RESTFUL WEB SERVİSLERİ ile E-SAĞLIK SİSTEMLERİ GERÇEKLEŞTİRİMİ

ALİ NİHAT ÇİÇEK

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

EYLÜL 2009
ANKARA

Fen Bilimleri Enstitü onayı:

_____

Prof. Dr. Ünver KAYNAK
Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Doç. Dr. Erdoğan DOĞDU
Anabilim Dalı Başkanı

Ali Nihat ÇİÇEK tarafından hazırlanan "RESTFUL WEB SERVİSLERİ ile E-SAĞLIK SİSTEMLERİ GERÇEKLEŞTİRİMİ" adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Doç. Dr. Erdoğan DOĞDU
Tez Danışmanı

Tez Jüri Üyeleri

Başkan: Doç. Dr. Erdoğan DOĞDU _____

Üye : Doç. Dr. Kemal BIÇAKÇI _____

Üye : Yrd. Doç. Dr. Murat ÖZBAYOĞLU _____

Üye : Yrd. Doç Dr. Bülent GÜMÜŞ _____

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Ali Nihat ÇİÇEK

Üniversitesi : **TOBB Ekonomi ve Teknoloji Üniversitesi**
Enstitüsü : **Fen Bilimleri**
Anabilim Dalı : **Bilgisayar Mühendisliği**
Tez Danışmanı : **Doç. Dr. Erdoğan DOĞDU**
Tez Türü ve Tarihi : **Yüksek Lisans – Eylül 2009**

Ali Nihat ÇİÇEK

RESTFUL WEB SERVİSLERİ ile E-SAĞLIK SİSTEMLERİ GERÇEKLEŞTİRİMİ

ÖZET

Günümüzde çoğu hastane bilgi sistemi, hastanelerin kendilerine özgü işleyişi ve talepleri doğrultusunda geliştirilen, hiçbir standarda dayanmayan bilgi sistemleri olarak çalışmaktadır. Şu sıralar yazılım firmaları bu sağlık bilgi sistemlerini Sağlık Seviye 7 (Health Level 7 – HL7) gibi standartlara uyumlu hale getirmeye çalışmaktadır. Fakat hastane otomasyonlarının baştan hastanelere özelleştirilmiş şekilde geliştirilmesi sonucu bu çevirme işlemi çok maliyetli olabilmektedir. Sağlık bilgi sistemlerinin hem sağlık hizmetlerinden faydalanacak kişiler hem de bu sektörde çalışanlar için daha kaliteli hizmet sağlaması ve bununla beraber sağlık masraflarının azaltılmasını başarmak için Elektronik Sağlık Kayıtları'nın (Electronic Health Records - EHR) farklı uygulamalar üzerinde paylaşımı ve transferi kaçınılmaz hale gelmiştir. Bu sayede er ya da geç elektronik sağlık kayıtlarının belirli bir standart kapsamında oluşturulması gerekmektedir.

Bu çalışmanın amacı standartlara uygun biçimde elektronik sağlık kaydı tutan web-tabanlı bir Hastane Bilgi Sistemi (HBS) geliştirmektir. Bu web-tabanlı bilgi sistemi sadece sağlık kuruluşunun kendi iş süreçlerine özel yapılandırılmış değil, aynı zamanda dinamik bir yapı ile hem yeniliklere anında adapte olabilmeli hem de ulusal ve uluslararası sağlık bilgi sistemleri ve diğer uygulamalarla da beraber çalışabilen bir yapıya sahip olmalıdır.

Bu çalışma kapsamında HL7 mesajlaşma ve içerik standardı kullanılacaktır. Sistem Web 2.0 standartları ve Service Tabanlı Mimari (Service Oriented Architecture - SOA) yaklaşımı kullanılarak geliştirilecek, metotlara ve verilere web servisleri ile erişilecektir. Kullanılacak web servisleri Temsili Durum Transferi (Representational State Transfer – REST, RESTful Web Services) yaklaşımı ile elektronik sağlık kayıtlarının sadece basit HTTP komutları ile erişimini sağlayacaktır. Çalışmanın sonunda basit bir hasta takip sistemi prototip olarak sunulacaktır.

Anahtar Kelimeler: HL7, ESK, RESTful, HBS

University : TOBB University of Economics and Technology
Institute : Institute of Natural and Applied Sciences
Science Programme : Computer Engineering
Supervisor : Associate Professor Dr. Erdoğan DOĞDU
Degree Awarded and Date : M.Sc. – September 2009

Ali Nihat ÇİÇEK

IMPLEMENTING E-HEALTH SYSTEMS USING RESTFUL WEB SERVICES

ABSTRACT

Currently, most of the hospital information systems (HIS) are custom developed information systems and recently they are being slowly converted into some standards' based software systems, such as HL7-complaint (Health Level 7) HIS. Sooner or later all systems need to convert their Electronic Health Records (EHR) into these standards due to the fact that systems need to share data and services to better serve the customer, workers, and all parties involved in healthcare, and also to lower the cost of medical care.

The purpose of this study is to develop a web-based HIS based on standard EHR. This web-based system should rely not only on the institution's specific business structure but also has a dynamic structure to adapt to new innovations and being interoperable with other electronic health systems such as national health or international health care applications, or other applications. By these improvements, the quality of health process lifecycle for healthcare workers and patients will be increased.

In the scope of this study, the messaging and content standard HL7 will be used. The system is developed using Web 2.0 standards and Service Oriented Architecture (SOA) approach where data and processes are accessed via web services. Web services in the system are developed using a RESTful approach where EHR items are accessed and manipulated via simple HTTP commands. At the end of the work, a demonstration of a patient follow-up management prototype implementation will be shown.

Key words: HL7, EHR, RESTful, HIS

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren hocam Doç. Dr. Erdoğan DOĞDU'ya, yine kıymetli tecrübelerinden faydalandığım TOBB Ekonomi ve Teknoloji Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyelerine teşekkürü bir borç bilirim.

İÇİNDEKİLER

ÇİZELGELERİN LİSTESİ

ŞEKİLLERİN LİSTESİ

KISALTMALAR

ABC	Address, Binding, Contract
AJAX	Asenkron JavaScript ve Xml
AKS	Adres Kayıt Sistemi
ANSI	American National Standards Institute
AR-GE	Araştırma ve Geliştirme
ASP	Active Server Pages
CDA	Clinical Document Architecture
CORBA	Common Object Request Broker Architecture
CRUD	Create, Retrieve/Read, Update, Delete
DCOM	Distributed Component Object Model (Paylaştırılmış Parçacıklı Nesne Modeli)
DMIM	Domain Message Information Model
DTD	Document Type Definition
EHR	Electronic Health Record
ERP	Enterprise Resource Planning (Kurumsal Kaynak Planlaması)
ESK	Elektronik Sağlık Kaydı
EUROSTAT	Avrupa Birliği İstatistik Ofisi
GIF	Graphics Interchange Format. Bir imaj dosyası uzantısı
GTC	Generic Typesetting System
HIPAA	Health Insurance Portability and Accountability Act
HL7	Health Level 7 (Sağlık Seviye 7)
HMD	Hierarchical Message Description
HTML	Hyper Text Markup Language. İnternet üzerinde web sayfası oluşturmak için kullanılan bir dildir.
HTTP	Hyper Text Transer Protocol. İnternette sunucular ve kullanıcılar arasındaki bilgilerin aktarılmasını sağlayan kuralları ve yöntemleri düzenleyen bir sistemdir.
IIS	Internet Information Services
ISO	International Standards Organization (Uluslar arası Standart Örgütü)
JPEG	Joint photographic experts Group. Bir imaj dosyası uzantısı.
JSON	Javascript Object Notation
JSP	Java Server Pages
Medula	Medikal Ulak. TC Sosyal Güvenlik Kurumu'nun sağlık kuruluşlarının faturalarını kontrol ettiği web tabanlı uygulama.
Mernis	Merkezi Nüfus İdaresi Sistemi
MKYS	Malzeme Kaynak Yönetim Sistemi
MSMQ	Microsoft Message Queuing
MSVS	Minimum Sağlık Veri Setleri
NHS	National Health Service (Ulusal Sağlık Servisi)
OECD	Organization for Economic Co-operation and Development (Ekonomik Kalkınma ve İşbirliği Örgütü bazen de İktisadi İşbirliği ve Gelişme Teşkilatı)
OWL	Web Ontology Language
PACS	Picture Archiving And Communication System

PHP	Personal Home Page.
RelaxNG	Regular Language for XML Next Generation
REST	Representational State Transfer
RIM	Reference Information Model
RMI	Remote Method Invocation.Uzaktan nesnelere ileti göndermeye yarayan teknolojidir.
RMIM	Refined Message Information Model
RPC	Remote Procedure Call
SDO	Standards Developing Organization
SGK	Sosyal Güvenlik Kurumu
SKRS	Sağlık Kodlama Referans Sunucusu
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol (Basit Nesne Ulaşım Protokolü)
SQL	Structured Query Language
TCP	Transmission Control Protocol
UDDI	Universal Discovery, Description and Integration
UML	Unified Modelling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USVS	Ulusal Sağlık Veri Sözlüğü
W3C	World Wide Web Consortium
WCF	Windows Communication Foundation
WEB	İnternet Ağı
WHO	World Health Organization (Dünya Sağlık Örgütü)
WSDL	Web Services Description Language
XSD	XML Schema Definition
XML	Extensible Markup Language (Genişletilebilir İşaretleme Dili)
Xmlns	XML Namespace

GİRİŞ

1.1 Giriş ve Çalışmanın Amacı

Bilgi teknolojilerinin sağlık sektörüne girişinden önce hastaneler ve diğer sağlık kuruluşları hasta arşiv bilgileri ve muayene kayıtlarını eski usul kalem, kağıt, daktilo vb. gerçek ortam materyalleri yardımı ile yaparlardı. Tabi bu yol ile sarf malzeme sarfiyatı maksimuma ulaşırken arşivleme için yüksek metrajlı alanlar kullanılıyordu.

Sabahları polikliniklerin önünde uzun uzadıya kuyruklar ve açılması beklenen sekreterlik bankoları, randevu veya muayene sırası alabilmek için birbirinin üzerine çıkan hastalar ve hasta yakınları, durumdan şikayetçi olan personel ve doktorlar bu sistemde uzun süredir çalışmaktadır ve kimse bu mevcut durumdan memnun olmamaktadır. Kısacası bu işin içinde rolü olan herkes perişan bir haldedir. Büyük devlet hastanelerinde bu resim daha da acılı bir hale gelmektedir. Çünkü küçük sağlık birimlerinde yapılamayan birçok tetkik veya ameliyat büyük hastanelerde yapılmakta olup ülkenin dört bir yanından büyük illerdeki büyük hastanelere hücum edilmektedir.

Bilgi teknolojilerinin yaygınlaşması sonucu kurumlar da işleyişlerini kısmen bir otomasyona bağlama yoluna geçtiler. Böylece hem kurum yönetimi karar destek sistemleri verilerine bakarak yöneticileri yönlendirecek, hem kayıtlar bilgisayar ortamında tutularak elektronik sağlık kayıtları oluşturulacak hem de süreç olarak işleyiş daha da kolaylaşacaktı. Verileri tutmak ve bu verilere bağlı olarak istatistik çalışması yapmak gerçekten eskiye göre daha da artış gösterdi. Böylelikle istatistik çalışmaları ve buna bağlı olarak kullanılan kaynakların neler olduğu daha rahat belirlenmektedir. Yönetimler de kararlarını bu veriler üstünden daha verimli şekilde verebilmektedir. Elbette karar destek sistemleri mükemmel denecek kadar gelişim göstermedi. Günümüzde kullanılan otomasyonların raporlamaları gerek veri bütünlüğünün tam sağlanamaması, gerekse işleyişin daha tam oturmamış olmasından dolayı %100 doğru bilgiyi sağlayamamaktadır.

Fakat bilgisayar otomasyonuna geçmek hasta, personel ve doktorlar açısından hayatı daha da kolaylaştırmadı. Hatta otomasyon sisteminin çeşitli sebeplerden dolayı (teknik veya teknik olmayan) durma ihtimalinde hastanenin de durması kaçınılmaz olmakta ve büyük sıkıntılar yaşanır hale gelmektedir. Tabii ki bu sıkıntılar yazılımın ve gerekli donanımın kalitesiyle doğru orantılı olduğundan çok iyi bir sistemin 7/24 ayakta kalabilmesi yazılımı sağlayan şirketin analizi doğrultusunda sağlanabilmektedir.

Yazılımın çökmediğini ve sistemin durma olasılığın hiç olmadığının varsayıldığı ideal bir koşulda bile işleyişin sıkıntısız olduğunu dile getirmek imkansız. Yazılım teknik olarak problemsiz çalışsa bile işleyiş mantığı olarak her zaman bir eksik çıkıyor. Sağlık sektöründeki iş akışının belirli bir standarda henüz oturtulmamış olması, her hastanenin kendi belirlediği kriterler üzerinden işe devam etmesi anlamına geliyor ve pratikte de böyle oluyor. Hasta kabulden muayene aşamasına, randevu, kabul, muayene, işlem-tetkik, yatış, sevk gibi süreçler her hastanede farklı şekilde işliyor. Bu işleyişlerini otomasyon yazılımlarına iyi oturtabilen bir sağlık kuruluşu bile yeri geliyor devletin mevzuat değişikliği yüzünden sıkıntıya girebiliyor. Yani sadece hastane işleyişinin standarda bağlı olması yetmiyor, bu standardı devlet, ülkemiz için Sağlık Bakanlığı, koymalıdır ve bütün hastaneler buna uygun hareket etmelidir. Şu an Sağlık Bakanlığı sadece tedavinin nasıl yapılması gerektiği, hangi işlemlerin yapılabileceğini ve bu işlemlerin kaç parya ve hastaların sigorta tiplerine göre belirli ödeme şartlarını standartlaştırmış durumdadır. Her sene veya senede birden fazla kez bu mevzuat değişebilmektedir. Ayrıca sağlık bakanlığı hastane otomasyonlarının hangi özellikleri bünyesinde barınması gerektiğini de belirlemiştir [1]. Ama ne yazık ki iş akışı olarak hiçbir standart bulunmamakta ve tekrar belirtmek gerekirse her kuruluş kendisine uygun gelen bir yol bulmaya çalışmakta ve kazandığı alışkanlıklardan daha kolayını bulsa dahi vazgeçmemek için direnmektedir. Bu farklılaşma gerek aynı otomasyon yazılımının yeni sürümüyle gerekse farklı bir firma ile anlaşarak yeni bir otomasyona geçiş ile mümkün olabilir.

Yazılımın oluşturulmasındaki problemler de yukarda bahsedilen olaylar sayesinde meydana çıkmaktadır. İhtiyaç analizi ilk seferde ne kadar iyi yapılırsa yapılsın, her

an değişebilmektedir. Sağlık yazılımlarında da yazılım süreci şuana kadar bitirilmemiştir. Yani bir paket program hazırlayıp büyük sağlık kuruluşlarına satış yapılsa dahi o paket program iş süreçleri için tamamen yeterli olmayacak ve sürekli yenilenmesi gerekecektir. Günümüzdeki sağlık sisteminde hizmet veren bilişim araçları sağlık kuruluşlarına göre özel olarak tasarlanmakta ve yürütülmektedir. Bu durum gerek sağlık denetim kuruluşlarını gerekse bu sağlık hizmetlerinden yararlanan vatandaşların her geçen gün farklı uygulamalarla karşılaşıp esas alması gereken sağlık hizmetlerinin yanında zamansal ve maddi kayıplara sebep olmaktadır.

Buna sebep olan en büyük etken de önceki kısımlarda bahsedildiği gibi hastane bilgi yönetim sistemlerinin gelişi güzel oluşturulmasıdır. Şuan asıl amacı bilgi toplayarak karar destek mekanizması oluşturmak olan bu sistemlerin standardizasyondan uzak olması nedeniyle veritabanları adeta veri çöplüğüne dönmüş vaziyettedir. Bu durum hem sağlık kuruluşlarının yönetilmesini zorlaştırmakta hem de hastalara verilen sağlık hizmetinin kalitesini düşürmektedir. Problemin en önemlisi de hastaların mağdur olmasında hala bir gelişme olmamasıdır.

İşleyişin standarda bağlanması hem yazılım firmalarının daha rahat ve nitelikli program yazmasına, hem de sağlık kuruluşlarının iş akış süreçlerinin gelişmesine ve verdikleri hizmetin daha kaliteli olmasına olanak sağlar.

Bu çalışmanın amacı bir işleyiş standardı oluşturarak ülkedeki tüm hastaneleri benzer şekilde çalışır hale getirmek ve bu sayede yazılım firmalarının uygulama geliştirirken sadece hastaneye bağlı kalmadan belirli bir standart çerçevesinde doğru işi yapabilmesini sağlamaktır. Teknolojik açıdan ise hem uygulama, kendisine katılan yeni özelliklere hemen adapte olabilecek hem de farklı uygulamalarla bütünleşmeyi kendi işleyişinde değişiklik yapmadan ya da minimum değişiklikle sağlayabilecektir. Bilgi sistemlerinin geldiği nokta itibari ile de bu uygulamanın web tabanlı olması artık kaçınılmaz bir gerçektir. Web tabanlı olmasının avantajı ileriki bölümlerde ayrıntılı biçimde açıklanacaktır.

E-SAĞLIK BİLEŞENLERİ

Bilgi teknolojilerinin tarihteki hızlı yükselişi ve internetin hayata girişiyle programlama teknikleri ve yaklaşımlar da değişti. Kullanıcıların uygulamalara sadece ofislerinde veya evlerinde değil, gittiği her yerde ihtiyaç duyması ve bununla beraber artık uygulama kurmanın ve güncellemenin bir dert haline gelmesi yazılımların web tabanlı olmasını cesaretlendirdi ve yaygınlaştırdı. Bu bölümde web tabanlı elektronik sağlık sistemi geliştirirken kullanılacak olan teknolojiler, standartlar ve yaklaşımlardan bahsedilecektir.

1.2 Web Servisleri

Web'in hayata girmesi sayesinde kuruluşlar kendi içlerinde, başka şirketlerle veya müşterileri ile aralarındaki etkileşimde kolaylık sağladılar. Fakat bu durum şirketlerin ihtiyaçlarını tam olarak karşılayamadı. Oluşturulan web siteleri sadece tanıtım amaçlı kullanılıyordu. Bu durum kullanıcılarla uygulamaların veya uygulamalarla uygulamaların birbirleriyle haberleşmesine olanak tanımıyordu. Bu sebeple işletmeler veya kişiler internette sınırlı bir çerçeve içindelerdi. XML'in gelişmesi ve web servislerinin doğuşu ile geniş bilgisayar ağları daha hareketli bir ortam haline geldi.

HTML'in ihtiyaçları tam anlamıyla karşılayamadığı sınırlı yapısı, yazılım dünyasında daha dinamik bir yapının ihtiyaç duyulmasını sağladı. XML, öğrenilmesi ve kullanılması çok kolay olan fakat bu kolaylığın karşısında içinde bulunduğu uygulamalara tamamen yeni bir standart kazandıran bir yapı taşıdır. HTML'in aksine dinamik bir biçimde yapılandırılmış verilerin transferini sağlar. Hiç bir platforma bağlı olmadığından verilerin istenildiği gibi yapılaşmasına imkan verir.

2000 yılından bu yana birçok alanda internet ve web teknolojileri yeni bir devrim başlatmıştır. Birçok araştırmacıya göre XML tabanlı web servisleri bu devrimin ikinci adımını oluşturmaktadır. Web servisleri bilgisayar ağları sayesinde yayınlanıp, istenince aranıp bulunarak erişilebilen ve bir uygulamanın kendisi veya en küçük

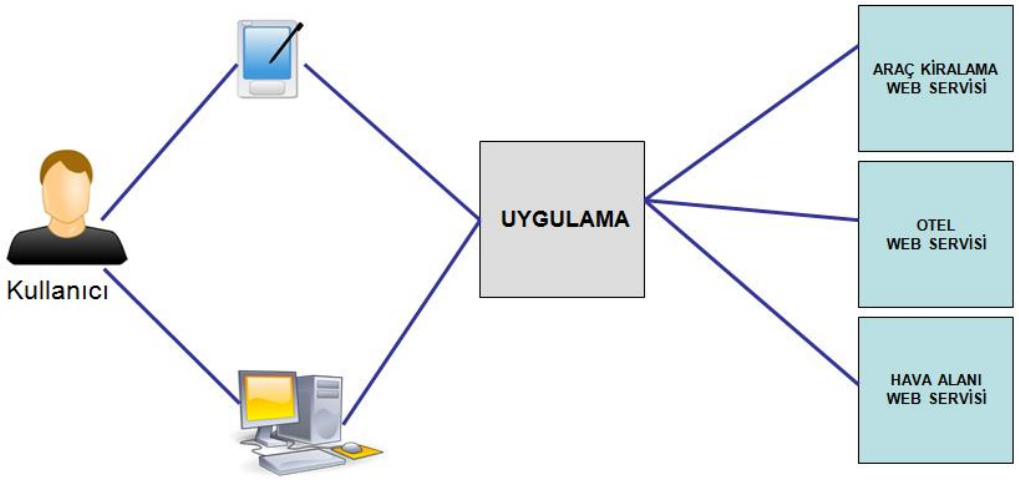
fonksiyonel parçasıdır. Bu servisler değişik kurumsal iş süreçlerini gerçekleştirip uygulamalar arası haberleşme sağlayabilmektedir.

IBM, Oracle, Microsoft, HP ve daha birçok firma web tabanlı uygulama geliştirme üzerine yoğun bir şekilde çalışmakta ve web servisleri yazılım-uygulama geliştirme araçlarını yine web tabanlı yazılım geliştiren firmalara ve programcılara sunmaktadır. Bu yoğun destek ve web platformunun geniş kitlelerce kullanılması neticesinde uygulamalar gün geçtikçe web servisi kullanımını arttırmıştır. Günümüzde web servisleri yazılım dünyasının vazgeçilmezleri arasında yer almaktadır. Internet teknolojilerinin bu denli gelişiminden sonra bugünkü web ortamı olmadan ağ tabanlı bilgi sistemlerinin düşünülmesi çok zordur.

Web ortamındaki gelişmeler üç safhada incelenebilir. Belge Web'i (Document Web) ile internetin ilk aşamadaki kullanımını, yani basit HTML sayfalarının HTTP protokolü üzerinden biçimlendirilmiş statik belgeler olarak kullanıcılara sunumunu ifade ediyor. Nitekim internetin kullanılmaya başlamasıyla kurumlar ve kuruluşlar basit HTML sayfaları ile kendilerini tanıtan web sayfaları ile geniş kitlelere açılıyorlardı. Daha sonraki aşamada ise Uygulama Web'i (Application Web) devreye girdi. Bu yapı işletmelerin veya her hangi birinin müşterilerine veya ziyaretçilerine web üzerinden bazı iş süreçlerini yaptırma gereksinimi sonucunda ortaya çıktı [45]. Bu süreçte sunucu tarafında çalışan programlar vasıtasıyla hazırlanan dinamik HTML belgeleri ile kullanıcı ve iş uygulaması arasında bağlantı sağlandı. Bu dinamik HTML sayfaları değişik teknolojilerle gerçekleştirilebilir. Bunlara örnek olarak Microsoft'un ASP, Java'nın JSP ve PHP gösterilebilir. Bu diller sayesinde kod geliştiriciler basit HTML sayfaları üzerine script ekleyerek verileri sunucu tarafına gönderebilmeye ve bu verilerin sunucu tarafında işlenerek tekrar kullanıcı karşısına getirilmesini sağlayan web uygulamaları yazmaya başladır.

Son nesil olan Servis Web'i ile de kuruluşların diğer işletmelerle veya şahıslarla olan iş süreçlerini bütünleştirme gereksinimi karşılanmış oldu. Bu yapının temel taşı web servisleridir. Web servislerindeki temel amaç bir bilgi sistemindeki program parçalarının etkileşimini sağlamaktır. Örnek olarak Şekil 2.1'de farklı web

servislerini kullanarak bir iş gezisi planlaması ve rezervasyon işlemlerini tek bir arayüz üzerinden yapan bir uygulama senaryosu gösterilmiştir. Senaryoda kullanıcı bir iş gezisi planı yapmaktadır ve uygulama yardımı ile gideceği yere ulaşabilmesi için hava yolu rezervasyonunu ve havaalanına inişinden sonra da kiralayacağı aracı ve kalacağı oteli ayarlayabilmektedir. Kullanıcı tek bir web sayfasına girerek havayolu şirketinin, araç kiralama şirketinin ve otelin web servislerinden yaralanmaktadır.



Şekil 2.1- Web Servis Uygulaması

Microsoft'tan DCOM, Sun'dan RMI veya Object Managament Group'tan CORBA ve benzeri teknolojilerden farklılık gösteren web servisleri belirli nesne modellerine özgü protokolleri kullanmaz. Web servisleri iletişimde, standart Web protokollerini ve veri formatlarını kullanır. Örneğin Hypertext Transfer Protocol (HTTP), Extensible Markup Language (XML) ve Simple Object Access Protocol (SOAP) gibi. Bu Web standartlarını destekleyen sistemlerin tümü Web Servislerini destekler. [2][3][4]

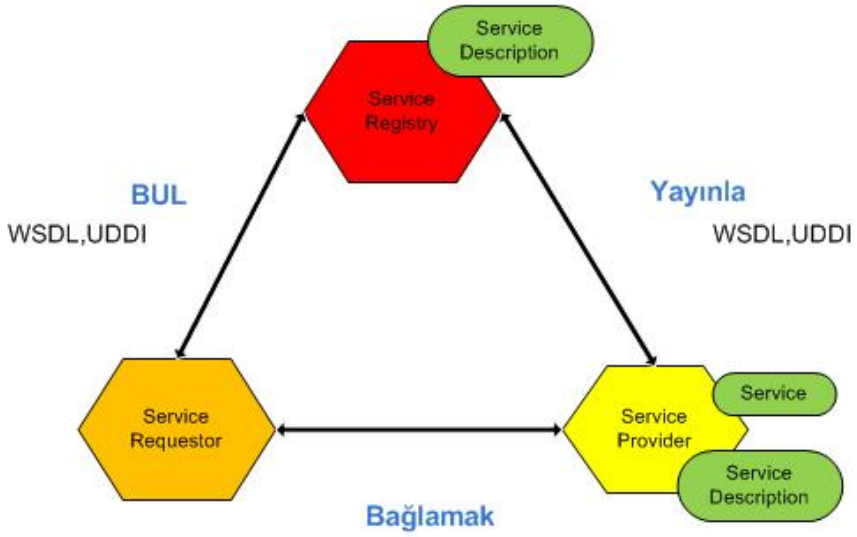
Web servislerinin kullanımı için üç ana bileşenin birbiriyle bağlantı kurması gerekir; servis sağlayıcı, servis istemcisi ve servis kayıt birimi. Bu bileşenler günümüzde açık internet standartlarını oluşturtan standartlardandır. Çıktığı günden beri gelişmekte

olan bu modelle ilgili olarak kullanılan çekirdek standartlar SOAP, WSDL ve UDDI'dır.

Servis sağlayıcı istemcilerin sunucuda bulunan servislerle haberleşmesini sağlar. Servis sağlayıcı kendi bünyesinde bulunan web servis tanımlarını servis kayıt birimine kaydederek bu servislerin nasıl çağırılacağını belirtir.

Servis sağlayıcı tarafında bulunan web servisleri çağırıp kullanan istemci uygulamalara da servis istemcileri denir. Web servislerin çağırılması sırasında gönderilecek parametreleri servis kayıt biriminden arar, bulur ve çağırır. Bu çalışma işleyişi ekran başındaki kullanıcı tarafından çalıştırılabileceği gibi herhangi bir tetikleme olayı ile de istem gerçekleştirilebilir.

Servis sağlayıcılarının yayınladıkları web servis tanımlarının saklandığı ve bu servislerin aranınca bulunmasını sağlayan birim ise servis kayıt birimidir. Servis sağlayıcılar servis kayıt birimlerini tarayarak ulaşmak istediği servisler hakkında bilgi alabilir. Servis kayıt birimi her servisin nasıl çağırılacağı konusunda da açıklama bilgileri barındırır [45]. Şekil 2.2'de web servisi çalışma prensibi ve akışı incelenebilir.



Şekil 2.2 - Web Servis Çalışma Prensibi [45]

Yukarıdaki şekilde bir web servisi istemcisi (SOAP Client) servis kayıt biriminden (UDDI) web servisini bulur. İstemci istek yapabilmek için bir SOAP mesajı hazırlar. Bu mesaj XML tabanlı bir dokümanıdır. İstemci SOAP mesajını alır ve içindeki bilgileri web veya uygulama sunucusunda çalışan SOAP istek dinleyicisine iletir. İstek dinleyici gelen isteklere cevap veren sunucu taraflı programlardır.

Web servis sunucusu gelen SOAP mesajını değerlendirir ve gerekli parametreleri göndererek istenen servisteki metotları çağırır. Çağırılan metotlar çalışır ve yine sonuçları bir SOAP mesajı ile istemciye geri gönderilir. Son olarak da istemci, yaptığı isteğin cevabını (request - response) alır ve değerlendirir.

SOAP, XML tabanlı mesajların transferi için kullanılan ve genellikle HTTP/HTTPS protokollerini üzerinde çalışan bir haberleşme protokoldür**.** FTP ve SMTP gidi diğer iletişim protokolleri üzerinden de kullanılabilir. XML tabanlı olduğu için dil ve platform bağımsız ve bu sayede esnek bir yapıya sahiptir [5].

SOAP'ın değişik örnekleri olmasına karşın en çok kullanılanı RPC modelidir ki bu modelde istemci-sunucu mantığı ile çalışır. Bu protokolle istemci uygulama kullanacağı web servisi belirtir ve gerekli parametreleri paketler. Bu paketlere SOAP mesajı adı verilir. İstemci kullanmak istediği uygulamaya bir istek paketi gönderir ve karşılığında sunucu hemen bir cevap paketi karşılığı verir. Bu mesajları üç ana yapı ile bir XML dokumanı oluşturur, Zarf/başlık/gövde (Envelope/Header/Body).

Bir SOAP zarfı içerisinde opsiyonel olarak SOAP başlığı ve gövdesi barındırabilir. Ama SOAP yapısı olduğu anlaşılması için XML Namespace'inde (`xmlns:soap`) mutlaka SOAP yazmalı ve XML dokümanında bir zarf bulunmalıdır.

Temel SOAP iskeleti aşağıdaki gibidir (Çizelge 2.1);

Çizelge 2.1 - SOAP Mesaj Yapısı

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Header>
    ...
  </soap:Header>
  <soap:Body>
    ...
    ...
      <soap:Fault>
       ...
      </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

Zorunlu olan zarf kısmı XML dokümanının içerisindeki <soap:Envelope> etiketi ile tanımlanmıştır. Bu etiket olmadan SAOP XML'lerinin SOAP anlamı kalkmış olur yani bir XML dokümanının SOAP mesajı olduğu XML Namespace'te "soap" yazmasının yanı sıra bu dokümandaki zarf (envelope) etiketiyle belirlenir.

Header/Başlık <soap:Header> etiketi ile ise isteğe bağlıdır ve SOAP mesajının uygulamalarla ilişkili spesifik bilgilerini belirtir. Eğer bir başlık etiketi olacaksa zarfın ilk alt elamanı yani ilk çocuğu olmalıdır.

Esas olan SOAP mesaj içeriği, XML dokümanının gövde (body) kısmında taşınır. Gövde etiketi de yine zarf elemanının çocuğudur. Gövdenin içerisinde opsiyonel hata (fault) çocuğu da yer alabilir ki bu elemanla hata mesajları taşınır.

Çizelge 2.2 - SOAP İsteği (SOAP Request)

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body>
    <m:GetPatient xmlns:m="http://www.example.com/patients">
      <m:TCKimlikNo>12345678910</m:TCKimlikNo>
    </m:GetPatient>
  </soap:Body>
</soap:Envelope>
```

Çizelge 2.3 - SOAP Cevabı (SOAP Response)

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body>
    <m:GetPatientResponse
xmlns:m="http://www.example.com/patients">
      <m:Name>Ali</m:Name>
    </m:GetPatientResponse>
  </soap:Body>
</soap:Envelope>
```

Yukarıdaki örneklerde bir SOAP isteği ve karşılığında gelen cevap mesajı vardır. Görüldüğü üzere bir zarf yaratılmış ve içinde de sadece gövde elemanı gömülmüştür. Mesajın içeriği de gövdenin içinde belirtilmiştir. Örnekte bir hasta sorgulama servisi yazılmıştır. İstek olarak GetPatient metoduna TC Kimlik No parametresi gönderiliyor (Çizelge 2.2, Çizelge 2.3.)

Dönen cevap mesajında da aynı şekilde GetPatientResponse nesnesi dönüş yapıyor ve içerisinde Name değişkeni, değeri ise Ali. GetPatient sonuna otomatik olarak response takısı eklenmiştir. Bu metot isimleri ve parametreler SOAP namespace'i ile ilişkili olmayıp uygulama tanımlı yapılardır [5].

