T.C.
BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI
YÜKSEK LİSANS PROGRAMI

WEB SERVİSLERİ TABANLI COĞRAFİ BİLGİ SİSTEMLERİ

HAZIRLAYAN

Fırat TURAN

Ankara - 2006

T.C.
BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI
YÜKSEK LİSANS PROGRAMI

WEB SERVİSLERİ TABANLI COĞRAFİ BİLGİ SİSTEMLERİ

HAZIRLAYAN

Fırat TURAN

DANIŞMAN

PROF. DR. HAYRİ SEVER

Ankara - 2006

T.C.
BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

WEB SERVİSLERİ TABANLI COĞRAFİ BİLGİ SİSTEMLERİ

Fırat TURAN

Bu tez, 21.09.2006 tarihinde aşağıda üye adları yazılı jüri tarafından kabul edilmiştir.

Ünvan	Adı Soyadı	İmza
Prof. Dr.	Hayri Sever	.....
Doç. Dr.	Haşmet Gürçay	.....
Prof. Dr.	Ümit Karakaş	.....

ONAY

/ / 2006

Fen Bilimleri Enstitü Müdürü
Prof. Dr. Emin AKATA

ÖZ

WEB SERVİSLERİ TABANLI COĞRAFİ BİLGİ SİSTEMLERİ

Fırat Turan

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
Ankara, 2006

Açık kaynaklı coğrafi konumsal konsorsiyumu, coğrafi bilgi sistemleri komitesi olarak geniş çapta bünyesinde çeşitli sayıda veri modelleri ve çevrim içi servisleri adapte etmektedir. Web servis özelliği olmayan versiyonlara göre, servis odaklı web harita servislerinin farklı bir istek/yanıt mekanizması bulunmaktadır. Bu web harita servis mekanizması da çeşitli öznitelikler ile genişletilebilir. Bu tez çalışmasında, CBS servisleri ile web servislerinin ekstra bilgi ve kabiliyetlerinin olduğu fark edilmiştir. Ayrıca OGC standartları ile web servis standartlarının birbirine uygun olduğu, CBS servisleri ile OGC uyumlu servislerin arasında yararlı ilişkilerin kurulduğu, CBS uygulamalarındaki web servislerinde servis üreticilerinin, web servisleri şeklinde kullanılabildiği de gözlemlenmiştir.

Bu tez çalışmasında amaç, web servisleri temelleri ile coğrafi bilgi sistemleri özelliklerine bağlı olarak uygulamalar geliştirmek, ve bu uygulamaları tamamen açık kaynaklı araçlar ve yazılımlar kullanarak geliştirmektir. Araştırmalar sonucunda CBS yayınlayıcı araç olarak, OGC uyumlu UMN MapServer kullanılmasına karar verilmiştir. Ancak bu araç üzerinde fark edilen nokta olarak, web servislerine ait WSDL dokümanı oluşturamadığı anlaşılmış ve bu oluşturamama yazılımını gerçekleştirdiğimiz web servislerinin başka platformlarda kullanılmasını desteklemediğinden bunun çözümünü geliştirmek, tez uygulamasında çözülmesi gereken bir problem olarak tespit edilmiştir. Uygulama bazında web harita servislerine (WMS) odaklanılmıştır. UMN MapServer, açık kaynak coğrafi konumsal servis özelliklerini, bu servislerin WSDL dokümanlarını oluşturmadan

destek vermektedir. Uygulamamızda SOAP istemci/sunucu mekanizması kullanılarak, UMN MapServer, harita sunucusunun OGC uyumlu WSDL dokümanı elde edilmesi sağlanmıştır. Böylelikle UMN MapServer harita sunucu aracının önemli bir eksikliği tez uygulaması çalışmamızda giderilmiştir.

Anahtar Kelimeler: OGC (Açık Coğrafi konumsal Konsorsiyumu), Web Servis, GIS (Coğrafi Bilgi Sistemi, CBS), WMS (Web Harita Servisi), WFS (Web Özellik Servisi), XML (Genişletilebilir Metin Dili), OWS (Açık Coğrafi konumsal Web Servis), Harita Sunucusu

ABSTRACT

WEB SERVICES BASED GEOGRAPHIC INFORMATION SYSTEMS

Fırat Turan

MASTER THESIS
IN
THE DEPARTMENT OF COMPUTER ENGINEERING

Ankara, 2006

The Open Geospatial Consortium (OGC) defines a number of standards, both for data models and for online services, that has been widely adopted in the Geographical Information System (GIS) community. Web service oriented WMS has a different request/response paradigm from non-web service versions, we have extended cascading WMS by adding request handler functionality. In this thesis, we will describe our group's efforts to implement GIS services according to OGC standard specifications in accordance with the web services approach. In this thesis work we have also realized extra information and capabilities about GIS services and web services. And also in this thesis we can say that there are a lot of benefits about relationship in between GIS and web services. To be able to benefit from the web services in the GIS applications, all the service providers should provide their services as Web Services.

The goal of this thesis work, to integrate web services approach and GIS properties in to application development, and the applications related are pure open source tools and software. During the investigations OGC compliant UMN MapServer GIS publisher tool had used for GIS part of the implementation. Our implementation status have used OGC compliant web services which is based web mapping services and generate it's WSDL document with UMN MapServer as an open source GIS tool. This work focuses on the web map service (WMS). UMN

MapServer supports OGC compliant web services but without generated WSDL documents. In this work we have also control this tool of lack in our implementation status. We find that the OGC standards are very compatible with web services standards.

Key Words: OGC (Open Geospatial Consortium), Web Service, GIS (Geographic Information System), WMS (Web Map Service), WFS (Web Feature Service), XML (Extensible Mark-up Language), OWS (Open Geospatial Web Service), Map Server

TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesinde beni yönlendiren, yeni yaklaşımlar geliştirme konusunda sürekli fikir ve kaynak sağlayan tez danışmanım, Prof. Dr. Hayri Sever'e, bana hep destek olan sevgili babam Mustafa Turan'a, annem Hatice Turan'a, kardeşim Funda Turan'a teşekkür ederim.

İÇİNDEKİLER

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

KISALTMALAR

OGC	:	Open Geospatial Consortuim (Açık Coğrafi konumsal Konsorsiyumu)
WMS	:	Web Map Service (Web Harita Servisi)
WFS	:	Web Feature Service (Web özellik Servisi)
WSDL	:	Web Service Description Language (Web Servisi Tanımlama Dili)
XML	:	Extensible Markup Language (Genişletilebilir Metin Dili)
GML	:	Geography Markup Language (Coğrafi Metin Dili)
UDDI	:	Universal Description, Discovery and Integration (Evrensel Tanımlama, Keşif ve Entegrasyon)
API	:	Application Programming Interface (Uygulama Programlama Arayüzü)
IBM	:	International Business Machine (Uluslararası İş Makinaları)
HTTP	:	Hyper Text Transfer Protocol (Yüksek Metin İletişim Protokolü)
IDE	:	Integrated Development Environment (Entegre Geliştirme Ortamı)
ISO	:	International Organization for Standardization (Uluslararası Standart Organizasyonu)
SOAP	:	Simple Object Access Protocol (Basit Nesne Erişim Protokolü)
HTML	:	Hyper Text Markup Language (Hiper Metin İşaretleme Dili)
DOM	:	Document Object Model (Doküman Nesne Modeli)
WWW	:	World Wide Web (Dünya Çapında Web)
URL	:	Uniform Resource Locator (Evrensel Kaynak Konumlandırıcısı)
URI	:	Uniform Resource Identifier (Evrensel Kaynak Tanımlayıcısı)
XSD	:	XML Schema Definition (XML Şema Tanımı)

WCS : Web Catalog Service (Web Kapsam Servisi)

WRS : Web Services Registry (Web Servis Kaydı)

WS-I : Web Services for Interoperability (Web Servisleri için Birlikte işlerlik)

SOA : Service Oriented Architecture (Servis Odaklı Mimari)

GIS : Geography Information System (Coğrafi Bilgi Sistemi) (CBS)

COM : Component Object Model (Bileşen Nesne Modeli)

OWS : OGC Web Service (OGC Web Servisi)

DTD : Document Type Definition (Doküman Tipi Tanımlama)

GET/POST: Al/Yayınla

BÖLÜM 1

GİRİŞ

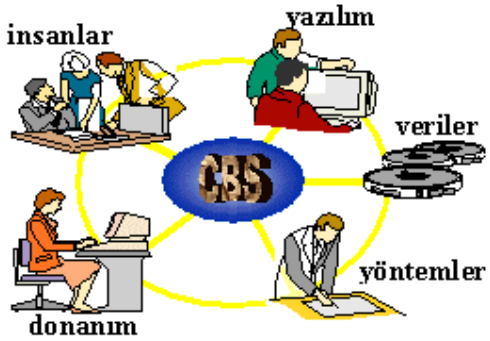
Açık kaynak kodlu CBS Konsorsiyumu (OGC) 318 şirket, hükümetler aracılığı ve üniversiteler işbirliğiyle yapılan ve alınan ortak kararlarla, halka açık ara yüzlü şartnamelerin geliştirildiği uluslararası endüstriyel konsorsiyumdur. Açık CBS şartnameleri birlikte işleyen çözümleri yani coğrafi imkanlı web, kablosuz ve yer tabanlı sunucuları ayrıca merkezi bilgi işlemleri desteklemektedir. Şartnameler teknoloji geliştiricilere kompleks konumsal bilgi ve erişebilir servisle ve her tür kullanışlı uygulamalara yetki vermektedir. Açık CBS®, açık kaynak kodlu CBS konsorsiyumunun kayıtlı ticari markasıdır ve OGC tarafından üretilmiş şartnameler ve dokümanlarla ortak model ismidir. OpenGIS şartnameleri tek bir karar işlemiyle OGC (Açık kaynak kodlu CBS Konsorsiyumu) endüstrisi tarafından desteklenmiş, hükümet ve akademik üyelere coğrafi veri işleme teknolojilerine bir host bilgisayar altında tüm işletim sistemlerinin (Pc, Mac, Suns gibi) bir arada çalışması için olanak vermesi için geliştirilmiştir. OGC kendi web sunucularını yani OWS (açık kaynak kodlu CBS konsorsiyum web servisleri) sistem mekanizmasını destekler. Açık kaynak kodlu CBS konsorsiyum web servisleri dinamik bir şekilde bağlanabilen, açık tek bir host altında çalışan sistemler zincirini kullanan ve bu şekilde dinamik uygulamalar üreten bir web coğrafi veri işleme servisidir. Açık kaynak kodlu CBS konsorsiyum web servisleri, web erişimli coğrafi veri işlemleri ve yer sunucuları tarafından istenen, gelecekte yapılacak uygulamaları bir araya getirme olayına imkan verecektir. OWS ileride yayımlanan, yer eden ve web de aranan kendinden içerikli, kendinden tanımlı modüler uygulamalar olacaktır. OWS sistemleri birçok sunucuların sıralı olarak bağlanmasına izin verir ve aynı zamanda bu sunucuların kendi dahili işlerinde bağımsız olmalarını, gerekirse bunların patentli olmalarını sağlar. GIS yöntemleri ve çevreyi görselleşmek , idare etmek ve coğrafi konumsal analiz yapmak için tanıtır. Bu yöntemler ve çevreler birlikte çalışabilirlik konusunda bazı problemlere sahiptir. Farklı organizasyonlar ve ticari satıcılar kendi data modellerini ve hafıza yapılarını geliştirmektedirler. Eğer GIS servisleri birlikte çalışamıyorlarsa, aynı organizasyon

veya ticari satıcıya ait olsalar dahi birbirleriyle etkileşimde bulunamazlar. Coğrafi uygulamaların doğası, muntazam bütünleşme ve çeşitli sağlayıcılardan konumsal veri paylaşımı gerektirir. Servislerin birlikte çalışabilirliği organizasyonların ve sağlayıcıların GIS ve Grid hesaplamaların ana amacına karşıdır. Bu problemi çözmek için açık kaynak kodlu CBS konsorsiyumu, CBS servisleri için şartnameleri yayımlayarak tanıtmıştır. Açık kaynak kodlu CBS konsorsiyumu karsız operasyon ve servislerle ilgili coğrafi veri için standartların gelişmesine yol gösteren uluslararası organizasyondur. Açık kaynak kodlu CBS konsorsiyumu farklı alanlardan örneğin özel endüstri ve CBS için açık ve genişletilebilir yazılım uygulaması programlama ara yüzüne katkıda bulunan çeşitli insanlara sahiptir. Bunun yansıra biz söylenenden daha çok birlikte işlerlik problemi olduğuna inanmaktayız. CBS servisleri örnek olarak OGC tarafından tanımlananlar, SOA (servis odaklı mimari) çevresindeki dağıtım sistemlerini kurmak için harcanılan büyük çabanın bir parçasıdır. Bu sistemler dağıtılmış sunucuları mesaja yönelik mimari, bağların çözülmesine izin verilmesi, ölçeklenirlik, hata toleransı, çapraz organizasyonel servis koleksiyonlarını birleştirir. Web servis standartları, SOA idealleri ve Grid hesaplamaları (bir haritada kesişen yatay ve dikey hatlar sistemi) ortak birbirlerine yaklaşık gereksinimlere sahiptir. GIS servislerinin web sunucu versiyonları uygulayarak, bunları bilimsel yatay ve dikey hatlar sistemi ile direkt birleştirebiliriz. Bu tanımlamalardan sonra probleme sonuç tanımı, problemi anlama, tez çalışmasının planı ve tez organizasyonlarını tez raporunun ikinci bölümünde ayrıntıları ile tanımlanmaktadır.

OGC uyumlu web servisleri, OGC'nin geliştirdiği birtakım prosedürlerin, XML ve GML metin dilleri tabanlı web servisleri olarakta ifade edilebilirler. Bu tez çalışmasında metin dilleri tabanlı web servislerinin bir uygulama üzerinde gelişimi ve kullanımını incelemekteyiz. Coğrafi bilgi sistemlerindeki veri tiplerinin günümüz internet web teknolojisini kullanarak, görsel platformda nasıl sergilendiğini, en iyi web servisleri teknolojisi ile hem de platform, işletim sistemi ve dil bağımsız bir şekilde kullanıldığını rahatlıkla söyleyebiliriz. Tez raporunun içeriğinden önce kısaca Coğrafi bilgi sistemlerinde, veri yapılarının ve veri tiplerinin neler içerdiğini, nasıl tanımlandıklarını anlatarak giriş yapıp, daha sonra bu Coğrafi Bilgi Sistemleri (CBS)

süreçlerinin, web servislerinin hangi yöntemlerini kullanarak son kullanıcılara yansıtıldığından bahsedebiliriz.

