T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SERVİS ODAKLI MİMARİNİN (SOM) İŞ SÜREÇLERİ ÜZERİNE UYGULANABİLMESİ İÇİN BİR METODOLOJİNİN GELİŞTİRİLMESİ

DOKTORA TEZİ

Deniz Herand

Enstitü Anabilim Dalı : ENDÜSTRİ MÜHENDİSLİĞİ

Tez Danışmanı : Prof. Dr. İ. Hakkı CEDİMOĞLU

Temmuz 2013

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SERVİS ODAKLI MİMARİNİN (SOM) İŞ SÜREÇLERİ ÜZERİNE UYGULANABİLMESİ İÇİN BİR METODOLOJİNİN GELİŞTİRİLMESİ

DOKTORA TEZİ

Deniz Herand

Enstitü Anabilim Dalı : ENDÜSTRİ MÜHENDİSLİĞİ

Bu tez 01 / 07 / 2013 tarihinde aşağıdaki jüri tarafından Oybirliği ile kabul edilmiştir.

Prof. Dr. **Orhan TORKUL** **Jüri Başkanı**	**Prof. Dr.** **İsmail Hakkı CEDİMOĞLU** **Üye**	**Prof. Dr.** **Alptekin ERKOLLAR** **Üye**
Prof. Dr. **Muzaffer KAPANOĞLU** **Üye**	**Doç. Dr.** **Özden ÜSTÜN** **Üye**	

TEŞEKKÜR

Öncelikle çalışmalarımda katkılarını hiçbir zaman esirgemeyen, tezin oluşmasında ve sonuçlandırılmasında beni sürekli yönlendiren, problem bölgelerine sonuç odaklı yaklaşmam konusunda yardımcı olan tez danışmanım Sayın Prof. Dr. İsmail Hakkı CEDİMOĞLU'na sonsuz teşekkürlerimi sunarım.

Aynı şekilde çalışmam süresince engin bilgi ve tecrübeleriyle çalışmanın şekillenmesinde önemli katkılar sağlayan tez izleme komitesi üyeleri Sayın Prof. Dr. Orhan TORKUL ve Sayın Prof. Dr. Alptekin ERKOLLAR'a ayrıca teşekkür ederim.

Çalışmam süresince bana her türlü bilgi desteğini sunan Sayın Prof. Dr. Alfred ZİMMERMANN'a, bölüm başkanım Sayın Prof. Dr. Haldun AKPINAR'a, bölüm öğretim üyelerimizden Sayın Öğr. Gör. Murat GÜNSUR'a ve değerli dostumuz Sayın Sezgin ATASOY'a desteklerinden dolayı minnet duygularımı sunmak isterim.

Son olarak çok uzun ve zahmetli bir süreç olan doktora süreci boyunca maddi manevi desteklerini hiç esirgememiş bulunan değerli annem, babam ve kardeşime teşekkür eder saygılarımı sunarım. Bununla birlikte eşime bana sağladığı sınırsız moral sebebiyle sonsuz teşekkürlerimi sunarım. Her zaman sıkıntılılarımızı dinleyen ve yardımcı olmaya çalışan Sayın Doç. Dr. Çağla ERSEN CÖMERT'e ve Dr. Filiz GÜRDER'e manevi yardımlarından dolayı özel olarak teşekkür ederim.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

SOA : Servis Odaklı Mimari

IT : Information Technology – Enformasyon Teknolojileri

WS : Web Service – Web Servisi

WSDL : Web Services Description Language – Web Servisi Tanımlama Dili

WS – Policy : Web Services Policy – Web Servisi İlkeleri

XML : Extensible Markup Language – Genişletilebilir İşaretleme Dili

XML–Shema : Xml Dosyalarını Sınıflama Dili

MXML : Mining Xml

SLA : Service Level Agreement – Servis Seviye Anlaşmaları

PM : Process Mining – Süreç Madenciliği

UML : Unified Modelling Language - Birleşik Modelleme Dili

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar Kelimeler: Web Servisleri, Servis Odaklı Mimari (SOA), Servis Geliştirme Kriterleri, İş Süreç Yönetimi (BPM), Eski Sistemler (Legacy Systems), SOA Yönetim Platformları, Kurumsal Kaynak Planlama Sistemleri.

Küresel rekabet nedeniyle, tüm ürünlerin yaşam döngüleri her geçen gün kısalmaktadır. Yazılım ürünlerinde ise bu durum çok daha hızlı bir şekilde gerçekleşmektedir. Kurumlar için yazılım ürünlerinin değişim etkisi diğer birçok ürüne göre daha yüksektir. Bu nedenle, bu değişim etkisinin azaltılma ihtiyacı, şirket karlılığı için her geçen gün daha belirgin bir hal almaktadır.

Yazılım ürünlerinin değişim etkisinin yüksek olmasının sebebi yazılımların, esnek olmayan kırılgan, taşınmaz, gereksiz karmaşık, gereksiz tekrar içeren, anlaşılması zor bir yapıda geliştirilmelerinden kaynaklanmaktadır. Bu sorunun çözümü için günümüzde birçok teknik ve yaklaşım geliştirilmiştir. Son yıllarda bu alanda adından oldukça fazla söz ettiren Servis Odaklı Mimari yaklaşımı ise bunlardan biridir. Bu yaklaşım, özel bir yazılım tekniği gerektirmeden, barındırdığı kriterler çerçevesinde yazılımın değişim etkisini düşürebilmeyi öngörmektedir.

Kurumların SOA uygulamaya geçişi esnasında birçok sorunla karşılaşılabilmektedir. Karşılaşılan bu sorunlar ilerleyen zamanlarda katlanarak kurumlar için çekilmez bir hal alabilmektedir. Belirli bir yol haritası yoksunluğunda başlanan SOA'ya geçiş süreci ve ardından doğru bir uygulamanın takip edilmemesi, SOA'nın, kendisini kurtarıcı olarak gören kuruma, yararından daha fazla zarara yol açabildiği ise tecrübe edilen proje istatistikleriyle kanıtlanmaktadır. Kurumların SOA uygularken yaptıkları en büyük hata SOA'nın barındırdığı büyük resmi kaçırmalarından kaynaklanmaktadır. Kurumlar genellikle SOA'yı bir yazılım tekniği gibi algılayıp sadece teknik açıdan yararlı olabilecek küçük faydalarından yararlanmaya çalışmaktadırlar. Bu sebeple birçok kurum ilerleyen zamanlarda karşılaştıkları problemler sebebiyle SOA uygulamaktan vazgeçmekte ve SOA'nın yararlı olabileceğine dair inançlarını yitirebilmektedir. Bu çalışmada bu sorunun çözümü için "İş Süreçlerine SOA Uygulama Metodolojisi" geliştirilmiştir.

Geliştirilen metodoloji, bir üniversitenin Uzaktan Eğitim Merkezi'nin iş süreçlerine uygulanmıştır. Uygulama sonucunda altı servisli bir yazılım yapısı geliştirilmiştir. Bu yapı SOA ürünleri sunan, yazılım firmalarının ürünleri kullanılarak teknik ve görsel olarak son hallerine kavuşturulmuştur. Uygulama sonucunda elde edilen yazılım yapısının fonksiyonel olarak ihtiyaçlara cevap verebildiği ve değişim etkisinin, geliştirilen servisler aracılığıyla düşürüldüğü görülmüştür.

Geliştirilen metodolojinin SOA uygulamayı ve SOA'nın sağladığı en büyük fayda olan değişim etkisinin düşürülmesi özelliğinden faydalanmayı arzulayan tüm kurumlar tarafından bir yol gösterici olarak kullanılabileceği düşünülmektedir.

DEVELOPMENT OF A METHODOLOGY FOR SOA APPLICATIONS ON BUSINESS PROCESSES

SUMMARY

Keywords: Web Services, Service Oriented Architecture (SOA), Service Design Patterns, Business Process Management (BPM), Legacy Systems, SOA Management Platforms and Enterprise Resource Planning Systems.

Product lifecycles decrease every day due to global competition reasons. This event happens in the software products even more quickly. The change effect of software products on corporations is higher compared to other products. Therefore, the need for reduction of this change effect is becoming every day more explicit from the perspective of corporate profitability.

The reason for this high change effect of software is that it is usually developed inflexible, importable, unnecessarily complex, unnecessarily repetitive and less understandable in structure. Nowadays, some techniques and approaches are developed with the objective to solve this problem. SOA is one of these approaches which have made a reputation in recent years. The goal of this approach is to decrease the change effect within its own criteria framework without the need for any other special software technique.

Corporations encounter many problems while making transitions to SOA applications. These problems can increase with time and become intolerable for many of them. The statistics show that transitions to SOA without a proper guideline and without correct application follow-up can be more harmful than beneficial to corporations which may have seen SOA as a rescuer. The biggest mistake is that corporations usually miss the big picture which SOA contains while trying to apply it. Corporations which usually assume SOA as a software technique try to make use its minor benefits, which can be helpful only from technical perspective. Due to these reasons, many corporations which encounter problems over time cancel the application of SOA and lose their belief that it can be beneficial. Therefore, this study develops a methodology which can be assumed as a guideline for SOA Applications on Business Processes.

The developed methodology was applied on business processes of an external education centre of a university. A software structure which included six services was developed at the end of the application. This structure was completed technically and visually through using of SOA products supplied by leading software concerns. It was observed that the achieved software structure could meet the functional needs and the change effect was very low.

It is thought that the developed methodology can help as a guideline to corporations which want to apply SOA together with its most important benefit of decreased change effect.

BÖLÜM 1. GİRİŞ

1.1. Motivasyon

Servis Odaklı Mimari (Service Oriented Architecture - SOA) son yıllarda yazılım mimarisi alanında çok önemli bir yer tutmaktadır. SOA bir yaklaşım olup, bir yazılımın mimarisini oluştururken otomatikleştirilmesi gereken süreçlere ait fonksiyonların ele alınarak, bu fonksiyonların, belirli bazı prensipler temelinde servisler haline getirilmesini ve son olarak bu servislerin birbiriyle uyumlu ve belirli bir süreç akışına göre hareket ettirilerek (servislerin birleştirilebilmesi) otomatikleştirme işleminin gerçekleştirilmesi esasına dayanmaktadır.

Bu tezin konusu "İş Süreçlerine Servis Odaklı Mimari Uygulama Metodolojisi"'dir. Literatürde SOA ayrıntılı olarak incelenmiş, SOA uygulayan bir organizasyonun bir SOA'yı ne kadar iyi uyguladığının belirlenebilmesi için SOA olgunluk modelleri (SOA Maturity Model) geliştirilmiştir. Ancak bu olgunluk modellerinde yüksek bir seviye elde edebilmek için, organizasyonun nasıl hareket etmesi gerektiği, başka bir ifade ile en baştan hangi adımları izlemesi gerektiğine dair henüz bir uygulama metodolojisi geliştirilmemiştir. Literatürde SOA olgunluk modellerinin analizini en ayrıntılı şekilde gerçekleştirmiş olan Madlen Büttner de [1] 'Günümüzde SOA olgunluk modellerinin sayısının bu kadar çok arttığı göz önünde bulundurulursa, SOA uygulamaları için, özellikle SOA olgunluk modelleri formunda bir metodolojik destek ihtiyacının mevcut olduğu rahatlıkla görülebilir' yorumunda bulunmuştur. Bununla birlikte birçok bilimsel kaynak ve SOA uygulamayı arzulayan organizasyon da böyle bir metodolojinin eksikliğine değinmektedir. Tam olarak bu sorunun çözümü için bu tez kapsamında tüm iş süreçlerine uygulanabilecek genel bir metodoloji geliştirilmiştir.

Böyle bir metodoloji, SOA'nın sağladığı avantajları ile birlikte organizasyonlar için vazgeçilmez bir yaklaşım haline gelmesi ve organizasyonların bu önemli teknolojiyi

uygularken duydukları ihtiyaçtan dolayı geliştirilmeye karar verilmiştir. SOA'nın vazgeçilmez olduğu birçok analiz şirketinin raporlarında görülebilmekle birlikte, SOA, Garter'in stratejik teknoloji listesinde de uzunca bir süre ilk on teknoloji içinde yer almıştır. Bu gün artık bu listede iyi bir sebepten dolayı yer almamaktadır, zira stratejik teknoloji listesinde yer alan, bugünkü hemen hemen her teknolojiyi desteklemekte ve uzmanlar tarafından gizli güç teknoloji olarak görülmektedir [2],[3].

1.2. Çalışmanın Hedefleri ve Amacı

Bu çalışmanın amacı, yazılım sistemlerindeki değişim etkisinin azaltılmasını hedefleyen SOA yaklaşımının, uygulama uzmanları tarafından doğru algılanıp uygulanabilmesi için, uygulamaya başlangıç safhasını ve ilerleyen zamanlardaki ihtiyaç duyulan değişikliklerin sisteme entegrasyonunu kolaylaştıran, anlaşılabilir, gerekli durumlarda kapsam olarak genişletilebilen, SOA'nın hedeflerini eksiksiz olarak gerçekleştirebilen ve iş süreçleri üzerine kolaylıkla uygulanabilen, genel bir SOA uygulama metodolojsinin geliştirilmesidir.

1.3. Çalışmanın Yapısı

Çalışma bu bölüm ile birlikte toplam altı ana bölümden oluşmaktadır. Giriş bölümü olan bu birinci bölüm, bu çalışmaya neden gereksinim duyulduğunu açıklayan motivasyon bölümü, çalışmanın hedefleri ve amacı bölümü ve çalışmanın yapısını anlatan üç alt bölümden oluşmaktadır. İkinci bölüm, SOA'nın ve metodolojinin anlaşılabilmesi için gerekli olan temel konuları ele almaktadır. Üçüncü bölüm, metodolojinin çözüm getirmesi gereken problemleri, bu problemlerin örneklerini, ve literatür taramasını içermektedir. Dördüncü bölümde, çalışmanın amacı bölümde ifade edilen hedefler doğrultusunda "İş Süreçlerine SOA Uygulama Metodolojisi" geliştirilmiş ve metodolojinin her adımı açıklanmıştır. Dördüncü bölümde geliştirilen metodoloji, beşinci bölümde bir üniversitenin uzaktan eğitim merkezinin iş süreçlerine uygulanmıştır. Değerlendirme bölümü olan altıncı bölüm ile çalışma sonlandırılmıştır.

BÖLÜM 2. TEMEL KONU BAŞLIKLARI

2.1. Servis Odaklı Mimari

Pazar taleplerini dikkate alarak, sürekli yeni fikirler üretebilen ve fikirlerini hızlı bir şekilde pazara sunabilen, esnek ve çevik bir kurum yapısı oluşturabilmek, günümüzde bütün kurum yöneticilerinin hedefidir. Hızla yenilenen taleplerin karşılanma süresi de giderek kısalmaktadır. Yenilenen taleplere kısa sürede cevap verilememesi ise çoğu kez pazar kayıplarıyla sonuçlanabilmektedir [4],[5].

Bu sorunlar, fikirlerin hızlıca hayata geçmesini sağlayacak teknolojilere olan ihtiyacı ve bu teknolojileri sunmakla görevli IT birimleri üzerindeki yükü sürekli olarak arttırmaktadır. Bununla birlikte, doğru teknoloji kullanımının stratejik önemi giderek artmakta ve dijital iş dünyası ile gerçek dünya arasındaki sınırlar giderek ortadan kalkmaktadır [5],[7].

Dolayısıyla günümüzde, kurumların IT departmanları ile İş departmanları birbirleriyle çok daha uyumlu olmak ve birlikte çalışmak zorundadır. IT departmanlarının, taleplerin sürekliliği ve artan yoğunluk ile başa çıkabilen ve gelen talepleri öncelikle kurumun kendi iş gücü ve uzmanlık kapasitelerinin kullanılarak hızlıca ve kurum içerisinde karşılaşılacak en az tepki ile hayata geçirebilen bir yapı kazanabilmesi için yeni yaklaşımlar gerekmektedir. Servis odaklılık yaklaşımının mimari yansıması olan Servis Odaklı Mimari (SOA: Service Oriented Architecture), tam olarak bu değişen iş ortamının gerekliliklerini karşılamak için geliştirilmiş bir kurumsal mimari modelidir [5].

Forrester ve Gartner gibi firmaların araştırma sonuçlarına göre, SOA tarafından vaat edilen etkin sonuçlara sadece, SOA'yı bir kurumsal mimari modeli olarak ele alan kurumlar ulaşabilmektedir. SOA karar verici ve uygulayıcı tüm kademelerin ciddi ve sabırlı adımlarıyla, büyüyen süreçler şeklinde ele alınması gerekmektedir. Bir

SOA'nın vaat ettikleri ise, o SOA'nın uygulanma sürecinin giderek olgunlaşmasıyla ve artarak devam etmektedir. SOA'yı sadece bir teknoloji olarak algılayıp, stratejik bir vizyondan yoksun olarak yola çıkan firmalar ise, büyük resmi kaçırmakta ve beklenen faydaları görememektedir [5],[8].

2.1.1. Servis nedir?

Servis odaklı yaklaşımın temelini, servis kavramı oluşturmaktadır. Bir servis, kendisinden beklenen görev için birbiri ile uyumlu çalışan bir grup operasyondan (fonksiyon) oluşmaktadır. SOA yaklaşımı kapsamında bir servis, standart bir arabirime sahip kavramsal bir bileşen olarak tanımlanabilir [4],[5]. Bir servis, bu hali ile 'kavramsal servis' olarak isimlendirilebilir. Kavramsal servisler, aynı kavram bütünlüğünde, bilgisayar uygulamalarına dönüştürüldüklerinde ise, 'dijital servis' halini almaktadır [9].

Dijital servisler, kavramsal servislerin dijital yansıması gibidir [4],[5]. Kurumların, tedarikçileri ile ilgili işlemlerini yürütmek için kurdukları "Satınalma Birimi" üzerinden bir örnek verilerek bu durum açıklanabilir. Satınalma Birimi, işlevi ile ilgili tüm hizmeti kurum adına tek noktada alarak, iş süreçlerinde bir verimlilik hedeflemektedir. Bu birimin SOA için bir dijital servis olarak geliştirilmesi gerekirse, bir "Satınalma Servisi" oluşturulup, bu servise bağlı olarak da "Tedarikçi Bilgilerini Getir", "Yeni Tedarikçi Oluştur", "Tedarikçi Bilgilerini Güncelle", "Tedarikçiyi Sil" gibi satın alma işlevini yerine getirirken kullanacağı uygun operasyonlar tanımlanabilir [5],[10].

Daha teknik bir örnek olarak ise, bir tarayıcının (scanner) işlevlerinin servis olarak tanımlanması verilebilir. "Tarayıcı Servisi" muhtemelen "Belgeyi Tara", "İptal Et", "Çözünürlüğü Azalt", "Çözünürlüğü Arttır" "Kapan", "Açıl" gibi operasyonlar içerecektir [11].

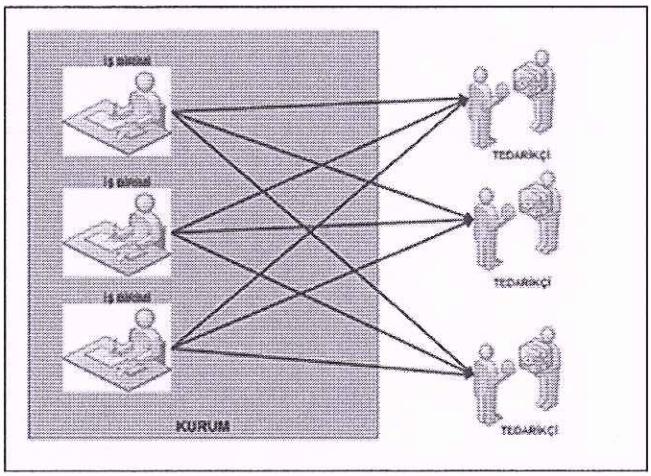
Gerçek dünyada bulunan nesneleri bir servis olarak kavramlaştırmanın en kolay yolu, o nesneyi bir kara kutu olarak düşünmek ve dışarıdan aldığı direktif ve bu direktiflere vereceği cevapları birer operasyon olarak tanımlamaktır. Örneğin bir iş departmanına, organizasyon yapısı ve iş süreçleri dikkate alınmaksızın dışarıdan

bakıldığında, bu departmanda yerine getirilen görevler, servis operasyonları olarak tanımlanabilir [5].

2.1.2. SOA'nın amacı

SOA'nın ortaya çıkışına yol açan en büyük neden, kurumlarda sık yaşanan değişimlerin etkilerini en aza indirme gayreti olarak tanımlanabilir. Bir başka deyişle, değişikliğe gidilen bir nesnenin zincirleme değişiklikler reaksiyonuna yol açmasını engelleme çabası olarak da nitelendirilebilir. Eğer bir bileşenin kendi görevlerini yerine getirmesi için bir diğer bileşenden faydalanması gerekiyor ise, bu ilişki bir bağlanma (coupling) olarak nitelendirilir. Satış departmanının bir satış işlemini tamamlaması için stok, lojistik ve muhasebe gibi departmanlara duyduğu ihtiyaç bir bağlanma örneğidir. Lojistik departmanının mal teslimi için kabul formlarında yapacağı değişikliğin satış departmanı tarafından öğrenilmesi ve kendini bu değişikliğe uyarlama çabası ise değişim etkisine (change impact) verilebilecek iyi bir örnektir [5],[7].

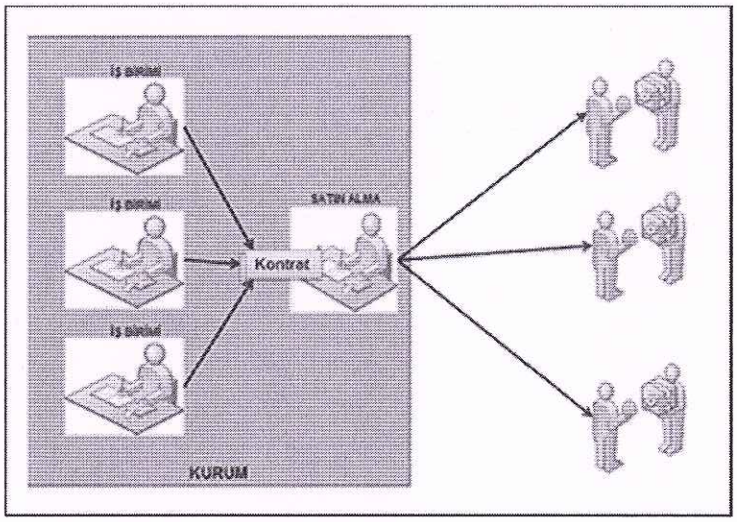
Bir başka bağlanma örneği olarak, iş süreçlerinin içerisine dahil edilen bir tedarikçi firma düşünülebilir. İş süreçlerinde, bu tedarikçi firmaya ihtiyaç duyan her iş birimi ile tedarikçi arasında bir eşleme oluşacaktır. Bu eşleme ilişkisinde doğal olarak tedarikçi ile olan ilişkiler, tedarikçinin gereksinimlerini temin etme şekli (formlar, sipariş alma, durum takibi vb.) ve diğer faktörler önemli olacaktır. Pazarda alternatiflerin oluşması ile tedarikçi sayısının zamanla artacağı düşünülebilir. Her tedarikçi, yeni ilişkileri, formları ve kendi iş süreçlerini beraberinde getirecektir. Bir tedarikçinin değiştirilmesi gerektiği zaman, o tedarikçi ile çalışmaya alışmış ve uyum geliştirmiş tüm iş birimlerinde sorunlar yaşanabilecektir [5].



Şekil 2.1. Klasik Yaklaşımdaki Değişim Etkisi [5]

Klasik yaklaşımda mevcut tedarikçiler ile sıkı bir ilişki içerisinde bulunulduğundan, ilgili tüm entegrasyon noktalarında, her yeni tedarikçi ile bir uyum süreci yaşama zorunluluğu ortaya çıkmaktadır (Bkz. Şekil 2.1). Bu uyum için harcanacak enerji, değişim etkisi için verilebilecek bir başka örnektir. Değişim etkilerinin yüksek olduğu bir şirkette alınan her karar, yapılacak her değişiklik çaba ve zaman gerektireceğinden, şirketin hareket kabiliyetini doğal olarak kısıtlamaktadır. Böyle bir durumda şirketin esnekliğinden ve çevikliğinden söz edilememektedir [5],[6].

SOA, bu durumdan kurtulmak için bağlanma ilişkisinin sözleşmeler (kontrat) üzerinden kurulması ilkesini getirmektedir (gevşek bağlanma). Servisler amaçlarını, sundukları yeteneklerini (operasyonlar), servisin nasıl kullanılacağını, mesaj yapılarını ve diğer özelliklerini bir servis sözleşmesi ile olası istemcilere sunmaktadır. Servisi kullanan taraf yani istemci, bağlanacağı servise, servisin nasıl ve kim tarafından sunulduğu ile ilgilenmeden, sunulmuş olan standart sözleşme üzerinden bağlanmaktadır. Sözleşme yerine getirildiği sürece, sözleşmenin arka planında olup biten değişiklikler, istemciye yansımamaktadır [5].



Şekil 2.2. Servis Odaklı Bir Kurumda Değişim Etkisi [5]

Tedarikçiler örneği için şirketlerin genellikle izlediği yol bir satınalma departmanı kurup iş birimlerinin taleplerini standart yollar ile toplamaktır. Yeni tedarikçi ekleme ya da tedarikçi değişikliği örneği ele alınacak olursa, servis odaklı bir kurum, tedarikçi değişikliğini hemen hemen hiç hissetmeden, sadece satın alma biriminde gerçekleştireceği uyum süreci ile yapabilecektir (Bkz. Şekil 2.2). İşte bu SOA'nin vaat ettiği esneklik ve çevikliğin en önemli örneklerinden birisidir [5].

2.1.3. SOA'nın sağladığı faydalar üzerine bir örnek

Örneğin açıklaması: Almanya kökenli iki rakip firma, Türkiye'de üretim sektöründe faaliyet göstermektedir. Her iki firmanın, Almanya'da bulunan üst yönetim kademeleri tarafından, Türkiye'de bulunan şubelerine (Firma A ve Firma B), içeriği aynı olan birer talep yöneltilmiştir. Aldıkları talepler karşısında, Türkiye'de faaliyet gösteren birimler, iş süreçlerini değiştirmek durumunda kalmaktadır.

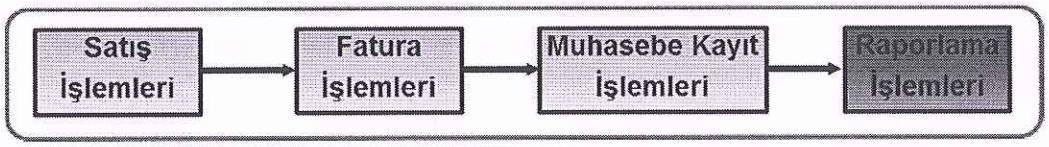
Örneğin amacı: İş süreçleri değişiminin, farklı kurumsal mimarilere sahip Firma A ve Firma B'nin IT sistemleri üzerinde yaratacağı değişim etkisini ortaya çıkarmaktır.

Firmaların IT sistemlerinin özellikleri: Firma A'da, SOA yaklaşımı uygulanmamakta ve günümüzde önemli bir pazar payına sahip olan Kurumsal Kaynak Planlama (ERP: Enterprise Resource Planning) Sistemi (örn: SAP 7.2) kullanılmaktadır. ERP

sistemleri, kurumların, tedarikten dağıtıma kadar olan tüm iş süreçlerinin, bütünleşik bir veri/bilgi yönetim sistemi desteğiyle yönetmesini sağlayan, geniş kapsamlı ve modüler yapıya sahip, yazılım paketleri olarak tanımlanabilirler [12]. Firma B'nin IT sistemi ise, SOA yaklaşımı uygulanarak geliştirilen bir sistemdir.

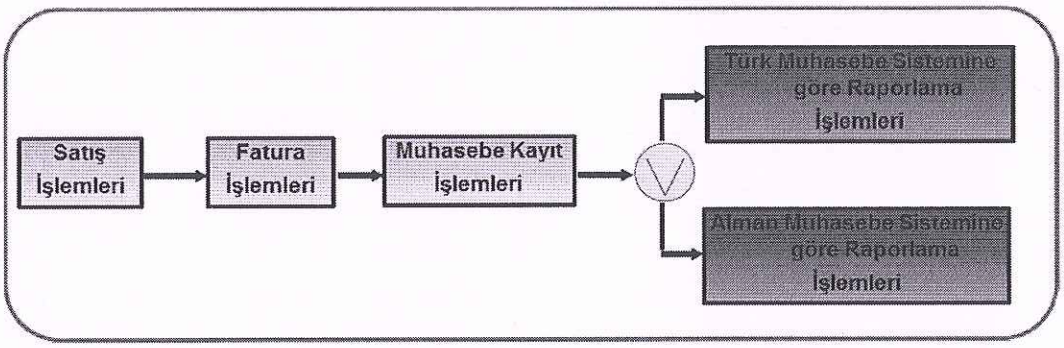
Talep: Firma A ve Firma B, Türkiye'de faaliyet gösterdikleri için, her iki firmanın finansal raporları, farklı yapıda olan IT sistemleri tarafından, doğal olarak Türk muhasebe sistemine göre düzenlenmektedir. Ancak bu firmaların Almanya'da bulunan üst yönetim birimleri, kendilerinin acilen Alman muhasebe sistemine göre düzenlenmiş finansal raporlara ihtiyaç duyduğunu bildirmektedir.

Talebe Yönelik Süreç Değişimi: Firma A ve Firma B'nin finansal rapor hazırlama üzerine mevcut ve genel iş süreçleri aynı olup, sadeleştirilmiş olarak, aşağıdaki modelde yansıtılmaktadır:



Şekil 2.3. Firma A ve Firma B'nin Finansal Raporlama Süreçleri

Firma A ve Firma B, yöneltilen talepler karşısında, finansal rapor hazırlama süreçlerini, aşağıdaki şekilde değiştirmek durumunda kalmışlardır. Değişim ile birlikte, yukarıdaki modelde kırmızı ile renklendirilen raporlama süreci, aşağıdaki kırmızı ile renklendirilmiş olan şekle dönüştürülmektedir. Yeni iş süreçlerine göre finansal raporlar, aşağıdaki şekilde, ihtiyaç duyulan ülkenin muhasebe sistemine göre hazırlanabilecektir.

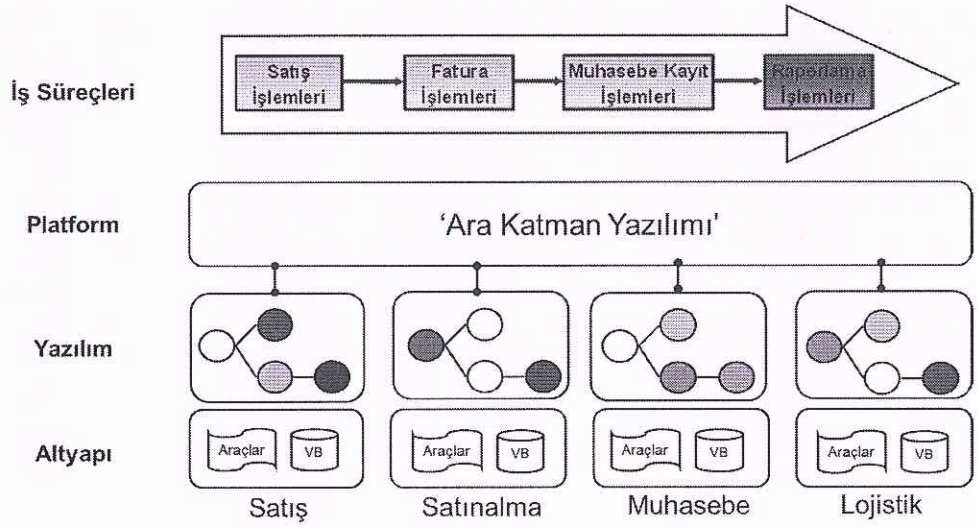


Şekil 2.4. Firma A ve Firma B'nin Değişen Finansal Raporlama Süreçleri

Değişim Etkisi: Bu bölümde, yukarıda modellenerek açıklanan süreç değişiminin, her iki firmanın IT sistemleri üzerinde yarattığı değişim etkisinin analizi yapılacaktır. Öncelikle SOA'den yoksun ve ERP sistemi kullanan Firma A'nın durumu incelenecektir [13].

2.1.3.1. ERP sistemi kullanan firma A'nın durumu

Şekil 2.5, ERP sistemi kullanan Firma A'nın IT sisteminin, süreç değişiminden önceki durumunu yansıtmaktadır. Şekildeki yatay bölümler, firmanın IT sisteminin katmanlarını göstermektedir. Dikey bölümler ise ERP sisteminin modüllerini ifade etmektedir. Satış, Satınalma, Muhasebe ve Lojistik modülleri bir ara katman yazılımı (middleware) aracılığıyla entegre edilmekte ve birbiri ile veri alış verişinde bulunabilmektedir. Yazılım katmanı, her modülün faaliyetlerine özel yazılımlarını canlandırmaktadır. Bu katmandaki daireler, bulundukları modüle özel yazılımın bir bölümünü (programlama dili ve türüne göre: fonksiyon, metot, prosedür, sınıf, nesne v.b.) simgelemektedir. Daireler arasındaki çizgiler, yazılım bölümlerinin, birbirine bağlı olarak hareket ettiği ve birbirinden ayrılamadığını göstermektedir. Farklı renklendirilen daireler, yazılımın, birbirini bütünleyen, farklı bölümlerini göstermektedir. Farklı modüllerde bulunan aynı renkli daireler ise, aynı özellikteki yazılım bölümlerinin farklı modüllerde kullanıldığını simgelemektedir. Buna örnek olarak müşteri işlemleri ile ilgili fonksiyonların, hem muhasebe hem de satış modüllerinde kullanılabileceği örneği verilebilir (Bkz. Şekil 2.5).



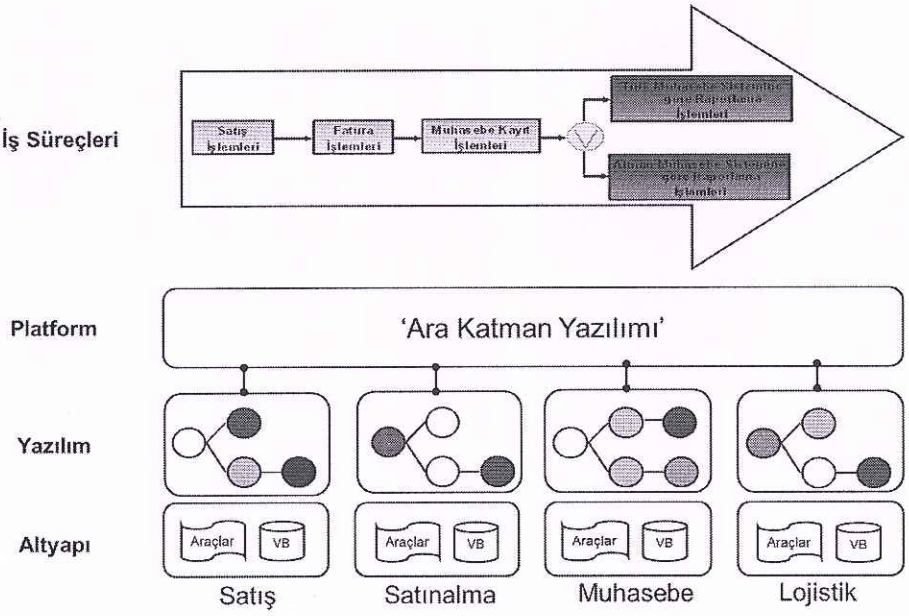
Şekil 2.5. ERP Sistemi Kullanan Firma A'nın IT Sistemi

İş süreçlerinde gerçekleştirilen değişikliğin, dijital dünyaya yansıtılabilmesi için, öncelikle Firma A'nın muhasebe modülünde değişikliğe gidilmesi gerekmektedir. Zira arzulanan süreç değişikliği (finansal raporlama sistemi) ile ilgili yazılım bölümleri genellikle ERP sistemlerinin muhasebe modülünde bulunmaktadır. Ancak, bu tarz sistemler üzerindeki değişimler, günümüzde sadece o sistemin uzmanları tarafından gerçekleştirilebilmektedir. Uzman kişiler ise, genellikle, ERP hizmeti sunan firmalar tarafından eğitilmekte ve kendilerine uzmanlık sertifikası verilebilmektedir. Gelinen bu durum, ERP hizmeti alan firmalar tarafından genellikle istenmeyen, teknoloji bağımlılıklarına ve uzman ihtiyacına yol açmaktadır. Modüller arasında veri alışverişine olanak sağlayan ara katman yazılımına olan bağımlılık ve iş süreçlerinin otomatikleştirilmesi için, modüler yapıda geliştirilen yazılımların mimari yapısına olan bağımlılık, teknoloji bağımlılığına örnek olarak verilebilir. Teknolojik altyapı konusunda uzmanlaşmış kişilere olan bağımlılık ise, uzmanlara olan bağımlılığın bir örneğidir.

Bununla birlikte unutulmamalıdır ki, günümüz ERP sistemleri üzerindeki değişimlere, sadece o ERP sistemini kullanan kişiler tarafından gereksinim duyulmamaktadır. ERP sistemi sunan firmaların, belirli periyotlarla kar ve kalite amaçlı olarak pazara sundukları sistemlerin yeni sürümlerinin kullanımına duyulan ihtiyaç (veya zorunluluk), aynı şekilde değişimlere yol açabilmektedir. O halde,

günümüzde kullanılan modüler ERP sistemlerinde gerçekleştirilecek her türlü değişim durumlarında, zaman, kalite, maliyet ve risk gibi ölçütler, genel olarak, kullanılan ERP sistem teknolojilerine ve bu teknolojileri uygulayabilen uzmanlara bağımlı kalmaktadır [15].

Tüm bu koşullar altında sağlanan değişimin ardından, Firma A'nın IT Sisteminin muhtemel yeni şekli aşağıdaki gibi olacaktır:



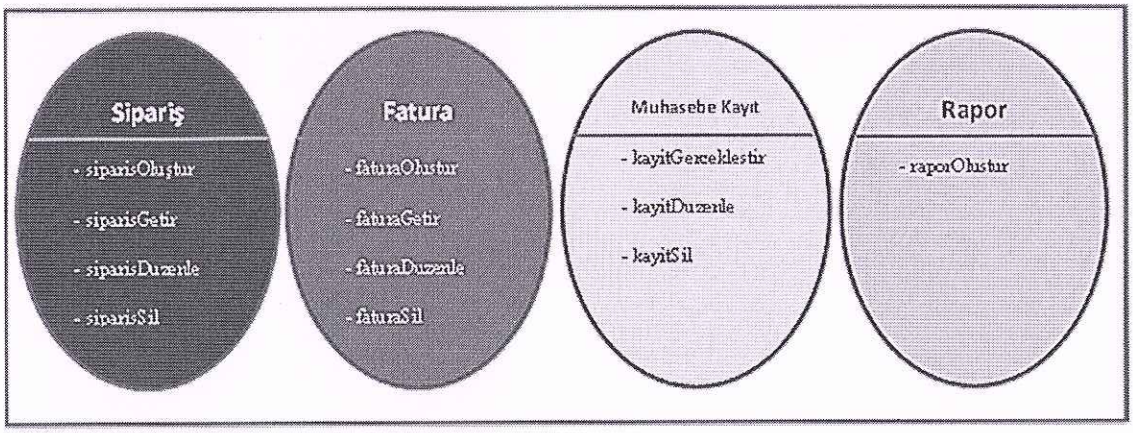
Şekil 2.6. Süreç Değişiminin Ardından Firma A'nın Muhtemel Yeni IT Sistemi

Şekil 2.6'da, Firma A'nın IT sistemine ait muhasebe modülüne, bir yazılım bölümünün (mor renkli daire ile ifade edilen bölüm) eklendiği görülmektedir. Ancak, talep edilen değişiklik için, sadece belirtilen ekleme işleminin yeterli olmadığı görülmektedir. Muhasebe modülünün mevcut yazılımına ait bölümler, birbirine standart ve ayrılabilen bir arabirim ile bağlı olmadıkları için, birlikte hareket etmektedir. Bu sebeple mevcut yazılım üzerinde yapılacak herhangi bir değişim (veya ekleme), yazılımın diğer bölümlerinde de bir takım değişiklikler (değişim etkisi) yapma zorunluluğunu doğurabilmektedir. Örnek üzerinde, gerçekleştirilen bu değişim, muhasebe modülündeki mavi renkli yazılım bölümünün, turuncu renkli bir bölüme dönüştürülmesi ile canlandırılmıştır. Bununla birlikte, hayatta değişmeyen tek şeyin değişim olduğu düşünülürse, bu sistem üzerindeki bir sonraki değişim

gereksiniminde, yine aynı şekilde teknoloji ve uzmanlara olan bağımlılık devam edecektir. Değişim etkisinin ise, buna bağlı olarak ilerleyen zamanlarda da azalmayacağı söylenebilir.

2.1.3.2. SOA uygulayan firma B'nin durumu

Aşağıdaki şekilde, IT Sistemi SOA yaklaşımına göre geliştirilmiş olan Firma B'nin, talep edilen süreç değişiminden önce sahip olduğu servislerden dört tanesi, içerdikleri fonksiyonlar (kavramsal servislerdeki operasyonlar) ile birlikte gösterilmektedir.



Şekil 2.7. Firma B'nin IT Sistemine Ait Servislerden Bir Kesit

Firma B'nin servisleri, SOA'nın öngördüğü kriterlere göre geliştirilmiştir. SOA'ın öngördüğü servis geliştirme kriterleri ve bu kriterlerin açıklamaları kısaca aşağıdaki gibidir [5],[14];

- Servis Sözleşmeleri (Service Contracts): Servisler, amaçlarını, sundukları yeteneklerini, servisin nasıl kullanılacağını, mesaj yapılarını ve diğer özelliklerini gösteren bir servis sözleşmesi ile sunarlar. Bir servisi kullanacak her türlü servis istemcisi, bu sözleşmeye göre hareket eder. Servis sözleşmeleri, istemciler tarafından kolayca yorumlanabilmeli ve standart bir şekle sahip olmalıdırlar.

- Gevşek Bağlanma (Loose Coupling): Servis istemcilerinin, bir servise olan bağımlılığı sadece o servisin sözleşmesine olan bağımlılıkla sınırlı kalmalıdır. Servisin iç işleyişinin nasıl olduğu, nasıl gerçekleştirildiği servis istemcilerini bağlamamalıdır.

- Kavramsallaştırma (Abstraction – Conceptualization): Kavramların özelliklerinin ve davranışlarının tanımlanmasına kavramsallaştırma denmektedir. SOA'da kalite kriteri olarak kavramsallaştırma ise, bir servisin, servis olmasını sağlayan özelliklerin tanımlanmasıdır. Kavramsallaştırma yapılırken, servisin isimlendirilmesinin ve servis içerisinde tanımlanan fonksiyonların birbirleriyle uyumu da (cohesion) son derece önem arz etmektedir. Bir servisin tanımlanması esnasında (encapsulation) yine servis istemcilerini ilgilendirmeyen gereksiz ayrıntıların sözleşmeye konmaması (information hiding) dikkat edilmesi gereken bir başka önemli durumdur. İyi tanımlanamamış servislerin ideal formlarına kavuşuncaya kadar değişmesi beklenen bir durumdur. Bu sebeple tanımlamalar dikkatli bir şekilde ve ideal durumu kazandıktan sonra yapılmalıdır.

- Yeniden Kullanım (Reuse): Çalışma mantığı, servislere ve fonksiyonlara dağıtılarak, servislerin yeniden kullanımı özendirilmelidir.

- Keşfedilebilirlik (Discoveribility): Servis sözleşmeleri, servisi kullanacak hedef kitle tarafından kolayca erişilebilir olmalıdır. Bu nedenle, servis sözleşmeleri, hedef kitle tarafından erişilebilen servis kayıt defterlerinde kayıt altına alınarak, servislerin olası istemciler tarafından keşfedilebilmesi sağlanmalıdır.

- Otonomi (Authonomy): Servisin, yeteneklerini kesintisiz ve güvenilir bir şekilde sunabilmesi için, kullandığı kaynaklar üzerindeki otoritesi tam olmalıdır.

- Birleştirilebilirlik (Composibility): Servisler, içerdikleri fonksiyonlar ile kendi görevlerinden başka, daha büyük ve farklı problem çözümlerinde kullanılabilmelidir. Bileşimin boyu ve büyüklüğüne bakılmaksızın, servisler, aktif çözüm bileşenleri olarak görev alabilmelidir. Bu işlem Kurumsal Servis

Yolu (ESB – Enterprise Service Bus) adı verilen bir ara yazılım aracılığıyla gerçekleştirilebilmektedir. ESB, servis fonksiyonları arasında oluşturulan ağı dikkate alarak, servislerin birbiri ile veri alışverişinde bulunmasını sağlamaktadır.

- Durumsuzluk (Statelesness): Durum, servis operasyonlarının, aynı istemci tarafından artarda kullanıldığı durumlarda, iki operasyon kullanımı arasında, istemci kullanımına özel verilerin saklanmasına verilen addır. Bu işlem, servisin işleyişini etkileyebileceğinden servislerin mümkün olduğu kadar durumsuz tasarlanması gerekmektedir.

Yukarıda açıklanan kriterler çerçevesinde geliştirilen Firma B'ye ait servislerin içerdikleri fonksiyonların açıklamaları aşağıdaki gibidir. Yatay şekilde yazılan kısım fonksiyonun adını belirtmektedir. Fonksiyonun adının sol tarafında bulunan parantez içindeki ifade, fonksiyonun girdisini, sol tarafındaki ifade ise, çıktısını belirtmektedir.

