T.C.
MALTEPE ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

KURUMSAL SERVİS ODAKLI MİMARİ KAVRAMI, TEKNOLOJİSİ VE TASARIMI

Ali BEKLEN

Yüksek Lisans Tezi

Tez Danışmanı
Yrd. Doç. Dr. Turgay Tugay Bilgin

İSTANBUL – 2009

T.C.
MALTEPE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

KURUMSAL SERVİS ODAKLI MİMARİ KAVRAMI, TEKNOLOJİSİ VE TASARIMI

YÜKSEK LİSANS TEZİ

Ali BEKLEN

Tez Danışmanı
Yrd. Doç. Dr. Turgay Tugay Bilgin

İSTANBUL – 2009

ÖZET

Yüksek Lisans Tezi, Kurumsal Servis Odaklı Mimari Kavramı, Teknolojisi ve Tasarımı, T.C. Maltepe Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı.

Servis odaklı mimari, yazılımın gerçekte nasıl uygulandığına odaklanan yazılım sistemleri oluşturmaya yönelik bir yaklaşımdır. Tüm işlevlerin veya servislerin bir açıklama dili kullanılarak tanımlandığı ve iş süreçlerini gerçekleştirmek için çağrılabilen arabirimlerin bulunduğu bir uygulama mimarisidir. Arabirimler, platformdan bağımsız olduğu için, bir servis, herhangi bir programlama dilindeki bir işletim sistemini kullanan herhangi bir aygıttan kullanabilir.

Servis odaklı mimari, kendine özgü kuralları olan, finans, sigorta ve kamu gibi farklı alanlarda hayata geçirilmiş bir yaklaşımdır. Servislerin iyi tanımlanabilmesi için detaylı bir analize ve modellemeye ihtiyaç duyar.

Bu tezin amacı, servis odaklı mimariyi oluşturan teknoloji, mimari yaklaşım ve programlama modellerini açıklamak ve Genişletilebilir Bağlantılı Metin Dili'nin (XML) bu mimari içindeki önemini araştırmaktır. Tez ile birlikte sunulan örnek senaryo ile servis odaklı mimari modellenerek, bu mimarideki teknoloji ve yaklaşımın nasıl hayata geçirildiği gösterilmeye çalışılmıştır.

Bu tez 2009 yılında tamamlanmıştır ve 107 sayfadan oluşmaktadır.

Anahtar Kelimeler: Servis, yazılım mimarisi, SOA, XML, web servis

ABSTRACT

Master Thesis, Service Oriented Architecture Concept, Technology and Design. T.C. Maltepe University, Institute of Natural Sciences, Department of Computer Engineering.

Service oriented architecture is an approach to build software systems that is focused on how the software is actually implemented. This is an application architecture in which functions or services have interfaces that are called to perform business processes and they are all defined using a description language. Interfaces are platform independent and they should get invoked by any other devices using any operating systems which are written in any programming language.

Service oriented architecture is an approach which has its own unique policies and has been implemented in different industries like finance, insurance and government. It needs to have detailed analysis and modeling to design well-defined services.

This thesis aims to describe the technology, architectural approach and programming models behind the service oriented architecture and to research the importance of Extensible Markup Language(XML) in this architecture. By prototyping the service oriented architecture with the case study which is given with the thesis, it is aimed to show, how technology and the approach of this architecture are implemented.

. This thesis has been completed in 2009 and consists of 107 pages.

Keywords: Service, software architecture, SOA, XML, web service

TEŞEKKÜR

Seçtiğim tez konusu üzerinde çalışmama olanak sağlayan Maltepe Üniversitesi rektörü sayın Prof. Dr. Kemal KÖYMEN'e ve Fen Bilimleri Enstitüsü Müdürü sayın Prof. Dr. E. Murat Esin'e, tez süreci boyunca destek ve yardımlarını esirgemeyen değerli bilgilerinden istifade ettiğim danışman hocam Yrd. Doç. Dr. Turgay Tugay Bilgin'e, çalışmalarım sırasında gerekli kaynakların sağlanmasına yardımcı olan IBM Türk firmasına, süreçler konusunda yardım ve desteklerini benden esirgemeyen sayın Öğr. Gör. Fatih YÜCALAR, sayın Ar. Gör. Selim Bayraklı ve sayın Ar. Gör. Derya Ersoy'a, örnek senaryonun geliştirilmesinde yardımlarını esirgemeyen mesai arkadaşlarım Arden Agopyan ve Emrah Utku Barkana'ya, tez sürecinde bana gösterdiği olağanüstü anlayış ve destek için çok değerli eşim Canan Beklen'e ve kızım Eda'ya, manevi desteğini benden hiçbir zaman esirgemeyen çok değerli aileme ve çalışmalarım sırasında emeği geçen herkese teşekkürlerimi sunarım.

İÇİNDEKİLER

KISALTMALAR

Kısaltma	İngilizcesi	Türkçesi
ACID	Atomicity, Consistency, Isolation, Durability)	Atomik olma, Tutarlılık, Yalıtım, Dayanıklılık
API	Application Programming Interface	Uygulama Programlama Arayüzü
BLOB	Binary Large Object	İkili Geniş Nesne
BPEL	Business Process Execution Language	İş Süreci Yürütme Dili
BPM	Business Process Management	İş Süreci Yönetimi
BPMN	Business Process Modeling Notation	İş Süreci Modelleme Gösterimi
BPMS	Business Process Management Systems	İş Süreci Yönetimi Sistemleri
BRM	Business Rules Management	İş Kuralları Yönetimi
CLOB	Character Large Object	Karakter Geniş Nesne
CORBA	Common Object Request Broker Architecture	Genel Nesne İstek Aracı Mimarisi
COTS	Commercial off-the-shelf	Kullanıma Hazır Ticari Uygulamalar
DBMS	Database Management Systems	Veritabanı Yönetim Sistemleri
DDL	Data Definition Language	Veri Tanımlama Dili
EAI	Enterprise Application Integration	Kurumsal Uygulama Entegrasyonu
HTML	HyperText Markup Language	Zengin Metin İşaret Dili
HTTP	HyperText Transfer Protocol	Zengin Metin Aktarma Protokolü
IIOP	General Inter-ORB(Object Request Broker) Protocol	Genel Nesne İstek Aracısı Protokolü

J2EE	Java Enterprise Edition	Java Kurumsal Sürümü
JCA	Java Connector Architecture	Java Bağlayıcı Mimari
JMS	Java Message Service	Java Mesaj Servisi
JSF	Java Server Faces	Java Sunucu Yüzleri
JSR	Java Specification Request	Java Tanım İsteği
LAN	Local Area Network	Yerel İletişim Ağı
LOB	Line of Business	Ana İş Dalı
PHP	Personal Home Page	Kişisel Ana Sayfa
PKI	Public Key Infrastructure	Genel Anahtar Altyapısı
QoS	Quality of Service	Servis Kalitesi
RDBMS	Relational Database Management Systems	İlişkisel Veritabanı Yönetim Sistemleri
REST	Representational State Transfer	Temsili Durum Transferi
RMI	Remote Method Invocation	Uzak Method Çağrısı
SCA	Service Component Architecture	Servis Bileşen Mimarisi
SDO	Service Data Object	Servis Verileri Nesnesi
SEI	Software Engineering Institute	Yazılım Mühendisliği Enstitüsü
SLA	Service Level Agreement	Servis Düzeyi Sözleşmesi
SOA	Service Oriented Architecture	Servis Odaklı Mimari
SOAP	Simple Object Access Protocol	Basit Nesne Erişimi Protokolü
SQL	Structured Query Language	Yapısal Sorgu Dili
SSL	Secure Sockets Layer	Güvenli Yuva Katmanı
TeX	Text Executive Processor	Metin Çalıştırma İşlemcisi
UDDI	Universal Description Discovery and Integration	Evrensel Açıklama, Keşif ve Tümleştirme
W3C	World Wide Web Consortium	Dünya Çapında Ağ Birliği
WS	Web Service	Web Servis
WSA	Web Service Architecture	Web Servis Mimarisi
WSDL	Web Service Definition Language	Web Servisi Tanımlama Dili

XML	eXtensible Markup Language	Genişletilebilir Bağlantılı Metin Dili
XPath	XML Path Language	XML Yol Dili
XSLT	Extensible Stylesheet Language Transformations	Genişletilebilir Biçem Dili Dönüşümü
XQuery	XML Query	XML Sorgu

ŞEKİLLER

TABLOLAR

1. SERVİS ODAKLI MİMARİ'YE GİRİŞ

Büyük ölçekli yazılım projeleri, çeşitli sektörlerde farklı iş problemlerini çözümlemek üzere yazılım mühendisliği disiplinleri uygulanarak geliştirilmektedir. Ancak her kurum hayata geçirdiği uygulamaların alt yapılarında farklı teknoloji ve yaklaşımlar kullanmıştır. Bazı projelerde ise herhangi bir mimari yaklaşım kullanılmadan uygulamalar geliştirilmiştir. Bu tip durumlar zaman içinde daha da büyüyen ve karmaşıklaşan uygulamaları, birbirleriyle entegre olamaz, bakımı güçlükle yapılan, tekrarlanan işlerin yapıldığı veya platformlara bağlı uygulamalara dönüştürmüştür.

Bu tip sorunları, uygulama katmanlarını birbirinden soyutlayarak çözmeyi hedefleyen servis odaklı mimari (SOA), yaklaşımı, mimari, servis ve tanım gibi alt başlıklarla bu bölümde incelenmiştir.

1.1. Mimari Nedir?

Yazılım mimarisi, ana bileşenler bakımından yazılım sistemlerinin tanımı, bağlantıları ve birbirleri arasındaki bilgi alışverişidir. Mimari, bir yapının en iyi şekilde tanımlanmış gerekliliklerin planı, aynı zamanda şimdi ve ileriki zamanlarda sistemlerin bu gereklilikleri yerine getirmek için sahip olacağı karakteristiklerdir.

"Yazılım mimarisinin temel amacı; karmaşık yazılım sistemlerinin çözümünde yardımcı olmak ve dış etkenlerle gelen değişikliklere ya da ilavelere iş alanında, kurumsal ve teknik ortamlarda yardımcı olmaktır" [1].

Yazılım mimarisinin endüstriyel olarak tek bir tanımı yoktur. Yazılım mühendisliği enstitüsü (SEI) yazılım mimarisi ile ilgili bir çok tanım yapmıştır. Bazı tanımlar daha fazla detay sağlar ve yukarıdaki amacı içerir. Diğerleri sadece özet olarak tanım içerir ancak başka bakış açısından mimari bakışını dile getirir. Yazılım mimarisinin genel tanımı aşağıdaki gibi yapılmıştır:

“Yazılım mimarisi:

- Organizasyonun yazılım sistemi
- Yapısal elemanların seçimi ve sistemin oluşturduğu arayüzleri, birbirleri ile olan ilişkisi
- Büyük alt sistemlerde elemanların bileşimi
- Organizasyonu yönlendiren mimari stil, elemanlar ve onların arayüzleri, bileşimleri, beraber çalışmaları ve durumlarıyla ilgili önemli kararları içine alır” [2].

Yazılım mimarisi sadece yapısal ve davranış olarak değil aynı zamanda kullanım, işlevsellik, performans, tekrar kullanılabilirlik, esneklik, anlaşılabilirlik, ekonomik ve teknolojik kısıtlamalar, ödünleşme ve estetikle de ilgilenir. Çoğu tanım, yazılım mimarisinin bileşik sistemleri tanımladığında hem fikirdir ancak perspektif olarak sistemin ne olduğu ve birleşiğin ne anlama geldiği konusunda farklılık gösterir. Bu tanım detaylıdır ve birçok perspektifi ve kapsamı içerir.

Geleneksel yazılım mimarisi, yazılım uygulamalarının geliştirilmesine odaklanılırken servis odaklı mimari, çözümlerin işletme ve çapraz kurumsal yöntemlerle üretilmesi, servisler ve iş süreçlerinin birbirleri ile olan etkileşimine odaklanır.

Yazılım mimarisi aşağıdaki dört soruya cevap verecek şekilde tanımlanmalıdır:

1. Önemli olan kavramlar nelerdir?
2. Bunlar arasındaki ilişkiler nelerdir? Bu ilişkiler sistem davranışlarını nasıl tanımlıyor?
3. Hangi kavramlar ve ilişkiler değerleri yukarı taşıyor?
4. Genel sistem, amacı haricinde bireysel bölümlerin amacına nasıl hizmet etmiş oluyor?

Tipik yazılım mimari tanımı, sınıfların yapısını (ana kavramı) veya bileşenleri, bunların ilişkilerini ve bunların yazılım ürünündeki değerlerini tanımlar. Benzer

olarak, SOA servislerin yapısını(ana kavramı), bunların ilişkilerini ve süreçlere ve sonuçlara getirdiği değerleri tanımlar [3].

1.1.1. Mimari stiller

Ürünlerin çoğu, uygulamalar ve şirketler, birbirine benzer olmasına rağmen, farklı mimarilere sahiptir. Örneğin, bir şirketteki e-ticaret uygulamasının mimarisi muhtemel olarak başka şirketteki e-ticaret uygulamasına büyüklük ve işlev bakımından benzerdir. Ancak özel bir mimari ile sunduğu çözüm tanımlamaları arasında farklılık vardır. Bunun endüstriyel olarak ortak adı mimari stildir.

"Mimari stil, kelime anlamıyla kavramlar, ilişkiler ve ilişkilerin daha iyi amaç ve belirli bir mimari oluşturmak için nasıl birleştirilebileceği gibi bir takım kısıtlamalar anlamına gelir. Bir mimari stil, ortak prensipler ve özellikler olarak mimariler ailesindendir. Bir başka deyişle, bir mimari stil iyi tanımlanmış, işletmeye ait çözümlerin ortak kalıplarını içerir" [3]. Örneğin istemci/sunucu, 3 katman, n katman ve kurumsal uygulama entegrasyonu (EAI) yaklaşımlarının hepsi birer mimari stildir.

Kurumsal çözümler için mimari stil seçimleri bazı gereklilikler sonrası mühendislik ödünlerine göre seçilir. Örneğin; n katmanlı mimari stil şu gereksinimleri yerine getirmelidir:

- Dağılma,
- Ölçeklenebilirlik,
- Esnek arayüz,
- Cihaz bağımsız,
- Ticari servis kullanılabilirliği,
- Uygulama entegrasyonu ve diğerleri.

Özellikle n katmanlı mimari web tabanlı ve diğer tip erişilebilir bilgi ve müessese içinde mevcut servisler olarak tasarlanmıştır. SOA işlevsel servisleri destekleyen, yani temel birim olarak tasarlama, yaratma ve işlevsel çözümleri dizme gibi, bir

mimari stil olarak tanımlanabilir. SOA çözümlerinin tanımları, uygulama ve kurulumu tanımlayan birçok model bu stili tamamlar.

1.1.2. Mimari ile ilgili ilkeler ve uygulamalar

"Mimari açıdan ön plana çıkan en önemli ilke ve uygulamalar:

- İlişkilerin ayrıştırılması
- Mimari ile ilgili görünüm
- Değişimin sağlanması
- Soyutlama
- Tutarlılık
- İşletme ile ilgili türetme
- Kalıplar
- Basitleştirme
- Erişim" [3].

İlişkilerin ayrıştırılması, mimarinin en temel ilkesidir. İlişkiler bağımsız elemanlar bağımsız kalmaya devam etsin diye hep ayrı tutulur. Bunun yararı da sistemin bir parçasının diğer bir parçayı etkilememesidir. Bir başka deyişle bağımsız olarak değişebilir. Örnek olarak uygulamadan arayüzün ayrılması verilebilir.

Mimari ile ilgili görünüm, bir başka önemli ilişkinin ayrıştırılmasını sağlar. Bunu, bazı özelliklerin dışa aktarılması ya da dışarıdan alınması ve çeşitli paydaşların bilgilerini sunarak sağlar. Mimari ile ilgili görünüm ya da bakış açısı yazılım geliştirilmesinin belli başlı ilişkileri ya da kurumsal çözümlerin tüm hayat döngüsünde bir rol oynayan önemli işletme grupları ve organizasyonları belirtmesi için tasarlanmıştır. Tipik yazılım görünümü mantıklı yayılımcı, süreç ve bilgisayar ağından oluşur. Tipik kurumsal mimari görünümü işletme, bilgi, uygulama ve teknolojidir. SOA, uygulama ve tasarım bu mimari ilişkilere işaret etmektedir.

Değişimin yerleşmesi, mimarinin esneklik sağlayabilmesine bağlıdır. Böylece daha sonraki uygulama gereksinimleri daha kolay sağlanabilir. Esnek mimari, ileriki uygulama gereksinimlerini ve değişebilecek alanları belirtir. Endüstriyel eğilimleri takip etmek potansiyel değişiklik alanlarının saptanmasına yardımcı olur. Bu alanlar mimari de ayrıntılı olarak tanımlanmalıdır. Eğer özellikler ve bağımsızlık, orijinal tasarımda yer almamış ise, mimari üstü kapalı bağlantılar içerecektir. Değişimler gerektiğinde bu bağlantılar ile uğraşmak oldukça zordur

Soyutlama; ayrım, değişimin algılanması ve ilişki ayrıştırılmasında kullanılan önemli bir mimari bileşendir. Endüstride " bilgisayar ortamındaki bir sorun soyut bir katman eklenerek çözülebilir" diye bir deyiş vardır. Bu soyutlama katmanı daha fazla esneklik ile iki katman arasında dolaylı bir yol sağlar. Soyutlama yüksek seviyede iletişim sağlar. Örneğin veritabanına doğrudan yazmak yerine, yapısal sorgu dili (SQL) yazarak ulaşılabilir. SQL yüksek seviyede iletişim modeli, aynı zamanda soyutlama ve en alt seviye veritabanı arayüzü üzerinde dolaylı katman sağlar. Soyutlama, yüksek üretimi (bir SQL satırı birçok alt seviye veritabanı arayüzüne karşılık gelir.) ve birçok değişik veri saklanmasını destekler.

Mimarinin ana amaçlarından biri tutarlılık ve tekrar kullanımı desteklemesidir. Yazılım mimarisi ve servis odaklı mimari arasındaki farklılıklardan biri kapsamdır. SOA, tüm kurumlara tutarlı servisler vermek amacını güder. Böylece SOA farklı çözüm ailelerinde tekrar tekrar kullanılabilir. SOA iş özelliklerini geliştirmeyi desteklediği takdirde değişik iş süreçleri tarafından tekrar kullanılabilir.

İşletmesel türetme, en önemli mimari ilkelerden biridir. Mimarinin amacı kurumsal işlevleri desteklemektir.

Model, özel gereksinimlere çözüm olan bir örnek ve mimariyi açıklamak için bir araçtır. Model kavramı ilk olarak Berkeley'de bir mimarlık profesörü olan Christopher Alexander, tarafından ele alınmıştır. Christopher Alexander "Her bir model bizim ortamımızda tekrar tekrar oluşan bir problemi ve bu problemin temel

çözümünü tanımlar ki bu çözümü milyon kere, her seferinde farklı şekilde kullanabilirsiniz" demektedir [3].

Basitleştirme, bir mimarinin kolay çözümler yaratıp mimariye uymasıdır. Mimari sadece sistemin ne yaptığını anlatmaz, sistemin nasıl yaratılacağını ve bileşenlerini de belirtir.

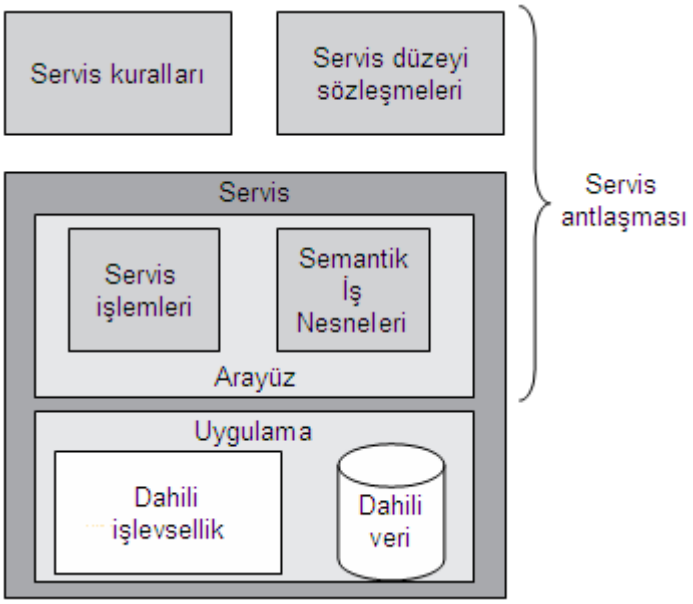
Erişim de, mimari bireylerin erişim mekanizmalarını ve bilişim sistemlerinde ortak bir algılamayı tanımlar. Bu mimari ilkeler mimari tanımının değişik seviyelerde yapabilmesine yardımcı olur.

1.2. Servis Nedir?

"SOA'nın temel fikri servistir. Servis, servis anlaşmasıyla yapılabilen iş işlevselliğinin ayrık bir birimi olarak tanımlanır. Servis antlaşması, servis sunucusu ve servis kullanıcısı arasındaki etkileşimi belirtir. Bunlar:

- Servis arayüzü
- Arayüz dokümanı
- Servis kuralları
- Servis kalitesi(QoS)
- Performans" [4].

Servis ve başka yazılım bileşenleri (objeler veya öğeler) arasındaki en önemli faklılıklardan biri servisin harici olarak yönetilmesidir. QoS ve performans, servis düzeyi sözleşmeleri(SLA) tarafından yönetilir. Buna ek olarak tasarımdan geliştirmelere ve bakıma kadar tüm servis hayat döngüsü yönetilmektedir. Şekil 1.1'de servisin bileşenleri gösterilmektedir.



Şekil 1.1 Servis bileşenleri [4]

Servisin iki ana bakış açısı bulunmaktadır. Şekil 1.1'de, servisin üst tarafı servis arayüzünü ve alt tarafı ise servis uygulamasını göstermektedir. Servis, özellikle arayüzü uygulamadan ayırır.

Servis arayüzü servis operasyonlarını belirtir. Bunlar servisin ne yaptığı, servisin operasyona giren ve çıkan parametreleri, servisin özellikleri ve sağladığı protokolleridir. Bir servis birden fazla değişik ancak birbirleri ile alakalı operasyon içerir. Servis uygulaması, servisin arayüzde sağladığı özelliklerdir. Uygulama, var olan uygulamalara diğer servislerin yeterliliklerini birleştirerek uyumlaştırmaya, servis için yazılan kodlara ve ya bunların tümüne bağlı olabilir. Burada önemli olan servis kullanıcılarının sadece servisin ne yaptığını bilmeleridir nasıl kodlandığını bilmeleri değildir. Servis sağlayıcısı servisin arayüzünü ve davranışını değiştirmediği sürece servis uygulamasını değiştirmekte serbesttir. Örneğin yeni bir servis büyük olasılıkla varolan bir uygulamanın fonksiyonuna bağlı olacaktır. Arayüz son halini aldığında kullanıcılar servisi kullanmaya başlayabilir. Bu arada servis sağlayıcı yeni modern uygulama yapabilir ve artık desteklenmeyen ortamda çalışan eski uygulamayı kullanımdan kaldırabilir. Servis kullanıcıları servisin davranışı ve servis anlaşması değişene kadar durumu kesinlikle farketmezler. Diğer bir deyişle, servis

arayüzü kullanıcıların ilişki kurduğu nokta olarak düşünebilir. Arayüz stil ve ilişkinin detayını tanımlar. Uygulama kısmı servis sağlayıcının sağladığı özellikleri tanımlar. Bu bağlantı noktası görüşü çözümlerin tasarlanmasına izin verir.

Arayüz ve uygulama ile ilgili farklı görüşler vardır. Bunlar uygulanan fonksiyonlar ve uygulanan bilgilerdir. Bir başka deyişle servis, fonksiyonel servis işlemleri setinin ve işleme giren ve çıkan ilişkili sanal verilerin birleşimidir. Sanal veri, veri depolama ve uygulamadan bağımsız olan iş varlıklarının (kurumsal şemaya bağlı olarak) soyutlanmasıdır. Servis işlem imzası, işlemle giren ve çıkan parametreleri açıklar. Bilgi modeli (veya kurumsal şema) giren ve çıkan sanal verinin anlamını ve yapısını tanımlar [4].

Servis arayüzündeki sanal bilgi ve servis uygulamasındaki mantıklı ve fiziksel verinin farklılığı kritiktir. Servis arayüzü seviyesinde önemli olan, iş sürecinin olanak vermesi ve tamamlanması için servisler arasında iletilmek zorunda olan bilgidir. Bu bilgi, sürece katılan tüm servislerde ortak ve üzerinde fikir birliğine varılan bilgi olmalıdır. Buna rağmen dahili olarak bu servislerin bir çoğu farklı kapsamdaki bilgiye ve muhtemelen farklı formata sahiptir. Tüm servislerin dahili veri modellerinin farklı ayrıntılarının bilinmesi ve onlar üzerinde uzlaşılması zorunlu değildir. Bunun yerine arayüzün uygulamadan ayrılması genel (sanal) tanımlama ve dahili (fiziksel) uygulama arasındaki çevirmenin kolaylaşmasına izin verir.

1.2.1. Servis özellikleri

Şekil 1.1'de gösterilen servisin belirli yapısına ek olarak, iyi servisler aşağıdaki özelliklere sahiptir:

Modülerlik ve öğe boyutu: SOA içinde iş süreçleri, bağımsız modüler servislere ayrılmıştır. Servisler başka modüler servislerden oluşabilir ve yeni birleşik servisler oluşturmak için gerektiğinde karışık ve eşlemeli şekilde kullanılabilir.

Öğe boyutu bir servisin işlevsel zenginliğinin kalitesini gösterir. Bir servis ne kadar büyük öğe boyutuna sahipse, servisin sunduğu işlev o kadar zengin ya da büyük olacaktır. Büyük öğe boyutlu servisler, tek bir servis işlemi içinde daha üst düzeyde işlevsellik sunar. Böylece, belirli bir iş etkinliğini yerine getirmek için gerekli adımları azaltarak karmaşıklığı ve ağ üzerindeki ek yükü azaltmaya yardımcı olur. Genellikle daha küçük görevleri büyük öğe boyutlu tek bir işlemde birleştirerek gerçekleştirilir. Küçük öğe boyutlu servis işlemleri, belirli bir somut görevi tamamlamak için küçük miktarlarda bilgi alışverişi sağlar. Büyük öğe boyutlu servise örnek olarak bir sigorta fiyat teklifinde kullanılan servis verilebilir. Küçük öğe boyutlu servis (diğerlerinin yanı sıra fiyatlandırma servisi tarafından kullanılabilecek olan), başvuru sahibinin posta koduna dayalı olarak risk bilgilerini verebilir.

Sarmalama: Servisler, servis arabiriminin (servisin ne yaptığı) servis uygulamasından (servisin nasıl yapıldığı) kesin çizgilerle ayrıldığı bir yapı sergiler. Sarmalama, servisin iç uygulama ayrıntılarını ve veri yapılarını yayınlanan arabirim işlemlerinden ve semantik modelden gizler.

Gevşek bağlaşım: Bağlaşım, bir servis kullanıcısı ile sağlayıcısı arasındaki bağımlılık sayısını tanımlar. Gevşek bağlaşımlı servisler az sayıda, iyi bilinen ve yönetimli bağımlılığa sahiptir. Sıkı bağlaşımlı servisler çok sayıda bilinen ve daha da önemlisi bilinmeyen bağımlılıklara sahiptir. Bağlaşım derecesi bir sistemin esnekliğini ve genişletilebilirliğini doğrudan etkiler.

Sorumlulukların ayrılması: Servisler belirli görevlerden ya da belirli kaynakların yönetiminden sorumludur. Servis tasarımının temel özelliklerinden biri, sorumluluğun belirli işlevler ya da tek bir servis içindeki bilgi için ayrılmasıdır. Bu tasarım, her işlev için tutarlılık sağlar ve yinelemeyi azaltır.

Özerklik: Özerklik, servislerin ve bu servisleri kullanan çözümlerin birbirinden bağımsız şekilde konuşlandırılmasını, değiştirilmesini ve sürdürülmesini sağlayan özelliktir. Özerk bir servisin yaşam döngüsü diğer servislerden bağımsızdır.

Yeniden kullanım: Modülerlik, sarmalama, gevşek bağlaşım, sorumlulukların ayrılması ve özerklik; servislerin birden çok iş sürecinde birleştirilmesini ya da birden çok yerden ya da birden çok bağlamda, birden çok servis kullanıcısı tarafından erişilmesini sağlar. Diğer bir deyişle servisler, süreçlerin ve birleşik servislerin yapılandırılmasında yapı taşları olarak paylaşılır ve yeniden kullanılır.

Dinamik keşif ve ilişkilendirme: Servisler, tasarım sırasında bir tasarım süresi servis havuzunun kullanılmasıyla keşfedilebilir. Servislerin çalıştırılması sırasında dinamik olarak keşfedilmesi kuramsal olarak mümkün olsa da, bunun uygulanması pratikte pek mümkün değildir.

Ancak, servis kullanıcıları servislerin çalıştırılması sırasında sağlayıcılara dinamik olarak bağlanabilir. Bu senaryoda, kullanıcılar belirli bir servisin kaydını isterler ve uygun servis sağlayıcısına dinamik olarak yönlendirilirler ve bağlanırlar. Servis kullanıcısının servis sağlayıcısına dinamik olarak bağlanması, gevşek bağlaşımı geliştirir ve aracılık gibi ek özellikleri etkinleştirir.

Durumsuz: Servis işlemleri durumsuzdur. Kendilerinden istenilen en son işlemi anımsamazlar ya da sonraki işlemin ne olacağına ilişkin bir bilgiye sahip değillerdir. Servisler, diğer servislerin bağlamından ya da durumundan bağımsızdırlar. Yalnızca işlevsellik açısından bağlıdırlar. Durumsuz servisler daha iyi esneklik, ölçeklenebilirlik ve güvenirlik sağlar. Durumsuz servislere örnek olarak uzun süre çalışan servis etkileşimleri verilebilir.

Öz tanımlamalı: Servis sözleşmesi servis arabiriminin, servisin işlemlerinin, giriş ve çıkış parametrelerinin ve şemasının eksiksiz bir tanımlamasını sağlar. Sözleşme, işlemlere ilişkin önkoşullar ile sonraki koşullar ve sınırlamalar da içerebilir.

Oluşturulabilirlik: Servisler başka servislerden oluşturulabilir ve dolayısıyla yeni servisler ya da iş süreçleri oluşturmak için başka servislerle birleştirilebilir.

İlke tarafından yönetilen: Servis kullanıcıları ile sağlayıcıları arasındaki (servisler ile servis etki alanları arasındaki) ilişkiler ilkeler ve servis düzeyi sözleşmeleri (SLA'lar) tarafından yönetilir. İlkeler, farklı kullanıcıların servisle nasıl etkileştiklerini, diğer bir deyişle neleri yapmalarına izin verildiğini tanımlar.

Konumdan, dilden ve protokolden bağımsız: Servisler konum olarak şeffaf ve protokolden/platformdan bağımsız olacak şekilde tasarlanmıştır. Diğer bir deyişle, yetkili her kullanıcı tarafından, her platformda ve her yerden erişilebilir olacaklardır.