WSDL ise bir web servisinin hangi işlemleri yaptığını, nasıl çağırılacağını belirten ve yine XML tabanlı bir dildir. Bir web servisiyle ilgili işlevler, kullanılan veri yapıları

gibi tüm bilgileri XML formatında saklayan ve web servisini bütünüyle tanımlayan bir belgedir. Web servisleri WSDL belgeleri olmadan da çalışabilir ama bu koşulda istemcilerin her birinin web servis içeriğini ve yapısını kendi bünyelerinde biliyor olmaları gerekir. Bu da zaten web servis mantığına aykırı bir yaklaşım olur.

Bir WSDL dokümanı üç ana bileşenden oluşur:
Tip (Type): Web servisi tarafından kullanılacak veri yapıları tanımını saklar. İletişimde kullanılacak veri yapıları böyle tanımlanır. Eğer sadece basit (primitive) tipleri (string, int, float vb.) kullanılacaksa bu etikete ihtiyaç yok denebilir. Ama bir veri yapısı (structure) hasta(TcKimlikNo, Isim, Soyisim) gibi bir veri yapısı döndürecek veya parametre olarak alınacaksa bu bir tip (type) olarak belirtilmelidir. Web servis tanımları hazırlanırken basit tiplerle birlikte karmaşık veri yapıları ve nesneler de kullanılabilir.

Mesaj (Message): Metotların girdi ya da çıktılarından her biri mesaj olarak değerlendirilir. Örneğin hastaBilgisiBul(TCNo) metodu için, sadece kimlik no içeren bir girdi mesajı ve hasta tipinden oluşan bir çıktı mesajı tanımlanması gerekir. Daha sonra servisler kullanılırken bu mesajlar kullanılır.

Port Tipi (PortType): Web servislerin sağlamış olduğu metotlar/operasyonlar bu elemanda tanımlanır (operation). Her bir metodun girdi ve çıktı için hangi mesaj tiplerini kullanacağı bilgisi de burada belirtilir. Port tiplerine metotları bir arada barındıran bir sınıf ya da kütüphane gözüyle bakılabilir.

Port tiplerindeki operasyonlar dört değişik biçimde sıralanır;

One-way: Operasyon bir mesaj alabilir ama herhangi bir cevap dönmez.
Request-Response: Bir istek mesajı alınır ve karşılığında bir cevap mesajı döner.
Solicit-Response: Bir istek mesajı gider ve cevap mesajının gelmesi beklenir.
Notification: Bir mesaj yollanır ve hiçbir cevap beklenmez (Çizelge 2.4).

Çizelge 2.4 - One Way

```
<message name="newPatient">
  <part name="Name" type="xs:string"/>
  <part name="Surname" type="xs:string"/>
</message>
<portType name="patientOperations">
  <operation name="setName">
    <input name="newPatientValues" message="newPatient"/>
  </operation>
</portType >
```

One-way örneğinde bir setName metodu ve input olarak yeni hasta bilgisi alıyor, yeni ekleme işlemini yapıyor ve bir cevap dönmüyor (Çizelge 2.4)

Çizelge 2.5 - Request – Response

```
<message name="patientInput">
  <part name="tcKimlikNo" type="xs:string"/>
</message>
<message name="patientOutput">
  <part name="name" type="xs:string"/>
</message>
<portType name="patientOperations">
  <operation name="getPatient">
    <input message="patientInput"/>
    <output message="patientOutpur"/>
  </operation>
</portType>
```

Request-response örneğinde ise getPatient metodu hangi hasta bilgisini getireceğini aldığı TC Kimlik No girdisiyle beraber işlem yapıp daha sonra bulunan hasta bilgisini geri döndürüyor (Çizelge 2.5) [6].

UDDI platformdan bağımsız, web servislerini tanıtmaya, işletmeleri bulmaya ve işletmelerin yayınladıkları servisleri entegre etmeye yarayan bir çerçevedir (framework). Web servislerle ilgili bilgilerin saklanması için kullanılır ve bu bilgiler WSDL dili ile tanımlanır. İletişimi SOAP mesajları ile sağlanır [7].

Eğer herhangi bir sektör bir UDDI hizmeti verecekse, işletmeler bu hizmete kayıt olurlar ve kendi servislerini sunarlar. Böylece o sektörle ilgili iş yapanlar aradıkları

hizmeti kolayca bulabilirler. Basit bir örnekle anlatmak gerekirse bir uçuş hizmeti veren şirketler kendi servislerini UDDI'a kaydettirerek tur firmalarının gerektiğinde rezervasyon ve satış veya firmaların diğer hizmetlerinde faydalanabilirler.

Internet uygulamalarının gelişmesiyle ortaya atılan Web 2.0 kavramıyla da web siteleri sadece düz HTML kodlarıyla kullanıcılara sadece kaynak göstermeyi bıraktığı yukarıda vurgulanmıştır. İkinci nesil internet uygulamaları sadece kullanıcılara belirli kaynakları göstermek yerine, kullanıcın daha çok rol aldığı interaktif bir platform sunmaktadır. Hatta öyle ki kullanıcıların Windows tabanlı uygulamalarda yapabildiği her şey artık web ortamına sunulmuştur. İnternetin sınır tanımayan yapısının da yardımıyla hem haberleşme hem de her türlü uygulama desteği kişisel bilgisayarlarda kullanılan programlardan web ortamına taşınmıştır. İnteraktif uygulamalara örnek olarak, Facebook, Flickr veya birçok blog sitesi gösterilebilir. Bunun yanında tabii sayması güç çok çeşitli web uygulamaları mevcuttur. Internet bankacılığı, elektronik posta ve doküman yönetimleri, e-devlet ve e-sağlık uygulamalarıyla neredeyse tüm hayat internet üzerine dökülmüş oldu. Tabi böylelikle farklı uygulamaların birlikte çalışabilirliği açısından birbirlerine servis sağlamaları da kaçınılmaz halde geldi. İşte bu yüzden web servislerinin günümüzde en çok kullanılan web tabanlı teknoloji olduğu tartışılmaz bir gerçektir. Gelecekte de web tabanlı uygulamalar hayatın vazgeçilmez parçası olacaktır [9].

Ülkemizde de iyi bir hastane otomasyonunun sahip olması gereken özellikler itibariyle birçok web servis kullanması gerekmektedir. Bunlara örnek olarak Nüfus ve Vatandaşlık İşleri Genel Müdürlüğü'nün Mernis ve AKS web servisleri, Sağlık Bakanlığı'nın Sağlık-NET web servisleri ya da Sosyal Güvenlik Kurumu'nun Medula web servisleri verilebilir. Tabii iyi bir web uygulamasının bu servisler dışında da farklı uygulamalarla birlikte çalışabilmesi talep edildiğinde hemen entegre olabilmesi iyi bir özellik olacaktır. Bu çalışma da bu noktaya üzerinde de durulmaktadır.

1.3 Service Oriented Architecture (SOA)

Yine son yıllarda adından çokça bahsettiren bir yaklaşım, servis tabanlı mimari (service oriented architecture - SOA). Aslında düşünülen teknolojiden ziyade bir sıyrılma ve iş geliştirmede üstün performans, hız ve başarı istatistiklerini yükseltmekti. Küreselleşme ve beraberinde gelen büyük firmaların rekabeti ekonomilerde dalgalanma yaparak iş dünyasının daha zor bir yer hale gelemsini sağladı. İş süreçlerini daha verimli hale getirmek ve kayıpları minimize etmek isteyen yöneticiler çareyi bilişim teknolojilerinde aradılar ve kurumsal kaynak planlaması (ERP) burada devreye girdi. ERP ilk zamanlar sadece kaynakları yönetmek gibi görünse de sonunda direk olarak bir kurumun yaptığı tüm işleri yansıtan bir kavram olmaktan başka bir anlama gelmediğini ispatladı.

Tabii değişen dünya ve değişen iş şartları ve bilişim teknolojilerinin getirdiği yenilikler parametre olarak kullanılınca ERP'ler sürekli olarak değişim göstermek durumunda kaldı. Kurumlar birbirleriyle rekabetini sadece pazarda sergiledikleriyle değil aynı zamanda kullandıkları ERP yazılımlarıyla da göstermektedir. ERP yazılımlarını güncelleyemeyen birçok kuruluş mevcut ve bu kuruluşların başarı oranları da bu yüzden gelişim gösterememektedir. Şirketlerin sahip olabileceği birçok veriyi iyi şekilde yönetememesi ve verimli bilgilere çevirememesi demek fırsatları kaçırması hatta neyi yöneteceğini bilmemesi anlamına gelmektedir [12].

Belli bir zaman geçtikten sonra bu uygulamaları güncellemekte hem yazılım hizmetini verenler hem de yazılımı kullanan ya da kullandıran yöneticiler/analistler için de zor olmaya başladı. Yazılım dünyasının getirdiği fonksiyon üzerine fonksiyonlar, modül üzerine modüller ve bu modüllerin birbirleriyle olan sıkı bağları (coupling) ERP uygulamalarının yeniden yapılanmalarını gittikçe zorlaştırdı. Bu nedenle AR-GE çalışması yapanlar çözümü yeni bir yaklaşımda aradılar ve sonucunda servis tabanlı mimari kavramı gündeme geldi.

Kitaplarda, seminerlerde veya internette pek çok tanımı bulunan SOA basit olarak iş süreçlerinin yani servislerin birbirinden bağımsız halde birleştirilerek istenilen

durumlarda sınırsız sayıda tekrar tekrar kullanılabilmesine, gerektiğinde birbirleriyle yer değiştirerek veya bazılarını kullanımdan kaldırıp yeni servisler ekleyerek yazılımların dinamik olarak kullanılabilmesini sağlayan bir yaklaşımdır.

SOA'dan bahsederken elbette servis kavramı üzerinde de durulmasında fayda var. Servis denilen aslında birden fazla tekrarlanan iş adımıdır. SOA çıkmadan önce uygulamalarda karmaşık bir şekilde bu iş adımları gerektiği yerlerde çağırılıp lazım olduğunda başka yerlerden de çağrılıyordu. Böylece uygulamaların birbiriyle olan bağı artıyor ve karmaşıklığı artıyordu. SOA'da bu iş adımları parçalara ayrılıyor ve servis yani verilen küçük hizmet olarak adlandırılıyor. Bu servisler servis tabanlı mimari çatısı altında yeni iş süreçleri tanımlıyor ve bu bağımsız servisler sayesinde istendiği zaman istenildiği yerden çağırılarak büyük bir modülerlik sağlıyor [10][11][13].

Yazılım dünyasından örnek vermek gerekirse bir bankacılık uygulamasında "müşteriye hesap açma", "döviz fiyatları güncelle" gibi sık kullanılan küçük iş parçaları servis olarak nitelendirilir. Elektronik sağlık sistemlerinde ise "hasta kaydı oluştur", "hasta hesap dökümü" veya "hasta işlemlerini listele" yine birer servistir.

Bu servisler birbirleriyle iç süreçte yani tek bir uygulamada haberleşebildiği gibi harici uygulamalarla da bağlantı kurabilir yapıda olmalıdır ki SOA yapısına uygun olsun. Örneğin bir hastane otomasyonu bir hastası için özel bir diyaliz merkezinden randevu alabiliyor ve özel diyaliz merkezinin bu servisini herhangi bir hastane, herhangi bir platformdan bağımsız şekilde dilediği zaman kullanabiliyorsa bu servis tabanlı mimarinin örneğidir.

Özetle SOA;

- Bir programlama dili değil, bir uygulama geliştirme mantığıdır.
- Tamamen platformdan bağımsız, esnek uygulamalar geliştirmeyi hedefler.
- İş süreçlerini parçalara bölerek modüler bir yapı oluşturur. Bu yapıların tamamı bir uygulama oluşturabileceği gibi başka uygulamaların parçaları da

istenildiğinde çağırılıp kullanılabilir. Uygulamaya yeni servisler kolaylıkla eklenip çıkarılabilir. Yapboz (puzzle) parçalarının aksine lego parçaları gibi servisleri ekleyip çıkarmak kolaydır. Servisler birbirleriyle sıkı bağlı değildir (loosely coupled). Bu kolaylık teknolojide veya iş süreçlerinde değişiklik olduğu zaman kurumların bu değişikliklerden geri kalmasını engeller.

- Sadece tek bir uygulama içi değil birden fazla uygulamanın haberleşmesi sağlanabilir. Servisler birbirlerine veri alıp göndererek haberleşir. Uygulamaların hatta çoklu platformların entegrasyonu servislerin veri transferleri ile gerçekleşir.
- Genelde web servisleri kullanıldığından web servislerin tanımı da WSDL dokümanlarında bulunur. Veri alışverişi SOAP mesajlarıyla yapılır.
- UDDI yardımıyla bir indeks oluşturularak var olan WSDL'leri bulmak ve kullanmak kolaylaşır.

1.4 Elektronik Sağlık Kaydı (Electronic Health Record - EHR)

1960'larda var olmaya başlayan Elektronik Sağlık Kaydı kavramı 1990'larda yaygın biçimde kullanılmaya başladı. Elektronik sağlık kayıtlarının anlamı Health Information Management Systems Society's (HIMSS) sitesinde yayınlamış olduğu açıklayıcı tanımdan anlaşılabilir; "Elektronik Sağlık Kaydı, insanların uzun süreli yani doğumundan ölümüne kadar bütün sağlık durumlarının ve aldığı bakımlar ile ilgili kayıtların elektronik ortamda tutulmasına denir. Elektronik sağlık kaydı bilgileri dahilinde; hastanın nüfus bilgileri kapsamında ilerleme notları, sorunları, ilaçları, hayati belirtileri, tıbbi geçmişi, bağışıklıkları, laboratuar verileri ve radyoloji raporları vardır. Medikal görüntülerin (x-ray, mr gibi) entegre edilmesi, elektronik sağlık kaydına bütünlük kazandırır ve eğer bir sağlık problemi varsa tam bir tanı konmasını sağlar" [16].

Elektronik Hasta Kaydı ise hastanın bir sağlık kuruluşuna kayıt yaptırdığı aşamadan itibaren bir eşsiz bir referans numarası ile hastanın o kuruluştaki tüm bilgilerini kapsayan çerçevedir. Elektronik sağlık kaydının farkı, hastanın tıbbi geçmişi her

nerede eklenmiş ya da değiştirilmiş olursa olsun o hastanın bütün tıbbi bilgilerini içeren bir formdur.

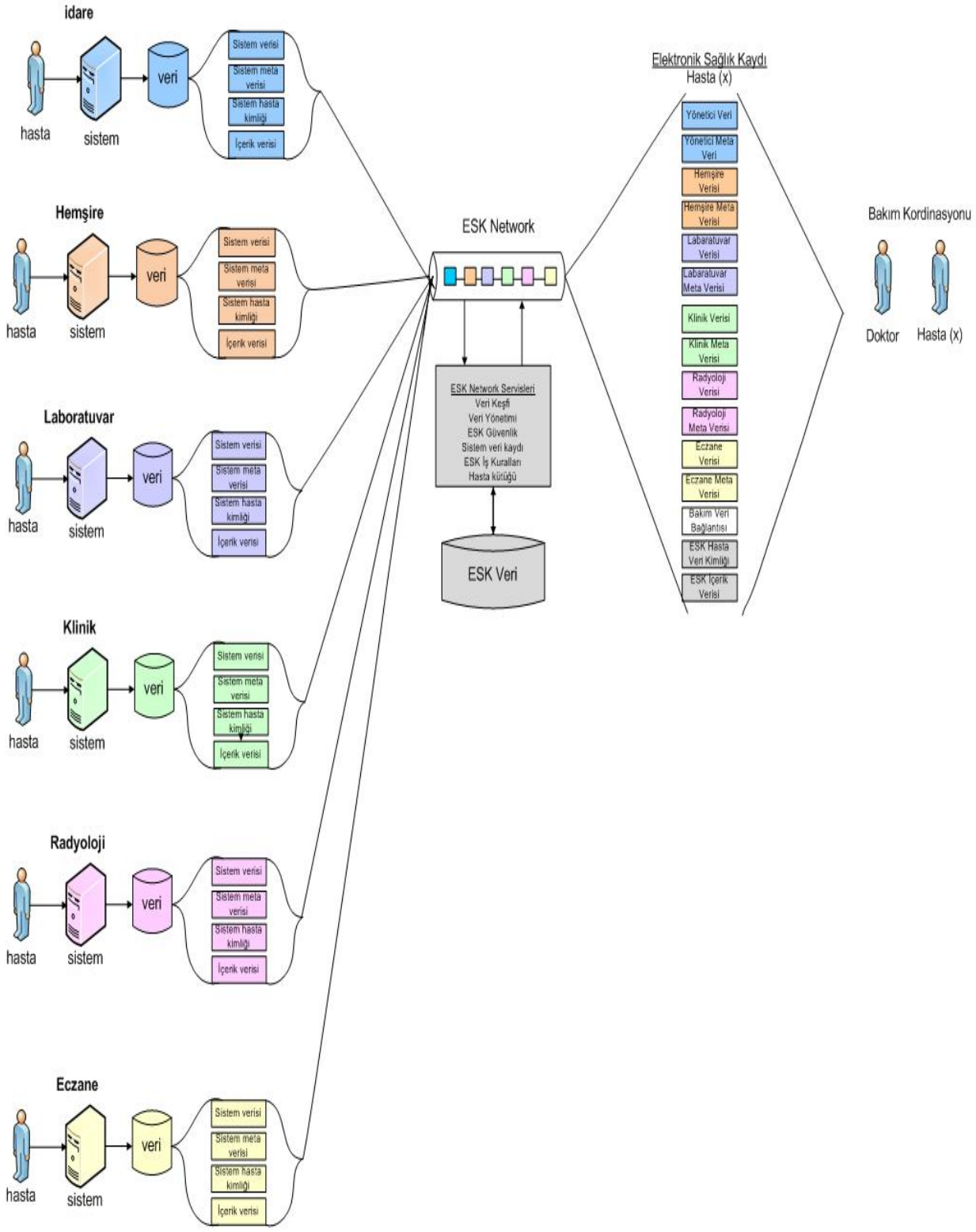
Elektronik Sağlık Kaydı (ESK) doktorlar için bilgi kaynağıdır. Güvenlidir, gerçek zamanlıdır, hasta merkezlidir. ESK her gerektiği yerde ve zamanda kayıta dayalı olarak karar vermeye yardım sağlar. Klinisyenlerin iş akışını sıraya dizer, iletişi sağlar, gecikme veya boşlukları uyarır. Direk klinik bakımla ilgili olmayan kullanımlar için veri toplanmasına yardım eder. Örneğin; faturalama, kalite yönetimi, sonuçların raporlanması, kaynak planlaması, hastaların izlenmesi ve raporlanması, uzman doktorların veya asistanların performans puanları bilgileri vb.

Elektronik sağlık kayıtlarıyla, tıbbi hatalar ve bu hatalara dayalı sonuçlar azaltılır, sağlık sektöründeki maliyet düşürülür. Ayrıca ESK; ulusal ve uluslararası sağlık hizmeti veren tüm kuruluşların sağlık ve hasta kayıtlarına ulaşabilmesini sağlayan bir sistem altyapısıdır. Kurulan sistemlerin belirli bir standarda uygun olması, farklı uygulamaların birlikte çalışabilmesini sağlamaktadır. Bu tez çalışmasının da en büyük amaçlarından biri birlikte çalışabilir sağlık uygulamalarının bir hastanın elektronik sağlık kaydını paylaşmasıdır.

Elektronik Sağlık Kaydı, bir sağlık uygulamasının hastaya ait bilgileri girebileceği her bir modülden girilen verilerle oluşur. Hastanın kayıt aşamasının yapıldığı kayıt kabul gişelerinden, doktor muayenesi ve müdahalesi yapılan polikliniklere, eğer hastanın yatış işlemi gerektiren bir tedavi süreci varsa yatan hasta servislerine, işlem ve tetkikleri için laboratuarlara kadar sağlık kurumunun hastayla ilgili her bir biriminde girilen veriler olabilir.

Çalışmanın örnek sağlık uygulaması geliştirme bölümünde hastane işleyiş bütünlüğünü sağlayan ve elektronik sağlık kaydı oluşturan modüllerin ayrıntıları incelenecektir. Kısaca Elektronik Sağlık Kaydı Bileşenleri aşağıdaki Şekil 2.3'de gösterilmiştir.

2005 yılında İngiltere'de gerçekleştirilen "National Health Service" (NHS) elektronik sağlık kayıt sistemi uygulamalarına örnek olabilir. NHS'in amacı 60 milyon hastanın elektronik sağlık kaydını 2010 yılına kadar bünyesinde bulundurabilmekti. İngiltere, Galler ve İskoçya bazında yapılan bu uygulama başarıya ulaşmış bir çalışma olarak değerlendirilmektedir [18].



Şekil 2.3 - Elektronik Sağlık Kaydı Bileşenleri [17]

1.5 Health Level 7 (HL7)

1.5.1 HL7 Nedir

HL7 (Health Level Seven), ANSI (American National Standards Institute) tarafından kabul edilmiş, elektronik sağlık sistemleri ve sağlık bilişimi alanında standartlar geliştiren kuruluşlardan biridir. Bu tip kuruluşlara SDO (Standards Developing Organization) denir.

Sağlık sektöründe standart geliştiren birçok kuruluşun tersine HL7 tamamen klinik ve yönetsel veriler ve bu verilerin transferleri üzerine yoğunlaşmıştır. Diğer organizasyonlar tıbbi cihazlar, eczane ya da depo, dijital görüntüleme veya sigorta (hak sahipliği/hizmet karşılığı alabilme) işlemleri gibi belirli sağlık uygulama alanları için zaman zaman spesifikasyon veya protokol olarak da adlandırılan standartlar üretir.

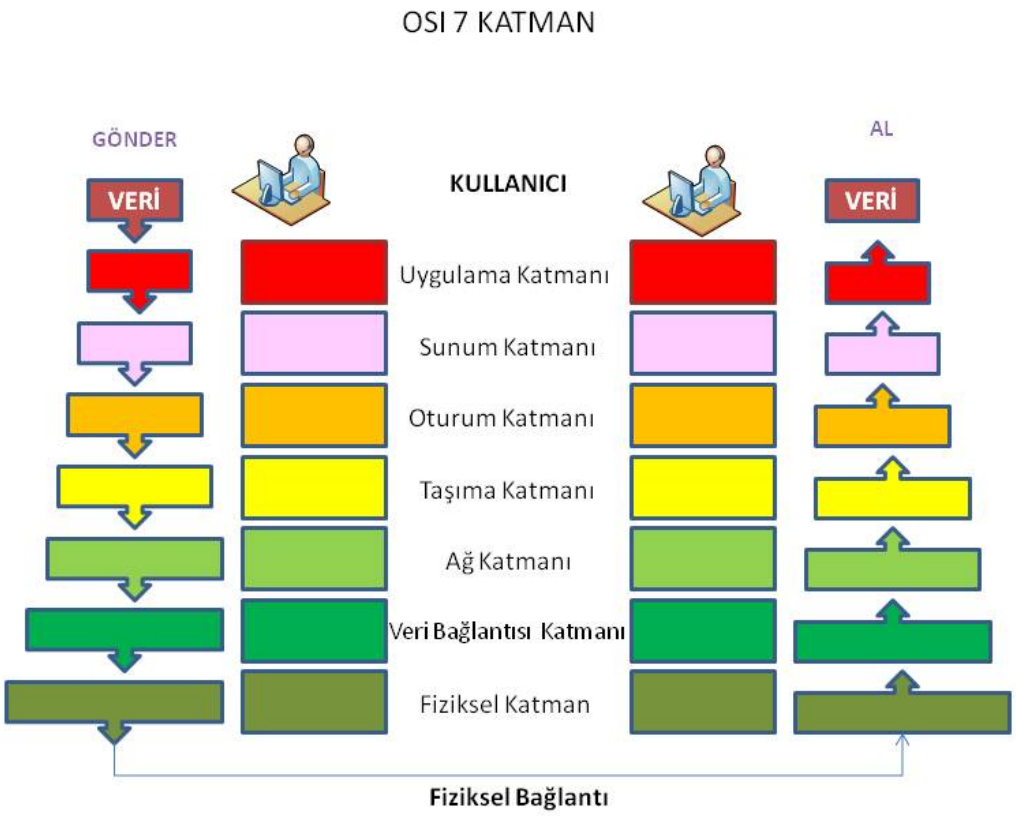
Merkezi Ann Arbor, Michigan, ABD olan HL7, diğer SDO'lar gibi kar amaçlı çalışmayan gönüllü bir organizasyondur. 1987'nin Mart ayında kurulmuştur ve geliştirdiği standartla aynı ismi taşımaktadır. Amaç olarak sağlık uygulamalarının değişik platformlarda değişik yapılarda olmasına rağmen aralarındaki bağlantı ara yüzünü oluşturmak için standart belirmeyi hedef olarak almıştır [21]. HL7 standartları HL7 üyeleri tarafından yaratılır ve geliştirilir. Bu üyeler sağlık hizmeti sunan veya sağlık hizmetlerinin bedelini ödeyen kurum/kuruluşlar, tıbbi cihaz, sistem, çözüm üreten firmalar, danışmanlar, devlet otoriteleri olabildiği gibi sağlık alanında çalışan, klinik ve yönetsel çalışmalar yapan bireyler de olabilir. Yeni eklenecek ya da üzerinde değişiklik yapılacak maddelerle sektördeki gelişmeleri tartışmak için bu üyeler dünyanın çeşitli yerlerinde toplantılar düzenler. Çok sık yaşanan yanlış anlamaların aksine HL7 kesinlikle yazılım geliştirmez. HL7 organizasyonu sağlık bilişiminde görev alan uzmanlarının ve mühendislerinin elektronik ortamdaki sağlık bilgilerinin mesajlar halinde transferi, yönetilmesi ve sağlık otomasyonları ile bütünleşmesini sağlayan standartlar oluşturmak üzere çalışan uluslararası bir topluluktur. ANSI tarafından akredite edilmiş diğer tüm

SDO'lar gibi HL7 de, oy bütünlüğü, açıklık ve yarar dengesi sağlamakla beraber titiz ve iyi tanımlanmış faaliyet planları düzenler. Bu planlamalar sonucunda çok detaylı tanımlamalar ortaya çıkar. Bu tanımlamalar standartlaşarak birbirinden bağımsız sağlık kuruluşları arasında çok önemli klinik ve yönetsel veri setlerinin karşılıklı transferini sağlar. Bu da HL7 mesajlaşma standardı olarak tanımlanır [19].

HL7'ın organizasyon yapısı teknik komiteler ve özel ilgi gruplarına ayrılan çalışma grupları olarak ayrılmıştır. Teknik komiteler standartların oluşturulması ve bu standartları oluşturan içeriğin ne olduğu üzerinde durur. Özel ilgi grupları ise HL7 standartları içerisinde ihtiyaç duyulan yeni olguların araştırılması ve geliştirilmesi için test ortamı hazırlar.

Ülkemizde de HL7 standardı Tıp Bilişim Derneği'nin Sağlık Bakanlığı ile organize çalışmaları ile takip edilmektedir. Tıp bilişim Derneği'nin görevi HL7 standartlarını inceleyip değerlendirmek ve gelişimine katkıda bulunmakla birlikte bu standartların ülkemizde yaygınlaştırılmasını desteklemektir [20].

HL7 ifadesindeki Seviye 7, ISO'nun (International Standards Organization) OSI (Open System Interconnection) modelindeki 7. Seviye olan uygulama katmanından gelmektedir. OSI 7 layer bir network mimarisidir ki, verileri bir uygulamadan alıp bir başka bilgisayardaki uygulamaya taşır. Bu model verilerin transfer sürecini parçalara bölerek 7 adımda bir makineden diğer makineye veri taşımayı modeller. Düşük seviyelerdeki katmanlar, fiziksel ve veri bağlantısı (physical ve data link) donanımsaldır. Bunlardan sonrakiler yazılımsal olarak ele alınır. HL7'deki 7'nin geliş sebebi ise yedinci katman yani uygulama katmanın veri alışverişinde verinin tanımlanması ve işlenmesi uygulama seviyesinde yapılmasına olanak sağlamasıdır. Bu katmanda aynı zamanda güvenlik kontrolleri, kullanıcı tanımı, verinin uygunluğu ve transferin yapılandırılması gibi işlemleri destekler. Aşağıda OSI 7 Katman gösterimi verilmiştir (Şekil 2.4).



Şekil 2.4 - OSI 7 Katmanlı Referans Modeli

1.5.2 HL7'ın Önemi

Dünyada uluslararası olmak dâhil devam etmekte olan birçok sağlık standardı geliştirme çabası vardır. Bu durumda sağlık sektöründe çalışan bilişimcilerin kafasında HL7'a neden ihtiyaç duyulsun sorusu meydana çıkıyor. Fakat diğer standart çalışmaları sağlık sektörünün sadece belirli noktalarını temel alırken; örneğin radyoloji-pacs modülleri, randevu mekanizmaları vb, HL7 sağlık organizasyonunun tümüne konsantre olarak veri iletişimi de dahil olmak üzere toplu bir standart geliştirmeyi hedefler. Bununla beraber HL7 1980'lerin sonundan beri kitlesini dünya çapında arttırarak uluslararası bir standart akreditasyonu sağlamaya çalışmaktadır. Birçok ülkede HL7 standardı kullanılarak uygulama geliştirmeye

başlanmıştır. Hayatın ilerleyişi de artık bilgisayarlar üzerinden gerçekleştirildiği için sağlık sektörü gibi önemli bir sektörün verilerinin de değişik uygulamalar arası gönderimi, alımı ya da değiş tokuş edilmesi için bir standart olması uygulamaların birlikte çalışabilirliğini sağlayacaktır. Birlikte çalışabilirlik her açıdan önemlidir. Sadece sağlık sektörü için dahi düşünülecek olursa bir hastanın bir sağlık kuruluşuna gittiğinde yaptırdığı kayıt, yapılan işlemler, konulan tanılar, radyolojik kayıtları örneğin röntgen filmleri veya pacs görüntüleri, hesap kayıtları gibi bilgileri hastanın bir sonra başka bir sağlık kuruluşuna gittiğinde oraya göndermesi ve bu verilerin tanınması önemli bir mevzudur. Sadece sağlık kuruluşlarının aralarında haberleşmesi değil bu kuruluşların devlet kurumlarıyla da haberleşmesi gerekebilir. Örneğin adli vakalarda hasta geçmişinin ne olduğunu devletten yapılan sorgular sonucunda görebilmek mümkün olmalıdır. Tabi hasta kayıtlarının ne ölçüde paylaşılacağı da çok ayrı bir konudur ve ülkeden ülkeye göre değişir. İnsan hakları mahremiyeti her ülkede farklı olgulara dayanarak tamamının paylaşılması da yasak olabilir. Mesela hastaya en son gittiği hastanede konulan tanı herkes tarafından görülememelidir çünkü bu hastanın özel bilgisidir ve paylaşımı hasta mahremiyetini bozmaktır vb. Ama bunun yanında bir hastane bir laboratuarla bütünleşmiş çalışıyorsa ve hastanın yaptırması gereken tahlilleri laboratuara gönderir. Laboratuarda yaptığı tahlilleri ve sonuçlarını geri hastaneye iletir. Bu tarz veri iletişimlerinde de bir standart olması birlikte çalışabilirlik açısından çok önemlidir.

Ülkemizde de E-Devlet projeleri kapsamında birçok dönüşüm projesi başlamıştır ve değişik sektörlerde kullanılan uygulamalar ile devlet uygulamaları birbiriyle haberleşme yapmaktadır. Sağlık-NET veya Medula bunlara örnek teşkil edebilir. Fakat şuan için bu uygulamalardan sadece Sağlık-NET HL7 standardını desteklemektedir.

1.5.3 HL7'ın Yararları

HL7, üyelerinin ihtiyaçlarını en hızlı şekilde karşılamak amacıyla bir protokol setini geliştirebilir. Dahası, HL7 çalışmaları, halen kullanılmakta olan kimisi olgunlaşmış teknolojilerin ürünü olan birçok Hastane Bilgi Yönetim Sistemi ve

Departmantal/Klinik Sistemlerin özgün ihtiyaçlarına da yoğunlaşmaktadır. HL7 bir taraftan yeni ortaya çıkan ihtiyaçların karşılanmasına odaklanırken, diğer yandan Amerika'da ve uluslararası alanda sürdürülmekte olan diğer standart geliştirme faaliyetleriyle koordinasyonu ve uyumu sağlamaktadır. Arjantin, Avustralya, Kanada, Çin, Çek Cumhuriyeti, Finlandiya, Almanya, Hindistan, Japonya, Kore, Litvanya, Hollanda, Yeni Zelanda, Güney Afrika, İsviçre, Tayvan, Türkiye ve İngiltere HL7 üyesi ve HL7'nin yaygın olarak uygulandığı ülkelerdir. HL7 üyelerinin (Kullanıcılar, firmalar, kurum/kuruluşlar ve danışmanlar) farklı gereksinimlerini tespit etmek ve desteklemek için ciddi gayret sarf etmektedir. Üyelerinin ihtiyaçlarını, gereksinimlerini, önceliklerini ve ilgi alanlarını bilen HL7, üyelerinin bulundukları organizasyonlara katkıda bulunmaları için onlara destek olur. HL7 komite yapısı, dengeli oylama prosedürleri ve herkese açık olan üyelik politikaları, gereksinimlerin HL7 dâhilinde dengeli, uyumlu, kalite ve tutarlılığı eşit oranda sağlayacak şekilde değerlendirilmesini sağlar [20].

