KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

JEODEZİ VE FOTOGRAMETRİ MÜHENDİSLİĞİ ANABİLİM DALI

KONUMSAL VERİ ALTYAPILARININ WEB SERVİSLERİ İLE GERÇEKLEŞTİRİLMESİ: MEVCUT DURUM ANALİZİ VE GELECEK YÖNELİMLERİNİN BELİRLENMESİ

DOKTORA TEZİ

Harita Yük. Müh. Halil AKINCI

EYLÜL 2006
TRABZON

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

JEODEZİ VE FOTOGRAMETRİ MÜHENDİSLİĞİ ANABİLİM DALI

KONUMSAL VERİ ALTYAPILARININ WEB SERVİSLERİ İLE GERÇEKLEŞTİRİLMESİ: MEVCUT DURUM ANALİZİ VE GELECEK YÖNELİMLERİNİN BELİRLENMESİ

Harita Yük. Müh. Halil AKINCI

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü'nce
"Doktor"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 15.08.2006
Tezin Savunma Tarihi : 25.09.2006

Tez Danışmanı : Doç. Dr. Çetin CÖMERT
Jüri Üyesi : Prof. Dr. Tahsin YOMRALIOĞLU
Jüri Üyesi : Prof. Dr. Emin Zeki BAŞKENT
Jüri Üyesi : Prof. Dr. Aslan DİLAVER
Jüri Üyesi : Prof. Dr. Dursun Zafer ŞEKER

Enstitü Müdürü: Prof. Dr. Emin Zeki BAŞKENT

Trabzon 2006

ÖNSÖZ

"Konumsal Veri Altyapılarının Web Servisleri ile Gerçekleştirilmesi: Mevcut Durum Analizi ve Gelecek Yönelimlerinin Belirlenmesi" başlıklı çalışma, Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Jeodezi ve Fotogrametri Mühendisliği Anabilim Dalında doktora tezi olarak hazırlanmıştır.

Doktora tezimin danışmanlığını üstlenerek, tez çalışması süresince üst görüşü ve önerileri ile yol gösteren, değerli katkıları ile bana destek olan, ilgi ve yardımını esirgemeyen değerli hocam Sayın Doç. Dr. Çetin CÖMERT'e sonsuz teşekkürlerimi sunarım.

Doktora tez izleme komitesinde görev alarak, çalışmalarım sırasında bilimsel desteklerini esirgemeyen değerli hocalarım Prof. Dr. Tahsin YOMRALIOĞLU ve Prof. Dr. Emin Zeki BAŞKENT'e teşekkür ederim.

Çalışmaya maddi kaynak sağlayan Karadeniz Teknik Üniversitesi Rektörlüğüne teşekkür ederim. Tez aşaması boyunca daima desteğini gördüğüm sevgili eşim Orman Yüksek Mühendisi Hazan ALKAN AKINCI'ya teşekkür ederim. Ayrıca bugünlere gelmemde büyük fedakarlıklar gösteren anneme, babama ve kardeşlerime sonsuz teşekkürlerimi sunarım.

Halil AKINCI

Trabzon 2006

İÇİNDEKİLER

ÖZET

Günümüzün aşırı rekabetçi ve değişken iş modelinin gereği olan hızlı, kaliteli ve ekonomik "servis sağlama" işlevi ancak, ilgili servis sağlayıcıların işbirliği ya da birlikte çalışmaları ile mümkündür. İşbirliği için ise "birlikte işlerlik" altyapılarına ihtiyaç vardır. Bu ihtiyaç, özellikle son yıllarda yalnızca konumsal veri ve servisler alanında değil, "e-iş" ve "e-devlet" gibi diğer pek çok alanda çok belirgin bir biçimde hissedilmektedir. "Konumsal Veri Altyapıları (KVA)" konumsal veri yönetimine yönelik birlikte işlerlik altyapılarıdır. KVA'lar yerel, bölgesel, ulusal ve uluslararası düzeyde gerçekleştirilebilir. Ulusal düzeydeki, Ulusal KVA (UKVA) olarak bilenenler en çok bilinen örneklerdir. UKVA ülke düzeyindeki tüm kamu kurumları, yerel yönetimler, özel sektör ve konumsal veri ile iş yapan bütün kesimler arasında "birlikte işlerliği" sağlayacak ve istemcilere aradıkları veri ve servislere anlık erişim olanağı sağlayacak olan bir altyapıdır. Bununla birlikte, mevcut KVA gerçekleştirimleri hala "insan yönelimli Web" olarak nitelenen tarzdadır. Yani insan kullanıcılara yönelik olan, aranan bir veri ya da servisin bulunmasının özellikle aranan konumsal veri gibi anlamsal içeriği çok zengin olabilen veriler olduğunda çok kolay olmadığı tarzdır. Oysa gelecek, uygulamaların birbirleri ile doğrudan konuşabilmelerine olanak tanıyacak "Anlamsal (Semantik) Web" dedir. Anlamsal Web'e geçişi sağlayacak olan yaklaşım ise Servis Yönelimli Mimari (SYM) ve onun halen en popüler gerçekleştirim şekli olan Web Servisleri (WS) dir. SYM nin çıkış noktası aslında birlikte işlerliğin sağlanması olmuştur. Ancak KVA'ların, WS teknolojisi ile gerçekleştirilebilmeleri Dünya genelinde henüz başarılabilmiş bir görev değildir. Çünkü böyle bir gerçekleştirimin, teknik birlikte işlerlik altyapısını tanımlayan, üzerinde anlaşma sağlanmış bir çerçeve mevcut değildir. Bu soruna bir çözüm getirmeyi hedeflemiş olan bu tez çalışmasında söz konusu gerçekleştirimin nasıl başarılabileceğini belirleyen temel soruların bütüncül, basitleştirici ve genel bir çerçevede irdelenmesine olanak tanıyacak bir çatı geliştirilmiştir.

Anahtar Kelimeler: Birlikte İşlerlik, Ulusal Konumsal Veri Altyapısı, Web Servisleri, Servis Yönelimli Mimari, Servis Düzenleme, Anlamsal Web,

SUMMARY

Implementation of Spatial Data Infrastructures with Web Services: Analyzing the Current Status and Determining the Future Directions

The provision of rapid, quality and economical services, the requirement of Today's highly dynamic and competitive business models, can only be achieved by collaborations of the involved parties, which actually require "Interoperability infrastructures". The need for interoperability infrastructures has been felt not only in the spatial data arena but also in many areas like "e-business" or "e-government" for the past several years. Spatial Data Infrastructures (SDI)" are "interoperability infrastructures" for the spatial data. SDIs s can be realized in local, regional, national and international levels. The ones at the national level are known as National SDI (NSDI) and are most popular. Providing interoperability among the state organizations, private sector, local governments, and all other spatial data using communities, an NSDI will enable "ad-hoc" access to the data and services. Nevertheless, the current SDI implementations are mostly in so called "human-oriented Web" form. That is they have been designed fort he human user, where it is not easy to find a piece of data or service especially when the requested is generally of a semantically reach state like spatial data. Whereas the future is of a form of the Web, where applications can directly "talk" to the each other and locate their data themselves. This is what is known as "Semantic Web". The paradigm which will enable the Semantic Web is the Services Oriented Architecture (SOA) and its currently most popular way of implementation, Web Services (WS). The main driving force behind the SOA has, in fact, been interoperability. Despite the immense World wide activity, however, the implementation of SDIs with WS technology is not something achieved yet. The reason for this has been the lack of a commonly accepted framework which would define the technical interoperability infrastructure in this undertaking. Addressing this problem has been the goal of this thesis. And towards that end, a framework which enables dealing with the involved issues from a unifying, distilling and general perspective has been developed in this thesis.

Key Words: Interoperability, National Spatial Data Infrastructures, Web Services, Services Oriented Architecture, Service Composition, Semantic Web

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SEMBOLLER DİZİNİ

BİT	: Bilgi ve İletişim Teknolojileri
BPEL4WS	: Business Process Execution Language for Web Services
BPELJ	: Business Process Execution Language for Java
CBS	: Coğrafi Bilgi Sistemleri
CORBA	: Common Object Request Broker Architecture
CQL	: Common Catalogue Query Language
CSW	: Catalogue Service Web
D-U-N-S	: Dun & Bradstreet Number Identifier System
ebRIM	: ebXML Registry Information Model
ebRS	: ebXML Registry Services and Protocols
ebXML	: electronic business XML
EVBİS	: Emlak Vergisi Bilgi Sistemi
FGDC	: Federal Geographic Data Committee
GML	: Geography Markup Language
GOS	: Geospatial One-Stop
GSDI	: Global Spatial Data Infrastructure
HGK	: Harita Genel Komutanlığı
HTN	: Hierarchical Task Networks
IDL	: Interface Definition Language
IIOP	: Internet Inter-ORB Potocol
IIS	: Internet Information Server
INSPIRE	: Infrastructure for Spatial Information in Europe
İDF	: İmar Durumu Formu
İPM	: İmar Planlama Müdürlüğü
JRMP	: Java Remote Method Protocol
KBS	: Kent Bilgi Sistemleri
KTH	: Konum Temelli Hizmetler
KVA	: Konumsal Veri Altyapıları
NAICS	: North American Industry Classification System
OAI	: Open Archive Initiative

OASIS : Organization for the Advancement of Structured Information Standards
OGC : Open Geospatial Consortium
ORB : Object Request Broker
RPN : Reverse Polish Notation
SD : Servis Düzenleme
SHOP2 : Simple Hierarchical Ordered Planner 2
SOA : Service Oriented Architecture
SOAP : Simple Objects Access Protocol
SVG : Scalable Vector Graphics
SYM : Servis Yönelimli Mimari
TAKBİS : Tapu Kadastro Bilgi Sistemleri
TKGM : Tapu ve Kadastro Genel Müdürlüğü
TRSICS : Thomas Register Supplier Identifier Code System
TSÖ : Temel Sorgulanabilir Öznitelikler
TUCBS : Türkiye Ulusal Coğrafi Bilgi Sistemi
UDDI : Universal Description Discovery, and Integration
UKVA : Ulusal Konumsal Veri Altyapısı
UN/CEFACT : United Nations Centre for Trade Facilitation and Electronic Business
URI : Uniform Resource Identifier
W3C : World Wide Web Consortium
WAF : Web Accessible Folder
WCS : Web Coverage Service
WFS : Web Feature Service
WMS : Web Map Service
WS : Web Servisleri
WSBPEL : Web Services Business Process Execution Language
WS-CDL : Web Services Choreography Description Language
WSDL : Web Services Description Language
WSFL : Web Services Flow Language
XML : Extensible Markup Language
YZP : Yapay Zeka Planlama

1. GENELBİLGİLER

1.1. Giriş

Günümüzün çok rekabetçi ve değişken Dünya'sında, genel anlamda herhangi bir "servis sağlama" işleminin hızlı, kaliteli ve ekonomik olması kaçınılmazdır. Böylesi servisler ise ancak, farklı servis sağlayıcılarının etkin işbirliği ya da birlikte çalışabilmeleri ile mümkündür. Bunun için de "birlikte işlerlik" altyapılarına ihtiyaç vardır. Bu ihtiyaç özellikle son on yılda yalnızca konumsal veri ve servisler alanında değil, "e-iş" gibi diğer bütün alanlarda çok belirgin bir biçimde ortaya çıkmıştır. Konumsal açıdan, anılan birlikte işlerlik altyapıları, "Konumsal Veri Altyapıları (KVA)" olarak adlandırılmaktadır. Ulusal Konumsal Veri Altyapısı (UKVA) ile bir ülke genelini kapsayan KVA kastedilmektedir. UKVA, ülke genelinde tüm kamu kurumları, yerel yönetimler, özel sektör ve konumsal veri ile iş yapan bütün kesimler arasında "birlikte işlerliği" sağlayacak ve vatandaşlar dahil ilgililere, gereksinim duydukları veri ve servislere "anlık" erişim ve kullanım olanağı tanıyacak bir altyapıdır.

KVA'ların önemi bütün Dünya'da yeterince iyi anlaşıldığı için, bu alanda ulusal ve uluslararası çok sayıda proje mevcuttur. Amerika, Kanada ve İngiltere gibi ülkelerin genellikle en az on yıldır süregelen UKVA çalışmaları yanında, uluslararası boyutta özellikle son beş yıldır sürmekte olan çok sayıda girişim mevcuttur. Bir örnek, Avrupa geneline yönelik bir bilgi altyapısının kurulması için önerilmiş olan INSPIRE (Infrastructure for Spatial Information in Europe) projesidir (INSPIRE, 2005). Diğer yandan işlerin kurumsal boyutundan ziyade teknik boyutları ile uğraşan uluslararası organlar mevcuttur. Bunlardan en çok bilinenleri Açık CBS Konsorsiyumu (OGC), Uluslararası standartlar örgütünün 211 nolu teknik komisyonu (ISO TC211) ve Web teknolojileri için de WWW (W3C) konsorsiyumudur.

Genel olarak KVA'lar birlikte işlerlik altyapıları olarak algılanabilir. Birlikte işlerlik ise, genel anlamda farklı dil ya da kavramlar kullanan tarafların birbirleri ile konuşabilmesi olarak tanımlanabilir. Dolayısıyla söz konusu altyapı bunu sağlamalıdır. Herhangi bir KVA birlikte işlerlik altyapısının iki temel bileşeni vardır. Biri kurumsal anlamdaki diğeri ise teknik anlamdaki birlikte işlerlik altyapısıdır. Bu tezin ilgi alanı, UKVA ya da herhangi bir ölçekteki KVA'nın teknik birlikte işlerlik altyapısıdır.

1.2. Problemin Tanımı

Bu tezin çıkış noktası, UKVA'nın teknik olarak nasıl ve hangi yolla gerçekleştirilebileceğinin belirlenmesi olmuştur. Ancak genellik arz etmesi bakımından amaç geniş ölçekli bir KVA'nın, Servis Yönelimli Mimari (SYM) ye dayalı, teknik birlikte işlerlik altyapısının nasıl gerçekleştirilebileceğinin belirlenmesi olarak konmuştur. Problem ise bunun nasıl başarılabileceğidir.

Web Servisleri (WS) teknolojik ve endüstriyel açıdan son birkaç yılın en aktif araştırma ve uygulama alanlarından biri durumundadır. KVA alanında ulusal ve uluslararası sayısız girişim, uluslararası düzeyde çok yoğun bir tempoda devam eden standart geliştirme çalışmaları, sayıları hızla artan bilimsel çalışmalar, konumsal Web servisleri ile yeni bir sektör ve trendi yaratan ticari ve açık kod yazılımlar bu alandaki büyük heyecanı yansıtmaktadır. Ancak, gelinen noktada SYM ya da WS yönelimli geniş ölçekli bir KVA'nın teknik birlikte işlerlik altyapısının nasıl gerçekleştirileceği henüz tanımlanamamıştır. Bu durum, bu noktada WS yönelimli bir KVA gerçekleştirmek isteyenlerin işini, gerek teknik gerçekleştirimciler ve gerekse KVA'ları kurmak ve yaşatmaktan sorumlu karar vericiler bazında çok zor kılmaktadır. Söz konusu gerçekleştirimcilerin çözmesi gereken başlıca sorunlardan bir kaçı şunlardır: Web servisi tanımlama, bulma ve servis düzenleme için hangi teknoloji, araç, standart ya da belirtimler kullanılmalıdır? Servis yayınlama ve bulmada merkezi ya da dağıtık ağ yapılandırılmalarından hangisi tercih edilmelidir? Servis kataloglama ve katalog federasyonları nasıl gerçekleştirilecektir? Hangi ticari ve/veya açık kod yazılımlarının tercih edileceğine nasıl karar verilecektir? WS yönelimli KVA'lara geçişin ana amacı olan birlikte işlerlik, acaba gerçekten sağlanabilecek midir yoksa sorun olmaya devam mı edecektir? Bu birlikte işlerlik "söz dizimsel" ya da "anlamsal" düzeyde mi sağlanmalıdır? Mevcut araçlar hangisini olanaklı kılmaktadır? Belirli kuşak araç ve teknolojilerle yapılan gerçekleştirimler arasında geçiş olanaklı olacak mıdır? Mevcut uygulamalar (legacy applications) WS mimarisine uyarlanabilecek midir?

1.3. Çalışmanın Amacı

Bu çalışmada, Dünya genelinde gerek akademik ve gerekse endüstriyel gerçekleştirim boyutunda en aktif araştırma ve geliştirme alanlarından biri olan KVA'ların, Web teknolojilerinde en son kuşak teknoloji olarak kabul edilen SYM ve onun halen en popüler gerçekleştirim şekli olan WS teknolojileri ile gerçekleştirilebilmesine yönelik strateji ve olabilirliğin belirlenmesi amaçlanmıştır. Ayrıca Dünya genelinde bu alandaki gelecek yönelimlerinin belirlenmesi amaçlanmıştır.

1.4. Metodoloji

Bu çalışmanın gerçekleştirilmesinde sırasıyla aşağıdaki gibi bir yol izlenmiştir.

- Birlikte işlerlik altyapıları geliştirmeye yönelik teknolojilerin incelenmesi.
- Servis Yönelimli Mimari'nin (SYM) incelenmesi.
- SYM'nin halen en popüler gerçekleştirim şekli olan Web Servisleri (WS) teknolojilerinin incelenmesi ve pratikte uygulanması.
- WS teknolojilerine dayalı e-belediye modelinin teknik gerçekleştirim yolunun gösterilmesi.
- Mevcut durumun analizi ve WS yönelimli KVA teknik gerçekleştirimlerinin nasıl başarılabileceğinin belirlenebilmesine yönelik, çok sayıda akademik çalışma, çeşitli ulusal ve uluslararası proje ve girişim, ilgili uluslararası standart ve belirtim geliştirme kuruluşlarının ilgili standart ve belirtimleri, ticari ve açık kaynak kodlu yazılımların ilgili ürünleri, ulusal ve uluslararası düzeyde KVA gerçekleştirimlerinin detaylı olarak incelenmesi.
- Teknik gerçekleştirim stratejisinin belirlenmesi.
- Gelecek yönelimlerinin belirlenmesi.

1.5. Ulusal Konumsal Veri Altyapısı

Türkiye için Ulusal Konumsal Veri Altyapısı (UKVA) ilk olarak Cömert ve Banger (1995) tarafından önerilmiştir. Gerek bu çalışmada ve gerekse daha sonra (Cömert, 1996), (Cömert ve Banger, 1996), (Cömert, 1998) gibi birçok çalışmada UKVA tanıtılmış, acilen

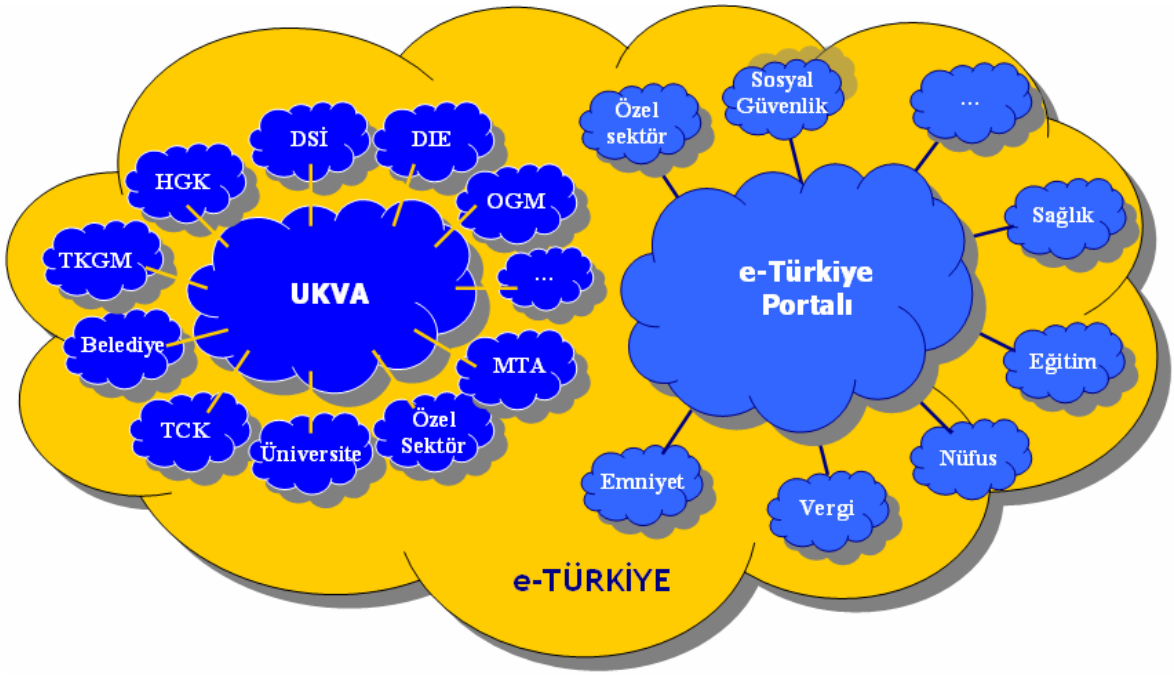
işlevselleştirilmesinin Türkiye için taşıdığı hayati önem vurgulanmış ve bunun için ilk olarak yapılması gerekenler belirlenmiştir. Ancak, UKVA'nın Türkiye için ilk defa önerilmesinden sonra geçen yaklaşık on yıllık sürede veri yönetimi, dağıtık sistemler ve Web teknolojilerinde çok önemli gelişmeler olmuştur. Örneğin Coğrafi Bilgi Sistemleri (CBS) yazılımı üreten firmalar kendi yazılımlarının İnternet harita sunucularını geliştirmişlerdir. Bu yazılımlar yardımıyla kullanıcılar artık kendi sistemlerinde herhangi bir CBS yazılımı olmadan, henüz kısıtlı da olsa bir takım CBS fonksiyonlarını İnternet üzerinde yerine getirebilmektedir. Diğer yandan, Java dili ve CORBA teknolojilerindeki gelişmeler "Object Web" kavramını gündeme getirmiştir (Orfali ve Harkey, 1998). Nihayet "Web servisleri" yaklaşımı gündemdedir (W3C, 2004d). Doğal olarak bu gelişmeler, UKVA kavramına yeni boyutlar getirmiş ve UKVA'nın kullanacağı teknolojik altyapının yeniden tanımlanmasını zorunlu kılmıştır.

UKVA, ülke düzeyinde kamu kurumları, özel sektör, yerel yönetimler ve konumsal veri ile iş yapan bütün kesimler arasında "birlikte işlerliği" sağlayacak ve vatandaşlar dahil ilgililere, gereksinim duydukları veri ve servislere anında erişim ve kullanım olanağı tanıyacak bir altyapı olarak tanımlanabilir. "Birlikte işlerlik" (interoperability), çok genel olarak donanım ve yazılım olarak farklı sistemlerin birbirleri ile "iletişim kurabilmesi" ya da daha iddialı bir sözcükle, "konuşabilmesi" olarak tanımlanabilir. UKVA isimlendirmesinde "altyapı" ile kastedilen, konumsal verinin toplanması, işlenmesi, dağıtımı, kullanımı, güncellenmesi ve güvenliğinin sağlanması için gerekli tüm teknolojiler, politikalar, standartlar, insan kaynakları ve ilgili faaliyetlerin tümünün çerçeve tanımıdır (OMB,1992).

UKVA'nın temel ilkesi "ortaklaşa kullanım", yani aynı veri ve servisin farklı kullanıcılar tarafından kullanılabilmesini sağlamaktır. Bunun gerekçesi ise ekonomik, hızlı ve doğru çözümler üretmektir. Burada "ortaklaşa kullanım" ile kastedilen, daha yaygın bilinen ismi ile veri paylaşımıdır. Web teknolojileri alanındaki son gelişmelerin etkisi ile buna "servis paylaşımı" da eklenmiştir. UKVA, çözüm üreticiler için hızlı, ekonomik ve doğru çözümleri, hizmet sunucuları için kaliteli ve hızlı hizmeti, karar vericiler için de hızlı ve doğru kararları olanaklı kılacaktır. UKVA, doğrudan ve dolaylı pek çok etkiyle ülke ekonomisine çok önemli katkılar sağlayacaktır.

Ülke düzeyinde ilgili bütün kamu ve özel sektör kuruluşları, yerel yönetimler, üniversiteler ve çeşitli diğer kuruluşlar, UKVA'da sunucu veya istemci ya da hem sunucu hem de istemci konumunda olabilirler (Şekil 1). Harita Genel Komutanlığı (HGK) ve Tapu

Kadastro birimleri ağırlıklı olarak sunucu, üniversiteler istemci, belediyeler ise hem istemci hem sunucu konumlarına örnektir. UKVA'da sunucu ve/veya istemci konumunda bulunacak her bir katılımcı taraf, kendi sorumluluğundaki konumsal veri, meta veri ve servisleri üretip, güncelleyerek UKVA üzerinden kullanıma sunacaktır. Merkezi bir üretim ve dağıtım söz konusu değildir.



Şekil 1. UKVA ve e-Türkiye algılaması

1.6. Neden UKVA?

Neden UKVA sorusunun cevabı aslında neden e-Türkiye sorusununki ile aynıdır. Çünkü UKVA ve e-Türkiye ile aşılmaya çalışılan sorunlar aynıdır. Aradaki fark UKVA'nın konumsal veriye, e- Türkiye'nin ise genelde konumsal olmayan veriye yönelik olmasıdır. Aslında UKVA, e-Türkiye'nin pek çok bileşeninden yalnızca biri, fakat aynı zamanda da en önemlisidir. Çünkü coğrafi veri çok geniş bir kullanım alanına sahiptir. E-Türkiye ve UKVA, "Bilgi ve İletişim Teknolojileri" nin yeterli ve yerinde kullanımını sağlayarak, Türkiye'nin bir "Bilgi Toplumu" olabilmesinin yolunu açacaktır. Bu da ekonomik ve teknolojik açıdan olduğu kadar, kültürel açıdan da yeni ufuklar açacaktır. UKVA'nın başlıca yararları aşağıdaki gibi özetlenebilir.

- UKVA'nın işlevsel hale gelmesi ile bugüne kadar bir türlü gerçekleştirilememiş ya da bir şekilde kesintiye uğramış, Türkiye'nin bir "bilgi toplumu" olabilmesi için hayati önem taşıyan pek çok proje gerçekleştirebilecektir. Bu projelere en çarpıcı örnek, bir türlü hayata geçirilemeyen ve bir şekilde tamamlanmış olsa bile, UKVA olmadan yaşatılması mümkün olmayan Kent Bilgi Sistemleri (KBS) projeleridir. Bu bağlamda, UKVA'nın henüz işlevsel olmadığı Ülkemizde, KBS ya da herhangi bir KVA girişimi, zaten acil kaynak ihtiyacındaki Ülkemiz açısından kaynak israfından başka bir anlam taşımayacaktır.
- UKVA'nın belki de en önemli katkısı, yeni "iş alanları" yaratacak olmasıdır. Çok büyük ve etki alanı çok geniş bir proje olması nedeniyle, gerek kurulması ve işletilmesi sürecindeki faaliyetler ve gerekse etki alanları itibariyle yaratacağı yeni iş olanakları, haritacılık dahil pek çok sektörde istihdam yaratacaktır. Örneğin UKVA'nın halen önündeki en önemli engel olan Tapu Kadastro Bilgi Sistemlerinin (TAKBİS) eksikliği, özel sektörün bu alanda bir an önce devreye sokulmasını gerektirmektedir. Ayrıca, TAKBİS in güncel tutulmasında da özel sektör için çok geniş olanaklar mevcuttur. Çünkü UKVA'ya sürekli olarak veri sağlamak durumundaki TAKBİS'in bu işlevini yalnızca kamu imkanları ile yerine getirmesi, bugünkü yapı ile hem olanaksız hem de gereksizdir.
- UKVA, ülke düzeyinde konumsal veri erişimi ve kullanımını hızlı ve ekonomik hale getirecektir. Çünkü konumsal verinin ilk elden toplanması, pahalı olmasının yanında zaman alıcı bir işlemdir. Kaldı ki, günümüz uygulamalarının veri ihtiyacı tek bir kurumun veri toplama kapasitesinin çok üzerindedir. UKVA ile herhangi bir kurum ya da kullanıcı, ihtiyacı olan veri ya da servisleri yeniden oluşturmak yerine, doğrudan UKVA üzerinden sağlayabilecektir. Dolayısıyla UKVA, kaynak israfının önleyerek, maliyetleri düşürecek ve ülke ekonomisine çok büyük bir katkı yapacaktır.
- UKVA, şimdiden acil durum yönetimi gibi pek çok uygulama için kritik önem taşıyan ve Konum Temelli Hizmetler (KTH) gibi geleceğin uygulamaları açısından çok büyük öneme sahip olacak "anlık" veri ve servis iletişimine olanak tanıyacaktır.
- UKVA ile her tür bilgi kayıt altına alınacağından, "oto-kontrol" mekanizmaları işler hale gelecektir. Bunun her alanda yaratacağı etki çok büyük olacaktır. Örneğin UKVA sayesinde emlak vergileri, internet üzerinden ilgili Web servisleri ile anlık olarak hesaplanabilecek ve ülkemizde büyük oranlara varan emlak vergisi kayıpları önlenebilecektir. Bu, acil kaynak ihtiyacındaki belediye ve ülke ekonomileri için çok önemli bir katkı olacaktır. Bu yapı, beyan sistemini ortadan kaldıran yasal bir değişiklik

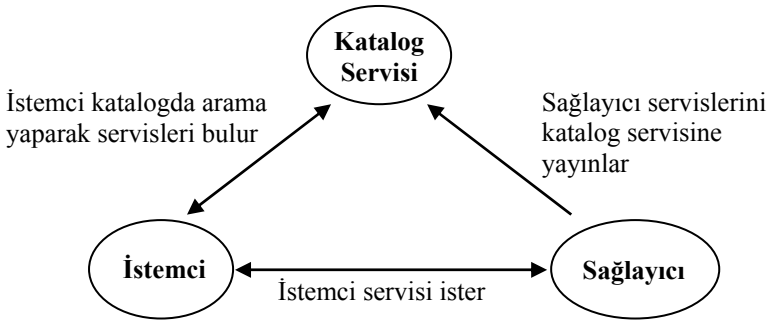
gerektirmekte ve Tapu kadastro ve ilgili belediye birimlerinin çevrimiçi (on-line) servislerine gereksinim duymaktadır (Cömert ve Akıncı, 2003).

- UKVA, "değerlenmiş ürünler" için de son derece uygun bir ortam hazırlayacaktır. Örneğin belediyeler, emlak bilgilerini belirli ücretler dahilinde özel emlak bürolarına sunabilecek, emlak büroları bu bilgilere kendi bilgilerini ekleyerek, yeni ürünler üretebileceklerdir. Bu ürünlerin kullanıcıları da ilgilendikleri emlakların, örneğin şehir içindeki konumlarını harita üzerinden görebilecek ya da örneğin bir önceki yıl ödenen emlak vergisi tutarını sorgulayabileceklerdir. Benzer şekilde turizm şirketleri, kamu kurumlarından sağlayacakları bilgilerle, örneğin "etkileşimli gezi planı" gibi yeni ürünleri müşterilerinin kullanımına sunabileceklerdir.
- UKVA ile kamu ve özel sektörün iş yapma tarzı değişecek, bu kurumlar UKVA'ya uygun ürün ve hizmetler üretmek durumunda kalacaklardır. Örneğin, Türkiye'de konumsal veri yönetimi alanında çok az kullanılmakta olan "meta veri", UKVA ile önem kazanacak, kurumlar artık ürettikleri verinin kalitesini meta veri ile belgeleyeceklerdir.
- UKVA ile ilgili bütün tarafların birbirlerinden ve vatandaşların yerel yönetimler ve kamu kurumlarından aldıkları hizmetlerin kalitesinde çok büyük oranda bir iyileşme görülecektir. İnsanımızın özellikle kamu kurumlarında ve belediyelerde basit bir işlem için bile günlerce uğraşması, öteden beri yaygın şikayet konusudur. Hizmet kalitesinin "çağdaş" düzeylere ulaşması, vatandaşı mutlu ederken, ülkesine ve devletine olan güvenini tesis edecektir. Toplumların mutluluk ve üretkenliklerinin toplumu oluşturan bireylere bağlı olduğu düşünüldüğünde, UKVA'nın bu açıdan yapacağı katkı daha iyi anlaşılacaktır.
- UKVA ile yıllardır hep konuşulan ancak bir türlü başarılamayan "Kamunun yeniden yapılandırılması" başarılabilir. Çünkü kurumların hak ve yükümlülüklerinin UKVA kapsamında yeniden belirlenmesi, sadece kamu kurumlarının yeniden yapılandırılmasını değil, aynı zamanda kamu ve özel sektörün birlikte çalışabilme modellerini de gündeme getirecektir. UKVA'nın hayata geçirilmesi ve özel sektör rolünün güçlendirilmesi, kamu kurumlarında öteden beri şikayet konusu olan "hantallığı" gidererek, hizmetlerin hızlı ve ekonomik bir biçimde yerine getirilmesini sağlayacaktır.

1.7. Servis Yönelimli Mimari

Günümüzün oldukça rekabetçi ve dinamik dünyası, herhangi bir servisin hızlı, ekonomik ve kaliteli olmasını gerektirmektedir. Hızlı, ekonomik ve kaliteli servisler, sadece birlikte işleyen (interoperable) uygulamalar ile gerçekleştirilebilir. Farklı programlama dilleri kullanılarak geliştirilen, ağ üzerinde faklı yerlerde bulunan ve farklı platformlara sahip bilgisayarlar üzerinde koşan uygulamaların, belirli görevleri yerine getirmek için, birlikte işleyebilmelerini sağlayan çeşitli sistemler ve yazılım mimarileri geliştirilmiştir. Şu an oldukça popüler ve yaygın olan yazılım mimarisi, Servis Yönelimli Mimari (Service Oriented Architecture) ya da kısaca SYM (SOA) olarak adlandırılmaktadır. SYM, uygulamaların son kullanıcılara servis olarak sunulduğu dağıtık sistemleri gerçekleştirmek için bir yaklaşımdır (Colan, 2004). Web servisleri, SYM'yi gerçekleştirmenin en iyi ve şu anki en popüler yolu olarak kabul edilmektedir (McGovern vd., 2003; Colan, 2004; Weerawarana vd., 2005). W3C (2002), bir Web servisini, İnternet tabanlı protokoller aracılığıyla XML tabanlı mesajları kullanarak diğer yazılım uygulamaları ile doğrudan etkileşimleri destekleyen, arayüzleri ve bağlantıları XML tabanlı diller kullanılarak tanımlanabilen ve bulunabilen ve bir URI (Uniform Resource Identifier) tarafından tanımlanan bir yazılım uygulaması olarak tanımlamaktadır. Üst düzey bir görüşle bir Web servisi, belirli bir görevi gerçekleştirmek için internet üzerinden çağrılabilen bir uygulama olarak tanımlanabilir. Bir Web servisi, görevini yerine getirmek için başka Web servislerini çağırabilir. Bir Web servisleri ortamı, servis sağlayıcılarının sahip oldukları Web servislerini bir katalog servisi aracılığıyla yayınladığı ve istemcilerin katalogdan servisleri bulup, uygulamalarını gerçekleştirmek için onları sağlayıcılardan istedikleri bir ortam olarak kavramsallaştırılabilir (Şekil 2).

SYM, kendine has özellikleri olan özel bir yazılım mimarisidir. Servis tasarımcılarının ve geliştiricilerinin SYM'nin özelliklerini anlamaları, Web servislerinin en etkin şekilde kullanımını sağlayabilmeleri açısından önemlidir (McGovern vd., 2003). SYM, birlikte işlerliği (interoperability) sağlayan, gevşek bağlı (loosely coupled), genel (coarse-grained) arayüzlere sahip, uygulama geliştirmek için düzenlenebilen (composable), katalog servislerinde arama yapılarak bulunabilen (discoverable) ve dinamik olarak bağlanılabilinen (dynamically bound) servislerin geliştirilmesini destekleyen bir mimaridir.



Şekil 2. Servis yönelimli mimarinin bileşenleri (Vinoski, 2002).

Bulunabilirlik ve dinamik olarak bağlanılabilirlik: SYM mimarisinde, servis sağlayıcılar sahip oldukları Web servislerini katalog servisleri aracılığıyla yayınlarlar. İstemciler ise, katalog servislerinde arama yaparak ihtiyaç duydukları Web servislerini bulurlar. Bir istemci, ihtiyaç duyduğu bir Web servisini bulmak için bir katalog servisine sorgu gönderir. Katalog servisi istemciye, aradığı özellikleri sağlayan Web servisleri ile ilgili kayıtlar gönderir. Söz konusu kayıtlar, katalog servisinin Web servislerini tanımlamak için kullandığı meta verilerdir. Bir Web servisinin meta verisi, servisin arayüzünü tanımlayan dokümanın adresini ve servisin sağlayıcıdaki adresini içerir. İstemci, servislerin arayüz dokümanlarını inceleyerek hangi servisi kullanacağına karar verir. İstemci daha sonra ilgili sağlayıcıya bağlanır. Bir Web servisinin arayüzü, servisi yürütmek için gerekli olan bilgileri içerir. İstemci, servisin arayüzünde tanımlanan formatta bir istek mesajı üretir ve yine servisin arayüzünde tanımlanan iletişim protokolünü kullanarak istek mesajını sağlayıcıya gönderir. Sağlayıcı, istek mesajını alır ve servisi koşturur. Sağlayıcı daha sonra, servisin arayüzünde tanımlanan formatta bir yanıt mesajı üreterek istemciye gönderir. İstemci, servisi yürütmek için ihtiyaç duyduğu tüm bilgileri koşum anında elde eder ve tasarım anında servisle ilgili herhangi bir bilgiye ihtiyaç duymaz. Servisin arayüz dokümanı dinamik olarak bulunur ve mesajlar dinamik olarak oluşturulur. Böylece istemcilerin, servislerin arayüzlerinde meydana gelen değişikliklerden etkilenmemeleri sağlanır.

Servis düzenleme: Servis düzenleme (service composition), bir kullanıcı tarafından gerçekleştirilmek istenen karmaşık bir uygulamanın, tek bir Web servisi tarafından yerine getirilememesi durumunda, uygun Web servislerinin birleştirilerek uygulamanın gerçekleştirilmesi şeklinde tanımlanabilir. Servis düzenleme, SYM de uygulama geliştirmenin yoludur ve kullanıcılara, farklı sağlayıcılar tarafından sunulan Web

servislerini kullanarak uygulama geliştirme olanağı sağlamaktadır. Servis düzenleme sonucunda "birleşik servis" (composite service) adı verilen yeni bir Web servisi geliştirilir. Bir birleşik Web servisi, görevini yerine getirmek için diğer Web servislerini kullanır. Servis düzenleme, bölüm 2.4 de detaylı olarak ele alınacaktır.

Gevşek bağlılık: Bağlılık (coupling), bir istemci uygulama ile bir sunucu uygulama arasındaki bağımlılık ilişkisini tanımlar. "Sıkı bağlılık" (tight coupling) ve "gevşek bağlılık" (loose coupling) olarak adlandırılan iki çeşit bağlılık vardır. Bir sistemin bağlılık derecesi, sistemin değiştirilebilirliğini doğrudan etkiler. Sıkı bağlı sistemlerde, sunucu uygulamanın arayüzünde yapılan bir değişiklik, istemci uygulamada da değişikliklerin yapılmasını gerektirir. Örneğin bir sunucu uygulamasında, bir operasyonun parametrelerinin yerlerinin değiştirilmesi, parametrelerden birkaçının çıkarılması, yeni parametrelerin eklenmesi veya parametrelerin isimlerinin ya da tiplerinin değiştirilmesi, istemci uygulamada söz konusu operasyonu çağırmak için kullanılan rutinlerde de değişikliklerin yapılmasını gerektirir. Ayrıca bir istemcinin, bir sağlayıcı tarafından sunulan bir uygulamayı kullanabilmesi için ne kadar çok bilgiye ihtiyacı olursa, bağlılık derecesi de o kadar artmaktadır. Başka bir ifadeyle, eğer bir istemci bir sunucu uygulamanın adresi, yayınlandığı platform ve geliştirildiği programlama dili gibi detaylarını biliyorsa istemci ile sunucu sıkı bağlıdır.

Birlikte işlerliği gerçekleştirmek için geliştirilen CORBA (OMG, 2002) ve RMI (SUN, 2000) gibi dağıtık nesne sistemleri sıkı bağlı (tightly coupled) sistemlerdir. CORBA da bir sunucu nesnenin (uygulamanın) arayüzü IDL (Interface Definition Language) (OMG, 2002), RMI de ise java dili kullanılarak tanımlanır. Her iki sistemde de arayüz bir derleyici tarafından derlenerek, "*stub*" ve "*skeleton*" adı verilen yazılım bileşenleri elde edilir. *Stub*, sunucu nesnesinin istemci tarafındaki temsilcisidir ve sunucu nesnenin gerçekleştirdiği tüm metotlara sahiptir. İstemci uygulaması, sunucu nesnenin bir metodunu çağırmak için *stub* nesnesindeki ilgili metodu çağırır. *Stub* nesnesinin görevi, metot parametrelerini serileştirmek (marshalling), *skeleton* ile bağlantı kurmak ve serileştiren veriyi *skeleton* nesnesine göndermektir. *Skeleton* nesnesi ise sunucuda yer alır ve *stub* nesnesi tarafından gönderilen serileştirilmiş metot parametrelerini okur, parametreleri sunucu nesnesinin yazıldığı programlama dilindeki ilgili veri tiplerine dönüştürür (unmarshalling) ve sunucu nesnesinin ilgili metodunu çağırır. *Skeleton*, metodun döndürdüğü değeri serileştirilerek (marshalling) *stub* nesnesine gönderir. *Stub* nesnesi, gelen değeri okur, istemci uygulamanın yazıldığı programlama dilindeki ilgili veri tipine

dönüştürür (unmarshalling) ve değeri istemci uygulamasına aktarır. Sunucu nesnenin arayüzünde yapılacak olan bir değişiklik, *stub* ve *skeleton* nesnelerinin çalışmamasına neden olacaktır. Bu nedenle, arayüzün yeniden derlenerek *stub* ve *skeleton* nesnelerinin yeniden yaratılması ve istemci uygulamasında gerekli değişikliklerin yapılması gerekmektedir.

Eğer bir istemci uygulaması, bir sunucu uygulamasını çağırmak için, sunucu hakkında detaylı bilgiye ihtiyaç duymuyorsa, istemci ve sunucu uygulaması gevşek bağlıdır. Gevşek bağlılık, istemci ve sunucu uygulamalarının, birbirlerini etkilemeden değiştirilebilmelerine olanak sağlar. Her yazılım mimarisi, uygulamalar arasında gevşek bağılığı gerçekleştirmek için çalışır. SYM, istemci ve sunucu arasında gevşek bağlılığı destekler (McGovern vd., 2003). SYM, gevşek bağlılığı, bulunabilirlik ve dinamik olarak bağlanılabilirlik özelliği sayesinde sağlar. İstemci, katalog servislerinde arama yaparak ihtiyaç duyduğu bir Web servisini bulur ve servisin arayüz dokümanındaki bilgileri kullanarak servisi çağırır. İstemci sadece servisin arayüzüne bağımlıdır. Servisin hangi programlama dili kullanılarak gerçekleştirildiği kullanıcı açısından önemli değildir. Her ne kadar SYM mimarisi gevşek bağlılığı desteklese de, servis düzenleme sonucunda elde edilen birleşik Web servisleri sıkı bağlıdır. Bir birleşik Web servisi, görevini yerine getirmek için diğer Web servislerini kullanır. Bileşen servislerden birinin arayüzünün değişmesi, birleşik servisin gerçekleştirim kodunda da değişiklik yapılmasını gerektirmektedir.

Genel arayüzlere sahiplik: Bir Web servisinin işlevi, tasarım anında bir servis tasarımcısı tarafından belirlenir. Web servislerinin işlevsellik derecesini belirlemek için kullanılan tasarım ilkesi, "*granularity*" olarak adlandırılmaktadır. Web servisleri, *granularity* düzeylerine göre, özel (fine-grained) ve genel (coarse-grained) servisler olmak üzere ikiye ayrılır. Özel servisler, belirli küçük görevleri yerine getiriler ve küçük miktarda veri döndürürler. Genel servisler ise, birkaç özel servis tarafından gerçekleştirilebilecek görevleri yerine getirirler ve büyük miktarda veri döndürürler (Schmelzer, 2006). Örneğin, *poligonGetir* isimli bir Web servisi, istemciye bir poligon noktasının sadece koordinatlarını döndürüyorsa özel, poligon noktasının hem koordinatlarını hem de röper krokisini döndürüyorsa genel bir Web servisidir. Özel servisler, genel servislere göre kullanıcılara daha fazla esneklik sağlarlar. Ancak bir kullanıcının özel servisleri kullanması durumunda istediği verileri elde etmesi uzun zaman alır. Çünkü kullanıcının internet üzerinden çağırması gereken servis sayısı artmaktadır. Örneğin bir kullanıcının bir poligon noktasının

koordinatlarını ve röper krokisini elde edebilmesi için iki ayrı Web servisini (örneğin, *poligonGetir* ve *poligonRöperGetir*) çağırması gerekmektedir. Genel servisler, internet üzerinden çağrılması gereken servis sayısını azaltırlar. Ancak genel servisler kullanıcıların ihtiyaç duyduklarından daha fazla miktarda veri döndürebilirler. Örneğin *poligonGetir* Web servisinin bir genel servis olduğunu ve bir poligon noktasının hem koordinatlarını hem de röper krokisini döndürdüğünü kabul edersek, bir poligon noktasının sadece koordinatlarına ihtiyaç duyan bir kullanıcı, poligonun röper krokisini de almış olacaktır.

Servis düzenleme, SYM nin en önemli özelliklerinden biridir ve kullanıcıların mevcut Web servislerini bir araya getirerek yeni uygulamalar geliştirmesine olanak sağlamaktadır. Bir servis düzenlemede kullanılan Web servislerinin özel ya da genel olması, gerçekleştirilecek olan uygulamanın performansını doğrudan etkilemektedir. Bir birleşik Web servisi tarafından kullanılan servislerin özel servisler olması durumunda, uygulama için ihtiyaç duyulan verilerin elde edilmesi uzun zaman alacaktır. Çünkü birleşik servisin internet üzerinden çeşitli özel servisleri çağırması ve bu servislerden gelecek olan yanıt mesajlarını beklemesi gerekmektedir. Birleşik servisin kullandığı özel servis sayısının fazla olması, düzenlemede kullanılan özel servislerin aynı anda başka istemciler tarafından da kullanılması ve internet trafiğinin yoğunluğu uygulamanın yavaş gerçekleştirilmesine neden olur. Birleşik servisin, uygulamayı gerçekleştirmek için genel servisler kullanması durumunda ise, uygulama tarafından ihtiyaç duyulmayan verilerde elde edilir. Ancak, birleşik servis tarafından çağrılan servis sayısının azalması ve bir genel servis tarafından döndürülen veriler içerisinden istenen bir verinin çekip çıkarılmasının, aynı verinin internet üzerinden bir özel servisi çağırarak elde edilmesinden daha hızlı olması nedeniyle uygulama daha hızlı gerçekleştirilir. Bu nedenle, özellikle bir kurum içerisindeki uygulamalarda özel servislerin, farklı kurumlar tarafından sunulan servislerin kullanılarak uygulamaların geliştirileceği durumlarda ise genel servislerin kullanılması önerilmektedir (Colan, 2004).

Birlikte işlerlik: SYM'nin en önemli hedefi, birlikte işlerliği sağlamaktır (Colan, 2004; Mahmoud, 2005). Birlikte işlerlik, farklı uygulamaların birbirleriyle konuşabilmesi olarak tanımlanabilir. SYM, birlikte işlerlik hedefine ulaşmak için, birlikte işlerliğin geleneksel yaklaşımlarından farklı bir felsefe izlemektedir. SYM, uygulamalar birbirleriyle konuşabildiği, birbirlerinden belirli servisleri isteyebildiği ve birlikte işleyebildiği sürece uygulamaların ne kadar farklı oldukları ve ağ üzerinde nerede bulundukları ile ilgilenmez. Bunu yerine getirebilmek için bazı ana gereksinimler vardır. Birincisi, uygulamalar

arasındaki iletişimi kolaylaştırmak için ortak bir dile ihtiyaç vardır. İkincisi, uygulamaların birbirlerine gönderecekleri istekleri düzenleyecek bir anlaşmaya ihtiyaç vardır. Üçüncüsü, uygulamalar arasında istekleri taşıyacak bir mesajlaşma çatısına ihtiyaç vardır. Son olarak, servis sağlayıcılarının sahip oldukları servisleri yayınlayacakları ve istemcilerin servisleri arayıp bulacakları bir katalog servisine ihtiyaç vardır. Aslında Web servisleri, bu gereksinimleri sağlayan teknolojiler topluluğu olarak görülmektedir. Şöyle ki, Web servisleri, ortak dil gereksinimi için XML (eXtensible Markup Language) (W3C, 1998), anlaşma gereksinimi için WSDL (Web Services Description Language) (W3C, 2001), mesajlaşma çatısı için SOAP (Simple Objects Access Protocol) (W3C, 2000) ve katalog servisi için ebXML (electronic business XML) Registry ve UDDI (Universal Description Discovery, and Integration) (UDDI.org, 2000) standartlarını kullanır.

CORBA, DCOM ve RMI gibi birlikte işlerliği sağlamak için geliştirilen dağıtık nesne sistemleri, sunucu uygulamalarını tanımlamak ve istemci uygulaması ile sunucu uygulaması arasındaki iletişimi sağlamak için belirtime özel arayüz tanımlama dilleri ve iletişim protokolleri kullanırlar. Örneğin CORBA da bir sunucu uygulamasının arayüzü, IDL dili kullanılarak tanımlanırken, sunucu uygulama ile istemci uygulama arasındaki iletişim IIOP (Internet Inter-ORB Potocol) (OMG, 2002) protokolü kullanılarak gerçekleştirilir. Benzer şekilde Java RMI de bir sunucu uygulamasının arayüzü Java dilinde tanımlanır ve istemci uygulama ile sunucu uygulama arasındaki iletişimi gerçekleştirmek için JRMP (Java Remote Method Protocol) (SUN, 2000) protokolü kullanılır. Söz konusu birlikte işlerlik sistemleri, platform ve/veya yazılım bağımlıdırlar. Örneğin RMI, sadece java programlama dili kullanılarak geliştirilen istemci ve sunucu uygulamalarının birlikte işlerliğini sağlar. DCOM, istemci ve sunucu uygulamalarının Windows platformu üzerinde koşmasını, CORBA ise istemci ve sunucu uygulamalarında aynı ORB (Object Request Broker) yazılımının kullanılmasını gerektirmektedir. Bu nedenle, bir CORBA uygulaması, bir DCOM veya bir RMI uygulaması ile birlikte çalışamaz.

Web servisleri ise, WSDL, SOAP gibi XML tabanlı standartları ve HTTP, SMTP ve FTP gibi standart internet protokollerini kullanarak birlikte işlerliği gerçekleştirirler. Bir servis sağlayıcısı, sahip olduğu Web servislerini WSDL dilini kullanarak tanımlar. WSDL, bir Web servisinin sahip olduğu operasyonları, operasyonların istek ve yanıt mesajlarını, mesajlar aracılığıyla taşınacak olan verileri ve tiplerini, Web servisini çağırmak için kullanılacak olan uygulama protokolünü ve servisin adresini tanımlayan XML tabanlı bir

dildir. Servis sağlayıcılar, Web servislerini katalog servisleri aracılığıyla yayınlarlar. Kullanıcılar, katalog servislerinde arama yaparak ihtiyaç duydukları Web servislerinin WSDL dokümanlarını elde ederler. Bir istemci uygulaması, bir Web servisini çağırmak için bir SOAP işlemcisi (SOAP processor/handler) kullanır. SOAP işlemcisi, servisin WSDL dokümanında tanımlanan formatta bir SOAP istek mesajı üretir. İstek mesajı, Web servisinin bir operasyonunu çağırmak için gerekli olan parametreleri içeren bir XML dokümanıdır. Operasyon parametreleri, SOAP işlemcisi tarafından istemci uygulamasının yazıldığı programlama dilindeki veri tiplerinden, XML şema (W3C, 2004a) veri tiplerine dönüştürülür. Oluşturulan istek mesajı internet üzerinden Web servisine gönderilir. Servis mesajları, HTTP, SMTP ve FTP gibi standart internet protokolleri üzerinden taşınır. Web servisinin koştuğu sunucudaki SOAP işlemcisi, gelen istek mesajını okur ve parametreleri Web servisinin gerçekleştirim kodunun yazıldığı programlama dilindeki veri tiplerine dönüştürerek servisin ilgili operasyonunu çağırır. Yanıt mesajının oluşturulmasında ve istemci uygulamasına gönderilmesinde yukarıdaki işlemler tekrarlanır.

1.8. Web Servisleri Tanımlama Dili

Web Servisleri Tanımlama Dili (Web Services Description Language / WSDL), Web servislerini tanımlamak için geliştirilen XML tabanlı bir dildir. WSDL belirtiminin ilk sürümü (version 1.0), Microsoft, IBM ve Ariba yazılım firmaları tarafından Eylül 2000 de geliştirilmiştir. Adı geçen yazılım firmaları, Mart 2001 de belirtimin ikinci sürümünü (version 1.1) geliştirmiş ve standart olarak kabul edilmesi için W3C ortaklığına sunmuştur. W3C, WSDL belirtimini aynı tarihte "W3C notu" (W3C Note) olarak yayınlamıştır (W3C, 2001). WSDL belirtiminin son sürümü (version 2.0), W3C tarafından Mart 2006 da yayınlanmıştır (W3C, 2006). WSDL belirtimi W3C tarafından henüz standart olarak kabul edilmemiştir. W3C nin WSDL çalışma grubu belirtim üzerindeki çalışmalarına devam etmektedir.

WSDL, Web servislerinin arayüzlerini tanımlayan, platformdan ve dilden bağımsız bir dildir. WSDL, herhangi bir programlama dili kullanılarak gerçekleştirilen ve herhangi bir platform üzerinden sunulan Web servislerini tanımlama yeteneğine sahiptir (Nagappan vd., 2003). WSDL, bir Web servisinin sahip olduğu operasyonları, operasyonların istek ve yanıt mesajlarını, mesajlar aracılığıyla taşınacak olan verileri (operasyon parametrelerini) ve tiplerini, Web servisini çağırmak için kullanılacak olan uygulama protokolünü ve Web

servisinin adresini tanımlar. Bir WSDL dokümanı, *definitions*, *types*, *message*, *portType*, *binding* ve *service* olarak adlandırılan altı element içerir.

Definitions elementi, bir WSDL dokümanının "kök" (root) elementidir ve WSDL dokümanı içerisinde yer alan elementlerin ve özniteliklerin "isim uzaylarını" (namespace) içerir. D*efinitions* elementinin "*targetNamespace*" özniteliği, WSDL dokümanının bir XML şema dokümanı gibi kendisine referans etmesine olanak tanır.

Types elementi, istemci ile Web servisi arasında mübadele edilen mesajlardaki parametrelerin tiplerini tanımlamak için kullanılan XML şema dokümanlarının "isim uzaylarını" içerir. WSDL, servis parametrelerinin tiplerini tanımlamak için, varsayılan (default) tip sistemi olarak, W3C tarafından geliştirilen tip sistemini (W3C, 2004a) kullanır. WSDL ayrıca, servis tanımlarında, kullanıcılar tarafından tanımlanan veri tiplerinin kullanılmasına da olanak sağlar.

Message elementi, istemci ile Web servisi arasında mübadele edilen mesajları tanımlar. Her bir *message* elementi, bir Web servisine gönderilen ya da Web servisi tarafından döndürülen bir istek ya da yanıt mesajını temsil eder. Mesajlar, bir operasyonun istek parametrelerini ya da operasyonun döndürdüğü değeri temsil eden bir ya da daha fazla "*part*" elementi içerebilir. Bir *part* elementinin "*name*" özniteliği bir parametrenin ya da döndürülen değerin ismini, "*type*" özniteliği ise tipini tanımlar.

PortType elementi, bir Web servisi tarafından desteklenen operasyonları soyut (abstract) olarak tanımlar. *PortType* elementi, bir ya da daha fazla "*operation*" elementi içerebilir. Her bir *operation* elementi Web servisi tarafından gerçekleştirilen bir operasyonu temsil eder. *Operation* elementinin "*name*" özniteliği operasyonun ismini, "*input*" ve "*output*" elementleri ise operasyonun istek ve yanıt mesajlarını tanımlar.

Binding elementi, belirli bir *portType* tarafından tanımlanan operasyonlar ve mesajlar için mesaj formatını ve protokol detaylarını tanımlar (Nagappan vd., 2003). İstemci ile Web servisi arasında hangi etkileşim modelinin (rpc/document) kullanılacağı, mesajları kodlamak için hangi kodlama stilinin (literal/encoded) kullanılacağı ve mesajları iletmek için hangi uygulama protokolünün kullanılacağı, *binding* elementi tarafından tanımlanır. Bir *portType*, birden fazla *binding* elementi tarafından tanımlanabilir.

Service elementi ise Web servisinin adresini tanımlar. Ek 1. de bir Web servisini tanımlayan örnek bir WSDL dokümanı yer almaktadır.

1.9. Basit Nesne Erişim Protokolü

Basit Nesne Erişim Protokolü (Simple Object Access Protocol / SOAP), XML dokümanlarının internet üzerinden uygulamalar arasında değiş tokuş edilmesine olanak sağlayan bir mesajlaşma protokolüdür. SOAP, DevelopMentor yazılım firması tarafından geliştirilmiştir. SOAP belirtiminin ilk sürümü (SOAP 1.0), RogueWave, IONA, ObjectSpace, Digital Creations, UserLand, Microsoft ve DevelopMentor yazılım firmaları tarafından 1999 yılında yayınlanmıştır. Belirtimin ikinci sürümü (SOAP 1.1), IBM ve Lotus yazılım firmalarının da katılımıyla 2000 yılında geliştirilmiş ve standart olarak kabul edilmesi için W3C ortaklığına sunulmuştur (Nagappan vd., 2003). W3C, Mayıs 2000 de SOAP 1.1 belirtimini "W3C Notu" olarak yayınlanmıştır (W3C, 2000). SOAP belirtiminin son sürümü (SOAP 1.2), W3C tarafından 2003 yılında yayınlanmıştır (W3C, 2003a).

SOAP, HTTP protokolü için bir genişletmedir (Newcomer, 2002) ve XML mesajlarının, HTTP istek ve yanıt mesajları ile taşınmasına olanak sağlamaktadır. SOAP mesajları, "*envelope*", "*header*" ve "*body*" olarak adlandırılan üç ana elementten oluşur. *Envelope,* bir SOAP mesajının kök elementidir. *Envelope*, *header* ve *body* elementlerini içerir ve bir HTTP mesajı içerisindeki bir SOAP mesajının başlangıç ve bitişini belirtir. *Header*, seçmeli bir elementtir. Bir SOAP mesajı, birden fazla *header* elementi içerebilir. *Header*, güvenlik bilgileri gibi bir uygulamaya özel bilgilerin SOAP mesajları ile taşınmasına olanak sağlamaktadır. *Header* elementinin içerdiği bilgilerin, SOAP mesajını alan sistem tarafından işlenip işlenemeyeceğini belirtmek için "*mustUnderstand*" özniteliği kullanılır. *Header* elementinin *mustUnderstand* özniteliğinin değeri "1" ya da "true" ise SOAP mesajını alan sistem (SOAP düğümü), SOAP mesajını ve *header* elementinde yer alan bilgileri işlemek zorundadır. *mustUnderstand* özniteliğinin değeri "0" ya da "false" ise, SOAP mesajını alan sistem mesajı başka bir SOAP düğümüne iletir.

Her SOAP mesajı bir adet *body* elementi içerir. *Body* elementi, bir SOAP mesajı tarafından taşınan verileri içerir ve bir operasyon çağrısını temsil eder. Bir istemcinin, bir Web servisini çağırmak için servise gönderdiği SOAP istek mesajı, çağırılan operasyonun adını, operasyonun girdi parametrelerini ve değerlerini içerir. Örneğin aşağıdaki XML dokümanı, bir Web servisinin, "Topla" isimli operasyonunu çağırmak için, bir istemci tarafından Web servisine gönderilen bir SOAP istek mesajını göstermektedir. Mesajın *body* elementi, operasyonun adını (Topla), operasyonun girdi parametrelerini (sayi1 ve

sayi2), parametrelerin tiplerini (float) ve parametrelerin değerlerini (20.25, 12.43) içermektedir.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <SOAP-ENV:Body>
      <Topla>
         <sayi1 xsi:type="xsd:float">20.25</sayi1>
         <sayi2 xsi:type="xsd:float">12.43</sayi2>
      </Topla>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Web servisleri tarafından istemcilere gönderilen SOAP yanıt mesajlarının *body* elementleri, istemciler tarafından çağrılan operasyonların çıktı parametrelerini ve değerlerini içerirler. Örneğin aşağıdaki XML dokümanı, bir SOAP yanıt mesajını göstermektedir. Mesajın *body* elementi, operasyonun çıktı parametresini (sonuc), parametrenin tipini (float) ve parametrenin değerini (32.68) içermektedir.

```
<?xml version="1.0"?>
<env:Envelope
   xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <env:Body>
      <ToplaResponse>
         <sonuc xsi:type="xsd:float">32.68</sonuc>
      </ToplaResponse>
   </env:Body>
</env:Envelope>
```

SOAP mesajları, Web servislerinin WSDL dokümanlarından yaratılırlar. İstemci uygulamalarının ve Web servislerinin yerleştirildiği ve sunulduğu, örneğin Apache ve Microsoft Internet Information Server (IIS) gibi HTTP sunucularının, SOAP mesajlarını işleyebilmeleri için bir SOAP işlemcisine (SOAP processor/handler) sahip olmaları gerekmektedir. Bir SOAP işlemcisinin görevi, bir Web servisinin WSDL dokümanında tanımlanan formatta SOAP mesajları yaratmak ve SOAP mesajlarını istemcilere veya Web servislerine göndermektir.

2. YAPILAN ÇALIŞMALAR

2.1. W3C Web Servisleri Mimarisinde E-Belediye Gerçekleştirimi

Belediye faaliyetleri, belediyenin gerek kendi birimleri arasında ve gerekse belediye ile kamu kurumları ve özel sektör arasında "birlikte işlerliği" (interoperability) gerektirmektedir. Ancak Türkiye'de mevcut işleyişte böyle bir birlikte işlerlikten söz etmek mümkün değildir. Çünkü Türkiye'de belediyeler, birkaç münferit pilot proje dışında, Kent Bilgi Sistemlerini kuramamışlardır. Diğer yandan belediyelerin iletişimde bulunduğu diğer kurumlar ve Tapu Kadastro bir türlü kendi bilgi sistemlerini kurabilmiş değildir. Özellikle Tapu Kadastro'nun bu eksikliği, bugüne kadarki süreçte Kent Bilgi Sistemlerinin önündeki en büyük engellerden biri olmuştur. Dolayısıyla belediyelerin gerek kendi birimleri ve gerekse diğer kurumlar ve Tapu Kadastro ile olan iletişimi ve veri alış-verişi hala, büyük ölçüde geleneksel yöntemlerle gerçekleşmektedir.

Geleneksel işleyişten kaynaklanan sorunlar, bu çalışmada Trabzon belediyesi örnek alınarak belirlenmiş ve sınıflandırılmıştır. Buna göre, geleneksel işleyişte büyük ekonomik kayıplar, yüksek maliyetler, oto-kontrol eksiliği, işlemlerin yavaşlığı ve servis kalitesinin düşüklüğü söz konusudur. Her ne kadar bu çalışmada Trabzon belediyesi örnek alınmışsa da, bu sorunlar aslında Türkiye'de kamu hizmetleri sektöründe, yıllardır genel şikayet konusu olan fakat bir türlü çözülemeyen sorunlardır.

Elektronik Devlet (e-devlet) ve Elektronik Belediye (e-belediye) ile hedeflenen, devlette ve belediyede geleneksel işleyişte yaşanan bu sorunların mevcut "Bilgi ve İletişim Teknolojileri" (BİT) kullanımı ile giderilmesidir. Dolayısıyla e-devlet, işlemleri hızlandıracak, hizmet kalitesini yükseltecek, maliyetleri düşürecek ve ekonomik kayıpları önleyecektir. Dahası e-devlet, BİT kullanımının sağladığı esneklik ve internet'in sunduğu geniş olanaklarla, yeni hizmetlerin ve dolayısıyla sektörlerin yolunu açacaktır. Dünya Bankası, e-devleti, devleti daha erişilebilir, etkin ve güvenilir bir yapıya dönüştürecek BİT kullanımı olarak tanımlamaktadır (World Bank, 2002). Yine Dünya Bankası'nın bir başka anlatımıyla e-devlet, devlet ile vatandaşlar (D-V), devlet ile özel sektör (D-Ö) ve devlet ile devlet (D-D) arasındaki iletişimi daha sıcak, uygun, şeffaf ve ucuz hale getirecektir.

E-devlet ve e-belediye gerçekleştirilmesi çok kolay hedefler değildir. Bunlar, çok sayıda teknik, kurumsal ve yönetimsel gereksinimin karşılanmasını gerektiren ve

dolayısıyla da uzun zaman isteyen projelerdir. Bu çalışmanın ilgi alanı, teknik gereksinimler ya da daha somut olarak, e-devlet ya da e-belediye'nin dayanacağı birlikte işlerlik altyapısıdır. E-devlet şu anda Dünya'da pek çok gelişmiş ve gelişmekte olan ülkenin gündeminde olmakla birlikte, söz konusu altyapı henüz netleşmemiştir. Bu çalışma ile hedeflenen, henüz çok yeni ve popüler bir teknoloji olan Web servislerinin, e-belediye birlikte işlerlik altyapısı için kullanılabilirliğinin araştırılmasıdır. Türkiye'de halen çeşitli firmalar belediyelere yönelik otomasyon sistemleri ve e-belediye tarzı uygulamalar geliştirmektedir. Ancak Web servisleri konusunun Dünya'da ve özellikle Türkiye'de daha çok yeni olması nedeniyle, literatürde Web servislerine dayalı her hangi bir e-belediye sistemine rastlanmamıştır. Bu bakımdan bir ilki teşkil eden bu çalışmanın, kendi alanında önemli bir katkı yapması beklenmektedir.

Web servisleri, birlikte işlerlik için bugüne kadar önerilen çözümlerden en geçerli, en basit ve en yeni olanıdır. En geçerli olmasından kasıt bugüne kadar önerilen sistemlerin uygulanmalarının zor olması ve belirli bir sistem içinde birlikte işlerlik sağlansa bile farklı sistemler arasında sorun çıkmış olmasıdır (Nagappan vd., 2003). Bu nedenle diğer birlikte işlerlik modelleri, bir belediye içerisinde çalışsa bile, belediyenin diğer kurumlar ile olan iletişiminde sorun çıkarabileceği nedeniyle benimsenmemiştir. Bir başka ifadeyle bu çalışmanın temel hedefi, yalnızca bir belediyeye yönelik değil, en az değişiklikle başka belediyeler için de uyarlanabilecek bir altyapının önerilmesidir. Bir o kadar önemlisi, bu altyapının aynı zamanda UKVA ve e-devlet için de geçerli olabilmesidir.