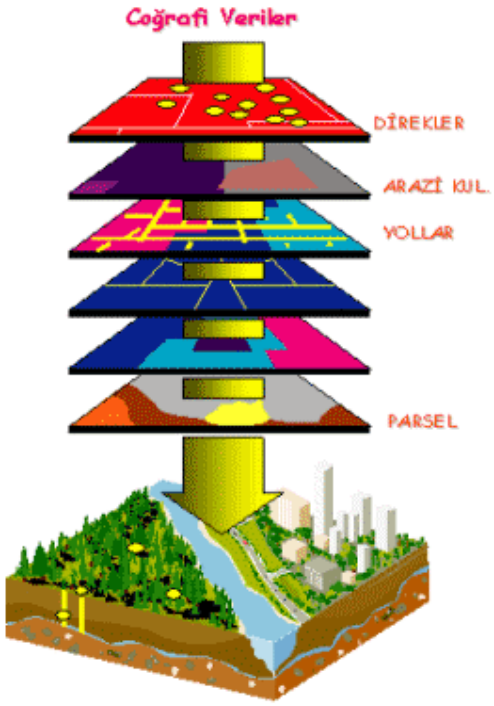
Geographical Information Systems (GIS), veya Coğrafi Bilgi Sistemleri (CBS) günümüzde konuma dayalı her türlü grafik ve tanımlayıcı bilgiyi entegre ederek kullanıcıya sunan bilgi-teknoloji tabanlı bir bilgi sistemidir. Başka bir deyişle mekandaki konumu belirlenmiş verilerin kapsanması, yönetimi, işlenmesi, analiz edilmesi, modellenmesi ve görüntülenebilmesi işlemlerini kapsayan insan ,yazılım, veriler, yönetim çemberinden oluşan bir sistemdir.



Şekil 1.1

(Kaynak Numarası: 60)

Harita bilgisi olarak nitelendirilen, konuma bağımlı grafik ve grafik olmayan yazılı bilgilerin bir sistem içerisinde bütünleştirilmesi ile ortaya çıkan bu sistem bilgiye hızlı ve sağlıklı ulaşım imkanı sağlamaktadır.



Şekil 1.2

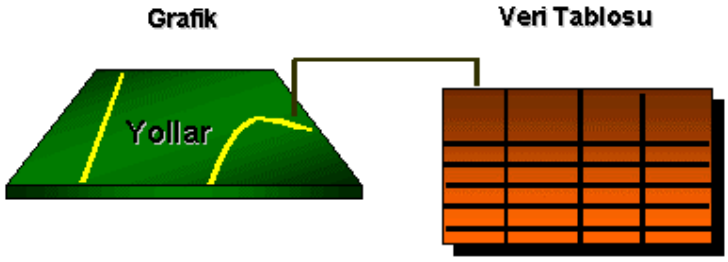
(Kaynak Numarası: 60)

Yukarıdaki şekilden de anlaşılacağı gibi CBS' de aynı coğrafi bölgeye ait farklı katmanlardan oluşan veriler bilgisayar ortamında saklanmakta ve gerektiğinde istenilen katmanlar arasında ortak analiz yapılabilme yeteneği farklı bilgisayar yazılımları ile kazandırılmaktadır. CBS' de veri iki şekilde tarif edilmektedir; konumsal veriler ve mekana ait sözel veriler. Konumsal veriler, nehirler, göller, yollar, jeolojik oluşumlar, orman türü, yerleşmeler, meteorolojik oluşumlar ….vb gibi coğrafi bilgiler ve özelliklerden oluşan veriler birbirlerinden bağımsız olarak tanımlanmaktadır. Konumsal veriler bilgisayar ortamında iki farklı şekilde saklanmakta ve kullanılmaktadır.

i- VEKTÖR burada konumsal veriler nokta, çizgi ve çokgenlerden oluşan harita elemanları ile tanımlanan

ii- GRID (RASTER) mekan üzerindeki verilerin düzenli dizilmiş karelere (piksel) aktarılması ile tanımlanan

Bu konumsal verilerin tanımlanmasında kullanılan öznitelik verileri ise (RDMS) ilişkisel veri tabanlarında saklanmaktadır. Kullanılan yazılımların yeteneklerine göre konumsal veriler ile nitelik bilgileri eşleştirilmektedir.



Şekil 1.3

(Kaynak Numarası: 60)

Konumsal veriyi tarif eden RASTER veya GRID olarak hazırlanmış tabaka ile bu tabakanın nitelik verilerinin saklandığı tablo arasındaki ilişki yukarıdaki şekilde görülmektedir. Grafik veri üzerindeki yol, nehir, bina, arazi parçaları……vb gibi coğrafi yapılar kendi içlerinde tanımlanan hiyerarşiye göre dizinlemede ve tablolarda saklanmaktadır. İlişkisel veri tabanlarında saklanan bu veriler amaca göre analiz edilebilmektedir. Konuyu biraz daha sadeleştirecek olursak CBS yeryüzündeki nesneler ve onlarla ilgili öznitelik bilgilerinin kullanıldığı bir sistem olarak Uzaktan Algılama (UA, Remote Sensing), Bilgisayar Destekli Tasarım (CAD) ve Veri Tabanı Yönetim Sistemlerinden (RDBMS) farklı olarak hepsinin birleştirildiği bütünleşik bir sistemdir. CBS de en önemli etkenlerinden biri olan yazlım konusuna gelece olursak piyasada bu konu üzerinde çok fazla yazılım bulunmakla beraber bunlardan ülkemizde en çok kullanılanları Arcinfo, Arcview, Mapinfo, Microstation, Autocad ve Netcad dir. İlişkisel veri tabanı olarak da bu yazılımlar kendi bünyelerinde veritabanlarını barındırmakla beraber dışardanda sözel veriyi alabilmektedir. Bunlar D-base, Oracle, SQL Server, Access, Paradox en çok kullanılanlarıdır.

Bu tez çalışmasında, Coğrafi Bilgi Sistemleri ve temel özelliklerinden hareketle, günümüz web tabanlı teknolojilerde, platform bağımsız uygulamaların kullanıldığı web servis özelliklerinin incelenmesi, konumsal verinin web tabanlı olarak

görüntülenmesi ve görüntü üzerinden konumsal veriye erişimin sağlanması, verinin güncellenmesi gibi süreçlerin incelenmesine karar verilmiştir. Buna göre araştırmalarda OGC uyumlu web servislerine ve bu servisleri kullanabilen açık kaynak kodlu araçlara önem verilmiştir. OGC uyumlu web servislerinden Web Harita Servisi (WMS) ve Web Özellik Servisi (WFS) yazılımları yapılıp, açık kaynaklı CBS aracı olan UMN MapServer üzerinden web ortamında görüntüleme sağlanmıştır. Konumsal verinin kaynağı olarakta yine açı kaynaklı PostGIS coğrafi konumsal veritabanı kullanılmıştır. UMN MapServer aracıda bu veritabanını desteklemektedir. Görüntüleme işleminin yapılabilmesi için bu web servislerinin XML ağaç şemaları, OGC uyumlu olabilecek şekilde tasarlanmıştır. Web servislerinin kendi prosedürlerine göre OGC konsorsiyumu tarafından hazırlanmış XML şemaları mevcuttur, bu şemalar incelenerek, uygulama mimarisi için XML şemaları tasarlanmış ve web servisleri özellikleri ile fonksiyonları üzerinden çalıştırılması sağlanmıştır. OGC uyumlu web servislerinin kullanımı, konumsal veriyi görüntüleyecek CBS aracının yapısına göre de değişmektedir ve günümüzde web servislerinin kullanımı iki şekilde sağlanmaktadır. Bunlar SOAP istek/yanıt yapısı ile HTTP GET/POST (AL/YAYINLA) yapısı şeklindedir. Burada SOAP istek/yanıt, web servisinin ne yaptığını, nasıl kullanıldığını ve ne gibi özelliklere sahip olduğunu belirten WSDL dilinin oluşumu ile kullanılmakta, HTTP GET/POST (AL/YAYINLA) ise, kullanılan araç üzerinden url'ler ve bunlara eklenen, parametreler ile veri iletişimi sağlanmakta ve sonuca gidilmektedir. Araştırmalar sonucunda kullanılan CBS aracının WSDL dokümanı oluşturamadığı ve bu dokümanın eksikliğinden dolayı, geliştirilen web servislerinin başka platformlarda SOAP istek/yanıt çerçevesinde kullanılamayacağı anlaşılmıştır. Bu eksikliğin giderilmesi için, açık kaynaklı bileşenler ve bu bileşenlerin desteklediği yazılımlar geliştirilerek, CBS aracı olarak kullanılan UMN MapServer'ın WSDL dili üretemem eksikliği bu tez çalışmasında giderilmiştir. Sonuç olarak bu tez çalışmasında, OGC, OGC uyumlu web servisleri ve özellikleri, OGC destekli araçların araştırılması, açık kaynak kodlu araçlar ve bileşenlerin araştırılması yapılmış olup, bu araştırmalara göre uygun görülen araç ve yazılımların eksikliklerin giderilmesi ve OGC uyumlu web servislerinin kullanıldığı mimari bir uygulama programlama arayüz yapısı geliştirilmiştir. Aynı zamanda Coğrafi Bilgi Sistemleri (GIS-CBS) ile Servis Odaklı Mimarinin (SOA) bütünleştiği bir yapı tasarlanmıştır.

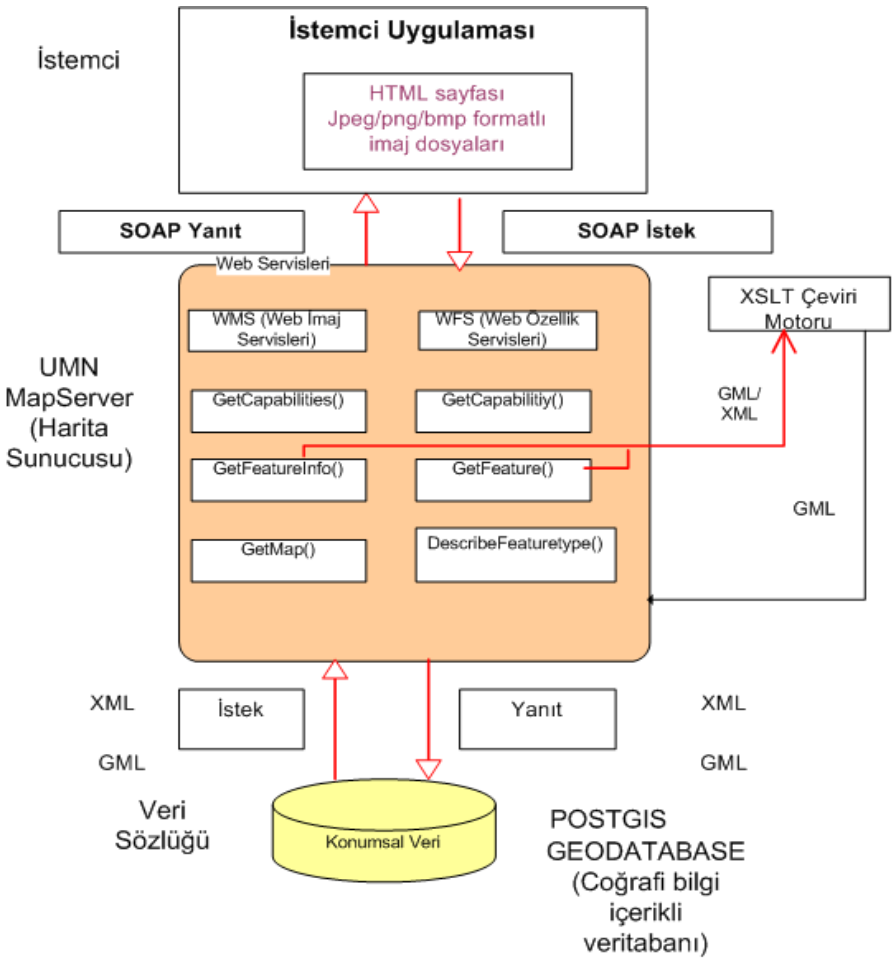
Tez raporunun birinci bölümünde, tez içeriğinin temel özellikleri, tez konusu, kullanılan senaryolar ve coğrafi bilgi sistemleri hakkında temel anlamda giriş bilgileri verilmektedir. İkinci bölümde temel CBS servisleriyle OGC Web servis şartnameleri karşılaştırılmakta, OGC ve OGC web servisleri mimarisi hakkında detaylı bilgi verilmektedir. Tez konusu araştırmalarında WMS ve WFS servislerinin yapıları ve kullanım senaryoları ile uygulama yapısı için tasarlanan web servis sistem mimarisi anlatılmaktadır. 3.bölümde WMS ve WFS için OGC tabanlı web servisleri hakkında detaylı bilgiler verilmektedir. 4.bölümde, CBS görsel servisleri için konumsal veri birlikte işlerliği ve bunun için de genel mimari tanımlanmakta olup, XML tabanlı veri aktarım servislerinin, veri etkileşimi ve veri yapısı standardı olarak sunulan, PostGIS veri tabanı hakkında bilgiler verilmektedir. 5.bölümde, XML dokümanlarındaki web servis teknolojileri, SOAP tekniği, WSDL dokümanı, UDDI mekanizması ve HTTP AL/YAYINLA yöntemi ve ayrıca CBS'nin bakış açısından avantajları açıklanmaktadır. Bu başlıklar altında alt başlıklar olarak; Web servislerinin CBS servislerine katkısı, uygulamalar sırasında mücadele edilen teknik problemler, OGC uyumlu CBS görüntülenmesine web servisleri katmak, web servislerinin kullanımı durumunda WMS servislerine geçerli istekler yaratmak, basamaklı WMS kapasitesini bağlamak, WMS'nin uygulama detayları ve önerilen görsel sistemi içeren diğer servisler. Bütün bu yapıların kullanım dili olarak, PHP dilinin seçimi ve bu dil ile geliştirilmiş açık kaynak kodlu bileşenlerden bahsedilmektedir. 6.bölümde ise, gelecekteki çalışmalar ve sonuç anlatılmaktadır.