1.2.2. Dinamik keşif ve ilişkilendirme

Servislerin çok fazla dile getirilen özelliklerinden biri dinamik keşif ve ilişkilendirme kavramıdır. Dinamik keşif, bir servisin kullanıcısının bir servisin var olduğunu keşfetmek ve servisi kullanmaya başlamak için gerekli tüm bilgileri almak üzere belirli bir merkezi konuma gidebileceği anlamına gelir. Geliştirme sırasında faydalı olan bir kavramdır. Senaryo şu şekildedir: Şöyle işler yapan bir servise ihtiyaç duyuyorsunuz ve aradığınız servisleri bulmak için bir yere gidiyorsunuz. Bir servis bulduğunuzda, nasıl kullanacağınızı belirleyebilirsiniz. Çoğu kuruluşta bu, bir servis havuzu birleşimi ve ağızdan ağıza yöntemiyle yapılır [3].

Bilgisayar programının nasıl yapılandırıldığı düşünüldüğünde; belirli şeyleri belirli bilgilerle yaparak belirli hedefleri gerçekleştirmeniz gereklidir. Program bilgileri toplamakla işe başlar ve ardından istenilen sonucu elde etmek için algoritmaları bu bilgilere uygular. Ancak, neyi gerçekleştirmesi gerektiğini bilmeyen ve bilinmeyen hedefleri gerçekleştirmek için her nasılsa kullanabileceği rastgele işlevleri aramaya çıkan programlar genelde yazılmaz. Bir programın çalıştırma süresinde servisleri dinamik olarak keşfetmesinin, bunları nasıl kullanacağını belirlemesinin ve yeniden yapılandırmasının ardında yatan fikir, sistemlerin nasıl oluşturulduğu, sınandığı ve konuşlandırıldığıyla taban tabana zıttır. Ancak "dinamik keşif kuramsal olarak mümkün olsa da bu fikirden etkilenmiş bir kurumsal servis odaklı mimariyi bir süre daha görmek mümkün olmayacak" [3].

Bir diğer konu da dinamik ilişkilendirme konusudur. Buradaki ana fikir, bir programın bir servis arabiriminin tanımını okuyacağı ve servisi kullanmak için istek ve yanıt işlemeyi dinamik olarak yapılandıracağı yönündedir. Servisin kullanıcısı bir kullanıcı arabirimi üzerinden etkileşimde bulunan bir kişiyse, kullanıcıya servisin girdilerini soran, istekleri oluşturan ve gönderen; ardından sonuçları görüntüleyen bir formu dinamik olarak oluşturacak bir programın yazılması mümkündür. Ancak, servisin kullanıcısı SOA'da genellikle olduğu gibi başka bir programsa, bu çok anlamlı olmayacaktır. Bir servisi kullanmak için bir program yazılıyorsa, servis isteğine hangi parametrelerin gönderileceği ve servis yanıtının nasıl işleneceği bilinmesi gerekir. Genel olarak, bunun dinamik olması istemez ya da gerek duyulmaz veya bunu yapabilecek kadar akıllı bir programı yazma maliyeti göze alınmaz. Dolayısıyla, kuramsal olarak mümkün olan dinamik arabirim keşfinin olduğu birkaç örnek olsa da uygulanabilir olmayacaktır. Ancak, dinamik ilişkilendirmenin diğer yönleri ve kullanıcının sağlayıcı uç noktasına dinamik bağlantısı faydalıdır.

1.2.3. Servis bileşenleri tanımları

Servis bileşenleri tanımları aşağıdaki gibidir :

Servis : Bir servis sözleşmesine dayalı olarak bir işlevselliği sağlamak için kullanılan belirli bir stildir. Servislerin etkileşimi servis arabiriminde belirtilir. Servis, belirli özelliklere sahip sağlayıcılarla etkileşimde bulunması gereken kullanıcılar için bağlantı olarak işlev görür. Genellikle, bir servis gevşek bağlaşımla daha kapsamlı tanımlanır ve veri odaklı etkileşim stilinden çok süreç odaklı etkileşim stili sunar.

Arabirim: Bir servisle etkileşimi tanımlar. İlgili özellikleri ilişkili işlemler kümesi üzerinden gruplandırır. Arabirim, servis işlemlerinin girdileri ve çıktıları ile bu işlemlerin önkoşullarını, sonraki koşullarını ve sınırlamalarını tanımlar. Arabirim, arabirim tanımlama dili olarak (WSDL gibi) ya da arabirim sınıfı olarak (Java gibi) belirlenebilir.

Süreç: İş süreci, belirli bir kurumsal hedefi gerçekleştirmeye yönelik koordinasyonlu görevler ve etkinlikler kümesidir. SOA açısından, bir süreç, iş servislerini koordine etmek için kullanılabilir.

Öğe boyutu: Belirli bir etkileşimdeki işlevselliğin boyutu ya da miktarı anlamında kullanılır. Örneğin, çok küçük öğe boyutlu bir etkileşim, bir nesnenin tek bir özniteliğini belirlemek ya da almak olabilir. Çok büyük öğe boyutlu bir etkileşimse, tek bir etkileşimdeki bir nesneler toplamının tüm değerlerini almak olabilir. Bunlar, arabirim öğe boyutlarına örneklerdir. Öğe boyutu, belirli bir etkileşimin değeri anlamında da kullanılır. Örneğin, Üye Ekle servisi Adresi Doğrula servisine göre daha yüksek öğe boyutlu bir servistir. Bir servisin ve arabirimin uygun öğe boyutu, düşünülen kullanımına bağlıdır ve servis tipi için geçerlidir.

Birleşik servis: Diğer servislerin birleşimi olan bir uygulamaya sahip bir servistir.

Atomik servis: Uygulanması için başka bir servise gerek duymayan ya da başka bir servisi kullanmayan bir servistir. Servis oluşturmanın en alt düzeyidir.

Temel servis: İş kuralları motoru, veri yönlendirme servisi ya da iş akışı sistemi gibi başka servislerin oluşturulmasında kullanılan bir yardımcı programdır. Bu servisler belirli işletme işlevleri sağlamaz; bunun yerine servislerin oluşturulmasında yüksek düzeyde teknik beceriler sağlar. (Altyapı servisi ve temel servis terimleri eşanlamlı olarak kullanılmaktadır.)

İş servisi: İş değerinin (Bloke Hesabı Değerlendir ya da Ödemeleri Yeniden Hesapla gibi) daha yüksek öğe boyutunu sunan belirli türde bir servistir. Genellikle birden çok alt düzeyde ya da daha küçük öğe boyutlu servisten oluşur.

Etki alanı servisi: Belirli bir iş etki alanında işletme işlevselliği sağlayan alt düzey bir servistir. Bir etki alanı içinde önemli ve paylaşılan özellikler sunar; ancak, etki alanı dışında açıklanmaya yönelik değildir. Örneğin, alacaklı verilerinin doğruluğunun denetlenmesi, alacak hakkı işlemenin farklı yönlerince paylaşılabilen;

ancak, başka etki alanları tarafından kullanılmayan bir işlevdir. Etki alanı servisleri, iş servislerinin oluşturulmasında kullanılan ortak bir işlevsellik sağlar.

Yardımcı program servisi: Yardımcı program servisleri, en küçük ya da en az büyüklüğe sahip öğe boyutlu servislerdir. İşletme çapında ortak işlevler sunan alt düzey servisler sağlar (örneğin, adres defteri işlevi ya da parça numarası doğrulama).

Tümleştirme servisi: Var olan uygulamaları işletmenin geri kalanı tarafından kullanılabilen hizmetler olarak açıklar ve birden çok veri kaynağı üzerinde yayılmış işletme verilerine tutarlı ve birleşik erişim sağlar. Tümleştirme servislerinin öğe boyutu, açıkladıkları mevcut sistemlere kısmen bağlıdır. Tümleştirme servisleri genellikle işletme modeli ile uygulama modeli arasında bir dönüştürme içerir.

Dış servis: İşletme dışındaki tedarikçiler ya da iş ortakları tarafından sağlanan sistemlere ve uygulamalara erişim sağlar (örneğin, kredi kartı doğrulama ya da gönderi izleme). Dış servislerin öğe boyutu belirli bir servis sağlayıcıya bağlıdır. Geleneksel olarak bu servisler göreceli şekilde küçük öğe boyutlu olsa da, yeni yeni görülmeye başlayan servis olarak yazılım sağlayıcılar tüm alanlarda çok çeşitli servisler oluşturmaktadır.

İşletme iş süreci: İşletme içindeki (ya da dışındaki) iş etki alanlarını yayan belirli türde bir iş sürecidir.

İş akışı: Bir sürecin bir dizi adıma, etkinliğe, koşula ve benzeri yapılara ayrıldığı bir bilgi işleme stilidir. Bir adımdan bir sonrakine çalışma etkinlikleri akışı, koşullu değerlendirmeye dayalıdır. İş akışı genellikle, iş akışı geliştirmeyi, işi kuyruğa göndermeyi ve süreç yönetimini destekleyen bir iş akışı yönetim sistemi tarafından yürütülür. Çoğunlukla iş akışı sistemleri kişiler tarafından yürütülen etkinlikler içerir; burada iş öğeleri kişinin gelen kutusuna gönderilir ve tamamlanmış öğeler bu kişinin giden kutusuna aktarılır.

Uyumlulaştırma: Genellikle iş servislerinden iş süreçleri ya da küçük servislerden birleşik servisler oluşturmak için uygulanan belirli tipte bir iş akışıdır ve insanlar tarafından gerçekleştirilen etkinlikler içermez. Uyumlulaştırma genellikle, birbirleriyle doğrudan bağımlılığı olmaması için diğer parçalar arasında etkileşimi yöneten, denetleyen ya da yönlendiren bir yönetici ya da denetleyici içerir.

İş Süreci Yönetimi (BPM): Wikipedia bu kavramı, "insanları, kuruluşları, uygulamaları, belgeleri ve diğer bilgi kaynaklarını kapsayan operasyonel iş süreçlerini tasarlamak, uygulamak, denetlemek ve çözümlemek için yöntemleri, teknikleri ve araçları içine alan, yönetim ve bilgi teknolojisinin kesiştiği noktada gelişmekte olan bir bilgi ve araştırma alanı," olarak tanımlamaktadır. Bu kavram, uyumlulaştırma teknolojisine ek olarak iş süreçlerinin yönetimini vurgulayan bir süreç yapılandırma tipi olarak düşünülebilir. BPM sistemlerinin ana işlevi, süreçlerin planlanan iş hedeflerini karşıladığından emin olmak için izlenmesidir. Ayrıca, denetleme, raporlama ve diğer işlevleri de içerebilirler.

İş Süreci Modeli: Alt düzey servislerden üst düzey süreçlerin yürütülmesini ve oluşturulmasını tanımlamada kullanılan bir modeldir. Süreç modelleri uyumlulaştırma ya da BPM araçları tarafından yürütülür. Hem iş servisleri hem de işletme iş süreçleri, İş Süreci Modelleri tarafından tanımlanabilir.

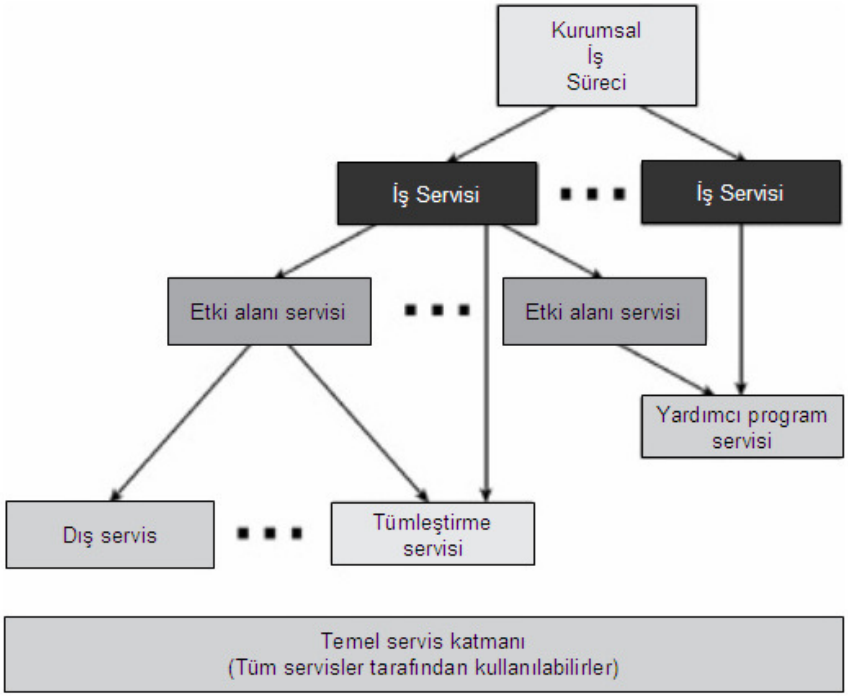
1.2.4. Servis öğe boyutu

Öğe boyutu, servisin boyutunu tanımlar. Bu boyut, kilobayt cinsinden kod boyutu anlamında kullanılmaz. Tek bir istek/yanıt ileti alışverişinde gerçekleştirilen işletme işlevi miktarı anlamında kullanılır.

SOA hakkında cevaplanması gerekli önemli sorulardan biri: "Bir servis ne kadar büyük olmalıdır?" sorusudur. Genel kabul gören cevap, servislerin büyük öğe boyutlu olması gerektiğidir; ancak, bu o kadar da basit değildir. Bir servis için tek bir doğru öğe boyutu yoktur. Doğru öğe boyutu daha çok, aşağıda sıralananlar gibi birçok etmene bağlıdır:

- Servisin hedeflenen kullanıcıları kimlerdir (iş ortakları, iş süreçleri, diğer servisler)?
- Topoloji ve performans gereksinimleri nelerdir (LAN, WAN, vb.)?
- Servisin hedeflenen kapsamı nedir?

Karmaşık bir sistem ya da ortamda, çok çeşitli servis öğe boyutlarının görülmes iolasıdır. Şekil 1.2'de aşağıdaki servis tipleri ve öğe boyutlarının bir sıra düzeni gösterilmektedir:



Şekil 1.2 Servis sıra düzeni

İşletme iş süreçleri: Bu iş süreçleri tüm işletmeye yayılır ve temel servisleri kullanabilir.

İş servisleri: İş servisleri en büyük öğe boyutlu servislerdir. İş servisleri işletmeye yüksek düzeyde, birleşik işletme işlevleri sunar. İşlevler ve bilgiler, iş süreçlerinin gerek duyduğu semantik ve sözdizimiyle yakından eşleşir. Bu düzeydeki veri

tümleştirme servisleri işletme süreçleri tarafından gerek duyulan birleşik verileri destekler. İş servisleri aşağıdakilerden biri olabilir:

- Ana iş dalı (LOB) servisleri: İşletmenin geri kalanına dışarıdan açıklanan belirli LOB işlevleri
- Ortak iş servisleri: Tüm uygulamaların temel işletme işlevlerini paylaşmasını ve ortak bir işleyiş göstermesini sağlar (örneğin, Yapılandırılmış Bilgi Yönetimi)

Etki alanı servisleri: Etki alanı servisleri orta büyüklükte öğe boyutuna sahiptir. İş etki alanına özgü işle ilgili servisler sağlar ve bu etki alanındaki birçok farklı iş servisi (örneğin, üyelik doğrulama) tarafından kullanılır; ancak, etki alanı dışında açıklanamaz.

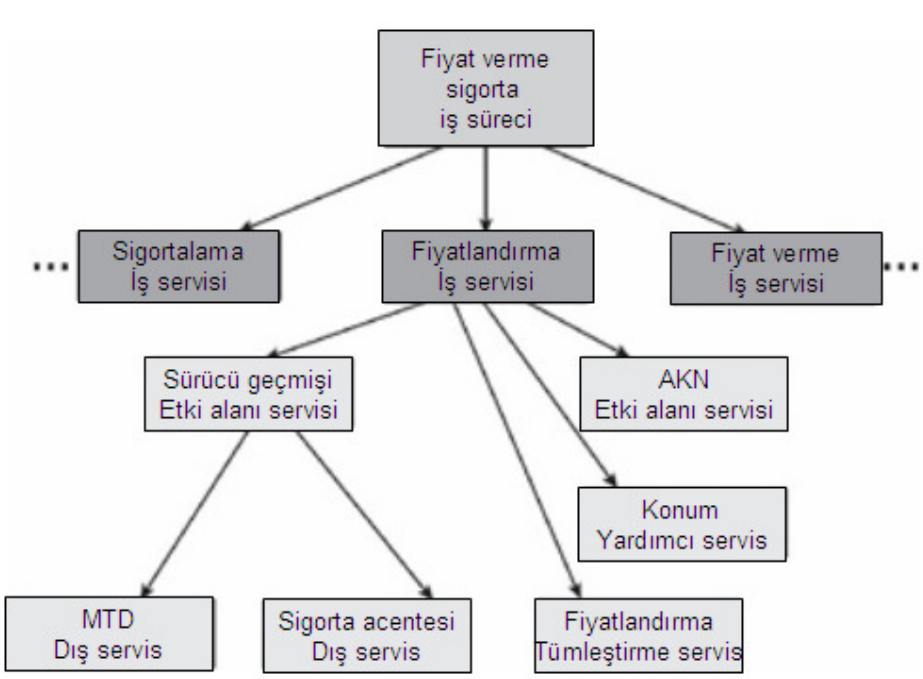
Yardımcı program servisleri: Yardımcı program servisleri en az büyüklüğe sahip öğe boyutlu servislerdir. İşletme çapında ortak işlevler sunan alt düzey servisler sağlar (örneğin, adres defteri ya da parça numarası doğrulama).

Tümleştirme servisleri: Bu servisler, var olan uygulamaları işletmenin geri kalanı tarafından kullanılabilen hizmetler olarak açıklar ve birden çok veri kaynağı üzerinde yayılmış işletme verilerine tutarlı ve birleşik erişim sağlar. Tümleştirme servislerinin öğe boyutu, açıkladıkları mevcut sistemlere kısmen bağlıdır. Tümleştirme servisleri genellikle, işletme modeli ile uygulama modeli arasında hem işlevsel hem de bilgi düzeyinde bir dönüştürmeyi içerir.

Dış servisler: Bu servisler, işletme dışındaki tedarikçiler ya da iş ortakları tarafından sağlanan sistemlere ve uygulamalara erişim sağlar (örneğin, kredi kartı doğrulama ya da gönderi izleme). Dış servislerin öğe boyutu belirli servis sağlayıcıya bağlıdır. Geleneksel olarak bu servisler göreceli şekilde küçük öğe boyutlu olsa da, yeni yeni görülmeye başlayan servis olarak yazılım sağlayıcılar tüm alanlarda çok çeşitli servisler oluşturmaktadır.

Temel servisler: Bu servisler, bir iş etki alanından bağımsız daha yüksek düzeyde servisler oluşturmada kullanılan küçük öğe boyutlu özellikler sağlar (örneğin, güvenlik, günlük kaydı ve uyumlulaştırma). Bunlar, CORBA ya da COM gibi altyapıları destekleyen, geleneksel olarak servis biçiminde adlandırılan özelliklerdir. Bunlar servis olarak adlandırılabilmekle birlikte bu servisler ile burada belirtilen işle ilgili servisler arasında bir ayrıma gidilmesi gereklidir. Bu servislere bazen teknik ya da altyapı servisleri de denir.

Örneğin, otomobil sigortası poliçeleri için fiyat teklifleri sağlayan yeni bir iş süreci oluşturılması istendiğini varsayalım. Fiyat teklifi sürecindeki adımlar, bilgi toplamayı, isteği doğrulamayı, isteği sigortalamayı, poliçeyi fiyatlandırmayı, teklifi oluşturmayı ve bunu müşteriye göndermeyi içerir. İş sürecinin bu başlıca adımları ya da görevleri iş servisleri tarafından uygulanır. Şekil 1.3'de, özellikle bu iş sürecini göstermek için değiştirilmiş bir servis sıra düzeni gösterilmektedir.



Şekil 1.3 Servis sıra düzeni örneği

Bir fiyat oluşturmak için sigorta şirketinin yapması gerekenler:

- Motorlu taşıtlar dairesinden sürücünün geçmiş kayıtlarının belirlenmesi

- Bağımsız sigorta acentesinden sürücünün geçmiş kayıtlarının belirlenmesi
- Sürücünün ikamet adresine göre riskin belirlenmesi
- Aracın marka ve modelinin belirlenmesi
- Şirketin ana çerçevesinde çalışmakta olan mevcut fiyatlandırma uygulamasının kullanılmasıdır.

Şirket, otomobil ana iş dalı içinde bir "sürücü geçmişi" servisi uygulamıştır. Bu etki alanı servisi iki dış servis kullanmaktadır. Biri sürücünün ehliyet kaydını almak için motorlu taşıtlar dairesinden (MTD) ve diğeri, sürücünün sigortaya bulunduğu hasar ödemesi talepleri geçmişini almak için sigorta acentesinden. Şirket, marka ve model bilgilerini almak için bir de araç kimlik numarası (AKN) etki alanı servisi uygulamıştır. Bu servislerin her ikisi de orta büyüklüktedir ve genellikle otomobil ana iş dalında faydalıdır. Bir konum yardımcı program servisi sürücünün ikametgah adresiyle ilgili bilgileri almak için kullanılır. Bu servis girdi olarak posta kodunu alır ve risk bilgilerini döndürür. Son olarak şirket var olan ana çerçeve fiyatlandırma işlemini kapsayan bir fiyatlandırma tümleştirme servisi yazmıştır.

Fiyatlandırma iş servisi, geri kalan tüm servisler için bir koordinatör görevi görmektedir. Fiyatlandırma işlemi iş süreci tarafından çağrıldığında, fiyatlandırma işlemi diğer servislerin her birindeki işlemleri çağırır (Sürücü Geçmişi, AKN ve Konum), döndürülen bilgileri derler, fiyatlandırma tümleştirme servisini çağırır, yanıtı biçimlendirir ve bunu iş sürecine geri gönderir; iş süreci sonraki adım olan fiyat verme iş servisine geçer.

Bir servisin ne kadar büyük olması gerektiğini belirlemenin yollarında biri, servisin modülerliğinin servis kullanıcısının beklentilerini karşılayabiliyor olmasıdır. Örneğin, iş süreçleri ile iş servisleri arasındaki ilişkiyi, bir sürecin görevlere ayrıldığını ve görevlerin iş servislerinin işlemleri tarafından uygulandığını belirtilmiştir. Dolayısıyla, bir iş servisinin doğru öğe boyutu, bir iş sürecindeki görevin boyutuna karşılık gelir. Benzer şekilde, daha küçük servisleri bir araya getirerek daha büyük servisler oluşturulabilir. Bir iş sürecinin iş servislerinin bir

toplamı olduğu düşünüldüğü gibi, bir iş servisi de diğer küçük ve orta büyüklükte servislerin birleştirilmesiyle yapılandırılabilir. Bu nedenle, bir etki alanı servisinin öğe boyutu bir iş servisinin olağan ayrıştırılmasıdır.

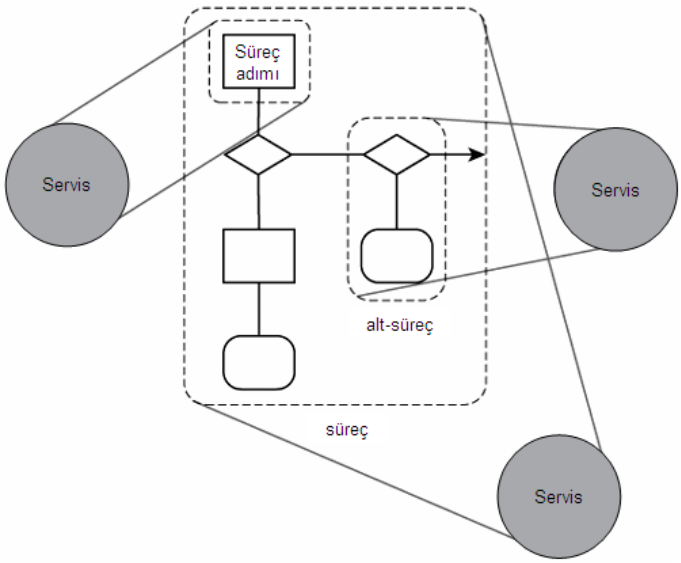
1.2.5. Servis sarmalama

Servisler, bağımsızlıklarını sürdürmek için mantığını belirgin bir bağlam içine sarmalar. Bu bağlam bir iş görevine, iş varlığına ya da diğer mantıksal gruplamaya özgü olabilir.

Bir servisin ele aldığı konu küçük ya da büyük olabilir. Dolayısıyla, servisin temsil ettiği mantığın boyutu ve kapsamı farklılık gösterebilir. Ayrıca, servis mantığı diğer servislerin sunduğu mantığı da kapsayabilir. Bu durumda, bir ya da birden fazla servis toplu bir serviste bir araya gelir [4].

Örneğin, iş otomasyonu çözümleri genellikle bir iş sürecinin uygulanmasıdır. Bu süreç, çözüm tarafından gerçekleştirilen işlemleri belirten bir mantıktan oluşur. Mantık, iş kuralları ve çalıştırma süresi koşullarına göre önceden tanımlı sırada yürütülen bir adımlar dizisine ayrılır.

Şekil 1.4'de gösterildiği gibi, servislerden oluşan bir otomasyon çözümü oluştururken, her servis ayrı bir adım ya da bir adımlar kümesinden oluşan alt süreç tarafından gerçekleştirilen bir görevi sarabilir. Bir servis tüm süreç mantığını da sarabilir. Sonraki iki durumda, servislerin gösterdiği daha büyük kapsam diğer servisler tarafından sarmalanan mantığı içine alabilir.



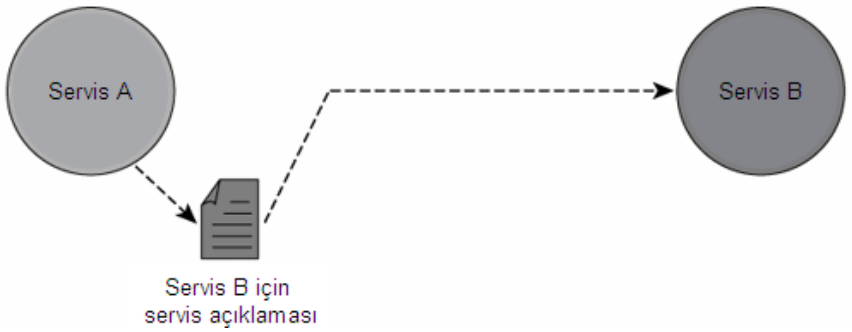
Şekil 1.4 Servisler değişen miktarlarda süreci sarmalayabilir

Servislerin sarmaladığı mantığı kullanmaları için iş etkinliklerinin yürütülmesine katılabilirler. Bunun için, onları kullanmak isteyenlerle ayrı ilişkiler oluşturmaları gerekir.

1.2.6. Servislerin ilişki oluşturması

SOA içinde, servisler diğer servisler ya da diğer programlar tarafından kullanılabilir. Servisle arasındaki ilişki, servislerin birbiriyle etkileşmesi için birbirinin farkında olması gerektiği anlayışına dayanır. Bu farkındalık, servis açıklamalarının kullanılmasıyla gerçekleştirilir.

En temel biçimiyle servis açıklaması, servisin adını ve servis tarafında beklenen ve döndürülen verileri belirtir. Servislerin servis açıklamalarını kullanma şekli, gevşek bağlaşımlı olarak sınıflandırılan bir ilişki kurulmasını sağlar. Örneğin, Şekil 1.5'de, servis A servis B'nin servisine ait olduğu için servis A servis B'nin farkındadır. Servis B'nin servis açıklamasına erişimi olduğu için, servis A, servis B ile iletişim kurmasında gerekli tüm bilgilere sahiptir [5].

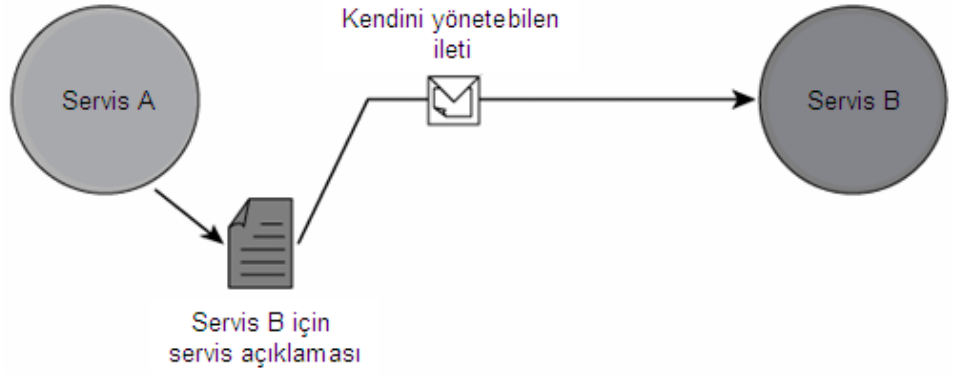


Şekil 1.5 Servis ilişkisi modeli

Servislerin etkileşmesi ve anlamlı bir şeyler gerçekleştirmesi için bilgi alışverişinde bulunmaları gerekir. Dolayısıyla, gevşek bağlaşımlı ilişkilerini koruyabilen bir iletişim çerçevesi gereklidir. Böyle bir çerçeveye örnek olarak ileti alışverişi verilebilir.

1.2.7. Servislerin iletişim kurması

Bir servis kendi yöntemiyle bir ileti gönderdiğinde, iletiye ne olduğuna dair ileti üzerindeki denetimini kaybeder. Bu nedenle iletilerin "bağımsız iletişim birimleri" olarak var olmaları gerekir. Bu da, iletilerin servisler gibi özerk olması gerektiği anlamına gelir. Bu nedenle, iletiler işleme mantığının kendilerine düşen bölümlerini kendi başlarına yönetmeleri için yeterli bir zeka ile donatılabilir (Şekil 1.6) [4].

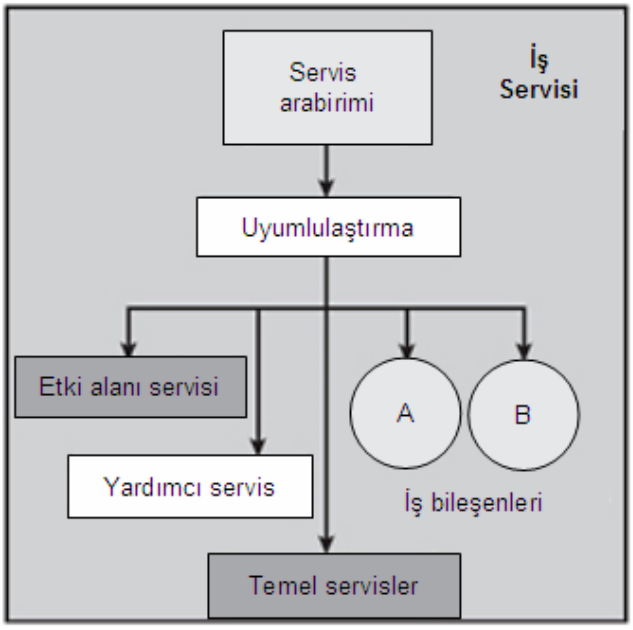


Şekil 1.6 Servis iletişim modeli

1.2.8. Ortak servis modelleri

Tüm servisler eşit oluşturulmamıştır. Farklı boyutlar içindeki kapsamlı değerler aralığı göz önünde bulundurulduğunda, karmaşık çeşitlilikte servisler olduğu görülmektedir. Servis tasarımında kullanılacak bir ortak servis tipleri kümesi tanımlayarak ve ardından bu tipleri uygulama sırasında servis boyutları arasında belirli değerleriyle eşleştirerek işler oldukça basitleştirilebilir. Bu ortak servis tiplerinin SOA uygulamalarında yaygın şekilde görülen servis modellerine de karşılık geldiği görülmektedir. Ortak servis tiplerinin sıra düzeni Şekil 1.2'de gösterilmektedir.