2.1.1. Geleneksel Belediyelerin Sorunları

Belediye faaliyetleri, belediyenin gerek kendi birimleri arasında ve gerekse belediye ile kamu kurumları ve özel sektör arasında birlikte işlerliği gerektirmektedir. Ancak, Türkiye'de geleneksel işleyişte böyle bir birlikte işlerlik söz konusu olmadığı için, çeşitli sorunlar mevcuttur. Bu sorunlar, bu çalışmada Trabzon belediyesi bazında belirlenmiştir. Bununla birlikte bu sorunlar D-V, D-Ö ve D-D sektörlerinde de aynen geçerlidir. Dünya Bankası, gelişmekte olan ve hatta gelişmiş ülkeler için benzer sorunlar tanımlanmıştır. Bu çalışmada belirlenen sorunlar, beş ayrı fakat birbiri ile ilişkili sınıfta toplanmıştır. Bunlar oto-kontrol mekanizmalarının eksikliği, büyük ekonomik kayıplar, yüksek servis maliyetleri, işlemlerin yavaşlığı ve düşük servis kalitesidir. Bu sorunlar aşağıda yalnızca özetlenmiştir. Bu konuda daha geniş bilgi için Şahin'e (2003) başvurulabilir.

Oto-kontrol: Oto-kontrol ile kastedilen, kurum içi ve kurumlar arası kontrol mekanizmalarının işletilmesidir. Ancak, kurum içi ve kurumlar arası birlikte işlerliğin eksikliği bunu olanaksız kılmaktadır. Oto-kontrol eksikliğinin en önemli sonuçlarından biri yüksek orandaki ekonomik kayıplardır. Bunun en çarpıcı örneği emlak vergilerinin toplanmasında yaşanmaktadır. Mevcut sistemde devlet kimden ne kadar vergi alacağını bilmemekte bu işi adeta mükellefe bırakmaktadır. Sonuç, %70 hatta %90 lara varan emlak vergisi kayıplarıdır (Ekinci, 1996). Bu konuda, Cömert ve Akıncı (2002) tarafından emlak vergilerinin eksiksiz toplanmasını sağlayacak yeni bir sistem önerilmiştir. EVBİS (Emlak Vergisi Bilgi Sistemi) olarak adlandırılan ve gerçekleştirimi de yapılan sistem, e-belediye'nin önemli bir bileşeni olacaktır.

Oto-kontrol aynı zamanda, verilerin güncelliği ve doğruluğu ile de ilgilidir. Oto-kontrol eksikliği, uygulamaların güncel olmayan veri kullanarak yanlış sonuçlar üretmesine yol açabilir. Geleneksel sistem, verinin güncellenmesi için gerekli teknik ve hukuki alt yapıdan yoksundur. Bu yüzden güncellemeler ya ihmal edilmekte ya da yardımcı arşivleme sistemleri ile güncellik sağlanmaya çalışılmaktadır. Örneğin imar planı güncellemelerinin yasal olarak planın yeniden çizilmesi yoluyla yapılması gerekir. Bu da, geleneksel yöntemlerle çok zor olduğundan, pratikte uygulanamamaktadır. Bunun yerine imar plan değişiklikleri ayrıca arşivlenir ve imar planında ilgili parsel üzerine değişiklik tarihi ve ilgili arşiv dosyasının numarası kurşun kalemle yazılır. Ancak özellikle büyük şehir belediyelerinde imar planı değişiklerinin bu şekilde çekip çevrilmesi hem çok zor hem de hata eğilimlidir. Öyle ki bazen değişikliklerin imar planına kaydedilmesi ihmal edilmiş olabilir. Bu da örneğin imar durumu verilmesi gibi, güncel veri gerektiren pek çok faaliyette yanlış sonuçlar üretilmesine neden olabilir.

Oto-kontrolün çok kritik diğer bir boyutu da saydamlık ve iyi yönetimdir. Dünya Bankası da (2002) bu duruma dikkat çekmiştir. Oto-kontrol kötü yönetim ve yolsuzluklarla savaşmak için gerekli mekanizmaların alt yapısını hazırlayacaktır.

İşlemlerin hızı: Geleneksel sistemde işlemler çok yavaş yürümektedir. Bunun bir nedeni, verinin arşivlenmesi, işlenmesi, analiz edilmesi ve sunulması işlevlerinin, hala geleneksel araçlarla yapılıyor olmasıdır. Diğer neden, birlikte işlerlik eksikliğidir. Nihayet faaliyetler için gerekli, başka birim ve/veya kurumlardan sağlanabilecek verilerin, vatandaştan istenmesidir. İşlemlerin hızı için çeşitli örnekler verilebilir. Bunlardan biri, aşağıda detaylı olarak incelenecek olan, imar durumu formlarının (İDF) hazırlanmasıdır. İşlemlerin yavaşlığı, diğer olumsuzlukları bir yana, kaynakların hızla ekonomiye

kazandırılması açısından da çok negatif bir etki yaratmaktadır. Örneğin inşaat ruhsatlarındaki gecikmeler doğrudan inşaat ve yan sektörlere kaynak aktarımını geciktirecektir.

Servis kalitesi: Servis kalitesi yalnızca vatandaşların memnuniyetsizliğine değil aynı zamanda ekonomik kayıplara da neden olmaktadır. Örneğin emlak vergisi kayıplarının önemli bir bölümü, emlak vergisi ödemelerinde yaşanan zorluklar nedeniyle, bu işin çoğu vatandaş tarafından ihmal edilmesi nedeniyledir. Burada ana problem aslında "servis anlayışından" kaynaklanmaktadır. Geleneksel işleyişte kendisine bir hizmet sunulacak olan vatandaş ta adeta bir "çalışan" olarak görülmektedir. Öyle ki, sayısal ortamda başka bir kurumdan doğrudan internet üzerinden elde edilebilecek bilgiler, vatandaştan istenmektedir. Örneğin tapudaki bir satış için vatandaştan yerin rayiç değerini belediyeden getirmesi istenmektedir. Oysa tapudan belediyeye bağlanacak olan görevli bu bilgiyi doğrudan belediye sitesinden alabilmeli ve hatta belediyeden elde edilecek bilgileri doğrudan kendi uygulamasına (satış işlemi) aktarabilmelidir.

Kamu kurumlarının birbirinden sağlayabilecekleri verileri vatandaştan istemeleri, vatandaşların yaygın şikayet konusudur ve çağdışı bir uygulamadır. Bu durum, Türkiye'de yukarıda anılan UKVA gibi bir altyapının henüz hayata geçirilememiş olmasından kaynaklanmaktadır. UKVA ile kurumlar ihtiyaçları olan verileri internet üzerinden birbirilerinden sağlayabileceklerdir. Bu durumda da vatandaş örneğin vergi beyannamesi vermek, ödeme yapmak vs. için uğraşmak, ödeme kuyruklarında zamanını boşa harcamak zorunda kalmayacak, yalnızca kendine gelen emlak vergisi faturasını internet üzerinden ya da diğer araçlarla ödeyebilecektir.

Servis kalitesinin diğer bir boyutu da bugün gelişmiş ülkelerde yaygın olarak kullanılmakta olan kamunun yönetime katılmasıdır. Gelişmiş ülkelerin çoğunda, çeşitli yönetim kararları ve projeler için vatandaşın görüşünü almaya yönelik siteler mevcuttur. Ülkemizde ise kamu katılımı bir yana, kamuya bir takım sonuçların sunulma şekli bile ilkeldir. Bunun çok karakteristik bir örneği imar planı uygulamaları sonucunda ortaya çıkan imar planı değişikliklerinin, "belediye panolarında" askıya çıkarılmasıdır. Bu panolardan vatandaşlar kendi parselleri ile ilgili bilgileri ancak zorlukla görebilmekte ve belki de vakit bulamadığında ilgili panoya gidip bakamamaktadır. Oysa bu bilgiler internet üzerinden ilgililerinin kullanımına sunulabilir.

Yüksek maliyetler: Geleneksel işleyişte maliyetlerin yüksek olmasının çeşitli nedenleri vardır. Bunlardan ilki, birlikte işlerlik eksikliği nedeniyle uygulama geliştirme

maliyetinin yüksek olmasıdır. Örneğin belediyeler ile diğer ilgili taraflar arasındaki veri alış verişi hala geleneksel yöntemlerle yapılmaktadır. Bu da oldukça zor ve zaman alıcı bir iştir. Ayrıca, faaliyetlerin yürütülmesinde belediyelerin kendi birimleri ve diğer kamu kurumlarından veri almayı gerektiren durumlarda internet kullanılmamaktadır. Bunun yerine, günümüzde ilkel sayılabilecek bir yöntemle, ilgili kurumlara bizzat gidilerek, veriler, çoğunlukla basılı harita ya da belge gibi "analog" bir formda alınmaktadır. Analog formda diğer birim ve kurumlardan alınan verilerin, sayısallaştırılarak belediye bilgi sistemine aktarılması hem maliyetleri yükseltmekte hem de faaliyetleri yavaşlatmaktadır. Veri alınacak kurumun il dışında olması durumunda ise, belirli bir personel bu iş için görevlendirilmekte ve bu personelin ulaşım ve diğer giderleri karşılanmaktadır. Bu duruma çok çeşitli örnekler verilebilir. Genel şikayet konusu olan örneklerinden biri, Ülkemizin temel veri sağlayıcı kurumlarından biri olan, HGK'nın veri sunma şeklidir. Buna göre, herhangi bir kurum HGK'dan veri sağlamak için, görevlendireceği bir personelini verileri almak üzere Ankara'ya göndermesidir. Belediyeler açından bir örnek ise, kıyılardaki yasal olmayan yapılaşmalarla ilgili denetimler için ihtiyaç duyulan kıyı kenar çizgisi haritalarının temini için, ilçe belediyelerinden bayındırlık il müdürlüklerine gidilmesidir. Bir diğer örnek ise, yukarıda da sıkça dile getirildiği gibi, tapu kadastro verilerinin, ilgili tapu ve kadastro müdürlüğüne bizzat gidilerek temin edilmesidir.

Diğer yandan, belediyelerde birçok faaliyet hala elle yürütülmekte, formlar elle doldurulmakta ve bilgiler geleneksel yöntemlerle arşivlenmektedir. Bu durum zaman alıcı ve hata eğilimli olması bir yana, gereğinden fazla personel istihdamını gerektirerek maliyetleri yükseltmektedir. Bu aslında yalnızca belediyelerin değil, Türkiye'deki kamu kurumlarının genel sorunudur.

Ekonomik kayıplar: Türkiye'de ekonominin kötü gidişi, yerel yönetimlere ayrılan bütçenin yetersizliği, belediye hizmetlerinin yürütülmesini olumsuz etkilemektedir. Halen birçok belediyede hizmetlerin yürütülmesi ve personel giderlerinin ödenmesi konusunda sorunlar yaşanmaktadır. Belediyeler, bu sorunları aşmak için yürüttükleri faaliyetlerde ekonomik kayıplarını en aza indirmek ve gelir getirecek yeni hizmetleri üretmek durumundadır.

Belediye faaliyetlerinde ekonomik kayıplar oto kontrol eksikliği, işlemlerin yavaşlığı ve hizmet kalitesinin düşüklüğü nedeniyle olabilir. Oto kontrol eksikliğinden kaynaklanan kayıplara en çarpıcı örnek emlak vergisinde yaşanan yüksek orandaki kayıplardır. Bu durum, ekonomik krizlerden bir türlü kurtulamayan ve halen de acil ek kaynak ihtiyacında

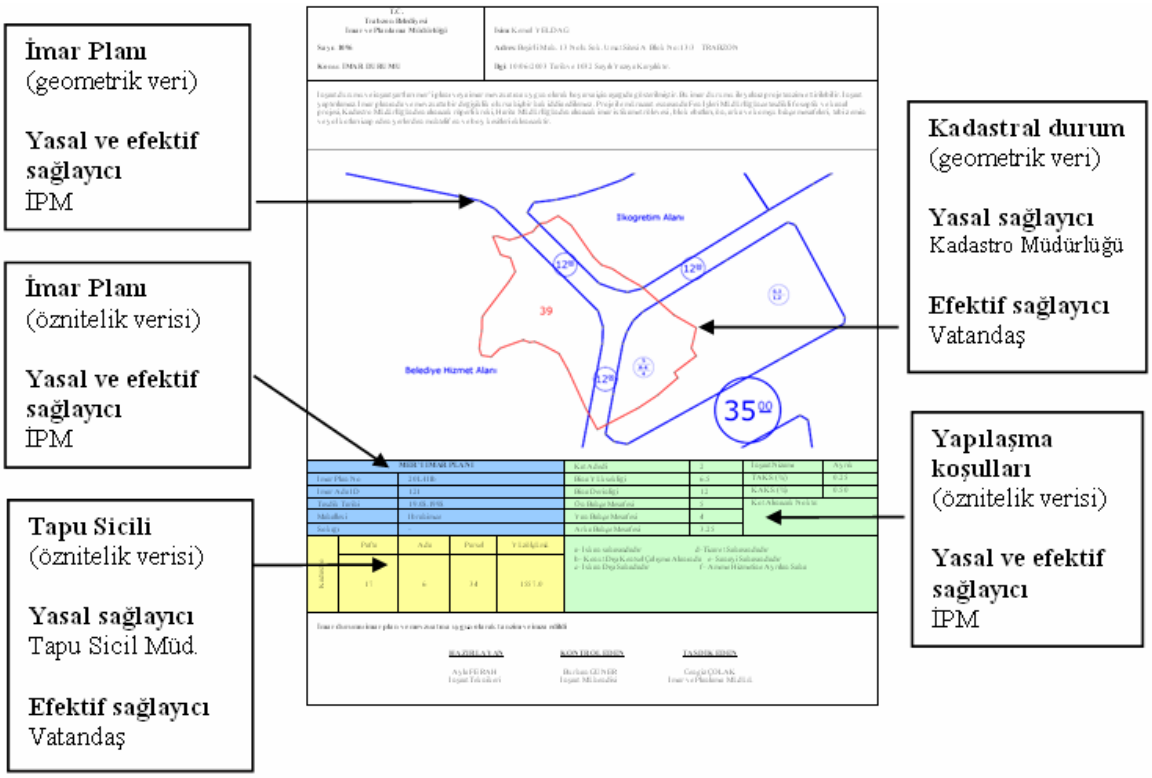
olan Ülkemiz açısından son derece önemlidir. Ayrıca emlak vergilerindeki bu kayıpların önlenmesi, Ülkemizde hep önerilen fakat bir türlü uygulamaya konamayan, vergi adaleti ve verginin tabana yayılması için de kaçınılmazdır. Emlak vergisi kaybının nedenleri yukarıda açıklanmıştı.

Diğer yandan faaliyetlerin yavaş yürümesi de ekonomik kayıplara neden olmaktadır. Buna karakteristik bir örnek, Trabzon belediyesi revizyon imar planı sürecidir. Revizyon imar planı yapımına Şubat 2002 de başlanmış, plan belediye meclisinde Mart 2003'te kabul edilmiştir. Sürenin bu kadar uzun olması, faaliyetin yürütülmesinde BİT kullanımının yetersiz oluşundan kaynaklanmıştır. Örneğin bu iş için önce kadastro paftalarının sayısallaştırması gerekmiştir. Ayrıca revizyon imar planını hazırlayacak olan özel firmaya sağlanan halihazır haritalar 1995–96 yıllarına ait olduğu için bunların güncellenmesi de zaman almıştır. Planın yapımına başlanan Şubat 2002 ile belediye meclisinde Mart 2003'te kabulü arasındaki süreçte belediye imar planını ile ilgili tüm uygulamaları durdurmuştur. Yani bu süre içerisinde imar durumu verilmesi, yapı ruhsatı verilmesi, ifraz ve tevhit gibi imar uygulamalarının tümü yapılamaz hale gelmiştir. Bu, kaynakların ekonomiye aktarılması açısından son derece olumsuz bir etki yapmıştır. Diğer yandan bu durum, belediyenin faaliyetler kapsamında topladığı harç gelirlerinin de kesilmesine neden olmuştur. İnşaat faaliyetlerinin azalması, özel sektörün yeni projeler alamaması gibi faktörler piyasayı hizmet ve mal alımı yönünden ekonomik olarak etkilemektedir.

2.1.2. Geleneksel Belediyelerden Örnek Bir Uygulama

İmar durumu formu (İDF), bir parsele ait tapu ve kadastro verileri ile imar planı verilerini içerir. Parselin imar planındaki konumunu grafik olarak gösterir ve yapılaşma koşullarını listeler. Yani İDF, grafik ve metin bilgiler içerir (Şekil 3). Belediyelerin, kamu kurumlarının ve özel sektörün pek çok uygulaması için İDF bilgilerine ihtiyaç vardır. Örneğin inşaat ruhsatı almak isteyen bir ilgili, İDF ile işe başlar. İDF nin oluşturulması, Tapu ve Kadastro kurumundan sağlanacak tapu kadastro verileri ile belediye İmar Planlama Müdürlüğü (İPM) den sağlanacak imar planı verilerinin aynı bir belgede toplanmasını içerir. Dolayısıyla, burada söz konusu olan farklı kurumlar tarafından çekip çevrilen verinin birleştirilmesidir. Şekil 3, bir İDF yi, gerekli verileri, bu verilerin yasal ve efektif sağlayıcılarını göstermektedir. Efektif sağlayıcı, uygulamada bu veriyi belediyeye

sağlayan, yani İDF talebinde bulunan ilgili ya da vatandaştır. İDF'ler, belediyelerin İPM birimleri tarafından "ilgili" nin isteği üzerine hazırlanır. İlgili, her ne kadar bir kamu kurumu, özel sektör, başka bir belediye birimi ya da vatandaş olabilirse de, burada vatandaş kastedilecektir.



Şekil 3. İDF verileri, yasal ve efektif veri sağlayıcılar

Bir vatandaşın bir belediyeden İDF alma senaryosu Şekil 4 de gösterilmiştir. Şekli özetlemek gerekirse, senaryo aşağıdaki gibidir:

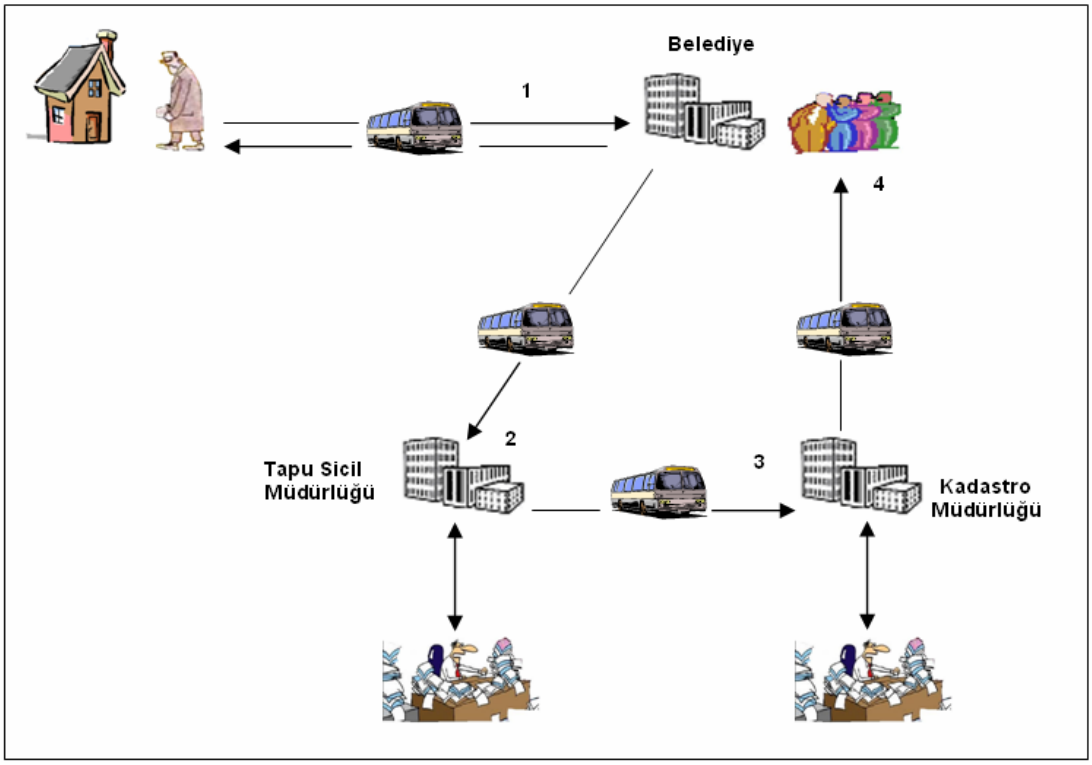
1. Vatandaş, belediye İPM'ye giderek parseline ait İDF yi talep eder. İPM kendisinden tapu ve kadastro bilgilerini ister.

2. Vatandaş bunun üzerine tapu sicil müdürlüğüne giderek, tapu bilgilerini talep eder. İzlenmesi gereken prosedür ve işlemlerin yavaşlığı nedeniyle bu iş vatandaş için "bunaltıcı" olabilir. Müdürlük görevli personeli açısından, ilgili kayıtların genellikle bir veri tabanı ara birimi yardımıyla bulunması çok da kötü sayılmayabilir.

3. Vatandaş, kadastro bilgileri (kadastro çapı) için kadastro müdürlüğüne gider. Kadastro müdürlükleri, bazı şehirlerde olduğu gibi, şehrin farklı yerlerinde iseler, vatandaşın bu iş için ayrı bir ulaşıma ihtiyacı vardır. Aynen yukarıda anıldığı gibi bu işlem

de uzun yasal prosedür gerektirir, yavaştır ve dolayısıyla vatandaş adına "can sıkıcıdır". Müdürlük görevli personeli için ise, ilgili paftanın bulunması ve bu paftadan parselin olduğu bölgenin "elle" çizilmesi zaman alıcıdır ve hata eğilimlidir. Müdürlüğün yoğunluğuna da bağlı olarak, vatandaşın belgeyi aynı gün alma şansı oldukça düşüktür.

4. Kadastro çapı ve tapu senedi bilgilerini toplamasının ardından vatandaş tekrar belediye İPM ye gider. Topladığı bilgileri İPM ye sunar. Ancak çoğu zaman İDF yi hemen alamaz. Çünkü İDF ler halen elle ya da yarı-otomatik yöntemlerle verilmektedir. Her iki işlem de zaman alıcı olduğundan, vatandaşa genellikle İDF yi almak için birkaç gün içinde geri gelmesi söylenir.

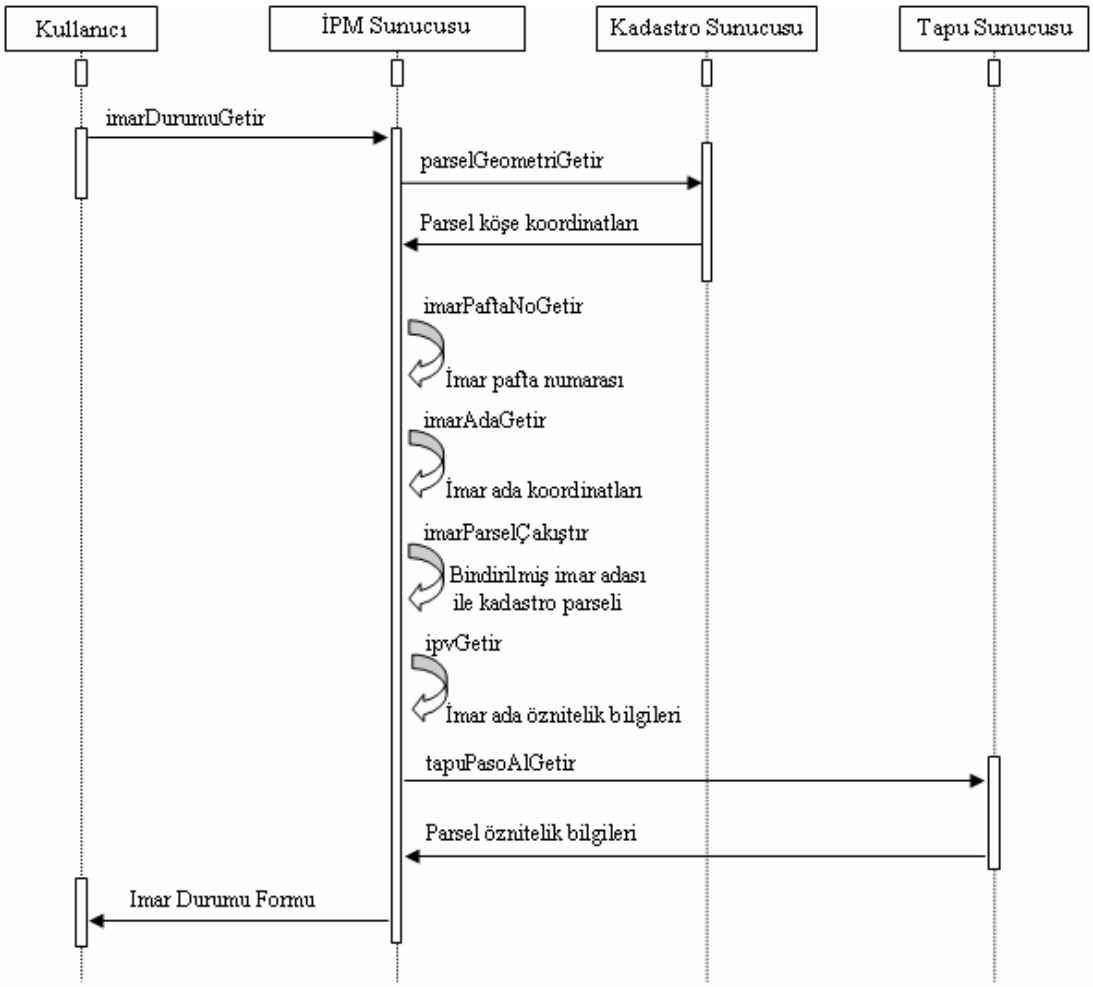


Şekil 4. Bir vatandaşın geleneksel belediyeden İDF alma serüveni

Özetle, iyimser bir tahminle, geleneksel bir belediyeden bir İDF almak isteyen vatandaşın bütün bir haftasını bu iş için harcayacağı söylenebilir. Bu, hiçbir şekilde kabul edilemeyecek bir hizmet sunum tarzıdır. Diğer yandan, yukarıda anılan problemler, bu örnekte belirgin olarak görülebilir; Öyle ki, işlemler çok yavaştır, maliyetler yüksektir ve servis kalitesi çok düşüktür. Problemin kaynağı yine veri sağlayıcı birim ve kurumlar arasındaki birlikte işlerlik eksikliğidir.

2.1.3. E-Belediye İçin Web Servisleri

E-belediye, belediyenin kurum içi, kurum dışı ve vatandaş ile ilgili her türlü iletişim, iş üretimi ve hizmet sunumunun elektronik ortamda gerçekleştiği bir belediye olarak tanımlanabilir. Şahin (2003), Trabzon Belediyesi'nin özellikle coğrafi veri içeren faaliyetleri incelenmiş ve bu faaliyetler bazında, e-belediye'ye yönelik bir dizi Web servisi belirlemiştir. Bu servisler ayrıca girdi ve çıktı parametreleri ile de belirtimlendirilmiştir. Bu servislerin bir kısmı, Cape Clear (2003) Web servisleri geliştirme ve kurma yazılımı kullanılarak gerçekleştirilmiştir. Tanımlanan servisleri kullanarak yeni bir servis geliştirme, çok kısa bir zamanda yapılabilmektedir. Örneğin ilgili servisleri kullanarak İDF hazırlayacak bir *imarDurumuGetir* servisinin geliştirilmesi, ilgili bütün servislerin hazır olması durumunda, yaklaşık on beş dakika almıştır. *İmarDurumuGetir,* İPM nin bir servisi olacak, gereksinim duyduğu verileri, ilgili Web servisleri vasıtasıyla, kendi veri tabanından ve diğer kurumların veri tabanından sağlayarak İDF yi üretecektir. Bir başka anlatımla, kendisi de bir Web servisi olan *imarDurumuGetir*, işini görmek için başka Web servislerini çağırır (Şekil 5).

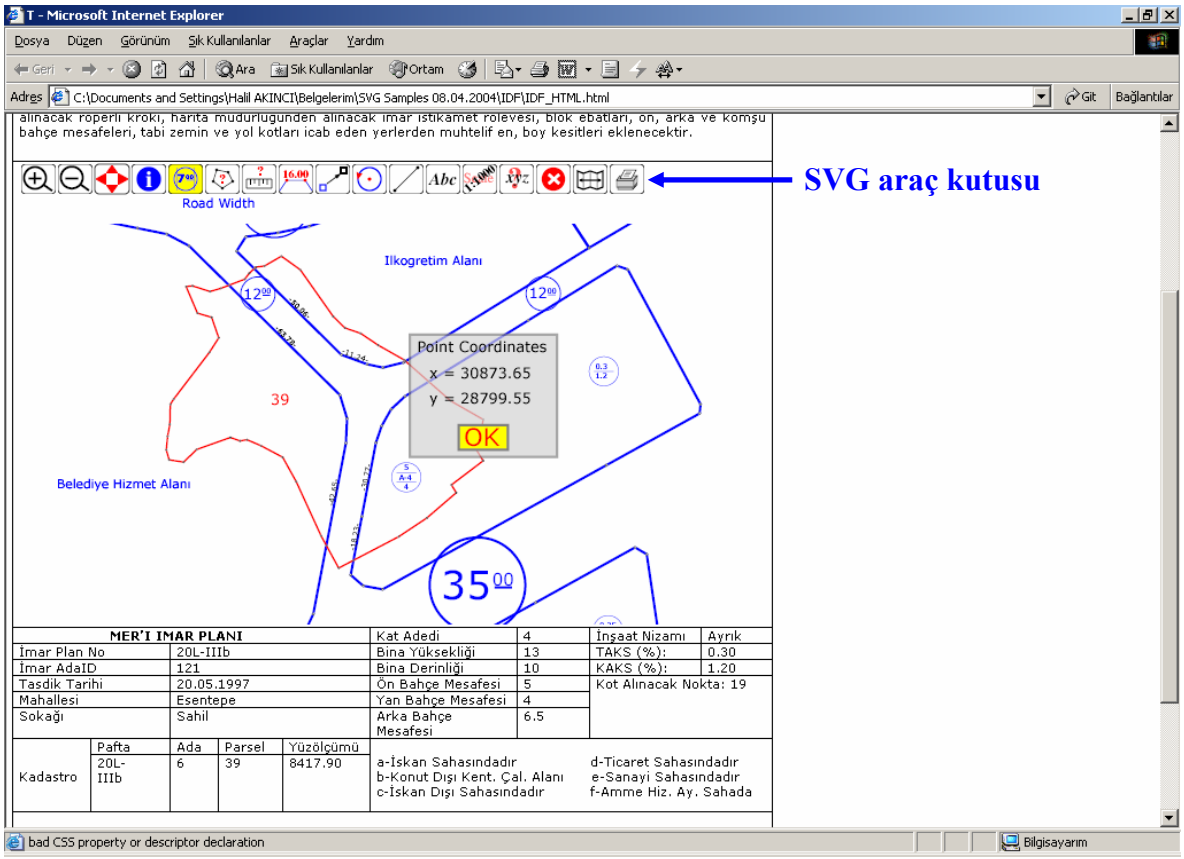


Şekil 5. İmarDurumuGetir web servisinin UML iş akış diyagramı

İPM görevlisi, *imarDurumuGetir* servisini kendi internet tarayıcısından çağırır ve parselin *taşınmazID* sini HTML formundan girer. *İmarDurumuGetir* Web servisi, parsel geometrisi için kadastro sunucusundan *parselGeometriGetir* servisini çağırır. Ardından ilgili imar adalarını getirecek olan *imarAdaGetir* servisini çağırır. Her iki servis de birer vektör harita sunar. Bu haritalar İPM nin kendi sunucusunda bulunan *imarParselÇakıştır* servisi tarafından çakıştırılır ve İDF nin grafik kısmı hazırlanmış olur. *İmarParselÇakıştır*, iki vektör haritanın çakıştırma işlemine ek olarak GML (Geography Markup Language) (OGC, 2003d) verisini SVG (Scalable Vector Graphics) (W3C, 2003b) verisine dönüştürür. *İmarDurumuGetir* servisi, İDF de yer alan tapu sicili ve imar planı verileri için İPM sunucusundan *ipvGetir* ve tapu sunucusundan *tapuPasoAlGetir* servislerini çağırır ve Şekil 6 da görülen İDF yi üretir.

Web servislerine dayalı bir e-belediye'de vatandaşın bir İDF alma senaryosu aşağıdaki gibi özetlenebilir:

1. Vatandaş, belediyeye bizzat giderek ya da internet üzerinden belediye İPM ye başvurarak bir İDF talep eder.
2. İPM görevlisi internet tarayıcısından *imarDurumuGetir* servisini çağırır.
3. *İmarDurumuGetir* servisi diğer Web servislerini çağırarak İDF yi üretir.
4. İPM görevlisi İDF çıktısını yazıcısından alır ve vatandaşa sunar. Vatandaş eğer internet üzerinden belediyeye bağlanmışsa ve yetkisi dahilinde ise İDF çıktısını kendi bilgisayarından da alabilir.



Şekil 6. İmarDurumuGetir web servisi ve ürettiği İDF

Bu senaryoda bir vatandaşın belediyeden bir İDF alması, yeterince hızlı bir iletişim hattı ile saniye ya da daha kötüsü, dakikalar içerisinde mümkün olur. Bu, yukarıda incelenen geleneksel yönteme göre olağanüstü bir iyileştirmedir. Bu örnekten hareketle,

Web servislerinin geleneksel belediyelerin yukarıda söz edilen sorunlarına getirdiği çözümler bağlamında aşağıdakiler söylenebilir.

Web servisleri, hızlı hizmetleri olanaklı kılar. Vatandaş ve diğer ilgililerin istekleri saniyeler içerisinde karşılanabilir. Bu da servis kalitesini çok büyük oranda iyileştirir ve gelirleri arttırıcı bir etki yaratır.

Web servisleri ile uygulama geliştirme ve hizmet sunma maliyetleri düşecektir. Çünkü bir kere temel servisler tanımlandıktan sonra, yeni servisler geliştirmek son derece basit ve hızlıdır. Yukarıda da belirtildiği gibi *imarDurumuGetir* servisinin geliştirilip kurulması, 15 dakika gibi bir zamanda mümkün olmuştur. Diğer taraftan veri transferi maliyetleri düşecektir. Çünkü geleneksel uygulamalardaki gibi bir dosyanın tümünün transfer edilmesinin aksine, Web servisleri yaklaşımında yalnızca ihtiyaç duyulan verinin transferi söz konusudur.

Kamu kurumları ve belediyelerin "kaliteli" hizmetler sunmasının çok anlamlı bir sosyolojik boyutu vardır. Böylesi hizmetler alan bir ülkenin insanları kendilerine değer verildiğini hissederek, ülkelerinin bir vatandaşı olmakla gurur duyacak, mutlu ve nihayet her alanda daha "üretken" olacaklardır.

Gerek belediyelerin ve gerekse kamu kurumlarının geleneksel işleyişten kaynaklanan örneğin ekonomik kayıplar gibi diğer sorunları ise ancak Web servislerinin de içinde yaşayacağı ve gelişeceği bir ortamın tesisi ile mümkün olacaktır. Web servisleri herkesin başkaları için servisler sunacağı bir ortam için öngörülmüştür. Bu nedenle Web servislerine dayalı bir e-belediye ya da e-devlet modelinin işleyebilmesi ancak, ilgili bütün tarafların en azından veri tabanlarını işlevsel hale getirmeleri ve kendi Web servislerini kullanıma sunmaları ile mümkün olacaktır. Örneğin, e-belediye için tapu kadastro kurumunun kendi veri tabanlarını kurması ve belediyeler dahil bir çok kamu kurumu ve özel sektör için Web servisleri sunması hayati önem taşımaktadır. Burada ihtiyacı duyulan aslında UKVA dır (Cömert ve Banger, 1995). UKVA, tapu kadastro dahil coğrafi veriyi bir şekilde kullanan bütün kamu kurumları, özel sektör, belediyeler, üniversiteler, vatandaşlar ve başka kurumların birbirlerine veri ve servisler sunmasını sağlayarak, e-belediye ve e-devlet için gerekli altyapıyı oluşturacaktır. Sonuç olarak, Web servislerine dayalı bir e-belediye'de işlerin hızlanacağı, maliyetlerin düşeceği, ekonomik kayıpların önleneceği ve servis kalitesinin yükseleceği görülmektedir. Ancak bu hedeflere ulaşılabilmesi için, UKVA'nın bir an önce hayata geçirilmesi ve belediyeler dahil tüm UKVA taraflarının kendi Web servislerini diğer tarafların kullanımına sunması gerekir.

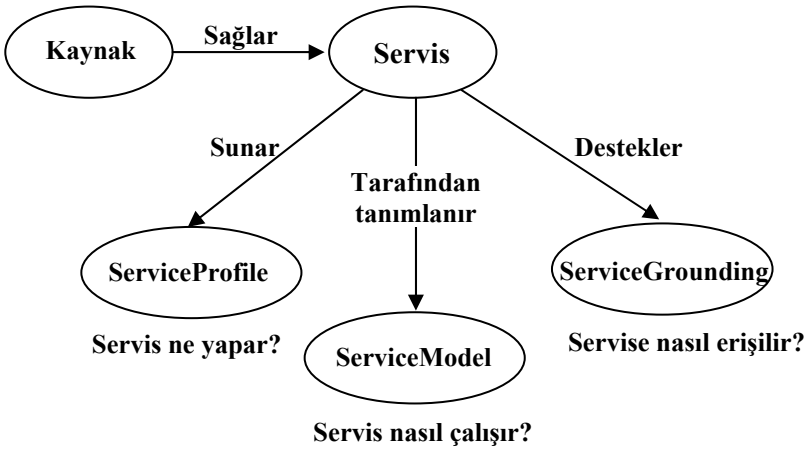
2.2. Servis Tanımlama

2.2.1. Web Servisleri Tanımlama Dili

Web Servisleri Tanımlama Dili (Web Services Description Language / WSDL), Web servislerini tanımlamak için geliştirilen XML tabanlı bir dildir. WSDL, Microsoft, IBM ve Ariba yazılım firmaları tarafından geliştirilmiştir. WSDL, bir Web servisinin sahip olduğu operasyonları, operasyonların istek ve yanıt mesajlarını, mesajlar aracılığıyla taşınacak olan verileri (operasyon parametrelerini) ve tiplerini, Web servisini çağırmak için kullanılacak olan uygulama protokolünü ve Web servisinin adresini tanımlar. Bir WSDL dokümanı, *definitions*, *types*, *message*, *portType*, *binding* ve *service* elementlerinden oluşur. Söz konusu elementler, birinci bölümde 1.8 başlığı altında açıklanmıştır.

2.2.2. OWL-S Anlamsal Web Servisleri Tanımlama Dili

OWL-S (W3C, 2004c), anlamsal Web servislerini tanımlama dilidir (Paolucci vd., 2004). Web servislerinin bir yazılım tarafından otomatik olarak bulunması, düzenlenmesi ve yürütülmesi amacıyla geliştirilmiştir (W3C, 2004c). OWL-S de bir Web servisi, servis profili (service profile), işlem modeli (process model) ve servis referansı (service grounding) olarak adlandırılan ve servisin farklı özelliklerini tanımlayan üç ayrı doküman tarafından tanımlanır (Şekil 7).



Şekil 7. OWL-S servis tanımı dokümanları (W3C, 2004c).

Servis profili, bir Web servisinin yeteneklerini tanımlar ve servisin bir yazılım tarafından otomatik olarak bulunmasını sağlar. Bir Web servisinin servis profili tanımında, servisin sağlayıcısı ile ilgili bilgiler ve servisin işlevsel ve işlevsel olmayan özelliklerini tanımlayan bilgiler yer alır. Sağlayıcı bilgileri, Web servisini sunan organizasyonla ilgili iletişim bilgilerini içerir. Servisin işlevsel özellikleri, servisin yeteneklerini tanımlayan girdi, çıktı, önkoşul ve etki parametreleridir. Servisin işlevsel olmayan özellikleri ise, servisin belirli bir taksonomideki kategori bilgileri ve servisin kalitesi ile ilgili bilgilerden oluşur. Servis profili ayrıca, bir servis ile ilgili işlevsel olmayan herhangi bir bilginin sağlayıcılar tarafından servis profili tanımına eklenmesine olanak sağlayan bir elemente (*serviceParameter*) sahiptir. Bir servis sağlayıcısı, *serviceParameter* elementini kullanarak servis profiline, örneğin servisin maksimum yanıt süresini veya servisin kullanılabileceği coğrafyayı tanımlayan bir bilgi ekleyebilir.

İşlem modeli, bir Web servisinin nasıl çalıştığını tanımlar. İşlem modeli, bir servisi bir işlem (process) olarak görür. İşlem modeli, üç tür işlem içerebilir. Bunlar, atomik işlem (AtomicProcess), basit işlem (SimpleProcess) ve birleşik işlem (CompositeProcess) dir. Bir Web servisi işlem modelinde ya bir atomik işlem ya da bir birleşik işlem olarak tanımlanır. Bir birleşik işlem, *Sequence*, *Any-Order*, *Choice*, *If-Then-Else*, *Iterate*, *Repeat-While* ve *Repeat-Until* gibi kontrol yapıları kullanılarak bir araya getirilen atomik veya birleşik işlemlerden oluşur. Bileşik işlem bir servis düzenlemeyi tanımlar. Bu nedenle işlem modeli, servis düzenleme dili olarak ta tanımlanmaktadır (Şirin vd., 2005). Herhangi bir içyapıya sahip olmayan (veya başka bir işlem içermeyen) bir işlem, atomik işlem olarak adlandırılır. Basit işlem ise, mevcut bir atomik veya birleşik işlemin soyut görünüşünü tanımlamak için tasarlanmıştır. Basit işlem, somut işlemleri tasarım zamanında önceden bilinen atomik veya birleşik işlemleri tanımlamak için kullanılır ve servislerin profil tanımlarına referans eder.

Servis referansı ise, bir Web servisine nasıl erişileceğini tanımlar. Bir Web servisinin referans tanımı, servisin WSDL dokümanına işaret eder. OWL-S dili kullanılarak tanımlanan bir anlamsal Web servisini yürütecek olan OWL-S işlemcisi, servisin adresi, kullanılacak olan iletişim protokolü ve mesaj formatı gibi servisin nasıl yürüteceği ile ilgili detayları, servisin referans tanımından ilgili WSDL dosyasına erişerek öğrenir. Bir servisin işlem modelinde tanımlanan her bir atomik işlem, servisin referans tanımında, referans edilen WSDL dokümanında tanımlanan bir operasyona resmedilir. Atomik işlemlerin girdi

ve çıktı parametreleri ise, WSDL dokümanında ilgili operasyonun girdi ve çıktı mesajlarına resmedilir.

OWL-S, servis tanımlama ile ilgili iki önemli kısıtlama getirmiştir. Buna göre bir Web servisi, en fazla bir işlem modeli tarafından tanımlanır ve bir referans sadece bir servis ile ilişkili olmak zorundadır.

2.3. Servis Kataloglama

2.3.1. Servis Katalogları ve Katalog Servisleri

Servis Katalogları (SK), SYM'nin kilit bileşeni durumundadır. Çünkü SYM'de servis sağlayıcılar Web servislerini katalog servisleri üzerinden kullanıma sunar, istemciler de aradıkları özellikteki Web servislerini bu sayede tespit ederler. Diğer bir anlatımla katalog servisi, sağlayıcıların Web servislerini katalog servisine kaydetmelerine ve istemcilerin istenen özellikleri sağlayan Web servislerini aramalarına olanak tanır. Burada söz konusu olan iki işlem, sırasıyla, SYM'nin meşhur "yayınla" (publish) ve "bul" (find) işlemleridir. Ancak Colan (2004) tarafından da işaret edildiği üzere, mevcut WS uygulamalarında SK bileşeninin genellikle eksik olduğu görülmektedir. (W3C, 2004d) gibi W3C'nin ilgili dokümanlarında bile konu yeterince detaylandırılmamıştır. Diğer bir anlatımla, istemcinin aradığı özellikteki servisleri bir şekilde "bulmuş olduğu" kabul edilmektedir. Oysa SYM'nin potansiyelleri ve sorunları bakımından, gerek teorik ve gerekse pratik olarak değerlendirilebilmesinde, onun en temel bileşeni olan katalog boyutundaki değerlendirmeler son derece kritik bir rol oynayacaktır.

SYM ve onun halen en geçerli gerçekleştirim şekli olan Web servisleri zaten, bütün işlerin Web servisleri ile yürütüldüğü işleyişler için öngörülmüştür; Çeşitli sunucuların Web servislerini sunduğu ve istemcilerin de bu servisleri kullanarak uygulamalarını gerçekleştirdiği bir işleyiş. Bu açıdan, Ryman'ın (2000) hava alanında "check-in" yaptırırken biletinin, bir şekilde birden çok müşteriye satılmış olduğunu fark eden yolcunun, biletinin "doğrulanmış" olup olmamasına ve kendi seyahat planına göre, yeni bilet arama ya da doğrulanmış biletini açık artırmada satma işlemlerini, Web servisleri aracılığı ile gerçekleştirebilme örneği, çok iddialı gibi görünse de WS ile hedeflenen ortamdır. Kullanıcıların Web servisleri ile böyle "iddialı" uygulamalar gerçekleştirebilmeleri için, çok yaygın bir servisler ağı ile desteklenmiş olmaları gerekir.

Böyle bir ortam buradan itibaren “Yaygın WS Ortamı (YWSO)” olarak anılacaktır. Anılan örnekte en azından havayolları, otel ve oto kiralama sektörlerinin Web servislerine ihtiyaç olacaktır. Benzer şekilde UKVA‘da da çok sayıda servis ve bunların sağlayıcıları söz konusu olacaktır. Bu durumda bir istemci gereksinimlerini karşılayacak özelliklerdeki bir ya da daha çok servisi nasıl bulacaktır? Bunun ne gibi boyutları vardır? Bu bağlamda katalogların önemi nedir? gibi sorular bu bölümde ele alınacak konunun özünü teşkil etmektedir.

Gerçekten de WS vizyonunda kritik nokta, istemcilerin aradıkları özellikleri sağlayan servisleri bulmaları ile ilgilidir. Aranan servis ya da servisler bulunduktan sonra geriye yalnızca servisin icrası ya da birden çok servis olması durumunda, servislerin belirli bir düzenleme ile icrası kalmış olacaktır. O halde servis bulmanın ne içerdiğine bakmak gerekir. Servis bulmanın her şeyden önce bir “sözdizimsel (syntactic)” bir de “anlamsal (semantic)“ boyutu vardır. Daha sonra, “bir servisi ne gibi özellikler tanımlar?” sorusu ile alakalıdır. Servisin örneğin hangi tip veriler için tanımlı olduğu bilgisi, servis tanımlayıcı bir özellik olarak kullanılmalı mıdır? Bir diğer boyut YWSO da, servislerin nasıl kataloglanacağıdır. Servisler merkezi bir katalog da mı depolanacaktır yoksa birden çok katalog mu olmalıdır? Birden çok ve farklı katalog kullanılması durumunda birlikte işlerlik sağlanabilecek midir? Bütün bu sorunlara yönelik ne gibi çözümler ve araçlar mevcuttur ve bunlar bir çerçeve de değerlendirilebilir mi?

Web Servisleri alanında Dünya genelindeki müthiş aktiviteye rağmen, ne konumsal WS ne de çok daha geniş bir kitle oluşturan konumsal olmayan WS alanında, YWSO lerin nasıl gerçekleştirileceği konusunda üzerinde görüş birliği sağlanmış bir gerçekleştirim modeli mevcuttur. Gerçekleştirim modeli bir yana, anılan sorunların bütüncül bir yaklaşımla ele alınmasını sağlayacak bir çatı bile mevcut değildir. Öyle ki WS alanında belirtim ve standart geliştirme ile ilgili çeşitli uluslararası kuruluşun (W3C, OASIS, WSMO, OGC) her birinin kendi çalışmaları vardır. Bunların hangilerinin yaygın kabul göreceği, farklı standartlar arasında nasıl uyum sağlanacağı henüz belirsizdir. Akademik çalışmaların çoğu ya çok genel ya da çok özeldir. Konumsal ve konumsal olmayan WS alanda faaliyet gösteren yazılım firmalarının kendi Web servisleri geliştirme ve yayma ortamlarının birlikte işlerliği henüz test edilmemiştir. Diğer yandan da piyasada çeşitli gerçekleştirimler mevcuttur. Bu durumda WS yönelimli KVA gerçekleştiricileri nasıl bir yol izlemelidir?

Anılan sorunların basitleştirici ve bütüncül bir yaklaşımla ele alınmasına yardımcı olacak bir çatıya yönelik olarak kataloglama bağlamında bir takım ölçütler belirlenebilir mi? noktasından hareketle bu çalışmada bir dizi ölçüt belirlenmiştir. Konumsal veri ve servis boyutlarını içeren böyle bir çalışma mevcut değildir. Bununla birlikte, konumsal olmayan veri ve servislere yönelik bazı çalışmalar mevcuttur. Paolucci vd. (2004) de bu eksikliğe dikkat çekmektedir. Az sayıdaki çalışmalardan biri olan (W3C, 2004d), Web servisleri mimarisine yönelik olarak bir dizi gereksinim belirlemiştir. Bunların bir kısmı katalog gereksinimleri ile aynıdır. Dustdar ve Treiber (2005), bu noktadan hareketle (W3C, 2004d) ölçütlerini genişletmişler bazılarını da çıkararak 12 ölçüt önermişlerdir. Bu ölçütler oldukça tutarlı bir çerçeve çizmekle birlikte, bu tez çalışmasının amaç ve kapsamını aşan "güvenlik" ve "sağlamlık (fault tolerance)" gibi ölçütler içermekte, bu tez çalışması için çok gerekli olan federasyon gibi konularla ilgili ölçütler bakımından da eksik durmaktadır. Çünkü söz konusu çalışmada tek katalog değerlendirmesi üzerinde durulmuştur. Ayrıca, Dustdar ve Treiber (2005), "bakış noktası" bazlı bir değerlendirme uygulamış ve "insan bakış açısı" ve "Web servisi bakış açısı" itibarı ile bir değerlendirme yapmıştır. Ardından, bu bakış açılarının da "mimari" ve "veri modeli" gibi iki farklı boyuta işaret ettiklerini belirtmiştir. Ancak bu bakış açılarının seçimi gerekçelendirilmemiştir. Najmi'nin (2005) LDAP, UDDI, ebRIM ve ISO 11179 standartlarını karşılaştıran, "yetenekler matrisi" çalışması, bu tez çalışmasının amacına göre çok detaylı olması bir yana, tümüyle birbirini dışlayan bir sınıflandırma içermemektedir. Bunun bir nedeni, çalışmanın henüz devam ediyor olması olabilir. Ayrıca Sivashanmugam vd. (2004) gibi, yeterince detaylı olmayan, örneğin "federasyon desteği" gibi, yalnızca belirli noktalardan ebRIM ve UDDI kataloglarını karşılaştıran bazı çalışmalar mevcuttur. Konumsal olmayan çalışma alanlarından olan bu çalışmaların hiçbirinde doğal olarak konumsal veri boyutu yoktur.

Bu çalışmada, bilgi kaynaklarının kataloglanmasına yönelik olarak geliştirilmiş olan çeşitli teknoloji ve tekniklerin basitleştirici ve birleştirici bir çevrede anlaşılmasını sağlayacak, gerçekleştirimci ve karar vericiler için yol gösterici olacak bir dizi ölçüt belirlenmiştir. Bu ölçütler, bir ya da daha çok katalog kullanılması gibi iki ana sınıf altında düşünülmüştür. Aslında amaç başlı başına UKVA'nın gerçekleştirimi ise, tek katalog seçimi çok önemli görülmeyebilir. Çünkü o durumda birden çok katalog kullanılması gerekecektir. Dolayısıyla bu katalogların birlikte işlerliği çok daha önemli olacaktır. Ancak UKVA gibi çok iddialı projelere geçişin, genellikle gözlendiği üzere, çok da eş zamanlı bir şekilde sağlanamaması nedeni ile tek katalog gerçekleştirimleri de önem kazanmaktadır.

Diğer yandan bu ölçütler, piyasadaki ticari katalog ürünleri için, genel bir "benchmark" niteliği taşımaktadır. Sonuç olarak, tek katalog sınıfında katalogun bilgi modeli, katalog arayüzleri, katalog uygulama profilleri ve anlamsal destek ölçütleri belirlenmiştir.

Kataloglama ölçütleri

- Bilgi modeli
  - Bilgi kaynağı türü
  - Anlatım gücü
  - Genişletilebilirlik
- Katalog arayüzleri
  - Bulma (sorgu) arayüzü
  - Yayınlama arayüzü
  - Diğer arayüzler
- Katalog uygulama profili
- Anlamsal destek

Bilgi modeli: Bir katalogun "bilgi modeli" aslında onun veri modelidir. Ancak kataloglar bağlamında kullanılan terim genellikle "bilgi modeli" olduğu için burada da bu isimlendirmeye bağlı kalınmıştır. Bu belki kataloglarda tutulan verinin de "bilgi kaynağı" olarak adlandırılmış olmasından kaynaklanmıştır. Bilgi modeli kendi içerisinde de çeşitli ölçütler içerir. Bunlar bilgi kaynağı türü, anlatım gücü ve genişletilebilirliktir. Aşağıda bunların her biri açıklanacaktır.

Bilgi kaynağı türü: Yukarıda da belirtildiği üzere, YWSO lar henüz erişilebilmiş bir hedef değildir. Bu nedenle, bugün bütün "sağlayıcılar" Web servisi sağlayabilecek durumda değildir. Bazı sağlayıcılar yalnızca veri sağlayabilir durumdadır. Bu nedenle mevcut uygulamada kataloglarda yalnızca Web servisleri değil, aynı zamanda XML şema dokümanları, "yetenekler" dokümanı, biçimlendirme dili dokümanları, çoklu ortam dosyaları ve belirtim dokümanları gibi çeşitli türde sayısal veriler kataloglanmaktadır. Bu bakımdan mevcut uygulamada kataloglarda tutulan bilgi çoğunlukla Web servisi olarak değil de "elektronik kapsam" ya da daha genel ifadeyle "bilgi kaynağı" olarak anılmaktadır (OASIS, 2005a; OGC, 2006). Burada en önemli sorulardan biri, "veriyi sunan servis varken o veriyi ayrıca sunmaya gerek var mıdır?" sorusudur. Bölümün başlığı da bu soru ile ilintilidir; yalnızca servisleri kataloglayan servis katalogları mı yoksa hem servis hem de veri kataloglayan katalog servisleri mi? Mevcut uygulama ve bu ölçüt açısından arzu edilen katalog servisleridir. Ancak, W3C Web servisleri mimarisine dayalı bir YWSO ya da UKVA ortamında söz konusu sorunun cevabı açık bir "hayır" dır.

Anlatım gücü: Bilgi kaynakları, onları özet olarak tanımlayan ve "meta veri" olarak anılan özellikleri ile bir kataloga kaydedilir. Belirli özelliklerde bilgi kaynağı arayanlar da bu özellikler yardımı ile arama gerçekleştirirler. Bir Web servisini tanımlayabilecek sayısız meta verilerden bir kaçı örneğin Web servisini sunan servis sağlayıcının hangi ticari sektöre ait olduğu, Web servisinin hangi tarihler arasında kullanıma açık olacağı, servisin hangi sınıflandırma sisteminde ne tür bir servis olduğu, servis sağlayıcının servis sunma kalite göstergesi olabilir. Bir veri grubu ise örneğin konumsal referans sistemi, veri kalitesi gibi özellikleri ile karakterize edilebilir. Hatta meta veriler arasındaki örneğin "bir Web servisi hangi tip bir Web servisinden sonra icra edilebilir?" ya da "bir servis hangi tip verileri kullanabilir?" gibi ilişkilerin temsil edilmesi gerekebilir.

Söz konusu tanımlayıcı özellikler ve ilişkiler aslında belirli "kavramsallaştırmalara" işaret eder. Bu kavramsallaştırmaların neler olması gerektiğini uygulamalar ya da kullanıcı kitlesi belirleyecektir. Bu kavramsallaştırmaların ne oranda temsil edilebildiği ise, kataloglamada kullanılan meta veri modelinin ya da katalogun "bilgi modeli" nin "anlatım gücü"ne bağlı olacaktır.

Geleneksel veri modellemede kavramsallaştırmalar belirli temel kavramlarla ifade edilir. Bunlar genelleştirme, ilişki kurma (association), oluşum (composition) ve bir araya getirme (aggregation) dir. UML (Unified Modelling Language) (OMG, 2005) dili bunlara ilave olarak, örneğin "bağımlılık" gibi daha pek çok kavram içermektedir. İlişki kurma iki ya da çok varlık arasında bir ilişki tanımlar. Aslında oluşum ve bir araya getirme de birer ilişkidir. Her ikisi de varlıklar arasında "parça-bütün" ilişkisi tanımlar. Aradaki fark, oluşumda bütün durumundaki nesnenin silinmesi, parçalar anlamsız durumda kalacağı için, parçaların silinmesini de gerektirir. Bir araya getirme de böyle bir durum yoktur. Örneğin gölet ve akarsu nesnelerinin bir araya getirilmesi ile oluşturulan bir "drenaj-ağı" nesnesinin silinmesi parça ya da bileşen durumundaki gölet ve akarsuların silinmesini gerektirmez. Genelleştirme genellikle özelleştirme ile birlikte anılır. Örneğin ISO 19115 meta veri standardında "*MD_identification*" sınıfı "*MD_dataIdentification*" ve "*SV_ServiceIdentification*" sınıflarının her ikisinin de üst sınıfı yani daha genel halidir. Genelleştirmenin ters yönü ise özelleştirme ifade eder.

"Anlatım gücü", veri modelleme literatüründe sıkça anılan bir konu olmakla birlikte, gerek tanımı ve gerekse anlatım gücünün neden yüksek olması gerektiği çok somut açıklanmamaktadır. Batini vd. (1992), bir şemanın anlatım gücünü, şemanın Varlık-İlişki (V-İ) modeli kavramları yardımıyla doğrudan, ilave bir açıklamaya gerek kalmadan

anlaşılabilmesine bağlı olarak tanımlamaktadır. Saltor vd. (1991), bir veri modelinin anlatım gücünü, modelin bir kavramsallaştırmayı doğrudan temsil edebilme derecesi olarak tanımlamakta ve konunun "yapısal" ve "davranışsal" iki farklı boyutuna işaret etmektedir. Tabi, bu yararlı bir sınıflandırmadır ancak, konunun meta veri modelleme açısından öneminin irdelenebilmesi için, biraz daha somutlaştırılması gerekir. Çünkü "doğrudan temsil etme" hala oldukça soyut durmaktadır.

Aslında konunun iki boyutu vardır birincisi kavramsal veri modeli ikincisi ise gerçekleştirim veri modeli boyutudur. Kavramsal veri modeli boyutunda modelin içerdiği kavramlar ne kadar fazla ise anlatım gücü de o kadar yüksek olacaktır. Örneğin V-İ modelinin anlatım gücü, ilişkisel modele göre daha yüksektir. Bu durumda V-İ modelindeki bir şema eğer ilişkisel olarak eşdeğerine dönüştürülerek gerçekleştirilecekse bazı kavramlar kaybolabilir. Örnek olarak V-İ modelindeki "zayıf varlık tipi" alınabilir. İlişkisel şemaya geçişte bu kavram kaybolacağı için, kavramın ifade ettiği anlamın yakalanması ancak uygulama programları ile mümkün olacaktır ki, bu tip işlevler bir yandan uygulama programlarının yükünü artırırken, bir yandan da veri tabanının yönetimini zorlaştıracaktır. Oysa aynı kavram V-İ modeli gibi varlık tabanlı bir yapıda örneğin Nesne Yönelimli bir veri tabanı ve Nesne Yönelimli Programlama (NYP) ile gerçekleştirilirse, aynı anlamın gerektirdiği davranış, nesneler vasıtasıyla doğrudan gerçekleştirilebilecektir. Çünkü NYP, yeniden-kullanılabilirliğin çok üst düzeyde olması nedeni ile de davranış modellemede kolaylık sağlamaktadır (Meyer, 1988). Bununla birlikte, bu ölçütle kastedilen kavramsal düzeydeki anlatım gücüdür.

Katalog servisleri bilgi modeli olarak ya kendi modellerini ya da bir meta veri modelini kullanabilirler. ISO 19115 (ISO, 2003) ve FGDC kapsam standardı (FGDC, 1998) verilere yönelik, ISO 19119 (ISO, 2001) ise Web servislerine yönelik meta veri standartlarına örnektir. Bu ölçüt açısından arzu edilen, katalogun anlatım gücü yüksek olan bir bilgi modeline sahip olmasıdır.

Genişletilebilirlik: Genişletilebilirlik, bilgi modeline, yeni kavramsallaştırmaların eklenebilmesindeki kolaylık olarak tanımlanabilir. Yeni kavramsallaştırmalar, bilgi modeline yeni sınıflar ya da ilişkiler eklenmesini gerektirebilir. Geleneksel veri modellemede bugün gelinen nokta itibarı ile yaygın kabul edilen bir gerçek, bütün ihtiyaçları karşılayacak genel bir model tasarlamanın imkansız olduğudur (Wiederhold, 1992). Bu bakımdan, aynen veri modellemede olduğu gibi, meta veri modellemede de meta veri modelinin anlatım gücünün yüksek olmasından daha önemlisi belki de, modelin

ihtiyaçlara kolaylıkla uyarlanabilir ya da "genişletilebilir" olmasıdır. Bu açıdan varlık bazında tanımlama ve ilişki kurma içeren "varlık tabanlı" modeller avantaj sağlamaktadır. Gerek genelleştirme ve gerekse oluşum hiyerarşileri ve ilişki kurma vasıtasıyla varlıklar arasında yeni ilişkiler kolaylıkla tanımlanabilir. Ancak ilişki kurma yoluyla kavramsallaştırmaların ve dolayısıyla, anlatım gücünün artırılması, bir yandan sorgulanabilir özellikleri artırırken, diğer yandan katalog gerçekleştirimini ve farklı katalog bilgi modellerinin kullanıldığı bir ortamda gerek söz dizimsel ve gerekse anlamsal birlikte işlerliği zorlaştıracaktır.

Katalog arayüzleri: Katalog arayüzleri, katalogun iki temel bileşeninden biri olarak düşünülebilir. Bu bileşenlerden biri katalog bilgi modeli, diğeri katalog arayüzleridir. Bilgi modeli, bilgi kaynaklarının katalogda nasıl tanımlanacaklarını belirler. Katalog arayüzleri ise bir bilgi kaynağını katalogda yayınlama, arama gibi işlevlerin yürütülmesine olanak tanırlar. Örneğin CSW katalog servisleri, OGC CSW belirtimi tarafından tanımlanan arayüzleri gerçekleştirirler. CSW belirtiminde, bir CSW katalog servisinin gerçekleştirmesi gereken beş arayüz (OGC_Service, Discovery, Manager, BrokeredAccess, Session) tanımlanmıştır. Bunlardan "OGC_Service" ve "Bulma" (Discovery) arayüzleri, bir CSW katalog servisinin sağlaması zorunlu olan arayüzleridir. Bir CSW katalog servisinin gerçekleştirebileceği arayüzler, katalog servisinin kullandığı uygulama protokolüne bağlı olarak değişir. Örneğin, HTTP protokolü "durumsuz" (stateless) bir protokol olduğu için, bu protokolü kullanan bir CSW katalog servisi "Oturum" (Session) arayüzünü gerçekleştiremez. CSW belirtimi, katalog servisi arayüzlerinin Z39.50, CORBA/IIOP ve HTTP protokollerindeki gerçekleştirimlerini tanımlamaktadır. Arayüzler ve operasyonlar, farklı protokolleri kullanan katalog servislerinde farklı isimlerle adlandırılırlar. Örneğin, HTTP protokolünü kullanan CSW gerçekleştirimlerinde, "Yönetim" (Manager) arayüzü, "Yayınlama" (Publication) arayüzü olarak adlandırılmaktadır.

Sağlayıcılar, sahip oldukları bilgi kaynaklarını tanımlayan meta verileri, yayınlama arayüzünü kullanarak CSW katalog servislerinde yayınlarlar. Yayınlama arayüzü, seçmeli olan "*Transaction*" ve zorunlu olan "*Harvest*" operasyonlarına sahiptir.

Katalog arayüzleri açısından istenen ya da gereksinim duyulanın ne olduğu konusunda bilgimiz dahilinde bir çalışma yoktur.

Bulma (sorgu) arayüzü: Sorgu arayüzü, istemcilerin katalogu sorgulamak için kullandıkları arayüzdür. Sorgu arayüzü açısından istenen ya da gereksinim duyulanın ne olduğu konusunda bilgimiz dahilinde bir çalışma yoktur. Genel olarak arzu edilen,

gerektiğinde parametrik ve "anlık" (ad-hoc) sorgu yapılabilmesidir. UDDI gibi bazı kataloglar yalnızca parametrik sorgu desteği vermektedir. Bu da kullanıcının aradığı özelliklerin "parametre" olarak önceden tanımlanmamış olması durumunda sorgu yapılamaması nedeni ile kısıtlayıcı bir yoldur. Anlık sorgulamayı destekleyen kataloglar genellikle bir filtreleme dili ya da SQL ile sorgulamaya izin verirler (OGC, 2005e; OASIS, 2005a). Bu durumda da "sorgulama dilinin anlatım gücü" (Dustdar ve Treiber, 2005) önem kazanmaktadır. Bir başka ifadeyle sorgu dilinin anlatım gücünün yüksek olması istenir. Konumsal veri açısından, sorgulama dilinin OGC CSW filtreleme dilinde olduğu gibi, konumsal operatörleri desteklemesi gerekir.

Yayınlama arayüzü: SYM de "yayınlama" işlemi ile kastedilen, sağlayıcıların, sahip oldukları bilgi kaynaklarını tanımlayan meta verileri, bir katalog servisine koyarak, servis ya da veri kaynaklarını başkalarının kullanımına açmalarıdır. Sağlayıcılar bu işlevi, katalogun yayınlama arayüzünü kullanarak yerine getirirler. Yayınlama arayüzü açısından arzu edilebilecek noktalardan biri, katalogun hem sağlayıcıların doğrudan yayınlama işlemi yapmasına, hem de katalogun tüm kullanıcılardan yayın toplaması yolu ile yayınlama yapılmasına olanak tanımasıdır. Örneğin OGC CSW de yayınlama arayüzü, seçmeli olan "*Transaction*" ve zorunlu olan "*Harvest*" operasyonlarına sahiptir.

Diğer arayüzler: Katalog arayüzleri açısından istenen ya da gereksinim duyulanın ne olduğu konusunda bilgimiz dahilinde bir çalışma yoktur. Arayüz fonksiyonlarının fazlalığı artı bir puandır. Örneğin ebXML katalog servisinin sunduğu "olay bildirme" (event-notification) fonksiyonu katalog içeriğindeki değişiklikleri abone olan kullanıcılara bildirmektedir. Bu sayede örneğin katalogda yeni yayınlanmaya başlayan ya da katalogda yayınlanması durdurulan servislerin bu durumları kullanıcılara bildirilebilir.

Katalog uygulama profili: OGC (2005a), CSW bağlamında bir katalog uygulama profilini, katalogun belirli bir protokole göre olan gerçekleştirimi olarak tanımlamaktadır. CSW belirtimi, katalog servisi arayüzlerinin Z39.50, CORBA/IIOP ve HTTP protokollerindeki gerçekleştirimlerini tanımlamaktadır. Bir CSW katalog servisinin gerçekleştirebileceği arayüzler, katalog servisinin kullandığı uygulama protokolüne bağlı olarak değişir. Örneğin, HTTP protokolü "durumsuz" (stateless) bir protokol olduğu için, bu protokolü kullanan bir CSW katalog servisi "Oturum" (Session) arayüzünü gerçekleştiremez. Temel belirtim, uygulama profili ve katalog servisi arasındaki ilişkiyi, OGC (2005a)'da, "...bir katalog servisi belirtimine uyan birden çok uygulama profili ve bir uygulama profiline uyan birden çok katalog servisi olabileceği..." şeklinde özetlenmiştir.