Sipariş servisi:

- public String *siparisOluştur* (Siparis siparis): Uygulamanın ön yüzü kullanılarak oluşturulan sipariş nesnesi servise verilerek kayıt yapılır. Sonuç olumlu ise 1 olumsuz ise 0 döner.

- public Siparis *siparisGetir* (long siparisID): Mevcut bir siparişe ID numarası ile erişimi sağlar. Siparis nesnesi döner. Eğer bulunamazsa *null* döner.

- public String *siparisDuzenle* (Siparis yeniSiparis): Girilmiş bir sipariş üzerinde değişiklik yapılmasını sağlar. Değiştirilen sipariş nesnesini alır, ID parametresi değiştirilmez, böylelikle mevcut sipariş bulunur ve üzerinde değişiklik yapılabilir. Sonuç olumlu ise 1 olumsuz ise 0 döner.

- public String *siparisSil* (long siparisID): ID'si verilen siparişin silinmesi gerçekleştirilir. Sonuç olumlu ise 1 olumsuz ise 0 döner.

Fatura servisi:

- public Fatura *faturaOlustur* (Siparis siparis): Önyüzde oluşturulan Siparis nesnesinden Fatura nesnesi oluşturarak database'e kaydı yapılır. Sonuç olarak oluşturulan Fatura nesnesi döndürülür.

- public Fatura *faturaGetir* (long faturaID): ID'si verilen Fatura nesnesini döndürür. Eğer bulunamazsa *null* döner.

- public String *faturaDuzenle* (Fatura yeniFatura): Girilmiş bir fatura üzerinde değişiklik yapılmasını sağlar. Değiştirilen fatura nesnesini alır. ID parametresi değiştirilmez, böylelikle mevcut fatura bulunup değiştirilebilir. Sonuç olumlu ise 1 olumsuz ise 0 döner.

- public String *faturaSil* (long faturaID): ID'si verilen faturanın silinmesi gerçekleştirilir. Sonuç olumlu ise 1 olumsuz ise 0 döner.

Muhasebe kayıt servisi:

- public MuhasebeKaydi *kayitGerceklestir* (Fatura fatura): Bu fonksiyon, aldığı Fatura nesnesinden MuhasebeKaydi nesnesi oluşturarak, bu nesnenin veritabanına kaydını gerçekleştirir. Sonuç olarak MuhasebeKaydi döner.

- public String *kayitDuzenle* (MuhasebeKaydi muhasebeKaydi): Girilmiş bir muhasebe kaydının üzerinde değişiklik yapılmasını sağlar. Değiştirilen muhasebe kaydı nesnesini alır. ID parametresi değiştirilmez, böylelikle mevcut muhasebe kaydı bulunur ve değiştirilebilir. Sonuç olumlu ise 1 olumsuz ise 0 döner.

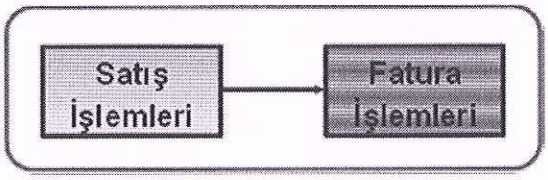
- public String *kayitSil* (long kayitNo): ID'si verilen muhasebe kaydının silinmesi gerçekleştirilir. Sonuç olumlu ise 1 olumsuz ise 0 döner.

Rapor servisi:

- public Rapor *raporOlustur* (String yil, String periyot): İstenen yıl ve periyot için database'den rapor çekilir. Sonuç olumlu ise Rapor nesnesi olumsuz ise *null* döner.

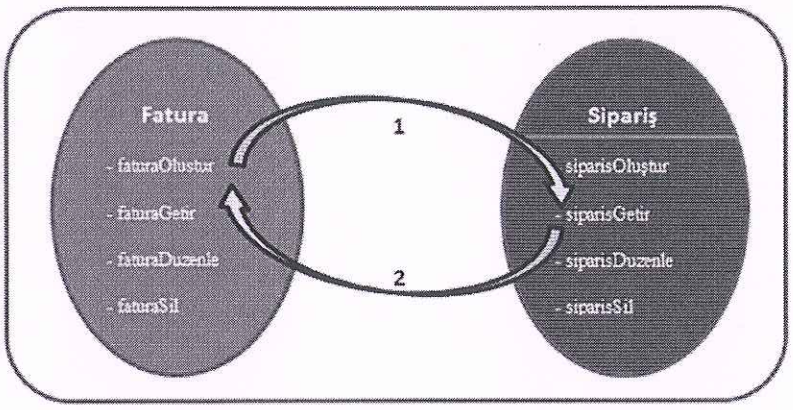
SOA kriterleri çerçevesinde geliştirilen servisler ve bu servislerin sahip olduğu fonksiyonlar açıklandıktan sonra, şimdi servislerin bir süreci otomatikleştirmek için nasıl kullanılabileceği açıklanacaktır. Servisler geliştirildikleri teknolojiden bağımsız oldukları için, süreçler aşağıdaki şekilde kolaylıkla otomatikleştirilecektir.

Fatura oluşturma süreci:



Şekil 2.8. Fatura Oluşturma Süreci

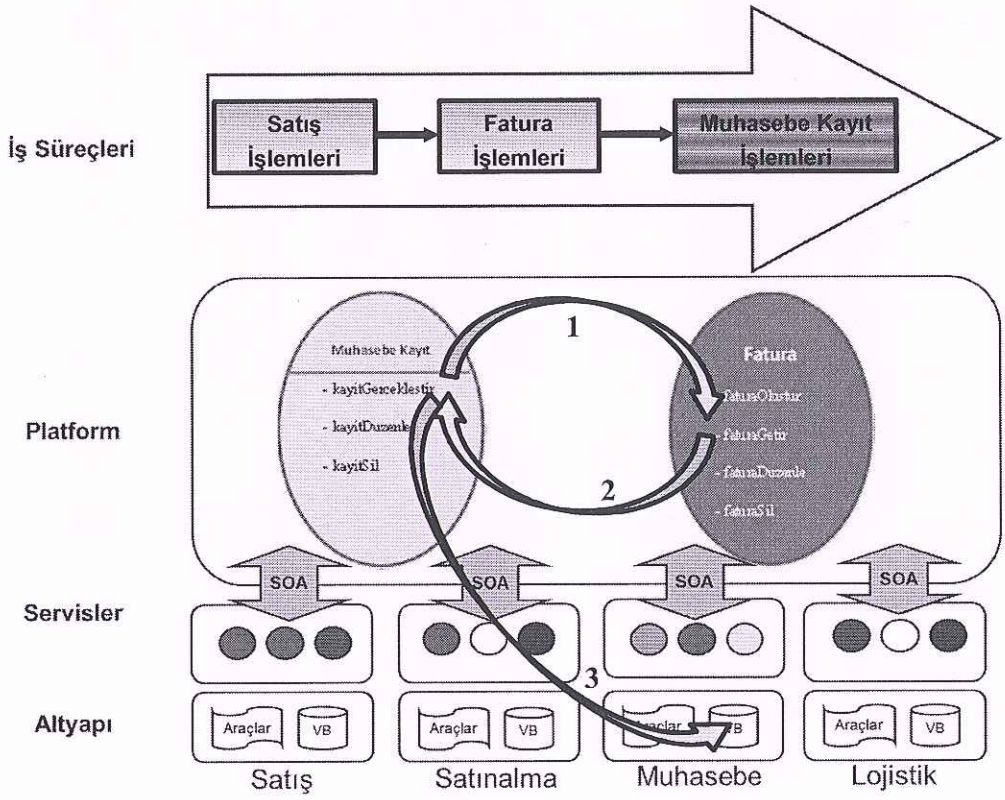
Fatura oluşturma süreci, sadeleştirilmiş olarak yukarıdaki modelde görüldüğü şekilde ilerlemektedir. Bu sürecin, servislerin birleştirilebilme özelliği kullanılarak otomatikleştirilmesi işlemi, aşağıdaki şekilde Fatura ve Satış servislerinin görevlendirilmesi ile sağlanabilir.



Şekil 2.9. Servisler Üzerinde Fatura Oluşturma Süreci

Açıklama: İşlem, süreci başlatacak fonksiyona sahip olan servisin çağırılması ile başlamaktadır. Süreci başlatan fonksiyon, ihtiyaç duyduğu verileri elde edebilmek için diğer servislerin fonksiyonları ile haberleşme yapabilmektedir. Bir fonksiyonun ihtiyaç duyduğu verileri hangi servis ve fonksiyondan alabileceğinin önceden bir ağ oluşturularak belirlenmesi gerekmektedir. Ağın oluşturulmasını ve ağdaki haberleşme sırasının takip edilerek verilerin taşınmasını, kurumlarda, Kurumsal servis yolu olarak adlandırılan bir platform sağlamaktadır. Fatura oluşturma örneğinde, Fatura servisi, ağ üzerinde 1 ile numaralandırılmış ok ile sipariş servisinden bilgi talep etmektedir. Sipariş servisi 2 numaralı ok ile cevabı iletmektedir. Fatura servisinin *faturaOlustur* fonksiyonu, aldığı cevabı girdi olarak kullanmakta ve işlemi yürütmektedir. Sürecin değişmesinin gerektiği durumlarda, ağ üzerindeki sıralama değiştirilebilir ve ihtiyaca göre, servislerin, başka servisler ile birleştirmesi sağlanabilir. SOA'nın sağladığı bu imkanla görülmektedir ki, geliştirildikleri teknolojiden bağımsız servisler, birbirine gevşek bağlı oldukları için değişime olanak vermekte, yapı olarak esnek ve çevik kalmaktadır. Aşağıda, farklı bir süreç olarak Muhasebe Kayıt süreci, Firma B'nin IT sistemi üzerinde gösterilmektedir.

Muhasebe kayıt süreci:

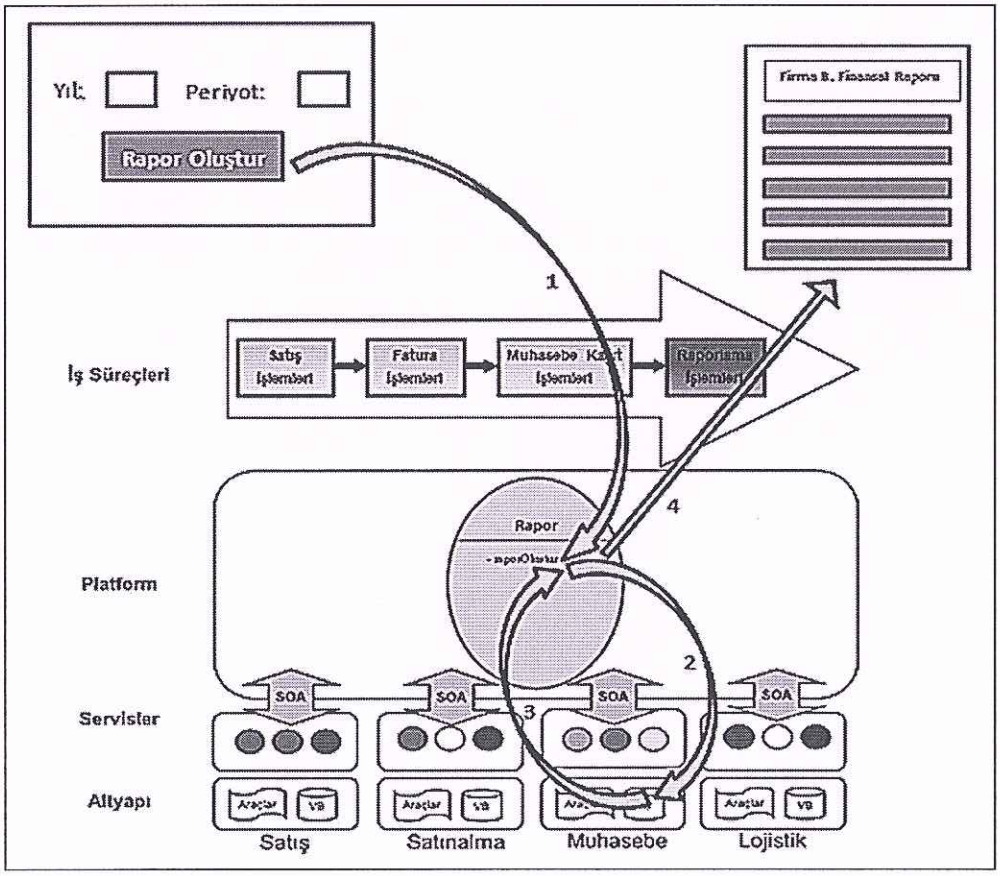


Şekil 2.10. SOA Uygulayan Firma B'nin IT Sistemi Üzerinde Muhasebe Kayıt Süreci

Açıklama: Dikkat edilecek olursa Firma A'nın IT sistemindeki Yazılım katmanının yerini servis katmanı ve Platform kavramını da ESB (Kurumsal Servis Yolu) almıştır. Servisler katmanında bulunan servislerin birbirinden bağımsız olduğu (Servislerin arasında çizgi yoktur. Firma A'nın sisteminde, yazılım katmanındaki yazılım bölümleri birbirine bağımlı olarak hareket edebilmekte idi. Bu durum, yazılım bölümlerinin çizgiler ile birleştirilerek modellenmesi ile gösteriliyordu.) açıkça görülmektedir. Bu bağımsızlık sayesinde bir süreç otomatikleştirilirken herhangi bir uzmana olan bağımlılık ortadan kalkmaktadır. Zira servisler ile ağ oluşturulması uzmanlık gerektirmemektedir. Muhasebe kayıt sürecinde, aynı şekilde, ESB aracılığıyla, Muhasebe Kayıt servisinin *kayitGerceklestir* fonksiyonu, Fatura servisinin *faturaGetir* fonksiyonundan gereken verileri alarak işlemini yürütme ve böylelikle süreç otomatikleştirilebilmektedir.

Bu aşamaya kadar Firma B tarafından uygulanan SOA'dan bazı kesitler alınarak, SOA'ın özellikleri tanıtılmıştır. Bu aşamadan sonra, gelen talep karşısında, süreç değişiminin, Firma B'nin üzerindeki değişim etkisi ve SOA'ın değişim durumlarında sağladığı kolaylıklar gösterilecektir.

Firma B'de talep öncesi rapor oluşturma süreci:

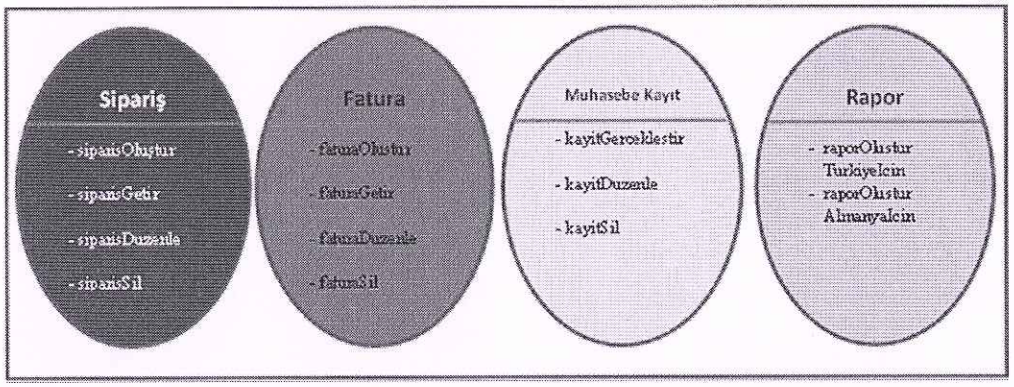


Şekil 2.11. Süreç Değişiminden Önce Firma B'nin IT Sistemi Üzerinde Raporlama Süreci

Açıklama: Yukarıdaki şekilde, raporlama sürecindeki işlemler, Firma B'nin IT sistemi üzerinde canlandırılmıştır. İşlem sırası ile şu şekilde gerçekleşmektedir; bir arayüz aracılığıyla, oluşturulacak finansal rapora ait yıl ve periyot bilgisini alan *raporOlustur* fonksiyonu, bağlı bulunduğu veri bankasından ilgili verileri alarak, finansal raporu oluşturmaktadır. Ancak gelen talep sonrasında, kırmızı ile işaretlenen

Raporlama süreci değişecek ve hem kullanıcı arayüzü ve hem de sistemdeki servislerin durumu bir sonraki bölümde anlatıldığı şekilde değişecektir.

Firma B'de talep sonrası rapor oluşturma süreci:



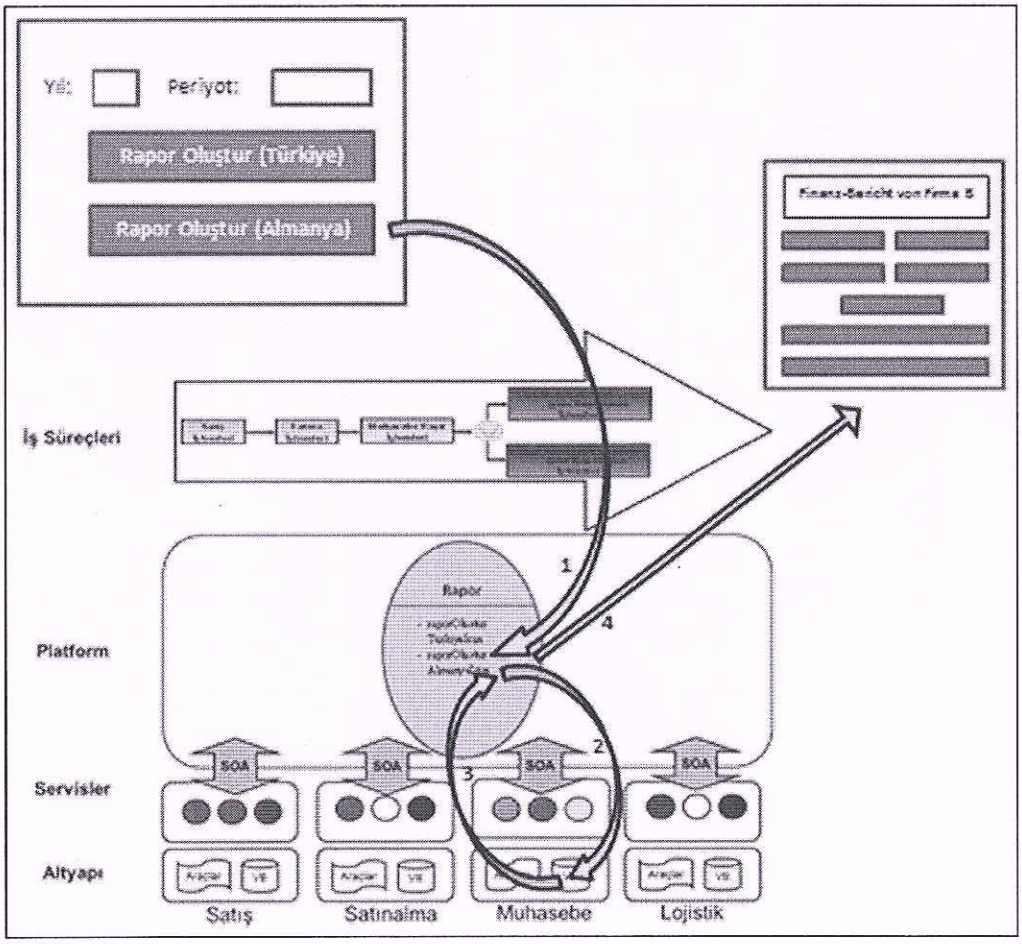
Şekil 2.12. Süreç Değişiminin Ardından Firma B'nin Servisleri

Servislerin yeni durumu yukarıda gösterildiği şekilde olacaktır. Görüldüğü üzere talep değişimi için sadece Rapor servisi üzerinde yapılacak bir değişiklik yeterli olacaktır. Ancak, bu değişiklik yapılmayarak, sisteme sadece talep edilen işlemi gerçekleştirebilecek bir servis eklenmesi yoluyla da talebe uygun değişim sağlanabilirdi. Değişen Rapor servisinin fonksiyonları aşağıdaki son halini almıştır.

Rapor servisi:

- public Rapor *raporOlusturTurkiyeIcin* (String yil, String periyot): İstenen yıl ve periyot için veritabanından Türkiye'ye göre rapor çekilir. Sonuç olumlu ise Rapor nesnesi olumsuz ise null döner.

- public Rapor *raporOlusturAlmanyaIcin* (String yil, String periyot): İstenen yıl ve periyot için veritabanından Almanya'ya göre rapor çekilir. Sonuç olumlu ise Rapor nesnesi olumsuz ise *null* döner.



Şekil 2.13. Süreç Değişiminin Ardından Firma B'nin IT Sistemi Üzerinde Raporlama Süreci

Açıklama: Süreç değişiminin ardından, sistemde bulunan Rapor servisine bir fonksiyon daha eklenmiştir. Bununla birlikte, kullanıcı arayüzü de yeni işlemleri gerçekleştirebilmek için değiştirilmiştir. Son duruma göre, Firma B'nin finansal raporları, Rapor servisinin ilgili fonksiyonunu çalıştırmak kaydıyla, ihtiyaç duyulan ülke sistemine göre oluşturulabilecektir. Bu örnekte, bir servise yeni bir fonksiyon eklenerek sorun çözülebilmiştir. Sistemde bu görevi gerçekleştirebilecek fonksiyona sahip başka bir servis bulunması durumunda, doğrudan o servis de sorun çözümü için görevlendirilebilmektedir. Görüldüğü gibi SOA uygulanan bir sistemde bağımlılık ortadan kalkmakta ve değişimin etkisi azalabilmektedir.

2.1.3.3. Sonuç ve SOA'nın tanımı

Bu örnek ile birlikte SOA'nın, kurumlarda yaşanan değişim durumlarında, günümüzde çoğunlukla kullanılan ERP sistemlerine oranla sağladığı avantajlar açıklanmıştır. Örnekte oldukça basit bir süreç üzerinde, basit bir değişiklik yapılarak, her iki sistemin değişikliğe karşı verdiği tepki ile, yaşanan değişim etkisi canlandırılmıştır. Piyasada faaliyet gösteren kurumların iş süreçlerinin çok daha geniş ve karmaşık olduğu göz önünde bulundurulursa, değişim gerektiren durumlarda, bu kurumların sistemlerinin göstereceği tepkiler ve yaşanabilecek değişim etkisinin hangi boyutlara ulaşabileceği de rahatlıkla tahmin edilebilir. SOA'nın, bahsedilen geniş kapsamlı kurumlarda da, aynı şekilde değişim etkilerini azaltabileceği söylenebilir. Sonuç olarak örnekteki iki sistem karşılaştırıldığında aşağıdaki tablodaki sonuçlar elde edilecektir.

Tablo 2.1. ERP ve SOA ile ERP Sistemlerinin Karşılaştırılması

ERP	SOA ile ERP
– Kurumlar, hem kurulum aşamasında hem de ilerleyen zamanlardaki değisim durumlarında özel teknolojilere ve uzmanlara bağımlı kalmaktadır. – Değişimler, istenmeyen etkiler oluşturabilmektedir. – Sistem üzerinde, anında müdahaleler yapılamamaktadır. – Esnek ve çevik bir yapıdan söz edilemez. – Sistemin aktif olması ve verim alınması oldukça hızlı	– Kurumlar, kurulum (uygulamaya başlangıç) aşamasında konunun uzmanlarına ihtiyaç duymakta, ancak ilerleyen zamanlardaki değişim durumlarında, özel teknolojiler ve uzmanlar ile ilgili bir bağımlılık yaşamamaktadırlar. – Değişim etkisinin hissedilmemesi amaçlanmaktadır. – Sistem üzerinde, anında müdahaleler yapılabilmektedir. – Sistem yapısı esneklik ve çeviklik kavramları üzerine kurulmuştur.

gerçekleşmektedir.	– Sistemin kurulabilmesi ve verim alınabilmesi için oldukça uzun bir zamana ihtiyaç duyulmaktadır.

Elde edilen sonuçlar ile birlikte, IT dünyasında, özel teknolojilere ve uzmanlara olan bağımlılığı ortadan kaldırmak ve kurumlardaki değişim etkisini azaltmakla görevli olan SOA'nın tanımı aşağıdaki şekilde yapılabilir.

SOA, esnek ve çevik kurumsal yapı tasarımı sağlayan, bilimsel bir yaklaşımdır.

2.1.3.4. SOA'nın dezavantajları ve zayıf yanları

SOA, sağladığı kolaylıklar ile birlikte, bir dizi dezavantajı da beraberinde getirmektedir. SOA'nın zayıf yanları ve dezavantajlarından bazıları şunlardır:

- Uygulamaya geçiş süreci: SOA'nın uygulanmaya başlanabilmesi için oldukça uzun bir planlama ve tasarım süreci gerekmektedir.
- Maliyet: SOA'nın uygulamaya geçiş aşaması, birçok farklı kurumsal mimari modeline göre oldukça yüklü bir maliyet gerektirmektedir.
- Eski sistem (Legacy system): Eski sistemler, SOA yaklaşımını negatif yönde etkilemektedir. Ancak, sıfırdan SOA uygulamaya geçişin de maliyetli olduğu düşünülürse, eski sistemin de dahil edilebileceği bir SOA uygulama olanağının sağlanması gerektiği görülebilir.
- SOA'nın benimsenmesi: IT personelinin, bugüne kadar kullandıkları SOA harici yaklaşımlardan uzaklaşarak, SOA bilimsel yaklaşımını benimsemesi kolay olmamaktadır.
- Bilimsel kaynak: SOA (özellikle Türkiye'de) oldukça yeni bir bilimsel yaklaşım olduğu için, SOA'ya dair (Türkçe) açık ve anlaşılabilir bilimsel kaynaklar zor bulunmaktadır.
- Kurumsal mimari uzmanları: SOA'nın uygulanmaya başlangıç aşaması için, yeterli donanıma sahip uzman sayısı yeterli değildir.

- Uygulama sayısı: (Özellikle Türkiye'de) SOA hakkında gereken tecrübenin kazanılabilmesi için, uygulanmış olan proje sayısı yeterli değildir.
- Servis seviye anlaşmaları: Servis sözleşmelerinin barındırdığı servis seviye anlaşmaları için uluslararası bir standart geliştirilmemiştir.
- Servis kaydı: Servis kaydı için uluslararası bir standart geliştirilmemiştir.
- Olay güdümlü SOA (Event Driven SOA): SOA'ya yapay zeka kazandırmak için geliştirilen Olay Güdümlü SOA (SOA 2.0) yaklaşımının kapsamı ve standartları henüz belirlenmemiştir.
- SOA yönetim platformları (SOA Suites): SOA yönetim yazılımları, piyasada çok sayıda bulunmaktadır. Ancak kurumlar, bu konuda seçim aşamasında zorlanmaktadırlar.
- SOA'yı destekleyen bilimsel yöntemler: SOA birçok farklı bilimsel yöntem ile birlikte çalışmaktadır (örn: BPM, BPR, Process Mining). SOA'nın uygulanmaya başlanabilmesi için bu yöntemlerin de bilinmesi gerekmektedir.
- SOA kavramı: SOA bilimsel bir yaklaşımdır. Ancak günümüzde birçok kişi ve kurum tarafından salt bir teknoloji olarak algılanmaktadır.
- Uygulama stratejisi: SOA doğru algılandığı durumlarda da, standart bir uygulama stratejisinin noksanlığından dolayı, çoğu zaman verimli sonuçlar elde edilebilecek şekilde uygulanamamakta, hatta birçok SOA projesi başlamadan son bulabilmektedir.
- Olgunluk modelleri: SOA olgunluk modelleri, piyasada çok sayıda bulunmaktadır. Ancak kurumlar, bu konuda seçim aşamasında oldukça zorlanmaktadır. Bununla birlikte piyasadaki olgunluk modellerinin hiçbiri SOA uygulama stratejisi içermemektedir.

2.2. Servis Geliştirme Kriterleri

Şu ana kadar SOA'nın faydalarından ve genel olarak yapısından bahsedilmiştir. Servislerin de SOA'nın yapı taşı olduğu belirtilmiştir. Bu aşamada ise, biraz daha detaya inilerek SOA'yı tasarlarken dikkat edilmesi gereken tasarım prensiplerine değinilecektir. Bu prensipler aslında doğrudan servisleri ilgilendirmektedir. Servislerin, ileriki bölümlerde anlatılan kriterlere göre geliştirildiği durumlarda, SOA kalitesi de orantılı olarak artmaktadır.

Kriterler aşağıdaki listede görüldüğü şekildedir [7];

- Servis sözleşmeleri
- Gevşek bağlanma
- Kavramsallaştırma
- Yeniden kullanım
- Keşfedilebilirlik
- Otonomi
- Birleştirilebilirlik
- Durumsuzluk

2.2.1. Servis sözleşmeleri

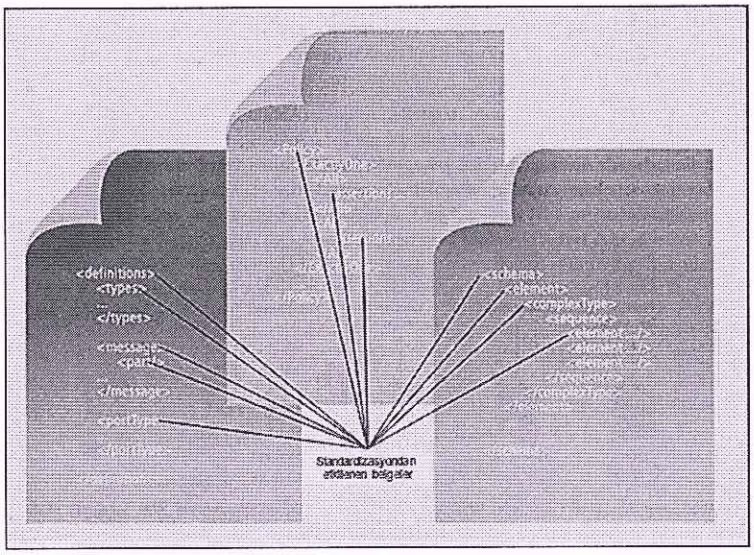
Servislerin birbiriyle iletişime geçebilmesi için birleştirilmeleri, iş süreçlerinin otomatikleştirmesi için gerekli bir durumdur. Ancak bir iletişimin gerçekleşebilmesi için, günlük hayatta da olduğu gibi, tarafların birbirini anlayabilmesi gerekmektedir. Bu durum SOA'da standart bir sözleşme geliştirilerek sağlanmaktadır. Böylelikle, bir servisin diğeri ile iletişime geçebilmesi için sadece geliştirilen sözleşmeye bağlı kalması yeterli olmaktadır. İletişime geçilecek servisin, nasıl ve ne şekilde geliştirildiği hiçbir önem teşkil etmemektedir [14].

Şekil 2.14. Servis Sözleşmesi'nin Yapısı [7]

Şekil 2.14'te örnek bir servis sözleşmesinin yapısı görülmektedir. Servisler iki ayrı bölümden oluşmaktadırlar. Bu bölümler teknik ve sözel bölümlerdir. Şekil 2.15 teknik kısımla ilgili örnekleri ve standardizasyondan etkilenen birimleri göstermektedir. Teknik kısım aşağıda ifade edilen üç ayrı bölümden oluşmaktadır [16];

- WSDL: Web Services Description Language: Verilerin, XML tabanında iletimi için programlardan ve platformalardan bağımsız tanımlama dili.
- XML-Schema: XML belgelerini sınıflandırma ve bu belgelerin yapı ve içeriğinin tanımlanma dilidir.
- WS-Policy: Servis sunucularının, XML tabanında servis kullanıcıları için uyması gereken güvenlik kalite ve versiyon ile ilgili kuralları kullanıma sunan şartnamedir.

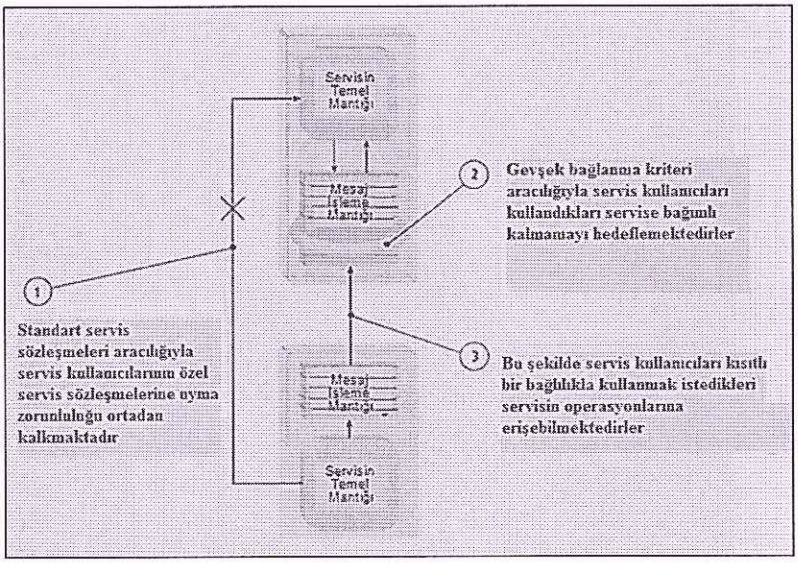
SLA ise servis sözleşmesinin sözlü detaylarını içeren kısmıdır. Bunlara örnek vermek gerekirse, servisin hangi saatler arasında kullanıma açık olacağı, servis yeteneklerinin garanti edilen cevap zamanları, yeteneklerin ortalama cevap süresi, kullanım istatistikleri, servis satın alınıyorsa veya satışa sunuluyorsa maliyeti vs. verilebilir [17].



Şekil 2.15. Teknik Servis Belgeleri [7]

2.2.2. Gevşek bağlanma

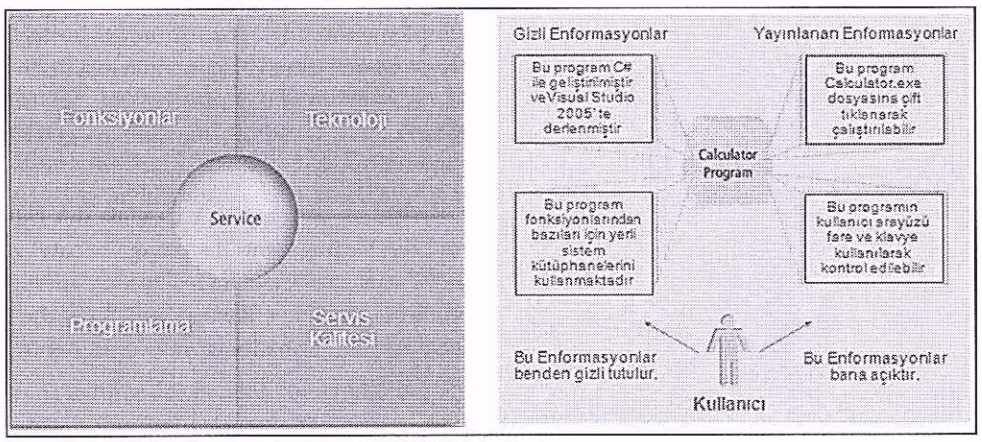
Servisler arası gevşek bağ kullanıcı servisin sunucu servise kolayca bağlanabilmesi ve veri iletişimine geçebilmesini esas alır. Bunun için servisin temel işlevlerden uzak olarak sadece mesaj işleme mantığına uyması gerekmektedir. Aksi takdirde negatif bir bağlılık meydana gelmekte bu da teknolojiye olan bağımlılığı beraberinde getirmektedir. Gevşek bağlanma kriteri sayesinde, servislerin kendilerine özel olan sözleşmelere uyma zorunluluğu ortadan kalkmaktadır. Bu durum servis sözleşmeleri sayesinde gerçekleşmektedir [18].



Şekil 2.16. Servisler Arası Gevşek Bağlanma [7]

2.2.3. Kavramsallaştırma

Bu kriter, sunucu servise ait meta verilerin, dengeli bir oranda yayınlanması gerektiği esasına dayanmaktadır. Kötü niyetli kullanıcıların sahip olmaması gereken verilerin yayınlanmaması sunucu servisimizin güvenliği için önem teşkil etmektedir. Şekil 2.17'de hesap makinası adlı bir servis ile ilgili, sol taraftaki meta verilerin sağ taraftaki simetrisinde, örnek olarak hangi verilerin yayınlanması ve hangilerinin yayınlanmaması gerektiği canlandırılmaktadır [9],[19].



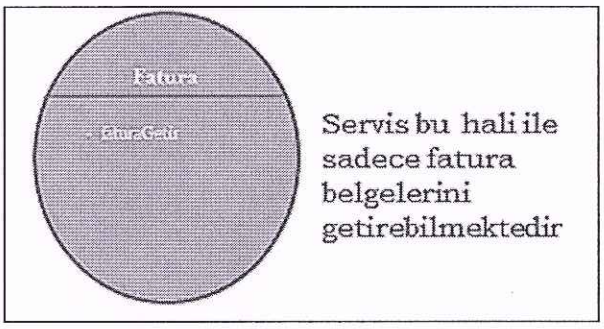
Şekil 2.17. Servislerin Kavramsallığı [7]

2.2.4. Yeniden kullanım

SOA'nın önem verdiği konulardan biri olan yeniden kullanım, servislerin birçok defa kullanılabilecek bir yapıda geliştirilmeleri gerektiği esasına dayanmaktadır. Organizasyonları yüksek bir maliyetten kurtaran bu kriter aynı zamanda, zaman ve iş yükü açısından da büyük faydalar sağlamaktadır. Yeniden kullanım, aşağıdaki gibi üçe ayrılarak incelenebilir.

2.2.4.1. Taktiksel yeniden kullanım

Taktiksel tekrar kullanılabilirlik, bir servisin, sadece anlık otomatikleştirmesi gereken iş biriminin hedef alınması ve bu servisin gelecek için devamlı geliştirmeye açık bırakılarak geliştirilmesi esasına dayanmaktadır. Örneğin anlık olarak sadece faturalama belgelerini çağırabilecek bir servise ihtiyaç duyuluyorsa bunu bu yeteneğe sahip bir servis geliştirerek gerçekleştirebiliriz, ancak servisin gelecekte başka yetenekleri de içerebilecek şekilde geliştirilmesi gerekmektedir. Şekil 2.18'de bu durum örneklendirilmiştir [14],[20].



Şekil 2.18. Taktiksel Yeniden Kullanım [7]

2.2.4.2. Hedefli yeniden kullanım

Servisin şimdiki ve yakın gelecekte ihtiyaç duyulacak fonksiyonları (yetenekleri) içerecek şekilde geliştirilmesi esasına dayanır. Belirli fonksiyonlara anlık olmasa da gelecekte ihtiyaç duyacağımızdan emin olduğumuz durumlarda servisin bu kriter

doğrultusunda geliştirilmesi gerekmektedir. Şekil 2.19 bu durumu örneklendirmektedir [9],[21].

Şekil 2.19. Hedefli Yeniden Kullanım

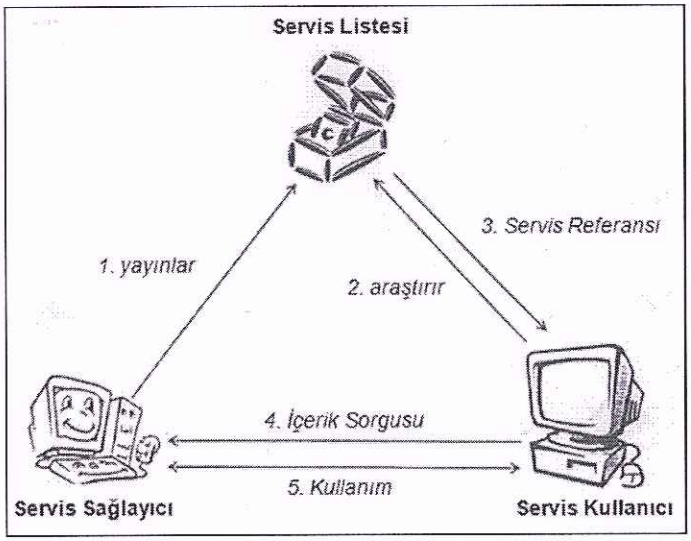
2.2.4.3. Eksiksiz yeniden kullanım

Kusursuz bir servis envanterine sahip kurumlar, servislerini acele geliştirmek zorunda kalmadan, geniş ve bu alanda ihtiyaç duyulabilecek her türlü fonksiyonu içerebilecek şekilde geliştirebilirler. Sonuçta servis içindeki herbir yeteneğin gerektiğinde başka bir servisin yeteneği tarafından çağırılabileceği ve kullanılabileceği düşünülürse, servisin mümkün olduğunca tam fonksiyonel geliştirilmesi organizasyonun yararına olacaktır. Şekil 2.20 bu durumu örneklendirmektedir [14],[22].

Şekil 2.20. Eksiksiz Yeniden Kullanım

2.2.5. Keşfedilebilirlik

Çok büyük organizasyonlar düşünüldüğünde, kullanılması gereken servis sayısı göz önüne alınırsa, aranan özellikteki bir servisin keşfedilebilmesi zorlaşmaktadır. Bu durum, daha da geniş bir yapıda, organizasyonlar arası servis paylaşımı olarak ilerletilirse servislerin bulunabilirliği ya da keşfedilebilirliği daha da zorlaşmaktadır. Bu sorun, ancak Şekil 2.21'deki gibi bir yapıyla çözülebilir. Bunun için, servislerin kaydedilebileceği bir liste görevi gören bir yapı, servisleri bu listeye kaydedecek sunucular ve servis ihtiyacı olan kullanıcıların birbirine bağlanabilmesine ihtiyaç duyulmaktadır. Bu şekilde, sunucular servislerini ve özelliklerini yayınlamaktadırlar. Kullanıcılar ise, listeden aradıkları servisin bu güne kadar yayınlanıp yayınlanmadığını, yayınlandıysa da sunucuyla nasıl iletişime geçeceklerini buradan öğrenebilirler [23],[26].

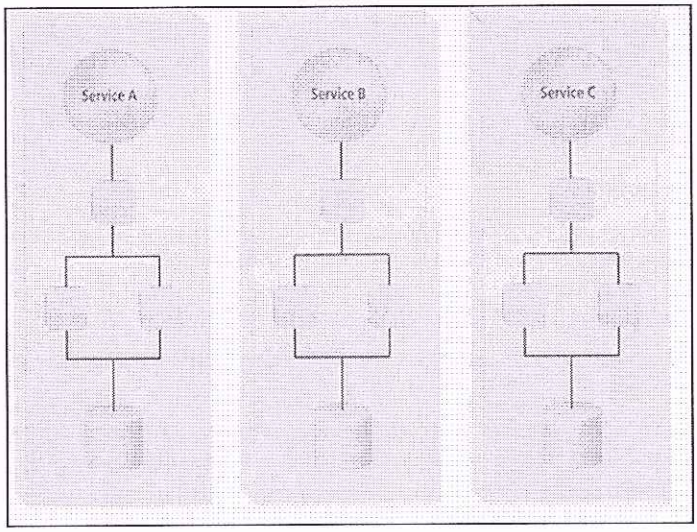


Şekil 2.21. Servislerin Keşfedilebilirliği [26]

2.2.6. Otonomi

Bu kriter, servislerin mümkün olduğunca otonom bir yapıya sahip olması gerektiği esasına dayanmaktadır. Otonom yapıya sahip olmak, kendi başına hareket edebilmesi anlamına gelmektedir, yani kendi kararlarını verebilmek için özgür ve kontrol sahibi olmak ve bunun için dışardan herhangi bir müdahale gerekmemesi demektir. Yazılım

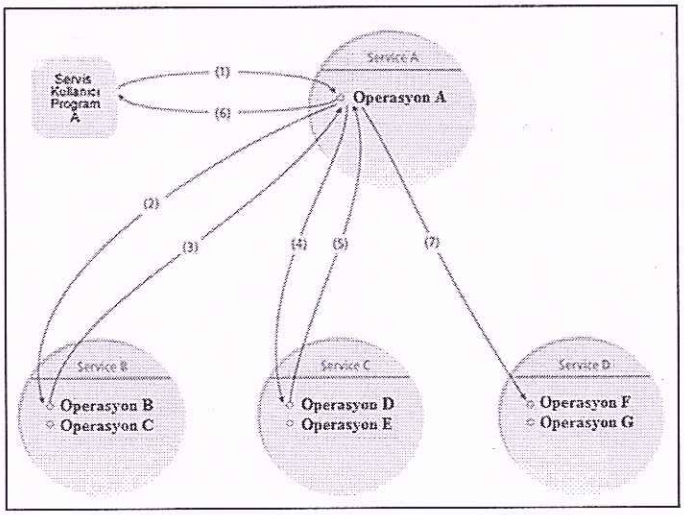
açısından özerklik bir programın fonksiyonlarını yürütebilme özgürlüğü olarak tanımlanır. Burada ise bir servisin platforma veya uygulamaya bağlı olmaması anlamı taşır. Şekil 2.22' deki yapı, servisler açısından oldukça özerk bir yapıdır. Bir servisin daha özerk hale gelmesi neredeyse imkansızdır. Servislerden hiçbiri diğerinin platformuna uymak zorunda değildir çünkü herbiri bağımsız platformlara ve uygulamalara bağlıdır [14],[24].