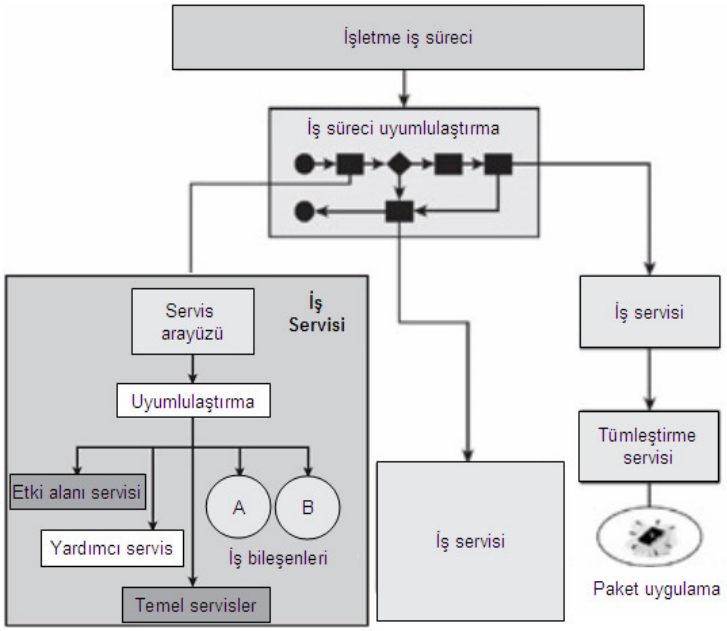
Bir iş servisini gösteren Şekil 1.7'de iş servisi ve iş süreci açıklanmaktadır. Daha üst düzey iş servisi daha alt düzey, etki alanı, yardımcı program ve temel servislerden oluşturulur. Bu farklı servislerin iş servisi oluşturmak için nasıl bir araya geleceğini tanımlamak üzere uyumlulaştırma kullanılır. İş servisi ilk servis arabirimi tarafından açıklanır.



Şekil 1.7 İş servislerinin oluşturulması

Örnek olarak, sigorta ödemesine uygunluğu belirlemeye yönelik bir iş servisi öncelikle kullanıcının servisi yürütme yetkisine sahip olduğunu doğrulamak için temel servisleri kullanacaktır. Ardından, hak talebi doğrulaması gerçekleştirmek için etki alanı servislerini ve adresleri doğrulamak için yardımcı program servislerini kullanacaktır. Son olarak, hak talebinin sigortalının poliçesi kapsamında olup olmadığını belirlemek için özel iş mantığını (iş bileşenleri) ya da kurallarını kullanacaktır.

SOA, genellikle tek bir iş servisinden daha geniş bir kapsama sahiptir. Birden çok ana iş dalından özellikleri birleştirerek iş süreçlerinin işletme düzeyinde nasıl oluşturulacağını tanımlayan bir ana iş dalı ya da işletme kapsamına da sahip olabilir. Şekil 1.8'de bir işletme iş sürecinin iş servislerinin yanı sıra, paketli uygulamalar, eski sistemler ya da kullanıma hazır ticari uygulamalardan nasıl oluşturulduğu gösterilmektedir. Sonraki iki olayda, sistemler işlevlerini servisler gibi uygun şekilde sağlayamadığı için bir tümleştirme servisi tarafından sarılırlar. Ancak, iş sürecinin tümleştirme servisine bir iş servisi üzerinden eriştiğine dikkat edilmelidir. Bu durum, mimari için önemli bir kısıtlamadır. Oluşturulan servislerin sırasını ve etkileşimini (uyumlulaştırma) tanımlamak için bir iş süreci yönetimi sistemi kullanılır.



Şekil 1.8 İşletme iş süreci

İş servisi, iş süreçleri ve servis boyutları bağlamında incelendiğinde:

- İş servisinin kapsamı, genellikle ana iş dalı ya da işletme düzeyinde olan iş sürecinin kapsamını desteklemek için yeterli olmalıdır.
- İş servisinin sahipliği, iş sürecini ve kuruluşlar ya da birimler arasında gerekli işbirliğini desteklemelidir. Doğru sahiplik modeli bir işletmenin yapısına, ilkelerine, finansmanına, güven derecesine dayanır. Burada önemli olan, bir iş sürecinin bir iş servisini kullanması ve bu servise güvenmesi için, bazı güvenilir kuruluşlar servisin sahibi olmalı ve iş süreci sahibine bir garanti düzeyi sunmalıdır.
- İş servisinin öğe boyutu, iş süreci içinde genel bir etkinlikte gerçekleştirilen işlevin öğe boyutuyla iyi düzeyde eşleşmelidir. Bu genellikle büyük ya da orta boyutlu bir servistir.
- İş servisinin oluşturulması diğer servislerin oluşturulmasına dayanır. Bu yaklaşımın bazı faydaları, daha küçük birimleri birleştirerek büyük öğe boyutlu iş servislerinin oluşturulmasına olanak tanıması, sınırlı kapsama sahip işletme işlevlerinin dolaylı, sınırlı ve denetimli kapsamın ötesinde açıklanmasına olanak tanıması, iş süreçlerinin ve var olan sistemlerin semantik ve işlevsel uyumsuzluğunun azaltılmasına olanak tanımasıdır [3].

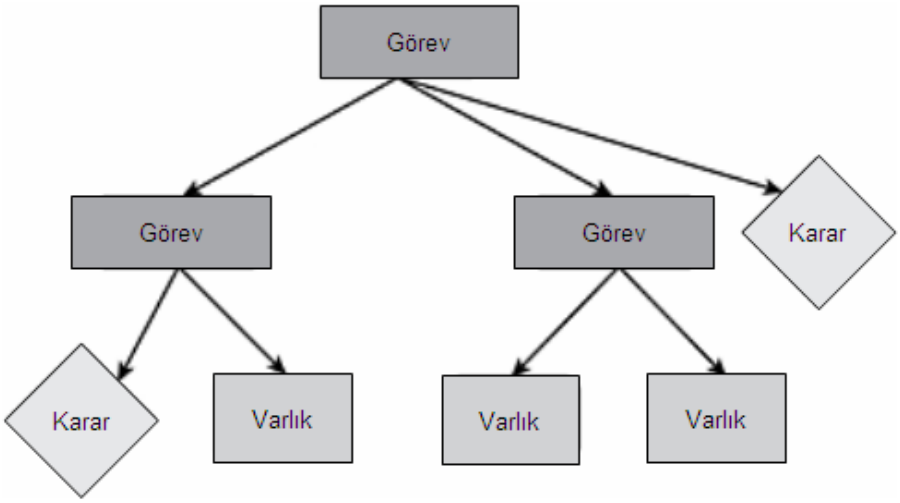
1.2.9. Servis tipleri ve amaçları

Servisin boyutu, kapsamı, sahipliği ve yapılanışından ayrı olarak bir diğer önemli nokta da, servisin hedeflenen amacıdır. Bu farklı servis tiplerini anlamak için, konunun ayrılması mimari ilkesi uygulanabilir. Tasarımcılar, uygulamaları oluşturmada önemli bir kavram olarak verilerin mantıktan ayrılması yaklaşımını uygulamaktadır. Bu yaklaşım, bağlaşık farklı konuların yalnızca birbirinden ayrılmasını değil, bunların uygulanacağı özel ortamları da olanaklı kılar [6].

İş süreç yönetimi (BPM), iş akışının özel bir ortamda yürütülmesini ve yönetilmesini ve işin yeni süreçleri hızlı şekilde modelleyerek gereksinimlere kolayca yanıt vermesini sağlayan iş sürecinin iş akışını ya da şemasını mantığın geri kalanından

ayıran bir örnektir. SOA bunu iş servislerini iş süreçlerinin temel yapı taşları olarak sağlayarak hızlandırır.

Benzer şekilde İş Kuralları Yönetimi de (BRM), kuralların özel bir ortamda yürütülmesini ve yönetilmesini; yeni iş gereksinimlerini desteklemek için kolayca değiştirilmesini sağlayan, iş kurallarının ya da kararlarının uygulama mantığının geri kalanından ayrılmasına yönelik bir örnektir. Aynı şekilde SOA, bu işlemi iş kuralları ve kararlarını açıklayan servisler sağlayarak hızlandırır (Şekil 1.9'a bakınız).



Şekil 1.9 Servis kullanımı ve bağımlılığı modeli

Servis katmanları, genellikle geniş kapsamlı üç servis kategorisi ile yapılandırılır:

Görev servisleri: Bir sigorta fiyat teklifinin fiyatını hesaplayan ya da bir adresin biçimini doğrulayan bir servis gibi bir işletme işlevini uygulayan servislerdir. Görev servisleri, ayrı yardımcı program servislerinden büyük iş servislerine kadar farklı boyutlarda olabilir. Daha küçük servisler amaç olarak daha geneldir ve yeniden kullanım için daha yüksek potansiyele sahiptir. İş servisleri genellikle daha küçük servislerin büyük birleşimleridir ve bir ya da birden fazla belirli süreci desteklemek için tasarlanabilir. Dolayısıyla, süreçler arasında geniş kapsamlı yeniden kullanım

için daha az potansiyel bulunur. Bu durum başka yeniden kullanılabilir bölümlerden oluşturulduğu için bir sorun oluşturmaz.

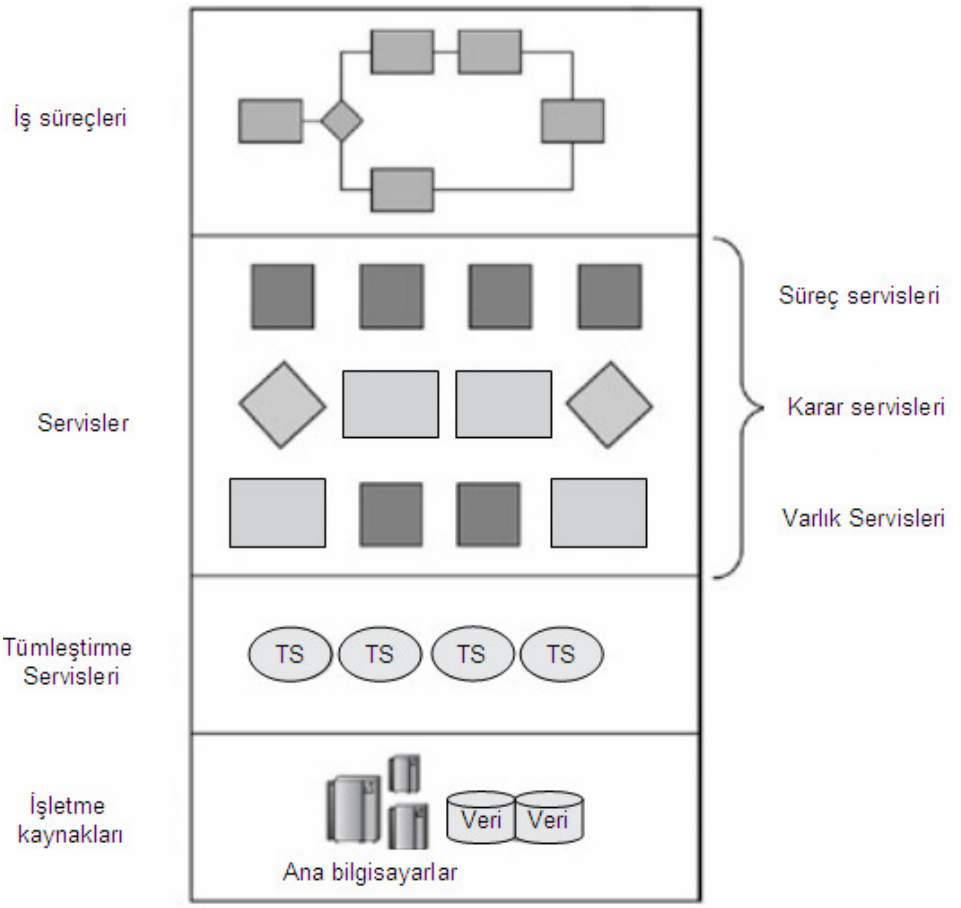
Varlık servisleri: Öncelikle iş varlıklarına erişimi yönetmeye yönelik servisler. İş varlıklarına örnek olarak müşteriler, politikalar, hak talepleri vb. verilebilir. Temel iş bilgisi kavramlarına karşılık gelir. Varlıklar genellikle boyut olarak orta-büyük aralığındadır. Varlıklar, belirli bir iş sürecinden bağımsız olma ve daha çok birden çok farklı iş sürecinin parçası olma eğilimindedir. Varlık servisleri yeniden kullanım için yüksek potansiyele sahiptir. Burada ele alınan, alt düzeyde veri şeması öğeleri değil iş varlıkları olduğunu göz önünde bulundurulmalıdır.

Genel olarak, görev servisleri etkindir ve değer katmak için bir şeyler yapar. Varlık servisleri, görevleri uygulamak için gerekli bilgileri uyarlayarak ve sağlayarak görev servislerini destekler. İş semantiği yerine iç verilerin açıklanmaması için varlık servislerini tasarlarken dikkatli olunmalıdır.

Karar servisleri: İş kararları sağlamak için iş kurallarını yürüten servislerdir. Karar servisine örnek olarak kredi uygunluğu onayı verilebilir. Karar servisleri genellikle karmaşık sorulara evet/hayır yanıtları sağlar ya da vergi yönetmelikleri gibi sık değişen dış kuralları destekler. Karar servisleri çoğunlukla başka servisleri oluşturur ve boyut olarak küçük-orta büyüklüktedir.

Bu farklı servis tipleri, bir iş sürecinin etkinliklerini destekleyen esnek iş becerileri sağlamak için birleştirilebilir. En iyi uygulamalar, bağımlılıkların azaltılmasına, bağlaşımın sınırlandırılmasına ve esnekliğin arttırılmasına yardımcı olan çeşitli modeller, teknikler ve araçlar sunar. Şekil 1.10'da, bağımlılığı azaltmak ve varlık servislerinin yeniden kullanımını artırmaya yönelik tasarlanan tipik bir modelin üst düzey bir gösterimi verilmektedir. Model, birden çok sürecin birleşimini uyumlulaştıran bir görev düzeyi süreç servisini göstermektedir. Her alt süreç bir ya da birden çok varlık servisine erişim sağlar. Süreç servisi, oluşturulmasının bir parçası olarak bir karar servisini de kullanabilir. Ancak, varlık servisinin başka bir varlık servisini doğrudan çağırmasına izin verilmez.

Şekil 1.10 servisi bu ek kavramları dahil edecek şekilde genişletmiştir. Daha önce de olduğu gibi, iş süreçlerinin görevleri servisler tarafından uygulanır (çoğunlukla görev odaklı servisler). Yüksek düzeyde, görev odaklı iş servisleri daha küçük başka servislerden oluşur. Artık süreç, varlık ve karar servisi tiplerinin daha zengin bir kümesini kullanarak yeni ve farklı servis derlemeleri oluşturulabilir. Bu şekilde, esnek ve değiştirilebilir kuralların faydaları, SOA'nın sunduğu modülerlik, esneklik ve yeniden kullanım olanaklarıyla birleştirilebilir.



Şekil 1.10 Servis tipleri ve katmanları

1.3. Servis Odaklı Mimari

SOA kavramını ele almanın birkaç yolu vardır. En temel biçimiyle SOA, yazılımın gerçekte nasıl uygulandığına odaklanan yazılım sistemleri oluşturmaya yönelik bir yaklaşımdır. Bu yaklaşımı ve uygulama şeklini anlamak için, tanımın tek tek bölümlerine ve tipik bir kurumsal SOA uygulamasına bakılması gereklidir.

Aşağıda SOA tanımlarına genel örnekler verilmiştir. Bu tanımların çoğu mimarinin teknik yönlerine odaklanırken, bazıları iş özelliklerini de içerir. Ancak hepsinin birleştiği ortak bir nokta vardır:

İş tanımı: BT ile sorunları azaltmak ya da ortadan kaldırmak ve rekabet avantajı için çevik bir iş ortamı oluştururken BT' nin iş değerini sayısal olarak ölçmek için bir araya gelen iş, süreç, kurumsal, yönetişim ve teknik yöntemlerin bir kümesidir.

En geniş kapsamlı teknik tanım: Gevşek bağlaşımı, yeniden kullanımı ve sistemler arasında birlikte çalışırlığı teşvik eden işletme çapında bir bilgi teknolojisi mimarisidir.

Detaylı teknik tanım: Tüm işlevler ve servislerin bir açıklama dili kullanılarak tanımlandığı ve iş süreçlerini gerçekleştirmek için çağrılabilen arabirimlerin olduğu bir uygulama mimarisidir. Her etkileşim diğer etkileşimlerin her birinden ve tümünden bağımsızdır ve iletişim aygıtlarının protokollerini birbirine bağlar. Arabirimler, platformdan bağımsız olduğu için, bir müşteri bir servisi, herhangi bir dildeki bir işletim sistemini kullanan herhangi bir aygıttan kullanabilir.

Yaygın olmayan ortak payda tanımı: Uygulama işlevlerinin, gevşek bağlaşımlı bileşenler (servisler) olarak oluşturulduğu ve birlikte çalışırlığı desteklemek ve esneklik ile yeniden kullanımı geliştirmek için ayrıntılı şekilde tanımlandığı bir sistem mimarisidir.

En dar tanımı: SOA, SOAP, WSDL ve UDDI gibi web servisi teknolojilerini kullanan çözüm mimarileriyle eşanlamlıdır. Burada SOA, "W3C web servisleri mimarisi (WSA) ile uyumlu herhangi bir ürün ve proje mimarisi" [3] olarak tanımlanır.

1.3.1. Servis odaklı mimari özellikleri ve faydaları

Mimari, farklı kuruluşların anında ortaya çıkan gereksinimlerini karşılayan servisleri bağımsız şekilde uygulamasını; ancak aynı zamanda daha üst düzey iş süreçleri ve işletme çözümleriyle birleştirmesini de sağlar. Bunun için servislerin aşağıdaki özelliklere sahip olması gerekir:

- Aynı boyut, şekil, biçim, işlev ve diğer özelliklere sahip olmaları
- İşletme standartlarına uymaları
- Teknik düzeyde iletişim kurmaları
- Semantik düzeyde iletişim kurmaları
- Sorumluluklar açısından boşluklarının ve çakışmalarının olmaması [7].

Belirtildiği üzere mimarinin aşağıdaki sorulara yanıt vermesi gerekir:

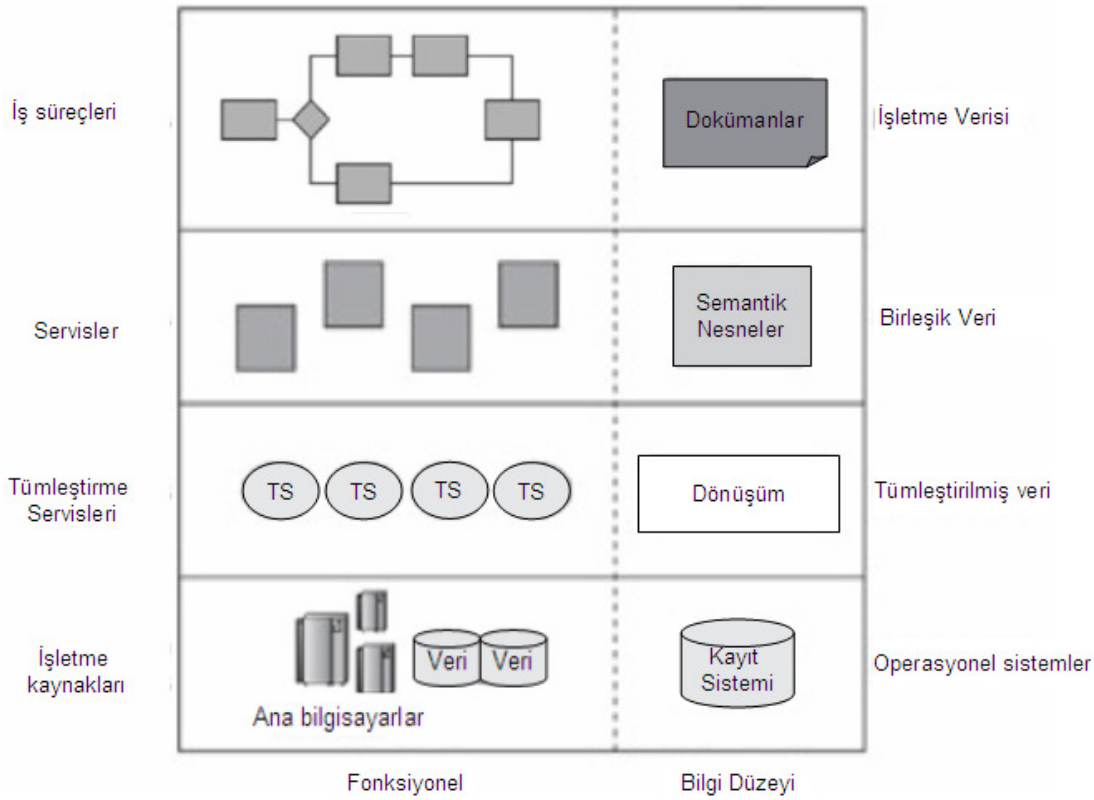
- Önemli parçalar nelerdir?
- Parçalar arasındaki ilişkiler nelerdir?
- Kendilerinden üst düzeyde değer sunmak için nasıl birleşirler?

SOA açısından önemli parçalar şunlardır:

- **Süreçler:** Üst düzey işletme işlevleri, genellikle yayılan uygulamalar ya da ana iş dallarıdır.
- **Servisler:** İşletme işlevlerinin modüler birimleri
- Tümleştirme: Var olan uygulamaların ve/veya verilerin servisler olarak bağlanması ve açıklanması
- **Var olan sistemler:** Var olan eski sistemler, kullanıma hazır ticari uygulamalar (COTS) ve işletmenin kullanmak istediği veriler

- **Belgeler:** Satınalma emri ya da kurumsal belgeler gibi iş bilgilerinin üst düzey birimleri
- **Semantik:** Süreçlerde alışverişi yapılan bilgilerin altta yatan anlamı
- **Dönüştürme:** Bilgilerin bir biçimden ya da semantikten bir diğerine dönüştürülmesi
- **İletişim:** Servislerin birbiriyle iletişim kurma becerisi

Şekil 1.11, her katman için iki önemli kavramla birlikte katmanlı bir SOA mimarisini göstermektedir. Solda, sistemleri ve süreçleri yapılandırmada kullanılan işlevsel kavramlar yer alır. Sağda ise, bu farklı işlevsel düzeylerde verileri geçirmek, tanımlamak ya da işlemek için kullanılan bilgi kavramları bulunur. Diğer bir deyişle, işletmelerin süreç ve bilginin bir birleşimi olduğu gerçeği açıkça görülebilir. Her katman soyutlamaya gerek duyar. Ancak çoğunlukla SOA, önemli veri kavramlarını göz ardı ederek işlevsel konulara odaklanır. Bunun yerine, işlevsel ve veri soyutlamalarının birlikte sunulması gerekir. Katmanlar arasındaki bağlantılar işlevler arasındaki ilişkileri gösterir.



Şekil 1.11 SOA'nın mimari öğeleri

İşletme kaynakları ve operasyonel sistemler: Bu katman, MİY (Müşteri İlişkileri Yönetimi) ve KKP (Kurumsal Kaynak Planlaması) paket uygulamaları ve daha eski nesne odaklı uygulamalar da dahil olmak üzere var olan uygulamalar ile eski ve COTS sistemlerinden oluşur. Bu uygulamalar, işletmenin operasyonel sistemlerinde tek bir mantıksal iş birimini gösteren işletme işlemleri sağar. Bir işlemin yürütülmesi bir ya da birden çok veri kaydının Kayıt Sistemi'nde (KS) okunmasına, yazılmasına ya da değiştirilmesine neden olur. İşlemler belirli ve yapılandırılmış bir arabirime sahiptir ve yapılandırılmış yanıtlar döndürür. Bu katmandaki veriler var olan uygulamalar ya da veritabanlarında bulunur.

Tümleştirme servisleri: Tümleştirme servisleri, var olan uygulamalar arasında tümleştirme ve var olan uygulamalara erişim sağlar. Tümleştirme servisleri ile iş servislerinin ayrılması esnek bir işletme ortamının sürdürülmesi için çok önemlidir. Bu genellikle, veri ve işlevlerin iş servisi düzeyinde istenenden; var olan sistemlerde gerçekte olanaklı olana dönüştürme çalışmalarını içerir.

İş servisleri: İş servisleri, tüm işletme çapında yüksek düzeyde işletme işlevleri sağlar. Bu katman, süreçler ve var olan sistemler arasında doğrudan bağımlılığı kırarak alttaki katmanın servis arabirimi soyutlamasını ve tümleştirmesini sağlar. Servisler, servis düzeyi sözleşmelerine (SLA'lar) uyumu sağlamaktan sorumlu yönetilen işletme varlıkları kümesidir. İş servisleri, işlemlerin mantıksal gruplaması üzerinden iş becerileri sağlar. Örneğin, Müşteri Profilleme bir servis olarak görüntüleniyorsa, içerdiği işlemlerin mantıksal gruplaması şunlar olabilir:

- Telefon Numarasına Göre Müşteri Ara,
- Müşterileri Ada Göre Listele
- Yeni Müşteri Verilerini Kaydet.

Tüm işlemlerin aynı operasyonel sistemlerden gelmeyebileceğini ya da bazı durumlarda, işlemlerin birden çok benzer sistemde çoğaltılabileceğini göz önünde bulundurulmalıdır. Böylece, iş servisleri ilgili işletme işlemlerinin sanal uygulamasını sağlar. İş servisleri semantik veri nesneleri, servisler arasında paylaşılması ya da geçirilmesi gereken bilgileri tanımlayan sanal veriler üzerinde çalışır. Genellikle birden çok var olan sistemden toplanır. İş servisi katmanının çok sayıda farklı servis tipinden oluştuğu göz önünde bulundurulmalıdır.

İş süreçleri: İş süreci, bir iş kuralları kümesine göre belirli bir sırada yürütülen bir dizi işlemden oluşur. İş süreci çoğunlukla, İş Süreci Modelleme Gösterimi (BPMN) ile uyumlu ve özel iş süreci yönetimi sistemi (BPMS) tarafından yürütülen bir İş Süreci Modeli'nde tanımlanır. İşlemlerin sıralaması, seçimi ve yürütülmesine uyumlulaştırma denir. İş süreçleri, uzun süre çalışan bir işlemler ya da etkinlikler kümesi sağlar. İş servislerinden oluşur ve genellikle birden çok servis çağırma işlemini kapsar. İş süreçleri iş belgeleri üzerinde çalışır. Süreçler ve belgeler, İş Süreci Modeli ile ortak semantik veri modeli uyarınca, alttaki katmanın servislerinden ve nesnelerinden oluşur. Bu süreçlerin kapsamı tüm işletmedir. İş süreçlerine örnek olarak şunlar verilebilir:

- Yeni Çalışanı Başlat,

- Ürün ya da Servis Sat,
- Siparişi Karşıla.

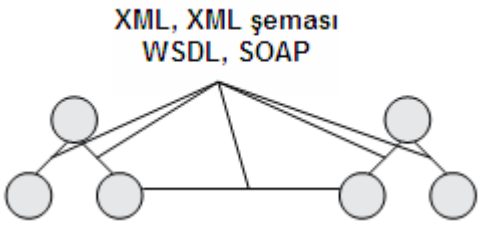
1.3.2. Servis odaklı mimari özellikleri

- SOA, servis odaklı bilgi işlem platformunun merkezinde yer alır.
- SOA servis kalitesini artırır.
- SOA temelde özerktir.
- SOA açık standartlara dayalıdır.
- SOA sağlayıcı çeşitliliğini destekler.
- SOA gerçek birlikte çalışırlığı teşvik eder.
- SOA keşfe ön ayak olur.
- SOA birleştirmeyi teşvik eder.
- SOA mimari oluşturulabilirliği destekler.
- SOA yeniden kullanılabilirliği teşvik eder.
- SOA genişletilebilirliği vurgular.
- SOA servis odaklı iş modellemesi paradigmasını destekler.
- SOA soyutlama katmanlarını uygular.
- SOA işletme çapında gevşek bağlaşımı teşvik eder.
- SOA kurumsal çevikliği destekler.
- SOA bir yapı taşıdır.
- SOA bir evrimdir.
- SOA gelişmeye devam etmektedir.
- SOA gerçekleştirilebilir bir idealdir [4].

SOA açık standartlara dayalıdır: Web servislerinin en önemli özelliği belki de, veri alışverişinin açık standartlar tarafından yönetildiği gerçeğidir. Bir ileti bir web servisinden bir diğerine gönderildiğinde, küresel olarak standartlaştırılmış ve kabul gören bir dizi protokol üzerinden aktarılır.

Ayrıca, iletinin kendisi de hem biçim hem de yükünü nasıl gösterdiği açısından standartlaştırılmıştır. SOAP, WSDL, XML ve XML Şeması'nın kullanılması, iletilerin tam bağımsız olmasını sağlar ve servislerin birbiriyle iletişim kurması için birbirinin servis açıklamaları dışında başka bir bilgiye gerek duymadığı temel uzlaşımını destekler. Açık ve standartlaştırılmış ileti alışverişi modelinin kullanılması, temel servis mantığının tip sistemleri paylaşması gereğini ortadan kaldırır ve gevşek bağlaşımlı paradigmayı destekler [4].

Servis odaklı mimari, bu açık, sağlayıcıdan bağımsız iletişim çerçevesini tam olarak kullanır ve teşvik eder (Şekil 1.12). SOA, özel geliştirilmiş teknolojinin rolünü servisin sarmaladığı uygulama mantığının uygulanması ve barındırılmasıyla sınırlandırır. Servisler arası iletişim fırsatı dolayısıyla her zaman bir seçenektir.



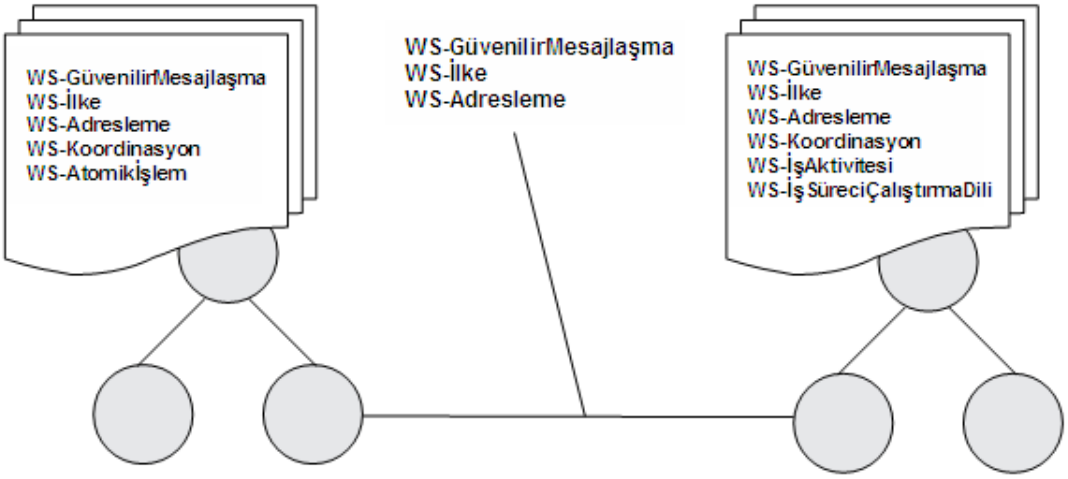
Şekil 1.12 Standart açık teknolojiler çözüm sınırlarının içinde ve dışında kullanılır

SOA mimari oluşturulabilirliği destekler: Oluşturulabilirlik, SOA'nın farklı düzeylerde gerçekleştirilebilen, kökleri derinlere inen bir özelliğidir. Örneğin, SOA oluşturulabilir servislerin geliştirilmesini ve evrimini teşvik ederek, esnek ve son derece yüksek düzeyde uyum gösteren iş süreçlerinin otomasyonunu destekler. Yukarıda belirtildiği gibi servisler mantığın bağımsız birimleri olarak var olurlar. Dolayısıyla bir iş süreci, her biri bir sürecin bir bölümünün yürütülmesinden sorumlu bir dizi servise ayrılabilir.