1.5.4 HL7'ın Geçmişi ve Yapısı

HL7 alış veriş sırasında standardında belirtilen çevirme kuralları dâhilinde verileri mesajlara çevirir. Bir HL7 mesajı bir sorgunun cevabı olabileceği gibi beklenmeyen zamanda gelen bir güncelleme de olabilir. Sorgulardan kast edilen basit bir laboratuar sonucu istemi bile olabilir, örneğin 75432 arşiv numaralı hastanın patoloji işlemleri sonucu gibi. Bir talebe karşılık cevap olarak gelmeyen güncelleme mesajları ise hasta kaydının tamamı veya değişmiş kısmı olabilir. Bunların yanı sıra HL7 ile istenildiği zaman sorgu yapılabilir veya gönderilmiş bir sorguyu tekrardan gönderilebilir. Sorgular ve beklenmeyen zamanlı güncelleme mesajları görüntü veya kayıt tabanlı olabilir.

Görüntü tabanlı mesajlar (display messages) bu mesajları alan sistemler tarafından kayıt edilmesi zorunlu olmayan bu nedenle de sistemde tutulmayan mesajlardır. Bu mesajlar genellikle güncelleme verilerini taşırlar ve sadece ekranda görünürler veya yazıcı gibi aygıtlardan çıktı olarak alınırlar bu nedenle de sadece görüntüsü düzgün olması için görüntü düzeyinde formatlıdırlar. Kayıt tabanlı mesajlar (record oriented)

ise alanlara bölünmüş olarak biçimlendirilmiştir ve mesajı alan sistemler tarafından bu biçimlendirmeye göre kayıt altına alınır. Bu iki mesaj tipi de açık ASCII karakterleri şeklinde olduğu için çıplak gözle okunabilirliği vardır. Mesaj, verinin sistemler arası transferindeki atomik birimdir [21].

Bir kayıt mesajı birden fazla alandan oluşabilir. Bu alanlara örnek vermek gerekirse hasta adı, yaşı, cinsiyeti vb. alanlar hasta mesajını oluşturur. Bu alanlar belirli alanlardır ve belirli mesajları oluştururlar. Anlamlı veriler bu şekilde ortaya çıkar. Alanların tipi alfabetik, nümerik, tarih veya alfa-nümerik olabilir. Mesajdaki alanlar birer ayraç ile ayrılırlar. Bu alanlar mantıksal gruplarlar oluştururlar ve bu gruplara segment adı verilir. Kısacası mesajları da segmentler oluşturur. Hasta kimliği, alerji bilgileri veya isteğe bağlı ya da tekrar eden alanlar segment örneği olabilir.

Mesaj değiş tokuşunun ne zaman yapılacağı HL7'nin kontrolündeki tetikleyici durumlar ile sağlanır. Bu tetikleyici durumlar HL7 yapısındaki bir listede belirtilmiştir. Tetikleyici bir durum oluştuğunda mesajlar tek olarak ya da gruplanarak bir seferde yollanır. HL7 standardı, gerçek hayattaki bir sağlık kaygısının, iki ya da daha fazla sistem arasındaki veri değiş tokuşu gereğini oluşturduğunu farz eder. Örnek vermek gerekirse, birçok hastane ADT(kabul/taburcu/transfer) sistemini ve bir hasta fatura sistemi kullanmaktadır. Fatura kaydı hastaneye kabul edilen her hasta için oluşturulur. Hasta kabul edildiğinde ki bu bir tetikleyici durum anlamına gelir (A01÷kabul/ziyaret bildirimi), hasta hakkındaki genel bilgiler toplanır ve ADT sistemine girilir. Bir HL7 kabul/ziyaret uyarıcı mesajı (istem dışı güncelleme) ADT sisteminden, hasta fatura sistemine gönderilir. ADT sisteminden gelen ve uyarıcı mesaj ile toplanan hasta bilgileri (örnek olarak, isim, adres, doğum tarihi, tanılar, sigorta bilgileri vb.) hasta fatura sisteminin henüz kabul edilmiş hasta için bir fatura kaydı oluşturmasına olanak tanır [21].

HL7 standardı, iki sitem arasındaki hata mesajların nasıl iletileceğini kontrol eden OSI protokollerini takip eder. Bilgi iki sistem arasında değiş tokuş yapıldığında, alıcı sistem mesajın geçerliliğini kontrol eder ve gönderen sisteme, bilginin başarılı

şekilde alındığı, mesajın bütün gerekli bölümleri veya alanları içermediği ya da gönderen sistemle iletişim kurulamadığı gibi bilgileri içeren bir cevap mesajı yollar. Diğer durumlarda alıcı sistem, gönderen sisteme bir onay mesajı gönderme ihtiyacını önlemek için gelen bilgiyi belleğe kaydeder. Kullanılacak onay mesajı örneği ara yüz anlaşma kontratında belirtilmiş olup mesajın baş kısmına yansıtılır.

HL7 amaç bildirisi değiş tokuş yapılacak anahtar veri kümelerini belirtmiştir. Bunlar; klinik hasta bakımı ve yönetimi ile sağlık hizmeti servislerinin değerlendirilmesi ve teslimidir. 1988'de HL7 standardı ilk açıklandığında içeriği daha çok hasta bilgisi ayarları ile sınırlıydı. Bu ayarlar; ADT, siparişler, gözlemler ve finansal veriler üzerinedir. O zamandan günümüze, HL7'nin etki alanı günümüz sağlık hizmetlerine uygun bir şekilde genişlemiştir. Bugün en sık kullanılan sürüm olan 2.3 sürümü (sık kullanılmasının sebebi eski olması en güncel sürüm 3.x), ayakta tedavi edilen hasta mesajlarını da içerdiği gibi, günümüzdeki yaygın yönetilmiş bakımları ve kuruluşlar arasındaki iş akışı ihtiyacını da yansıtır. Bu sürüm bir gönderme bölümü içerir. Bu bölüm başlıca sağlık sağlayıcıları, uzmanlar, ödeme yapanlar, laboratuarlar ve diğer hasta havaleleri kapsamındaki kuruluşları arasındaki veri değiş tokuşunu sağlayan mesajlar üretir. Yeni planlama bölümü servisler ya da kaynaklar için randevu planlama ile ilgili durum verisi içeren mesajlar içerir. Hasta bakım bölümü bir hastanın sağlık sorunlarını, ilgili hedefleri ve bu hedeflere ulaşmak için izlenmesi gereken yolları takip etmek için tutulan kayıtların iletimi için mesajlar sağlar. Yeni bir bölüm olan medikal kayıt bölümü ise ilgili alanı destekleyen mesajlar üretir.

Bir takım organizasyonlar proje geliştirmek, sağlık pazarındaki ihtiyaçlara çözüm bulmak için düzenli olarak HL7 ile işbirliği içindedirler. CDC (Centers for Disease Control and Prevention) kendi bünyesindeki bağışıklık kayıtları için HL7 2.3 sürümündeki mesajların geliştirilmesinde HL7 ile birlikte çalışmıştır. Bu mesajlar çeşitli sağlık birimleri ile devlet ve cemiyet veritabanlarının sorgulanmasına, güncellenmesine ve bağışıklık kayıtlarının değiş tokuşuna olanak verir. CDC HL7 standardını aynı zamanda elektronik laboratuar kayıtları tutulmasında ve kanser kayıtları için de kullanır. 1997 de duyurulan CDC'nin DEEDS(Data Elements for Emergency Depart Systems) biriminin 1.0 sürümü HL7 mesaj bölümlerini kullanarak

acil birimlerin sistemleri tarafından kullanılan veri elementlerinin standartlaştırılmasında yardımcı olmuştur. Bu veriler standart hale geldiğinde gözetim, toplum risk değerlendirmeleri ve araştırmalar gibi çeşitli kamu sağlığı konuları için toplanabilir ve birleştirilebilir. HL7, 1996 HIPAA'ya (Health Insurance Portability and Accountability Act) karşılık olarak sigorta poliçelerindeki alacakları otomatikleştirmek amacıyla bir kavram standardı sağlayabilmek için bir başka standart kuruluşu olan Health Care Financing Administration ile işbirliği içerisindedir [21].

Versiyon 2'nin segment yapısı gün geçtikçe ihtiyaçları karşılamamaya ve yetersiz kalmaya başladı. HL7 mesajları geliştiren teknik komitelerin kendilerine göre mesajlar üretmesi ile birlikte ortaya aynı bilgiye erişebilmek için farklı formatlı mesajlar çıkmıştır. Bu da HL7'nin içinde bile mesaj uyumsuzluğu problemini doğurmuştur. Ayrıca mesajların içerisine herkesin ekleyebileceği opsiyonel alanların olması en sonunda standartlaşmaya ters bir mantık oluşturmaktadır.

Versiyon 2'nin dezavantajlarından biri de uygulamaların HL7 uyumlu olduğunu kontrol edebilen bir mekanizma olmayışıdır. Uygulama sadece bir tane bile HL7 uyumlu mesaj içeriyor olsa dahi komple HL7 uyumlu olarak kabul edilebilmektedir.
Yazılım geliştiriciler için de Versiyon 2 mesajlarını bilgisayarlara otomatik tanıtma gibi teknolojileri kullandıramaması sorun teşkil etmektedir. HL7 mesajlarını anlayabilmek için her uygulamaya mesaj dönüştürücü arayüzler yazmak kaçınılmazdır. Tek bir uygulama içerisinde çok büyük problem olmasa dahi farklı uygulamaları birbirleriyle haberleştirmek, mesaj opsiyonluluğundan doğan esneklik yüzünden güçlük çıkarmaktadır. Bu durum da tabii zaman ve maddi kayıplara sebep olmaktadır.

1.5.5 HL7 Versiyon 3

Versiyon 2'nin yetersiz kalması HL7 organizasyonunu harekete geçirdi ve komple yeni bir sistem olan versiyon 3 üzerine çalışmalar başladı. Versiyon 3 geliştirilirken

v2 ile uyumlu olsun diye bir düşünce üzerinde durulmadı, v3 tamamen yeni temeller üzerine kuruldu.
XML tabanlı olan versiyon 3 mesajlarında en büyük hedef opsiyonelliğin azaltılarak mesaj uyumsuzluğu oluşturabilen vakaların önüne geçmektir. Bu sürümün uygulama geliştiriciler için en net sürüm olduğu HL7 yetkilileri tarafından açıklanmaktadır [19]. Böylece farklı firmalar yazılımlarının birbirlerine uyumluluğunu test ederken zorlanmayacaktır. V3'de nesneye yönelik programlama ile mesajlar üretilir ve mesajlar tanımlarını yine V3'ün gelişmesine en büyük pay sahibi olan anlamsal sözlükten alır. Bu sözlüğün adı Referans Enformasyon Modeli'dir (Reference Information Model - RIM).

Referans Enformasyon Modeli ile v3 metodolojisi nesne tabanlı mimari üzerine dayandırılmıştır. Sağlık kayıtlarının dünyadaki ihtiyaçlar doğrultusunda tasarlanarak belirli bir standarda oturtulmuş gösterimidir. Bu gösterim yazılı açıklamalardan daha çok görsel grafiklere ve UML'i andıran HL7'ın kendine özgü dili ile gerçekleştirilmiştir. RIM bu kapsamda sağlık kayıtlarının saklanmasında ve transferinde oluşacak mesajların doğru tanımlanması için yol gösterir. RIM yazılım geliştirme için doğruluğun ve bütünlüğün arttırılmasını sağlayarak farklı uygulamaların birlikte çalışabilmesi için yapılan entegrasyon çalışmalarında zaman kaybını minimuma indirerek masrafları azalmakta büyük rol oynar.

RIM sayesinde sağlık uygulamalarında kullanılacak olan klinik verileri belirli sınıf tanımlarına gömülmüştür. HL7 uyumlu olmak isteyen bir uygulama da RIM'den faydalanılmalıdır. Fakat her sağlık uygulamasın kendi içerisinde belirli alanlara dağılımından oluşan parçacıkları vardır ve bu parçacıkları geliştiren yazılımcılar farklı ekipler olabileceği gibi aynı bile olsa o an odaklandıkları alan üzerinde RIM gibi büyük bir sözlükte kaybolabilirler. Her nesne tabanlı dilde olduğu gibi HL7'da da sınıflardan yeni nesneler yaratılabilmektedir. Bu sayede RIM'deki sınıflardan sağlık uygulamasının belirli alanlarına göre nesneler kopyalanarak sadece o alana özgü sınıflar yaratılabilir. HL7'da ilk önce alanların mesaj modellemeleri oluşturulur. Bu modellemeye Alan Mesaj Bilgi Modeli (Domain Message Information Model - DMIM) denir. DMIM oluşturulduktan sonra da sıra bu sınıf kümesinden tek tek

sağlık kayıtlarını oluşturacak mesajların oluşturulması için bilgi modelleri üretilmelidir. Bu bilgi modeli çeşitli filtreleme aşamalarından geçtiği için HL7'da Rafine edilmiş Mesaj Bilgi Modeli (Refined Message Information Model - RMIM) olarak tanımlanır [35].

DMIM'den gerekli sınıflar RMIM ile de mesajların içerikleri belirlenir. Fakat HL7 V3 XML tabanları mesajları oluşturmak için sadece DMIM ve RMIM yeterli olmamaktadır çünkü oluşturulan bu modellerin nasıl XML'e dönüştürüleceği konusunda bir bilgi saklanmamaktadır. XML mesajlarını oluşturacak bir şema ihtiyacı bu aşamada çıkmaktadır. Bu yüzden birde RMIM'den XSD'ye dönüştürme yapılması gerekmektedir. HL7'da bu dönüştürme işlemini Hiyerarşik Mesaj Tanımı (Hierarchical Message Description - HMD) kavramı altında tablolar yapmaktadır. HMD'lerden XSD'ye çevirme işlemleri HL7'ın kendi sitesinde yayınlanan araçlarla veya bazı üçüncü parti yazılımlarla yapılabilmektedir [22][35].

Anlatılan bu modellemelerle HL7 V3'ün çok dinamik bir yapıya sahip olduğu ve genel yapıları (generic) barındıran RIM'den spesifik nesnelere doğru indirgenebildiği kanıtlanmaktadır. HL7 mesajları içerisinde klinik doküman bilgilerinin anlaşması için de yine ayrı bir standart mevcuttur. HL7 Klinik Doküman Mimarisi (Clinical Document Architecture - CDA) ile transfer edilecek olan klinik bilgilerinin formatları ve içerikleri belirlenmektedir. CDA dokümanları da diğer HL7 mesajlarından farksız olarak yapısını RIM'den almaktadır. Fakat CDA bir HL7 mesajı içerisinde transfer edilebileceği gibi farklı mesaj yapıları içerisinde, hatta XML tabanlı olmasının getirdiği esneklikle tek başına dahi gönderilebilir.

Çizelge 2.6 - CDA Doküman Örneği [23]

```
<ClinicalDocument>
  ... CDA Header ...
  <structuredBody>
    <section>
      <text>...</text>
      <observation>...</observation>
      <substanceAdministration>
        <supply>...</supply>
      </substanceAdministration>
      <observation>
```

```
        <externalObservation>
          ...
        </externalObservation>
      </observation>
    </section>
    <section>
      <section>...</section>
    </section>
  </structuredBody>
</ClinicalDocument>
```

Çizelge 2.6'da görüldüğü gibi CDA mesajı bir başlıktan ve gövdeden oluşmaktadır. Başlık kısmı içerisinde dokümanın kimlik bilgileri, kimler tarafından ve ne zaman hazırlandığı, hangi dilde olduğu, içerisinde neyin taşındığı gibi tanıtım bilgileri tutulur. Gövde kısmında ise asıl klinik bilgilerinin kendisi taşınır. Çizelge 2.7'de örnek bir klinik dokümanı incelenebilir.

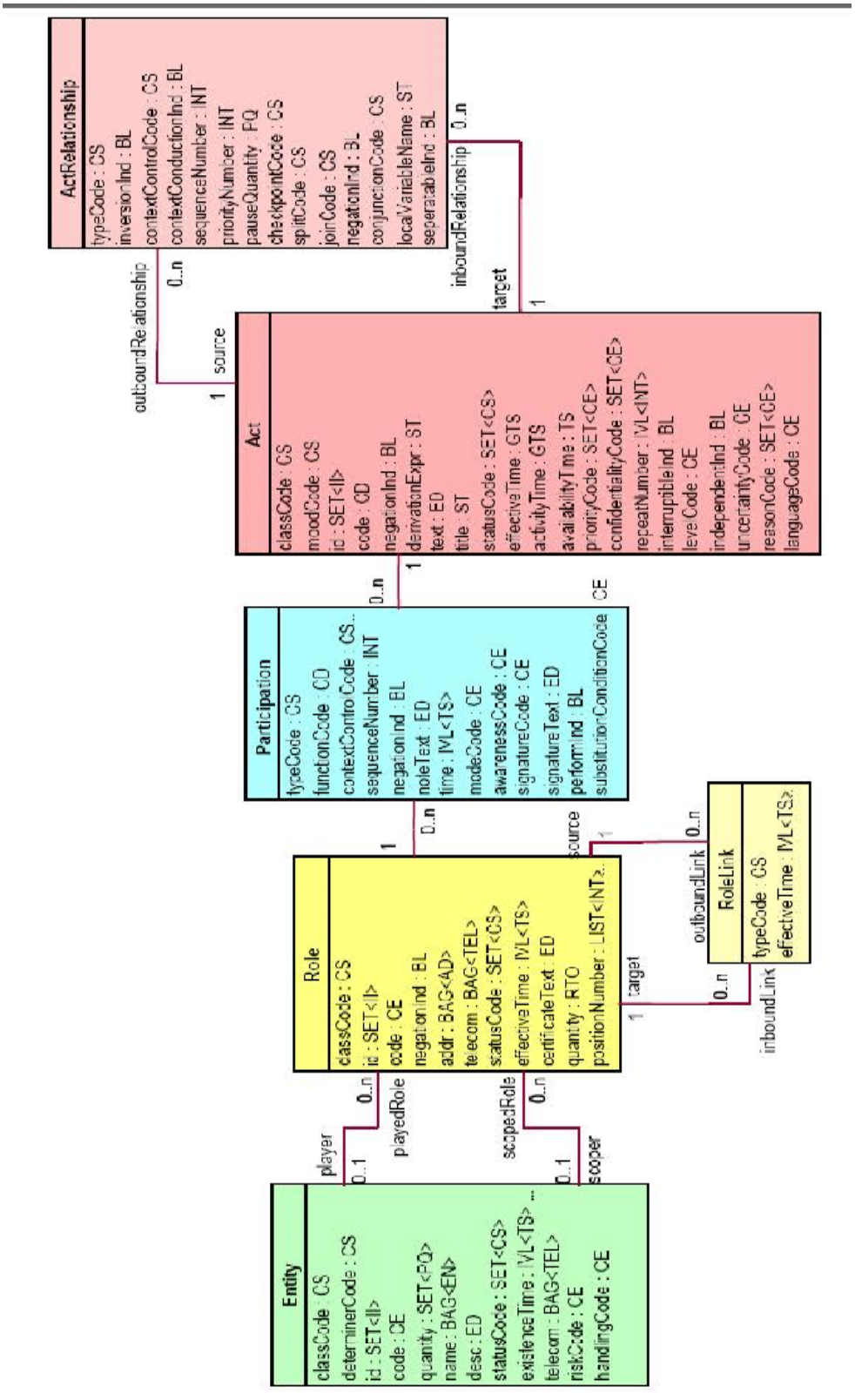
Çizelge 2.7 - Örnek Mesaj İçeriği [23]

```
<section>
  <code code="10153-2"
   codeSystem="2.16.840.1.113883.6.1"
   codeSystemName="LOINC"/>
  <title>Past Medical History</title>
  <text>
    There is a history of <content ID="a1">Asthma</content>
  </text>
  <entry>
    <observation classCode="OBS" moodCode="EVN">
      <code code="195967001"
       codeSystem="2.16.840.1.113883.6.96"
       codeSystemName="SNOMED CT"
       displayName="Asthma">
        <originalText>
          <reference value="#a1"/>
        </originalText>
      </code>
      <statusCode code="completed"/>
    </observation>
  </entry>
</section>
```

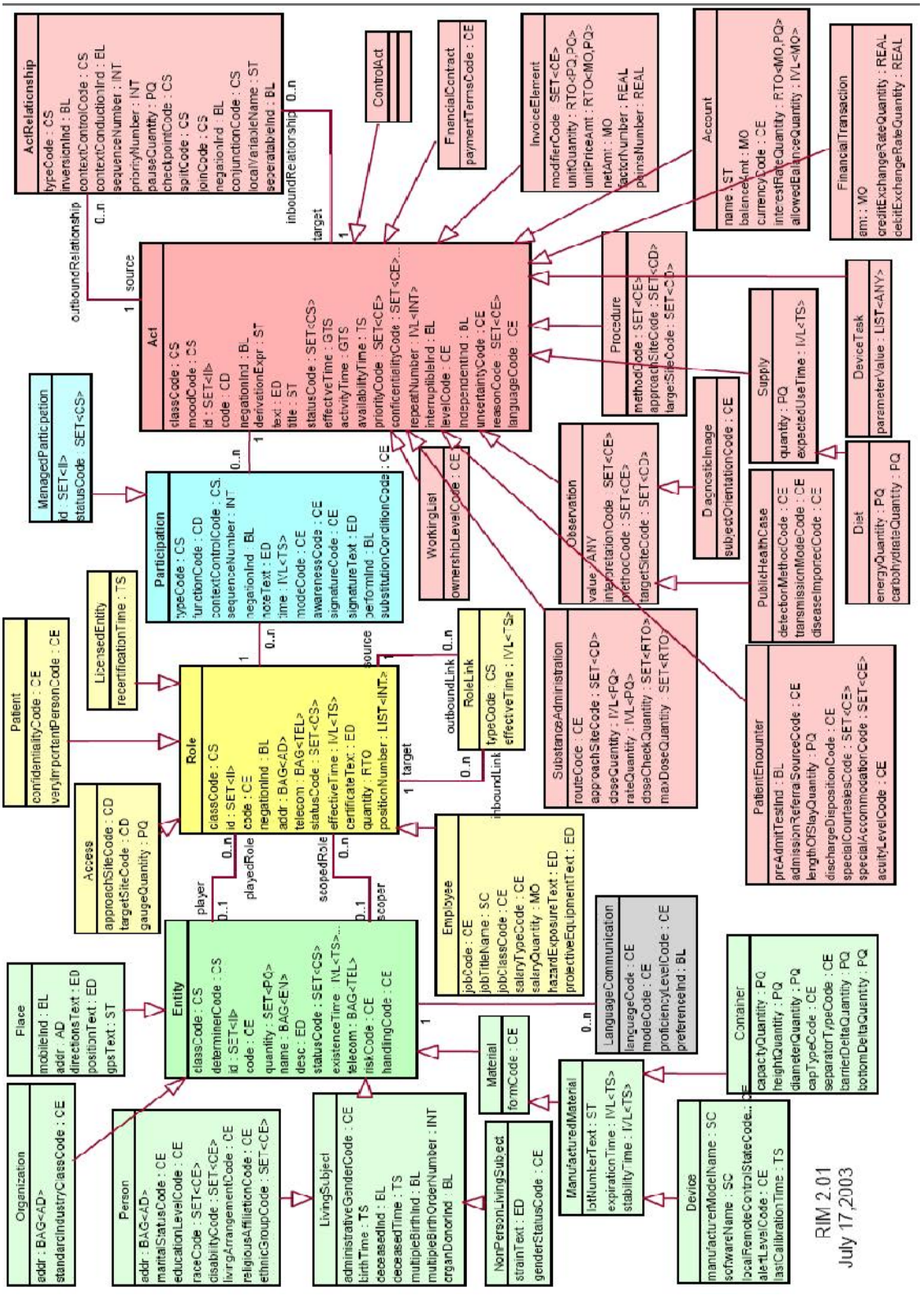
1.5.6 Referans Enformasyon Modeli – RIM

HL7 V3'ün mesaj alt yapısını oluşturan RIM'in içerisinde altı adet ana sınıf vardır. Bu ana sınıflar aşağıdaki gibi tanımlanır;

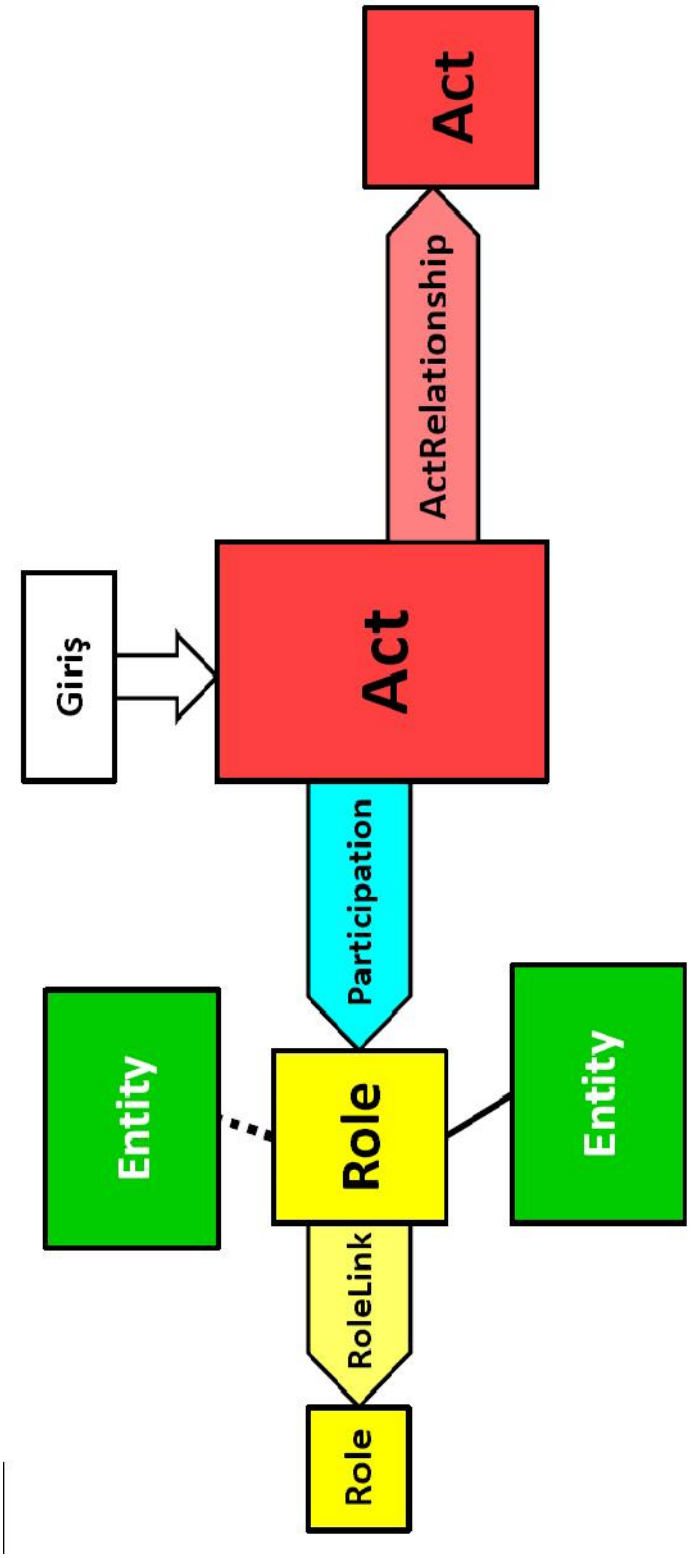
- Act: Sağlık sürecindeki tüm eylemlere (action) denir. Muayene, yapılan tahlil/tetkikler, muayene sonrası izlem, muhasebe olayları vb. RIM'in en temel sınıfıdır.
- Entity: Sağlık sürecinde bulunan canlı veya cansız varlıklara denir. Kişiler, tıbbi sarf malzemeler, cihazlar vb.
- Role: Entity'lere yüklenen rolleri ve görevleri temsil eder. Örneğin kişilerin Doktor veya Hasta rolleri gibi.
- Participation: Bir Role'ün bir Act'e nasıl katıldığını ifade eder. Örneğin doktor rolü ameliyata katıldığında ameliyatı yapan olarak nitelendirilirken hasta ise ameliyat edilen olarak nitelendirilir.
- ActRelationship: birden fazla Act arasındaki ilişkiyi ifade eder. Örneğin doktorun hastadan Tetkik istemesi, Muayene sırasında aldığı kararın sonucudur. Bu tarz bir ilişki, "neden" (Reason) ilişkisi olarak tanımlanmaktadır.
- RoleLink: Bu sınıf da Role'ler arasındaki bağlantıyı temsil eder.
- Aşağıda bu temel altı sınıfın sınıf diyagramları ve HL7 RIM sınıflarının birbirleriyle olan ilişkilerinin UML'e benzeyen diyagramı gösterilmiştir; Şekil 2.5, Şekil 2.6, Şekil 2.7



Şekil 2.5 - Temel RIM Sınıfları [24]



Şekil 2.6 - RIM'den Türeyen Sınıflar [24][25]



Şekil 2.7 - HL7 RIM Gösterimi [24][25]

RESTFUL WEB SERVİSLERİ

Bu başlık altında HTTP protokolü için çok önemli bir isim olan Roy Fielding'in Web 2.0 yaklaşımıyla da örtüşen RESTful Web Servisleri anlatılacaktır.

1.6 RESTful Nedir

REST, Roy Fielding (Roy Fielding HTTP için de ciddi çalışmalar da bulunmuştur) tarafından doktora araştırmasında ağ sistemlerinden bir mimari stili tasvir etmek için kullanılan bir deyimdir [28]. REST, İngilizce Representational State Transfer (Temsili Durum Transferi) kelimelerinin akronimidir. REST'i anlayabilmek için öncelikle temsili durum transferi kalıbını açıklamak gerekmektedir.

Web kaynaklardan oluşmuştur. Bir kaynak, bir ilgi gurubunun herhangi bir parçasıdır. Örnek olarak Ford otomotiv şirketi bir Mondeo modeli araba kaynağı olarak değerlendirilebilir. İstemci bu kaynağa http://www.ford.com/mondeo olarak da erişebilir. Burada kaynağın "temsili" kullanıcıya döner (örn: fordmondeo.html). Temsil, istemci uygulamayı bir "durum" a getirir. Sonuç olarak istemci bir köprü (hyperlink) sayesinde başka bir kaynak olan fordmondeo.html kaynağına erişir. Yeni temsil, istemci uygulamayı başka bir duruma sokar. Böylelikle istemci uygulama her kaynak temsilinde durum değiştirir, durumu transfer eder. Bu da temsili durum transferi (representational transfer) olarak adlandırılır. Roy Fielding ise bu durumu şöyle açıklıyor: "Temsili durum transferi bir web sitesinin ne kadar iyi planlandığının resmini hayal etmek için tasarlanmıştır. İyi tasarlanmış bir web sitesi, kullanıcının bir uygulama kullanarak web sayfalarından oluşan bir ağda(bir sanal durum makinesi), çeşitli linkleri seçerek (durum geçişleri) her seferinde kullanıcı için transfer edilen ve kullanımına hazır hale getirilen sonraki sayfa (uygulamanın bir sonraki durumunun temsili) ile sonuçlanan bir işleyişe sahip olmalıdır" [26].

REST bir standart değildir. Bu yüzden REST ile ilgili W3C şirketler birliğinden bir tanımlama ya da beyanname beklemek veya herhangi bir firmanın REST geliştirme araçları ya da programları satmasını beklemek yersizdir. REST sadece bir mimari

stildir. Bir standart olmadığı için kullanıcılar sadece bu stili öğrenerek kullanabilirler, farklı şekillere sokamazlar. REST bir standart olmadığı gibi çeşitli standartları kullanır. Bu standartlar:

- HTTP
- URL
- XML/HTML/GIF/JPEG vb. (Kaynak Temsilleri)
- text/xml, text/html, image/gif, image/jpeg, vb. standartlardır [26].

RESTful web servislerinin http metotları ve çoğu servisin kullanıma sunduğu CRUD operasyonlar ile aralarında doğal bir eşleştirme vardır. Kesin ve hızlı kurallar olmamasına rağmen aşağıdaki talimatlar pek çok durumda uygulanabilir. HTTP metotları aşağıdaki gibi özetlenebilir;

- GET bir kaynağa sorgu yapmak için ya da bir kaynaktan veri almak için kullanılır. Kaynaktan kullanıcıya geri dönen veri ise istenilen kaynağın bir temsilidir.
- PUT yeni bir kaynak yaratmak için kullanılır. Web servisi bu duruma başarılı ya da hatalı olarak yanıt verir.
- POST ise var olan bir kaynağın veya verinin güncellenmesinde kullanılır. Yeni bir kaynak yaratmak veya kaynağı silmek için de kullanılabilir.
- DELETE ise var olan bir kaynağın veya verinin silinmesi işleminde kullanılır [27].