Şekil 2.22. Kendi Fiziksel Ortamlarına Ait Otonom Servisler [7]

2.2.7. Birleştirilebilirlik

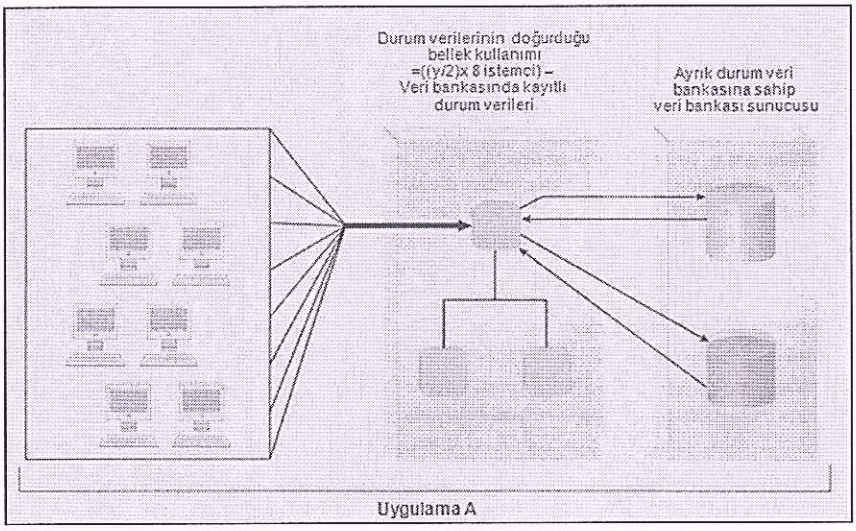
Servisler, iş süreçlerini otomatikleştirmek için birbiriyle iletişime geçerek hareket etmek durumundadırlar. Süreçlerin otomatikleştirilebilmesi için, servislere ait fonksiyonların ihtiyaç duydukları girdi ve çıktı parametlerinin belirlenmesi gerekmektedir. Son olarak, bir işlemi gerçekleştirebilmek için birbirinin yeteneklerine ihtiyaç duyan servislerin, işlem sıralarının belirlenerek, birleştirilmelerinin sağlanması gerekmektedir. Bu kriter Şekil 2.23'de örneklendirilmiştir [9],[25].



Şekil 2.23. Servislerin Birleştirilebilirliği [7]

2.2.8. Durumsuzluk

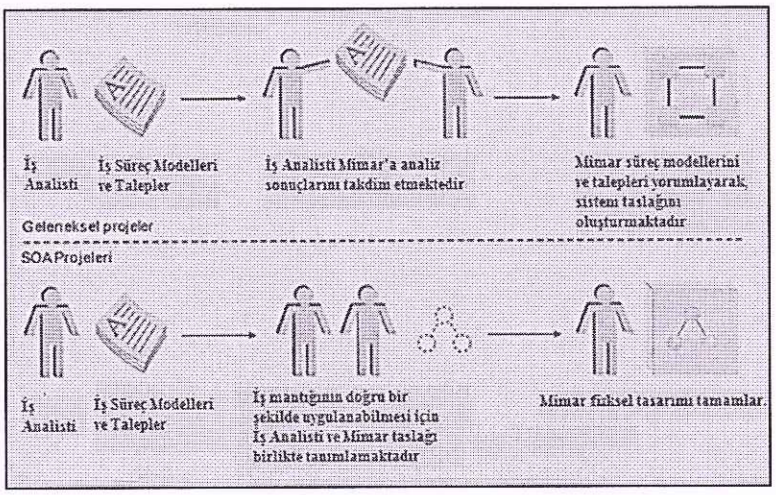
Bir servis, belirli bir işlemin gerçekleştirilmesi için çağırılmaktadır, ardından gereken hizmeti görmekte ve son olarak kendisinin görevi son bulmaktadır. Ancak servis ile ilgili tam da bu aşamada, aktif olup olmadığı veya durumu ile ilgili benzer verilerin kaydedilmesi ve yönetilmesi gerekmektedir. Bunun sebebi ise, iletişimdeki iki servis arasındaki bağın teknik bir sebepten dolayı kopması durumunda, iletişimin tekrar sağlanabilmesi için, servislerin tekrar göreve çağırılması gerekliliği gibi örnekler düşünülebilir. Şekil 2.24'te bu tür verilerin kaydedilme işlemi, veri bankası sunucuna ikinci bir veri bankası eklenerek sağlanmıştır. Aksi takdirde bu verilerin kullanıcı ve sunucu tarafından paylaşılarak kaydedilmesi gerekmektedir. Bu örnekte veri bankası sunucusu, uygulamanın sunucusuna bu yükü bırakmamakta ve bu işlemi gerçekleştirmektedir. Böylece, uygulama içindeki servislerin durum verileriyle ilgilenmemesi sağlanmaktadır [7].



Şekil 2.24. Servislerin Durumsuzluğu [7]

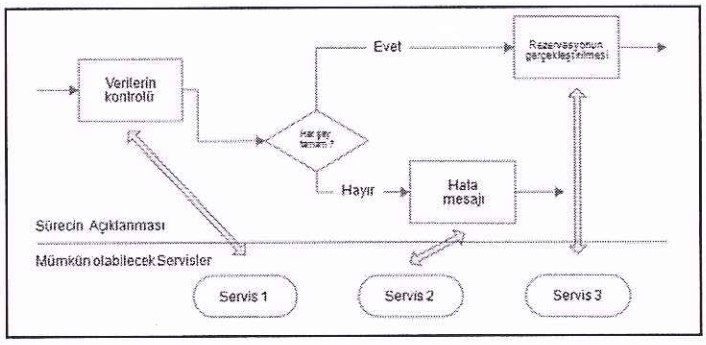
2.3. SOA için İş Süreç Yönetimi

İş Süreç Yönetimi (BPM – Business Process Management) ve Servis Odaklı Mimari'nin birbirini bütünleyen kavramlar olarak görülmeleri gerekmektedir [27]. Şekil 2.25'ten de net olarak anlaşılabileceği gibi, geleneksel yazılım projelerinde, iş analisti tarafından modellenen iş süreçleri ve talepler, yazılım mimarları tarafından yorumlanmakta ve sistemler, gene aynı yazılım mimarları tarafından tasarlanmaktadır. Ancak günümüz SOA projelerinde, iş analisti ve mimarlar, iş mantığını (fonksiyonel olarak) doğru bir şekilde yorumlayabilmek ve oluşturulması gereken servisleri belirleyebilmek amacıyla, çok daha fazla beraber çalışmaktadırlar [28].



Şekil 2.25. Geleneksel Projeler ve SOA Projelerinin Karşılaştırılması [7]

Kaliteli bir iş süreç yönetimi uygulaması, SOA için çok önemli bir ilk adım oluşturabilmektedir [29]. Böyle bir uygulama, servislerin, bölüm 2.2'de anlatılan servis tasarım prensiplerine göre geliştirilebilmesini desteklemekte ve kolaylaştırmaktadır. Böylelikle iş süreçlerinin düzeninden sorumlu olan iş süreç yönetimi, oluşturulacak servisler için çok sağlam bir temel görevi görebilmektedir [32]. Bölüm 4'te açıklanan SOA uygulama metodolojisinin en son adımında, bu temelde geliştirilen servislerin yönetilerek, iş süreçlerinin otomatikleştirilmesi sağlanmaktadır. Şekil 2.26'da basit bir iş süreç modelinden SOA için servislerin nasıl türetilebileceği örnek olarak gösterilmektedir.



Şekil 2.26. İş Süreçleri Üzerinden Servislerin Türetilmesi [32]

İş süreç yönetimi, iş süreçlerinin belgelendirilmesi, yapılandırılması, daha iyi hale getirilmesi ve bilişim teknolojileri ile desteklenmesiyle ilgilenen bir bilim dalıdır [33]. İş süreç yönetimi, organizasyonların pazardaki rekabet gücünü arttırabilmek için, iş süreçlerinin sürekli iyileştirilmesini hedeflemektedir. Bu hedef, bir iş süreç yönetimi döngüsü ile gerçekleştirilmektedir ve bu döngü, strateji, dizayn, geliştirme ve kontrol adımlarından oluşmaktadır [34].

İş süreç yönetimi, SOA'ya dizayn aşamasındaki işlev ve teknikleri ile destek olabilmektedir [27]. Dizayn aşamasında bir kurumun süreçlerinin belirlenmesi, analizi ve belgelendirilmesi gerçekleştirilmektedir. İlk olarak mevcut süreçlerinin tanımlanıp, iş süreç modeli (mevcut iş süreç modeli) olarak belgelendirilmesi ve bu modelin analizinin yürütülmesi gerekmektedir. Analizin sonucu olarak hedef süreçler, bilgisayar temelli simülasyon için, hedef süreç model olarak belgelendirilmektedir [35].

Kaliteli bir SOA uygulaması için, iş süreç yönetiminin dizayn aşamasındaki bu işlemlerin tümünün özverili bir şekilde yürütülmesi gerekmektedir. Bölüm 4'te açıklanan iş süreçlerine SOA uygulama metodolojisinin ikinci adımına geçilebilmesi için, tüm bu işlemlerin en iyi şekilde gerçekleştirilmesi gerekmektedir.

Bununla birlikte uygulamada görülmüştür ki süreçlerin manuel modellenmesi yoluyla süreç bilgisinin çıkarılması zaman alıcı ve maliyeti çok yüksek olabilmektedir. Süreçler hakkındaki bilgi sahipleri, genellikle bu bilgilerini biçimsel ve kurallara uygun olarak tarif edebilecek durumda olamamaktadırlar. Bu durumda da modelleme maliyetini katlayacak uzmanların bilgilerine başvurulmasına gerek duyulmaktadır. Ayrıca belirtilen sebeplerden dolayı modellenen bilginin düzenli bir şekilde güncellenmesi genellikle sağlanamamaktadır. Bu şekilde çok maliyetli olan bir modelleme işleminin modelleme sona erdiğinde, eskidiği daha doğrusu süreçlerin bu arada değişebildiği durumlara da az rastlanmamaktadır. Bu tarz bir modellemede, kaynak bilgi kuruluşun denetiminde olmadığı için model hem kabul sorunlarını hem de ekonomik sorunları beraberinde getirmektedir [36].

Bu sorunun aşılabilmesi için özel bir tekniğe ihtiyaç duyulmuş ve bu da süreç madenciliğinin doğuşuna sebep olmuştur. Süreç Madenciliği (Process Mining) kavramı 90'lı yıllara ve J.E. Cook ve A.L. Wolf isimli yazarlara dayanmaktadır. Bu yazarlar olaylara (event) bağlı verilerden süreç modellerinin türetilmesi ile ilgilenmişlerdir ve çalışmaları da "Process Discovery" (Süreç Keşfi) ismi ile tanınmıştır [37],[38],[39].

Süreç Madenciliği teknolojisi, Veri Madenciliği'nin özel bir şekli olarak düşünülebilir. Veri Madenciliği'nin amacı büyük veri gruplarından (veri) ihtiyaç duyulan bilginin çıkarılması veya ayıklanmasıdır (madencilik). Süreç Madenciliği ise, bilginin çok belirgin bir şeklini hedeflemektedir: Süreç bilgisi [40].

Süreç Madenciliği, olay kayıtlarına (event logs) dayalı iş süreçlerinin tersine mühendisliği (reverse engineering) olarak tanımlanabilir. Olay kayıtları "süreçlere vakıf bilgi sistemleri" tarafından üretilmektedir. Bu olay kayıtları, durum verilerini (case ids), olay adlarını ve türlerini (event names and types), olayların yaratıcılarını ve zaman verilerini içermektedirler. Olay kayıtları bunun dışında müşteri verilerini, ürün çeşitlerini ve diğer karar özniteliklerini de içerebilmektedir. Olay kayıtları genellikle denetim için tutulmaktadırlar [41].

Teknik olarak ise, bir arayüz (interface) aracılığı ile bir uygulamaya (application) ait olay kayıtları, mxml (mining xml) formatına dönüştürülmektedir. Bu hali ile olay kayıtları keşfedilmeye ve analiz edilmeye hazır hale gelmektedir. Böylece aşağıdaki soruları cevaplayabilecek verilerin üretilmesi sağlanmaktadır [41];

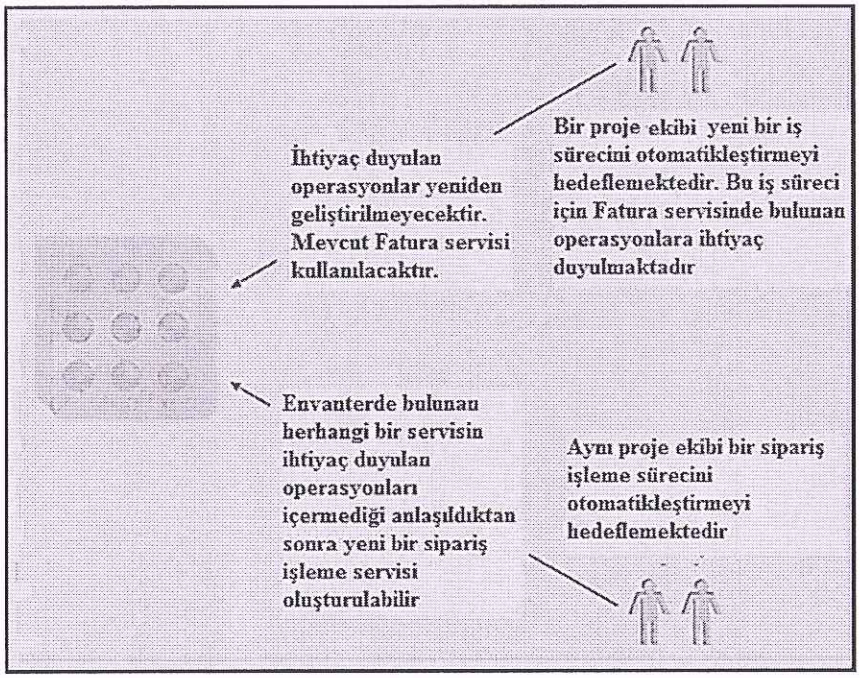
- Her olay ne kadar zaman almaktadır?
- Bekleme zamanları ne kadardır?
- Olayların barındırdığı işlemlerin süreleri ne kadardır?

Süreç Madenciliği, performans ve organizasyonel verileri kullanarak iş süreçlerini, iş akışları şeklinde görsel ve açık bir hale getirmektedir. Barındırdığı uygunluk yeteneği sayesinde hedef iş süreç modeli, uygulamanın son kayıtları ile karşılaştırılmaktadır. Elde edilen uygunluk raporlarında, hedef model ile kayıtlar

arasındaki farklar belirlenebilmektedir. Birden fazla uygulama kullanan (örn. ERP veya CRM) kuruluşlar için de süreç madenciliği (süreç tasarımı) önemli fırsatlar sağlamaktadır. İş süreç yönetiminin esasında, mevcut iş süreçlerinin hedef iş süreçleriyle karşılaştırılabilmesi büyük önem teşkil etmektedir. Süreç madenciliği ise, teknik olarak bu yeteneğe sahiptir [41].

2.4. Servis Odaklılık ve Merkezi Mantığın Sağlanması

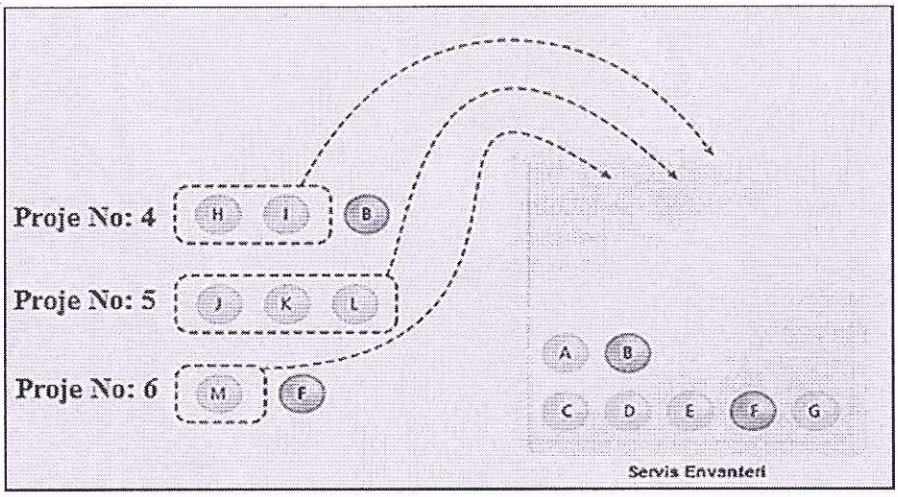
Servis odaklı bir mimarinin ne anlama geldiğinin tam olarak anlaşılabilmesi için, öncelikle servis odaklılığın ne anlama geldiğinin anlaşılması gerekmektedir. Servis odaklılık, merkezi mantığın sağlanmasıyla elde edilmektedir. Servis mantığı ise, servisin temelde hangi görevi gerçekleştirmek için geliştirildiğine dayanmaktadır. Örnek vermek gerekirse, bir faturalama servisi mantığı, kapsam olarak faturalama işlevlerini barındırmaktadır. O halde faturalama servisi, faturalama için gereken fonksiyonları içermelidir ve dolayısıyla faturalama görevini gerçekleştirmelidir (otomatikleştirmelidir). Merkezileştirilen temel bir mantık kendisinin birden fazla yerde kullanılabilmesi imkanını doğurmaktadır. Bu durumu şu şekilde açıklayabiliriz; az önce örnek olarak gösterilen faturalama servisi belli bir amaca hizmet etmek için geliştirileceği için, kendisinin bir organizasyonun sadece belli bir biriminde değil, ihtiyaç duyulan birden fazla biriminde de (örn: satınalma, lojistik, arşiv vs.) kullanılabilmesi sağlanmış olmaktadır. Daha geniş düşünüldüğünde faturalama servisi, başka organizasyonlarda da kullanılabilecek bir hal kazanmaktadır. Bu durum, organizasyonlara servis odaklı sistemler geliştirebilme ve kullanabilme yeteneği kazandırmaktadır [42],[43],[44].



Şekil 2.27. Proje Ekipleri'nin Servis Odaklılık ve Merkezi Mantığa Göre Eylemleri [7]

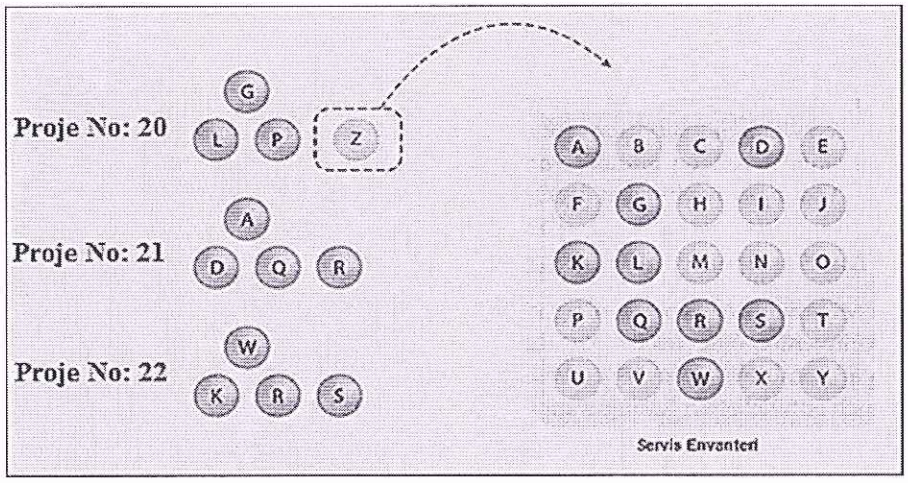
Şekil 2.27, servis odaklı bir yaklaşım izleyen bir organizasyonun hareket şeklini göstermektedir. Böyle bir organizasyona ait yazılım geliştirme proje grubu, servis odaklı bir yazılım mimarisi tasarlarken bir servisin veya servis mantığının envanterde bulunup bulunmama durumuna göre iki ayrı hareket şekli sergilemektedir. Eğer ihtiyaç duyulan servis mantığı envanterde bulunuyorsa, bu mantık yeniden geliştirilmeden kullanılabilmektedir. Envanterde bulunmayan bir mantığa ihtiyaç duyulduğunda ise, bu mantığın geliştirilmesi veya temin edilmesi gerekmektedir [45],[46].

Servis Envanteri bir organizasyon veya organizasyonun belli bir bölümü için geliştirilmiş veya temin edilmiş servislerin toplamını simgeleyen bağımsız bir standarttır. Organizasyonun ihtiyaç duyduğu süreçler, otomatikleştirilmeye başlandığında servis envanteri gelişmeye başlamaktadır. Artan projeler yeni servis ihtiyaçlarını doğuracağı için, servis envanteri, sürekli olarak gelişmeye devam etmektedir [9].



Şekil 2.28. Tamamlanmamış Servis Envanteri [7]

Şekil 2.28'den de görülebileceği gibi servis odaklı bir yapıya geçildiği zaman, yeni projeler, yeni servis ihtiyacını doğurduğundan, ihtiyaç duyulan servisler de geliştirildikçe veya temin edildikçe kurumlarda bulunan servis envanteri genişlemektedir. Proje için gereken servis, envanterde hâlihazırda mevcutsa bu servis yeniden geliştirilmemektedir. Burada önemli bir kavram olan servis mantığı ortaya çıkmaktadır. Servis mantığı, eğer ihtiyaç duyulan mantığı gerçekleştiriyorsa, yani programlama açısından bakıldığında ihtiyaç duyulan fonksiyon veya metod envanterdeki bir serviste varsa, bu servis direk olarak kullanılabilmektedir [14].



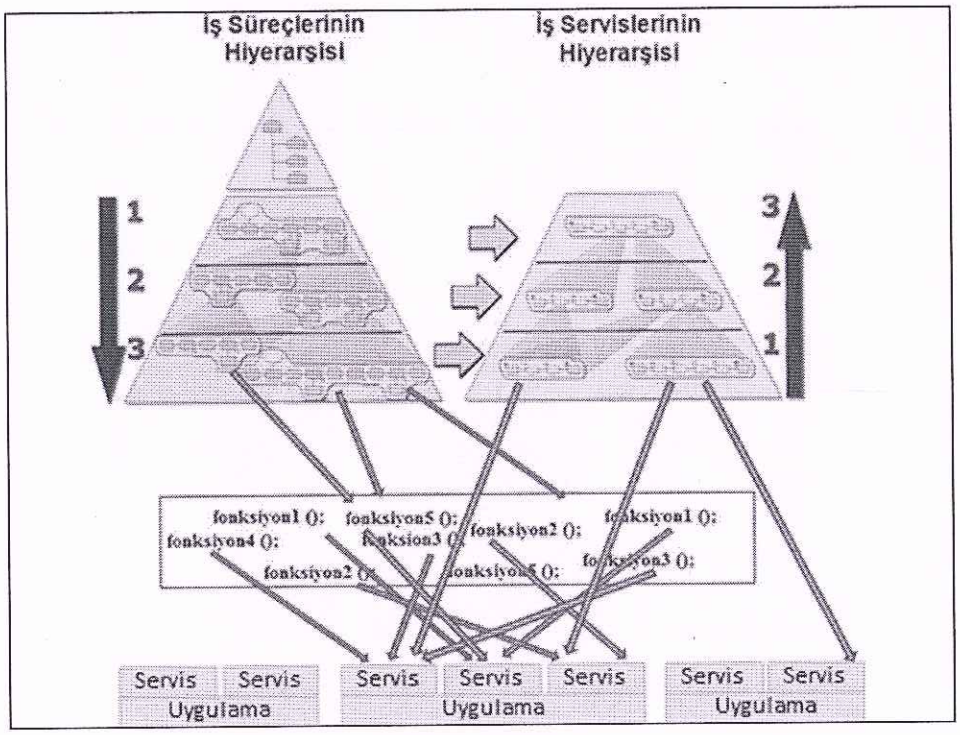
Şekil 2.29. Tamamlanmış Servis Envanteri [7]

Şekil 2.29'da gelişmiş bir servis envanteri görülmektedir. Buradan görülebileceği gibi, artan proje sayısının getirdiği servisler, zaman ilerledikçe organizasyonun yeni servis geliştirme ihtiyacını düşürmektedir ve var olan servisler organizasyon için yeterli olmaya başlamaktadır. Dolayısıyla, mimari geliştirilirken belirlenen servisler, bu kurala da uyacak şekilde, yeniden kullanılabilirliği sağlayacak şekilde yapılandırılmalıdır [9].

Bir SOA için servis odaklılığın ve merkezi mantığın sağlanmasının önemi açıklandıktan sonra şimdi de bu konuların temelinde, bir SOA'nın nasıl kazanılabileceği (uygulanabileceği veya geliştirilebileceği ifadeleri de kullanılabilir) üzerinde durulacaktır. Bu konuda çeşitli stratejiler geliştirilmiştir. Bu stratejilerden literatürde en geniş yer bulan üç tanesi şunlardır: yukarıdan aşağıya uygulama stratejisi (top-down implementation), aşağıdan yukarıya uygulama stratejisi (bottom-up implementation) ve ortada buluşma uygulama stratejisidir (meet in the middle implementation). Şimdi sırası ile bu stratejiler açıklanacaktır.

2.4.1. Yukarıdan aşağıya uygulama stratejisi

Yukarıdan aşağıya uygulama esnasında, uygulama yapılacak kurumdaki iş süreçlerinin analizi yapılmakta ve ardından bu süreçler yukarıdan aşağıya doğru detaylandırılarak mantıksal olarak tutarlı modüller (kavramsal servisler) olarak türetilmektedir [47]. Böylece servisler ve iş süreçleri arasında oldukça iyi bir uyum elde edilmektedir [1].

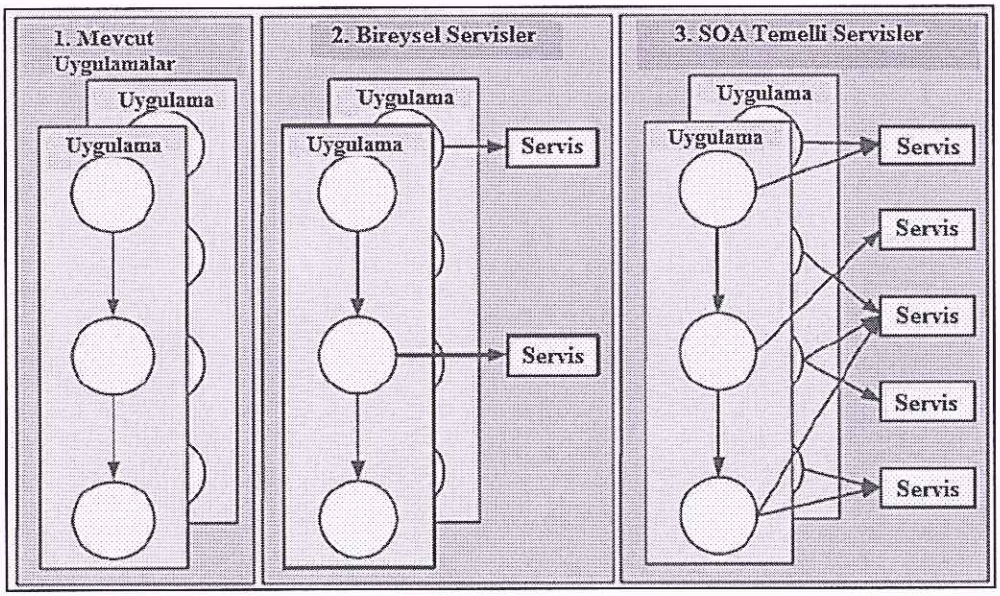


Şekil 2.30. Yukarıdan Aşağıya Uygulama Stratejisi [48]

Şekil 2.30 yukarıdan aşağıya uygulama stratejisinin nasıl gerçekleştirilebileceğini göstermektedir. Şeklin sol tarafında, iş süreçleri hiyerarşik olarak detaylandırılmaktadır. Detaylandırma işlemi mümkün olan son noktaya kadar yapılmaktadır. Daha sonra detaylandırılan süreçlerin mantıksal olarak tutarlı modüller haline getirilebilmeleri için, öncelikle bu süreçlerin barındırdığı fonksiyonların ortaya çıkarılması gerekmektedir. Kaynak No: [48]'de yukarıdaki hiyerarşik yapı, arada fonksiyonlar tabakası olmadan ele alınmış, ancak SOA' da merkezi mantığın sağlanması açısından araya bir katman daha ekleme gereği duyulmuştur. Böylelikle aynı mantığa hizmet edecek fonksiyonların doğru ve tek bir serviste toplanabilmesi sağlanırken, aynı işleve sahip fonksiyonların da kod tekrarı oluşturması engellenmektedir. Sonuç olarak, elde edilen (kavramsal) servisler, şekildeki gibi sol tarafa simetrik olarak detaylandırılan süreçlere uygun olarak dizilerek, süreçlerin kavramsal olarak otomatikleştirilmesi sağlanmaktadır. Tüm bu aşamalardan sonra, elde edilen kavramsal servislerin geliştirilmesi işlemine başlanabilmektedir.

2.4.2. Aşağıdan yukarıya uygulama stratejisi

Aşağıdan yukarıya uygulama stratejisinde bir kurumun uygulama portföyünden (application portfolio) başlanmaktadır. Burada bulunan fonksiyonlar paketleme teknolojileri (wrapping-technologies) aracılığıyla servislere dönüştürülürerek kullanıma sunulmaktadır. Prensip olarak mevcut sistemler daha kapsamlı bir servis tabakası ile genişletilmektedir. Sonrasında adım adım daha fazla tekrar kullanılabilir servisler hazırlanmaktadır. Bu servisler standart servis arayüzleri olarak uygulamaların fonksiyonlarından oluşmaktadır. Bununla birlikte, servisler ancak bir sonraki adımda tekil iş süreçlerini otomatikleştirebilecek hale gelirler ve daha sonra tüm iş süreçleri için gruplanabilmektedirler [49]. Şekil 2.31 bu yöntemi şematik olarak göstermektedir.



Şekil 2.31. Aşağıdan Yukarıya Uygulama Stratejisi [49]

2.4.3. Yukarıdan aşağıya ve aşağıdan yukarıya uygulama stratejilerinin analizi

Literatürde yukarıdan aşağıya uygulama stratejisinin faydaları hakkında bilgiler bulunabilmektedir. Yukarıdan aşağıya yaklaşımı bir akış analizi gerektirmektedir. Bu aşamada her bir servisin dizayn ve parametreleri ayrıntılı olarak analiz edilecektir. Böylelikle servislerin tekrar kullanım potansiyeli maksimize edilmiş olmakla birlikte iş süreçlerine göre yönetilebilmeleri desteklenmiş olacaktır. Bu sebepten dolayı Thomas Erl yukarıdan aşağıya yaklaşımı yüksek kalitede bir servis mimarisi olarak görmektedir [4]. Bu yöntem mantıklı bir uygulamayı ve önemli iş fonksiyonlarının da servis formunda temsilini garantilemektedir [47],[49].

Ancak Liebhart, yukarıdan aşağıya uygulama stratejisinin yüksek yatırımlara bağlı olmasını dezavantaj olarak görmektedir. Örneğin ilk servisin tasarlanması ve oluşturulabilmesinden önce kapsamlı bir envanter analizi gerçekleştirilmek zorundadır [49]. Ayrıca bu strateji prensip olarak ancak eğer kurumda halihazırda kullanılan bir enformasyon teknolojileri sistemi bulunmuyorsa uygulanabilir. Ancak gerçek hayatta çoğunlukla bu durum geçerli olmamaktadır. Kurumlar, makul bir çaba ile değiştirilemeyecek ve bir servis odaklı mimariye entegre edilmek zorunda olan çok sayıda sistem barındırmaktadırlar [47].

Buna karşılık, aşağıdan yukarıya SOA uygulama stratejisinin yararı ise mevcut sistemlerin kullanımının devam etmesidir. Bu da uygulamanın daha düşük maliyetler ile gerçekleştirilebileceği anlamına gelmektedir. Ayrıca yukarıdan aşağıya uygulama stratejisine oranla servislerin daha kısa bir zamanda kullanıma hazır olacağını ve SOA uygulamanın ilk yararlarının daha erken elde edileceğini de söylemek mümkündür [9].

Aşağıdan yukarıya uygulama stratejisinin dezavantajı ise servislerin yapılandırılmasıyla ilgilidir. Zira bu servisler, süreçlere ait yetersiz miktarda fonksiyonlara sahip olmaktadırlar [49]. Bu da ayrıca ek bir yük getirmektedir, çünkü "aşağıdan yukarıya yaklaşımı" ile elde edilen servisler genel olarak daha fazla bakım gerektirmektedirler. Bunun dışında "yukarıdan aşağıya yaklaşımı" ile elde edilen servislere oranla ömürleri daha kısadır.

Höss ve Weisbecker'e göre yukarıdan aşağıya yaklaşımı, aşağıdan yukarıya yaklaşımı ile kombine edilmelidir. Bu uygulama stratejisi ise 'Ortada Buluşma Uygulama Stratejisi' olarak adlandırılmaktadır ve birçok yazar tarafından favori olarak gösterilmektedir, zira bu kombine strateji her iki uygulama stratejisinin avantajlarını kullanmaya ve dezantajlarını uzaklaştırmaya imkan sağlamaktadır [4],[47],[50],[51],[52].

2.4.4. Ortada buluşma uygulama stratejisi

Ortada buluşma stratejisinin uygulanmasına dair bir yöntem şu şekilde olabilir; mevcut uygulamaların hesaba katılarak, servis adaylarının iş süreçlerinden türetilmesi. Bu vesile ile elde edilen servisler iki farklı servis türüne ayrılacaktır. Aşağıdan yukarıya uygulama stratejisi aracılığı ile oluşan servisler ince taneli (az fonksiyonlu anlamına gelmektedir, ing: fine-grained) dijital servisler haline getirilecekler, buna karşılık yukarıdan aşağıya uygulama stratejisi aracılığı ile türetilen servisler ise bu teknik servisleri kullanan iş servisleri olarak düzenleneceklerdir [49].

Bölüm 4'teki metodoloji ortada buluşma yaklaşımına dayanmaktadır ancak çok sayıda ön çalışma ve analiz sayesinde maliyet ve iş yükünü önemli bir ölçüde azaltmayı hedeflemektedir. Örneğin aşağıdan yukarıya uygulama stratejisi aracılığı ile mevcut sistemin (eski sistem) tüm fonksiyonlarını servisler haline getirmek yerine, analizler sonucunda sadece yeni geliştirilen servis odaklı mimaride kullanılabilecek fonksiyonlarını servisler haline getirme yoluna gidilmiştir. Bununla birlikte metodoloji sadece bir kurumun kendisine uygulayabileceği değil aynı zamanda başkalarına da uygulayabileceği bir özelliğe sahiptir ve örnek olarak bir kurumun eski sistemini hiç bilmeyen kişilerin de kendisini kullanılabilmesi için nasıl hareket etmesi gerektiği yönünde öneriler ve teknikler sunmaktadır.

2.5. SOA için Eski Sistemler ve Entegraston Yaklaşımları

Eski sistemler, organizasyonların iş süreçlerini otomatikleştirmek için yeni bir sistem uygulamaya başlayana kadar uyguladıkları sistemler olarak tanımlanabilirler. Kabul etmek gerekir ki IT'nin bu denli hızlı ilerleyişi sistemlerin o kadar hızlı bir şekilde eski kalmasına sebep olmaktadır. Sistemlerin hızlı değişen Pazar koşullarına ayak uydurabilmesi için, işlevinin değiştirilmesi, hızlandırılması, güvenlik koşullarının ve kullanım kolaylıklarının arttırılması gerekmektedir. Ancak her yeni sistem yüklü bir maliyet oluşturmakta ve kendisinden sonra gelecek sisteme kadar yaşayabilmektedir. Organizasyonlar ise büyük yatırım yaptıkları sistemlerini yeni bir sisteme geçerken heba etmek istememektedirler veya en azından kullanılabilecek bölümlerini yeni sisteme entegre ederek kullanmayı hedeflemektedirler [29],[30].

2.5.1. Eski sistem analizi

Yeni sistemini SOA yaklaşımı kapsamında hedefleyen bir kurumun, eski sisteminin kullanılmaya değer kısımlarını kullanabilmesi için bir dizi işlem gerçekleştirmesi gerekmektedir. Bunlardan ilki, eski sistem fonksiyon haritasının çıkarılmasıdır. Bununla kastedilen eski sistemin gerçekleştirdiği fonksiyonların, yani otomatikleştirdiği süreçlerin belirlenmesidir. Bu işlem için iki yol izlenebilir, bunlardan ilki eğer belgelenmiş bir sistem mevcut ise belgelemenin notasyon tekniğine göre inceleyerek belirleme ve eğer mevcut değil ise otomatikleştirilen süreçleri inceleyerek belirlemedir. Bir sonraki işlem ise, çıkarılan fonksiyon haritasına göre belirlenen fonksiyonların, hedeflenen süreçler için gereksinim duyulan fonksiyonların hangilerini karşılayabileceğinin tespitidir. Bu işlemin yürütülme şekli bölüm 4.3'te daha detaylı olarak açıklanmıştır [29],[31].

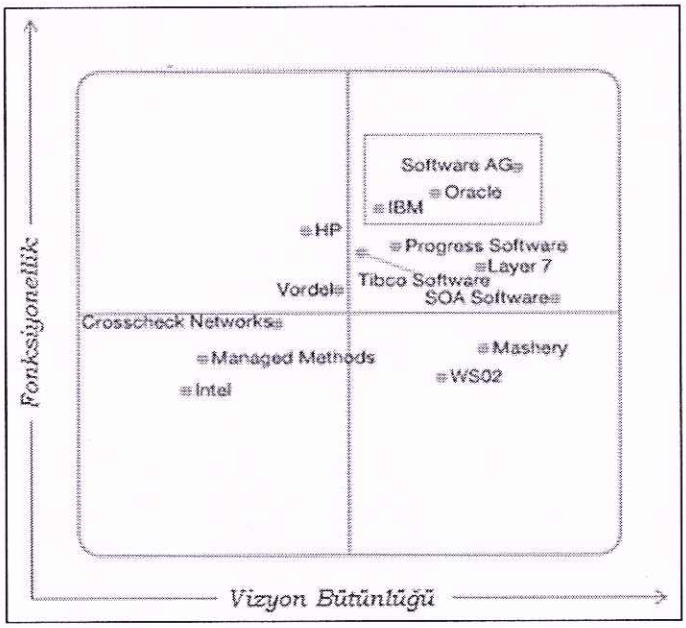
2.5.2. Entegrasyon yaklaşımları

Yapılan araştırmalara göre eski sistemlerin SOA ile entegrasyonu konusunda, yeniden geliştirme (redevelopment), paketleme (wrapping) ve göç ettirme (migration) olmak üzere üç ayrı yaklaşım geliştirilmiştir. Yeniden geliştirme tüm SOA temelli sistemi en başından ayrıntılı olarak geliştirmektir. Yeniden geliştirme yaklaşımı söz konusu olduğunda, birçok kurum iş süreçlerinin eski uygulamalarla

ilerleyebildiği sürece, eski sisteminin tamamının yeni geliştirilmiş bir sistem ile yer değiştirmesinin riskini alamamaktadır. Bu sistemlerin yer değiştirmesi riski, kurumun rutin işleyişinde sorunlara sebep olabilmektedir. Bununla birlikte, bir sistemin geliştirilmesi uzun bir zaman dilimi ve kaynak gerektirmektedir ki bunlar kuruma çok ağır yük olabilmektedir. Alternatif bir yaklaşım olarak ta eski sistemdeki bir bileşenin paketlenmesi yolu ile bu bileşen yeni ve daha erişilebilir bir hale getirilebilir. Bir diğer yaklaşım olan göç ettirmeye göre, eski sistem bileşenlerinin yeni bileşenler ile yerlerinin değiştirilmesi aracılığıyla, orijinal veriler ve fonksiyonellik sabit tutularak, ortamın daha esnek bir hale getirilmesi sağlanmaktadır [53]. Diğer bir çalışmada, bir SOA bileşeni olarak, eski sistemlerin yeniden kullanılabilirliğinin potansiyeli araştırılmıştır. Buna göre eski sistemlerin servis olarak fonksiyonelliği ortaya çıkarılabilir. Ancak bunun için mevcut sistemin SOA'ya nasıl dönüştürülebileceğinin bir analizi gerekmektedir. Bu analiz SOA'nın talep ettiği özel etkileşimleri ve eski sistem bileşenleri üzerinde yapılması gereken önemli değişiklileri içermelidir [54],[55].

Bisbal ve arkadaşlarının araştırmalarına göre, göç ettirme yaklaşımını gerçekleştirme çabaları her zaman başarılı olamamaktadır çünkü eski sistemler, yavaş, bakımı pahalı ve zor olan eski donanımlar üzerinde çalışmaktadırlar. Diğer bir zorluk ise eski yazılımların bakımı, belgeleme eksikliği ve yazılımın anlaşılabilirliği yüzünden çok pahalı olabilmektedir. Eski uygulamalardaki arayüzlerin net olmamasından dolayı da, eski sistemlerin diğer sistemlerle entegrasyonu zor olabilmektedir. Eski bir sistemin geliştirilmesi ve değiştirilmesi büyük zorluklar taşımaktadır. Bu yüzden, eski sitemlerin, göç ettirme süreçlerine olumsuz yönde etki gösterebilecek ve daha önceden söz edilen zorlukların üstesinden gelmeye yardımcı olabilecek koşullar, önceden sağlanmalıdır [53],[55].

2.6. SOA Yönetim Platformları



Şekil 2.32. Gartner'in Araştırmasına Göre Lider SOA Yönetim Platformları [56]

SOA Yönetim Platformları, lider yazılım firmaları tarafından SOA yaklaşımının öngördüğü faaliyetlerin gerçekleştirilmesini kolaylaştırmak için geliştirilmiş olan yazılımlardır. SOA'nın talep ettiği servis geliştirme kriterleri bu tarz platformların yoksunluğunda oldukça zor bir şekilde gerçekleştirilebilmektedir. Şekil 2.32'de görüldüğü gibi, birçok lider yazılım firması farklı isimler altında ve farklı özelliklere sahip SOA yönetim platformları geliştirmiştir. Ancak Oracle, Software AG ve IBM firmalarının fonksiyonellik ve vizyon bütünlüğü açısından incelendiğinde diğer firmalara göre başrolde yer aldıkları görülebilmektedir [57].

Tablo 2.2. Lider SOA Yönetim Platformları'nın Güçlü ve Zayıf Yanları [57]

		Yazılım Firması ve Ürün		
		Oracle	Software AG	IBM
		SOA Suite	Crossvision	SOA Foundation
Ürüne Özel Kriterler	Performans	8	9	7
	Yönetim ve Konfigürasyon İmkanı	8	Bilgi Yok	6
	Standartların Desteklenmesi	7	9	6
	Güvenlik	8	8	8
	Interoperabilite	8	9	8
	Hata Yakalama	8	8	8
Üründen Bağımsız Kriterler	İşletim Sistemlerinin Desteği	9	8	8
	Maliyet	8	6	5
	Belgelendirme	8	7	6
	Kullanıcı Odaklılık	subjektif	subjektif	subjektif
	Erişilebilirlik	8	5	5
	Yazılım ve Donanım Talebi	7	7	7
	TOPLAM	79	76	68

[57] numaralı kaynakta Oracle, Software AG ve IBM firmalarının SOA yönetim platformları, ürüne özel ve üründen bağımsız kriterler olmak üzere iki ana kriter altında karşılaştırılmıştır. Her bir kriter, karşılanma derecesi olarak en düşükten en büyüğe doğru ve bir ile on arasında değişen bir puanlama tablosu eşliğinde puanlanmıştır. Kriterler arasından yönetim ve konfigürasyon imkanı ve kullanıcı odaklılık tam ve objektif bilgiler içermedikleri için puanlanmaya dahil edilmemişlerdir. Sonuç olarak bu kriterler ışığında Orecle'ın en fonksiyonel SOA yönetim platformu olduğuna kanaat getirilmiştir [57].

BÖLÜM 3. PROBLEM TANIMI VE LİTERATÜR TARAMASI

3.1. Problem Tanımı

Bilgi teknolojisi kullanımı, hemen hemen tüm sektörlerdeki işletmeler için vazgeçilmez hale gelmiştir. Bununla birlikte kurumsal uygulamalar, iş süreçlerinin yürütülmesine destek vermekte ve hatta bunları kısmen otomatikleştirmektedirler [58]. Birçok alanda olduğu gibi özellikle yazılım dünyasında da bir problemi birçok yöntemle çözmek mümkündür. Aynı işi yapan bir yazılım birçok farklı şekilde tasarlanarak ortaya çıkarılabilmektedir. Yazılım tasarımında kesin çizgilerle en iyi tasarım ya da doğru tasarım olarak bir tasarımı nitelendirmek zor olsa da genel olarak daha iyi tasarımın taşıdığı özellikler bilinmektedir. Kötü bir tasarımın belirtilerini ise kısaca aşağıdaki gibi sıralayabiliriz [59],[60].

- Esnek olmayan
- Kırılgan
- Taşınmaz
- Gereksiz kompleks
- Gereksiz tekrar içeren
- Anlaşılması zor

Kötü tasarım belirtilerinin açıklamaları aşağıdaki gibidir;

Esnek olmayan: Kısaca değişim maliyetinin yüksek olmasıdır. Geliştirilen yazılımın herhangi biryerinde değişiklik yapılmak istendiğinde, kodun birçok yerinde değişiklik yapmak gerekiyorsa bu yazılımın esnek olmadığını göstermektedir [59],[60]. Bu konu bölüm 2.1.3'te ayrıntılı bir uygulama örneği ile tanıtılmıştır.

Kırılgan: Yazılımda yapılması gereken en ufak bir değişiklikte çok sayıda bug (sistem hatası) ile karşılaşılması geliştirilen yazılımın kırılgan yapıda olduğunu göstermektedir. Kırılganlık yazılımcıların en kötü kabusu olduğu gibi, yazılımcı değişiklik yapmaktan iyice korkar hale gelebilmekte ve kırılgan olan yazılım müdahale edilmedikçe dahada kötü hal alabilmektedir [59],[60].