Çünkü bir uygulama profili oluşturmada, bütün durumlarda olmasa bile çoğunlukla söz konusu olan ana modelin daraltılması ya da özelleştirilmesidir.

Katalog uygulama profili ile kastedilen, katalogun belirli bir gerçekleştirim topluluğu ya da alanı (domain) için özelleştirilmesidir. Bu özelleştirme bazen genişletme de içerebilir. Bu durumda birden çok katalogun aynı sorgu ile sorgulanabilmesi için sorgunun ortak sorgulanabilir özellikler üzerinden yapılması gerekir. Uygulama profilleri bazındaki farklılıklar ise ancak farklı protokollere sahip olan sağlayıcılardaki dönüştürücülerle sağlanabilir. Bu durumda özetle söylenebilecek olan şey, farklı uygulama profili sayısı arttıkça birlikte işlerliği sağlamanın zorlaşacak olduğudur.

Anlamsal destek: Bu ölçüt katalog bilgi modelinin anlamsal olarak genişletilebilmesine yöneliktir. Anlamsal destek aslında genişletilebilirlik ölçütü altında da düşünülebilirdiyse de, katalogun anlamsal Web (Berners-Lee vd., 2001) açısından kullanılabilirliğinin değerlendirilebilmesi açısından ayrı bir ölçüt olarak düşünülmüştür. Bu alandaki başlıca çalışmalar (Doğaç, 2003; Doğaç vd., 2003; Doğaç vd., 2004; Doğaç vd., 2005) olarak sayılabilir. Bu çalışmalarda genel olarak iki yolla anlamsal zenginleştirme uygulanmıştır. Doğaç (2003) ve Doğaç vd., (2003) te uygulanan yöntemde, Web servisleri belirli "alan ontolojilerine" (domain ontologies) dayalı olarak DAML-S dilinde tanımlamış ve bu tanımlamalar, ebRIM bilgi modelinin *ClassificationScheme* ve *ClassificationNode* sınıflarına resmedilmiştir. Ancak bu yöntem yalnızca taksonomilerin bir dereceye kadar temsil edilmesine olanak tanımaktadır. Diğer yandan, hiyerarşik yapıdaki taksonomilerin aksine çizge (graph) yapıdaki ontolojiler ise bu yolla temsil edilemeyecektir. Çünkü bu yöntem, ontolojinin iki sınıfı ya da kavramı arasındaki ilişkiyi temsil edecek bir ebXML katalog servisi mekanizması kullanmamaktadır. Bu durumu, ISO 19119 "Coğrafi servis taksonomisi" (Geographic Services Taxonomy) bağlamında basit bir örnekle açıklamak gerekirse, verilen bir zaman ve iki nokta arasında en kısa yolu belirleyerek istemciye sunan bir "en kısa yol servisi" alınabilir. Bu servisin hem "konumsal" hem de "zamansal" bir servis olduğu bilgisinin doğrudan temsil edilmesi, zamansal ve konumsal kategorilerinin taksonomide bulunduğu hiyerarşik düzeye bağlı olacaktır. Çünkü ebRIM bilgi modeli bir kategorinin yalnızca birinci derecedeki hiyerarşik "üst" üne ait bilgi içerir.

Doğaç vd. (2004) ve Doğaç vd. (2005) gibi daha sonraki çalışmalarda uygulanan ikinci yöntemde ise, Web servisleri arasında kullanılan ontoloji gereği temsil edilmesi istenen ilişkilerin, *Association* nesneleri vasıtası ile temsil edilmesi yoluna gidilmiştir. Bunun için, "OWL Lite" dilinde yapılan Web servisi tanımlamaları, çalışma ekibi

tarafından geliştirilen bir ayrıştırıcı (parser) yardımı ile otomatik olarak, ebRIM bilgi modeli nesnelerine resmedilmiştir. Ayrıştırıcı, her bir OWL sınıfı için bir *ClassificationNode* nesnesi, her bir nesne özelliği için de bir *Association* nesnesi oluşturmaktadır. Temsil edilmek istenen ilişkinin ebRIM bilgi modelinin yerleşik (built-in) *Association* nesne tipleri kapsamında olmaması durumlarına yönelik olarak, yeni *Association* nesne tipi tanımlamaları eklenmiştir. Bunlardan bazıları, "*subClassOf*" "*subPropertyOf*", "*inverseOf*", "*intersectionOf*" ve "*succeed*" gibi *Association* tipleridir. *Association* nesnelerine yüklenen anlamlar artık herhangi bir uygulama programı, SQL gibi bir sorgu dili ile ortaya çıkarılabilir. Doğaç vd. (2005) bu amaçla SQL için tanımlanmış olan "saklanmış sorgu" mekanizmasını kullanmıştır. Saklanmış sorgular üzerinden de çeşitli parametrik sorgular tanımlanmıştır. Bu sayede örneğin bir Web servisinin, kendisinden sonra kullanacağı Web servislerini ekranda kullanıcıya gösteren bir parametrik sorgu gerçekleştirilebilmektedir. Bu yolla, bir "eğlence servisi" vasıtası ile bir eğlence planı yapmaya çalışan bir kullanıcı, bu servisten önce sırası ile "otel rezervasyon" ve "uçak rezervasyon" servislerine de ihtiyacı olduğunu belirleyebilir. Ancak belirli bir uygulama için hangi servislerin kullanılabileceğinin bu şekilde belirlenmesi, tümüyle "önceden tanımlı" (hard-coded) ilişkiler olduğu için, anlamsal bir servis bulma ve düzenleme olarak nitelendirilemez.

Literatürde yukarıda anılan çalışmalara oranla daha yeni bir çalışma olarak tespit edilen Di vd. (2006)'nın çalışması, konumsal verilere yönelik olması ile Doğaç ve ekibinin yukarıda anılan çalışmalarından farklı, ancak anlamsal desteğin bir ebXML KS de nasıl ve ne dereceye kadar temsil edilebileceğinin değerlendirilmesi açısından benzerdir. Di vd. (2006) veri ve servis tanımlama için OWL-S dilini kullanmış ve bu tanımlamaları "RDF üçlüleri" olarak anılan kaynak (resource/subject), özellik (property/predicate) ve değer (value/object) formatına dönüştürerek "bilgi tabanı"na koymuştur. Bilgi tabanında bu şekilde kavramsal olarak tanımlı hale gelen veri ve servislerin örnekleri ise OWL-S tanımlamalarının ebXML KS bilgi modeline resmedilmesi ile elde edilen şemaya göre ebXML KS ye konmuştur. Bu resmetmede OWL özellikleri (property) katalog nesnelerinin (registry object) *Slot* bileşenlerine, OWL aksiyomları da katalog nesneleri arasındaki ilişki tiplerine (Association) karşılık gelmiştir. Bu, aynen Doğaç ve ekibinin, bu resmetme için geliştirdikleri bir yazılımı kullanarak uyguladığı yöntemdir. Di vd. (2006) verilen örnek "heyelan riskini" belirleyen bir servis ihtiyacındaki kullanıcının hangi bileşen servisleri kullanarak böyle bir servis düzenleyebileceğinin belirlenmesidir. Söz konusu

servis düzenlemenin, Di vd. (2006) de anlatılan tarzda yapılması durumunda, yine aynı kaynakta söz edildiği gibi bir çıkarsama ve dolayısıyla bilgi tabanına gerek yoktur. Çünkü örnekteki servis düzenleme tamamen hiyerarşik bir tarzda servislerin ne tip girdi ve çıktılar kullandığının ve bunların hangi veri setleri olduğu ilişkilerine göre yapılmaktadır. Bu bilgiler zaten ebXML KS ye yukarıda açıklanan şekilde, yani girdi/çıktı tipleri *Slot* ve servisin hangi veriler üzerinde çalıştığı da *Association* olarak aktarılmıştı. Bu durumda da girdi çıktı tipi eşleştirmeleri aslında tamamen sözdizimsel (syntactic) dir. Di vd. (2006)'de söz edilen bilgi tabanı ve çıkarsamacı ancak, Di vd. (2006)'nın çalışmalarının bir sonraki adımında gerçekleştirmeyi düşündükleri tam otomatik servis düzenleme için olabilir.

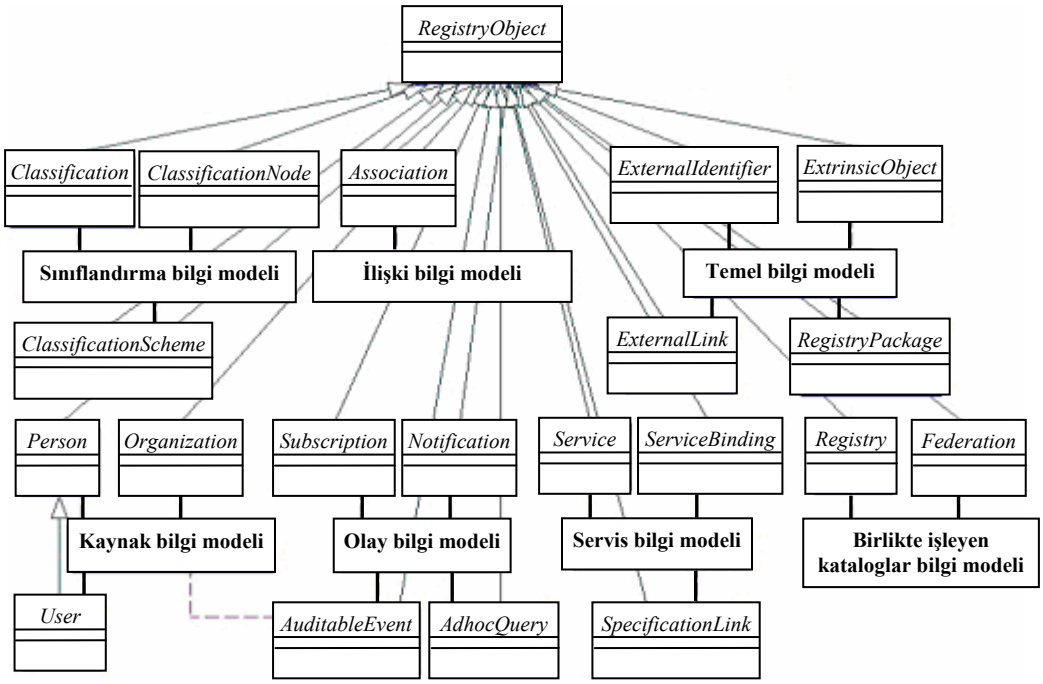
2.3.2. EbXML Katalog Servisi

ebXML girişimi, elektronik-iş'i (e-iş) desteklemek için XML tabanlı bir altyapı geliştirmek amacıyla, UN/CEFACT (United Nations Centre for Trade Facilitation and Electronic Business) ve OASIS (Organization for the Advancement of Structured Information Standards) ortaklığı tarafından Kasım 1999 da başlatılan ve Mayıs 2001 de tamamlanan bir projedir (Manes, 2002). Bu ortaklık tarafından geliştirilen e-iş altyapısı, bir kısmı OASIS, bir kısmı da UN/CEFACT tarafından yönetilen belirtimlerle tanımlanmıştır. OASIS, ebXML Registry Teknik Komisyonu'nu Mayıs 2001 de kurmuş ve ebXML Katalog Servisinin (ebXML KS) ikinci sürümü, teknik komisyon tarafından Nisan 2002 de standart olarak onaylanmıştır. ebXML KS'nin, şu an yürürlükte olan son sürümü (Version 3.0) komisyon tarafından Mayıs 2005 de onaylanmıştır.

ebXML KS, birbirini tamamlayan iki ayrı standart tarafından tanımlanır. İlk standart, "ebXML Registry Services and Protocols" (ebRS) (OASIS, 2005a), bir ebXML KS'nin desteklemesi gereken arayüzleri, ikinci standart ise, "ebXML Registry Information Model" (ebRIM) (OASIS, 2005b), ebXML KS'nin bilgi modelini tanımlar.

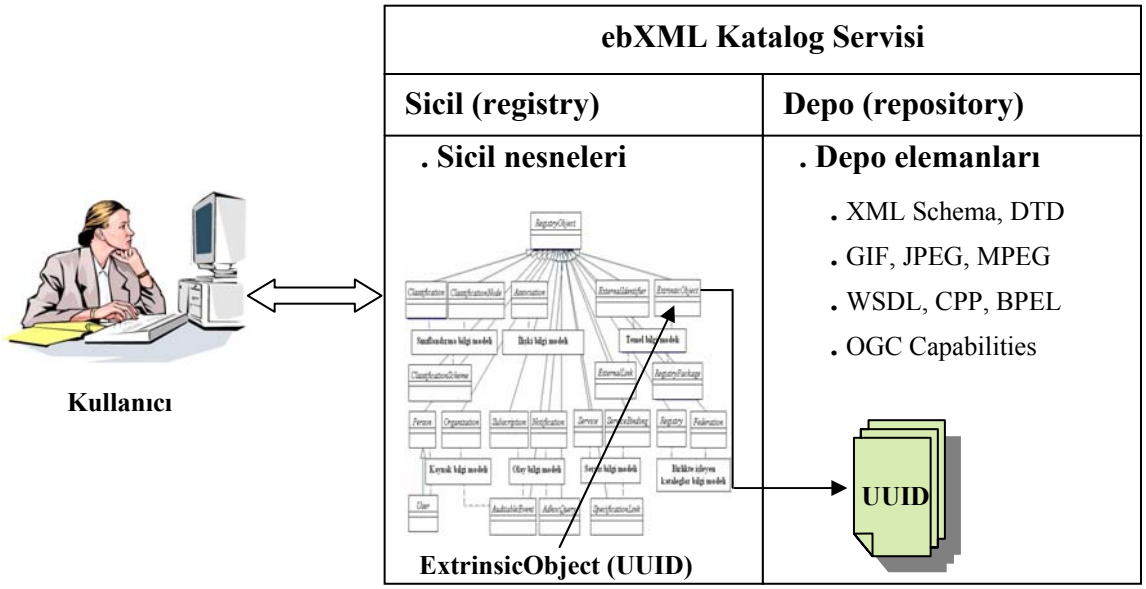
Bilgi modeli: ebXML KS'nin bilgi modeli, nesne yönelimli bir tasarıma sahiptir. Bilgi modeli, her türlü bilgi kaynağını tanımlamaya olanak sağlayan otuzdan fazla meta veri sınıfı içerir. Bilgi modelinde yer alan meta veri sınıflarının birçoğu, *RegistryObject* sınıfının bir alt sınıfıdır (Şekil 8). *RegistryObject* sınıfı, diğer sınıfların bilgi kaynaklarını tanımlamak için kullandıkları ortak meta veri özniteliklerini sağlayan bir süper sınıftır. Bir katalog servisinin bilgi modeli, bilgi kaynağı türü, anlatım gücü ve genişletilebilirlik ölçütlerine göre değerlendirildiği için, bu bölümde sadece ebXML KS bilgi modelinin söz

konusu ölçütler açısından değerlendirilmesine olanak sağlayan meta veri sınıfları açıklanacaktır. Bilgi modelindeki diğer sınıflarla ilgili detaylı bilgiler, ebRIM standardında (OASIS, 2005b) bulunabilir.



Şekil 8. EbXML katalog servisi bilgi modeli (OASIS, 2005b).

Bilgi kaynağı türü: ebXML KS, her türlü bilgi kaynağını tanımlamaya olanak sağlayan bir bilgi modeline sahiptir. Bir ebXML KS, sicil (registry) ve depo (repository) olarak adlandırılan iki ana bileşenden oluşur (Şekil 9). ebXML KS terminolojisinde, WSDL, BPEL gibi XML dosyaları, GIF, JPEG gibi görüntü dosyaları ve WAV, MPEG gibi çoklu ortam dosyalarını da içeren her türlü elektronik doküman, "depo elemanları" (repository items) olarak adlandırılır ve ebXML KS'nin depo bileşeninde yer alır. Depo elemanlarını tanımlamak için kullanılan meta veri sınıfları ise, "sicil nesneleri" (registry objects) olarak adlandırılır ve ebXML KS'nin sicil bileşeninde yer alırlar (OASIS, 2005b).



Şekil 9. EbXML katalog servisi mimarisi

Depo elemanlarını tanımlamak için kullanılan temel meta veri sınıfı, *ExtrinsicObject* sınıfıdır. Her depo elemanı, sicilde bir *ExtrinsicObject* nesnesi tarafından tanımlanır (Şekil 9). Başka bir sunucuda bulunan bir depo elemanı, *ExternalLink* nesnesi kullanılarak tanımlanır. Web servislerini tanımlamak için bilgi modelinin *Service*, *ServiceBinding* ve *SpecificationLink* sınıfları kullanılır. *Service* sınıfı, Web servislerini tanımlamak için kullanılan temel meta veri sınıfıdır. *ServiceBinding* sınıfı, bir Web servisinin erişim adresini tanımlar. *SpecificationLink* sınıfı ise, Web servisinin nasıl kullanılacağını tanımlayan WSDL veya OGC yetenekler dokümanı gibi bir teknik dokümana erişim için bir referans tanımlar. *SpecificationLink* nesnesi, teknik doküman depo tarafından tutuluyor ise bir *ExtrinsicObject* nesnesine, doküman başka bir sunucuda ise bir *ExternalLink* nesnesine referans eder. Bilgi kaynağı sağlayıcılarını tanımlamak için ebRIM'in *Organization* sınıfı kullanılır. Bilgi kaynakları ve sağlayıcılar, *Classification* sınıfı kullanılarak sınıflandırılırlar. *ClassificationScheme* sınıfı, sicile kayıtlı bir sınıflandırma sistemini (taksonomi) temsil eder. Bir taksonomi içerisindeki kategoriler, *ClassificationNode* sınıfı kullanılarak tanımlanırlar. *Classification* sınıfı, bir taksonomi içerisindeki bir kategoriyi kullanarak, bir bilgi kaynağını veya bir organizasyonu sınıflandırır. Örneğin bir OGC WFS servisi, ISO 19119 Coğrafik Servis Taksonomisinde, "Feature Access Service-Web Feature Service (WFS)" kategorisi kullanılarak sınıflandırılır.

Anlatım gücü: Bir katalog servisinin bilgi modelinin anlatım gücü, kavramsal veri modeli boyutunda, modelin içerdiği kavramlar ile ölçülmektedir. ebXML KS, Nesne-Yönelimli bir bilgi modeline sahip olduğu için bilgi modeli, veri modellemede kullanılan genelleştirme, ilişki kurma, oluşum ve bir araya getirme gibi veri modeli kavramlarını içermektedir. Bu nedenle ebXML KS bilgi modelinin anlatım gücü yüksektir. ebXML KS bilgi modeli ayrıca, sicil nesneleri arasında ilişki kurmaya olanak sağlayan "*Association*" sınıfına sahiptir. *Association* sınıfının "*sourceObject*" ve "*targetObject*" öznitelikleri, aralarında ilişki kurulacak olan sicil nesnelerini belirtir. Sicil nesneleri arasındaki ilişkinin tipi, *Association* sınıfının "*associationType*" özniteliği tarafından belirlenir. ebRIM standardı, bir ebXML katalog servisinin kullanacağı ilişki tiplerini önceden tanımlanmıştır (Tablo 1). Örneğin, bir Web servisi ile servisi sunan organizasyon arasındaki ilişki "SubmitterOf" ilişki tipi ile tanımlanmaktadır. Ayrıca bilgi modeli, kullanıcıların yeni ilişki tipleri tanımlamasına olanak sağlamaktadır.

Tablo 1. EbRIM standardı tarafından tanımlanan ilişki tipleri

İlişki tipi	**Açıklaması**
RelatedTo	Kaynak nesnenin, hedef nesne ile ilişkili olduğunu belirtir.
HasFederationMember	Bir katalog federasyonuna üye olmak isteyen bir ebXML KS tarafından kullanılması gereken ilişki tipidir.
HasMember	Hedef nesnesinin, kaynak nesnesinin bir üyesi olduğunu belirtir.
Contains	Kaynak nesnenin, hedef nesneyi içerdiğini belirtir.
EquivalentTo	Kaynak nesne ile hedef nesnenin eşdeğer olduğunu belirtir.
Extends	Kaynak nesnenin, hedef nesneyi genişlettiğini belirtir.
Implements	Kaynak nesnenin, hedef nesne tarafından sağlanan fonksiyonelliği gerçekleştirdiğini belirtir.
InstanceOf	Kaynak nesnenin, hedef nesnenin bir örneği olduğunu belirtir.
Supersedes	Kaynak nesnenin, hedef nesneyi geçersiz kıldığını belirtir.
Uses	Kaynak nesnenin, hedef nesneyi kullandığını belirtir.
Replaces	Kaynak nesnenin, hedef nesnenin yerine geçtiğini belirtir.
SubmitterOf	Hedef nesnenin, kaynak nesne tarafından yayınlandığını belirtir.
ResponsibleFor	Bir organizasyonun, bir registry nesnesinin yönetiminden sorunlu olduğunu gösterir.
OwnerOf	Kaynak nesnenin, hedef nesnenin sahibi olduğunu belirtir.
OffersService	Kaynak nesnenin, hedef nesneyi bir servis olarak sunduğunu belirtir.

Genişletilebilirlik: ebXML KS, genişletilebilir bir bilgi modeline sahiptir. *Slot* sınıfı, bilgi modelinin genişletilebilirliğine olanak sağlayan en önemli sınıftır. Bir kullanıcı, *Slot* sınıfını kullanarak bilgi modelindeki herhangi bir sicil nesnesine ilave öznitelikler ekleyebilir. Örneğin bir servis sağlayıcı, servis parametrelerinin servis tanımında yer almasını sağlamak için, "*Service*" sınıfına girdi ve çıktı isminde iki yeni öznitelik ekleyebilir. Bilgi modelinin genişletilebilirliğine olanak sağlayan bir diğer sınıf, *Association* sınıfıdır. Kullanıcılar, ebRIM standardı tarafından önceden tanımlanan ilişki tiplerinin, sicil nesneleri arasındaki ilişkileri tanımlamak için yetersiz kaldığı durumlarda, yeni ilişki tipleri tanımlayarak bilgi modelini genişletebilirler. Örneğin, ebXML KS'nin anlamsal Web servislerini tanımlamak için kullanılması durumunda, hem alan (domain) ontolojileri arasındaki ilişkileri hem de bir anlamsal Web servisini tanımlayan OWL-S servis ontolojisindeki sınıflar arasındaki ilişkileri tanımlamak için yeni ilişki tiplerine ihtiyaç duyulur. Örneğin ebRIM standardı tarafından tanımlanan mevcut ilişki tipleri, OWL-S servis tanımında bir sınıfın bir diğer sınıfın alt sınıfı olduğunu gösteren "*subClassOf*" ilişki tipini içermemektedir. Bilgi modeline yeni ilişki tiplerinin eklenmesi ile ilgili ayrıntılı bilgiler, hem ebRIM bilgi modelinin anlamsal desteğinin anlatıldığı bölümde, hem de OGC CSW katalog servislerinin ebRIM profilinin anlatıldığı bölümde bulunabilir.

Katalog Arayüzleri: ebXML KS, kullanıcılarına oldukça zengin bir arayüz desteği sunmaktadır. Ancak bu bölümde, ebXML KS'nin bulma ve yayınlama arayüzüne ilave olarak "yaşam süreci yönetim" (lifecycle management) arayüzü ve "olay bildirme" (event notification) arayüzü açıklanacaktır. ebXML KS'nin, bu çalışma kapsamında ele alınmayan örneğin "güvenlik arayüzü" gibi diğer arayüzleri ile ilgili bilgiler, ebRS standardında bulunabilir.

Bulma arayüzü: İstemciler, bir ebXML KS tarafından yayınlanan bilgi kaynaklarını bulmak için ebXML KS'nin "sorgu yönetim" (query management) arayüzünü kullanırlar. Sorgu yönetim arayüzü, istemcilerin iki türlü sorgu yapmalarına olanak sağlar. İlk sorgu yöntemi anlık sorgu (adhoc query), ikincisi ise saklanmış sorgu (stored query) olarak adlandırılır. Anlık sorguda istemci bir sorgu mesajı yazar ve bir ebXML KS'ye gönderir. Sorgu mesajı, bilgi kaynaklarını tanımlamak için kullanılan meta veri sınıflarının özniteliklerinden oluşan ve SQL ya da ebRIM Filter sözdizimine göre yazılan bir sorgu ifadesi içerir. Örneğin aşağıdaki ebRIM filter sorgu ifadesi, bir istemcinin bir ebXML KS de, ISO 19119 coğrafik servis taksonomisinin, "Feature Access Servise- Web Feature

Service (WFS)" kategorisine göre sınıflandırılmış tüm Web servislerini bulmasına olanak sağlar.

```
<ServiceQuery>
  <ClassificationQuery>
    <ClassificationNodeQuery>
      <PrimaryFilter comparator="EQ" domainAttribute="path"
        value="/${ISO19119_SCHEME_ID}/WFS" xsi:type="StringFilterType"/>
    </ClassificationNodeQuery>
  </ClassificationQuery>
</ServiceQuery>
```

Ancak bir ebXML KS kullanıcısı, SQL ve ebRIM Filter sözdizimini veya ebXML KS'ye kayıtlı taksonomileri ve bu taksonomilerin kategorilerini bilmeyebilir. Bu nedenle, istemcilerin bir ebXML KS'yi kolay bir şekilde sorgulayabilmesi için, istemciler tarafından sıkça kullanılacağı düşünülen sorgular veya birden fazla meta veri sınıfının özniteliklerini içeren ve istemciler tarafından yazılması zor olan karmaşık sorgular, ebXML KS yöneticisi tarafından önceden yazılır ve bilgi modelinin *AdhocQuery* sınıfı kullanılarak ebXML KS'ye kayıt edilir. Bu şekilde ebXML KS'ye kaydedilen sorgulara "saklanmış sorgu" adı verilir. *AdhocQuery* sınıfı, bir sorgunun ne yaptığını ve hangi parametrelere ihtiyaç duyduğunu açıklayan bir "açıklama" (description) özniteliğine sahiptir. İstemci, ebXML KS'ye kayıtlı tüm sorguları listeleyerek sorguların açıklama bilgilerini okur ve hangi sorguyu kullanacağını belirler. İstemci daha sonra, koşturacağı sorgunun id'sini ve parametrelerini ebXML KS'ye göndererek sorguyu gerçekleştirir.

ebXML KS, hem anlık hem de saklanmış sorgu yönteminde istemcilerin, anahtar kelime-tabanlı (keyword-based) bir sorgu gerçekleştirmesine olanak sağlar. Çünkü her iki yöntemde de sorgu ifadeleri sadece meta veri sınıflarının özniteliklerinden oluşmaktadır. ebXML KS, istemcilerin içerik tabanlı (content-based) sorgular yapmasına da olanak sağlamaktadır. Ancak istemcilerin içerik tabanlı sorgu yapabilmesi için, içeriğin bir "içerik yönetim servisi" (content management service) tarafından önceden kataloglanmış olması gerekmektedir. Bir içerik yönetim servisi, genellikle "içerik kataloglama" (content cataloging) ve "içerik doğrulama" (content validation) servislerinden oluşur. Bu bölümde sadece içerik kataloglama servisi ele alınacaktır.

Bir içerik kataloglama servisi, istemcilerin ebXML KS'ye yayınladıkları dokümanların (içeriğin), ebXML KS tarafından otomatik olarak kataloglanması sağlar. İçerik kataloglama servisinin doküman içerisindeki hangi bilgileri kataloglayacağını,

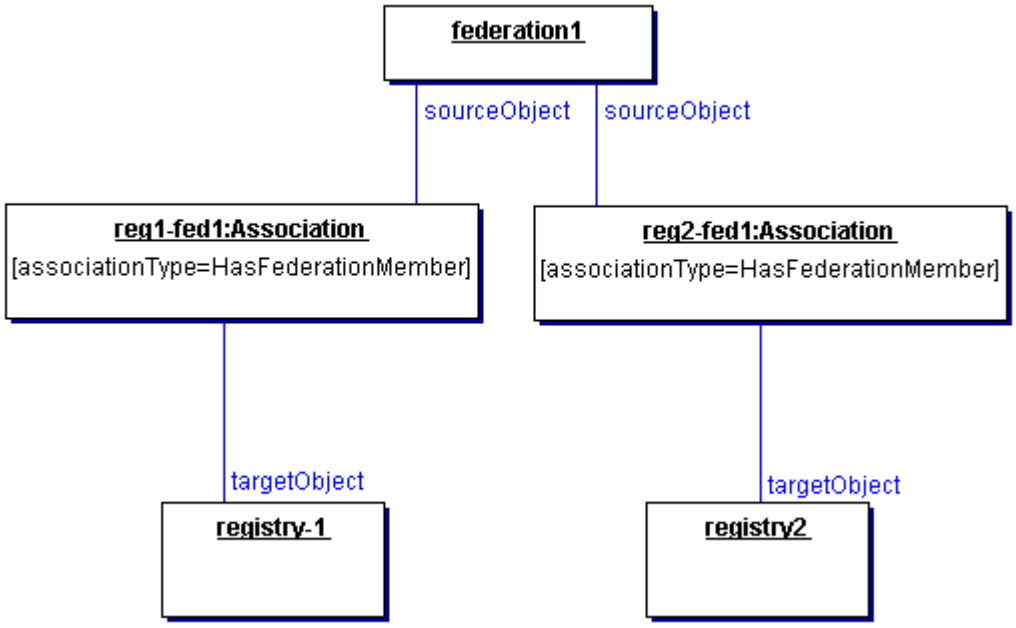
dokümana özel bir "kontrol dosyası" belirler. Örneğin, bir WSDL dokümanı ile bir OGC yetenekler (capabilities) dokümanı farklı kontrol dosyaları kullanılarak kataloglanır. İçerik kataloglama servisi, kontrol dosyasını kullanarak dokümandan elde ettiği içerik bilgilerini, *ExtrinsicObject* sınıfına, *Slot* sınıfını kullanarak öznitelik olarak ekler. İstemciler, bu ilave öznitelikleri kullanarak içerik tabanlı sorgu gerçekleştirebilirler.

OGC (2006), ebXML KS'ler de içerik tabanlı sorgu yapmaya olanak sağlayan başka bir yöntem önermektedir. OGC tarafından geliştirilen CSW-ebRIM profilinde, bir ebXML KS'nin depo bileşeninde yer alan XML dokümanlarının, XPath sorgu dili kullanılarak içerik tabanlı sorgulanabileceğini belirtilmektedir.

Yayınlama arayüzü: Bir sağlayıcı, sahip olduğu bilgi kaynaklarını bir ebXML KS'ye yayınlamak için, ebXML KS'nin yaşam süreci yönetim arayüzünü kullanır. Yaşam süreci yönetim arayüzünün "*SubmitObject*" metodu, sicil nesnelerinin ve depo elemanlarının bir ebXML KS'ye kayıt edilmesini sağlar. Bir ebXML KS'ye, bilgi kaynaklarını yayınlama şu şekilde gerçekleşir. Sağlayıcı, bir bilgi kaynağını tanımlamak için kullanacağı sicil nesnelerinin öznitelik değerlerini, bir istek mesajına (SubmitObjectRequest) elle yazar ve istek mesajını, bilgi kaynağı ile birlikte *SubmitObject* metodunu kullanarak ebXML KS'ye gönderir. ebXML KS, istek mesajı ile gelen bilgi kaynağını depo bileşenine kaydeder. ebXML KS, yine istek mesajı ile gelen sicil nesnelerinin özniteliklerini kullanarak, sicil bileşeninde ilgili nesnelerin örneklerini (instance) yaratır. Örneğin, konumsal bir veri setini tanımlayan XML formatındaki bir ISO 19115 meta veri dosyasının, bir ebXML KS'ye yayınlanması şu şekilde gerçekleşir. Sağlayıcı, istek mesajına, *ExtrinsicObject* sınıfının öznitelik değerlerini elle yazar ve istek mesajını XML dokümanı ile birlikte ebXML KS'ye gönderir. ebXML KS, UUID formatında bir id üretir ve XML dokümanını depo bileşenine bu id ile kaydeder. ebXML KS, sicil tarafında ise aynı id ye sahip bir *ExtrinsicObject* nesnesi yaratır.

Diğer arayüzler: ebXML KS'nin kullanıcılara sunduğu önemli arayüzlerden birisi de "olay bildirme" (event notification) arayüzüdür. Olay bildirme arayüzü, ebXML KS de belirli bir olayın gerçekleşmesi durumunda, bu olay ile ilgilenen kullanıcıların haberdar edilmesine olanak sağlar. Örneğin istemci, ebXML KS'ye yeni bir WFS servisinin yayınlanması durumunda veya kullandığı bir Web servisinin güncellenmesi durumunda, ebXML KS'nin kendisini uyarmasını isteyebilir. ebXML KS, bir olayın gerçekleştiğini kullanıcıya genellikle e-mail göndererek bildirir.

Federasyon desteği: ebRIM, ebXML KS'ler arasında bir katalog federasyonunun kurulmasına olanak sağlayan, birlikte çalışan kataloglar veya başka bir ifadeyle federasyon bilgi modeline sahiptir. Federasyon bilgi modeli, "*Federation*" ve "*Registry*" sınıflarından oluşur. *Federation* sınıfı, katalog federasyonuna ev sahipliği yapan ebXML KS'yi, *Registry* sınıfı ise federasyona üye olan bir ebXML KS'yi temsil etmek için kullanılır. Katalog federasyonu, ebXML KS'lerin birinde bir *Federation* nesnesi yaratılarak kurulur. Bu federasyona üye olmak isteyen diğer ebXML KS'ler ise bir *Registry* nesnesi yaratırlar. Bir ebXML KS'nin bir federasyona üyeliği, *Association* sınıfının *HasFederationMember* ilişki tipi kullanılarak kurulur (Şekil 10).



Şekil 10. EbXML kataloglarında katalog federasyonunun kurulması (OASIS, 2005b).

Bir ebXML KS'ye gönderilen sorgunun, katalog federasyonu açısından önemli olan iki parametresi vardır. Bunlar, *federated* ve *federation* parametreleridir. Eğer bir ebXML KS'ye gönderilen sorgunun *federated* parametresinin değeri *true* ise ve ebXML KS bir federasyona üye ise, sorgu aynı federasyona üye olan diğer tüm ebXML KS'lere gönderilir. Bir ebXML KS birden çok federasyonun üyesi olabilir. OASIS, bir katalog federasyonunun en önemli özelliği olan ve bir sorgunun federasyon içerisinde sadece sorguyu cevaplayabilecek olası kataloglara gönderilmesini sağlayan bir sorgu yönlendirme mekanizması geliştirmemiştir. Bu nedenle, bir ebXML KS'ye gönderilen sorgu, ebXML

KS'nin üyesi olduğu tüm federasyonlara gönderilir. Bununla birlikte, OASIS sorgu yönlendirmenin kullanıcılar tarafından tanımlandığı bir yöntem geliştirmiştir. Eğer kullanıcı, sorgunun hangi federasyon tarafından cevaplanabileceğini biliyorsa, sorgunun *federation* parametresine ilgili federasyonun id'sini yazar. Bu durumda sorguyu alan ebXML KS, sorguyu sadece ilgili federasyona gönderir. OASIS, federasyon bilgi modeli ile bir katalog federasyonunun nasıl kurulabileceğini ve katalog servislerinin federasyona nasıl üye olabileceklerini tanımlamıştır. Ancak katalog federasyonunun neye göre kurulacağı, hangi ölçütlerin göz önünde bulundurulacağı gibi konular hem ebRIM hem de ebRS standartlarında tanımlanmamıştır.

Katalog uygulama profili: ebXML katalog servisleri, özelleştirilebilir bir bilgi modeline sahiptir. ebRIM bilgi modelinin özelleştirilmesini gerektiren iki durum söz konusudur. Birincisi, ebXML katalog servislerinin farklı ilgi alanları tarafından kullanılmak istenilmesi durumunda bilgi modeli ilgi alanının ihtiyaçlarını karşılamak için özelleştirilebilmektedir. ebRIM bilgi modelinin bir ilgi alanına özel gerçekleştirimi, genellikle bir "uygulama profili" tarafından tanımlanmaktadır. Örneğin, ebXML katalog servislerinin, sağlık hizmetleri alanında kullanılabilmesi için HL7 (Healthcare Level 7) profili geliştirilmiştir. Bilgi modeli ikinci olarak, farklı ilgi alanları tarafından kullanılan katalog servislerine bilgi modeli desteği sağlamak için özelleştirilebilmektedir. Örneğin OGC, CSW katalog servislerinin bilgi modelini tanımlamak için CSW-ebRIM profilini geliştirmiştir (OGC, 2006). CSW-ebRIM profiline uygun olarak geliştirilen katalog servisleri, ebRIM bilgi modelini ve CSW belirtimi tarafından tanımlanan arayüzleri gerçekleştirirler. CSW, ebXML katalog servislerinden sadece bilgi modeli desteği almıştır. ebXML KS bilgi modelinin özelleştirilebilirliği, CSW'nin ebRIM profilinin açıklandığı bölümünde daha detaylı olarak ele alınacaktır.

Anlamsal destek: ebXML KS, Web servisi tanımlamalarını anlamsal olarak zenginleştirmeye olanak sağlayan bir bilgi modeline sahiptir. Bu alandaki başlıca çalışmalar (Doğaç, 2003; Doğaç vd., 2003; Doğaç vd., 2004; Doğaç vd., 2005) olarak sayılabilir. Bu çalışmalarda genel olarak iki yolla anlamsal zenginleştirme uygulanmıştır. Doğaç (2003) ve Doğaç vd., (2003) te uygulanan yöntemde, Web servisleri belirli "alan ontolojilerine" (domain ontologies) dayalı olarak DAML-S dilinde tanımlamış ve bu tanımlamalar, ebRIM bilgi modelinin *ClassificationScheme* ve *ClassificationNode* sınıflarına resmedilmiştir. Ancak bu yöntem yalnızca taksonomilerin, bir dereceye kadar temsil edilmesine olanak tanıyacaktır. Diğer yandan, hiyerarşik yapıdaki taksonomilerin

aksine çizge (graph) yapıdaki ontolojiler ise bu yolla temsil edilemeyecektir. Çünkü bu yöntem, ontolojinin iki sınıfı ya da kavramı arasındaki bir ilişkiyi temsil edecek bir ebXML KS mekanizması kullanmamaktadır. Bu durumu, ISO 19119 "Coğrafi servis taksonomisi" (Geographic Services Taxonomy) bağlamında basit bir örnekle açıklamak gerekirse, verilen bir zaman ve iki nokta arasında en kısa yolu belirleyerek istemciye sunan bir "en kısa yol servisi" alınabilir. Bu servisin hem "konumsal" hem de "zamansal" bir servis olduğu bilgisinin doğrudan temsil edilmesi, zamansal ve konumsal kategorilerinin taksonomide bulunduğu hiyerarşik düzeye bağlı olacaktır. Çünkü ebXML KS bilgi modeli bir kategorinin yalnızca birinci derecedeki hiyerarşik "üst"üne ait bilgi içerir.

Doğaç vd. (2004) ve Doğaç vd. (2005) gibi daha sonraki çalışmalarda uygulanan ikinci yöntemde ise, Web servisleri arasında kullanılan ontoloji gereği temsil edilmesi istenen ilişkilerin, *Association* nesneleri vasıtası ile temsil edilmesi yoluna gidilmiştir. Bunun için, OWL Lite dilinde yapılan Web servisi tanımlamaları, çalışma ekibi tarafından geliştirilen bir ayrıştırıcı (parser) yardımı ile otomatik olarak, ebXML KS bilgi modeli nesnelerine resmedilmiştir. Ayrıştırıcı, her bir OWL sınıfı için bir *ClassificationNode* nesnesi, her bir nesne özelliği için de bir *Association* nesnesi oluşturmaktadır. Temsil edilmek istenen ilişkinin ebXML KS bilgi modelinin yerleşik (built-in) *Association* nesne tipleri kapsamında olmaması durumlarına yönelik olarak, yeni *Association* nesne tipi tanımlamaları eklenmiştir. Bunlardan bazıları, "*subClassOf*" "*subPropertyOf*", "*inverseOf*", "*intersectionOf*" ve "*succeed*" gibi *Association* tipleridir. *Association* nesnelerine yüklenen anlamlar artık herhangi bir uygulama programı, SQL gibi bir sorgu dili ile ortaya çıkarılabilir. Doğaç vd. (2005) bu amaçla SQL için tanımlanmış olan "saklanmış sorgu" mekanizmasını kullanmıştır. Saklanmış sorgular üzerinden de çeşitli parametrik sorgular tanımlanmıştır. Bu sayede örneğin bir Web servisinin, kendisinden sonra kullanacağı Web servislerini ekranda kullanıcıya gösteren bir parametrik sorgu gerçekleştirilebilmektedir. Bu yolla, bir "eğlence servisi" vasıtası ile bir eğlence planı yapmaya çalışan bir kullanıcı, bu servisten önce sırası ile "otel rezervasyon" ve "uçak rezervasyon" servislerine de ihtiyacı olduğunu belirleyebilir. Ancak belirli bir uygulama için hangi servislerin kullanılabileceğinin bu şekilde belirlenmesi, tümüyle "önceden tanımlı" (hard-coded) ilişkiler _ki örnekte *succeed* ilişkisi vasıtasıyla_olduğu için, anlamsal bir servis bulma ve düzenleme olarak nitelendirilemez.

Literatürde yukarıda anılan çalışmalara oranla daha yeni bir çalışma olarak tespit edilen Di vd. (2006)'nın çalışması, konumsal verilere yönelik olması ile Doğaç ve ekibinin

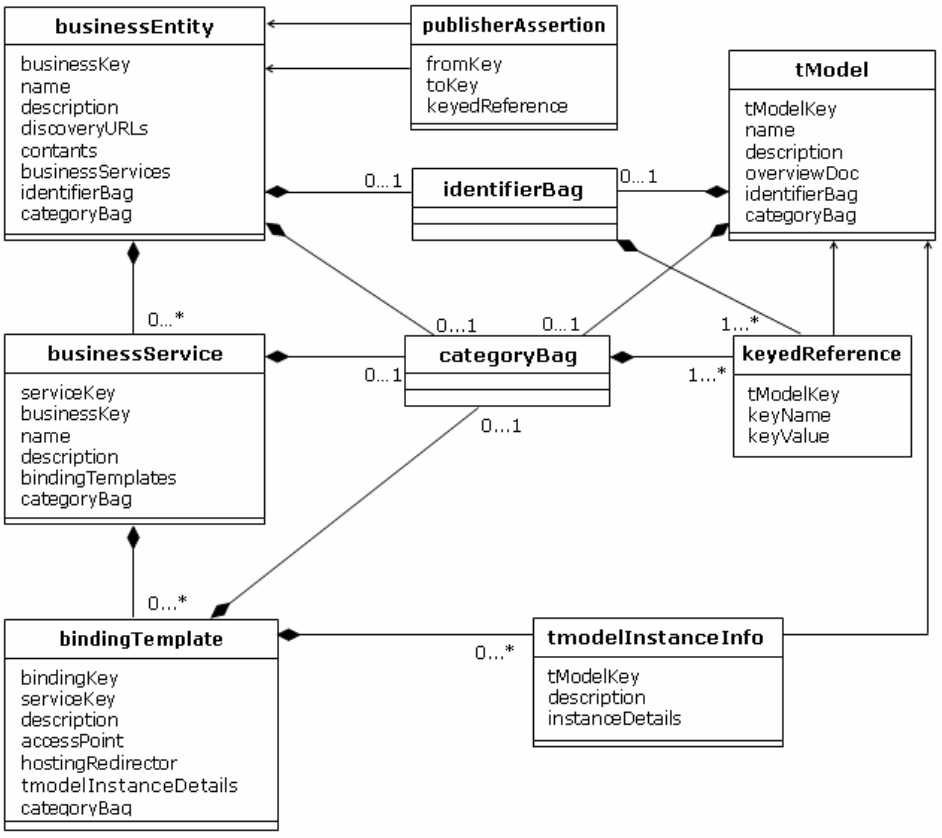
yukarıda anılan çalışmalarından farklı, ancak anlamsal desteğin bir ebXML KS de nasıl ve ne dereceye kadar temsil edilebileceğinin değerlendirilmesi açısından benzerdir. Di vd. (2006) veri ve servis tanımlama için OWL-S dilini kullanmış ve bu tanımlamaları "RDF üçlüleri" olarak anılan kaynak (resource/subject), özellik (property/predicate) ve değer (value/object) formatına dönüştürerek "bilgi tabanı"na koymuştur. Bilgi tabanında bu şekilde kavramsal olarak tanımlı hale gelen veri ve servislerin örnekleri ise OWL-S tanımlamalarının ebXML KS bilgi modeline resmedilmesi ile elde edilen şemaya göre ebXML KS ye konmuştur. Bu resmetmede OWL özellikleri (property) katalog nesnelerinin (registry object) *Slot* bileşenlerine, OWL aksiyomları da katalog nesneleri arasındaki ilişki tiplerine (Association) karşılık gelmiştir. Bu, aynen Doğaç ve ekibinin, bu resmetme için geliştirdikleri bir yazılımı kullanarak uyguladığı yöntemdir. Di vd. (2006) verilen örnek "heyelan riskini" belirleyen bir servis ihtiyacındaki kullanıcının hangi bileşen servisleri kullanarak böyle bir servis düzenleyebileceğinin belirlenmesidir. Söz konusu servis düzenlemenin, Di vd. (2006) de anlatılan tarzda yapılması durumunda, yine aynı kaynakta söz edildiği gibi bir çıkarsama ve dolayısıyla bilgi tabanına gerek yoktur. Çünkü örnekteki servis düzenleme tamamen hiyerarşik bir tarzda servislerin ne tip girdi ve çıktılar kullandığının ve bunların hangi veri setleri olduğu ilişkilerine göre yapılmaktadır. Bu bilgiler zaten ebXML KS ye yukarıda açıklanan şekilde, yani girdi/çıktı tipleri *Slot* ve servisin hangi veriler üzerinde çalıştığı da *Association* olarak aktarılmıştı. Bu durumda da girdi çıktı tipi eşleştirmeleri aslında tamamen sözdizimsel (syntactic) dir. Di vd. (2006)'de söz edilen bilgi tabanı ve çıkarsamacı ancak, Di vd. (2006)'nın çalışmalarının bir sonraki adımında gerçekleştirmeyi düşündükleri tam otomatik servis düzenleme için olabilir.

Sonuç olarak, ebRIM OWL ya da OWL-S dilinde anlamsal olarak tanımlanmış bilginin temsiline olanak tanımaktadır. Ancak yukarıda Doğaç ve ekibinin çalışmaları ile ilgili olarak ta belirtildiği üzere, anlamsal bilgi bir ebXML KS de ancak "önceden kodlandığı" (hard-coded) kadarı ile temsil edilebilir. Çünkü bu durumda herhangi bir çıkarsama söz konusu olmayacaktır. Zaten bir ebXML KS de bir çıkarsamacı değildir. Bir örnekle açıklamak gerekirse, bir bilgi tabanı ve çıkarsamacının söz konusu olduğu bir bilgi temsilinde "erkekler oje sürmez" ve "Harun Efe erkektir" aksiyomları, "Harun Efe oje sürmez" çıkarsaması için yeterlidir ve örneğin bir "Tanımlama Mantığı" (Description Logics / DL) çıkarsamacısı bunu çıkarsayabilir. Oysa ebXML KS de her üç aksiyomun da katalogda bulunması gerekir.

2.3.3. Evrensel Tanımlama Bulma ve Birleştirme

Evrensel Tanımlama Bulma ve Birleştirme (Universal Description Discovery and Integration / UDDI), Microsoft, IBM ve Ariba yazılım firmalarının kurduğu, "UDDI.org" olarak adlandırılan bağımsız bir ortaklık tarafından geliştirilmiştir. UDDI belirtiminin ilk sürümü, Eylül 2000 de bu ortaklık tarafından bilişim dünyasına duyuruldu (Cerami, 2002). Herkesin kullanımına açık ilk UDDI katalog servisleri, Mayıs 2001 de Microsoft ve IBM firmaları tarafından hizmete sunuldu. UDDI belirtiminin yenilenen ikinci sürümü, Haziran 2001 de ilan edildi. UDDI.org, Temmuz 2002 de OASIS birliğine dahil oldu (Newcomer, 2002) ve UDDI'ın ikinci sürümü Mayıs 2003, şu an yürürlükte olan son sürümü (Version 3.0.2) ise Şubat 2005 de OASIS tarafından standart olarak kabul edildi.

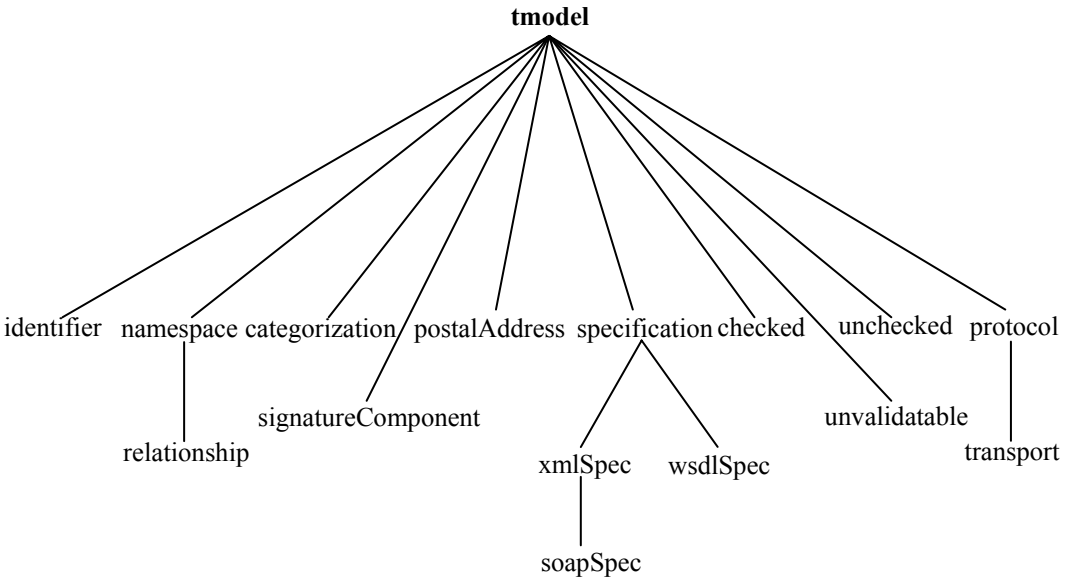
Bilgi modeli: UDDI, Web servislerini ve servis sağlayıcılarını tanımlamaya olanak sağlayan bir bilgi modeline sahiptir. UDDI bilgi modeli, dört temel sınıftan oluşur. Bunlar, "*businessEntity*", "*businessService*", "*bindingTemplate*" ve "*tmodel*" sınıflarıdır (Şekil 11).



Şekil 11. UDDI bilgi modeli (UDDI.org, 2000).

BusinessEntity sınıfı, servis sağlayıcılar ile ilgili meta verileri tanımlamak için kullanılır. *BusinessEntity* sınıfı, ebRIM bilgi modelinde, bilgi kaynağı sağlayıcılarını tanımlamak için kullanılan "*Organization*" sınıfına karşılık gelir ve servis sağlayıcıları ile ilgili iletişim ve sınıflandırma bilgilerini içerir. Bir servis sağlayıcısı tarafından sunulan Web servisleri, *businessService* sınıfı kullanılarak tanımlanır. *BindingTemplate* sınıfı ise, bir Web servisini çağırmak için gerekli olan bilgileri içerir. Servislerin erişim adresleri, *bindingTemplate* sınıfının "*accessPoint*" özniteliği tarafından tutulur. *businessService* ve *bindingTemplate* sınıfları sırasıyla, ebRIM bilgi modelindeki "*Service*" ve "*ServiceBinding*" sınıflarına karşılık gelmektedir.

UDDI bilgi modelinin en önemli ve anlaşılması bir o kadar da zor olan sınıfı, "*technical model*" veya kısaca "*tmodel*" sınıfıdır. UDDI belirtimi (OASIS, 2004), *tmodel* sınıfının ne işe yaradığını ve görevinin ne olduğunu tam olarak ortaya koymamıştır. Bu durum literatürde de dile getirilmektedir (Cerami, 2002; Yu, 2006). *Tmodel*, bir tip tanımlama mekanizmasıdır ve servis sağlayıcılarının ve Web servislerinin tiplerini tanımlamaya olanak sağlamaktadır. Bir UDDI katalog servisinin desteklemesi gereken tip sistemleri ya da diğer adıyla "taksonomiler", UDDI belirtimi tarafından önceden tanımlanmıştır. UDDI katalog servisleri tarafından kullanılan taksonomiler, UDDI tip taksonomisine (Şekil 12) göre sınıflandırılırlar.



Şekil 12. UDDI tip taksonomisi (OASIS, 2004).

Servis sağlayıcılarının ve Web servislerinin tiplerini tanımlamak için kullanılan taksonomiler, UDDI tip taksonomisindeki "sınıflandırıcı" (categorization) taksonomiler kategorisine girerler. Servis sağlayıcılarının kimliklerini (id) tanımlamak için kullanılan taksonomiler ise, "tanımlayıcı" (identifier) taksonomiler kategorisine girerler. NAICS (North American Industry Classification System) ve ISO 3166 (Geographic Code System) sınıflandırıcı, D-U-N-S (Dun & Bradstreet Number Identifier System) ve TRSICS (Thomas Register Supplier Identifier Code System) ise tanımlayıcı taksonomilere örnek olarak gösterilebilirler. Aşağıdaki *tmodel* tanımı, UDDI katalog servislerinde NAICS taksonomisini tanımlamak için kullanılmaktadır. *Tmodel* tanımı, NAICS taksonomisinin, UDDI tip taksonomisine göre bir sınıflandırıcı taksonomi olduğunu göstermektedir. *Tmodel*, ebRIM bilgi modelinde taksonomileri tanımlamak için kullanılan *ClassificationScheme* sınıfına eşdeğerdir.

```
<tModel tModelKey="uddi:uddi.org:ubr:categorization:naics:1997">
  <name>ntis-gov:naics:1997</name>
  <description xml:lang="en">
    Industry Category System: NAICS (1997 release)
  </description>
  <overviewDoc>
    <overviewURL useType="text">
      http://uddi.org/taxonomies/UDDI_Taxonomy_tModels.htm#NAICS
    </overviewURL>
  </overviewDoc>
    <categoryBag>
      <keyedReference tModelKey="uddi:uddi.org:categorization:types"
        keyName="uddi-org:types:categorization"
        keyValue="categorization" />
      <keyedReference tModelKey="uddi:uddi.org:categorization:types"
        keyName="uddi-org:types:checked"
        keyValue="checked" />
    </categoryBag>
</tModel>
```

UDDI katalog servislerinde, servis sağlayıcılarının ve Web servislerinin tiplerini tanımlamak için "*categoryBag*" sınıfı, servis sağlayıcılarının id'lerini tanımlamak içinde "*identifierBag*" sınıfı kullanılır. *CategoryBag* sınıfı sınıflandırıcı, *identifierBag* sınıfı ise tanımlayıcı taksonomileri kullanırlar. Her iki sınıfında kendilerine ait öznitelikleri yoktur. *CategoryBag* ve *identifierBag* sınıfları, "*keyedReference*" sınıfını içerirler. *KeyedReference* sınıfının, "*tmodelKey*", "*keyName*" ve "*keyValue*" olarak adlandırılan üç özniteliği vardır. *TmodelKey* özniteliği, tip ya da kimlik tanımlamada kullanılacak olan taksonomiyi

gösterir. *KeyName* özniteliği, taksonomisi içerisindeki bir kategoriyi, *keyValue* ise kullanılan kategorinin taksonomi içerisindeki kodunu (id'sini) göstermektedir. Örneğin aşağıdaki *businessEntity* tanımı, bir servis sağlayıcısının NAICS ve ISO 3166 taksonomilerine göre sınıflandırılmasını göstermektedir. *BusinessEntity* tanımı, sağlayıcının NAICS taksonomisine göre bir eğitim kurumu olduğunu ve ISO 3166 taksonomisine göre de Trabzon'da faaliyet gösterdiğini tanımlamaktadır.

```
<businessEntity
  businessKey="88378742-e829-49d4-9c1b-00152a7f5889">
  <name xml:lang="tr-tr">Karadeniz Technical University</name>
  <contacts>
    …
  </contacts>
  <categoryBag>
    <keyedReference
      tModelKey=" uddi:uddi.org:ubr:categorization:iso3166"
      keyName="Trabzon"
      keyValue="513527"/>
    <keyedReference tModelKey=" uddi:uddi.org:ubr:categorization:naics:1997"
      keyName="Colleges and Universities, n.e.c."
      keyValue="8221"/>
  </categoryBag>
</businessEntity>
```

CategoryBag sınıfı, ebRIM bilgi modelindeki *Classification* sınıfına, *identifierBag* sınıfı ise *ExternalIdentifier* sınıfına karşılık gelmektedir. UDDI bilgi modelindeki meta veri sınıflarının, ebRIM bilgi modelindeki eşdeğerleri Tablo 2'de yer almaktadır.

Tablo 2. UDDI ve ebRIM meta veri sınıfları arasındaki eşleşmeler

UDDI	**ebRIM**
businessEntity	Organization
businessService	Service
bindingTemplate	ServiceBinding
tmodel	ClassificationScheme ExternalLink
categoryBag	Classification
identifierBag	ExternalIdentifier

Tmodel sınıfı ayrıca, bir taksonomiyi temsil etmeyen belirtimleri tanımlamak için de kullanılmaktadır. Örneğin bir Web servisini tanımlayan WSDL dokümanı, UDDI da

tmodel sınıfı kullanılarak tanımlanır. *Tmodel* sınıfı, WSDL dokümanının adı ve adresi ile ilgili bilgiler içerir. WSDL dosyasının adresi, *tmodel* sınıfının "*overviewURL*" özniteliği tarafından tutulur. UDDI katalog servislerinde WSDL dokümanlarını tanımlamak için kullanılan *tmodel* nesneleri, UDDI tip taksonomisinin "*wsdlSpec*" kategorisi kullanılarak sınıflandırılırlar. ebRIM bilgi modelindeki *ExternalLink* nesnesinin, bir ebXML KS'nin depo bileşeninde yer almayan elektronik dokümanları tanımlamak için kullanıldığını ve dokümanların adreslerini içerdiğini hatırlarsak, *tmodel* sınıfının bu kullanım şekli ile *ExternalLink* nesnesi ile eşleştiğini görmüş oluruz. Aşağıdaki XML kodu, tarafımızdan geliştirilen ve Microsoft firmasının UDDI katalog servisine kayıt edilen "Toplama" Web servisini tanımlayan WSDL dokümanının, katalogdaki *tmodel* tanımını göstermektedir.

```
<tModel tModelKey="16f5e541-1a86-4df4-85da-c259c03117e9">
  <name>ToplamaWS tmodel</name>
  <description xml:lang="tr-tr">
    Toplama Web servisine ait WSDL dokümanını tanımlayan tmodel
  </description>
  <overviewDoc>
    <description xml:lang="tr-tr">ToplamaWS WSDL dokümanı</description>
    <overviewURL>http://jeodeziprog:8000/ToplamaWS.wsdl</overviewURL>
  </overviewDoc>
        <categoryBag>
          <keyedReference
            tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
            keyName="Specification for a web service described in WSDL"
            keyValue="wsdlSpec"/>
        </categoryBag>
</tModel>
```

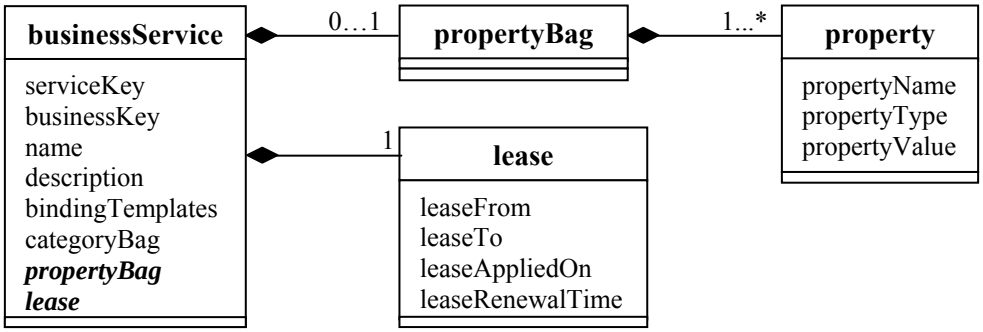
Bilgi kaynağı türü: UDDI, ebXML KS de olduğu gibi her türlü elektronik dokümanı tanımlamaya olanak sağlayan bir bilgi modeline sahip değildir. UDDI, sadece Web servisleri ve servis sağlayıcıları ile ilgili meta verileri içeren bir katalog servisidir. Bu nedenle UDDI'ı "servis katalogu" olarak adlandırmak daha doğru olur.

Anlatım gücü: UDDI, ebRIM de olduğu gibi nesne yönelimli bir bilgi modeline sahip değildir. Literatürde, UDDI bilgi modelinin hiyerarşik bir yapıya sahip olduğu, bilgi modelindeki meta veri sınıfları arasında sadece içerme (containment) ilişkisinin olduğu ve bilgi modelinin bir XML şema tarafından tanımlandığı belirtilmektedir (UDDI.org, 2000; OASIS, 2004). Her ne kadar literatürde UDDI bilgi modeli için "hiyerarşik bir yapıya sahiptir" denilse de, aslında UDDI bilgi modeli "varlık" (entity) tabanlıdır ve varlıklar

arasında sadece oluşum (composition) ve ilişki (association) ilişkisi vardır. Bu nedenle UDDI bilgi modelinin anlatım gücü zayıftır.

Ayrıca UDDI bilgi modeli, ebRIM bilgi modelinde meta veri sınıfları arasındaki ilişkileri tanımlamak için kullanılan *Association* sınıfı gibi bir sınıfa sahip değildir. Bilgi modeli, sadece servis sağlayıcılar (*businessEntity*) arasındaki ilişkileri tanımlamaya olanak sağlayan "*publisherAssertion*" sınıfına sahiptir. Servis sağlayıcılar arasındaki ilişkiler, "UDDI ilişkiler taksonomisi" tarafından tanımlanmaktadır. UDDI ilişkiler taksonomisi, 3 ilişki tipine sahiptir. "*Peer-peer*" ilişki tipi, sağlayıcıların aynı düzeyde olduğunu, "*parent-child*" ilişki tipi bir sağlayıcının diğerinin alt birimi olduğunu, "*identity*" ilişki tipi ise sağlayıcıların aynı olduğunu göstermek için kullanılmaktadır.

Genişletilebilirlik: UDDI, genişletilebilir bir bilgi modeline sahiptir ve kullanıcıların bilgi modelinde genişletme yapmalarına olanak sağlamaktadır. ShaikhAli (2003), UDDI bilgi modelinde bazı genişletmeler yapmış ve UDDIe (Extended UDDI) olarak adlandırılan açık kaynak kodlu bir UDDI katalog servisi geliştirmiştir (Şekil 13).



Şekil 13. UDDI bilgi modelinde yapılan genişletmeler (ShaikhAli, 2003).

ShaikhAli (2003), UDDI bilgi modeline, "*lease*" ve "*propertyBag*" olarak adlandırdığı iki yeni sınıf eklemiştir. Her iki sınıf ta *businessService* sınıfını genişletmek için kullanılmaktadır (Şekil 13). *PropertyBag* sınıfı, kullanıcıların servis tanımlarına ilave öznitelikler eklemesine olanak sağlamaktadır. *PropertyBag*, seçmeli bir sınıftır ve bir ya da daha fazla "*property*" sınıfı içerebilmektedir. *Property* sınıfı, "*propertyName*", "*propertyType*" ve "*propertyValue*" olarak adlandırılan üç özniteliğe sahiptir. *PropertyName*, servis tanımına eklenecek olan özniteliğin adını, *propertyType* özniteliğin tipini, *propertyValue* ise özniteliğin değerini göstermektedir. Özniteliğin tipi, kullanıcılar tarafından tanımlanmaktadır. *PropertyBag* sınıfı, ebRIM bilgi modelinde, meta veri

sınıflarına kullanıcılar tarafından ilave öznitelikler eklenmesine olanak sağlayan *Slot* sınıfı ile aynı göreve sahiptir. Örneğin, internet üzerinden otomatik imar çapı çıkaran bir Web servisi sunan bir belediye, servisin belirli bir ücret karşılığında kullanılabileceğini göstermek için, servis tanımına aşağıdaki gibi bir öznitelik ekleyebilir.

```
<propertyBag>
  <property>
    <propertyName> Servis_Kullanım_Ücreti </propertyName>
    <propertyType> Currency (YTL) </propertyType>
    <propertyValue> 30 </propertyValue>
  </property>
</propertyBag>
```

Zorunlu olan *lease* sınıfı ise, bir Web servisinin katalog servisinde sağlayıcının belirlediği zaman diliminde sunulmasını sağlamaktadır. Örneğin, programlarında bir takım güncellemeler yapan bir yazılım firması, müşterilerinin güncellemeleri indirmek için kullanacakları Web servisini, *lease* sınıfı sayesinde sadece bir haftalığına katalog servisinde sunabilir. *Lease* sınıfı, "*leaseFrom*", "*leaseTo*", "*leaseAppliedOn*" ve "*leaseRenewalTime*" olarak adlandırılan dört özniteliğe sahiptir. *LeaseFrom* özniteliği, katalogun servisi yayınlamaya başlayacağı tarihi, *leaseTo* özniteliği yayınlamanın durdurulacağı tarihi, *leaseAppliedOn* özniteliği servisin kataloga kaydedildiği tarihi ve *leaseRenewalTime* özniteliği ise, yayınlanma süresi dolan bir Web servisinin yeniden yayınlanmaya başlandığı tarihi göstermektedir. Tarihler, "ay/gün/yıl saat:dakika:saniye" formatında girilmektedir. Katalog servisi, yayınlanma süresi dolan Web bir servisini, sağlayıcı yeni bir yayınlanma tarihi belirtmez ise katalogdan otomatik olarak kaldırmaktadır.

Katalog arayüzleri: UDDI katalogları, yayınlama ve sorgu arayüzü gibi, bir katalog servisinin kullanıcılarına sunması gereken temel arayüzlerin yanı sıra, seçmeli olan güvenlik (security) arayüzüne, sorumluluk ve sahiplik transfer (custody and ownership transfer) arayüzüne ve abonelik (subscription) arayüzüne sahip olabilirler. Güvenlik arayüzü ile sorumluluk ve sahiplik transfer arayüzü, bu çalışmanın kapsamı içerisinde yer almadığı için bu bölümde, söz konusu arayüzlere değinilmeyecektir. Bu arayüzler ile ilgili ayrıntılı bilgi UDDI belirtiminde (OASIS, 2004) bulunabilir.

Bir servis sağlayıcısı, sahip olduğu Web servislerini bir UDDI katalog servisine yayınlamak veya yayınladığı Web servisleri ile ilgili bilgileri güncellemek ya da silmek için UDDI'ın "yayınlama" (publication) arayüzünü kullanır. Servis sağlayıcı, yayınlama

arayüzünün "*save_service*" ve "*save_binding*" metotlarını kullanarak servis bilgilerini, "*save_business*" metodunu kullanarak sağlayıcı bilgilerini yayınlar. Adı geçen metotlar, parametre olarak, bilgi modelindeki ilgili sınıfların özniteliklerini alırlar. Örneğin *save_business* metodunun parametreleri, *businessEntity* sınıfının özniteliklerinden oluşur.