Oluşturulabilirliğe daha kapsamlı bir örnek olarak, çok sayıda WS-* belirtimi yayın düzeyinden evrimleşmiş ikinci nesil web servisleri çerçevesi verilebilir. Bu

belirtimlerin modüler yapısı SOA'nın yalnızca gerek duyduğu işlevsel yapı taşlarından oluşturulmasına olanak tanır.

Bu esnekliği sağlayansa, ikinci nesil web servisleri belirtimlerinin özellikle SOAP ileti alışverişi modelini kullanacak şekilde tasarlanmış olmasıdır. Ayrı belirtimler, bir ya da birden fazla belirli özelliği sağlayan modüler uzantılardan oluşur. Belirli bir sağlayıcı platformu tarafından desteklenen WS-* uzantılarının sunumu arttıkça, yapılandırma esnekliği yalnızca gerçekten gerek duyulan özellikleri uygulayan çözümler oluşturmaya devam edilmesini olanaklı kılacaktır (Şekil 1.13). Diğer bir deyişle, WS-* platformu düzenli ve iyileştirilmiş servis odaklı mimarilerin, uygulamaların, servislerin ve hatta iletilerin oluşturulmasını sağlar.

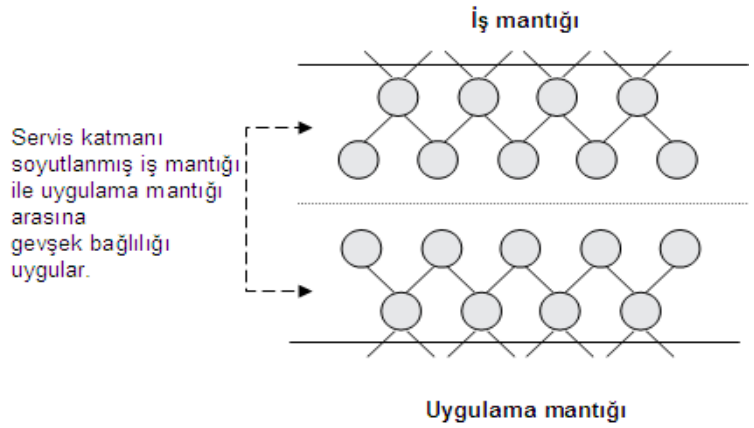


Şekil 1.13 Farklı uzantılardan farklı çözümler oluşturulabilir ve gerekli ortak uzantıları desteklediği sürece birlikte çalışmaya devam edebilirler.

SOA işletme çapında gevşek bağlaşımı teşvik eder: Belirtildiği üzere, gevşek bağlaşımlı servislerle teknik bir mimari oluşturmanın temel faydası servis mantığının bağımsızlığıdır. Servislerin yalnızca birbirinin farkında olması gerekir; böylece bağımsız şekilde evrimleşebilirler.

İşletmeye bir bütün olarak bakıldığında; servis odaklı ilkelerin hem iş modellemeye hem de teknik tasarıma uygulandığı bir işletmede gevşek bağlaşım kavramı yaygın şekilde kullanılır.

Standartlaştırılmış servis soyutlama katmanları uygulanarak, bir işletmenin iş ve uygulama teknolojisi etki alanları arasında, gevşek bağlaşımlı bir ilişki de elde edilebilir (Şekil 1.14). Her ucun yalnızca bir diğerinin farkında olması gerekir. Dolayısıyla her etki alanı daha bağımsız şekilde evrimleşebilir. Bunun sonucunda, kurumsal çeviklik olarak bilinen iş ve teknoloji ile ilgili değişim özelliğini daha iyi barındıran bir ortam elde edilir.



Şekil 1.14 İş ve uygulama mantığını soyutlayan servis katmanları gösterimi.

Servis işi gösterimi, servis soyutlama ve servis katmanlarının kullanımıyla iş ve uygulama mantığı arasında gevşek bağlaşımın kullanılması sayesinde SOA, kurumsal çevikliği arttırır.

2. SERVİS ODAKLI MİMARİ VE WEB SERVİSLERİ

SOA yapısal olarak web servislerine o kadar çok dayalıdır ki; servis odaklı mimariyi gerçekleştirmek için kullanılan web servisleri kavramları ve teknolojisi birçok ortak SOA özelliğini etkilemiş ve bu özelliklere katkıda bulunmuştur. Dolayısıyla, SOA'nın anlaşılması, birinci ve ikinci nesil web servisleri uzantıları tarafından oluşturulan genel çerçeveye yakından bakılmasını gerektirir.

2.1. XML'in Özellikleri

XML'in açılımı Genişletilebilir Bağlantılı Metin Dili'dir ve genellikle büyük harfleri eXtensible Markup Language biçiminde yazılarak akronimi gösterilir. XML, bir belgeyi bölümlerine ayıran ve belgenin farklı bölümlerini tanımlayan semantik etiketleri tanımlamada kullanılan bir kurallar dizisidir. Diğer alana özgü bağlantılı metin dillerinin yazılabileceği bir sözdizimini tanımlayan bir meta bağlantılı metin dilidir.

XML ile ilgili olarak anlaşılması gereken ilk şey, bu dilin, HTML ya da TeX gibi diğer bağlantılı metin dilleri gibi olmadığıdır. Bu diller sabit sayıda öğeyi tanımlayan sabit bir etiketler kümesi tanımlar.

XML, bir meta bağlantılı metin dilidir. Etiketlere gerek duydukça bu etiketlerin oluşturulabildiği bir dildir. Bu etiketler bazı genel ilkelere göre düzenlenmelidir; ancak anlam olarak oldukça esnektirler. Örneğin, soyağacı üzerinde çalışılıyorsa ve soyadların, adların, tarihlerin, doğum tarihlerinin, evlat edinmelerin, ölümlerin, mezarlıkların, ailelerin, evliliklerin, boşanmaların tanımlanması gerekiyorsa, bunların her biri için etiketler oluşturulabilir. Verilerin paragraflara, liste öğelerine, tablo hücrelerine ve bazı çok genel kategorilere uyması için zorlamanıza gerek yoktur [8].

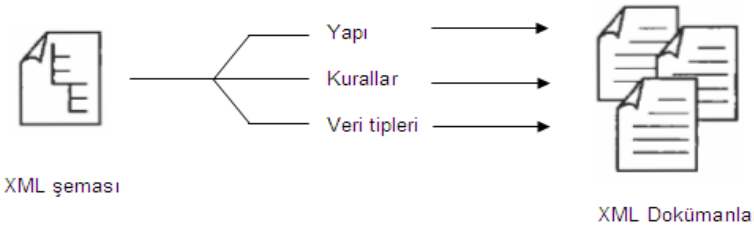
XML ile ilgili olarak ikinci önemli konu, XML bağlantılı metin dilinin belgenin yapısını ve anlamını tanımladığıdır. Sayfadaki öğelerin biçimlendirilmesini tanımlamaz. Biçimlendirme stil sayfası olan bir belgeye eklenebilir. Belgenin kendisi yalnızca belgede ne olduğunu belirten etiketler içerir; belgenin nasıl göründüğünü belirten etiketler içermez.

XML için tasarım hedefleri şunlardır:

- XML doğrudan Internet üzerinden kullanılabilir olmalıdır.
- XML çok çeşitli uygulamaları desteklemelidir.
- XML, Standart Genelleştirilmiş Bağlantılı Metin Dili (SGML) ile uyumlu olmalıdır.
- XML belgelerini işleyen programların yazılması kolay olmalıdır.
- XML içindeki isteğe bağlı özelliklerin sayısı en az sayıda, ideal olarak sıfır olmalıdır.
- XML belgeleri anlaşılır ve açık olmalıdır.
- XML tasarımı hızlı şekilde hazırlanabilmelidir.
- XML tasarımı biçimsel olmalıdır.
- XML belgelerinin oluşturulması kolay olmalıdır.

2.2. XML Şemaları

"XML şemaları XML belgelerinin yapısını tanımlar. Şekil 2.1'de gösterildiği gibi, yapı, doğrulama kuralları, tip kısıtlamaları ve öğeler arası ilişkiler sağlayarak XML belgesinin bütünlüğünü korur. Diğer bir deyişle, XML şemaları XML verileriyle nelerin yapılabileceğini ve nelerin yapılamayacağını belirtir" [9].



Şekil 2.1 XML şemaları ile XML belgeleri arasındaki ilişki [9]

XML Şeması:

- XML belgesinin içeriğini tanımlar ve sınırlandırır
- Özel tiplerin tanımlanmasını sağlar
- Basit tipleri ve karmaşık tiplerin tanımlanmasını sağlar

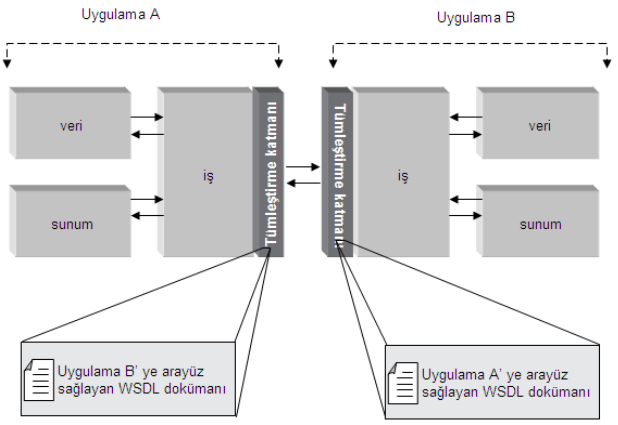
XML Şeması, programlama dillerinde ve veritabanı uygulamalarında yaygın olarak bulunan çoğu tipi gösteren yaklaşık 46 yerleşik basit tipi tanımlar.

2.3. SOAP (Basit Nesne Erişimi Protokolü) ve WSDL (Web Servisi Tanımlama Dili)

Bu bölümde web servis ve basit nesne erişimi protokolünün tanımı yapılmaktadır.

2.3.1. WSDL (Web servis tanımlama dili)

Web servislerinin, diğer servisler ve uygulamalar tarafından keşfedilebilmeleri ve bunlarla arabirim oluşturabilmeleri için tutarlı bir şekilde tanımlanması gerekir. WSDL, web servisi tanımlarının açıklaması için başlıca dili sağlayan bir W3C belirtimidir. Web servisleri çerçevesinin sunduğu tümleştirme katmanı, standart, tüm dünyada kabul gören ve desteklenen bir programlama arabirimi oluşturur. Şekil 2.2'de gösterildiği gibi WSDL, standartlaştırılmış uç nokta tanımlamaları sağlayarak bu katmanlar arasında iletişimi sağlar [9].



Şekil 2.2 Web servislerini uygulamalara gösteren WSDL belgeleri [9]

WSDL servis uç noktası ya da yalnızca uç nokta olarak bilinen bir servis sağlayıcısı için irtibat noktasını tanımlar. Uç nokta arabiriminin biçimsel bir tanımını sağlar (servis sağlayıcısıyla iletişime geçmek isteyen istekte bulunanlar istek iletilerini nasıl yapılandıracaklarını tam olarak bilebilirler) ve servisin fiziksel konumunu (adres) belirtir.

WSDL servis açıklaması (WSDL servis tanımı ya da yalnızca WSDL tanımı olarak da bilinen) iki kategoriye ayrılabilir:

- Soyut açıklama
- Somut açıklama

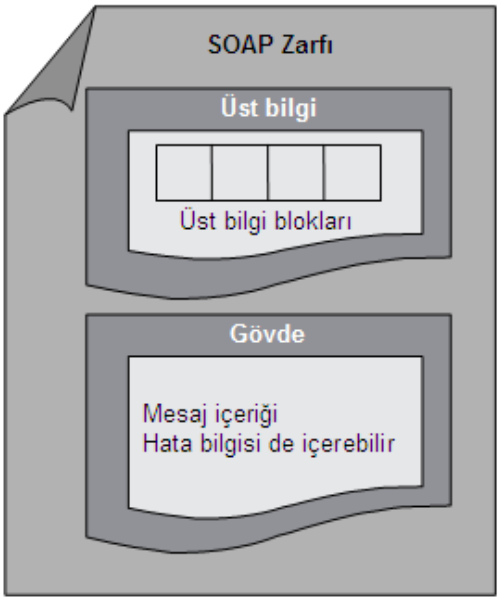
2.3.2. SOAP (Basit nesne erişimi protokolü)

SOAP belirtiminin temel amacı standart ileti biçimini tanımlamaktır. Bu biçimin yapısı oldukça basittir; ancak, genişletilebilme ve özelleştirilebilme özelliği SOAP ileti alışverişini, SOA'ların çoğu önemli özelliğinin arkasındaki itici güç olarak konumlandırmıştır.

Zarf, üstbilgi ve gövde: Her SOAP iletisi zarf olarak bilinen bir barındırıcı içinde paketlenmiştir. Bu metaforun anlattığı gibi zarf, iletinin tüm parçalarını barındırmaktan sorumludur.

Her ileti, meta bilgilerin yerleştirilebileceği bir üstbilgi alanı içerebilir. Çoğu servis odaklı çözümde, bu üstbilgi bölümü genel mimarinin çok önemli bir parçasını oluşturur ve isteğe bağlı bir alan olmasına rağmen genelde kullanılır. Önemi, çok sayıda uzantının uygulanabileceği üstbilgi bloklarının kullanımıyla ilgilidir [9].

Gerçek ileti içeriği, genellikle XML biçimlendirilmiş verilerinden oluşan ileti gövdesi tarafından barındırılır. Mesaj gövdesinin içeriğine genellikle mesajın içeriği adı verilir. (Şekil 2.3)



Şekil 2.3 SOAP iletisinin temel yapısı

Her ileti, meta bilgilerin yerleştirilebileceği bir üstbilgi alanı içerebilir. Çoğu servis odaklı çözümde, bu üstbilgi bölümü genel mimarinin çok önemli bir parçasını oluşturur ve isteğe bağlı bir alan olmasına rağmen genelde kullanılır. Önemi, çok sayıda uzantının uygulanabileceği üstbilgi bloklarının kullanımıyla ilgilidir.

Gerçek ileti içeriği, genellikle XML biçimlendirilmiş verilerinden oluşan ileti gövdesi tarafından barındırılır. Mesaj gövdesinin içeriğine genellikle mesajın içeriği adı verilir.

2.4. Web Servislerinin Özellikleri

W3C'nin Web Servisleri Mimarisi Çalışma Grubu, web servisinin tanımı üzerinde aşağıdaki ortak görüş üzerinde uzlaşmaya varmışlardır:

"Web servisi, arabirimleri ve bağlantıları XML yapay olguları olarak tanımlanabilen, açıklanabilen ve keşfedilebilen bir URI tarafından tanımlanan bir yazılım uygulamasıdır. Web servisi, Internet tabanlı protokoller üzerinden gönderilip alınan

XML tabanlı iletiler kullanan diğer yazılım aracılarıyla doğrudan etkileşimleri destekler ” [10].

Temel web servisleri her yerde kullanılan iki teknolojinin güçlerini birleştirir: Evrensel veri açıklama dili olan XML ile tarayıcılar ve Web sunucuları tarafından kapsamlı şekilde desteklenen HTTP aktarma protokolü.

Web servisleri = XML + aktarma protokolü (HTTP gibi)

Web servislerinin bazı temel özellikleri şunlardır:

- Web servisleri bağımsızdır.

  İstemci tarafında başka bir yazılıma gerek yoktur. XML ve HTTP istemci desteğine sahip bir programlama dili, başlamanız için yeterlidir. Sunucu tarafında, yalnızca bir Web sunucusu ile sunucu uygulamacığı motoru gereklidir. Web servisinin tek bir kod satırı yazılmaksızın var olan bir uygulamayı etkinleştirmesi mümkündür.

- Web servisleri kendinden açıklamalıdır.

  Ne istemci ne de sunucu istek ve yanıt mesajlarının biçimi ve içeriği dışında başka hiçbir şey bilmez ve gerek duymaz (gevşek bağlaşımlı uygulama tümleştirmesi). İleti biçiminin tanımı iletiyle birlikte aktarılır. Dış meta veri havuzlarına ya da kod oluşturulmasına gerek yoktur.

- Web servisleri modülerdir.
  Web servisleri, işletme işlevlerini Web üzerinden konuşlandırmak ve bu işlevlere erişmek için kullanılan bir teknolojidir; J2EE, CORBA ve diğer standartlar bu web servislerini uygulamaya yönelik teknolojilerdir.

- Web servisleri Web üzerinden yayımlanabilir, bulunabilir ve çağrılabilir.

Bunun için gerekli standartlar şunlardır:

- SOAP (Basit Nesne Erişimi Protokolü),
- WSDL (Web Servisi Tanımlama Dili),
- UDDI (Evrensel Açıklama, Keşif ve Tümleştirme), web servisi açıklamalarını aramak için kullanılabilen bir sistem kütüğü mekanizmasıdır.

- Web servisleri dilden bağımsızdır ve birlikte çalışabilir özelliktedir.

  Servis sağlayıcısı ile servis isteğinde bulunan arasındaki etkileşim tamamıyla platformdan ve dilden bağımsız olacak şekilde tasarlanmıştır. Bu etkileşim WSDL belgesinin, bir ağ protokolüyle birlikte (genellikle HTTP) arabirimi tanımlamasını ve hizmeti açıklamasını gerektirir. Servis sağlayıcı ile servis isteğinde bulunan birbirinin kullandığı platformlar ve dillere ilişkin bir fikir sahibi olmadığı için, birlikte çalışırlık özelliği sunulur.

- Web servisleri yapısal olarak açık ve standart tabanlıdır.

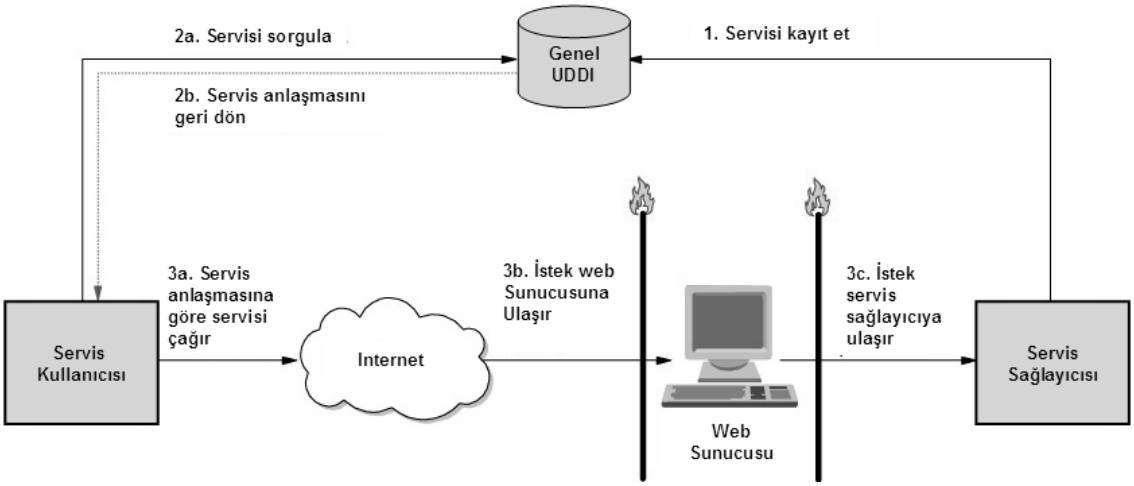
  XML ve HTTP, web servislerinin teknik temelidir. Web servisi teknolojisinin büyük bir bölümü açık kaynak projeleri kullanılarak oluşturulmuştur. Dolayısıyla sağlayıcıdan bağımsız olma ve birlikte çalışırlık gerçekçi hedeflerdir.

- Web servisleri dinamiktir.

  Dinamik e-iş, UDDI ve WSDL ile web servisi açıklaması ve keşfi otomatikleştirilebileceği için, web servisleri kullanılarak gerçekleştirilebilir.

- Web servisleri oluşturulabilir niteliktedir.

Basit web servisleri, iş akışı teknikleri kullanılarak ya da web servisi uygulamasından alt katman web servisleri çağrılarak daha karmaşık servislerde bir araya getirilebilir.



Şekil 2.4 SOA modeline dayalı tipik bir Web servisi işbirliği

2.4.1. Web servis rolleri

Bir web servisi, kullanıldığı ilişkilendirmeye bağlı olarak farklı roller üstlenebilir. Örneğin, bir servis bir iletinin başlatıcısı, aktarıcısı ya da alıcısı olabilir. Dolayısıyla bir servis açık şekilde istemci ya da sunucu olarak etiketlenemez; ancak bunun yerine belirli bir senaryodaki işleme sorumluluğuna dayalı olarak rolünü değiştirebilen bir yazılım birimi olarak etiketlenir.

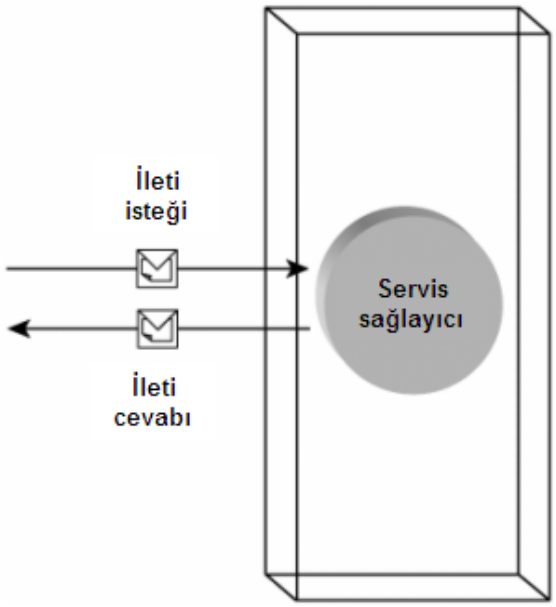
Bir web servisinin rolünü belirli bir iş görevinde birden çok defa değiştirmesi sık görülen bir durumdur. Özellikle de SOA içindeki bir web servisinin farklı iş görevlerinde farklı roller üstlenmesi sık görülür.

2.4.2. Servis sağlayıcısı

Servis sağlayıcısı rolü aşağıdaki koşullarda web servisi tarafından üstlenilir:

- Web servisi servis isteğinde bulunan gibi dış bir kaynak tarafından çağrılır.

- Web servisi, kendi özellikleri ve işleyişine ilişkin bilgi sağlayan yayınlanmış bir servis açıklaması sunar.



Şekil 2.5 İstek iletisinin alıcısı olarak Web servisi, bir servis sağlayıcısı olarak sınıflandırılır [5].

Servis sağlayıcısı rolü, klasik istemci-sunucu mimarisinde sunucu rolüyle eşanlamlıdır. Servis sağlayıcısını çağırırken kullanılan ileti alışverişi tipine bağlı olarak, servis sağlayıcısı bir istek iletisine bir yanıt iletisiyle yanıt verebilir.

Servis sağlayıcısı terimi, web servisini sunmaktan sorumlu kuruluşu (ya da kişiyi) tanımlamak için de kullanılır. Servis rolünü servisin gerçek sağlayıcısından ayırmaya yardımcı olması için, aşağıdaki daha nitelikli terimler bazen kullanılır:

- Servis sağlayıcı varlığı (web servisini sunan kuruluş ya da kişi)
- Servis sağlayıcı aracısı (sahibinin adına aracı olarak hareket eden web servisinin kendisi) [4].

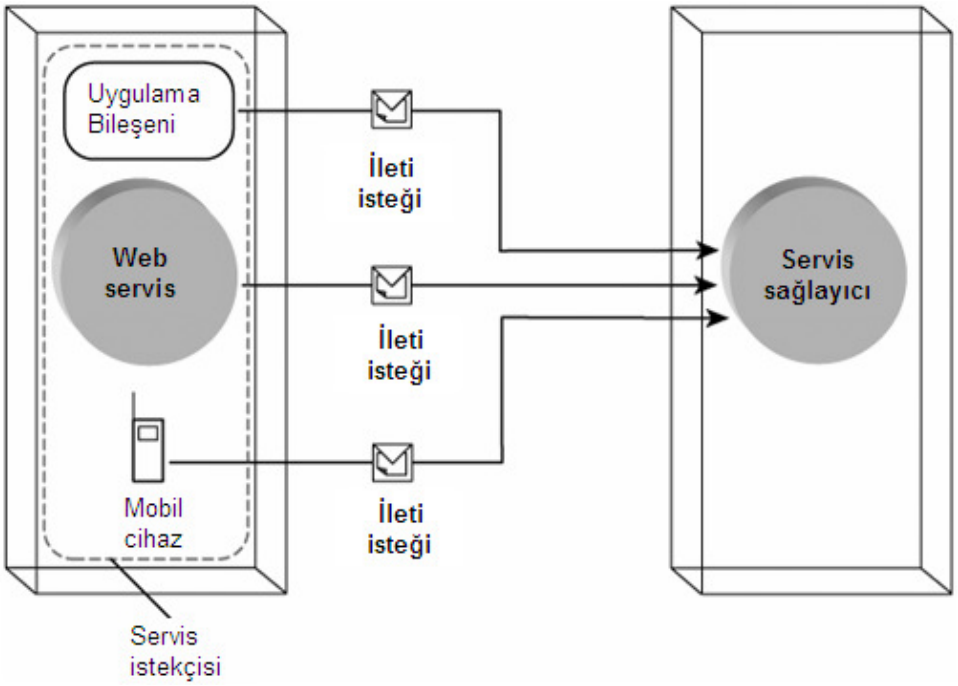
Ancak, çağrılan servisi servis sağlayıcı olarak adlandırmak çok yaygın şekilde görülür.

2.4.3. Servis isteğinde bulunan

Servis sağlayıcı tarafından anlaşılabilen bir istek iletisi yayınlayabilen bir işleme mantığı birimi servis isteğinde bulunan olarak sınıflandırılabilir. Web servisi her zaman bir servis sağlayıcısıdır; ancak servis isteğinde bulunan olarak da işlev görebilir.

Web servisi aşağıdaki koşullarda servis isteğinde bulunan rolünü alır:

- Web servisi ileti göndererek bir servis sağlayıcısını çağırdığında
- Web servisi, kullanılabilen servis açıklamalarını inceleyerek en uygun servis sağlayıcısını arayıp değerlendirdiğinde



Şekil 2.6 İstek iletisinin göndericisi, servis isteğinde bulunan olarak sınıflandırılır[5].

Servis isteğinde bulunan, tipik istemci-sunucu ortamında istemciye karşılık gelir ve servis sağlayıcının doğal karşılığıdır. Servis sağlayıcının bir istek iletisine yanıt olarak bir ileti gönderdiğinde servis isteğinde bulunan olarak hareket etmediğinin belirtilmesi gerekir. Servis isteğinde bulunan, bir servis sağlayıcısı ile bir sohbet başlatan bir yazılım programı olarak görülebilir [4].

Servis sağlayıcısında, olduğu gibi bu terimde de bazı netleşmemiş yönler bulunur. Servis isteğinde bulunan hem web servisinin kendisini hem de web servisinin sahibini temsil edebilir. Dolayısıyla genişletilmiş terimler bulunur:

- Servis isteğinde bulunan varlık (web servisini isteyen kuruluş ya da kişi)
- Servis isteğinde bulunan aracı (sahibinin adına aracı olarak hareket eden web servisinin kendisi)

2.5. Servis Modelleri

Servis modelleri, servislerin gerçek dünyada kullanılma şeklidir; sundukları uygulama mantığının yapısının yanı sıra genel çözüm içindeki işle ilgili rollerine dayalı bir sınıflandırmaya sahiptirler. Bu sınıflandırmalar servis modelleri olarak bilinirler.

2.5.1. İş servisi modeli

SOA içinde iş servisi en temel yapı taşını gösterir. İyi tanımlanmış işlev sınırı içinde belirgin bir iş mantığı kümesini kapsar. Tamamıyla özerktir; ancak iş servislerinin sıklıkla servis yapılanışlarına katılması beklendiği için tek başlarına yürütülme sınırlaması yoktur.

İş servisleri SOA'lar içinde aşağıdaki şekillerde kullanılır:

- İş mantığını temsil eden temel yapı taşları olarak
- Kurumsal bir varlığı ya da bilgi kümesini göstermek için
- İş süreci mantığını göstermek için
- Servis oluşturma üyeleri olarak

Soyutlama katmanları etrafında bir SOA oluştururken, iş servisi modeli iş servisi katmanına karşılık gelir.

2.5.2.Yardımcı program servis modeli

Olası yeniden kullanım için tasarlanmış genel bir web servisi ya da servis aracısı yardımcı program servisi olarak sınıflandırılabilir. Bu sınıflandırmayı elde etmenin anahtarı, yeniden kullanılabilir işlevlerin tamamıyla genel olması ve yapısı gereği uygulamaya özgü olmamasıdır.

Yardımcı program servisleri SOA'lar içinde aşağıdaki şekillerde kullanılır:

- SOA içinde yeniden kullanım özelliğini etkinleştiren servisler olarak çözümden bağımsız ara servisler olarak
- SOA'nın yapısında bulunan birlikte çalışırlık özelliğini teşvik eden servisler olarak en yüksek düzeyde özerkliğe sahip servisler olarak

2.5.3.Denetleyici servis modeli

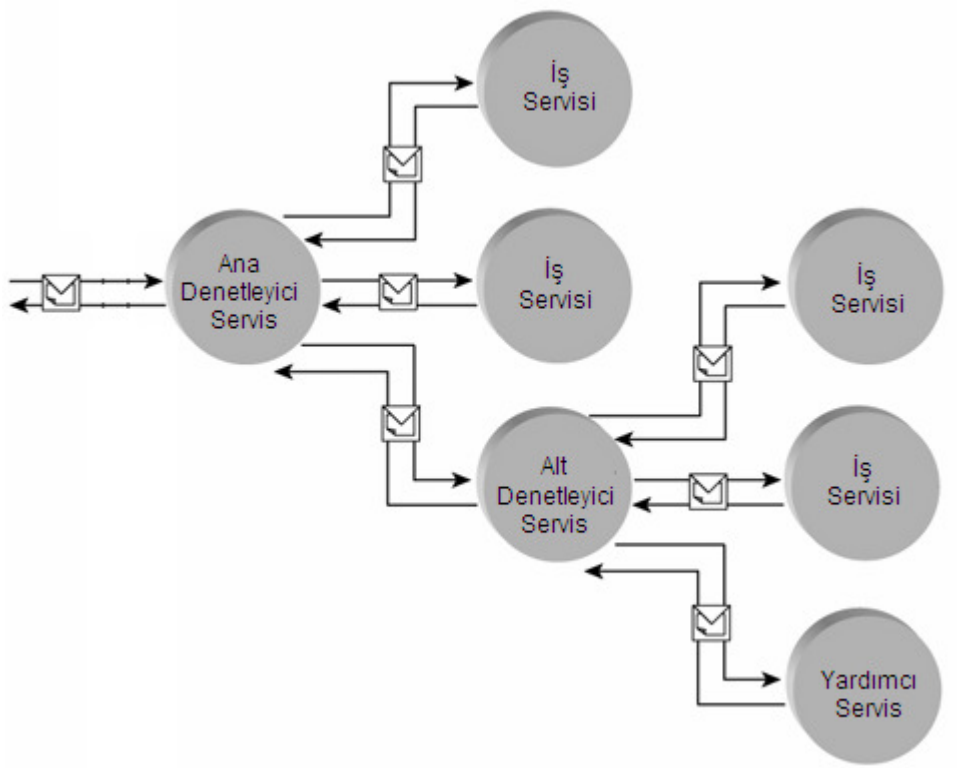
Servis yapılanışları, her biri genel iş görevinin yürütülmesine katkıda bulunan bağımsız bir servisler kümesinden oluşur. Bu servislerin derlenmesi ve koordinasyonu genellikle servisin kendi içinde bir görevdir ve adanmış bir servisin birincil işlevi olarak atanabilir ya da iş görevini tamamıyla bağımsız olarak yürütebilen bir servisin ikincil işlevi olabilir. Denetleyici servisi bu rolü, servis oluşturma üyelerinin üst servisi olarak hareket ederek yerine getirir.