BÖLÜM 2

OGC WEB SERVİSLERİ MİMARİSİ

2.1 Problemin Tanımlanması

Bu tez raporu, tasarım ve web servislerinin web harita sunucusundan OGC şartnamelerinin yeniden yapılandırılmasının mimarisi hakkında detay bilgiler vermektedir. Araştırmalarda ve uygulamada, uyumlu web harita servis yetenekleriyle uygulama detaylarına odaklanılmaktadır. Bu uygulama detayları yardımcı yayınlarda tanımlanır. Bu, web servis standartlarındaki, CBS sunucularındaki çevirilerin soruşturulması için bizim tarafımızdan yapılan büyük çabanın bir kısmıdır. Bu alandaki bazı erken çalışmalar WMS ile rapor edilir. Bu dokümanda servis ara yüzlerinin standart WSDL tanımlaması belirtilmektedir. Ayrıca, bu dokümanda ilk önce bazı arka plan bilgisi ve bazı ilgili çalışmaların açıklamasında verilmektedir. Bu tez çalışmasında OGC uyumlu web servisleri, müşteri ve sunucu arasındaki metotları uygulama durumu için Şekil 2.1'i kullanmaktadır. Daha önce belirtildiği gibi bu şekil OGC uyumlu web harita servislerinde ve web şekil servislerinde kullanılan ve hatta postGIS açık kaynak kodlu veri tabanıyla ilgili konumsal veri sunucusu kullanılan web servis sistem mimarisiyle de ilgilidir.



Şekil 2.1 Problemin Tanımı için geliştirilen Webservis Sistem Mimarisi

Bu şekilde gösterilenler SOAP istek/karşılık ve istemci/sunucu sistem mimarisindeki SOAP mekanizmasını WSDL ile birleştirir ve bu mimari post GIS uygulamasında bulunan konumsal veri sunucusundan veri alır. Web servis metotları XML sorgusunun bir parçası olarak müşteriden veri alır ve bunları sunucuya gönderir, ve sunucu da XML sorgusunun çıktısını iletir, bu sorgular müşteri görsel kısmı için url olarak örnek verilerek Getmap() yöntemlerine gönderilir. Web servis metotlarını ve Bölüm 2.2'deki probleme sonuç tanımlamalarındaki uygulamaları detaylı olarak tanımlamaktadır. Ayrıca bu tezdeki problem tanımı OGC uyumlu web servislerini nasıl oluşturacağını ve onların WSDL dokümanlarını nasıl yaratacağımızı tanımlar. İlk

olarak web servislerinin ne olduğunu OGC uyumunun anlamını, OGC web servislerini ve problem tanımlamalarından önce web sevislerinin yapısı açıklanabilir.

2.2 Problemi Anlama

Bu problemi anlamadaki amaç, web harita servisleri tabanlı OGC web servislerini ve özellik servisleri kapasitelerini uygulama durumlarında test etmektir. Problemi anlamak için iki parçaya bölünen yolun ilki, OGC web servis mimarisinin tasarımı ve bunun anlamının ne olduğudur. İkincisi ise OGC web servislerini oluşturmaktır. Haritalama özellik servisleri mimarileri ve fonksiyonları. Bunları önümüzdeki 2.2.1'de ve 2.2.2'de detaylı bilgi vererek tanımlanmaktadır.

2.2.1 Web Haritacılık ve Özellik Servisleri (Mapping and Featuring)

Web Özellik Servisi

WFS örnek olarak coğrafi konumsal verileri saklar ve onları müşteriden gelen isteklere göre hizmete sunar. WFS müşterileri web harita sunucuları ve diğer WFS'lere sahiptir. WFS özellikli vektör verisi sağlar. Vektör verileri GML de şifrelenir, bir coğrafi bilginin saklanması ve taşınması için XML şifrelemesi yapılır, coğrafi özellikler ve geometriyi de içerir. Açık CBS WFS şartnamesine göre, temel web özellik servisleri getCapabilities (kapasiteleri alma), describeFeatureType (özellik tipi tanımlama) ve getFeature (özellik alma) dır. Eğer WFS (özellik servisi) kurumun bütün rapor ve kayıtlarını gösteriyorsa, bu WFS iki fazla servis sağlar. Bu iki servis "transaction" (rapor ve kayıtlar) ve "lockFeature" (kilitleme özelliği) servisleridir. Bizim WFS uygulamalarımız temel WFS uygulamalarıdır, böylece transaction (işlem görme) ve lockFeature (ilgili özelliği kilitleme) kabiliyetleri yoktur. Temel WFS'leri uyguladığımızdan bu yana, WMS temel WFS servislerini kullanır. Bunlar: getCapabilities, describeFeatureType ve getFeature dur. WMS, WFS'ye WFS'lerin