Abonelik arayüzü, istemcilere, bir UDDI katalogundaki değişiklikleri takip etme olanağı sağlar. Bir istemci, abonelik arayüzünün "*save_subscription*" metodunu kullanarak izlemek istediği değişikliklere abone olur. Örneğin istemci, bir UDDI da arama yaparak bulduğu ve kullandığı bir Web servisinin güncellenmesi durumunda veya belirli bir sağlayıcının kataloga yeni servisler yayınlanması durumunda, katalogun söz konusu değişiklikleri kendisine bildirmesini isteyebilir. İstemci, içerik değişikliklerinden iki şekilde haberdar olur. İstemci, ya abonelik arayüzünün "*get_subscriptionResults*" metodunu kullanarak değişik bilgilerini katalogdan kendisi alır ya da "*notify_subscriptionListener*" metodunu kullanarak değişikliğin katalog tarafından kendisine otomatik olarak bildirilmesini sağlar.

Bulma Arayüzü: Bir UDDI katalog servisini sorgulamak ve Web servislerini bulmak için UDDI'ın sorgu (inquiry) arayüzü kullanılır. Sorgu arayüzü, anahtar kelime eşleştirmeye (keyword match) dayanır. Web servisleri UDDI katalogunda, isimlerine göre, sağlayıcılarına göre, sunuldukları coğrafik yere göre veya belirli bir taksonomiye göre aranarak bulunur. Örneğin bir istemci, katalogda, sağlayıcısı KTÜ Jeodezi ve Fotogrametri Mühendisliği Bölümü olan, Trabzon'dan sunulan, ismi "Web Feature Service" olan ve NAICS taksonomisinin, "Bilgi Servisleri" (Information Services) kategorisine göre sınıflandırılan Web servislerini arayabilir. Ancak aynı servisi, eğer "Türkiye'den sunulan servisler" kategorisinde de kataloga kaydetmemişse, bu kategori altında bulamayacaktır. Çünkü ebXML KS den farklı olarak UDDI KS de taksonomiler temsil edilmemektedir. Bu açıdan ebXML KS daha esnektir.

UDDI bulma arayüzü bir filtreleme ya da SQL arayüzü içermez. Dolayısıyla ebXML KS de mümkün olan "anlık" sorgulamalar UDDI KS de yapılamaz.

UDDI KS de içerik tabanlı sorgulama da mümkün değildir. Bunun nedeni, UDDI KS nin bir depo (repository) bileşenine sahip olmaması ve Web servislerin WSDL tanımlarının UDDI KS de bulunmayıp doğrudan servis sağlayıcıda bulunmasındadır.

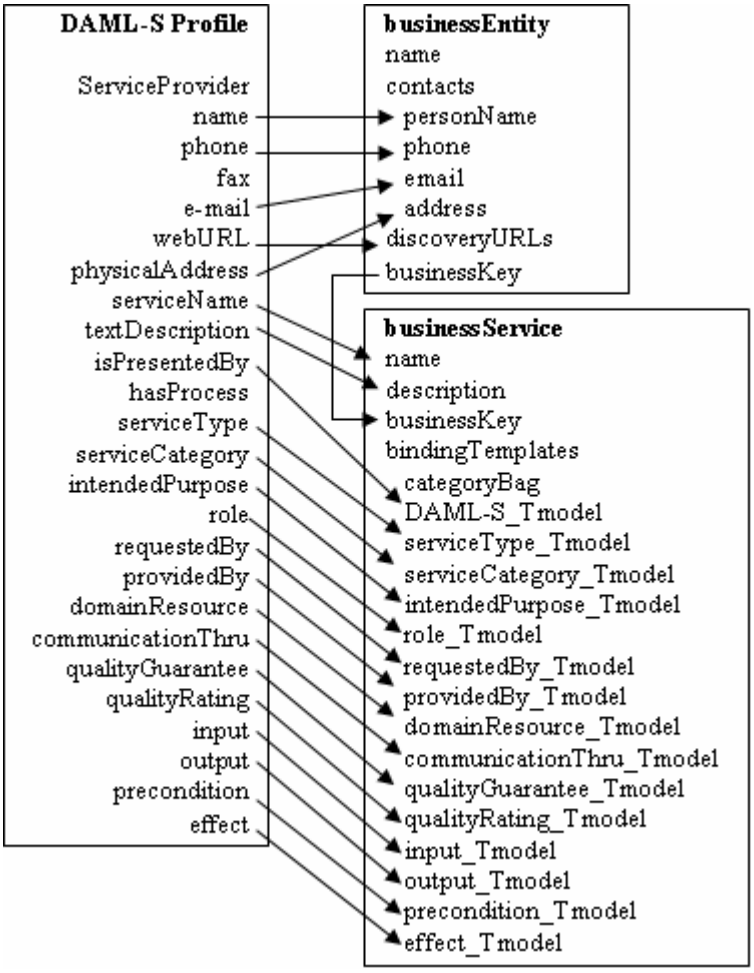
Katalog Uygulama Profili: UDDI, ebXML KS ve OGC CSW gibi katalog servislerine göre daha basit bir bilgi modeline sahiptir. Bir bilgi modelinin veya bilgi modeli olarak gerçekleştirilecek olan örneğin ISO 19115 gibi bir meta veri standardının, bir ilgi alanına

özel gerçekleştirimi, genellikle bilgi modelinin veya meta veri standardının bir alt seti olan bir uygulama profili tarafından tanımlanır. UDDI bilgi modeli, diğer katalog bilgi modellerine göre oldukça az sayıda meta veri sınıfı içermektedir. Bir UDDI katalog servisi sağlayıcısının, UDDI bilgi modelinin tamamını gerçekleştirmesi, sağlayıcıya fazla bir yük getirmeyecektir. Bu nedenle UDDI'ın bir ilgi alanı için özelleştirilmesine veya bir UDDI profilinin tanımlanmasına gerek yoktur. Literatürde ve Web de yapılan araştırmalarda da, bir UDDI profiline veya UDDI'ın bir ilgi alanına özel gerçekleştirimine rastlanamamıştır.

Anlamsal destek: UDDI, Web servisi tanımlamalarını anlamsal olarak zenginleştirmeye olanak sağlayan bir bilgi modeline sahiptir. UDDI kataloglarına anlamsal destek kazandırmak için önemli akademik çalışmalar yapılmıştır (Paolucci vd., 2002a; Paolucci vd., 2002b; Paolucci vd., 2004; Doğaç vd., 2002a; Doğaç vd., 2002b). Örneğin Paolucci vd. (2002a), Web servislerinin UDDI da sadece isimlerine göre değil de sahip oldukları yeteneklere göre bulunmasına olanak sağlayan anlamsal bir servis bulma mekanizması geliştirmiştir. Paolucci vd. (2002a), Web servislerini DAML-S dilini kullanarak anlamsal olarak tanımlamış ve DAML-S servis tanımlarını UDDI servis tanımlarına dönüştürmüştür (Şekil 14). Paolucci, DAML-S servis tanımlarında yer alan ve UDDI servis tanımlarına doğrudan dönüştürülemeyen DAML-S servis parametreleri için UDDI da yeni bir *tmodel* oluşturmuştur. Örneğin bir DAML-S servis tanımında yer alan sağlayıcı bilgileri, UDDI da *businessEntity* sınıfının özniteliklerine dönüştürülürken, servisin girdi ve çıktı parametrelerini tanımlamak için UDDI da "*input_tmodel*" ve "*output_tmodel*" olarak adlandırılan iki yeni tmodel yaratılmıştır (Şekil 14). *Input_tmodel* ve *output_tmodel*, DAML-S servis tanımında, bir Web servisinin girdi ve çıktı parametrelerini tanımlamak için kullanılan ontolojileri temsil etmektedirler. Paolucci vd. (2002a), oluşturulan yeni *tmodel* tanımlamalarını Web servislerini sınıflandırmak için kullanmıştır. Örneğin aşağıdaki *categoryBag* tanımı, bir Web servisinin girdi ve çıktı parametrelerini tanımlamak için, "finans ontolojisini" (*financialOntology*) temsil eden *input* ve *output* tmodel'lerinin kullanımını göstermektedir.

```
<categoryBag>
  <keyedReference tModelKey="UDDI input_tmodel id'si"
    keyName="input"
    keyValue="financialOntology:ticker"/>
  <keyedReference tModelKey="UDDI output_tmodel id'si"
    keyName="output"
    keyValue=" financialOntology:quote"/>
</categoryBag>
```

Paolucci vd. (2002a) ayrıca, servis istekleri ile servis yetenekleri arasındaki anlamsal eşleşme düzeylerini hesaplamak için bir eşleştirme algoritması tanımlamış ve bu algoritmayı gerçekleştiren bir eşleştirici (DAML-S/UDDI Matchmaker) geliştirmiştir. Eşleştirici, UDDI kataloglarını, yetenek tabanlı eşleştirmeyi gerçekleştiren ilave bir anlamsal katmanla genişletmek için kullanılmaktadır.



Şekil 14. DAML-S ve UDDI arasındaki eşleşmeler (Paolucci vd., 2002a).

2.3.4. OGC Katalog Servisleri

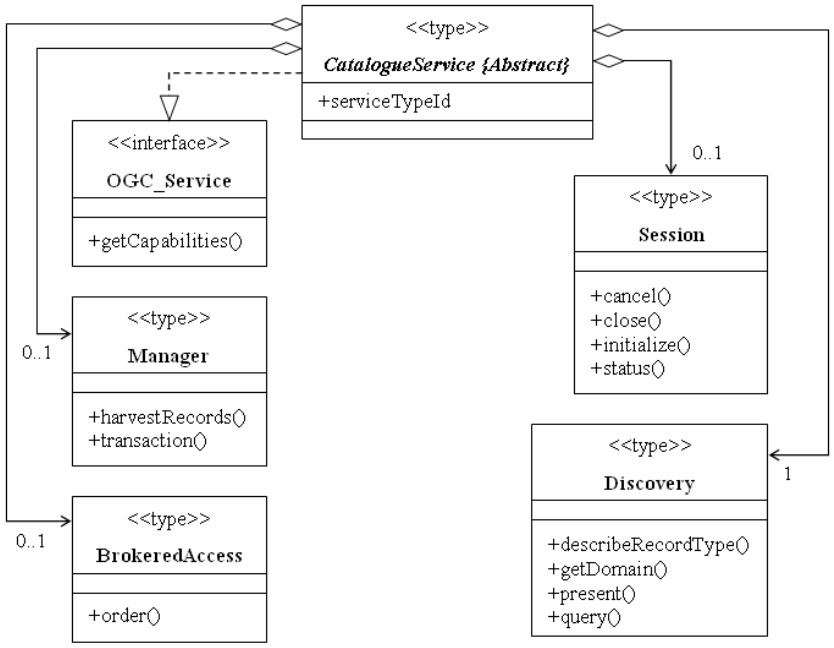
OGC, katalog servisleri ile ilgili çalışmalarına 1998 yılında başlamış ve Ağustos 1999 da katalog servis belirtiminin ilk sürümünü (OGC, 1999) yayınlanmıştır. OGC'nin "Katalog Revizyon Çalışma Grubu" (Catalogue RWG), kullanıcı gereksinimlerini ve bilgi teknolojilerindeki gelişmeleri dikkate alarak değişik zamanlarda katalog servisinde çeşitli

düzenlemeler yapmış ve belirtimin yeni sürümlerini yayınlamıştır. Yürürlükte olan son sürüm (CSW 2.0.1) (OGC, 2005b), Mayıs 2005 de yayınlanmıştır.

Bilgi modeli: CSW belirtimi, OGC uyumlu bir katalog servisinin desteklemesi gereken arayüzleri ve bu arayüzlerin farklı uygulama protokollerindeki gerçekleştirim esaslarını tanımlayan temel belirtimdir. CSW belirtiminde, bir OGC katalog servisinin gerçekleştirmesi gereken bilgi modeli tanımlanmamıştır. Bir CSW katalog servisinin bilgi modeli, OGC tarafından geliştirilen bir "katalog uygulama profili" tarafından tanımlanır. OGC, CSW katalog servisinin bilgi modelini tanımlamak için, 2004 yılına ebRIM profilini (OGC, 2004b), 2005 yılında ise ISO19115/19119 profilini (OGC, 2005a) geliştirmiştir. CSW'nin bilgi modelleri ile ilgili detaylar, uygulama profillerinin açıklandığı ileriki bölümlerde ele alınacaktır.

Katalog arayüzleri: CSW katalog servisleri, CSW belirtimi tarafından tanımlanan arayüzleri gerçekleştirirler. CSW belirtiminde, bir CSW katalog servisinin gerçekleştirmesi gereken 5 arayüz (OGC_Service, Discovery, Manager, BrokeredAccess, Session) tanımlanmıştır (Şekil 15). "OGC Servis" (OGC_Service) ve "Bulma" (Discovery) arayüzü, tüm CSW katalog servisleri tarafından gerçekleştirilmesi zorunlu olan arayüzlerdir. Bir CSW katalog servisinin gerçekleştirebileceği arayüzler, katalog servisinin kullandığı uygulama protokolüne bağlı olarak değişir. Örneğin, HTTP protokolü "durumsuz" (stateless) bir protokol olduğu için, bu protokolü kullanan bir CSW katalog servisi "Oturum" (Session) arayüzünü gerçekleştiremez. HTTP protokolünü kullanan bir CSW katalog servisi, OGC Servis ve Bulma arayüzlerinin yanı sıra, seçmeli olan "Yönetim" (Manager) arayüzünü de gerçekleştirebilir. CSW belirtimi, katalog servisi arayüzlerinin Z39.50, CORBA/IIOP ve HTTP protokollerindeki gerçekleştirimlerini tanımlamaktadır. Arayüzler ve operasyonlar, farklı protokolleri kullanan katalog servislerinde farklı isimlerle adlandırılırlar. Örneğin, HTTP protokolünü kullanan CSW gerçekleştirimlerinde, Yönetim arayüzü, "Yayınlama" (Publication) arayüzü olarak adlandırılmaktadır. Ayrıca, Bulma arayüzünün "query" ve "present" operasyonları birleştirilerek, "GetRecords" olarak adlandırılan yeni bir operasyon oluşturulmaktadır.

Bu bölümde sadece OGC Servis ve Yayınlama arayüzlerine değinilecektir. Bulma arayüzü, CSW katalog servisinin sorgulama modelinin anlatıldığı bir sonraki bölümde ele alınmaktadır. Diğer arayüzler (BrokeredAccess, Session) ve tüm arayüzlerin CORBA/IIOP ve Z39.50 protokollerindeki gerçekleştirimleri ile ilgili ayrıntılı bilgiler, CSW belirtiminde bulunabilir.



Şekil 15. OGC CSW katalog servisi arayüzleri (OGC, 2005b).

OGC servis arayüzü, OGC servis modeline göre tüm OGC Web servislerinin desteklemesi zorunlu olan "*GetCapabilities*" operasyonunu içermektedir. *GetCapabilities* operasyonu, kullanıcılara, bir CSW katalog servisinin sorgu yeteneklerini, servisin gerçekleştirebildiği operasyonları ve bu operasyonların parametrelerini gösteren ve "yetenekler" (capabilities) dosyası olarak adlandırılan bir XML dokümanı döndürür. Kullanıcı, bu XML dokümanını inceleyerek katalog servisinin neler yapabileceği (yetenekleri) hakkında bilgi sahibi olur.

Sağlayıcılar, sahip oldukları bilgi kaynaklarını tanımlayan meta verileri, yayınlama arayüzünü kullanarak CSW katalog servislerine yayınlarlar. Yayınlama arayüzü, seçmeli olan "*Transaction*" ve zorunlu olan "*Harvest*" operasyonlarına sahiptir.

Transaction operasyonu, meta verilerin bir CSW katalog servisine sağlayıcılar tarafından yayınlanmasına olanak sağlar. Ayrıca sağlayıcılar, *transaction* operasyonunu kullanarak kataloglardaki meta veri kayıtlarını güncelleyebilir veya silebilirler. Meta verisini yayınlamak isteyen bir sağlayıcı, *transaction* operasyonunu kullanarak katalog servisine XML formatında bir istek mesajı gönderir. İstek mesajı, sağlayıcı tarafından katalogun bilgi modeline uygun olarak yazılan meta verileri içerir. CSW katalog servisi, gelen istek mesajını okur ve meta verileri kataloga kaydeder.

Harvest operasyonu ise, meta verilerin bir CSW katalog servisi tarafından otomatik olarak bir sunucudan alınmasını ve kataloga kaydedilmesini sağlar. *Harvest* operasyonunu kullanarak meta verilerini bir CSW katalog servisine yayınlamak isteyen bir sağlayıcı, ilk olarak bir meta veri editörü kullanarak sahip olduğu bilgi kaynaklarını tanımlayan XML formatında meta veri dosyaları üretir. Sağlayıcı daha sonra katalog servisine bir istek mesajı (*HarvestRequest*) gönderir. İstek mesajı, meta veri dosyalarına erişim adresini, meta veri standardının adını ve sağlayıcı tarafından belirtilen bir zaman aralığını içerir. Katalog servisi, kullanıcı tarafından belirtilen adrese bağlanır ve meta veri dosyalarını alır, bir yazılım (XML parser) kullanarak meta verileri dosyalardan okur ve kataloga kaydeder. Katalog servisi, sağlayıcı tarafından belirtilen zaman aralığında, yeniden belirtilen adrese bağlanarak katalog kayıtlarını günceller.

CSW katalog servisleri, katalog içeriğindeki değişikleri kullanıcılara otomatik olarak bildiren bir "olay bildirme" (event notification) arayüzüne sahip değildir.

Katalog uygulama profilleri, CSW katalog servislerine, bilgi modeli desteğinin yanı sıra arayüz desteği de sağlayabilirler. Örneğin, CSW-ebRIM profilinin ilk sürümünde (OGC, 2004b), ebRIM bilgi modelini gerçekleştiren CSW katalog servislerinin "Bulma" arayüzü, "*WRS-Retrieval*" olarak adlandırılan seçmeli bir arayüz ile genişletilmişti. CSW-ebRIM profilinin ikinci sürümünde ise, "Bulma" arayüzüne "*GetRepositoryItem*" olarak adlandırılan zorunlu bir operasyon eklenmiştir. CSW-ebRIM profilinin ikinci sürümü, ilk sürümü geçersiz kıldığı için "*WRS-Retrieval*" arayüzü hakkında bilgi verilmeyecektir. Ancak "*GetRepositoryItem*" operasyonu, CSW-ebRIM profilinin açıklandığı ileriki bölümde ele alınacaktır.

Bulma arayüzü: Bir CSW katalog servisinde arama yapmak için CSW'nin Bulma arayüzü kullanılır. Bulma arayüzü, biri seçmeli olan dört adet operasyon (*DescribeRecord*, *GetRecords*, *GetRecordById*, *GetDomain*) içerir. Katalog servisini sorgulamak ve meta veri kayıtlarını elde etmek için kullanılan temel operasyon, "*GetRecords*" operasyonudur. *GetRecords* operasyonu, CSW belirtimi tarafından tanımlanan CQL (Common Catalogue Query Language) veya OGC Filter (OGC, 2005e) sözdizimine uygun olarak yazılan bir sorgu ifadesi içerir. Sorgu ifadesi, katalog servisinde bilgi kaynaklarını tanımlamak için kullanılan meta veri sınıflarının öznitelikleri kullanılarak yazılır. Bu nedenle, kullanıcının sorgusunu yazabilmesi için katalogun bilgi modelini bilmesi gerekmektedir. Kullanıcı, katalog servisinin bilgi modelini bilmiyorsa, Bulma arayüzünün "*DescribeRecord*" operasyonunu kullanarak, katalogdan bilgi modelinin şema tanımını almalıdır. *GetRecords*

operasyonunun, dördü zorunlu, toplam on yedi adet parametresi vardır. Kullanıcı, bu parametrelerin alabileceği değerleri öğrenmek için, arayüzün seçmeli olan "*GetDomain*" operasyonunu kullanır. Örneğin kullanıcı, *GetRecords* operasyonunun "*outputFormat*" parametresinin alabileceği değerleri öğrenmek için katalog servisine bir istek mesajı gönderir. Katalog servisi de kullanıcıya, bu parametrenin alabileceği değerleri (Text, XML, HTML) gösteren bir liste döndürür.

Dağıtık arama desteği: *GetRecords* operasyonu, "*DistributedSearch*" olarak adlandırılan seçmeli bir parametre içermektedir. *DistributedSearch* parametresi, kullanıcının bir CSW katalog servisine gönderdiği sorgunun, sorgunun gönderildiği katalog servisi eğer bir katalog federasyonuna üye ise, federasyon içerisindeki diğer katalog servislerine gönderilip gönderilmeyeceğini belirtmektedir. OGC, CSW belirtimi ve katalog uygulama profilleri ile "bir CSW katalog servisi hangi arayüzleri gerçekleştirmelidir?" ve "katalog servisi nasıl bir bilgi modeline sahip olmalıdır?" gibi sorulara yanıt vermektedir. OGC, katalog servisleri arasında federasyonun nasıl kurulacağı ve sorgu yönlendirmenin nasıl yapılacağı gibi konuları, uygulama profillerini gerçekleştiren katalog sağlayıcılarına bırakmıştır.

OGC, bir kullanıcının, bir sorgu üzerinde değişiklik yapmadan ve katalog servislerinin bilgi modellerinin detaylarını bilmeden, aynı sorguyu kullanarak tüm CSW katalog servislerinde arama yapabilmesini sağlamak için Tablo 3'te yer alan "Temel Sorgulanabilir Öznitelikleri" (TSÖ) geliştirmiştir (OGC, 2005b). TSÖ'leri kullanarak katalogda arama yapacak olan bir kullanıcının, "*DescribeRecord*" operasyonunu kullanarak katalogun bilgi modelini öğrenmesine gerek yoktur. Kullanıcının, TSÖ'leri kullanarak yazdığı sorguyu, *GetRecords* operasyonu ile bir CSW katalog servisine göndermesi yeterlidir.

TSÖ'ler, kullanıcıların, özellikle farklı bilgi modellerini gerçekleştiren CSW katalog servislerinin oluşturduğu bir katalog federasyonu içerisinde arama yapmaları durumunda önem kazanmaktadır. Bir kullanıcının, federasyon içerisindeki bir katalog servisinin bilgi modeline göre yazdığı sorgu, farklı bir bilgi modelini gerçekleştiren katalog servisi tarafından yürütülemez. Dolayısıyla, TSÖ'ler katalog servislerinin birlikte işlerliğini sağlarlar.

Tablo 3. Temel sorgulanabilir öznitelikler

Özniteliğin Adı	**Açıklaması**	**Veri Tipi**
Subject	Kaynağın içeriğinin konusu	String
Title	Kaynağın adı	String
Abstract	Kaynağın içeriğinin özeti	String
AnyText	Bir katalogda, karakter veri tipine sahip tüm alanlarda aranacak olan bir metin	String
Format	Kaynağın gösterim formatı	application/xml, text/html, text/plain
Identifier	Kaynağın katalogdaki ID si	Identifier
Modified	Kaynağın en son değişim tarihi	Date-ISO 8601
Type	Kaynağın türü	Dataset, Service
BoundingBox	İlgilenilen coğrafik alanı tanımlamak için kullanılan sınırlayan dikdörtgen koordinatları	numeric
CRS	Sınırlayan dikdörtgen koordinatlarının tanımlandığı koordinat sistemi	Identifier urn:opengis:crs:EPSG:4326
Association	Bire-bir ilişki	String ve Identifier

Anlamsal Destek: Bir katalog servisinin anlamsal desteği, sahip olduğu bilgi modelinin, bilgi kaynaklarını anlamsal olarak tanımlayabilme yeteneğine bağlıdır. Bu nedenle, CSW katalog servislerinin anlamsal desteği, CSW'nin bilgi modellerini tanımlayan uygulama profilleri açıklanırken ele alınacaktır.

2.3.4.1. OGC Katalog Servisleri - ebRIM Profili

CSW-ebRIM profili, OGC tarafından CSW katalog servislerine bilgi modeli desteği sağlamak için geliştirilmiştir. Profilin ilk sürümü (version 0.9.3) 2004 yılında, ikinci sürümü (version 1.0.0) ise 2006 yılında yayınlanmıştır. ebRIM bilgi modeli ve CSW katalog servisleri hakkında yukarıdaki bölümlerde detaylı bilgiler verilmişti. Dolayısıyla bu bölümde, CSW-ebRIM profilinin, özellikle ebRIM bilgi modeli ve CSW arayüzleri üzerinde yaptığı genişletmeler ele alınacaktır.

Bilgi modeli: CSW-ebRIM profiline uygun olarak geliştirilen bir CSW katalog servisi, ebRIM bilgi modelini gerçekleştirir. OGC, ebRIM bilgi modelinin konumsal bilgi toplumunun katalog servisi gereksinimlerini karşılayabilmesi için özelleştirilebileceği noktaları belirlemiş ve bu noktaları kullanarak bilgi modelinde bir takım genişletmeler yapmıştır (OGC, 2006). Söz konusu genişletmeler aşağıda açıklanmıştır.

- ebRIM bilgi modelindeki nesne tipleri konumsal bilgi ihtiyaçları doğrultusunda genişletilmiştir. ebRIM bilgi modelindeki bütün sicil (registry) nesneleri, "*objectType*" olarak adlandırılan bir özniteliğe sahiptir. *ObjectType* özniteliği, değerini ebRIM'in "ObjectType taksonomisi"nden (ObjectType ClassificationScheme) almaktadır. *ExtrinsicObject* (EO) ve *ExternalLink* nesneleri dışındaki tüm nesnelerin *objectType* özniteliklerinin değeri, nesne isimleri ile aynıdır. Örneğin, *Service* nesnesinin *objectType* özniteliğinin değeri yine "*Service*" dir. EO nesnesinin *objectType* özniteliğinin değeri ise, bu nesnenin tanımladığı içeriğin tipine bağlı olarak değişmektedir. Örneğin, EO nesnesi bir XML şema dokümanını tanımlıyorsa, *objectType* özniteliği "XML Schema" değerini alır. ebRIM'in ObjectType Taksonomisi, konumsal bilgi toplumundaki bilgi kaynaklarını tanımlayacak kadar geniş bir kategori içermemektedir. Örneğin, söz konusu taksonomide, ISO 19115 meta veri dosyalarını tanımlamak için kullanılacak bir obje tipi yoktur. Bu nedenle CSW-ebRIM profili, ObjectType Taksonomisine Tablo 4'de yer alan obje tiplerini eklemiştir.

Tablo 4. CSW-ebRIM profili tarafından tanımlanan yeni obje tipleri

Nesne tipi	**Açıklama**
ServiceProfile	Web servislerinin yeteneklerini tanımlayan dokümanları tanımlamak için kullanılan obje tipi (OWL-S servis ontolojisinden alınmıştır). Örnek: OGC yetenekler dokümanı
ServiceModel	Bir Web servisin nasıl çalıştığını tanımlayan bir dokümanı tanımlamak için kullanılan obje tipi (OWL-S servis ontolojisinden alınmıştır). Örnek: WSDL dokümanı
ServiceGrounding	Bir Web servisine nasıl ulaşılacağını tanımlayan bir dokümanı tanımlamak için kullanılan obje tipi (OWL-S servis ontolojisinden alınmıştır). Örnek: WSDL dokümanı
Dataset	ISO 19115 meta veri dosyası gibi bir konumsal veri setini tanımlayan dokümanları tanımlamak için kullanılan obje tipi.
Schema	UML, XMI, XML Schema, RELAX NG ve ASN.1 gibi şema dokümanlarını tanımlamak için kullanılan obje tipi.
Stylesheet	XSLT, CSS ve SLD gibi, bilgi kaynaklarını biçimlendirmek için kullanılan dokümanları tanımlamak için kullanılan obje tipi.
Document	Standartlar, belirtimler, kullanım kılavuzları gibi her türlü dokümanı tanımlamak için kullanılan obje tipi.
Annotation	Bir kaynağı ile ilgili açıklayıcı bilgiler içeren dokümanları tanımlamak için kullanılan obje tipi.
Image	Harita, fotoğraf veya animasyon gibi görsel kaynakları tanımlamak için kullanılan obje tipi.

- Konumsal bilgi kaynakları arasındaki ilişkileri temsil etmeye yönelik yeni ilişki tipleri eklenmiştir. Bunun nedeni, ebXML standardının konumsal olmayan, "e-iş sektörüne yönelik olarak tasarlanmış olmasıdır. Örneğin ebRIM de, bir konumsal veri seti ile bu veriyi sunan OGC Web servisi arasındaki ilişkiyi tanımlamak için kullanılabilecek bir ilişki tipi yoktur. Eklenen ilişki tipleri Tablo 5'de yer almaktadır.

Tablo 5. CSW-ebRIM profili tarafından tanımlanan yeni ilişki tipleri

İlişki tipi	**Açıklama**
OperatesOn	Bir Web servisi ile bu servis tarafından sunulan veri seti arasındaki ilişkiyi tanımlamak için kullanılır. ISO 19119 standardından alınmıştır.
Presents	Bir Web servisi ile servisin yetenekler dokümanı arasındaki ilişkiyi tanımlamak için kullanılır. OWL-S Servis ontolojisinden alınmıştır.
Supports	Bir Web servisi ile servise nasıl ulaşılacağını tanımlayan grounding dosyası arasındaki ilişkiyi tanımlamak için kullanılır. OWL-S Servis ontolojisinden alınmıştır.
DescribedBy	Bir Web servisi ile servisin nasıl çalıştığını tanımlayan işlem modeli dosyası arasındaki ilişkiyi tanımlamak için kullanılır. OWL-S Servis ontolojisinden alınmıştır.
Annotates	Bir registry nesnesini, bir açıklayıcı kaynak ile ilişkilendirir.
RepositoryItemFor	Bir ExternalLink nesnesi ile bir ExtrinsicObject arasında ilişki kurmak için kullanılır.
GraphicOverview	Bir sayısal konumsal veri setini, bu veri setini gösteren bir resim ile ilişkilendirmek için kullanılır.

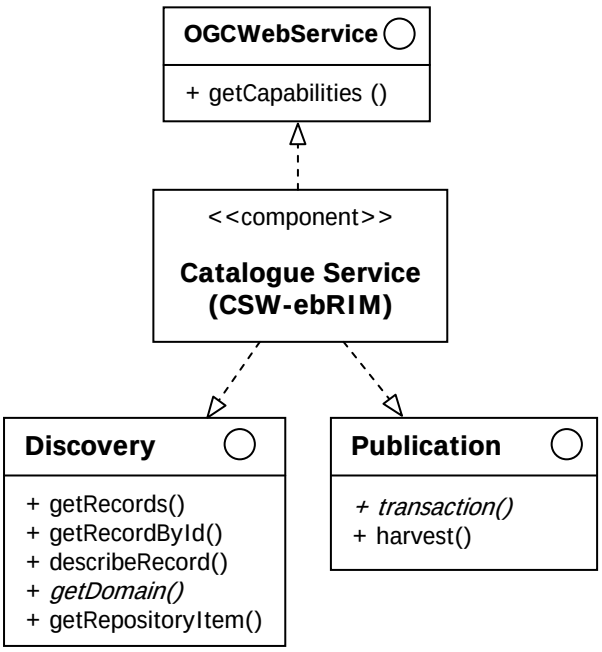
- ISO 19119 Coğrafi Servis Taksonomisi ve DIGEST FACC "Detay Kodları" taksonomisi gibi, konumsal servis ve verilere yönelik olan ve ebXML bilgi modelinin desteklemediği yeni taksonomilerin kullanılabilmesi için gerekli destek sağlanmıştır.
- Farklı bilgi kaynağı yayınlayıcılarının farklı *Slot* isimleri kullanmasının önlenmesi ve dolayısıyla birlikte işlerliğin sağlanabilmesi amacıyla standart *Slot* isimleri belirlenmiştir. Bu *Slot* isimleri için, Dublin Core meta veri terimleri esas alınmıştır (Tablo 6).

Tablo 6. CSW-ebRIM profili tarafından tanımlanan slot isimleri

Slot ismi	Açıklama
http://purl.org/dc/elements/1.1/contributor	Kaynağın içeriğine katkıda bulunan organizasyon
http://purl.org/dc/elements/1.1/coverage	Kaynağın içeriğinin kapsamı
http://purl.org/dc/elements/1.1/creator	Kaynağı yaratan organizasyon
http://purl.org/dc/elements/1.1/date	Kaynağın yaşam sürecindeki bir olayla ilişkili tarih
http://purl.org/dc/elements/1.1/description	Kaynağın içeriğinin tanımı
http://purl.org/dc/elements/1.1/format	Kaynağın formatı
http://purl.org/dc/elements/1.1/identifier	Kaynağın id si
http://purl.org/dc/elements/1.1/language	Kaynağın tanımlandığı dil
http://purl.org/dc/elements/1.1/publisher	Kaynağı sunan organizasyon
http://purl.org/dc/elements/1.1/relation	Kaynakla ilişkili bir diğer kaynağa erişmek için kullanılan referans
http://purl.org/dc/elements/1.1/rights	Kaynakla ilgili sağlayıcı ve istemci hakları.
http://purl.org/dc/elements/1.1/source	Kaynağın türetildiği kaynak
http://purl.org/dc/elements/1.1/subject	Kaynağın içeriğinin konusu
http://purl.org/dc/elements/1.1/title	Kaynağa verilen isim
http://purl.org/dc/elements/1.1/type	Kaynağın içeriğinin türü
http://purl.org/dc/terms/abstract	Kaynağın içeriğinin özeti
http://purl.org/dc/terms/modified	Kaynağın en son değiştirildiği tarih
http://purl.org/dc/terms/spatial	Kaynağın konumsal özellikleri
http://purl.org/dc/terms/temporal	Kaynağın zamansal özellikleri
http://purl.org/dc/terms/valid	Kaynağın geçerlilik tarihi

Katalog arayüzleri: CSW-ebRIM profiline uygun olarak geliştirilen bir katalog servisi, CSW belirtimi tarafından tanımlanan arayüzleri gerçekleştirmek zorundadır. CSW-ebRIM profili, katalog servislerinin HTTP protokolünü kullanmalarını istemektedir. Bu nedenle katalog servisleri, CSW'nin "OGC Service", "Yayınlama" ve "Bulma" arayüzlerini gerçekleştirebilirler. Arayüzlerin içerdikleri operasyonlar, CSW belirtiminde anlatıldığı şekilde işlemektedir. Bununla birlikte, ebRIM profili CSW'nin standart "Bulma" arayüzünü, gerçekleştirilmesi zorunlu olan "*GetRepositoryItem*" operasyonu ile genişletmiştir (Şekil 16). İstemcilerin, bir katalog servisinin depo (repository) bileşeninde yer alan dokümanlara ulaşmalarını sağlayan *GetRepositoryItem* operasyonu, OASIS ebRS standardı tarafından tanımlanan bir operasyondur. ebXML katalog servislerinde, katalogun

depo bileşeninde yer alan bir dokümanın, sicil tarafında EO nesnesi kullanılarak tanımlandığını ve EO nesnesi ile dokümanın aynı id'ye sahip olduğunu hatırlayalım. *GetRepositoryItem* operasyonu, parametre olarak EO nesnesinin id'sini alır ve geriye EO nesnesi tarafından tanımlanan dokümanı döndürür.



Şekil 16. CSW-ebRIM profilini gerçekleştiren katalog servislerinin desteklemesi gereken arayüzler (OGC, 2006).

Bulma arayüzü: CSW-ebRIM profilini gerçekleştiren bir katalog servisi, CSW belirtimi tarafından tanımlanan bulma arayüzünü destekler. Katalog servisi, Bulma arayüzünün *GetRecords* operasyonu kullanılarak sorgulanır. *GetRecords* operasyonu, ebRIM bilgi modelindeki meta veri sınıflarının öznitelikleri kullanılarak yazılan bir sorgu ifadesi içerir. CSW-ebRIM profili, sorgu ifadelerinin CQL ve OGC Filter sözdizimlerine ilave olarak XPath ve XQuery sorgu dilleri kullanılarak ta yazılmasına olanak sağlamaktadır.

CSW-ebRIM profili, kullanıcıların katalog servislerinde içerik tabanlı sorgu yapmasına olanak sağlayan bir yöntem tanımlamaktadır. "Derin arama" (deep search) adı verilen bu yöntemde, kullanıcıların XPath sorgu dilini kullanarak katalog servisinde içerik tabanlı sorgular gerçekleştirebileceği belirtilmektedir. Kullanıcılar, katalog servisine gönderdikleri XPath sorgusu ile sadece "*isOpaque*" özniteliğinin değeri "*false*", "*mimeType*" özniteliğinin değeri "XML" olan EO nesneleri tarafından tanımlanan XML

formatındaki dokümanları sorgulayabilirler. Derin arama yöntemini gerçekleştiren katalog servisleri, bu özelliklerini, kullanıcılara gönderdikleri yetenekler dosyasında belirtirler.

CSW-ebRIM profili, bu profili gerçekleştirecek olan bir katalog servisinin gerçekleştirmesi gereken saklanmış sorguları da tanımlamaktadır. Profil tarafından tanımlanan saklanmış sorgular, bir katalog servisinde kullanıcıların yaygın olarak kullanacağı düşünülen sorgulardan oluşmaktadır. ISO 19119 Coğrafi Servis taksonomisindeki kategorilere uyan Web servislerini bulmak için tasarlanan "*findService*" sorgusu, profil tarafından tanımlanan saklanmış sorgulara örnek olarak gösterilebilir. Diğer sorgularla ilgili bilgiler, CSW-ebRIM profilinde bulunabilir.

Dağıtık arama desteği: CSW belirtiminde, katalog servislerinin dağıtık sorgulamayı (DistributedSearch) destekledikleri belirtilmişti. Ancak ebRIM profili, dağıtık sorgulamanın nasıl yapılacağını tanımlayan bir öneri içermemekte ve bu özelliğin gerçekleştirimini katalog sağlayıcılarına bırakmaktadır.

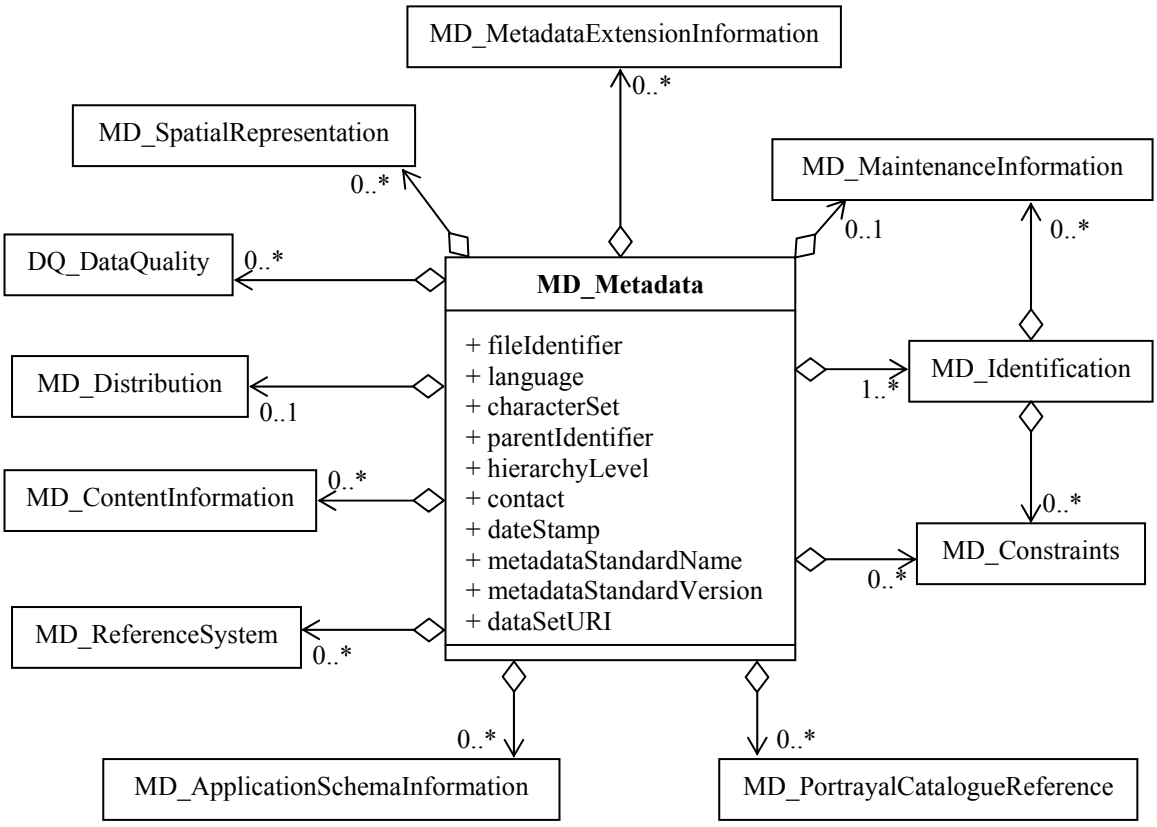
Anlamsal destek: ebRIM bilgi modelinin anlamsal desteği, ebXML Katalog Servislerinin anlamsal desteğinin değerlendirildiği yukarıdaki bölümde açıklanmıştı. Ancak CSW-ebRIM profili, ebRIM bilgi modeli için ilave bir anlamsal genişletme içermemektedir. Ayrıca CSW-ebRIM profilinde, bu profili gerçekleştiren CSW katalog servislerinin, bilgi modelinin sağladığı mevcut anlamsal desteği kullanabilecekleri yönünde bir açıklama yer almamaktadır.

2.3.4.2. OGC Katalog Servisleri – ISO 19115/19119 Profili

OGC, CSW belirtimi ile bir katalog servisinin gerçekleştirmesi gereken arayüzleri ve operasyonları tanımlamıştır. OGC, bugüne kadar yayınladığı CSW belirtimlerinde, katalog servisleri için bir bilgi modeli tanımlamamış ve katalog sağlayıcılarının da belirli bir bilgi modelini kullanmalarını şart koşmamıştır (OGC,2005b). Ancak "OWS-1" girişiminde (OGC, 2002), katalog bilgi modeli olarak ebRIM standardının kabul edildiği bildirilmesine ve 2004 yılında CSW-ebRIM profilinin yayınlanmasına rağmen, OGC, 2005 yılında katalog servisleri için yeni bir bilgi modeli tanımlayan ISO 19115/19119 profilini yayınlamıştır. CSW-ISO 19115/19119 profili, Almanya'nın North Rhine Westphalia (NRW) eyaletinin UKVA çalışmalarında, katalog servisi gerçekleştirimlerinde, bilgi modeli olarak ISO 19115 ve ISO 19119 meta veri standartlarından geliştirilen bir uygulama profilinin kullanılmasından esinlenerek geliştirilmiştir.

Bilgi modeli: CSW-ISO 19115/19119 profiline uygun olarak geliştirilen CSW katalog servisleri, bilgi modeli olarak ISO 19115 (ISO, 2003) ve ISO 19119 (ISO, 2001) meta veri standartlarını gerçekleştirirler. ISO 19115 konumsal veriler, ISO 19119 ise Web servisleri ile ilgili meta verileri tanımlamak için geliştirilen standartlardır.

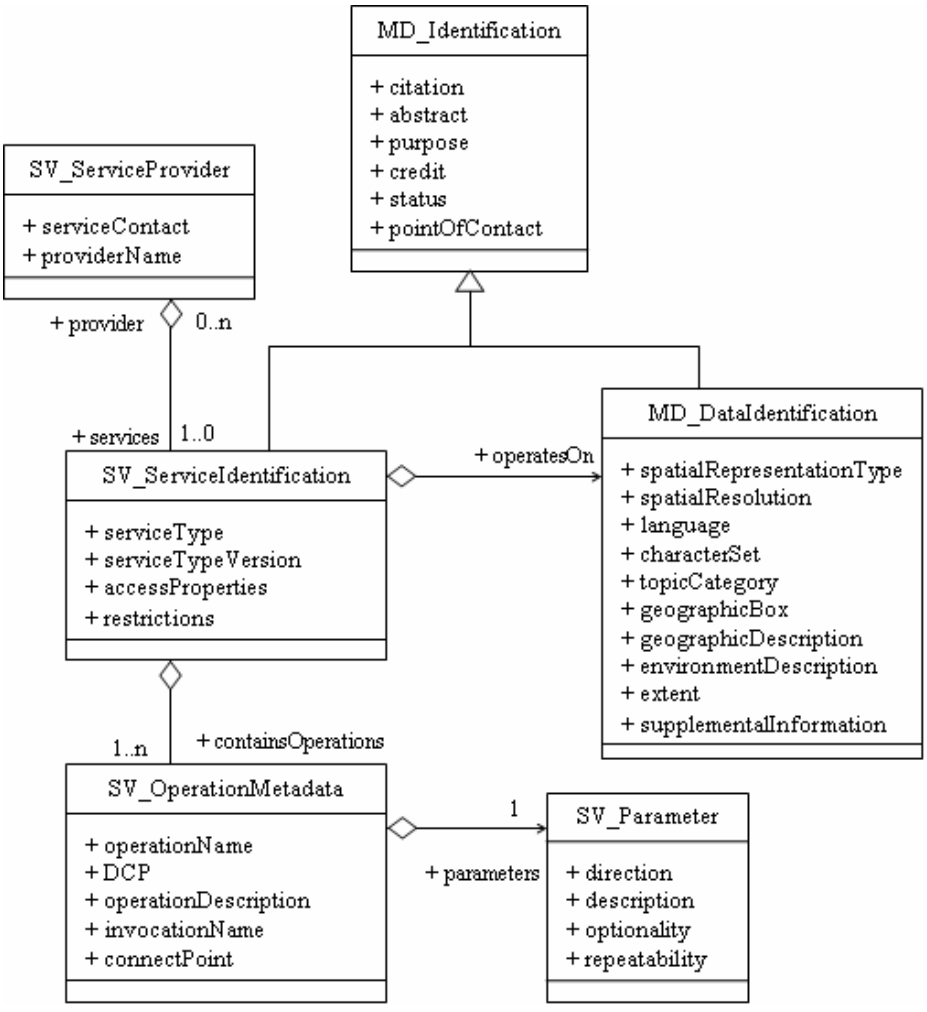
Bilgi kaynağı türü: CSW-ISO 19115/19119 profili, ebRIM bilgi modelinde olduğu gibi her türlü elektronik dokümanı tanımlamaya olanak sağlayan genel bir bilgi modeline sahip değildir. Profil, sadece konumsal verilerin ve Web servislerinin tanımlanmasına olanak sağlayan bir bilgi modeli sunmaktadır (Şekil 17). Örneğin bilgi modeli, bir SLD dosyasını veya GML formatından SVG formatına dönüşüm yapmak için kullanılan bir XSLT dosyasını tanımlamak için kullanılamaz.



Şekil 17. ISO 19115 de konumsal verileri tanımlamak için kullanılan meta veri sınıfları (ISO, 2003a).

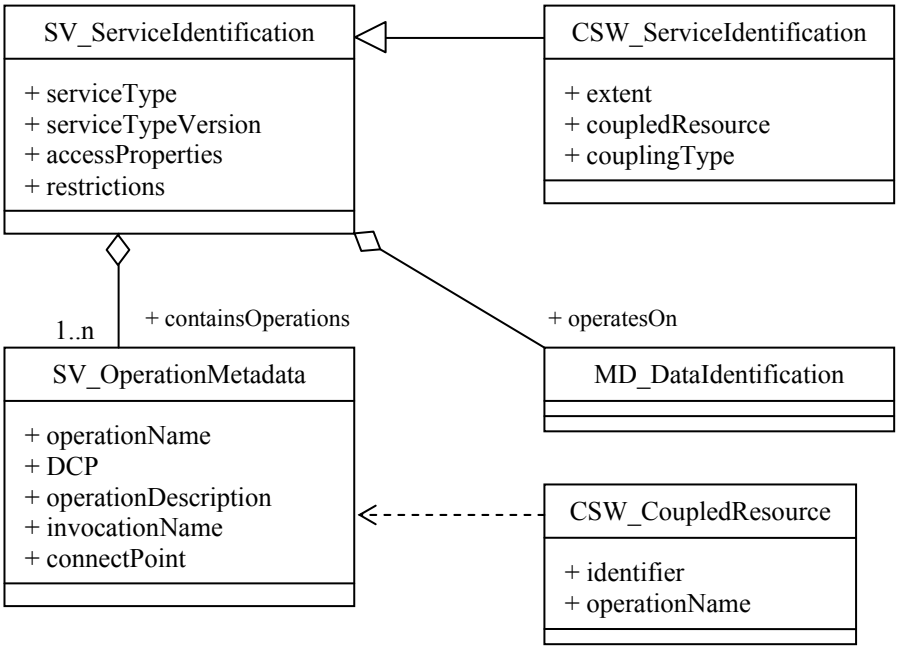
Anlatım gücü: ISO 19115 meta veri standardı, konumsal verilerle ilgili meta verileri tanımlayan 12 adet temel meta veri sınıfı (Şekil 17) ve yaklaşık 300 adet öznitelik içerir. ISO 19119 ise, Web servislerini tanımlayan 4 adet meta veri sınıfı içermektedir (Şekil 18).

ISO 19115 ve 19119, nesne-yönelimli bir tasarıma sahip değillerdir. Ancak meta veri sınıfları arasındaki ilişkileri tanımlamak için, genelleştirme (generalization) ve bir araya getirme (aggregation) gibi nesne yönelimli modellemede kullanılan kavramları kullanırlar. Örneğin, ISO 19115 meta veri standardında yer alan ve konumsal verilerle ilgili meta verileri tanımlamak için kullanılan ana sınıf olan "*MD_Metadata*" sınıfı, 11 adet meta veri sınıfını içerir (Şekil 18). Bu sınıflardan sadece "*MD_Identification*" sınıfı meta veri tanımı içerisinde yer alması zorunlu olan sınıftır. Diğer sınıflar ise seçmelidir. Ayrıca *MD_Identification* sınıfı, "*MD_DataIdentification*" ve "*SV_ServiceIdentification*" sınıflarının her ikisinin de üst sınıfı yani daha genel halidir (Şekil 18). Her iki meta veri standardında da, ebRIM bilgi modelinde olduğu gibi meta veri sınıfları arasındaki ilişkileri tanımlamak için kullanılan Association sınıfı gibi bir sınıf yoktur.



Şekil 18. ISO 19119 da servis meta verisini tanımlamak için kullanılan meta veri sınıfları (ISO, 2003b).

Genişletilebilirlik: ISO 19115 ve 19119 meta veri standartları, genişletilebilir bir bilgi modeli sunmaktadır. Her iki standart da, kullanıcıların bilgi modeline yeni meta veri sınıfları eklemesine veya bir meta veri özniteliğinin alabileceği değerlerin genişletilmesine olanak sağlamaktadır. CSW-ISO19115/19119 profili, ISO 19115 standardında yer alan meta veri sınıflarında bir genişletme yapmamıştır. Ancak profil, ISO 19119 bilgi modelinde bir takım genişletmeler yapmış ve bilgi modeline iki yeni sınıf eklemiştir (Şekil 19).

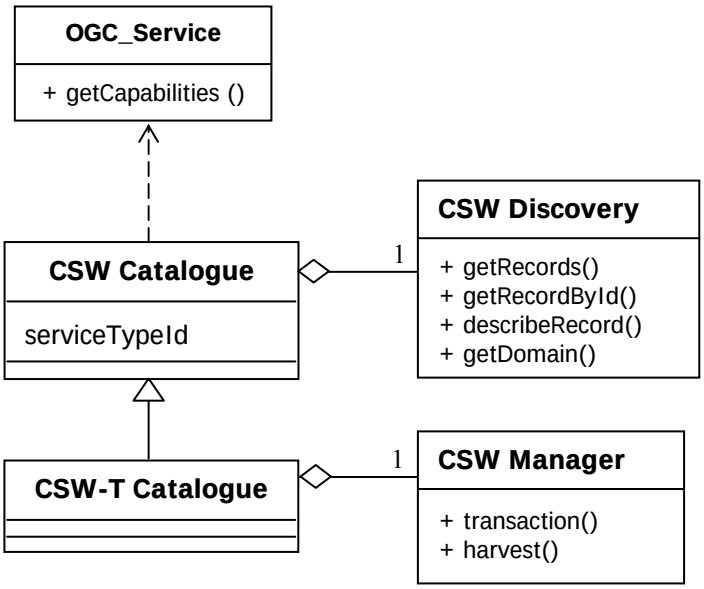


Şekil 19. ISO 19119 için CSW genişletmeleri (OGC, 2005b).

CSW-ISO 19115/19119 profili, Web servislerini, "gevşek bağlı" (loosely coupled), "sıkı bağlı" (tightly coupled) ve "karışık bağlı" (mixed coupled) Web servisleri olarak üçe ayırmaktadır. Söz konusu bağlılık düzeyleri, bir Web servisi ile bir konumsal veri seti arasındaki ilişkiyi temsil etmektedir. Sıkı bağlı Web servislerinde, bir Web servisi belirli bir veri seti ile ilişkilidir. Örneğin, bir OGC WFS servisi sadece kadastro verisini sunuyor ise bu sıkı bağlı bir Web servisidir. Bu tür Web servislerinde servis meta verisi, hem Web servisi hem de veri seti ile ilgili meta veri tanımlarını içerir. Gevşek bağlı Web servislerinde ise bir Web servisi belirli bir veri seti ile ilişkili değildir. Ancak servis, belirli bir veri tipi ile ilişkili olabilir. Örneğin bir Web servisi, sadece nokta tipine sahip olan verileri sunuyor ise söz konusu servis, gevşek bağlı bir servistir. Gevşek bağlı Web

servislerinde servis meta verisi, bir veri seti ile ilgili meta veri tanımlarını içermez. Karışık bağlı Web servislerinde ise bir Web servisi hem belirli bir veri seti ile hem de belirli bir veri tipi ile ilişkili olabilir. CSW-ISO 19115/19119 profili, Web servisleri ile konumsal veri setleri arasındaki söz konusu bağlılık ilişkilerini temsil edebilmek için bilgi modeline, "*CSW_ServiceIdentification*" ve "*CSW_CoupledResource*" sınıflarını eklemiştir.

Katalog arayüzleri: CSW-ISO 19115/19119 profiline uygun olarak geliştirilen CSW katalog servisleri, CSW belirtimi tarafından tanımlanan OGC Service, yayınlama ve bulma arayüzlerini gerçekleştirebilirler. CSW-ISO 19115/19119 profili, katalog servislerini gerçekleştirdikleri arayüzlere göre "salt okunur" (read-only) ve "işlevsel" (transactional) katalog servisleri olarak ikiye ayırmaktadır (Şekil 20). Salt okunur (CSW) katalog servisleri, sadece OGC servis ve bulma arayüzünü gerçekleştirirler. İşlevsel katalog servisleri (CSW-T) ise, bu arayüzlere ilave olarak yayınlama arayüzünü de gerçekleştirirler. OGC servis ve yayınlama arayüzlerinde yer alan operasyonlar, yukarıda CSW'nin katalog arayüz desteğinin anlatıldığı bölümde açıklanan şekilde yürütülürler. CSW-ISO 19115/19119 profili, CSW arayüzlerinde bir genişletme yapmamıştır. Ayrıca profilin, "olay bildirme" arayüzü desteği de yoktur.



Şekil 20. Salt-okunur ve işlevsel katalog servisleri (OGC, 2005b).

Bulma arayüzü: CSW-ISO 19115/19119 profilini gerçekleştiren bir katalog servisi, CSW belirtimi tarafından tanımlanan bulma arayüzünü gerçekleştirir. Ancak bu profili gerçekleştiren katalog servislerinde, kullanıcılar sorgularını ISO 19115 ve ISO 19119 meta veri standartlarında yer alan meta veri sınıflarının özniteliklerini kullanarak yazarlar. Ayrıca ISO profilini gerçekleştiren katalog servislerinde, *GetRecords* operasyonun *outputFormat* parametresi, sadece "XML" değerini alabilir.

Katalog uygulama profili: CSW-ISO 19115/19119 profili, katalog servisi sağlayıcılarına özelleştirilebilir bir bilgi modeli sunmaktadır. ISO 19115 meta veri standardı, 12 adet temel meta veri sınıfı ve yaklaşık 300 adet öznitelik içermektedir. Bu sınıfların ve özniteliklerin büyük bir çoğunluğu seçmelidir. Bir katalog servisi sağlayıcısının, bu meta veri sınıflarının veya özniteliklerin tamamına ihtiyacı olmayabilir. Örneğin, sadece vektör formatta konumsal veri sunan bir organizasyona ait katalog servisinde, "*MD_SpatialRepresentation*" sınıfının bir alt sınıfı olan "*MD_GridSpatialRepresentation*" sınıfının bilgi modelinde yer almasına gerek yoktur. Bu nedenle sağlayıcılar, bilgi modelini kendi ihtiyaçları doğrultusunda özelleştirebilirler. Ancak bilgi modelinin bu şekilde her kurum ya da her uygulama için özelleştirilmesi, katalog servisleri arasında birlikte işlerlik problemlerine sebep olabilir.

Anlamsal destek: CSW-ISO 19115/19119 profili tarafından tanımlanan bilgi modeli, anlamsal bir destek içermemektedir. Bu konuyla ilgili olarak gerek OGC de gerekse akademik çevrelerde bir çalışmaya rastlanamamıştır. Ayrıca bilgi modelini oluşturan ISO standartlarında da bu konuyla ilgili bilgiler yer almamaktadır.

2.4. Servis Düzenleme

"Servis düzenleme" (SD) bağlamında, İngilizce literatürde "*servis composition*", "*servis chaining*", "*service orchestration*" ve "*service choreography*" terimleri ile karşılaşılmaktadır. Servis düzenlemeden daha genel bir anlamı ifade etmek üzere ise "*collaboration*", "*coordination*", "*conversations*" kavramlarına atıfta bulunulmaktadır. Konunun geleneksel anlamdaki eşdeğerleri olarak ta "iş işlem yönetimi" (Business Process Management/BPM) ve "iş-akışı" (workflow) gibi terimler kullanılmaktadır. Şurası kesindir ki kullanılan isim ne olursa olsun amaç, Web servislerinin ya da işlemlerin bir işbirliğine olanak sağlayacak tarzda birbirleri ile ilişkilendirilmesidir (Peltz, 2003). SYM'nin başarısı

ve endüstri tarafından tercih edilmesi, servis düzenlemedeki başarıya bağlı olduğu için, SYM açısından en kritik konulardan biridir.

Bir dizi Web servisinin belirli bir düzende "işbirliği" yapmaları SYM de uygulama geliştirmenin yoludur. Basit bir örnekle açıklamak gerekirse, SYM ortamında bir "yer seçimi" (site selection) uygulamasını geliştirmek isteyen bir kullanıcı, bu işi çeşitli servis sağlayıcılarından sağlayabileceği Web servisleri ile gerçekleştirebilir. Amaca uygun "bileşen" servislerden bazıları örneğin ilgili coğrafi alana ait eğim, yol ve yükseklik bilgilerini sunan servisler olabilir. Bu türden fakat farklı yeteneklere sahip servisler farklı sunucular tarafından sunuluyor olabilir. O halde bir dizi servis arasında bir işbirliğinin tanımlanabilmesi için öncellikle istenen özellikleri sağlayan servislerin bulunması, daha sonra bu servisleri belirli bir koşum sırasında düzenleyen yeni bir servisin oluşturulması gerekmektedir.

Yeni düzenlenen serviste bir Web servisi gibi yayınlanabilir ve başka bir servis düzenlemede bileşen servis olarak kullanılabilir Daha önceden oluşturulmuş servis düzenlemenin yeni düzenlemelerde kullanılabilmesine, literatürde "özyineli düzenleme" (recursive composition) (Peltz, 2003) denir. Aslında özyineli düzenleme, ilkesel olarak anlamlı durmakla birlikte, uygulanması çok kolay olmayan bir yöntemdir. Çünkü düzenlemenin içerdiği servis örneklerinin, somut servis düzenleme anında bulunamaması durumunda sorun yaşanacaktır. Bu anlamda, özyineli düzenleme SYM felsefesi ile çok da uyumlu değildir. Çünkü SYM felsefesi ile öngörülmüş olan bize göre, yeni servisler ihtiyaç duyulduklarında mevcut servislerden tanımlanır ya da düzenlenir. Çünkü hazır bir düzenlemenin aranan özelliklere uygunluğunun belirlenmesi bir yana, bu düzenlemeyi gerçekleştirecek servis örneklerinin koşum anında bulunması ya da bulunamaması durumunda gerekli önlemin alınması, düzenlemenin soyut ve somut olarak yapılmasından daha zor olabilir. Ancak mevcut düzenlemelerin düzenli olarak güncellenmesi bu sorunu hafifletebilir.

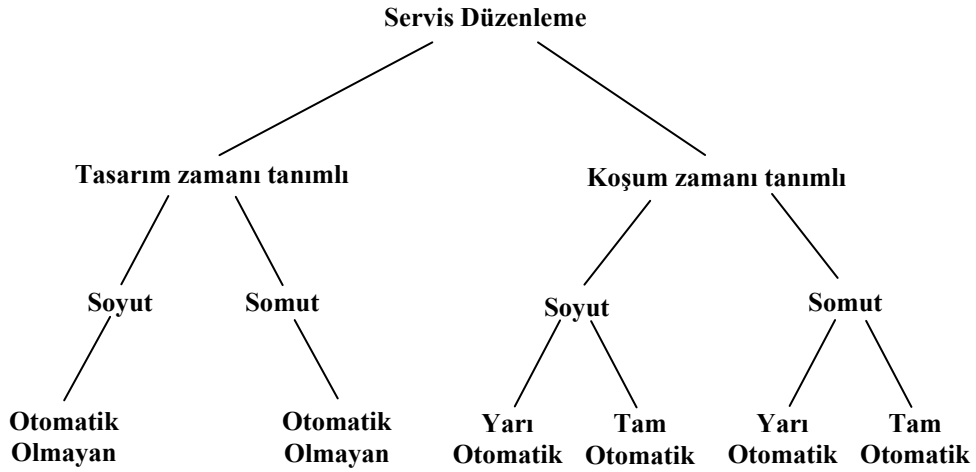
Servis düzenleme ilgili bir diğer konu, İş akışı Yönetimi (İaY) (Workflow Management/WfM) dir. Bu konuda çeşitli çalışmalar (Alameh, 2003) bulunmakla birlikte, konuyu temel ya da kritik boyutları ile ortaya koyabilen kaynak sayısı çok azdır. Örneğin Cardoso vd. (2003), İaY ve "Kuruluş Kaynak Planlama" (Enterprise Resource Planning/ERP) sistemleri arasındaki farkı vurgulamış ve bu iki kavramın birbirine karıştırıldığını belirtmiştir. Ancak Cardoso vd. (2003), aradaki farkı net bir biçimde, daha üst düzey bir bakış açısı altında verememiştir.

Diğer yandan, servis koreografisi ve servis orkestrasyonu aynı şeyler değildir. Fakat fark çoğunlukla yeterince açık bir şekilde vurgulanmamakta, hatta bazen bunlar aynı kavramlarmış gibi gösterilmektedir. Bu da bazen yanlış değerlendirmelere yol açmaktadır. Hendler vd. (2002) bu karmaşaya işaret etmiştir. W3C (2004d) de bile bu ayırım yeterince iyi bir şekilde yapılamamıştır. Öyle ki W3C (2004d) servis koreogafiyi, "…çok sayıda, bağımsız ve işbirliği yapan ajanın, bir amaç durumunu başarmaya yönelik olarak bir görev icra etmelerinde, aralarındaki mesaj değişimini düzenleyen iş akışı ve koşulların tanımı..." olarak tanımlamaktadır. Aynı dokümanda bu tanımın, W3C WS Koreografi çalışma grubununkinden "farklı bir soyutlama" olduğu da belirtilmektedir. Diğer yandan, yine aynı doküman orkestrasyonu, "…yararlı bir fonksiyon icra etmek için bir Web servisinin diğer bir Web servisini çağırmasındaki iş akışı ve koşulların tanımı..." olarak tanımlamaktadır. W3C WS Koreografi çalışma grubu ise koreografinin temel kullanım amacını, "...birbirleriyle işbirliği yapan bir dizi Web servisi arasındaki etkileşimin akışının, kesin bir biçimde tanımlanması..." olarak belirtmektedir (W3C, 2004d). Böyle bir tanımlama için de WS-CDL (Web Services Choreography Description Language) koreografi dilini önermektedir. WS-CDL (W3C, 2004f) dili aslında Oracle tarafından geliştirilmiş ve Eylül 2003 te, W3C koreografi çalışma kurumuna teslim edilmiştir. Bu grubun çalışmaları sonucunda ilk "taslak" Nisan 2004 de tamamlanmıştır.

Literatürde ve ilgili diğer çalışmalarda SD yöntemlerinin sınıflandırılması konusunda bir görüş birliği yoktur. Bu tez çalışması kapsamında böyle bir sınıflandırma önerilmiştir. Literatürde, SD yöntemlerini sınıflandıran sadece iki çalışma vardır. Konumsal Web servisleri alanındaki standartları belirleyen ISO/TC211 Teknik Komitesi tarafından geliştirilen "ISO 19119" standardı (ISO, 2001) tarafından tanımlanan SD yöntemleri, (Bernard vd., 2003) ve (Alameh, 2003) gibi sadece birkaç çalışmada kullanılmıştır. ISO 19119 da kanaatimizce tanımlama ve icra kavramları karıştırılmıştır. Arpınar vd. (2005) tarafından önerilen sınıflandırma ise net bir şekilde tanımlanamamıştır.

Bir SD uygulaması, genel olarak üç temel işlemden oluşur. Bu işlemler sırasıyla, katalog servislerinde arama yapılarak uygulama gereksinimlerini karşılayacak olan Web servislerinin bulunması, bulunan Web servislerinin girdi ve çıktı parametrelerini karşılaştırarak düzenlemede hangi servislerin hangi sırada kullanılacağının belirlenmesi ve son olarak servislerin belirlenen sırada yürütülmesidir. SD yöntemleri, bu işlemlerin "insan" veya "yazılım" tarafından yapılmasına bağlı olarak sınıflandırılmalıdır. Servis düzenleme, tasarım veya koşum zamanında, soyut ve somut düzeyde tanımlanabilir. Soyut

ve somut servis düzenleme, düzenlemenin insan ya da yazılım tarafından tanımlanmasına bağlı olarak, "otomatik olmayan", "yarı otomatik" ve "tam otomatik" olarak adlandırılan üç kategoriye ayrılabilir (Şekil 21).