Denetleyici servisleri SOA'lar içinde aşağıdaki şekillerde kullanılır:

- Oluşturabilirlik ilkesini desteklemek ve uygulamak için
- Yeniden kullanım fırsatlarını değerlendirmek için
- Diğer servislerde özerkliği desteklemek için

Denetleyici servislerinin kendilerinin alt servis oluşturma üyeleri olabileceğini göz önünde bulundurun. Bu durumda oluşturma işlemi tümüyle bir denetleyici tarafından koordine edilerek daha büyük bir oluşturma işlemine dönüştürülür. Bu durumda tüm servis oluşturma işleminin üst servisi olarak işlev gören bir ana denetleyici servisinin

yanı sıra oluşturma işleminin bir bölümünü koordine etmekten sorumlu bir alt denetleyici servisi de bulunabilir.



Şekil 2.7 Ana denetleyici, alt denetleyici, dört iş servisi ve bir yardımcı program servisinden oluşan servis oluşturma işlemi [5].

2.6. İkinci Nesil Web Servisleri / WS-*Uzantıları

Web servisleriyle ilişkili çok çeşitli belirtimler bulunur. Bu belirtimler çeşitli tamamlanmışlık düzeylerindedir ve çeşitli standartlar kurumları ve kuruluşları tarafından desteklenir ya da sürdürülür. Belirtimler birbirini tamamlayabilir, birbiriyle örtüşebilir ya da birbiriyle rekabet edebilir. Web servisi belirtimlerine toplu olarak genellikle "WS-*" denilir; ancak, bu terimin sürekli olarak başvurduğu tek bir yönetilen belirtimler kümesi ya da tüm belirtimlerin sahibi bir kuruluş yoktur. Referans terimi olan "WS-*" daha çok, çoğu belirtimin "WS-" örnek gibi kullanılarak adlandırılmasında genel bir düğüm olarak kullanılır [11].

2.6.1.Bağlam ve işlem yönetimi

İlk web servisleri teknolojileri, servis etkinliği boyunca bağlamın yapılandırılmış şekilde sürdürülmesini destekleme becerisinden yoksundu. Etkin ve durumlu bir bağlam olmadan, web servisleri bağımsız olarak hareket eder ve dağıtılmış işlemleri destekleyemezdi. Bu problemi çözmek için WS-Coordination ve WS-Transaction belirtimi tasarlanmıştır.

WS-Coordination belirtimi, WS-Transaction belirtiminde açıklanan protokolleri kullanarak atomik ve uzun süre çalışan işlemleri desteklemek için uygulanan bir bağlam yönetim sistemi sağlar.

WS-Transaction belirtiminin atomik işlem bölümü, WS-AtomicTransaction adlı ayrı bir belirtim tarafından geçersiz kılınır. Benzer şekilde WS-BusinessActivity belirtimi WS-Transaction içindeki ilgili koordinasyon tipi tanımının yerini alır [12].

Referansta Bulunulan Belirtimler:

- **WS-Transaction:** Web Servisleri İşlemleri belirtimleri, web servisleri etki alanları arasında işlemsel birlikte çalışırlık için mekanizmaları tanımlar ve servisin işlemsel özelliklerini web servisleri uygulamalarına dönüştürecek araçlar sağlar.

  Web Servisleri İşlemleri belirtimleri şunlar için genişletilebilir bir koordinasyon çerçevesi (WS-Coordination) ve belirli koordinasyon tipleri tanımlar:
  - Kısa süreli, ACID işlemleri (WS-AtomicTransaction)
  - Uzun süre çalışan işletme işlemleri (WS-BusinessActivity)

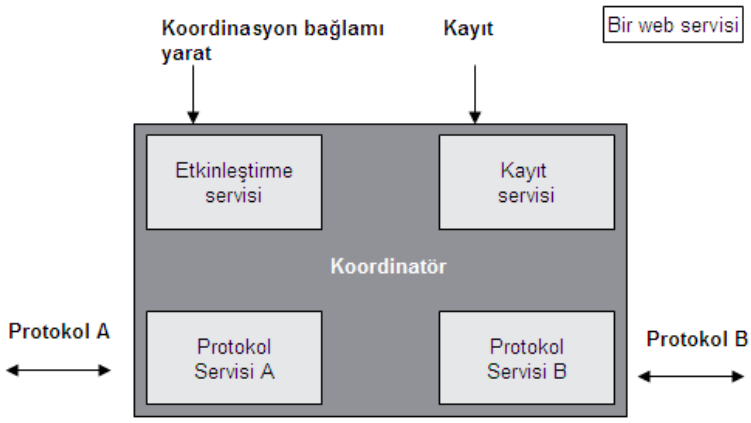
- **WS-Coordination :** Web servisi belirtimlerinin (SOAP [SOAP 1.1] [SOAP 1.2] ve WSDL) geçerli kümesi, web servisi birlikte çalışırlığı için protokoller tanımlar. Web servisleri, çok büyük dağıtılmış bilgi işlem birimleri oluşturan çok sayıda katılımcıyı bir araya getirmektedir; bu bilgi işlem birimlerine

etkinlikler denilmektedir. Elde edilen etkinlikler genellikle, katılımcıları arasında karmaşık ilişkiler nedeniyle yapı olarak karmaşıktır. Bu tür etkinliklerin yürütülmesinin tamamlanması çoğunlukla, iş gecikmeleri ve kullanıcı etkileşimleri nedeniyle uzun sürer.Bu belirtim, bir koordinatör ve bir dizi koordinasyon protokolü kullanarak etkinliklerin koordinasyonu için genişletilebilir bir çerçeve tanımlar. Bu çerçeve katılımcıların dağıtılmış etkinliklerin sonucu üzerine tutarlı bir uzlaşmaya varmalarını sağlar. Bu çerçevede tanımlanabilen koordinasyon protokolleri, basit kısa süreli işlemler için protokoller ve karmaşık uzun süreli iş etkinlikleri için protokollerin de aralarında olduğu çok çeşitli etkinlikleri barındırabilir. Örneğin, WS-AtomicTransaction [WSAT] ve WS-BusinessActivity [WSBA] belirtimleri bu belirtimi kullanır ve bu belirtim üzerine kurulur [12].

WS-Coordination Modeli : Bu belirtim, aşağıdaki bileşen servislerinden oluşan bir koordinasyon servisi (ya da koordinatör) için bir çerçeve tanımlar:

- Bir uygulamanın bir koordinasyon örneği ya da bağlamı oluşturmasını sağlayan bir işleme sahip etkinleştirme servisi.
- Uygulamanın koordinasyon protokollerini kaydetmesini sağlayan bir işleme sahip bir kayıt servisi.
- Koordinasyon protokollerinin koordinasyon tipine özgü kümesi.

WS-Coordination modeli Şekil 2.8'de gösterilmektedir.



Şekil 2.8 WS-Coordination modeli

Uygulamalar, bir etkinlik için koordinasyon bağlamı oluşturmak üzere Etkinleştirme servisini kullanır. Koordinasyon bağlamı bir uygulama tarafından alındığında, ardından uygun araçlarla başka bir uygulamaya gönderilir.

Bağlam, uygulamanın izleyeceği koordinasyon davranışını belirleyen etkinliğe kayıt yaptırmak için gerekli bilgileri içerir.

Ayrıca, koordinasyon bağlamını alan bir uygulama özgün uygulamanın Kayıt servisini kullanabilir ya da aracılık eden güvenilir bir koordinatör tarafından belirtilen servisi kullanabilir. Bu şekilde web servislerinin rastgele bir toplamı ortak işlemlerini koordine edebilir.

- **WS-AtomicTransaction:** Atomik işlemler ya-hep-ya-hiç özelliğine sahiptir. Bir işlem katılımcısı tarafından kesinleştirmeden önce yapılan işlemler yalnızca deneme amaçlıdır; genellikle kalıcı ve işlemin dışında görünür değillerdir. Bir uygulama bir işlem üzerinde çalışmayı tamamladığında, koordinatörden işlemin sonucunu belirlemesini ister. Koordinatör, katılımcılardan bildirim isteyerek işleme hatası olup olmadığını belirler. Katılımcıların tümü başarılı şekilde yürüttüklerine ilişkin bildirimde bulunursa, koordinatör yapılan tüm işlemleri kesinleştirir. Katılımcılardan biri işlemi iptal etmesi gerektiğini bildirirse ya da hiç yanıt vermezse, koordinatör yapılan tüm işlemleri iptal eder. Kesinleştirme işlemi katılımcıları deneme işlemlerini son durumuna getirmeye yönlendirir; örneğin işlemi kalıcı yapıp işlemin dışında da görünür kılmak gibi. İptal işlemi katılımcıları deneme işlemlerini sanki hiç olmamışlar gibi yapmaya yönlendirir. Atomik işlemlerin bir çok uygulama için son derece değerli olduğu kanıtlanmıştır. Tutarlı hata ve kurtarma semantiği sağarlar; böylece uygulamalar, üzerinde karşılıklı uzlaşılan bir sonuç kararını belirlemenin mekanizmasıyla uğraşmak ya da çok sayıda olası tutarsız durumda kurtarma işleminin nasıl yapılacağını bulmak zorunda kalmazlar.

Bu belirtim Atomik İşlemlerin sonucunu yöneten protokolleri tanımlar. Var olan işlem yürütme sistemlerinin, mülkiyete dayalı mekanizmalarını sarmak ve farklı sağlayıcı uygulamalarında birlikte çalışabilmek için WS-AtomicTransaction belirtimini kullanması beklenir.

Bu belirtim Atomik İşlemler için aşağıdaki protokolleri tanımlar:

- **Tamamlanma:** Tamamlanma protokolü kesinleştirme işlemini başlatır. Her protokolün kayıtlı katılımcılarına dayalı olarak koordinatör Geçici (2 Aşamalı Kesinleştirme) 2PC ile başlar ve ardından Kalıcı 2PC ile devam eder. Nihai sonuç başlatıcıya gönderilir.
- **2 Aşamalı Kesinleştirme (2PC):** 2PC protokolü kayıtlı katılımcıların bir kesinleştirme ya da iptal kararına varmalarını koordine eder ve nihai sonucun tüm katılımcılara bildirilmesini sağlar. 2PC protokolünün 2 türü vardır:
  - **Geçici 2PC:** Bu protokol için önbellek sistem kütüğü gibi geçici kaynakları yöneten katılımcılar.
  - **Kalıcı 2PC:** Bu protokol için veritabanı sistem kütüğü gibi kalıcı kaynakları yöneten katılımcılar.

Katılımcılar bu protokollerden sadece birine kaydolabilir.

- **WS-BusinessActivity:** Bu belirtim, bir iş sürecinin etkinliklerini yürütürken ortaya çıkan istisnaları işlemek için iş mantığı uygulayan etkinlikleri koordine etmede kullanılan iki İş Etkinliği koordinasyon tipinin tanımını sağlar. İşlemler hemen uygulanır ve kalıcıdır. Bir hata durumunda telafi edici işlemler çağrılabilir. WS-BusinessActivity, mülkiyete dayalı mekanizmalarını sarmak ve güven sınırları ile farklı sağlayıcı uygulamaları arasında birlikte çalışabilmek için var olan iş sürecini ve iş akışı sistemini etkinleştiren protokolleri tanımlar. İş etkinlikleri aşağıdaki özelliklere sahiptir:
  - Bir iş etkinliği uzun süre içinde çok sayıda kaynağı tüketebilir.
  - Etkinliğe dahil olan çok sayıda atomik işlem olabilir.

- Bir iş etkinliğindeki ayrı görevler, iş etkinliğinin tamamlanmasından önce görülebilir; bunların sonuçları bilgisayar sistemi dışında etkilere sahip olabilir.
- Bir isteğe yanıt verilmesi çok uzun sürebilir. Bir yanıt göndermeden önce bir kişinin onayı, montaj, üretim ya da teslimat yapılması gerekebilir.
- Bir iş istisnasının bir etkinliğin mantıksal olarak geri alınmasını gerektirdiği bir durumda, iptal işlemi genellikle yeterli olmaz. İstisna işlem mekanizmaları, bir telafi edici görev biçiminde, daha önce tamamlanmış bir görevin etkilerini tersine çevirmek için iş mantığına gerek duyabilir.
- Bir iş etkinliğindeki katılımcılar, tüm güven ilişkilerinin açık şekilde tanımlandığı farklı etki alanlarında olabilir.

Bu belirtimde tanımlanan İş Etkinliği protokolleri aşağıdaki tasarım noktalarına sahiptir:

- Tüm durum geçişleri, uygulama durumu ve koordinasyon meta verileri dahil olmak üzere güvenilir şekilde kaydedilir.
- Nihai olmayan tüm bildirimler, koordinatör ve katılımcı arasında durumun tutarlı bir görünümünü sağlamak için kabul edilir. Bir koordinatör ya da katılımcı iş ortağının durumunu isteyebilir ya da bunu gerçekleştirmek için bildirimleri göndermeyi yeniden deneyebilir.
- Her bildirim ayrı bir ileti olarak tanımlanır. Aktarma düzeyi istek/yanıt yeniden denemesi ve zamanaşımı uzun süre çalışan etkinlikler için uçtan uca anlaşma koordinasyonu gerçekleştirmek için yeterli mekanizmalar değildir.

2.6.2.İş süreci tanımı ve yürütmesi

Web servislerini yapılandırılmış bir iş akışında oluşturmak için, standart bir sözlük gerekir. Web Servisleri için İş Süreci Yürütme Dili, uyumlulaştırmayı destekleyen

ara katman yazılımı tarafından yürütülebilen çalıştırma süresi komut dosyalarında derlenebilen süreç açıklaması sözlüğü sağlar.

WS-BPEL: Web servislerini kurumsal tümleştirme alanına getirir. Sistem tümleştirmesi standart protokoller kullanarak basit etkileşimlerin gerçekleştirilmesinden daha fazlasını gerektirir. Bir tümleştirme platformu olarak Web Servislerinin tam potansiyelinin kullanılması, yalnızca uygulamalar ve iş süreçleri karmaşık etkileşimlerini standart süreç tümleştirme modeli kullanarak tümleştirdiğinde mümkün olur. WSDL tarafından doğrudan desteklenen etkileşim modeli durumsuz istek-yanıt modeli ya da korelasyonsuz tek yönlü etkileşimlerdir [12].

İş etkileşimleri için modeller genellikle eşler arası ileti alışverişi sıralarını, hem istek-yanıt hem de tek yönlü, durumlu, iki ya da daha çok sayıda tarafın dahil olduğu uzun süre çalışan etkileşimler olduğunu kabul eder. Bu tür iş etkileşimlerini tanımlamak için, iş süreçleri tarafından etkileşimlerinde kullanılan ileti alışverişi protokollerinin biçimsel bir açıklamasına gerek vardır. İşleme katılan tarafların iç uygulamalarını açıklamadan her birinin gözlemlenebilir ileti alışverişi davranışını açıklamak için Soyut Süreç kullanılabilir. İş süreci davranışının genel yönlerini iç ya da özel yönlerden ayırmanın iki önemli nedeni vardır. Birincisi, işletmeler kendi şirket içi kararlarını ve veri yönetimi ilkelerini iş ortaklarına kesinlikle açıklamak istemez. Diğeri ise, durum böyle olmasa da, genel sürecin özel süreçten ayrılmasının, süreç uygulamasının özel yönlerinin gözlemlenebilir davranışı etkilemeden değiştirme özgürlüğünü sunmasıdır. Gözlemlenebilir davranış, platformdan bağımsız bir şekilde açıkça tanımlanmalı ve işletme çapında iş önemine sahip davranış yönlerini yakalamalıdır.

2.6.3.Güvenlik

Web servislerin güvenliği sağlamak için özel bir çerçeve tanımı bulunmaktadır.

WS-Security: WS-Security çerçevesi, tamamlayıcı belirtimlerin bir yığınından oluşan kapsamlı bir güvenlik modeli oluşturur. İleti yolu boyunca SOAP iletilerini

korumak için güvenlik önlemleri oluşturur ve ilkelerin oluşturulmasını ve güven sınırlarının birleştirilmesini destekler. Temel WS-Security belirtimleri, bir dizi yerleşik XML güvenlik belirtimi ile eklemeler yapılarak tamamlanmıştır.

Bu belirtim esnektir; PKI, Kerberos ve SSL'in de aralarında olduğu çok çeşitli güvenlik modelleri içinde web servislerinin güvenliğini sağlamak için temel olarak kullanılmak üzere tasarlanmıştır. Bu belirtim özellikle, birden çok güvenlik simgesi biçimi, birden çok güven etki alanı, birden çok imza biçimi ve birden çok şifreleme teknolojisi için destek sağlar. Simge biçimleri ve bunları kullanmaya ilişkin semantik, ilgili profil belgelerinde tanımlanır. Bu belirtim üç temel mekanizma sağlar: ileti, ileti tümleşikliği ve ileti gizliliğinin bir parçası olarak güvenlik simgelerinin gönderilmesi becerisi. Bu mekanizmalar kendi başlarına web servisleri için eksiksiz bir güvenlik çözümü sağlamaz. Bunun yerine bu belirtim, daha kapsamlı güvenlik modelleri ve güvenlik teknolojileri kullanılması için diğer web servisi uzantıları ve uygulamaya özgü daha üst düzey protokollerle birlikte kullanılabilen bir yapı taşıdır. Bu mekanizmalar bağımsız şekilde (örneğin, bir güvenlik simgesini geçirmek için) ya da sıkı bir bağlaşım içinde (örneğin, bir iletinin ya da iletinin bir bölümünün imzalanması ve şifrelenmesi; ardından, imzalama ve şifreleme için kullanılan anahtarlarla ilişkili güvenlik simgesi ya da simge yolunun sağlanması) kullanılabilir [12].

2.6.4.Güvenirlik

Bir çözümün işletme düzeyinde otomasyon gerçekleştirebilmesi için, iletişim çerçevesinin arızalara karşı güvenli, esnek ve verimli olması gerekir. Aşağıdaki WS-ReliableMessaging belirtimleri, güvenilir ileti teslimi, öz yönetimli ileti alışverişi ve ileti ekleri ile ilgili önemli özellikler sunar.

Web Servisleri Güvenilir İleti Alışverişi (WS-ReliableMessaging): Yazılım bileşeni, sistem ya da ağ hatalarında güvenilir şekilde iletişim kurmak isteyen iki web servisi için genellikle bir gereksinimdir. Bu belirtimin temel hedefi iletilerin güvenilir aktarımı için modüler bir mekanizma oluşturmaktır. Kaynak ile hedef

arasında iletilerin güvenilir aktarımını tanımlamak, izlemek ve yönetmek için bir ileti alışverişi protokolü tanımlar. Birlikte çalışırlık için gerekli bir SOAP ilişkilendirme de tanımlar. Ek ilişkilendirmeler de tanımlanabilir.

Birçok hata bir sohbeti kesintiye uğratabilir. İletiler kaybolabilir, yinelenebilir ya da yeniden düzenlenebilir. Ayrıca, anasistemler hatalarla karşılaşabilir ve hassas duruma gelebilir.

WS-ReliableMessaging belirtimi, Güvenilir İleti Alışverişi (RM), hedefi tarafından algılandığı şekilde ilettiği her iletinin yapısını doğru şekilde belirlemek için Güvenilir İleti Alışverişi (RM) Kaynağı'nı etkinleştiren birlikte çalışılabilir bir protokol tanımlar; böylece, iletinin alındığına ilişkin herhangi bir şüpheyi ortadan kaldırır. Protokol aynı zamanda, bir RM Hedefi'nin aldığı iletilerin hangilerini daha önce aldığını etkili şekilde belirlemesini de sağlar; alınıp alınmadığı bilinmeyen bir iletinin RM Kaynağı tarafından yeniden iletilmesinden kaynaklanan yinelenen iletimleri filtreler. RM Hedefi'nin Aldığı iletileri Uygulama Hedefi'ne, iletilerin düzensiz şekilde alınması durumunda bir uygulama kaynağı tarafından gönderildikleri sırada teslim etmesini de sağlar. Bu belirtimin RM Kaynağı ya da RM Hedefi varlıklarının kapsamına hiçbir kısıtlama getirmediği de göz önünde bulundurulmalıdır. Örneğin, ikisi de birden çok WSDL Bağlantı Noktası ya da Uç Noktası'na yayılabilir.

Protokol, sıralı teslim, yineleme kaldırma ve garantili ileti almanın da aralarında olduğu geniş kapsamlı güvenirlik özeliklerinin uygulanmasını sağlar. Protokol tek bir süreç yaşam süresine yayılabilen bellek içi süreklilikten; çoğu uç durumlarda kurtarılabilir olan çoğaltılmış kalıcı saklama alanına kadar uzanan sağlamlık özellikleri de uygulayabilir. Uç noktaların bu güvenirlik özelliklerinin çoğunu ya da birkaçını protokolü kullanan uygulamanın doğru işletimi için gerekli şekilde uygulaması beklenir. Güvenirlik özelliklerinin hangisinin etkinleştirildiğine bakılmaksızın, hat protokolü değişmez.

2.6.5.İlkeler

Servis odaklı işletmede, üst düzey iş kurallarının, güvenlik kurallarının ve açıklayıcı özelliklerin servisler grubuna ilke olarak uygulanabilmesi için soyutlanması faydalı olacaktır.

WS-İlke: WS-İlke çerçevesi, bu tür ilkelerin açıklanmasına olanak tanıyan bir belirtimler kümesinin yanı sıra bunları web servislerine eklemenin standart yollarını içerir. Güvenlik (ve diğer iş) ilkelerinin aracılar ve uç noktalar üzerindeki özelliklerini ve kısıtlamalarını (örneğin, gerekli güvenlik simgeleri, desteklenen şifreleme algoritmaları ve gizlilik kuralları) ve ilkelerin servisler ve uç noktalarla nasıl ilişkilendirileceğini tanımlayan bir belirtimler kümesini gösterir [10].

3. SERVİS ODAKLI MİMARİ PROGRAMLAMA MODELİ

"SOA programlama modeli, SOA ilkelerine, tekniklerine ve en iyi uygulamalara dayalı işletmeleri ve BT dönüşümlerini olanaklı kılmak için günümüzde kullanılmakta olan sayıları her geçen gün artan yazılım teknolojileri, iyi uygulamalar, araçlar ve platformları yönetmede programcıların ve programcı olmayanların karşılaştıkları güçlükleri azaltma çabasıdır" [6]. SOA programlama modeli, yazılım geliştirme disiplininin geçirdiği on yıllara dayalı gelişiminden yararlanan en iyi uygulamaların bir toplamı olarak düşünülebilir. Araçları, teknikleri ve en iyi uygulamaları iki belirtimde birleştirir:

- Servis Bileşeni Mimarisi (SCA)
- Servis Verileri Nesnesi (SDO)

BEA, Oracle, Sun, IBM, RogueWave, Iona, SAP ve diğer önde gelen şirketler tarafından ortaklaşa geliştirilmiş olan bu gelişmekte olan SCA ve SDO standartları, birleşik uygulamaların tutarlı bir şekilde oluşturulmasını kolaylaştırmaktadır. SCA, servisleri kullanmaları için geliştiricilere tek bir programlama modeli sağlarken; SDO veri kaynaklarını göstermeleri ve kullanmaları için onlara yine tek bir programlama modeli sunar. SCA ve SDO standartlarına dayalı bu yeni SOA programlama modelinin, SOA tabanlı uygulamalar ve ekosistemler geliştirmede daha kolay ve basit bir çerçeve olmasını sağlayan bu standartlaştırmadır.

3.1. Servis Bileşeni Mimarisi

SCA, SCA çerçevesinin, uygulamaları hem bileşenin iç uygulamasından hem de erişim yönteminden koruduğu, daha esnek ve dinamik SOA tabanlı uygulamalar geliştirmeye yönlendiren bir belirtimler kümesidir. SCA belirtimi dört bölümden oluşur:

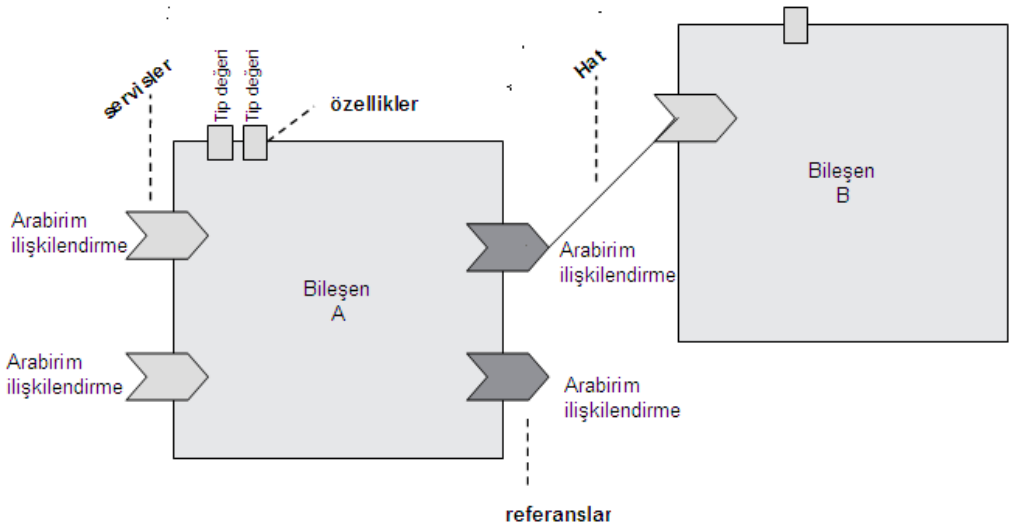
- **Derleme modeli:** Birleşik SOA uygulamalarının nasıl tanımlandığını açıklar.
- **İstemci uygulaması belirtimleri:** SCA standardının farklı dillerde nasıl uygulanacağını tanımlar (örneğin, Java, C++, PHP).

- **İlişkilendirme belirtimleri:** Çeşitli erişim yöntemlerinin nasıl kullanılacağını tanımlar (örneğin, Web Servisleri, JMS, RMI-IIOP, REST).
- **İlke çerçevesi:** Bildirimsel bir şekilde güvenlik, işlemler, sohbetler, güvenilir ileti alışverişinin nasıl ekleneceğini tanımlar [6].

En alt düzeyde, aşağıdaki altı SCA öğesi bulunur:

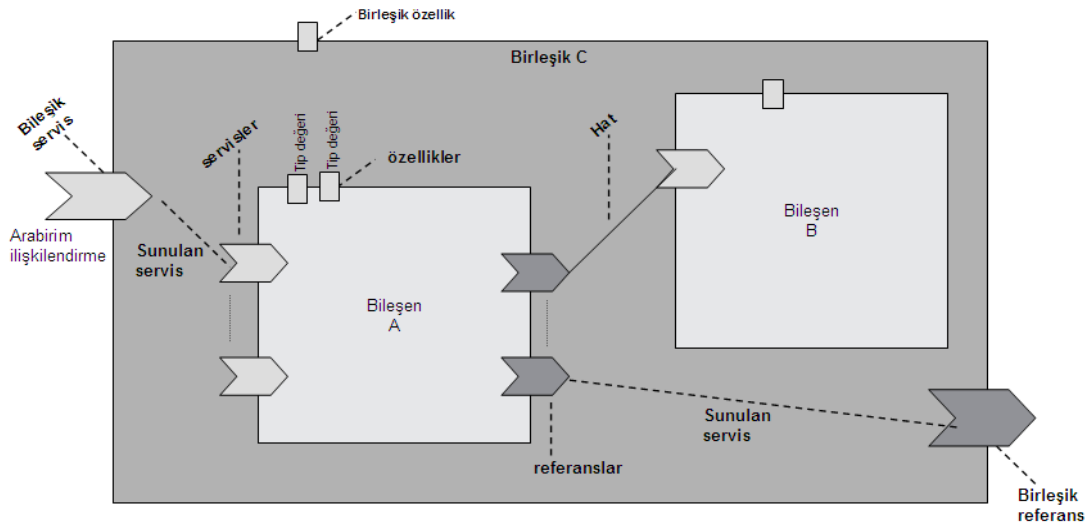
- **Servis:** SCA bileşenine ya da bir birleşime giriş noktasını gösterir.
- **Referans:** Belirtilen SCA bileşeninin kapsamı dışında başka bir şey tarafından sağlanan bir servise bir işaretçiyi gösterir.
- **Arabirim ilişkilendirme:** İki şeyi gösterir: bir arabirim ile bir ilişkilendirme. Arabirim, bir Java arabirimi, WSDL bağlantı noktası tipi, bir BPEL ortağı bağlantısı, bir C++ sınıfı, vb. tarafından gösterilen servisin dış bildirimidir. İlişkilendirme, erişim yöntemini tanımlar (SOAP/HTTP, JMS, RMI/IIOP, SCA, vb.). Arabirim ilişkilendirme bir servise ya da bir referansa eklenebilir.
- **Özellik:** Bir bileşenin bazı belirli özelliklerini tanımlamak için kullanılabilecek bir {tip, değer} çiftini gösterir.
- **Uygulama tipi:** İş mantığını kodlamak için kullanılan belirli SCA bileşeni için uygulama kodunun tipini tanımlar. Örnek uygulama tipleri durum makinelerini, insani görevleri, Java kodunu vb. içerir.
- **Hat:** İki bileşenin birbirine bağlanma mekanizmasını gösterir. Genellikle bir bileşenin referansı başka bir bileşen tarafından açıklanan bir servise bağlanır.

SCA Bileşeni bir ya da birden fazla servisi açıklayabilir; bir ya da birden fazla referans kullanabilir; bazı belirli özellikleri tanımlayan bir ya da birden fazla özelliğe sahip olabilir ve SCA hattını kullanan bir ya da birden çok bileşene bağlanabilir. Şekil 3.1 SCA'nın yapı taşlarını göstermektedir.



Şekil 3.1 SCA'nın yapı taşları [6]

SCA'nın yapı taşları, birleşik uygulamalar oluşturmak için birleştirilebilir ve bir araya toplanabilir. Birleşik uygulama, birlikte büyük öğe boyutlu bir işletme işlevi sağlayan ilgili SCA bileşenlerinin bir birleşimi olarak düşünülebilir. Birleşik uygulama, yalnızca birleşik uygulamanın iç uygulanması için gerekli bazı bileşenleri içerebilir; arabirimleri bağlamdaki birleşik uygulamanın kapsamı dışında gerekmeyebilir. Birleşik uygulama, bir ya da birden çok birleşik servisi ya da birleşik referansı açıklayabilir ve birden çok birleşik özelliğe sahip olabilir. Şekil 3.2 böyle bir örneği göstermektedir.



Şekil 3.2 Birleşik uygulama örneği [6]

Şekil 3.2'de B bileşeni, birleşik uygulamanın bir bölümü olarak açıklanan hiçbir servis ya da referansa sahip değildir. Bu şekilde, yalnızca birleşik uygulamanın iç uygulaması için kullanılan bileşenler olabileceği ve bu bileşenlerin servisleri ve/veya referansları açıklamasına gerek olmadığı gösterilmektedir.