servis edebileceği tipleri ve bu tiplerdeki desteklenen operasyonları öğrenmek için getCapabilities isteğini gönderir. Bu getCapabilities isteği tüm IS (bilgi servisleri) tarafından aracılık edilir. WMS istenilen özellikleri sağlayan belirli WFS adresleri almak için isteğini yapar, WMS ve IS arasındaki bağlantı Bölüm 3 Şekil 3.7'de gördüğünüz tez uygulamaları için gelecek çalışmada araştırılmış olacaktır. Ne zaman WMS müşterileri WMS'ye getFeatureInfo isteği yollasa, WMS bir getFeature isteği yaratır ve bunu WFS'ye gönderir. WFS'nin url adresi IS kullanılarak bulunur. Uygun bir WFS seçildikten sonra WMS özellikli veri almak için getFeature isteği yapar. Basit bir istek Şekil 2.5'de gösterilmiştir. XML'de şifrelenen GML dosyası bu isteğe cevap olarak SOAP zarfında döndürülür.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<wfs:GetFeature outputFormat="GML2"
xmlns:gml="http://www.opengis.net/gml"
xmlns:wfs="http://www.opengis.net/wfs"
xmlns:ogc="http://www.opengis.net/ogc">
 <wfs:Query typeName="boundary_lines">
<wfs:PropertyName>FNODE</wfs:PropertyName>
<wfs:PropertyName>TNODE</wfs:PropertyName>
<wfs:PropertyName>WORLD</wfs:PropertyName>
<wfs:PropertyName>coordinates</wfs:PropertyName>
<wfs:PropertyName>minx</wfs:PropertyName>
<wfs:PropertyName>miny</wfs:PropertyName>
<wfs:PropertyName>maxx</wfs:PropertyName>
<wfs:PropertyName>maxy</wfs:PropertyName>
  <ogc:Filter>
    <ogc:BBOX>
      <ogc:PropertyName>coordinates</ogc:PropertyName>
      <gml:Box>
        <gml:coordinates>-155,28 -120,50</gml:coordinates>
      </gml:Box>
    </ogc:BBOX>
  </ogc:Filter>
 </wfs:Query>
</wfs:GetFeature>
```

Şekil 2.2 WMS'den WFS'ye geçiş için kullanılan örnek GML

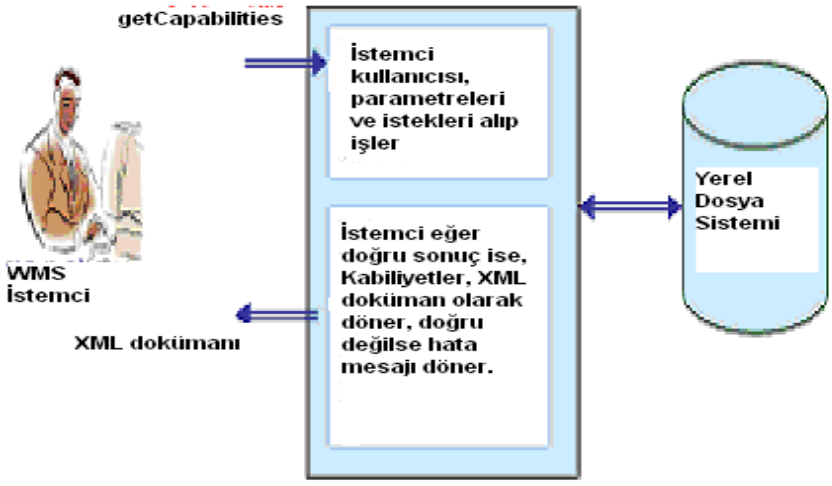
Görsel Servis - WMS

WMS, CBS görsel sisteminin anahtar servisidir. WMS coğrafi veriden harita üretir. Harita kendisi bir veri değildir. Haritalar gelişmemiş coğrafi veriden, vektörden veya özet verilerden bilgi yaratır. Haritalar genellikle resimsel şekil haline getirilir. Örneğin jpeg, gif ve png.WMS ayrıca ölçeklenebilir vektör grafiklerinde bulunan vektör tabanları grafiksel elemanlardan haritalar üretir. WMS üç ana servis sağlar; bunlar

getCapabilities, getMap, getFeatureInfo'dur. GetCapabilities ve getMap servisleri harita üretmek için gereklidir ama getFeatureInfo opsiyonel bir servisdir. Bu servisler ve uygulamalarımız alt bölümlerde açıklanmıştır.

WMS Servis Kabiliyetleri

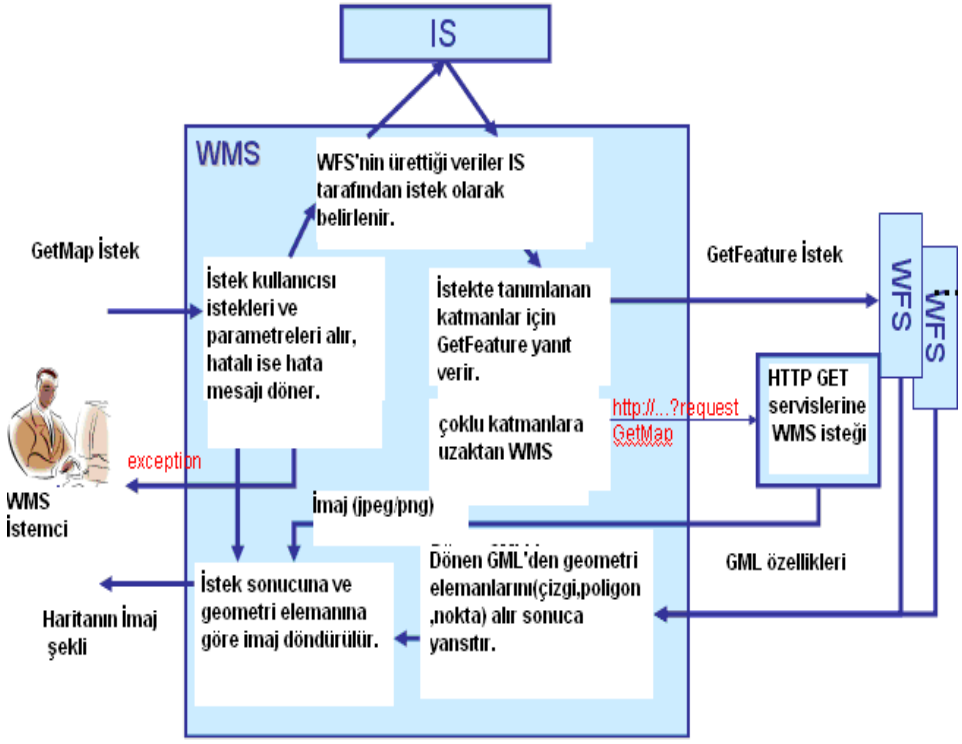
WMS müşterisi WMS'den harita istemeden önce, WMS'nin hangi bağlayıcı kutularının ne tabakalarda olduğunu bilmesi gerekir. GetCapabilities isteği WMS müşterilerine WMS ile ilgili bilgi almasına izin verir. GetCapabilities (Kabiliyetleri alma) isteği sunucuya kendi kapasitesini bildirmesine izin verir. Örneğin; uygun tabakalarla , destekli çıktı projeksiyonları, destekli çıktı şekilleri ve genel servis bilgileri. Bu isteği aldıktan sonra WMS, XML dokümanı ile birlikte WMS sunucusu hakkında metaveri döndürür. Bu kapasite dosyaları, yerel dosya sisteminde saklanır ve getCapabilities isteği yapan müşterilere yollanır. Bu isteği doğrularsa o zaman kapasite dosyalarını WMS müşterisine SOAP ek parçası olarak yollar. Eğer WMS isteği idare etmekte tesadüfen bir problemle karşılaşırsa o zaman SOAP' tan WMS müşterisi de bir istisna mesajı yollar. Temel GetCapabilities isteği Şekil 2.3'te gösterilmiştir.



Şekil 2.3 getCapabilities iş şeması

WMS Harita Görüntüleme

WMS tarafından sağlanan bir başka servis arayüz de getMap isteğidir. GetMap servis ara yüzü haritaya erişim sağlar. Harita üretmek için zincirlenmiş işlemler Şekil 2.4'de örneklendirilmiştir. Bu istek müşteri tarafından getCapabilities isteği bitirildikten ve uygun tabakalar belirlendikten sonra yapılır. GetMap isteği alındıktan sonra WMS Şekil 2.4'ün üzerinden geçer ve eğer her işlem başarılı ise getMap isteğindeki tanımlanmış biçimdeki görüntüyü sonuç olarak döndürür. Bütün desteklenmiş görüntü biçimleri WMS Capabilities dokümanında tanımlanmıştır. Görüntü biçimleri için yapılan istekler, WMS'nin Capabilities dosyasına göre olmalıdır. Görüntü WMS müşterisinin SOAP mesajının ek parçası olarak döndürülür. Eğer WMS, isteği idare etmekte bir problemle karşılaşırsa o zaman SOAP'tan WMS müşterisine bir istisna mesajı yollar. WMS ilk olarak parametreleri böler ve onların değerlerini getMap ten alır. Parametrelere dayanarak WMS başka WMS servislerine bazı isteklerde bulunmaya gerek duyabilir. WMS ilk olarak hangi tabakaların hangi bağlanmış kutularda ve hangi formda olduğuna karar verir. WMS döndürülmüş WFS'ye istekte bulunur ve GML biçiminde istenilmiş özellikli veri alır. Eğer parametre tanımlamasındaki döndürülen görüntü biçimindeki bir jpeg isteği ise o zaman, WMS geometri elemanlarını kullanarak döndürülmüş özellikli veri yaratır. Bu şema dosyalarını kullanarak GML dosyasından özellikle veriyi görüntülemek için geometri elemanları türetilir. GML deki bu geometri elemanları başlıca nokta, poligon, çizgi bandı, doğrusal daire, çoklu nokta, çoklu poligon, çoklu geometri, ...vs. WFS den dönen özelliklerden görüntü yaratmak için Java Graphics2D ve Java AWT kütüphanelerini kullandık. Her tabaka için farklı grafikler objesi yaratırız. Eğer her tabaka için farklı grafikler objesi ayırırsanız o zaman Java kütüphaneleri bu grafik objelerini kaplamanıza izin verir.



Şekil 2.4 getMap iş şeması.

WMS Özellik Alma

Bu isteğe bağlı bir WMS servisidir. Harita yaratmak için gerekli değildir. Bu sadece kullanıcının ileriki safhalarda haritada özellikli tipler hakkında bilgiye gereksinim duyduğunda kullanılır. Buna rağmen biz bunu coğrafi fiziksel uygulamalar için, kurulan etkileşimli arayüzlerde çok yararlı bulduk. GetFeatureInfo metodu bize bunları input olarak kullanan simülasyon kodlarına ek bilgiler yollamamız için izin verir. Örneğin; deprem hata boyutları ve materyal özellikleri. GetFeatureInfo şu şekilde çalışır: kullanıcı (x,y) kartezyen koordinatı ve ilgili tabakaları sağlar. Ve bilgiyi HTML, GML veya ASCII formda geri alır. Bütün bu desteklenen biçimler WMS capabilities dosyasında tekrardan tanımlanmıştır. Şekil 2.5, WMS tarafından başarıyla yapılan WMS müşterilerinin GetFeatureInfo dan yaptığı isteğin cevabını gösterir. Şekildeki sunumu daha somut hala getirmek için biz özellikli bilginin text/html tipinde istendiğini varsaydık. GetFeatureInfo isteğindeki bu değer parametrede “info-format”

olarak tanımlanır. GetFeatureInfo servis arayüzü ikiden daha fazla bilgi biçimleri destekler. Bunlar yalın text ve GML biçimleridir. Şekil 2.5'de GetFeatureInfo istek işlemini örneklerle göstermek için xpath operasyonuyla ayrılan GML dokümanları seçilmiştir. WMS'den gelen getMap teki bütün işlemler getMap için getFeatureInfo işlemiyle, WFS'den istenilen özellikler aynı olana kadar açıklanmıştır. Tekrar söylemek gerekirse uzaktan yapılan yardımlar SOAP mesajları kullanılarak yapılır. Şekil 2.5'de açıklandığı gibi harita üretmek yerine WFS'den özellikli veri koleksiyonu alındıktan sonra, WMS tüm geometrik olmayan elemanları ve GML dosyalarında dönen tüm değerleri listeler. GetMap isteği için WMS GML dosyalarından dönen geometri elemanlarıyla uğraşır fakat GetFeatureInfo için geometrik olmayan elemanlarla uğraşır. WMS, coğrafi konumsal olmayan listedeki elemanlardan geometrik olmayan elemanları html'ye çevirmek için yeni bir XML dosyası yaratır. XML dosyası basit, sadece geometrik olmayan elementleri özellikleri ve değerleri içeren GML'in başka bir formudur. Şekil 2.5'de GetFeatureInfo daki bütün işlemleri göstermek için, istenilen bilginin HTML biçiminde olduğunu varsaydık. Geometrik olmayan elemanlardan yeni bir XML dosyası yaratıktan sonra WMS, yeni yaratılmış XML dosyasından, XML verisinden şifreli url verisine transfer edilen geniş kapsamlı xpath operasyonunu kullanarak HTML dosyası yaratır. Problemi anlamayı iki parçada belirtebiliriz. Yukarıdaki bölümlerde ve onların alt bölümlerinde görebilirsiniz.

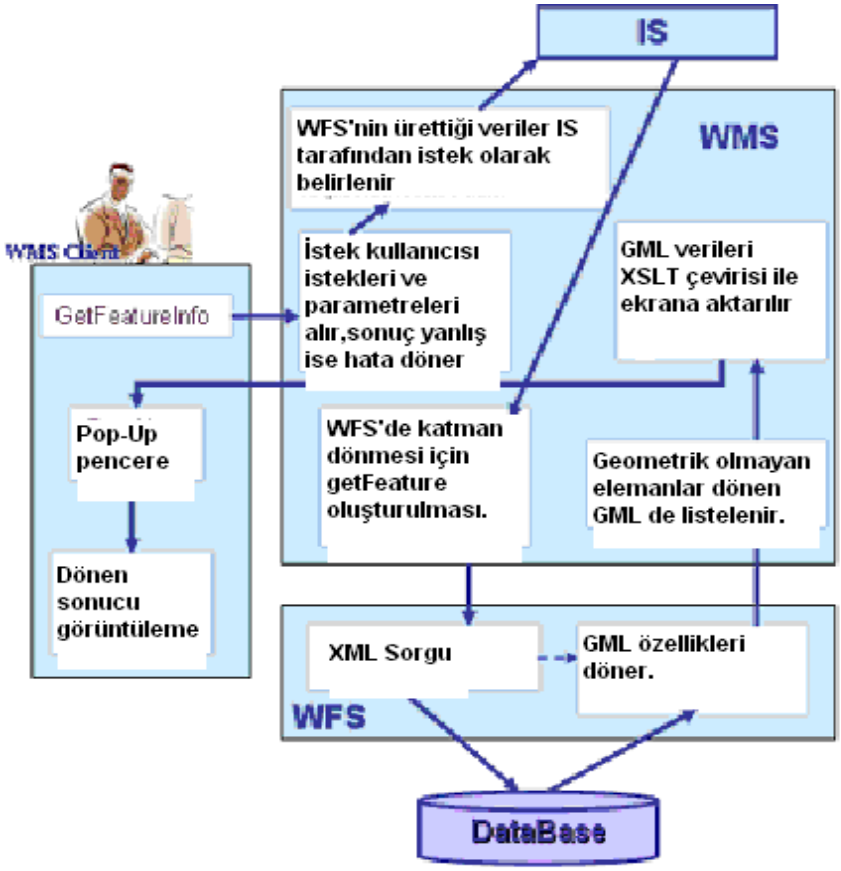


Figure 2.5 getFeatureInfo iş şeması.

2.3 Tez Çalışmasının Planı

İlk önce WMS ve WFS sunucularıyla uyumlu sade yapılı OGC uygulaması plan olarak ele alınmaktadır. Bu sunucular http üzerinden http al/yayınla istekleri yaparak iletişim kurmaktaydılar. Sonra, http tabanlı görsel sistemi servis tabanlı ters parçalara çevirmek için geniş kapsamlı algoritma aşamaları geliştirildi. Bu aşamalar Şekil 2.1'de ve 3.2'de listelenmiştir. Kendi fonksiyonalitesi için arayüz kümesi tanımlı OGC web servisler (OWS) için WSDL tanımlanır. OWS'nin sağladığı bütün istekler ve cevaplar için uygun XML şeması yaratılır. Bu şemalar OGC OWS şartnamelerinde tanımlanan HTTP/POST (yayınla) ve HTTP/GET (al) değer ve özelliklerine göre yaratılır. Buradaki genel işlem döngüsündeki hedef OWS'in WSDL dosyasından müşteri istek dokümanları oluşturulmasıdır. Bu işlem süreçleri sonunda, OGC CBS sunucusuyla