Şekil 21. Servis düzenleme yöntemleri

SYM de servis düzenleme, soyut ve somut düzeyde gerçekleşebilir. Somut düzenleme hangi servis örneklerinin hangi koşum düzeninde koşacağının ve birlikte işlerliklerinin nasıl sağlanacağının tanımlandığı düzenlemedir. Birlikte işlerlik sağlamadan kasıt örneğin servis parametrelerinin tip olarak uyumunun eşleştirme sırasında belirlenmesi dışında örneğin parametre sayısındaki farklılığın giderilmesi olabilir. Servis düzenleme belirli bir dilde ifade edilir. Bunun için en yaygın kullanılan dil ise BPEL4WS (OASIS, 2006) dır. Soyut düzenleme kavramı ise bütün servis örneklerinin belirtilmediği, koşum zamanında ortaya çıkabilecek beklenmedik durumlar nedeniyle bir kısım bileşen servislere ait tanımların soyut bırakıldığı tanımlama şeklidir. Örneğin bir seyahat planlama servisi düzenlemesinde düzenlemeyi tanımlayan doküman içerisinde "...ulaşım servisi rezervasyonu yapacak olan bileşen servisin hava, kara ya da deniz ulaşım rezervasyonu servislerinden herhangi birinin olabileceği…" nin belirtilmesi, soyut servis düzenlemeye örnektir. Koşum zamanında ilgili sağlayıcılardan hangi servisler sağlanabilirse onlar kullanılacaktır. Buradaki ima, anlaşılabileceği üzere, soyut servis tanımının koşum anında somut servis tanımına dönüştürüldüğüdür. Eğer servis düzenleme tasarım anında tümüyle somut yapılsaydı ve koşum anında örneğin bu servislerden biri sağlayıcı tarafından ilgili katalogda artık yayınlanmıyor olsa idi bu servis örneği bulunamayacak ve uygulama

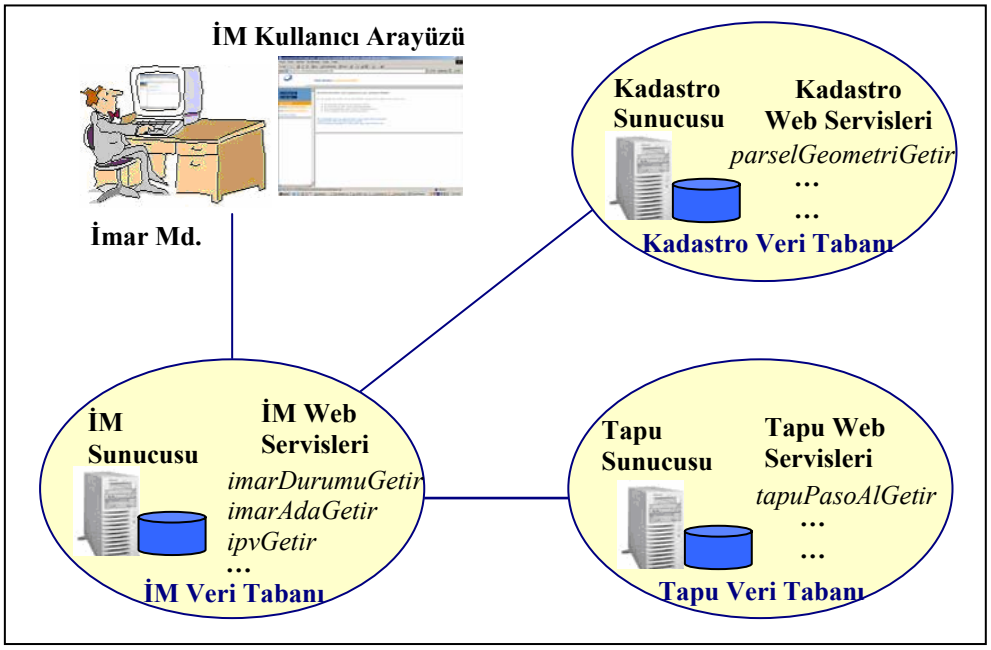
gerçekleştirilemeyecek idi. Soyut servis düzenleme kavramı ile bu gibi sorunların çözülmesi hedeflenmiştir.

Tasarım zamanı somut düzenleme BPEL işlem dokümanı ile tanımlanmış olan bir düzenlemedir. Böyle bir düzenleme için servislerin ve sağlayıcılarının bilinmesi gerekir. Bu da ancak otomatik olmayan bir yolla insan kullanıcı tarafından yapılabilir. Koşum zamanında somut servis oluşturma ise "yarı otomatik" ve "tam otomatik" olarak gerçekleştirilebilir. Yarı otomatik yöntemde yazılım, servis örneklerini bulur, kullanıcı örnekleri işlevsel ve işlevsel olmayan özellikleri bazında teke indirir. Tam otomatik yöntemde ise alternatif servis örnekleri bulma ve bunları teke indirme yazılım tarafından gerçekleştirilir. İlerleyen bölümlerde, somut servis düzenlemenin tasarım ve koşum zamanında hangi teknolojiler kullanılarak nasıl tanımlandığı ve otomatik olmayan, yarı otomatik ve tam otomatik olarak nasıl gerçekleştirildiği ele alınacaktır.

2.4.1. Otomatik Olmayan Yöntem

Otomatik olmayan yöntem, servis düzenlemenin insan tarafından gerçekleştirildiği yöntemdir. Bu yöntemde, mevcut Web servislerini kullanarak yeni bir uygulama geliştirmek isteyen bir kullanıcının, geliştireceği uygulama için ne tür Web servislerine ihtiyacı olduğunu ve Web servislerini nasıl düzenleyeceğini bilmesi gerekmektedir. Bu yöntemde kullanıcı, katalog servislerinde arama yaparak uygulama gereksinimlerini karşılayan Web servislerini bulmalıdır. Kullanıcı ayrıca, Web servislerinin girdi ve çıktı parametrelerini karşılaştırarak, eşleşen Web servislerini belirlemelidir. Bu yöntemde servis eşleştirme, çoğunlukla servis bulma işlemi gerçekleştirilirken yerine getirilmektedir. Çünkü bir kullanıcı bir katalog servisinde arama yaparken, uygulamada kullanacağı bir Web servisine ya veri sağlayan ya da bu servisin ürettiği verileri kullanan Web servislerini aramaktadır. Kullanıcı, uygulama gereksinimlerini karşılayan Web servislerini bulduktan sonra, bir Web servisleri geliştirme aracı kullanarak, uygulamayı gerçekleştiren yeni bir Web servisi yaratır. Yaratılan bu yeni Web servisi, "birleşik servis" (aggregate/composite service) olarak adlandırılır (ISO, 2001). Birleşik servis, uygulamayı gerçekleştirmek için, kullanıcı tarafından kataloglarda arama yapılarak bulunan "bileşen" servisleri kullanır. Birleşik servisin, bileşen servisleri hangi sırada çağıracağına, birleşik servisin gerçekleştirim kodunu yazan kullanıcı karar verir. Birleşik servisi kullanan bir istemci, bu servisin başka Web servislerini kullanarak uygulama gerçekleştirdiğini bilmez.

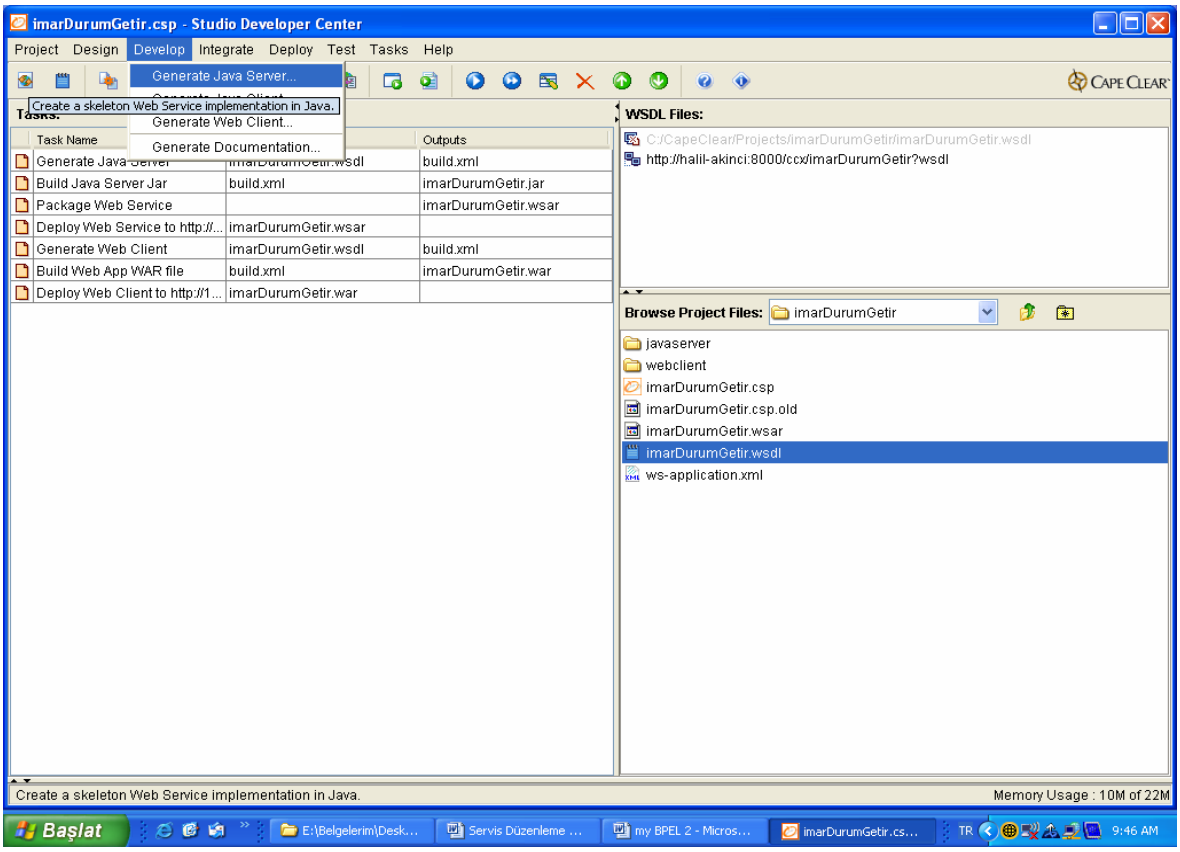
Bu tez çalışması kapsamında geliştirilen ve internet üzerinden otomatik imar çapı üreten "*imarDurumuGetir*" Web servisi, otomatik olmayan somut servis düzenleme yöntemine uyan bir birleşik Web servisidir. *İmarDurumuGetir* Web servisi, bir kadastro parselinin imar çapını üretmek için, kadastro müdürlüğü, tapu sicil müdürlüğü ve belediye imar müdürlüğü tarafından sunulan Web servislerini kullanmaktadır (Şekil 22).



Şekil 22. İmarDurumuGetir web servisinin kullandığı servisler ve sunucuları

İmarDurumuGetir Web servisini geliştirmek için kullanılan ve söz konusu servisin ihtiyaç duyduğu verileri sağlayan tüm Web servisleri, tarafımızdan geliştirilmiştir. Dolayısıyla, ilgili Web servislerinin hangi sağlayıcılar tarafından sunulduğu ve adresleri önceden bilindiği için, *imarDurumuGetir* Web servisi geliştirilirken katalog servislerinde arama yapmaya ihtiyaç duyulmamıştır. *İmarDurumuGetir* Web servisi, Cape Clear Studio (2003) Web servisleri geliştirme yazılımı kullanılarak gerçekleştirilmiştir (Şekil 23). Servisin gerçekleştirim kodu, Java programlama dili kullanılarak yazılmıştır. Gerçekleştirim kodu, *imarDurumuGetir* Web servisinin ihtiyaç duyduğu verileri sağlayan Web servislerini çağıran ve imar durumu formunu oluşturan kodlar içermektedir. Web servislerinin hangi sırada çağrılacağı tarafımızdan belirlenmiştir. Örneğin, *imarDurumuGetir* Web servisi, parametre olarak kullanıcılardan bir parselin taşınmaz numarasını almakta ve sırasıyla kadastro sunucusundan *parselGeometriGetir* ve imar

müdürlüğü sunucusundan *imarAdaGetir* Web servislerini çağırmaktadır. Bir Web servisinin döndürdüğü veri, bir sonraki servise parametre olarak aktarılmıştır. Örneğin, *parselGeometriGetir* Web servisinin döndürdüğü parsel köşe koordinatları, *imarAdaGetir* Web servisini çağırmak için kullanılmıştır. *imarDurumuGetir* Web servisi, Cape Clear Server sunucu yazılımı kullanılarak sunulmuştur. *İmarDurumuGetir* Web servisinin işleyişi ile ilgili detaylar bölüm 2.1. de yer almaktadır.



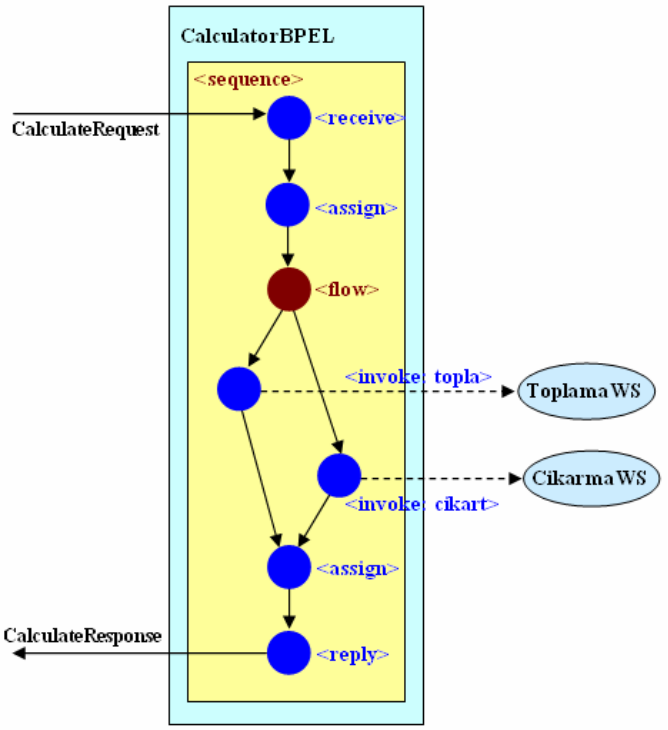
Şekil 23. Cape Clear Studio yazılımında imarDurumuGetir web servisinin geliştirilmesi

Web servisleri alanında faaliyet gösteren ve bu alanla ilgili teknolojiler geliştiren yazılım firmaları, uygulamaların kolay ve hızlı bir şekilde gerçekleştirilmesine olanak sağlamak için çeşitli servis düzenleme dilleri geliştirmiştir. CommerceNet tarafından geliştirilen eCo, Microsoft tarafından geliştirilen XLANG ve IBM tarafından geliştirilen WSFL (Web Services Flow Language) söz konusu servis düzenleme dillerinin ilk örneklerindendir (Peltz, 2003). Bu dillerden sonra geliştirilen BPEL4WS (Business Process Execution Language for Web Services), XLANG ve WFSL dillerinin yerini

almıştır. BPEL, Ağustos 2002 de BEA, IBM ve Microsoft yazılım firmaları tarafından geliştirilmiş ve Nisan 2003 de standart olarak kabul edilmesi için OASIS'e sunulmuştur (Juric vd., 2004). BPEL belirtiminin şu an yürürlükte olan son sürümü (version 2.0), Mayıs 2006 da (OASIS, 2006) yayınlanmıştır. BPEL, OASIS tarafından henüz standart olarak kabul edilmemiştir. OASIS'in WSBPEL teknik komitesi (Web Services BPEL Technical Committee), BPEL belirtimi üzerindeki çalışmalarına devam etmektedir.

BPEL, XML ve WSDL standartları üzerine kurulmuştur. BPEL, XML tabanlı bir dildir. Bir BPEL işlemi (BPEL process), mevcut Web servislerini kullanarak yeni bir uygulama geliştirmek için, BPEL şemasına uygun olarak geliştirilen bir XML dokümanıdır. Bir BPEL işlemi, düzenlemede kullanılan Web servislerini çağırmak için, bu servislerin WSDL dokümanlarında tanımlanan operasyonları ve mesajları kullanır. BPEL işlemleri bir WSDL dokümanı tarafından tanımlanır ve bir Web servisi olarak sunulur. Bir BPEL işlemi, servis düzenlemeyi gerçekleştirmek için çeşitli aktiviteler kullanır. Bu aktiviteler, "basit aktiviteler" ve "yapısal aktiviteler" olmak üzere ikiye ayrılır (OASIS, 2006). Basit aktiviteler, istemcilerden gelen istek mesajlarını almak (*receive*), değişkenlere değer atamak *(assign*), bir Web servisinin bir operasyonunu çağırmak (*invoke*), istemciye bir yanıt mesajı göndermek (*reply*), koşum anında oluşan hataları göstermek (*throw*), BPEL işlemini belirli bir süre bekletmek (*wait*) ve sonlandırmak (*terminate*) gibi genel görevleri yerine getirirler. Yapısal aktiviteler ise, basit aktiviteleri içerirler ve bu aktivitelerin nasıl yürütüleceklerini belirtirler. *Sequence*, *flow*, *switch*, *while*, *if*, *repeatUntil*, *forEach* ve *pick* BPEL belirtimi tarafından tanımlanan yapısal aktivitelerdir. *Sequence* ve *flow*, BPEL işlemlerinde en çok kullanılan yapısal aktivitelerdir. Bir *sequence* aktivitesi, belirli bir sırada yürütülecek olan bir ya da daha fazla basit aktiviteyi içerir. Aktiviteler, *sequence* elementi içerisindeki sıralarına göre yürütülürler. Bir *sequence* aktivitesi, *sequence* elementi içerisindeki son aktivite yürütüldüğü zaman tamamlanır. *Flow* aktivitesi ise, içerdiği aktivitelerin aynı anda (concurrently) yürütülmesine olanak sağlar. Yapısal aktiviteler, iç içe kullanılabilirler. Örneğin bir *sequence* aktivitesi, bir *flow* aktivitesini içerebilir.

Bu tez çalışması kapsamında, BPEL dili kullanılarak örnek bir servis düzenleme uygulaması geliştirilmiştir. Söz konusu uygulamada, bir istemci tarafından gönderilen iki sayının toplamını ve farkını bulmak için, *ToplamaWS* ve *CikarmaWS* olarak adlandırılan iki Web servisini çağıran, basit bir BPEL işlemi (*CalculatorBPEL*) geliştirilmiştir (Şekil 24).



Şekil 24. CalculatorBPEL işleminde kullanılan basit ve yapısal aktiviteler

CalculatorBPEL işleminde sırasıyla şu aktiviteler kullanılmıştır. Bir istemci tarafından BPEL işlemine gönderilen istek mesajını almak için *receive* aktivitesi kullanılmıştır. İstek mesajının içerdiği parametrelerin değerlerini, *ToplamaWS* ve *CikarmaWS* Web servislerini çağırmak için kullanılacak olan istek mesajlarındaki parametrelere atamak için *assign* aktivitesi kullanılmıştır. Aşağıdaki BPEL kodu, istemciden gelen istek mesajının alınmasını ve parametrelere değer atanmasını gerçekleştirmektedir.

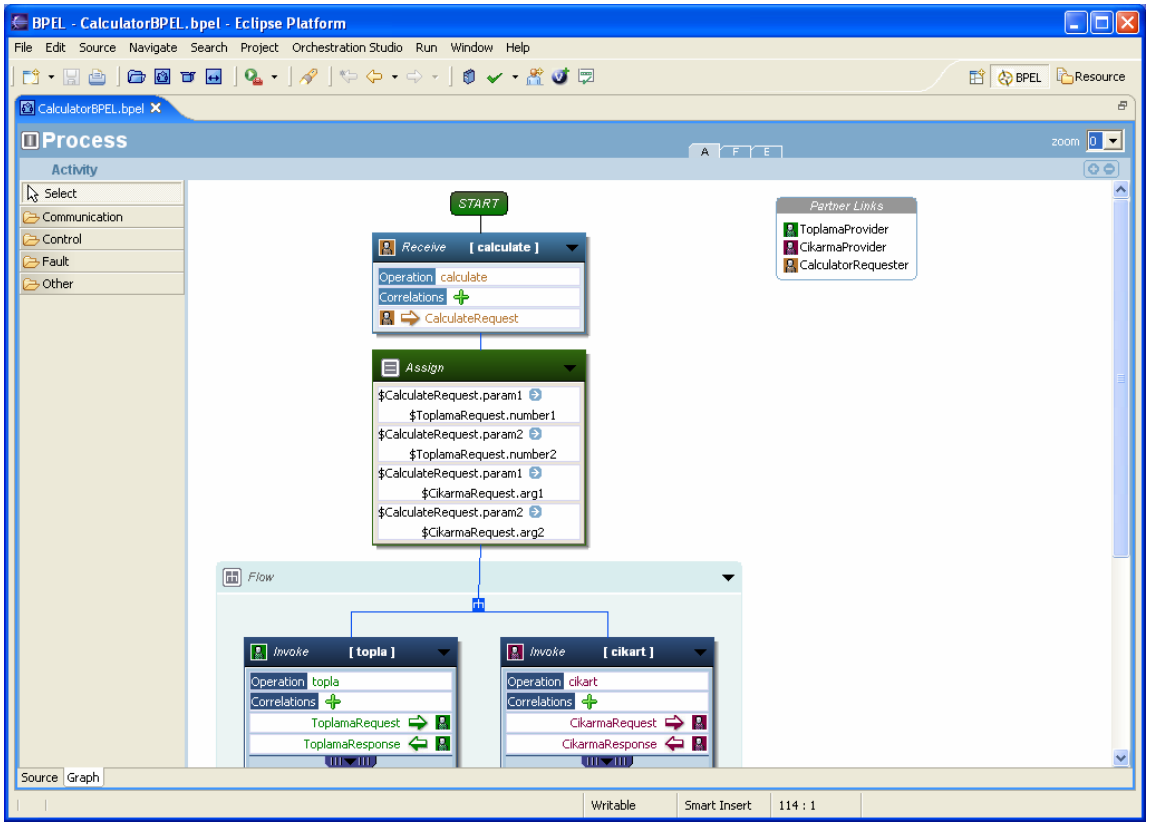
```
<bpws:receive partnerLink="CalculatorRequester"
    portType="ns8:CalculatorBPELWSPortType"
    operation="calculate" createInstance="yes" variable="CalculateRequest">
</bpws:receive>
<bpws:assign>
  <bpws:copy>
        <bpws:from variable="CalculateRequest" part="param1"></bpws:from>
        <bpws:to variable="ToplamaRequest" part="number1"/>
  </bpws:copy>
  <bpws:copy>
        <bpws:from variable="CalculateRequest" part="param2"></bpws:from>
        <bpws:to variable="ToplamaRequest" part="number2"/>
  </bpws:copy>
```

```
  <bpws:copy>
        <bpws:from variable="CalculateRequest" part="param1"></bpws:from>
        <bpws:to variable="CikarmaRequest" part="arg1"/>
  </bpws:copy>
  <bpws:copy>
        <bpws:from variable="CalculateRequest" part="param2"></bpws:from>
        <bpws:to variable="CikarmaRequest" part="arg2"/>
  </bpws:copy>
</bpws:assign>
```

CalculatorBPEL işleminin, koşum anında *ToplamaWS* ve *CikarmaWS* Web servislerini aynı anda çağırmasını sağlamak için *flow* aktivitesi kullanılmıştır. *Flow* aktivitesi, *sequence* aktivitesinde olduğu gibi, bir aktiviteyi gerçekleştirmek için bir önceki aktivitenin sonuçlanmasını beklememekte ve dolayısıyla uygulamanın daha hızlı gerçekleştirilmesini sağlamaktadır. *CalculatorBPEL* işleminin istemciye gönderdiği yanıt mesajı tek bir parametre içermektedir. Bu nedenle, "*concat*" XPath metodu kullanılarak, *ToplamaWS* ve *CikarmaWS* Web servislerinin sonuçlarının birleştirilmesi ve elde edilen yeni değerin *assign* aktivitesi kullanılarak istemciye gönderilecek olan yanıt mesajının ilgili parametresine atanması sağlanmıştır. Son olarak, *CalculatorBPEL* işleminin yanıt mesajını istemciye göndermesini sağlamak için *reply* aktivitesi kullanılmıştır. Aşağıdaki BPEL kodu, *ToplamaWS* ve *CikarmaWS* Web servislerinin sonuçlarının birleştirilmesini, elde edilen yeni değerin yanıt mesajı parametresine atanmasını ve yanıt mesajının istemciye gönderilmesini gerçekleştirmektedir.

```
<bpws:assign>
  <bpws:copy>
    <bpws:from
        expression="concat(bpws:getVariableData('ToplamaResponse', 'result')
       ,bpws:getVariableData('CikarmaResponse', 'arg') )">
    </bpws:from>
    <bpws:to variable="CalculateResponse" part="calc_response"/>
  </bpws:copy>
</bpws:assign>
<bpws:reply partnerLink="CalculatorRequester"
    portType="ns8:CalculatorBPELWSPortType"
    operation="calculate" variable="CalculateResponse">
</bpws:reply>
```

CalculatorBPEL işleminin BPEL dokümanı Ek 2. de yer almaktadır. Bir BPEL işleminin içerdiği BPEL kodlarının insanlar tarafından elle yazılması oldukça zordur ve hata eğilimlidir. Bu nedenle birçok yazılım firması BPEL işlemlerinin kolay bir şekilde geliştirilmesine olanak sağlayan araçlar geliştirmiştir. Bu araçlar, kullanıcıların BPEL işlemlerini, BPEL aktivitelerini temsil eden nesneleri kullanarak, bir iş akış şeması oluşturur gibi şematik olarak oluşturmalarına olanak sağlamakta ve BPEL dosyalarını bu şematik gösterimlerden otomatik olarak üretmektedir. *CalculatorBPEL* işlemi, Cape Clear Orchestration Studio yazılımı kullanılarak geliştirilmiştir (Şekil 25).



Şekil 25. Cape Clear Orchestration Studio yazılımında BPEL işleminin geliştirilmesi

BPEL kullanarak uygulama geliştirmek isteyen bir kullanıcının ilk olarak, katalog servislerinde arama yaparak, uygulama gereksinimlerini karşılayan Web servislerini bulması gerekmektedir. Kullanıcı, kataloglardan bulduğu tüm Web servislerinin WSDL dokümanlarını inceleyerek, düzenlemede hangi servisleri kullanacağına karar vermelidir. Kullanıcının daha sonra, Cape Clear Orchestration Studio gibi bir BPEL işlem geliştirme aracında, ilgili Web servislerinin WSDL dokümanlarını kullanarak, uygulamayı

gerçekleştirecek olan BPEL işlemini geliştirmesi gerekmektedir. Kullanıcının uygulamayı doğru bir şekilde gerçekleştirebilmesi için, Web servislerinin hangi operasyonlarını kullanacağını, operasyonları hangi sırada çağıracağını, BPEL işlemi içerisinde hangi basit ve yapısal aktiviteleri kullanacağını bilmesi gerekmektedir. Dolayısıyla kullanıcının sıradan bir Web servisleri kullanıcısı değil, XML, WSDL, BPEL ve XPath standartlarını bilen teknik bir eleman olması gerekmektedir. Servis düzenlemenin BPEL dili kullanılarak gerçekleştirilmesinin en önemli avantajı, düzenlemede kullanılan Web servislerinin arayüzlerinin değişmesi durumunda, BPEL işleminin güncellenmesinin veya yeniden üretilmesinin kolay olmasıdır.

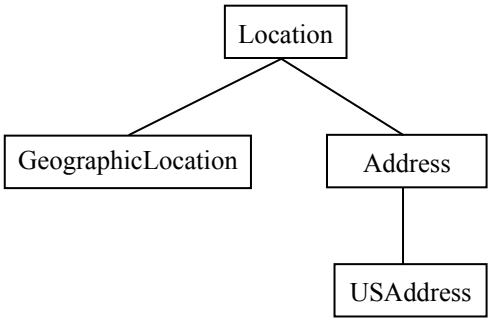
2.4.2. Yarı Otomatik Yöntem

Yarı-otomatik yöntem, servis düzenlemenin insan ve yazılım tarafından birlikte gerçekleştirildiği yöntemdir. Bu yöntemde, düzenlemede kullanılacak olan Web servisleri, insan tarafından katalog servislerinde arama yapılarak bulunur. Uygun Web servisleri bulunduktan sonra, servis parametrelerinin karşılaştırılarak, eşleşen servislerin belirlenmesi bir yazılım (reasoner) tarafından otomatik olarak gerçekleştirilir. Eşleştirme anlamsal olarak yapılır. Bu nedenle Web servislerinin, WSDL gibi sözdizimsel bir dil yerine, DAML-S veya OWL-S gibi anlamsal bir dil kullanılarak tanımlanmış olması gerekmektedir. Eşleştirme, Web servislerinin işlevsel (functional) ve işlevsel olmayan (non-functional) özellikleri kullanılarak gerçekleştirilir. İşlevsel özellikler, bir Web servisinin girdi, çıktı, önkoşul, etki/son koşul (IOPE parameters) parametreleridir. İşlevsel olmayan özellikler ise genellikle servis kalitesi parametreleri (QoS parameters) olarak anılan özelliklerdir. Anlamsal eşleştirme için en yaygın kullanılan yöntemlerden biri mantık tabanlı içerme (logic based subsumption) algoritmalarıdır. Anlamsal eşleştirme için çeşitli içerme algoritmaları geliştirilmiştir (Paolucci vd., 2002a; Şirin vd., 2003a). Yazılımın, düzenlemenin herhangi bir adımında, birden fazla eşleşen servis bulması durumunda düzenlemede hangi servisin kullanılacağına yine insan karar verir. Bu aşamada genellikle servislerin işlevsel olmayan özellikleri kullanılır. Şirin vd. (2003a), Şirin vd. (2003b), Şirin vd. (2004a) ve Lemmens vd. (2006) tarafından yapılan çalışmalar, yarı-otomatik servis düzenleme yöntemine örnek olarak gösterilebilir.

Şirin vd. (2003a), DAML-S dili kullanılarak tanımlanan anlamsal Web servislerinden, yarı-otomatik SD yöntemini kullanarak yeni bir anlamsal Web servisi yaratan prototip bir uygulama geliştirmiştir. Şirin vd. (2003a), söz konusu uygulamada servislerin önceden bulunduğunu kabul etmiş ve düzenlemenin nasıl yapılacağı konusuna odaklanmıştır. Geliştirilen prototip, düzenleyici (composer) ve çıkarsama motoru (inference engine) olarak adlandırılan iki temel bileşenden oluşmaktadır. Çıkarsama motoru, DAML-S servis tanımlarındaki ontoloji bilgilerini, RDF formatına (subject, predicate, object) dönüştürüp kendi bilgi tabanında (knowledge base) saklayan ve içerdiği mevcut çıkarsama kurallarını kullanarak eşleşen servisleri bulma yeteneğine sahip olan bir yazılımdır. Düzenleyici ise, kullanıcının servis düzenlemeyi gerçekleştirdiği bir grafik kullanıcı arayüzüdür. Düzenleyici, servis eşleşmelerini bulmak için çıkarsama motoru ile iletişim kurmakta ve her adımda, mevcut eşleşme sonuçlarını kullanıcıya göstererek düzenlemenin oluşturmasına yardımcı olmaktadır.

Şirin vd. (2003a) tarafından geliştirilen prototip uygulamada servis düzenleme şu şekilde gerçekleştirilmektedir. Düzenleyici, bilgi tabanına kayıtlı tüm Web servislerini kullanıcıya göstermektedir. Kullanıcı, bu servislerden herhangi birini seçerek düzenlemeyi başlatmaktadır. Seçilen servisin girdi parametrelerini belirlemek için bilgi tabanına bir sorgu gönderilmektedir. Servisin her bir girdi parametresi için bilgi tabanına yeni bir sorgu gönderilmekte ve girdi parametresi için uygun veriyi sağlayan olası Web servisleri çıkarsama motoru tarafından bulunarak düzenleyicide kullanıcıya gösterilmektedir. Çıkarsama motoru tarafından kullanılan anlamsal eşleştirme algoritması, parametre eşleştirmesi için "tam eşleşme" (exact match) ve "genel eşleşme" (generic match) olarak adlandırılan iki eşleştirme yöntemi kullanmaktadır. Buna göre, düzenlemede birbiri ardına kullanılacak olan iki Web servisinin girdi ve çıktı parametreleri, aynı ontolojide aynı kavramlar kullanılarak tanımlanmış ise tam eşleşme söz konusudur. Eğer bir servisin çıktı parametresinin tanımlandığı kavram, aynı ontolojide, bir başka servisin girdi parametresinin tanımlandığı kavramın alt sınıfı ise genel eşleşme söz konusudur. Örneğin, kullanıcı tarafından seçilen servisin, Şekil 26'da sınıf hiyerarşisi verilen örnek bir ontoloji parçasında, "Address" tipinde girdi parametresine sahip olduğunu kabul edelim. Kullanıcı, çıktısı "Address" tipinde olan Web servislerini aramaktadır. Çıktı parametreleri "Address" tipinde olan servisler tam eşleşme, "USAddress" tipinde olan servisler ise genel eşleşmeyi sağlayan servislerdir. Eşleştirme sonuçları, çıkarsama motoru tarafından tam eşleşmeler genel eşleşmelerden önce gelecek şekilde sıralanmaktadır. Kullanıcıların, düzenlemede

tam eşleşmeyi sağlayan Web servislerini kullanmaları önerilmektedir. Eğer çıkarsama motoru, bir Web servisine girdi parametresi sağlayan birden fazla Web servisi bulursa, kullanıcı bulunan servisleri, servis kalitesi (QoS) parametreleri gibi Web servislerinin işlevsel olmayan özelliklerine göre sorgulayarak düzenlemede hangi servisi kullanacağını belirlemektedir. Düzenlemenin her adımında, kullanıcı tarafından seçilen bir Web servisi için yukarıdaki işlemler tekrarlanmaktadır. Kullanıcı, bu şekilde adım adım ilerleyerek düzenlemede kullanacağı tüm Web servislerini belirlemektedir. Düzenleyici, bütün Web servisleri belirlendikten sonra bir DAML-S birleşik işlemi (CompositeProcess) yaratarak servis düzenlemeyi tanımlamaktadır. Şirin vd. (2003a), OWL-S dilinin DAML-S'in yerini alması nedeniyle sonraki çalışmalarında (Şirin, 2004a) aynı uygulamayı OWL-S dilini kullanarak gerçekleştirmiştir.



Şekil 26. Konumla ilgili sınıfların basit bir hiyerarşisi (Şirin, 2004a).

Şirin vd. (2003a) tarafından geliştirilen uygulamanın en önemli dezavantajı, düzenlemede kullanılan tüm Web servislerinin aynı ontolojide tanımlanmasıdır. Web servislerinin farklı ontolojiler kullanılarak tanımlanması durumunda, çıkarsama motorunun, servislerin eşleşme düzeylerini hesaplayabilmesi için, ontolojiler arasındaki ilişkileri de dikkate alması gerekmektedir. Ancak Grau vd. (2004), hem OWL nin farklı ontolojiler arasındaki ilişkileri tanımlamak, hem de mevcut OWL çıkarsamacılarının farklı ontolojilerden bilgi çıkarmak için yetersiz olduğunu, bu nedenle DDL (Distributed Description Logics) ve ε-connections gibi yeni tekniklerin kullanılması gerektiğini belirtmektedir. Şirin vd. (2003a), geliştirdikleri prototip uygulamada somut servis düzenlemeyi gerçekleştirmektedir. Çünkü düzenlemede, bilgi tabanına kayıtlı servis örneklerini kullanılmaktadır. Bu nedenle, düzenlemede kullanılan servislerden herhangi birinin kullanılabilir olmaması durumunda oluşturulan servis düzenleme çalışmayacaktır.

Servis düzenleme alanında, konumsal bilgi toplumunda önemli akademik çalışmalar yapılmaktadır. (Di vd., 2006) ve (Lemmens vd., 2006), bu alanda yapılan en son çalışmalardır. Lemmens vd. (2006) servis düzenlemenin temel işlem adımlarını, servis bulma, soyut servis düzenleme (abstract composition), somut servis düzenleme (concrete composition) ve servis yürütme olarak tanımlamıştır.

Lemmens vd. (2006) servis düzenlemedeki soyut ve somut servis düzenleme (Şirin vd., 2005) ayırımını vurgulamıştır. Servis bulma ve soyut servis düzenlemeyi gerçekleştirmek için *GeoMatchMaker*, somut servis düzenlemeyi gerçekleştirmek için *ICD* (Integrated Component Designer) isimli iki yazılım bileşeni geliştirmiştir. Örnek uygulama olarak, bir bölgedeki potansiyel tehlikeli alanları gösteren bir risk haritası üreten bir servis düzenleme ele alınmıştır.

Lemmens vd. (2006) servis düzenleme sonucunda elde edilecek yeni servisin, herhangi bir şekilde önceden tanımlı olmadığı yaklaşımı benimsemiştir. Bu yaklaşımda servis eşleştirme (Şirin vd., 2005) servislerin girdi ve çıktı parametreleri vasıtasıyla yapılır. Yani, servis düzenleme ile oluşturulacak yeni servis için, bir S^{i+1} servisinin bir S^{i} servisinden sonra kullanılabilir olması, S^{i+1} servisinin girdi parametresi tipinin, S^{i} servisinin çıktı parametresi tipi tarafından kapsanan bir tip olması ilkesi esas alınır. *GeoMatchMaker*, bir OWL çıkarsamacısı (RacerPro) ve bilgi tabanını kullanarak bu servis eşleştirme işlemini gerçekleştirir. Sınıf ve birey bazında dört çeşit eşleştirme tanımlamıştır. Birey, genelde ya da bir bilgi tabanında belirli bir sınıf tanımına ait bir örnektir. Söz konusu uygulamada *ADLGazetteer*, *LocSpot* sınıfına ait bir örnektir. Lemmens vd. (2006), *GeoMatchMaker* prototipinde bu eşleştirmelerden ikinci tipte olan, sınıfı bireye karşı karşılaştıran tipte eşleştirme kullanıldığını belirtmektedir. Buna göre, kullanıcı ilk olarak, servis düzenleme ile elde edilecek yeni servisin, etrafında bir risk haritası üreteceği kentin adını girer, örnekte "Enschede". *GeoMatchMaker* girdi tipi kent ismi olan servis örneklerini, anılan ikinci tip eşleştirme ile bulur, örnekte *ADLGazetteer*. *ADLGazetteer'in* çıktı tipi "nokta (point)" olduğundan bu servisten sonra gelebilecek servisler, girdi tipi nokta olan servisler olacaktır. *GeoMatchMaker* yine aynı eşleştirme ile *ADLGazetteer* den sonra gelebilecek servisleri bulur, örnekte *MapExtender, BboxWMSMaker, BboxCreate*. Kullanıcı bunlardan birini, Lemmens vd. (2006)'nın tabiriyle "isteğin kavramını netleştirmek" suretiyle seçer. Buradan anlaşılan kullanıcının bu üç servisin çıktılarına bakarak karar verdiğidir. Kullanıcı bir nokta etrafında bir sınırlayan dikdörtgen üretmek istediği için *BboxCreate* servisini seçer. Aynı eşleştirmeyi bir kez daha uygulayarak

GeoMatchMaker bu kez girdisi sınırlayan dikdörtgen olan servisleri bulur, örnekte *MakeGetMapRequest*.

Yeni düzenlenen servisin bileşenleri durumundaki tüm Web servisleri, *ADLGazetteer*, *BboxCreate* ve *MakeGetMapRequest,* belirlendikten sonra, *GeoMatchMaker* bir OWL-S birleşik işlem (CompositeProcess) dokümanı oluşturarak, yeni oluşturulan servisin işlem modelini (process model) tanımlamış olur. *ICD*, *GeoMatchMaker* tarafından üretilen OWL-S birleşik işlemini, bir BPEL işlemine dönüştürür. Bu BPEL işlemi artık bir BPEL işlemcisi, tarafından gerçekleştirebilir, örnekte kullanılan BPEL işlemcisi *Oracle BPEL Process Manager* dır.

Lemmens vd. (2006)'nin, "soyut servis düzenleme" olarak nitelediği, yukarıda açıklanan servis düzenleme işlemi, Şirin vd. (2005)'te ifade edilen soyut servis düzenleme den farklıdır. Şirin vd. (2005)'ye göre soyut servis düzenlemedeki amaç, düzenlemede kullanılacak olan Web servislerini tasarım anında sabitleştirmemek, uygun Web servislerini koşum zamanında bularak bir esneklik sağlamaktır. Bu Web servisleri literatüründe "hata kaldırabilme" (fault-tolerance) olarak anılmakta ve herhangi Web servisleri icra ortamında arzu edilen bir özellik olarak nitelenmektedir. Gerçekten de herhangi bir soyut servis tanımının, koşum anında somut bir servise dönüştürülmesi işlemi, bileşen servis örneklerinden birinin bulunamaması durumunda, eğer gerekli hata kaldırabilme önlemleri alınmamışsa, başarısız olacaktır. Buradaki hata kaldırabilme önlemi, örneğin aranan tipte bir servis örneğinin koşum anında bulunamaması durumunda, alternatif olarak ne tip servis örneklerinin tercih edilebileceğinin, soyut servis tanımının içine sokulması olarak tanımlanabilir. Bu amaçla Şirin vd. (2005), OWL-S servis tanımı için bir genişletme önermiştir. Bu genişletme, bir OWL-S servis tanımının işlem modeli içerisinde, yeni bir işlem tipi olarak *AbstractProcess* tanımlamasının bulunabilmesi şeklindedir. Şirin vd. (2005) ayrıca, somut servis düzenlemesini oluşturacak, bileşen servis örneği alternatifleri arasından, "en uygun" olanlarının seçilebilmesine, belirli bir derecelendirme (ranking) bazında olanak tanıyacak bir genişletme de önermiştir. Buna göre servisler "Servis Kalitesi Parametreleri" (Quality of Services / QoS parameters) bazında derecelendirilmektedir. Lemmens vd. (2006)'da *GeoMatchMaker* tarafından üretilen ve "soyut servis düzenlemesi" olarak anılan düzenlemede ise, Şirin vd. (2005) de bahsedilen anlamda bir soyutlama söz konusu değildir. Bu tez çalışması kapsamında önerilen sınıflandırmaya göre, Lemmens vd. (2006)'daki düzenleme, yarı-otomatik bir düzenlemedir. Çünkü servis örneği alternatifleri yazılım tarafından bulunmakta,

alternatifler arasından seçim ise kullanıcı tarafından yapılmaktadır. Ancak burada kullanıcının bu seçme işlemini kolaylaştıracak, örneğin Şirin vd. (2004a) teki gibi bir araçtan bahsedilmemektedir. Örneğin *MapExtender, BboxWMSMaker, BboxCreate* servis örneği alternatiflerinden *BboxCreate* alternatifinin seçiminin nasıl yapıldığı, Lemmens vd. (2006)'de açık olarak belirtilmese de, bunun için kullanıcının bu servisler hakkında bir şekilde bilgi sahibi olması gerekir ki, bu da çok pratik bir çözüm gibi görünmemektedir. Özellikle çok sayıda servis örneği ve farklı ontolojilerin söz konusu olduğu durumlarda kullanıcının işi çok zor olacaktır.

Dinamik servis düzenleme, servis güncelliği açısından önem arz etmektedir. Servis düzenlemenin yapıldığı an ile somut servis düzenlemenin yapıldığı an arasında, somut servislerin tanımları değişmişse ya da artık katalogda yayınlanmıyorlarsa somut servis oluşturulamayacaktır.

Lemmens vd. (2006), düzenlemede kullanılan Web servislerini WSDL-S, servis tanımlarında kullanılan alan ontolojilerini ise OWL-DL dillerini kullanarak tanımlanmıştır. Lemmens vd. (2006), Web servislerini yayınlamak için hangi katalog servisinin kullanıldığını belirtmemiştir.

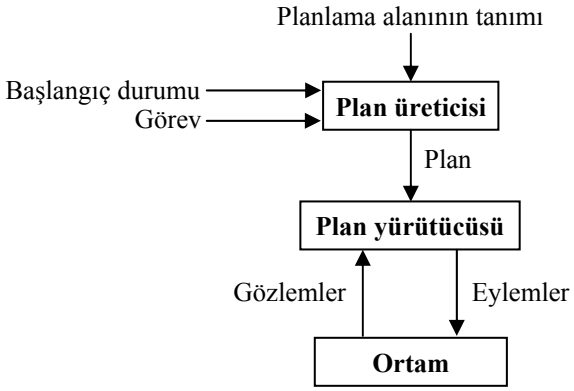
2.4.3. Tam Otomatik Yöntem

Tam otomatik SD yöntemi, tüm SD işlem adımlarının bir yazılım tarafından gerçekleştirildiği yöntemdir. Tam otomatik SD uygulamalarını gerçekleştirmek için kullanılması gereken tekniklerin henüz tam olarak netleşmediği belirtilmektedir (Peer, 2005). Otomatik SD uygulamaları günümüzde, "Yapay Zeka Planlama" (Artificial Intelligence Planning) yöntemleri kullanılarak gerçekleştirilmektedir. Otomatik SD uygulamalarında kullanılan Yapay Zeka Planlama (YZP) yöntemleri, Rao ve Su (2004) ve Peer (2005) tarafından sınıflandırılmıştır.

YZP yöntemleri, bir SD uygulamasını bir planlama problemi olarak ele alırlar. Bir YZP planlama problemi genellikle, olası eylemlerin (actions) tanımını, başlangıç durumunun (initial state) tanımını ve istenen görevin (goal/task) tanımını içerir (Peer, 2005). YZP plan üreticisi (planner), söz konusu tanımları kullanarak, istenen görevi gerçekleştirmek için kullanılması gereken eylemleri içeren bir plan üreten yazılım bileşenidir.

Hierarchical Task Networks (HTN), otomatik SD uygulamalarını gerçekleştirmek için kullanılan YZP yöntemlerinden biridir. HTN de bir planlama problemi, (*S, T, D*) üçlüsü tarafından tanımlanmaktadır (Erol vd., 1995; Wu vd., 2003; Nau vd., 2005). Bu üçlüden "*S*" başlangıç durumunu, "*T*" başlangıç görev ağını (initial task network) ve "*D*" planlama alanını (planning domain) temsil etmektedir. Başlangıç durumu, HTN plan üreticisinin, bir planı oluşturmaya başlamadan önceki mevcut durumu temsil eder. Başlangıç görev ağı, belirli koşulların sağlanması durumunda gerçekleştirilecek olan görevler setidir. Planlama alanı ise, planlama problemini çözmek için kullanılacak olan operatörler ve metotlar setidir. Operatörler, plan yürütücüsünün gerçekleştirebileceği basit görevleri (primitive tasks) tanımlarlar ve bir dizi önkoşula ve etkiye sahip olabilirler. Bir operatör yürütülmeden önce, sahip olduğu önkoşulların sağlanmış olması gerekmektedir. Metotlar, karmaşık görevleri (non-primitive tasks), basit görevlere ayırmak için kullanılan yöntemlerdir. Bir metot, bir dizi önkoşula sahip olabilir. Bir metodun uygulanabilmesi için önkoşullarının sağlanmış olması gerekir.

Parametre olarak (*S, T, D*) üçlüsünü alan bir HTN plan üreticisi, geriye bir plan döndürür (Şekil 27). Plan, istenen görevi gerçekleştiren operatörleri içerir. Bir HTN plan üreticisi, bir planı şu şekilde oluşturur. Plan üreticisi, başlangıç görev ağından, istenen görevi yerine getirmek için gerçekleştirilmesi gereken ilk görevi seçer. Eğer seçilen ilk görev bir basit görev ise plan üreticisi bu görevi yerine getirecek olan operatörü seçer. Başlangıç durumunun, seçilen operatörün önkoşulunu sağlaması durumunda plan üreticisi bir sonraki görevi belirler. Eğer seçilen ilk görev bir karmaşık görev ise, plan üreticisi uygun bir metot seçerek karmaşık görevi basit görevlere ayırır. Planlama işlemi, planın tamamı basit görevlerden oluşuncaya kadar sürdürülür.



Şekil 27. Planlama için basit bir kavramsal model (Nau vd., 2005).

Bir HTN plan üreticisinin, bir planlama problemini nasıl çözdüğü, aşağıda örnek bir uygulama ile açıklanmaktadır. Aşağıdaki sözde-kodlar (pseudocode), basit bir planlama alanını tanımlamaktadır. Söz konusu planlama alanı, bir şahsın bir restorandan yiyecek almak için gerçekleştirmesi gereken görevleri tanımlayan operatörleri ve metotları içermektedir. Uygulamada, restorana gitmek için iki yöntem tanımlanmıştır. Bu yöntemler, "*yürüyerek-git*" ve "*taksiyle-git*" metotları kullanılarak temsil edilmiştir. *Yürüyerek-git* metodu, bir önkoşula sahiptir. Bu önkoşul, restorana yürüyerek gidilebilmesi için şahıs ile restoran arasındaki mesafenin 2 km'den az olmasını gerektirmektedir. Eğer bu önkoşul sağlanırsa, *yürüyerek-git* metodu görevi basit bir göreve ayırmaktadır. Bu görev, "*yürü*" görevidir. *Taksiyle-git* metodunun da bir önkoşulu vardır. Bu önkoşul, şahsın taksi ücretini ödeyecek kadar paraya sahip olmasını gerektirmektedir. Eğer bu önkoşul sağlanırsa, *taksiyle-git* metodu görevi üç basit göreve ayırmaktadır. Bunlar, "*taksi-çağır*", "*taksiye-bin*", "*taksi-parasını-öde*" görevleridir.

method yürüyerek-git
  görev: git(a, x, y)
  önkoşul: mesafe(x, y) < 2
  alt görevler: yürü(a, x, y)

method taksiyle-git
  görev: git(a, x, y)
  önkoşul: nakit(a) ≥ 2.5 + 1 * mesafe(x,y)
  alt görevler: taksi-çağır (a,x) → taksiye-bin(a,x,y) → taksi-parasını-öde(a,x,y)

operator yürü(a, x, y)
  önkoşul: yer(a) = x
  etki: yer(a) ← y

operator taksi-çağır(a, x)
  etki: yer(taksi) ← x

operator taksiye-bin(a,x,y)
  önkoşul: yer(taksi) = x, yer(a) = x
  etki : yer(taksi) ← x, yer(a) ← y

operator taksi_parasını-öde(a,x, y)
  önkoşul: nakit(a) ≥ 2.5 + 1 * mesafe(x,y)
  etki: nakit(a) ← nakit(a) − [2.5 + 1 * mesafe(x,y)]

Yukarıdaki metot ve operatör tanımlarında yer alan sembollerden, "a" yiyecek alacak şahsı, "x" bu şahsın bulunduğu yeri, "y" ise restoranın yerini temsil etmektedir. Şahıs, restorana ulaştıktan sonra siparişini vermeli ve yiyeceklerin parasını ödemelidir. Bu görev

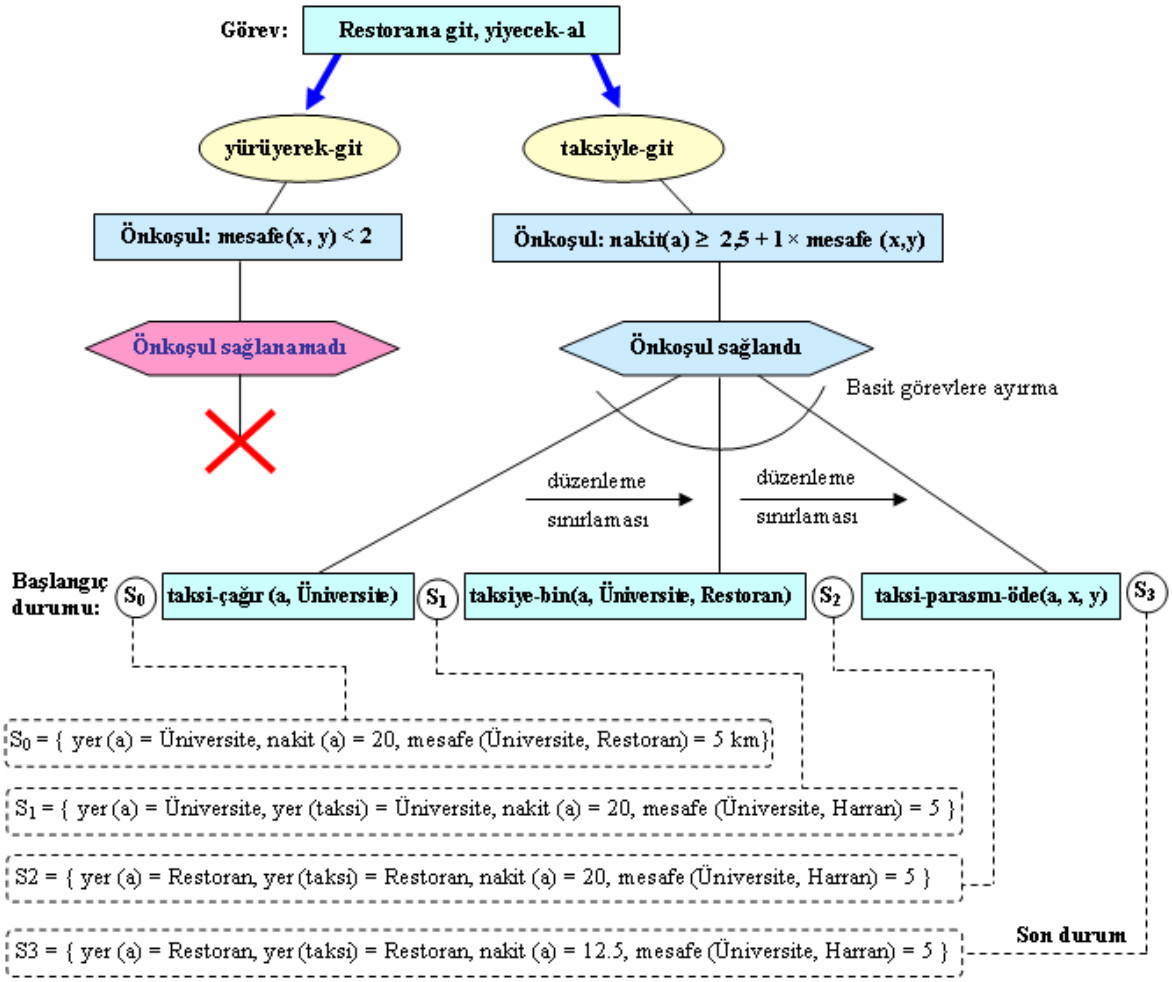
"*yerinde-sipariş-ver*" metodu ile temsil edilmiştir. Bu görevin bir önkoşulu vardır. Bu önkoşul, şahsın yiyeceklerin ücretini ödeyecek kadar paraya sahip olmasını gerektirmektedir. Eğer bu önkoşul sağlanırsa, *yerinde-sipariş-ver* metodu görevi iki basit göreve ayırmaktadır. Bunlar, "*sipariş-ver*" ve "*siparişlerin-parasını-öde*" görevleridir.

method yerinde-sipariş-ver
görev: yiyecek-al (a,y,z)
önkoşul: nakit(a) ≥ ücret (z)
alt görevler: *sipariş-ver* (a,y,z) → *siparişlerin-parasını-öde* (a,y,z)

operator sipariş_ver(a, y, z)
önkoşul: yer(a) = y

operator siparişlerin-parasını-öde(a, y,z)
önkoşul: nakit(a) ≥ ücret (z)
etki: nakit(a) ← nakit(a) − ücret (z)

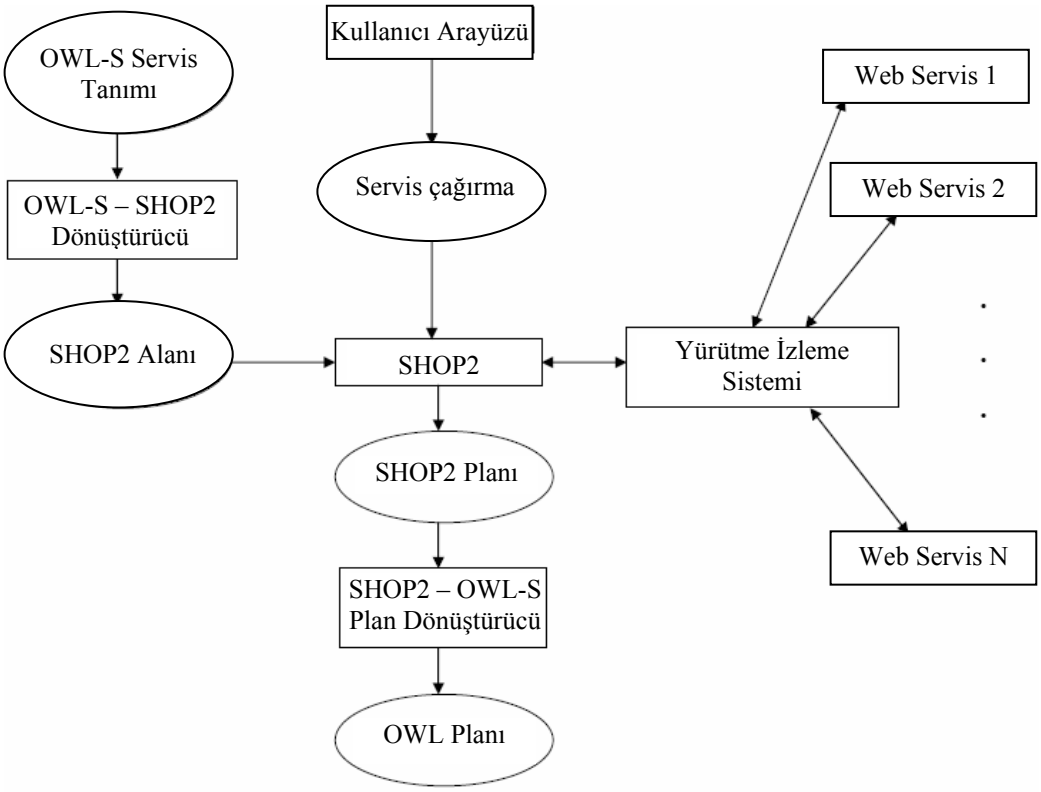
Bu problemin başlangıç durumu şu şekildedir. Şahıs, üniversitede bulunuyor, 20 YTL si var ve 5 km uzaklıktaki bir restorandan 4 lahmacun ve 2 ayran almak istiyor. Lahmacunların tanesi 2 YTL, ayranların tanesi 1 YTL dir. HTN plan üreticisinin, restorana gitme görevini yerine getirmek için ilk olarak "*yürüyerek-git*" metodunu seçtiğini kabul edelim. Plan üreticisi, başlangıç durumunu kontrol ettiğinde, "*yürüyerek-git*" metodunun önkoşulunun sağlanamadığını görür ve "*taksiyle-git*" metodunu seçer. *Taksiyle-git* metodunun önkoşulu sağlandığı için plan üreticisi söz konusu metodu, yukarıda belirtilen şekilde basit görevlere ayırır (Şekil 28). Plan üreticisi, sırasıyla ilgili basit görevlerin önkoşullarına bakar, sağlanamayan koşul olup olmadığını kontrol eder ve tüm önkoşullar sağlanıyorsa, restorandan yiyecek alma görevini yerine getirmek için, "*yerinde-sipariş-ver*" metodunu seçer. Plan üreticisi, "*taksiyle-git*" metodundan sonra oluşan son koşulu kontrol ettiğinde, şahsın istediği yiyecekleri alacak kadar parasının kaldığını, dolayısıyla *yerinde-sipariş-ver* metodunun önkoşulunun sağlandığını belirler. Plan üreticisi, *yerinde-sipariş-ver* metodunu, yukarıda metot tanımında belirtilen şekilde basit görevlere ayırır. Plan üreticisi, sırasıyla "*sipariş-ver"* ve "*siparişlerin-parasını-öde*" basit görevlerinin ön koşullarına bakar, sağlanamayan koşul olup olmadığını kontrol eder ve tüm ön koşullar sağlanıyorsa planı üretir. Şekil 28, HTN plan üreticisinin planı oluşturmak için izlediği işlem adımlarından bir bölümünü şematik olarak göstermektedir.



Şekil 28. Bir planlama probleminin çözümü (Nau vd., 2005).

HTN planlama yöntemi, Wu vd. (2003) ve Şirin vd. (2004b) tarafından otomatik SD uygulamalarını gerçekleştirmek için kullanılmıştır. Şirin vd. (2004b), servis düzenlemenin bir planlama problemi olarak ele alınabileceğini ve kullanıcılar tarafından istenen bir görevi gerçekleştirmek için kullanılması gereken Web servislerinin bir HTN plan üreticisi tarafından otomatik olarak bulunabileceğini ileri sürmüştür. Şirin vd. (2004b), servis düzenlemenin kullanıcıların müdahalesi olmadan otomatik olarak gerçekleştirilebilmesi için, Web servislerinin "bilgisayar tarafından okunabilir" (machine-readable) bir tanımlama dili kullanılarak tanımlanması gerektiğini belirtmiş ve bunun için Web servislerini OWL-S dilini kullanarak tanımlamıştır. Şirin vd. (2004b), plan üreticisi olarak bir HTN planlama sistemi olan SHOP2 (Simple Hierarchical Ordered Planner 2) (Nau vd., 2003) yazılımını kullanmıştır. Şirin vd. (2004b), OWL-S servis tanımlarını, HTN planlama alanı tanımlarına dönüştürmek için bir dönüştürücü (OWL-S–SHOP2 dönüştürücüsü) geliştirmiştir (Şekil 29). OWL-S–SHOP2 dönüştürücüsü, HTN planlama alanını

tanımlamak için, OWL-S işlem modelindeki (process model) atomik işlemleri (atomic process) HTN operatörlerine, birleşik işlemleri (composite process) ise HTN metotlarına dönüştürmektedir. SHOP2 plan üreticisi, operatör ve metot tanımlarını kullanarak istenen bir görevi gerçekleştiren HTN planını üretmektedir.



Şekil 29. Sistem mimarisi (Şirin vd. 2004b).

Şirin vd. (2004b), kullanıcılar tarafından istenen görevlerin tanımlanmasına olanak sağlayan bir arayüz geliştirmiştir. Kullanıcı arayüzü, istenen herhangi bir görev için OWL-S servis tanımlarının yüklendiği basit bir arayüzdür. Kullanıcı, bu arayüzden bir servis seçtiği zaman servisin girdi parametrelerinin girilmesine olanak sağlayan bir form ile karşılaşır. Söz konusu form, servisin girdi parametrelerini tanımlamak için kullanılan ontolojilerden üretilir. Kullanıcı tarafından seçilen servis, istenen görevi yerine getirmek için tanımlanan bir OWL-S birleşik servisidir. Planlama işlemi, kullanıcı tarafından seçilen servisin tüm girdi parametreleri sağlandıktan sonra başlamaktadır. SHOP2 plan üreticisi, istenen görevi yerine getirecek olan planı üretir, oluşturulan plan "SHOP2–OWL-S Plan Dönüştürücüsü" tarafından OWL-S formatına (OWL-S CompositeProcess) dönüştürülür ve "OWL-S Web servis yürütücüsü" tarafından yürütülür. SHOP2–OWL-S plan

dönüştürücüsü ve OWL-S Web servis yürütücüsü, Şirin vd. (2004b) tarafından geliştirilen diğer yazılımlardır.

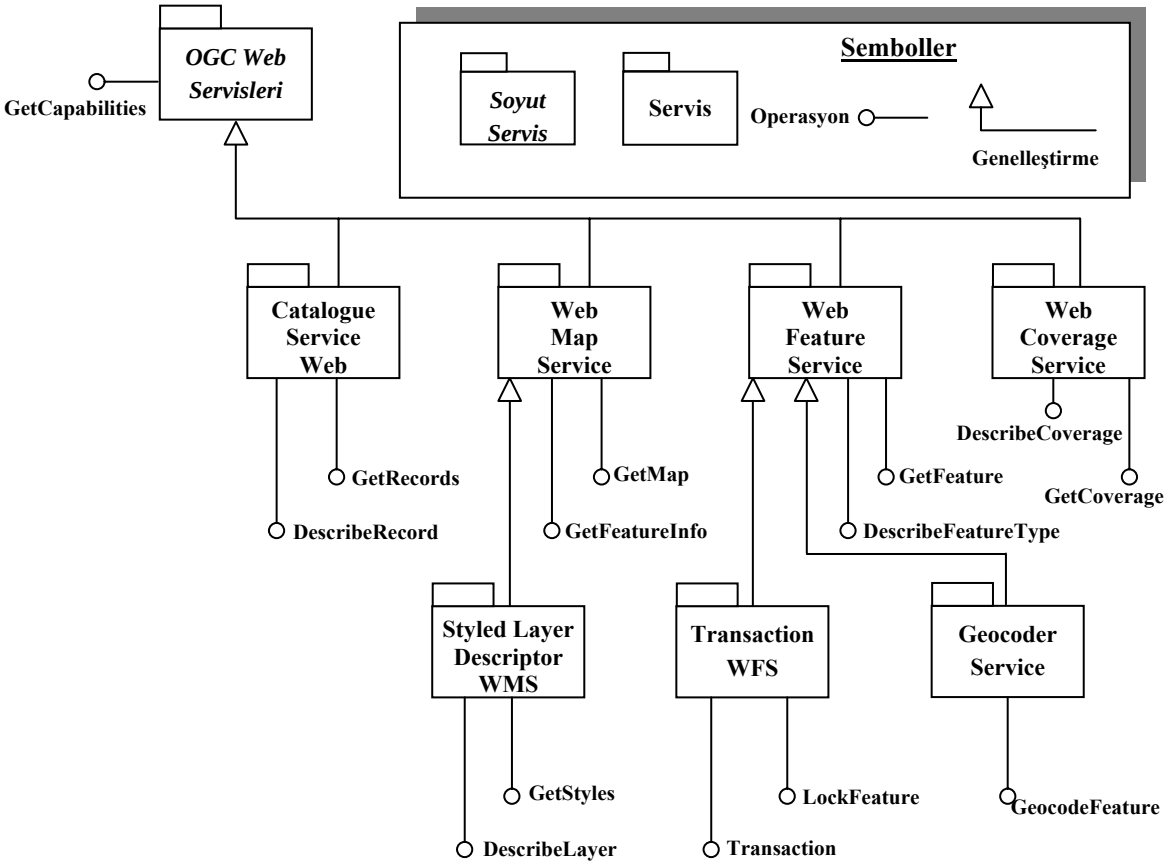
YZP sistemlerinin en önemli dezavantajı olarak, söz konusu sistemlerin bir "hata giderme" (fault-tolerance) mekanizmasına sahip olmamaları ve bu nedenle düzenlemede kullanılan bir Web servisinin artık kullanılabilir olmaması durumunda oluşturulan planın yürütülememesi gösterilmektedir (Majithia vd., 2004).

2.5. OGC Web Servisleri Mimarisi

1994 yılında kurulan Open Geospatial Consortium (OGC), Coğrafi Bilgi Sistemleri (CBS) şirketlerinden, kamu kurumlarından ve üniversitelerden oluşan, yaklaşık 300 üyeye sahip, kar amacı gütmeyen uluslararası bir endüstri birliğidir. Birliğin amacı, mevcut CBS sistemleri arasındaki birlikte işlerlik (interoperability) problemlerini belirlemek ve bilgi teknolojilerindeki gelişmeleri yakından izleyerek bu problemleri giderecek, herkesin kullanımına açık arayüz belirtimleri geliştirmektir. OGC, bu amaç doğrultusunda çeşitli birlikte işlerlik girişimleri (interoperability initiative) başlatmış ve bu girişimlerin sonucunda, WCS (Web Coverage Service) (OGC, 2003b), WMS (Web Map Service) (OGC, 2004c) ve WFS (Web Feature Service) (OGC, 2005c) Web servislerini geliştirilmiştir.

2.5.1. Servis Tanımlama

OGC, Web servislerinin yapısını şekillendiren kapsamlı bir servis modeline sahiptir (Doyle ve Reed, 2001). Bu modele göre her OGC Web servisi, operasyonlarından biri *GetCapabilities* olan bir arayüze sahip olmak zorundadır (Şekil 30). *GetCapabilities* operasyonu istemcilere, bir OGC Web servisinin desteklediği operasyonları, sahip olduğu coğrafik verileri tanımlayan ve "servis meta verisi" (service metadata) olarak adlandırılan bir XML dokümanı döndürür. OGC Web servisleri, bir servis meta veri dokümanı tarafından tanımlanır. Servis meta veri dokümanı, "yetenekler dokümanı" (capabilities document) olarak ta adlandırılır.



Şekil 30. OGC web servisleri mimari diyagramı (OGC, 2001).