Yukarıda açıklandığı gibi ilişkilendirme, bir servisin erişim mekanizmasını tanımlar. SCA, ilişkilendirme için bildirimsel ve açık bir belirtim sağlar. Bu belirtim, yalnızca şu anda desteklenen SCA ilişkilendirmelerinin (WSDL, JMS, JCA ve EJB) bildirimsel şekilde gösterilebileceği değil, yeni ilişkilendirmelerin de oluşturulabileceği anlamına gelir.

SCA içindeki ilkeler tek bir bildirimsel şekilde oluşturulabilir. İlkelere örnek olarak, hattaki tüm trafiğin dijital olarak imzalanması, bir servis bileşeninin Servis Kalitesi (QoS) gereksinimlerinin belgelenmesi vb. verilebilir.

3.2. Servis Verileri Nesneleri

"Servis Verileri Nesneleri (SDO) veri kaynağı tipleri arasında veri programlamayı birleştiren bir programlama modeli için bir belirtimdir; ortak uygulama modelleri için güçlü bir destek sağlar; uygulamaların, araçların ve çerçevelerin verileri daha kolay sorgulamasını, görüntülemesini, ilişkilendirmesini, güncellemesini ve iç gözlemlemesini sağlar" [6].

Böylece SDO, türdeş olmayan veri kaynaklarına veri erişimi için birleşik bir teknik üzerinde standartlaştırmaya yardımcı olur; uygulama kodunun veri erişimi mantığından bağlaşıklığını kaldırma mekanizmasını basitleştirir. Birleşik veri erişimi geliştiriciyi birden çok veri kaynağına erişim mekanizmalarını öğrenme zorunluluğundan kurtarır ve dolayısıyla SOA programlamanın karmaşıklığını azaltır. Bu kolaylık dolaylı olarak, birleşik veri bağımlılığına sahip uygulamaların geliştirme süresinin azaltılmasında olumlu bir etkiye sahiptir.

Tipik Web tabanlı uygulamalarda, bir işlemde bazı verilerin alınması ve ardından yeni işlemde güncellenmiş verilerin süreklilik için gönderilmesi genel bir gereksinimdir. SDO, verilerle etkileşmek için etkili bir koşut zamanlı mekanizma sağlamak üzere iyimser koşut zamanlı semantik veri erişimi modeli kullandığında bağlantısı kesilmiş veri kullanımı modelini uygular; dolayısıyla Web uygulama kullanımının tipik iş düzeyi semantiğinin çok benzer bir simülasyonunu olanaklı kılar.

Veri erişimi ve işleme için araçların ve çerçevelerin geliştirilmesi çalışmasını uygulayıcıların ellerine bırakmak yerine, bu belirtim, türdeş olmayan kaynaklar üzerinde verilere tek tip erişim için araçlara ve çerçevelere yapısında olan bir destek sunar. Özellikle, veri tipleri, ilişkiler ve kısıtlamaların da içinde bulunduğu verilere iç gözleme olanak tanıyan bir meta veri API çerçevesi sağlar. Bu çerçeve meta veriler etrafında genel araçların geliştirilmesini olanaklı kılar [6].

4. SERVİS ODAKLI MİMARİ VE XML VERİ MODELİ

4.1. XML Verilerinin Önemi ve Değeri

XML sektör standartlarının daha da yaygınlaşmasıyla, SOA ortamlarına doğru olan yönelimle ve dağıtım teknolojilerinin hızlı şekilde benimsenmesiyle, her gün daha çok miktarda XML verisi web haber kaynağı, satınalma emirleri, işlem kayıtları SOA ortamlarındaki iletiler, finansal işlemler, sigorta uygulamaları ve diğer sektöre özgü ve sektörler arası veriler olarak oluşturulmaktadır. Diğer bir deyişle, XML verileri ve belgeleri değerli bilgiler içeren önemli bir işletme varlığına dönüşmüştür (müşteri bilgileri, işlem verileri, sipariş kayıtları ve operasyonel belgeler gibi) [13].

XML varlıklarının büyümesi ve yaygınlaşması zorlukları ve fırsatları bir arada sunmaktadır. XML verileri kullanıldığında ve içerdiği bilgilerin değeri açığa çıkarıldığında, işletmeler için operasyonlarını kolaylaştırmada, öngörü kazanmada ve çevikleşmelerinde fırsatlara dönüşebilecektir.

Öte yandan, XML verileri bir işletmenin operasyonların daha kritik bir öneme sahip oldukça, XML verilerinin güvenliği, bakımı, aranması ve paylaşılması noktasında zorluklar ortaya çıkmaktadır. Kullanımına bağlı olarak XML verilerinin güncellenmesi, denetlenmesi ve geleneksel verilerle tümleştirilmesi gerekir. Bu görevlerin tümü, geleneksel veri varlıklarına sağlanan güvenirlik, kullanılabilirlik ve ölçeklenebilirlik özellikleriyle gerçekleştirilmelidir.

XML verilerinin potansiyelini ortaya çıkarmak için, işletme sınıfı ilişkisel veritabanı yönetim sistemlerinin depolama ve yönetim servislerine benzer servislere gerek vardır.

4.2. XML Verilerinin Yönetilmesinde Geleneksel Yöntemler

"XML verileri için aşağıdaki ortak yaklaşımlar bulunur:

- XML belgelerinin dosya sistemlerinde saklanması
- XML verilerinin ilişkisel veritabanlarında büyük nesnelere doldurulması
- XML verilerinin birden çok ilişkisel sütun ve tabloda parçalarına ayrılması verilerin salt XML veritabanı sistemlerinde yalıtılması" [13].

XML verilerini yönetmeye ve paylaşmaya yönelik bu seçenekler çoğunlukla performans gereksinimlerini karşılayamamaktadır. Dosya sistemleri basit görevler için iyidir; ancak yüzlerce ya da binlerce belge olduğunda ölçekleme yapamazlar. Koşut zamanlılık, kurtarma, güvenlik ve kullanılabilirlik konuları yönetilemez bir hal alabilmektedir. Veritabanı yönetim sistemlerinin (DBMS) kullanılması dosya sistemlerinin bu kısıtlamalarının bazılarını hafifletmektedir; ancak, DBMS çözümü de XML verilerinin yönetilmesinde kendi sınırlamalarına sahiptir. İlişkisel veritabanı yönetim sistemlerini kullanmaya yönelik iki genel yaklaşıma doldurma ve parçalarına ayırma denir.

- **Doldurma:** XML belgesinin bir bütün halinde ilişkisel veritabanı içindeki tek bir VARCHAR ya da büyük nesne (CLOB ya da BLOB) sütununda saklanmasını içerir. Bu yaklaşım, yapılması gereken tek şey XML belgelerini bir bütün halinde saklamak ve ardından çağırmak olduğu sürece işe yarar. Ancak, XML belgelerinin içeriği üzerinde sorgu yürütülmesi; parçaları ya da belirli öğeleri/öznitelikleri/alt ağaçları alınması gerektiğinde, performans ek yükü nedeniyle son derece kullanışsız olan her belgenin çalıştırma süresinde taranmasını içerir.

- **Parçalarına ayırma:** XML verilerinin ilişkili sütunlarına/sütunlarından ve tablolarına/tablolarından eşleştirilmesini içerir. XML belgesini veritabanında saklamak için, belge ayrı sütunlarda saklanan çeşitli parçalarına (öğeler ve öznitelikler) ayrılır; bu işlem bir miktar ek yük getirir. XML verilerinin

karmaşıklığı ve normalleştirme kuralları XML belgelerinin onlarca tabloda yüzlerce sütuna yayılmasına neden olabilir. Benzer şekilde, belgeyi yeniden oluşturmak için bu sütun ve tabloların tümüne karmaşık sorgular ve gereksiz karmaşıklık yaratan çok tablolu birleştirmeler kullanılarak erişilmesi gerekir. Bazı belgelerin yeniden oluşturulması ya da dijital imzalarda olduğu gibi verilerin özgün doğruluğunun korunması da imkansız olabilir. Ayrıca, ilişkisel veri modellerinin esnemezlik özelliğinin XML veri biçimlerinin esnek yapısına karışmasına da yol açar.

XML veri yapılarının ve XML verilerinin daha etkili yönetimine olanak tanıyan, XML verileri için bir dizi özel DBMS tanıtılmıştır. Bu tezin örnek senaryosunda kullanılacak DBMS bu tip bir yapı da olacaktır.

4.3. Veri Gösterimi Olarak XML

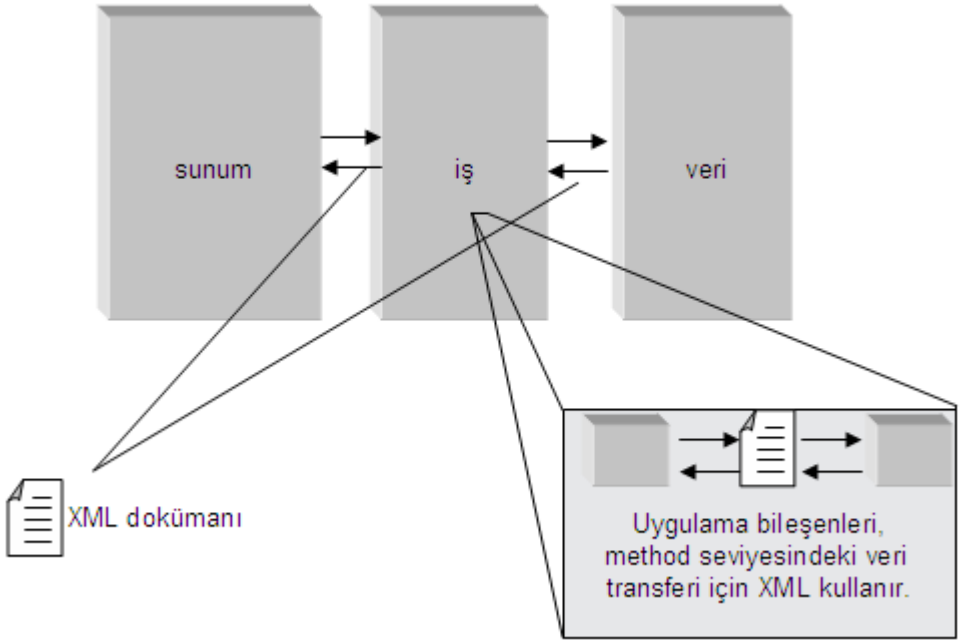
Veri gösterimi teknolojisi olarak XML, çağdaş tümleştirme çözümlerinin temel bir bölümüdür. Dolayısıyla, tümleşik mimarinin hem operasyonel hem de kavramsal yönünde çok önemli bir role sahiptir. Günümüzdeki tümleştirme platformları, XML belirtimlerinin çekirdek ailesini, bu platformların çoğu üçüncü taraf ürünlerinin yanı sıra özel çözümlerin tamamlayıcı bir parçası olacak şekilde kapsamaktadır.

4.3.1.XML veri gösteriminin mimari de konumlandırılması

XML veri gösterimi özelliklerini düzgün şekilde tümleştirmek için en önemli adım, hem uygulama hem de tümleştirme mimarilerinde nasıl faydalı olacaklarını anlamaktır. Böylece XML belgelerin uygulama ortamları katmanları içinde stratejik olarak konumlandırılabilir. XML' in yapıya bu düzeyde nasıl dahil edildiği, yalnızca bileşenler arası ve katmanlar arası iletişimi etkilemez, aynı zamanda gelecekteki uygulamalar arası veri alışverişinin temelini de oluşturur [9].

4.3.2. Uygulamalar içinde veri aktarma biçimi olarak XML

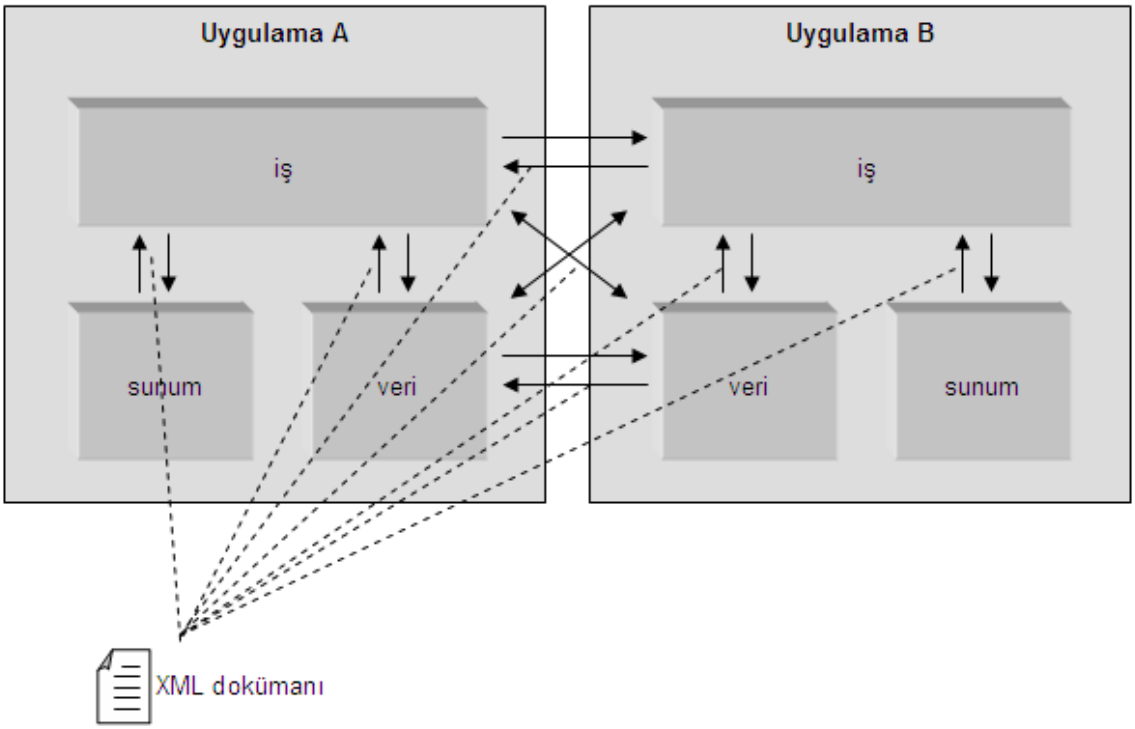
Çok katmanlı mimarinin çeşitli katmanları içinde ve arasında akan veriler genellikle XML belgeleri tarafından barındırılır. Bu nedenle XML, uygulama verilerinin standart aktarma biçimi olarak konumlandırılmıştır ve uygulama mimarisinin temel bölümlerinden biri olmuştur.



Şekil 4.1 N katmanlı uygulama

4.3.3.Uygulamalar arasında veri aktarma biçimi olarak XML

Uygulamaları tümleştirirken XML belgeleri, verileri göstermek ve genel uygulama arabirimlerine/arabirimlerinden aktarmak için kullanılır.

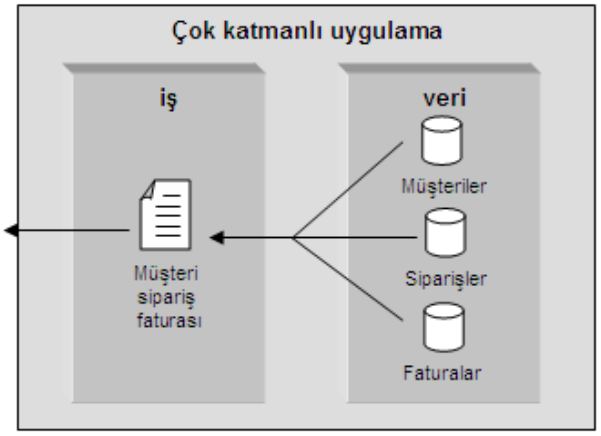


Şekil 4.2 XML belgelerinin, tümleşik ortamlarda standart veri aktarma yöntemi olarak kullanımı

Web servisleri özellikle tümleştirme mimarisi farklı platformları birleştirdiğinde bu uygulama uç noktalarını göstermek için kullanılır. Web servisleri çerçevesi içinde, XML belgeleri genellikle SOAP iletileri kullanılarak aktarılır.

4.3.4.Uygulamalar içinde veri köprüsü olarak XML

XML veri gösteriminin kullanımı, analistlere farklı veriler ile iş modelleri arasındaki boşluğu kapatma olanağı sunmuştur. Bu durum, özellikle verileri farklı biçimlerde ve yapılarda saklayan bilgi havuzlarından verileri gösterirken bir dizi zorluk getirecektir.



Şekil 4.3 Çok katmanlı bir uygulama da üç farklı veri kaynağından alınan verileri yakalayan bir XML belgesi

4.3.5.XML veri gösterimini tümleştirme stratejileri

"Bir XML belgesi modellediğinde, aşağıdaki etmenleri projeye katılması gereken stratejik kararların alınmasını gerekebilir:

- Performans
- Güvenlik
- Genişletilebilirlik
- Yeniden kullanılabilirlik
- Veri erişimi" [9].

Bu konular, yalnızca XML'in verileri nasıl gösterdiği ile ilgili değil, aynı zamanda kullanılması planlanan tamamlayıcı XML teknolojileri ile de ilgilidir.
Örneğin, tanımlanan XML belgesi yapısı şunları belirleyecektir:

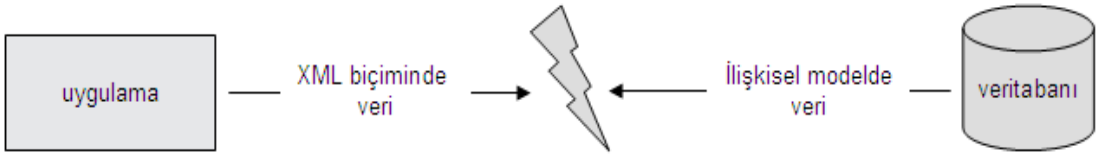
- Bu belgenin örneklerini iletmek için bant genişliği gereksinimleri
- Programlama API'si kullanarak bu belgeyi ayrıştırmak için işleme gereksinimleri
- Belgenin ilgili şema tanımının yapısı
- İlgili şemaya göre bu belgeyi doğrulamak için işleme gereksinimleri
- İlgili dönüştürme stil sayfalarının tasarımı
- İlgili dönüştürme stil sayfaları için işleme gereksinimleri

- Belge tarafından barındırılan verilerin yapısıyla ilgili güvenlik gereksinimleri
- Zaman içinde değişen veri modeli olarak genişletilecek belge yapısının özelliği
- Uygulama içinde paylaşılacak belgenin (ya da belgenin bölümlerinin) özelliği
- Uygulama bileşenleri ve son kullanıcılar tarafından kullanılan veri gösteriminin kalitesi

Bu etmenler projeye katılırsa, sağlam bir XML temeli tasarlamada doğru olunduğu düşünülür [9].

4.4. XML ve İlişkisel Veritabanlarının Karşılaştırılması

Her türlü veri tümleştirme stratejisini oluşturmadan önce, ilişkisel veritabanları arasındaki temel farklılıkların, XML teknolojisi kümesi ve bunların verilerinizle göreli ilişkilerinin öncelikle anlaşılması önemlidir.



Şekil 4.4 Geleneksel uygulamaların bilgi akışı

XML belgeleri ve ilişkisel veritabanları verileri çok farklı şekillerde gösterir ve yapılandırır. Bu durum iki ortam arasına görünmez sınırlar çizer ve bu veri platformlarının etkili şekilde işbirliğinde bulunması, iki farklı kültür arasında anlaşma müzakereleri yürütmek kadar zor olabilir.

4.4.1.Veri depolama ve güvenlik

Veri yönetimi sistemlerinin en temel özelliği bilgileri güvenli şekilde saklayabilme becerileridir. Burası, ilişkisel veritabanlarının XML' in basit dosya biçimiyle kıyaslanamayacak özellikler sunduğu noktadır.

Tablo 4.1 İlişkisel veritabanı ve XML güvenlik karşılaştırma [9]

	İlişkisel Veritabanı	**XML**
Fiziksel Depolama	Denetimli depolama ortamı	Düz metin
Güvenlik	• Uygulamaya dayalı güvenlik ya da işletim sistemiyle tümleştirilmiş bir güvenlik sistemi sağlar. • Veriler ve yapılarına ilişkin çoğu açıdan öğe boyutlu denetim sağlar.	• Yerleşik güvenlik yoktur. • Erişim denetimi dosya ya da klasör düzeyinde ayarlanır ya da uygulama tarafından yönetilir.

4.4.2.Veri gösterimi

XML belgeleri ve ilişkisel veritabanları veri gösterimi konusuna farklı yönlerden yaklaşmaktadır. XML belgeleri birleşik ve yapılandırılmış bir sıradüzeni getirirken, DBMS'ler daha esnek bir ilişkisel model sunmaktadır.

Bu iki platform, XML belgesi sıra düzenlerinin ilişkisel veritabanlarında yeniden oluşturulması ve ilişkisel veri modellerinin XML belgelerinde gösterimi zor olduğu için çok büyük tümleştirme güçlükleriyle karşı karşıyadır.

Tablo 4.2 İlişkisel veritabanı ve XML veri gösterimi karşılaştırma [9]

	İlişkisel Veritabanı	**XML**
Veri Modeli	Satır ve sütunların olduğu, tablo halinde veri varlıklarından (tablolar) oluşan ilişkisel veri modeli	Öğe ve öznitelik düğümlerinin olduğu belge yapılarından oluşan sıra düzensel veri modeli.
Veri tipleri	İkili veriler için destek de dahil olmak üzere sayı, karakter, tarih gibi	XSD şemaları, karşılaştırılabilir veri

	çok çeşitli veri tipleri sunulmaktadır.	tipleri kümesi ile donatılmıştır. İhtiyaca göre özel veri tipi tasarlanabilir
Veri öğesi ilişkileri	Sütun tanımları, DDL kurallarına göre tablolar içinde ve arasında birbiriyle ilişki kurabilir.	Referanslar, öğeler arasında açıkça ya da yapısal olarak tanımlanabilir.

4.4.3.Veri Bütünlüğü ve Doğrulama

Tablo 4.3 İlişkisel veritabanı ve XML veri bütünlüğü ve doğrulama karşılaştırma tablosu [9]

	İlişkisel Veritabanı	**XML**
Şema	İlişkisel şemaların gevşek yapısı, veri varlıklarının nasıl var olacağı ve birbiriyle ilişki kuracağına ilişkin büyük bir esneklik sağlar.	• XML şemaları, bir öğe sıradüzeniyle sınırlı oldukları için daha katı bir yapıdadır. • XML Şema dili gibi daha karmaşık şema teknolojileri, öğelerin ayrılmasını gerçekleştiren işlevler sağlar.
Referans bütünlüğü	İlişki kısıtlamalarının uygulandığından emin olmak için kapsamlı destek verilir. Genel özellikler, basamaklı silme ve güncelleme işlemleriyle değişikliklerin ilgili sütunlara yayılması becerisini içerir.	• Referanslar öğeler arası ilişkilere belirli bir düzeye kadar benzese de, RDBMS benzeri referans bütünlüğünde kıyaslanabilir bir zorlama yoktur.

		• Ayrıca, ilişkisel veritabanları referans bütünlüğünü verilerin değiştirilme anında zorlarken, XML ayrıştırıcısı tarafından ayrı bir doğrulama adımına gerek duyulur.
Tamamlayıcı doğrulama	Doğrulama, tetikleyiciler ve kayıtlı yordamların kullanımıyla tamamlanabilir.	Öğeleri ve öznitelikleri özel kurallara göre doğrulamak için XSD şemaları tasarlanabilir. XSLT ve Schematron gibi ek teknolojiler kullanılarak doğrulama düzeyi daha da iyileştirilebilir.

4.4.4.Veri sorgulama ve dizinleme

Tablo 4.4 İlişkisel veritabanı ve XML veri sorgulama ve dizinleme karşılaştırma [9]

	İlişkisel Veritabanı	**XML**
Sorgu Dilleri	Çoğu ticari RDBMS, sektör standardı SQL'i destekler. Çoğu, mülkiyete dayalı uzantılar ekler.	• Tekil XML belgelerinde sorgu çoğunlukla DOM ve SAX API'leri üzerinden ya da XPath ifadeleri kullanılarak yürütülür. • XML teknolojisi, belgeler arası aramaları

		da destekleyen kapsamlı bir sözdizimi sağlayan XQuery olanağı SQL ile çok benzerdir.
Sorgu Sonucu İşleme	SQL, sorgu sonuçlarını gruplayan, sıralayan ve özelleştiren bazı çıktı parametreleri sağlar.	XML biçimlendirilmiş verilerinin çıktısını işlemek için XSLT ve XQuery kullanılabilir. Her iki dil de gruplayabilir, sıralayabilir ve karmaşık veri işleme gerçekleştirebilir.
Birden çok havuzda sorgu yürütme	Havuzların her biri sorguyu yürütmek için kullanılan protokolü desteklediği takdirde, çeşitli veritabanı platformları çoklu veri kaynağı sorgularına olanak tanır.	• XPath birden çok XML belgesinde sorgu yürütmezken, XQuery yürütebilir. • XSLT birden çok belgeyi de sorgulayabilir (belge işlevi kullanarak).
Dizinleme	• Karmaşık dizinler desteklenir ve sütun düzeyinde özelleştirilebilir. • RDBMS dizinleri üzerinde, iyileştirilmiş sorgu yürütme için ince ayar yapılabilir.	• XML teknolojisi kümesi karşılaştırılabilir dizinleme uzantısını desteklemez. • XML belge dizinleri genellikle özel uygulamalar tarafından oluşturulur ve sürdürülür.

		• En modern RDBMS'ler, öznitelik ya da düğüm düzeyinde XML belgelerinin dizinlenmesini de destekler.

4.4.5.Çeşitli ek özelliklerin karşılaştırılması

Tablo 4.5 İlişkisel veritabanı ve XML'in ek özelliklerini karşılaştırma [9]

	İlişkisel Veritabanı	**XML**
İşlemler	• Çoğu veritabanı işlem ve geri alma desteği sunar ve çoğu genel ACID3 özelliklerini destekler. • Bazı RDBMS'ler işlem desteğini birden çok veritabanına genişleten iki aşamalı kesinleştirme özellikleriyle birlikte gönderilir.	XML platformu henüz sektör standardı bir işlem teknolojisi sunmamaktadır (WS-Coordination ve WS-Transaction kullanımıyla ACID özellikleri için destek sunulsa da).
Çok kullanıcılı erişim kaydı kilitleme	Ardışık kullanım sırasında verilerin bütünlüğünü korumak için çoğu veritabanı, veriler güncellenirken verilere erişimi denetleyen araçlar sağlamaktadır. RDBMS'ler genellikle farklı kilitleme modellerinin yanı sıra sayfa düzeyinde ya da satır düzeyinde kayıt kilitlemeyi destekler (örneğin, karamsar, iyimser).	• Fiziksel dosyalar olarak bulunan XML belgelerine erişim, XML ayrıştırıcısı üzerinden uygulama tarafından dosya G/Ç denetimlidir. • Karşılaştırılabilir kilitleme özellikleri yoktur.

Platform bağımlılığı	• İlişkisel veritabanları, sağlayıcıya bağımlı bir depolama platformunu zorunlu kılan ticari ürünlerdir. • Ancak verilerin, farklı sağlayıcıların veritabanları arasında kolayca geçişi yapılabilir.	XML belirtimleri ailesi açık sektör standartlarıdır ve hiçbir ticari platforma bağlı değildir.
Şema bağımlılığı	Şemalar bir veritabanının zorunlu bir bölümüdür. Şema özellikleri veritabanı yazılımı tarafından sağlanır.	• Şemalar XML belgelerinin isteğe bağlı bölümleridir. • Kullanıldığında mevcut çeşitli şema teknolojilerinden biri seçilebilir.
Şema yeniden kullanımı	Şema, veritabanı örneğine bağlıdır. Şemalar DDL ile sarılabilir, ancak modelleme araçları olmadan kolayca yeniden kullanılamaz.	• Şema bağımsız bir varlıktır. • Birden çok belge aynı şemayı kullanabilir.
İç içe yerleştirme	Tablo sütunları genellikle yapısal iç içe yerleştirme sağlamaz.	XML öğeleri iç içe yerleştirilmiş alt öğeler içerebilir.

5. ÖRNEK OLAY : MÜŞTERİ TAKİP SİSTEMİ

Örnek olayın amacı, servis odaklı mimarinin prototipini yaratarak, olması gereken örnek mimari yapıyı ve uygulamasını göstermektir. Bunun yanısıra veri modelinin XML olarak tasarlanmasının entegrasyon ve önyüz katmanlarının hayata geçirilmesine olan katkısı da incelenecektir. Örnekte kullanılacak araçlar servis odaklı mimari konusunu uygulamaya geçirmek için ürünleri IBM firmasına ait olacaktır. Bu araçların senaryodaki kullanım amaçları aşağıda listelenmiştir:

- **IBM DB2 Kurumsal Veri Tabanı Sunucusu:** IBM tarafından geliştirilen ilişkisel ve XML veritabanı sunucusu olan DB2 ' da örnek olaya ait veri modeli barındırılacaktır. Burada ileride detayı anlatılacak olan hibrid veri modeline ilişkin tanımlar ve XML veri tiplerine ait XML şemaları tutuluyor olacak. XML verilere ilişkin şema doğrulaması bu sunucuda ele alınıyor olacak.
- **IBM WebSphere Kurumsal Servis Yolu (WebSphere Enterprise Service Bus):** Örnek olaya ait uygulamada süreç ve önyüz katmanına ait veri iletim ve dönüştürme ihtiyacı kurumsal servis yolu üzerinde giderilecek böylece verinin iletimi konusunun uygulama içerisinden çıkarılıp ayrı bir katmanda çalıştırılması sağlanarak veri iletiminde oluşacak değişikliklerde uygulamanın değiştirilmesine ihtiyaç olmaması sağlanacaktır.
- **IBM WebSphere Süreç Sunucusu (IBM WebSphere Process Server):** Örnek olayda kullanılacak sürecin uygulama içerisinde yazılmasını önleyerek servis odaklı mimarideki süreç katmanının kurgulanacağı araç süreç sunucusu olacak. Böylece süreçlerin geliştirilmesi ve değiştirilmesinden uygulama katmanın asgari düzeyde etkilenmesi sağlanacaktır.
- **IBM WebSphere Portal Sunucusu (WebSphere Portal Server) :** Örnek olayımıza ait uygulamanın sunum katmanını çalıştıracak sunucumuz olacak. Rol tabanlı çalışmayı, kimlik doğrulama ve yetkilendirme katmanını üstleniyor olacak. Bu sunucu üzerinde çalışacak uygulamalarımızın bir arada çalışmasını da bu uygulama sunucusu sağlıyor olacak. Uygulama sunucusu üzerinde geliştireceğimiz uygulamalara literatürde portlet adı veriliyor.

Geliştireceğimiz portletler JSR286 standardında oluyor olacak. Böylece servis odaklı mimari'de son kullanıcının senaryoyla buluştuğu ve süreci çalıştırdığı katman da bu sunucu üzerinden uygulamaya geçiriliyor olacak.

- **IBM Rational Uygulama Geliştirici (Rational Application Developer):** Ön yüze ait uygulamalar Java uygulama geliştirme dili kullanarak JSR286 portlet standardına uygun olarak bu aracı kullanarak geliştirecektir.
- **IBM WebSphere Entegrasyon Geliştirici:** Örnek olayımıza ilişkin veri dönüşüm ve iletim senaryoları bu araç aracılığı ile tasarlanıyor olacak. Böylece veri dönüşümü katmanı uygulama katmanından geliştiriliyor ve uygulamaya alınıyor olacak.