Taşınmaz: Yeniden kullanılamayan koda, modüle, yazılıma kısaca taşınmaz denmektedir. Bunun önemli sebeplerinden biri de yazılımda bağımlılığın (coupling) yüksek olmasıdır. Örneğin bir proje için yazılan ve belirli bir işi yapan bir sınıfın, başka bir projede işe yarayacağı için o projeye taşınmak istenebileceği düşünülebilir. Kopyalanan sınıf diğer projenin kaynak koduna atıldığında, diğer projede hatalar çıkmaya başlamaktadır. Hatalar dikkatlice incelediğinde kopyalanan sınıfın içindeki sınıf değişkenlerinin diğer projede bulunamadığı görülmektedir. Bulunamayan sınıfların da alınıp projeye dahil edildiğinde ise, bu sınıfların içindeki diğer değişken olarak tanımlanmış sınıfların bulunamadığı görülebilmektedir. Kısaca ufak bir sınıfın veya kodun başka bir proje veya modülde kullanılmak istenmesi durumunda, zincirleme olarak birçok sınıfın da diğer kısma taşınması gerekiyorsa, bu durum yazılımın hantallaşmış ve taşınmaz olduğunu belirtmektedir [59],[60].

Gereksiz kompleks: Bir yazılımın gelecekte de kullanılabilmesi amacıyla birçok gereksiz özelliği barındırması onu aşırı ve gereksiz kompleks hale getirmektedir. Örnek olarak, gerekmediği halde daha esnek olabilmesi ve ileride başka bir kod eklenmesi durumunda mevcut kodun hiç değiştirilmemesi ve çok hızlı çalışması amaçlı heryerde Design Pattern kullanılması, anlaşılması ve yönetilmesi zor, aşırı ve gereksiz bir yazılım ortaya çıkarabilmektedir [59],[60].

Gereksiz tekrar içeren: Kod tekrarı, aynı işi yapan sınıfların tekrarı veya metodların tekrarı olarak tanımlanabilmektedir. Bu durum aynı zamanda kırılgan ve esnek olmayan bir yapıya yol açabilmektedir. Kodun bir kısmında değişiklik yapılması gerektiğinde aynı kod birden fazla yerde tekrarlandığı için, birden çok yerde değişikliğe gitmek zorunda kalınmaktadır. Bununla birlikte hata çıkma oranı ve maliyet aynı oranda artabilmektedir [59],[60].

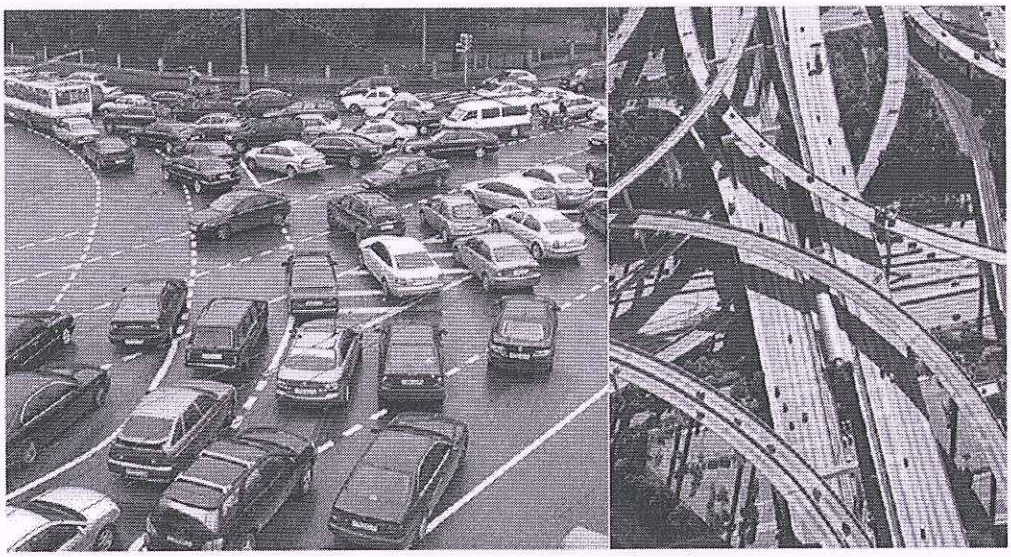
Anlaşılması zor: Geliştirilen bir kodun, bir birey tarafından okunduğunda işlevi anlaşılamıyorsa bu kodun okunabilirliğinin az olduğu anlamına gelmektedir. Kısaca yazılım geliştirirken amaç bir makinenin anlayabileceği kodu yazmak yerine insanların anlayabileceği kodu yazmak olmalıdır [59],[60].

Yazılım dünyasında yukarıda anlatılan kötü tasarım belirtileri ile sıklıkla karşılaşabildiğimiz günümüzde, katmanlı ve dağıtık yazılım sistemleri olarak karakterize edilebilen yüksek kaliteli kurumsal uygulamaların geliştirilmesi ve entegrasyonu büyük önem teşkil etmektedir. Son yıllarda servis odaklı mimari kavramı, kurumsal uygulamaların geliştirilmesi ve entegrasyonunda, bu kötü tasarım belirtilerinden uzak durulabilmesi anlamında önemli bir mimari tarzı haline gelmiştir. Kullanıcı açısından bakıldığında SOA, yazılım olarak gerçekleştirilen servislerde işe yönelimi (iş süreçleri odaklı olması) vurgulamaktadır. Mimari açıdan ele alındığında ise modülerlik, gevşek bağlantı (örn. Platform, yer, protokol, ve format bağımsızlığı), aşamalı yapı ve akış bağımsızlığı önemli SOA prensiplerindendir [58].

Bir servis odaklı mimarinin, uygulanmaya geçilmesi aşamasında, servislerin tasarımı önemli bir görev teşkil etmektedir [32],[61]. Tasarım aşamasında, ilk önce servislerin kimliklendirilmesi (kavramsallaştırılması) gerekmektedir ve sonra hangi servislere hangi fonksiyonların (metotlar veya yetenekler) atanacağı yönündeki ilk önemli karar aşaması söz konusu olmaktadır. Bu noktada, ihtiyaç duyulan servisler, servis adayı olarak adlandırılmaktadır. Daha sonra servis adaylarının her birinin servis tasarımı formunda ayrıntılı özelliklendirilmesi gerekmektedir. Böyle bir servis tasarımı, ayrıca servisi tanımlayan servis arayüzünü, geliştirilecek olan servis bileşenini ve gerekirse de servislerin işlevselliklerini gerçekleştirebilecekleri (teknik) arayüzlerini ifade eder [4],[62],[63]. Servis tasarımı, servislerin geliştirilebilmesi için önemli bir temel teşkil etmektedir. Bununla birlikte servis tasarımının, bir Servis Odaklı Mimarinin yapısı ve ayrıca da servislerin kalite özellikleri üzerinde önemli etkileri bulunmaktadır. Bir Servis Odaklı Mimarinin bilişim teknolojileri ile ilgili olan hedeflerinin ne ölçüde tutturulabileceği de servis tasarımının potansiyeline bağlıdır. Bundan dolayı servisin gevşek bağlantı ya da özerklik gibi kalite özelliklerinin en iyi şekilde sağlanabilmesi için, servis tasarım

aşamasının, servis adayının kimliklendirilmesi ve akabinde özelliklendirilmesi işlemleri büyük bir özenle gerçekleştirilmelidir [64].
Bilişim Teknolojileri Mimarlarını servis tasarımında destekleyebilmek için anlaşılabilir ve kabul görür bir yönteme ihtiyaç duyulmaktadır. Bu yöntem sistematik olarak iş analizi sonuçlarından ve doğal olarak bunların temelinde yatan iş süreçlerinden servis taslakları oluşturabilme imkanı vermelidir. Bunlar ayrıca öyle tasarlanmalıdır ki, servise dönüştüklerinde seçilen kalite özellikleri ile birlikte sonuç alınabilmelidir [64].

Günümüz literatüründeki kaynaklarda, SOA yaklaşımı temelinde servis geliştirme kriterleri belirlenmiş, ancak bu kriterlerin nasıl sağlanabileceğine (pratikte nasıl uygulanabileceği) dair standardize edilmiş bir metodoloji geliştirilmemiştir. Bununla birlikte SOA proje tecrübelerine dayanan birçok kaynakta belirtilmiştir ki, belirli bir metodoloji temelinde uygulanmayan bir SOA, birçok sorunu da beraberinde getirebilmektedir. Doğru uygulanan bir SOA, yazılım dünyasında karşılaşılabilecek birçok sorunu ortadan kaldırabilmekle birlikte, SOA'nın asıl amacının yakalanamayarak, belirli bir metodolojiden yoksun gerçekleştirilebilecek bir uygulaması da istenileni vermekten çok problem halini alabilmektedir [5],[8]. Bu durum Şekil 3.1'deki resimde bulunan arabaların servisleri ifade ettiği düşünülerek canlandırılmaya çalışılmıştır. Şeklin sağ tarafındaki yapının, belirli bir planlamanın uygulanması ve çok daha zahmetli süreçlerden geçilerek oluşturulabileceği açıkça görülebilmekte, ayrıca bu yapıdaki servislerin sol taraftaki kaosu yaratabilmesi ise nerdeyse imkansız hale getirilebilmektedir. Bu tez kapsamında geliştirilen ve bölüm 4'te tanıtılan 'İş Süreçlerine Servis Odaklı Mimari Uygulama Metodolojisi', ifade edilen tüm bu sorunların minimize edilebilmesini amaçlamaktadır.



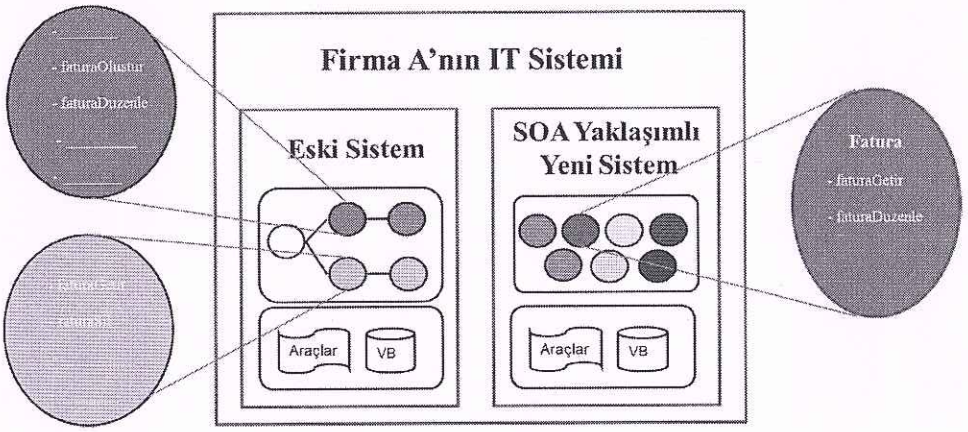
Şekil 3.1. İki Farklı SOA Yaklaşımı'nın Karşılaştırılması [65]

3.2. Karşılaşılabilecek Problemlere Dair Örnekler

Eski Sistem (Legacy System) Üzerinden Doğabilecek Problemler:

Eski sistemler, kısaca kurumların teknolojik veya mimari açıdan daha yeni bir sistem kullanmaya başlayana kadar kullandıkları sistemlere verilen isimdir. Eski sistemler, kendilerinden daha yeni bir sistem uygulamaya çalışan firmaların IT yöneticileri için genellikle problem oluşturabilmektedir. Özellikle SOA tarzı bir düzen sistemine geçiş yapmak isteyen kurumların yazılım mimarları, kullanılan ve kurum yöneticileri tarafından terk edilmek istenmeyen (SOA'dan yoksun) eski sistemlerin oluşturduğu problemler ile sıklıkla karşılaşabilmektedir. Bölüm 2.5'te eski sistemlerin analizleri ve kullanılabilmeleri ile ilgili genel bilgiler verilmiştir. Aşağıda bu konu ile ilgili karşılaşılabilecek iki ayrı problem üzerinde durulmuştur.

Problem 1: Eski sistemde mevcut olan fonksiyonların keşfedilemeyip yeniden geliştirilmesi.



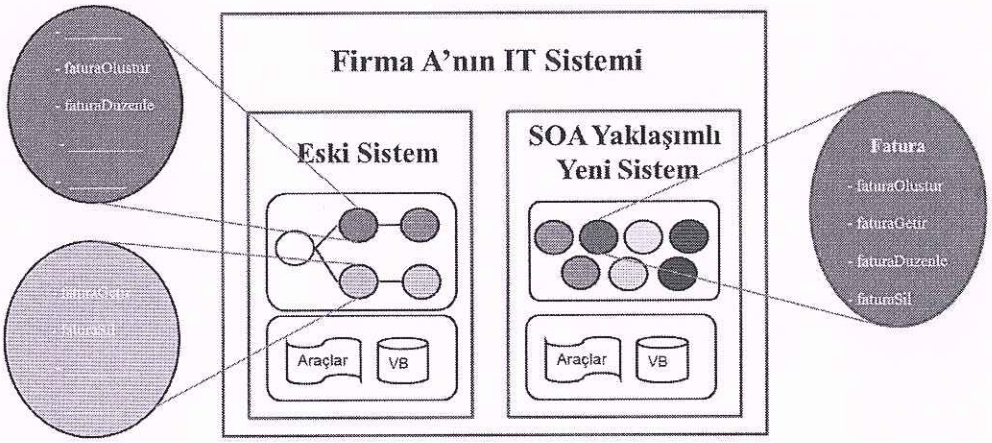
Şekil 3.2. Eski Sistem Üzerinden Doğabilecek Kod Tekrarı Problemi'nin Canlandırılması

Şekil 3.2'de Firma A'nın, SOA yaklaşımından yoksun bir eski sistemden ve SOA yaklaşımlı yeni sistemden oluşan IT sistemi canlandırılmaktadır. Şekil üzerinde de görüldüğü gibi eski sistemde var olan belirli fatura fonksiyonlarının fark edilemediği ve bu sebeple bunlara denk fonksiyonların yeni sistemde fatura servisi fonksiyonları olarak yeniden geliştirilmek zorunda kalındığı görülmektedir. Sonuçta problem olarak Firma A'nın IT sistemi üzerinde kod tekrarı yapılmış olup, Firma A belirli bir maddi zarara uğramıştır. Problemin sebebi ise eski sistemin yeterli derecede ve doğru tekniklerle incelenmemiş olmasıdır.

Problem 2: İhtiyaç duyulan fonksiyonların eski sistemde bulunup kullanılabilecek durumda olmalarına rağmen, birer SOA temelli servis fonksiyonu olarak geliştirilmesinin daha verimli olabilmesi.

Şekil 3.3'te Firma A'nın, SOA yaklaşımından yoksun bir eski sistemden ve SOA yaklaşımlı yeni sistemden oluşan IT sistemi canlandırılmaktadır. Bir önceki örneğin aksine, bu kez firmanın eski sisteminde var olan belirli fatura fonksiyonlarının fark edilebildiği ancak buna rağmen bu fonksiyonların yazılım mimarları tarafından birer SOA temelli servis fonksiyonu olarak geliştirilmesinin daha verimli olabileceği

düşünülmektedir. Zira Fatura fonksiyonları, çok kullanılan fonksiyonlardır ve bu sebeple bir fatura servisi altında derlenmeleri ve böylelikle de servisin tekrar kullanılabilirliğinin arttırılması hedeflenmektedir. Böylelikle genel olarak sistemin verimliliği arttırılabilmektedir.

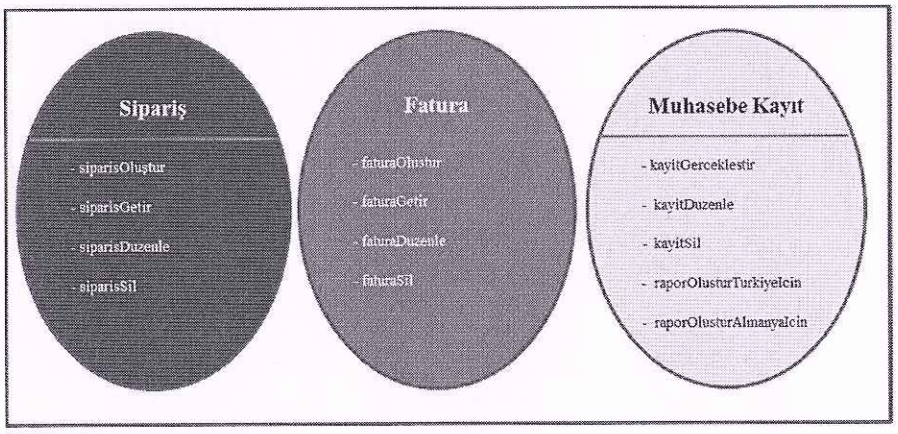


Şekil 3.3. Eski Sistem Üzerinden Doğabilecek Verimlilik Problemi'nin Canlandırılması

Kavramsallık Üzerinden Doğabilecek Problemler:

Bölüm 2.1.3.2'de kavramsallık kavramı üzerinde durulmuştur. Kavramsallaştırma yapılırken servisin isimlendirilmesinin ve servis içerisinde tanımlanan fonksiyonların birbirleriyle uyumu son derece önem arz etmektedir. Aşağıda bu konu ile ilgili karşılaşılabilecek iki ayrı problem üzerinde durulmuştur.

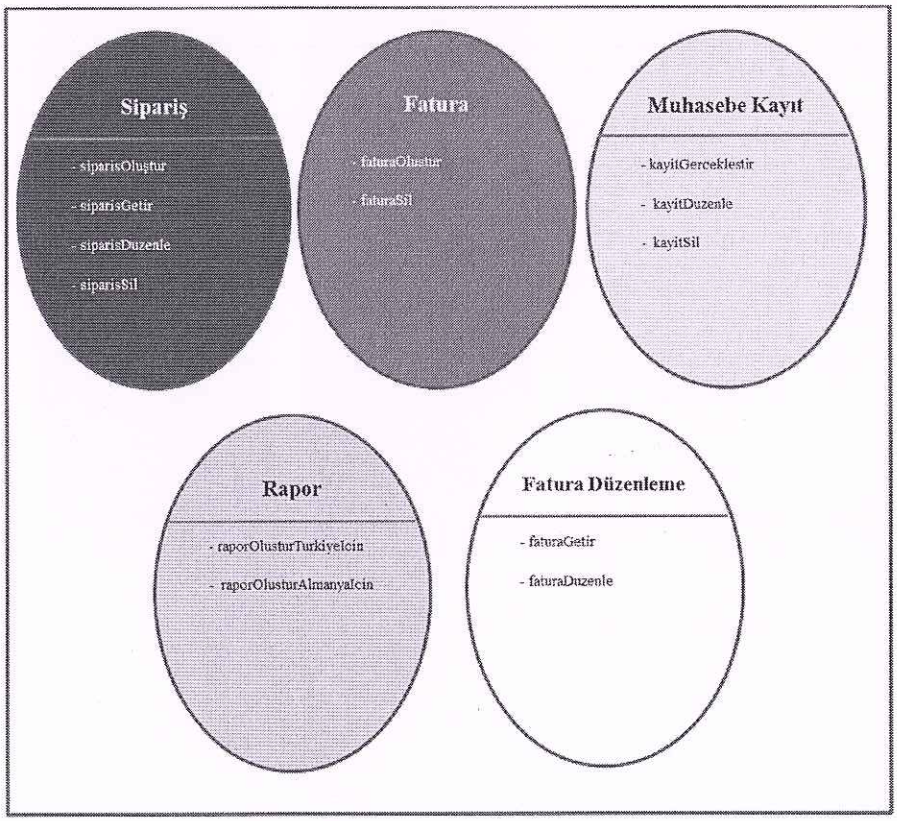
Problem 1: Gerekli olduğu halde yeni bir servisin geliştirilmemesi.



Şekil 3.4. Kavramsallık Üzerinden Doğabilecek Yeni Bir Servisin Yaratılmaması Problemi'nin Canlandırılması

Şekil 3.4'te Firma A'nın servis envanterinden bir kesit alınmıştır. Firmanın servis envanterindeki Muhasebe Kayıt servisinin fonksiyonları yakından incelendiğinde kavramsallık açısından bir uyumsuzluğun söz konusu olduğu görülecektir. Servisin kapsadığı iki ayrı ülkenin muhasebe sistemine göre finansal rapor oluşturmak için kullanılan raporOlusturTurkiyeIcin ve raporOlusturAlmanyaIcin fonksiyonlarının kavramsallık açısından yerlerinin mevcut servis olmadığı anlaşılabilmektedir. Ancak yeni bir servis oluşturmak istemeyen yazılım mimarları servisi bu hali ile de bırakabilir ve servis SOA'nın gerektirdiği diğer tüm servis geliştirme kriterlerini gerçekleştiriyor olabilir. Buradaki sorun kavramsal olarak yerleri yanlış olan fonksiyonların ilerki projelerde farkedilemeyip yeniden geliştirilmek zorunda kalınabileceğidir. Bununla birlikte farkedilebilmeleri durumunda da Muhasebe Kayıt servisinin farklı amaçlar için gereksiz yere tekrar kullanılmak zorunda kalınabileceği unutulmamalıdır. Çözüm olarak ise Şekil 3.6'daki gibi sadece bu fonksiyonların amaçlarına özel ve bunları kapsayan yeni bir servis oluşturulması önerilebilir.

Problem 2: Gereksiz servislerin geliştirilmesi.

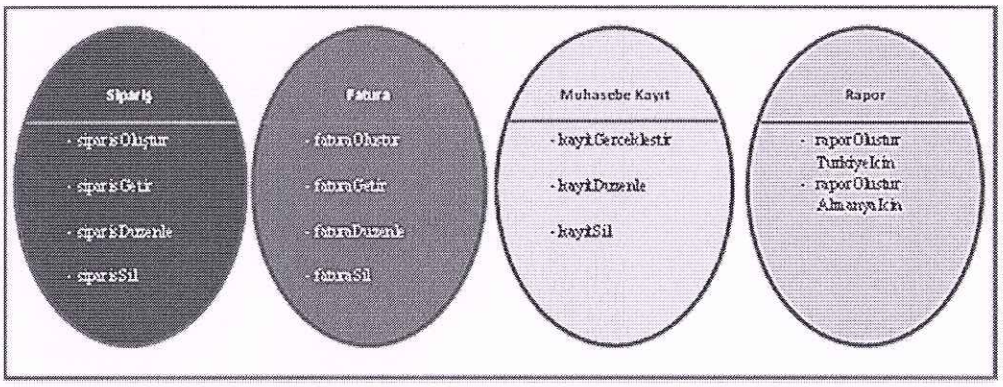


Şekil 3.5. Kavramsallık Üzerinden Doğabilecek Gereksiz Servislerin Yaratılması Problemi'nin Canlandırılması

Şekil 3.5'te Firma A'nın servis envanterinden bir kesit alınmıştır. Bu örnekte yukarıdaki örneğin aksine gereksiz servis oluşturmanın zararları üstünde durulacaktır. Servis envanteri yakından incelenirse, Fatura ve Fatura Düzenle adında iki ayrı servisin geliştirildiği ve bu iki ayrı servisin fonksiyonlarının temelde fatura işlemleri ile ilgili olduğu görülebilir. Bu yüzden kavramsallık açısından iki servisin fonksiyonlarının tek bir servisin altında toplanması daha yararlı olabilmektedir. Buradaki sorun servislerin tekrar kullanılabilirliğinin arttırılamaması ile ilgilidir. Farklı servislerin kapsamında bulunan ve aynı amaca hizmet eden fonksiyonların kavramsallık açısından tek bir servis altında toplanmaları o servisin tekrar kullanımını arttırabilmektedir. Bununla birlikte gereksiz servisler, servis envanterinde kalabalık yaratabilmekte ve fonksiyonların gözden kaçmasına sebep olabilmektedir. Örnek olarak ise Fatura servisinin fonksiyonlarının eksik olduğunu

düşünen bir yazılım mimarı, bu fonksiyonları tamamlamayı düşünebilir ve bunların diğer bir serviste bulunduğunu fark etmeyebilir. Bu kod tekrarı SOA'nın asla kabullenebileceği bir şey değildir. Çözüm olarak ise Şekil 3.6'daki gibi Fatura ve Fatura Düzenleme servislerinin fonksiyonların tek bir servis altında toplanması önerilebilir.

Şekil 3.6'daki envanterin kavramsallığının yukarıdaki her iki envanterdeki kavramsallıktan daha verimli bir kavramsallık olduğu söylenebilir. Zira şekil 3.6'daki envanterde bulunan servisler, kimlikleri ve sahip oldukları fonksiyonları ile SOA'nın öngördüğü kavramsallık kriterlerini diğer iki envanterde bulunan servislere göre daha fazla destekleyebilmektedirler.



Şekil 3.6. Kavramsallık Üzerinden Doğabilecek Problemlerin Çözümünün Canlandırılması

Service Level Agreement (SLA) – Servis Seviye Anlaşmaları Üzerinden Doğabilecek Problemler:

Servis Seviye Anlaşmaları, SOA'nın öngördüğü servis geliştirme prensiplerinden servislerin özellikleri ile ilgili bilgileri barındıran Servis Sözleşmeleri'nin bir parçasıdır. Servis Sözleşmeleri teknik ve teorik olmak üzere iki bölümden oluşmaktadır. Teknik bölüm sözleşmenin makineler tarafından anlaşılabilen kısmını oluşturmaktadır. Teorik bölüm ise servisler ile ilgili özelliklerin insanlar tarafından da anlaşılabileceği Servis Seviye Anlaşmaları'ndan oluşmaktadır. Servis Seviye Anlaşmaları örnek olarak servisin erişilebilirlik zamanları, servis fonksiyonlarının ortalama cevap süreleri gibi birçok bilgi içerebilmektedir. Bununla birlikte Servis

Seviye Anlaşmalarının daha farklı meta veriler ile genişletilmesi önerilmektedir. Bu sayede servis geliştirme kriterlerinden olan Servislerin Keşfedilebilirliği desteklenebilmektedir. Aşağıda, örnek olarak Şekil 3.6'daki Rapor servisine ait ve içerdiği fonksiyonların özelliklerinin açıklamaları ile genişletilen Servis Seviye Anlaşmasının, oluşturabileceği bir problem üzerinde durulmuştur.

Gösterim şekli 1:

Rapor servisi:

- public Rapor *raporOlusturTurkiyeIcin* (String yil, String periyot): İstenen yıl ve periyot için veritabanından Türkiye'ye göre rapor çekilir. Sonuç olumlu ise Rapor nesnesi olumsuz ise null döner.
- public Rapor *raporOlusturAlmanyaIcin* (String yil, String periyot): İstenen yıl ve periyot için veritabanından Almanya'ya göre rapor çekilir. Sonuç olumlu ise Rapor nesnesi olumsuz ise *null* döner.

Gösterim Şekli 2:

Tablo 3.1. Bir Servis'in Fonksiyonlarına Ait Özelliklerin Tablo Halinde Gösterim Şekli

Rapor servisi	
raporOlusturTurkiyeIcin – Fonksiyonu	Girdi: İstenen yıl ve periyot. Çıktı: Türk Muhasebe Sistemine göre düzenlenmiş Rapor nesnesi.
raporOlusturAlmanyaIcin – Fonksiyonu	Girdi: İstenen yıl ve periyot. Çıktı: Alman Muhasebe Sistemine göre düzenlenmiş Rapor nesnesi.

Bir Servisin fonksiyonlarının özellikleri birden çok farklı biçimde gösterilebilir ve bunun için bir standart geliştirilmemiştir. Yukarıdaki örnekte rapor servisinin fonksiyonlarının özellikleri iki farklı biçimde gösterilmiştir. İlk gösterim şekli,

fonksiyonları liste halinde ayrıntılı bir biçimde açıklamaktadır. İkinci gösterim şekli ise, tablo halinde fonksiyonların girdi ve çıktılarını göstermektedir. Buradaki problem ise bu işlemin bir standardının olmaması ve böylelikle de servislerin keşfedilebilirliğinin desteklenememesidir. Zira bu çok biçimlilik sebebi ile bazı fonksiyonlar veya bu fonksiyonların özellikleri yazılım mimarlarının gözünden kaçabilmekte ve bu durum da kod tekrarına sebep olabilmektedir.

3.3. Literatür Taraması

Tarif edilen problemin çözümü noktasında günümüze kadar bazı çalışmalar yapılmıştır. Bu konudaki en önemli çalışmalar aşağıda listelenmiştir;

1. Büttner; *SOA – Reifegradmodelle Analyse und Weiterentwicklung von Reifegradmodellen für SOA*: Kaynakta, önde gelen yazılım şirketlerinin geliştirdikleri SOA olgunluk modelleri karşılaştırılmış olup, karşılaştırılan bu modellerin genel bir uygulama metodolojisi barındırmadığı ve böyle bir metodolojinin geliştirilip, uygulama uzmanlarının bu büyük ihtiyacının karşılanması gerektiğinden bahsedilmiştir. [1]
2. Erl; *Service-Oriented Architecture – concepts, Technology and Design*: Kaynakta servis odaklı mimari taslaklarının neleri içermeleri gerektiği, hangi teknolojileri barındırdığı ve dizayn aşamasında nelere dikkat edilmesi gerektiği konusunda geniş bilgi verilmiştir. Ancak, adım adım nasıl bir sıralama ile uygulamanın sağlanabileceği belirtilmemiştir. [4]
3. Erl: *Entwurfsprinzipien für serviceorientierte Architektur*: Çalışmada, servis odaklı mimarinin yapı taşları olan servislerin, hangi prensipler temelinde geliştirilmesi gerektiği konusuna geniş yer ayrılmıştır. Her servis, ancak belirli özellikleri taşıdığı durumda SOA temelli bir servis olarak kabul görebilmektedir. Bu servislerin, ne şekilde bu özellikleri taşıyacak hale getirilebileceği ve bu işlem için hangi adımların izlenmesi gerektiğine ise değinilmemiştir. [7]
4. Erl: *SOA Design Patterns*: Kaynakta, çeşitli SOA örnekleri ışığında, SOA yaklaşımının sunduğu kolaylıklar gösterilmiştir. Gösterilen örnekler oldukça

aydınlatıcı olup, uygulamalarda yön gösterebilecek yapıdadır. Bu örnek yapıya ulaşmak için izlenmesi gereken metodlara değinilmemiştir. [9]

5. Erl; *Web Service Contract Design and Versioning for SOA*: SOA temelli servis sözleşmelerinin nasıl geliştirilmesi gerektiği üzerine odaklanmış kaynak, bu sözleşmelerin farklı sürümlerde nasıl kullanabileceğine dair örnekler sunmuştur. Servis sözleşmeleri, SOA prensiplerinden sadece bir tanesini oluşturmaktadır, kaynakta bu prensibin diğer prensipleri nasıl etkileyebileceği konusuna ayrıntılı olarak yer verilmiştir. Erl'in bu çalışması da bir metodoloji içermemektedir. [14]
6. Meydanoğlu v.d.; *Geschäftsprozessmanagement und SOA*: Kaynakta iş süreçleri ile SOA arasındaki yakın ilişkiden söz edilmiş ve servislerin iş süreçlerine göre nasıl türetilebileceği konusuna değinilmiştir. Ancak SOA genel hatları ile ele alınmış olup, detaya inilmemiştir. [27]
7. Herand v.d.; *A healthcare management system for Turkey based on a service-oriented architecture*: Kaynakta SOA'nın Türkiye'deki sağlık sistemine nasıl uygulanabileceğine dair bilgi verilmiştir. Sıralama olarak doğru bir uygulama takip eden kaynak, isimlendirilmemiş bir metodoloji içermektedir. [29]
8. Dreifus v.d.; *Systematisierung der Nutzenpotentiale einer SOA*: Kaynakta, SOA'nın sağladığı olanaklardan bahsedilmiştir. Ayrıca bu olanakların sistematize edilmesi gerektiği belirtilmiştir. Böylece kullanıcıların elde etmek istedikleri olanaklara göre SOA'yı kullanabilmeleri sağlanmıştır. Ancak bu kaynak ta bir metodoloji içermemektedir. [34]
9. Zimmermann v.d.; *Capability Diagnostics of Enterprise Service Architectures using a dedicated Software Architecture Reference Model*: Kurumsal servis mimarilerinin kalite testlerinin nasıl yapılabileceğine değinen kaynak, aynı SOA olgunluk modellerindeki yapıyı takip ederek bir kalite seviyesi belirleme yoluna gitmiştir. Bu çalışma da SOA üzerine bir uygulama metodolojisi içermemektedir. [42]
10. Zimmermann v.d.; *A Pattern Language for Architecture Assessment of Service-oriented Enterprise Systems*: Yazılım mimarilerinin değerlendirilebilmesi için örnek bir dil geliştiren kaynak, bu işlemin nasıl yapılabileceğine dair genel bilgiler içermektedir. Ancak değerlendirme

işlemini açıklayan bu kaynak, değerlendirdiği kalitenin nasıl elde edilebileceğine dair bilgi vermemektedir. [44]

11. Bräbender v.d.; *Geschäftsprozessmanagement als Grundlage für SOA*: İş süreçlerinin SOA'nın dizaynında büyük rol oynadığını gösteren kaynak, iş süreçlerini hiyerarşik sıraya göre dizip bu işlemin takip edilmesini önermektedir. İş süreçlerinin önemini vurgulayan kaynak, genel olarak SOA uygulamayı ele almamaktadır ve genişletilmesi gereken teorileriler içermektedir. [48]
12. Liebhart: *SOA goes real – Serviceorientierte Architekturen erfolgreich planen und einführen*: Servis odaklı mimarilerin nasıl doğru bir şekilde planlanması ve uygulanması gerektiği üzerine geliştirilmiş bir kaynaktır. SOA'nın önem verdiği noktalara özellikle değinen kaynak oldukça geniş ve detaylı bir çalışma olmuştur. Ancak, pratik ve kolay anlaşılabilir bir kaynak olmaktan biraz uzak kalmıştır. [49]
13. Marcs v.d.; *Service-oriented Architecture. A Planning and Implementation Guide for Business and Technology*: SOA'nın işletme ve teknoloji açısından nasıl planlanması gerektiğine dair bir yol gösterici olarak işlev görebilecek kaynak, önemli detaylar içermektedir. İş süreçlerine üzerine uygulama geliştirme konusuna özellikle değinmiştir. Bu çalışma kapsamında geliştirilen bir uygulama metodolojisi için oldukça değerli bir kaynaktır. [50]
14. Josuttis; *SOA in der Praxis. System-Design für verteilte Geschäftsprozesse*: Kaynak, dağıtık iş süreçlerine SOA'nın nasıl uygulanabileceği üzerine pratik örnekler içermektedir. Pratik örnekleri ile kaynak, önemli bir uygulama imkanı sunmaktadır. Aynı şekilde bir uygulama metodolojisi için oldukça değerli bir kaynaktır. [51]
15. Billing; *SOA im IT-Service-Management*. Kaynak, servis yönetiminin nasıl gerçekleştirilebileceğini göstermektedir. Uygulama uzmanları tarafından geliştirilen ve bir kitap bölümü olan kaynak, önemli pratik bilgiler içermektedir. Bu çalışma kapsamında geliştirilen metodolojinin önemli bir parçası olmuştur. [52]
16. Zimmermann; *An Architectural Decision Modeling Framework for Service-Oriented Architecture Design*: SOA için mimari karar modeli geliştirmiş bu doktora tezi, önemli bir SOA kaynağı olarak görülebilir. Özellikle, hangi

süreçler için, hangi servislerin geliştirilmesi gerektiği üzerine çok önemli bir karar destek kaynağı halini almıştır. Tezin içerdiği bilgiler geliştirilen metodoloji için köklü bir kaynak oluşturmaktadır. [58]

17. Perepletchikov v.d.; *Formalising Service-Oriented Design*: Biçimsel bir model çerçevesinde, servis odaklı sistemlerin nasıl geliştirilebileceğine değinen kaynak, önceden kalite tahmininde bulunabilmeye imkan vermektedir. Öncelikle ihtiyaçların belirlenmesi ile oluşturulmuş model, son derece yararlı bir SOA uygulama kaynağıdır. Ancak model pratik ve hızlı kullanılabilecek bir yapıda değildir. [61]
18. Engels v.d.; *Quasar Enterprise*: SOA temelli servislerin belirlenmesi açısından önemli bir bilgi kaynağıdır. Ancak benzerleri ile bu kaynak arasında belirgin farklar vardır. İş süreçleri üzerine yoğun bir şekilde odaklanmış olan kaynak, UML diyagramları ile oluşturulan taslaklar çerçevesinde hareket etmeyi uygun görmüştür. Kaynak, genel bir SOA uygulama metodolojisi içermemektedir. [62]
19. IBM; *Rational Unified Process for Service-Oriented Modeling and Architecture (RUP/SOMA)*: Servis odaklı yapının özelleştirilmesi hedeflenen kaynakta, analiz aşamasında UML ve BPMN kullanılmıştır. Çalışma aynı zamanda, iş süreçlerinden servislerin nasıl türetilebileceğine dair örnekler barındırmaktadır. Kaynak genel bir uygulama metodolojisi içermemektedir. [63]
20. Gebhart; *Qualitätsorientierter Entwurf von Anwendungsdiensten*: Kaynak, SOA temelli servisler için, kalite odaklı taslak geliştirmeyi hedeflemektedir. Çalışmada, iş süreçlerinin analizinden başlanarak, yüksek kalitede servis geliştirilebilmenin yolları aranmıştır. SOA'nın hedeflerini gerçekleştirebilecek bir yol takip eden kaynak, geliştirilen metodoloji için önemli bir kaynak görevi görmüştür. [64]
21. Erradi v.d.; *Service Oriented Architecture Framework (SOAF)*: SOA temelli servislerin geliştirilebilmesi aşamasında destek olabilecek, teorik açıdan oldukça geniş bir kaynaktır. Ancak, kaynağın pratik açıdan zayıf olması ve standart bir dil kullanmamış olması, okuyucu için algılama zorluğu meydana getirmektedir. Genel hatlarıyla kaynak, önemli bir SOA uygulama kaynağı olarak görülebilir. [66]

22. OMG; *Service oriented architecture Modeling Language (SoaML)*: SOAML ile OMG, SOA için yeni UML araçları sunmaktadır. Bu araçlar, Servis Odaklı Mimari'nin kapsam ve önemini açıkça ifade etmektedirler. Ancak SOAML, herhangi bir SOA uygulama yöntemi sunmamaktadır. [67]
23. Humm v.d.; *Normalform für Services*: SOA temelli servisler için bir yapı biçimi oluşturan kaynak, çok sayıdaki SOA prensiplerinin, servisler üzerinde somutlaştırılmasını sağlamaktadır. Kaynak aynı zamanda, servislerin bu kriterleri sağlayıp sağlamadığını kontrol edilebilme konusunda da faydalı olmaktadır. Geliştirilen biçimsel yapı, genel bir uygulama metodolojisi içermemektedir. [68]

Listelenen yirmi üç çalışmanın ışığında, tespit edilen eksikliklerin göz önüne alınıp eklenmesi ile geliştirilen bölüm 4'teki metodoloji sayesinde, şirketlere daha kolay SOA uygulama imkanı sağlanabileceği düşünülmektedir.

Yürütülen bu çalışma ile, daha önceki çalışmalardan da yararlanılarak elde edilen metodoloji, bilime, dağıtık yazılım mimarisinin en gelişmiş hali olan SOA'nın uygulamaya geçiş aşamasının, kolaylıkla nasıl gerçekleştirilebileceği konusunda ışık tutacaktır.

3.4. Talepler

Problem tanımı bölümünde tarif edilen ihtiyaçların karşılanabilmesi için branşlara veya sektörlere özel bir yapıdan uzak, tüm iş süreçlerine uygulanabilecek bir metodolojiye ihtiyaç duyulmaktadır. Bu sebepten dolayı elinizde bulunan doktora tezi, "İş Süreçlerine Servis Odaklı Mimari Uygulama Metodolojisi" olarak isimlendirilerek, tezde, her branşa ve sektöre ait iş süreçlerine uygulanabilecek bir SOA uygulama metodolojisinin geliştirilmesi amaçlanmıştır. Sorunun çözümüne yardımcı olabilecek birçok çalışma mevcuttur ve bu çalışmaların önemli bir kısmı literatür taraması bölümünde listelenmiştir, ancak bunlar birbirinden bağımsız, bir bütünlük içermeyen ve ideal bir SOA'nın neleri içermesi gerektiğine dair prensipleri, yaklaşımları, teknikleri içeren çalışmalardır. Bu tezde çözümü aranan problem ise

bugüne kadar yapılan çalışmalarda ifade edilen idealin "NASIL?" elde edilebileceği ile ilgilidir.

Bu konuda, öncelikle SOA projelerine katılmış, tecrübeli uzmanlar ve akademisyenlerin yönlendirmeleri doğrultusunda belirlenen talepler genellenerek bir talep listesi oluşturulmuştur. Çözüm olarak geliştirilecek metodolojinin sağlaması gereken kriterlere dair talepler aşağıda listelendiği şekildedir [69],[70];

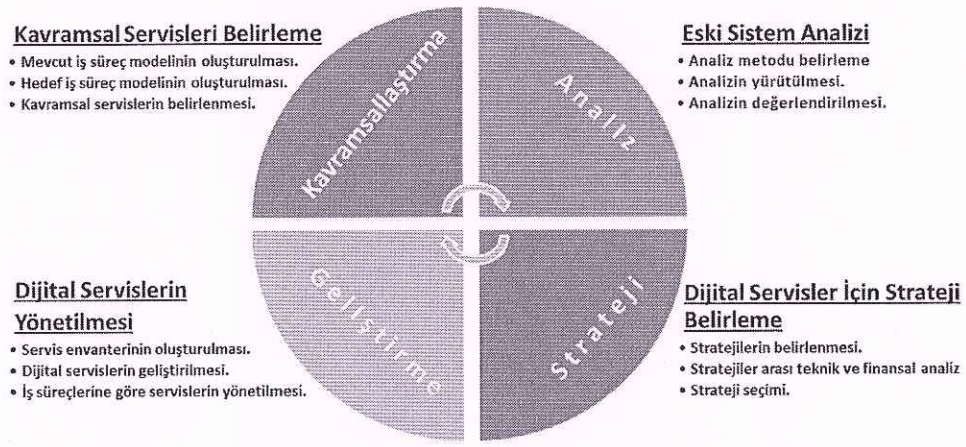
- Servis kaydının, tasarım ve uygulama aşamasında yaygın olarak kullanımı.
- Servis kaydının, uygulama çeşitlendirme, uzaktan yayınlama ve servis keşfi için yaygın olarak kullanımı.
- İş süreç modelleme araç ve standartlarının kullanımı (BPMN, BPDM).
- Servislerin güvenlik kurallarına dair, gelişmiş servis yönetimi izleme, uyarı (alerting), giriş (logging) ve uygulamalar.
- SLA'ya (Service Level Agreement) göre servis performansı.
- Uygulama kodlarının iş süreçlerine hızlı uyumunu sağlayabilmek için standart arayüzlerin kullanılması.
- Kurumsal mimari tarafından yürütülen IT (Information Technology) organizasyonundaki iş stratejilerinin derinlemesine anlaşılması.
- Mimari standartların sıkı bir şekilde uygulanması.
- Süreç sapmaları, değişimleri ve servis tekrar kullanımı için ölçüm tanımlayıcıları.
- Uygulamaların kavramsal tasarımlarının (blueprint) tanımlanması ve kullanılması.
- Mimari politikalara bağlı projelerin puanlanması ve bunun kurumsal SOA'ya katkısının sağlanması için yönergelerin geliştirilmesi.
- Bu yönergelere uyulabilmesi için projelerin puanlandırılması.
- Net bir şekilde tanımlanmış uygulamalar, enformasyon ve teknolojileri barındıran referans mimarileri ve güvenlik mimarileri.
- Uygulama kodlarından soyutlanmış iş politikaları, servis olarak kullanıma hazır iş kuralları.
- En iyi uygulamaların bir araya getirilip paylaşılması.

BÖLÜM 4. İŞ SÜREÇLERİNE SOA UYGULAMA METODOLOJİSİ

4.1. Metodolojinin İçeriği

SOA yaklaşımının iş süreçlerine nasıl uygulanabileceği, günümüz SOA kaynaklarında ve SOA olgunluk modellerinin hiçbirinde açık bir şekilde ifade edilmemiştir. Bu konuda SOA olgunluk modelleri çok kısıtlı veya genellikle hiçbir bilgi içermemektedirler. Bununla birlikte SOA literatüründe, SOA olgunluk modeli tasarımcılarının, bir Servis Odaklı Mimari Uygulama Metodolojisi geliştirilmesinin mutlak gerekliliği yönünde görüş bildirdikleri bilgisine sıkça rastlanmaktadır [1]. Bu tez çalışması kapsamında geliştirilen bu yöndeki ilk çalışma, zamanla olgunluk modeli tasarımcıları ve kurumlar tarafından olgunluk modellerine eklenebilir.

Bu tez çalışması kapsamında geliştirilen 'İş Süreçlerine SOA Uygulama Metodolojisi' birbirini takip eden dört adımdan oluşmaktadır. Bu adımlar Şekil 4.1'de de görülebileceği üzere sırası ile 'Kavramsal Servisleri Belirleme', 'Eski Sistem Analizi', 'Dijital Servisler için Strateji Belirleme' ve 'Dijital Servislerin Yönetilmesi' olarak adlandırılmıştır. Bünyesinde barındırdığı iş süreçlerine SOA yaklaşımı uygulayacak her kurumun bu adımları sırası ile takip etmesi, uygulanacak SOA'nın kalitesi açısından büyük önem arz etmektedir.