Servis meta veri dokümanları, "*ServiceIdentification*", "*ServiceProvider*", "*OperationsMetadata*" ve "*Contents*" olarak adlandırılan dört bölümden oluşur (OGC, 2005d). *ServiceIdentification* bölümü, bir OGC Web servisi ile ilgili temel meta verileri içerir. Örneğin bir OGC Web servisinin tipi, sürümü, kullanım ücreti ve erişim kısıtlamaları gibi bilgileri bu bölümde yer alır. *ServiceProvider* bölümü, servisi sunan organizasyonla ilgili bilgiler içerir. Organizasyonun adı, Web sitesinin adresi ve iletişim bilgileri bu bölümde yer alır. *OperationsMetadata* bölümü, servisin gerçekleştirdiği operasyonları ve parametrelerini tanımlayan bölümdür. *Contents* ise, servis tarafından sunulan konumsal verilerle ilgili meta verilerin tanımlandığı bölümdür. Bu bölümde yer alan meta veriler, ISO 19115 meta veri standardının "*MD_DataIdentification*" sınıfına göre tanımlanır. *Contents*, servisinin tipine bağlı olarak farklı isimler alabilir. Örneğin *Contents*, WFS servislerini tanımlayan servis meta veri dokümanlarında *FeatureTypeList*, WMS servislerini tanımlayan servis meta veri dokümanlarında ise *Layer* olarak adlandırılmaktadır. Servis meta veri dokümanları ayrıca, *Filter_Capabilities* olarak

adlandırılan seçmeli bir bölüm içerebilirler. *Filter_Capabilities* bölümü, bir OGC Web servisinin desteklediği konumsal ve sayısal operatörleri tanımlayan bölümdür. Kullanıcılar, bir OGC Web servisini sorgulamak için bu bölümde yer alan operatörleri kullanırlar. Ek 3. de, bir OGC WFS servisini tanımlayan örnek bir servis meta veri dokümanı yer almaktadır.

2.5.2. Servis Bulma

OGC Web servisleri mimarisinde, bir kullanıcının aradığı servisleri bulmasını sağlayan mekanizma, CSW katalog servisleridir. Servis meta verileri, servis sağlayıcıları tarafından katalog servislerine kayıt edilirler. Kullanıcılar, katalog servislerinde arama yaparak ihtiyaç duydukları OGC Web servislerini bulurlar. Kullanıcıların bir CSW katalog servisinde arama yapabilmeleri için sırasıyla aşağıdaki CSW operasyonlarını kullanmaları gerekmektedir.

GetCapabilities: GetCapabilities operasyonu kullanıcılara, CSW nin servis meta veri dokümanını döndürür.

DescribeRecord: DescribeRecord operasyonu kullanıcılara, CSW bilgi modelinin şema tanımını gösteren bir XML dokümanı döndürür. CSW de arama yapacak olan bir kullanıcının, sorgusunu yazabilmesi için CSW'nin bilgi modelini bilmesi gerekmektedir. Örneğin kullanıcı bir organizasyon tarafından sunulan WFS servislerini arıyor ise bilgi modelinde, organizasyonları ve servisleri tanımlamak için kullanılan meta veri sınıflarının özniteliklerini bilmelidir.

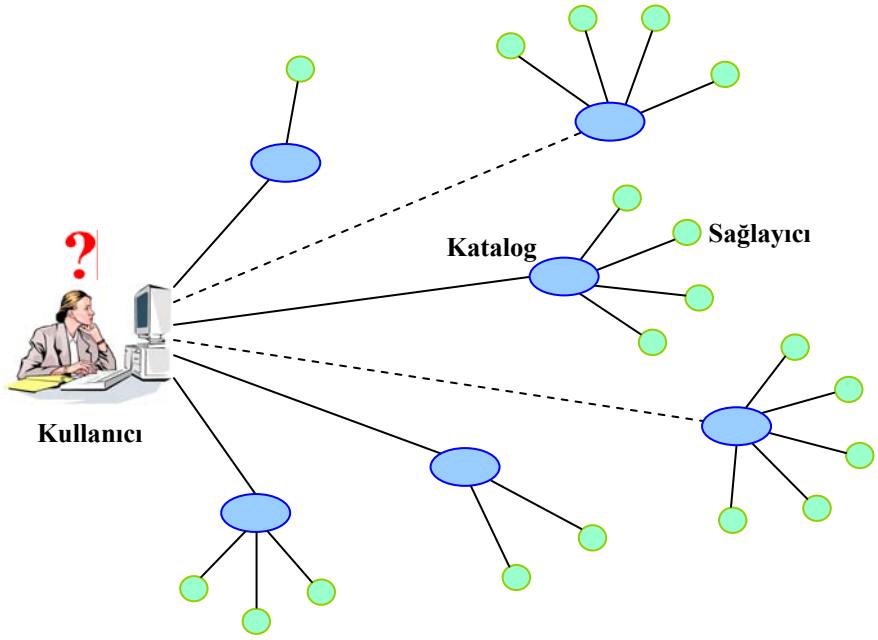
GetRecords: GetRecords operasyonu, kullanıcıların bir CSW katalog servisinde arama yapmasına ve meta veri kayıtlarını elde etmesine olanak sağlar. GetRecords operasyonu, CSW nin şema tanımına göre yazılan ve OGC Filter sözdizimine uyan bir sorgu ifadesi içerir.

OGC Web servisleri, farklı kamu kurumları veya özel sektöre ait birçok katalog servisi tarafından sunulabilir (Şekil 31). Bir kullanıcının, aradığı OGC servislerinin hangi kataloglar tarafından sunulduğunu bilip bilmemesine ve katalog servisleri arasında bir federasyonun olmasına bağlı olarak, kataloglarda arama yapmak için kullanabileceği üç yöntem vardır.

Kullanıcı eğer bir uygulama için ihtiyaç duyduğu servisleri hangi kataloglarda arayacağını biliyorsa, ilgili katalog servislerinde sırasıyla yukarıda tanımlanan CSW operasyonlarını çağırır. Bu yöntemin üç temel dezavantajı vardır. Birincisi, CSW

operasyonlarını çağırmak için yazılan istek mesajları XML formatında olduğu için kullanıcı XML söz dizimini bilmelidir. İkincisi, kullanıcı GetRecords operasyonundaki sorgu ifadesini yazabilmek için OGC Filter söz dizimini bilmelidir. Üçüncüsü, kullanıcı CSW belirtimini bilmeli ve CSW yanıt dokümanlarını anlayabilmelidir.

Kullanıcı, eğer aradığı servisleri hangi kataloglarda bulacağını bilmiyorsa, sırasıyla tüm katalog servislerinde arama yapmalıdır. Bu durumda kullanıcının, sırasıyla tüm katalog servislerinde CSW operasyonlarını çağırması gerekmektedir. Bu yöntemin en önemli dezavantajı, en kötü durumda tüm katalog servislerinde arama yapmak gerekirse istenen servislerin bulunması oldukça uzun zaman alacaktır.



Şekil 31. Katalog servisleri ve servis sağlayıcıları

Kullanıcı, katalog servislerinin arasında bir federasyonun olması durumunda, katalogları sorgulamak için "dağıtık arama" (distributed search) yapabilir. Kullanıcı bu durumda sorgusunu yazmak için "Temel Sorgulanabilir Öznitelikleri" (TSÖ) kullanır. TSÖ'ler, bir kullanıcının katalog servislerinin bilgi modellerinin detaylarını bilmeden kataloglarda arama yapmasına olanak tanır. Bu durumda kullanıcının *GetCapabilities* ve *DescribeRecord* operasyonlarını kullanmasına gerek yoktur. Kullanıcının, TSÖ'leri kullanarak yazdığı bir *GetRecords* istek mesajını, herhangi bir katalog servisine göndermesi aramayı gerçekleştirmesi için yeterlidir. İstek mesajını alan katalog servisi,

aynı mesajı federasyona üye olan tüm katalog servislerine dağıtır. Bu yöntemin en önemli dezavantajı, TSÖ'lerin kullanılması durumunda anahtar kelimelere dayalı bir arama gerçekleştirildiği için, kataloglarda servisleri tanımlamak için kullanılan anahtar kelimelerin, sorgudaki anahtar kelimeler ile eşleşmemesi durumunda, aranan servisler kataloglarda olsa dahi bulunamama riskleri vardır. Örneğin aşağıdaki XML kodu, bir kullanıcının yol verisini sunan WFS servislerini bulmak için TSÖ leri kullanarak yazdığı bir *GetRecords* istek mesajını göstermektedir. İstek mesajında, aranan WFS servislerini tanımlamak için "yol" ve "wfs" anahtar kelimeleri kullanılmıştır. Eğer kataloglarda, ilgili WFS servislerini tanımlamak için "ulaşım" ve "wfs" anahtar kelimelerinin kullanıldığı kabul edilirse, kullanıcı aradığı servisleri bulamayacaktır.

```
<?xml version="1.0" encoding="UTF-8"?>
<GetRecords
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:csw="http://www.opengis.net/cat/csw"
  version="2.0.0"
  outputFormat="application/xml"
  outputSchema="csw:Record">
  <csw:Query typeNames="csw:Record">
    <csw:ElementName>summary</csw:ElementName>
    <csw:Constraint>
      <ogc:Filter>
        <ogc:And>
          <ogc:PropertyIsLike wildCard="%" singleChar="_" escape="\">
            <ogc:PropertyName>/csw:Record/csw:AnyText</ogc:PropertyName>
            <ogc:Literal>%yol ağı%</ogc:Literal>
          </ogc:PropertyIsLike>
          <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>/csw:Record/csw:Subject</ogc:PropertyName>
            <ogc:Literal>yol</ogc:Literal>
          </ogc:PropertyIsEqualTo>
          <ogc:PropertyIsEqualTo>
          <ogc:PropertyName>/csw:Record/csw:Subject</ogc:PropertyName>
            <ogc:Literal>wfs</ogc:Literal>
          </ogc:PropertyIsEqualTo>
        </ogc:And>
      </ogc:Filter>
    </csw:Constraint>
  </csw:Query>
</GetRecords>
```

2.5.3. Servis Düzenleme

OGC Web servisleri, konumsal veri sunmak için geliştirilmiş "insan-odaklı" servislerdir. Bir OGC Web servisi tarafından sunulan konumsal verileri elde etmek isteyen bir kullanıcının, ilgili servis tarafından gerçekleştirilen bir dizi operasyonu kullanması gerekmektedir. Örneğin bir WFS servisi tarafından sunulan detayları elde etmek isteyen bir kullanıcı, sırasıyla şu WFS operasyonlarını kullanmalıdır. Kullanıcı ilk olarak *GetCapabilities* operasyonunu kullanarak servisin yetenekler dokümanını almalıdır. Kullanıcı eğer özniteliklere dayalı bir sorgu gerçekleştirecekse, *DescribeFeatureType* operasyonunu kullanarak WFS den ilgili detayların şema dokümanlarını almalı ve bu dokümanları inceleyerek sorgusunda kullanacağı öznitelikleri belirlemelidir. Kullanıcı son olarak, ilgili detayların özniteliklerini kullanarak OGC Filter sözdizimine uyan bir sorgu yazmalı ve sorguyu *GetFeature* operasyonunu kullanarak WFS servisine göndermelidir. Ayrıca bir kullanıcının, bir WFS servisinden konumsal verileri alabilmesi için servise bir takım verileri göndermesini gerektiren durumlar söz konusu olabilir. Örneğin, yol ve parsel verilerinin iki ayrı WFS servisi tarafından sunulduğunu kabul edelim. Bir kullanıcının, yola komşu olan parselleri elde etmek istemesi durumunda, ilk olarak yol verisini sunan WFS servisinden yol detaylarını alması ve bu detayları parsel verisini sunan WFS servisine göndererek burada bir analiz gerçekleştirmesi gerekmektedir. OGC Web servisleri, mevcut işleyişleri ile uygulamaların insanlar tarafından gerçekleştirilmesine olanak sağlayan bir modele sahiptir ve bu nedenle servis düzenlemeyi gerçekleştirmek için uygun değillerdir.

OGC Web servislerini kullanarak servis düzenlemeyi gerçekleştirmenin iki yolu vardır. Bunlardan birincisi, OGC Web servislerini BPEL işlemleri içerisinde kullanarak servis düzenlemeyi gerçekleştirmektir. Bu yöntem, OGC Web servislerinin WSDL dili kullanılarak tanımlanmış olmasını gerektirmektedir. OGC, WCS, WMS ve WFS Web servislerinin WSDL tanımlarını üretmek ve söz konusu servisleri kullanarak servis düzenlemeyi gerçekleştirmek için çeşitli çalışmalar yapmıştır (OGC, 2003c; OGC, 2004d). OGC Web servislerini gerçekleştiren yazılımlar geliştiren Ionic, Galdos ve MapInfo gibi yazılım firmalarının da katıldığı bu çalışmalarda, OGC Web servislerini tanımlayan WSDL dokümanları geliştirilmiştir. OGC Web servislerinin gerçekleştirdikleri operasyonlar, zorunlu ve seçmeli parametrelere sahiptir. Örneğin, bir WFS servisinden detayların alınmasına olanak sağlayan *GetFeature* operasyonunun, altı adet seçmeli (*handle*, *outputFormat*, *propertyName*, *filter*, *maxFeatures*, *featureVersion*) parametresi vardır.

Ancak WSDL seçmeli parametrelerin tanımlanmasına olanak tanımadığı için, OGC Web servislerinin WSDL tanımlarında tüm operasyon parametreleri zorunlu olarak tanımlanmıştır. OGC tarafından, WFS servislerini tanımlamak için geliştirilen WSDL dokümanından alınan aşağıdaki mesaj tanımı, *GetFeature* operasyonunun istek mesajını (*GetFeatureRequest*) ve operasyonun girdi parametrelerini göstermektedir.

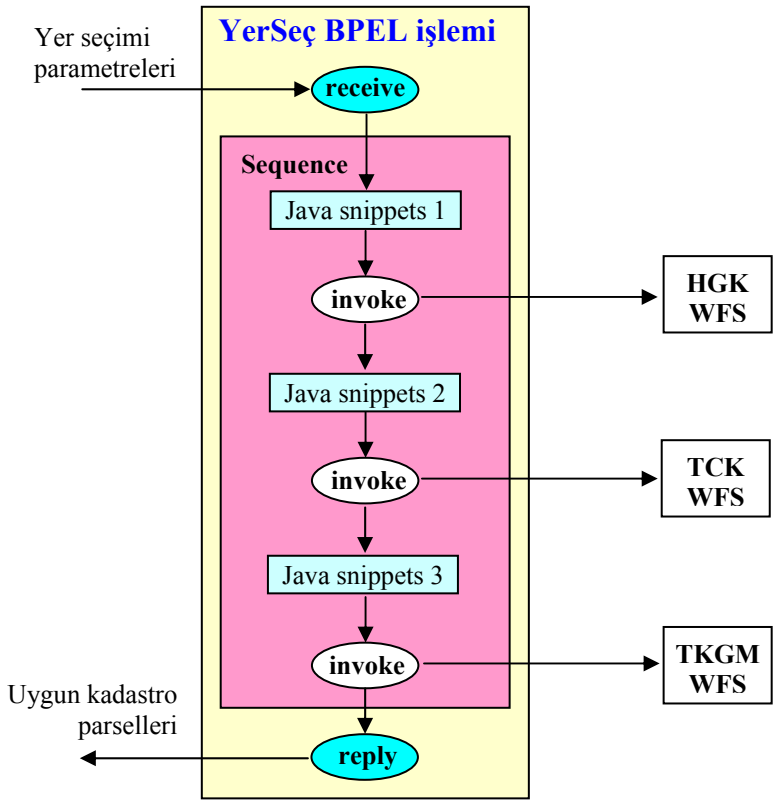
```
<wsdl:message name="GetFeatureRequest">
<wsdl:part name="REQUEST" type="xsd:string" wfs:value="GetFeature" />
<wsdl:part name="SERVICE" type="xsd:string" wfs:value="WFS" />
<wsdl:part name="VERSION" type="xsd:string" />
<wsdl:part name="TYPENAME" type="xsd:string" />
<wsdl:part name="FEATUREID" type="xsd:string" />
<wsdl:part name="PROPERTYNAME" type="xsd:string" />
<wsdl:part name="FEATUREVERSION" type="xsd:string" />
<wsdl:part name="MAXFEATURES" type="xsd:postiveInteger" />
<wsdl:part name="FILTER" type="xsd:string" />
<wsdl:part name="BBOX" type="xsd:string" />
</wsdl:message>
```