Bazı durumlarda, güncelleme ve silme işlemleri POST eylemi ile de yapılabilir. Buna örnek olarak PUT veya DELETE eylemlerini desteklemeyen tarayıcı programları ile donatılmış servisler gösterilebilir.

Kullanıcıların, yazılım geliştiricilerin ya da web tasarımcılarının “create”, “read”, “update”, “delete” (crud) fonksiyonlarını tanımlayabilmeleri için gereken kelimeler http teknolojisinde de olduğu gibi PUT, GET ve POST, DELETE olarak belirlenmiştir. Bu kelimeler SQL veritabanı geliştirme dili ile de benzerlikler göstermektedir fakat REST stili ile HTTP kesinlikle birbirlerinden bağımsız ve

özeldir. Tekrar vurgulamak gerekirse REST bir standart değil, mimari bir yaklaşımdır. HTTP metotlarının SQL metotları ile olan benzerlikler ve karşılıkları aşağıdaki tabloda gösterilmiştir (Çizelge 3.1).

Çizelge 3.1 - SQL ve HTTP kelimeleri arasındaki ilişki

Eylem	SQL	HTTP
Create	Insert	PUT
Read	Select	GET
Update	Update	POST
Delete	Delete	DELETE

1.7 RESTful Web Servislerinin Özellikleri

Web mimarisinin altında yatan mantık, mimarideki elementlere uygulanacak bir takım kısıtlamaların oluşturduğu bir stil olarak açıklanabilir. Her kısıtlamanın oluşturduğu etkiyi araştırarak web kısıtlamaları sonucu ortaya çıkan özellikleri tanımlayabiliriz. Yeni ve modern bir mimari stil yaratmak için, ilave kısıtlamalar sonradan uygulanabilir [28]. Bu bölümde mimari bir stil olarak REST tabanlı web servislerin özellikleri adım adım açıklanacaktır.

1.7.1 Başlangıç Olarak Sıfır Değeri

Mimari tasarım olarak bina mimarisi ve yazılım mimarisi olarak iki tane temel prensip vardır. İlk prensipte tasarımcı işe bir boşluk, temiz bir sayfa ya da bir çizim tahtası ile işe başlar ve hedeflenmiş kıstaslara ulaşana kadar bilinen elemanlarla bir mimari ortaya çıkarana kadar devam eder. İkinci prensipte ise tasarımcı sistem gereksinimlerini bir bütün halde, sınırlılık olmadan ele alarak başlar ve daha sonra zamanla belirlenen kısıtlamaları, tasarım alanında farklılık yaratmak ve tasarımın sistemin davranışına işleyerek bir harmoni yaratmasını sağlamak amacıyla sisteme

ekler. İlk prensip yaratıcılığı ve sınırları olmayan bir görüşü desteklerken, ikincisi kısıtlanmayı ve sistem içeriğini anlamayı vurgular. REST yaklaşımı ikinci tasarım prensibine dayanarak geliştirilmiştir [28].

Sıfır değeri, basit olarak kısıtlamaların boş kümesidir. Mimari bir perspektif ile bakılacak olunursa, sıfır değeri öğeler arasında ayırt edilebilir herhangi bir sınırın bulunmaması olarak tanımlanabilir. Bu REST'in tarifinin başlangıç noktasıdır.

1.7.2 İstemci – Sunucu

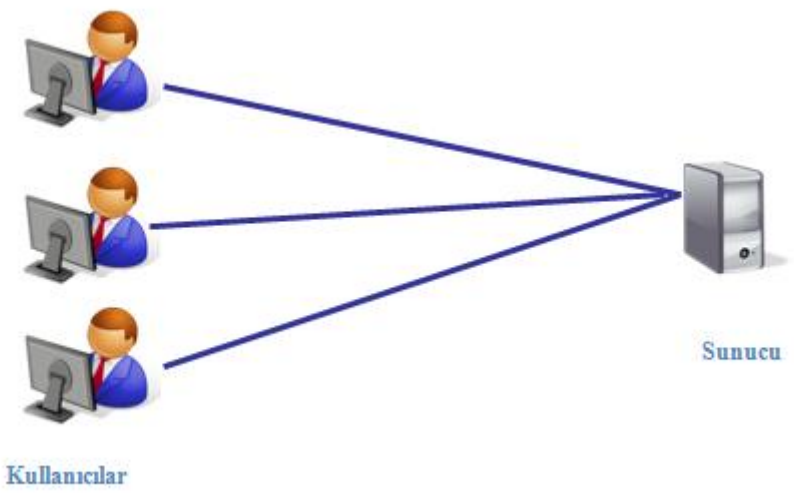
Tasarıma eklenecek ilk kısıtlama istemci-sunucu (client-server) mimari stilidir. Bağlantının ayrılması, istemci-sunucu kısıtlamalarının temel prensibidir. Kullanıcı ara yüzü ile veri saklama birimlerinin birbirinden ayrılmaları, kullanıcı ara yüzünün birçok platform arasında taşınabilirliğini ve sunucu elemanlarının basitleştirilmesi ile ölçeklendirilebilmenin artmasını sağlar. İstemci ve sunucunun ayrılması, Web'in gelişmesinde önemli rol oynar ve bileşenlerin birbirlerinden bağımsız olarak geliştirebilmelerine ve farklı uygulamaların birlikte çalışabilirliğine de olanak verir (Şekil 3.1) [28].

Şekil 3.1 - İstemci – Sunucu

1.7.3 Durumsuzluk

İstemci-sunucu altyapılı sistemlerde iletişim sağlandığında uygulamanın herhangi bir durumda (state) olmaması gerekmektedir. Böylece kurulacak olan sistem istemci-

durumsuz-sunucu yapısında olacaktır. Bu yapıda istemciler istediği zaman sunucuya istek yapacak ve sistem herhangi bir durumda olmadığı için her bir istemcinin isteğine karşılık uygun cevap içerikleri dönecektir. Oturum durumu ise basit web sitelerinde olduğu gibi istemci tarafında saklanabilir (Şekil 3.2) [28].



Şekil 3.2 – Durumsuzluk

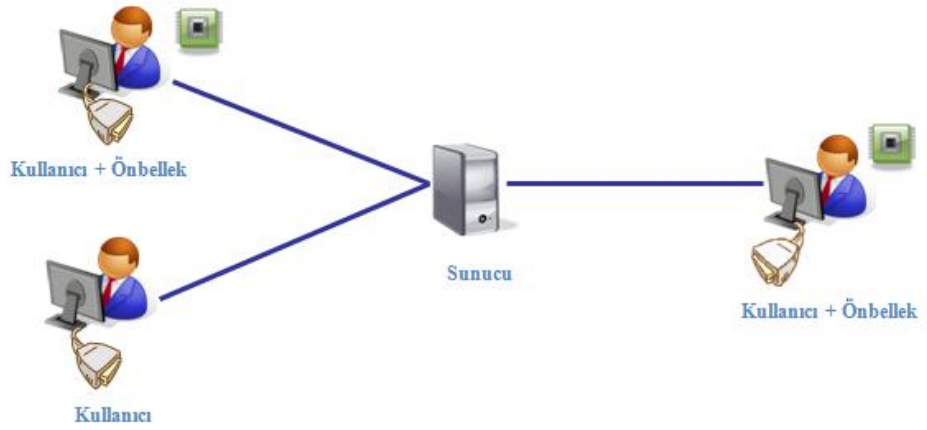
Bu sınırlılığın sonucunda görünebilirlik, güvenirlik ve ölçeklendirebilirlik gibi özellikler daha gelişmiş bir yapıya kavuşur. Görünebilirlik artar çünkü bir izleme mekanizması her isteğin içeriğini belirlemek için isteğin ötesine bakmak zorunda kalmaz. Güvenirlik artar çünkü istemci-sunucu stili, kısmi hataların düzeltilmesini kolaylaştırır. İstekler arası durumları saklamamak sunucuda daha fazla kaynak anlamına gelir, aynı zamanda sunucu, istekler arasında kaynak kullanımını yürütmek zorunda olmadığı için uygulamaların daha basit olmasına da olanak sağlar. Böylece ölçeklendirebilirlik artmış olur [28].

Çoğu mimari seçim gibi durumsuzluk kısıtlaması da bazı özelliklerden ödün verilmesini gerektirir. Seri isteklerin gönderimi sırasında çok fazla tekrar eden veri gönderimi yapılır ve bu, sistemin performansını, sunucu tarafında paylaşılacak veri kalmayıncaya kadar olumsuz yönde etkiler. Buna ek olarak uygulama durumunun kullanıcı tarafında tutulması, sunucunun, uygulamanın tutarlılığı üzerindeki

kontrolünün azalmasını sağlar. Bu durum uygulamanın çoklu kullanıcı sürümleri arasında semantiklerin doğru uygulanmasına bağlı olana dek sürer [28].

1.7.4 Ön Belleğe Alma

Ağ verimliliğinin artması için istemci-durumsuz-sunucu stiline, önbelleğe alma kısıtlamasının da eklenmesi gerekir. Önbelleğe alma kısıtlaması, bir istek ve cevap arasındaki verinin, dolaylı ya da açık olarak, önbelleğe alınabilir veya önbelleğe alınamaz ifadeleri ile etiketlenmesini gerektirir. Eğer bir cevap önbelleğe alınabilir ise bir kullanıcı önbelleği o cevap verisini daha sonra aynı istekler için tekrar kullanabilme yetkisine sahip olmuş olur (Şekil 3.3) [28].

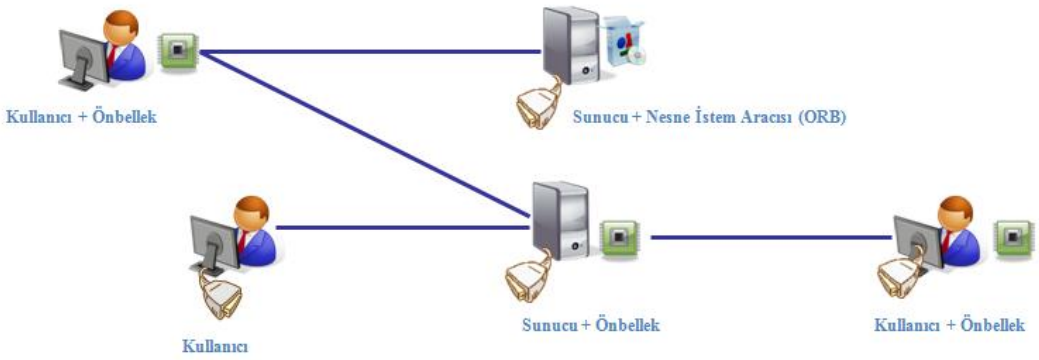


Şekil 3.3 - Önbelleğe Alma

Önbelleğe alma kısıtlamalarının eklenmesindeki avantaj, bir dizi etkileşimdeki ortalama gecikmeyi azaltarak, kısmen ya da tamamen bazı etkileşimleri ortadan kaldırma, verimliliği, ölçeklendirilebilirliği ve kullanıcı tarafından algılanan performansı arttırma potansiyelidir. Bu kısıtlamadaki verilen ödün ise eğer kullanıcı önbelleğindeki veri olması gerekenden eski veya farklı bilgiler içeriyorsa, sunucudan gelen cevaplar için bu bilgileri kullanmaya devam eder ve böylece güvenirliği düşürmüş olur [28]. Basit HTML web sayfalarında da bu özellikler vardır ve kurabiyeler (cookie) ya da geçici dosyalar ile önbellekleme yapılır.

1.7.5 Aynı Arayüz

REST mimari stilinin onu diğer ağ tabanlı tasarımlardan ayıran temel özelliği istemciler ve sunucular arasında ortak bir ara yüze sahip olmasıdır. Bir bileşenin ara yüzüne yazılım mühendisliğinin genelleme prensibinin eklenmesiyle tüm sistemin mimarisi basitleştirilmiş olur ve sistemdeki veri değiş tokuş takibi kolaylaşır. Uygulamalar, sağladıkları servislerden ayrılarak birbirlerinden bağımsız geliştirilmeye elverişli bir ortam sağlamış olurlar. Ortak ara yüz kullanmanın engeli ise bilginin, bir uygulamanın özel gereksinimlerine uygun olmaktansa standartlaştırılmış bir biçimde transfer edilmesinin verimliliği düşürmesidir. Ama bu da birlikte çalışabilir uygulamalar geliştirmek için artı puandır. REST ara yüzü, geniş bant ile çeşitli yüksek kapasiteli medya verileri taşımak, web'in genel durumunu optimize etmek için tasarlanmıştır [28].



Şekil 3.4 - Ortak Arayüz

REST arayüzlerindeki yaklaşım, kaynakların temsiller üzerinden kullanıcıya ya da diğer uygulamalara aktarılması, kendi tanımlarını içeren mesajların olmasıdır (Şekil 3.4 - Ortak Arayüz).

1.7.6 Katmanlı Sistem

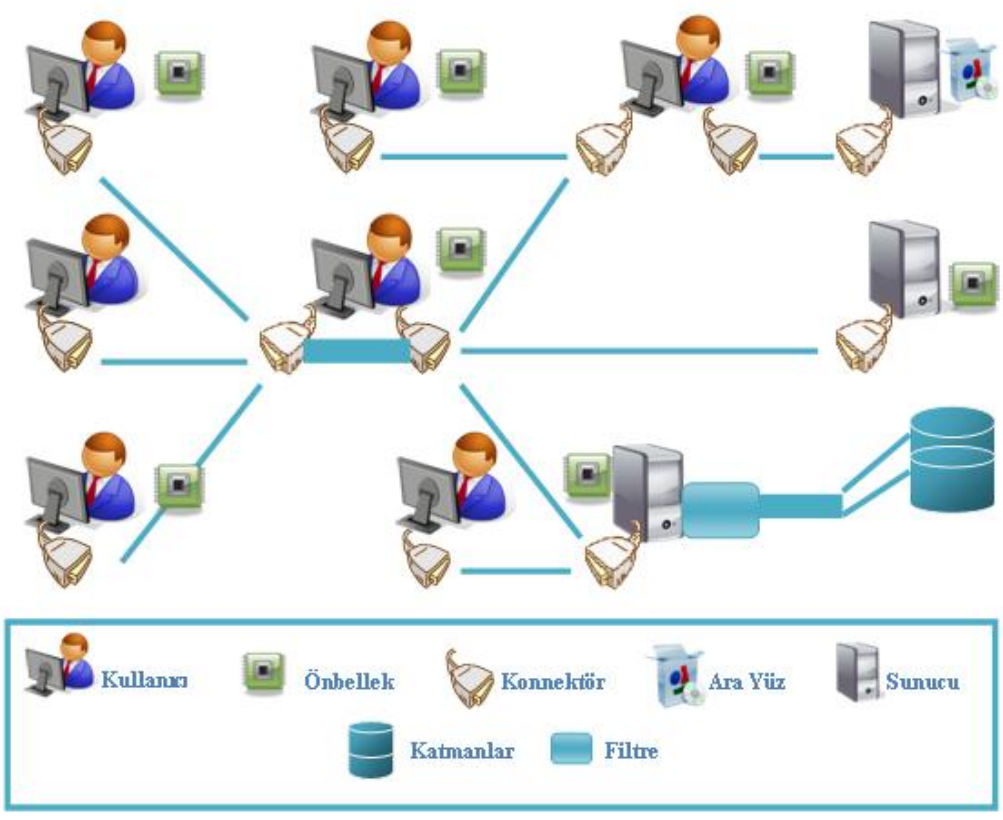
İnternet ölçeklendirme gereksinimlerinin gelecekteki geliştirilmeye açık olmaları için katmanlı sistem kullanılması önerilmektedir. Katmanlı sistem stili bileşenlerin davranışlarını kısıtlaması ile bir mimarinin hiyerarşik katmanlardan oluşmasını sağlar. Öyle ki, her katman hemen altında ya da üstünde etkileşimde olduğu katmanın ötesine ulaşamaz. Zaten HTTP'de bir transfer katman protokolüdür ve RESTful web servisleri bu protokole dayanmaktadır.

Tek bir katmanın sistem bilgisini öğrenmesini engellemek için genel sistem karmaşıklığına sınır koymak ve alt tabaka bağımsızlığını desteklemek gerekir. Katmanlar eski servisleri sarmalayarak yeni servisleri de eski servis kullanıcılarından ayırmada kullanılabilirler. Aynı zamanda ortak kullanılan bir aracı elemana nadiren kullanılan fonksiyonları kaydırarak bileşenlerin basitleştirilmesinde kolaylık sağlarlar. Aracı elemanlar aynı zamanda çoklu ağların ve işlemciler arasındaki servislerin ağırlık balansını ayarlayarak sistemlerin ölçeklendirilebilirliğini arttırır [28].

Bu çalışma sırasında uygulama katmanı içerisinde de katmanlar sağlanarak HTTP isteklerinin kaynaklara ulaşmadan incelenip bölünerek hangi metotlarda ne yapmak istediğinin anlaşılabilmesi için denemeler yapılmıştır. Microsoft NET teknolojisi kullanılarak bir http isteği daha kaynağa ulaşmadan yakalanıp, isteğin hangi kaynağa hangi eylemi gerçekleştireceği anlaşılabilmiştir.

Katmanlı sistem kullanmanın dezavantajı ise verinin işlenmesine yük ve gecikme katarak, kullanıcı tarafında performans düşüklüğüne neden olabilmesidir, fakat bu durum yine yazılımın düzgün tasarlanmasına bağlıdır. Önbellek desteği veren ağ tabanlı bir sistemde bu durum aracılardaki ortak kullanılan önbelleklerin getirdiği faydalar ile dengelenebilir. Organize bir alanın izin verdiği kadar paylaşımlı önbellek eklemek, fark edilebilir derecede performans getirisi ile sonuçlanabilir. Katmanlar, verinin organize sınırlar arasında iletilirken güvenlik duvarlarının ihtiyaç duyduğu güvenlik talimatlarının uygulanmasını da sağlarlar.

Katmanlı sitem ve ortak ara yüz kısıtlamalarının birleşimi, mimari özelliklerin “boru ve filtre” (pipe-and-filter) tarzına benzemesine yol açar. REST etkileşimi iki yollu olmasına rağmen, gelişmiş medya etkileşimleri ve büyük veri akışları, bir veri-akış ağındaki gibi işlenebilir. Bu işlem, içeriğin olduğu gibi geçmesi için veri akışına seçilerek uygulanan filtreler ile sağlanır. REST ile aracı elemanlar mesajın içeriğini aktif olarak dönüştürebilirler çünkü mesajlar kendi kendilerini tanımlayıcı niteliktedirler [28]. Daha önce de bahsi geçen kendi tanımlarını barındıran mesaj yapısı Şekil 3.5’de gösterilmiştir.



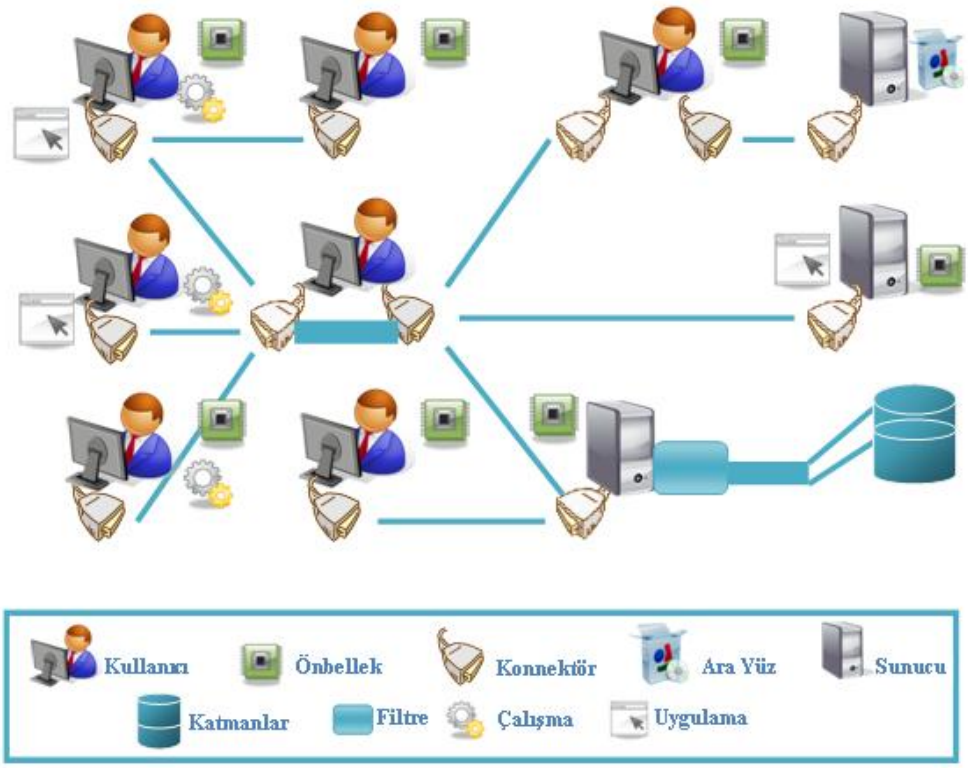
Şekil 3.5 - Katmanlı Mimari

1.7.7 İhtiyaç Anında Kod

İhtiyaç duyulduğunda koda ulaşabilme stilinde, istemci bir takım kaynağa ulaşır fakat bu kaynakları nasıl işlemden geçireceğini bilmez. Uzaktan kontrollü sunucuya

bu kaynağı nasıl işleyeceğini belirten kod için bir istek gönderir, kodu alır ve çalıştırır. REST yaklaşımındaki bir diğer özellikte budur. REST kullanıcı fonksiyonelliğini genişletilmesine, küçük uygulamalar veya komut dosyaları halinde bulunan kodları karşıdan yükleyip çalıştırılmasına olanak sağlar. Buna örnek olarak HTML ve Javascript ikilisi gösterilebilir. Kaynaklar HTML temsilleri ile kullanıcıya gösterilirken bir yandan da alt tarafta Javascript kodları işlemleri gerçekleştir. Hem de bunların hepsi istemci tarafında gerçekleşmektedir. Tabi bu örnek sadece istemci tarafında çalışan kodlar için değil, istemcinin ihtiyaç duyduğu anda sunucuda çalışacak olan fonksiyonların çağırılması olarak da nitelendirilir. Web servisleri ile günümüzde geliştirilen uygulamalar bu şekilde çalışmaktadır.

Bu sayede, istemci tarafında bulundurulması gereken özellik sayısını azaltarak kullanıcı tarafının daha basit tutulabilmesini sağlar. Sistem kurulumundan sonra yeni özellikler yüklenebilmesi sistemin genişletilebilmesi özelliğini geliştirir fakat aynı zamanda görünebilirliği düşürür ve bu yüzden REST için isteğe bağlı bir kısıtlamadır. İsteğe bağlı bir kısıtlama kavramı çelişki gibi görünebilir. Fakat sistemin mimari yapısında, birçok organize sınırlar içermek gibi bir nedeni vardır. Bunun anlamı, isteğe bağlı kısıtlamalar sadece tüm sistemin belli bir alanlarında etkili olduğunda mimari yapı bu kısıtlamalardan yarar sağlar veya zarar görür. Örnek olarak, bir organizasyonun tüm kullanıcı yazılımlarının Java programlarını desteklediği bilinirse, organizasyonun servisleri, karşıdan yüklenebilir Java sınıfları ile ileri düzey yarar sağlayacak şekilde tasarlanabilirler. Aynı zamanda, organizasyonun güvenlik duvarı, dış kaynaklardan Java programlarının transferini önleme ihtimali olsa da web'in geri kalanı için bu kullanıcılar ihtiyaç anında kod stilini desteklemiyor görünürler. İsteğe bağlı bir kısıtlama bize arzu edilen davranışın genel durumda destekleyen bir mimari tasarlamamıza olanak verir fakat bu kısıtlamanın bazı içerikleri de engelleyebileceği unutulmamalıdır [28]. Bu sorun istemci-sunucu bazlı olup web tabanlı olmayan uygulamalarda sıkça yaşanmaktadır. Web servislerinin popüler olmasında bu durumun da katkısı büyüktür.



Şekil 3.6 - İhtiyaç Anında Kod

1.8 RESTful Web Servislerinin Dizayn Prensipleri

- Bir REST ağında web servisi oluşturmanın anahtarı servis olarak sunulacak tüm kavramsal girdilerin belirlenmesidir. Bu tez çalışmasında prototip olarak geliştirilecek olan uygulamada hasta ekleme özelliği bu tip servis olarak örnek gösterilebilir.
- Her kaynağa bir URL yaratılmalıdır. Bu URL'lerdeki kelimeler fiillerden değil isimlerden oluşturulmalıdır. Örnek URL: www.etu.edu.tr/bolumler
- Kaynaklar, kullanıcıların bu kaynaklardan sadece temsil mi alacaklar yoksa kaynaklarda değişiklik yapabilecekler mi sorularına göre kategorize edilmeleri gerekir. Bunu da gelen isteklere göre kategorize ederek davranışlarını yani metotların gerçekleştireceği eylemleri ona göre ayarlamalıdır. Put, delete vb. işlevler.
- HTTP GET komutu ile erişilebilen her kaynak yan etkilere açık olmamalıdır. Bu, kaynağın sadece bir temsil ile kullanıcıya dönmesi anlamına gelir.

Kaynağa başvurmak kaynağı değiştirmek olarak sonuçlanmamalıdır. Örnek olarak bir web sayfasına ulaşınca karşımıza gelen HTML sayfasını sadece görülebilir ama değiştirilebilir olmaması durumu verilebilir.

- Kullanıcıların daha alt katmanlardaki verilere ulaşabilmesi veya bu veriler ile ilgili bilgilere ulaşabilmesi için kaynak temsillerine köprüler (hyperlink) konulmalıdır. Örnek: http://www.etu.edu.tr/bolumler/cs/
- Verilerin ortaya koymak için aşamalı tasarım kullanılmalıdır. Tüm bilgilerin tek bir cevap belgesine konulmaması gerekir. Daha fazla bilgi için köprüler kullanılmalıdır.
- Cevap olarak dönecek verinin bir taslak (DTD, W3C Taslağı, RelaxNG, veya Schematron, XSD) kullanılarak belirtilmesi yararlı olabilir. POST veya PUT komutlarına gereksinim duyan servisler için bu komut cevaplarının formatlarını belirlemek için de bir taslak kullanılmalıdır. Örneğin isteklere cevap olarak XML dokümanı gönderildi. Bu dokümanın belirli bir formatı olmazsa istemci bu veriyi kullanmakta zorluk çekebilir veya yanlış kullanabilir.
- Kullanılacak servislerin bir WSDL dokümanı ile mi yoksa basit olarak bir HTML dokümanı ile mi başvurulacağı tanımlanmalıdır [26][29].

1.9 RESTful ve SOAP Karşılaştırması

Bu çalışmaya başlamadan önce web servisleri denince aklıma gelen WSDL tanımları ve SOAP mesajları idi. Fakat RESTful ile web servis kullanırken aslında bu iki standarda ihtiyaç olmadığını, her şeyin daha da basite indirgenebileceği gerçeği ile karşılaştım.

XML'in gücünden web servisleri bölümünde bahsedilmişti. Gerçekten de esnekliği ve bu esnekliğini verilerin transferinde kullanmasındaki başarısı tartışılmaz bir gerçek. Fakat her türlü durumda XML'e ihtiyaç duyuluyor mu? İşte günümüz teknolojilerinde sıkça kullanılan web servis kavramında XML sadece veri taşımak için değil iletişimin her aşamasında gidip gelen bir zarf olarak kullanılmaktadır.

Sistemlere girerken yapılan kimlik denetimleri, servis tanımları, metot bilgileri ve daha niceleri. Kullandığımız yapının ne olduğunu belirlemek için gönderdiğimiz bilgilerin yanında bir de kullandığımız standartların başlıkları ve gerekli diğer bilgiler işin içine girince transfer edilen zarfların boyutu da gereksiz yere büyüyor.
REST yaklaşımında bunlardan kaçınılarak uygulamaların aslında basit web sayfalarından farksız, sadece HTTP'nin kullandığı eylemlerle tüm uygulamayı çalıştırabilecek bir mimari hedeflenmektedir. Yapının güzelliği de basitliğindedir. Yapılan isteklerde veya bu isteklerin cevaplarında ister XML, ister JSON veya istenirse düz string değerler gidip gelebilir.

Gerçek hayattan örneklerle yukarıda savunulanları desteklemenin uygun olduğunu düşünüyorum. REST veya SOAP servilerinden birini seçmek zorunda olan Thomson Yayıncılığın yaşamış oldukları örnek gösterilebilir. Thomson Yayıncılık, hazırda bulunan karakter dizisi yazılımlarına ek olarak geliştirilecek paket için bu iki web servisi arasında seçim yapmak zorunda kalmış ve yazılımcılar, SOAP yazılımına göre sunduğu üstün performans, güvenirlilik ve ölçeklendirebilirlik özelliklerinden dolayı REST yaklaşımını kullanmaya karar vermiştir. Bay Thomson yapılan çalışma sonucunda oluşturulan özel dizin sistemini (generic typesetting system-GTC) kullanarak çok daha kolay bir şekilde çeşitli veri kaynaklarından dizin oluşturmaktadır. Bu durumu, "Bu teknolojiyi kullanmadan önce her yeni veri kaynağı için özel bir çözüm yazmak zorunda kalıyorduk, şimdi ise RESTful web servisleri sayesinde Thomson'a özel olan dizin akışlarını hızlı bir şekilde oluşturabiliyoruz." diye açıklıyor [30].

Aşağıda SOAP ile REST sistemlerinin birbirlerine göre avantajları yer almaktadır. REST web servislerinin SOAP web servislerine göre avantajları şöyledir:

- RESTful sistemler temsilleri önbelleğe alma destekleri sayesinde gelişmiş cevap alma süreleri ve düşürülmüş sunucu yükleme zamanları sunmaktadırlar. Bunun yanında SOAP'ın zorunlu tuttuğu zarf yapısını kullanmadığı için gidip gelen verilerin boyutları da daha düşük olmaktadır.
- Oturum durumlarına erişmeye daha az ihtiyaç duydukları için RESTful sistemler sunucu ölçeklendirilmesinde SOAP sistemlere göre daha

gelişmişlerdir. Bu, değişik sunucuların aynı oturumda farklı istekler ile başa çıkabilmesi anlamına gelir.

- Sadece tarayıcı program her uygulamaya ve kaynağa ulaşabildiği için, istemci tarafında diğer servislere göre daha az yazılım çalıştırılmasına olanak tanır.
- Ek katman mesajlaşma yapıları http protokolünün en üst seviyesinde yer aldığı için ticari yazılımlara ve gereçlere daha az dayalıdır.
- İletişime alternatif yaklaşımlar ile karşılaştırıldığında, bu yaklaşımlara eşit derecede fonksiyonellik sağlar.
- Kaynakları temsilde köprüler kullandığı için ayrı bir kaynak bulma mekanizmasına ihtiyaç duymaz.
- RPC'den daha uzun zaman uyumluluk ve değişime daha fazla yatkın olması;
- Html gibi doküman tiplerinin ileri veya geri kırılım göstermeden yenilenebilme özelliği ve
- Yeni içerik tiplerinin, eski içerik tiplerine destek vermeyi sürdürebilmesi ile kaynakların yeni içerik tipleri eklemeye olanak sağlaması SOAP web servisine olan üstünlükleri olarak sayılabilir.

SOAP web servislerinin REST web servislerine göre avantajları ise;

- SOAP servisleri, kurumsal bilgi sistemlerinde API merkezli sonuçlar doğururlar.
- SOAP servisleri tamamlanmamış işlemli modeller için daha uygun olabilir, zira her kullanılan web servislere ekleme yaparak veya eni servis tanımlayarak yeni ihtiyaçlar çözümlenebilir. RESTful'da ise dizin yapısı çok karmaşık uygulamalarda belki sorun yaratabilir.
- SOAP servisleri genellikle bütünleşme odaklıdır ve sistemler arası uyumluluk uygulamaları için kullanılırlar.
- Güvenlik, kayıt, ekleme ve bir dizi iletişim modelleri (senkron, asenkron, istek/geri çağırma, uyarı vb.) sunduğundan geliştirilmesi ve tasarımı biraz daha karmaşıktır.

Ayrıca REST ve SOAP web servislerinin standartlar, araçlar, yazılımcı desteği ve güvenlik gibi kritik başlıklardaki karşılaştırılmaları Çizelge 3.2'de kısaca açıklanmıştır.