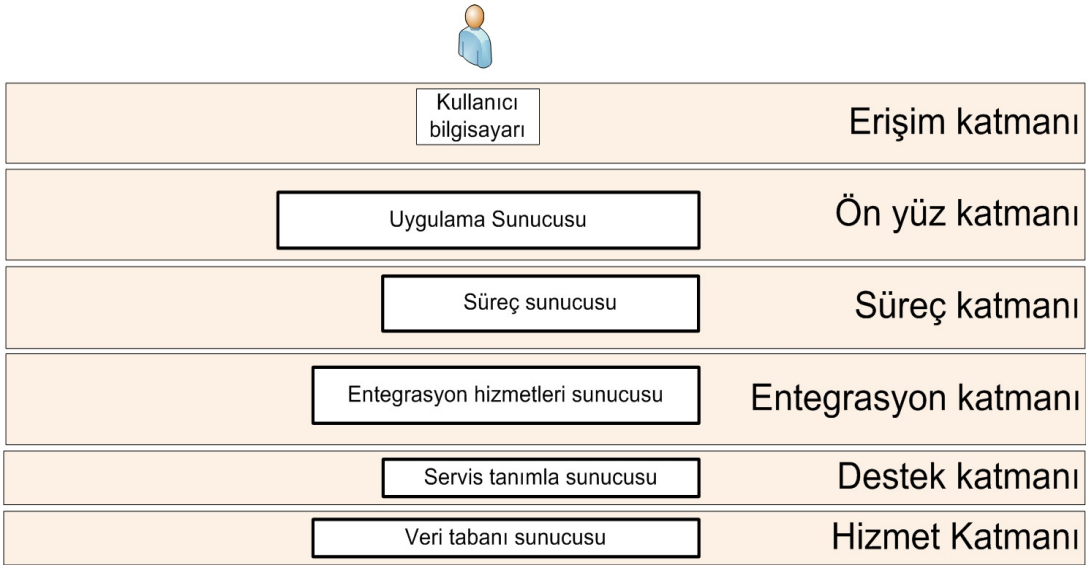
5.1. Teknik Hedefler

Bu örnek olay da servis odaklı mimarinin uygulamaya geçirilişi hakkında bir model hazırlanmıştır. Örnek olay ile aşılmak istenen teknik hedefler :

- Süreç katmanının servis odaklı mimari de anlatıldığı üzere ayrı bir katmanda tasarlanmasının uygulama geliştiricilere sağladığı esnekliği anlatmak.
- Entegrasyon katmanını uygulama geliştirme katmanından ayrı bir katmanda geliştirmek.
- Servisleri yönetilebilir bir sunucudan kullanarak servis odaklı mimari deki servis meta modelini yönetebilme ve tekrar kullanılabilirliliği göstermek.
- Ön yüz katmanında JSR 286 standardında portlet kullanarak, bu katmanda ki uygulama geliştirme sürecini anlatmak.
- Veri modelinde hibrid bir yapı kullanarak XML ve ilişkisel veri modelinin bir arada kullanımını sağlamak.

5.2. Teknik Çözüm

Örnek olayın hangi yapıda işlediğini anlatmak için veri modeli, uygulamaya ait mantıksal mimari, katmanlı mimari, entegrasyon mimarisi ve örnek olay senaryosu kullanılmıştır.



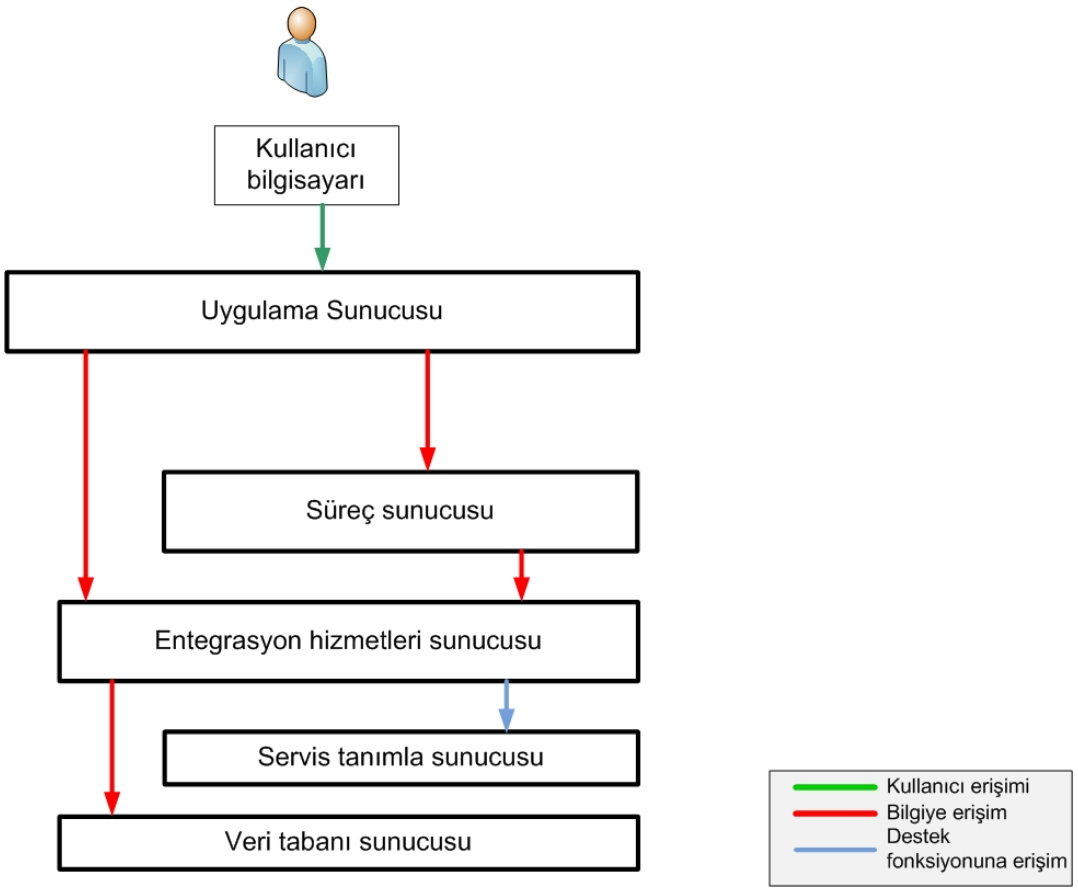
Şekil 5.1 Katmanlı mimari gösterimi

Katmanlı Mimari: Şekil 5.1'de görüldüğü üzere örnek olay altı katmandan oluşmaktadır. Bu katmanların temel hedefi servis odaklı mimarinin amaçlarından olan tekrar kullanılabilirlik ve modülerliği sağlamaktır. Örnek olaya ait tasarlanan mimari katmanlar :

- **Erişim katmanı:** Erişim katmanı, kullanıcının sisteme erişmek için kullandığı katmandır. Bu katmandaki araçlar ve protokoller çeşitlilik gösterebilirler. Bu tezin senaryosunda erişim katmanındaki araç internet tarayıcı, protokol ise HTTP olacaktır.
- **Ön yüz katmanı:** Ön yüz katmanı, erişim katmanından gelen istekleri karşılayarak alt katmanlara bu isteğin iletilmesini sağlayacak ve alt katmanlardan gelen saf veriyi ön yüz bileşenleri ile birleştirerek kullanıcıya görünecek son ekranın oluşturulmasından sorumlu olacaktır.
- **Süreç katmanı:** Süreç katmanı, örnek olaya ait sürecin işletilmesinden sorumlu olacak katmandır. Süreç tasarım aracında tasarlanan süreçler bu katmanda çalışacaktır.
- **Entegrasyon katmanı :** Entegrasyon katmanı, süreç katmanın ve ön yüz katmanının ihtiyaç duyduğu verinin iletimini ve dönüşümünü sağlamakla sorumludur. Örnek olay senaryosunda ilişkisel veri tabanından aldığı XML

bilgiyi web servis cevabına dönüştürme ve ön yüz katmanından gelen XML verinin ilişkisel veri tabanın istediği XML biçime dönüşümünü sağlayacaktır.

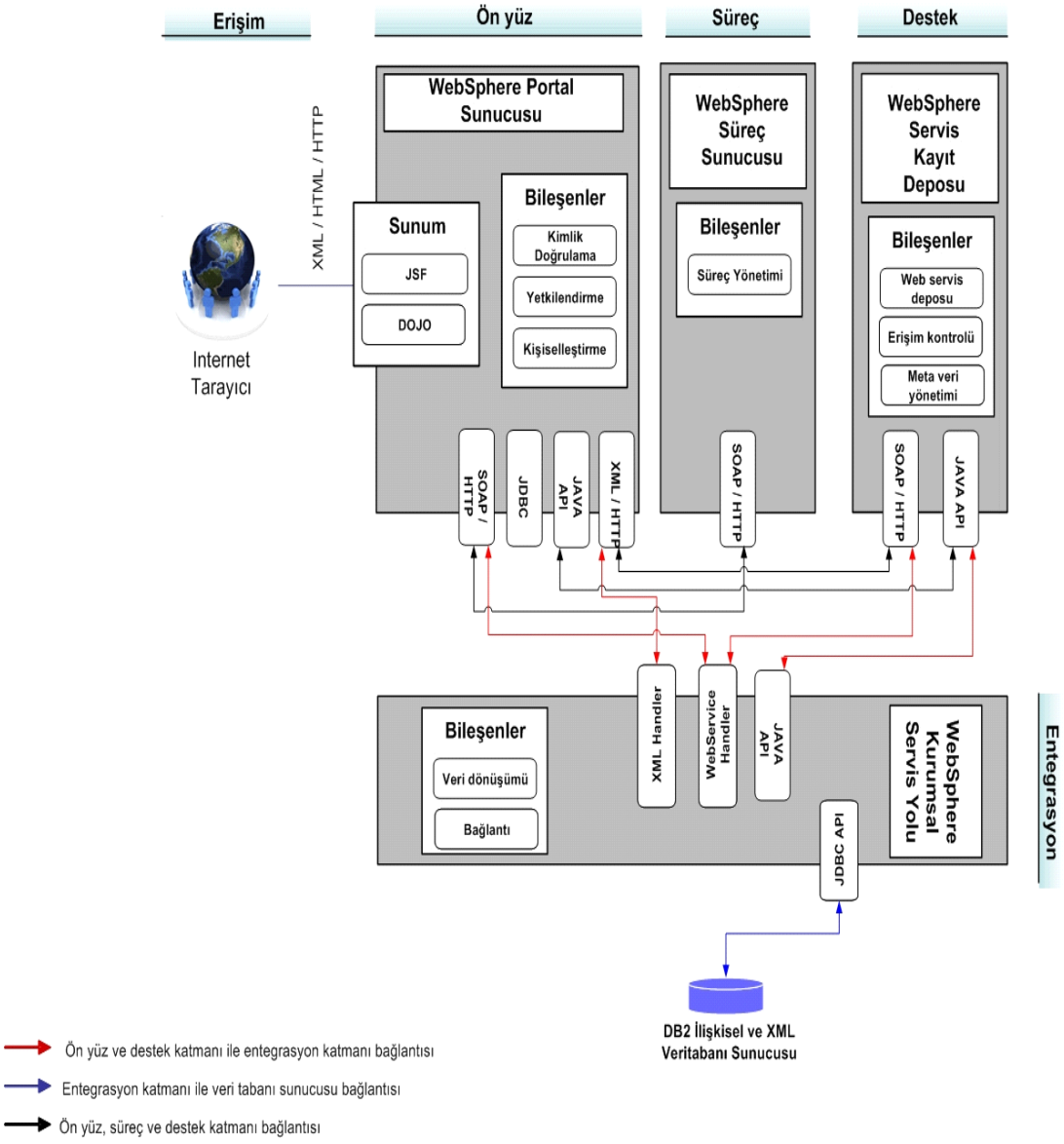
- **Destek katmanı:** Destek katmanı, web servislerin tanımının yapıldığı ve erişildiği katman olacaktır. Böylece web servislerin yönetimi ayrı bir katmanda ele alınarak entegrasyon katmanından veya ön yüz katmanından aynı web servisin ortak kullanımı sağlanabilecektir. Aynı zamanda bu katmanda tutulan web servislerin tanımlarının yapılabilmesi için web servislere ait meta bilgileri de bu katmanda yönetilecektir. Böylece web servislerin yaşam döngüsü süreci servis odaklı mimari ile uyumlu duruma getirilecektir.
- **Hizmet katmanı:** Hizmet katmanı, örnek olaya ait verilerin tutulacağı veri tabanı sunucusundan oluşmaktadır.



Şekil 5.2 Mantıksal mimari gösterimi

Mantıksal mimari: Şekil 5.2'de gösterilen mantıksal mimari, örnek olayın hayata geçirilmesinde kullanılacak sunucu tiplerini ve aralarındaki iletişimi göstermek amacıyla tasarlanmıştır. Örnek olay senaryosunda beş tip sunucu kullanılacaktır. Bu sunuculular ve amaçları:

- **Uygulama sunucusu:** Uygulama sunucusu kullanıcıdan gelen istekleri karşılamak üzere mimariye konulmuştur. Alt yapısında portlet standardı olan JSR 286 taşıyıcısı bulunmaktadır. Bu uygulama sunucusu aynı zamanda Java Enterprise Edition (JEE) standardını destekleyen bir sunucu olduğu için yazılan uygulamalar java uygulama geliştirme dili ile geliştirilmiştir.
- **Süreç sunucusu :** Süreç sunucusu, senaryoya ilişkin sürecin çalıştırıldığı sunucudur. Tasarladığımız örnek olaya ilişkin süreç herhangi bir önyüz çalıştırmıyor olacak. Sadece sürecin yönetimini üstlenecek. Sürece ilişkin kurallar ve koşullar burada çalıştırılıyor olacak.
- **Entegrasyon hizmetleri sunucusu:** Veri dönüşümü ve iletiminden sorumlu olacak bu katmana uygulama sunucusu ya da süreç sunucundan erişilebilir olacak.
- **Servis tanımlama sunucusu :** İletişim için kullanılan web servisler , uygulama içerisinden yönetilmemek için bu sunucuya kurulacak ve bu sunucu üzerinden gerekli web servis erişimi sağlanıyor olacak.
- **Veri tabanı sunucusu:** Veri tabanı sunucusu tasarlanan hibrid veri modelini tutarak sistemin ihtiyaç duyduğu veriyi sağlayacak.



Şekil 5.3 Entegrasyon mimarisi

Entegrasyon mimarisi: Şekil 5.3'de görülen örnek olaya ait entegrasyon mimarisi katmanlar arasındaki iletişimin teknik detaylarını ve her bir katmanda hizmet veren sunuculara ait bileşenleri göstermek amacıyla tasarlanmıştır.

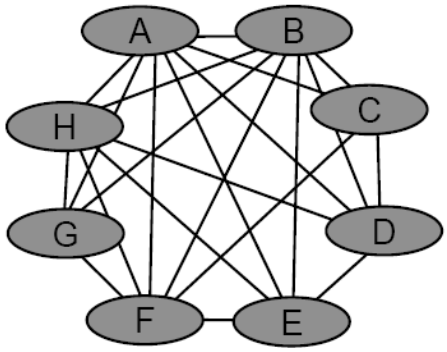
Erişim katmanından ön yüze erişim, HTTP protokolü üzerinden XML ve HTML kullanarak sağlanmaktadır. Ön yüz katmanına gelen istek portal sunucusu üzerinde JSF ve DOJO kullanılarak yazılmış portlet yardımı ile karşılanır. Buradan yapılacak

işin türüne göre istek HTTP üzerinden XML kullanarak SOAP ile süreç katmanına ya da entegrasyon katmanına gidebilir.

Aynı istek, ihtiyaca göre JAVA API, HTTP ya da SOAP kullanarak entegrasyon katmanına da aktarılabilir.

Örnek senaryonun farklı protokol ve haberleşme yöntemlerini içeriyor olmasının temel sebebi uygulamaya ait farklı katmanların, servis odaklı mimari içerisinde kendilerine ait özel alanlarda uzmanlaşmış olduğunu göstermektir. Böylece uygulamanın dış dünya ile iletişimi API ve noktadan noktaya iletişim düzeyinden servis odaklı olarak belirli bir bağlantı katmanı üzerinden gerçekleşmiş olur. Şekil 5.4'de farklı fonksiyonların birbirleriyle doğrudan bağlı olduğu görülmektedir. Bu durumun yarattığı en temel problemler :

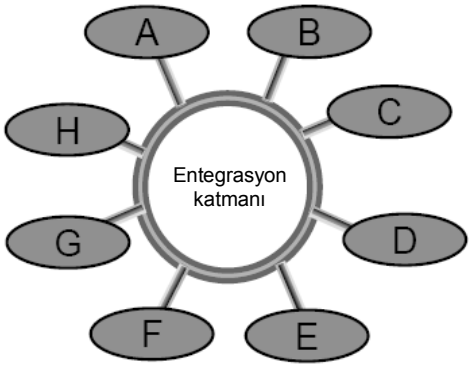
- Her bir bileşen diğer bileşene erişirken özel bir program arayüzü kullanır.
- Komplekslik ve iki bileşen arasında taşınan verinin boyutu üstel olarak artış gösterir.
- Bir bileşeni fiziksel olarak taşımak etkileşimde olduğu her noktanın değiştirilmesini gerektirir.



Şekil 5.4 Fonksiyonların birbirleriyle doğrudan bağlı olma durumu

Şekil 5.5'de fonksiyonların birbirlerine bir entegrasyon katmanı kullanarak bağlanması durumu gösteriliyor. Örnek olaya ilişkin senaryo bu bağlantı yöntemini kullanarak esnekliği göstermeye çalışmıştır. Bu bağlantı yönteminin temel avantajları:

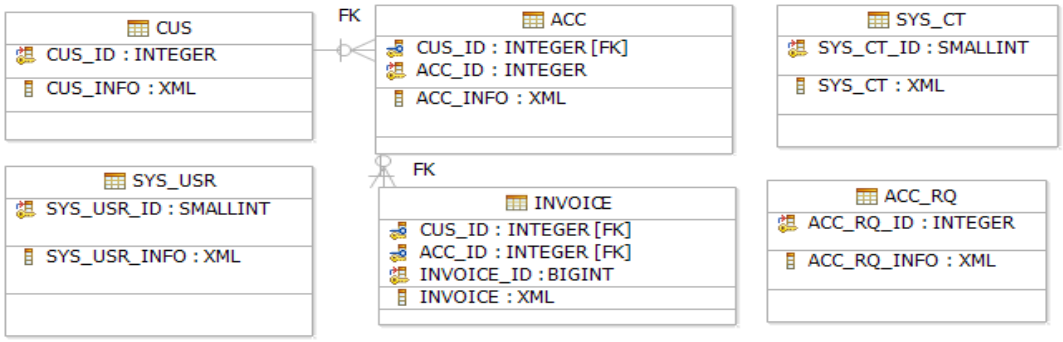
- Her bir bağlantı entegrayon katmanına bir adaptör yardımıyla bağlanır ve bu adaptör veri üzerinde her türlü dönüşümü gerçekleştirir.
- Komplekslik ve taşınan veri ihtiyacı bileşen ihtiyacı ile doğru orantılı artar.
- Bileşenler uygulamalara etki etmeden taşınabilirler.



Şekil 5.5 Fonksiyonların entegrasyon katmanı kullanarak bağlanma durumu

5.3. Veri modeli

Örnek senaryoya ait tasarlanan veri modeli Şekil 5.6 ‘ da görüldüğü üzere altı tablodan oluşmuştur. Veri modeli hibrid yapıda tasarlanmış olup her tabloyu oluşturan anahtar kolonların dışında XML tipinde kolonlar da bulunmaktadır. XML tipindeki kolonların veri doğrulamasının veri tabanı sunucusu üzerinde yapmak üzere XML şemaları hazırlanmıştır. Verinin doğrulanmasını veri tabanı sunucusu üzerinde yapmak uygulama geliştirme katmanına ait sunucu tarafında yapılacak doğrulama işleminin veri tabanı katmanına aktarılmasını da sağlamıştır. Bu durum verinin doğrulunu merkezi bir ortamda garanti altına almayı sağlamış olmakla birlikte aynı zamanda doğrulama kriterlerinde oluşacak bir değişiklikte uygulamayı değiştirme ve değişen uygulamayı sunucu üzerine tekrar koyma operasyonundan da bizi kurtarmıştır.



Şekil 5.6 Hibrid veri modeli

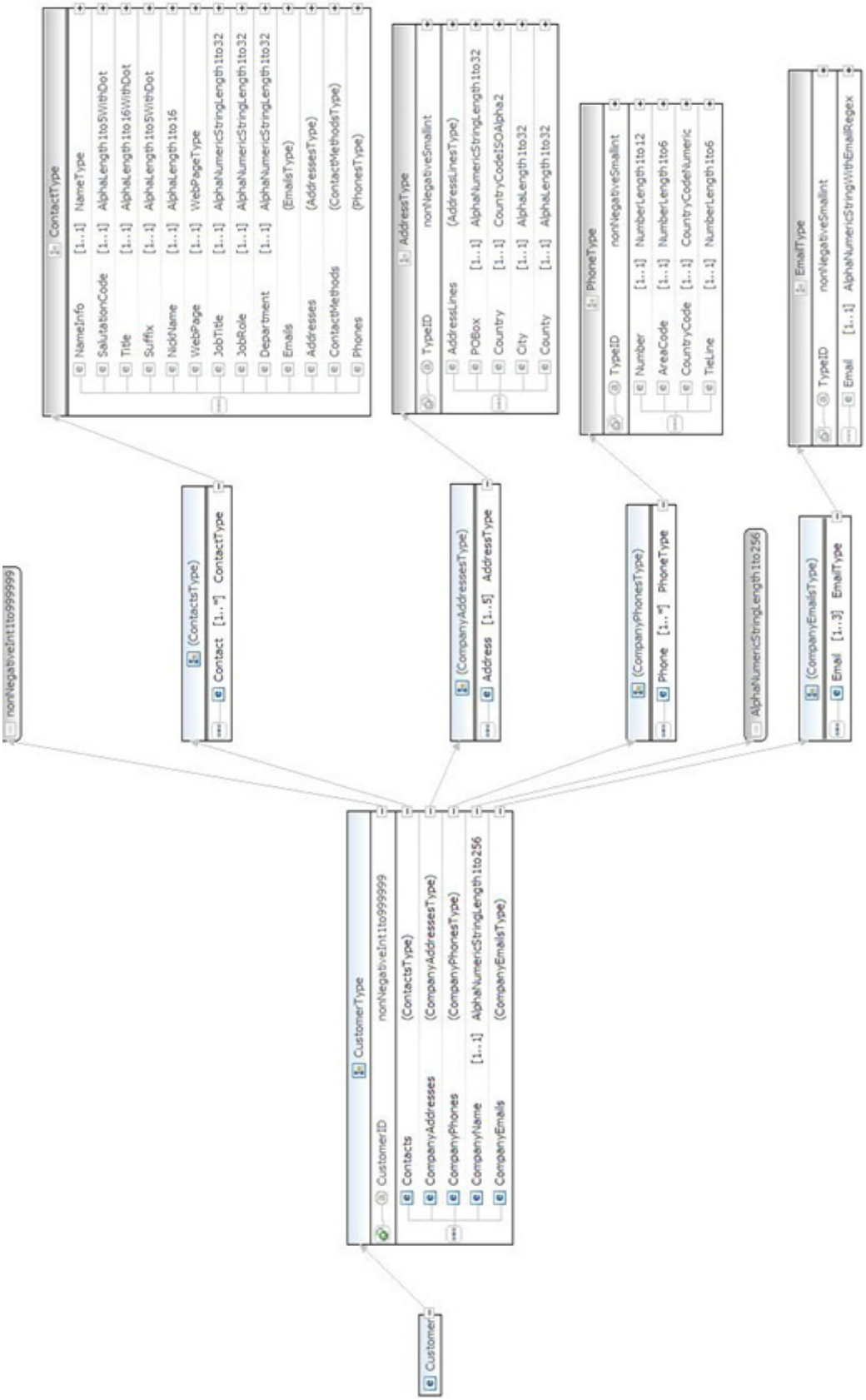
Hibrid veri modeli temelde XML veri tutma yapısının ilişkisel veri tutma yapısı ile birlikte modellenmesidir. Bu modele göre XML tipindeki kolonların içinde bulunan XML modeller barındırdıkları özelliklere göre indekslenebilirler. Bu modelin temel faydaları :

- Veri tabanından alınan verinin XML dönüşümünü uygulama içerisinde yapılmasını ortadan kaldırmak.
- İki uygulama arasında ihtiyaç duyulacak veri aktarımı operasyonun veri tabanı sunucusu seviyesinde yapılabilmesini sağlamak.
- Dijital imza kullanacak uygulamalarda imzalanmış XML verinin değiştirilmeden veri tabanı sunucusunda yönetilmesini sağlamak.
- Kolon yapısı farklılık gösterebilecek satır yapılarının yönetilebilmesi [14].

Örnek senaryo veri modeline ait tablolar ve tablolarda bulunan XML kolonların doğrulamasını yapmak üzere tasarlanmış XML şemalar:

CUS: CUS, müşteri bilgilerini tutmak üzere tasarlanmış bir tablodur. İki kolondan oluşmuştur. Kolonlar ve açıklamaları:

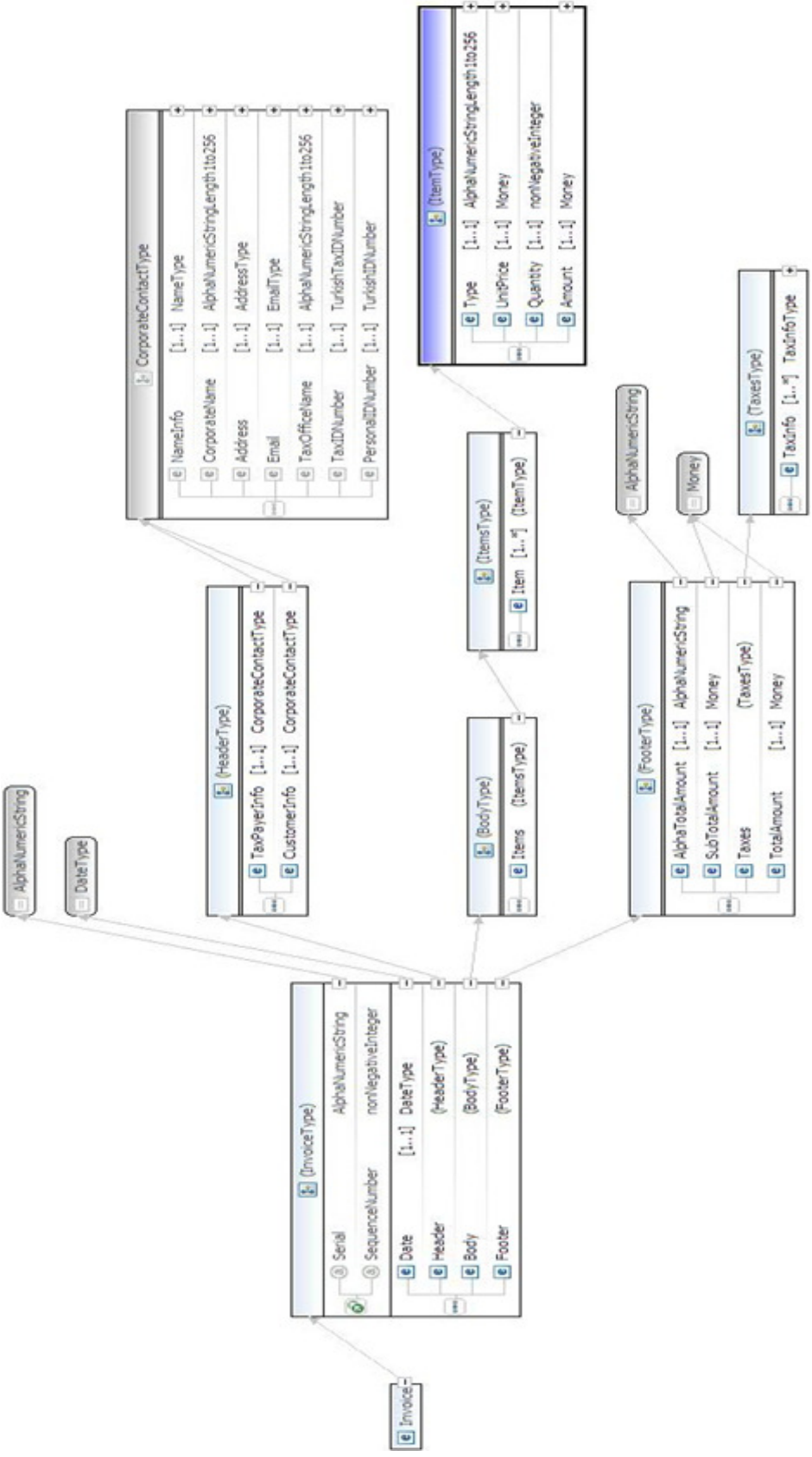
- **CUS_ID:** CUS_ID isimli kolon müşteriye sistem tarafından verilen tekil numarayı tutar. Aynı zamanda tablonun birincil anahtarıdır.
- **CUS_INFO:** CUS_INFO isimli kolon ise müşteriye ait bilgilerin XML olarak tutulmasını sağlayan kolondur. Bu kolona gelecek bilginin doğrulunu sağlamak üzere SYSMNG_CusInfo isimli şema hazırlanmıştır. Şekil 5.7 ‘ de görüldüğü üzere bu şemada müşteriye ait alanlar ve alanlara ait özellikler tanımlanmıştır.