OGC Web servislerinin WSDL tanımlarını üretmek, bu servisleri bir BPEL işleminde kullanarak bir servis düzenleme uygulamasını gerçekleştirmek için tek başına yeterli değildir. Çünkü OGC Web servislerinden konumsal verileri elde etmek için kullanılan operasyonlar, parametre olarak bir sorgu ifadesi (filter) içermekte ve OGC Web servislerinin mevcut işleyişinde sorgu ifadeleri kullanıcılar tarafından yazılmaktadır. Örneğin aşağıdaki XML kodu, kadastro parsellerini sunan bir WFS servisinden, mülkiyeti Maliye Hazinesi'ne ait olan parselleri elde etmek için yazılan bir *GetFeature* istek mesajını ve içerdiği sorgu ifadesini göstermektedir. BPEL belirtimi (OASIS, 2006), aşağıdaki gibi bir sorgu ifadesinin bir BPEL işlemi içerisinde oluşturulmasına olanak sağlayan bir aktiviteye sahip değildir. Bu nedenle, OGC Web servisleri ile servis düzenlemeyi gerçekleştirebilmek için, sorgu ifadelerinin BPEL işlemleri içerisinde oluşturulmasına olanak sağlayan araçlara ihtiyaç vardır.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<GetFeature outputFormat="GML2" xmlns:gml="http://www.opengis.net/gml">
  <Query typeName="kadastro">
    <Filter>
      <PropertyIsEqualTo>
        <PropertyName>MALIK</PropertyName>
        <Literal>Maliye Hazinesi</Literal>
      </PropertyIsEqualTo>
    </Filter>
```

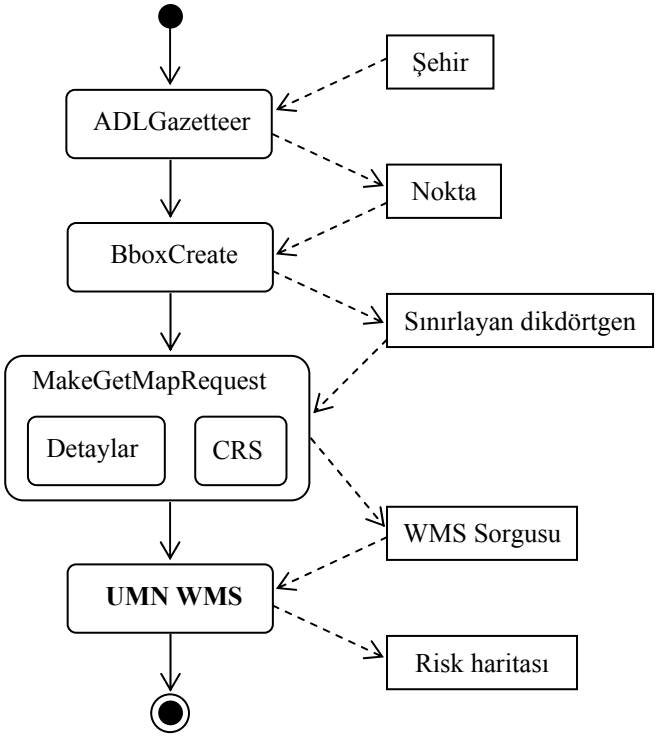
```
  </Query>
</GetFeature>
```

IBM ve BEA yazılım firmaları tarafından geliştirilen BPELJ (BPEL for Java) (Blow vd., 2004) dili, BPEL ve Java programlama dillerinin bir BPEL işleminde birlikte kullanılmasına olanak sağlamaktadır. BPELJ, "*java snippets*" olarak adlandırılan ve döngü koşulları, dallanma koşulları, değişkenlerin hazırlanması ve servis mesajlarının hazırlanması gibi işlemleri yerine getiren küçük java kodlarının, BPEL işlemleri içerisinde kullanılmasına olanak sağlamaktadır. Söz konusu java kodları, BPEL işlemleri içerisinde, OGC Web servislerine gönderilecek olan sorgu ifadelerini oluşturulmak için kullanılabilir. Bu nedenle, BPELJ kullanılarak OGC Web servisleri ile servis düzenleme uygulamaları geliştirilebilir. Örneğin bir yer seçimi uygulamasının, OGC Web servisleri kullanılarak BPELJ de nasıl gerçekleştirileceğini ele alalım. Bir kullanıcının, eğimi %10 dan az olan, mevcut yollara 1 km mesafede olan ve mülkiyeti Maliye Hazinesi'ne ait olan uygun yerleşim alanlarını belirlemek istediğini kabul edelim. Söz konusu uygulama için gereksinim duyulan eğim verisinin Harita Genel Komutanlığı (HGK), yol verisinin Karayolları (TCK) ve parsel verisinin Tapu ve Kadastro Genel Müdürlüğü (TKGM)'ne ait WFS servisleri tarafından sunulduğunu kabul edelim. Kullanıcının yer seçimi uygulamasını gerçekleştirebilmesi için, üç adet java kodu içeren bir BPEL işlemi (YerSeç) geliştirmesi gerekmektedir (Şekil 32). Birinci java kodu, kullanıcı tarafından düzenlemeyi gerçekleştiren BPEL işlemine gönderilen yer seçimi parametrelerini kullanarak, HGK WFS servisinden eğimi %10 dan az olan poligonları elde etmek için gerekli sorguyu oluşturmalıdır. İkinci java kodu, eğimi %10 dan az olan poligonların içerisinde kalan yolları elde etmek için, TCK WFS servisine gönderilecek olan sorguyu oluşturmalıdır. Üçüncü java kodu ise, TCK WFS servisi tarafından döndürülen ve eğimi %10 dan az olan poligonların içerisinde kalan yollara 1 km mesafedeki Hazine parsellerini elde etmek için TKGM WFS servisine gönderilecek sorguyu oluşturmalıdır.



Şekil 32. BPELJ ve OGC web servisleri ile servis düzenleme

OGC Web servisleri ile servis düzenlemeyi gerçekleştirmenin bir diğer yolu da, bir servis düzenleme uygulamasında, OGC Web servislerini düzenlemenin son servisi olarak kullanmaktır. Örneğin Lemmens vd. (2006), bir şehirdeki potansiyel tehlikeli alanları gösteren bir risk haritası üreten örnek bir servis düzenleme uygulaması geliştirmiştir. Geliştirilen servis düzenleme uygulamasında risk haritası, bir WMS servisi (UMN WMS) tarafından üretilmektedir. Risk haritasını üretmek için WMS servisine gönderilecek olan sorgu, *ADLGazetteer*, *BboxCreate* ve *MakeGetMapRequest* Web servislerini içeren bir servis düzenleme tarafından oluşturulmaktadır. Lemmens vd. (2006), söz konusu uygulamayı gerçekleştirmek için *GetRiskMap* adında yeni bir birleşik Web servisi geliştirmiştir. *GetRiskMap* Web servisi, kullanıcılardan parametre olarak bir şehir ismi almakta ve sırasıyla *ADLGazetteer*, *BboxCreate* ve *MakeGetMapRequest* Web servislerini çağırarak, WMS servisine gönderilecek olan sorguyu oluşturmaktadır (Şekil 33). *GetRiskMap* Web servisi, üretilen sorguyu UMN WMS servisine göndererek WMS servisinden risk haritasını alıp kullanıcılara göndermektedir.



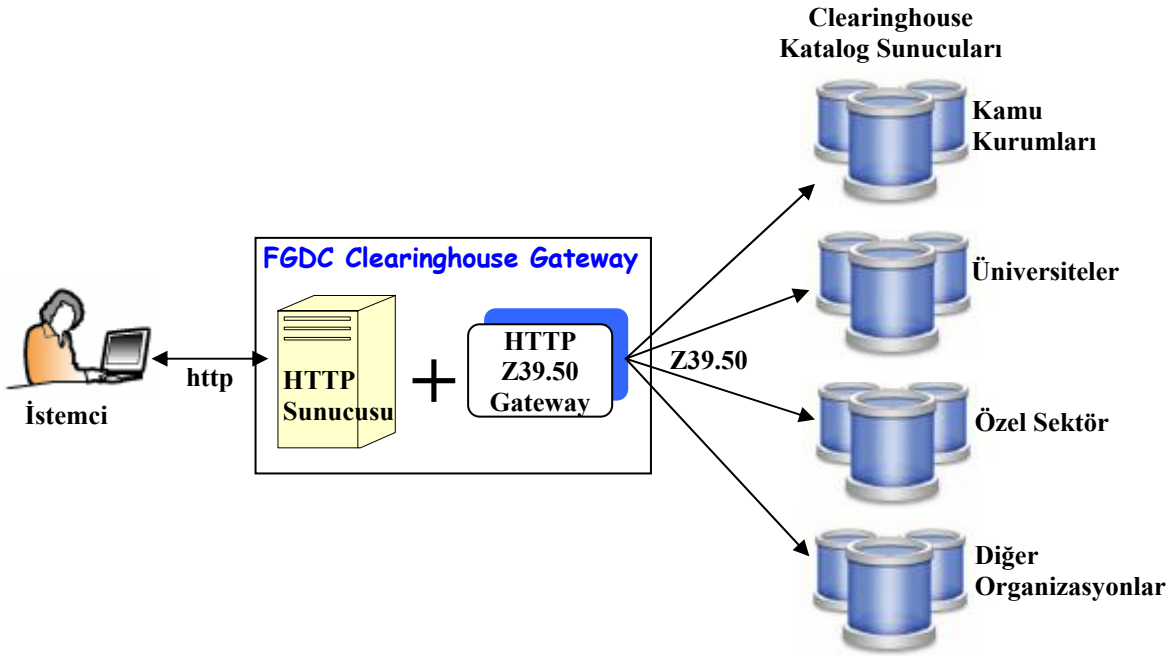
Şekil 33. GetRiskMap web servisinin UML aktivite diyagramı (Lemmens vd., 2006).

2.6. Geleneksel UKVA Gerçekleştirim Yöntemleri

2.6.1. FGDC UKVA Sunucusu (Clearinghouse)

Kamu kurumları, özellikle doğal afetlerle, endüstriyel kazalarla, çevresel sorunlarla veya ulusal güvenliği tehdit eden terörist saldırılarla karşılaştıkları zaman, kamunun can ve mal güvenliğini korumak amacıyla, hızlı ve doğru kararlar verebilmek için konumsal veriye ihtiyaç duyarlar. Konumsal verinin elde edilmesi, zaman ve maliyet açısından oldukça zahmetlidir. Amerikan hükümeti tarafından konumsal verinin üretimi, yönetimi ve dağıtımı için yılda 4 milyar dolardan fazla para harcandığı tahmin edilmektedir (FGDC, 2005a). Bu paranın büyük bir bölümü, özellikle konumsal verinin toplanması ve yönetimi için harcanmaktadır. Amerikan hükümeti, 1994 yılında, konumsal verinin tüm kullanıcılar tarafından paylaşımını sağlamak, veri toplama maliyetlerini düşürmek ve kurumların doğrudan hizmet üretimine geçmesini sağlamak amacıyla, Ulusal Konumsal Veri Altyapısı (UKVA) nın kurulmasına karar vermiş ve UKVA çalışmalarını koordine etmesi için FGDC (Federal Geographic Data Committee) komitesini kurmuştur. FGDC komitesinin

ana görevi, konumsal verinin toplanması, paylaşılması ve tanımlanması için standartlar geliştirmek ve konumsal veri üreticileri, yöneticileri ve kullanıcılarından oluşan UKVA ağını (Clearinghouse) kurmaktır (FGDC, 2005b). Clearinghouse, konumsal verilerin depolandığı merkezi bir veri tabanı veya verilerin adreslerini içeren bir Web sitesi değildir. Clearinghouse, ortak bir arayüz aracılığı ile sorgulanabilen konumsal veri katalogları federasyonudur (Şekil 34) (FGDC, 2005a).



Şekil 34. Clearinghouse mimarisi (Schweitzer, 2003).

Bilgi modeli: Bir Clearinghouse katalogu, FGDC CSDGM meta veri standardına uygun meta veriler içerir. Katalogda, sadece konumsal verilerle ilgili meta veriler yer alır. Katalog, Web servis tanımlarını içermez. Bir Clearinghouse katalogunun bilgi modeli, Z39.50 protokolünün FGDC tarafından geliştirilen "GEO" profili tarafından tanımlanır. GEO profili, istemcilerin bir Clearinghouse katalogunu sorgularken kullanmaları gereken zorunlu öznitelikleri ve bir Z39.50 GEO sunucusunun gerçekleştirmesi gereken konumsal operatörleri tanımlar. GEO profilde, 20 adet sorgu özniteliği (Tablo 7) ve 5 adet konumsal operatör (Overlaps, Fully Enclosed Within, Encloses, Fully Outside Of, Near) yer alır (FGDC, 2000).

Tablo 7. Meta veri sorgulamalarında kullanılması gereken zorunlu öznitelikler

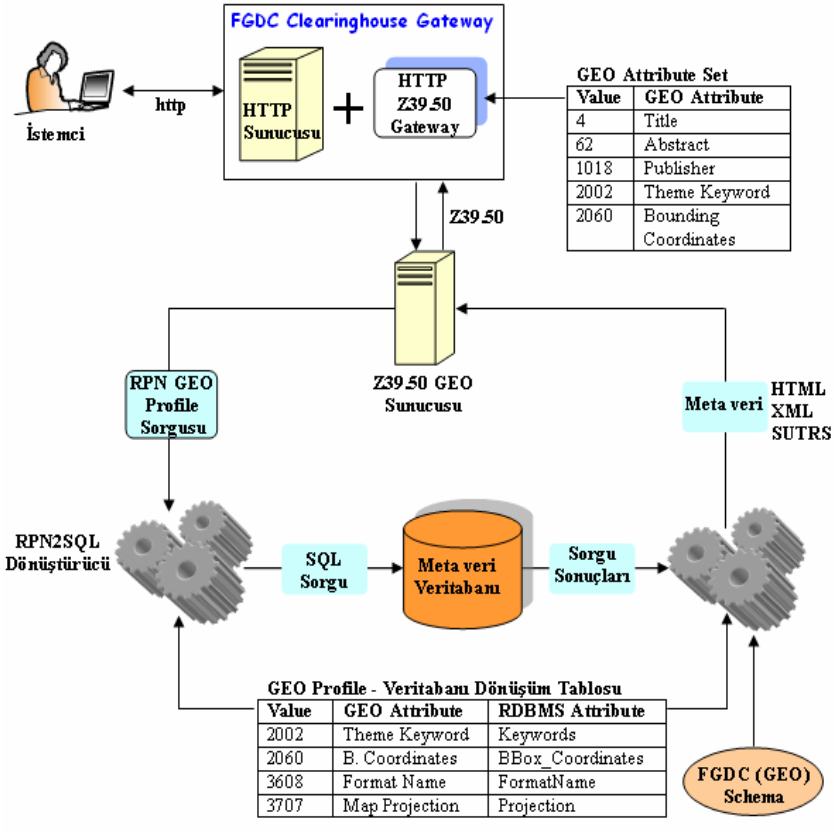
Öznitelik	**Veri tipi**	**Açıklama**
Title (4)	String	Veri setinin adı
Publication Date (31)	Date String	Veri setinin yayınlandığı tarih
Abstract (62)	String	Veri seti ile ilgili özet bilgi
Originator (1005)	String	Veri setini üreten organizasyon
Any (1016)	String	Katalogda string veri tipine sahip tüm alanlarda aranacak olan metin
Publisher (1018)	String	Veri setini yayınlayan organizasyon
Theme Keyword (2002)	String	Veri setinin içeriğini tanımlamak için kullanılan anahtar kelime
Purpose (2003)	String	Veri setinin ne amaçla üretildiğini açıklayan özet bilgi
West Bounding Coordinate (2038) (WBC)	Numeric String	Veri setinin enlem ve boylan cinsinden sınırlayan dikdörtgen koordinatları
East Bounding Coordinate (2039) (EBC)	Numeric String	
North Bounding Coordinate (2040) (NBC)	Numeric String	
South Bounding Coordinate (2041) (SBC)	Numeric String	
Bounding Coordinates (2060)	Coordinate String	
Time Period Information (2062)	Date String	Bir olayın tarih ve saati ile ilgili bilgi
Beginning Date (2072)	Date String	
Ending Date (2073)	Date String	
Progress (3108)	String	Veri setinin durumu (Complete, Planned, In work)
Extent (3148)	Numeric String	(NBC – SBC) * (EBC – WBC)
Geospatial Data Presentation Form (3805)	String	Konumsal verinin temsil ediliş formu
Calendar Date (3903)	Date String	Bir olayın tarihi ile ilgili bilgi

Sorgu modeli: Clearinghouse, katalogları sorgulamak ve meta verileri elde etmek için "ANSI/NISO Z39.50 arama ve bilgi edinme" protokolünü kullanır. Z39.50 protokolü, istemci/sunucu mimarisini gerçekleştiren sistemlerde, bir istemcinin, bir sunucunun veritabanında arama yapmasına ve veritabanı kayıtlarını elde etmesine olanak sağlayan standart bir protokoldür (NISO, 2003). Z39.50, ilgi alanından bağımsız bir protokoldür. Protokolün, bir ilgi alanına özel gerçekleştirimi, bir "uygulama profili" tarafından tanımlanır. FGDC, Clearinghouse kataloglarındaki konumsal meta verilerin Z39.50 protokolü kullanılarak sorgulanmasını ve elde edilmesini sağlamak amacıyla GEO profilini geliştirmiştir. GEO profil, Z39.50 protokolünü kullanan istemci ve sunucuların, "Z39.50 Type–1 RPN" (Reverse Polish Notation) sorgu formatını kullanmalarını ister. RPN temelde, karmaşık hesaplamaların parantezler kullanılmadan ifade edilmesine olanak sağlayan ve bilgisayarlar tarafından işlenmesi kolay ve hızlı olan bir formattır. Örneğin, "((1+2)*3)+4" şeklindeki bir hesaplama RPN'de "4321+*+" veya "12+3*4+" şeklinde ifade edilir. Bir Clearinghouse katalogunda, örneğin Trabzon Belediyesi tarafından 2004 yılından sonra yayınlanmış imar planı verilerini arayan bir istemci, kataloga "find @attr 1=4 @attr 2=3 İmar Planı AND @attr 1=31 @attr 2=5 2004 AND @attr 1=1018 @attr

2=3 Trabzon Belediyesi" şeklinde bir RPN sorgusu gönderir. RPN sorgusu, katalogda "SELECT * FROM Database WHERE Title=İmar Planı AND Publication Date>2004 AND Publisher=Trabzon Belediyesi" şeklinde bir SQL sorgusuna dönüştürülür.

Uygulama geliştirmek için konumsal veriye ihtiyaç duyan bir istemci, internetten Clearinghouse sorgu sayfasına bağlanır. Sorgu sayfası, Z39.50 GEO profili tarafından belirlenen standart sorgu özniteliklerini içeren bir HTML formudur. İstemci sorguyu iki şekilde gerçekleştirebilir. Eğer istemci sorguyu göndermesi gereken katalogu biliyorsa, sorgu formunu doldurur, formda yer alan Clearinghouse katalog listesinden sorgulamak istediği katalogu seçer ve sorguyu başlatır. İstemcinin HTML formuna girdiği bilgiler, Clearinghouse sunucusunda bulunan "HTTP-Z39.50 Gateway" yazılımı tarafından RPN formatına dönüştürülür ve ilgili kataloga gönderilir. Sorguyu alan Z39.50 GEO katalog sunucusu, bir yazılım (RPN2SQL wrapper) kullanarak sorguyu SQL formatına dönüştürür. Sunucu, meta veri veritabanına bağlanır ve SQL sorgusunu yürütür. Sorgu sonucunda elde edilen meta veriler, FGDC meta veri standardına uygun şekilde HTML, XML veya SUTRS (Simple Unstructured Text Record Syntax) formatında kodlanarak istemciye gönderilir. İstemci, eğer sorguyu hangi kataloga göndermesi gerektiğini bilmiyorsa, sorgu tüm Clearinghouse kataloglarına gönderilir. Kataloglardan gelen yanıtlar, Clearinghouse da bir araya getirilir ve istemciye gönderilir. Şekil 35, Clearinghouse'un meta veri sorgulama modelini göstermektedir.

Bir Clearinghouse katalogunun bu şekilde belirli öznitelikler kullanılarak sorgulanması, bir OGC CSW katalog servisinin temel sorgulanabilir öznitelikler kullanılarak sorgulanmasına benzemektedir. Ancak Clearinghouse'un, OGC CSW de olduğu gibi kullanıcıların içerik tabanlı veya anlık sorgulamalar yapmasına olanak sağlayan bir filtreleme dili desteği yoktur.



Şekil 35. Clearinghouse meta veri sorgulama modeli

Arayüz desteği: Z39.50 protokolü, bir sunucu veritabanındaki bilgilerin aramasını ve elde edilmesini sağlar. Protokolün, bir istemcinin bir sunucunun veritabanına yeni kayıtlar eklemeye veya mevcut kayıtları güncellemeye olanak sağlayan operasyonları yoktur. Bu nedenle bir konumsal veri sağlayıcı, Z39.50 protokolünü kullanarak meta verilerini mevcut bir Clearinghouse kataloguna yayınlayamaz. Bir Clearinghouse katalogu, Clearinghouse düğümü olarak adlandırılır. Konumsal veri sağlayıcıların, sahip oldukları verileri tanımlayan FGDC meta verilerini Clearinghouse aracılığı ile yayınlayabilmeleri için, Clearinghouse ağına yeni bir düğüm olarak eklenmeleri gerekmektedir. Sağlayıcının, bir Clearinghouse düğümü olabilmesi için FGDC standardına uygun meta verilerini içeren bir veritabanına ve bu veritabanını sorgulayan ve GEO profilini gerçekleştiren bir Z39.50 sunucu yazılımına sahip olması gerekmektedir. Sağlayıcı, sunucunun adresini ve veritabanının adını Clearinghouse'a kayıt ettiği zaman, meta verisini yayınlamış olur. Clearinghouse sorgu modelini kullanarak meta veri arayan bir istemci, sorgunun tüm kataloglara gönderilmesi durumunda yeni eklenen Clearinghouse katalogunu da sorgulamış olur.

Federasyon desteği: Clearinghouse, konumsal verilerin depolandığı merkezi bir veri tabanı veya verilerin adreslerini içeren bir Web sitesi değildir. Clearinghouse, ortak bir arayüz aracılığı ile sorgulanabilen konumsal veri katalogları federasyonudur (FGDC, 2005a). Ancak Clearinghouse, bir katalog federasyonunun en önemli bileşeni olan ve bir sorgusunun, sadece sorguyu cevaplayabilecek ilgili kataloglara gönderilmesini sağlayan bir sorgu yönlendirme (Query Routing) mekanizmasına sahip değildir. Bu nedenle eğer istemci sorguyu hangi kataloga göndermesi gerektiğini bilmiyorsa, sorgu tüm Clearinghouse kataloglarına gönderilmektedir.

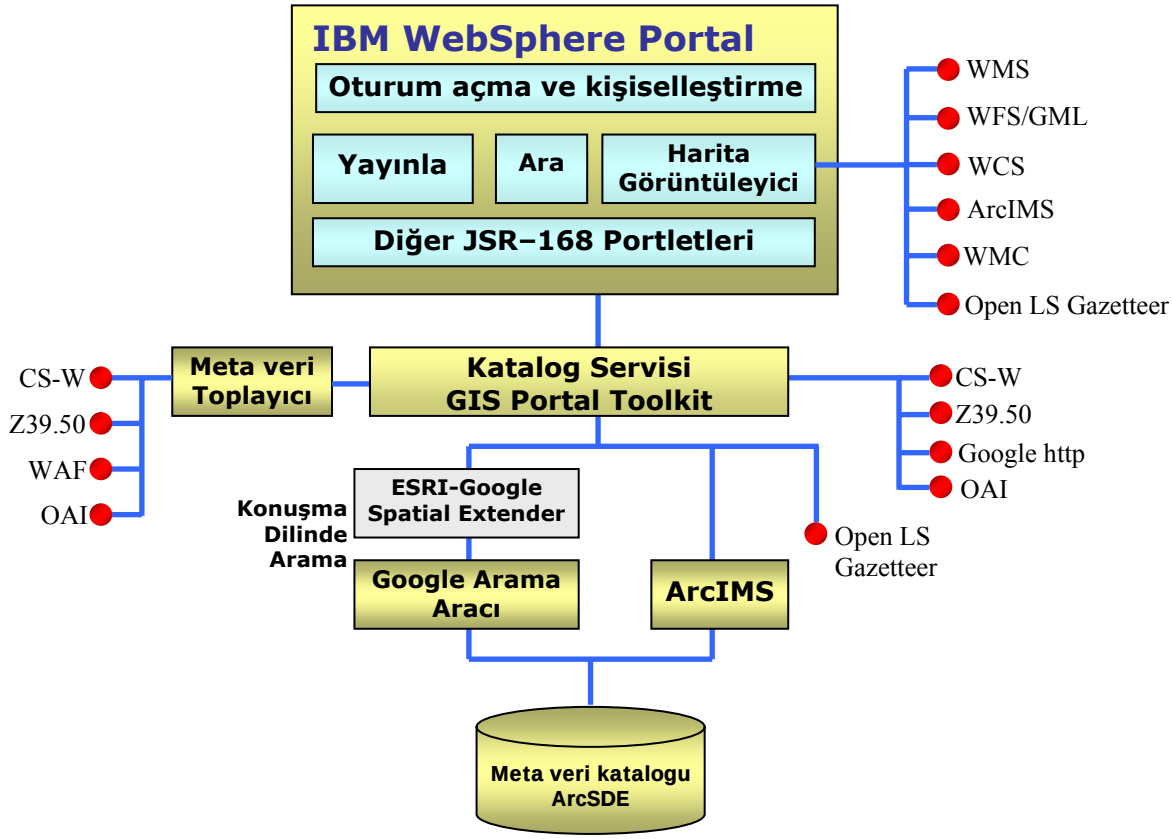
2.6.2. GOS Konumsal Portal Gerçekleştirimi

GOS (Geospatial One-Stop), Başkan George Bush'un onayıyla 2002 yılında başlatılan ve 24 farklı girişimden oluşan e-devlet programının bileşenlerinden biridir. GOS girişimi, özellikle Clearinghouse'un hayata geçmesinden sonraki sekiz yıllık süreçte, bilgi teknolojilerinde yaşanan gelişmeler ve özellikle Web servislerinin ortaya çıkması nedeniyle hem UVKA'nın teknoloji altyapısını yenilemek hem de tüm kamu kurumlarının, özel sektörün ve vatandaşların konumsal veriye hızlı, kolay ve daha ucuz bir şekilde erişimini sağlamak amacıyla başlatılmıştır. GOS'un temel görevi, konumsal veri ve servislere erişim sağlayan Web tabanlı bir "portal" kurmaktır (FGDC, 2005c).

Literatürde ve Web de yapılan aramalarda, yaygın kabul görmüş bir portal tanımına rastlanamamıştır. SUN yazılım firmasına göre portal, genellikle kişiselleştirmeyi, tek bir oturum açmayı, farklı kaynaklardan elde edilen içerikleri birleştirmeyi sağlayan ve bilgi sistemlerinin sunum katmanlarına ev sahipliği yapan Web tabanlı bir uygulamadır (SUN, 2003). ESRI ye göre portal, konumsal veriler için tek bir erişim noktasıdır (ESRI, 2004). OGC ye göre, birçok kaynak için tek bir erişim noktası sağlayan bir giriş kapısı gibi hareket eden bir Web sitesidir (OGC, 2004a). Yine OGC ye göre, GOS bağlamında, bir konumsal veri topluluğu, daha doğrusu veriyi sağlayan servisler topluluğu için çevrimiçi erişim noktasıdır (OGC, 2003a).

Konumsal portal belirtimi, 2003 yılında OGC tarafından yayınlanmıştır (OGC, 2003a). Aynı tarihte ESRI firması, prototip bir portal geliştirme ihalesini kazanmış ve Haziran 2003 de işlevsel ilk GOS portalı (www.geodata.gov) hizmete sunulmuştur. İlk GOS portal gerçekleştirimi, kamu kurumlarının sahip oldukları konumsal verileri tanımlayan meta verilerini yayınladıkları bir Clearinghouse katalogu gibi görev yapmıştır.

2004 yılının sonlarında GOS portalını genişletmek için yeni bir ihale düzenlenmiş, ihaleyi yine ESRI firması kazanmış ve 2005 yılında ESRI firması ile 2.38 milyon dolarlık beş yıllık bir kontrat yapılmıştır. İkinci nesil portal gerçekleştiriminden beklenenler, birlikte işlerlik ve açıklık, kullanım kolaylığı, hızlı arama, genişletilmiş performans ve genişletilmiş fonksiyonellik olmuştur (Ball, 2005).



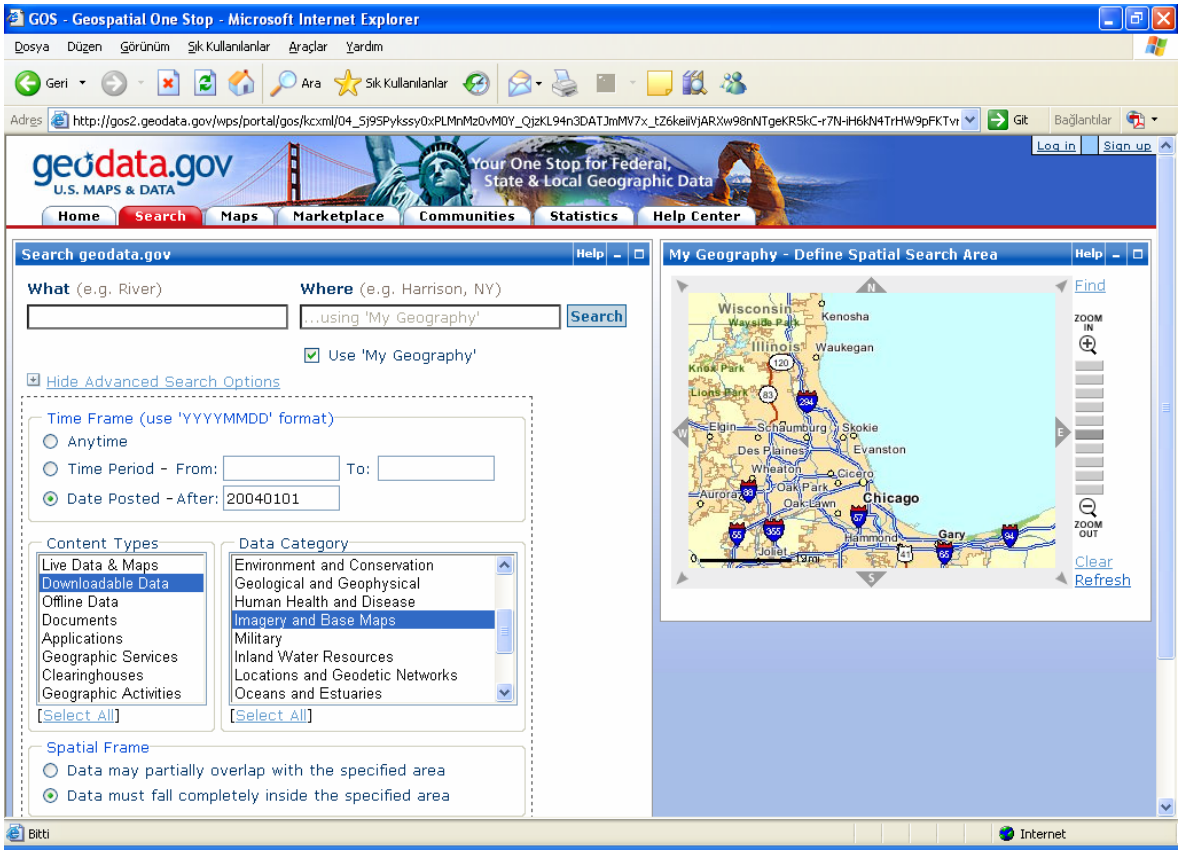
Şekil 36. GOS portal mimarisi (Ball, 2005).

İkinci nesil portal gerçekleştirimi, ilk GOS portalını bir meta veri katalogundan etkileşimli bir portala dönüştürmüştür. Portalın ikinci sürümü, merkezi bir meta veri kataloguna sahiptir ve ArcIMS ve ArcSDE teknolojileri üzerine kuruludur (Şekil 36). ArcIMS, portal mimarisinin en önemli bileşenidir ve meta veri yönetim, haritalandırma ve veri yükleme (download) servislerini sağlamaktadır. ArcSDE, sağlayıcılar tarafından yayınlanan ve bir ilişkisel veri tabanında (Oracle 9.2.0.5, Microsoft SQL Server 2000, IBM 8.2) saklanan meta verilere erişmek için kullanılmaktadır (ESRI, 2003). GOS portalında birlikte işlerliği sağlamak için OGC Web servisleri kullanılmaktadır. SAFE firması

tarafından geliştirilen *SpatialDirect* yazılımı, kullanıcıların veri dosyalarını 20 değişik formatta indirebilmelerine olanak sağlamaktadır. Google tarafından geliştirilen standart "Google Arama Aracı" (Google Search Appliance), kullanıcıların portalda daha hızlı arama yapmalarına olanak sağlamaktadır.

Bilgi modeli: GOS Portal, katalog bilgi modeli olarak FGDC meta veri standardını gerçekleştirir. Ancak FGDC'nin gelecekte meta veri standardı olarak ISO 19115 meta veri standardını kullanacağı ve bu değişimin gerçekleşmesi durumunda, portalında konumsal verileri tanımlamak için ISO 19115 meta veri standardına geçeceği bildirilmektedir (OGC, 2003a).

Sorgu modeli: GOS Portalın, kullanıcıların hem içerik tabanlı hem de parametrik sorgular yapmasına olanak sağlayan bir sorgu modeli vardır. Parametrik sorgulamada kullanıcılar, belirli meta veri öznitelikleri kullanarak portal katalogunda arama yaparlar. Kullanıcıların katalogu sorgulamak için kullandıkları öznitelikler, Z39.50 GEO profili tarafından tanımlanan sorgu özniteliklerinden daha az sayıdadır ve sadece belirli bir coğrafik alan içerisinde kalan, belirli bir tarihte üretilmiş, belirli bir veri kategorisindeki konumsal verilerin aramasına olanak sağlarlar (Şekil 37). Portalın parametrik sorgu modeli, bir OGC CSW katalog servisinin, temel sorgulanabilir öznitelikler kullanılarak sorgulanmasına benzemektedir.



Şekil 37. GOS portalda parametrik sorgulama

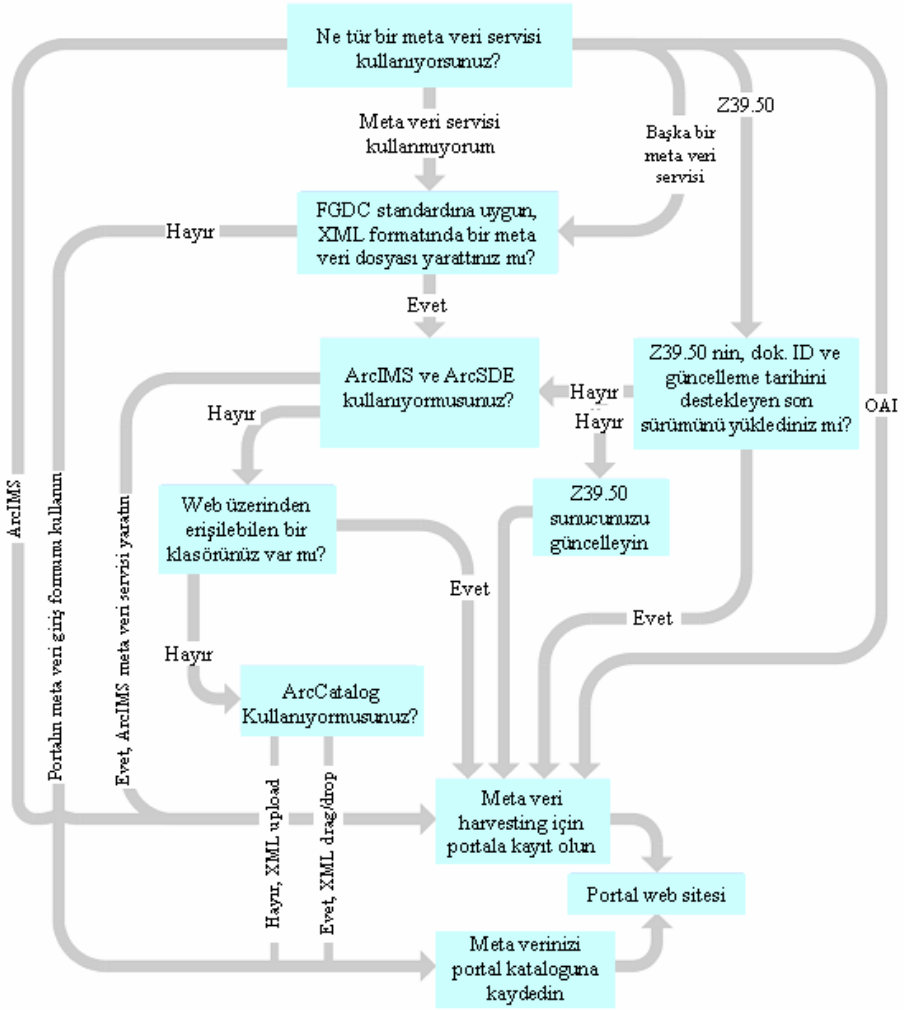
GOS Portalın, OGC CSW de olduğu gibi kullanıcıların içerik tabanlı veya anlık sorgular yapmasına olanak sağlayan bir filtreleme dili desteği yoktur. Bununla birlikte, Google'dan alınan ve ESRI tarafından genişletilerek portal mimarisine monte edilen standart Google Arama Aracı, kullanıcıların Google da bir Web sayfası arar gibi portalda konumsal veri aramalarına olanak sağlar. Örneğin "Chicago'ya ait 30 cm piksel çözünürlüklü, gerçek renkli, sayısal ortofoto görüntüsü" arayan bir kullanıcı, portalın standart arama kutusuna, aramak istediği veriyi tanımlayan bilgileri yazarak kolay bir şekilde aramasını gerçekleştirebilir (Şekil 38). Google Arama Aracı, kullanıcıların içerik tabanlı sorgu yapmasını sağlar.

Şekil 38. GOS portalda içerik tabanlı meta veri sorgulama

Standart Google arama aracının işlevselliği, ESRI tarafından geliştirilen ilave bir yazılım (Spatial Extender) tarafından genişletilmiştir. Genişletilen arama aracı, bir Web servisi (ArcWeb Services Gazetteer) kullanarak yer isimlerine göre konumsal arama yapmaya olanak sağlar (Schutzberg, 2005). Örneğin, Chicago'ya ait yolları arayan bir kullanıcının arama kutusuna "roads" ve "Chicago" kelimelerini yazması yeterlidir. Sistem, ilk olarak Gazetteer servisini kullanarak Chicago'nun sınırlayan dikdörtgen koordinatlarını bulmakta ve katalogda bu koordinatlar içerisinde kalan yolları aramaktadır. Google arama aracının, katalog aramalarını 10 kat daha hızlandırdığı belirtilmektedir (Ball, 2005).

Arayüz desteği: GOS Portal, SYM'nin yayınla ve bul olarak adlandırılan iki temel fonksiyonunu gerçekleştirir. Bir konumsal veri sağlayıcısının, meta verisini portalın kataloguna yayınlamak için kullanabileceği üç yöntem vardır (URL-1). Bu yöntemlerden ilk ikisi, meta verinin sağlayıcı tarafından portal kataloguna yayınlanmasına olanak tanır. Üçüncü yöntem ise meta verinin, sağlayıcı katalogundan portal tarafından otomatik olarak toplanmasını (metadata harvesting) ve portal kataloguna yüklenmesini sağlar (Şekil 39). Bir sağlayıcının, bu yöntemlerden hangisini kullanması gerektiğini, sahip olduğu veri seti sayısı belirler (URL-2). Birinci yöntemde sağlayıcı, GOS portalından kendi Web tarayıcısına indirdiği bir HTML meta veri giriş formuna, bir konumsal veri setini tanımlayan ve FGDC standardına uyan meta veri bilgilerini elle girip formu onaylar. Portal, form bilgilerini okuyarak meta verileri kataloga kaydeder. Bu yöntem, özellikle veri seti sayısı beşten az olan sağlayıcılar için önerilmektedir. Sahip olduğu veri seti sayısı yirmiden az olan sağlayıcılar için önerilen ikinci yöntemde, sağlayıcı, örneğin ArcCatalog gibi bir meta veri editörü kullanarak, sahip olduğu her bir veri seti için FGDC meta veri standardına uygun, XML formatında bir meta veri dosyası üretir. Sağlayıcı, daha sonra bu XML dosyalarını portal üzerindeki bir HTML formunu kullanarak sırasıyla portala yükler. Portal, bir parser aracılığıyla XML dosyalarını okur ve meta verileri kataloguna kaydeder. Veri seti sayısı yirmiden fazla olan, ArcIMS meta veri servisi, Z39.50 GEO meta veri

servisi, OAI (Open Archive Initiative) meta veri servisi veya OGC CSW gibi bir katalog servisine veya XML formatındaki meta veri dosyalarını içeren Web üzerinden erişilebilen bir klasöre (Web Accessible Folder) sahip olan sağlayıcılar için önerilen üçüncü yöntemde, bir sağlayıcı, katalog sunucusunun veya Web klasörünün internet adresini portala kayıt eder. Portal, sağlayıcının katalog sunucusuna veya Web klasörüne ilk kez bağlandığında tüm meta verileri alır ve kendi kataloguna kaydeder. Portal, sağlayıcı tarafından belirtilen zaman aralıklarında tekrardan sağlayıcının katalog sunucusuna veya Web klasörüne bağlandığı zaman, güncellenen veya yeni kayıt edilen meta veriler olup olmadığını kontrol eder. Güncellenen veya yeni kayıt edilen meta veriler varsa sadece bu meta verileri alır ve portal kataloguna kaydeder.



Şekil 39. GOS portalda meta veri yayınlama seçenekleri (ESRI, 2004).

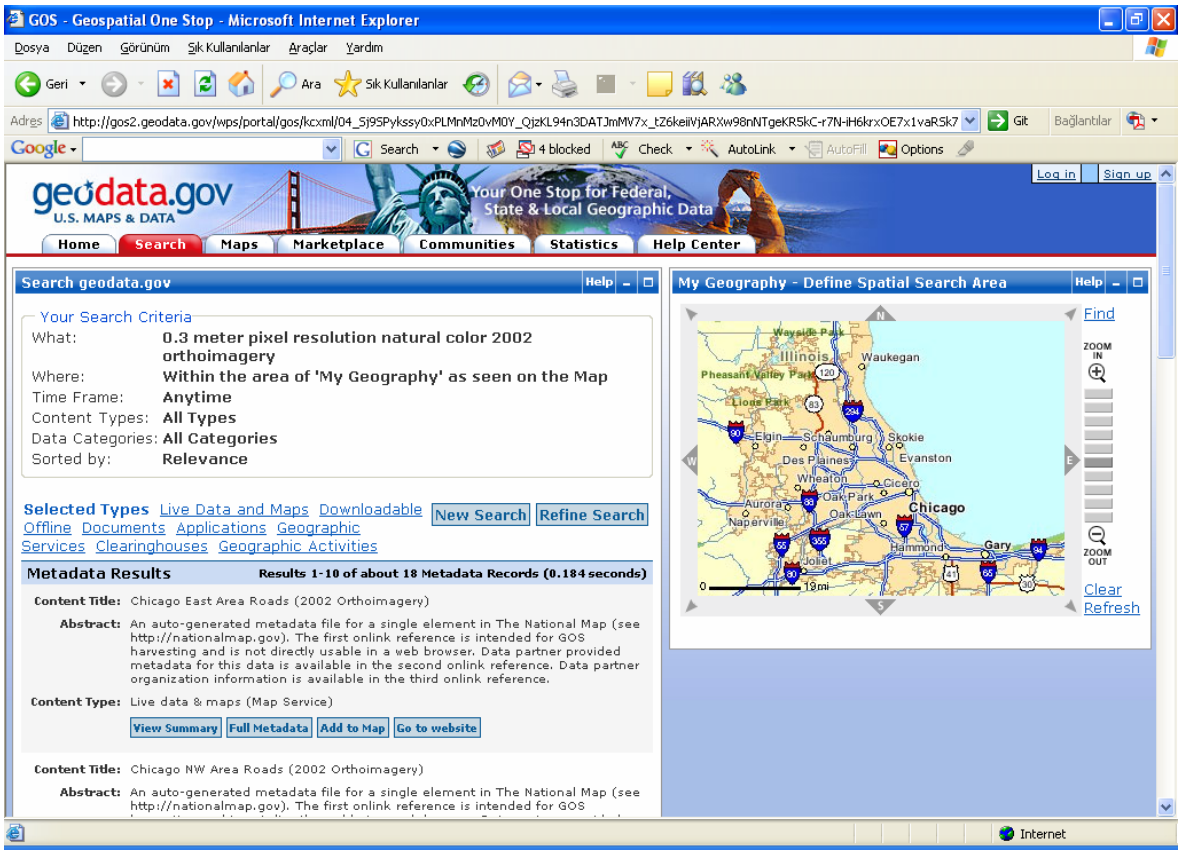
GOS portal ayrıca, belirli bir aktivitenin gerçekleşmesi durumunda, bu aktivite ile ilgilenen kullanıcıları haberdar eden bir uyarı (alert) servisine sahiptir. Örneğin, Chicago kent rehberini hazırlayan bir kullanıcı, uyarı servisine abone olarak portala, Chicago kent merkezi ile ilgili yeni verilerin eklenmesi durumunda, servisin kendisini haberdar etmesini isteyebilir. Servis, uyarı içeriğini, kullanıcının kişisel portal sayfasında gösterebilir veya kullanıcıya e-posta ile gönderebilir.

Federasyon desteği: GOS portal, tek bir merkezi meta veri kataloguna sahip olduğu için burada bir katalog federasyonu söz konusu değildir. Dolayısıyla tüm sorgulamalar, merkezi katalogda gerçekleşmektedir. Avrupa Konumsal Veri Altyapısı'nın kurulması için geliştirilen Avrupa Konumsal Veri Portalı (European Geoportal), GOS portal ile aynı mimariye sahiptir. Avrupa portalında da merkezi bir katalog servisinin kullanılmasına gerekçe olarak, dağıtık katalog sorgulamaları için standart olarak kabul edilmiş bir belirtimin henüz geliştirilememesi ve dağıtık aramayı destekleyen katalog servislerinin gerçekleştirimlerinin zor olması gösterilmektedir (Bernard vd., 2005).

2.6.2.1. GOS Portalda Uygulama Geliştirme

Konumsal Portal, "insan-odaklı Web" (Cerami, 2002) nitelemesine uyan bir Web teknolojisidir. Portal teknolojisi kullanılarak gerçekleştirilen bir KVA da, uygulama geliştirmek isteyen bir kullanıcı, portal katalogunda arama yaparak uygulama gereksinimlerini karşılayan konumsal verileri bulmalı, bu verileri bir servis aracılığıyla sağlayıcılardan alarak kendi sistemindeki bir uygulama programına aktarmalı ve uygulamayı kendi sisteminde gerçekleştirmelidir. Ancak portalda arama yaparak, bir uygulama için ihtiyaç duyulan konumsal verileri bulmak oldukça zahmetli ve zor bir işlemdir. Kullanıcının aradığı bir veri için farklı sağlayıcılar tarafından sunulan birden fazla veri bulması durumunda, ilgili verileri tanımlayan meta veri dokümanlarını inceleyerek uygulamasında hangi veri setini kullanacağına karar vermesi gerekmektedir. Örneğin GOS Portal da, içerik tabanlı sorgu yöntemini kullanarak, Chicago kent merkezini kapsayan bir alan içerisinde, "30 cm piksel çözünürlüklü, gerçek renkli ve 2002 yılından sonra üretilmiş ortofoto görüntüsü" arayan bir kullanıcı, farklı sağlayıcılar tarafından sunulan 18 adet veri seti bulmaktadır (Şekil 40). Kullanıcının, uygulamasında hangi veri setini kullanacağını belirleyebilmesi için sırasıyla tüm veri setlerini tanımlayan meta veri dokümanlarını incelemesi gerekmektedir. Bu işlem uzun zaman alır ve kullanıcı acısından

oldukça bunaltıcıdır. Ayrıca GOS portal bilgi modeli olarak FGDC meta veri standardını gerçekleştirdiği için, sorgu sonucunda kullanıcıya gönderilen meta veri kayıtları FGDC formatındadır. Bu nedenle, kullanıcının meta veri kayıtlarını inceleyebilmesi için FGDC meta veri sınıfları ve öznitelikleri hakkında bilgi sahibi olması gerekmektedir.



Şekil 40. GOS portalda içerik tabanlı bir arama sonucunda elde edilen meta veri kayıtları

GOS Portal da bir yer seçimi uygulamasının nasıl gerçekleştirileceğini ele alalım. Örneğin bir kullanıcının, belirli bir bölgede eğimi %10 dan az olan, mevcut yollara 1 km mesafede olan ve mülkiyeti Maliye Hazinesi'ne ait olan uygun yerleşim alanlarını belirlemek istediğini kabul edelim. Kullanıcı, eğim verisini bulmak için parametrik ya da içerik tabanlı sorgu yöntemlerinden birini kullanarak portalda arama gerçekleştirir. Kullanıcının, eğim verisi için farklı sağlayıcılar tarafından sunulan birden fazla veri seti bulması durumunda, söz konusu veri setlerini tanımlayan meta veri dokümanlarını inceleyerek uygulamada hangi sağlayıcı tarafından sunulan verileri kullanacağına karar vermesi gerekmektedir. Portal mimarisinde, konumsal veri sağlayıcıları sahip oldukları

konumsal verileri OGC Web servislerini (WFS, WMS ve WCS) kullanarak sunarlar. Kullanıcı, eğim verisini sunan sağlayıcıyı belirledikten sonra, portal tarafından sunulan standart harita görüntüleme aracını (Map Viewer Tool) kullanarak ilgili veriyi sunan OGC Web servisine bağlanır ve veri setinin tamamını harita görüntüleme aracında görüntüler. Kullanıcı, harita görüntüleme aracı tarafından sunulan seçim fonksiyonlarını kullanarak istediği bölge içerisinde kalan eğim verilerini seçer ve bu verileri portal tarafından desteklenen formatlardan birinde kendi bilgisayarına indirir. Kullanıcı, yol ve kadastro verileri içinde benzer işlemleri gerçekleştirir ve ilgili verileri kendi bilgisayarına indirir. Kullanıcı daha sonra, bir CBS yazılımı kullanarak yer seçimi uygulamasını kendi sisteminde gerçekleştirir.

3. BULGULAR

Bu tez çalışmasının çıkış noktası, UKVA'nın teknik gerçekleştirim stratejisi ve mimarisinin belirlenmesi olmuştur. Diğer bir anlatımla, UKVA'nın teknik birlikte işlerlik altyapısının hangi teknoloji, standart ve araçlar üzerine kurulacağının belirlenmesi hedeflenmiştir. Burada temel ilke ise, UKVA'nın kurulması stratejisi bakımından benimsenmiş olan, "UKVA'nın mevcut en son teknoloji ile kurulması" (Cömert ve Akıncı, 2005) olmuştur. Daha önce de belirtildiği gibi, herhangi bir birlikte işlerlik altyapısının gerçekleştirimi için bugün, en son teknoloji olarak yaygın kabul görmüş olarak önerilen, Servis Yönelimli Mimari (SYM) ve onun mevcut en popüler gerçekleştirim şekli olan Web Servisleri (WS) dir. Bu ilkenin benimsenmesi, söz konusu görevi, kapsamı belirli bir teknolojiye indirgemesi bakımından basitleştirebilir şeklinde algılanabilirse de gerçekte durum tam tersidir. Çünkü WS, konumsal veri toplumunun yıllardır özlemini duyduğu ve uğruna büyük çaba harcadığı birlikte işlerlik açısından, çok güzel bir gelecek vaat etmekle birlikte, özellikle belirli konularda üzerine kurulduğu teknolojilerin henüz olgunlaşma sürecini yaşadığı bir paradigmadır. Dolayısıyla, gerek kapsamın genişliği ve gerekse kullanılacak teknolojilerin henüz olgunlaşmamış olması, bu tezde çözülmeye çalışılan problemi oldukça zorlaştırmıştır.

Bu tezin amacı olan geniş ölçekli bir KVA'nın, SYM mimarisine dayalı, teknik birlikte işlerlik altyapısının nasıl gerçekleştirilebileceği konusunda somut bir çatı ve bir görüş birliği henüz söz konusu değildir. Bunun en önemli nedeni, yukarıda belirtildiği üzere WS'nin gerek konumsal ve gerekse konumsal olmayan arenada henüz olgunlaşma sürecini yaşayan çok yeni bir teknoloji olmasıdır. Bununla birlikte, WS teknolojik ve endüstriyel açıdan son birkaç yılın en aktif araştırma ve uygulama alanlarından biri durumundadır. KVA alanında ulusal ve uluslararası sayısız girişim, uluslararası düzeyde çok yoğun bir tempoda devam eden standart geliştirme çalışmaları, sayıları hızla artan bilimsel çalışmalar, konumsal Web servisleri ile yeni bir sektör ve trendi yaratan ticari ve açık kod yazılımlar bu alandaki büyük heyecanı yansıtmaktadır. Ancak, gelinen noktada SYM ya da WS yönelimli geniş ölçekli bir KVA'nın teknik birlikte işlerlik alt yapısının nasıl gerçekleştirileceği henüz tanımlanamamıştır. Bu durum, bu noktada WS yönelimli bir KVA gerçekleştirmek isteyenlerin işini, gerek teknik gerçekleştirimciler ve gerekse KVA'ları kurmak ve yaşatmaktan sorumlu karar vericiler bazında çok zor kılmaktadır. Söz

konusu gerçekleştirimcilerin çözmesi gereken başlıca sorunlardan bir kaçı şunlar olabilir: Gerçekleştirim mimarisi ne olacaktır? OGC ve W3C WS mimarileri birlikte işletilebilecek midir? Web servisi tanımlama, bulma ve servis düzenleme için hangi teknoloji, araç, standart ya da belirtimler kullanılmalıdır? Servis kataloglama ve katalog federasyonları nasıl gerçekleştirilecektir? Hangi ticari ve/veya açık kod yazılımların tercih edileceğine nasıl karar verilecektir? WS yönelimli KVA'ya geçişin ana amacı olan birlikte işlerlik acaba gerçekten sağlanabilecek midir yoksa sorun olmaya devam mı edecektir? Bu birlikte işlerlik "söz dizimsel" ya da "anlamsal" düzeyde mi sağlanmalıdır? Mevcut araçlar hangisini olanaklı kılmaktadır? Belirli kuşak araç ve teknolojilerle yapılan gerçekleştirimler arasında geçiş olanaklı olacak mıdır? Mevcut uygulamalar (legacy applications) SYM mimarisine adapte edilebilecek midir? WS alanında en önemli sorunlardan biri olarak kabul edilen güvenlik sorunu nasıl çözülecektir?

Konumsal ve konumsal olmayan Web Servisleri alanında Dünya genelindeki müthiş aktiviteye rağmen, WS yönelimli KVA gerçekleştiricilerinin anılan sorunlarının çözüm bulduğu bir gerçekleştirim modeli mevcut değildir. Gerçekleştirim modeli bir yana, anılan sorunların bütüncül bir yaklaşımla ele alınmasını sağlayacak bir çatı bile mevcut değildir. Öyle ki, WS alanında belirtim ve standart geliştirme ile ilgili çeşitli uluslararası kuruluşun (W3C, OASIS, WSMO, OGC) her birinin kendi çalışmaları vardır. Bunların hangilerinin yaygın kabul göreceği, farklı standartlar arasında nasıl uyum sağlanacağı henüz belirsizdir. Akademik çalışmaların, anlaşılabilenlerinin çoğu ya çok genel ya da çok özeldir. Tüm sorunları bütüncül bir yaklaşımla ele almakla birlikte, bazı temel sorunlar açısından henüz çözüm sunamayan (Fensel vd., 2005) gibi az sayıda çalışma vardır ki onlarda da bu bölümün ilerleyen kısımlarında belirtilecek olan bazı eksikler mevcuttur. Diğer yandan konumsal ve konumsal olmayan alanda faaliyet gösteren yazılım firmalarının Web servisleri geliştirme ve yayma ortamları ancak belirli düzeyde çözüm sunabilmekte, bazı temel sorunlara bir çözüm sunamamaktadırlar. Nihayet piyasada bölgesel, ulusal ve uluslararası ölçekte KVA gerçekleştirimleri mevcuttur.

Bütün bu belirsizlikler ve yukarıda bazı örnekleri verilen temel sorunlar ortada iken, gerçekleştirimciler nasıl bir yol izlemelidir? Hatta karar vericiler KVA gelişmelerini gerek kurulma ve gerekse yaşatma sürecinde nasıl yönlendirmeli ve kontrol etmeli, bu süreçte yukarıda anılan sorunlara nasıl çözüm üretmelidir? Bunun için kendilerine genel ve aynı zamanda bütüncül bir yaklaşımla anılan sorunları çözmeye olanak tanıyacak bir çatı sunulabilir mi?

Bu sorunun yanıtı, bu tezin ilgi odağını oluşturmuştur. Bu sorunun yanıtını bulmayı hedefleyen bu tez kapsamında yürütülen çalışmalar üç temel aşama olarak sınıflandırılabilir. İlk aşamada, dağıtık nesne modeli ortamlarının tanınması amacı ile CORBA ve RMI ortamlarında birlikte işlerlik uygulamaları geliştirilmiştir. Burada temel uygulama, Türkiye'de yıllardır çok önemli bir sorun olarak dile getirilen ve halen de çözümlenmemiş bulunan emlak vergilerinin doğru ve kayıpsız olarak hesaplanmasına olanak tanıyacak bir uygulama olmuştur (Cömert ve Akıncı, 2002).

Bu çalışmaların ardından başlayan ikinci aşamada, o tarihlerde bütün Dünya'da hızla popülaritesi artmaya başlayan WS teknolojisinin farkına varılarak tez çalışmaları bu alana doğru yönlendirilmiştir. İzleyen aşamada WS teknolojisine dayalı ve "e-belediye" gerçekleştirimine olanak tanıyacak uygulamalar geliştirilmiştir. Burada iki temel hedef vardı. Birincisi e-belediye modelinin, en son kuşak Web teknolojisi olan, WS teknolojisi ile gerçekleştirilebileceğini göstermekti. Bu amaca yönelik olarak ayrıca bir yüksek lisans çalışması (Şahin, 2003) yürütülmüş ve bu çalışma kapsamında konumsal veri ile en çok uğraşan belediye birimleri ve onların en temel uygulamaları incelenerek, bu uygulamalara yönelik Web servisleri tasarlanmıştır. Bu doktora tezi kapsamında ise Şahin (2003) tarafından tasarlanmış olan Web servislerinin bir kısmı Cape Clear (2003) Web servisleri geliştirme ve kurma yazılımı kullanılarak gerçekleştirilmiştir. Bu uygulamada ayrıca, istemci taraflı işlevleri "güçlü istemci" (thick-client) felsefesi doğrultusunda istemci tarafında geliştirebilmek için SVG (W3C, 2003b) kullanılmıştır.

WS tabanlı e-belediye uygulamasının gerçekleştirilmesinde diğer önemli hedef ise, gerek tasarım ve gerekse teknik gerçekleştirim mimarisi açısından, e-belediye gerçekleştiriminde uygulanan yaklaşımın, UKVA için de uygulanabileceğini göstermekti. Buradaki çıkış noktası, UKVA'nın da sonuçta e-belediye gibi bir "e-altyapı" olması algılaması olmuştur. Öyle ki, UKVA ve e-belediye modellerinin her ikisinde de, bileşen sistemler ve birlikte işlemesi gereken uygulamalar söz konusu idi. Aralarındaki tek fark kapsam ya da ölçektedir. E-belediye ya da UKVA uyumlu "belediye birlikte işlerlik altyapısı", ülke geneline hizmet verecek olan UKVA'nın bileşenlerinden yalnızca biri durumundadır.

Tez çalışmasının üçüncü ve nihai aşamasında ise çok geniş bir literatür, OGC, ISO, W3C ve OASIS gibi standart geliştirme kuruluşlarının belirtim ve standartları, yazılım firmalarının ticari ürünleri ve nihayet GOS gibi, ilgili market gerçekleştirimleri detaylı olarak irdelenmiştir. Böylece tezin hedefi olan KVA'ların SYM ile gerçekleştirilebilmeleri

için gerekli olan mevcut durumun analizi ve gelecek yönelimlerinin belirlenmesi gerçekleştirilmiştir. Bu aşamada, servis ya da veri tanımlama, yayınlama, bulma ve servis düzenleme işlemlerinin nasıl gerçekleştirileceği sorularına cevap aranmıştır. Tarama yapılan kapsamın çok geniş olması yanında, literatürün çoğunun temel felsefe ve kritik önem arz eden konularda, konuyu net bir şekilde ortaya koyamamaları, bu aşamadaki çalışmaları oldukça zorlaştırmıştır. Bu aşamada ayrıca, bu tez çalışması kapsamında gündeme getirilen bazı konularda ne gibi çözümleme ya da netleştirmeler getirilebileceği de araştırılmıştır. Bu değerlendirmeler tezin tek bir bölümünde olmayıp bir kısmı tez anlatımı içerisinde belirli konular altında, bir kısmı da bu bölümde sunulan ve bu tezde belirlenen ölçütler vasıtası ile verilmiştir.

Ölçütler kaynak tanımlama, kaynak yayınlama, kaynak yayınlama/bulma kapsama alanı belirleme, istek ve yayın eşleştirme ve uygulama geliştirme olarak belirlenmiştir.

Ölçüt isimleri, hem veri ve hem de servisleri kapsayan bir değerlendirmeye olanak tanıyacak tarzda bir isimlendirme ile belirlenmiştir. Bu durum her iki tür kaynağa yönelik olarak yapılmış çalışmaların bütüncül tek bir çatı altında değerlendirilebilmesi açısından son derece önemli ve katkı yapıcıdır.

Sınıflamayı oluşturan ölçütler SYM deki temel işlevlere yönelik olarak belirlenmiştir. Bu bakımdan ölçütlerin kendi içinde tutarlı olması amaçlanmıştır. Dolayısıyla sınıflandırma ana ölçütlerin her biri bazında ayrı ya da tek bir bütün olarak değerlendirmeye olanak tanımaktadır. Buna göre bir SYM gerçekleştirimi, yalnızca kaynak yayınlama bazında ya da diğer SYM işlevlerinin tümü açısından değerlendirilebilir.

Sınıflandırmada SYM yi karakterize eden temel işlemler arasındaki sınırlar yeterince keskin bir biçimde tanımlanmaya çalışılmıştır. Çünkü, bu tez çalışması kapsamında gerçekleştirilen geniş literatür taramasında, pek çok çalışmada gözlemlenen bir eksiklik olarak, belirli ayırımların yeterince netleştirilmemesi, çalışmanın katkısı bakımından çok olumsuz bir etki yapmaktadır. Bu konunun önemi, Sivashanmugam vd. (2004) tarafından da vurgulanmıştır.

Bulgular, genel bir düzeyde belirlenmiş olmaları nedeni ile sadece UKVA için değil, herhangi bir ölçekte KVA kurma ve yaşatma girişimine de yol gösterici niteliktedir.

Bu çatı ile herhangi bir akademik çalışma, girişim, proje, market gerçekleştirimi kolaylıkla değerlendirilebilmektedir. Bu gibi örnek değerlendirmeler, izleyen kısımda verilmektedir. Değerlendirmede akademik çalışma, girişim, proje, market gerçekleştiriminin hepsini kapsamak üzere “çalışma” (work) sözcüğü tercih edilmiştir.

Ayrıca yazım içerisinde “kaynak” sözcüğü veri ve servisleri her ikisinden daha genel bir düzeyde ifade etmek üzere kullanılmıştır.

Kaynak tanımlama: Bu ölçüt, çalışmanın temel olarak önerdiği ya da dayandığı servis ya da veri tanımlama ile ilgili özellikleri tanımlar. Burada iki alt ölçüt tanımlanmıştır. Biri tanımlanan bilgi kaynağı türüdür. Bu veri ya da servis olabilir. Veriden kasıt konumsal ve konumsal olmayan, çoklu ortam verisi dahil, her tür veridir. Servis ile kastedilen Web servisi yani internet üzerinden erişilebilen bir kod parçasıdır. Diğer alt ölçüt, tanımlamanın sözdizimsel mi ya da anlamsal mı olduğunu belirleyen “tanımlama temeli” dir. “Anlamsal” tanımlamadan kasıt, makine ya da yazılım kodunun bilgi çıkarsaması yapmasına olanak tanıyan bir tanımlamadır. “Sözdizimsel” tanımlamada ise bunun tersi bir durum söz konusudur. Yani sözdizimsel bir veri tanımı, makine ya da yazılım kodunun bilgi çıkarsaması yapmasına olanak tanımayan bir tanımlamadır. Dolayısıyla sözdizimsel tanımlamalara dayanan ortamlarda anlam çıkarsama ya da anlamın bir şekilde bilinmesi insan kullanıcıya kalmaktadır. Sözdizimsel veri ve servis tanımı için kullanılabilecek araç/standart/belirtim/teknolojilere popüler örnekler olarak geleneksel şema tanımlama, XML Schema ve WSDL verilebilir. Anlamsal veri ve servis tanımı için kullanılabilecek araç/standart/belirtim/teknolojilere popüler örnekler olarak ise WSDL-S, RDF, DAML, DAML-S, OWL ve OWL-S verilebilir. Bunun dışında çeşitli çalışmaların kendilerine özel olarak geliştirdikleri araç ya da belirtimler de söz konusudur. O nedenle bu ölçüt, yalnızca standart konumundaki belirtimleri değil, henüz standart olarak kabul edilmemiş belirtimleri de kapsamaktadır. Çalışmaların değerlendirilmesine yönelik olarak hazırlanmış ve ilerleyen kısımlarda sunulacak olan tablolarda, bu durumu netleştirici açıklamalar yer almaktadır. Örneğin METEOR-S (Sivashanmugam vd., 2004) çalışmasında, servis tanımı için W3C ya da OASIS standartları yerine, çalışma grubunun geliştirdiği MWSCF (Meteor Web Services Composition Framework) çerçevesinin bileşenlerinden biri olan ve anlamsal olarak genişletilmiş WSDL olarak algılanabilecek türden bir dil kullanmaktadır**.**

“Tanımlanan bilgi kaynağının türü” açısından arzu edilen, yalnızca servislerin değil fakat aynı zamanda verinin de tanımlanabilmesidir. Aslında SYM de tanımlanması gereken servislerdir ancak, Web Servisleri henüz istenen yaygınlık düzeyine ulaşamadığından, bazı sunucular yalnızca veri sunuyor olabilir. O nedenle herhangi bir gerçekleştirimde hem veri ve hem de servis tanımlamanın bir arada kullanılabileceği yaklaşım ve araçlar tercih edilmelidir.

Tanımlama temeli, kaynak tanımlamanın "sözdizimsel" ya da "anlamsal" bir temelden hangisine dayandığını belirtmek içindir. Kaynak tanımlamanın "sözdizimsel" ya da "anlamsal" olması, bir gerçekleştirimin, her şeyden önce, anlamsal Web ortamlarına uygunluğu açısından önem arz etmektedir. Öyle ki anlamsal olarak tanımlanmış servisler anlamsal Web ortamında doğrudan gerçekleştirilebilir; Aynen sözdizimsel bir tanımlamanın sözdizimsel Web ortamında doğudan gerçekleştirilebileceği gibi. Oysa sözdizimsel bir tanımlamanın anlamsal Web ortamında gerçekleştirilmesi için tanımlamanın anlamsal bir forma dönüştürülmesi gerekir.

Kaynak yayınlama: Kaynak yayınlama SYM nin en temel bileşenlerinden biri durumundadır. SYM'de servis sağlayıcılar Web servislerini bir katalog servisi üzerinden kullanıma sunar, istemciler de aradıkları özellikteki Web servislerini bu sayede tespit ederler. Bununla birlikte, P2P (Peer-to-Peer) (Aberer vd., 2005) tarzı gerçekleştirimlerde katalog ve sağlayıcı ayrımı bazen çok belirgin olmadığı için, katalog kavramı vurgusu ölçüt isimlendirmesi bazında değil de araçlar bazında yapılmıştır. Diğer bir anlatımla, SYM nin katalog boyutu, daha üst düzey bir algılamayı ifade eden "kaynak yayınlama" içerisinde düşünülmüştür.

Bu ölçüt altındaki ilk alt ölçüt "yayınlama referansı" dır. Bu ölçüt ile kastedilen, kaynağın katalog ya da reklam edildiği yer her neresi ise, orada hangi bilgilerle temsil edildiğidir. Kaynağı temsil eden, kaynağın belirli bir standart ya da belirtime göre tanımı olabilir. Bu tür tanımlar veri bağlamında "meta veri", servis bağlamında da "servis meta verisi" olarak anılır. Bu durumda, örneğin ebRIM ya da UDDI katalogunda, bu katalogların bilgi modelleri ya da bu katalogların konumsal veriye yönelik profillerine göre yapılacak olan servis tanımları yayınlanır. Bu durum, "tanım referanslı" yayınlama olarak adlandırılmıştır. Katalog içermeyen P2P gerçekleştirimlerinde, her bir eş, kendi bilgi kaynakları ile anlamsal olarak bir şekilde ilişkili, bilgi kaynaklarını içeren eşlere ait adresleri barındırır (Castano vd., 2003). Dolayısıyla "yayınlama" tabiri bu durumu da kapsamaktadır. Diğer yandan anlamsal ilişkiler kaynak tanımından türetilerek eşlere konduğu ve dolayısıyla arama zamanında sorgu eşleştirme tanım bazında yapılacağı için, "tanım referanslı" nitelemesi P2P gerçekleştirimlerini de kapsamaktadır.

Pratikte, özellikle veri türü kaynakların temsili için kullanılan, fakat servis türü kaynaklar için de uygulanabilecek, anahtar kelimeler ile endeksleme yaygındır. Anahtar kelimeler örneğin Google aramalarında kullanılan doküman içerisinde geçen sözcüklerin tümü ya da belirli bir ölçüte göre kaynağı temsil eden sözcükler olabilir. Bu tür yayınlama

“endeks referanslı” yayınlama olarak anılmıştır. Yani endeks referansından asıl kaynak olan dokümana ya da bilgi kaynağına ulaşılmaktadır. Örneğin Google, “endeks referanslı” bir yayınlama uygulamaktadır. Bunun bir sebebi, Google’in ilgi alanını oluşturan kaynağın metin dokümanı tarzı veriler olmasıdır. Buradan anlaşılacağı üzere, çatı yalnızca Web servislerini değil, veri kaynaklarına yönelik çalışmaların değerlendirilmesine ve anlaşılmasının basitleştirilmesine olanak tanımaktadır.

Her ne kadar SYM bir servis katalogu kullanmayı öngörse de, bugün mevcut WS gerçekleştirimlerinin çoğunda, daha önce de belirtildiği gibi, bir katalog bileşeni söz konusu değildir. Katalog boyutuna önem veren çalışmalar son birkaç yıldır artmaya başlamıştır. Bu bakımdan, herhangi bir SYM gerçekleştirimin kaynak yayınlama içerip içermediği, başlı başına bir ölçüt olarak değerlendirilecektir.

Yayınlama referansı açısından arzu edilen tanım referanslı yayınlamadır. Çünkü eğer bilgi kaynağı tanım referanslı olarak yayınlanırsa, sözdizimsel ya da anlamsal olarak gerçekleştirilebilecek bir tanım eşleştirme ile aranan tanımlamanın varlığına kesin olarak karar verilebilir. Oysa endeks referansı yardımı ile bilgi kaynaklarının bulunmasında, kaynağın varlığına endeksin kaynağı temsil edebilme derecesi ile orantılı bir kesinlik düzeyinde karar verilebilir. Endeks referanslı yayınlamada da yine aramanın doğruluğu açısından arzu edilen anlamsal endekslerdir.

Endeks referanslı yayınlama için endeks referansını detaylandıran alt ölçüt, “endeks temeli”dir. Endeks temeli, endeksin sözdizimsel mi yoksa anlamsal mı olduğunu belirtmek içindir. Sözdizimsel temelde istatistiksel doküman özetleme, doküman içinden anahtar sözcük vd. endeksi oluşturabilir. Anlamsal temelde ise, örneğin “semantic latent index” (Tang vd., 2003) gibi anlamsal endeksler kullanılabilir.

Tanım temeli ölçütü ile tanımlanmaya çalışılan yine kaynağın sözdizimsel mi yoksa anlamsal bir tanımlama ile mi yayınlandığının belirtilmesidir. Mevcut çalışmaların çoğunda kullanılan geleneksel katalog servislerinin kullanılması durumunda tanım temeli ya sözdizimsel ya da anlamsal zenginleştirilmiş sözdizimsel olabilir. Her iki durumda da tanımın anlamsal kullanımı söz konusu olmadığı için sözdizimsel nitelemesi kullanılmıştır. Bu tarz çalışmalar katalog servisleri kısmında detaylı olarak ele alındığından burada ayrıca bahsedilmeyecektir. Anlamsal tanımlama, Tanımlama Mantığı (TM / DL (Description Logics)) temelli OWL-S ve DAML-S gibi yukarıda kaynak tanımlama kısmında belirtilen diller ile yapılabilir. Anlamsal tanımlama durumunda servisin yayınlanması bir bilgi tabanı vasıtası ile olabilir.

Bu ölçüt kapsamındaki araç/standart/belirtim/teknolojiler, ebXML katalog servisi, UDDI katalog servisi, OGC CSW katalog servisi ve ISO 19115/19119 veri ve servis meta veri tanımlama standartları ve bilgi tabanları olabilir. Bunlardan bilgi tabanları dışındakiler katalog servisleri kısmında anlatılmıştır. Yine aynı kısımda bu araçların nasıl değerlendirilebileceği de belirlenmişti.

Bilgi tabanları, tezin kapsamı gereği bilinmesi gerekli olan, fakat detaylı analizleri bu tez kapsamını aşan bir konudur. O nedenle bu konuda, bu tez çalışmasını yürütmeye yetecek düzeyde bir çalışma yapılmıştır. Buna göre kısaca özetlemek gerekirse, TM de tanımlanan veri ya da servisler, OWL ya da OWL-S dilinde ifade edilir. Bir TM bilgi tabanı "TBox" ve "ABox" olarak adlandırılan iki temel bileşen içerir (Baader vd., 2003). "Harun Efe erkektir", "Erkekler Oje Sürmez", "O halde Harun Efe de Oje Sürmez". Bunlardan Erkek tanımı ve bu tanım kapsamındaki "Erkekler Oje Sürmez" iddiası TBox ta bulunması gereken kavramlardır. "Harun Efe erkektir" ise kavram örneği (individual/instance) dir ve Abox ta yer alır. Tbox ve Abox ta bulunan bilgiler bazen sırasıyla *intensional* ve *extensional* bilgi olarak ta anılır (Şirin vd. 2005). "O halde Harun Efe de Oje Sürmez" ise mantıksal bir program kodunun bu Tbox ve Abox bileşenlerinde bulunan bilgileri kullanarak yapabileceği bir çıkarsamadır. Bir bilgi temsilini "anlamsal" yapan boyut ta bu nokta, yani çıkarsamanın program ya da çoğunlukla tabir edildiği şekliyle "makine" tarafından yapılabilmesidir.

Kaynak yayınlama/bulma kapsama alanı belirleme: Bu ölçüt altında kaynak yayınlama ve kaynak bulmanın kapsama alanının tanımlanabilmesi amaçlanmıştır. Ölçütün yayınlama bazında da düşünülmesi, yayınlanacak bir kaynağın hangi kapsama alanında yayınlanması gerektiğinin belirlenmesi içinde kapsama alanı belirlemeye ihtiyaç olmasındandır. Bir başka anlatımla, eğer yayın tüm sağlayıcılara yönelik olarak yapılmıyorsa, hangi sağlayıcılara yönelik olarak yapılacağı nasıl belirlenecektir? İki temel alt ölçütü vardır. Bunlardan biri "tüm sağlayıcılar", diğeri ise "seçili sağlayıcılar" dır.

Kaynak yayınlamanın amacı, aslında kaynağın bulunmasıdır ve kaynağın en hızlı şekilde bulunması esastır. Eğer bütün kaynak sağlayıcılara ait kaynak referansları tek bir yerde tutulursa bu "tüm sağlayıcılar" ölçütünün işaret ettiği durumdur, o zaman aranan kaynağın hem bulunması ve hem de kaynak referanslarının yönetimi ve yaşatılması için, bilgisayar kaynakları bakımından güçlü ve dolayısıyla pahalı sistemlere ihtiyaç vardır. Kataloglama bağlamında bu tip sistemler genellikle "merkezi" kataloglar olarak anılır. Bu durumda performansın yükseltilebilmesi için bir yol, Google mimarisi'nde (Brin ve Page,

2000) olduğu gibi bilgi kapsamının farklı sunuculara, yine merkezi kontrol altında bölünmesi olabilir. Ancak bu durumda da kontrolün yine merkezi olması nedeni ile tek sunucu bazında oluşabilecek başarısızlıkların (failure) önlenebilmesi için, her bir sunucunun bir de kopyası tutulmaktadır ki, bu da toplamda sistem maliyetini artıran bir durumdur.

"Seçili sağlayıcılar" içeren bir kapsama alanı ölçütü ile aslında P2P ağ yapısının temsili amaçlanmıştır. P2P ağ yapısı WS mimarisi için önerilen son yılların en popüler ve en aktif araştırma konularındandır. P2P sistemler konusunda çok sayıda çalışma bulunmakla birlikte, merkezi yapı ile olan farkların, üst düzey bir bakışla çok net bir biçimde ortaya konduğu bir kaynağa rastlanamamıştır. Bu tespit, bu konuda "genel" ve "saflaştırıcı" bir çatı tanımlamayı hedefleyen Hoschek (2002) tarafından da dile getirilmiştir. P2P ağların en belirleyici özelliklerinden biri, merkezi bir kontrolün bulunmamasıdır. Böylece hem performans hem güncelleme kolaylaşmaktadır. Diğer avantajlar başarısızlıklara karşı sağlamlık (robustness) ve gerek veri hacmi ve gerekse bağlı tarafların sayısı itibariyle ölçeklenebilirlik olarak belirtilmektedir (Knowledge Web, 2005).

"Kapsama alanı belirleme temeli", P2P terminolojisi ile "komşu belirleme temeli" olarak algılanabilir. P2P ağlarda arama komşu eşler üzerinden yönlendirildiği için bir eşin komşularının kim olduğu çok önemlidir. Kapsama alanı belirleme, anlamsal ya da anlamsal olmayan bir temele dayanabilir. Anlamsal olmayan kapsama alanı belirlemede en yaygın kullanılan yöntemlerden biri, bir *hash* fonksiyonu kullanmaktır. Bunlardan en popüler olanları CAN (Ratnasamy vd., 2001) ve Chord (Stoica, vd., 2001) dir. Bu tip P2P sistemler, bazen "rijid" ya da "yapılandırılmış" (structured) P2P olarak anılır. Ancak bu durumda anlamsal olarak birbirine hiçte yakın olmayan eşler aynı kapsama alanında yer alabilir. Yani P2P sistemler terminolojisi ile birbirine komşu durumuna gelebilir. P2P ağlarda kaynak arama, kaynak referansı itibarı ile kaynağı içermesi en olası komşular üzerinden yönlendirildiği için, bu durum anahtar sözcük ya da endeks referanslı aramalar için uygun olabilirse de, tanım referanslı aramalar için uygun olmayabilir. Çünkü kaynağın bulunması ya çok zaman alacak ya da ağdaki eşlerden birinde mevcut olsa bile bulunamayabilecektir. Çünkü P2P aramaların bir diğer karakteristiği, aramada bir eşin kaçıncı derecede komşusuna kadar bakılacağının belirtilmesidir. Bu "hop count (HC)" olarak anılır. Dolayısıyla HC değerinin çok yüksek tutulması ağın neredeyse tamamının sorgulanması anlamına gelecektir ki bu, hem diğer kullanıcılar açısından ağ trafiğine

olumsuz etki yapacak, hem de istemci eş açısından arama zamanını artıracaktır. Diğer yandan HC nin çok düşük tutulması aranan kaynağın ağda mevcut olsa bile bulunamaması sonucunu doğurabilecektir.

Anlamsal Örtü Ağları (AÖA/Semantic Overlay Networks (SON)), anlamsal olarak benzer eşlerin (peer) gruplanması ilkesine dayanır (Crespo ve Garcia-Molina, 2003). Ancak bu anlamsal benzerlik nasıl tanımlanacaktır? Önemli sorun ve gelecek araştırma konularından biri budur. Eşlerin belirli kriterlere göre gruplanması bazen "katalog federasyonu" ismi altında işaret edilen bir konudur. Her durumda konunun özü eşlerin bir şekilde gruplanması ve kaynak aramanın bu şekilde iyileştirilmesidir. Bu yöndeki çalışmalara bir örnek (Paik vd., 2004) tür. Burada belirli ölçütlere göre "katalog toplulukları" tanımlanmıştır. Bu ölçütler anlamsal temelli ölçütler olmadıklarından bu ve benzeri çalışmalar aynı zamanda kapsama alanının anlamsal olmayan ölçütler bazında belirlenmesine de örnek teşkil etmektedir. Kapsama alanı belirleme temeli açısından arzu edilen, kapsama alanlarının AÖA vasıtası ile belirlenmesidir. Ancak bugüne kadar bu açıdan ne konumsal ne de konumsal olmayan arenada yaygın kabul görmüş bir çözüm önerilmiş değildir. Dolayısıyla bu konu, çok net bir biçimde geleceğin en önemli araştırma konuları arasındadır.

Kaynak yayınlama/bulma kapsama alanı belirleme ölçütü kapsamında belirlenen bir diğer alt ölçüt ise "kapsama alanı belirleme tarzı" dır. Genel olarak kapsama alanları, ancak yeni eşlerin ağa katılması ya da yeni kaynakların yayınlanması durumunda değişecektir. Kapsama alanı statik ya da dinamik bir tarzda belirlenebilir. Statik tarz ile kapsama alanının tasarım zamanında belirlenmesi kastedilmektedir. Günümüz P2P gerçekleştirimlerinin büyük çoğunluğu bu tarzı kullanır. "Dinamik tarz" ile son yıllarda popülarite kazanan dinamik ya da değişken ağlardaki (mobile Networks) (Elenius ve Ingmarsson, 2004) durum kastedilmektedir. Bu ağlarda ağ eşleri, donanım bazında bugün yaygın geçerliğin sağlandığı "tak ve çalıştır" felsefesi ile ağa eklenebilmekte ve dolayısıyla ağ kapsama alanları dinamik ve hatta anlık olarak değişebilmektedir. Değişken ağlar, geleceğin uygulamaları açısından önemli potansiyeli bulunan araştırma alanlarından biridir. Bu nedenle kapsama alanlarının anlık olarak güncellenebilmesi bu ölçüt açısından arzu edilen durumdur.

Kaynak bulma: Bu ölçüt ile kapsanmaya çalışılan, aranan kaynağı sunan sağlayıcı ya da eşlerin bulunması işlevidir. Burada iki durum söz konusudur. Birincisi sağlayıcıların biliniyor farz edilmesi durumudur. OGC Web servislerini kullanan pek çok araştırma ve

market gerçekleştirimi çalışmalarındaki durum budur. Sağlayıcılarının bilinmemesi durumu gerçekçi olan durumdur ve bu durumda bir tür yönlendirme mekanizmasına ihtiyaç vardır. Burada tüm sağlayıcılara yönlendirme (flooding) durumu merkezi katalog ya da tüm sağlayıcıları içeren kapsama alanı durumudur. P2P deki HC değerinin çok yüksek tutulması da, yukarıda açıklandığı gibi aynı anlama gelebilir. Seçili sağlayıcıları yönlendirme ise yukarıda açıklandığı şekilde belirlenecek olan kapsama alanına göre yapılacaktır.

İstek ve yayın eşleştirme: İstemciden bir formda gelen isteğin yayınlanan kaynak referansı ile eşleştirilmesi işleminin kapsamını değerlendirmek içindir. "Eşleştirme kaynak türü" alt ölçütü, eşleştirmenin veri ya da servise yönelik olması durumunu belirler. "Eşleştirme temeli" eşleştirmenin sözdizimsel ya da anlamsal mı yapıldığını belirtmek içindir. Anlamsal eşleştirme için en yaygın kullanılan yöntemlerden biri "mantık tabanlı içerme" (logic based subsumption) algoritmalarıdır. Bu tür eşleştirmeler, "eşleştirme gerçekleşme şekli" ölçütü altında yer alan, eşleştirmenin program tarafından ya da "tam otomatik" olarak yapılması durumuna ve "eşleştirme referansı" altında yer alan tanım eşleştirmeye örnektir. Eşleştirme sözdizimsel temelde tanım eşleştirme olarak ta gerçekleştirilebilir. OGC "yetenekler" dokümanının insan kullanıcı tarafından, istemci isteğine yönelik olarak incelenmesi bu tip bir eşleştirmeye ve aynı zamanda otomatik olmayan" yani insan tarafından yapılan eşleştirmeye örnektir. "Eşleştirme özellikleri" ile, tanım bazlı eşleştirmede kullanılan özellikler kastedilmektedir. WS literatüründe bu özellikler işlevsel (functional) ve işlevsel olmayan (non-functional) özellikler olarak anılmaktadır. İşlevsel özellikler servisin girdi, çıktı, önkoşul ve son koşul/etki (IOPE) parametreleridir. İşlevsel olmayan özellikler ise genellikle servis kalitesi (QoS) parametreleri olarak anılan özelliklerdir. Bu kavramlarla ilgili detaylı bilgi servis düzenleme kısmında verildiği için burada tekrar edilmeyecektir.

Uygulama geliştirme: Uygulama geliştirme ile kastedilen kullanıcıların herhangi bir uygulamayı nasıl gerçekleştirdikleridir. SYM de uygulama geliştirmenin yolu servis düzenleme ya da son yıllarda popülarite kazanan servis işbirliğinin tanımlanması olarak algılanabilecek servis koreografisidir. O nedenle bu ölçüt altında vurgulanmaya çalışılan bir servis düzenlemenin söz konusu olup olmadığıdır. Bu nedenle burada iki seçenek belirlenmiştir. Bunlardan biri servis düzenlemesiz uygulama geliştirme şeklidir. OGC Web servislerine dayalı pek çok gerçekleştirim ve hatta mevcut GOS ve UKVA gerçekleştirimleri bu gruba girmektedir. Diğer ölçüt servis düzenleme yoluyla uygulama

geliştirmedir. Bu, SYM nin öngördüğü yöntemdir. Servis düzenlemede amaç belirli bir uygulamayı gerçekleştirmek için kullanılabilecek bir dizi servisin belirli bir koşum düzeninde bir araya getirilmesidir. Bunun için servislerin bulunması, koşum düzeninin belirlenmesi ve birlikte işlerliklerinin sağlanması söz konudur.

SYM de servis düzenleme soyut ve somut düzeyde gerçekleşebilir. Somut düzenleme hangi servis örneklerinin hangi koşum düzeninde koşacağının ve birlikte işlerliklerinin nasıl sağlanacağının tanımlandığı düzenlemedir. Birlikte işlerlik sağlamadan kasıt örneğin servis parametrelerinin tip olarak uyumunun eşleştirme sırasında belirlenmesi dışında örneğin parametre sayısındaki farklılığın giderilmesi olabilir. Bu konuda bir örnek servis düzenleme kısmında OGC servislerini kullanan bir düzenlemede verilmişti. Söz konusu düzenleme belirli bir dilde ifade edilir. Bunun için en yaygın kullanılan dil ise BPEL dir. Soyut düzenleme kavramı ise bütün servis örneklerinin belirtilmediği, koşum zamanında ortaya çıkabilecek beklenmedik durumlar nedeniyle bir kısım bileşen servislere ait tanımların soyut bırakıldığı tanımlama şeklidir. Örneğin bir seyahat planlama servisi düzenlemesinde düzenlemeyi tanımlayan doküman içerisinde "...ulaşım servisi rezervasyonu yapacak olan bileşen servisin hava, kara ya da deniz ulaşım rezervasyonu servislerinden herhangi birinin olabileceği…" nin belirtilmesi, soyut servis düzenlemeye örnektir. Koşum zamanında ilgili sağlayıcılardan hangi servisler sağlanabilirse onlar kullanılacaktır. Buradaki ima, anlaşılabileceği üzere, soyut servis tanımının koşum anında somut servis tanımına dönüştürüldüğüdür. Eğer servis düzenleme tasarım anında tümüyle somut yapılsaydı ve koşum anında örneğin bu servislerden biri sağlayıcı tarafından ilgili katalogda artık yayınlanmıyor olsa idi bu servis örneği bulunamayacak ve uygulama gerçekleştirilemeyecek idi. Soyut servis düzenleme kavramı ile bu gibi sorunların çözülmesi hedeflenmiştir.

Servis düzenleme için gerekli olan eşleştirmeler yukarıda ve servis düzenleme kısmında açıklandığı gibi gerçekleştirilir. Ayrıca eşleştirmenin tasarım anında ya da koşum zamanında gerçekleştirilmesi söz konudur. Doğal olarak, söz konusu olan bir eşleştirme olduğu için, eşleştirmenin gerçekleşme şekilleri burada da geçerlidir. Buna göre tasarım zamanı tanımlı soyut düzenleme BPEL Abstract ve OWL-S işlem şablonu (process template) yoluyla olabilir. BPEL Abstract durumunda somutlaştırma "kullanım profili (usage profile) vasıtası ile olur. Şirin vd. (2005) teki tercihler (preferans) gereği ile olduğu gibi.

Tasarım zamanı somut düzenleme BPEL işlem dokümanı ile tanımlanmış olan bir düzenlemedir. Böyle bir düzenleme için servislerin ve sağlayıcılarının bilinmesi gerekir. Bu da ancak otomatik olmayan bir yolla insan kullanıcı tarafından yapılabilir. Koşum zamanında somut servis düzenleme ise yarı-otomatik ve tam-otomatik olarak gerçekleştirilebilir. Yarı otomatik yöntemde yazılım, servis örneklerini bulur, kullanıcı örnekleri işlevsel ve işlevsel olmayan özellikleri bazında teke indirir. Tam-otomatik yöntemde ise alternatif servis örnekleri bulma ve bunları teke indirme yazılım tarafından gerçekleştirilir. Bunun için kullanılan tekniklerden biri Yapay Zeka Planlama (YZP) dır (Peer, 2005). Bir örnek Şirin vd. (2005) te uygulana HTN (Hierarchical Task Network) planlamadır.

Servis işbirliğinin tanımlanması yoluyla (Service Choreography) uygulama geliştirme de bir dizi servisin işbirliğini gerektirir. Ancak koreografide bu işbirliği, servis düzenlemedeki gibi belirli bir perspektiften_ki bu yeni düzenlenmekte olan servisin perspektifidir_değil de genel standart bir perspektiften tanımlanır. Servis koreografisi ve servis düzenleme arasındaki fark henüz çok net bir biçimde ortaya konamamıştır ancak koreografi standartlaşma gerektirmektedir ki bu da "görev" (task) ontolojileri vasıtası ile olabilir. Yani bir "satıcı" ve "alıcı" nın rol ve görevleri tanımlandıktan sonra, koşum zamanında geriye sadece alıcı ve satıcının kimler olduğunun belirlenmesi kalacaktır. Her ikisi de kendileri için belirlenmiş kurallara göre ilgili işlemeleri yürütecektir.

Tablo 8, Web servisleri mimarisine dayalı KVA gerçekleştirimlerini sınıflandırmak için geliştirilen ölçütleri ve bu ölçütler ile ilgili özet bilgileri içermektedir. Tablo 9, Tablo 10 ve Tablo 11 de, konumsal ve konumsal olmayan başlıca WS gerçekleştirimleri ve UKVA market gerçekleştirimlerinin belirlen ölçütlere değerlendirmesi yapılmıştır.

Tablo 8. Web servisleri mimarisinde KVA gerçekleştirimleri için sınıflandırma ölçütleri

Ölçüt	**Açıklama**
Kaynak Tanımlama	Servis ya da veri tanımlama ile ilgili özellikleri tanımlar.
Tanımlanan bilgi kaynağı türü	Tanımlanan bilgi kaynağının türü, veri ya da servis olabilir. Veri ile, konumsal ve konumsal olmayan, çoklu ortam verisi dahil her tür veri kastedilmektedir. Servis ile Web servisi kastedilmektedir.
Veri	
Servis	
Tanımlama temeli	Tanımlamanın sözdizimsel mi yoksa anlamsal mı olduğunu belirleyen ölçüttür. Anlamsal tanımlama, makine ya da yazılım kodunun bilgi çıkarsaması yapmasına olanak tanıyan bir tanımlamadır. Sözdizimsel tanımlamada ise bilgi çıkarsaması insan tarafından yapılmaktadır.
Sözdizimsel	
Anlamsal	
Araç/standart/belirtim/teknoloji	Geleneksel şema tanımlama, XML Schema ve WSDL sözdizimsel, RDF, RDF-S, DAML, DAML-S, OWL, OWL-S ve WSDL-S anlamsal veri ve servis tanımı için kullanılabilecek araç/standart/belirtim ve teknolojilere örnek olarak verilebilir.
Kaynak Yayınlama	Servis ya da veri yayınlama ile ilgili özellikleri tanımlar.
Yayınlama referansı	Kaynağın katalog ya da reklam edildiği yer her neresi ise, orada hangi bilgilerle temsil edildiğini belirlemek içindir. Tanım referanslı yayınlama ile ebRIM ya da UDDI kataloglarında, bu katalogların bilgi modelleri ya da bu katalogların konumsal veriye yönelik profillerine göre yapılacak olan servis tanımları kastedilmektedir. Endeks referanslı yayınlama ile özellikle veri türü kaynakların temsili için kullanılan, fakat servis türü kaynaklar için de uygulanabilecek, anahtar kelimeler ile endeksleme kastedilmektedir.
Endeks referanslı	
Tanım referanslı	
Endeks temeli	Endeks temeli, endeksin söz dizimsel mi yoksa anlamsal mı olduğunu belirtmek içindir. Söz dizimsel temelde istatistiksel doküman özetleme, doküman içinden anahtar sözcük vd. endeksi oluşturabilir. Anlamsal temelde ise, örneğin "semantic latent index" gibi semantik endeksler kullanılabilir.
Sözdizimsel	
Anlamsal	
Tanım temeli	Tanım temeli, kaynağın söz dizimsel mi yoksa anlamsal bir tanımlama ile mi yayınlandığını belirtmek içindir. Geleneksel katalog servislerinin kullanılması durumunda tanım temeli ya söz dizimsel ya da anlamsal zenginleştirilmiş söz dizimsel olabilir. Anlamsal tanımlama, Tanımlama Mantığı (Description Logics) temelli OWL-S ve DAML-S gibi diller ile yapılabilir.
Sözdizimsel	
Anlamsal zenginleştirilmiş sözdizimsel	
Anlamsal	
Araç/standart/belirtim/teknoloji	Bu ölçüt kapsamındaki araç/standart/belirtim/teknolojiler ebXML katalog servisi, UDDI katalog servisi, OGC CSW katalog servisi ve ISO 19115/19119 veri ve servis meta veri tanımlama standartları ve bilgi tabanları olabilir. Anlamsal tanımlama durumunda yayınlanma bir bilgi tabanı vasıtası ile olabilir.
Geleneksel	
Anlamsal zenginleştirme için genişletilmiş geleneksel	
Bilgi Tabanı	
Kaynak Yayınlama/Bulma Kapsama Alanı Belirleme	Kaynak yayınlama ve kaynak bulmanın kapsama alanının belirlenmesi ile ilgili özellikleri tanımlar.
Kapsama alanının karakteri	Kapsama alanının karakterini belirlemek içindir. İki temel alt ölçütü vardır. Bunlardan biri "tüm sağlayıcılar", diğeri ise "seçili sağlayıcılar" dır. Eğer bütün kaynak sağlayıcılara ait kaynak referansları tek bir yerde tutulursa bu "tüm sağlayıcılar" ölçütünün işaret ettiği durumdur. Kataloglama bağlamında bu tip sistemler genellikle "merkezi" kataloglar olarak anılır. Seçili sağlayıcıları içeren bir kapsama alanı ölçütü ile P2P ağ yapısının temsili amaçlanmıştır.
Tüm sağlayıcılar	
Seçili sağlayıcılar	

Tablo 8'in devamı

Ölçüt	**Açıklama**
Kapsama alanı belirleme temeli	Kapsama alanı belirleme temeli, P2P terminolojisi ile "komşu belirleme temeli" olarak algılanabilir. Kapsama alanı belirleme, anlamsal ya da anlamsal olmayan bir temele dayanabilir. Anlamsal olmayan kapsama alanı belirlemede en yaygın kullanılan yöntemlerden biri, bir hash fonksiyonu kullanmaktır. Anlamsal kapsama alanı belirlemede en yaygın kullanılan yöntem ise, "Anlamsal Örtü Ağları" tanımlamaktır.
Anlamsal olmayan	
Anlamsal	
Kapsama alanı belirleme tarzı	Kapsama alanı statik ya da dinamik bir tarzda belirlenebilir. Statik tarz ile kapsama alanının tasarım zamanında belirlenmesi kastedilmektedir. Dinamik tarz ile, son yıllarda popülarite kazanan dinamik ya da değişken ağlardaki (mobile Networks) durum kastedilmektedir.
Statik Tasarım zamanı tanımlı	
Dinamik Koşum zamanı tanımlı	
Kaynak Bulma	Aranan bir bilgi kaynağını sunan sağlayıcı ya da eşlerin bulunması ile ilgili özellikleri tanımlar.
Aranan bilgi kaynağı türü	Aranan bilgi kaynağının türü, veri ya da servis olabilir. Veri ile, konumsal ve konumsal olmayan, çoklu ortam verisi dahil her tür veri kastedilmektedir. Servis ile Web servisi kastedilmektedir.
Veri	
Servis	
Sağlayıcılar biliniyor	Aranan bilgi kaynağını sunan sağlayıcılar biliniyor olabilir. OGC Web servislerini kullanan pek çok araştırma ve market gerçekleştirimi çalışmalarındaki durum budur.
Sağlayıcılar bilinmiyor	Aranan bilgi kaynağını sunan sağlayıcılar bilinmiyor olabilir. Bu durumda bir tür yönlendirme mekanizmasına ihtiyaç vardır. Sağlayıcıların bulunması, "kaynak yayınlama/bulma kapsama alanı belirleme" ölçütündeki kapsama alanı karakterini belirleme yöntemine göre bulunur.
İstek ve yayın eşleştirme	İstemciden gelen isteğin, yayınlanan kaynak referansı ile eşleştirilmesi işleminin kapsamını değerlendirmek içindir.
Eşleştirme kaynak türü	Eşleştirme kaynak türü, eşleştirmenin veri ya da servise yönelik olması durumunu belirler.
Veri	
Servis	
Eşleştirme temeli	Eşleştirme temeli, eşleştirmenin sözdizimsel mi yoksa anlamsal mı yapıldığını belirtmek içindir. Anlamsal eşleştirme için en yaygın kullanılan yöntemlerden biri "mantık tabanlı içerme" (logic based subsumption) algoritmalarıdır. Eşleştirme sözdizimsel temelde gerçekleştirilebilir. OGC "yetenekler" dokümanının insan kullanıcı tarafından, istemci isteğine yönelik olarak incelenmesi bu tip eşleştirmeye örnektir.
Sözdizimsel	
Anlamsal	
Eşleştirme referansı	Eşleştirmede kullanılan referansı belirtmek için kullanılan ölçüttür. Bu endeks eleştirme ve tanım eşleştirme olabilir.
Endeks eşleştirme	
Tanım eşleştirme	
Eşleştirme özellikleri	Eşleştirme özellikleri ile tanım bazlı eşleştirmede kullanılan özellikler kastedilmektedir. Web servisleri literatüründe bu özellikler işlevsel (functional) ve işlevsel olmayan (non-functional) özellikler olarak anılmaktadır. İşlevsel özellikler servisin girdi, çıktı, önkoşul ve son koşul/etki (IOPE) parametreleridir. İşlevsel olmayan özellikler ise genellikle servis kalitesi (QoS) parametreleri olarak anılan özelliklerdir.
İşlevsel özellikler	
İşlevsel olmayan özellikler	
Araç/standart/belirtim/teknoloji	Eşleştirme tabloları (Mapping tables), Crosswalks, Dağıtık Hash Tabloları (DHT), Şema eşleştirme tanımlamaları (GAV, LAV vd.), Mediators, Wrappers, Agents, İçerme algoritmaları (Subsumption algorithms), Ontolojiler, Ontoloji dönüşüm tanımlamaları ve çıkarsamacı (reosoner) bu ölçüt kapsamındaki araç/standart/belirtim ve teknolojilere örnektir.

Tablo 8'in devamı

Ölçüt	**Açıklama**
Eşleştirme gerçekleşme şekli Otomatik olmayan (non-automated) Yarı otomatik (semi-automated) Tam otomatik (full automated)	Eşleştirmenin insan ya da yazılım tarafından yapıldığını belirtmek içindir. Otomatik olmayan eşleştirme yönteminde eşleştirme insan tarafından yapılır. Tam otomatik yöntemde eşleştirme yazılım tarafından otomatik olarak gerçekleştirilir. Yarı otomatik yöntemde ise eşleştirme insan ve yazım tarafından birlikte gerçekleştirilir.
Uygulama geliştirme	Kullanıcıların herhangi bir uygulamayı nasıl gerçekleştirdiklerini tanımlar. Bu ölçüt altında vurgulanmaya çalışılan bir servis düzenlemenin söz konusu olup olmadığıdır.
Servis düzenlemesiz API	Kullanıcılar uygulamalarını servis düzenleme yapmadan gerçekleştirebilir. OGC Web servislerine dayalı pek çok gerçekleştirim ile mevcut GOS ve UKVA gerçekleştirimleri bu gruba girmektedir.
Servis düzenleme yoluyla Soyut Tasarım zamanı (design-time) tanımlı Koşum zamanı (run-time) tanımlı Somut Tasarım zamanı (design-time) tanımlı Koşum zamanı (run time) tanımlı	Servis düzenleme, tasarım veya koşum zamanında soyut veya somut düzeyde gerçekleşebilir. Somut düzenleme hangi servis örneklerinin hangi koşum düzeninde koşacağının ve birlikte işlerliklerinin nasıl sağlanacağının tanımlandığı düzenlemedir. Soyut düzenleme ise bütün servis örneklerinin belirtilmediği, koşum zamanında ortaya çıkabilecek beklenmedik durumlar nedeniyle bir kısım bileşen servislere ait tanımların soyut bırakıldığı düzenleme şeklidir.
Servis işbirliğinin tanımlanması yoluyla (Service Choreography)	SYM de uygulama geliştirmenin bir diğer yolu son yıllarda popülarite kazanan servis işbirliğinin tanımlanması olarak algılanabilecek servis koreografisidir. Servis işbirliğinin tanımlanması yoluyla uygulama geliştirme de bir dizi servisin işbirliğini gerektirir. Ancak koreografide bu işbirliği, servis düzenlemedeki gibi belirli bir perspektiften değil de genel standart bir perspektiften tanımlanır. Servis koreografisi ve servis düzenleme arasındaki fark henüz çok net bir biçimde ortaya konamamıştır.
Araç/standart/belirtim/teknoloji	BPEL Abstract, OWL-S işlem şablonları, Yapay Zeka Planlama (örn. HTN planlama), içerme algoritmaları (subsumption algorithms), ontolojiler, ontoloji dönüşüm tanımlamaları, çıkarsamacı (reosoner), BPEL işlemci (BPEL engines), OWL-S işlemci, Pi4 araçları (Pi4 tools) bu ölçüt kapsamındaki araç/standart/belirtim ve teknolojilere örnektir.

Tablo 9. Konumsal olmayan akademik çalışmaların belirlenen sınıflandırma ölçütlerine göre değerlendirilmesi

Çalışma / Ölçüt	METEOR-S	Şirin vd. (2005)	WSMO
Kaynak Tanımlama (description)			
Tanımlanan bilgi kaynağı türü			
Veri			
Servis	x	x	x
Tanımlama temeli			
Söz dizimsel			
Anlamsal	x	x	x
Araç/Standart/Belirtim/Teknoloji	MWSCF Meteor Web Services Composition Framework (genişletilmiş WSDL)	OWL-S	WSML
Kaynak Yayınlama (publish / advertise)			
Yayınlama referansı			
Endeks referanslı			
Tanım referanslı	x	x	x
Endeks temeli			
Sözdizimsel			
Anlamsal			
Tanım temeli			
Sözdizimsel			
Anlamsal zenginleştirilmiş Sözdizimsel	x		
Anlamsal		x	x
Araç/standart/belirtim/teknoloji			
Geleneksel			
Anlamsal zenginleştirme için genişletilmiş geleneksel	Genişletilmiş UDDI		
Bilgi Tabanı		x	x
Kaynak Yayınlama / Bulma Kapsama Alanı Belirleme		Yok	
Tüm sağlayıcılar			
Seçili sağlayıcılar	x (EXTRO registry ontolojileri)		x
Kapsama alanı belirleme temeli			
Anlamsal olmayan			
Anlamsal	x		x
Kapsama alanı belirleme tarzı			
Statik	x		x
Dinamik			
Kaynak Bulma (discovery)			
Aranan bilgi kaynağı türü			
Veri			
Servis	x	x	x

Tablo 9'un devamı

Çalışma Ölçüt	METEOR-S	Şirin vd. (2005)	WSMO
Sağlayıcılar biliniyor		x	
Sağlayıcılar bilinmiyor	x		x
İstek ve yayın eşleştirme (match making)			
Eşleştirme kaynak türü			
Veri			
Servis	x	x	x
Eşleştirme temeli			
Söz dizimsel			
Anlamsal	x	x	x
Eşleştirme referansı			
Endeks eşleştirme			
Tanım eşleştirme	x	x	x
Eşleştirme özellikleri			
İşlevsel özellikler	x	x	x
İşlevsel olmayan özellikler		x	x
Araç/standart/belirtim/teknoloji	İçerme algoritmaları, Çıkarsamacı, Ontolojiler	İçerme algoritmaları, Çıkarsamacı, Ontolojiler	İçerme algoritmaları, Çıkarsamacı, Ontolojiler
Eşleştirme gerçekleşme şekli			
Otomatik olmayan (non-automated)			
Yarı otomatik (semi-automated)	x		
Tam otomatik (full-automated)		x	
Uygulama geliştirme			Üzerinde çalışılıyor
Servis düzenlemesiz API			
Servis düzenleme yoluyla	x	x	
Soyut			
Tasarım zamanı (design time) tanımlı	x	x	
Koşum zamanı (run time) tanımlı			
Somut			
Tasarım zamanı (design time) tanımlı			
Koşum zamanı (run time) eşleştirme	x	x	
Servis işbirliğinin tanımlanması yoluyla (Service Choreography)			
Araç/standart/belirtim/teknoloji	BPEL işlemci, MWSCF işlem şablonu,	OWL-S işlem şablonu, OWL-S işlemci, HTN Planlama, Ontolojiler, Çıkarsamacı (Pellet)	

Tablo 10. Konumsal akademik çalışmaların belirlenen sınıflandırma ölçütlerine göre değerlendirilmesi

Çalışma / Ölçüt	Lemmens vd. (2006)	Lutz ve Klien (2006)	Essid vd. (2004)
Kaynak Tanımlama (description)			
Tanımlanan bilgi kaynağı türü			
Veri		x	x
Servis	x		
Tanımlama temeli			
Sözdizimsel			x
Anlamsal	x	x	
Araç/standart/belirtim/teknoloji	WSDL-S	OWL-DL	Geleneksel şema tanımlama
Kaynak Yayınlama (publish / advertise)			Yok
Yayınlama referansı			
Endeks referanslı			
Tanım referanslı	x	x	
Endeks temeli			
Sözdizimsel			
Anlamsal			
Tanım temeli			
Sözdizimsel			
Anlamsal zenginleştirilmiş sözdizimsel			
Anlamsal	x	x	
Araç/standart/belirtim/teknoloji			
Geleneksel		CSW, ISO 19115	
Anlamsal zenginleştirme için genişletilmiş geleneksel			
Bilgi Tabanı	x	x	
Kaynak Yayınlama / Bulma Kapsama Alanı Belirleme	Yok		Yok
Tüm sağlayıcılar		x	
Seçili sağlayıcılar			
Kapsama alanı belirleme tabanı			
Anlamsal olmayan			
Anlamsal			
Kapsama alanı belirleme tarzı			
Statik			
Dinamik			
Kaynak Bulma (discovery)			
Aranan bilgi kaynağı türü			
Veri		x	x
Servis	x		
Sağlayıcılar biliniyor	x		x
Sağlayıcılar bilinmiyor			

Tablo 10'un devamı

Çalışma / Ölçüt	Lemmens vd. (2006)	Lutz ve Klien (2006)	Essid vd. (2004)
İstek ve yayın eşleştirme (match making)			
Eşleştirme kaynak türü			
Veri		x	x
Servis	x		
Eşleştirme temeli			
Söz dizimsel			x
Anlamsal		x	
Eşleştirme referansı			
Endeks eşleştirme			
Tanım eşleştirme	x	x	x
Eşleştirme özellikleri			
İşlevsel özellikler	x	x Veri tipi	x Veri tipi
İşlevsel olmayan özellikler			
Araç/standart/belirtim/teknoloji	İçerme algoritmaları, çıkarsamacı, Ontolojiler	İçerme algoritmaları, Çıkarsamacı, Ontolojiler	Şema eşleştirme tanımlamaları, Mediators, Wrappers
Eşleştirme gerçekleşme şekli			
Otomatik olmayan (non-automated)			
Yarı otomatik (semi-automated)	x	x	
Tam otomatik (full-automated)			
Uygulama geliştirme			
Servis düzenlemesiz API		x	x
Servis düzenleme yoluyla	x	yok	yok
Soyut			
Tasarım zamanı (design time) tanımlı			
Koşum zamanı (run time) tanımlı	x		
Somut			
Tasarım zamanı (design time) tanımlı			
Koşum zamanı (run time) tanımlı	x		
Servis işbirliğinin tanımlanması yoluyla (Service Choreography)			
araç/standart/belirtim/teknoloji	OWL-S işlem şablonu, Ontolojiler, Çıkarsamacı (RacerPro), BPEL işlemci,		

Tablo 11. UKVA gerçekleştirimlerinin belirlenen sınıflandırma ölçütlerine göre değerlendirilmesi

Çalışma / Ölçüt	GOS Portal	Clearinghouse	EU GeoPortal
Kaynak Tanımlama (description)			
Tanımlanan bilgi kaynağı türü			
Veri	x	x	x
Servis	x		
Tanımlama temeli			
Söz dizimsel	x	x	x
Anlamsal			
Araç/standart/belirtim/teknoloji	FGDC Meta veri standardı	FGDC Meta veri standardı	FGDC Meta veri standardı
Kaynak Yayınlama (publish / advertise)			
Yayınlama referansı			
Endeks referanslı			
Tanım referanslı	x	x	x
Endeks temeli			
Sözdizimsel			
Anlamsal			
Tanım temeli			
Sözdizimsel	x	x	x
Anlamsal zenginleştirilmiş sözdizimsel			
Anlamsal			
Araç/standart/belirtim/teknoloji			
Geleneksel	ArcSDE + İVTYS FGDC	Z39.50 + İVTYS FGDC	ArcSDE + İVTYS FGDC
Anlamsal zenginleştirme için genişletilmiş geleneksel			
Bilgi Tabanı			
Kaynak Yayınlama / Bulma Kapsama alanı belirleme			
Tüm sağlayıcılar	x	x	x
Seçili sağlayıcılar		x	
Kapsama alanı belirleme temeli			
Anlamsal olmayan			
Anlamsal			
Kapsama alanı belirleme tarzı			
Statik	x	x	x
Dinamik			
Kaynak Bulma (discovery)			
Aranan bilgi kaynağı türü			
Veri	x	x	x
Servis	x	x	x
Sağlayıcılar biliniyor			
Sağlayıcılar bilinmiyor			

Tablo 11'in devamı

Çalışma / Ölçüt	GOS Portal	Clearinghouse	EU GeoPortal
İstek ve yayın eşleştirme (match making)			
Eşleştirme kaynak türü			
Veri	x	x	x
Servis			
Eşleştirme temeli			
Söz dizimsel	x	x	x
Anlamsal			
Eşleştirme referansı			
Endeks eşleştirme			
Tanım eşleştirme	x	x	x
Eşleştirme özellikleri			
İşlevsel özellikler	x Veri tipi	x Veri tipi	x Veri tipi
İşlevsel olmayan özellikler			
Araç/standart/belirtim/teknoloji			
Eşleştirme gerçekleşme şekli			
Otomatik olmayan (non-automated)			
Yarı otomatik (semi-automated)			
Tam otomatik (full-automated)			
Uygulama geliştirme			
Servis düzenlemesiz API	x API	x API	x API
Servis düzenleme yoluyla			
Soyut			
Tasarım zamanı (design time) tanımlı			
Koşum zamanı (run time) tanımlı			
Somut			
Tasarım zamanı (design time) tanımlı			
Koşum zamanı (run time) tanımlı			
Servis işbirliğinin tanımlanması yoluyla (Service Choreography)			
Araç/standart/belirtim/teknoloji			

4. İRDELEME

Bu çalışmada, KVA'ların Web servisleri ile gerçekleştirilmelerindeki temel konuların genel, basitleştirici ve bütünleştirici bir yaklaşımla ele alınmasını sağlayacak bir çerçeve geliştirilmiştir. Doğal olarak çalışmanın katkısı, "herhangi bir KVA gerçekleştirimcisi ya da karar vericisi için bu çerçeve yol gösterici bir rol oynayabilecek midir?", "bunu nasıl sağlayacaktır?" gibi soruları ne dereceye kadar cevapladığına bağlı olacaktır.

Bu çerçevenin en önemli katkısı, halen olgunlaşma sürecinde olan ve üzerinde henüz uzlaşma sağlanamamış çeşitli konuları içeren Web servisleri alanında, ilgili konuları kategorize etmiş olmasıdır. Gerçekleştirimciler için konuların bu şekilde sınıflandırılması basitleştirme sağladığı için katkı oldukça açıktır.

Bu çerçeve ile herhangi bir akademik çalışma, gerçekleştirim ya da ilgili yazılım kolaylıkla değerlendirilebilir. Çerçeve hem veri ve hem de servisleri kapsamaktadır. Bu durum her iki tür kaynağa yönelik olarak yapılmış çalışmaların bütüncül tek bir çatı altında değerlendirilebilmesi açısından son derece önemlidir. Çerçeve geneldir. Yalnızca UKVA için değil, herhangi bir ölçekte KVA kurma ve yaşatma girişimine de yol gösterici niteliktedir.

Çerçeveyi oluşturan ölçütler SYM deki temel işlevlere yönelik olarak belirlenmiştir. Bu bakımdan, ölçütlerin kendi içinde tutarlı olması amaçlanmıştır. Dolayısıyla sınıflandırma, ana ölçütlerin her biri bazında ayrı ya da tek bir bütün olarak değerlendirmeye olanak tanımaktadır. Buna göre bir SYM gerçekleştirimi, yalnızca kaynak yayınlama bazında ya da diğer SYM işlevlerinin tümü açısından değerlendirilebilir.

Çerçeve SYM nin temel işlevleri olan kaynak yayınlama, kaynak bulma ve uygulama geliştirme konularında bir sınıflandırma sunmaktadır. Çalışma kapsamında ayrıca bu sınıflandırmanın her biri sınıfı bazında avantajlar dezavantajlar ortaya konmuştur**.** Çerçeveden yararlanmak isteyen gerçekleştirimci ve karar vericiler bu bilgiler ışığında çeşitli kararlar alabilirler.

Çerçevenin ölçütler bazında işaret ettiği avantajlar ve dezavantajlar açısından çok somuta indirgeyerek bir değerlendirme yapıldığında anlamsal temelli yaklaşımların tercih edilmesi gerektiği anlaşılmaktadır. Bu zaten geleceğin anlamsal Web'in olacağı genel eğilimi ile de uyumludur. Ancak anlamsal Web gerçekleştirimlerine yönelik yaklaşım ve araçlar henüz mevcut değildir ve çeşitli sorunlar mevcuttur. Örneğin tam otomatik ve hatta

yarı otomatik servis düzenleme üzerinde görüş birliği sağlanmış çözümler mevcut değildir. Web servisi tanımının hangi parametreler içermesi gerektiği konusu da çözümlenememiştir. P2P yaklaşımlar, merkezi servis yayınlama ve bulmaya tercih edilmeleri için yeterli gerekçe sunmaktadır. Anlamsal örtü ağlarının, değişken ağların çok önemli potansiyelleri olduğu anlaşılmaktadır. Ancak bu yaklaşımları ve Web servislerini tek bir çerçevede toplayan bir gerçekleştirim üzerinde yoğun olarak çalışılmakla birlikte, henüz başarılamamıştır.

Bu noktada kritik sorulardan biri, "Bugün bir KVA gerçekleştirmek durumunda olan bir gerçekleştirimci, bu belirsizlikler ortamında nasıl bir yol izlemelidir?" sorusudur. SYM ye geçişte anılan sorunların henüz çözümlenememiş olması nedeni ile son zamanlarda bir ara çözüm olarak önerilen "Portal" gerçekleştirimleri, özellikle popüler CBS yazılım firmalarının da desteklediği bir çözüm durumundadır. Ancak portal gerçekleştirimleri ile SYM nin, örneğin servis düzenleme gibi, sorunlarına çözüm sunulmamaktadır. Portal yaklaşımı ile önerilen yalnızca, servis ya da "portlet" (SUN, 2003) sağlayıcılarının kullanıcı tarafından bilindiği durumdaki uygulama geliştirmedir ki burada zaten yeni bir durum ya da çözüm önerme söz konusu değildir. Yoksa P2P ağ ortamında portlet yayınlama, portal federasyonu ve portlet iş akışı geliştirme gibi konular da henüz çözüm önerilmemiştir. Bu durumda gerçekleştirimci, servis sağlayıcıların kullanıcı tarafından bilindiği durumdaki uygulama geliştirme tarzı ile çalışabilir. Bu tarzın portal gerçekleştirimi ile gerçekleştirilmesi de mümkündür ancak bu durumda kullanılacak yazılım araçlarının SYM deki özel yazılımlardan bağımsız uygulama geliştirebilme özgürlüğünü sunması gerekmektedir.

5. SONUÇLAR VE ÖNERİLER

Yerel, bölgesel, ulusal ve uluslararası ölçekteki KVA'ların, en son kuşak Web teknolojisi olarak kabul edilen Web servisleri teknolojisi ile gerçekleştirilebilmeleri Dünya genelinde henüz başarılabilmiş bir görev değildir. Çünkü böyle bir gerçekleştirimin, teknik birlikte işlerlik altyapısını tanımlayan, üzerinde anlaşma sağlanmış bir çerçeve mevcut değildir. Bu durum, WS yönelimli bir KVA gerçekleştirmek isteyenlerin işini, gerek teknik gerçekleştirimciler ve gerekse KVA'ları kurmak ve yaşatmaktan sorumlu karar vericiler bazında çok zor kılmaktadır.

Bu soruna bir çözüm getirmeyi hedeflemiş olan bu tez çalışmasında öncelikle, söz konusu gerçekleştirimci ve karar vericilerin çözmesi gereken başlıca sorular belirlenerek, bunlara nasıl çözüm getirilebileceği araştırılmıştır. Bu sorunlardan bir kaçı şunlardır: Web servisi tanımlama, bulma ve servis düzenleme için hangi teknoloji, araç, standart ya da belirtimler kullanılmalıdır? Servis yayınlama ve bulmada merkezi ya da dağıtık ağ yapılandırılmalarından hangisi tercih edilmelidir? Servis kataloglama ve katalog federasyonları nasıl gerçekleştirilecektir? Hangi ticari ve/veya açık kod yazılımların tercih edileceğine nasıl karar verilecektir? WS yönelimli KVA'lara geçişin ana amacı olan birlikte işlerlik, acaba gerçekten sağlanabilecek midir yoksa sorun olmaya devam mı edecektir? Bu birlikte işlerlik "sözdizimsel" ya da "anlamsal" düzeyde mi sağlanmalıdır? Mevcut araçlar hangisini olanaklı kılmaktadır? Belirli kuşak araç ve teknolojilerle yapılan gerçekleştirimler arasında geçiş olanaklı olacak mıdır? Mevcut uygulamalar (legacy applications) WS mimarisine adapte edilebilecek midir?

Bu tez çalışmasında, WS yönelimli bir KVA'nın teknik birlikte işlerlik altyapısının nasıl gerçekleştirilebileceğini belirleyen temel soruların bütüncül, basitleştirici ve genel bir çerçevede irdelenmesine olanak tanıyacak bir çatı geliştirilmiştir. Genel bir düzeyde belirlenmiş olması, çatıyı sadece UKVA için değil, herhangi bir ölçekte KVA kurma ve yaşatma girişimi için de kullanılabilir kılmaktadır.

Çatının dayandığı temel ölçütler olarak kaynak tanımlama, kaynak yayınlama, kaynak yayınlama ve bulma kapsama alanı belirleme, kaynak bulma, istek ve yayın eşleştirme ve uygulama geliştirme olarak belirlenmiştir. Her bir temel ölçüt altında ayrıca çeşitli alt ölçütler belirlenmiştir. Her ne kadar tezin temel ilgi alanı WS olsa da, tezin aynı zamanda mevcut KVA gerçekleştirimlerini analiz etme hedefi bulunması nedeni ile çatı, yalnızca

WS türündeki Web kaynaklarının değil, aynı zamanda veri türündeki Web kaynaklarının her bir ölçüt bazında değerlendirilmesine yöneliktir.

Bu çalışmada geliştirilen çatı Dünya genelinde bir ilktir. Bununla birlikte, kabul görmesi ya da katkı yapması, bu noktada bir KVA kurmak durumundaki gerek teknik geliştirici ve gerekse karar vericilere yol gösterebilmesi ile mümkün olacaktır. Gerçekten de bu çatının bunu ne dereceye kadar başarabileceğinin irdelenmesi gerekir.

Çatı, mevcut durumun analizi bakımından gerekli ve yeterlidir. Gereklidir çünkü bu çalışma kapsamında çok sayıda akademik çalışma, çeşitli ulusal ve uluslararası proje ve girişim, ilgili uluslararası standart ve belirtim geliştirme kuruluşlarının ilgili standart ve belirtimleri, ticari ve açık kaynak kodlu yazılımların ilgili ürünleri, ulusal ve uluslararası düzeyde KVA gerçekleştirimlerinin detaylı incelenmesi yapılmıştır. Ancak bu incelemelerde incelenen kaynakların çoğunluğunun konuları bütüncül ve üst düzey bir bakış açısı ile sunmamaları bu çalışmayı zorlaştıran başlıca etkenlerden olmuştur. Bu sorun, WS mimarisine dayalı bir KVA gerçekleştiriminde gerek teknik gerçekleştirimcilerin ve gerekse karar vericilerin de yüzleşmek durumunda oldukları bir sorundur. Bu çalışmada geliştirilen çatı ilgili konuları bütüncül ve genel bir yaklaşımla kategorize etmiş ve anlaşılmalarını kolaylaştırmıştır. Bu çatı sayesinde anılan türlerde herhangi bir çalışmanın analizi son derece kolaylaşmıştır. Örneğin "Bulgular" bölümünde tablolar halinde verilen çalışma analizleri dakikalar içinde tamamlanmıştır.

Web teknolojileri alanındaki bir sonraki büyük hedef, "Anlamsal Web" (Semantic Web) (Berners-Lee vd., 2001) dir. Yani uygulamaların birbirleri ile "konuşabildiği" Web ortamıdır. Ancak bu hedefe ulaşılabilmesi için aşağıda belirtilen araştırmalara ihtiyaç vardır. Şu anda anlamsal Web'e yönelik çeşitli araçlar mevcuttur. Anlamsal Web yönelimli KVA gerçekleştirimleri için eksikliği mevcut araçlardan daha fazla hissedilen, ontolojiler ya da daha üst düzey bir bakışla kavramsallaştırmalardır.

Tabi ideal olan anlamsal Web yönelimli uygulamalar olduğu için çatıya göre herhangi bir çalışmanın bir diğerine göre tercih edilmesi her bir ölçüt bazında anlamsal Web gerçekleştirimine daha yakın olması ile mümkün olacaktır. Çünkü anlamsal Web gelecektir.

Çatıya göre yapılmış olan değerlendirmelerde, mevcut tüm ulusal ve uluslararası KVA gerçekleştirimlerinin "insan yönelimli" gerçekleştirimler olduğu görülmüştür. Bunlarda insan kullanıcılar hala "veri toplama" ve uygulamalarını başka bir Uygulama Programı Arayüzü (UPA) da geliştirmek durumundadır. Burada Google ile doküman

aramadaki gibi merkezi bir arama söz konudur ancak bulunan konumsal veri içerikli alternatiflerin teke indirilmesi hiçte kolay değildir. Mevcut portal ya da UKVA sunucusu gerçekleştirimlerinin geleneksel veri değişimi yoluyla sağlanabilen birlikte işlerlik düzeyinin üzerine eklediği iyileştirmeler, transfer formatı (GML) üzerindeki görüş birliği ve aranan veriyi içerme ihtimali bulunan sağlayıcıların belirli temel parametrelere göre bir portal sunucusu tarafından bulunmasıdır. Ancak burada da bütün sağlayıcılar sorgulanarak arama yapılmaktadır. Buradaki performans kaygılarını azaltmak için GOS portalda Google arama araçları kullanılmış ve performansta on kat iyileşme sağlanmıştır (Ball, 2005).

Peki, eldeki nispeten durağan araçlarla mevcut UKVA gerçekleştirimlerinden daha iyisi başarılabilir mi? sorusunun cevabı, "evet"tir. Her ne kadar WS teknolojisi henüz olgunlaşmakta olan bir teknoloji olarak kabul edilse de, nispeten durağan konuma gelmiş bileşenleri söz konusudur. Örneğin BPEL4WS servis düzenleme için kullanılabilecek durumdadır. Aynı şekilde WSDL yaygın kullanılmaktadır. Bu bakımdan (Lemmens vd., 2006) ve (Klein vd., 2006) deki çalışmalar mevcut KVA market gerçekleştirimlerinden daha iyisinin yapılabileceğinin gösterilmesine yönelik önemli adımlardır. Ancak anılan çalışmaların bu çalışmada belirlenen çatı açısından önemli eksiklikleri vardır.

Bu çalışmada ayrıca, Web kaynaklarının kataloglanmasına yönelik olarak geliştirilmiş olan çeşitli teknoloji ve tekniklerin bütüncül, basitleştirici ve genel bir çevrede anlaşılmasını sağlayacak, gerçekleştirimci ve karar vericiler için yol gösterici olacak bir dizi ölçüt belirlenmiştir. Aslında amaç başlı başına UKVA gibi geniş ölçekli bir KVA gerçekleştirimi ise, tek katalog seçimi çok önemli görülmeyebilir. Çünkü o durumda birden çok katalog kullanılması gerekecektir ki o zaman da bu katalogların birlikte işlerliği çok daha önemli olacaktır. Ancak UKVA gibi çok iddialı projelere geçişin, genellikle gözlendiği üzere, çok da eş zamanlı bir şekilde sağlanamaması nedeni ile tek tek katalog gerçekleştirimleri de önem kazanmaktadır. Diğer yandan bu ölçütler, piyasadaki ticari katalog ürünleri için, genel bir "*benchmark*" niteliği taşımaktadır. Bu ölçütler, katalog bilgi modeli, katalog arayüzleri, katalog uygulama profili ve anlamsal destektir.

Bu tez çalışması kapsamında ayrıca, WS teknolojilerine dayalı e-belediye modeli önerilerek buna yönelik uygulamalar geliştirilmiştir. Bu alanında Türkiye'deki ilk çalışmadır. Konumsal veri bazında olması ve önerdiği yeni fikirler bakımından Dünya'da da bir ilktir. Benzer yalnızca bir çalışmaya rastlanmıştır. O da (Medjahed vd., 2003) tür ki bu çalışma konumsal olmayan veriye yöneliktir. Bununla birlikte Medjahed vd. (2003), bu tez kapsamında yapılan çalışmaya göre çok daha ticari boyutta ele alınmış olan ve öyle

olması nedeniyle de, örneğin WS ler alanında en temel sorunlardan biri olarak dillendirilen, güvenlik konusunu da ele alan daha kapsamlı bir çalışmadır. WS yönelimli e-belediye uygulaması vasıtası ile önerilen yeni fikirlerden biri, özellikle e-belediye uygulamaları gibi rutin uygulamalar açısından, uygulamaların tümüyle tarayıcı üzerinden gerçekleştirilebileceğini önermesi olmuştur. Bu sayede hem işlemlerin hızı internet trafiğinden etkilenmeyecek, hem de, çok daha önemli olarak, ilave ticari yazılımlara gerek kalmayacaktır. Durum buysa ESRI, INTERGRAPH gibi büyük ticari CBS yazılım üreticileri kendilerine nasıl bir yol çizmelidir sorusu ise tartışma ve araştırma konusudur. Bu da bu çalışmanın gündeme getirdiği sorulardan biridir.

Bu çalışmada ayrıca "servis düzenleme" yöntemleri için bir sınıflandırma önerilmiştir. Bunun da çok önemli bir katkı yapması beklenmektedir. Çünkü bugün literatürde ve ilgili diğer çalışmalarda servis düzenleme yöntemlerinin sınıflandırılması konusunda bir görüş birliği yoktur. Bu tez çalışması kapsamında böyle bir sınıflandırma önerilmiştir.

Çalışmanın hedefi gereği kapsamın çok geniş olması nedeniyle, bu çalışmada çok geniş bir literatür taraması yapılmıştır. Taranan çalışmalardan çok önemli görülenlerinin irdelenmesi tezin tek bir bölümünde yapılmamış, tez anlatımı içerisinde belirli konular altında yer almıştır.

Bu tez çalışması kapsamanda ayrıca, aşağıda sıralanan gelecek yönelimleri ve araştırma alanları tespit edilmiştir. Bu alanlarda gerçekleştirilecek araştırmalar, Dünya genelinde bu çok aktif araştırma alanında Ülkemizin her anlamdaki rekabet gücüne katkı yapacaktır.

Konumsal Web servislerine yönelik ontolojilerin tanımlanması en öncelikli araştırma alanlarından biridir. Çünkü anlamsal Web de veri ya da servis tanımlamalarının ontolojiler vasıtası ile yapılması gerekmektedir. Bu ihtiyacı vurgulayan (Dolbear vd., 2005), (Kuhn, 2005) ve (Klein vd., 2006) gibi çalışmalar olmakla birlikte bunun nasıl yapılacağı ve nasıl yapılması gerektiği konusunda (Kuhn, 2003b) dışında bilgimiz dahilinde olan bir çalışma mevcut değildir. Bu açıdan Klein vd. (2006) tarafından işaret edilen bir sorun, konumsal Web servislerine yönelik uygulama alanlarının (domain) çok geniş olması ve bu alanlarının birbirleri ile olan ilişkileri nedeni ile alan sınırlarının belirlenmesindeki güçlüktür.

Diğer önemli konu, anlamsal tanımlamada bugün en yaygın kullanılan teknik olan Tanımlama Mantığının (TM), Zadeh (2003), Kuhn (2003b) ve Sheth vd. (2005) tarafından işaret edilen yetersizlikleridir. Örneğin Kuhn (2003b), TM'nin eylemleri modellemedeki

eksikliğine işaret etmiştir. Kuhn (2003b), "cebrik modeller"in bu açıdan TM den daha doğal temsillere olanak tanıdığına işaret etmektedir. Yine TM yetersizlikleri bağlamında Zadeh (2003) ise artık "kelimelerle hesaplama" (computing with words) gibi yeni modellere ihtiyaç olduğunu vurgulamaktadır. Bu durumda geleceğin ihtiyaçlarına cevap verecek anlamsal tanımlamalar ya da bir üst maddede ontoloji tanımlama olarak belirtilen kavramsallaştırmalar (conceptualizations), hangi bilim, teknoloji ve araçlarla yapılmalıdır? Bu soru, anlamsal Web'in geleceği ve başarısı açısından araştırılması acil önem arz eden diğer bir konudur.

Anlamsal Örtü Ağları (Semantic Overlay Networks) çok önemli potansiyeli olan araştırma alanlarından biridir. Bu ağlarda anlamsal benzerliğin nasıl tanımlanacağı ya da eşlerin nasıl gruplandırılacağı konusu en önemli araştırma alanlarından biridir.

Merkezi kontrollü ağlara göre daha ölçeklenebilir, ucuz ve sağlam (robust) olarak nitelenen P2P ağlar, WS ile birlikte kullanılma potansiyeli zaten yaygın biçimde vurgulanan, ancak henüz net bir biçimde ortaya konamamış diğer bir araştırma alanıdır. Bu ağlarda kapsama alanı belirleme ve istek yönlendirme konularında daha çok araştırmaya ihtiyaç vardır.

Konumsal bilgi işleme ve yönetimi, bilgisayar bilimleri ve hatta Kuhn'un (2003b) son derece ufuk açıcı çalışmasında işaret ettiği "kavramsal (cognitive) dilbilimi" ve diğer ilgili alanlarda Ülkemizdeki lisans ve yüksek lisans programlarında yukarıda değinilen geleceğin araştırma alanlarına yönelik açılımlar ve beslemeler, Ülkemizin her anlamdaki rekabet gücü bakımından kaçınılmazdır. Bu konulardaki eğitim sistemimizin eksiklikleri bu çalışmanın yürütülmesi sırasında çok yoğun bir şekilde hissedilmiştir.

Bu tez çalışmasında çok geniş bir literatür, OGC, ISO, W3C ve OASIS gibi standart geliştirme kuruluşlarının belirtim ve standartları, yazılım firmalarının ticari ürünleri ve nihayet GOS gibi, ilgili market gerçekleştirimleri detaylı olarak irdelenmiştir. İncelenen bu çalışmaların çoğunun temel felsefe ve kritik önem arz eden konularda, konuyu net bir şekilde ortaya koyamamaları, tez çalışmasını oldukça zorlaştırmıştır. O nedenle özellikle akademik çalışmalarda konunun açık ve basitleştirici bir perspektiften ortaya konması, çalışmanın katkısı ve kabul görmesi açısından büyük önem taşımaktadır. Son yıllarda yabancı literatürde sıkça gözlenen bu sorun, "Geoinformatics" dergisindeki köşesinde[1] Werner Kuhn tarafından da dile getirilmiştir. Kuhn, sorunun çözümünün Einstein'in "şeyleri mümkün olduğunca basitleştirin fakat daha basit (olduğundan daha basit)

[1] Geoinformatics dergisi Ocak 2004 sayısı

yapmayın”[2] özdeyişinde olduğuna işaret etmiştir. Einstein’in bize göre muhteşem ve hayatın her alanında yol gösterici olabilecek olan bu felsefesi, bütün araştırmacıların rehberi olmalıdır.

[2] “Make things as simple as possible but not simpler”

6. KAYNAKLAR

Aberer, K., Alima, L. O., Ghodsi, A., Girdzijauskas, S., Haridi, S. ve Hauswirth, M., 2005. The essence of P2P: A reference architecture for overlay Networks, Fifth IEEE Interational Conference on Peer-to-Peer Computing, Konstanz, Germany.

Alameh, N., 2003. Chaining Geographic Information Web Services, IEEE Internet Computing, 7, 5, 22–29.

Arpınar, I. B., Zhang, R., Aleman-Meza, B. ve Maduko, A., 2005. Ontology-driven Web services composition platform, Information Systems and E-Business Management, 3, 2, 175 – 199.

Baader, F., McGuinness, D. L., Nardi, D. ve Patel-Schneider, P., 2003. The Description Logic Handbook: Theory, implementation, and applications, Cambridge University Pres, Cambridge, UK.

Ball, M., 2005. GeoSpatial One-Stop Moves to Phase Two Weaving a Tapestry of National Geospatial Data Coverage, GeoWorld, http://www.geoplace.com/uploads/ FeatureArticle/0505sdi.asp, 29 Mayıs 2006.

Batini, C., Ceri, S. ve Navathe, S., 1992. Conceptual Database Design: An Entity-Relationship Approach, The Benjamin/Cummings Publishing Company, Inc., California, USA.

Bernard, L., Einspanier, U., Lutz, M. ve Portele, C., 2003. Interoperability in GI Service Chains – The Way Forward, 6th AGILE Conference on Geographic Information Science, Lyon, France.

Bernard, L., Kanellopoulos, I., Annoni, A. ve Smits, P., 2005. Europen geoportal-one stop towards the establishment of a Europen Spatial Data Infrastructure, Computer, Environment and Urban Systems, 29, 15-31.

Berners-Lee, T., Hendler, J. ve Lassila, O., 2001. The Semantic Web, Scientific American, 284, 4, 34–43.

Blow, M., Goland, Y., Kloppmann, M., Leymann, F., Pfau, G., Roller, D. ve Rowley, M., 2004. BPELJ: BPEL for Java Technology, A Joint White Paper by BEA and IBM, http://www-128.ibm.com/developerworks/library/specification/ws-bpelj/, 9 Ocak 2005.

Bowers, S. ve Ludascher, B., 2004. An ontology-driven framework for data Transformation in scientific workflows, In International Workshop on Data Integration in the Life Sciences (DILS'04), Leipzig, Germany.

Brin, S. ve Page, L., 2000. The Anatomy of a Large-Scale Hypertextual Web Search Engine, http://www-db.stanford.edu/~backrub/google.html, 5 Nisan 2003.

Cape Clear, 2003. Cape Clear 4 User's Guide, http://www.capeclear.com, 25 Mayıs 2003.

Cardoso, J. ve Sheth, A., 2003. Semantic e-Workflow Composition, Journal of Intelligent Information Systems, 21, 3, 191-225.

Castano, S., Ferrara, A., Montanelli, S., Pagani, E. ve Rossi, G. P., 2003. Ontology-addressable contents in P2P Networks, First WWW International Workshop on Semantics in Peer-to-Peer and Grid Computing, Budapest, Hungary.

Cerami, E., 2002. Web Services Essentials, O'Reilly & Associates, Inc., Sebastopol, USA.

Colan, M., 2004. Service-Oriented Architecture expands the vision of Web services, Part 1: Characteristics of Service-Oriented Architecture, http://www-128.ibm.com/developerworks/webservices/library/ws-soaintro.html, 13 Kasım 2004.

Cömert, Ç. ve Banger, G., 1995. Türkiye için Ulusal Konumsal Veri Altyapısı, Devlet İstatistik Enstitüsü Araştırma Sempozyumu, Ankara, Bildiriler Kitabı, 6-10.

Cömert, Ç., 1996. Ulusal Konumsal Veri Altyapısı için Veri Değişim Standardının Belirlenmesi, Doktora Tezi, KTÜ Fen Bilimleri Enstitüsü, Trabzon.

Cömert, Ç. ve Banger, G., 1996. Ulusal Konumsal Veri Altyapısı, 2. Coğrafi Bilgi Sistemleri Sempozyumu, İstanbul, Bildiriler Kitabı, 49-61.

Cömert, Ç., 1998. Ulusal Konumsal Veri Altyapısı Üzerine, HKMO Bülteni, 84-86.

Cömert, Ç. ve Akıncı, H., 2002. Application Development in an interoperable GIS environment: A new system for real estate taxation in Turkey, 3rd International Symposium on Remote Sensing of Urban Areas, İstanbul, Turkey.

Cömert, Ç. ve Akıncı, H., 2003. Emlak Vergilendirmede yeni bir sistem EVBİS: Emlak Vergisi Bilgi Sitemi, 9. Türkiye Harita Bilimsel ve Teknik Kurultayı, Ankara, Bildiriler Kitabı, 781-790.

Cömert, Ç. ve Akıncı, H., 2005. Architectures for Geospatial Web Services: Issues and Implementations, GML And Geo-Spatial Web Services Conference, Vancouver, British Columbia, Canada.

Crespo, A. ve Garcia-Molina, H., 2002. Routing Indices for Peer-to-Peer Systems, 22nd International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria.

Crespo, A. ve Garcia-Molina, H., 2003. Semantic Overlay Networks for P2P Systems, Stanford Technical Report, http://www-db.stanford.edu/~crespo/publications/op2p.pdf, 21 Şubat 2006.

Di, L., Zhao, P., Yang, W. ve Yue, P., 2006. Ontology-driven Automatic Geospatial-Processing Modeling based on Web-service Chaining, Sixth Annual NASA Earth Science Technology Conference (ESTC2006), Adelphi, Maryland, USA.

Dogaç, A., Cingil, İ., Laleci, G. ve Kabak, Y., 2002a. Improving the Functionality of UDDI Registries through Web Service Semantics, 3rd VLDB Workshop on Technologies for E-Services (TES-02), Hong Kong, China.

Dogaç, A., Laleci, G., Kabak, Y. ve Cingil, İ., 2002b. Exploiting Web Service Semantics: Taxonomies vs. Ontologies, IEEE Data Engineering Bulletin, 25, 4, 10–16.

Dogac, A., Kabak, Y. ve Laleci, G. B., 2003. A Semantic-Based Web Service Composition Facility for ebXML Registries, 9th International Conference of Concurrent Enterprising, Dipoli Congress Center, Espoo, Finland.

Dogac, A., 2003. A Tutorial on Exploiting Semantic of Web Services through ebXML Registries, eChallenges 2003, Bologna, Italy.

Dogac, A., Kabak, Y. ve Laleci, G. B., 2004. Enriching ebXML Registries with OWL Ontologies for Efficient Service Discovery, 14th International Workshop on Research Issues on Data Engineering: Web Services for E-Commerce and E-Government Applications (RIDE'04), Boston, USA.

Dogac, A., Kabak, Y., Laleci, G. B., Mattocks, C., Najmi, F. ve Pollock, J., 2005. Enhancing ebXML Registries to Make them OWL Aware, Distributed and Parallel Databases, 18, 9–36.

Dolbear, C., Goodwin, J., Mizen, H. ve Ritchie, J., 2005. Semantic Interoperability between topographic data and a flood defence ontology, http://www.ordnancesurvey.co.uk/partnership/research/publication/docs/2005/Semantic Interoperability_CDolbear_sem.pdf, 22 Mayıs 2006.

Doyle, A. ve Reed, C., 2001. Introduction to OGC Web Services, OGC Interoperability Program White Paper, http://ip.opengis.org/ows/010526_OWSWhitepaper.doc, 9 Kasım 2005

Dustdar, S. ve Treiber, M., 2005. A View Based Analysis on Web Service Registries, Distributed and Parallel Databases, 18, 2, 147-171.

Ekinci, A., 1996. İstanbul Kent Bilgi Sistemi, Sarıyer pilot projesi yöneticisi, Kişisel İletişim.

Elenius, D. ve Ingmarsson, M., 2004. Ontology-based Service Discovery in P2P Networks , First International Workshop on Peer-to-Peer Knowledge Management, Boston, Massachusetts, USA.

Erol, K., Hendler, J. ve Nau, D. S., 1995. Semantics for Hierarchical Task-Network Planning, Technical Research Report CS-TR-3239, Computer Science Department, Institute for Systems Research, University of Maryland.

ESRI, 2003. Introducing ESRI's GIS Portal Toolkit, ArcNews Winter 2003-2004 Issue, http://www.esri.com/news/arcnews/winter0304articles/introducing.html, 5 Mayıs 2006.

ESRI, 2004. GIS Portal Technology, ESRI White Paper, http://www.esri.com/library/whitepapers/pdfs/gisportal.pdf, 5 Mayıs 2006.

Fensel, D., Bruijn, J., Lara, R., Arroyo, S., Gomez, J.M. ve Han, K., 2005. A Unified Semantic Web Services Architecture based on WSMF and UPML, International Journal of Web Engineering and Technology, 2, 3, 148-180.

FGDC, 1998. Content Standard for Digital Geospatial Metadata, Federal Geographic Data Committtee, FGDC-STD-001-1998.

FGDC, 2000. Z39.50 Application Profile for Geospatial Metadata or "GEO", Version 2.2, http://www.blueangeltech.com/standards/GeoProfile/geo22.htm, 5 Mayıs 2006.

FGDC, 2005a. The National Geospatial Data Clearinghouse, http://www.fgdc.gov/library/factsheets/documents/chouse.pdf, 5 Mayıs 2006.

FGDC, 2005b. The National Spatial Data Infrastructure, http://www.fgdc.gov/nsdi/library/factsheets/documents/nsdi.pdf, 5 Mayıs 2006

FGDC, 2005c. Geospatial One-Stop: Encouraging partnerships to enhance access to geospatial information, http://www.fgdc.gov/library/factsheets/documents/gos.pdf, 5 Mayıs 2006.

Fujii, K. ve Suda, T., 2004. Dynamic Service Composition Using Semantic Information, 2nd International Conference on Service Oriented Computing (ICSOC'04), New York, USA.

Grau, B. C., Parsia, B. ve Sirin, E., 2004. Working with multiple ontologies on the semantic web, Third International Semantic Web Conference (ISWC2004), Lecture Notes in Computer Science, 3298, 620-634.

Granell, C., Poveda, J. ve Gould, M., 2004. Incremental Composition of Geographic Web Services: an Emergency Management Context, 7th AGILE Conference on Geographic Information Science, Heraklion, Greece.

Hendler, J., Berners-Lee, T. ve Miller, E., 2002. Integrating Applications on the Semantic Web, Journal of the Institute of Electrical Engineers of Japan, 122, 10, 676-680.

Hoschek, W., 2002. Peer-to-Peer Grid Databases for Web Service Discovery, CERN IT Division, European Organization for Nuclear Research, http://dsd.lbl.gov/~hoschek/publications/p2pGridDB.pdf, 17 Mart 2006.

INSPIRE, 2005. INSPIRE Web sitesi, http://www.ec-gis.org/inspire/, 18 Ekim 2005.

ISO, 2001. Geographic information — Services, ISO/DIS 19119, ISO TC 211/WG 4.

ISO, 2003. Geographic information — Metadata, ISO/FDIS 19115, ISO TC 211.

Juric, M. B., Mathew, B. ve Sarang, P., 2004. Business Process Execution Language for Web Services, Packt Publishing, Birmingham, UK.

Karnstedt, M., Hose, K. ve Sattler, U., 2004, Distributed Query Processing in P2P Systems with incomplete schema information, In Proc. Of the Int. Conf. CAiSE (Workshop DIWeb'04), Riga, Latvia.

Klien, E., Lutz, M. ve Kuhn, W., 2006. Ontology-based discovery of geographic information services—An application in disaster management, Computers, Environment and Urban Systems, 30, 102–123.

Knowledge Web, 2005. Knowledge Web FP6–507482, Research Deliverables, Deliverable 1.2.10, http://knowledgeweb.semanticweb.org/, 22 Mart 2006.

Kuhn, W., 2003a. Semantic Reference Systems, International Journal of Geographical Information Science, 17, 5, 405–409.

Kuhn, W., 2003b, Why Information Science needs Cognitive Semantics and what it has to offer in return, DRAFT Version 1.0, Meaning and Computation Laboratory, Department of Computer Science and Engineering, University of California, http://musil.uni-muenster.de/documents/WhyCogLingv1.pdf, 14 Aralık 2005.

Kuhn, W., 2005. Geospatial Semantics: Why, of What, and How?, Lecture notes in computer science, Journal on data semantics III, 3534, 1-24.

Lemmens, R., Granell, C., Wytzisk, A., de By, R., Gould, M. ve Oosterom, P. V., 2006. Semantic and syntactic service description at work in geo-service chaining, 9th AGILE Conference on Geographic Information Science, Visegrad, Hungary.

Lutz, M. ve Klien, E., 2006. Ontology-based retrieval of geographic information, International Journal of Geographical Information Science, 20, 3, 233–260.

Mahmoud, Q. H., 2005. Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI), http://java.sun.com/developer/technicalArticles/WebServices/soa/, 8 Ocak 2006

Majithia, S., Walker, D. W. ve Gray, W. A., 2004. Automated Web Service Composition Using Semantic Web Technologies, First International Conference on Autonomic Computing (ICAC'04), New York, USA

Manes, A. T., 2002. Standardizing Web Services, Upgrade Magazine, www.siia.net, http://www.siia.net/upgrade/archive/0809_02/manes.pdf, 22-25.

McGovern, J., Tyagi, S., Stevens, M. ve Mathew S., 2003. Java Web Services Architecture, Morgan Kaufmann, San Francisco, USA.

Medjahed, B, Rezgui, A., Bouguettaya, A. ve Ouzzani, M., 2003. Infrastructure for E-Government Web Services. IEEE Internet Computing, 7, 1, 58-65.

Meyer, B., 1988. Object-Oriented Software Construction, First Edition, Prentice-Hall, Inc. Upper Saddle River, NJ, USA.

Nagappan, R., Skoczylas, R. ve Sriganesh, R. P., 2003. Developing Java Web Services, Wiley Publishing Inc., Indiana, USA.

Najmi, F., 2005. Registry Capability Comparison Matrix, http://ebxmlrr.sourceforge.net/tmp/Registry_Capability_Matrix.html, 26 Aralık 2005.

Nau, D., Au, T. C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D. ve Yaman, F., 2003. SHOP2: An HTN planning system, Journal of Artificial Intelligence Research, 20, 379–404.

Nau, D., Au, T. C., Ilghami, O., Kuter, U., Munoz-Avila, H., Murdock, J. W., Wu, D. ve Yaman, F., 2005. Applications of SHOP and SHOP2, IEEE Intelligent Systems, 20, 2, 34-41.

Newcomer, E., 2002. Understanding Web Services XML, WSDL, SOAP and UDDI, Addison-Wesley, Indianapolis, USA.

NISO, 2003. Information Retrieval (Z39.50): Application Service Definition and Protocol Specification, ANSI/NISO Z39.50–2003, Bethesda, Maryland, U.S.A.

OASIS, 2004. Universal Description, Discovery and Integration (UDDI), Version 3.0.2, http://uddi.org/pubs/uddi-v3.0.2-20041019.htm, 28 Ağustos 2005.

OASIS, 2005a. ebXML Registry Services and Protocols, Version 3.0, OASIS Standard.

OASIS, 2005b. ebXML Registry Information Model, Version 3.0, OASIS Standard.

OASIS, 2006. Web Services Business Process Execution Language Version 2.0, Committee Draft, http://www.oasis-open.org/committees/download.php/18714/wsbpel-specification-draft-May17.htm, 29 Haziran 2006.

OGC, 1999. Catalogue Interface Implementation Specification, Version: 1.0, OGC 00–034, OGC Implementation Specification.

OGC, 2001. Web Map Service (WMS) Implementation Specification, Version 1.1.1, OGC 01-068r2, OGC Implementation Specification.

OGC, 2002. OWS–1 Registry Service (WRS), OGC 02-050r5, Version: 0.7.2.

OGC, 2003a. GOS-Portal Implementation Architecture, OpenGIS Interoperability Program Report, OGC 03-xxx, Version 0.1.

OGC, 2003b. Web Coverage Service (WCS), Version 1.0.0, OGC 03-065r6, OGC Implementation Specification.

OGC, 2003c. OWS 1.2 SOAP Experiment Report, OGC 03-014, Version: 0.8, OGC Discussion Paper.

OGC, 2003d. Geography Markup Language (GML) Implementation Specification, OGC 02-023r4, Version: 3.0, OGC Implementation Specification.

OGC, 2004a. Geospatial Portal Reference Architecture, A Community Guide to Implementing Standards-Based Geospatial Portals, OpenGIS Discussion Paper, OGC 04-039, Version 0.2.

OGC, 2004b. OGC Catalogue Services – ebRIM (ISO/TS 15000–3) profile of CSW, OGC 04-017r1, Version: 0.9.1, OGC Discussion Paper.

OGC, 2004c. Web Map Service (WMS), Version 1.3.0, OGC 04-024, OGC Implementation Specification.

OGC, 2004d. OWS 2 Common Architecture: WSDL SOAP UDDI, OGC 04-060r1, Version: 1.0.0, OGC Discussion Paper.

OGC, 2005a. OpenGIS Catalogue Services Specification 2.0 - ISO19115/ISO19119 Application Profile for CSW 2.0, OGC 04-038r2, Version: 0.9.3, OGC Recommendation Paper.

OGC, 2005b. OGC Catalogue Services Specification, OGC Implementation Specification, Version 2.0 with corregendum (OGC 04-021r3), Eds.: D. Nebert, A. Whiteside, Open Geospatial Consortium Inc.

OGC, 2005c. Web Feature Service (WFS) Implementation Specification, Version 1.1.0, OGC 04-094, OGC Implementation Specification.

OGC, 2005d. OGC Web Services Common Specification, Version 1.0.0, OGC 05-008, OGC Implementation Specification.

OGC, 2005e. Filter Encoding Implementation Specification, OGC 04–095, version 1.1.0, OGC Implementation Specification.

OGC, 2006. OGC Catalogue Services – ebRIM (ISO/TS 15000-3) profile of CSW, OGC 05-025r3, Version: 1.0.0, OGC Application Profile.

OMB, 1992. Office of Management and Budget (OMB), Circular No. A-16, http://www.whitehouse.gov/omb/, 15 Kasım 2003.

OMG, 2002. The Common Object Request Broker Architecture: Core Specification, Version 3.0, formal/02-11-01, http://www.omg.org/technology/documents/corba_spec_catalog.htm, 22 Ocak 2003.

OMG, 2005. Unified Modeling Language: Infrastructure, version 2.0, formal/05-07-05, http://www.omg.org/cgi-bin/doc?formal/05-07-05, 29 Haziran 2006.

Orfali, R. ve Harkey, D., 1998. Client/Server Programming With Java and CORBA, John Willey & Sons, Inc., New York, USA.

Paik, H., Banatallah, B. ve Toumani, F., 2004. WS-CatalogNet: Building Peer-to-Peer e-catalog, Lecture notes in computer science, 3055, 256-269.

Paolucci, M., Kawamura, T., Payne, T. ve Sycara, K., 2002a. Importing the Semantic Web in UDDI, International Workshop on Web Services, E-Business, and the Semantic Web (WES 2002), Toronto, Canada.

Paolucci, M., Kawamura, T., Payne, T. ve Sycara, K., 2002b. Semantic Matching of Web Services Capabilities, The Semantic Web - ISWC 2002: First International Semantic Web Conference, Sardinia, Italy.

Paolucci, M., Soudry, J., Srinivasan, N. ve Sycara, K., 2004. A Broker for OWL-S Web Services, First International Semantic Web Services Symposium, Southampton, UK.

Peltz, C., 2003. Web services orchestration A review of emerging technologies, tools, and standards, Hewlett Packard Company.

Peer, J., 2005. Web Service Composition as AI Planning - a Survey, Technical Report, University of St.Gallen, Switzerland.

Probst, F. ve Lutz, M., 2004. Giving Meaning to GI Web Service Descriptions, 7th AGILE conference on Geographic Information Science, Heraklion, Greece.

Rao, J. ve Su, X., 2004. A Survey of Automated Web Service Composition Methods, First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), San Diego, California, USA.

Ratnasamy, S., Francis, P., Handley, M., Karp, R. ve Shenker, S., 2001. A scalable content addressable network, ACM SIGCOMM Conference, San Diego, CA, USA.

Russel, S. ve Norvig, P., 1995. Artificial Intelligence: A Modern Approach, Prentice-Hall Inc, London, UK.

Ryman, A., 2000. Understanding Web Services, http://www7b.software.ibm.com/wsdd/ library/techarticles/0307_ryman/ ryman.html, 27 Haziran 2003.

Saltor, F., Castellanos, M. ve Garcia-Solaco, M., 1991. Suitability of data models as canonical models for federated databases, SIGMOD Record, 20, 4, 45–48.

Schmelzer, R., 2006. Solving the service granularity challenge, Service Orientation Research, Advisory, and Expertise, http://www.zapthink.com/ report.html?id=ZAPFLASH-200639, 12 Haziran 2006.

Schutzberg, A. ve Francica, J., 2005. The Technology Behind the New Geodata.gov and the Non-Technology Challenges Ahead, Directions Magazine, http://www.directionsmag.com/article.php?article_id=784, 5 Mayıs 2006.

Schweitzer, P. N., 2003. National Spatial Data Clearinghouse: The real story, Digital Mapping Techniques '03 Workshop, Millersville, Pennsylvania, USA.

ShaikhAli, A., 2003. UDDIe: An Extended Registry for Web Services, Proceedings of the Service Oriented Computing: Models, Architectures and Applications, SAINT-2003 IEEE Computer Society Pres, Orlando Florida, USA.

Sheth, A., Ramakrishnan, C. ve Thomas, C., 2005. Semantics for the Semantic Web: The implicit, the formal and the powerful, International Journal on Semantic Web and Information Systems, 1, 1, 1–18.

Sirin, E., Hendler, J. ve Parsia, B., 2003a. Semi-automatic Composition of Web Services Using Semantic Descriptions, Web Services: Modeling, Architecture and Infrastructure Workshop in International Conference on Enterprise Information Systems (ICEIS 2003), Angers, France.

Sirin, E., Hendler, J. ve Parsia, B., 2003b. Interactive Composition of Semantic Web Services, 12th International World Wide Web Conference, Budapest, Hungary.

Sirin, E., Parsia, B. ve Hendler, J., 2004a. Filtering ad Selecting Semantic Web Services with Interactive Composition Techniques, IEEE Intelligent Systems, 19, 4, 42-49

Sirin, E., Parsia, B., Wu, D., Hendler, J. ve Nau, D., 2004b. HTN Planning for Web Service Composition Using SHOP2, Journal of Web Semantics, 1, 4, 377–396.

Sirin, E., Parsia, B. ve Hendler, J., 2005. Template-based Composition of Semantic Web Services, AAAI Fall Symposium on Agents and the Semantic Web, Virginia, USA.

Sivashanmugam, K., Miller, J. A., Sheth, A. P. ve Verma, K., 2004. Framework for Semantic Web Process Composition, International Journal of Electronic Commmerce, 9, 2, 71-106.

Srinivasan, N., Paolucci, M. ve Sycara, K., 2004. Adding OWL-S to UDDI, implementation and throughput, First International Workshop on Semantic Web Services and Web Process Composition, SWSWPC 2004, San Diego, CA, USA.

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F. ve Balakrishnan, H., 2001. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, ACM SIGCOMM 2001 Conference, San Diego, California, USA.

SUN, 2000. Remote Method Invocation: Introduction, http://java.sun.com/developer/onlineTraining/rmi/, 23 Eylül 2002.

SUN, 2003. JSR-00168 Java Portlet Specification, Version 1.0, http://jcp.org/aboutJava/communityprocess/final/jsr168/index.html, 5 Mayıs 2006.

Şahin, N., 2003. E-belediye için Web servisleri tasarımı, Yüksek Lisans Tezi, KTÜ Fen Bilimleri Enstitüsü, Trabzon.

Tang, C., Xu, Z. ve Dwarkadas, S., 2003. Peer to Peer Information Retrieval Using Self Organizing Semantic Overlay Networks, SIGCOMM'03, Karlsruhe, Germany.

UDDI.org, 2000. UDDI Technical White Paper, Ariba Inc., IBM Corp., and Microsoft Corp., http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf, 24 Temmuz 2003.

URL-1, http://www.geo-one-stop.gov/metadata/CreatePublishMeta.pdf, Creating and Publishing Metadata in Support of Geospatial One-Stop and the NSDI, 3 Mart 2006.

URL-2, http://www.fgdc.gov/dataandservices/implementation, Clearinghouse Implementation, 3 Mart 2006.

Vinoski, S., 2002. Web Services Interaction Models, Part 1: Current Practice, IEEE Internet Computing, 6, 3, 89–91.

Voges, U., 2004. Cataloging Geo-Resources by Metadata, GeoInformatics, 4, 2004, 36–39.

Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H. ve Hübner, S., 2001, Ontology-based integration of information - a survey of existing approaches, In IJCAI-01 Workshop: Ontologies and Information Sharing, Seattle, WA, USA.

W3C, 1998. Extensible Markup Language (XML) 1.0, W3C Recommendation, REC-xml-19980210, http://www.w3.org/TR/2004/REC-xml-20040204, 15 Ocak 2000.

W3C, 2000. Simple Object Access Protocol (SOAP) 1.1, W3C Note, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/, 29 Haziran 2004.

W3C, 2001. Web Services Description Language (WSDL) 1.1, W3C Note, http://www.w3.org/TR/2001/NOTE-wsdl-20010315, 30 Nisan 2003.

W3C, 2002. Web Services Description Requirements, W3C Working Draft, http://www.w3.org/TR/ws-desc-reqs/, 11 Kasım 2005.

W3C, 2003a. Simple Object Access Protocol (SOAP) 1.2, W3C Recommendation, http://www.w3.org/TR/2003/REC-soap12-part1-20030624/, 18 Kasım 2004.

W3C, 2003b. Scalable Vector Graphics (SVG) 1.1 Specification, W3C Recommendation, http://www.w3.org/TR/2003/REC-SVG11-20030114/, 29 Ocak 2003.

W3C, 2004a. XML Schema Part 2: Datatypes Second Edition, W3C Recommendation, http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/, 11 Mart 2005.

W3C, 2004b. OWL Web Ontology Language Reference, W3C Recommendation, http://www.w3.org/TR/owl-ref/, 15 Şubat 2006.

W3C, 2004c. OWL-S: Semantic Markup for Web Services, W3C Member Submission, http://www.w3.org/Submission/OWL-S/, 15 Şubat 2006.

W3C, 2004d. Web Services Architecture, W3C Working Group Note, http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/, 11 Kasım 2005.

W3C, 2004e. Web Services Architecture Requirements, W3C Working Group Note, http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/, 11 Kasım 2005.

W3C, 2004f. Web Services Choreography Description Language, W3C Working Draft, http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/, 8 Nisan 2006.

W3C, 2006. Web Services Description Language (WSDL) 2.0 Part 0: Primer, W3C Candidate Recommendation, http://www.w3.org/TR/wsdl20-primer/, 10 Haziran 2006.

Weerawarana, S., Curbera, F., Leymann, F., Storey, T. ve Ferguson, D. F., 2005. Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More, Prentice Hall Ptr, USA.

Wiederhold, G., 1992. Mediators in the Architecture of Future Information Systems, IEEE Computer, 25, 3, 38-49.

Wiederhold, G., 1999. Mediation to Deal with Heterogeneous Data Sources, Second International Conference on Interoperating Geographic Information Systems, Zurich, Switzerland .

World Bank, 2002. E-government handbook for developing countries, Center for Democracy and Technology, http://www.cdt.org/egov/handbook/2002-11-14egovhandbook.pdf, 22 Ocak 2003.

Wu, D., Sirin, E., Hendler, J., Nau, D. ve Parsia, B., 2003. Automatic Web Services Composition Using SHOP2, 13th International Conference on Automated Planning and Scheduling (ICAPS 2003), Trento, Italy.

Yu, L., 2006. Understanding UDDI's tModel, http://www.codeproject.com/soap/understandingTModels.asp, 17 Şubat 2006.

Zadeh, L., 2003. From Search engines to question answering systems: The need for new tools, 12th IEEE international Conference on Fuzzy Systems, Lecture notes in computer science, 2663, 15-17.

7. EKLER

Ek 1. WSDL Dosyası

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
   name="ToplamaWS"
   targetNamespace="http://jeodeziprog/ToplamaWS.wsdl"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://jeodeziprog/ToplamaWS.wsdl"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsd1="http://jeodeziprog/ToplamaWS.xsd1">
   <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
     Created using Cape Clear Studio WSDL Editor - http://www.capeclear.com
   </wsdl:documentation>
   <wsdl:types>
      <xsd:schema
         targetNamespace="http://jeodeziprog/ToplamaWS.xsd1"
         xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema"
         xmlns:xsd1="http://jeodeziprog/ToplamaWS.xsd1"> </xsd:schema>
   </wsdl:types>
   <wsdl:message name="ToplamaResponse">
      <wsdl:part name="result" type="xsd:float"/>
   </wsdl:message>
   <wsdl:message name="ToplamaRequest">
      <wsdl:part name="number1" type="xsd:float"/>
      <wsdl:part name="number2" type="xsd:float"/>
   </wsdl:message>
   <wsdl:portType name="ToplamaWSPortType">
      <wsdl:operation name="topla">
         <wsdl:input message="tns:ToplamaRequest"/>
         <wsdl:output message="tns:ToplamaResponse"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="ToplamaWSBinding" type="tns:ToplamaWSPortType">
      <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="topla">
         <soap:operation
           soapAction="capeconnect:ToplamaWS:ToplamaWSPortType#topla"/>
         <wsdl:input>
            <soap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               namespace="http://jeodeziprog/ToplamaWS/binding"
               parts="number2 number1"
               use="encoded"/>
```