uyumlu olan bağımsız web servisleri yaratıldıktan sonra, mimari yapı bu tür sunucuyu başka geniş kapsamlı OGC sunucusuna çevirmeye hazırdır denilebilir.

2.4 Tez Organizasyonu

Bu tez organizasyonunda, OGC sunucu mimarisini ve onların kabiliyetlerini ve OGC uyumlu servis dokümanlarını desteklemeyi, nasıl web servis mimarisinde tasarlanabileceği araştırılmıştır. Uygulama durumu için WSDL bağlı SOAP istek/cevap programları organize edilmiştir. CBS servisi yayımları, açık kaynak uygulama programlama ara yüzleri ile, entegre geliştirme ortamları için açık kaynak kodlu araçları kullanılmıştır. Müşteri ve sunucu tarafları için PHP ve Java programlama dilleri kullanılmıştır. Web servisleri www standartlarıyla organize edilmiştir. WMS ve WFS web servisleri de, web servis ve OGC standartlarına sahiptir, böylece bu tez çalışması için OGC ve www standartlarıyla organize olmuştur denilebilir. Tez uygulama içeriğinin organizasyonu özetle, OGC uyumlu WMS ve WFS XML web servislerinin istek/yanıt süreçlerinde nasıl işlendiğinin gösterildiği açık kaynak kodlu araçların kullanıldığı ve bu araçlardan UMN MapServer CBS yayınlayıcı aracının eksikliği olan WSDL dokümanı üretememesinin gelişimi sağlanmaktadır.

2.5 Dağıtılmış Mimariler Üzerinde Coğrafi İşlem Servisleri

OGC özgün olarak dağıtılmış hesaplama platformlarına odaklanmaktadır veya (özellikle CORBA, OLE/COM ve SQL) üç ayrı esas olan yüksek seviye servis uygulama modelleri olan DCP'lere odaklanabilmektedir. HTTP, WWW ve WebMapping TestBed (Web haritalama test ortamı) sonraki şeklinin ortaya çıkmasıyla OGC Dağıtılmış coğrafi veri işleme sokulmuştur. Bu yeni model hala şekil almakta ve büyüyecek olan OGC Web servislerinin etrafında yapının çekirdeğini oluşturmaktadır.

2.5.1 Mimari Değerlendirme

OGC Web servislerinin vizyonu birçok faktörden etkilenebilmektedir:

-Verilen servislerin bağımsız şekilde geliştirilmiş olan uygulamalarına ihtiyaç duyulması. Örneğin: coğrafi kodlama

-Birlikte çalışabilen zincir servisleri uygulamasına ihtiyaç duyulması.

-Servislerin bağımsızca yeni objeler yaratmak için destek olunmasına ihtiyaç duyulması

-Servis tiplerine dayalı servislerin belirli yeni obje bulmasına, servis içeriğine, servislerin karakteristik ve kalite faktörlerine ihtiyaç duyulması

-Belirli veri ambarları ile, hangi veriye ait servislerin kullanılabileceğini keşfetmeye ihtiyaç duyulması.

-Servislerin direkt veri içeriği olmadığı sürece, sıkıca birbirlerine bağlanılmasının tespit edilmesine ihtiyaç duyulması

-Servislerde giriş kontrolü, güvenlik ve e-ticaret ile izin verilmeye ihtiyaç duyulması.

-Toplam iş akışı işleminin sağlanabilmesi için servislerin bağımsız şekilde eklenti zincirlemesine izin verilmesinin istenmesi.

Bu faktörler, teknoloji geliştiricileri ve kullanıcıları için OGC'nin nasıl bir güce sahip olduğunu, tanımlanan mimari kararlar verilirken doğru dengeyi bulmakta çektikleri sıkıntıları da aynı zamanda tanımlamaktadır. WebMapping TestBed'in (web harita test ortamı), yani WMS'nin bir sonraki evrede başlıca odaklanması gereken mimari kararların adresinin belirtilmesi bu bağlamda gerekmektedir..

İstek/Cevap Kapasiteleri

Verilen bir servis kendi servis tipine erişebilmeyi sağlamalıdır ve bu aynı zamanda servis örnek meta verisidir. Tarihsel olarak OGC'nin içinde "Capabilities" (kabiliyetler) istek ve cevapları bilinir.

Servis Tipleri ve Arayüz Kalıtımı

Servis, meta verisi ve cevap mekanizma kapasitelerini kullanarak, servis tiplerinin ve kalıtım ağaçları olan diğer servis tiplerinin sınıflandırılmasını geliştirebilmektedir.

2.6 Rol ve Ortak Mimarinin Tanımı

Açık CBS web servisleri (OWS), dinamik coğrafi konumsal uygulamaların bireysel bileşenleridir. Ayrıca coğrafi konumsal problemler için kurulan çözümlerin ve tüm paradigmanın parçalarıdır. Bu paradigma (Konumsal Web) kavramsal ve kısıtlanmış uygulamaların OWS üzerinde nasıl çalıştığını empoze eder. Kavramsal kısıtlamalar servis tanıtımı, çok katmanlı dağıtma kapasitesi, kendini tanımlama ve istatistikleri olmayan operasyonları içerir. Bu kısıtlamalar genellikle işlevsel adreslerdir. Uygulama kısıtlamaları ortak XML şifrelemesini, HTTP taşımasını, sıkıca tanımlanmış arayüz söz dizimini, servis tanımlamaları için belirli bilgi modellerini ve diğer metaveriler içerir. Bu kısıtlamalar genellikle adreslerin birlikte işlerliğidir. Birlikte alınırsa bu kısıtlamalar birlikte çalışabilen, başarılı OWS uygulamaları örneklerine yol göstermek için ortak temel ve mimari sistemi oluşturur. Basit bir görüş açısıyla bakarsak daha fazla ortak kısıtlamalar bir defaya mahsus da olsa bütün servisleri memnun edebilir, daha az iş ise, ya önceden varolan servis tiplerine uygulanır ya da yenileri tanımlanır.
Herhangi bir fonksiyonel web ile çoğu değerini web'e veren ölçü ve farklılıklarla daha hızlı değerlendirilmelidir. OWS ortak mimarisi uygulamalara ve plana göre yerleştirmek için konseptlere rehberlik sistemi, terminoloji, temel kalıplar ve organizasyon prensiplerini tanımlamaktadır. OWS ortak mimarisi ortak arayüzler kurabilmekte, protokolleri değiş tokuş edebilmektedir ve yararlanılan herhangi bir uygulamayı sunabilmektedir. OpenGIS uygulama şartnameleri uygulamalarla OWS mimarisinin uymasını nasıl sağlayabilmeleri için kılavuzluk sağlamaktadır. OpenGIS servisleri OpenGIS uygulamalı şartnamelerine uyan uygulamalardır. OpenGIS uygulamaları olarak adlandırılan uyumlu uygulamalar, işlemsel çevreye ayak uydurabilmek için sisteme "plug into" eklenti olabilir. Bu hareketli sistemlerde girişimci gelişmelere gevşekçe ikili alınır. Ortak arayüzlere uygulamalar kurarak her uygulama başka bir uygulamadaki geliştirme ortamına veya çalışma ortamına bağlı olmaksızın yapılır. Yeni uygulamalar ve servisler, diğer uygulamalara etki etmeden

eklenebilir, modifiye edilebilir veya yerine başka bir şey konabilir. Buna ek olarak işlemsel iş akışı dinamik olarak değiştirilebilir. Ve kriz durumlarına hızlı tepki vermeye izin verir. Bunları takip eden alt başlıklarda OpenGIS web servisleri ve onların etkileşimleri tanınmıştır.

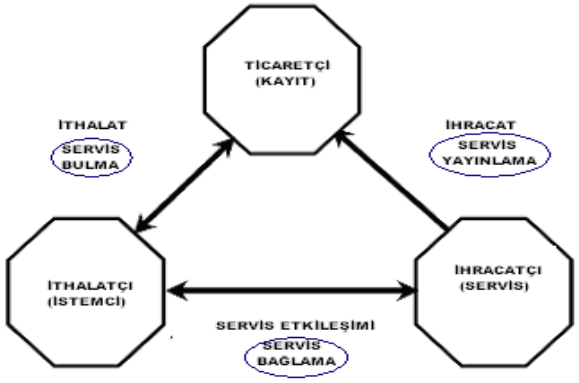
2.6.1 CBS' de Servis Odaklı Mimari (SOA - Service Oriented Architecture)

Servis odaklı mimari (SOA) kalıbı servis sağlayıcılarına temel ve dağıtılmış işlemsel network ile birlikte servis tüketicilerine roller sağlar. Bu kalıp işlemsel görevlerin veya çözümlerin bilinmesi gereken tiplerden bir bireysel servis, bağımsız şekilde eklenti (Ad-hoc) konfigürasyonunun geliştirilmesini vurgular. Tanımlanmış bir servisin sağlayıcı ya da tüketici bileşen tanımlarına odaklanır. Ayrıca SOA, aynı tanımlanmış servisler üzerindeki bileşenle aralarındaki etkileşimi, servis isteklerinin adlandırılmış haline, servis cevaplarına veya servis istisnalarına odaklanır. Bu paradigma, OGC Web Servislerinin işlemsel bilgi bakış açılarına ve buna bağlı raporları da sunmuştur denilebilir.

2.6.2 Servis Ticareti Yayımlamak-Bulmak-Bağlamak (Publish-Find-Bind)

Servis ticareti elverişli servis örneklerinin keşfedildiği temel kavram adreslendirilmesidir. ODP ticaret fonksiyonu, özel tipler sağlaması ile servisler arası paylaşımı kolaylaştırır. Bu ticaretçi uygulama kayıt servisi, reklam servislerinin isteği için öneri ve karşılaştırma yapar. Kapasiteyi yayımlamaya veya bir servisi teklif etmeye "export" (ihracat) denir. Yayınlanmış tekliflere veya keşfedilen servislere karşı istenilen servisin eşleştirilmesine "import" (ithalat) denir. Bu ayrıca "publish-find-bind" servis etkileşimi tarzında da tanımlanabilir. Ticari fonksiyon ayrı bir dokümanda inceden inceye ve teknik olarak bilgisayarlar ile ODP ticari fonksiyonunun hızlanmasıyla OMG ticaret şartnamelerinde oldukça arıtılmıştır. En önemlisi, bir ticaretçi, siteler ve uygulamalar sıklıkla dağıtılmış sistemler değiştiği sürece, servis sağlayıcıları ve istekçiler arasında dinamik bir destek bağı kurar. Temel roller ve etkileşimler Şekil 2.6'da yansıtılmıştır. PFB terminolojisinin eşiti daire içine alınmış

şekilde gösterilmiştir. Bir ticaretçi ihracatçı objelerden servislere teklifte bulunur ve bazı kriterlere dayanarak ithalatçı objelerine servis teklifleri döndürür.



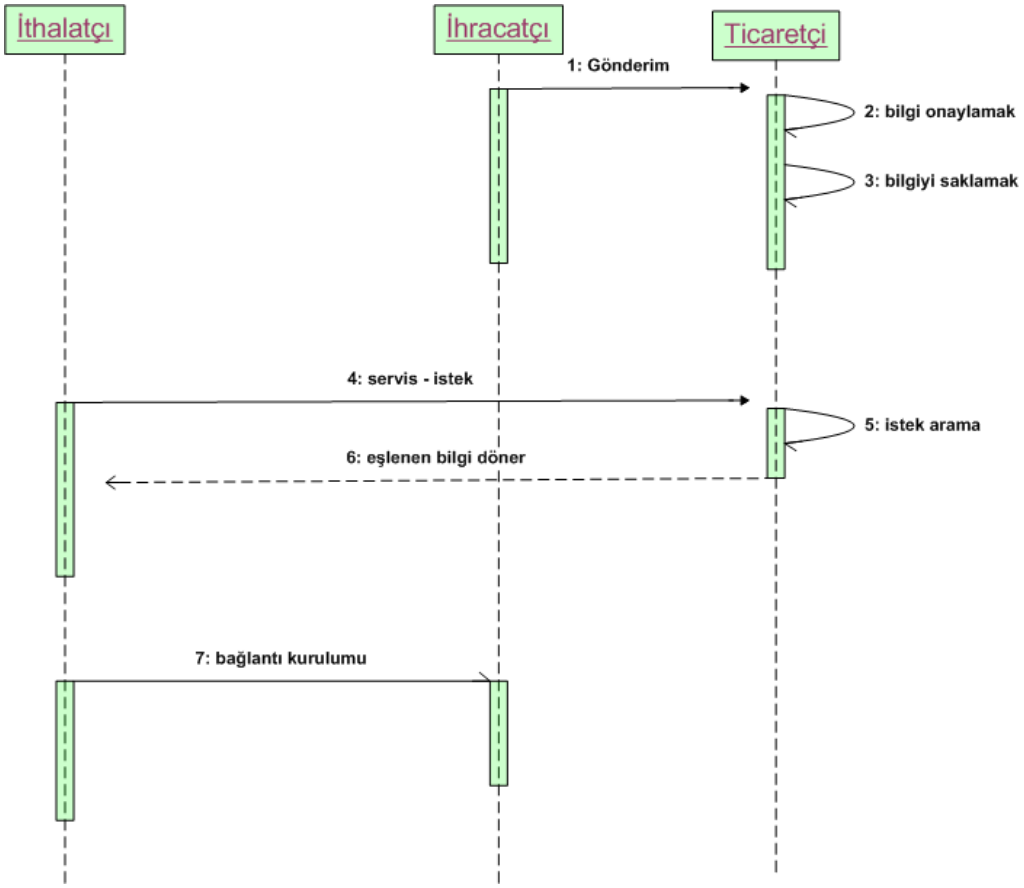
Şekil 2.6 Servis ticareti

Trader- (ticaretçi) – ihracatçı objelerden servis tekliflerine kaydeden ve bazı kriterlere dayanarak ithalatçı objelerini servis tekliflerine döndüren bir roldür.

Exporter-(ihracatçı)- ticaret objeleriyle birlikte servis tekliflerini kaydeden roldür.

Importer-(ithalatçı)- ticaret objelerinden sağlanan bazı kriterler ve servis teklifleri elde eden roldür.

Yürürlükteki ticaretçi, servis tabanlı mimaride "çöpçatan rolünü oynar. Bu, Şekil 2.7' de gayri resmi dizi diyagramında gösterilmilşir.



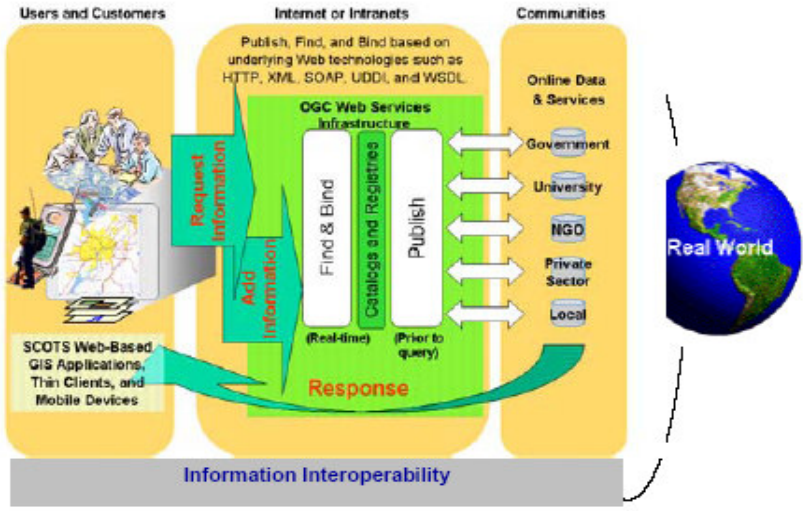
Şekil 2.7 Servis ticareti etkileşimi