Çizelge 3.2 -RESTful ve SOAP web servislerinin karşılaştırılması[29][30]

	REST	SOAP
Standartlar	REST güncel internet standartlarını kullanacağını vaat etmektedir.	SOAP tabanlı yaklaşımlar kendi bünyelerinde bir dizi standartlar içerir.
Araçlar	Ticari amaçla üretilmiş araçlar yoktur fakat birçok geliştirme aracı http ve xml gibi REST standartlarını da destekler.	Uygulama geliştirme şirketleri SOAP tabanlı uygulamaları günümüzde herhangi diğer bir uygulama gibi gelişimini ve yayılmasını sağlamak için yazılımlar geliştirmektedir.
Yazılımcı Desteği	Günümüzde RESTful geliştiren yazılımcı azınlıktadır, ayrıca şirketler, kullanıcıların RESTful sistemler talep etmediklerini belirtiyorlar fakat REST yeni bir yaklaşım ve SOAP tabanlı web servislerinin baskın olduğu piyasalarda istikrarlı bir şekilde yaygınlaşmaktadır. Firmalar REST desteğini yeni yeni vermeye başlamışlardır.	BEA Sistemleri, IBM ve Microsoft gibi belli başlı yazılım geliştirme şirketleri, web servislerinin geliştirilmesinde SOAP tabanlı kitler tavsiye etmektedir.
Güvenlik	RESTful sistemleri savunan geliştiriciler, bu sistemlerin daha güvenli olduğunu ileri sürüyorlar çünkü REST internetin şu anda var olan güvenlik altyapısını kullanmaktadır.HTTP protokolünün desteklendiği güvenlik protokolleri, HTTPS, SOCKS, SSL vb.	SOAP güvenlik konusunda hala geliştirilmekte olan bir servistir fakat web yöneticilerine, web servislerini kimlerin kullandığı ve bu kullanıcıların yetkileri gibi durumlarda daha fazla kontrol sunma konusunda oldukça umut vermektedir.

Tasarımcılar için güvenli, istikrarlı, verimli gibi kavramlar ne kadar önemli ise web servisi kullanacak bir mimari tasarlarken sistemin performansı da o kadar önemlidir.

Kullanıcı tarafında hissedilecek performans kayıpları verimliliğin düşmesi ve hazırlanan sistemin kullanıcılar tarafından tercih edilmemesi gibi pek çok açıdan dezavantajlı sonuçlar doğurabilir.

1.10 Değişik Programlama Dillerinde RESTful

Bu bölümde, aralarındaki farkı daha iyi anlayabilmek için, RESTful bir sistemin HTTP GET ve HTTP POST isteklerinin çeşitli programlama dillerindeki örnek kodları karşılaştırılmıştır.

1.10.1 C# ile RESTful Kullanımı

C# ile http GET isteği kodlamanın anahtar sınıfları System.Net deki HttpWebRequest ve HttpWebResponse dur. Aşağıdaki metot bir istekte bulunarak gelen cevabı tek, uzun bir karakter dizisinde saklar (Çizelge 3.3).

Çizelge 3.3 - C# ile HTTP GET Kullanımı [47]

```
static string HttpGet(string url)
{
    HttpWebRequest req = WebRequest.Create(url) as HttpWebRequest;
    string result = null;
    using (HttpWebResponse resp = req.GetResponse() as
HttpWebResponse)
    {
        StreamReader reader = new
StreamReader(resp.GetResponseStream());
        response = reader.ReadToEnd();
    }
    return result;
}
```

Eğer çağırılan adres parametreler içeriyorsa bu parametreler tam anlamıyla kodlanmalıdır (boşluk %20 vb.). Bu tür kodlamaları yapmak için System.Web'in altında UrlEncode metodunu çağıran HttpUtility sınıfı kullanılabilir. Çizelge 3.3 ve Çizelge 3.4'de örnek C# kodları incelenebilir.

HTTP POST istekleri için form kodlamalarına ek olarak URL kodlaması da gerekmektedir.

Çizelge 3.4 - HTTP Post [47]

```
 static string HttpPost(string url, string[] paramName, string[]
paramVal)
{
    HttpWebRequest req = WebRequest.Create(new Uri(url)) as
HttpWebRequest;
    req.Method = "POST";

    req.ContentType = "application/x-www-form-urlencoded";

    // Build a string with all the params, properly encoded.
    // We assume that the arrays paramName and paramVal are
    // of equal length:
    StringBuilder params = new StringBuilder();
    for (int i = 0; i < paramName.Length; i++)
    {
        params.append(paramName[i]);
        params.append("=");
        params.append(HttpUtility.UrlEncode(paramVal[i]));
        params.append("&");
    }

    // Encode the parameters as form data:
    byte[] formData =
    UTF8Encoding.UTF8.GetBytes(params.toString());
    req.contentLength = formData.Length;

    // Send the request:
    using (Stream post = req.GetRequestStream())
    {
        post.Write(formData, 0, formData.Length);
    }

    // Pick up the response:
    string result = null;
    using (HttpWebResponse resp = req.GetResponse() as
HttpWebResponse)
    {
        StreamReader reader =  new
StreamReader(resp.GetResponseStream());
        result = reader.ReadToEnd();
    }

    return result;
}
```

1.10.2 Java ile RESTful Kullanımı

Java programlama dilinde HTTP GET isteğini kullanmak için, bir URL objesindeki openConnection metodunu çağırarak ulaşılan HttpURLConnection sınıfı anahtar sınıftır. Fakat openconnection metodu formatı gereği bir süper sınıf döndürür

(URLConnection) ve sonucu bir alt sınıfa yönlendirir. . Aşağıdaki metot bir istekte bulunarak gelen cevabı tek, uzun bir karakter dizisinde saklar.

Çizelge 3.5 - JAVA ile HTTP GET Kullanımı [48]

```
public static String httpGet(String urlStr) throws IOException
{
      URL url = new URL(urlStr);
      HttpURLConnection conn = (HttpURLConnection)
url.openConnection();

      if (conn.getResponseCode() != 200)
      {
          throw new IOException(conn.getResponseMessage());
      }

      // Buffer the result into a string
      BufferedReader rd = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
      StringBuilder sb = new StringBuilder();
      String line;
      while ((line = rd.readLine()) != null)
      {
            sb.append(line);
      }
      rd.close();

      conn.disconnect();
      return sb.toString();
}
```

Çizelge 3.5'de verilen kod çok basit düzeye indirgenerek verilmiştir. Daha kolay anlaşılması için normalde barındırması gerektiği try/catch/finally bloklarından arındırılmıştır. C# metotlarında olduğu gibi burada da parametrelerin kodlanmasına dikkat edilmesi gerekir.

Aşağıdaki kodda olduğu gibi Java programlama dilinde de HTTP POST istekleri için de URL kodlaması gerekmektedir.Çizelge 3.6'da http post örneği gösterilmiştir.

Çizelge 3.6 - JAVA ile HTTP POST Kullanımı [48]

```
public static String httpPost(String urlStr, String[]
paramName,String[] paramVal) throws Exception
{
      URL url = new URL(urlStr);
      HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
```

```
        conn.setRequestMethod("POST");
        conn.setDoOutput(true);
        conn.setDoInput(true);
        conn.setUseCaches(false);
        conn.setAllowUserInteraction(false);
        conn.setRequestProperty("Content-Type", "application/x-www-
form-urlencoded");

        // Create the form content
        OutputStream out = conn.getOutputStream();
        Writer writer = new OutputStreamWriter(out, "UTF-8");
        for (int i = 0; i < paramName.length; i++)
        {
              writer.write(paramName[i]);
              writer.write("=");
              writer.write(URLEncoder.encode(paramVal[i], "UTF-8"));
              writer.write("&");
        }
        writer.close();
        out.close();

        if (conn.getResponseCode() != 200)
        {
              throw new IOException(conn.getResponseMessage());
        }

        // Buffer the result into a string
        BufferedReader rd = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
        StringBuilder sb = new StringBuilder();
        String line;
        while ((line = rd.readLine()) != null)
        {
              sb.append(line);
        }
        rd.close();

        conn.disconnect();

        return sb.toString();
}
```

1.10.3 Javascript ile RESTful Kullanımı

Tıpkı diğer diller gibi JavaScriptler ile de RESTful sistemler geliştirmek mümkündür. Eğer oluşturulacak JavaScript sunucu tarafında çalıştırılacaksa, sunucuyu sağlayan firmanın HTTP istekleri için hazırladığı dokümantasyona uygun oluşturulmalıdır. AJAX uygulamaları geliştirmede tecrübeli bir yazılımcı için

Javascript kullanarak RESTful bir sistem geliştirmek oldukça kolaydır çünkü her AJAX isteği, bir HTTP isteğidir ve birçok açıdan AJAX uygulamaları RESTful'dur. Javascript programlama dilinde HTTP istekleri XMLHttpRequest objesini içerir. Burada objenin ismi aldatıcı olabilir çünkü ne istek ne de cevap XML içermek zorunda değildir. Ne yazık ki XMLHttpRequest objelerini yaratmanın standart bir yolu yoktur. Aşağıdaki metot bir çapraz-tarayıcı yolu ile çözüm sunmaktadır.

Bir XMLHttpRequest objesi hem GET hem de POST ile istek yollamaya izin verir fakat anında bir cevap döndürmez. Bu durumda isteğin bittiğinde başvurulan bir geri çağırma (callback) fonksiyonu tanımlanmalıdır. Sunucu-istemci etkileşimi sırasında geri çağırma fonksiyonu etkileşimin farklı aşamalarında birçok kez çağırılır. Tarayıcı programa da bağlı olan bu durumda, fonksiyon 4 defaya kadar çağırılabilir. Bu aşamada bizi ilgilendiren aşama son aşamadır. Verilen kod parçasında da olduğu gibi son aşamaya gelinip gelinmediğini anlamak için readyState alanı kullanılabilir. Çizelge 3.7 ve Çizelge 3.8'de javascript örnek kodları incelenebilir.

Çizelge 3.7 - Javascript - XMLHttpRequest [49]

```
function createRequest()
{
    var result = null;
    if (window.XMLHttpRequest)
    {
        // FireFox, Safari, etc.
        result = new XMLHttpRequest();
        if (typeof xmlhttp.overrideMimeType != 'undefined')
        {
            result.overrideMimeType('text/xml'); // Or anything
else
        }
    }
    else if (window.ActiveXObject)
    {
        // IE
        result = new ActiveXObject("Microsoft.XMLHTTP");
    }
    else
    {
        // No known mechanism -- consider aborting the application
    }

    return result;
}
```

Çizelge 3.8 - JavaScript için readyState [49]

```
var req = createRequest(); // defined above
// Create the callback:
req.onreadystatechange = function() {
if (req.readyState != 4) return; // Not there yet
if (req.status != 200)
{
    // Handle request failure here...
    return;
}
// Request successful, read the response
var resp = req.responseText;
```

Eğer dönen cevap bir XML cevabı ise (sunucu tarafındaki MIME type tex/xml ile belirtilir) bu cevap aynı zamanda responseXML özelliği ile okunabilir de. Bu özellik bir XML dokümanı içerir ve Javascript'in DOM(doküman obje modeli) yönlendirme hizmetleri gibi kullanılabilir.

XMLHttpRequest objesini ve geri çağırma fonksiyonunu hazırladıktan sonra istek fonksiyonu hazırlanabilir. Bu koda örnek Çizelge 3.9 ve Çizelge 3.10'da get ve post olarak gösterilmiştir.

Çizelge 3.9 - Javascript ile HTTP GET

```
req.open("GET", url, true);
req.send();
```

Çizelge 3.10 - Javascript ile HTTP POST

```
req.open("POST", url, true);
req.setRequestHeader("Content-Type", "application/x-www-form-
urlencoded");
req.send(form-encoded request body);
```

1.11 Gelecekte RESTful

Eleştirmenlere göre kullanıcılar REST sistemlerini kullanmaya başlayıp deneyim kazandıkça, REST servislerinin yaygın bir şekilde benimsenmesine en büyük engelin, bu servisler ile ilgili araç eksikliği olduğu ortaya çıkmıştır. Bugüne kadar büyük yazılım şirketlerinden hiçbirinin REST servisleri ile uygulama geliştirme girişimi olmamasına rağmen, bu servisleri ciddi şekilde takip ettikleri, BEA'nın

teknik direktörü David Orchard'ın "REST ile ilgili bazı karakteristik özellikleri inceliyoruz" sözleri ile anlaşılabilir. Buna rağmen REST'in büyük bir sıçrama yapmadan önce yinede araç geliştirme uzmanları tarafından benimsenmesine ihtiyacı vardır.

Zap Think firmasının üst düzey analistçisi olan Ronald Schmeltzer, "Bir ürün perspektifinden bakıldığında REST neredeyse görünmez" diyor ve ekliyor: "Eğer REST kullanıcıları bu teknolojiyi ayakta ve güncel tutmak istiyorlarsa, bu servisleri web servisleri oluşturan veya tüketen araçlar ile beraber kullanmak zorunda kalacaklar". Yazılım geliştirenler için REST kaynakları sunan bir web sitesinin sahibi ve bağımsız bir web mimarisi danışmanı olan Mark Baker, Schmeltzer'in görüşüne katılmıyor: "REST web servislerini geliştirmek için neredeyse her http uyumlu araç kullanılabilir. Şu anda birçok REST araç kiti mevcut, sadece insanların bunların ne olduğu bilmiyor" diyor. Baker'a göre sunucularda çalışan Java programlarını geliştirmekte kullanılan araçlar, REST tabanlı web servislerini geliştirmek için de kullanılabilir. Bu araçlar için Baker, "http tanımlarını takip ediyorlar ve bu sayede dolaylı olarak REST tasarımının sınırları içerisinde kalmış oluyorlar" ifadesinde bulunuyor [30].

Son olarak Orchard, REST'in bir gün SOAP ile beraber kullanılarak yazılımcılar için web servisleri tasarımları geliştirmeleri için yeni bir yol sunabileceğini düşünüyor ve ekliyor: "Bazen bir problemin çözümü için tek bir yol yeterli olmayabilir" [30].

Bu tez çalışmasında da az önce bahsedilen SOAP ve REST yazılım mimarilerinin birlikte tüketilmesiyle ile geliştirilecek olan yazılımın diğer uygulamalarla ne kadar rahat birlikte kullanılabilir olacağı savunulmaktadır.

Microsoft'un Visual Studio .NET veya IBM'in WebSphere gibi yazılım geliştirme programları uzun zamandır otomatik olarak SOAP tabanlı web servisleri oluşturabilmektedir.

.NET Framework 3.5 sürümü ile REST'e tam destek vermiştir. WCF ile REST servisleri yaratmak veya daha önceki sürümlerde de *.ashx HTTP Handler dosyaları kullanarak REST yaklaşımında uygulama geliştirmek mümkündür.

GEÇMİŞ E-SAĞLIK ÇALIŞMALARI

Geçmişten günümüze elektronik sağlık sistemlerinde yapılan akademik ve sektöre yönelik, özellikle ülkemizdeki piyasayı etkileyen ulusal ve uluslar arası sağlık uygulamaları bu kısımda incelenmiştir.

1.12 Sağlık-NET

Sağlık Bakanlığı'nın 58 ve 59. hükümet programlarında bulunan Acil Eylem Planında belirtilen ve kamu yönetimi reformu kapsamında "Sağlıklı Toplum" çerçevesi altında Sağlıkta Dönüşüm Programı başlatılmıştır. Bu programın amacı sağlık sektöründe hastaların hem daha çabuk tedavi edilebilmesi ve hastalıktan önce hastalığın önlenmesi, hem de tüm nüfusun eşit şartlarda sağlık hizmetlerinden yararlanabilmesidir. Ayrıca bu alanda hizmet veren kuruluşların da hizmet kalitesini arttırarak zaten sağlıklarında problem olan insanların daha uygun koşullarda bakımını üstlenmek hedeflenmektedir. Sağlık-NET kuruluşların belirli standartlara uygun elektronik sağlık uygulamalarıyla hem bu kuruluşların bünyelerinde yönetimi iyileştirecek hem de verdiği hizmetin daha verimli olmasını sağlayacaktır. Sağlıkta dönüşüm projeleri ile bilgisayar otomasyonları daha verimli kullanılarak sistem yönetimini geliştirilecek, buna bağlı olarak finansman ve hizmet sunumu iyileştirilecektir. Bunlarla beraber sağlık sigortası çatısı genişletilip finansal açıdan da devam ettirilebilir şekilde tüm ülkeye yayılacaktır. Sağlıkta Dönüşüm Programı sağlık alanında etkinlik, verimlilik ve hakkaniyet kavramları ele alınarak oluşturulmuştur. Bu kavramlar şöyle genişletilebilir;

- Halkın sağlık düzeyinin geçerli durumdan daha üst seviyeye taşınması, hastalıkların tedavisi ve önlenmesi
- Sağlık kuruluşlarının kaynakları en verimli şekilde kullanması;
    - İnsan kaynaklarının dağılımı
    - Malzeme yönetimi
    - Akılcı ilaç kullanımı
- Devlet kaynakları en verimli şekilde kullanması,

- Koruyucu hekimlik ile hastalıkların engellenmesi
- Aile hekimliği ile büyük hastanelerin gereksiz yere işgal edilmemesi
- Büyük hastaneleri yurdun çeşitli bölgelerinde yaygınlaştırmak

- SGK kapsamında tüm halkın eşit koşullarda sağlık hizmeti alması

Bahsi geçen bu kavramlar ile ilgili Sağlık Bakanlığı'nın vizyonu, ülkemiz genelinde sağlık alanında görevli tüm çalışanların katkısıyla ulusal sağlık bilgi sistemi oluşturmaktır. Bu bilgi sistemine yetki verilmiş kişi ve kuruluşların yanı sıra bütün vatandaşlar erişebilecektir. Böylece her bireyin doğumundan yaşamına tüm sağlık geçmişine internet aracılığı ile ulaşabilmesi hedeflenmektedir. Böylece kişilerde farkındalık yaratılmış olacak ve kendi sağlıklarını da daha iyi takip edeceklerdir. Bununla beraber sağlık kuruluşlarının da kişilere ait sağlık geçmişine belirli düzeyde erişimi ile hastane iş süreçleri verimliliği iyileştirilerek sağlık hizmet kalitesi artacak ve buna rağmen sağlık bakım maliyetleri düşecektir [31].

Sağlık-NET Sağlıkta Dönüşüm Program kapsamında Ulusal Sağlık Bilgi Sistemi olarak geçmektedir. Şekil 4.1'de ülkemizdeki sağlıkta elektronik dönüşüm projeleri gösterilmektedir.

Şekil 4.1 - Sağlıkta E-Dönüşüm [31]

Sağlık kuruluşlarının kullandığı yazılımlarla entegre olması zorunlu tutulmuş Sağlık-NET, bu kuruluşlarda oluşan her tip veriyi toplamayı, toplanan verilerden de uygun

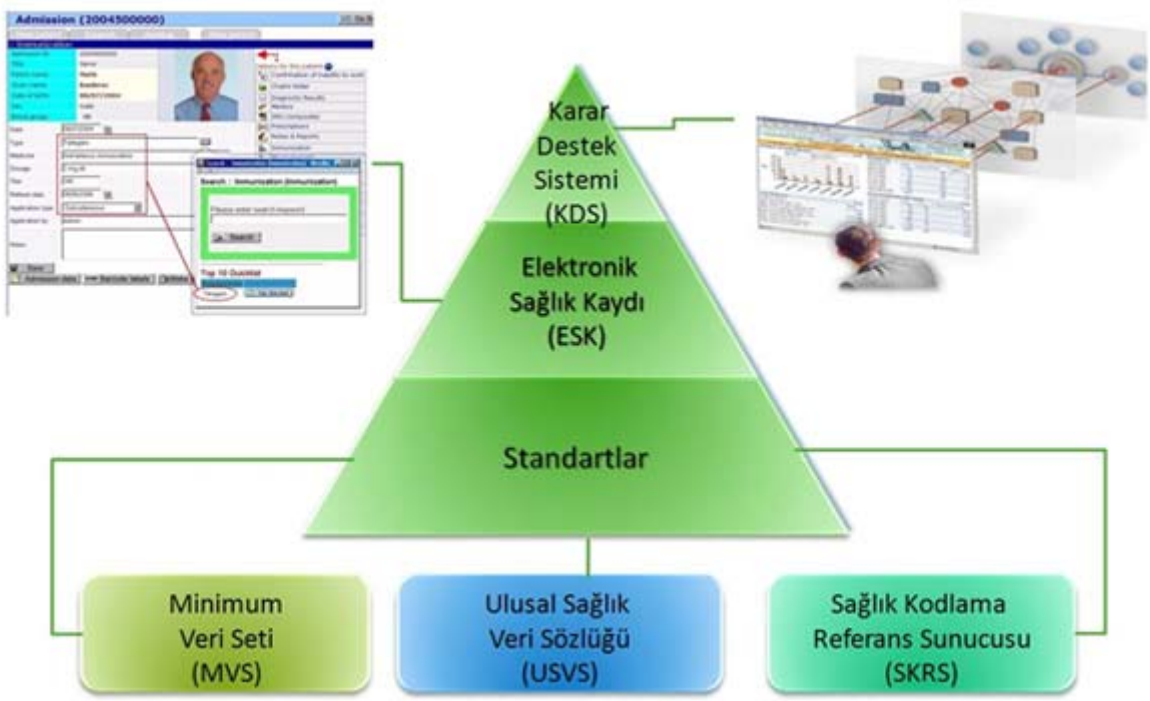
bilgilere ulaşmayı ve bu sayede sağlık hizmetlerinin kalitesini arttırmayı planlamaktadır. Tabii bu bütünlüğün sağlanabilmesi için Sağlık-NET'in kuruluşların kullandığı uygulamalarla entegre olması ve birlikte çalışabilmesi gerekmektedir. Bu derece geniş kapsamlı olan bir çalışmanın da güvenli ve hızlı tepki veren bir uygulama olması, veri yapılarının ve mesajlaşmanın da standartlara uygun bir şekilde sağlanması beklenmektedir.

Sağlık-NET sisteminde günümüz yazılım teknolojileri ve network altyapısı düşünülerek tek bir ortak veri tabanı üzerinde veri saklamak planlanmıştır. Hastaların vatandaşlık numarası gibi tekil bir referansı olması, farklı uygulamalar tarafından hasta bilgilerine erişmede veya veri eşleştirmede kullanılacak bir değer olacaktır. Sağlık-NET'in özellikler aşağıdaki gibi sıralanabilir;

- Sağlık sektöründe hizmet veren aile hekimliği, hastane bilgi sistemi ve bunlara benzer farklı uygulamalardan standart veri alışverişi yapabilen ve aynı zamanda bu verileri ortak erişimli bir veritabanında saklayabilen bir altyapı sağlamak
- Ülke genelinde barındırdığı verilerden oluşan veri ambarı ile hastalık istatistikleri, sağlık harcamaları, bölgesel sağlık analizleri, bulaşıcı hastalıklar vb. kritik bilgilere ulaşarak sağlık kuruluşlarına ve ulusal yönetime karar destek mekanizması oluşturmak
- Karar destek mekanizması sayesinde sağlık alanında tehdit oluşturacak (bulaşıcı hastalık, sağlık harcamalarında patlama vb.) unsurlarda erken uyarı verebilmek ve hatta önleyici bir mekanizma oluşturmak
- Uluslararası örgütlerin (EUROSTAT, WHO, OECD) talep ettiği istatistiksel verileri sağlamak
- Vatandaşların kendi sağlık geçmişlerini bilgisayar ortamından takip edebilmesini sağlamak [32]

Sağlık-NET'in veri yapılarına ve bu verilerin transferlerine yönelik standartlaşma çalışması üç ana gruba bölünmüştür, Ulusal Sağlık Veri Sözlüğü – USVS, Minimum

Veri Setleri – MSVS, Sağlık Kodlama Referans Sunucusu – SKRS. Ortak veri tabanında tutulan içeriğe Elektronik Sağlık Kaydı – ESR ve bu veri tabanından çıkacak analiz çalışmalarını sağlayan göstergelere Karar Destek Sistemi bileşeni olarak isimlendirme yapılmıştır. Aşağıda Sağlık-NET'in bileşenleri gösterilmiştir.



Şekil 4.2 - Sağlık-NET Bileşenleri [32]

Şekil 4.2'da gösterilen Sağlık-NET bileşenleri aşağıdaki detaylı açıklanmaktadır.
Sağlık Kodlama Referans Sunucusu – SKRS, Sağlık-NET ile entegre olacak olan bilgi sistemlerinin ihtiyaç duyduğu verilerin standart bir biçimde, XML formatında ulaşılmasını sağlayacak platformdur. SKRS kapsamında aşağıdaki bileşenler mevcuttur [33];

- Tanı Sınıflama Sistemi (ICD–10)
- Tanı Sınıflama Sistemi (ICPC2)
- İlaç ve İlaç Sınıfları Kodlama Sistemi (ATC)
- Sağlık Uygulama Tebliği (SUT) Kodları
- Klinik Kodları

- Branş Kodları
- Sağlık Kurumu Kodları
- Adres Kodları
- Aşı Listesi
- Aşı Takvimi Değerleri Listesi
- Meslek Grupları Listesi
- Parametreler Listesi
- Bebek İzlem Listesi
- Gebe İzlem Listesi
- Çocuk İzlem Listesi
- Persentil Değerleri Listesi
- Olası Tanı Kriterleri
- Enfeksiyon Etkenleri Tanı Kriterleri
- Tümör Yerleri
- Kesin Tanı Kriterleri
- Histoloji Kodları
- Doktor Bilgi Bankası
- Sözlük Veri Kapsamı Alan Kodları

E-Sağlık uygulamalarının bünyelerine kaydedeceği bu standart verilerin SKRS'den çekildiğinde cevap dönen kodlarını Sağlık-NET'e veri kaydederken göndermeleri gerekmektedir. Bu sayede Sağlık-NET standartlaşmayı sağlayabilmektedir. Örneğin bir hastane hastasına çıkmış olduğu ilaç bilgisini Sağlık-NET'e kaydedeceği zaman İlaç verisinin SKRS'de kayıtlı olan sistem koduyla (ilaç sistem kodu: AF7BB2C3-3AEF-433A-BD0A-EA7416D3D586) beraber ilacın adını ve kodunu göndermelidir. Şekil 4.3'da örnek SKRS veri elemanı görüntülenmektedir.

İlaç

Metadata Türü	: VERİ ELEMANI	*İdari Durumu*	: Kullanımda
Veri Elemanı No.	: 111	*Oluşturulma Tarihi*	: 01.06.2007
Sürüm No.	: 1	*Sürüm Tarihi*	: 01.06.2007
Kayıt Otoritesi	: Sağlık Veri Standartları Geliştirme Komisyonu	*Kaldırılma Tarihi*	:
Kaynak Organizasyon	: İlaç Eczacılık Genel Müdürlüğü (İEGM)		
Kaynak Doküman	:		

Tanımlayıcı ve niteleyici özellikler

Tanımı : Hastalıkların teşhis ve tedavileri, hastaların yakınmalarını hafifletme, hastalıklardan korunma ya da fizyolojik etkinlikleri düzeltmek amacıyla kullanılan her türlü kimyasal bileşim ilaç olarak değerlendirilir.

Bu amaçların dışında kullanılan ve alışkanlık yapan bazı maddeler, ilaç kapsamına girmezler. Aşılar, serumlar ve teşhis ajanları ise ilaç kapsamına girerler.

Bağlamı : Epidemiyolojik araştırmalarda, planlamada, maliyet analizinde, standart tanı tedavi kılavuzlarında kullanılır.

Ek Değerlendirmeler :

İlişkisel ve gösterimsel özellikler

Veri Tipi : Alfanumerik *Alan Büyüklüğü* : 14 *Format* : A(14)

SKRS Sistem Kodu : AF7BB2C3-3AEF-433A-BD0A-EA7416D3D586

Doğrulama Kuralları :

Kullanım Klavuzu : Bir muayene işleminde bir veya birden fazla ilaç kaydı yapılabilir. Ancak, "komplikasyon yok" seçilmiş ise tek seçim yapılmış olmalıdır.

Toplama Metodları :

İlişkili Veri Elemanları : Majistral İlaç Açıklama, İlaç Adedi

Bulunduğu Veri Setleri :

Adı	*Zorunluluk*	*Tekrar*
Reçete MSVS	Zorunlu	1+
HIV İzlem MSVS	Seçimli	1+

Veri Kapsamı : *Alan Adı* *Kodu*

Sağlık Kodlama Referans Sunucusu'nda yer alan İLAÇLAR tablosundan seçilmelidir.

Şekil 4.3 - Örnek SKRS Elemanı [33]

Ulusal Sağlık Veri Sözlüğü ve Minimum Sağlık Veri Seti (USVS, MSVS) tanımları da aşağıdaki gibidir.

USVS Türkiye'de kullanılan e-sağlık uygulamalarında verilerin yapısının nasıl bir standarda bağlanması gerektiğini gösteren bir sözlük çalışmasıdır. Sözlük kapsamında veri elemanlarının tanımı mantıksal açıklamalarla beraber bu verilerin ne anlama geldiği de vurgulanmıştır. Toplamda 10 ayrı küme altında 46 adet veri seti ve 261 adet veri elemanı bulunmaktadır. Bir veri elemanı birden fazla veri seti içerisinde yer alabilir [34]. Örnek veri elemanı Şekil 4.4'de sunulmuştur.

Meslek

Metadata Türü	Veri Elemanı	**Idari Durumu**	Kullanımda
Veri Elemanı No	59	**Oluşturulma Tarihi**	01.06.2007
Kayıt Otoritesi	Sağlık Veri Standartları Geliştirme Komisyonu	**Sürüm Tarihi**	01.06.2007
Kaynak Organizasyon		**Kaynak Döküman**	

Tanımlayıcı ve niteleyici özellikler

Tanımı	Kişinin meşgul olduğu işi veya görevi ifade eder.
Bağlamı	Kişinin mensubu olduğu meslek ile taşıdığı hastalık arası ilişkilerin değerlendirilmesinde kullanılır. Kişinin mesleği ile ilişkili olabilecek iş kazaları ve olası hastalıkların tespitinde kullanılır. Meslek bilgisinin doğru girilmesi, hastalık tanısının belirlenmesinde de yardımcı olmaktadır.
Ek Değerlendirmeler	

İlişkisel ve gösterimsel özellikler

Veri Tipi	Numerik	**Alan Büyüklüğü**	6	**Format**	N(6)
SKRS Kodu	512d0cb3-d0b3-487c-ab1e-1343fc7ff611				
Doğrulama Kuralları					
Kullanım Klavuzu	Bu veri, 0-15 yaş aralığında olan kişiler için gerekli değildir. Sağlık kurumları kendi yerel ihtiyaçlarını göz önüne alarak, Sağlık Kodlama Referans Sunucusu'nda yer alan MESLEKLER tablosundan seçilmiş özel bir tablo oluşturup, kuruma veya hekime özel hızlı listeler kullanabilirler.				
Toplama Metodları	Sağlık Kurumları kendi yerel ihtiyaçlarını gözönüne alarak, Sağlık Kodlama Referans Sunucusunda yer alan MESLEKLER tablosundan seçilmiş özel bir tablo oluşturup, kuruma veya hekime özel hızlı listeler kullanabilirler.				
İlişkili Veri Elemanları					

Bulunduğu Veri Setleri

Adı	Zorunluluk	Tekrar
► Hasta Özlük Bilgileri MSVS	Koşullu	Hayır
► HIV Tespit MSVS	Koşullu	Hayır

Veri Kapsamı

Alan Adı	Kodu
Sağlık Kodlama Referans Sunucusu'nda yer alan MESLEKLER tablosundan seçilmelidir.	

Şekil 4.4 - Örnek veri elemanı – Hasta özlük bilgileri veri setinden Meslek [34]

MSVS ise bunca veri içerisinden sadece Sağlık Bakanlığının toplanmasında zorunlu tuttuğu verileri ifade etmektedir. Böylece bu zamana kadar kağıt üzerinde toplanan

verileri bilgisayar ortamında toplanmaya başlayacaktır. Örnekte Hasta Kabul MSVS'sinde (Şekil 4.5) zorunlu veri elemanları ve bunların tekrar bilgileri mevcuttur.

Hasta Kabul MSVS

Sürüm No	1.0	**Oluşturulma Tarihi**	01.06.2007
İdari Durumu	Kullanımda	**Sürüm Tarihi**	01.06.2007

Tanımlayıcı ve niteleyici özellikler

Kapsamı	Kabul, sağlık kurumu veya aile hekimi tarafından hastanın tedavi ve bakım sorumluluğunun üstlenilmesi olayını ifade eder. Kabulden sonra hastane ya da aile hekimi tarafından hastanın ihtiyaç duyduğu sağlık hizmetinin verilmesine başlanır. Bu veri seti, hastanın sağlık kurumuna veya aile hekimine kabul işlemlerine ait verileri içerir.
Bağlamı	Hastanın sağlık hizmeti almaya başlaması sırasında alınan bilgiler, verilen sağlık hizmetinin, kurumlar arası ve kurum için sevklerin analizinde ve hizmet planlamasında kullanılan önemli bilgiler içermektedir.
Toplama Metodları	Hasta Kabul MSVS; hastanın kabulünden sonra aldığı sağlık hizmetleri (muayene, ağız diş sağlığı hizmetleri vb.) ve hasta çıkış işlemleri ile ilişkilendirilir ve ilgili olan veri setleri ile birlikte paket olarak bildirilir.