Şekil 5.7 Müşteri verisi doğrulama şeması

ACC: ACC, müşteriye ait hesap bilgilerini tutmak üzere tasarlanmıştır. ACC tablosuna ait kolon detayları:

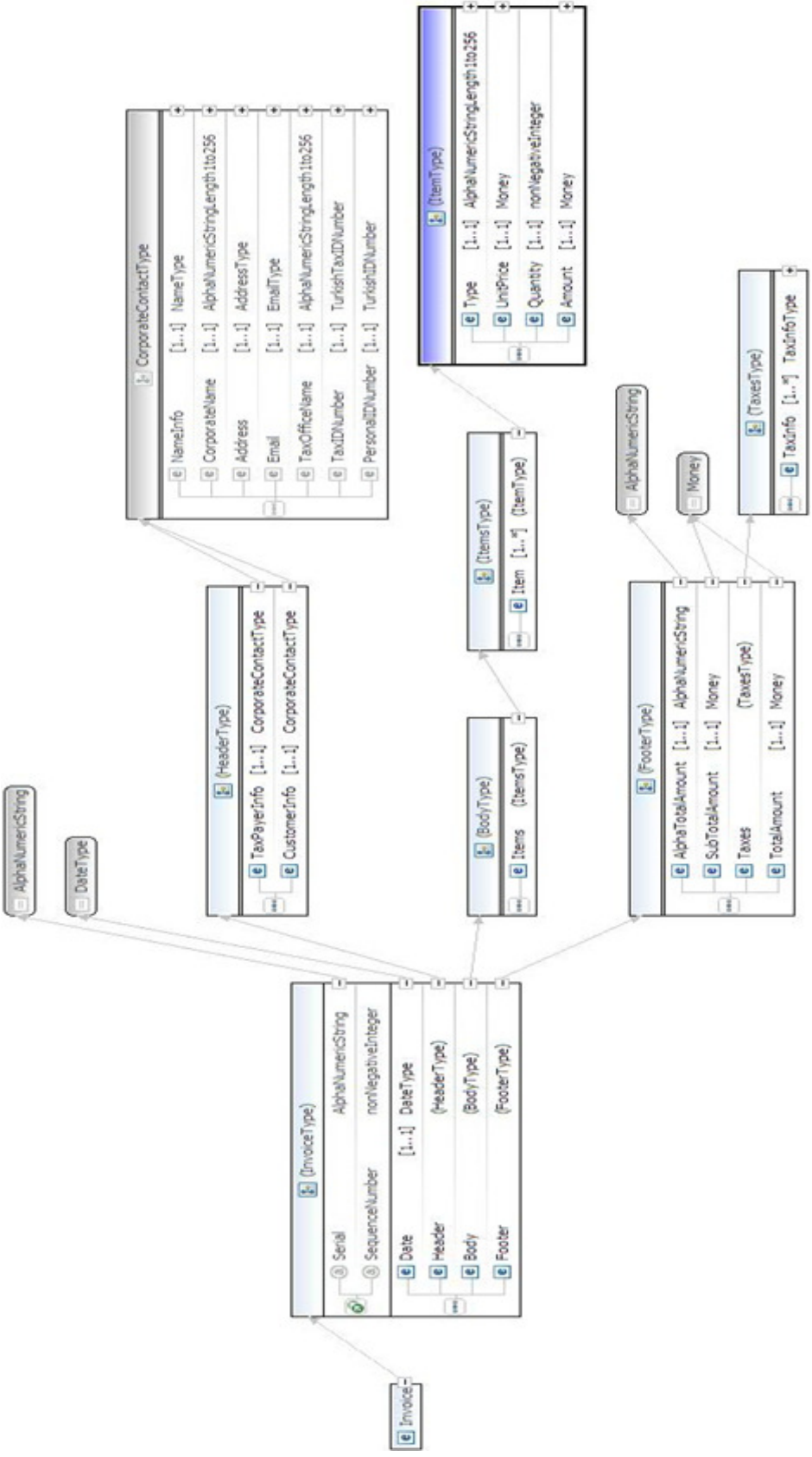
- **CUS_ID:** CUS tablosundan yabancı anahtar olarak bu tabloya entegre edilerek müşteri bilgisinin tablolar arası tutarlılığı sağlanmıştır.
- **ACC_ID:** Müşteriye ait her bir hesaba sistem tarafından atanan tekil numaradır. Bu kolon tabloda birincil anahtar durumundadır. Tabloda tekilliği sağlamak için CUS_ID ve ACC_ID kolonu bir arada kullanılır.
- **ACC_INFO:** Müşteriye ait hesabın tanım detaylarını XML olarak bu kolon tutar. Müşteriye ait hesabın veri doğrulunu bu kolon için tasarlanan SYSMNG_AccInfo şeması sağlar. Şekil 5.8'de bu şemanın yapısı görülmektedir. Bu tabloya veri girişinin olabilmesi için mutlaka müşteri bilgisinin girilmiş olması gerekmektedir. Hesaba bağlı müşteri, hesaplar silinmeden silinmemelidir. Bu bağlılığın güvenliği veri tabanı sunucusu tarafından sağlanmaktadır.



Şekil 5.8 Müşteriye ait hesap verisi doğrulama şeması

INVOICE : INVOICE müşteriye ait hesap için kesilen lisans faturalarını tutmak üzere tasarlanmıştır. INVOICE tablosuna ait kolon detayları :

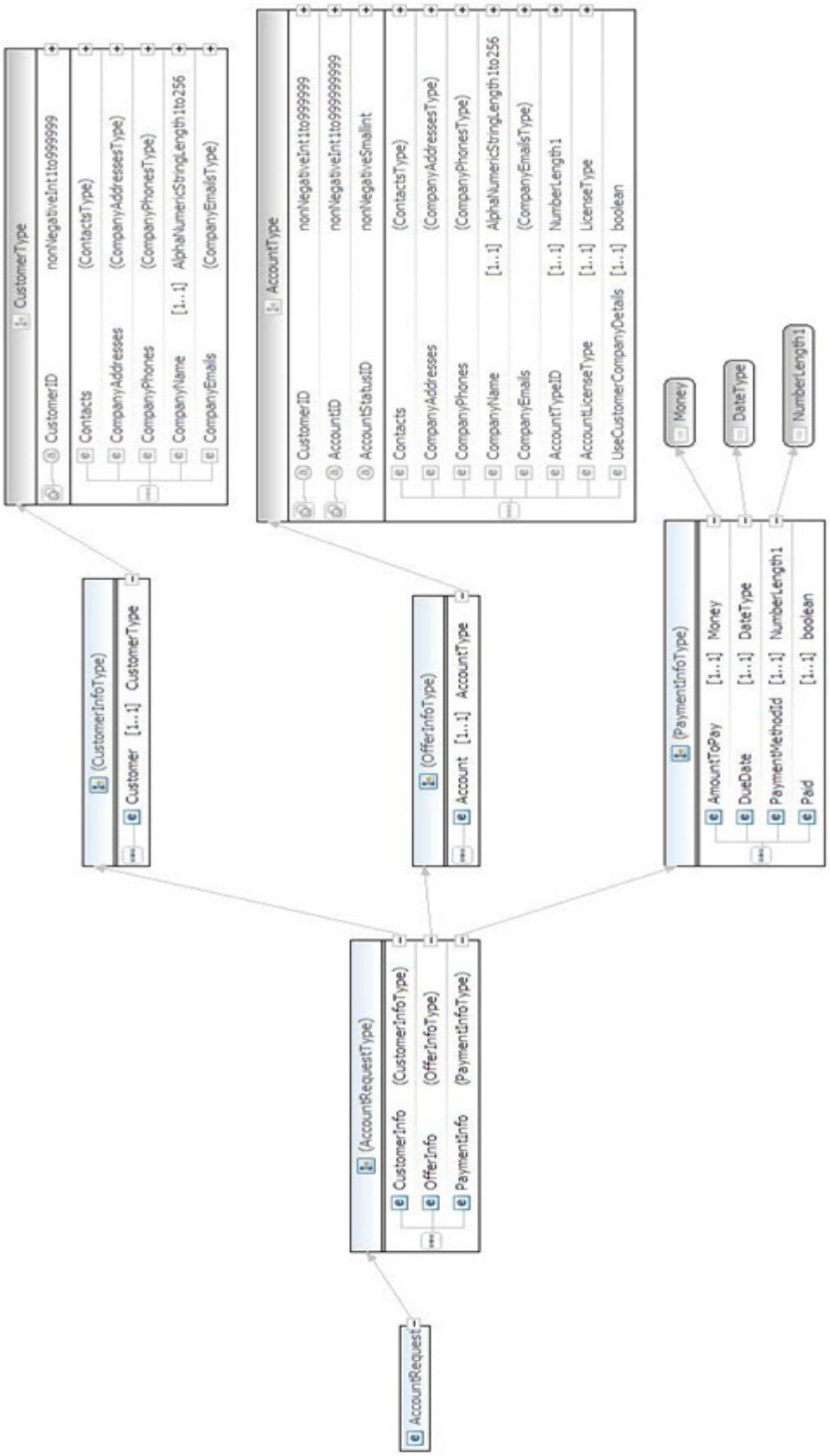
- **CUS_ID:** CUS tablosundan yabancı anahtar olarak bu tabloya entegre edilerek müşteri bilgisinin tablolar arası tutarlılığı sağlanmıştır.
- **ACC_ID:** ACC tablosundan yabancı anahtar olarak bu tabloya entegre edilerek hesap bilgisinin tablolar arası tutarlılığı sağlanmıştır.
- **INVOICE_ID:** Her bir faturaya özel atanan tekil numaradır. Bu kolon tabloda birincil anahtar durumundadır. Tabloda tekilliği sağlamak için CUS_ID ve ACC_ID ile birlikte kullanılır.
- **INVOICE:** Faturaya ait bilgilerin XML olarak tutulmasını sağlayan kolondur. Bu kolondaki fatura verisinin doğruluğunu SYSMNG_Invoice şeması sağlamaktadır. Bu şemanın yapısı Şekil 5.9 ‘ da görülmektedir. Invoice tablosuna veri girişinin olabilmesi için mutlaka müşteri ve hesap tanımlarının olması gerekir aksi takdirde bu tabloya veri girişi mümkün olmayacaktır.



Şekil 5.9 Fatura verisi doğrulama şeması

ACC_RQ: ACC_RQ, müşterinin hesap açma veya güncelleme talep bilgisini tutmak üzere tasarlanmış bir tablodur. Bu tabloya gelen hesap açma istekleri için müşterinin daha önceden sistemde tanımlı olmasına gerek yoktur. Çünkü ilk hesap açma talepleri kendi içerisinde müşteri bilgilerini de içeriyor olabilir.

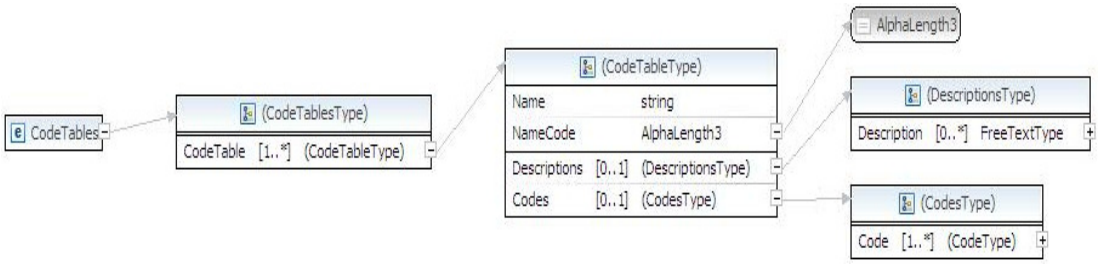
- **ACC_RQ_ID:** Her bir hesap açma veya güncelleme talebi için sistem tarafından üretilen tekil numaradır. Bu numara ile kaydın tablo içerinde tekilliği sağlanır. Bu kolon tabloya ait birincil anahtardır.
- **ACC_RQ_INFO:** Hesap açma veya güncelleme talebine ilişkin detay bilgileri tutan XML kolondur. Bu XML ' in içerisinde, yeni hesap talebi için müşteri bilgisinin tamamı olabilirken, mevcut bir müşterinin hesap talebi için müşterinin sadece müşteri numarası XML'in içerisinde yer alır. Aynı şekilde var olan hesap üzerindeki değişiklik talepleri için müşterinin hesap bilgilerinin tamamı değil sadece hesap numarası XML'in içerisinde yer alır. Ancak yeni taleplerde hesaba ilişkin tüm detay da bu XML içerisinde tutulabilir. ACC_RQ_INFO için yaratılan veri doğrulama şeması bu varyasyonları destekler biçimde tasarlanmıştır. Bunlara ek olarak, bu XML'de ödeme tipi ve durumu da tutulur. Şekil 5.10 bu XML'e ait verilerin doğrulamasını yapmak üzere tasarlanan SYSMNG_AccRq şemasının yapısını göstermektedir.



Şekil 5.10 Hesap talepleri verisi doğrulama şeması

SYS_CT: SYS_CT tablosu örnek senaryonun ihtiyaç duyduğu kod tablolarını tutmak için tasarlanmıştır. Örnek olarak ülkeler, ülkelere bağlı şehirler, para birimleri, hesap tipleri gibi kullanıcıya seçimlik olarak gelecek parametreler bu tablonun içeriklerini oluşturacak verilere örnek olarak verilebilir.

- **SYS_CT_ID:** Her bir kod tablosuna ait tekil numarayı tutan alandır. Bu kolon aynı zamanda birincil anahtar özelliğine de sahiptir.
- **SYS_CT:** Kod tabloları XML olarak bu kolonda saklanır. Her kod tablosunun içeriği farklı olacağından XML'in yapısı genel düşünülmüştür. XML içerisinde her bir kodun çok dilli bir yapıyı nasıl desteklediğini örneklemek için çok dillilik fonksiyonu entegre edilmiştir. Kod tablosunun veri bütünlüğünün doğrulamasını yapmak için MTS_CodeTable şeması tasarlanmıştır. Bu şemanın yapısı Şekil 5.11'de görülmektedir.



Şekil 5.11 Kod tablosu veri doğrulama şeması

SYS_USR: SYS_USR uygulamayı kullanan kullanıcıların fonksiyon bazında erişim yetkilerini tutabilmek için tasarlanmıştır. Bu tabloda sistem kullanıcılarının detay bilgileri de tutulmaktadır.

- **SYS_USR_ID:** Kullanıcıya ait sistem tarafından üretilen tekil numaradır. Tablo da birincil anahtar konumundadır.
- **SYS_USR_INFO:** Kullanıcı detayları XML olarak bu kolonda saklanır. Kullanıcı tanım verisini doğrulamak için SYSMNG_SysUsr şeması bulunmaktadır. Bu şemanın yapısı Şekil 5.12 ' de görülmektedir.

Şekil 5.12 Sistem kullanıcısı veri doğrulama şeması

5.3.1.Ortak veri tiplerinin tutulduğu şemalar

Genel tablo tanımlarının dışında, XML kolonlarda verilerin doğrulunu sağlayan şemalarda ortak kullanılan tipler ve kuralların her şema da tekrarlanmaması için MTS_SimpleTypes ve MTS_CommonTypes şemaları tasarlanmıştır. Bu şemalar içerikleri bakımında ortak veri tipleri ve doğrulama kuralları içerirler.

MTS_SimpleTypes: Bu şema, diğer şemalarda kullanılan ve karmaşık olmayan veri tiplerini içerir. Bu veri tipleri basit anlamda uzunluğu kısıtlanmış alfanümerik değerlerin tanımı, ondalık hanesi kısıtlanmış sayısal değerler, sadece pozitif ya da negatif sayılar gibi basit tanımlamalardan oluşur. Bu tanımlamalar XML şemasında standart olarak verilen veri tiplerinin dışına çıkmamıştır. Bu şema da tanımlanmış örnek bir basit tip aşağıda görülmektedir:

```
<xs:simpleType name="nonNegativeSmallint">
    <xs:annotation>
        <xs:documentation xml:lang="en">Değeri 1 ile 32767 arasındaki pozitif nümerik değer.xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:integer">
        <xs:minInclusive value="1"/>
        <xs:maxInclusive value="32767"/>
    </xs:restriction>
</xs:simpleType>
```

Yukarıdaki basit XML veri tipinin adı “nonNegativeSmallint”‘dir. Bu veri tipindeki “minInclusive” ve “maxInclusive” özelliklerinde tanımladığı üzere, bu tipi kullanan diğer şemalara ait alanlar değer 1 ile 32767 arası nümerik değerlerden birini alabilirler. Bu değerler dışında diğer karakter ya da nümerik değerlerden birini alan alanlar buradaki tanımdan dolayı doğrulama süzgecinden geçmezler.

MTS_CommonTypes : Bu şema, diğer şemalarda ortak kullanılan karmaşık veri tiplerini ve kurallarını tutmak için tasarlanmıştır. Karmaşık veri tiplerini basit veri tiplerinde ayıran en önemli özellik bu veri tiplerinin XML şemalarında tanımlı olmamasıdır. Uygulamanın ihtiyacına göre özel olarak geliştirilir. Tüm özellik ve alt alanları özel olarak tasarlanır. Karmaşık veri tipleri içerisinde basit veri tipi ya da başka bir karmaşık veri tipi içerebilir. Karmaşık veri tipi, bir kişinin detay bilgilerini içeren ve alt alanlardan oluşan bir tip olacağı gibi, farklı basit veri tiplerini bir araya getiren karmaşık bir tipte olabilir. Bu şema da tanımlanmış örnek bir karmaşık tip aşağıda görülmektedir:

```
<xs:complexType name="ContactType">
    <xs:annotation>
        <xs:documentation xml:lang="en">Genel kontak kurulacak
        kişi.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="NameInfo" minOccurs="1" maxOccurs="1"
        type="NameType" />
        <xs:element name="SalutationCode" minOccurs="1" maxOccurs="1"
        type="AlphaLength1to5WithDot" />
        <xs:element name="Title" minOccurs="1" maxOccurs="1"
        type="AlphaLength1to16WithDot" />
        <xs:element name="Suffix" minOccurs="1" maxOccurs="1"
        type="AlphaLength1to5WithDot" />
        <xs:element name="NickName" minOccurs="1" maxOccurs="1"
        type="AlphaLength1to16" />
        <xs:element name="WebPage" minOccurs="1" maxOccurs="1"
        type="WebPageType" />
        <xs:element name="JobTitle" minOccurs="1" maxOccurs="1"
        type="AlphaNumericStringLength1to32" />
        <xs:element name="JobRole" minOccurs="1" maxOccurs="1"
        type="AlphaNumericStringLength1to32" />
```

```
            <xs:element name="Department" minOccurs="1" maxOccurs="1"
            type="AlphaNumericStringLength1to32" />
            <xs:element name="Addresses" >
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="Address" minOccurs="1"
                        maxOccurs="5" type="AddressType" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
</xs:complexType>
```

Yukarıda ismi "ContactType" olan tek bir karmaşık tip görülmektedir. Bu tip kendine ait alanlardan oluşmaktadır. Bu alanların başka karmaşık ya da basit alanları kullandığıda görülmektedir. Örnek olarak, Address alanı diğer bir karmaşık tip olan AddressType'ı kullanırken, Department alanı AlphaNumericStringLength1to32 isimli basit tipini kullandığı görülmektedir.

Bu iki şemadan da görüldüğü üzere şemaları birbirlerine referans vererek kullanmak, veri doğrulaması için gerekli yapının tekrar kullanılabilirliliğini arttırdığı gibi, tasarımı da sadeleştirmiştir. Ayrıca doğrulamayı kod içerisinden çıkarıp, okunur bir yapı da tarif etmek doğrulama kurallarının bakımını da kolaylaştırmaktadır.

Alan tiplerinin şemalar içerisinde tanımlanması ve uygulamaya özel karmaşık alan tiplerinin oluşturulması XML şemalarda mümkünken, bu ilişkiyi ilişkisel veri modelinde kullandığımız kolon yapılarında yapmamız mümkün olmamaktadır.

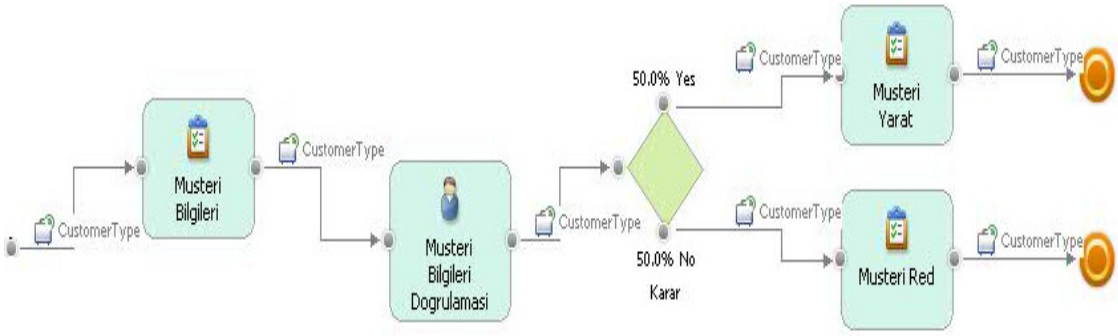
5.4. Örnek Olay Süreçleri

Örnek olayda iki adet süreç ve süreçleri yürütmek üzere tasarlanmış roller, ekranlar, servisler, XML şemaları ve bağlantı noktaları bulunmaktadır. Süreçler de kullanılan roller toplam dört adettir:

- Müşteri
- Satış temsilcisi
- Satış yöneticisi
- Muhasebe elemanı

Yukarıdaki rollerin yer aldığı iki adet süreç bulunmaktadır:

- **Müşteri yaratma:** Bu senaryo gereği, müşteri firmayla iş yapmak üzere başvuruda bulunur. Yeni müşteriden alınacak bilgileri müşteri kaydı için senaryomuzda hazırlanan SYSMNG_CusInfo XML şeması belirlemektedir. Firma için müşterinin istenen tüm bilgilerinin doldurulması zorunludur ve akış gereği müşterinin yerleşik adresi mutlaka Türkiye, Azerbeycan ya da Kıbrıs ‘ da olmalıdır ve bu kural süreç katmanında tanımlanan bir kuralla kontrol edilmektedir. Süreç gereği ayrıca satış temsilcisi müşteri bilgilerinin doğruluğunu onaylayacaktır. Şekil 5.13’de müşteri yaratma süreci görülmektedir. İlgili tasarım süreç modelleme aracında gerçekleştirilmiştir. Böylece sürecin akışı uygulama geliştirme katmanında soyutlaştırılmıştır. Ayrıca sürecin ihtiyaç duyduğu müşteri tipi (CustomerType) şeması ‘ da servis kayıt sisteminden gelmektedir. Böylece veri modeli için tasarladığımız şemayı süreç içinde de kullanarak gevşek bağlaşımlılık ve servis odaklı mimari prensipleri korunmaktadır.



Şekil 5.13 Müşteri yaratma süreci

Bu süreçte kullanılan servis müşteri yarat servisidir. Süreçte müşterinin yaratılması, ilgili XML verinin veri tabanı sunucusu üzerinde oluşturulması ve müşteri durum bilgisin aktif olması anlamına gelirken müşterinin red edilmesi durum bilgisinin reddedildi olması anlamına gelmektedir. Bu servis' te servis kayıt aracı üzerinde yönetilmektedir. Servis web servis olarak tasarlanmıştır ve kurumsal servis yolu üzerinde çalışmaktadır. Web servis olarak kurumsal servis yoluna gelen istek burada XML'e dönüştürülerek veri tabanı sunucusuna gönderilir. Böylece bağlantı katmanı da süreç katmanından ve uygulama katmanından soyutlanmış olur.

- Hesap Yaratma: Sistemde kayıtlı bir müşteri, firmadan lisans almak isteğinde mutlaka bir hesap açtırmalıdır. Hesap açma talebinin içerisinde satın almayı istediği lisansı seçebileceği gibi, var olan bir hesabının lisansını değiştirebilir ya da süresini uzatabilir. Bir müşteri birden fazla hesaba sahip olabilir. Senaryo gereği:
    - Müşterinin talebi Satış temsilcisi rolündeki kişiye gelir,
    - Satış temsilcisi, hesaba ait istenen lisans detaylarını ve lisansın fiyatını günceller ve talebi onaylar.
    - Satış temsilcisi tarafından onaylanan talep, satış müdürünün onayına gelir. Satış müdürü isteği onaylayabilir ya da reddedebilir.

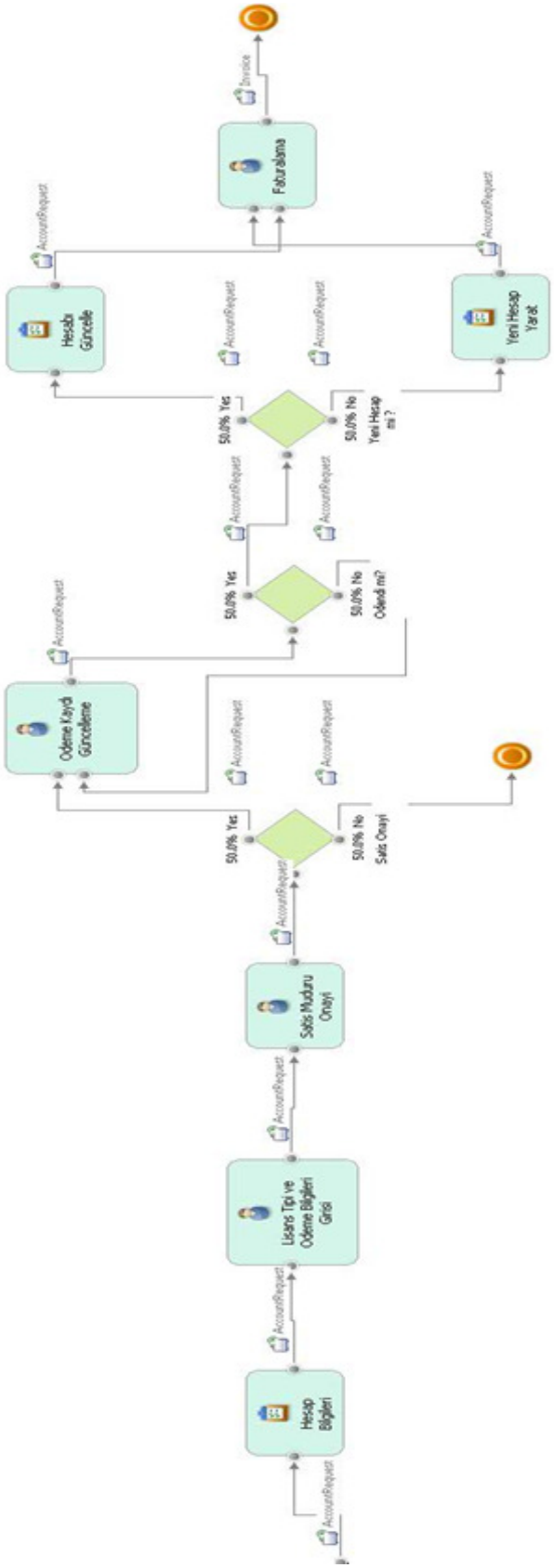
- Onaylan talep muhasebe elemanının onayına gelir. Muhasebe elemanı ödemenin alındığını doğrular ve hesabın güncellenmesi için talepte bulunur.
- Sistem, ödemesi alınan talebin ödeme bilgisini XML üzerinde kontrol eder ve hesabın daha önce sistemde olup olmadığını kontrol eder.
- Eğer sistemde olmayan bir hesapsa önce hesabı yaratır, eğer kayıtlı ise lisans bilgisini hesap üzerinde günceller.
- Son olarak, muhasebe elemanı isteği faturalar ve süreci bitirir.

Şekil 5.14'de süreç tasarım aracında tasarlanmış hesap yaratma süreci görülmektedir. Süreçte kullanılan servis ve XML şemalar bir önceki süreçte olduğu gibi servis kayıt sisteminde gelmektedir. Bu süreçte kullanılan servisler:

- Hesap yaratma
- Hesap güncelleme
- Faturalama
- Hesap isteği yaratma

Bu süreçte de bağlantı katmanlarında dönüşüm ve entegrasyon ihtiyacı için kurumsal servisi yolu kullanılmıştır.

Senaryo ' da kullanılan XML şemaların, senaryonun farklı katmanlarında kullanılması iş nesnelerini tanımlama ve kullanmada önemli avantajlar sağlamıştır. Bu konu etrafında gelişmiş ürünler veri tabanının ilişkisel kolonları üzerinde çalışmaktansa şemalar üzerinde çalışmayı tercih ederler. Böylece tasarımcıların şemalar içinde uygulamaya özel tanımladıkları tipleri de çeşitli katmanlarda kullanma olanağı sağlanmış olur.



Şekil 5.14 Hesap yaratma süreci

6. SONUÇ

Bu çalışmada servis odaklı mimarinin temel tanım ve bileşenlerinin genel açıklaması yapıldı.Özellikle servis tanımı üzerinde durularak nelerin servis olabileceği konusu anlatıldı. Servis odaklı mimari de kullanılan uygulama geliştirme bileşenlerinden de bahsedilerek mimarinin farklı katmanlarında kullanılabilecek teknolojiler detaylı olarak açıklandı.

Servis odaklı mimariyi uygulama da gösterebilmek için, bir örnek olay senaryosu tasarlanarak senaryonun mimari yapısı, senaryo da kullanılan ürünler, veri modeli ve süreçler modellenerek aktarıldı.

Servis odaklı mimarinin, tüm dünyada farklı sektörler için başarıyla uygulamaya alınmış olması mimarinin olgunluk seviyesini de yukarılara taşımış bulunmaktadır. Uygulama sorunlarına, katmanlar da soyutlama ile çözüm bulması ve platform bağımsız entegrasyon yöntemleri kullanıyor olması mimarinin en önemli avantajlarındandır.

XML'in bu mimaride ki yeri özellikle incelenerek XML ' in kullanım alanlarının neler olabileceği belirtildi. XML'in veri modeli olarak kullanılması, mimarinin katmanlarını geliştirirken gevşek bağlılığı ve tekrar kullanılabilirliliği destekleyerek uygulama sürecini kısalttığı örnek senaryo da görüldü.

7. ÖNERİLER

Servis odaklı mimarinin küçük ve orta ölçekli projelerde çevik metodoloji kullanarak geliştirilmesi ve hayata geçirilmesi, ayrıca bu mimari kullanılarak geliştirilen projelerin mühendisliğinde izlenecek yaklaşımların ayrı bir araştırma konusu olacağı düşünülmektedir.

Servis odaklı mimari de servislerin ve XML şemalarının nasıl bir metodoloji ve yaşam döngüsü ile yönetilmesi gerektiği ayrı bir araştırma konusu olabilir.

KAYNAKLAR

1. http://www.sei.cmu.edu/architecture/definitions.html, (07.03.2009)
2. Kruchten P., "The Rational Unified Process: An Introduction", Addison Wesley Professional, ISBN: 0-201-70710-1, Massachusetts, 2000
3. Rosen M., Lublinsky B., Smith K. T., Balcer M. J., "Applied SOA: Service-Oriented Architecture and Design Strategies", John Wiley & Sons, ISBN:9780470223659, Indianapolis, 2008.
4. Erl T., "Service-Oriented Architecture: Concepts, Technology, and Design", Prentice Hall PTR, ISBN: 0-13-185858-0, New Jersey, 2005
5. Hurwitz J., Bloor R., Kaufman M., Halper F., "Service Oriented Architecture For Dummies, 2nd Edition", John Wiley & Sons, ISBN:9780470376843 , Indianapolis, 2007.
6. Bieberstein N., Laird R. G., Jones K., Mitra T., "Executing SOA: A Practical Guide for the Service-Oriented Architect", IBM Press, ISBN:9780132353748, Boston, 2008
7. https://www.opengroup.org/projects/soa-book/protected/, (01.04.2009)
8. http://www.w3.org/TR/2008/REC-xml-20081126/, (08.04.2009)
9. Erl T., "Service-Oriented Architecture : A Field Guide to Integrating XML and Web Services", Prentice Hall PTR, ISBN: 9780131428980, New Jersey, Nisan 2004
10. http://www.w3.org/, (07.04.2009)
11. http://www.soaspecs.com/page2.asp, (16.03.2009)
12. http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec-os/wstx-wscoor-1.1-spec-os.html , (09.03.2009)
13. Chen W, Sammartino A., Goutev D., Hendricks F., Komi I., Wei M., Ahuja R., "DB2 9 pureXML Guide", IBM Redbooks , ISBN:9780738489834, New York, 2007.
14. http://www.ibm.com/developerworks/data/library/techarticle/dm-0709nicola/, (12.04.2009)

15. Powell G., “Beginning XML Databases”, Wiley, ISBN:9780471791201, Indianapolis, 2007
16. http://docs.oasis-open.org/soa-rm/v1.0/, (27.02.2009)

ÖZGEÇMİŞ

Ali BEKLEN, 1980 yılında İzmir / Balçova'da doğdu. Öğrenimlerini sırasıyla Ertuğrul Gazi İlkokulu, Güzelyalı Ortaokulu, Cumhuriyet Anadolu Meslek Lisesi Bilgisayar Bölümü'nde tamamladı. 1998 yılında Ege Üniversitesi Bilgisayar Programcılığı bölümünü kazandı ve 2000 yılında bölüm üçüncüsü olarak mezun oldu. Yaz stajını 2000 yılında Kentbank Bilgi Teknolojileri bölümünde tamamladı. 2003 yılında Anadolu Üniversitesi İktisat Fakültesi İktisat bölümünü kazandı ve 2007 yılında mezun oldu. 2000-2001 yılında Kentbank Bilgi Teknolojileri'nde yardımcı yazılım uzmanı olarak çalıştı. 2001-2006 yılları arasında Yapı Kredi Bilgi Teknolojileri'nde kıdemli yazılım uzmanı olarak görev aldı. 2006'dan bugüne IBM yazılım grubunda yazılım mimarı olarak çalışmaktadır. Profesyonel çalışma hayatında yurt içi ve yurt dışında yazılım konusunda çeşitli seminerlere konuşmacı ve katılımcı olarak katıldı. Ayrıca çeşitli üniversitelerde düzenlenen bilişim konferanslarında seminer vermiştir.