İhracat (export) yapmak için, bir obje tüccara servis örneği elverişli olan ve ara yüz tanımlarını içeren bir servis tanımı verir. İthalat (import) yapmak için, bir obje tüccardan belirli karakteristikleri olan servisler ister. Tüccar servis açıklamalarına karşı kontrol yapar ve cevabını servis örneklerine bağlanmak için gereken bilgiyle birlikte verir. Servislerin tipine kısıtlanmış tanımlara ve çeşitli poliçelere göre seçenekler birbirine uyan tekliflere uygulanır. Tercihlerin uygulamaları eşlenmiş tekliflere, ithalatçı (importer) ile döndürmek için bunların sırasını belirler.

2.7 Girişimde OpenGIS Web Servisleri Sistemi

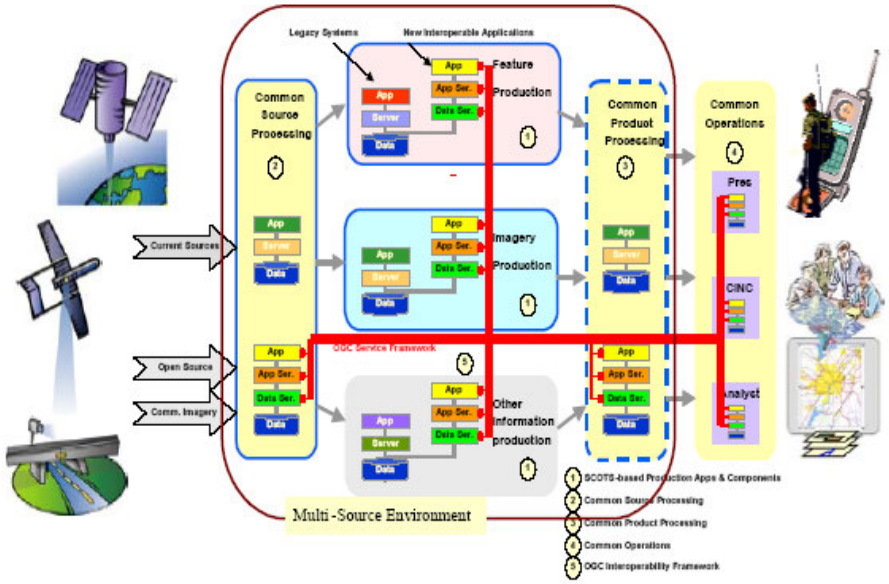
OpenGIS web servisleri sistemi yatırım kapsamlı birlikte işlerlik sağlayabilmek için Şekil 2.8'de yatırımın kısa süreli fonksiyonel parçalarını ve ortak arayüzler kümesini

sağlar. Yatırımın tanımı kullanıcılar ve müşteriler oluşumu, bilgisayar ağları ve coğrafi konumsal komineteler, gerçek yaşam özelliklerini ve fenomeni temsil eden veri holdingleridir.



Şekil 2.8 Girişimde OpenGIS web servisleri
(Kaynak Numarası : 30)

Bilgisayar ağ kaynakları daha geniş bir kullanım için yayımlanır. Uygulamalar basit istek-cevap etkileşimleri yoluyla, network kaynaklarıyla birlikte çoğalır, keşif yapar ve etkileşir. OpenGIS web servisleri sistemi gelecekteki dağıtılmış coğrafi konum ve mekan sistemleri için bir platformdur. Bu platform çoklu network uyumlu coğrafi konumlar ve mekan servislerini bir araya getirmek için uygulamalara imkan verir. Standart web servisleri teknolojilerini birleştirir. Komuniteler ve kaynaklara bağlı olarak bilgi birlikte işlerliğine izin verir. Gerçek dünya ve dağıtılmış kullanıcılar arasındaki bariyeri azaltır.



Şekil 2.9 Çok kaynaklı bilgi operasyonları

(Kaynak Numarası : 30)

Yukarıdaki şekilde, yatırım kapsamlı birlikte işlerliğe temel olan OpenGIS servisleri sisteminin tipik bilgi üretim çevresinin yerini aldığı kavram sistemi gösterilmektedir. Ürün dünyasının parçası olarak altı tane temel fonksiyonel parçalar yer almaktadır.

Ortak Kaynak İşlemi (Common Source Processing) kaynak kazanımlarını değerlendirmenin ve işleminin yapıldığı yerdir.

Özellik Üretimi (Feature Production) özellik ürünü ve işletmenin olduğu yerler.

Betimleme üretimi (Imagery Production) betimleme ürününün ve işletmenin olduğu yerler.

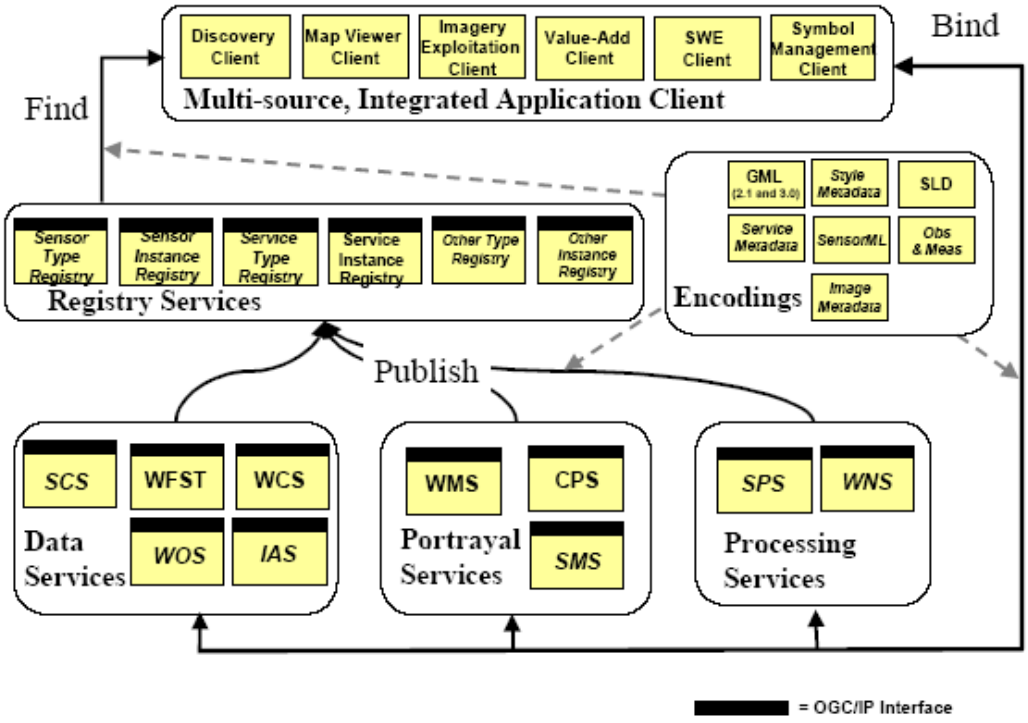
Diğer Bilgi Üretimi (Other Information Production) çok kaynaklı işlemlerin, özelleştirilmiş bilgi ürününün ve bunların işletilmesinin yapıldığı yerdir.

Ortak Ürün İşlemi (Common Product Processing) bitirme ve dağıtım kapasitesi ana ürünü.

Ortak İşlemler (Common Operations) müşterilerin yayılmış ürünleri kullanmak ve sergilemek için bulunan ortak kapasitelerdir.

2.7.1 OpenGIS Web Servisleri Sisteminin Bileşenleri

OpenGIS servisleri sistemi (Şekil 2.10) geniş yatırımlı birlikte işlerliği ve kısa süreli fonksiyonel parçaların ortak arayüzlerini sağlar.



Şekil 2.10 OPENGIS web servisleri iskelet bileşenleri
(Kaynak Numarası : 32)

OWS 1.2 servis test platformlarının mimari elemanları şunlardır:

Müşteri-İstemci Servisler (client service) Müşteri uygulamalarının kullanıcılarla etkileşim halinde olduğu, müşteri tarafının bileşenleri ve diğer servis taraflı müşteri uygulamalarının, servis tarafıyla etkileşimde olduğu uygulama servisleridir aynı zamanda veri servisleridir.

Kayıt Servisleri (registry service) sınıflandırmak, kayıt yapmak, tanımlamak, araştırma yapmak, sürdürmek ve network kaynakları hakkında erişebilir bilgi için ortak bir mekanizma sağlar.

İşlem-iş akışı servisleri (processing-workflow service) coğrafi konumsal veri ve metaveriyle işlem yapan değer ekleme servisi sağlayan uygulama kurma ve blok-servis kuruluşudur. OWS 1.2 için işlem-iş akışı servisleri Sensör Planlama Servisi (SPS) ve Web Hatırlatma Servisi içerir.

Betimleme servisi (portrayal service) betimleme servisleri coğrafi konumsal bilgileri görsellemeyi destekleyen özellikli kapasiteleri sağlar.Betimleme servisleri, verilen bir veya iki input, haritacılıkla betimlenen haritalar üretilmesi, arazi bakış açısı, dipnot görüntüleri, dinamik olarak değişen uzay ve zaman özellikli görüşlerin bileşenleridir. OWS 1.2 için, betimleme (portrayal) servisleri, Web Harita Servisini (WMS), Kapsam Betimleme (Portrayal) Servislerini (CPS) ve Stil İşletme Servislerini (SMS) içerir.

Veri servisi (data service) veri gereksinimini özellikle coğrafi konumsal veriyi sağlayan servis kurma blokları kuruluşudur. OWS 1.2 için veri servisleri, Web Nesne Servisini (WOS), Web Özellik Servisini (WFS), Sensor Koleksiyon Servisini (SCS), Görüntü Arşiv Servisini (IAS) ve Web Kapsam Servisini (WCS) içerir.

2.8 Arayüzler

Bir arayüz belki, bir veya daha fazla operasyonlar olarak tanımlanır, fakat tekrar kullanımın azami dereceye çıkarılması arayüzler kümesini tanımlamakta daha kullanışlı olacaktır. Diğer bir deyişle sabit ve en uygun yakınlık, birlikte işlerliği sentetik ve anlamsal seviyelerde terfi ettirmektir. Burada önerilen servis modeli, verilen herhangi bir servisin normal olarak uygulanmasını küçük gruplara ayıran hiyerarşik arayüzler içermektedir. Daha genel operasyonların uzmanlaşması ve yeniden kullanımı arayüz kalıtımı olarak karakterize edilir. Bu hedefin en büyük amaçlarından biri, bir servisten daha ortak elemanlarla vurgulanmasının sağlanmasıdır. Diğer bir önemli amaç ise kolay anlaşılır arayüz bileşenlerini sağlayarak inşa ve uzmanlaşmayı kolaylaştırmaktır. Özetle tasarım hedefi şu motivasyonları içerir:

-arayüzlerin tekrar kullanımı ve uzmanlaşmaya olanak vermek

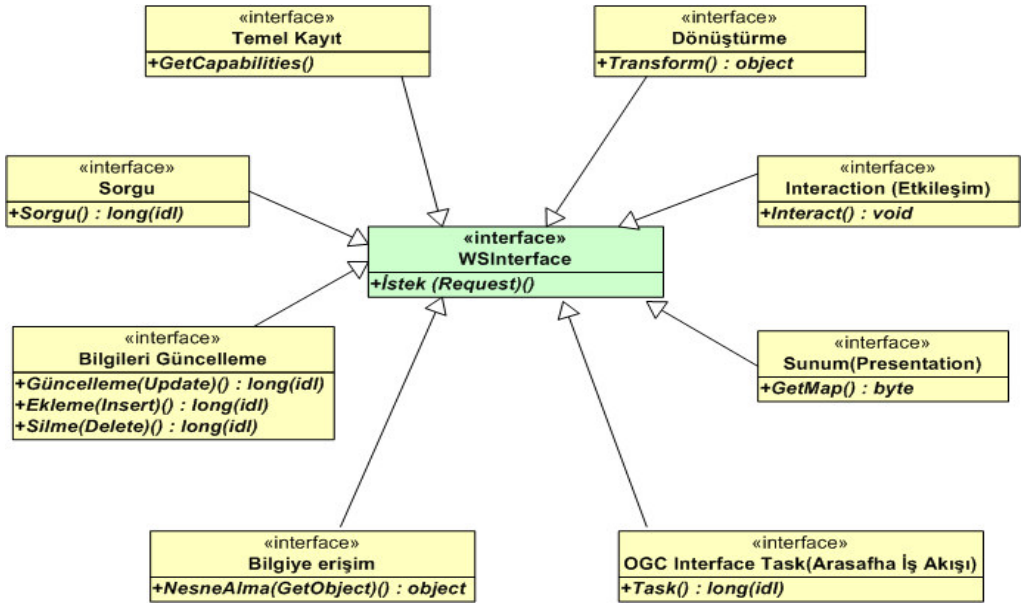
-birden fazla ortak elemanlara odaklanması (arayüzler, operasyonlar, mesajlar, içerik)

-yeni servisleri kolay inşa etmesi ve kavraması

-inşa ve servislerle etkileşimi “contracts” tanımlamak için kolay anlaşılır arayüzler sağlamak

2.8.1 Arayüz Hiyerarşisi

Önerilen arayüz hiyerarşisi, yüksek seviyeli arayüzlerin küçük sayılarla düzenlenmesi Şekil 2.11'de resimlendirilmiştir. Şekilde gösterilen sekiz arayüz, arayüzlerin detayı ve arayüz hiyerarşisi en iyi ölçüdeki (toplevel) arayüzlerden türerken aşağıda tanımlanmıştır. Her en iyi ölçüdeki arayüz, ayrıca o arayüzden türeyen bütün arayüzlere karşılık gelir. Bütün arayüzler soyut WSInterface arayüzü uzmanlaşması için gösterilmiştir; var olan operasyonları aşırı yükleyerek ekstra operasyonlara ve içerik tiplerine ekler. Sadece operasyon ve onun dönüş tipi burada tanımlanmıştır. Operasyonların dokümantasyonları servis uygulamaları için kendi kendilerine imzalanır. Bizim uygulama statümüz GetMap(), GetCapabilities() ve GetFeatureInfo() metotlarını kullanmıştır. Bunlarda Web Haritalama Servisi WMS ve WFS metotlarıyla alakalıdırlar.



Şekil 2.11 En iyi ölçüdeki arayüzler

2.9 Servis Tanımları

UML diyagramları aşağıdaki şekilde her servis tanımı için gösterilmiştir. Takip eden alt başlıklarda servis tiplerinin sınıfları ve onları tanımlayan kombinasyonları aynı zamanda arayüz içerikleri tartışılmıştır. Her servis uygulamaları, temel kayıt arayüzü ve servis bilgi içeriği uygulamaktadır.

2.9.1 Haritalama

Bu servis sınıfları, verilmiş ve oluşmuş sunum haritalarına konsantre olur ve verilen arayüzlere minimum şekilde uygular. Sunulan harita tabakaları hakkında bilgi sağlama için istek arayüzüde GetFeatureInfo uygulayabilirler. Buna ek olarak işlem arayüzleri de desteklenir.

2.9.2 Nesne İdaresi

Bu servisler minimum düzeyde istekleri, güncellemeleri ve transfer aryüzlerini uygun olduğu yerde destekleyebilirler. Servisler sınıfı olarak, kaynakların düzeltilmesine ve depolamaya, o kaynakların içeriğine veya tipine dikkat etmeden odaklanır.

2.9.3 Kayıtlar/Kataloglar

Kayıtlar veya kataloglar meta veri işletimine odaklanır ve istek, güncelleme, çevirme, görevlendirme veya başka meta veri üzerindeki operasyonlarına, meta veriden çalışma veya kurma sağlar. Her ne kadar obje ve özellikli idari servislere fonksiyonel olarak benzer olsa da sundukları meta veri ve içerik tarafından gerçekten ayırt edilmiştir.

2.9.4 Özellik İdaresi

Bu servisler özelleştirilmiş veri için obje idaresi uygularlar.

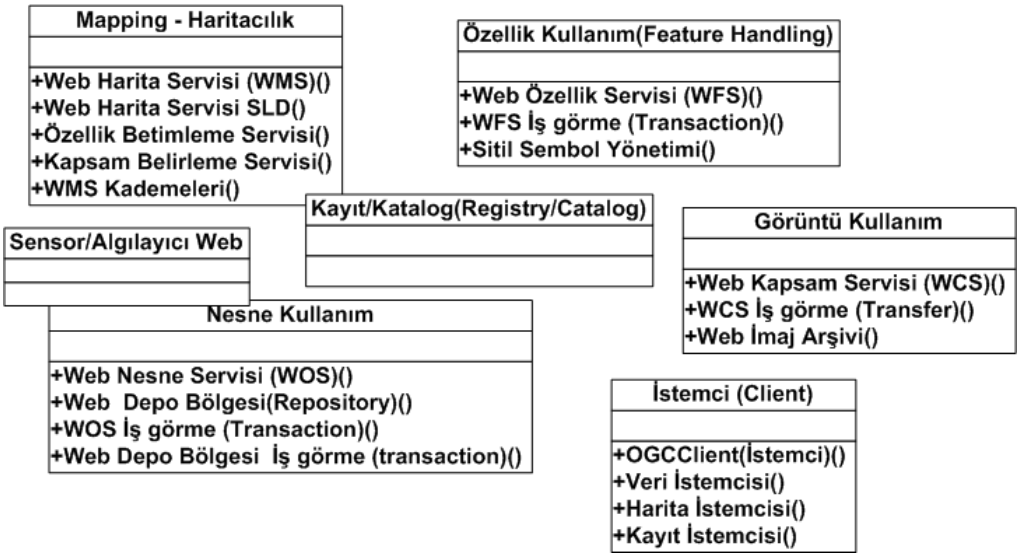
2.9.5 Görüntü İdaresi

Bu servisler görüntü verisi için obje idare arayüzleri uygularlar. Verilen çeşitli yollarda gördük ki, çok fazla sayıdaki servislerde olacak potansiyel bu arayüzlerde vardır.

2.10 Servis Tipi Açıklamaları

2.10.1 Genel

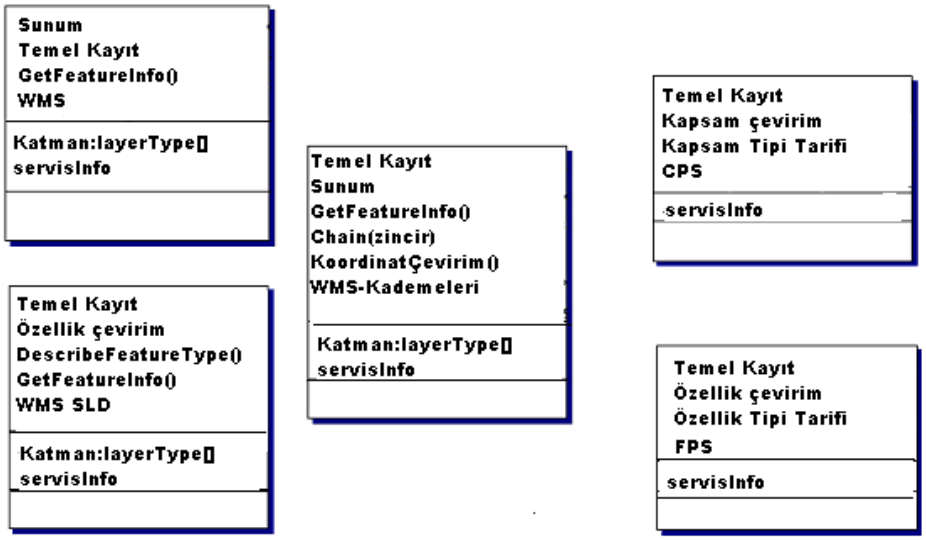
Şekil 2.12 özellik (fonksiyonalite) ve kullanım tanımlarını toplevel paketlerle sunar. Hangi arayüzlerin ve içerik tiplerinin hangi servis parçasında olduğu aşağıdaki bölümlerde tanımlanmıştır.



Şekil 2.12 OGC servislerinde sınıf diyagramları

2.10.2 Haritacılık Servisleri

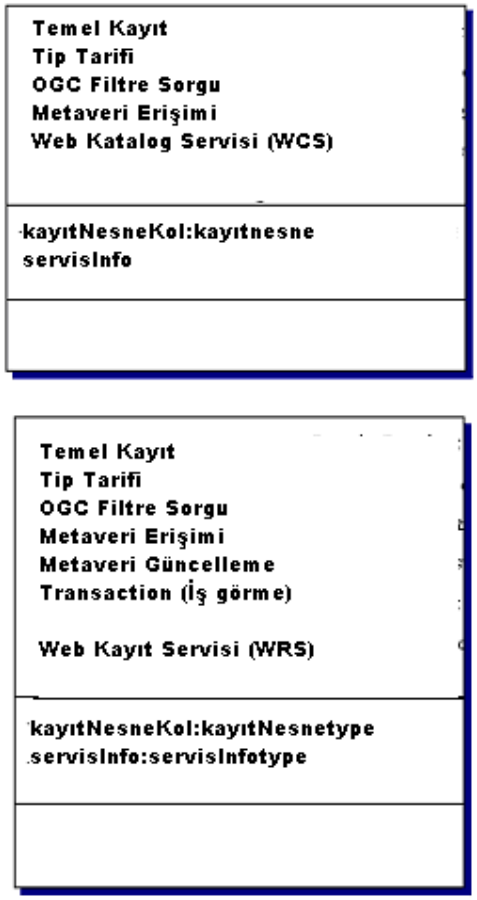