İlişkisel ve gösterimsel özellikler

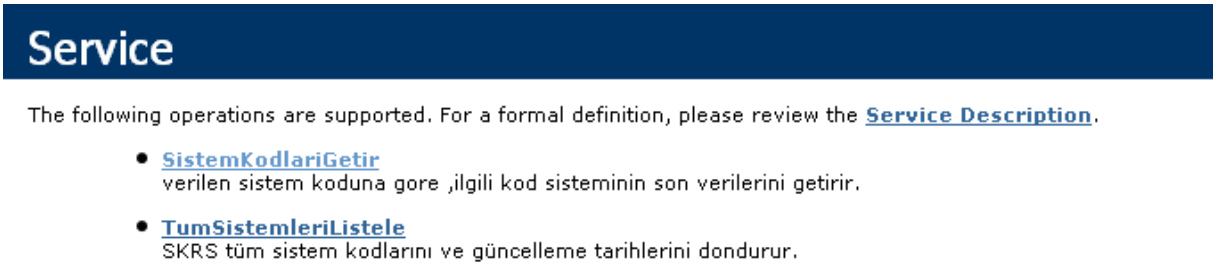
Kaynak Organizasyon	Strateji Geliştirme Başkanlığı (SGB)
Kayıt Otoritesi	Tedavi Hizmetleri Genel Müdürlüğü (THGM)
Ulusal Raporlama Düzenlemesi	Internet üzerinden Sağlık-NET portali aracılığıyla veya doğrudan uzak uygulamalar tarafından gönderilen elektronik mesajlarla HL7 V3 protokolü kullanılarak Ulusal Sağlık Bilgi Sistemi'ne bildirilir.

Veri Seti Elemanları

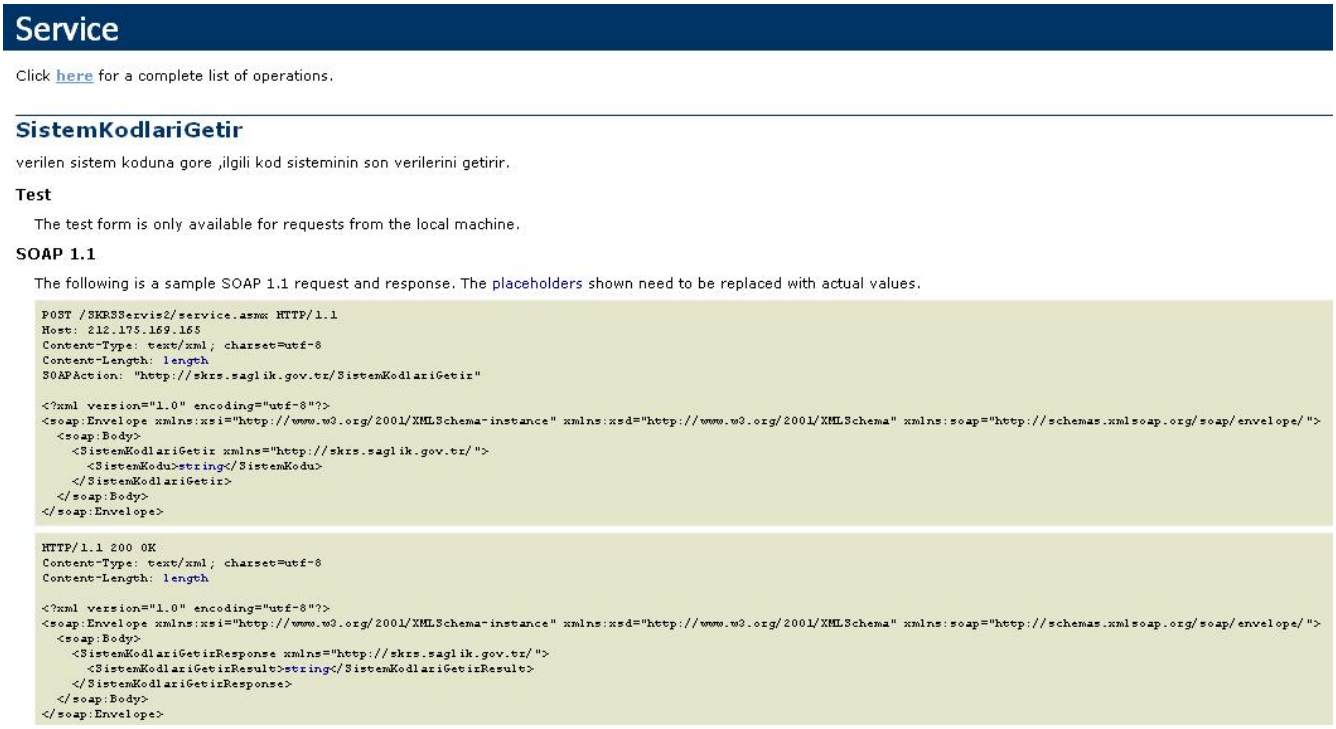
Veri Seti Elemanı Adı	Tekrar	Zorunluluk
➤ Kurum	1	Zorunlu
➤ Kabul Zamanı	1	Zorunlu
➤ Kabul Şekli	1	Zorunlu
➤ Geldiği (Poli)klinik	1	Koşullu
➤ Sevk Tanısı	1+	Koşullu
➤ Sevk Tarihi	1	Koşullu
➤ Vaka Türü	1	Zorunlu
➤ SGK Takip Numarası	1	Seçimli

Şekil 4.5 - Örnek MSVS - Hasta Kabul [34]

Sağlık-NET verileri bağladığı bu standartlar sayesinde ülkede hizmet veren e-sağlık uygulamalarıyla HL7 mesajlaşma standardı ile haberleşerek verileri depolamayı hedeflemektedir. Diğer uygulamalarla birlikte çalışabilmek için web servis teknolojisinden yararlanmıştır. Yayınladığı web servislerinden hangisinin ne zaman kullanılması gerektiği Sağlık-NET Entegrasyon Kılavuzunda detaylı şekilde açıklanmıştır [35].



Şekil 4.6 - Örnek Sağlık-NET Web Servisleri



Şekil 4.7 - Örnek SKRS Web Servisi, Sitem kodlarını getirme isteği ve cevabı

1.13 Artemis

Artemis projesinin başlangıç sebeplerinden temeli, sağlık sektöründe otomasyon uygulamalarının birlikte çalışabilirliğinin olmamasıdır. Hasta kayıtlarının bir uygulamadan diğerine transferinin mevcut olmaması, sağlık bilgilerinin standartlara uygun biçimde paylaşılamaması, bilgi sistemlerinin geliştiği bir dünyada büyük bir eksikliktir. Artemis projesinin amacı kuruluşlar arasında bu bilgilerin dünyada kabul

görmüş standartlar eşliğinde uygulamalar arasında transfer edilebilmesidir. Böylelikle sağlık personelinin hasta verilerine sadece kendi kullandığı otomasyon bünyesinde değil, hastanın geçmişinde gittiği kuruluşun otomasyonundan da bilgi edinebilmesi sağlanacaktır. Artemis, uygulamaların web servisler kullanarak P2P şekilde bağlantı sağlayarak bir platform kurmayı hedeflemiş bir projedir.

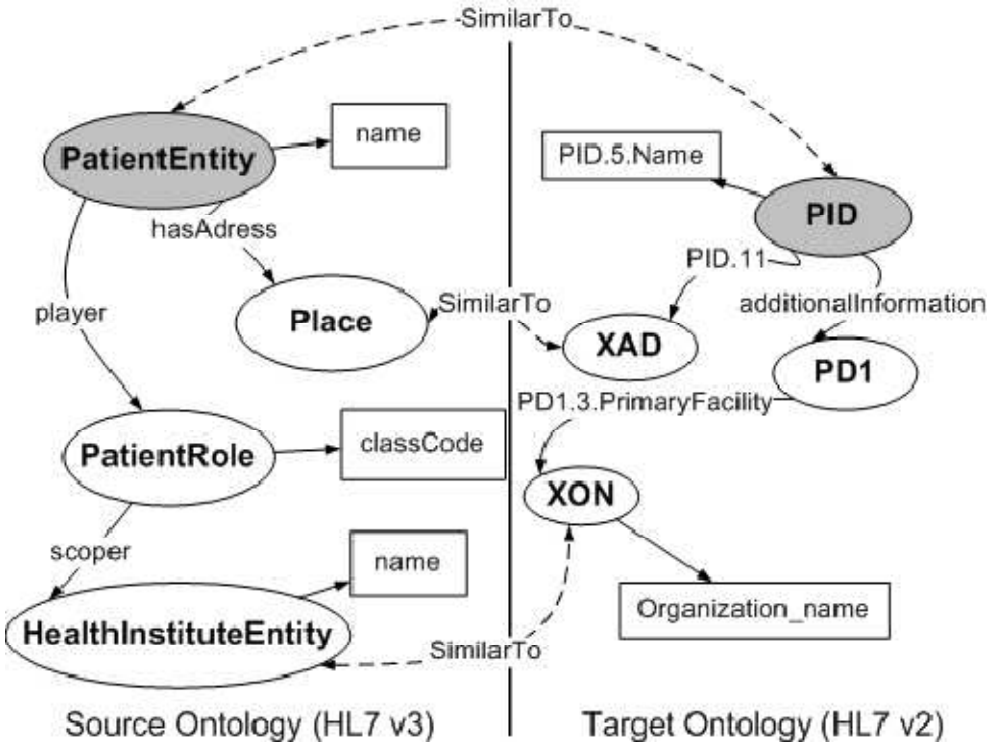
Artemis, Avrupa Komisyonu 6.çerçeve programı tarafından desteklenmektedir. Ülkemizden ODTU Yazılım AR-GE Ünitesi ana yüklenici ve proje koordinatörü olarak çalışmalara dahil olmuştur. Sağlık uygulamaları geliştiren firmalardan ise yine ülkemizden Tepe Teknoloji Corttex ismini verdiği sağlık otomasyon uygulamasını bu proje ile diğer uygulamalar ile birlikte çalışabilir hale getirmek için çalışmaktadır. Projeye dahil olan diğer ülkeler ve şirketler; Yunanistan'dan ALTEC, Almanya'dan Kuratorium Offis, İngiltere'den University of Southampton, IT Innovation Center ve South and East Belfast Health and Social Services Trust'dır. Artemis projesi 1 Ocak 2004'te başlamıştır ve 30 ay sürmesi planlanmaktadır. Proje için gereken toplam bütçe 2,957,604.00 Euro olup, Avrupa Komisyonundan destek olarak 1,989,000.00 Euro alınmıştır [36].

Artemis projesinde HL7, GEHR gibi sağlık bilgi sistemlerinde belirli bir düzeni yakalamak için kullanılan ve geliştirilmesine de devam edilen standartlar kullanılmaktadır. Fakat Artemis sadece bu standartların yeterli olmayacağını çünkü uygulama geliştiren firmaların kendilerine değişik standartları baz alabileceğini vurguluyor. Böylece iki program birbirlerine p2p seviyesinde direkt bağlı olsa dahi farklı standartlar kullanıyorlarsa birlikte çalışabilirliği zor sağlanacaktır. Çünkü bu standartlardan bazıları mesajlaşmaya yoğunlaşmışken diğerleri elektronik sağlık kayıtlarının yapısıyla ilgilenmektedir. Artemis, sağlık uygulamalarının birbirleriyle haberleşebilmeleri için web servis kullanımını yaygınlaştırmak gerektiğini savunmaktadır. Uygulamalar birbirlerine kendilerinden alınacak hizmetlerin web servislerini paylaşmalı, böylece birlikte kullanılabilirlik kavramı daha gerçekçi bir boyuta yükselecektir. Yazılım geliştiren firmalar birbirlerinin alternatifi olan sağlık standartlarını kullansa dahi web servisleri sayesinde gerek veri paylaşımı gerekse diğer uygulamadan hizmet alımı hızlı ve doğru şekilde yapılabilecektir.

Artemis yukarıda bahsi geçen teknoloji standartlarının yanı sıra verilere anlam yükleyerek uygulamalar arasında sadece insanların taleplerine göre hizmet veya veri paylaşımından çok, artık bilgisayarların kendi karar verebilecekleri girdi ve çıktıları sağlayan bir arabulucu uygulama çerçevesi olmayı planlamaktadır.

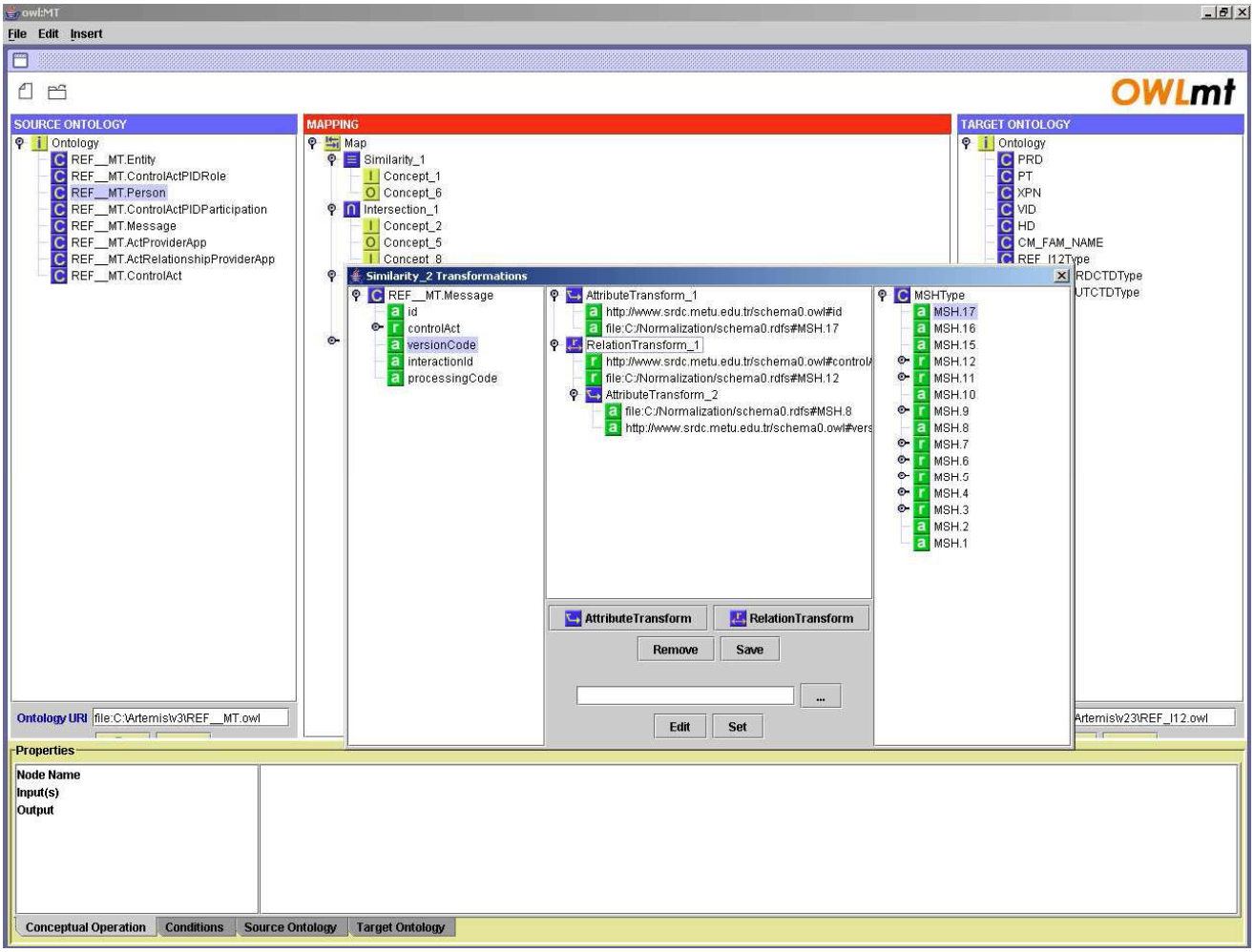
Bunu başarabilmek için semantik (anlamsal) web teknolojilerinden yararlanılmaktadır. Artemis mesajlara anlam kazandırmak için kendi yapısını kurmuştur ve Artemis Message Exchange Framework adını vermiştir. Bu çerçevede Artemis takımı mesajları nasıl OWL'ye (Web Ontology Language) çevireceklerini araştırmış ve neticesinde de OWLmt – OWL mapping tool ismi verilen bir araç geliştirmiştir. OWLms OWL-QL – OWL Query Language engine yardımı ile çalışmaktadır [36][37].

Artemis'te prototip olarak HL7 v2 mesajları HL7 v3 mesajlarıyla anlamsal olarak eşleştirilmiştir. Tabii projede sadece spesifik olarak HL7 mesajları değil, kullanılan standart ne olursa olsun her biri için verilere anlam OWL temsilleri eklenerek kaynak ve hedef veriler birbirleriyle anlamsal bağ oluşturabilecek halde geliyor. Sonuçta veri tiplerinin ne olduğunu bilmeden, dinamik bir biçimde anlamlara bakarak istemci uygulamanın girdilerini karşılık veren uygulamanın çıktıları ile bütünleşebilir hale getiriliyor (Şekil 4.8).



Şekil 4.8 - Nesneler arası anlam eşleştirme[37]

Artemis projesinde geliştirilen OWLmt ara katman olarak OWL-QL motorunu kullanarak farklı veri setlerini yükleyerek anlamsal eşleştirme yapabilmektedir. Kaynak ve hedef olarak aldığı girdileri eşleştiren örnek Şekil 4.9'da gösterilmiştir [38].



Şekil 4.9 - OWLmt Arayüzü [38]

1.14 Google ve Microsoft E-Sağlık Çözümleri

Günümüzdeki sağlık sistemlerinin karmaşıklığı ve kullanışsızlığı yüzünden, sağlık bilgileri çoğunlukla kağıt üzerinde dağınık ve kopuk bir şekilde saklanıyor. Dahası bir hastanın bir ya da birden fazla sağlık kuruluşundan, birçok doktor ve klinikle ilişkili sağlık kayıtları bulunabiliyor. Bu gibi standart olmayan ve istenilen zamanda istenilen yerden ulaşılamayan sağlık bilgilerinin işe yaramaz olması büyük yazılım firmalarına da piyasada yer açmış bulunmaktadır.

Bilgileri merkezi bir yerde, saklamak ve paylaşmak, kolay bir şekilde güncelleme yapmak için sağlık bilgi sistemine ihtiyaç duyulması sonucu Microsoft ve Google,

Sağlık sektöründe çalışan ve faaliyet gösteren kişiler ile son kullanıcıların rahat bir şekilde kullanabilecekleri sağlık arama motorları ve uygulamaları geliştirmektedir.

Microsoft, kişilerin kendi sağlık durumları üzerinde daha fazla kontrole sahip olması için HealthVault'u geliştirdi. HealthVault doktor, hastane, işveren, eczaneler, sigorta sağlayıcıları ve sağlık cihaz üreticileri ile birlikte çalışabilmektedir.

HealthVault, web tabanlı olup kullanıcıların sağlık kayıtlarını (örneğin; tansiyon, kollestrol seviyeleri, cerrahi durumları vb.) çevrimiçi (online) olarak saklayabildiği ve paylaşabildiği bir uygulamadır. HealthVault hesabı açmak basit elektronik posta hesabı açar gibi kolay bir sürece dahildir. Kullanıcılar tek bir hesapta sadece kendi verilerini tutabileceği gibi ailelerinin de sağlık kayıtlarını saklayabilmekteler. HealthVault kullanıcıları hastaneye gitmeden sağlayabildikleri sağlık verilerini de düzenli olarak sisteme girerek elektronik sağlık kayıtlarının bütünlüğünü koruyabilmektedir. Böylece bu kayıtları takip eden sağlık personeli için hem zaman kaybı olmamakta hem de karar verme süreci iyileşmektedir [39].

Microsoft'un diğer bir ürünü Amalga Hastane Bilgi Sistemi 2009 ise sağlık kuruluşlarına entegre olabilen, bütün hastane fonksiyonlarını yönetebilen, operasyonel verimliliği arttıran, hasta bilgilerini kolay takip etmeyi sağlayan, basit mimari ve kod bazlı, web tabanlı bir otomasyondur. Web tabanlı olmasının verdiği kolaylıkla versiyon güncellemelerini dinamik olarak merkezi sistemden dağıtabilmektedir. HL7 uyumlu olduğu için diğer üçüncü parti uygulamalarla sağlık kayıtlarını paylaşabilir [40].

Google'ın sağladığı sağlık çözümü ise Microsoft'un HealthVault'una benzer bir şekilde çalışmaktadır. Merkezi bir ortamda sağlık bilgilerinin depolanmasını ve yönetilmesini sağlayan Google Health, diğer çevrimiçi uygulamalarında da olduğu gibi sisteme ücretsiz olarak alınan kullanıcı adı ve şifre kullanılarak giriş yapılmasına olanak sağlar. Elektronik posta, takvim, dokümanlar gibi uygulamalar ile aynı platformdadır.

Google sağlık kullanıcılara; hastaneler, klinikler, eczaneler, sigorta şirketlerindeki sağlık kayıtları ile bunlara bağlı bilgilere ulaşma imkanı sağlamaktadır. Kullanıcıların sağlık profilinde, reçete bilgileri, alerjileri, geçirdiği hastalıklar ve tedavi geçmişindeki diğer bilgiler yer almaktadır. Bu sağlık kayıtları kullanıcıların isteğine göre doktorlarla veya diğer kullanıcılarla istendiği zaman paylaşılabilmektedir [41].

Hem Google hem de Microsoft tüm dünya kullanıcılarına daha rahat ulaşabilmek için öncelikle kişisel sağlık kayıtlarını takip etmeye yarayan web tabanlı uygulamalar üzerinde durmaktadırlar. Şimdilik sadece Microsoft Amalga ile hastane bilgi sistemleri üzerine yoğunlaşmıştır. Fakat paket program olarak sunulan bu hizmet ülkelerin değişik standartlarına ve süreçlerine uyum sağlamada ne kadar başarılı olabilir tartışılır. Diğer uygulamalarla birlikte çalışabilirliği açısından düşünülecek olursa veri alıp vermek için HL7 standardı kullanılmaktadır fakat dışarıya açılan web servislerinin ne kadar esnek olduğu konusunda bir bilgi bulunmamaktadır.

Sağlık sektöründeki bu büyük yazılım açığı dev bilgisayar firmaları tarafından bile büyük bir pazar rekabeti oluşturmaktadır.

RESTFUL WEB SERVİSLERİ İLE GERÇEKLEŞTİRİLEN BİR E-SAĞLIK UYGULAMASI

Bu bölümde bir hastane bilgi sisteminde olması gereken özellikler ve bu özelliklere bağlı olarak basit bir uygulama gerçekleştirimi olacaktır. Bu uygulama web tabanlı olup, çalışmanın önceki bölümlerinde ifade edilen yazılım teknolojilerine, standartlarına ve yaklaşımlarına uygun olarak geliştirilecektir.

1.15 Gereksinimler

Çalışmanın bu aşamasında ülkemizdeki hastanelerin standart bir işleyişe dahil olabilmeleri için ihtiyaç duyduğu özellikler listelenecektir. Günümüz hastane bilgi sistemlerinden beklenilen faydalar aşağıdaki gibi sıralanabilir;

- Hastane otomasyonunu kullanacak olan kullanıcıların yönetilebileceği, bu kullanıcıların belirli rollerle ilişkilendirilip, rollerin de hangi modüllerle ilişkili olup hangi özellikleri kullanıp kullanamayacaklarının ayarlanabildiği bir güvelik ve kullanıcı yönetimi modülü olmalı. (Authorization ve authentication)
- Hastane otomasyonunda kullanılacak olan nesnelerin (örneğin işlem tanımları, birim tanımları, hangi birim servis hangi birim poliklinik vb.) yönetilebileceği bir modül olmalı ve sadece salık kuruluşunun yetkili personelleri ile ilişkilendirilmeli.
- Müracaat eden her hastanın genel kimlik bilgileri, adres bilgileri, hasta kurum hastası ise hastaya ait sosyal güvenlik ve sevk bilgileri sisteme girilmelidir.
- Türkiye için, Mernis ile entegre olunmalı ve sistemde kayıtlı olmayan hastaların bilgileri uzun uzun girilmesi yerine TC Kimlik No ile Mernis'ten hasta bilgilerine ulaşılmalıdır.
- Bulunulan ülkeye ve sigorta tiplerine göre hak sahipliği sorgulamaya yönelik web servislere erişilebilinmelidir. Ülkemizde Sosyal Güvenlik Kurumu (SSK, Bağ-kur, Emekli Sandığı Emeklileri), Emekli Sandığı Çalışan ve Yeşilkart hastaları için hak sahipliği sorgulama web servisleri mevcuttur.

- Yine ülkeye göre adres bilgi sistemi ve kimlik numarası doğrulama ve sorgulama sistemlerine entegre olmalıdır.
- Hastalara ait özlük bilgilerinin mükerrer kaydı engellenmelidir.
- Hasta kaydı esnasında hastanın resmi çekilerek sisteme aktarılabilmelidir.
- Hasta kayıtlarına ilişkin barkod basılabilmelidir.
- Hastanın hastaneye her gelişinde hangi polikliniğe veya servise gideceğini gösteren bir barkod olmalıdır. Bu barkoda veya çıktıda mutlaka hastanın gittiği birime ait sıra numarası bulunmalıdır.
- Hastaların muayenesi, tedavisi vb. tüm tıbbi süreçler ile ilişkili veriler en ufak ayrıntısına kadar hastane bilgisayar otomasyonu bünyesinde kayıt altına alınmalıdır.
- Hasta arama fonksiyonları ile hastalar arasından çok farklı kriterlerde aramalar yapılarak hasta kayıtlarına ulaşılabilir. Örneğin belirli tarihlerde Dahiliye birimine gelen ve şeker hastalığı teşhisi konan hastalar vb.
- Hastalar internet ve telefon aracılığıyla gitmek istedikleri birimlere ve doktorlara kolaylıkla randevu alabilmelidir. Alınan randevuların takibi de hastane personeli tarafından otomasyondan yapılabileceği gibi yine internet ve telefon üzerinden de rahatlıkla yapılabilmelidir.
- Polikliniklerde hastalarının sıra takipleri için LCD veya Plazma TV gibi ekranlara sıradaki hastanın hangi poliklinikte hangi doktora gittiği listelenmelidir. Bu yapıda hastanın adı yerine sıra numarası kullanılması hastanın özel haklarının korunması açısından daha verimli olabilir.
- Hastanın yaptıracağı Radyolojik ve ileri düzey tetkikler için randevu verilebilmelidir. Bu randevuların takibi sağlık ekibi tarafından yapılabileceği gibi hastaya da randevusuyla ilgili çıktı verilmeli ve yine hasta gerekirse internetten hangi işlemi ne zaman yaptıracağını izleyebilmelidir.
- Görüntülü radyolojik işlemler için eğer hastanenin PACS cihazları varsa, kaydedilen hareketli görüntülere ihtiyaç anında otomasyondan ulaşılabilmeli ve görüntülenebilmelidir.
- Hastalar ayrıca tahlil sonuçlarına internet üzerinden ulaşabilmelidir.

- Sağlık personeline tedavi esnasında hastaların hastalık geçmişlerine göre uyarılar sistem tarafından verilmelidir. Örneğin Diyabet, astım ve benzeri hastalıklara sahip hastalar, ya da hastanın alerjik durumu için uyarılar verilmesi.
- Hasta başı monitörlerle hastanın tedavi süreci izlenebilmelidir. Yatan hasta servislerinde doktorlar veya asistanlar ellerinde gereksiz evraklardan kurtulmalıdır.
- İlaç etkileşimi uyarılarına sahip olmalıdır. Eğer hastaya uygulanan birden fazla ilaç varsa ve birbirleriyle değişik etkileşime girebiliyorlarsa, hastaya ilaç çıkılmadan bu ilaçlar konusunda sağlık personeli uyarılmalıdır.
- Hastaya çıkılan ilaç eğer birimin deposunda gözükmüyorsa, hastane içinde farklı bir birimin deposundan ya da hastanenin eczanesinden istek yapılabilmelidir. Eğer ilaca hiç ulaşılamıyorsa o ilacın muadili olan bir ilaç sistem tarafından sağlık personeline gösterilmelidir.
- Entegrasyon veya başka uygulamalarla haberleşmek, birlikte çalışabilirlik gerektiğinde değişikliklere gerek kalmaksızın ya da minimum değişiklik yaparak yeni koşullara adapte olabilmelidir.
- Ülkemiz için Sağlık-NET sistemi ile entegre olarak çalışmalı sağlık veri setlerini web servisler üzerinden merkeze iletmelidir. Bu gereksinim aynı zamanda devlet tarafından getirilen bir zorunluluktur.
- Yine ülkemizde geçerli olan MKYS ve MEDULA servislerine entegre olmalıdır.
- Tıbbi cihaz ve malzemeler barkodlanarak stokları takip edilebilmelidir.
- Hastane yönetiminin ihtiyaç duyabileceği, devletin zorunlu kıldığı ya da tıbbi sürelerle ilgili detaylı istatistikleri kısa sürede verebilmelidir.
- Faturalama süreçleri sorunsuz şekilde işlemelidir ki bu da ancak hastaların kabul alasından taburcu olmasına kadar olan süreçteki tüm verilerin eksiksiz ve hatasız girilmesiyle mümkün olmaktadır.
- Resmi kurumlara iletilecek raporlar düzenli bir şekilde verilebilmelidir. Hastane personeli neyin ne zaman olduğunun takvimini otomasyondan izleyebilmelidir.

- Hastane bilgi sistemi eğer idari süreçleri de bünyesinde barındıracaksa muhasebe, vezne ve personel takip gibi faklı modüller içermelidir.
- Personel takip modülünde personel özlük bilgilerinin kayıtları mükerrer olmamak koşulu ile saklanmalıdır. Bu personeller gereksinim listesinin başında belirtilen kullanıcı yönetimi modülü ile de ilişkili olabilir (yani personel otomasyonu kullananlardan biri olabilir).
- Personellerin işe başlama bilgileri ve sözleşme detayı, görev bilgileri ve hastanedeki pozisyonları, şuan ki çalışma durumu, izin bilgileri, maaş bilgileri ve bordro kayıtları, baktığı hastalara ve yaptığı tetkik/ameliyatlara göre performans ek ödeme hesapları gibi kritik bilgilerin oluşturulması ve kayıt altında tutulması gerekmektedir.
- Hastanede bulunan demirbaşların takip edilmesi bir ihtiyaçtır.
- Demirbaşlar gibi tıbbi cihazlarında kayıt altında olması gerekir. Bu cihazlarının nerede tutulduğu, ne zaman bakımının yapıldığı vb. gibi bilgilerin sistem bünyesinden eklenebilmesi ve gerektiğinde geçmişe yönelik sorgusunun yapılabilmesi şarttır.
- Hastane satın alma işlemlerinin yürütüleceği, ihalelerin takip edilerek satın almanın gerçekleştirileceği ve eğer hastane demirbaşı veya ilaç, sarf gibi ürünlerin alımı yapılıyorsa depo modülündeki ilaç-sarf stoklarının güncellenmesi, demirbaş ve cihaz takip modülündeki cihaz ve demirbaşların sayısının güncellenmesi gerekmektedir.
- Kullanıcıların yaptığı her işlem kayıt altında tutulmalıdır. Bu kayıtların detaylı incelenebilmesi için ayrıntılı arama yapılabilecek bir ekran olmalıdır. Hastane yönetiminin kararına göre işlem kayıtlarını her kullanıcı görüntüleyemeyebilir, bu yüzden burada da diğer safhalarda olduğu gibi yetkilendirme olmalıdır.

Günümüz teknolojisini düşündüğümüzde bu bahsedilen beklentiler yetersiz kalmaktadır. Hastane bilgi sisteminde yapılabilecekler teknolojik yeniliklerden bazılarını sıralayacak olursak;

- Hastalara verilecek bir akıllı kart (smartcard) vasıtasıyla hasta sağlık kurumuna kaydından itibaren o sağlık kuruluşuyla ilgili yaptırdığı her işlemin kaydını kart vasıtasıyla yapabilmelidir.
- Hastalar sadece kendi imkanlarıyla telefon ve internet üzerinden randevu almak yerine merkezi randevu sistemine belli noktalardan bir bankamatik benzeri cihazla girerek randevu alım işlemleri gerçekleştirilebilir. Tabii bu durum hastanenin bu tip cihazları nereye kuracakları ya da kimden izin alacakları konusun ortaya çıkıyor fakat otomasyon kuran kişilerle bir alakası olmadığı için burada bahsinin geçmesine gerek görülmemiştir. Bu tip kiosk cihazlar hastanenin kendi içine de yerleştirilebilir.
- Randevusu olan hastalar giriş bankolarına uğramadan polikliniklere direk gidebilirler.
- Tedavi altındaki hastaların nabız, kan basıncı gibi değerleri doktoruna otomatik olarak elektronik posta ve/veya kısa mesaj yoluyla gönderilebilir.
- Doktorlar ve diğer sağlık ekibi ellerinde taşıyabildikleri tablet bilgisayarlar yardımıyla hasta vizitlerinde hasta bilgilerine sistemden ulaşabilir, anında hastaya tetkik veya ilaç isteyebilirler. Tablet bilgisayarın kamerası varsa, sağlık personeli tedavi altında tuttuğu hastaların gelişimini kayıt altında tutmak amacıyla fotoğraf çekebilir ve bu fotoğraflara açıklama girebilir. Hastane otomasyonu resim, doküman gibi belgeleri saklarken sıkıntı yaşamamalıdır.
- Ambulansla taşınan hastanın kan ihtiyacına göre mevcut kan bankalarında elektronik ortamda rezerv bilgileri sorgulanabilir. Gerekirse hasta için hazırlanabilir. Gidilen sağlık kuruluşunda yapılacak müdahaleler için de ön hazırlık başlatılabilir. Bu madde yine birlikte çalışabilirliğin ne kadar önemli bir nokta olduğunu hatırlatan maddeler arasında yer alabilir.
- Hastalar hastane bilgi sistemi üzerinden doktorlarına danışabilip tavsiyeler alabilirler.
- Hasta yakınları yine hastane bilgi sistemi üzerinden internet aracılığıyla hastalarına ulaşıp sesli ve görüntülü olarak geçmiş olsun dileklerini iletebilirler. Farklı illerde yer alan hasta yakınları, ziyaretçi saati

uygulamasının kalkması veya steril ortamın bozulmaması gereken durumlar için faydalı bir çözüm olacaktır.