Şekil 4.1. İş Süreçlerine SOA Uygulama Metodolojisi

İş Süreçlerine SOA Uygulama Metodolojisi'nin ilk adımı olan Kavramsal Servisleri Belirleme adımı, ihtiyaç duyulan servislerin kavramsal olarak tasarlanmasını öngörmektedir. Bu adım üç alt adımdan oluşmaktadır. Bu adımlar, 'Mevcut İş Süreçlerinin Oluşturulması', 'Hedef İş Süreçlerinin Oluşturulması' ve 'Kavramsal Servislerin Belirlenmesi' adımlarıdır.

İş Süreçlerine SOA Uygulama Metodolojisi'nin ikinci adımı olan Eski Sistem Analizi adımı, kuruma ait eski sistemin, yeni sistemde kullanılabilecek kısımlarının belirlenmesini öngörmektedir. Bu adım üç alt adımdan oluşmaktadır. Bu adımlar, 'Analiz Metodu Belirleme', 'Analizin Yürütülmesi' ve 'Analizin Değerlendirilmesi' adımlarıdır.

İş Süreçlerine SOA Uygulama Metodolojisi'nin üçüncü adımı olan Dijital Servisler İçin Strateji Belirleme adımı, kurumun SOA yaklaşımı uygulamaya geçerken bir strateji belirlemesi gerektiği esasını öngörmektedir. Bu adım üç alt adımdan oluşmaktadır. Bu adımlar, 'Stratejilerin Belirlenmesi', 'Stratejiler Arası Teknik ve Finansal Analiz' ve 'Strateji Seçimi' adımlarıdır.

İş Süreçlerine SOA Uygulama Metodolojisi'nin dördüncü adımı olan Dijital Servislerin Yönetilmesi adımı, kurum için geliştirilen dijital servislerin, hedef iş süreçleri doğrultusunda yönetilmesi esasını öngörmektedir. Bu adım üç alt adımdan oluşmaktadır. Bu adımlar, 'Servis Envanterinin Oluşturulması', 'Dijital Servislerin

Geliştirilmesi' ve 'İş Süreçlerine Göre Servislerin Yönetilmesi' adımlarıdır. İlerleyen bölümlerde, bu bölümde belirtilen adımlar ayrıntılı bir şekilde açıklanmaktadır.

4.2. Kavramsal Servisleri Belirleme

Farklı sektör gruplarında bulunan kurumlar genellikle farklı iş süreçleri yürütürken, aynı sektör gruplarına ait olan kurumların da birbirinden farklı iş süreçleri yürütebildikleri bilinmektedir. Bu durum ürünler ve hizmetler üzerinde kalite farklılıklarının oluşabilmesi açısından da kaçınılmazdır. İş Süreçlerine SOA Uygulama Metodolojisi, her bir iş süreci için aynı şekilde uygulanmakta, ancak sonuçta elde edilen mimariler her zaman iş süreçlerine göre farklılık göstermektedir. Buna göre SOA yaklaşımında kalitenin yakalanabilmesi için öncelikle kurumun iş süreçlerinin ele alınması gerekmektedir. Ancak kesin olarak karar verilen, optimum ve değişikliklere mümkün olduğunca çabuk adapte edilebilecek iş süreçleri tasarlandıktan sonra, servis geliştirme işlemine başlanması gerekmektedir.

Sonuç olarak, dijital (teknik) yapıya yani servislerin geliştirilmesine geçilmeden önce bir dizi farklı yaklaşım veya (alt) bilim dalı ile süreçlerin ele alınması, kaliteli bir SOA uygulaması için kaçınılmazdır. Her şeyden önce yapılması gereken, hâlihazırda uygulanan gerçek iş süreçlerinin keşfedilmesi, ardından bu süreçler temelinde hedef iş süreçlerinin tasarlanması ve son olarak ta hedef iş süreçlerine göre kavramsal servislerin belirlenmesidir.

Yukarıdaki açıklamalara dayanarak, İş Süreç Yönetimi ve SOA arasındaki ilişki aşağıdaki şekilde özetlenebilir [71];

- İş Süreç Yönetimi ve SOA arasında çok yakın bir ilişki bulunmaktadır.
- Etkili bir SOA yaklaşımı için etkili bir İş Süreç Yönetimi önkoşuldur.
- Servisler, kavramsal olarak tasarlanan iş süreçlerinden türetilmektedirler. Bu durumda yürütülen İş Süreç Yönetimi kalitesi ise SOA yaklaşımının etkinliği arasında doğru bir orantı vardır.

- Etkili bir şekilde yürütülen İş Süreç Yönetimi ve servis geliştirme kriterleri dikkate alınarak geliştirilen servisler temelinde uygulanan SOA yaklaşımı, kalite, çeviklik ve rekabet gücünü beraberinde getirmektedir.

4.2.1. Mevcut iş süreç modelinin oluşturulması

Mevcut iş süreç modeli genel olarak, kurumun yürüttüğü iş süreçlerinin, kurum içi veya dışı çalışanlar tarafından anlaşılabilmesi ve iyileştirilebilmesi için oluşturulmaktadır. SOA temelli servisler de iş süreçlerinden türetildikleri için, öncelikle kurum içi mevcut iş süreçlerinin belirlenip hedefler doğrultusunda iyileştirilmesi gerekmektedir. Günümüzde kurumlar, mevcut iş süreçlerini temel olarak aşağıda ifade edilen iki farklı şekilde veya benzer metotlar kullanarak belirleyebilmektedirler [72],[73],[74],[75],[76].

1. Manuel Yöntem: Geleneksel olarak uygulanan yöntemdir. İş süreçlerinin manuel olarak takip edilerek belirlenmesini esas almaktadır. Bu yöntem ile belirlenen süreçler birçok notasyon türü (örn. BPMN) ile modellenebilmektedir.

2. Süreç Madenciliği: İş süreç yönetiminin önemli bir parçasıdır. Şirketler iş süreç yönetimi projelerine çoğunlukla süreçlerinin tam olarak ne olduğunu bilmeden başlamaktadırlar. Geleneksel metotlarla süreçlerin belirlenmesi hem doğruluk açısından sorunlu olabilmekte hem de çok zaman alabilmektedir. Süreç Madenciliği araçları başka bir şekilde elde edilmesi neredeyse imkansız doğruluk ve analiz imkanları sağlarken ciddi bir zaman tasarrufunu da beraberinde getirmektedirler. Bir süreç iyileştirme projesi söz konusu olması durumunda da süreçler arası karşılaştırmalı değerlendirme yapılabilmesi için bu verilere ihtiyaç duyulmaktadır [41].

Manuel olarak veya gerekli tekniklerin öğrenilerek süreç madenciliği aracılığıyla, mevcut iş süreç modeli oluşturulduktan sonra, tüm iş süreçleri, alt süreçleri ile birlikte açıklamalı ve ayrıntılı bir şekilde belgelendirilmek durumundadır. Bu işlem sadece SOA yaklaşımının uygulanabilmesi için önem arz etmemektedir. Bununla

birlikte zaman ve maliyet açısından iş süreçlerinin önemsendiği günümüzde, ihtiyaç duyulabilecek her türlü iş süreç bilgilerinin elde edilebilmesi açısından da son derece önemlidir.

4.2.2. Hedef iş süreç modelinin oluşturulması

Kurumlardaki hedef iş süreçleri, sadece hâlihazırda yürütülen mevcut iş süreçlerinin optimize edilmiş hali olabilmekle birlikte, temelde değişiklik gösterebilen süreçler de olabilmektedir. SOA yaklaşımı açısından, hedef iş süreçlerinin gelecek zamanlardaki olası değişikliklerin de dikkate alınarak, geniş çaplı projeler olarak belirlenmeleri gerekmektedir. Kavramsal servisler, temel olarak bu aşamada tasarlanan ve modellenen iş süreçlerinden türetilmektedir [77],[78].

Aynı şekilde hedef iş süreç modeli oluşturulduktan sonra da, tüm iş süreçleri ve barındırdıkları alt süreçler açıklamaları ile birlikte ayrıntılı olarak belgelendirilmelidir. Bu şekilde, yazılım mimarlarının da iş süreçlerini anlayabilmeleri kolaylaşmaktadır. Ayrıca yazılım mimarlarının hedef süreçlerden türettikleri SOA temelli servislerin de daha az hata ile tasarlanmaları sağlanabilmektedir.

4.2.3. Kavramsal servislerin belirlenmesi

Bu bölümde, bölüm 2.4'teki servis odaklılık ve merkezi mantığın sağlanmasına dair yapı esas alınarak ve yukarıdan aşağıya uygulama stratejisi kullanılarak kavramsal servislerin belirlenmesi işlemi gerçekleştirilmektedir. Bu işlem için öncelikle, bir önceki aşamada belirlenen hedef iş süreçleri temelinde, ihtiyaç duyulan tüm fonksiyonlar belirlenmektedir. Belirlenen bu fonksiyonlar, yapıları ve girdi ve çıktı parametreleri dikkate alınarak, tekrar etmeleri durumunda elenmektedir. Daha sonra kavramsal yapının sağlanabilmesi için tekrar içermeyen bu fonksiyonlar, amacına uygun olarak isimlendirilen servisler altında toplanarak sınıflandırılmaktadır.

4.3. Eski Sistem Analizi

SOA yaklaşımı uygulayarak sistem geliştirmeyi arzulayan kurumlar uygulama maliyetlerini düşürebilmek için genellikle hâlihazırda kullandıkları sistemleri kısmen de olsa yeni geliştirilecek sisteme entegre etme çalışmaları yürütmektedirler. Bu işlemin sağlanabilmesi için, kurumlara ait eski sistemlerin analizlerinin gerçekleştirilmesi gerekmektedir. Bu işlem için, sırasıyla, bir analiz metodunun belirlenmesi, analizin yürütülmesi ve analizin değerlendirilmesi adımlarının izlenmesi gerekmektedir.

4.3.1. Analiz metodu belirleme

Kurumların eski sistemlerinin incelenebilmesi oldukça karmaşık bir işlemdir. Eski sistemler, kurumların büyüklüklerine bağlı olarak binlerce satır yazılım ve yüksek kapasiteli donanımlardan oluşabilmektedir. Bu derece kapsamlı yapıların analizi için daha önce önerilen aşağıdaki iki metottan uygun olan bir tanesi kullanılabilir.

1. Sistem Belgelerinin Takibi: Kurumların, kullandıkları IT sistemlerini belgelendirmiş oldukları durumlarda izlenebilecek bir metottur. Eğer mevcut kullanılan sistem UML veya benzeri bir notasyon tekniği kullanılarak modellenmiş ise, bu model izlenerek analiz edilecek sistemin, SOA'ya entegre edilebilecek ve edilemeyecek kısımlarının belirlenmesi oldukça kolaylaşmaktadır.

2. İş Süreçlerinin İncelenmesi: Günümüzde kurumların kullandıkları IT sistemleri genellikle iyi belgelendirilmemektedir. Birçok kurum bu işlemler için zaman ve para ayırmayı tercih etmemektedir. Bu durumlarda kurumun bölüm 4.2.1'de oluşturduğu mevcut iş süreçlerinin incelenmesi gerekmektedir. Bu inceleme ile birlikte sistem üzerindeki olası fonksiyonların tespiti gerçekleştirilebilir. Bu tespit işleminden sonra belirlenen bu fonksiyonların sistem üzerindeki yerlerinin aranması ve bulunması çalışmalarının yürütülmesi gerekmektedir.

4.3.2. Analizin yürütülmesi

Bu bölümde, öncelikle eski sistemin sahip olduğu, ancak yeni sistemde yer alması söz konusu olmayan fonksiyonları barındıran iş süreçlerinin belirlenerek, elenmeleri gerekmektedir. Bu işlem için mevcut iş süreçlerinde bulunan, ancak hedef iş süreçlerinde yer almayan süreçlerin belirlenmesi ve sistemin bu fonksiyonlara yönelik analizinin gerçekleştirilmemesi sağlanmalıdır.

Bir sonraki işlem ise, içerdikleri fonksiyonlar itibari ile mevcut ve hedef iş süreçleri arasında birebir örtüşen iş süreçlerinin tespit edilmesidir. Bu yönde tespit edilen süreçler üzerinde, içerdikleri fonksiyonların girdi ve çıktı parametrelerinin uyuşma durumunun belirlenmesi gerekmektedir. Girdi ve çıktı parametreleri ve fonksiyonellikleri tam olarak örtüşen fonksiyonların yeniden geliştirilmemesi sağlanmış olacaktır.

4.3.3. Analizin değerlendirilmesi

Bu aşamada, bir önceki bölümdeki analize dayanarak hangi iş süreçlerine ait fonksiyonların SOA yaklaşımına dayanan yeni sistemde de kullanılabileceğine dair genel bir özet yapılmaktadır. Bu işlem bir sonraki aşamada gerçekleştirilen dijital servis geliştirme alternatifleri belirlenirken çok önemli bir rol oynamaktadır. Zira belirlenen bazı alternatifler eski sistemdeki fonksiyonların kullanımını öngörecektir.

4.4. Dijital Servisler İçin Strateji Belirleme

Bu bölüm, kavramsal servisler ile dijital servisler arasında bir geçiş noktası olarak görülebilir. Kavramsal servislerin teknik boyuta taşınması için, öncelikle, daha önceki bölümlerdeki çalışmalar ve bölüm 2.5.2'deki, eski sistemlerin SOA'ya entegrasyonuna dair yaklaşımlar doğrultusunda, belirli alternatiflerin oluşturulması sağlanmaktadır. Daha sonra bu alternatifler arasında teknik ve finansal analizler yürütülmektedir. Son olarak aralarından bir alternatifin seçilmesi işlemi ile birlikte bu bölüm son bulmaktadır.

4.4.1. Stratejilerin belirlenmesi

Eski sistemin de varlığı düşünülerek belirlenmesi gereken alternatifler, temelde, yeniden geliştirme, paketleme ve göç ettirme olarak özetlenebilir. Bu aşamada bu alternatiflerden hangilerinin uygulanabilir olduklarının belirlenmesi gerekmektedir.

Alternatif belirleme aşamasında, teknik ve finansal bakış açısı büyük önem teşkil etmektedir. Kurumlar, alternatif belirlerken, eski sistem bileşenlerinin web servisi bileşenleri olarak yeniden kullanılabilmesine, özel entegrasyon metodu belirlenmesine veya eski sistemi bir kenara bırakarak, sistemin yeniden geliştirilmesine karar verebilirler. Kurumların kendi yapılarına uygun bir alternatif belirleyebilmeleri durumunda, SOA kalitesi açısından başarılı sonuçlar elde etme olasılıkları yükselmektedir [55],[79].

4.4.2. Stratejiler arası teknik ve finansal analiz

Bu bölümde, bir önceki bölümde belirlenen alternatifler arasında teknik ve finansal analizlerin yapılması gerekmektedir. Bu işlem, bir sonraki aşamada alternatif seçerken yapılabilecek puanlama işlemi için büyük önem teşkil etmektedir. Yapılması gereken her bir alternatif için teknik ve finansal yarar ve zararların ortaya çıkarılmasıdır. Her bir alternatif, yapı itibari ile belirli hedefler içermektedir. Bu hedeflerin teknik ve finansal yükü ise bu aşamada ortaya çıkarılıp yorumlanarak ve bir sonraki bölümde puanlanarak, alternatif seçimi işlemi gerçekleştirilmektedir [80],[81].

4.4.3. Strateji seçimi

Dijital servisler için alternatif geliştirilirken, bir veya daha fazla alternatifin tanımlanmasına, yorumlanmasına, analizine ve sonrasında bu alternatifler arasından, gerçekleştirilecek SOA yaklaşımı için en iyi alternatifin seçilmesine ihtiyaç duyulmaktadır [55],[79].

Seçim işlemi için tablo 4.1'deki gibi bir yapının oluşturulması ve bu yapıdaki alternatiflerin kriterlerinin ne ölçüde karşıladığına dair puanlama işleminin

yürütülmesi gerekmektedir. Kurumlara özel sonuçların elde edilebilmesi için bu puanlar, kurum değerlerine göre belirlenen kriter katsayıları ile çarpılmaktadır. Kriter katsayıları yüzdelik olarak dağıtılabilir ancak toplam değerleri bir olmak durumundadır [83],[84],[85].

Tablo 4.1. Alternatif Seçim Tablosu

		Belirlenen Alternatifler			
		Alternatif 1	*Alternatif 2*	*Alternatif 3*	**Katsayı**
Kriterler	*Kriter 1*				
	Kriter 2				
	Kriter 3				
	Toplam Puan				

4.5. Dijital Servislerin Yönetilmesi

Bu bölümde, bir önceki bölümde seçilen alternatif kapsamında bir servis envanterinin oluşturulması gerekmektedir. Ardından bu envanterdeki yapıya uygun olarak dijital servislerin geliştirilmesi ve son olarak ta dijital servislerin iş süreçlerine göre yönetilmesi işlemleri gerçekleştirilmektedir.

4.5.1. Servis envanterinin oluşturulması

Servis envanteri oluşturulurken, kavramsal olarak tasarlanan ve bir sonraki aşamada geliştirilmelerine başlanacak olan servislerin hepsi için aşağıda gösterilen yapının gerçekleştirilmesi gerekmektedir. Buradaki amaç oluşturulan envanter sayesinde servislerin keşfedilebilirliğinin kolaylaştırılmasıdır [82].

Servis adı:

- çıktı parametresi *fonksiyon adı* (girdi parametresi): fonksiyonun ayrıntılı açıklaması.

4.5.2. Dijital servislerin geliştirilmesi

Bu aşamada, bölüm 4.3.2'de tasarlanan kavramsal servisler ve bir önceki bölümdeki servis envanteri baz alınarak dijital servislerin geliştirilmesinin sağlanmasıdır. Servislerin geliştirilebilmeleri, yani dijital yapıya geçirilebilmeleri için bir programlama dilinin belirlenmesine ve bir adet SOA yönetim platformuna ihtiyaç duyulmaktadır. SOA yönetim platformları hakkında bölüm 2.6'da ayrıntılı bilgi verilmiştir. Kurum yapısı ve hedeflerine göre, lider yazılım şirketleri tarafından geliştirilen bu SOA yönetim platformlarından birisinin, uygulanacak projeler için seçiminin gerçekleştirilmesi gerekmektedir. Bu seçim işlemi için bölüm 2.6'daki yapı kullanılabilir.

4.5.3. İş süreçlerine göre servislerin yönetilmesi

Dijital servislerin iş süreçlerine göre yönetilmesi işlemi, bölüm 2.2'de ifade edildiği şekilde yürütülmektedir. İş süreçlerinin otomatikleştirilebilmesi için, bir önceki bölümde dijital olarak elde edilmiş olan servislerin barındırdıkları fonksiyonların, gene iş süreç sıralamasına göre, birbirinden veri alış verişini gerçekleştirebilecek şekilde bağlanmaları gerekmektedir. Bu şekilde bir fonksiyonun çıktı parametresi diğer bir fonksiyonun girdi parametresi olabilmekte ve bu parametreler SOA yönetim platformu aracılığıyla taşınmaktadır. Bu özellik servislerin birleştirilebilmesi olarak adlandırılmaktadır.

4.6. Metodoloji'nin Güçlü ve Zayıf Yanları

Bu bölüme kadar, İş Süreçlerine SOA Uygulama Metodolojisi'nin adımları ve alt adımları ayrıntılı bir şekilde açıklanmıştır. Bu bölümde ise metodolojinin güçlü ve zayıf yanları karşılaştırılmaktadır. Bu karşılaştırma işlemi Tablo 4.2'de özet bir şekilde sunulmuştur.

Tablo 4.2. İş Süreçlerine SOA Uygulama Metodolojisi'nin Güçlü ve Zayıf Yanları

Güçlü Yanları	Zayıf Yanları
– SOA Kalitesi: Değişim etkisini düşürerek, SOA kalitesinin yükselmesini sağlamaktadır. – Hız: Kaliteli bir SOA uygulamasının, belirlenen adımların sırası ile takip edilerek, hızlı bir şekilde elde edilmesine yardımcı olmaktadır. – Hata Önleme: Uygulama alanını, barındırdığı birçok tekniğin yardımıyla denetleyen adımları sayesinde, hataların önlenmesi sağlanmaktadır. – Maliyet: Uygulama maliyeti oldukça düşük bir metodolojidir. Barındırdığı tekniklerin bir çoğu ek donanım ve yazılım gerektirmeden yürütülebilmektedir. Bununla birlikte daha doğru bir uygulama gerçekleştirmeye yardımcı olduğu oranda, SOA'nın gelecekte doğurabileceği maliyetleri de önlemektedir. – Esneklik: Metodoloji, ilerleyen zamanlar için daha gelişmiş teknikler, metotlar ve bakış açıları ile genişletilebilecek bir yapıdadır.	– İlk Metodoloji: Bu yönde geliştirilen ilk metodoloji olduğu için, farklı metodolojilerle karşılaştırılma imkanı yoktur. – Zayıf Kalma Riski: SOA uygulama alanında artan talepler ve ileride geliştirilebilecek metodolojiler karşısında zayıf kalabilme riski vardır. – Sürekli Geliştirilme İhtiyacı: Zayıf kalma ihtimalini düşürmek için SOA uygulayan ekipler haricinde, ayrıca metodolojinin geliştirilmesi ile ilgilenebilecek uzmanlara ihtiyaç duyulacaktır. – Teknik İhtiyacı: Metodoloji çok sayıda teknik, metot ve bakış açıları içermektedir. SOA uygulamalarının bu metodoloji önderliğinde başarılı bir şekilde yürütülebilmesi için tüm bu tekniklerin de öğrenilmesi gerekmektedir. – İyileştirilen Boyut Sayısı: Metodoloji SOA olgunluk modellerinin denetlediği, altyapı, mimari, enformasyon, proje yönetimi, finans, organizasyon ve yönetim boyutlardan sadece altyapı ve mimari boyutlarını iyileştirmektedir. Diğer boyutları da

	iyileştirebilecek şekilde genişletilmesi gerekmektedir.

4.7. Metodoloji'nin t-Testi ile Değerlendirilmesi

4.7.1. Literatürde t-testi

Literatürde, bir örneklem grubunun deneyden önce ve sonra olmak üzere iki kez sınanması gerektiği durumlarda, Eşli İki Örnek t-Testi'nin kullanıldığı birçok yayın ile karşılaşılabilmektedir. Özellikle SCI ve SSCI kapsamında yer alan dergilerde yayınlanan yayınlarda, bu amaçla çoğunlukla t-testinin kullanıldığı dikkat çekmektedir [86],[87],[88].

4.7.2. t-Testi'nin kullanılmasının sebebi

Eşli iki örnek t-testi; örneklerde, bir örnek grubunun deneyden önce ve sonra olmak üzere iki kez sınanması durumunda olduğu gibi doğal eşleşmenin olduğu durumlarda kullanılabilmektedir. Bu çözümleme aracı ve onun formülü, iki örnekli bir öğrenci t-testini, bir işlemden önce alınan gözlemler ve bir işlemden sonra alınan gözlemlerin büyük bir olasılıkla eşit popülasyon ortalamalı dağılımlardan gelmiş olup olmadıklarını belirlemek için uygulamaktadır. Bu t-test formu her iki popülasyonun da eşit olduğunu varsaymamaktadır [89].

Neyman-Pearson Lemma'ya göre de bu tarz sınamalarda t-testinin en etkili tekniklerden biri olduğu söylenebilir [90],[91].

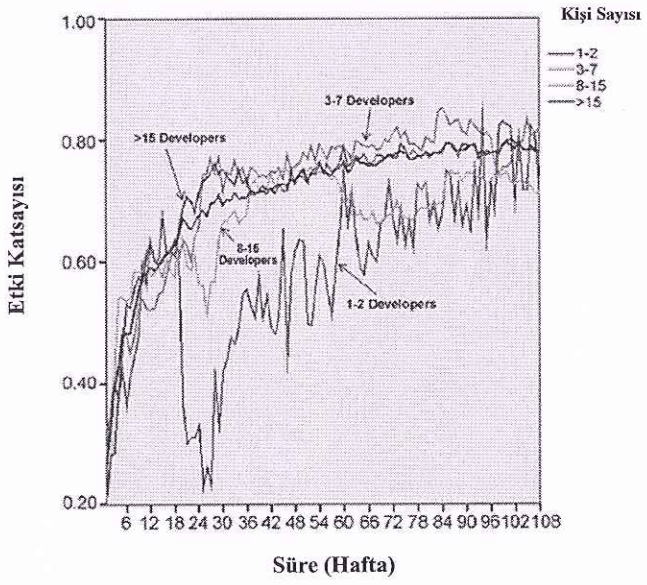
4.7.3. Metodoloji'nin t-testi ile sınanması

Metodoloji'nin t-testi ile sınanabilmesi için bir örnek olay çalışması yürütülmüştür [92],[93],[94].

Örnek olay çalışmasının detayları aşağıdaki gibidir;

- Çalışma 50 kişilik bir örneklem grubu ile gerçekleştirilmiştir.
- Her katılımcının çalışması bireysel bir SOA projesi olarak değerlendirilmiş ve proje yürütme süresi kayıt altına alınmıştır.
- Tüm SOA projeleri aynı birim üzerine uygulanarak gerçekleştirilmiştir (UZEM).
- Öncelikle metodoloji gösterilmeden aynı sonucu elde etmek için harcanan süreler (saat olarak) tespit edilmiştir (Tablo 4.3, 2. Sütun).
- Ardından metodoloji'nin öğretilmesi ile birlikte aynı sonucu elde etmek için harcanan süreler (saat olarak) tespit edilmiştir (Tablo 4.3, 3. Sütun).

Ancak doğru bir değerlendirmenin gerçekleştirilebilmesi için, metodoloji gösterilmeden harcanan çalışma süresi boyunca elde edilen tecrübenin, metodolojinin öğretilmesi esnasında harcanan süre değerine eklenmesi gerektiği de unutulmamalıdır.



Şekil 4.2: Yazılım Projelerinde Öğrenme Eğrisi [95]

Bunun için [95] numaralı kaynakta yazılım projelerini dikkate alarak oluşturulan öğrenme eğrisinin dikkate alınması gerekmektedir. Bu eğriye göre elde edilecek etki katsayısının, metodoloji gösterilmeden önceki çalışmada harcanan süre değeriyle çarpılarak, ikinci çalışmaya eklenmesi, yapılacak sınama işleminin daha doğru

olmasını sağlayabileceği söylenebilir. Bunun için öncelikle kazanılan tecrübe değerleri (ilk çalışma X öğrenme etkisi) belirlenmiştir (Tablo 4.3, 4. Sütun). Ardından bu değer ikinci çalışmanın süre değerlerine eklenmiştir (Tablo 4.3, 5. Sütun).

Tablo 4.3: Örnek Olay Çalışmasından Elde Edilen Sonuçlardan Bir Kesit

	Metodoloji Öncesi Değerler (saat)	Metodoloji Sonrası Değerler (saat)	MÖD X 0.35	MSD + (MÖD X 0.35)
Katılımcı 1	3	1.25	1.05	2.30
Katılımcı 2	3	1.25	1.05	2.30
Katılımcı 3	3.2	1.25	1.12	2.37
Katılımcı 4	2.28	1.25	0.798	2.05
Katılımcı 5	2.8	1.24	0.98	2.22
Katılımcı 6	2.4	1.24	0.84	2.08
Katılımcı 7	3	1.24	1.05	2.29
Katılımcı 8	2.86	1.23	1.001	2.23
.....				
Katılımcı 44	2.1	0.87	0.735	1.61
Katılımcı 45	2.08	0.89	0.728	1.62
Katılımcı 46	2.06	0.79	0.721	1.51
Katılımcı 47	2.15	0.78	0.7525	1.53
Katılımcı 48	2.11	0.71	0.7385	1.45
Katılımcı 49	2.13	0.71	0.7455	1.46
Katılımcı 50	2.11	0.71	0.7385	1.45

Elde edilen bu sonuçlara göre, yürütülecek t testi için aşağıdaki iki hipotez geliştirilebilir;

H0: Metodoloji Öncesi Değerler = Metodoloji Sonrası Değerler

H1: Metodoloji Öncesi Değerler ≠ Metodoloji Sonrası Değerler

Tablo 4.4: Eşli İki Örnek t Testi Sonuçları.

	Metodoloji Öncesi Değerler (saat)	*Metodoloji Sonrası Değerler (saat)*
Ortalama	2.6598	1.97253
Varyans	0.196797918	0.062215361
Gözlem	50	50
Pearson Korelasyonu	0.803918139	
Öngörülen Ortalama Farkı	0	
df	49	
t Stat	17.06450755	
P(T<=t) tek-uçlu	1.47347E-22	
t Kritik tek-uçlu	1.676550893	
P(T<=t) iki-uçlu	2.94693E-22	
t Kritik iki-uçlu	2.009575237	

Tablo 4.4'teki değerlerin yorumlanması gerekirse, % 95 anlamlılık düzeyinde (alfa=0.05 hata payıyla) alınan bu sonuçlara göre, hesaplanan t değerinin (t Stat), t Kritik iki-uçlu değerinden daha büyük bir değere sahip olması, H0 hipotezinin reddedildiği anlamına gelmektedir. Aynı şekilde HO hipotezinin doğru olma olasılığının (P(T<=t)) alfa (0.05) değerinden küçük olması da HO hipotezinin reddedildiği anlamına gelmektedir. Bununla birlikte korelasyonun yaklaşık olarak 0.80 gibi bir değerde çıkması veriler arasındaki uyumu da kanıtlamaktadır. Daha doğru bir ifade ile halihazırda hızlı çalışan bir kişi daha hızlı bir sonuç elde edebilmekte, ancak bununla birlikte karşılaştırmalı olarak daha yavaş çalışan bir kişi de aynı şekilde ilk çalışmasına göre yaklaşık aynı oranda daha hızlı bir sonuç alabilmektedir. Görüldüğü gibi t testi, korelasyon katsayısının anlamlılığını istatiksel olarak analiz edebilmek için oldukça uygun bir yöntemdir [96].

Sonuç olarak, t Testi sonuçlarına göre, iki değer grubu arasında belirgin bir fark olduğu söylenebilir (t=17.065). Bununla birlikte, İş Süreçlerine SOA Uygulama Metodolojisi'nin kullanılması ile birlikte, belirgin bir şekilde daha hızlı sonuç alınabileceği söylenebilir [96].

Tüm bu sonuçlara göre, genel bir finansal karlılık analizi yapmak gerekirse, üzerinde uygulamanın yürütüldüğü Uzaktan Eğitim Merkezi'nde, bir günde, 8 kişinin 4'er

saat, uzaktan eğitim yönetim sistemi üzerine çalıştığı görülmektedir. Toplamda 32 saati bulan bu çalışma, haftada 160 saate (32 x 5), ayda 640 saate (160 x 4) ve yılda 7680 saate (640 x 12) ulaşmaktadır. t-Testi sonuçlarına göre, bu saatler metodoloji ile birlikte, günde yaklaşık 24 saate (32 x 1.97 / 2.66), ayda 118 saate (160 x 1.97 / 2.66) ve yılda 474 saate (640 x 1.97 / 2.66) düşmektedir. Türkiye'deki büyük şehirlerde, yazılım uzmanlarının saatlik ücretinin, ortalama 100 TL (55 $) olduğu öngörüldüğü durumda (önde gelen üç büyük yazılım şirketinden alınan ortalama değer), metodoloji ile birlikte, tasarruf miktarı, günlük 800 TL ((32 x 100) – (24 x100)), haftalık 4.000 TL, aylık 16.000 TL ve yıllık 192.000 TL olarak hesaplanabilmektedir.

BÖLÜM 5. METODOLOJİ'NİN ÖRNEK İŞ SÜREÇLERİNE UYGULANMASI

Bu bölümde, öncelikle İş Süreçlerine SOA Uygulama Metodolojisi'nin uygulanacağı birimin yapısı hakkında bilgi verilmiş ve ardından metodolojinin adımları olan, kavramsal servisleri belirleme, eski sistem analizi, dijital servisler için strateji belirleme ve dijital servisleri yönetmenin, sırasıyla nasıl uygulanabileceği gösterilmiştir. Son olarak ta uygulamanın çıktıları olarak önemli ara yüzlerinden bir kısmı işlevleri ile birlikte tanıtılmıştır.

5.1. Uygulama Alanının Özellikleri

Elinizdeki tez kapsamında geliştirilen ve bölüm 4'te ayrıntılı bir şekilde açıklanan 'İş Süreçlerine SOA Uygulama Metodolojisi, bu bölümde bir üniversitenin uzaktan eğitim merkezinin iş süreçlerine uygulanmıştır.

Üzerinde uygulama yapılan uzaktan eğitim merkezinin içeriği ve yapısı aşağıda açıklandığı şekildedir [97];

Söz konusu uzaktan eğitim merkezi 1997 yılından bu yana sürdürdüğü uzaktan eğitim çalışmalarında bugün bulunduğu nokta itibariyle, 6 Önlisans, 8 Lisans, 8 Yüksek Lisans, 1 Lisans Tamamlama ve 2 Sertifika programında kayıtlı yaklaşık 8000 öğrencisi ile Türkiye'de bu konuda en önde gelen konumdadır.

Uzaktan eğitim merkezinde, kurulduğunda sadece bir arayüz üzerinden sade bir sayfa ile sunulan ilk online dersinin yayınlandığı sistemden, bugün dünya çapında birçok gelişmiş sistemde olmayan bir öğretim yönetim sistemi yazılımı kullanılmaktadır. Bu yazılımın geliştirilmesi çalışmaları ise sürekli olarak devam etmektedir. Bu aşamada ülkemiz üniversitelerinin uzaktan eğitim ve yönetim süreçlerini kolaylaştırmak, daha

kolay yönetilebilir, takip ve kontrol edilebilir hale getirmek için merkezi olarak yönetilebilen, kolay kullanılabilen, otomasyon sistemleriyle uyumlu, mobil teknolojileri destekleyen bir uzaktan eğitim yönetim sisteminin tasarımı, geliştirilmesi amaçlanmaktadır. Uzaktan eğitim yönetim sistemine SOA yaklaşımının uyarlanması ise değinilen tüm bu hedeflerin gerçekleştirilmesinde çok önemli bir rol oynayacaktır. Bunun sağlanabilmesi için de bölüm 4'te açıklanan metodoloji büyük bir kolaylık sağlayacaktır.

Uzaktan eğitim merkezinin sahip olduğu uzaktan eğitim yönetim sistemi, Türkiye'deki yüksek öğrenim kurumlarının ihtiyaçlarını karşılamak üzere geliştirilmiştir. Uzaktan eğitim yönetim sistemi geliştirilirken, üniversitelerin uzaktan eğitim konusunda ihtiyacı olabilecek eğitim, yönetim, içerik, destek ve pazarlama anlamındaki tüm istekleri göz önüne alınmıştır. Uzaktan eğitim yönetim sisteminin sağladığı faydalar kısaca şu şekilde özetlenebilir;

- Tek bir sistem ve yazılım ile içerik, sanal sınıf ve eğitim yönetim sisteminin yönetilebilmesi.
- Üniversite organizasyonuna uygun (fakülte, yüksekokul, uzaktan eğitim merkezi, öğrenci işleri) olması sebebi ile sorumlulukların paylaştırılması ve görevlerin dağıtımının yapılması.
- Sadece eğitim sisteminin değil aynı zamanda eğitim programının web sitesinin de yönetilmesi.
- Staj, belge talebi, not durum belgesi ve diğer öğrenci işleri gibi işlemler için, sistem üzerinden uzaktan eğitim öğrencilerine destek verilebilmesi.
- Entegre içerik üretim aracı ile derslerin çevrimiçi olarak geliştirilmesi.
- Onbinlerce öğrencinin sistemi kullanacağı varsayımı ile yönetimsel kolaylıklar sağlaması
- Sürekli gelişime açık olması

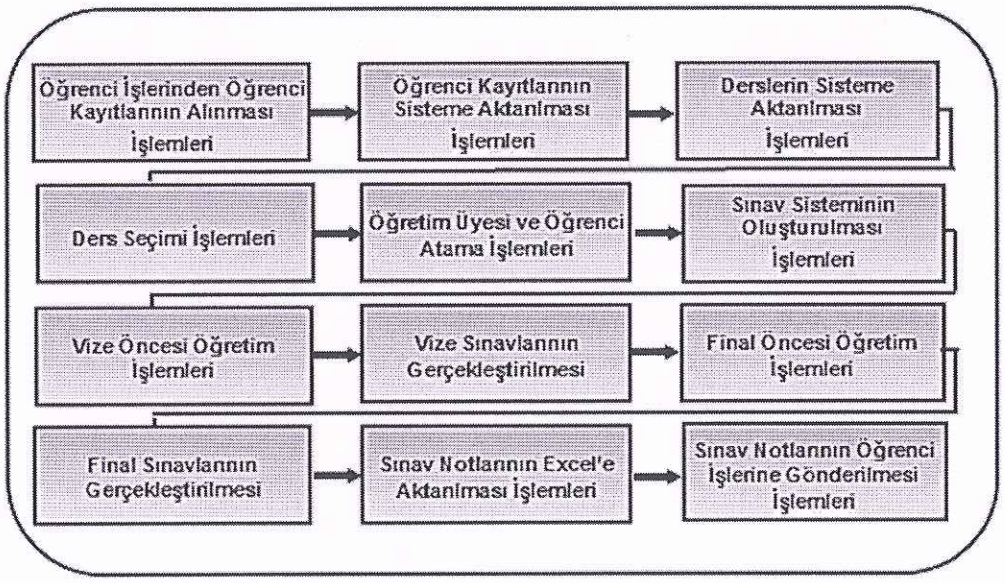
Böylelikle, yukarıda özellikleri aktarılan uzaktan eğitim merkezine, bölüm 4'te anlatılan metodolojinin adım adım uygulanması aşamasına gelinmiştir. Bu aşamalar bölüm 4'te açıklanmış olmakla birlikte uygulamanın da gerçekleştirilmesi ile birlikte oldukça somut bir hal kazanacaklardır.

5.2. Kavramsal Servisleri Belirleme

Bu bölümde, metodolojinin uygulanacağı Uzaktan Eğitim Merkezi için, uygulama kapsamında ihtiyaç duyulacak olan kavramsal servislerin belirlenmesi işlemi gerçekleştirilmiştir. Bu işlem için öncelikle mevcut iş süreçleri sonra da hedef iş süreçleri modellenmiş ve bu modeller ışığında kavramsal servisler belirlenmiştir.

5.2.1. Mevcut iş süreç modelinin oluşturulması

Mevcut iş süreç modelinin oluşturulması işlemi manuel olarak gerçekleştirilebileceği gibi daha doğru ve süreçlere özel ayrıntıların da dikkate alınabileceği süreç madenciği gibi teknik bir yönteme başvurularak ta oluşturulabilir. Uygulamanın yapılacağı uzaktan eğitim merkezinin, herhangi bir notasyon tekniğine dikkat edilmeksizin ve manuel olarak oluşturulan iş akış modeli Şekil 5.1'de gösterildiği şekildedir. Bu akış modeli temelinde oluşturulan iş süreç modeli ise Ek. C bölümünde bulunmaktadır. Şekil üzerindeki oniki tane ana sürecin her biri birçok alt süreçten oluşmaktadır. Tezin kapsamı dikkate alınarak bu süreçler detaylandırılmamıştır ancak her süreç hakkında açıklayıcı bilgiler verilmiştir.



Şekil 5.1. Uzaktan Eğitim Merkezi'nin Mevcut İş Akış Modeli

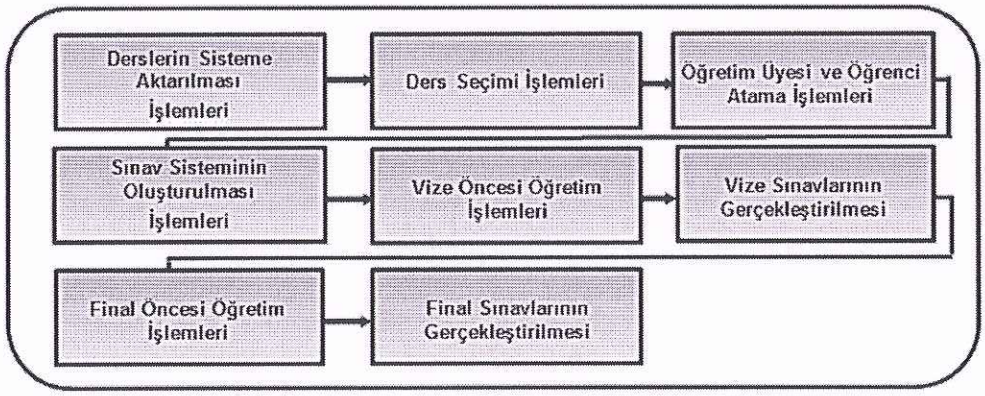
Uzaktan Eğitim Merkezi'nin mevcut iş süreçlerinin açıklamaları aşağıdaki gibidir;

- Öğrenci İşlerinden Öğrenci Kayıtlarının Alınması: Uzaktan Eğitim Merkezi öğrencileri, kayıtlarını merkezin bağlı bulunduğu üniversitenin öğrenci işlerine yapmaktadırlar. Ancak Uzaktan Eğitim Merkezi, öğrencilerin kayıt bilgilerine her zaman ihtiyaç duymaktadır. Bu sorunun çözümü için öğrencilerin kayıt bilgileri öğrenci işlerinden temin edilmektedir.
- Öğrenci Kayıtlarının Sisteme Aktarılması: Bu aşamada öğrenci işlerinden temin edilen öğrenci kayıt bilgileri, öğretim yönetim sistemine aktarılmaktadır.
- Derslerin Sisteme Aktarılması: Bu aşamada dönem içinde bölümlere özel yürütülecek dersler öğretim yönetim sistemine aktarılmaktadır.
- Ders Seçimi İşlemleri: Uzaktan Eğitim Merkezi öğrencileri, zorunlu olarak gördüğü dersler ile birlikte belirli bir sayıdaki dersi de seçmeli olarak görmektedirler. Bu derslerin seçimi ise bu aşamada gerçekleştirilmektedir.
- Öğretim Üyesi ve Öğrenci Atama: Öğretim yönetim sistemi üzerinden hangi öğretim üyesinin hangi öğrencilere ders vereceği ile ilgili atama işlemleri bu aşamada gerçekleştirilmektedir.
- Sınav Sisteminin Oluşturulması: Öğretim üyeleri bu aşamada eğitim dönemi içinde uygulayacakları sınav sistemini düzenlemektedirler. Sınav, proje ve ödev yüzdelerini atama işlemi bu aşamada gerçekleştirilmektedir.
- Vize Öncesi Öğretim: Vize sınavları öncesi öğretimin yürütülmesi ile ilgili tüm süreçler bu aşamada gerçekleştirilmektedir.
- Vize Sınavlarının Gerçekleştirilmesi: Dönem ortasına yakın bir periyotta vize öncesi öğretim boyunca işlenen derslerin sınanması için vize sınavları uygulanmaktadır. Bu işlemler ile ilgili tüm süreçler bu aşamada gerçekleştirilmektedir.
- Final Öncesi Öğretim: Vize sınavları sonrası öğretimin yürütülmesi ile ilgili tüm süreçler bu aşamada gerçekleştirilmektedir.
- Final Sınavlarının Gerçekleştirilmesi: Dönem sonunda öğretim dönemi boyunca işlenen derslerin sınanması için final sınavları uygulanmaktadır. Bu işlemler ile ilgili tüm süreçler bu aşamada gerçekleştirilmektedir.

- Sınav Notlarının Excel'e Aktarılması: Dönem boyunca uygulanan sınav, proje ve ödevler bu aşamada bir excel dosyasına aktarılır. Bu notlar bir vize ve bir de final notu belirlenecek şekle getirilir.
- Sınav Notlarının Öğrenci İşlerine Gönderilmesi: Bu aşamada, bir önceki aşamada elde edilen vize ve final notları, kaydedilmek üzere ve bir excel dosyası olarak öğrenci işlerine gönderilir.

5.2.2. Hedef iş süreç modelinin oluşturulması

Uzaktan Eğitim Merkezi yetkililerinin koordinatörlüğünde mevcut süreçlerin daha işlevsel bir hale getirilmesi hedeflenerek, hedef iş akış modeli Şekil 5.2'de gösterildiği şekilde oluşturulmuştur. Bu akış modeli temelinde oluşturulan iş süreç modeli ise Ek. C bölümünde bulunmaktadır. Hedef iş süreçleri, sadece manuel yolla oluşturulabilmektedir zira bu süreçlere dair herhangi bir kayıt bulunmamaktadır ve böylelikle başka bir teknik yönteme başvurulamamaktadır.



Şekil 5.2. Uzaktan Eğitim Merkezi'nin Hedef İş Akış Modeli

Uzaktan Eğitim Merkezi'nin hedef iş süreçlerine bakıldığında, bu süreçlerin mevcut iş süreçlerinden daha kısa oldukları görülebilir. Bunun temel sebebi ise mevcut iş süreçlerinde bulunan, Öğrenci İşlerinden Öğrenci Kayıtlarının Alınması, Öğrenci Kayıtlarının Sisteme Aktarılması, Sınav Notlarının Excel'e Aktarılması, Notların Öğrenci İşlerine Gönderilmesi süreçlerine, yeni oluşturulacak sistemde gerek duyulmayacak olunmasıdır. Yeni sistemde, öğrenci işleri ve uzaktan eğitim

merkezinde bulunan iki ayrı veri bankası tek bir veri bankasına indirgenmiş ve yukarıda ifade edilen dört sürece ait fonksiyonların tümü her iki birimin de kullanabileceği servislerde yerlerini almışlardır. Böylelikle, söz konusu dört sürecin öğrenci işlerinin süreçleri arasında yer almaları gerektiğinden ve uzaktan eğitim merkezinin ana süreçleri arasında bulunmalarının gereksizliğinden, bu süreçlere uzaktan eğitim merkezinin hedef süreçleri arasında yer verilmemiştir.