Ek 1'in devamı

```
            </wsdl:input>
            <wsdl:output>
               <soap:body
                  encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                  namespace="http://jeodeziprog/ToplamaWS/binding"
                  parts="result"
                  use="encoded"/>
            </wsdl:output>
         </wsdl:operation>
      </wsdl:binding>
      <wsdl:service name="ToplamaWS">
         <wsdl:port binding="tns:ToplamaWSBinding" name="ToplamaWSPort">
            <soap:address location="http://jeodeziprog:8000/ccx/ToplamaWS"/>
         </wsdl:port>
      </wsdl:service>
</wsdl:definitions>
```

Ek 2. BPEL İşlem Dosyası

```
<?xml version="1.0"?>
<bpws:process name="CalculatorBPEL"
targetNamespace=http://jeodeziprog.ktu.edu.tr/BusinessServices/CalculatorBPEL
xmlns:tns=http://jeodeziprog.ktu.edu.tr/BusinessServices/CalculatorBPEL
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:ns="http://jeodeziprog/ToplamaWS.wsdl"
xmlns:ns1="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:ns2="http://jeodeziprog/ToplamaWS.xsd1"
xmlns:ns3="http://schemas.xmlsoap.org/wsdl/"
xmlns:ns4="http://jeodeziprog/ToplamaWS/ToplamaWSPartnerLinks.wsdl"
xmlns:ns5="http://193.140.168.115/CikarmaWS.wsdl"
xmlns:ns6="http://193.140.168.115/CikarmaWS.xsd1"
xmlns:ns7="http://193.140.168.115/CikarmaWS/CikarmaWSPartnerLinks.wsdl"
xmlns:ns8="http://jeodeziprog.ktu.edu.tr/CalculatorBPELWS.wsdl"
xmlns:ns9="http://jeodeziprog.ktu.edu.tr/CalculatorBPELWS.xsd1"
xmlns:ns10="http://jeodeziprog.ktu.edu.tr/CalculatorBPELWS/CalculatorBPELWSPartner
Links.wsdl">

<bpws:import location="ToplamaWSPartnerLinks.wsdl"
   namespace="http://jeodeziprog/ToplamaWS/ToplamaWSPartnerLinks.wsdl"
   importType="http://schemas.xmlsoap.org/wsdl/">
</bpws:import>
<bpws:import location="CikarmaWSPartnerLinks.wsdl"
   namespace="http://193.140.168.115/CikarmaWS/CikarmaWSPartnerLinks.wsdl"
   importType="http://schemas.xmlsoap.org/wsdl/">
</bpws:import>
<bpws:import location="CalculatorBPELWSPartnerLinks.wsdl"
namespace="http://jeodeziprog.ktu.edu.tr/CalculatorBPELWS/CalculatorBPELWSPartner
Links.wsdl"
   importType="http://schemas.xmlsoap.org/wsdl/">
</bpws:import>

<bpws:partnerLinks>
  <bpws:partnerLink name="ToplamaProvider"
      partnerLinkType="ns4:ToplamaLinkType" partnerRole="ToplamaProvider">
  </bpws:partnerLink>
  <bpws:partnerLink name="CikarmaProvider" partnerLinkType="ns7:CikarmaLinkType"
      partnerRole="CikarmaProvider">
  </bpws:partnerLink>
  <bpws:partnerLink myRole="Requester" name="CalculatorRequester"
      partnerLinkType="ns10:CalculatorServiceLinkType">
  </bpws:partnerLink>
</bpws:partnerLinks>
```

Ek 2'nin devamı

```
<bpws:variables>
  <bpws:variable messageType="ns:ToplamaRequest" name="ToplamaRequest"/>
  <bpws:variable messageType="ns:ToplamaResponse" name="ToplamaResponse"/>
  <bpws:variable messageType="ns5:CikarmaRequest" name="CikarmaRequest"/>
  <bpws:variable messageType="ns5:CikarmaResponse" name="CikarmaResponse"/>
  <bpws:variable messageType="ns8:CalculateRequest" name="CalculateRequest"/>
  <bpws:variable messageType="ns8:CalculateResponse" name="CalculateResponse"/>
</bpws:variables>

<bpws:sequence>
  <bpws:receive partnerLink="CalculatorRequester"
      portType="ns8:CalculatorBPELWSPortType"
      operation="calculate" createInstance="yes" variable="CalculateRequest">
  </bpws:receive>
  <bpws:assign>
    <bpws:copy>
        <bpws:from variable="CalculateRequest" part="param1"></bpws:from>
        <bpws:to variable="ToplamaRequest" part="number1"/>
    </bpws:copy>
    <bpws:copy>
        <bpws:from variable="CalculateRequest" part="param2"></bpws:from>
        <bpws:to variable="ToplamaRequest" part="number2"/>
    </bpws:copy>
    <bpws:copy>
        <bpws:from variable="CalculateRequest" part="param1"></bpws:from>
        <bpws:to variable="CikarmaRequest" part="arg1"/>
    </bpws:copy>
    <bpws:copy>
        <bpws:from variable="CalculateRequest" part="param2"></bpws:from>
        <bpws:to variable="CikarmaRequest" part="arg2"/>
    </bpws:copy>
  </bpws:assign>
  <bpws:flow>
    <bpws:sequence>
      <bpws:invoke partnerLink="ToplamaProvider" portType="ns:ToplamaWSPortType"
          operation="topla" inputVariable="ToplamaRequest"
          outputVariable="ToplamaResponse">
      </bpws:invoke>
    </bpws:sequence>
    <bpws:sequence>
     <bpws:invoke partnerLink="CikarmaProvider" portType="ns5:CikarmaWSPortType"
        operation="cikart" inputVariable="CikarmaRequest"
        outputVariable="CikarmaResponse">
      </bpws:invoke>
    </bpws:sequence>
  </bpws:flow>
```

Ek 2'nin devamı

```
  <bpws:assign>
    <bpws:copy>
      <bpws:from expression="concat(bpws:getVariableData('ToplamaResponse', 'result')
          ,bpws:getVariableData('CikarmaResponse', 'arg') )">
      </bpws:from>
      <bpws:to variable="CalculateResponse" part="calc_response"/>
    </bpws:copy>
  </bpws:assign>
  <bpws:reply partnerLink="CalculatorRequester"
     portType="ns8:CalculatorBPELWSPortType"
     operation="calculate" variable="CalculateResponse">
  </bpws:reply>
  </bpws:sequence>
</bpws:process>
```

Ek 3. OGC WFS Yetenekler Dokümanı Örneği

```
<?xml version="1.0" encoding="UTF-8" ?>
<wfs:WFS_Capabilities xmlns:wfs="http://www.opengis.net/wfs"
xmlns:deegree="http://www.deegree.org/wfs" xmlns:gml="http://www.opengis.net/gml"
xmlns:ogc="http://www.opengis.net/ogc" xmlns:ows="http://www.opengis.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" updateSequence="0"
version="1.1.0" xsi:schemaLocation="http://www.opengis.net/wfs ../wfs.xsd">

<ows:ServiceIdentification>
  <ows:ServiceType>WFS</ows:ServiceType>
  <ows:ServiceTypeVersion>1.1.0</ows:ServiceTypeVersion>
  <ows:Title>deegree WFS 2 Test</ows:Title>
  <ows:Abstract>Test Web Feature Service</ows:Abstract>
  <ows:Keywords>
    <ows:Keyword>Persons</ows:Keyword>
    <ows:Keyword>deegree</ows:Keyword>
    <ows:Keyword>Test</ows:Keyword>
    <ows:Type>String</ows:Type>
  </ows:Keywords>
  <ows:Fees>None</ows:Fees>
  <ows:AccessConstraints>None</ows:AccessConstraints>
</ows:ServiceIdentification>

<ows:ServiceProvider>
  <ows:ProviderName>lat/lon GmbH</ows:ProviderName>
  <ows:ProviderSite xlink:href="http://www.lat-lon.de" xlink:type="simple" />
  <ows:ServiceContact>
    <ows:IndividualName>Markus Schneider</ows:IndividualName>
    <ows:PositionName>deegree core developer</ows:PositionName>
    <ows:ContactInfo>
      <ows:Phone>
        <ows:Voice>+49 228 184960</ows:Voice>
        <ows:Facsimile>+49 228 1849629</ows:Facsimile>
      </ows:Phone>
      <ows:Address>
        <ows:DeliveryPoint>Aennchenstr. 19</ows:DeliveryPoint>
        <ows:City>Bonn</ows:City>
        <ows:AdministrativeArea>Northrhine-Westfalia</ows:AdministrativeArea>
        <ows:PostalCode>53177</ows:PostalCode>
        <ows:Country>Germany</ows:Country>
      <ows:ElectronicMailAddress>mschneider@lat-lon.de</ows:ElectronicMailAddress>
      </ows:Address>
      <ows:OnlineResource xlink:href="http://www.lat-lon.de" xlink:type="simple" />
      <ows:HoursOfService>24x7</ows:HoursOfService>
      <ows:ContactInstructions>Don't call us. We'll call you.</ows:ContactInstructions>
    </ows:ContactInfo>
    <ows:Role>PointOfContact</ows:Role>
```

Ek 3'ün devamı

```
   </ows:ServiceContact>
</ows:ServiceProvider>

<ows:OperationsMetadata>
<ows:Operation name="GetFeature">
<ows:DCP>
<ows:HTTP>
  <ows:Get xlink:href="http://127.0.0.1:8080/deegree2/ogcwebservice?"
xlink:type="simple" />
  <ows:Post xlink:href="http://127.0.0.1:8080/deegree2/ogcwebservice"
xlink:type="simple" />
  </ows:HTTP>
  </ows:DCP>
<ows:Parameter name="resultType">
  <ows:Value>results</ows:Value>
  <ows:Value>hits</ows:Value>
  </ows:Parameter>
<ows:Parameter name="outputFormat">
  <ows:Value>text/xml; subtype=gml/3.1.1</ows:Value>
  </ows:Parameter>
  </ows:Operation>
<ows:Operation name="DescribeFeatureType">
<ows:DCP>
<ows:HTTP>
  <ows:Get xlink:href="http://127.0.0.1:8080/deegree2/ogcwebservice?"
xlink:type="simple" />
  <ows:Post xlink:href="http://127.0.0.1:8080/deegree2/ogcwebservice"
xlink:type="simple" />
  </ows:HTTP>
  </ows:DCP>
<ows:Parameter name="outputFormat">
  <ows:Value>text/xml; subtype=gml/3.1.1</ows:Value>
  </ows:Parameter>
  </ows:Operation>
<ows:Operation name="GetCapabilities">
<ows:DCP>
<ows:HTTP>
  <ows:Get xlink:href="http://127.0.0.1:8080/deegree2/ogcwebservice?"
xlink:type="simple" />
  </ows:HTTP>
  </ows:DCP>
<ows:Parameter name="AcceptVersions">
  <ows:Value>1.1.0</ows:Value>
  <ows:Value>1.0.0</ows:Value>
  </ows:Parameter>
<ows:Parameter name="AcceptFormats">
  <ows:Value>text/xml</ows:Value>
```

Ek 3'ün devamı

```
  </ows:Parameter>
<ows:Parameter name="Sections">
  <ows:Value>ServiceIdentification</ows:Value>
  <ows:Value>ServiceProvider</ows:Value>
  <ows:Value>OperationsMetadata</ows:Value>
  <ows:Value>FeatureTypeList</ows:Value>
  <ows:Value>ServesGMLObjectTypeList</ows:Value>
  <ows:Value>SupportsGMLObjectTypeList</ows:Value>
  <ows:Value>Filter_Capabilities</ows:Value>
  </ows:Parameter>
  </ows:Operation>
<ows:Operation name="GetFeatureWithLock">
<ows:DCP>
<ows:HTTP>
  <ows:Get xlink:href="http://127.0.0.1:8080/deegree2/ogcwebservice?"
xlink:type="simple" />
  <ows:Post xlink:href="http://127.0.0.1:8080/deegree2/ogcwebservice"
xlink:type="simple" />
  </ows:HTTP>
  </ows:DCP>
<ows:Parameter name="resultType">
  <ows:Value>results</ows:Value>
  <ows:Value>hits</ows:Value>
  </ows:Parameter>
<ows:Parameter name="outputFormat">
  <ows:Value>text/xml; subtype=gml/3.1.1</ows:Value>
  </ows:Parameter>
  </ows:Operation>
<ows:Operation name="GetGMLObject">
<ows:DCP>
<ows:HTTP>
  <ows:Post xlink:href="http://127.0.0.1:8080/deegree2/ogcwebservice"
xlink:type="simple" />
  </ows:HTTP>
  </ows:DCP>
<ows:Parameter name="outputFormat">
  <ows:Value>text/xml; subtype=gml/3.1.1</ows:Value>
  <ows:Value>text/xhtml</ows:Value>
  </ows:Parameter>
<ows:Parameter name="LocalTraverseXLinkScope">
  <ows:Value>0</ows:Value>
  <ows:Value>*</ows:Value>
  </ows:Parameter>
<ows:Parameter name="RemoteTraverseXLinkScope">
  <ows:Value>0</ows:Value>
  <ows:Value>*</ows:Value>
  </ows:Parameter>
```

Ek 3'ün devamı

```
  </ows:Operation>
<ows:Operation name="LockFeature">
<ows:DCP>
<ows:HTTP>
  <ows:Post xlink:href="http://127.0.0.1:8080/deegree2/ogcwebservice"
xlink:type="simple" />
  </ows:HTTP>
  </ows:DCP>
<ows:Parameter name="lockAction">
  <ows:Value>ALL</ows:Value>
  <ows:Value>SOME</ows:Value>
  </ows:Parameter>
  </ows:Operation>
<ows:Parameter name="srsName">
  <ows:Value>EPSG:4326</ows:Value>
  </ows:Parameter>
<ows:Constraint name="RemoteTraverseXLinkScope">
  <ows:Value>0</ows:Value>
  <ows:Value>*</ows:Value>
  </ows:Constraint>
<ows:Constraint name="LocalTraverseXLinkScope">
  <ows:Value>0</ows:Value>
  <ows:Value>*</ows:Value>
  </ows:Constraint>
<ows:Constraint name="DefaultMaxFeatures">
  <ows:Value>10000</ows:Value>
  </ows:Constraint>
<ows:Constraint name="DefaultLockExpiry">
  <ows:Value>5</ows:Value>
  </ows:Constraint>
  </ows:OperationsMetadata>

<wfs:FeatureTypeList>
<wfs:FeatureType xmlns:app="http://www.deegree.org/app">
  <wfs:Name>app:Philosopher</wfs:Name>
  <wfs:Title>European philosopher</wfs:Title>
  <wfs:Abstract>Describes famous European philosophers</wfs:Abstract>
<ows:Keywords>
  <ows:Keyword>philosopher</ows:Keyword>
  <ows:Keyword>test</ows:Keyword>
  </ows:Keywords>
  <wfs:DefaultSRS>EPSG:4326</wfs:DefaultSRS>
<wfs:OutputFormats>
  <wfs:Format>text/xml; subtype=gml/3.1.1</wfs:Format>
  </wfs:OutputFormats>
<ows:WGS84BoundingBox>
  <ows:LowerCorner>-180.0 -90.0</ows:LowerCorner>
```

Ek 3'ün devamı


```
  <ows:UpperCorner>180.0 90.0</ows:UpperCorner>
  </ows:WGS84BoundingBox>
  </wfs:FeatureType>
<wfs:FeatureType xmlns:app="http://www.deegree.org/app">
  <wfs:Name>app:SGID024_StateBoundary</wfs:Name>
  <wfs:Title />
  <wfs:DefaultSRS>EPSG:26912</wfs:DefaultSRS>
<wfs:Operations>
  <wfs:Operation>Query</wfs:Operation>
  </wfs:Operations>
<wfs:OutputFormats>
  <wfs:Format>text/xml; subtype=gml/3.1.1</wfs:Format>
  </wfs:OutputFormats>
<ows:WGS84BoundingBox>
  <ows:LowerCorner>-180.0 -90.0</ows:LowerCorner>
  <ows:UpperCorner>180.0 90.0</ows:UpperCorner>
  </ows:WGS84BoundingBox>
  </wfs:FeatureType>
<wfs:FeatureType xmlns:app="http://www.deegree.org/app">
  <wfs:Name>app:SGID024_Springs</wfs:Name>
  <wfs:Title />
  <wfs:DefaultSRS>EPSG:26912</wfs:DefaultSRS>
<wfs:Operations>
  <wfs:Operation>Query</wfs:Operation>
  </wfs:Operations>
<wfs:OutputFormats>
  <wfs:Format>text/xml; subtype=gml/3.1.1</wfs:Format>
  </wfs:OutputFormats>
<ows:WGS84BoundingBox>
  <ows:LowerCorner>-180.0 -90.0</ows:LowerCorner>
  <ows:UpperCorner>180.0 90.0</ows:UpperCorner>
  </ows:WGS84BoundingBox>
  </wfs:FeatureType>
<wfs:FeatureType xmlns:app="http://www.deegree.org/app">
  <wfs:Name>app:SGID100_RailroadsDLG100</wfs:Name>
  <wfs:Title />
  <wfs:DefaultSRS>EPSG:26912</wfs:DefaultSRS>
<wfs:Operations>
  <wfs:Operation>Query</wfs:Operation>
  </wfs:Operations>
<wfs:OutputFormats>
  <wfs:Format>text/xml; subtype=gml/3.1.1</wfs:Format>
  </wfs:OutputFormats>
<ows:WGS84BoundingBox>
  <ows:LowerCorner>-180.0 -90.0</ows:LowerCorner>
  <ows:UpperCorner>180.0 90.0</ows:UpperCorner>
  </ows:WGS84BoundingBox>
```

Ek 3'ün devamı

```
 </wfs:FeatureType>
 </wfs:FeatureTypeList>
  <ogc:Filter_Capabilities>
    <ogc:Spatial_Capabilities>
      <ogc:GeometryOperands>
        <ogc:GeometryOperand>gml:Envelope</ogc:GeometryOperand>
        <ogc:GeometryOperand>gml:Point</ogc:GeometryOperand>
        <ogc:GeometryOperand>gml:LineString</ogc:GeometryOperand>
        <ogc:GeometryOperand>gml:Polygon</ogc:GeometryOperand>
      </ogc:GeometryOperands>
      <ogc:SpatialOperators>
        <ogc:SpatialOperator name="Within" />
        <ogc:SpatialOperator name="Intersects" />
        <ogc:SpatialOperator name="Overlaps" />
        <ogc:SpatialOperator name="BBOX" />
        <ogc:SpatialOperator name="Contains" />
        <ogc:SpatialOperator name="Disjoint" />
        <ogc:SpatialOperator name="Beyond" />
        <ogc:SpatialOperator name="Equals" />
        <ogc:SpatialOperator name="Touches" />
      </ogc:SpatialOperators>
    </ogc:Spatial_Capabilities>
    <ogc:Scalar_Capabilities>
      <ogc:LogicalOperators />
      <ogc:ComparisonOperators>
        <ogc:ComparisonOperator>LessThanEqualTo</ogc:ComparisonOperator>
        <ogc:ComparisonOperator>Between</ogc:ComparisonOperator>
        <ogc:ComparisonOperator>EqualTo</ogc:ComparisonOperator>
        <ogc:ComparisonOperator>NotEqualTo</ogc:ComparisonOperator>
        <ogc:ComparisonOperator>GreaterThanEqualTo</ogc:ComparisonOperator>
        <ogc:ComparisonOperator>Like</ogc:ComparisonOperator>
        <ogc:ComparisonOperator>GreaterThan</ogc:ComparisonOperator>
        <ogc:ComparisonOperator>LessThan</ogc:ComparisonOperator>
      </ogc:ComparisonOperators>
      <ogc:ArithmeticOperators>
        <ogc:SimpleArithmetic />
        <ogc:Functions>
          <ogc:FunctionNames />
         </ogc:Functions>
      </ogc:ArithmeticOperators>
    </ogc:Scalar_Capabilities>
    <ogc:Id_Capabilities>
       <ogc:EID />
       <ogc:FID />
     </ogc:Id_Capabilities>
   </ogc:Filter_Capabilities>
</wfs:WFS_Capabilities>
```

ÖZGEÇMİŞ

1976 yılında Kayseri' de doğdu. İlk ve orta öğrenimini Kayseri'de tamamladı. 1992–1993 eğitim-öğretim yılında Karadeniz Teknik Üniversitesi, Mühendislik Mimarlık Fakültesi, Jeodezi ve Fotogrametri Mühendisliği Bölümü'nü kazandı. 5 Temmuz 1996 tarihinde aynı bölümden mezun oldu. 1996–1997 öğretim yılında Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Jeodezi ve Fotogrametri Mühendisliği Anabilim dalında Yüksek lisans programına başladı. Bayındırlık ve İskan Bakanlığı'nın açmış olduğu sınavı kazanarak, 09.01.1998 tarihinde Kayseri Bayındırlık ve İskan Müdürlüğü'ne Harita Mühendisi olarak atandı. 10.08.1999 tarihinde Fen Bilimleri Enstitüsünden Harita Yüksek Mühendisi olarak mezun oldu. Eylül 1999 da aynı üniversitede Doktora programına başladı. 22.03.2000 tarihinde Samsun Ondokuz Mayıs Üniversitesi, Jeodezi ve Fotogrametri Mühendisliği Bölümüne Araştırma Görevlisi olarak atandı. 2547 Sayılı Yasanın 35. maddesine istinaden 03.07.2000 tarihinde Doktora yapmak üzere K.T.Ü.'de görevlendirildi. Halen bu görevine devam etmektedir. İngilizce bilmektedir.