1.16 RESTful ile Prototip

Çalışmanın bu aşamasında RESTful web servisleri kullanılarak dar kapsamlı bir hasta takip sistemi gerçekleştirimi anlatılmaktadır. Bu prototipteki özellikler aşağıdaki gibi sıralanabilir;

- Örnek teşkil etmesi açısından bir klinik yönetim modülü oluşturulmuştur. Klinik yönetimi modülünde hastaların kayıt yaptırabilecekleri birimler oluşturulup, yönetilebilmektedir.
- Bir diğer modül olan Kayıt Yönetiminde yeni hasta kaydı açma, var olan hastaların kayıtlarını yönetebilme, hastaların her bir gelişi için hasta kabul işlemi ve bu kabullerin görüntülenmesi sağlanmıştır. Oluşturulan hasta kabullerin muayene ve tedavi bilgileri ile ilişkilendirilme düzeyi oluşturulmuştur.
- RESTful yaklaşımına uyumlu olmak için, yani başka bir deyişle artık işlevlerin ve fonksiyonların çağırılması yerine kaynak tabanlı olup (resource oriented) bu kaynakların temsili için her kaynağa bir URI adresi verildi. Bu adreslere ulaşırken HTTP'nin basit get, post, update ve delete komutları kullanıldı.
- Kod geliştirme dili olarak Microsoft .NET C# tercih edildi.

Bu ihtiyaçların hepsini kodlarken ilk olarak düşünülmesi gereken kaynakların neler olması gerektiği idi. Hastane bilgi sistemi tasarlarken RESTful yöntemi ile geliştirme yapılacaksa ilk olarak kaynakların ne olduğu belirlenmelidir. Yukarıda sayılan ihtiyaç listesi içerisinde yüzlerce kaynak bulunmaktadır. Örnek vermek gerekirse hasta, kabul, ilaç, sarf malzeme muayene vb.

Prototipi gerçekleştirilecek uygulama için gösterim amaçlı az sayıda kaynak seçildi. Bunlar, hasta, giriş ve birimdir.

HTTP'nin metotları kullanılacağı için geri kalan kısımda sadece kaynaklara hangi HTTP metotlarıyla ulaşılacağına, hangi metodun ne zaman ne için kullanacağını tasarlamak gerekmektedir.

RESTful başlığı altında anlatıldığı gibi kaynağa direk erişmek ve görüntülemek için GET metodu kullanıldı. Kaynakları güncellemek veya yeni bir kayıt eklemek için POST kullanıldı. Silme operasyonu için ise DELETE kullanıldı. Çizelge 5.1'de özetle kullanılan metotlar sunulmuştur.

Çizelge 5.1 - Prototipte Kullanılan HTTP Metotlar

HTTP METOT	AMAÇ
GET	Kaynakların bilgilerine ulaşmak için kullanıldı
POST	Yeni kaynak yaratmak veya var olanı güncellemek için kullanıldı
DELETE	Var olan kaynağı silmek için kullanıldı
PUT	Kullanılmadı, yeni kaynak yaratmak için kullanılabilirdi.

Geliştirilecek olan uygulama web tabanlı olacağından dolayı .NET Framework 3.5 altında Internet Information Service (IIS) sunucusu kullanarak bir web projesi geliştirilmeye başlandı. Çalışmanın başında normal web servisler kullanmak yerine RESTful web servisleri kullanmak için yöntemler araştırılmıştır. İlk etapta araştırmalar sonucu elde edilen C# ile RESTful geliştirme tekniklerinden biri olan tüm http isteklerini (http request) yakalayan bir denetleyici olan HTTPHandler sınıfı kullanılmıştır. Böylece yapılan isteklerin tipini algılayıp okuma, yaratma, güncelleme veya silme işlemleri yapılabilir hale getirildi.

Projeye HTTPHandler ekledikten sonra yaratılan sınıfın IHTTPHandler arayüzünden gelmesi gerekmekte, böylece ProcessRequest ve IsReUsable metotları zorunlu olarak tanımlanabilir ve kullanılabilir. Projenin konfigürasyon dosyası olan web.config içerisine istenilen http fiillerini gömerek gelen istekler karşısında hangi denetleyiciye gitmesi gerektiği ayarlanabilir. Çalışmanın başında bütün http işlemleri için tek bir

HTTPHandler yaratıldı ve istekler yakalandıkça http fiillerine göre operasyonlar düzenlendi [42][43].

Çizelge 5.2 ve Çizelge 5.3'de bit ashx uzantılı bir HTTPHandler içeriğini konfigürasyon dosyasındaki gerekli değişiklik incelenebilir.

Çizelge 5.2 - Deneme projesi, web.config

```
<httpHandlers>
   <add verb="*"
px"type="TESTService.MyHTTPHandler,TESTService"/>
pHandlers>
```

Çizelge 5.3 - Deneme projesi, MyHTTPHandler.ashx

```
namespace TESTService
{
    /// <summary>
    /// Summary description for $codebehindclassname$
    /// </summary>
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    public class MyHTTPHandler : IHttpHandler
    {

        public void ProcessRequest(HttpContext context)
        {
            //URI bilgisini bul, Or: www.his.com/patient -> resource =
patient
            string resource = GetPathInfo(context.Request.Path);

            //HTTP metodları, Get, Put, Post, Delete
            string method = context.Request.HttpMethod;

            switch (method)
            {
                case "GET":
                    GetResourceInfo(resource);
                    break;
                case "POST":
                    ModifyResource(resource);
                    break;
                case "PUT":
                    InsertResource(resource);
                    break;
                case "DELETE":
                    DeleteResource(resource);
                    break;

            }
        }
```

```
        private string GetPathInfo(string url)
        {
            string[] path = url.Split("/");
            string pathInfo = path[path.Length - 1];

            return pathInfo;
        }

        public bool IsReusable
        {
            get
            {
                return false;
            }
        }
    }
}
```

Web.config'e eklediğimiz kısımda, web uygulamamızdan hangi aspx sayfası hangi http komutu ile çağırılırsa çağırılsın ilk olarak MyHTTPHandler.ashx http denetleyicisine uğrayacak.

HTTPHandler'da bulunan ProcessRequest metodu ile hem isteğin hangi http fiili ile geldiğini (get, put, post, delete) hem de isteğin yapıldığı URL adresi yakalanarak RESTful bakışında hangi kaynaklar üzerinde işlem yapılacağı belli olur.

Fakat her farklı http adresi için uzun uzun kod yazmak ve yakalanan her kaynağı ayrı ayrı değerlendirmek servis tabanlı (service oriented) ve kaynak tabanlı (resource oriented) mimari için pek uygun bir yaklaşım değildir. Araştırmanın devam eden kısmında Microsoft'un tamamen servis tabanlı mimari için geliştirmiş olduğu kaynak bazlı ve http'nin metotları ile REST yaklaşımına uygun servisler üretmeyi sağlayan ve .Net Framework 3.0 ile ortaya çıkan (3.5 ile de iyice gelişen) bir yazılım geliştirme mimarisi olan WCF keşfedildi. Çalışmanın geri kalanı WCF servisleri üzerine kurulu olan bir hastane bilgi sistemi geliştirmek üzerine yapılmıştır. Prototip için de aynı durum söz konusudur, kod geliştirme WCF servisleri üzerine oturtulmuştur.

WCF'i kısaca tanımlamak gerekirse, servis tabanlı yazılım geliştirmeyi kendine tamamen amaç edinmiş, platform bağımsız çalışabilecek uygulama geliştirmek için altyapıya sahip ve bunun için dağıtık yapıları bir arada toplamayı hedefleyen, bunların yanı sıra güvenlik gibi uğraşılması güç konuları biraz daha basite indirgeyebilen ve kodlamasını kolaylaştıran bir yazılım geliştirme mimarisidir. WCF ile geliştirilen uygulamalarda entegrasyon, birlikte çalışabilirlik ve iş süreçlerinin değişikliğine adapte olabilme gibi özellikler ön plana çıkmaktadır.

WCF servisleri yazılırken, bir tane metotların tanımlandığı arayüz de yaratılır ve servisin içereceği metotlar bu arayüzde bulunur. Servisler de bu arayüzden türer ve içerdiği metotlar servislerde yazılır ve kullanılır. WCF'in yapısını jargonlaşmış hale gelen WCF'in ABC'si oluşturur. Adres, Bağlama ve Kontrat/Sözleşme (Address, Binding, Contract) [29][44];

- Adres, WCF servislerinin adresini yani nereden ve hangi transfer protokolleriyle ulaşılacağını belirler. WCF transfer protokolü olarak HTTP, TCP ve MSMQ'yu destekler. http://localhost/WCFService.svc, net.tcp://localhost:44333/WCFService.svc, net.msmq://localhost:33444/WCFService.svc gibi adreslerden servislere ulaşılabilir. Örneklerde görüldüğü üzere servislerin uzantısı .svc'dir.
- Bağlayıcılar ise uygulamada tanımlanan WCF servisleri ile nasıl bağlantı kurulacağını tanımlamak için kullanılır. Uygulamaların konfigürasyon dosyalarında yer alırlar.
- Kontrat veya bir başka deyişle sözleşmeler ise WCF servislerinde etiket olarak kullanılacağı birimlerin başına eklenir ve bu birimlerin ne iş yaptığını veya neyi tanımladığını belirtir. Başlıca dört temel sözleşme tipi vardır; ServiceContract, OperationContract, DataContract, FaultContract. İsimlerinden anlaşıldığı gibi ServiceContract tanımlanan servisin sınıfının bir servis olduğunu, OperationContract yazılan metotların başına eklendiği takdirde servis dışarıdan çağırıldığında metotların kullanılabilmesini sağlar. Aynı şekilde DataContract'da tanımlanan sınıfların başka uygulamalarca

servis çağırılıp kullanılabilmesine ve içeriğinin tanımlanabilmesine olanak sağlar. Bu sözleşmeler sayesinde artık servislerin farklı platformda çalışan uygulamalar tarafından çağırıldığında bile neyin ne olduğu anlaşılabilir. Örneğin Java tabanlı bir uygulamadan DataSet döndüren bir WCF servisi çağırıldığında, Java uygulaması dönen değerin DataSet olduğunu DataContract sayesinde anlayabilecektir.

WCF'in ABC'si, uygulamanın konfigürasyon dosyasında bulunan bitiş noktalarında (endpoint) içerisinde saklanır. Çizelge 5.4'de uygulamanın konfigürasyon dosyası gösterilmiştir. Görüldüğü gibi servis tanımı altında bir endpoint etiketi açılıyor ve içine adres, bağlayıcı ve sözleşme bilgileri giriliyor. Bir servis için birden fazla endpoint tanımlanabilir.

Çizelge 5.4 - Web tabanlı WCF uygulaması, Web.config

```
  <system.serviceModel>
    <services>
      <service name="WcfService1.Service1"
behaviorConfiguration="WcfService1.Service1Behavior">
        <!-- Service Endpoints -->
        <endpoint address="" binding="wsHttpBinding"
contract="WcfService1.IService1">

          <identity>
            <dns value="localhost"/>
          </identity>
        </endpoint>
        <endpoint address="mex" binding="mexHttpBinding"
contract="IMetadataExchange"/>
      </service>
    </services>
    <behaviors>
      <serviceBehaviors>
        <behavior name="WcfService1.Service1Behavior">
          <!-- To avoid disclosing metadata information, set the
value below to false and remove the metadata endpoint above
before deployment -->
          <serviceMetadata httpGetEnabled="true"/>
          <!-- To receive exception details in faults for
debugging purposes, set the value below to true.  Set to false
before deployment to avoid disclosing exception information -->
          <serviceDebug includeExceptionDetailInFaults="false"/>
        </behavior>
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>
```

RESTful yaklaşımı ile modelleme yapmak için de çok uygun olan WCF'te UriTemplate etiketi ile her bir kaynağa farklı uri adresi tanımlanabilmektedir. OperationContract olan her bir metodun başına eklenebilen bu etiket sonucunda servis çağırılırken servis yolunun sonuna da eklenen tanımlanan ekler getirilince istenilen kaynakla ilgili operasyon yapılabilir.

Çizelge 5.5'de arayüzün tanımı ve içerdiği metotlar bulunmaktadır. Ayrıca yukarıda bahsedilmiş olan sözleşmelerin bu kodda uygulanışı görülmektedir. Servisin servis olduğunu belirten ServiceContract, tanımlamaları yapılmış metotların serviste kullanılacak operasyon olduklarını söyleyen OperationContract Çizelge 5.5'den daha iyi anlaşılabilir. Üzerinde OperationContract olmayan metotlar, servis çağırıldığında gözükmeyecek ve kullanılamayacaktır.

Çizelge 5.5 - IPatientService arayüzü

```
[ServiceContract]
    public interface IPatientService
    {
        [OperationContract]
        void DoWork1();

        /*[OperationContract]
        [WebGet(UriTemplate = "/{patientID}")]
        string DoWork(string patientID);*/


        [OperationContract]
        [WebGet(UriTemplate = "/patient",
ResponseFormat=WebMessageFormat.Xml)]
        List<Patient> GetPatients();

        [OperationContract]
        [WebGet(UriTemplate = "/patient/archive/{archiveNo}",
ResponseFormat = WebMessageFormat.Json)]
        Patient GetPatientFromArchiveNo(string archiveNo);

        [OperationContract]
        [WebGet(UriTemplate =
"/patient/tckimlik/{TCKimlikNo}",ResponseFormat=WebMessageFormat.J
son)]
        Patient GetPatientFromTcKimlik(string TCKimlikNo);

        [OperationContract]
        [WebGet(UriTemplate = "/HL7Patient/tckimlik/{TCKimlikNo}",
ResponseFormat = WebMessageFormat.Xml)]
        Patient GetPatientFromTcKimlik(string TCKimlikNo);
```

```
        [OperationContract]
        [WebInvoke(Method="POST", UriTemplate = "/newpatient",
ResponseFormat=WebMessageFormat.Xml)]
        void AddPatient(Stream s);


    }
```

UriTemplate'larda yine yukarıdaki tabloda görüldüğü gibi tamamen esnek ve istenildiği gibi servis tanımının sonuna gelecek şekilde yazılabilmektedir. Prototip uygulamadaki PatientService.svc tamamen hasta işlemlerine ayrılmış bir servistir. Adreslemelerde de hasta bazlı kaynaklara göre modelleme yapılmıştır. Örneğin 1199876531 TC kimlik numaralı hasta bilgisi isteniyorsa http://localhost/HIS/PatientService.svc/patient/tckimlk/1199876531 adresinden sorgu yapılabilir. Prototip uygulamada hasta ararken TC Kimlik numarası, hastalara kayıt esnasında verilen arşiv numarası üzerinden arama yapılabilmektedir. Eğer tüm hastalar çekilmek isteniyorsa UriTemplate'ı "patient" olan adresten sorgu yapılabilir (http://localhost/HIS/PatientService.svc/patient). Kodlarda da görüldüğü gibi UriTemplate'a yazılan adreslerde eğer köşeli parantez-{} kullanılıyorsa o değer değişken bir değer anlamına gelmekte ve parametre olarak o değişken mutlaka gönderilmelidir. Aksi halde endpoint bulunamadı hatası alınır.

WCF servisleri çağırıldığında yine OperationContract etiketinin özellikleri sayesinde, istek yapılırken http metotlarından hangisi kullanıldıysa algılayıp uygulamaya ona göre davranışlar sergiletilebilir. Yine Çizelge 5.5'de her metodun başında bulunan WebGet ve WebInvoke özellikleri, http istek metotlarını yönetmeyi sağlamaktadır. Eğer OperationContract'ların başına bu değerler konmazsa, metotlar direk varsayılan WebGet olarak çalışmaktadır. Koddan da anlaşılacağı üzere WebGet get operasyonunu temsil etmektedir. WebInvoke ise diğer http metotlarını yönetir. WebInvoke (Method="POST") denildiği vakit servis isteğinin post metodu ile yapıldığında ilgili operasyonu çağırması beklenmektedir. Bu çalışmada yeni kayıt eklemek ve var olan kayıtları güncellemek için post istek şekli kullanılmıştır.

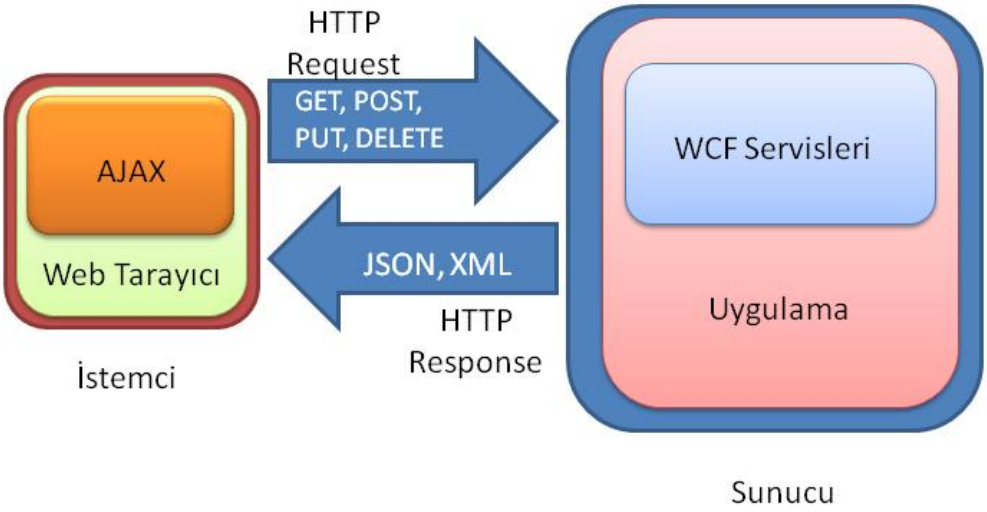
WCF'in bu kolaylıklarla dolu kullanım şeklini sayesinde her bir kaynağın temsili kısa kod parçaları ve metot tanımlamalarıyla mümkün olmaktadır. Bu özelliklere göre tez çalışmasındaki prototip için aşağıdaki kaynaklar, metotlar ve servisler kullanılmıştır;

- Unit, Patient ve Reception sınıfları
- Her bir sınıf için özel servisler. UnitService, PatientService, ReceptionService ve her birinin türetildiği arayüzler IPatientService, IReceptionService, IUnitService oluşturuldu. Arayüzler içerisinde metot tanımları eklendi ve bu metotların UriTemplate'ları ve http istek tipleri düzenlendi.
- Her Kaynak için belirtilen uri adresleri şöyle listelenebilir;
    - Unit için;
        - http://localhost/HIS/UnitService.svc/unit tüm birimleri getir
        - http://localhost/HIS/UnitService.svc/unit/name/{unitName} birim adı yazarak birim arama
        - http://localhost/HIS/UnitService.svc/unit/objectid/{objectID} birimin unique id'si üzerinde birim arama
        - http://localhost/HIS/UnitService.svc/newunit adresine yeni bir birim post ediliyor ve kayıt eklenmiş veya güncellenmiş oluyor.
    - Patient için;
        - http://localhost/HIS/PatientService.svc/patient ile sistemdeki tüm hastalar getirilecek.
        - http://localhost/HIS/PatientService.svc/archive/{archiveNo} ile hasta arşiv numarası ile sorgulama yapılıyor ve ilgili hasta var ise sonuç getiriliyor
        - http://localhost/HIS/PatientService.svc/patient/tckimlik/{TCKimlikNo} ile hasta TC Kimlik numarası irilerek arama yapılıyor.

        - http://localhost/HIS/PatientService.svc/newpatient adresine yeni bir hasta kaydı post ediliyor ve kayıt eklenmiş veya güncellenmiş oluyor.
    - Reception için;
        - http://localhost/HIS/ReceptionService.svc/reception/receptionno/{receptionNo} ile kabul almış hastaların kabul numarası girerek sorgusu yapılıyor.
        - http://localhost/HIS/ReceptionService.svc/newreception ile hastanın mevcut kabulü güncellenebilir veya yeni kabul oluşturulabilir.

Uygulama web tabanlı olduğundan dolayı, uygulamayı kullanacak olan kullanıcıların da web tabanlı olması ve böylece kurulum ve güncelleme masraflarından kurtulmak için web tarayıcılara Client olarak karar verildi. Aspx sayfalarından direk kod arkası (code behind) yani aspx sayfalarının kendi sunucu taraflı sınıflarını kullanmak yerine web servis kullanarak aslında kullanıcı tarafında çalışan uygulama da bağımsız hale gelmiş oluyor.

Normal bir aspx sayfasından web servis olarak o an ihtiyaç duyulan servis çağırılmalı ve hem iş süreçleri hem de kontroller servis tarafında saklandıktan sonra kullanıcıdan sadece gerekli veriler alınarak süreç ilerletilmelidir. HTML veya Aspx sayfasında da web servislerine asenkron şekilde ulaşmak ve sayfanın tekrar yüklenme maliyetinden kurtulmak için Ajax kullanıldı (Şekil 5.1). Çizelge 5.6'da örnek olarak hasta sorgusu yapılan ekran için servis çağırma kodu görüntülenebilir.



Şekil 5.1 - HTTP Metotları ile RESTful gerçekleştirimi

Çizelge 5.6 - Arşiv numarasından hasta ara

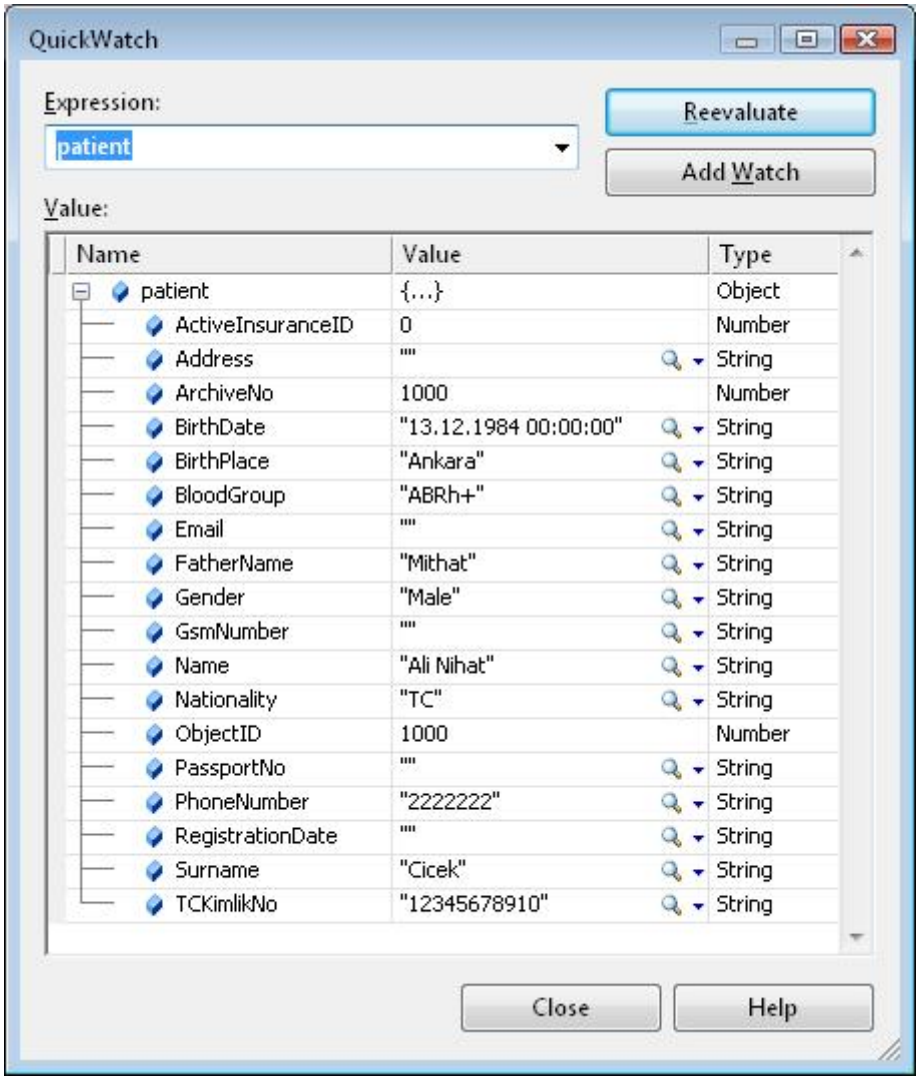
```
    var wRequest = new Sys.Net.WebRequest();
    wRequest.set_httpVerb("GET");
    var arcNo =
document.getElementById("txtPatientArchiveNo").value;
    var url =
"http://localhost/HIS/RegistrationManagement/PatientService.svc/pat
ient/archive/"+arcNo;
    wRequest.set_url(url);
    wRequest.add_completed(onArchiveNoChanged);

    wRequest.invoke();
```

Arşiv numarası 1000 olan bir hasta aratıldığında, Ajax kısmından WCF servislerine HTTP get metodu ile istek yapılmaktadır. URL adresi http://localhost/HIS/RegistrationManagement/PatientService.svc/patient/archive/1000 olacaktır. Adres yorumlanırsa "/patient" yolu verilerek hasta kaynaklarından "/archive/1000" arşiv numarası 1000 olan gelsin olarak anlaşılacaktır. Bu durumda WCF servisi URITemplate'dan yakaladığı adrese göre GetPatientFromArchiveNo metodunu çalıştıracak ve cevap dönecektir. Eğer geçerli veriler sistemde mevcut ise metodun dönüş yapacağı veri tipine bakarak (GetPatientFromArchiveNo için ResponseFormat=WebMessage.JSON) belirtilen formatta cevap verir. Verilen adreste 1000 arşiv numaralı hasta isteği yapıldığında, WCF servisinden geri gelen JSON formatındaki cevap şu şekilde olacaktır;
"{"ActiveInsuranceID":0,"Address":"","ArchiveNo":1000,"BirthDate":"13.12.1984

00:00:00","BirthPlace":"Ankara","BloodGroup":"ABRh+","Email":"","FatherName":"Mithat","Gender":"Male","GsmNumber":"","Name":"Ali Nihat","Nationality":"TC","ObjectID":1000,"PassportNo":"","PhoneNumber":"2222222","RegistrationDate":"","Surname":"Cicek","TCKimlikNo":"10894161718"}".

Bu cevap mesajı istemci tarafında gelince Javascriptin serializer kütüphanesinden yararlanılarak JSON nesnesini normal bir nesne haline getirerek kullanmaktan başka bir şey kalmıyor (Şekil 5.2). Çizelge 5.7'de geri dönen cevabın işlenmesi incelenebilir. Şekil 5.3'de ise bu mesaj işlendikten sonra prototip uygulamadaki görüntüsü gösterilmektedir. Şekil 5.4'de cevap tipi (ResponseFormat) XML olan cevap mesajı görüntülenebilir.

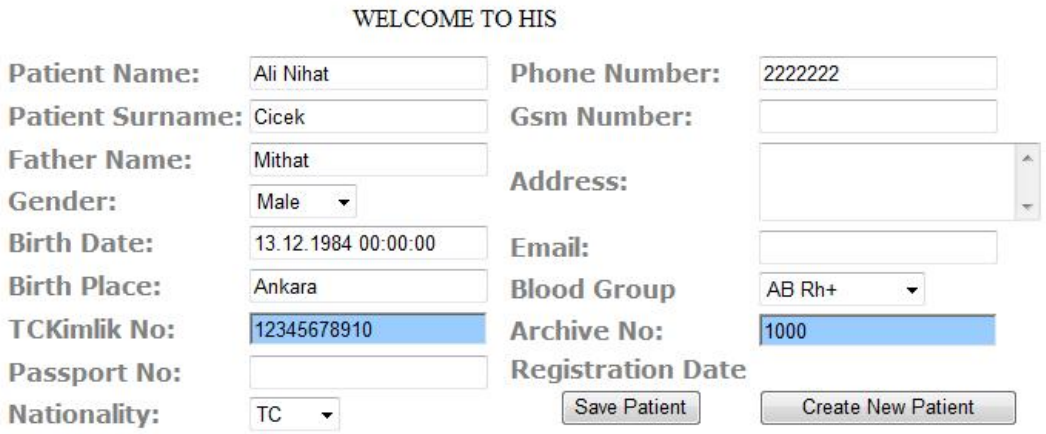


Şekil 5.2 - Serialize edilen JSON nesnesi

Çizelge 5.7 - Response mesajının işlenmesi

```
function onArchiveNoChanged( result )
{
    if(result.get_statusText() != "OK")
    {
        alert(result.get_statusText());
        return false;
    }

    var patient =
Sys.Serialization.JavaScriptSerializer.deserialize(result.get_respon
seData());
    if(patient!=null)
    {
        loadPatientDetails(patient);
    }
}
```

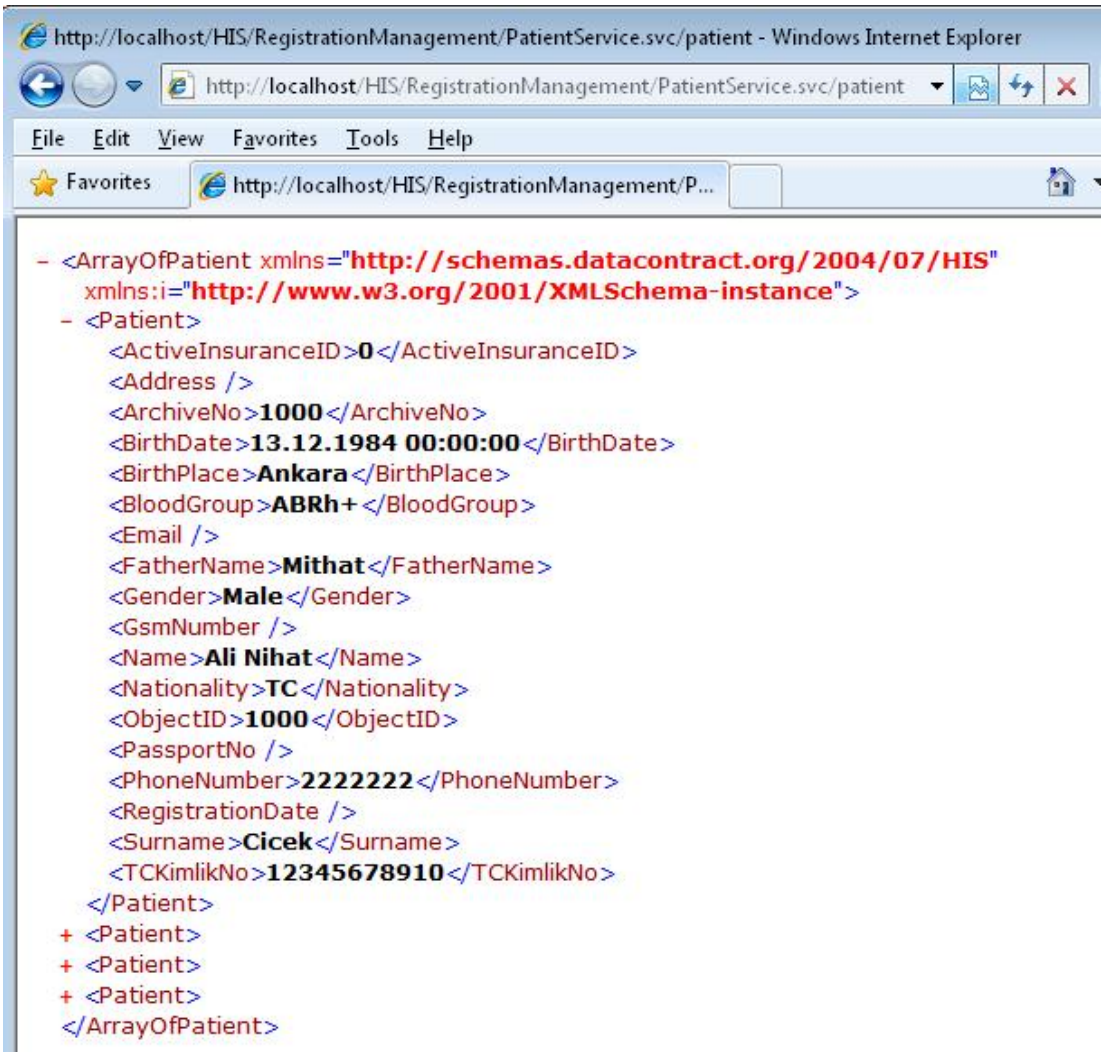


Şekil 5.3 - Hasta bilgileri ekranı

WCF ile birlikte gelen bir diğer kolaylık ise isteklerin ve isteklere karşılık giden cevapların mesaj formatlarının ayarlanabiliyor olmasıdır. XML veya JSON formatlı mesaj gönderilip alınabilmektedir. Prototip Internet Explorer üzerinden Ajax tarafından web servislerini çağırdığından dolayı, giden gelen verinin JSON formatında olması yazılımsal olarak büyük kolaylık sağlamaktadır. Çizelge 6.2'de bu formatların nasıl ayarlandığı incelenebilir.