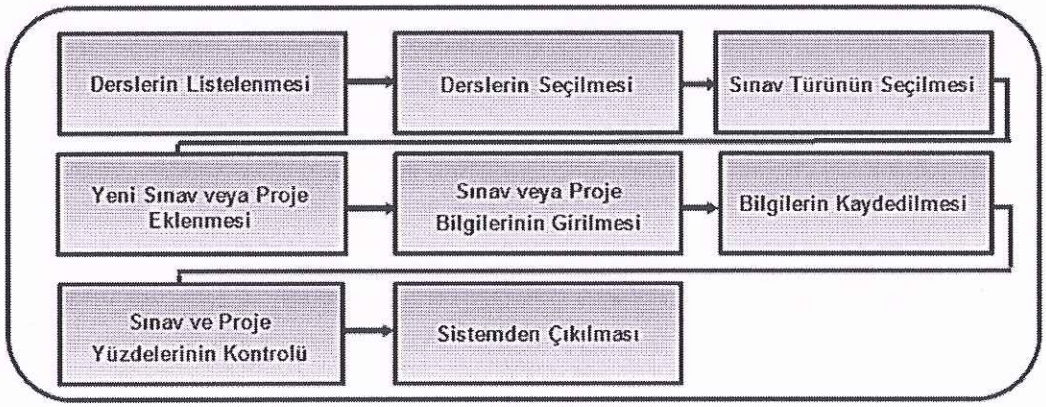
Uzaktan Eğitim Merkezi'nin hedef iş süreçlerinin açıklamaları aşağıdaki gibidir;

- Derslerin Sisteme Aktarılması: Bu aşamada, dönem içinde bölümlere özel yürütülecek dersler öğretim yönetim sistemine aktarılmaktadır.
- Ders Seçimi İşlemleri: Uzaktan Eğitim Merkezi öğrencileri zorunlu olarak gördüğü dersler ile birlikte belirli bir sayıdaki dersi de seçmeli olarak görmektedirler. Bu derslerin seçimi ise bu aşamada gerçekleştirilmektedir.
- Öğretim Üyesi ve Öğrenci Atama: Öğretim yönetim sistemi üzerinden hangi öğretim üyesinin hangi öğrencilere ders vereceği ile ilgili atama işlemleri bu aşamada gerçekleştirilmektedir.
- Sınav Sisteminin Oluşturulması: Öğretim üyeleri bu aşamada eğitim dönemi içinde uygulayacakları sınav sistemini düzenlemektedirler. Sınav, proje ve ödev yüzdelerini atama işlemi bu aşamada gerçekleştirilmektedir. Bu süreç, mevcut iş süreçlerindeki Sınav Sistemi'nin Oluşturulması sürecinden farklı bir yapıda düşünülmüştür. Öğrenci İşleri'nin son olarak bir vize ve bir de final notu talep ettiği bilindiği için bu ayarlamanın baştan yapılmasının daha uygun olabileceği düşünülmüştür. Bu aşamada öğretim üyesi en başta vize veya final sınav türünü seçerek, dönem içinde gerçekleştireceği sınav, proje ve ödev yüzdelerini direk olarak seçtiği sınav türünün üzerinde gerçekleştirecektir. Sonuç olarak vize ve final notları dönem sonunda otomatik olarak elde edilebilecektir.
- Vize Öncesi Öğretim: Vize sınavları öncesi öğretimin yürütülmesi ile ilgili tüm süreçler bu aşamada gerçekleştirilmektedir.
- Vize Sınavlarının Gerçekleştirilmesi: Dönem ortasına yakın bir periyotta vize öncesi öğretim boyunca işlenen derslerin sınanması için vize sınavları

uygulanmaktadır. Bu işlemler ile ilgili tüm süreçler bu aşamada gerçekleştirilmektedir.

- Final Öncesi Öğretim: Vize sınavları sonrası öğretimin yürütülmesi ile ilgili tüm süreçler bu aşamada gerçekleştirilmektedir.
- Final Sınavlarının Gerçekleştirilmesi: Dönem sonunda, öğretim dönemi boyunca işlenen derslerin sınanması için final sınavları uygulanmaktadır. Bu işlemler ile ilgili tüm süreçler bu aşamada gerçekleştirilmektedir.

Uygulamanın anlaşılabilir olabilmesi ve amacına ulaşabilmesi için, hedef iş süreçlerinden Sınav Sistemi'nin Oluşturulması, tez kapsamını aşmayacak şekilde detaylandırılarak ele alınmıştır. Sınav Sistemi'nin Oluşturulması sürecine ait alt iş akış modeli Şekil 5.3'te görselleştirildiği şekilde tasarlanmıştır. Bu akış modeli temelinde oluşturulan iş süreç modeli ise Ek. C bölümünde bulunmaktadır.



Şekil 5.3. Sınav Sistemi'nin Oluşturulması Sürecine Ait Alt İş Akış Modeli

Sınav sisteminin oluşturulması sürecine ait alt süreçlerin açıklamaları aşağıdaki gibidir;

- Derslerin Listelenmesi: Bu aşamada sistem tarafından, öğretim üyesinin dönem içinde verdiği derslerin bir listesi oluşturulacaktır.

- Derslerin Seçilmesi: Öğretim üyesi, bir önceki aşamada oluşturulan ders listesi üzerinden, sınav sistemini oluşturmak istediği dersin seçimini bu aşamada gerçekleştirecektir.
- Sınav Türünün Seçilmesi: Öğretim üyesi, bir önceki aşamada seçtiği dersin, üzerinde ayarlama yapacağı sınav türünü (vize veya final) bu aşamada belirleyecektir.
- Yeni Sınav veya Proje Eklenmesi: Öğretim üyesi bu aşamada, seçtiği sınav türü üzerine yeni sınav, proje ve ödevler ekleyebilecektir.
- Sınav veya Proje Bilgilerinin Girilmesi: Eklenen sınav proje veya ödevlere özel bilgiler ve bunların sınav türlerine olan yüzdelik etkileri bu aşamada belirlenebilecektir.
- Bilgilerin Kaydedilmesi: Sınavlarla ilgili girilen her türlü bilginin kayıt edilmesi bu aşamada gerçekleştirilecektir.
- Sınav ve Proje Yüzdelerinin Kontrolü: Sistem tarafından, sınav türlerine özel girilen yüzdelerin toplamının yüze eşit olup olmadığının kontrolünün yapılması bu aşamada gerçekleştirilecektir.
- Sistemden Çıkılması: Yukarıda açıklanan tüm süreçlerin sonunda sınav sistemi oluşturulduktan sonra, öğretim üyesi bu aşamada sistemden çıkış yapabilecektir.

5.2.3. Kavramsal servislerin belirlenmesi

Kavramsal servisler belirlenirken bölüm 4.2'de açıklandığı şekilde, öncelikle ihtiyaç duyulan fonksiyonlar belirlenmiştir. Sonra merkezi mantık yaklaşımı kapsamında bu fonksiyonların sınıflandırılması gerçekleştirilmiştir.

Gerçekleştirilecek uygulamada hedef iş süreç modeli temelinde aşağıdaki fonksiyonlara ihtiyaç duyulacağı belirlenmiştir. Fonksiyonlardan bir kısmına birden fazla süreçte ihtiyaç duyulmasına rağmen, aşağıdaki listede her fonksiyon sadece bir kez listelenmiştir.

Fonksiyon Listesi:

- kontrol(): Login sırasında kullanıcı kontrolü için kullanılacaktır.
- sinavOlustur(): Öğretim görevlisinin seçtiği ders için proje veya sınav tanımlamasını sağlar.
- idIleSinavBul(): Tanımlanmış bir sınavın id'si ile getirilmesini sağlar.
- sinavSil(): ID'si verilen sıavın silinmesi için kullanılacaktır.
- sinavGuncelle(): Tanımlanmış bir sınav üzerinde değişiklik yapmak için kullanılır.
- derseAitSinavlariGetir(): Bir ders için belli bir yılda tanımlanmış sınavların tamamını getirmek için kullanılır.
- ogrenciyeAitButunSinavSonuclariniGetir(): Öğrenci ve sınav ID'si ile sınav sonucunun getirilmesi için kullanılır.
- sinavaAitButunSinavSonuclariniGetir(): Belli bir yılda yapılan bir sınavın tüm sonuçlarını getirmek için kullanılır.
- dersOlustur(): Ders tanımlamak için kullanılır.
- idIleDersBul(): Dersin ID parametresi ile getirilmesi için kullanılır.
- dersSil(): Dersin silimesini sağlar.
- dersGuncelle(): Ders üzerinde güncelleştirme yapmak için kullanılır.
- butunDersleriGetir(): Veri tabanındaki tüm derslerin listesine ulaşmak için kullanılacaktır.
- profesoreAitDersleriGetir(): Öğretim görevlisinin belli bir yılda verdiği derslerin listesini verir.
- ogrenciyeAitDersleriGetir(): Bir öğrencinin belli bir yılda aldığı derslerin listesini verir.
- idIleSinavSonucuBul(): Sınav sonucunun sınavın ID parametresi ile getirilmesini sağlar.
- profesoreDersEkle(): Öğretim görevlisine ders atamak için kullanılır.
- profesordenDersCikar(): Öğretim görevlisine atanmış bir dersin silinmesi için kullanılır.
- profesordenButunDersleriCikar(): Öğretim görevlisine atanmış tüm derslerin silinmesi için kullanılır.

- kullanıcıAdiIleOgrenciGetir(): Kullanıcı adı ile öğrenci nesnesine ulaşmak için kullanılır.
- ogrenciyeDersEkle(): Öğrenciye yeni bir dersi almak üzere atamak için kullanılır.
- ogrencidenDersCikar(): Öğrencinin aldığı bir dersi kaldırmak için kullanılır.
- ogrencidenButunDersleriCikar(): Öğrencinin aldığı tüm derslerin silinmesi için kullanılır.
- sinavSonucuOlustur(): Bir öğrenciye ait sınav sonucun kaydedilmesi sağlar.
- sinavSonucuSil(): Girilmiş bir sınav sonucunun silinmesini için kullanılır.
- sinavSonucuGuncelle(): Girilmiş sınav sonucunda değişiklik yapılır.
- profesorOlustur(): Yeni bir öğretim görevlisi tanımlamak için kullanılır.
- kullaniciAdilleProfesorGetir(): Kullanıcı adı ile öğretim üyesine ulaşmak için kullanılır.
- idIleProfesorGetir(): ID ile öğretim üyesine ulaşmak için kullanılır.
- ogrenciGuncelle(): Öğrenci nesnesi üzerinde değişiklik yapmak için kullanılır.
- profesorSil(): Tanımlanmış bir öğretim üyesini silmek için kullanılır.
- profesorGuncelle(): Öğretim üyesi bilgilerinde değişiklik yapmak için kullanılır.
- butunProfesorleriGetir(): Bu fonksiyon veri tabanındaki tüm öğretim üyelerinin listesine ulaşabilmek için kullanılacaktır.
- ogrenciOlustur(): Öğrenci eklemek için kullanılır.
- idIleOgrenciGetir(): ID ile öğrenci nesnesine ulaşmak için kullanılır.
- ogrenciSil(): Tanımlanmış bir öğrenciyi silmek için kullanılır.
- butunOgrencileriGetir(): Bir dersi belli bir yılda alan öğrenci nesnesine ulaşmak için kullanılır.

Uygulama için ihtiyaç duyulan ve yukarıdaki şekilde listelenen fonksiyonlar aşağıdaki servis isimleri çerçevesinde sınıflandırılıp, kavramsallaştırılabilirler. Bu aşamadan sonra kavramsal servisler elde edilmekte ve metodoloji'nin ilk adımı tamamlanmaktadır.

Sınav servisi:

- sinavOlustur(): Öğretim görevlisinin seçtiği ders için proje veya sınav tanımlamasını sağlar.
- idIleSinavBul(): Tanımlanmış bir sınavın id'si ile getirilmesini sağlar.
- sinavSil(): ID'si verilen sıavın silinmesi için kullanılacaktır.
- sinavGuncelle(): Tanımlanmış bir sınav üzerinde değişiklik yapmak için kullanılır.
- derseAitSinavlariGetir(): Bir ders için belli bir yılda tanımlanmış sınavların tamamını getirmek için kullanılır.

Sınav sonuç servisi:

- sinavSonucuOlustur(): Bir öğrenciye ait sınav sonucun kaydedilmesi sağlar.
- idIleSinavSonucuBul(): Sınav sonucunun sınavın ID parametresi ile getirilmesini sağlar.
- sinavSonucuSil(): Girilmiş bir sınav sonucunun silinmesini için kullanılır.
- sinavSonucuGuncelle(): Girilmiş sınav sonucunda değişiklik yapılır.
- ogrenciyeAitButunSinavSonuclariniGetir(): Öğrenci ve sınav ID'si ile sınav sonucunun getirilmesi için kullanılır.
- sinavaAitButunSinavSonuclariniGetir(): Belli bir yılda yapılan bir sınavın tüm sonuçlarını getirmek için kullanılır.

Ders servisi:

- dersOlustur(): Ders tanımlamak için kullanılacaktır.
- idIleDersBul(): Dersin ID parametresi ile getirilmesi için kullanılacaktır.
- dersSil(): Dersin silimesini sağlayacaktır.
- dersGuncelle(): Ders üzerinde güncellerme yapmak için kullanılacaktır.
- butunDersleriGetir(): Veri tabanındaki tüm derslerin listesine ulaşmak için kullanılacaktır.

- profesoreAitDersleriGetir(): Öğretim görevlisinin belli bir yılda verdiği derslerin listesini getirecektir.
- ogrenciyeAitDersleriGetir(): Bir öğrencinin belli bir yılda aldığı derslerin listesini getirecektir.
- profesoreDersEkle(): Öğretim görevlisine ders atamak için kullanılacaktır.
- profesordenDersCikar(): Öğretim görevlisine atanmış bir dersin silinmesi için kullanılacaktır.
- profesordenButunDersleriCikar(): Öğretim görevlisine atanmış tüm derslerin silinmesi için kullanılacaktır.
- ogrenciyeDersEkle(): Öğrenciye yeni bir dersi atamak için kullanılacaktır.
- ogrencidenDersCikar(): Öğrencinin aldığı bir dersi kaldırmak için kullanılacaktır.
- ogrencidenButunDersleriCikar(): Öğrencinin aldığı tüm derslerin silinmesi için kullanılacaktır.

Profesör servisi:

- profesorOlustur(): Yeni bir öğretim görevlisi tanımlamak için kullanılacaktır.
- kullaniciAdilleProfesorGetir(): Kullanıcı adı ile öğretim üyesine ulaşmak için kullanılacaktır.
- idIleProfesorGetir(): ID ile öğretim üyesine ulaşmak için kullanılacaktır.
- profesorSil(): Tanımlanmış bir öğretim üyesini silmek için kullanılacaktır.
- profesorGuncelle(): Öğretim üyesi bilgilerinde değişiklik yapmak için kullanılacaktır.
- butunProfesorleriGetir(): Bu fonksiyon veri tabanındaki tüm öğretim üyelerinin listesine ulaşabilmek için kullanılacaktır.

Öğrenci servisi:

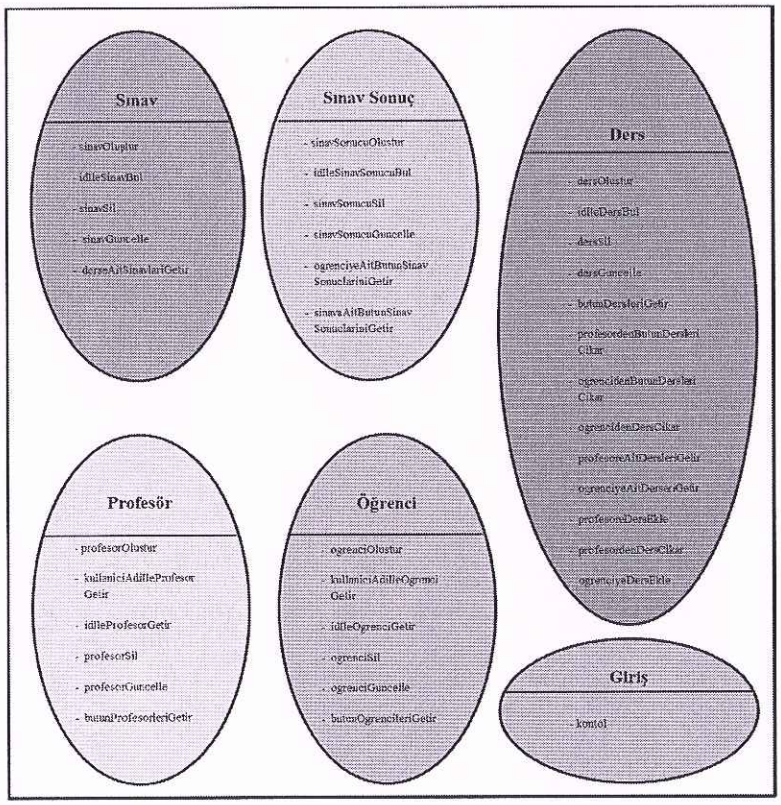
- ogrenciOlustur(): Öğrenci eklemek için kullanılacaktır.
- kullanıcıAdiIleOgrenciGetir(): Kullanıcı adı ile öğrenci nesnesine ulaşmak için kullanılacaktır.

- idIleOgrenciGetir(): ID ile öğrenci nesnesine ulaşmak için kullanılacaktır.
- ogrenciSil(): Tanımlanmış bir öğrenciyi silmek için kullanılacaktır.
- ogrenciGuncelle(): Öğrenci nesnesi üzerinde değişiklik yapmak için kullanılacaktır.
- butunOgrencileriGetir(): Bir dersi belli bir yılda alan öğrenci nesnesine ulaşmak için kullanılacaktır.

Giriş servisi:

- kontrol(): Login sırasında kullanıcı kontrolü için kullanılacaktır.

Bu uygulamada altı adet kavramsal servis tasarlanmıştır. Bu servisler Giriş, Sınav, Sınav Sonucu, Ders, Öğrenci ve Profesör servisleridir. Söz konusu kavramsal servisler fonksiyonları ile birlikte Şekil 5.4'teki gibi görsel hale getirilmişlerdir.



Şekil 5.4. Uygulamada Kullanılacak Kavramsal Servisler

5.3. Eski Sistem Analizi

Uygulamanın gerçekleştirileceği Uzaktan Eğitim Merkezi'nin öğretim yönetim sisteminin, SOA yaklaşımının uygulanacağı yeni sistem kapsamında kullanılabilecek kısımlarının belirlenebilmesi için analiz edilmesi gerekmektedir. Bu işlem için öncelikle bir analiz metodu belirlenmiş, ardından belirlenen metod çerçevesinde analizler yürütülmüş ve son olarak ta bu analizlerin değerlendirilmesi yapılmıştır.

5.3.1. Analiz metodu belirleme

Uygulamanın gerçekleştirileceği Uzaktan Eğitim Merkezi'nin, alanında Türkiye'de önde gelen, gelişmiş bir öğretim yönetim sistemine sahip olduğu bölüm 5.1'de belirtilmiştir. Bu tez kapsamında Uzaktan Eğitim Merkezi öğretim yönetim sisteminin SOA yaklaşımına uygun yeni bir sisteme dönüştürülmesi planlandığı için, merkezin sahip olduğu mevcut öğretim yönetim sistemi eski sistem olarak analiz edilecektir. Buradaki amacın eski sistem üzerindeki yararlı olabilecek bölümlerin yeni sisteme kazandırılması olduğu bölüm 2.5'te açıklanmıştır.

Analiz metodu belirlenirken ilk yapılması gereken eski sisteme dair bir belgelendirmenin yapılmış olup olmadığının araştırılmasıdır. Yapılan araştırmada Uzaktan Eğitim Merkezi'nin, Türkiye'de birçok kurumda olduğu gibi, öğretim yönetim sistemi için bir belgelendirme yapmamış olduğu belirlenmiştir. Sistem geliştirenlerin işten ayrılmaları ve yeni düzenlemelerin gerekliliği gibi birtakım önem teşkil eden durumlarda belgelendirme yoksunluğunun çok önemli mali kayıplara yol açabileceği de daha önceki bölümlerde belirtilmiştir.

Uzaktan Eğitim Merkezi'nde bir belge takibi yolu ile eski sistemin analizi yürütülemeyeceğinden, analiz işleminin, merkezin mevcut iş süreçleri üzerinden gerçekleştirilmesi gerekmektedir. İş süreçleri üzerinden yürütülmesi gereken eski sistem analizi, bölüm 5.2.1'de tasarlanan iş süreç modeli üzerinden gerçekleştirilebilecektir.

5.3.2. Analizin yürütülmesi

Uzaktan Eğitim Merkezi'nin Şekil 5.1'de görüldüğü şekilde tasarlanan mevcut iş süreç modelindeki süreçlerinden, Öğrenci İşlerinden Öğrenci Kayıtlarının Alınması, Öğrenci Kayıtlarının Sisteme Aktarılması, Sınav Notlarının Excel'e Aktarılması ve Sınav Notlarının Öğrenci İşlerine Gönderilmesine yeni sistem dahilinde ihtiyaç duyulmayacak olması sebebi ile bu süreçlerin ve barındırdıkları fonksiyonların analizine gerek kalmamaktadır.

Derslerin Sisteme Aktarılması, Ders Seçimi İşlemleri, Öğretim Üyesi ve Öğrenci Atama, Vize Öncesi Öğretim, Vize Sınavlarının Gerçekleştirilmesi, Final Öncesi Öğretim ve Final Sınavlarının Gerçekleştirilmesi süreçleri, barındırdıkları fonksiyonlar itibari ile birebir örtüşmektedir. Bununla birlikte fonksiyonların girdi ve çıktıları da örtüşmektedir.

Sınav Sisteminin Oluşturulması süreci ise aynı zamanda Uzaktan Eğitim Merkezi'nin hedef iş süreçleri arasında bulunmasına rağmen, tamamıyla farklı bir yapıdadırlar. Bu süreçte not hesaplama işlemi farklı olarak gerçekleştirilmektedir. Bu farkların açıklamaları bölüm 5.2.2'de detaylı olarak açıklanmıştır. Dolayısıyla sürecin barındırdığı fonksiyonlar da birbirinden tamamen farklıdır.

5.3.3. Analizin değerlendirilmesi

Bölüm 5.3.2'de yürütülen analize göre, mevcut iş süreç modelinde görülen ve eski sistem tarafından otomatikleştirilmiş olan yedi sürece ait fonksiyonların tümü, işlevleri ve girdi ve çıktı parametreleri itibari ile SOA yaklaşımı çerçevesinde geliştirilen sistemde kullanılabilecek durumdadır.

Sınav Sisteminin Oluşturulması sürecine ait fonksiyonların tamamının işlevleri ve girdi ve çıktı parametreleri farklı yapıdadır. Bu fonksiyonlar yeni sistemde kullanılabilecek durumda değildirler. Bu durumda yeni sistem için tasarlanan Sınav Sisteminin Oluşturulması sürecinin fonksiyonlarının yeniden geliştirilmesi gerekmektedir.

5.4. Dijital Servisler İçin Strateji Belirleme

Bu bölümde, bir sonraki bölümde geliştirilecek olan dijital servisler için bir strateji seçimi gerçekleştirilmiştir. Bu işlem için öncelikle belirli stratejilerin uygulanma potansiyelleri incelenmiş, ardından söz konusu stratejiler arasında teknik ve finansal bir analiz yürütüldükten sonra bölüm 4.4.3'te açıklanan seçim yöntemiyle bu stratejilerden bir tanesinin seçilmesi sağlanmıştır.

5.4.1. Stratejilerin belirlenmesi

Dijital servisler için stratejilerin belirlenmesi durumuna geldiğimiz bu aşamada, oluşturulmasını arzuladığımız SOA yaklaşımlı sistem ve eski sisteme dair veri ve analizlerin elimizde ve erişilebilir durumda bulunması gerekmektedir. Bu aşamanın kavramsal yapı ile teorik yapı arasında bir köprü görevi gördüğü ve verimli bir SOA için bu aşamaya kadar yürütülen aşamaların da aynı ölçüde verimli olarak gerçekleştirilmesi gerektiği daha önce de belirtilmiştir.

Elimizdeki sonuçlara göre, oluşturmak istediğimiz SOA ve eski sistemin SOA'ya entegrasyonu için teorik olarak bölüm 2.5.1'de açıklanan üç yaklaşımın her biri ayrı birer strateji olarak belirlenebilir. Dolayısıyla bu aşamada yeni sistemimiz için, yeniden geliştirme, paketleme ve göç ettirme adı altında üç ayrı strateji geliştirilmiştir. Şimdi, belirtilen bu stratejilerin her birinin uygulamamızda ne anlama geldiği açıklanacaktır.

Birinci stratejimiz olan yeniden geliştirme stratejisine göre, sistemimiz en baştan, ayrıntılı olarak SOA yaklaşımına göre tasarlanıp geliştirilecek ve tekrar işleve konulacaktır. İkinci stratejimiz olan paketlemeye göre ise mevcut öğretim yönetim sistemi üzerinde bulunan ve yeni sistemimizde kullanılabilecek durumda olan bileşenler paketlenerek yeni sistemimiz için erişilebilir bir hal kazanacaklardır. Son stratejimiz olan göç ettirmeye göre ise mevcut öğretim yönetim sistemi üzerindeki bileşenler, yeni bileşenler ile yer değiştirecek ancak orijinal veri ve fonksiyonellik sabit tutulacaktır. Bu sayede ortamın daha esnek bir hal kazanması sağlanacaktır.

5.4.2. Statejiler arası teknik ve finansal analiz

Bu bölümde, bir önceki bölümde belirlenen stratejier arasında teknik ve finansal bir analiz yürütülecektir. Buradaki amaç bir sonraki karar aşamasını mümkün olabildiğince kolaylaştırmaktır. Aşağıda sıra ile her üç stratejinin finansal ve teknik analizi yapılmıştır.

Birinci stratejimiz olan yeniden geliştirme stratejisine göre, sistemimiz en baştan, ayrıntılı olarak SOA yaklaşımına göre tasarlanıp geliştirilecek ve tekrar işleve konulacaktır. Buradaki en büyük risk ise sistemin tamamen yeni olacağı ve deneme süresinin kısıtlı olacağıdır. Bununla birlikte sistemin en baştan ve tamamıyla yeniden geliştirilmesi oldukça uzun bir zaman gerektirecek ve birim için yüklü bir maliyet oluşturacaktır.

İkinci stratejimiz olan paketleme stratejisine göre ise, mevcut öğretim yönetim sistemi bileşenleri paketlenerek yeni ve erişilebilir bir hal kazanacaklardır. Teknik açıdan oldukça maliyetli bir işlem olmasına rağmen, mali açıdan büyük bir yük getirmeyecek olan bir stratejidir. Ancak paketlenen bileşenler birçok zaman sistemin en baştan geliştirildiği ve her ayrıntısına dikkat edilecek olan yeniden geliştirme stratejisinde olduğu kadar işlevsel olmayacaktır.

Üçüncü stratejimiz olan göç ettirmeye göre ise, mevcut öğretim yönetim sistemi bileşenlerinin, yeni geliştirilmiş olan bileşenler ile yerleri değiştirilerek, ortamın daha esnek bir hal kazanması amaçlanmaktadır. Bu işlem gerçekleştirilirken orijinal veri ve fonksiyonların korunmasına özen gösterilecektir. Teknik ve mali açıdan diğer iki stratejinin ortasında gösterilebilecek bu strateji, eski sistemin, yeni bileşenleri teknik olarak kaldıramadığı durumlarda yeniden geliştirme stratejisine göre dezavantajlı bir strateji olmaktan kaçamamaktadır.

5.4.3. Strateji seçimi

Uzaktan Eğitim Merkezi'nin SOA'ya geçişi için belirlenen ve bir önceki bölümdeki analizler ve bölüm 4.4.3'te açıklanan yöntem doğrultusunda puanlanan stratejilere dair sonuçlar Tablo 5.1'de ifade edildiği şekildedir. Değerlendirme için, Finansal Maliyet, Teknik Maliyet ve Kullanım Süresi olmak üzere üç kriter kullanılmıştır. Bu kriterlere, Uzaktan Eğitim Merkezi'nin yapısı itibariyle ve bünyesinde çalışan teknik ekibin koordinatörlüğünde, katsayı olarak sırası ile 0.2, 0.3 ve 0.5 değerleri atanmıştır.

Bu durum aşağıdaki şekilde açıklanabilir;

- Uzaktan Eğitim Merkezi için finansal maliyet oldukça düşük bir önem taşımaktadır.
- Birim için teknik maliyet, finansal maliyetten daha önemli olmakla birlikte çok büyük bir önem taşımamaktadır.
- Birimin önem verdiği kriterlerin başında, oluşturulan sistemin ne kadar uzun süre ve verimli bir şekilde kullanılabileceği gelmektedir.

Tablo 5.1. Uzaktan Eğitim Merkezi'nde Stratejilerin Puanlanması

		Belirlenen Stratejiler			
		Strateji 1 **Yeniden Geliştirme**	*Strateji 2* **Paketleme**	*Strateji 3* **Göç Ettirme**	**Katsayı**
Kriterler	*Kriter 1* **Finansal Maliyet**	1	10	6	0.2
	Kriter 2 **Teknik Maliyet**	1	3	6	0.3
	Kriter 3 **Kullanım Süresi**	10	3	4	0.5
	Toplam Puan	***5.5***	***4.4***	***5***	

Puanlama işlemi esnasında her kriter, değeri 1 ile 10 arasında değişen ve en kötüden en iyiye doğru yükselen, bir ölçüm sistemi çerçevesinde değerlendirilmiştir. Bu durumda, örneğin strateji 1'in finansal maliyeti, strateji 2'nin finansal maliyetinden daha yüksektir ve bu sebepten dolayı da puanı daha düşüktür. Daha doğru bir ifade ile strateji 1, finansal bazda stratejiler arasındaki en pahalı stratejidir. Tüm bu değerlendirme işleminin sonucunda SOA'ya geçiş süreci için, elde ettiği en yüksek puanla, 1 numaralı stratejinin seçilmesinin uygun olabileceği söylenebilir.

5.5. Dijital Servislerin Yönetilmesi

İş Süreçlerine SOA Uygulama Metodolojisi'nin son adımı olan bu adımda, Uzaktan Eğitim Merkezi'nin öğretim yönetim sistemi için, bir önceki adımda belirlenen strateji çerçevesinde, öncelikle bir servis envanteri oluşturulmuştur, ardından dijital servisler geliştirilmiş ve son olarak ta dijital servislerin iş süreçlerine göre nasıl yönetilebileceğine dair bir örnek verilmiştir.

5.5.1. Servis envanterinin oluşturulması

Bölüm 5.2.3'te belirlenen kavramsal servisler için oluşturulan servis envanteri aşağıdaki şekildedir [98];

Giriş servisi:

- public String *kontrol* (String kullanici_adi, String parola): Bu fonksiyon kullanıcı giriş kontrolü için kullanılmaktadır. İki parametre almaktadır. Bunlar kullanıcının kullanıcı adı ve parolasıdır. Bu iki değer de String tipinde fonksiyona verilmelidir. Kontrol sonrasında kullanıcı adı ve parola doğru ise fonksiyon "Basarili" ifadesini döndürmektedir. Ancak kullanıcı adı veya parolasında hata olması durumunda "Hata" ifadesi döndürülmektedir.

Sınav servisi:

- public String *sinavOlustur* (Sinav sinav): Bu servis fonksiyonu ile uygulamanın arabirimi kullanılarak oluşturulan sınav nesnesinin veritabanına kaydı yapılmaktadır. Eğer fonksiyon çalışmasını başarılı bir şekilde sonlandırırsa geriye "Basarili" ifadesini döndürmektedir. İşlem sırasında bir hata ile karşılaşılması durumunda ise "Hata" ifadesi döndürülmektedir.

- public Sinav *idIleSınavBul* (String id): Herhangi bir sınav nesnesi yalnızca ona ait olan tanımlama numarası ile veritabanında aranmaktadır. Eğer sınav nesnesi veritabanında bulunursa değer olarak bu nesne döndürülmektedir. Ancak bir hata ile karşılaşıldığında veya nesne bulunamadığında "Hata" ifadesi döndürülmektedir.

- public String *sinavSil* (String sinavID): Bu fonksiyon benzersiz tanımlama numarası verilen bir sınav nesnesinin silinmesini sağlamaktadır. Eğer sınav nesnesi başarılı bir şekilde veritabanından silinirse sonuç olarak "Basarili" ifadesi döndürülmektedir. Ancak işlem sırasında bir hata ile karşılaşılırsa geriye "Hata ilfadesi döndürülmektedir.

- public String *sinavGuncelle* (Sinav sinav): Daha önceden sinavOlustur fonksiyonu ile oluşturulan bir sınav nesnesi bu fonksiyon yardımıyla güncellenebilmektedir. Bu fonksiyonun çalışması için güncellenmesi istenen sinav nesnesinin benzersiz tanımlama numarası gerekmektedir. Zira sadece bu numara ile kayıt veritabanında bulunabilmektedir. Eğer sınav nesnesi başarılı bir şekilde güncellenirse fonksiyon geriye "Basarili" ifadesini döndürmektedir. Ancak bir hata ile karşılaşılması veya verilen numara ile sınav nesnesinin bulunamaması durumunda fonksiyon "Hata" ifadesini döndürmektedir.

- public List<Sinav> *derseAitSinavlariGetir* (String dersID, String yil): Bu fonksiyon bir derse ait olan sınav nesnelerini veritabanından getirmek için kullanılmaktadır. Parametre olarak dersin benzersiz tanımlama numarası ve yıl verilmektedir. Fonksiyonun başarılı bir şekilde çalışmasını

sonlandırmasının ardından bulunan sınav nesneleri döndürülmektedir. Eğer herhangi bir sınav nesnesi bulunamazsa geriye null ifadesi döndürülmektedir.

Sınav sonuç servisi:

- public String *sinavSonucuOlustur* (SinavSonucu sinavSonucu): Bu servis fonksiyonu ile uygulamanın arabirimi kullanılarak oluşturulan sınav sonucu nesnesinin veritabanına kaydı yapılmaktadır. Eğer fonksiyon çalışmasını başarılı bir şekilde sonlandırırsa geriye "Basarili" ifadesi döndürülmektedir. İşlem sırasında bir hata ile karşılaşılması durumunda ise "Hata" ifadesi döndürülmektedir.
- public SinavSonucu *idIleSinavSonucuBul* (String sinavSonucuID): Herhangi bir sınav sonucu nesnesi yalnızca ona ait olan tanımlama numarası ile veritabanında aranmaktadır. Eğer sınav sonuç nesnesi veritabanında bulunursa değer olarak bu nesne döndürülmektedir. Ancak bir hata ile karşılaşıldığında veya nesne bulunamadığında null ifadesi döndürülmektedir.

- public String *sinavSonucuSil* (String sinavSonucID): Bu fonksiyon aracılığıyla benzersiz tanımlama numarası verilen bir sınav sonuç nesnesinin silinmesi sağlanmaktadır. Eğer sınav sonuç nesnesi başarılı bir şekilde veritabanından silinirse sonuç olarak "Basarili" ifadesi döndürülmektedir. Ancak işlem sırasında bir hata ile karşılaşılırsa geriye "Hata" ilfadesi döndürülmektedir.

- public String *sinavSonucuGuncelle* (SinavSonuc sinavSonuc): Daha önceden sinavSonucuOlustur fonksiyonu ile oluşturulan bir sınav sonuç nesnesi bu fonksiyon yardımıyla güncellenebilmektedir. Bu fonksiyonun çalışması için güncellenmesi istenen sinav sonuç nesnesinin benzersiz tanımlama numarası gerekmektedir. Zira sadece bu numara ile kayıt veritabanında bulunabilmektedir. Eğer sınav sonuç nesnesi başarılı bir şekilde güncellenirse fonksiyon geriye "Basarili" ifadesini döndürmektedir. Ancak bir hata ile karşılaşılması veya verilen numara ile sınav nesnesinin bulunamaması durumunda fonksiyon "Hata" ifadesini döndürmektedir.

- public List<SinavSonuc> *ogrenciyeAitButunSinavSonuclariniGetir* (String ogrenciID, String yil): Bu fonksiyon bir öğrenciye ait olan sınav sonuç nesnelerini veritabanından getirmek için kullanılmaktadır. Parametre olarak öğrencinin benzersiz tanımlama numarası ve yıl verilmektedir. Fonksiyonun başarılı bir şekilde çalışmasını sonlandırmasının ardından bulunan sınav sonuç nesneleri döndürülmektedir. Eğer herhangi bir sınav sonuç nesnesi bulunamazsa geriye null ifadesi döndürülür.

- public List<SinavSonuc> *derseAitButunSinavSonuclariniGetir* (String dersID, String yil): Bu fonksiyon bir derse ait olan sınav sonuç nesnelerini veritabanından getirmek için kullanılmaktadır. Parametre olarak dersin benzersiz tanımlama numarası ve yıl verilmektedir. Fonksiyonun başarılı bir şekilde çalışmasını sonlandırmasının ardından bulunan sınav sonuç nesneleri döndürülmektedir. Eğer herhangi bir sınav sonuç nesnesi bulunamazsa geriye null ifadesi döndürülmektedir.

- public List<SinavSonuc> *sinavaAitButunSinavSonuclariniGetir* (String sinavID, String yil): Bu fonksiyon bir sınava ait olan sınav sonuç nesnelerini veritabanından getirmek için kullanılmaktadır. Parametre olarak sınav nesnesinin benzersiz tanımlama numarası ve yıl verilmektedir. Fonksiyonun başarılı bir şekilde çalışmasını sonlandırmasının ardından bulunan sınav sonuç nesneleri döndürülmektedir. Eğer herhangi bir sınav sonuç nesnesi bulunamazsa geriye null ifadesi döndürülmektedir.

Ders servisi:

- public String *dersOlustur* (Ders ders): Bu servis fonksiyonu ile uygulamanın arabirimi kullanılarak oluşturulan ders nesnesinin veritabanına kaydı yapılmaktadır. Eğer fonksiyon çalışmasını başarılı bir şekilde sonlandırırsa geriye "Basarili" ifadesi döndürülmektedir. İşlem sırasında bir hata ile karşılaşılması durumunda ise "Hata" ifadesi döndürülmektedir.

- public Ders *idIleDersBul* (String id): Herhangi bir ders nesnesi yalnızca ona ait olan tanımlama numarası ile veritabanında aranmaktadır. Eğer ders

nesnesi veritabanında bulunursa değer olarak bu nesne döndürülmektedir. Ancak bir hata ile karşılaşıldığında veya nesne bulunamadığında "Hata" ifadesi döndürülmektedir.

- public String *dersSil* (String id): Benzersiz tanımlama numarası verilen bir ders nesnesinin silinmesi sağlanmaktadır. Eğer ders nesnesi başarılı bir şekilde veritabanından silinirse sonuç olarak "Basarili" ifadesi döndürülmektedir. Ancak işlem sırasında bir hata ile karşılaşılırsa geriye "Hata" ifadesi döndürülmektedir.

- public String *dersGuncelle* (Ders ders): Daha önceden dersOlustur fonksiyonu ile oluşturulan bir ders nesnesi bu fonksiyon yardımıyla güncellenebilmektedir. Bu fonksiyonun çalışması için güncellenmesi istenen ders nesnesinin benzersiz tanımlama numarası gerekmektedir. Zira sadece bu numara ile kayıt veritabanında bulunabilmektedir. Eğer ders nesnesi başarılı bir şekilde güncellenirse fonksiyon geriye "Basarili" ifadesini döndürmektedir. Ancak bir hata ile karşılaşılması veya verilen numara ile sınav nesnesinin bulunamaması durumunda fonksiyon "Hata" ifadesini döndürmektedir.

- public List<Ders> *butunDerslerıGetir* (): Bu fonksiyon sonucunda veritabanındaki bütün ders nesneleri getirilmektedir. Herhangi bir parametre almamaktadır. Eğer veritabanında ders nesnesi bulunamazsa null döndürülmektedir.

- public List<Ders> *profesoreAitDerslerıGetir* (String profesorID): Bu fonksiyon yardımıyla veritabanından bir profesöre ait dersler getirilmektedir. Parametre olarak profesör nesnesine ait olan benzersiz tanımlama numarası alınmaktadır. Veritabanında verilen profesör numarasına karşılık herhangi bir ders bulunamazsa geriye null döndürülmektedir.

- public List<Ders> *ogrenciyeAitDerslerıGetir* (String ogrenciID): Bu fonksiyon yardımıyla veritabanından bir öğrenciye ait dersler getirilmektedir. Parametre olarak öğrenci nesnesine ait olan benzersiz tanımlama numarası

alınmaktadır. Veritabanında verilen öğrenci numarasına karşılık herhangi bir ders bulunamazsa geriye null döndürülmektedir.

- public String *profesoreDersEkle* (String profesorID, String dersID): Bu fonksiyon yardımıyla bir profesöre ders ataması yapılabilmektedir. Fonksiyona parametre olarak verilen profesör ve derse ait benzersiz tanımlama numaralarıyla ders ile profesör arasında veritabanında bir ilişki oluşturulmaktadır. Eğer fonksiyon işlemini başarılı bir şekilde sonlandırırsa sonuç olarak "Basarili" ifadesi döndürülmektedir. Bir hata ile karşılaşılması durumunda ise geriye "Hata" ifadesi gönderilmektedir.

- public String *profesordenDersCikar* (String profesorID, String dersID): Daha önceden profesoreDersEkle fonksyionu kullanılarak tanımlanan herhangi bir ders bu fonksiyon yardımıyla silinebilmektedir. Parametre olarak profesör ve derse ait benzersiz tanımlama numaralarını alınmaktadır. Eğer fonksiyon işlemini başarılı bir şekilde sonlandırırsa sonuç olarak "Basarili" ifadesi döndürülmektedir. Bir hata ile karşılaşılması durumunda ise geriye "Hata" ifadesi gönderilmektedir.

- public void *profesordenButunDersleriCikar* (String profesorID): Bu fonksiyon yardımıyla bir profesör için tanımlanmış bütün dersler silinebilmektedir. Parametre olarak profesöre ait benzersiz tanımlama numarası alınmaktadır.

- public String *ogrenciyeDersEkle* (String ogrenciID, String dersID): Bu fonksiyon yardımıyla bir öğrenciye ders ataması yapılabilmektedir. Fonksiyona parametre olarak verilen öğrenci ve derse ait benzersiz tanımlama numaralarıyla ders ile öğrenci arasında veritabanında bir ilişki oluşturulmaktadır. Eğer fonksiyon işlemini başarılı bir şekilde sonlandırırsa sonuç olarak "Basarili" ifadesi döndürülmektedir. Bir hata ile karşılaşılması durumunda ise geriye "Hata" ifadesi gönderilmektedir.

- public String *ogrencidenDersCikar* (String ogrenciID, String dersID): Daha önceden ogrenciyeDersEkle fonksiyonu kullanılarak tanımlanan herhangi bir ders bu fonksiyon yardımıyla silinebilmektedir. Parametre olarak öğrenci ve

derse ait benzersiz tanımlama numaralarını almaktadır. Eğer fonksiyon işlemini başarılı bir şekilde sonlandırırsa sonuç olarak "Basarili" ifadesi döndürülmektedir. Bir hata ile karşılaşılması durumunda ise geriye "Hata" ifadesi gönderilmektedir.

- public void *ogrencidenButunDersleriCikar* (String ogrenciID): Bu fonksiyon yardımıyla bir öğrenci için tanımlanmış bütün dersler silinebilmektedir. Parametre olarak öğrenciye ait benzersiz tanımlama numarası alınmaktadır.

Öğrenci servisi:

- public String *ogrenciOlustur* (Ogrenci ogrenci): Bu servis fonksiyonu ile uygulamanın arabirimi kullanılarak oluşturulan öğrenci nesnesinin veritabanına kaydı yapılmaktadır. Eğer fonksiyon çalışmasını başarılı bir şekilde sonlandırırsa geriye "Basarili" ifadesi döndürülmektedir. İşlem sırasında bir hata ile karşılaşılması durumunda ise "Hata" ifadesi döndürülmektedir.

- public Ogrenci *kullanciAdiIleOgrenciGetir* (String kullaniciAdi): Herhangi bir öğrenci nesnesi yalnızca ona ait olan kullanıcı adı ile veritabanında aranmaktadır. Eğer öğrenci nesnesi veritabanında bulunursa değer olarak bu nesne döndürülmektedir. Ancak bir hata ile karşılaşıldığında veya nesne bulunamadığında null ifadesi döndürülmektedir.

- public Ogrenc *idIleOgrenciGetir* (String id): Herhangi bir öğrenci nesnesi yalnızca ona ait olan tanımlama numarası ile veritabanında aranmaktadır. Eğer öğrenci nesnesi veritabanında bulunursa değer olarak bu nesne döndürülmektedir. Ancak bir hata ile karşılaşıldığında veya nesne bulunamadığında null ifadesi döndürülmektedir.

- public String *ogrenciSil* (String id): Benzersiz tanımlama numarası verilen bir öğrenci nesnesinin silinmesi sağlanmaktadır. Eğer öğrenci nesnesi başarılı bir şekilde veritabanından silinirse sonuç olarak "Basarili" ifadesi döndürülmektedir. Ancak işlem sırasında bir hata ile karşılaşılırsa geriye "Hata" ifadesi döndürülmektedir.