Web haritacılık servislerine (WMS) ait sınıflar ve bu sınıflara ait yöntemler aşağıdaki tablolarda belirtilmiştir.OGC uyumlu web servislerinden WMS servisinin kullanılmasında bu tip sınıflar kullanılmaktadır.



Şekil 2.13 Haritacılık sınıf diyagramları

2.10.3 Kayıt/Katalog Servisleri

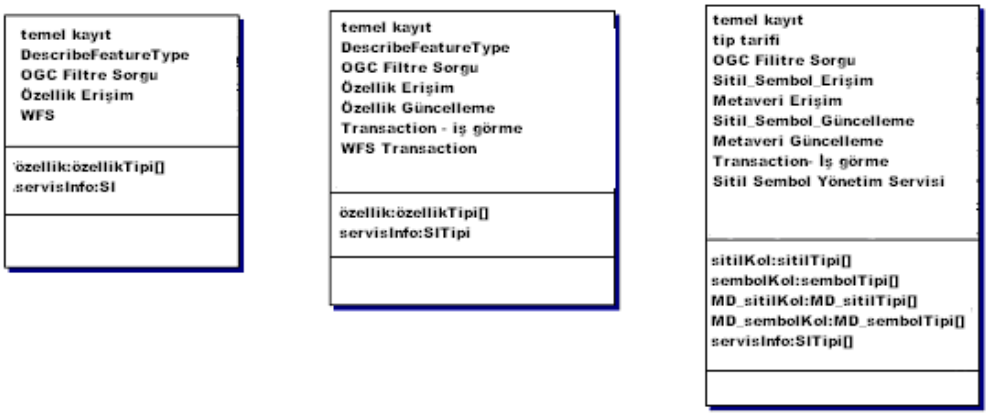
OGC uyumlu servislerden web kayıt servisi (WCS) herhangi bir uygulamada kullanılacak ise, uygulamnın içeriğindeki sınıflar ve bu sınıfların yöntemleri aşağıdaki tablolarda ifade edildiği gibidir.



Şekil 2.14 Kayıt/Katalog servis sınıfları

2.10.4 Özellik Belirleme Servisleri

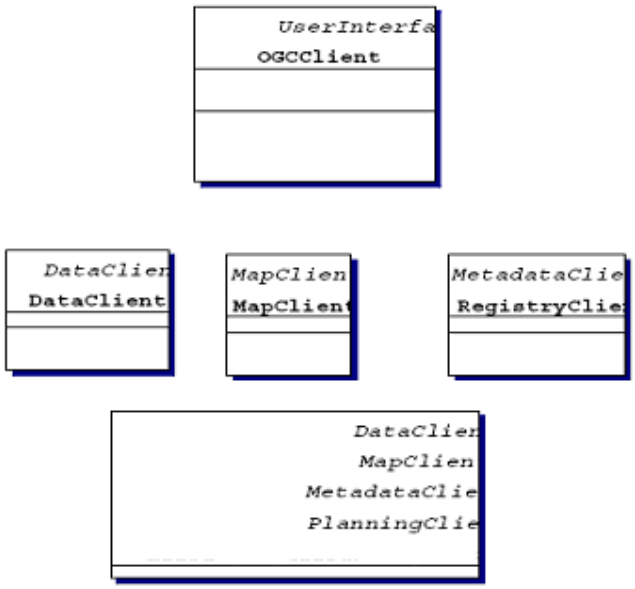
OGC uyumlu servislerden web özellik servisi (WFS) herhangi bir uygulamada kullanılacak ise, web servisine ait sınıf yapıları ve bu sınıflara ait yöntemler, fonksiyonlar aşağıdaki gibi ifade edilmelidir.



Şekil 2.15 Özellik belirleme servis sınıfları

2.10.5 İstemci Servisler

OGC uyumlu web servislerin istemci/sunucu entegrasyonlarında, istemci kısımlarında sunucu web servislerinin özelliklerini ve sonuçlarını kullanıcı platformuna taşıyacak bir takım özel sınıflara ve yöntemlere ihtiyaç duyulmaktadır. Bu ihtiyaçların karşılanması tamamen istemci tarafında geliştirilen sınıflar yardımı ile giderilmektedir. OGC uyumlu web servislerinin istemci tarafında nasıl kullanılması ve çağrılması gerektiğini ifade eden tablo yapısı aşağıdaki gibi ifade edilmektedir.



Şekil 2.16 İstemci servis sınıfları

BÖLÜM 3

OGC TABANLI HARİTALAMA VE ÖZELLİK SERVİSLERİ

OGC, GIS standartlarında web servisleri ortak uygulama şartnamelerini sonradan yayımlamıştır. Bu yüzden en son OGC web servis şartnamesiyle birtakım farklılıklar vardır. GIS web servis uygulamalarımızda WMS'nin getCapabilities ve getFeatureInfo fonksiyonları için döndürülen tipler dizgi olarak tanımlamıştır. Döndürülen dizgiler XML'de yapılanmış verilerdir. Dizgiler aslında istenilen biçime göre XML, HTML, plain text veya GML olabilir. GetMap istenmesi durumunda; WMS SOAP mesaja eklenmiş işleyici veri objeleri MIME tipinde yani imaj/jpeg olarak döndürür. OGC farklı tip isteklere göre farklı tipte döndürür. Döndürülen tipler için uygulamalarımız OGC web servisleri şartnamelerine göre geliştirilmiştir. GIS web servislerini ilk defa uygulamaya başladığımız zaman OGC bu yeni şartnameye sahip değildi. Uygulamadaki istek cevap objelerine ve onların şemalarına bu şartnameyi uygulayarak, OGC uyumlu servislerin özelliklerini kullanmış olacağız.

WFS tabanlı web servislerimize ilişkin olarak, istek ve cevap objelerinin XML formundaki tipi dizgidir. Her cevap ve istek kendine ait şema dosyasına sahiptir. Bunlar bu şemalara ve verilen parametrelere göre yaratılmıştır. Objeleri XML olarak yarattıktan sonra, web servis CBS çevresinde, XML objeleri uzantılı SOAP zarflarına konur. İstekler ve cevaplar SOAP mesajda HTML üzerinden taşınır.

3.1 CBS (GIS) için Web Servisleri

Web Servisleri çeşitli platformlar üzerinde çalışan farklı yazılım uygulamalarının birlikte işlerliğini anlatır. Web servisleri bir network üzerinden makinadan makinaya etkileşim ve birlikte işlerliği destekler. Her web servisinin makinadan makinaya okunabilir biçimde tanımlanmış bir arayüzü vardır. Web servis arayüzleri WSDL kullanılarak standartlaştırılarak tanımlanır. WSDL dosyaları servis ve servis protokolleri için input ve output özellikleri tanımlar. WSDL dosyaları XML dokümanları olarak yazılırlar. WSDL, web servislerini tanımlamak ve yerlerini

belirlemek için kullanılır. Web servisleri WSDL in dört ana elamanıyla tanımlanır. Bunlar "porttype", "message", "types" ve "binding" dir. Porttype web servisleri ve bu operasyonlar için sağlanan operasyonların veri elemanlarını tanımlar. Types web servisleri tarafından kullanılan veri tiplerdir. Binding ise protokol iletişimini tanımlar. Diğer sistemler SOAP mesajları kullanarak tanımlanan şekliyle web servisleriyle etkileşim halindedir. SOAP, dağıtımlı çevrede bilgi alışverişi yapmak için XML tabanlı mesaj protokolüdür. Bu XML dokümanları için çeşitli network taşıma protokolleri üzerinden standart paketleme yapıları sağlar. Bu yapı üç farklı parçadan oluşur. Bunlar zarf, şifreleme kuralları, ve Remote Procedure Call RPC (uzaktan prosedür çağrısı). SOAP, HTTP gibi başka protokollerle de bir kombinasyon içinde kullanılabilir. OGC uyumlu web servisleri HTTP üzerinden SOAP kullanmaktadır. Web servislerinin CBS alanındaki avantajları üç kategoride gruplaşabilir:

Dağıtım: Coğrafi konumsal verileri ve uygulamaları platformlar, işletim sistemleri ve bilgisayar dilleri üzerinden dağıtmak daha kolaydır. Bu işleçler platform ve dilce tarafsızdır.

Bütünleştirme: Uygulama geliştiricileri için, kendi uygulamalarında coğrafi konumsal özellik ve veriyi bütünleştirmek daha kolaydır.

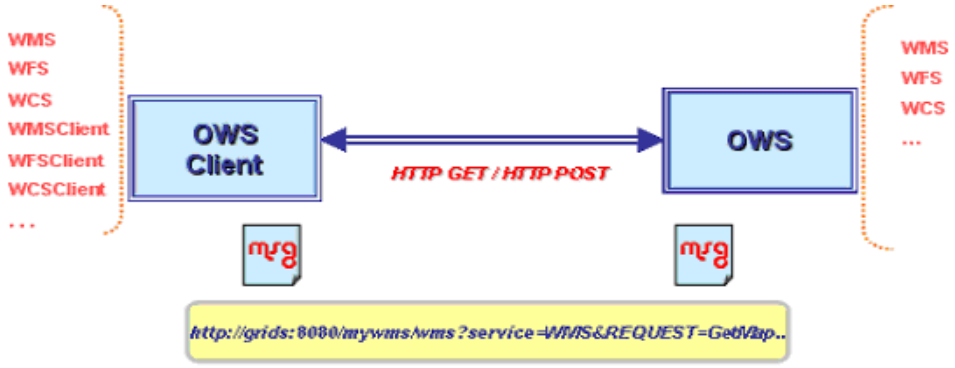
Altyapı: Altyapının avantajını web servisleri mimarisini inşa ederken görebiliriz. Altyapı bunu yaparken geliştirme araçlarını, uygulama servislerini, mesajlaşma protokollerini, güvenlik altyapısını ve iş akışı tanımlarını içerir.

GIS servisleri üç farklı kategoride gruplaşır. Bunlar veri servisleri, işlem servisleri ve kayıt veya katalog servisleridir. Veri servisleri belirli veri kümeleri ve verilerin parçalarına erişmek için yapılan tekliflerle sıkıca bağlanmıştır. İşlem süreci servisleri kişi tanımlı parametrelerle tanımlanmış işlem ve veri dönüşümü yapılmasını sağlar. Kayıt veya Katalog servisleri kullanıcıların ve uygulamaların sınıflandırma, iyi halde tutma, kayıt yapma, tanımlama, araştırma ve web servisleri hakkında bilgi erişimine

izin vermektedir. Coğrafi fiziksel uygulamalar için yapılan CBS web servislerinin geliştirilmesinde, WFS'yi veri servisleri olarak, IS'yi katalog-kayıt servisleri olarak ve WMS'yi de işlem servisleri olarak kullanabilmekteyiz.

3.2 Mimari Yapı

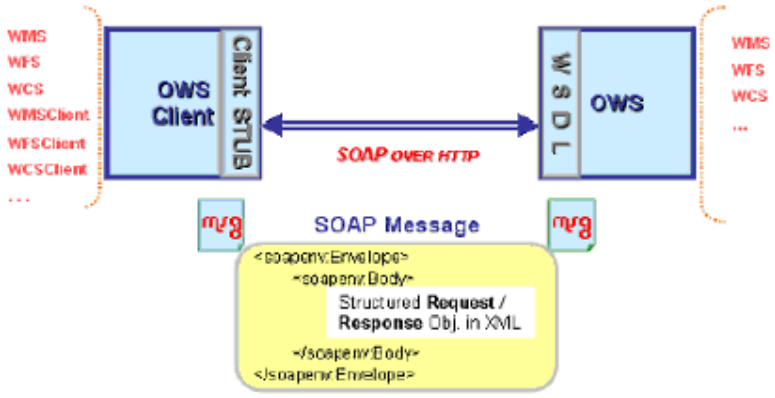
Bu bölüm OGC uyumlu CBS görselleme sistemleri için web servisleri teknolojilerinin bütünleşmesinin tüm detaylarını vermektedir. Şekil 3.1 ve 3.2'de dağıtılmış hesaplanan platformlar için iki farklı taslak hazırladım. Biri OGC kullanılarak ve şartnamelerinde standart olarak tanımlanmıştır. Diğeri ise web servisleri kullanılarak yapılmıştır.



Şekil 3.1 Mevcut OGC özellikleri ile HTTP AL/YAYINLA yönetimi ile servise istek yayınlanması

(Kaynak numarası : 49)

Her operasyonun çevrimiçi kaynakları OGC servisi tarafından yani HTTP url'siyle desteklenir. URL her operasyon için farklı olabilir veya servis sağlayıcısının kendi kararıyla aynı olabilir. Her url IETF RFC 2616 tanımıyla uymalıdır. Sadece istek parçası OGC WMS şartnameleri tarafından tanımlanan servis isteklerini kapsar. HTTP iki istek metodunu destekler: Al ve Yayınla. Bunların biri veya ikisi sunucu tarafından teklif edilebilir ve çevrimiçi kaynakların url'si iki durumda da değişir. Al yöntemi için destek zorunludur. Yayınla yöntemi için ise isteğe bağlıdır.



Şekil 3.2 Genişletilebilir SOAP zarflarındaki mesaj yapıları ile web servislerinin çağrılması

(Kaynak numarası : 49)

Sunucu uygulandığı zaman (web servis olarak) SOAP tarafından HTML üzerinden takas yapılır. Web servisleri standartlaştırılmış XML mesajlaşma sistemi tarafından istenir. SOAP, sunucular arası bilgi alışverişi için XML tabanlı protokoldür. SOAP çeşitli mesajlaşma sistemlerinde kullanabilinse de, çeşitli ulaşım protokolleriyle de getirilebilir. SOAP'ta asıl odaklanılması gereken HTTP ile RPC'nin ulaşımıdır. Şekil 3.4, 3.5, 3.6'da gösterilen Şekil 3.2'deki SOAP mesajının tanımlandığı kutu örnek isteklerle değiştirilmiştir. Bunlar web servisleri olarak uygulanan WMS fonksiyonalitelerinin çağrılması için oluşturulan şema dosyalarıdır. İstek örnekleri bu şema dosyalarına göre yaratılmıştır. İsim, numara ve tip istek parametreleri, sunucu tarafında geçerlilik ve şekillendirme için kontrol edilir. SOAP ana kısmında yapılandırılmış istekleri taşır. Sunucular WSDL dosyalarında tanımlanmış herkese açık arayüzleri çalıştırırlar. WSDL, web servis için tanımlanan herkese açık arayüzler için XML dilidir. Bu herkese açık arayüzler, herkese açık fonksiyonlardaki bilgileri, XML mesajları için veri tipi bilgilerini, kullanılacak olan belirli taşıma protokolleri hakkında bağlayıcı bilgiyi ve belirlenmiş servislerin yerini saptamak için adres bilgilerini içermektedir.

3.3 OGC uyumlu WMS Haritalamasında WSDL Dokümanı