Fakat uygulamanın web servisleri kendi içinden değil de dışarıdan farklı bir uygulama ile haberleşmesi gerekip, hastalara ait elektronik sağlık kayıtları paylaşılmak istenirse JSON formatlı mesajlar yetersiz kalmaktadır. Bu gibi

entegrasyon gereken durumlarda HL7 mesajları göndermek standartlara uymak açısından daha faydalı ve kullanışlı olacaktır. Çizelge 5.5'de HL7 Versiyon 3 mesajı döndüren (XML tabanlı) bir operasyon sözleşmesi de bulunmaktadır.



Şekil 5.4 - ResponseFormat = WebMassageFormat.Xml

Örnek uygulamada işlenen özellikler Çizelge 5.8'deki gibi özetlenebilir;

Çizelge 5.8 - RESTful Uygulaması

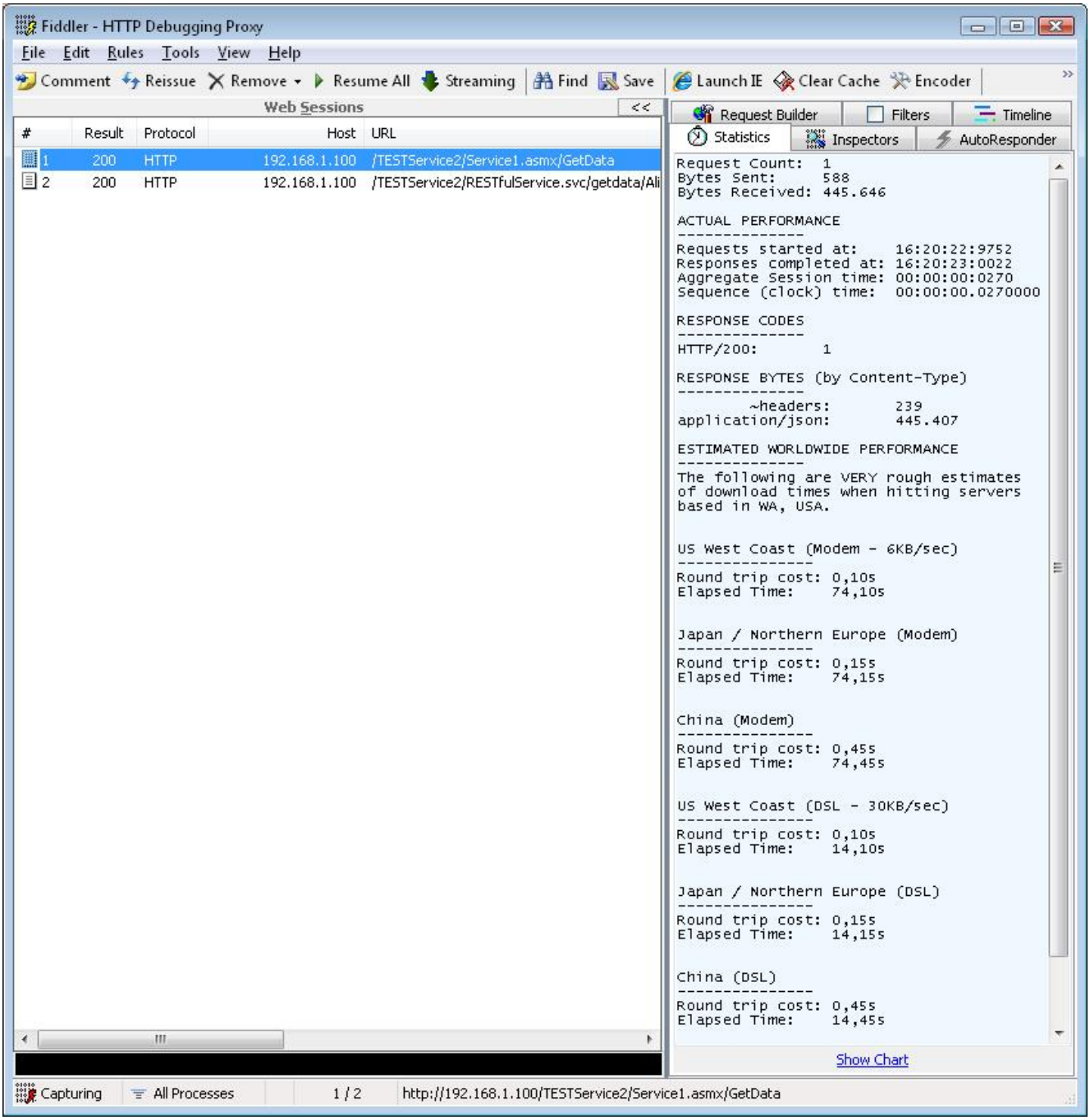
SERVİS	URL	HTTP METOT	ISTEK NESNESİ	ISTEK FORMATI	CEVAP NESNESİ	CEVAP FORMATI
PatientService	http://.../HIS/RegistrationManagement/PatientService.svc/patient	GET	-	-	Patient[]	XML
PatientService	http://.../HIS/RegistrationManagement/PatientService.svc/patient/archive/{archiveNo}	GET	-	-	Patient	JSON
PatientService	http://.../HIS/RegistrationManagement/PatientService.svc/patient/tckimlik/{TCKimlikNo}	GET	-	-	Patient	JSON
PatientService	http://.../HIS/RegistrationManagement/PatientService.svc/newpatient	POST	Patient	JSON	-	-
ReceptionService	http://.../HIS/RegistrationManagement/ReceptionService.svc/reception/{archiveno}	GET	-	-	Reception[]	JSON
ReceptionService	http://.../HIS/RegistrationManagement/ReceptionService.svc/reception/{receptionno}	GET	-	-	Reception	JSON
ReceptionService	http://.../HIS/RegistrationManagement/ReceptionService.svc/newreception	POST	Reception	JSON	-	-
ReceptionService	http://.../HIS/RegistrationManagement/ReceptionService.svc/reception/hl7/{receptionno}	GET	-	-	HL7Message	XML
UnitService	http://.../HIS/HISManagement/UnitService.svc/unit/	GET	-	-	Unit[]	JSON
UnitService	http://.../HIS/HISManagement/UnitService.svc/unit/name/{name}	GET	-	-	Unit	JSON
UnitService	http://.../HIS/HISManagement/UnitService.svc/unit/objectID/{objectID}	GET	-	-	Unit	JSON
UnitService	http://.../HIS/HISManagement/UnitService.svc/newunit	POST	Unit	JSON	-	-

1.17 WCF REST Yaklaşımı ve SOAP Karşılaştırması

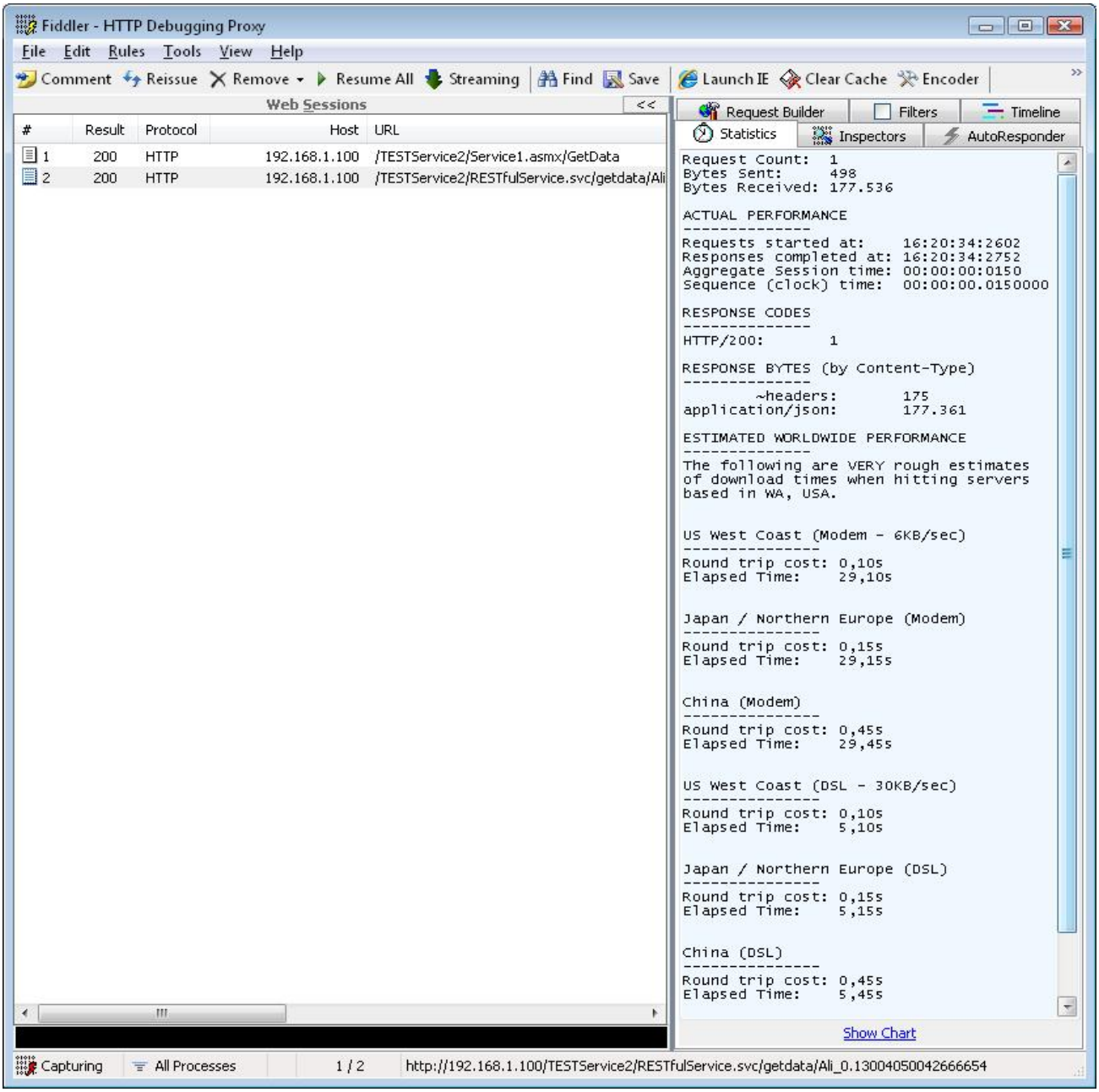
RESTful web servisleri ile SOAP web servisleri arasındaki temel fark SOAP tamamen operasyonlar üzerine kurulmuş bir yapıdır. Yani servisleri tasarlarken çağırılacak metotlar üzerine tasarlanır. REST stilinde ise kaynaklar ön plandadır. Kaynakların durumunu kullanıcıya veya diğer uygulamalara sunmak üzerine modelleme yapılır. REST'in operasyonları HTTP'nin get, post, put, delete metotlarından başkası değildir. İkisine de örnek vermek gerekirse SOAP tabanlı da HastaBilgisiBul(hastaNo) metodu bir servis olurken, REST'de ise http://localhost/HIS/hasta/21212 ile 21212 numaralı hasta bilgisine erişim gerçekleşmektedir.

SOAP'ın bir standart olduğu ve web servislerinin SOAP mesajlarıyla haberleşirken XML dokümanı kullandığı önceki kısımlarda vurgulanmıştı. SOAP'ın kendine özgü başlık yapısı ve zorunlu olan gövdesi sayesinde veriler transfer edilirken gereksiz yük taşınmaktadır. REST stiline destek veren WCF'te ise sadece veririn kendisi gidip gelmekte ve ekstra maliyete sebep olan kalıplar kullanılmamaktadır. Bu sebepten WCF servislerinin SOAP servislerine tercih edilmesinde hem giden gelen verinin boyut olarak daha karlı hem de haberleşme hızının daha performanslı olduğu iddia edilmektedir.

Karşılaştırma bir web browser üzerinden aynı işleve sahip olan metotlar üzerinde yapılmıştır. Parametre olarak aynı veri gönderilip, aynı veri seti geri döndürülmüştür. HTTP üzerinden giden gelen veri dinlenip, WCF REST servisi ve SOAP asmx servislerinin performansı gözlemlenmiştir. Hem giden gelen veri paketi içerisindeki başlık (header) hem de verinin gerek boyutu gerekse transfer hızı söz konusu olduğunda WCF servisleri büyük üstünlük sağlamıştır. HTTP istekleri Fiddler aracı ile izlenmiştir. Aşağıda Çizelge 6.1'den Çizelge 6.4'ya kadar karşılaştırma kodları incelenebilir. Fiddler aracının çıktıları ise Şekil 6.1 ve Şekil 6.2'de gösterilmiştir.



Şekil 6.1 - ASMX Web Servisinden Metot Çağırma



Şekil 6.2 - WCF Web Servisinden Metot Çağırma

Çizelge 6.1- Test için Person Class'ı

```
public class Person
        {
            private string name;
            private string surname;
            private string tcKimlikNo;
            private string birthDate;
            private string birthPlace;
            private string phoneNumber;
            private string address;
```

```
        private string email;
        private string fatherName;
        private string bloodType;

        public string Name
        {
            get { return name; }
            set { name = value; }
        }

        public string Surname
        {
            get { return surname; }
            set { surname = value; }
        }

        public string TCKimlikNo
        {
            get { return tcKimlikNo; }
            set { tcKimlikNo = value; }
        }

        public string BirthDate
        {
            get { return birthDate; }
            set { birthDate = value; }
        }

        public string BirthPlace
        {
            get { return birthPlace; }
            set { birthPlace = value; }
        }

        public string PhoneNumber
        {
            get { return phoneNumber; }
            set { phoneNumber = value; }
        }

        public string Address
        {
            get { return address; }
            set { address = value; }
        }

        public string Email
        {
            get { return email; }
            set { email = value; }
        }

        public string FatherName
        {
            get { return fatherName; }
            set { fatherName = value; }
        }
```

```
            public string BloodType
            {
                get { return bloodType; }
                set { bloodType = value; }
            }
```

Çizelge 6.2 - Test için Client Taraftan Servisleri Çağırmak

```
        function goForREST( )
        {
            var wRequest = new Sys.Net.WebRequest();
            //wRequest.get_headers()["Accept"] = acceptType;

            wRequest.set_httpVerb("GET");
            var url = "http://userx-
pc/TESTService/RESTfulService.svc/getdata/Ali";
            wRequest.set_url(url);
            wRequest.set_body("");
            wRequest.add_completed(onRESTCompleted);
            wRequest.invoke();
        }

        function onRESTCompleted( result )
        {
            if(result.get_statusText() == "OK")
            {
                var p =
Sys.Serialization.JavaScriptSerializer.deserialize(result.get_respon
seData());
                //alert(result.get_responseData());
                return;
            }
            else
            {
                alert(result.get_statusText());
                return;
            }
        }
        function goForSOAP( )
        {
            //TESTService.Service1.GetData("Ali","onSOAPCompleted");
            var obj = {"str":"Ali"+"_" + (Math.random()*
Math.random())};
            var webServicePath="http://userx-
pc/TESTService/Service1.asmx/";
            var webMethod="GetData";
            Sys.Net.WebServiceProxy.invoke(webServicePath,
                webMethod, false,obj, onSOAPCompleted,
                onSOAPFailed,"User Context",1000000);
        }
        function onSOAPCompleted(result)
        {
            alert(result);
        }
```

```
        function onSOAPFailed(result)
        {

        }
```

Çizelge 6.3 - Test için Asmx Servis Metodu

```
        [WebMethod]
        public Person[] GetData(string str)
        {
            Person[] pList = new Person[50];

            Person p = new Person();
            p.Name = "Ali";
            p.Surname = "Cicek";
            p.TCKimlikNo = "12345678910";
            p.BirthDate = "12.12.1955";
            p.BirthPlace = "Ankara";
            p.Address = "Ankara";
            p.Email = "a@a.com";
            p.FatherName = "BABA ADI";
            p.BloodType = "AB RH+";

            for (int i = 0; i < 50; i++)
            {
                pList[i] = p;
            }

            return pList;
            //return "Hello ";// +str;

        }
```

Çizelge 6.4 - Test için WCF Servisi

```
    [ServiceContract]
    public interface IRESTfulService
    {
        [OperationContract]
        [WebGet(UriTemplate = "/getdata/{str}", ResponseFormat =
WebMessageFormat.Json)]
        Person[] GetData(string str);
    }

    [DataContract]
    public class Person
    {
        private string name;
        private string surname;
        private string tcKimlikNo;
        private string birthDate;
        private string birthPlace;
        private string phoneNumber;
        private string address;
        private string email;
```

```
        private string fatherName;
        private string bloodType;

        [DataMember]
        public string Name
        {
            get { return name; }
            set { name = value; }
        }

        [DataMember]
        public string Surname
        {
            get { return surname; }
            set { surname = value; }
        }

        [DataMember]
        public string TCKimlikNo
        {
            get { return tcKimlikNo; }
            set { tcKimlikNo = value; }
        }

        [DataMember]
        public string BirthDate
        {
            get { return birthDate; }
            set { birthDate = value; }
        }

        [DataMember]
        public string BirthPlace
        {
            get { return birthPlace; }
            set { birthPlace = value; }
        }

        [DataMember]
        public string PhoneNumber
        {
            get { return phoneNumber; }
            set { phoneNumber = value; }
        }

        [DataMember]
        public string Address
        {
            get { return address; }
            set { address = value; }
        }

        [DataMember]
        public string Email
        {
            get { return email; }
            set { email = value; }
```

```
        }

        [DataMember]
        public string FatherName
        {
            get { return fatherName; }
            set { fatherName = value; }
        }

        [DataMember]
        public string BloodType
        {
            get { return bloodType; }
            set { bloodType = value; }
        }

    }


    public class RESTfulService : IRESTfulService
    {
        public Person[] GetData(string str)
        {
            Person[] pList = new Person[20];

            Person p = new Person();
            p.Name = "Ali";
            p.Surname = "Cicek";
            p.TCKimlikNo = "12345678910";
            p.BirthDate = "12.12.1955";
            p.BirthPlace = "Ankara";
            p.Address = "Ankara";
            p.Email = "a@a.com";
            p.FatherName = "BABA ADI";
            p.BloodType = "AB RH+";


            for (int i = 0; i < 20; i++)
            {
                pList[i] = p;
            }

            return pList;
            //return "Hello ";// +str + ", This is RESTful";
        }
    }
```

SONUÇ

Sonuç olarak günümüzde web servis kullanımının entegrasyon gerektiren uygulamalarda sektör ne olursa olsun yaygınlaştığı tartışılmaz bir gerçektir. Bilgisayar otomasyonu kullanmayan firmalar dahi artık bulundukları sektör içerisinde rakipleriyle rekabet edemez hale gelmektedir. Tabi kurum bulunduğu branşta tek olsa dahi günümüz şartlarında oluşan veri çöplüğünden kaynaklanan yönetim sıkıntısını manüel olarak hallederken güç kaybedecektir.

Sağlık sektöründe giden gelen veri yoğunluğu inanılmaz büyük ölçekli olduğundan dolayı sağlık kurumlarını, özellikle hastaneleri yönetmek epeyce zorlaşmıştır. Kullanılan bilgisayar otomasyonları dahi bu büyük veri ambarlarında doğru ve işe yarar veriyi bulmakta zorlanmaktadırlar. Özellikle ülkemizde her gün değişen sağlık kanunları ve bunun yanında geliştirilen diğer sistemlerle birlikte çalışabilirlik zorunlulukları ile hali hazırda hastanelerin kendi işleyişleri için istedikleri değişik istekler yüzünden yazılım firmaları değişen şartlara yetişebilmek için hem kaliteden ödün veriyor, hem de git gide karmaşıklaşan sistemler üzerinde ipin ucu kaçmışçasına geliştirme yapmaya çalışıyorlar. Bu da ciddi zaman ve para kaybına sebep olabiliyor.

Bu çalışma esnasında araştırılan yöntemler sonucu sağlık uygulamalarının ellerinde tuttuğu verileri sadece kendileri için düzgün saklamaları değil, diğer uygulamalarla da iletişime geçtiğinde fazla değişiklik yapmadan doğru bilgilerin transferinin sağlayabilecek durumda tutmaları gerektiği üzerinde durulmuştur. Kişilere ait elektronik sağlık kayıtlarının her zaman her sağlık kuruluşunda baştan yaratılması değil, gerektiğinde birinden diğerine transfer edilerek kişilerin sağlık geçmişlerinin de tedavi ve kontrol esnasında bulunmasının yararı büyük olacaktır. Sağlık kayıtlarının gerektiğinde sistemler arası transferinin gerekliliğinden dolayı verilerin standart biçimde saklanması çok ciddi bir iştir. Araştırma sırasında dünyanın sağlık alanında kabul ettiği HL7 standardının önemi üzerinde durulmuştur.

Bir diğer yararlı olacak çalışma uygulama geliştirirken alt yapının nasıl olması gerektiği hakkında yapılmıştır. Web tabanlı uygulamaların getirdiği esneklikten dolayı bu yönde ilerleme sağlanmaya çalışılmıştır. Günün sık kullanılan terimlerinden SOA üzerinde durulmuştur. Servis tabanlı uygulama geliştirilirken bir yandan da Web 2.0'ın kurallarına göre hareket edilmeye dikkat edilmiştir. Bu yüzden RESTful web servisleri tez çalışması içine dahil olmuştur.

REST stilinde geliştirme yapmak için araştırmalar yapılırken Microsoft'un .NET Framework 3.0 ile ortaya çıkan WCF mimarisi bulunmuştur. WCF ile normal SOAP tabanlı asmx web servisleri karşılaştırılmış ve WCF servislerinin üstünlüğü ispatlanmıştır.

Bu çalışmanın posteri Eylül 2009'da İstanbul'da gerçekleştirilen Elektronik Sağlık Konferansında yer almıştır [46].

Bu çalışmanın devamında da prototip olarak geliştirilen ve sadece RESTful servislerini test etmeye ve HL7 mesajları üremeye yarayan ufak çaplı uygulama geliştirilerek farklı modüller içeren tam kapsamlı ve farklı uygulamalarla birlikte çalışmaya açık, performansı yeni teknolojilerle desteklenen, servis tabanlı bir hastane otomasyonu oluşturmak hedeflenmektedir.

KAYNAKLAR

[1] Sağlık Bakanlılığı, Hastane Bilgi Sistemleri Alımı Çerçeve İlkeleri Dokümanı, http://www.saglik.gov.tr/BIDB/Genel/BelgeGoster.aspx?F6E10F8892433CFF1A9547B61DAFFE2AA510106937B3F39C, erişim tarihi: 31 Ağustos 2009

[2] Web Services Explained, http://www.service-architecture.com/web-services/articles/web_services_explained.html, erişim tarihi: 31 Ağustos 2009

[3] Web Services Definition, http://www.service-architecture.com/web-services/articles/web_services_definition.html, erişim tarihi: 31 Ağustos 2009

[4] W3Schools, Web Services Tutorial, http://www.w3schools.com/webservices/default.asp, erişim tarihi: 31 Ağustos 2009

[5] W3Schools, SOAP, http://www.w3schools.com/soap/default.asp, erişim tarihi: 31 Ağustos 2009

[6] W3Schools, WSDL, http://www.w3schools.com/wsdl/default.asp, erişim tarihi: 31 Ağustos 2009

[7] W3Schools, UDDI, http://www.w3schools.com/wsdl/wsdl_uddi.asp, erişim tarihi: 31 Ağustos 2009

[8] Wikipedia, Web 2.0, http://en.wikipedia.org/wiki/Web_2.0, erişim tarihi: 31 Ağustos 2009

[9] O'Reilly, T., Design Patterns and Business Models for the Next Generation of Software, http://oreilly.com/web2/archive/what-is-web-20.html, erişim tarihi: 31 Ağustos 2009

[10] SOA Definition, http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html, erişim tarihi: 31 Ağustos 2009

[11] What is Service Oriented Architecture, http://www.xml.com/lpt/a/1292, erişim tarihi: 31 Ağustos 2009

[12] Microsoft, ERP Nedir, http://www.microsoft.com/turkiye/dynamics/erp/erp_nedir.mspx, erişim tarihi: 31 Ağustos 2009

[13] Agopyan, A., IBM Yazılım Grubu, SOA Prensipleri, http://www.ardenagopyan.com/downloads/sunumlar/ibk-soaprensipleri.pdf, erişim tarihi: 31 Ağustos 2009

[14] Turkmia, İkibinli Yıllar Türkiye'sinde Sağlıkta Bilgi Stratejileri http://www.turkmia.org/files/16.PDF, erişim tarihi: 31 Ağustos 2009

[15] Sağlık Bakanlığı Bilgi İşlem Daire Başkanlığı, Ahmet Özçam, Sağlıkta E-Dönüşüm, http://www.sagliknet.saglik.gov.tr/portal_pages/notlogin/bilisimciler/docs/entegrasyon_egitimleri/Saglikta_e_Donusum_AHMET_OZCAM.ppt, erişim tarihi: 31 Ağustos 2009

[16] The Healthcare Information and Management Systems Society (HIMSS), Electronic Health Record, http://www.himss.org/ASP/topics_ehr.asp, erişim tarihi: 31 Ağustos 2009

[17] National Institutes of Health, National Center for Research Resources, Overview of EHR, http://www.ncrr.nih.gov/publications/informatics/EHR.pdf, erişim tarihi: 31 Ağustos 2009

[18] National Audit Office, Publications, Department of Health: The National Programme for IT in the NHS, http://www.nao.org.uk/publications/0506/department_of_health_the_nati.aspx, erişim tarihi: 31 Ağustos 2009

[19] Health Level 7 Official Site, Frequently Asked Questions, http://www.hl7.org/about/FAQs/index.cfm, erişim tarihi: 31 Ağustos 2009

[20] Tıp Bilişim Derneği, HL7 Özet Bilgi Dokümanı, http://www.turkmia.org/hl7_dokumanlar.php, erişim tarihi: 31 Ağustos 2009

[21] Van Hentenryck, K., HL7 Associate Executive Director, Health Level Seven: A Standard for Health Information Systems, Medical Computing Today, Haziran 2008

[22] HL7 V3 Resources, HL7 Australia, http://www.hl7.org.au/HL7-V3-Resources.htm, erişim tarihi: 31 Ağustos 2009

[23] Clinical Document Architecture, Release 2, http://www.hl7.org/v3ballot/html/infrastructure/cda/cda.htm, erişim tarihi: 31 Ağustos 2009

[24] HL7 Reference Information Model, http://www.hl7.org/v3ballot/html/infrastructure/rim/rim.htm, erişim tarihi: 31 Ağustos 2009

[25] Soysal, Ergin, Turkmia, HL7 V3 Eğitimi, Antalya, Haziran 2008

[26] Costello, R. L., Building Web Services the REST Way, http://www.xfront.com/REST-Web-Services.html, erişim tarihi: 31 Ağustos 2009

[27] Tyagi, S., RESTful Web Services, Ağustos 2006, http://java.sun.com/developer/technicalArticles/WebServices/restful/, erişim tarihi: 31 Ağustos 2009

[28] Fielding R. T.,2000, Architectural Styles and the Design of Network-based Software Architectures, Akademik inceleme, California Üniversitesi, Bilişim ve Bilgisayar Sistemleri,IRVINE, http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf, erişim tarihi: 31 Ağustos 2009

[29] Skonnard, A., PluralSight A Guide to Designing and Building RESTful Web Services with WCF 3.5, Ekim 2008, http://msdn.microsoft.com/en-us/library/dd203052.aspx, erişim tarihi: 31 Ağustos 2009

[30] McMillan, R., Network World, A RESTful Approach to Web services, Şubat 2007, http://www.networkworld.com/ee/2003/eerest.html?page=2, erişim tarihi: 31 Ağustos 2009

[31] Sağlık-NET Portalı, Sağlıkta E-Dönüşüm, http://www.sagliknet.saglik.gov.tr/portal_pages/notlogin/bilisimciler/bilisimciler_sagliknetilesagliktadonusum.htm, erişim tarihi: 31 Ağustos 2009

[32] Sağlık-NET Portali, Ulusal Sağlık Bilgi Sistemi, http://www.sagliknet.saglik.gov.tr/portal_pages/notlogin/bilisimciler/bilisimciler_sagliknetileulusalsaglik.htm, erişim tarihi: 31 Ağustos 2009

[33] Sağlık-NET Portalı, Sağlık Kodlama Referans Sunucusu, http://www.sagliknet.saglik.gov.tr/portal_pages/notlogin/bilisimciler/bilisimciler_startdart_skrs, erişim tarihi: 31 Ağustos 2009

[34] Sağlık-NET Portalı, Ulusal Sağlık Veri Sözlüğü, http://www.sagliknet.saglik.gov.tr/USVSBrowser/All.jsp, erişim tarihi: 31 Ağustos 2009

[35] Sağlık-NET Portalı, Entegrasyon Dokümanlarıi http://www.sagliknet.saglik.gov.tr/portal_pages/notlogin/dokumanlar/dokumanlar_dok.htm, erişim tarihi: 31 Ağustos 2009

[36] Artemis Projesi, http://www.srdc.metu.edu.tr/webpage/projects/artemis/, erişim tarihi: 31 Ağustos 2009

[37] Bicer, V., Laleci, G., Dogac, A., Kabak, Y., "Artemis Message Exchange Framework: Semantic Interoperability of Exchanged Messages in the Healthcare Domain" ACM Sigmod Record, Vol. 34, No. Haziran 2005 (Science Citation Index Expanded, Impact Factor: 00.675)

[38] Bicer, V., Laleci, G., Dogac, A., Kabak, Y., "Providing Semantic Interoperability in the Healthcare Domain through Ontology Mapping", eChallenges 2005, Ljubljana, Slovenia

[39] Microsoft HealthVault, http://www.healthvault.com/Industry/index.html, erişim tarihi: 31 Ağustos 2009

[40] Microsoft Amalga Hastane Bilgi Sistemi 2009, http://www.microsoft.com/amalga/products/microsoftamalgahis/default.mspx, erişim tarihi: 31 Ağustos 2009

[41] Google Health, http://www.google.com/intl/tr-TR/health/tour/index.html, erişim tarihi: 31 Ağustos 2009

[42] Altiere, J., Building API in .NET, Mart 2008, http://www.jaltiere.com/?p=41, erişim tarihi: 31 Ağustos 2009

[43] Ulsberg, A., REST ASP.NET Example, http://intertwingly.net/wiki/pie/RestAspNetExample, erişim tarihi: 31 Ağustos 2009

[44] Şenyurt, B. S., WCF etiketi ile ilgili makaleler, http://www.buraksenyurt.com/?tag=/wcf&page=5, erişim tarihi: 31 Ağustos 2009

[45] Godoro, Web Servisleri: İnternet Devriminde İkinci Aşama, http://www.godoro.com/divisions/ehil/Mecmua/Magazines/Articles/txt/html/article_WebServices.html, erişim tarihi: 31 Ağustos 2009

[46] Doğdu, E., Çiçek, A.N., **"**Implementing E-Health Systems Using RESTful Web Services", Poster, eHealth 2009 - Second International ICST Conference on Electronic Healthcare for the 21st Century, Eylül 2009

[47] Elkstein, M., Learn REST: A Tutorial: Using REST in C#, http://rest.elkstein.org/2008/02/using-rest-in-c-sharp.html, erişim tarihi: 31 Ağustos 2009

[48] Elkstein, M., Learn REST: A Tutorial: Using REST in Java, http://rest.elkstein.org/2008/02/using-rest-in-java.html, erişim tarihi: 31 Ağustos 2009

[49] Elkstein, M., Learn REST: A Tutorial: Using REST in Javascript, http://rest.elkstein.org/2008/02/using-rest-in-javascript.html, erişim tarihi: 31 Ağustos 2009

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı	: ÇİÇEK, Ali Nihat
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 13.12.1984 Ankara
Medeni hali	: Bekar
E-mail	: ncicek@etu.edu.tr

Eğitim

Lisans: Bilkent Üniversitesi / Bilgisayar Teknolojisi ve Bilişim Sistemleri, 2006

Yüksek Lisans: TOBB Ekonomi ve Teknoloji Üniversitesi / Bilgisayar Mühendisliği

İş Deneyimi

2007-2009, Entegre Enformasyon Sistemleri, Yazılım Uzmanı

Yabancı Dil

İngilizce

Yayınlar

Doğdu, E., Çiçek, A.N., "Implementing E-Health Systems Using RESTful Web Services", Poster, eHealth 2009" - Second International ICST Conference on Electronic Healthcare for the 21st Century, Eylül 2009