- public String *ogrenciGuncelle* (Ogrenci ogrenci): Daha önceden ogrenciOlustur fonksiyonu ile oluşturulan bir öğrenci nesnesi bu fonksiyon yardımıyla güncellenebilmektedir. Bu fonksiyonun çalışması için güncellenmesi istenen ders nesnesinin benzersiz tanımlama numarası gerekmektedir. Çünkü sadece bu numara ile kayıt veritabanında bulunabilmektedir. Eğer öğrenci nesnesi başarılı bir şekilde güncellenirse fonksiyon geriye "Basarili" ifadesini döndürmektedir. Ancak bir hata ile karşılaşılması veya verilen numara ile sınav nesnesinin bulunamaması durumunda fonksiyon "Hata" ifadesini döndürmektedir.

- public List<Ogrenci> *butunOgrencileriGetir* (): Bu fonksiyon sonucunda veritabanındaki bütün öğrenci nesneleri getirilmektedir. Herhangi bir parametre alınmamaktadır. Eğer veritabanında öğrenci nesnesi bulunamazsa null döndürülmektedir.

Profesör servisi:

- public String *profesorOlustur* (Profesor profesor): Bu servis fonksiyonu ile uygulamanın arabirimi kullanılarak oluşturulan profesör nesnesinin veritabanına kaydı yapılmaktadır. Eğer fonksiyon çalışmasını başarılı bir şekilde sonlandırırsa geriye "Basarili" ifadesi döndürülmektedir. İşlem sırasında bir hata ile karşılaşılması durumunda ise "Hata" ifadesi döndürülmektedir.

- public Profesor *kullanciAdiIleProfesorGetir* (String kullaniciAdi): Herhangi bir profesör nesnesi yalnızca ona ait olan kullanıcı adı ile veritabanında aranmaktadır. Eğer profesör nesnesi veritabanında bulunursa değer olarak bu nesne döndürülmektedir. Ancak bir hata ile karşılaşıldığında veya nesne bulunamadığında null ifadesi döndürülmektedir.

- public Profesor *idIleProfesorGetir* (String id): Herhangi bir profesör nesnesi yalnızca ona ait olan tanımlama numarası ile veritabanında aranmaktadır. Eğer profesör nesnesi veritabanında bulunursa değer olarak bu nesne

döndürülmektedir. Ancak bir hata ile karşılaşıldığında veya nesne bulunamadığında null ifadesi döndürülmektedir.

- public String *profesorSil* (String id): Benzersiz tanımlama numarası verilen bir profesör nesnesinin silinmesi sağlanmaktadır. Eğer profesör nesnesi başarılı bir şekilde veritabanından silinirse sonuç olarak "Basarili" ifadesi döndürülmektedir. Ancak işlem sırasında bir hata ile karşılaşılırsa geriye "Hata" ifadesi döndürülmektedir.

- public String *profesorGuncelle* (Profesor profesor): Daha önceden profesorOlustur fonksiyonu ile oluşturulan bir profesör nesnesi bu fonksiyon yardımıyla güncellenebilmektedir. Bu fonksiyonun çalışması için güncellenmesi istenen profesör nesnesinin benzersiz tanımlama numarası gerekmektedir. Çünkü sadece bu numara ile kayıt veritabanında bulunabilmektedir. Eğer profesör nesnesi başarılı bir şekilde güncellenirse fonksiyon geriye "Basarili" ifadesini döndürmektedir. Ancak bir hata ile karşılaşılması veya verilen numara ile profesör nesnesinin bulunamaması durumunda fonksiyon "Hata" ifadesini döndürmektedir.

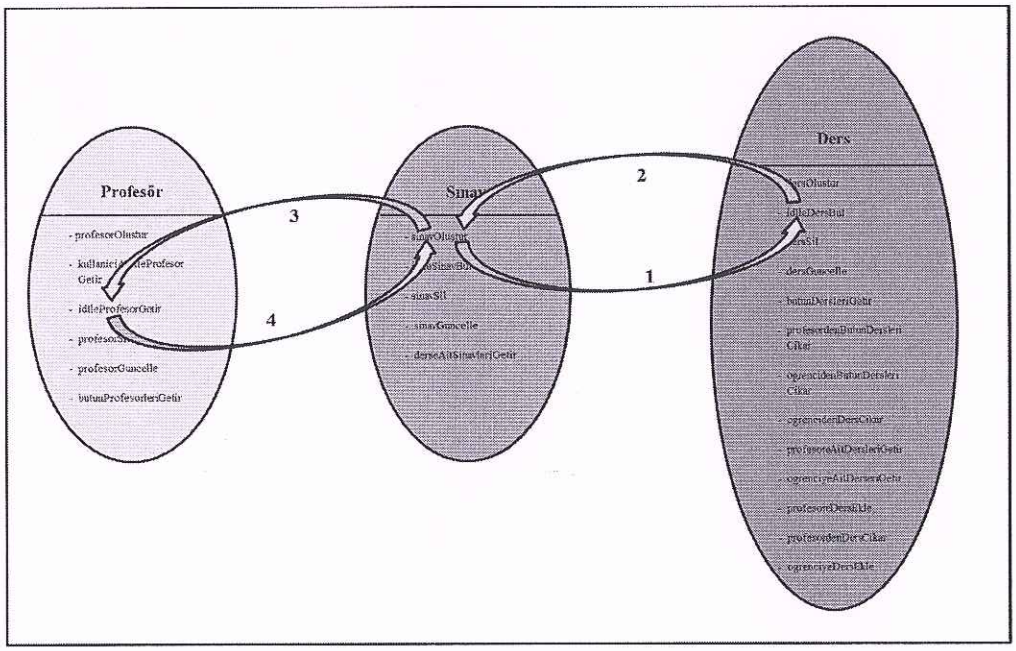
- public List<Profesor> *butunProfesorleriGetir* (): Bu fonksiyon sonucunda veritabanındaki bütün profesör nesneleri getirilmektedir. Herhangi bir parametre almamaktadır. Eğer veritabanında profesör nesnesi bulunamazsa null döndürülmektedir.

5.5.2. Dijital servislerin geliştirilmesi

Bu uygulamadaki dijital servisler, bölüm 5.2.3'te belirlenen kavramsal servisler ve bölüm 5.5.1'deki servis envanteri baz alınarak geliştirilmişlerdir. Servisler Java kodları ile geliştirilmiş olup, yönetim platformu olarak ta bölüm 2.6'daki analizler doğrultusunda Oracle SOA Suite 11g kullanılmıştır. Servis kodları, kapladıkları büyük yer sebebiyle tezin 6. Bölümündeki ekler kısmında yer almaktadır.

5.5.3. İş süreçlerine göre servislerin yönetilmesi

İş süreçlerine göre servislerin yönetilmesi, İş Süreçlerine SOA Uygulama Metodolojisi'nin en son aşamasıdır. Bu adıma gelindiğinde, hedeflenen iş süreç modelinin ve dijital servislerin hazır ve erişilebilir olmaları gerekmektedir. Ardından, iş süreçlerini otomatikleştirecek şekilde, servis ve servis fonksiyonlarının biribiriyle veri alışverişini gerçekleştirebileceği bağlantıların, servislerin birleştirilebilme özelliğinin kullanılarak sağlanması gerekmektedir. Şekil 5.5, Sınav Sisteminin Oluşturulması Sürecine ait servis yönetimini canlandırmaktadır.



Şekil 5.5. Sınav Sistemi'nin Oluşturulması Sürecine Ait Servislerin Yönetilmesi

Sınav sisteminin oluşturulması süreci, sınav servisine ait sinavOlustur fonksiyonu aracılığıyla gerçekleştirilmektedir. Bu fonksiyonun girdi parametresi (bölüm 5.5.1) olan sınav nesnesinin, hangi ders için sınav sisteminin oluşturulacağına ve dersi veren profesöre dair elde etmesi gereken bilgileri, ilgili Ders ve Profesör servislerinin idIleDersBul ve idIleProfesorGetir fonksiyonları ile haberleşerek elde edebileceği öngörüldüğü durumda, servislerin birleştirilebilmesi özelliğinin kullanılarak, şekil 5.5'teki örnekteki gibi bir bağlantı sisteminin kurulması aracılığıyla servislerin yönetilmesi gerekmektedir.

BÖLÜM 6. SONUÇLAR VE ÖNERİLER

6.1. Giriş

Sürekli değişmekte olan tüketici istek ve ihtiyaçları, kurum yapılarının, bu istek ve ihtiyaçları karşılayabilmek için mümkün olduğunca esnek ve çevik olmalarını gerektirmektedir. Müşteri memnuniyeti açısından, arzulanan ürün ve hizmetlerin, arzulanan kalitede, gecikmesiz ve uygun maliyetle sunulabilmesi çok önemli bir rol oynamaktadır. Bu durum karşısında kurumların gereksinim duydukları esnek ve çevik yapı, çoğunlukla iş süreçlerini otomatikleştirmek için kullandıkları yazılımlar aracılığıyla sağlanmaya çalışılmaktadır. Günümüzde, tüketici istek ve ihtiyaçları doğrultusunda sürekli değişmek durumunda olan iş süreçleri, kendilerini otomatikleştiren yazılımların da sürekli değişebilecek, esnek ve çevik bir yapıda geliştirilmelerini zorunlu kılmaktadır.

Yazılımların arzulandığı şekilde esnek ve çevik olabilmeleri için, kırılgan olmayan, taşınabilir, karmaşık olmayan, gereksiz tekrar içermeyen, anlaşılması kolay, düzenli ve kontrol edilebilir bir yapıda geliştirilmeleri gerekmektedir.

Arzu edilen bu yapı, değişim etkisinin minimuma indirilmesini hedefleyen SOA yaklaşımı aracılığıyla sağlanabilmektedir. Ancak SOA'nın oldukça dikkatli ve hedeflerinin iyice anlaşıldıktan sonra belirli bir metodolojinin takip edilerek uygulanması gerekmektedir. Aksi takdirde SOA, kullanım şekli okunmadan kullanılmış bir ilaç etkisi yaratabilmektedir.

Yazılım geliştirme ve özellikle SOA literatürü incelendiğinde araştırmacıların iş süreçlerine uygulanabilecek özel bir SOA Uygulama Metodolojisi'ne ihtiyaç duyduklarını belirttikleri görülmektedir. Bu çalışmada bu ihtiyacı karşılamak üzere bir metodoloji geliştirilmiş ve metodolojinin işleyişi açıklanmıştır.

Geliştirilen Metodoloji, Kavramsal Servisleri Belirleme, Eski Sistem Analizi, Dijital Servisler İçin Strateji Belirleme, Dijital Servislerin Yönetilmesi adımlarından oluşmaktadır. Bu adımların genel açıklamaları aşağıdaki şekildedir.

- Kavramsal Servisleri Belirleme: Bu adımda SOA uygulamaya başlamak isteyen kurumun mevcut iş süreçlerinden yola çıkılarak, hedef iş süreçleri modellenmekte ve bu doğrultuda ihtiyaç duyulan fonksiyonlar belirlenmektedir. Bu fonksiyonlar daha sonra kavramsal servisler altında sınıflandırılmaktadır.
- Eski Sistem Analizi: Bu adımda kurumun kullandığı eski sistemin kullanılabilecek kısımlarının yeni SOA yaklaşımlı sistemde de kullanılabilmesi hedeflenmektedir. Bunun için eski sistemin analiz edilebilmesi gerekmektedir. Bu işlem için öncelikle bir analiz metodu belirlenmektedir, bu metot kullanılarak analizi gerçekleştirilen sistemin analiz sonuçlarının değerlendirilmesi ile bu adım sonlandırılmaktadır.
- Dijital Servisler İçin Strateji Belirleme: Bu adımda öncelikle dijital servisler için olası stratejiler belirlenmektedir. Daha sonra belirlenen bu stratejiler arası teknik ve finansal analizler yürütülmektedir. Sonuç olarak dijital servislerin geliştirilebilmesi için bir strateji seçilmektedir.
- Dijital Servislerin Yönetilmesi: Son olarak bu adımda servislerin geliştirilme işlemine geçilmeden önce servis envanteri oluşturulmaktadır. Ardından dijital servislerin geliştirilmesi işlemi gerçekleştirildikten sonra, bu servislerin iş süreçlerine göre yönetilmesi ile bu adım sonlandırılmaktadır.

Çalışmanın son bölümünde, geliştirilen metodoloji bir üniversitenin uzaktan eğitim merkezinin iş süreçlerine uygulanmıştır. Uygulama sonucunda, metodolojinin fonksiyonunu yerine getirebildiği ve elde edilen yazılım yapısının esnek ve çevik olduğu ve değişim etkisinin de oldukça düşük olduğu görülmüştür.

6.2. Katkılar

Çalışmanın sağladığı katkılar aşağıdaki şekilde özetlenebilir:

- Kurumlarda, SOA'nın daha verimli bir şekilde uygulanabilmesi için genel bir uygulama metodolojisi geliştirilmiştir ve bu metodolojinin nasıl kullanılabileceği açıklanmıştır. Elde edilen bu metodoloji sayesinde SOA, uygulama uzmanları için, daha kolay algılanabilen ve uygulanabilen bir hal kazanmıştır.
- Geliştirilen metodoloji sayesinde, SOA uygulanırken, iş süreçleri üzerinden doğabilecek kayıplar azaltılmaktadır. Zira geliştirilen metodoloji, öncelikli olarak kurumlardaki iş süreçlerini ele almaktadır. Geliştirilen metodoloji ile birlikte, kavramsal servislerin tasarlanması, ancak mevcut ve hedef iş süreçleri ele alındıktan sonra sağlanmaktadır.
- Kurumlarda, SOA'ya geçiş öncesi kullanılan eski sistemlerin yararlı kısımlarının, yeni geliştirilen SOA tabanlı sistemde de kullanılması sağlanmıştır. Bu sayede, büyük maliyetlerle elde edilmiş olan eski sistemlerin, tamamen kullanılamaz bir hale gelmelerinin önüne geçilmiştir.
- Geliştirilen metodoloji, SOA'nın ortada buluşma teorisine dayandırılmıştır ve bu şekilde uygulama maliyeti düşürülmeye çalışılmıştır. Bu sayede diğer iki yaklaşım olan yukarıdan aşağıya ve aşağıdan yukarıya yaklaşımlarının avantajlı yanlarının kullanılması sağlanmıştır.
- Geliştirilen metodoloji, bir üniversitenin uzaktan eğitim merkezinin iş süreçleri üzerine uygulanmıştır. Uygulamanın, metodolojinin adımlarının takibi ile sorunsuz bir şekilde sonuca ulaşabildiği görülmüştür.
- SOA uygulama esnasında, strateji belirleme ve seçiminin önemi vurgulanmıştır. Kavramsal servislerin belirlenmesinden sonra, bu servislerin geliştirilebilmesi için stratejilerin belirlenmesi, belirlenen stratejiler arasında teknik ve finansal analizlerin yürütülmesi ve bu analizler doğrultusunda stratejilerden birinin uygulanmasına karar verilmesi sağlanmıştır. Bu şekilde, karar aşamasında hata yapma ihtimali düşürülmüştür.
- SOA tabanlı servis geliştirme ve yönetme işlemi gerçekleştirilirken, SOA yönetim platformu seçiminin önemi vurgulanmıştır. Bu aşamada, günümüzde

piyasada bulunan birçok SOA yönetim platformu üzerinde, performans ve maliyet analizinin yürütülmesi gerektiği belirtilmiştir. Bu sayede kurum için önem teşkil eden SOA yönetim platformu seçiminde hata yapma ihtimali düşürülmüştür.

- Geliştirilen metodolojinin etkinliği, elli kişilik bir grup çalışmasının sonuçlarından elde edilen veriler ışığında istatistiksel olarak test edilmiştir. Bu sonuçlara göre, belirli bir SOA uygulama projesi için, metodoloji öncesi harcanan zaman ile metodoloji sonrası harcanan zaman arasında pozitif yönde belirgin bir fark olduğu gözlenmiştir.

6.3. Kısıtlar

SOA yaklaşımı, uygulama alanının büyüklüğü oranında etkisini daha iyi gösterebilen bir yaklaşımdır. Bununla birlikte SOA projelerinin de kurum çapında ve tecrübeli proje ekipleri ile yürütülmesi gerekmektedir. Kurum bazında gerçekleştirilen SOA projeleri, kuruma bağlı olarak, beş yüz ile iki bin sayfa arasında ve proje ekibi çalışan sayısı da on ile iki yüz arasında değişebilmektedir. Aynı şekilde, İş Süreçlerine SOA Uygulama Metodolojisi'nin de kapsamı daha geniş ve çok sayıda SOA projesinde uygulanarak değerlendirilmesi gerekmektedir. Tez kapsamında ve tek kişi tarafından yapılabilecek hiçbir uygulamanın yeterlilik garantisi olamaz. Aynı kapsamda yapılacak birden çok uygulamaya dair analizler de yeterlilik garantisi veremez. Zira metodoloji her iş sürecine uygulanabilmekte ve her bir uygulamayı da sonuçlandırmaktadır. Yetkinlik ölçümünün yapılabilmesi için kapsamı genişletilmiş SOA projelerine ihtiyaç duyulmaktadır.

SOA uygulamalarının kalitesi, SOA olgunluk modelleri tarafından değerlendirilebilmektedir. İş Süreçlerine SOA Uygulama Metodolojisi'nin etkisinin ölçülebilmesi ve bu sayede geliştirilebilmesi için, metodolojinin uygulandığı projelerin profesyonel olgunluk ölçüm uzmanları tarafından değerlendirilmesi gerekmektedir. Bağımsız uzmanlar tarafından yapılacak değerlendirmeler tarafsızlığı da ortadan kaldırarak metodolojinin doğru bir şekilde geliştirilmesine katkı sağlayacaklardır.

SOA olgunluk modelleri SOA uygulamalarını, altyapı, mimari, enformasyon, proje yönetimi, organizasyon, finans ve yönetim boyutlarını ele alarak değerlendirmektedir. İş Süreçlerine SOA Uygulama Metodolojisi sadece altyapı ve mimari boyutlarını uygulayabilmek ve iyileştirebilmek için geliştirilmiştir. Bu durumda Metodoloji'nin etkisi güncel hali ile SOA olgunluk modelleri tarafından tam olarak değerlendirilemeyebilir.

6.4. Gelecek Çalışması

Bu çalışmada SOA'nın iş süreçlerine daha verimli bir şekilde uygulanabilmesi için bir metodoloji geliştirilmiştir. Geliştirilen metodoloji aracılığıyla kurumların iş süreçleri ve eski sistemleri daha yakından incelenmekte ve belirlenen stratejiler ışığında tasarlanan yeni sistemle entegrasyonları sağlanmaya çalışılmaktadır. Bu çalışma temel alınarak aşağıdaki şekilde yeni çalışma konuları geliştirilebilir.

- Geliştirilen metodolojinin ana ve alt adımları, olgunluk modellerinin denetlediği tüm boyutları iyileştirebilecek şekilde genişletilerek, daha ayrıntılı bir uygulama metodolojisinin geliştirilmesi sağlanabilir.
- Etkili olabileceği görüldüğü durumlarda veya ilerleyen zamanlarda daha yeni tekniklerin gelişmesiyle birlikte Metodoloji'nin, mevcut ana ve alt adımları üzerinde veya barındırdıkları tekniklerde, eklemeler ve güncellemeler yapılabilir.
- Metodoloji'nin farklı kurumların farklı iş süreçleri üzerinde uygulanması sağlanarak hangi kurum veya iş süreçlerinin yapısında daha iyi sonuçlar çıkarabileceği üzerine çalışmalar yapılabilir.
- Bu çalışmadaki metodoloji temel alınarak farklı adımların takip edildiği ve farklı metotların uygulandığı benzer metodolojiler geliştirilebilir. Benzer bir metodolojinin geliştirilmesi durumunda da metodolojiler arası analizler yapılabilir ve her birinin güçlü ve zayıf yanları ortaya çıkarılabilir.
- Metodoloji'nin bazı adımlarında piyasa tarafından kabul görmüş belirli tekniklerin kullanılması tavsiye edilmektedir. Bu teknikler yerine farklı tekniklerin kullanılması ile elde edilebilecek SOA kalitesi arasındaki ilişkiler üzerine istatiksel araştırmalar yapılabilir.

KAYNAKLAR

[1] BÜTTNER, M, SOA – Reifegradmodelle Analyse und Weiterentwicklung von Reifegradmodellen für SOA. Verlag Dr. Müller. 2010.

[2] MCKENDRICK, J, Gartner leaves SOA off 'Top 10 technology' list, but with good reason. http://www.zdnet.com/blog/service-oriented/gartner-leaves-soa-off-top-10-technology-list-but-with-good-reason/978. Erişim Tarihi: 08.01.2012.

[3] MCKENDRICK, J, Gartner leaves SOA off 'top ten' list – again. http://www.zdnet.com/blog/service-oriented/gartner-leaves-soa-off-top-ten-list-again/3159. Erişim Tarihi: 08.01.2012.

[4] ERL, T, Service-Oriented Architecture – concepts, Technology and Design. Prentice Hall. 2006.

[5] DİRİK, S, Şirketiniz için SOA – 1. http://www.bilgicagi.com/Yazilar/5739-sirketiniz_icin_soa_1.aspx. Erişim Tarihi: 02.05.2012.

[6] MORELAND, B., AFSHAR, M., The Path to SOA (part 1 of 3). http://www.ebizq.net/topics/system_management/features/7193.html. Erişim Tarihi: 07.11.2008.

[7] ERL, T, SOA - Entwurfsprinzipien für serviceorientierte Architektur. Addison-Wesley Verlag. 2008.

[8] GOASDUFF, L, FORSLING C, Bad Technical Implementations and Lack of Governance Increase Risks of Failure in SOA Projects. http://www.gartner.com/it/page.jsp?id=508397. Erişim Tarihi: 03.06.2012.

[9] ERL, T., SOA Design Patterns. Prentice Hall. 2009.

[10] ALTUNTAŞ, C., Affet bizi SOA seni yanlış anladık. http://www.cihataltuntas.com/?cat=11. Erişim Tarihi: 22.06.2012.

[11] REMPP, G., AKERMANN, M., LÖFFLER, M., LEHMANN, J., Model Driven SOA. Springer Verlag Berlin Heidelberg. 2011.

[12] YEGÜL, MF, TOKLU, B, Türkeye'de ERP Uygulamaları. http://www.mmo.org.tr/resimler/dosya_ekler/e8c15fed5f80800_ek.pdf?dergi=120. Erişim Tarihi: 02.05.2012

[13] HUVAR, M., FALTER, T., FIEDLER, T., ZUBEV, A., Anwendungsentwicklung mit Enterprise SOA. Galileo Press. 2008.

[14] ERL, T., Web Service Contract Design and Versioning for SOA. Prentice Hall. 2009.

[15] HACK, S., LINDEMANN, MA., Enterprise SOA Einführen. Galileo Press. 2007.

[16] WIEHLER, G., Mobility, Security und Web Services. Siemens Aktiengesellschaft. 2004.

[17] JACOBS, S., Reifegradmodelle. http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/is-management /Systementwicklung/reifegradmodelle. Erişim Tarihi:15.03.2012.

[18] JACOBS, S., CMMI - Enzyklopaedie der WI. http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/is-management/ Systementwicklung/reifegradmodelle/cmmi. Erişim Tarihi: 19.02.2012.

[19] AHLEMANN, F., SCHROEDER, C., TEUTEBERG, F., Kompetenz- und Reifegradmodelle für das Projektmanagement Grundlagen, Vergleich und Einsatz. http://www.bow.uni-osnabrueck.de/reifegradmodelle.pdf. Erişim Tarihi: 15.03.2012.

[20] PUGSLEY, A., Assessing your SOA Program. http://h20195.www2.hp.com/v2/GetPDF.aspx/4AA0-4824ENW.pdf. Erişim Tarihi: 26.02.2012.

[21] SOA MATURITY MODEL, Ein Vorgehensmodell zur Einführung einer serviceorientierten Architektur, http://www.progress.com.br /progress_ software/worldwide_sites/de/docs/whitepaper.pdf. Erişim Tarihi: 26.02.2012.

[22] ARSANJANI, A., HOLLEY, K., The Service Integration Maturity Model: Achieving Flexibility in the Transformation to SOA. http://iecexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4026979. Erişim Tarihi: 26.02.2012.

[23] INAGANTI, S., ARAVAMUDAN, S. vd., SOA Maturity Model. http://www.bptrends.com/publicationfiles/04-07-ART-The%20SOA%20 MaturityModel-Inagantifinal.pdf. Erişim Tarihi: 22.08.2012.

[24] AGILE PROZESSE durch Service Orientierung. http://www.bpm-soa-center.com/. Erişim Tarihi: 26.07.2011.

[25] ASIMOĞLU, E., Vorgehensweise von SOAMMI am Beispiel eines Standard-Software. İşletme Enformatiği Böl., Marmara Üniversitesi, İstanbul. 2010.

[26] DOSTAL, W, JECKLE, M, MELZER, I, ZENGLER B, Service-orientierte Architekturen mit Web Services. Spektrum Akademischer Verlag. 2005.

[27] BAYRAK MEYDANOĞLU, ES., HERAND, D., Geschäftsprozessmanagement und SOA: In Optimierung von Geschäftsprozessen, AKPINAR, H., ÖZTÜRK R. (ed.); Shaker Verlag, 2010; pp. 41-55.

[28] KNEUPER, R., CMMI : Verbesserung von Softwareprozessen mit Capability Maturity Model Integration. dpunkt.verlag. 2003.

[29] HERAND, D., GÜRDER F., TAŞKIN, H., YÜKSEL, EN., A healthcare management system for Turkey based on a service-oriented architecture. Informatics for Health and Social Care, 2012; pp. 1–19.

[30] CHU, SC., From Component-based to Service Oriented Software Architecture for Healthcare. 2005.

[31] ALBOAIE, L., BURAGA, SC., FELEA, V., TELEMON – an SOA-based e-Health System. Designing the Main Architectural Components. 9th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS. Suceava, Romania. 2008.

[32] HEUSER, O., HOLUBEK, A., Java Web Services in der Praxis. dpunkt.verlag. 2010.

[33] BECKER, J., MATHAS, C., WINKELMANN, A., Geschäftsprozessmanagement. Springer Verlag Berlin Heidelberg. 2008.

[34] DREIFUS, F., LEYKING, K., LOOS, P., Systematisierung der Nutzenpotentiale einer SOA, In Service-orientierte Architecturen - Chancen und Herausforderungen bei der Flexibilisierung und Integration von Unternehmensprozessen, NISSEN, V., PETSCH, M., SCHORCHT, H. (ed.); Deutscher Universitäts-Verlag, Wiesbaden, 2007; pp. 19-38

[35] EICKER, S., NAGEL, A., SCHULER, PM., Flexibilität im Geschäftsprozessmanagement-Kreislauf. ICB Research Report No. 21, Universität Duisburg Essen, Institut für Informatik und Wirtschaftsinformatik (ICB). 2007.

[36] SCHIMM, G., Process Mining – Anwendungen. http://www.processmining.de/1077.html. Erişim Tarihi: 26.07.2011.

[37] COOK, J.,WOLF, A., Discovering Models of Software Processes from Event-Based Data ACM Transactions on Software Engineering and Methodology. 1998; 7(3):215-249.

[38] COOK, J.,WOLF, A., Event Based Detection of Concurrency. In Proceedings of the Sixth International Symposium on the Foundation of Software Engineering (FSE-6). Orlando, FL. 1998; pp. 35-45.

[39] COOK, J., Process Discovery and Validation through Event-Data Analysis. University of Colorado. 1996.

[40] SCHIMM, G., Process Mining - Was ist Process Mining. http://www.processmining.de/1054.html. Erişim Tarihi: 26.07.2011

[41] BUSINESS PROCESS MINING. http://businessprocessmining. blogspot.com/. Erişim Tarihi: 26.07.2011.

[42] ZIMMERMANN, A., GROß, HJ., PILLER, G., BUCKOW, H., Capability Diagnostics of Enterprise Service Architectures using a dedicated Software Architecture Reference Model. IEEE International Conference on Services Computing. 2011.

[43] BUCKOW, H., PİLLER, G., GROß HJ., PROTT, K., Evaluating Service-oriented Vendor Platforms with a Dedicated Architecture Maturity Framework. Reutlingen Üniversitesi. 2011.

[44] ZIMMERMANN, A., LAUX, F., REINERS, R., A Pattern Language for Architecture Assessments of Service-oriented Enterprise Systems. Reutlingen Üniversitesi. 2011.

[45] DOMAIN MODEL FOR SOA. Realizing the Business Benefit of Service-Oriented Architecture. http://eudownload.bea.com/uk/events/soa/soa.pdf. Erişim Tarihi: 22.11.2008.

[46] KHOSHAFIAN, S., Service oriented enterprises. Auerbach Publications, Boca Raton. 2007.

[47] HÖSS, O., WEISBECKER A., v.d., Migration zu serviceorientierten Architekturen - top-down oder bottom-up? In HMD - Praxis der Wirtschaftsinformatik, 2007; pp. 39-47.

[48] BRABÄNDER, E., KLÜCKMANN, J., Geschäftsprozessmanagement als Grundlage für SOA. OBJEKTspektrum, Ausgabe 5, 2006; pp. 32-37

[49] LIEBHART, D., SOA goes real - Serviceorientierte Architekturen erfolgreich planen und einführen. Carl Hanser Verlag, München. 2007

[50] MARCS, E., BELL, M., Service-Oriented Architecture. A Planning and Implementation Guide for Business and Technology. John Wiley & Sons Inc., New Jersey. 2006.

[51] JOSUTTIS, N., SOA in der Praxis. System-Design für verteilte Geschäftsprozesse. 1. Aufl., dpunkt.verlag Gmbh, Heidelberg. 2008.

[52] BILLING, G., SOA im IT-Service-Management. In SOA-Expertenwissen, Methoden, Konzepte und Praxis serviceorientierter Architekturen. STARKE, G., TILKOV, S. (ed.); 1. Aufl., dpunkt.verlag GmbH, Heidelberg, 2007; pp. 671-680.

[53] BISBAL J., LAWLESS D.,WU B., GRIMSON J., Legacy information system: issues and directions. IEEE Software, 1999; 5(16):103–111.

[54] LEWIS G., MORRIS E., SMITH D., Analyzing the reuse potential of migrating legacy components to a service-oriented architecture. 10th European Conference on SoftwareMaintenance and Reengineering (CSMR 2006), 22–24 March 2006, Bari, Italy IEEE Computer Society, 2006.

[55] GALINIUM M., SHAHBAZ N., Factors affecting success in migration of legacy systems to service-oriented architecture (SOA). Enformatik Böl., Lund Üniversitesi, Sweden. 2009.

[56] MALINVERNO, P., PLUMMER, DC., Magic Quadrant for SOA Governance Technologies, http://www.gartner.com/technology/reprints.do?id=1-17R9KXL &ct=111024&st=sb. Erişim Tarihi: 26.03.2012.

[57] KIZILTAŞ, S., Evaluierung von Soa-Suiten Am Fallbeispiel Eines Unternehmens. İşletme Enformatiği Böl., Marmara Üniversitesi, İstanbul. 2012.

[58] ZIMMERMANN, O., An Architectural Decision Modeling Framework for Service-Oriented Architecture Design. Institut for Architecture and Application Systems. Universität Stuttgart, Almanya. 2009.

[59] ALTUNTAŞ, C., Kötü Tasarımın Belirtileri. http://www.cihataltuntas.com/?p=48. Erişim Tarihi: 03.06.2012

[60] ROBERT CM., MICAH, M., Agile Principles, Patterns, and Practices in C#. Prentice Hall. 2006.

[61] PEREPLETCHIKOV, M., RYAN, C., FRAMPTON, K., SCHMIDT, H., Formalising Service-Oriented Design. Journal of Software, Vol. 3, No. 2. 2008.

[62] ENGELS, G., HESS, A., HUMM, B., JUWIG, O. v.d., Quasar Enterprise, dpunkt.verlag. 2008.

[63] IBM. Building SOA Solutions Using the Rational SDP. IBM Redbooks. 2007.

[64] GEBHART, M., Qualitätsorientierter Entwurf von Anwendungsdiensten. Karlsruher Institut für Technologie, Almanya. 2011.

[65] FENAR, O., SOA'nın Hakimi Olmak (SOA Yönetişimi). http://www.cio-club.net/Makaleler/PDF/SOA_Corner_Haziran.pdf. Erişim Tarihi: 03.06.2012.

[66] ERRADI, A., ANAND, S., KULKARNI, N., SOAF: An Architectural Framework for Service Definition and Realization. SCC '06 Proceedings of the IEEE International Conference on Services Computing. 2006.

[67] OBJECT MANAGEMENT GROUP (OMG). Service oriented architecture Modeling Language (SoaML) Beta1, Specification for the UML Profile and Metamodel for services (UPMS). http://www.omg.org/cgi-bin/doc?soa/2006-9-9. 2009.

[68] HUMM, B., JUWIG, O., Eine Normalform für Services, In Software Engineering 2006. GI Edition Lecture Notes in Informatics (LNI) P-79,. BIEL, B., BOOK, M., GRUHN, V. (ed); Gesellschaft für Informatik, 2006; pp. 99-110.

[69] SOA Maturity Cheat Sheet. http://www.scribd.com/doc/2890015/oraclesoamaturitymodelcheatsheet. Erişim Tarihi: 25.02.2012.

[70] GERIC, S., VRCEK, N., Prerequisities for successful Implementation of Service Oriented Architecture. In Prceedings of the18th International Conference "Information and intelligent systems"; Varazdin 2007; pp. 199 - 207.

[71] BAJWA, IS. vd., SOA and BPM Partnership: A paradigm for Dynamic and Flexible Process and I.T. Management. http://www.waset.org/journals/waset/v45/v45-4.pdf. Erişim Tarihi: 04.08.2011.

[72] AALST, W., Process Mining: The next step in Business Process Management. www.processmining.org/_media/presentations/mining_au_process_days.ppt. Erişim Tarihi: 04.08.2011.

[73] AALST, W., Supporting the Full BPM Life-Cycle Using Process Mining and Intelligent Redesign. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.118.9596&rep=rep1&type=pdf. Erişim Tarihi: 04.08.2011.

[74] AALST, W., Workflow Mining: Discovering process models from event logs. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.77.4049&rep=rep1&type=pdf. Erişim Tarihi: 04.08.2011.

[75] KINDLER, E. vd., Incremental Workflow Mining for Process Flexibility. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.6204&rep=rep1&type=pdf. Erişim Tarihi: 04.08.2011.

[76] ŞAHİN, A., Process Mining and Information Technologies in Mergers and Acquisitions. İşletme Enformatiği Böl., Marmara Üniversitesi, İstanbul. 2011.

[77] SCHMELZER, HJ., SESSELMANN, W., Geschäftsprozessmanagement in der Praxis. Hanser Fachbuchverlag. 2006.

[78] HEINRICH, LJ., HEINZL, A., ROITHMAYR, F., Wirtschaftsinformatik-Lexikon. Oldenbourg. 2004.

[79] SMITH D., Migration of legacy assets to service-oriented architecture environment. 29th International Conference on Software Engineering, 20–26 May 2007, Minneapolis, USA. IEEE Computer Society, 2007.

[80] SÖDERSTRÖM, E., MAIER, F., Combined SOA Maturity Model (CSOAMM): Towards a Guide for SOA Adoption. In Enterprise Interoperability II. New Challenges and Approaches, GONCALVES, R., MÜLLER J., vd. (ed.); Springer Verlag, London 2007; pp. 389-400.

[81] HOFFMAN, D., Software Qualität. Springer Verlag. 2008.

[82] DREIFUS, F., LOOS, P., SOA - Reifegrad - Eine konzeptionelle Darlegung relevanter Erhebungsaspekte. In 2. Workshop Bewertungsaspekte serviceorientierter Architekturen, SCHMIETENDORF, A., MEVIUS, M., vd. (ed.); Shaker Verlag, Aachen 2007; pp. 101-114.

[83] RATHFELDER, C., GROENDA, H., ISOAMM: An Independent SOA Maturity Model. In DAIS 2008, LNCS 5053(2008), MEİER, R., TARZİS, S., (ed.); Springer Berlin/ Heidelberg, 2008; pp. 1-15.

[84] GERIC, S., Service-oriented Architectures Maturity Models. In Proceedings of International Doctoral Seminar; Smolenice, 2008; pp. 80-90.

[85] RUD, D., SCHMIETENDORF, A. vd., Analyse verfügbarer SOA - Reifegradmodelle - State - of - the - Art. In BSOA 2007, 2. Workshop Bewertungsaspekte serviceorientierter Architecturen, SCHMIETENDORF, A., MEVIUS, M., vd. (ed.); Shaker Verlag, Aachen 2007; pp. 115-126.

[86] ARTUT, PD., Experimental evaluation of the effects of cooperative learning on kindergarten children's mathematics ability, doi:10.1016/j.ijer.2010.04.001; International Journal of Educational Research. 2010.

[87] GORJIAN, B., HAYATİ, A., BARAZANDEH, E., An evaluation of the effects of art on vocabulary learning through multi-sensory modalities, doi:10.1016/j.protcy.2012.02.72. Procedia Technology. 2012.

[88] GRUBISIC, A., STANKOV, S., ROSIC, M., ZITKO, B., Controlled Experiment replication in evaluation of e-learning systems educational influence, doi: 10.1016/j.compedu.2009.03.014. Computers & Education. 2009.

[89] MICROSOFT; http://office.microsoft.com/tr-tr/excel-help/HP005203873.aspx; İstatistiksel çözümleme araçları hakkında; Erişim Tarihi: 26.02.2013

[90] NEYMAN J., PEARSON E., On the Problem of the Most Efficient Tests of Statistical Hypotheses. Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character 231: 289–337. doi:10.1098/rsta.1933.0009. JSTOR 91247. 1933.

[91] NEYMAN-PEARSON LEMMA; https://onlinecourses.science.psu.edu/stat414/node/307; PennState University. Erişim Tarihi: 26.02.2013.

[92] FRANK, U., Evaluation von Artefakten in der Wirtschaftsinformatik. In Evaluation und Evaluationsforschung in der Wirtschaftsinformatik, HÄNTSCHEL, I., HEINRICH L. J. (ed.); Oldenburg 2000.

[93] DREIFUS, F., SOA-Value-Management: Entwurf eines Methodenkomplexes zur Bestimmung der Wirtschaftlichkeit serviceorientierter Architekturen. Logos Verlag Berlin. 2009.

[94] SCHMELZER, HJ., SESSELMANN, W., Assessment von Geschäftsprozessen, In Qualität und Zuverlässigkeit (QZ); 43 (1998) 1; pp. 39-42.

[95] AU,YA., CARPENTER, D., CHEN, X, CLARK JG., Virtual organizational learning in open source software development projects; doi:10.1016/j.im.2008.09.004; Information & Management. 2008.

[96] NEWBOLD P., Statistics for Business and Economics. Prentice Hall. 2009.

[97] SAKARYA ÜNİVERSİTESİ, Uzaktan Eğitim Araştırma ve Uygulama Merkezi. http://www.uzem.sakarya.edu.tr/Makaleler.aspx?Makaleid=18. Erişim Tarihi: 02.11.2012.

[98] KULAKSIZ, O.,Webservice Implementierung für SOA. İşletme Enformatiği Böl., Marmara Üniversitesi, İstanbul. 2012.

[99] YAMAN, V., Serviceorientierung von Geschäftsprozessen bei der SOA-Migration. İşletme Enformatiği Böl., Marmara Üniversitesi, İstanbul. 2012.

[100] ARNDT, H., http://www.lehrer-online.de/dyn/bin/499546-502593-1-regeln_ tatischer_prozessmodellierung.pdf. Erişim Tarihi: 03.04.2012.

[101] EREIGNISGESTEUERTE PROZESSKETTE – EPK. http://www.der-wirtschaftsingenieur.de/index.php/ereignisgesteuerte-prozesskette-epk/. Erişim Tarihi: 03.04.2012.

[102] EPK-MODELLIERUNG. http://www.re-wissen.de/opencms/Wissen /Techniken/EPK-Modellierung.html. Erişim Tarihi: 03.04.2012.

[103] EPK-BESCHREIBUNG. https://svn.origo.ethz.ch/rpg-black/Modellierung /EPK_Beschreibung.pdf. Erişim Tarihi: 05.05.2012.

EKLER

Ek A. Uygulamaya Ait Dijital Servis Kodları

Beşinci bölümdeki uygulamaya ait altı adet servisin java programlama dili ile geliştirilmiş kodları aşağıdaki şekildedir [98];

Sınav sonuç servisi:

```
package com.marmara.elearning.examresult.service;

import com.marmara.elearning.examresult.domain.ExamResult;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

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

import javax.jws.WebService;

import javax.xml.ws.BindingType;
import javax.xml.ws.soap.SOAPBinding;

@WebService(serviceName = "ExamResultService", portName =
"ExamResultServiceSoap12HttpPort")
@BindingType(SOAPBinding.SOAP12HTTP_BINDING)
public class ExamResultService {

    public String createExamResult(ExamResult examResult) {
        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("INSERT INTO EXAM_RESULTS VALUES ( \n");
            sql.append("exam_result_id_seq.NEXTVAL, \n");
            sql.append("'" + examResult.getStudentId() + "', \n");
            sql.append("'" + examResult.getLessonId() + "', \n");
```

```
            sql.append("'" + examResult.getExamId() + "', \n");
            sql.append("'" + examResult.getNote() + "')");

            statement.execute(sql.toString());
            statement.close();

            con.close();
        } catch (Exception e) {
            e.printStackTrace();
            return "Failure";
        }
        return "Success";
    }

    public ExamResult loadExamResultById(String examResultID) {
        ExamResult examResult = new ExamResult();

        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con                                                    =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("SELECT * FROM EXAM_RESULTS \n");
            sql.append("WHERE ID = '" + examResultID + "'");

            ResultSet rs = statement.executeQuery(sql.toString());
            if (rs.next()) {
                examResult.setId(rs.getString("ID"));
                examResult.setExamId(rs.getString("EXAM_ID"));
                examResult.setStudentId(rs.getString("STUDENT_ID"));
                examResult.setLessonId(rs.getString("LESSON_ID"));
                examResult.setNote(rs.getString("NOTE"));
            }

            statement.close();
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
        }

        return examResult;
    }

    public String deleteExamResult(String examResultId) {
        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con                                                    =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("DELETE FROM EXAM_RESULTS \n");
            sql.append("WHERE ID = '" + examResultId + "'");
```

```
            statement.execute(sql.toString());
            statement.close();

            con.close();
        } catch (Exception e) {
            e.printStackTrace();
            return "Failure";
        }
        return "Success";
    }
    public String updateExamResult(ExamResult examResult) {
        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con                                                    =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("UPDATE EXAM_RESULTS SET \n");
            sql.append("EXAM_ID = '" + examResult.getExamId() + "',
\n");
            sql.append("STUDENT_ID = '" + examResult.getStudentId()
+ "', \n");
            sql.append("LESSON_ID = '" + examResult.getLessonId() +
"', \n");
            sql.append("NOTE = '" + examResult.getNote() + "', \n");
            sql.append("WHERE ID = '" + examResult.getId() + "'");

            statement.execute(sql.toString());
            statement.close();

            con.close();
        } catch (Exception e) {
            e.printStackTrace();
            return "Failure";
        }
        return "Success";
    }

    public List<ExamResult> getAllResultsOfStudent(String studentID,
String year) {
        List<ExamResult> examResults = new ArrayList<ExamResult>();

        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con                                                    =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara",  "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("SELECT * FROM EXAM_RESULTS er, EXAMS e \n");
            sql.append("WHERE STUDENT_ID = '" + studentID + "' AND
er.EXAM_ID = e.ID AND e.YEAR = '" + year + "'");

            ResultSet rs = statement.executeQuery(sql.toString());
            while (rs.next()) {
```

```
                ExamResult examResult = new ExamResult();
                examResult = loadExamResultById(rs.getString("ID"));
                examResults.add(examResult);
            }

            statement.close();
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
        }

        return examResults;
    }

    public List<ExamResult> getAllResultsOfLesson(String lessonID,
String year) {
        List<ExamResult> examResults = new ArrayList<ExamResult>();

        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con                                                    =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara",  "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("SELECT * FROM EXAM_RESULTS er, EXAMS e \n");
            sql.append("WHERE er.LESSON_ID = '" + lessonID + "' AND
er.EXAM_ID = e.ID AND e.YEAR = '" + year +  "'");

            ResultSet rs = statement.executeQuery(sql.toString());
            while (rs.next()) {
                ExamResult examResult = new ExamResult();
                examResult = loadExamResultById(rs.getString("ID"));
                examResults.add(examResult);
            }

            statement.close();
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
        }

        return examResults;
    }

    public   List<ExamResult>   getAllResultsOfExam(String   examID,
String year) {
        List<ExamResult> examResults = new ArrayList<ExamResult>();

        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con                                                    =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara",  "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
```

```
            sql.append("SELECT * FROM EXAM_RESULTS er, EXAMS e \n");
            sql.append("WHERE  er.EXAM_ID  =  '"  +  examID  +  "'  AND
er.EXAM_ID = e.ID AND e.YEAR = '" + year +  "'");

            ResultSet rs = statement.executeQuery(sql.toString());
            while (rs.next()) {
                ExamResult examResult = new ExamResult();
                examResult = loadExamResultById(rs.getString("ID"));
                examResults.add(examResult);
            }

            statement.close();
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
        }

        return examResults;
    }

}
```

Sınav servisi:

```
package com.marmara.elearning.exam.service;


import com.marmara.elearning.exam.domain.Exam;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

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

import javax.jws.WebService;

import javax.xml.ws.BindingType;
import javax.xml.ws.soap.SOAPBinding;

@WebService(serviceName = "ExamService", portName =
"ExamServiceSoap12HttpPort")
@BindingType(SOAPBinding.SOAP12HTTP_BINDING)
public class ExamService {

    public String createExam(Exam exam) {
        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("INSERT INTO EXAMS VALUES ( \n");
            sql.append("exam_id_seq.NEXTVAL, \n");
            sql.append("'" + exam.getWeight() +"', \n");
            sql.append("'" + exam.getTerm() +"', \n");
            sql.append("'" + exam.getType() +"', \n");
            sql.append("'" + exam.getYear() +"', \n");
            sql.append("'" + exam.getLessonId() + "')");

            statement.execute(sql.toString());
            statement.close();

            con.close();
        } catch(Exception e){
            e.printStackTrace();
            return "Failure";
        }
        return "Success";
    }

    public Exam loadExamById(String id) {
        Exam exam = new Exam();

        try {
```

```
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con                                                    =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("SELECT * FROM EXAMS \n");
            sql.append("WHERE ID = '" + id + "'");

            ResultSet rs = statement.executeQuery(sql.toString());
            if (rs.next()) {
                exam.setId(rs.getString("ID"));
                exam.setTerm(rs.getString("TERM"));
                exam.setType(rs.getString("TYPE"));
                exam.setWeight(rs.getString("WEIGHT"));
                exam.setYear(rs.getString("YEAR"));
                exam.setLessonId(rs.getString("LESSON_ID"));
            }

            statement.close();
            con.close();
        } catch(Exception e){
            e.printStackTrace();
        }

        return exam;
    }

    public String deleteExam(String examID) {
        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con                                                    =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("DELETE FROM EXAMS \n");
            sql.append("WHERE ID = '" + examID + "'");

            statement.execute(sql.toString());
            statement.close();

            con.close();
        } catch(Exception e){
            e.printStackTrace();
            return "Failure";
        }
        return "Success";
    }

    public String updateExam(Exam exam) {
        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
            con                                                    =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("UPDATE EXAMS SET \n");
            sql.append("WEIGHT = '" + exam.getWeight() + "', \n");
            sql.append("TERM = '" + exam.getTerm() + "', \n");
            sql.append("TYPE = '" + exam.getType() + "', \n");
            sql.append("YEAR = '" + exam.getYear() + "', \n");
            sql.append("LESSON_ID = '" + exam.getLessonId() + "'
\n");
            sql.append("WHERE ID = '" + exam.getId() + "'");

            statement.execute(sql.toString());
            statement.close();

            con.close();
        } catch(Exception e){
            e.printStackTrace();
            return "Failure";
        }
        return "Success";
    }

    public List<Exam> getExamsOfLesson(String lessonID, String year)
{
        List<Exam> exams = new ArrayList<Exam>();

        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con                                                    =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("SELECT * FROM EXAMS \n");
            sql.append("WHERE LESSON_ID = '" + lessonID + "' AND
YEAR = '" + year + "'");

            ResultSet rs = statement.executeQuery(sql.toString());
            while (rs.next()) {
                Exam exam = new Exam();
                exam.setId(rs.getString("ID"));
                exam.setTerm(rs.getString("TERM"));
                exam.setType(rs.getString("TYPE"));
                exam.setWeight(rs.getString("WEIGHT"));
                exam.setYear(rs.getString("YEAR"));
                exam.setLessonId(rs.getString("LESSON_ID"));
                exams.add(exam);
            }

            statement.close();
            con.close();
        } catch(Exception e){
            e.printStackTrace();
        }
```

```
            return exams;
        }

    }
```

Ders servisi:

```
package com.marmara.elearning.lesson.service;

import com.marmara.elearning.lesson.domain.Lesson;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

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

import javax.jws.Oneway;
import javax.jws.WebMethod;
import javax.jws.WebService;

import javax.xml.ws.BindingType;
import javax.xml.ws.soap.SOAPBinding;

@WebService(serviceName = "LessonService", portName =
"LessonServiceSoap12HttpPort")
@BindingType(SOAPBinding.SOAP12HTTP_BINDING)
public class LessonService {

    @WebMethod
    public String createLesson(Lesson lesson) {
        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("INSERT INTO LESSONS VALUES ( \n");
            sql.append("lesson_id_seq.NEXTVAL, \n");
            sql.append("'" + lesson.getName() + "', \n");
            sql.append("'" + lesson.getDescription() + "', \n");
            sql.append("'" + lesson.getSemester() + "')");

            statement.execute(sql.toString());
            statement.close();

            con.close();
        } catch (Exception e) {
            e.printStackTrace();
            return "Failure";
        }
        return "Success";
    }

    @WebMethod
    public Lesson loadLessonById(String id) {
        Lesson lesson = new Lesson();
```

```
        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara",
                                "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("SELECT * FROM LESSONS \n");
            sql.append("WHERE ID = '" + id + "'");

            ResultSet rs = statement.executeQuery(sql.toString());
            if (rs.next()) {
                lesson.setId(rs.getString("ID"));
                lesson.setName(rs.getString("NAME"));
                lesson.setDescription(rs.getString("DESCRIPTION"));
                lesson.setSemester(rs.getString("SEMESTER"));
            }

            statement.close();
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
        }

        return lesson;
    }

    @WebMethod
    public String deleteLesson(String id) {
        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara",
                                "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("DELETE FROM LESSONS \n");
            sql.append("WHERE ID = '" + id + "'");

            statement.execute(sql.toString());
            statement.close();

            con.close();
        } catch (Exception e) {
            e.printStackTrace();
            return "Failure";
        }
        return "Success";
    }

    @WebMethod
    public String updateLesson(Lesson lesson) {
        try {
            Connection con = null;
```

```
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara",
                                "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("UPDATE LESSONS SET \n");
            sql.append("NAME = '" + lesson.getName() + "', \n");
            sql.append("DESCRIPTION = '" + lesson.getDescription() +
"', \n");
            sql.append("SEMESTER = '" + lesson.getSemester() + "'
\n");
            sql.append("WHERE ID = '" + lesson.getId() + "'");

            statement.execute(sql.toString());
            statement.close();

            con.close();
        } catch (Exception e) {
            e.printStackTrace();
            return "Failure";
        }
        return "Success";
    }

    @WebMethod
    public List<Lesson> getAllLessons(){
        List<Lesson> lessons = new ArrayList<Lesson>();

        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con                                                    =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("SELECT * FROM LESSONS");

            ResultSet rs = statement.executeQuery(sql.toString());
            while (rs.next()) {
                Lesson lesson = new Lesson();
                lesson.setId(rs.getString("ID"));
                lesson.setName(rs.getString("NAME"));
                lesson.setDescription(rs.getString("DESCRIPTION"));
                lesson.setSemester(rs.getString("SEMESTER"));
                lessons.add(lesson);
            }

            statement.close();
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
        }

        return lessons;
    }
```

```
    @WebMethod
    public List<Lesson> getLessonsOfProfessor(String professorID) {
        List<Lesson> lessons = new ArrayList<Lesson>();

        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con                                                    =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("SELECT * FROM PROFESSOR_LESSON \n");
            sql.append("WHERE  PROFESSOR_ID  =  '"  +  professorID  +
"'");

            ResultSet rs = statement.executeQuery(sql.toString());
            while (rs.next()) {
                Lesson lesson = new Lesson();
                lesson = loadLessonById(rs.getString("LESSON_ID"));
                lessons.add(lesson);
            }

            statement.close();
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
        }

        return lessons;
    }

    @WebMethod
    public List<Lesson> getLessonsOfStudent(String studentID) {
        List<Lesson> lessons = new ArrayList<Lesson>();

        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara",
                            "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("SELECT * FROM STUDENT_LESSON \n");
            sql.append("WHERE STUDENT_ID = '" + studentID + "'");

            ResultSet rs = statement.executeQuery(sql.toString());
            while (rs.next()) {
                Lesson lesson = new Lesson();
                lesson = loadLessonById(rs.getString("LESSON_ID"));
                lessons.add(lesson);
            }

            statement.close();
            con.close();
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }

        return lessons;
    }

    @WebMethod
    public String addLessonToProfessor(String professorID, String
lessonID) {
        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara",
                                "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("INSERT INTO PROFESSOR_LESSON VALUES ( \n");
            sql.append("'" + professorID + "', \n");
            sql.append("'" + lessonID + "')");

            statement.execute(sql.toString());
            statement.close();

            con.close();
        } catch (Exception e) {
            e.printStackTrace();
            return "Failure";
        }
        return "Success";
    }

    @WebMethod
    public String removeLessonFromProfessor(String professorID,
String lessonID) {
        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con                                                    =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("DELETE FROM PROFESSOR_LESSON \n");
            sql.append("WHERE PROFESSOR_ID = '" + professorID + "'
AND \n");
            sql.append("LESSON_ID = '" + lessonID + "'");

            statement.execute(sql.toString());
            statement.close();

            con.close();
        } catch (Exception e) {
            e.printStackTrace();
            return "Failure";
        }
```

```
        return "Success";
    }

    @Oneway
    @WebMethod
    public void removeAllLessonsOfProfessor(String professorID){
        List<Lesson> lessons = getLessonsOfProfessor(professorID);
        for (Lesson lesson : lessons){
            removeLessonFromProfessor(professorID, lesson.getId());
        }
    }

    @WebMethod
    public String addLessonToStudent(String studentID, String
lessonID) {
        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con                                                    =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("INSERT INTO STUDENT_LESSON VALUES ( \n");
            sql.append("'" + studentID + "', \n");
            sql.append("'" + lessonID + "')");

            statement.execute(sql.toString());
            statement.close();

            con.close();
        } catch (Exception e) {
            e.printStackTrace();
            return "Failure";
        }
        return "Success";
    }

    @WebMethod
    public String removeLessonFromStudent(String studentID, String
lessonID) {
        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con                                                    =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("DELETE FROM STUDENT_LESSON \n");
            sql.append("WHERE STUDENT_ID = '" + studentID + "' AND
\n");
            sql.append("LESSON_ID = '" + lessonID + "'");

            statement.execute(sql.toString());
            statement.close();

            con.close();
```

```
        } catch (Exception e) {
            e.printStackTrace();
            return "Failure";
        }
        return "Success";
    }

    @Oneway
    @WebMethod
    public void removeAllLessonsOfStudent(String studentID){
        List<Lesson> lessons = getLessonsOfStudent(studentID);
        for (Lesson lesson : lessons){
            removeLessonFromStudent(studentID, lesson.getId());
        }
    }

}
```

Profesör servisi:

```
package org.marmara.elearning.professor.service;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;
import java.sql.Statement;

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

import javax.jws.WebService;

import javax.xml.ws.BindingType;
import javax.xml.ws.soap.SOAPBinding;

import org.marmara.elearning.professor.domain.Professor;

@WebService(serviceName = "ProfessorService", portName =
"ProfessorServiceSoap12HttpPort")
@BindingType(SOAPBinding.SOAP12HTTP_BINDING)
public class ProfessorService {
    public String createProfessor(Professor professor) {
            try {
                Connection con = null;
                Class.forName("oracle.jdbc.driver.OracleDriver");
                con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
                Statement statement = con.createStatement();

                StringBuffer sql = new StringBuffer();
                sql.append("INSERT INTO PROFESSORS VALUES ( \n");
                sql.append("professor_id_seq.NEXTVAL, \n");
                sql.append("'" + professor.getUsername() + "', \n");
                sql.append("'" + professor.getPassword() + "', \n");
                sql.append("'" + professor.getFirstname() +"', \n");
                sql.append("'" + professor.getSurname() + " )");

                statement.execute(sql.toString());
                statement.close();

                con.close();
            } catch(Exception e){
                e.printStackTrace();
                return "Failure";
            }
            return "Success";
        }

        public Professor loadProfessorByUsername(String username) {
            Professor professor = new Professor();

            try {
                Connection con = null;
```

```
                Class.forName("oracle.jdbc.driver.OracleDriver");
                con                                                    =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
                Statement statement = con.createStatement();

                StringBuffer sql = new StringBuffer();
                sql.append("SELECT * FROM PROFESSORS \n");
                sql.append("WHERE USERNAME = '" + username + "'");

                ResultSet                    rs                        =
statement.executeQuery(sql.toString());
                if (rs.next()) {
                    professor.setId(rs.getString("ID"));

professor.setFirstname(rs.getString("FIRSTNAME"));
                    professor.setSurname(rs.getString("SURNAME"));
                    professor.setUsername(rs.getString("USERNAME"));
                    professor.setPassword(rs.getString("PASSWORD"));
                }

                statement.close();
                con.close();
            } catch(Exception e){
                e.printStackTrace();
            }

            return professor;
        }
        public Professor loadProfessorById(String id) {
            Professor professor = new Professor();

            try {
                Connection con = null;
                Class.forName("oracle.jdbc.driver.OracleDriver");
                con                                                    =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
                Statement statement = con.createStatement();

                StringBuffer sql = new StringBuffer();
                sql.append("SELECT * FROM PROFESSORS \n");
                sql.append("WHERE ID = '" + id + "'");

                ResultSet                    rs                        =
statement.executeQuery(sql.toString());
                if (rs.next()) {
                    professor.setId(rs.getString("ID"));

professor.setFirstname(rs.getString("FIRSTNAME"));
                    professor.setSurname(rs.getString("SURNAME"));
                    professor.setUsername(rs.getString("USERNAME"));
                    professor.setPassword(rs.getString("PASSWORD"));
                }

                statement.close();
                con.close();
            } catch(Exception e){
                e.printStackTrace();
            }
```

```
            return professor;
        }

        public String deleteProfessor(String id) {
            try {
                Connection con = null;
                Class.forName("oracle.jdbc.driver.OracleDriver");
                con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
                Statement statement = con.createStatement();

                StringBuffer sql = new StringBuffer();
                sql.append("DELETE FROM PROFESSORS \n");
                sql.append("WHERE ID = '" + id + "'");

                statement.execute(sql.toString());
                statement.close();

                con.close();
            } catch(Exception e){
                e.printStackTrace();
                return "Failure";
            }
            return "Success";
        }

        public String updateProfessor(Professor professor) {
            try {
                Connection con = null;
                Class.forName("oracle.jdbc.driver.OracleDriver");
                con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
                Statement statement = con.createStatement();

                StringBuffer sql = new StringBuffer();
                sql.append("UPDATE PROFESSORS SET \n");
                sql.append("USERNAME = '" + professor.getUsername()
+ "', \n");
                sql.append("PASSWORD = '" + professor.getPassword()
+ "', \n");
                sql.append("FIRSTNAME = '" +
professor.getFirstname() + "', \n");
                sql.append("SURNAME = '" + professor.getSurname() +
"' \n");
                sql.append("WHERE ID = '" + professor.getId() +
"'");

                statement.execute(sql.toString());
                statement.close();

                con.close();
            } catch(Exception e){
                e.printStackTrace();
                return "Failure";
            }
            return "Success";
        }
```

```
        public List<Professor> getAllProfessors(){
            List<Professor> professors = new ArrayList<Professor>();

            try {
                Connection con = null;
                Class.forName("oracle.jdbc.driver.OracleDriver");
                con                                                    =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
                Statement statement = con.createStatement();

                StringBuffer sql = new StringBuffer();
                sql.append("SELECT * FROM PROFESSORS");

                ResultSet                    rs                        =
statement.executeQuery(sql.toString());
                while (rs.next()) {
                    Professor professor = new Professor();
                    professor.setId(rs.getString("ID"));

professor.setFirstname(rs.getString("FIRSTNAME"));
                    professor.setSurname(rs.getString("SURNAME"));
                    professor.setUsername(rs.getString("USERNAME"));
                    professor.setPassword(rs.getString("PASSWORD"));
                    professors.add(professor);
                }

                statement.close();
                con.close();
            } catch(Exception e){
                e.printStackTrace();
            }

            return professors;
        }

        public      List<Professor>      getProfessorsOfLesson(String
lessonID) {
            List<Professor> professors = new ArrayList<Professor>();

            try {
                Connection con = null;
                Class.forName("oracle.jdbc.driver.OracleDriver");
                con                                                    =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
                Statement statement = con.createStatement();

                StringBuffer sql = new StringBuffer();
                sql.append("SELECT * FROM PROFESSOR_LESSON \n");
                sql.append("WHERE LESSON_ID = '" + lessonID + "'");

                ResultSet                    rs                        =
statement.executeQuery(sql.toString());
                while (rs.next()) {
                    Professor student = new Professor();
                    student                                            =
loadProfessorById(rs.getString("PROFESSOR_ID"));
                    professors.add(student);
                }
```

```
            statement.close();
            con.close();
        } catch(Exception e){
            e.printStackTrace();
        }

        return professors;
    }
}
```

Öğrenci servisi:

```
package com.marmara.elearning.student.service;

import com.marmara.elearning.student.domain.Student;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;
import java.sql.Statement;

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

import javax.jws.WebService;

import javax.xml.ws.BindingType;
import javax.xml.ws.soap.SOAPBinding;

@WebService(serviceName = "StudentService",
            portName = "StudentServiceSoap12HttpPort")
@BindingType(SOAPBinding.SOAP12HTTP_BINDING)
public class StudentService {
    public String createStudent(Student student) {
        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara",
                                "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("INSERT INTO STUDENTS VALUES ( \n");
            sql.append("student_id_seq.NEXTVAL, \n");
            sql.append("'" + student.getUsername() + "', \n");
            sql.append("'" + student.getPassword() + "', \n");
            sql.append("'" + student.getFirstname() + "', \n");
            sql.append("'" + student.getSurname() + "', \n");
            sql.append("'" + student.getSchoolEntranceYear() + "',
\n");
            sql.append("'" + student.getStudentNo() + "')");

            statement.execute(sql.toString());
            statement.close();

            con.close();
        } catch (Exception e) {
            e.printStackTrace();
            return "Failure";
        }
        return "Success";
    }

    public Student loadStudentByUsername(String username) {
```

```
        Student student = new Student();

        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara",
                                "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("SELECT * FROM STUDENTS \n");
            sql.append("WHERE USERNAME = '" + username + "'");

            ResultSet rs = statement.executeQuery(sql.toString());
            if (rs.next()) {
                student.setId(rs.getString("ID"));
                student.setFirstname(rs.getString("FIRSTNAME"));
                student.setSurname(rs.getString("SURNAME"));
                student.setUsername(rs.getString("USERNAME"));
                student.setPassword(rs.getString("PASSWORD"));

student.setSchoolEntranceYear(rs.getString("ENTRANCE_YEAR"));
                student.setStudentNo(rs.getString("STUDENT_NO"));
            }

            statement.close();
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
        }

        return student;
    }

    public Student loadStudentById(String id) {
        Student student = new Student();

        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara",
                                "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("SELECT * FROM STUDENTS \n");
            sql.append("WHERE ID = '" + id + "'");

            ResultSet rs = statement.executeQuery(sql.toString());
            if (rs.next()) {
                student.setId(rs.getString("ID"));
                student.setFirstname(rs.getString("FIRSTNAME"));
                student.setSurname(rs.getString("SURNAME"));
                student.setUsername(rs.getString("USERNAME"));
                student.setPassword(rs.getString("PASSWORD"));
```

```
student.setSchoolEntranceYear(rs.getString("ENTRANCE_YEAR"));
                student.setStudentNo(rs.getString("STUDENT_NO"));
            }

            statement.close();
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
        }

        return student;
    }

    public String deleteStudent(String id) {
        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara",
                                "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("DELETE FROM STUDENTS \n");
            sql.append("WHERE ID = '" + id + "'");

            statement.execute(sql.toString());
            statement.close();

            con.close();
        } catch (Exception e) {
            e.printStackTrace();
            return "Failure";
        }
        return "Success";
    }

    public String updateStudent(Student student) {
        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara",
                                "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("UPDATE STUDENTS SET \n");
            sql.append("USERNAME = '" + student.getUsername() + "',
\n");
            sql.append("PASSWORD = '" + student.getPassword() + "',
\n");
            sql.append("FIRSTNAME = '" + student.getFirstname() +
"', \n");
            sql.append("SURNAME = '" + student.getSurname() + "',
\n");
```

```
            sql.append("ENTRANCE_YEAR = '" + student.getSurname() +
"', \n");
            sql.append("STUDENT_NO = '" + student.getStudentNo() +
"' ");
            sql.append("WHERE ID = '" + student.getId() + "'");

            statement.execute(sql.toString());
            statement.close();

            con.close();
        } catch (Exception e) {
            e.printStackTrace();
            return "Failure";
        }
        return "Success";
    }

    public List<Student> getAllStudents() {
        List<Student> students = new ArrayList<Student>();

        try {
            Connection con = null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara",
                                "123456");
            Statement statement = con.createStatement();

            StringBuffer sql = new StringBuffer();
            sql.append("SELECT * FROM STUDENTS");

            ResultSet rs = statement.executeQuery(sql.toString());
            while (rs.next()) {
                Student student = new Student();
                student.setId(rs.getString("ID"));
                student.setFirstname(rs.getString("FIRSTNAME"));
                student.setSurname(rs.getString("SURNAME"));
                student.setUsername(rs.getString("USERNAME"));
                student.setPassword(rs.getString("PASSWORD"));

student.setSchoolEntranceYear(rs.getString("ENTRANCE_YEAR"));
                student.setStudentNo(rs.getString("STUDENT_NO"));
                students.add(student);
            }

            statement.close();
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
        }

        return students;
    }
}
```

Giriş servisi:

```
package com.marmara.elearning.authorization.service;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

import javax.jws.WebService;

import javax.xml.ws.BindingType;
import javax.xml.ws.soap.SOAPBinding;

@WebService(portName = "LoginSoap12HttpPort")
@BindingType(SOAPBinding.SOAP12HTTP_BINDING)
public class Login {

    public String checkUser(String username, String password){
        Connection con = null;
        Statement statement = null;
        try {

            Class.forName("oracle.jdbc.driver.OracleDriver");
            con                                                    =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"marmara", "123456");
            statement = con.createStatement();

            StringBuffer studentSQL = new StringBuffer();
            studentSQL.append("SELECT * FROM STUDENTS \n");
            studentSQL.append("WHERE  USERNAME  =  '"  +  username  +
"'");

            ResultSet                        studentRS                 =
statement.executeQuery(studentSQL.toString());
            if (studentRS.next()) {
                if
(password.equals(studentRS.getString("PASSWORD"))){
                    return "Student," + studentRS.getString("ID");
                } else {
                    return "Failure";
                }
            }

            StringBuffer sql = new StringBuffer();
            sql.append("SELECT * FROM PROFESSORS \n");
            sql.append("WHERE USERNAME = '" + username + "'");

            ResultSet                     professorRS                  =
statement.executeQuery(sql.toString());
            if (professorRS.next()) {
                if
(password.equals(professorRS.getString("PASSWORD"))){
                    return                "Professor,"                 +
professorRS.getString("ID");
                } else {
                    return "Failure";
```

```
                }
            }
        } catch(Exception e){
            e.printStackTrace();
        } finally {
            try {
                statement.close();
                con.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return "Failure";
    }

}
```

Ek B. İş Süreçlerine SOA Uygulama Metodolojisi Değerlendirme Sonuçları

Tablo E.1. Örnek Olay Çalışmasından Elde Edilen Tüm Sonuçlar

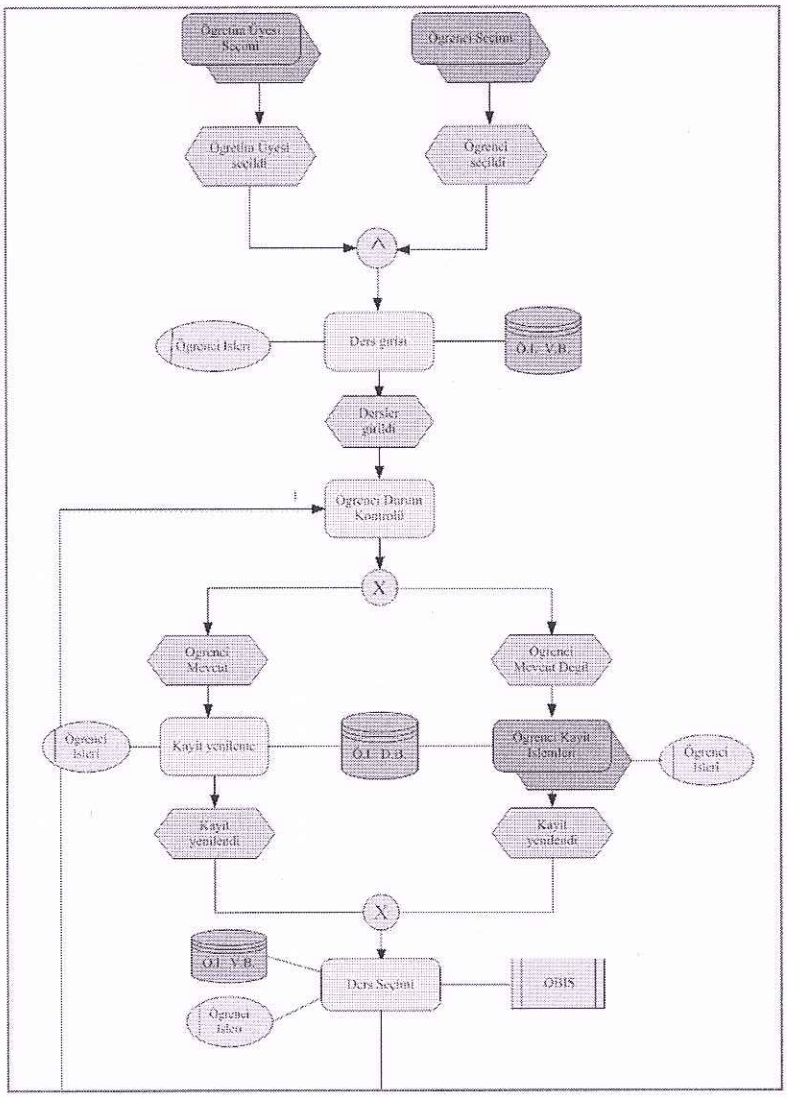
	Metodoloji Öncesi Değerler (saat)	Metodoloji Sonrası Değerler (saat)	MÖD X 0.35	MSD + (MÖD X 0.35)
Katılımcı 1	3	1.25	1.05	2.30
Katılımcı 2	3	1.25	1.05	2.30
Katılımcı 3	3.2	1.25	1.12	2.37
Katılımcı 4	2.28	1.25	0.798	2.05
Katılımcı 5	2.8	1.24	0.98	2.22
Katılımcı 6	2.4	1.24	0.84	2.08
Katılımcı 7	3	1.24	1.05	2.29
Katılımcı 8	2.86	1.23	1.001	2.23
Katılımcı 9	2.21	1.23	0.7735	2.00
Katılımcı 10	2.4	1.22	0.84	2.06
Katılımcı 11	2.6	1.22	0.91	2.13
Katılımcı 12	2.73	1.21	0.9555	2.17
Katılımcı 13	2.73	1.21	0.9555	2.17
Katılımcı 14	2.93	1.16	1.0255	2.19
Katılımcı 15	2.7	1.15	0.945	2.10
Katılımcı 16	2.22	1.1	0.777	1.88
Katılımcı 17	2.5	1.1	0.875	1.98
Katılımcı 18	2.4	1.1	0.84	1.94
Katılımcı 19	2.3	1.09	0.805	1.90
Katılımcı 20	3.1	1.09	1.085	2.18
Katılımcı 21	3.4	1.08	1.19	2.27
Katılımcı 22	2.68	1.05	0.938	1.99
Katılımcı 23	3.3	1.04	1.155	2.20
Katılımcı 24	3.4	1.03	1.19	2.22
Katılımcı 25	2.11	1.03	0.7385	1.77
Katılımcı 26	2.4	1.02	0.84	1.86
Katılımcı 27	2.3	1.02	0.805	1.83
Katılımcı 28	3.3	1.01	1.155	2.17
Katılımcı 29	3.4	1.01	1.19	2.20
Katılımcı 30	3.3	1	1.155	2.16
Katılımcı 31	3.2	1	1.12	2.12
Katılımcı 32	3.3	1	1.155	2.16
Katılımcı 33	2.18	1	0.763	1.76
Katılımcı 34	2.2	0.99	0.77	1.76
Katılımcı 35	3.3	0.98	1.155	2.14
Katılımcı 36	2.6	0.98	0.91	1.89
Katılımcı 37	2.9	0.97	1.015	1.99
Katılımcı 38	2.8	0.97	0.98	1.95
Katılımcı 39	2.7	0.97	0.945	1.92
Katılımcı 40	2.6	0.97	0.91	1.88
Katılımcı 41	2.21	0.96	0.7735	1.73
Katılımcı 42	2.21	0.96	0.7735	1.73
Katılımcı 43	3.1	0.75	1.085	1.84
Katılımcı 44	2.1	0.87	0.735	1.61
Katılımcı 45	2.08	0.89	0.728	1.62
Katılımcı 46	2.06	0.79	0.721	1.51
Katılımcı 47	2.15	0.78	0.7525	1.53
Katılımcı 48	2.11	0.71	0.7385	1.45
Katılımcı 49	2.13	0.71	0.7455	1.46
Katılımcı 50	2.11	0.71	0.7385	1.45

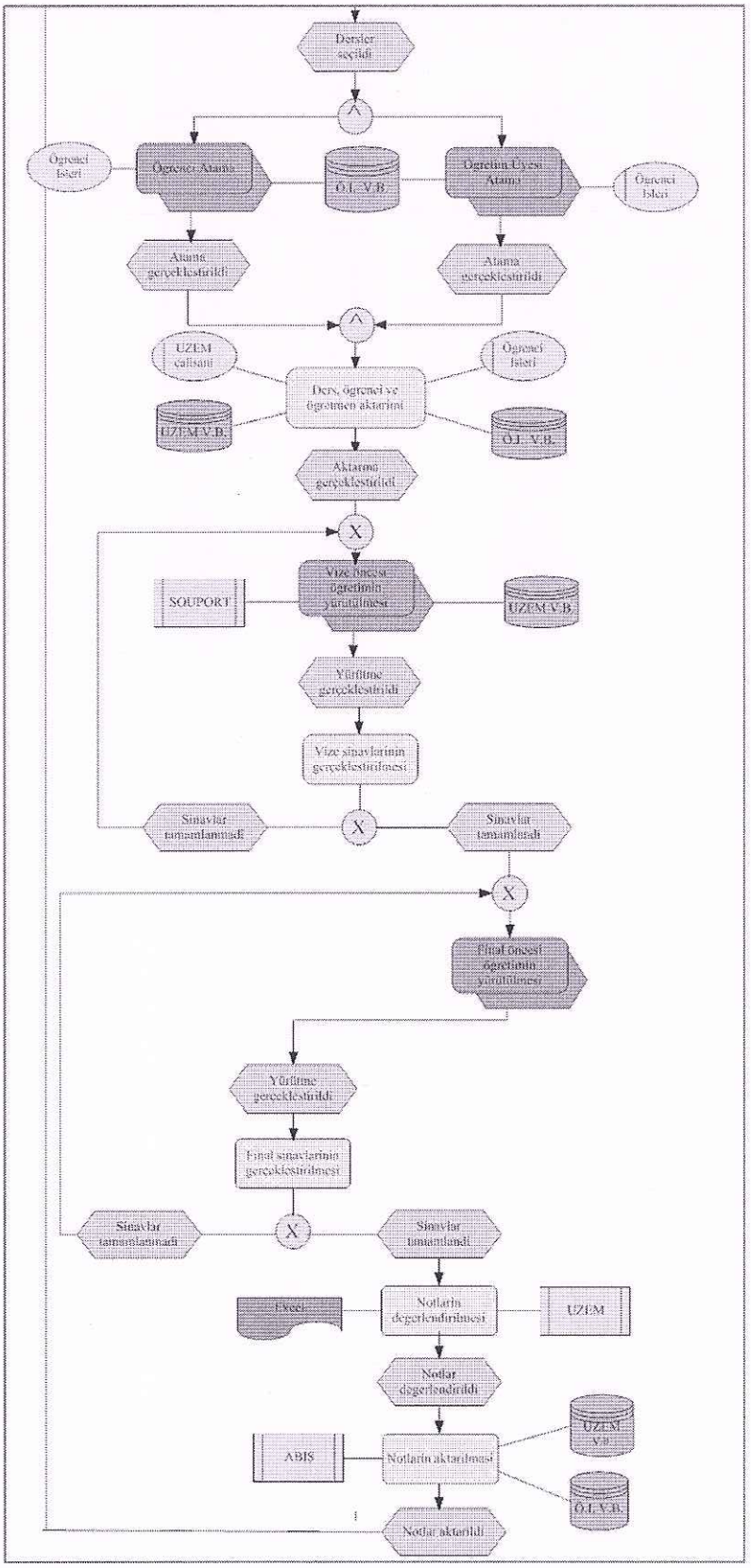
Ek C. eEPK ile Modellenen İş Süreçleri

Erweiterte Ereignisgesteuerte Prozesskette - eEPK (İng.: Event-driven process chain) Saarland Üniversitesi Enformatik Enstitüsü tarafından iş süreçlerinin görselleştirilebilmesi için geliştirilen grafiksel bir modelleme dilidir. [99], [100], [101],[102],[103].

Bölüm 5'teki uygulamada kullanılan iş süreçlerinin, eEPK ile modellenmiş halleri aşağıdaki gibidir;

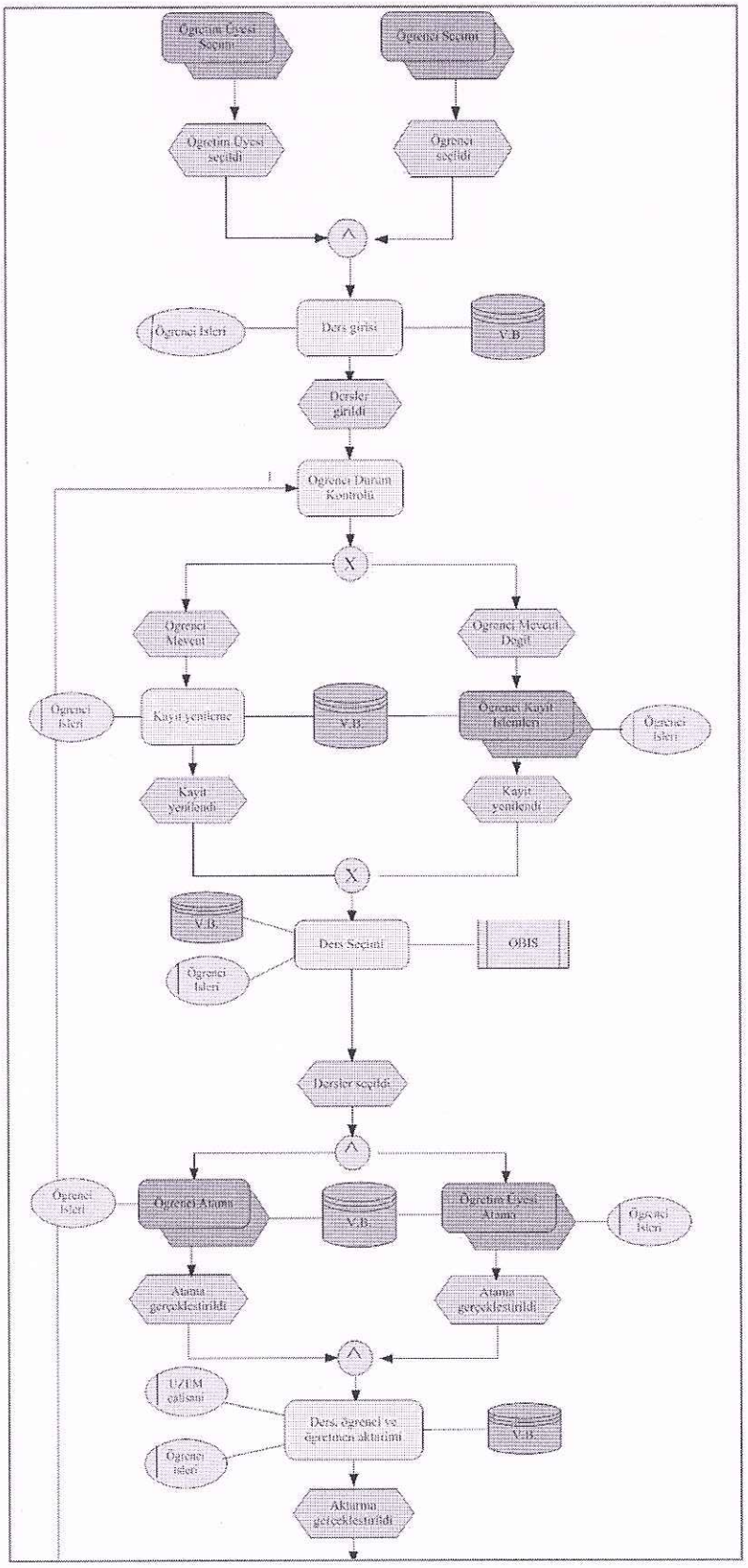
Mevcut iş süreç modeli:

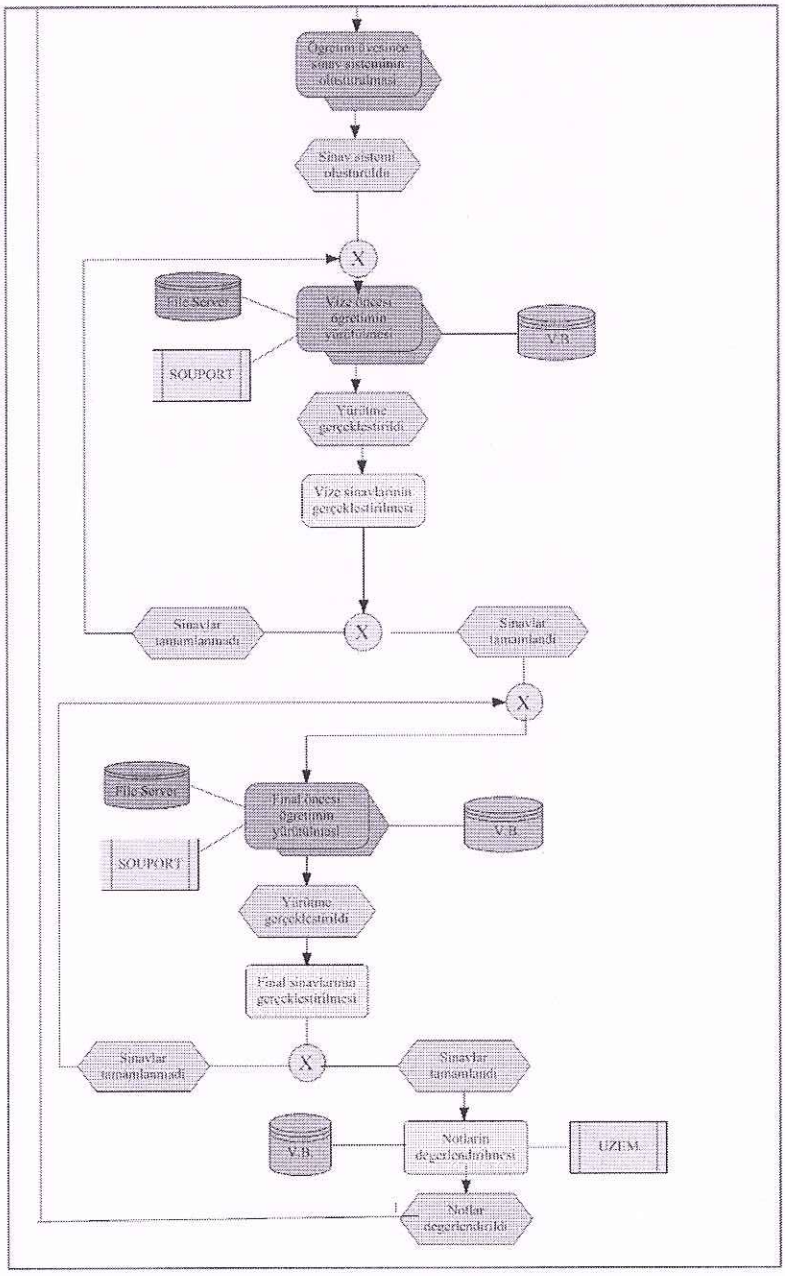




Şekil E.1: Uzaktan Eğitim Merkezi'nin Mevcut İş Süreç Modeli [99]

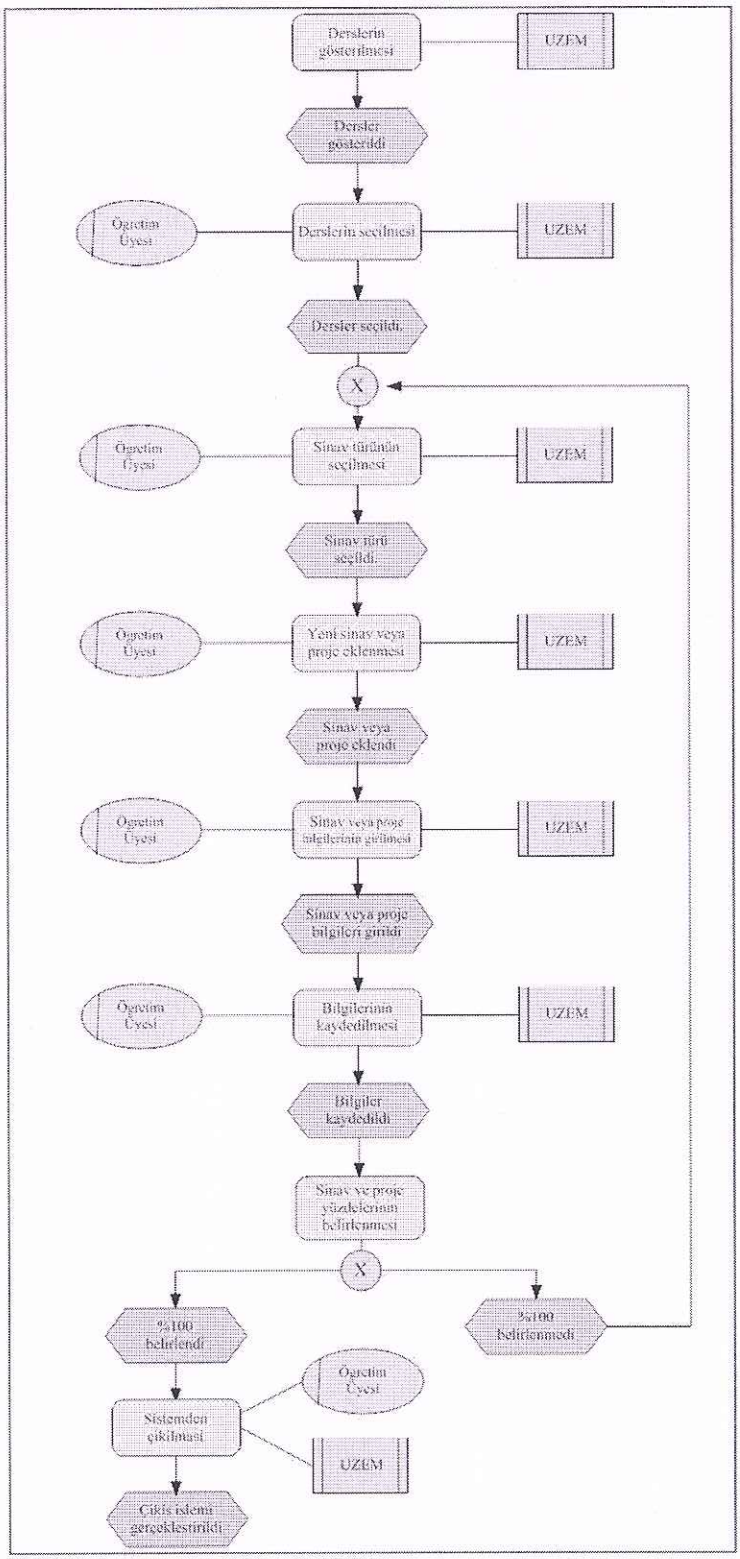
Hedef iş süreç modeli:





Şekil E.2: Uzaktan Eğitim Merkezi'nin Hedef İş Süreç Modeli [99]

Sınav sisteminin oluşturulması sürecine ait iş süreç modeli:



Şekil E.3: Sınav Sistemi'nin Oluşturulması Sürecine Ait İş Süreç Modeli [99]

ÖZGEÇMİŞ

Deniz Herand, 25.09.1980 tarihinde Bulgaristan'ın İsperih (Kemaller) şehrinde doğdu. İlköğrenimine Varna'da başladı. Bulgaristan'da 1985 yılı itibariyle Türk azınlığa karşı yürütülen olumsuz politikalar sebebiyle, 1989 yılında Bulgaristan'dan Türkiye'ye göç etti. İlköğrenimini Orgeneral Kami Güzey İlkokulu'nda, orta öğrenimini Şair Behçet Kemal Çağlar İlköğretim Okulu'nda ve lise öğrenimini Arnavutköy Korkmaz Yiğit Lisesi'nde tamamladı. 1998 yılında Viyana Teknik Üniversitesi'nde başladığı üniversite eğitimini 2005 yılında Marmara Üniversitesi Almanca İşletme Enformatiği Bölümü'nde tamamladı. Aynı yıl Marmara Üniversitesi Sosyal Bilimler Enstitüsü İşletme Enformatiği anabilim dalında başladığı yüksek lisans öğrenimini 2007 yılında tamamladı. Almanya Reutlingen Üniversitesi Enformatik Fakültesi'nde tamamladığı ikinci yüksek lisans öğreniminin ardından halen Sakarya Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği anabilim dalında doktora öğrenimine devam etmektedir. 2006 yılından beri Marmara Üniversitesi Almanca İşletme Enformatiği Bölümü'nde araştırma görevlisi olarak çalışmaktadır.