WMS coğrafi verilerden haritalar üretmektedir. Haritalar, genellikle WFS gibi bağlantılı CBS servislerinden elde edilmiş, işlenmemiş coğrafi veriden bilgi üretir. Haritalar genellikle jpeg, gif ve png gibi resimli biçimlerle gösterilir. WMS ayrıca ölçeklenebilir vektör grafiklerinden vektör tabanlı grafik elemanlar üretir. Şartnamelerde tanımlanan iki tür WMS vardır. Bunlar temel WMS ve SLD uyumlu WMS. Temel WMS için üç tane işlem tanımlanmıştır. Bunlar getCapabilities, getMap ve getFeatureInfo'dur. Eğer WMS SLD uyumlu ise o zaman dört tane daha desteklenmiş işlem olur. Bunlar describeLayer, getLegendGraphics, getStyles ve putStyles'dır. DescribeLayer, bir harita tabakasına XML tanımlaması sormak için kullanılır. GetLegendGraphics, büyük semboller elde etmek için kullanılır. GetStyles, WMS'den kullanıcı tanımlı stilleri düzeltmek için kullanılır. PutStyles, WMS'de kullanıcı tanımlı stilleri saklamak için kullanılır. Bu dokümanda buradan itibaren temel WMS'den bahsedeceğiz. HTTP, OGC WMS şartnamesiyle desteklenen dağıtılmış taşıma protokolüdür. WMS operasyonları url biçiminde istekler göndererek çağrılır. Bu url'lerin içeriği isteklerin parametrelerine ve operasyonlara bağlıdır. WMS, yeteneğini ve veri holdingini kapasite dokümanında yayınlar. Bu doküman XML'de şifrelenmiştir. WMS, coğrafi veri holdinglerini tabakalarda (layers) sınıflandırır ve bu tabakalar için uygun olan stiller hakkında bilgi verir. Her tabaka bir alt tabakaya sahip olabilir ve her alt tabakanın da kendileri için tanımlanmış farklı stilleri vardır.

Harita servislerinin web servis tabanlı uygulamalarında, HTTP'deki isim/değer çiftlerinden çok, web servislerinin yapılandırılmış XML mesajlarını gönderme yeteneğinden avantaj sağlamaktır. Bu biçimlenmiş mesajlar, standart XML ayrıştırıcıları kullanarak mesajların yapısal geçerliliğini kontrol eder ve daha basit işlemler için veri bağlayıcı sistemler kullanarak veri objeleriyle mesajları bağlar. Ayrıştırma araçları ve veri bağlama servisleri, web servisinin başlangıcında güçlü vurgulanarak XML araçlarının genel amacını temsil eder. Ayrıca web servisleri mesaj merkezlidir ve mesaja yönelik özel yazılımlar kullanarak uygulanabilirler. Böylece başlangıçta bu geliştirmeyi kullandım. Tablo 3.1 ve 3.2'de web servis tabanlı WMS arayüzünün, OGC uyumlu WMS mesajlarının haritalanmasını anlattım. Bu tablolardaki haritacılık işlevi şu anki WMS uygulamalarına uygulanmıştır.

WSDL dokümanında yer alan yöntemler(mesajlar)	WMS yapısında yer alan OGC-İstek tanımlamaları	WMS web servislerine yapılan istek yöntemleri
getCapabilitiyRequest	HTTP AL/YAYINLA SERVICE=WMS REQUEST=getCapabilities	SOAP içerisinde paketlenmiş şema dosya içerikleri
getMapRequest	HTTP AL/YAYINLA SERVICE=WMS REQUEST=getMap	SOAP içerisinde sıkıştırılmış şema dosya içerikleri
getFeatureInfoRequest	HTTP AL/YAYINLA SERVICE=WMS REQUEST=getFeatureInfo	SOAP içerisinde sıkıştırılmış şema dosya içerikleri

Tablo 3.1 OGC-WMS istekleri ile tanımlanan web servisinin WSDL dokümanına ait yöntemleri

WMS'nin WSDL arayüzü Şekil 3.3'de gösterilmiştir. Kısaca şu an ki konumuzla alakalı bölümleri gösterdim. Tablo 3.1'in ilk sütunu WMS tabanlı servislerde desteklenen istek mesajlarını listeler. İkinci sütun, benzer OGC uyumlu WMS isteklerini ve istek tiplerini tanımlamak için bunlarla bağlı parametreleri listeler. Son sütun ise Şekil 3.3'de gösterilen servislerin çağrılması için servis tabanlı WMS cevapları için OGC WMS cevaplarının haritalanmasını tanımlar.

WSDL mesajları	OGC-WMS Cevap Tipleri	WMS tarafından desteklenmiş sonuç olarak dönen tipler	WSDL'de WMS servisinin cevapları
getCapabilitiyResponse	İstek parametresi olarak tanımlanmış, Name:'FORMAT': Value:MIME tipte (default text/xml)	text/xml	Dizgi- veri tiplerinin cevap olarak dizgi şeklinde yazılması
getMapResponse	İstek parametresi olarak tanımlanmış, Name:'FORMAT': -MIME tipte (no default)	image/svg image/jpeg	Object(nesne)-veri tiplerinin (datahandler) veri işleyici obje olarak

			yazılması
getFeatureInfoResponse	İstek parametresi olarak tanımlanmış, Name: 'INFO_FORMAT': MIME tipte (no default)	text/plain text/HTML application/vnd.ogc.gml	Dizgi- veri tiplerinin cevap olarak dizgi şeklinde yazılması

Table 3.2 OGC-WMS cevapları ile WSDL dokümanında tanımlı WMS web servisi cevaplarına ait mesajlar

Tablo 3.2'nin ilk sütunu servis tabanlı WMS'lerle desteklenmiş cevap mesajlarını listeler. İkinci sütun OGC uyumlu WMS isteklerini ve cevap tiplerini tanımlamak için gerekli parametreleri listeler. Şu anda servis tabanlı WMS'lerimiz var ve bunlar da kapasite dosyalarına sahiptir. Bu dosyalarda herbir istek için desteklenmiş cevap tipleri tanımladım ve bunları üçüncü sütunda listeledim. Son sütun benzer istekler için esas cevap tiplerini ve OGC şartnamelerindeki desteklenmiş cevapları listeler. XML dizgilerinde, yapılandırılmış verilerin döndürülen tipleri, dizgi olarak tanımlanmıştır.

<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
targetNamespace="http://localhost:8080/wms/services/WMSServices"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http:// localhost:8080/wms/services/WMSServices"
xmlns:intf="http:// localhost:8080/wms/services/WMSServices"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns1="http://lang.java" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<wsdl:types><schema targetNamespace="http://lang.java"
xmlns="http://www.w3.org/2001/XMLSchema">
<import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>

<complexType
name="Object"><sequence/></complexType></schema></wsdl:types>
<wsdl:message name="getMapResponse">
<wsdl:part name="getMapReturn" type="tns1:Object"/>
</wsdl:message>
<wsdl:message name="getFeatureInfoResponse">
<wsdl:part name="getFeatureInfoReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getCapabilityRequest">
<wsdl:part name="request" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getMapRequest">
<wsdl:part name="request" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getFeatureInfoRequest">
<wsdl:part name="request" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getCapabilityResponse">
<wsdl:part name="getCapabilityReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:portType name="WMSServices">
<wsdl:operation name="getMap" parameterOrder="request">
<wsdl:input message="impl:getMapRequest" name="getMapRequest"/>
<wsdl:output message="impl:getMapResponse" name="getMapResponse"/>
</wsdl:operation>
<wsdl:operation name="getCapability" parameterOrder="request">
<wsdl:input message="impl:getCapabilityRequest"
name="getCapabilityRequest"/>
<wsdl:output message="impl:getCapabilityResponse"
name="getCapabilityResponse"/>
</wsdl:operation>
<wsdl:operation name="getFeatureInfo" parameterOrder="request">

<wsdl:input message="impl:getFeatureInfoRequest"
name="getFeatureInfoRequest"/>
<wsdl:output message="impl:getFeatureInfoResponse"
name="getFeatureInfoResponse"/>
</wsdl:operation>
</wsdl:portType>
+<wsdl:binding name="WMSServicesSoapBinding" type="impl:WMSServices">
+<wsdl:service name="WMSServicesService">
</wsdl:definitions>

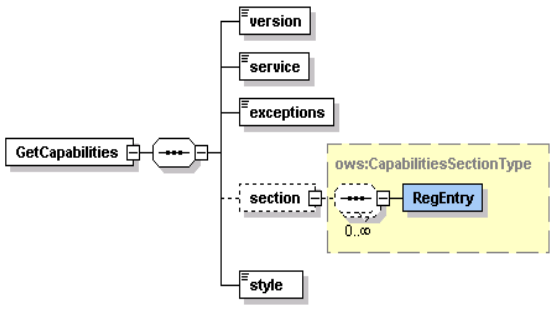
Şekil 3.3 WMS servisine ait WSDL dokümanı

3.4 Web Servis Kullanma Durumunda WMS Servisleri için Geçerli İstek Yaratma

WMS'nin web servis versiyonunu geliştirirken, önceden var olan HTTP AL/YAYINLA yöntemlerini WSDL arayüzlerine çevirdim. Bunu yaparken birtakım küçük problemlerle karşılaştım. WMS iç uygulamaları şu anki WMS şartnameleriyle uyumludur, fakat servis arayüzleri ve bunların servisleri çağırma yolu farklıdır. Servisler HTTP üzerinden SOAP yolu ile çağrılırlar. İstekler XML dokümanları şeklinde yaratılır ve SOAP istek mesajlarının ana kısmıyla bitirilir. Bunlar Şekil 3.4 ve 3.6'da gösterilmiştir. WMS operasyonlarının çağırılması şartnamelere göre olmalıdır. OGC uyumlu istekler WMS şartnamelerinde iyi şekilde tanımlanmıştır, istekler OGC ile uyumlu şartnamelerde tanımlanan ve bu kurallara uyan isim, sayı ve değerlere sahip olmalıdır. Bu bölümde bu istekleri XML şema dosyaları formunda tanımladım. Bu dosyalar, WMS tarafında web servisleri olarak uygulanmış operasyonlarda kullanılmak üzere oluşturulmuştur.

İstemciler WMS müşteri tarafında yaratılır. İstek hazır olduğu zaman, müşteri WMS'ye SOAP mesajı olarak bir istek gönderir. WMS, her servis için web servislerini planına göre yerleştirir. Müşteriler, müşteri fonksiyonlarını belirli web servisler çağrılmadan kullanmalıdır. WMS'deki bütün servisler tek bir dizgi parametresi alır. Bu parametre kendine bir istek yapar. Bu istekler aslında dizgi biçiminde XML

dokümanlarıdır. Şekil 3.4 ve 3.6'da gösterilen şema dosyaları, OGC WMS şartnamelerinde tanımlanan benzer OGC HTTP AL/YAYINLA yöntemlerinin tüm eleman ve değerlerini içerir.



Şekil 3.4 GetCapabilities istek şeması.

(Kaynak numarası : 8)

GetMap isteği bizim WMS uygulamalarımız için oluşturulmuştur. Daha model yaratma yeteneğini uygulamadık. WMS destekli model yaratma SLD uyumlu WMS olarak adlandırılır fakat biz bunu uygulamalarımızı anlatırken tartışmadık. OGC ve SLD şartnameleri, kullanıcı tanımlı simgeleme özelliği için bir mekanizma tanımlar. Bir SLD uyumlu WMS, web özellik servisinden gelen özellikli veriyi düzeltir ve kullanıcı tarafından sağlanan açık simgeleme bilgisini sağlamak için uygulamalar yapar. Projemizde şu zamana kadar temel WMS'yi uyguladık fakat WMS'nin getMap isteğine bağlı elemanları kullanmadık.

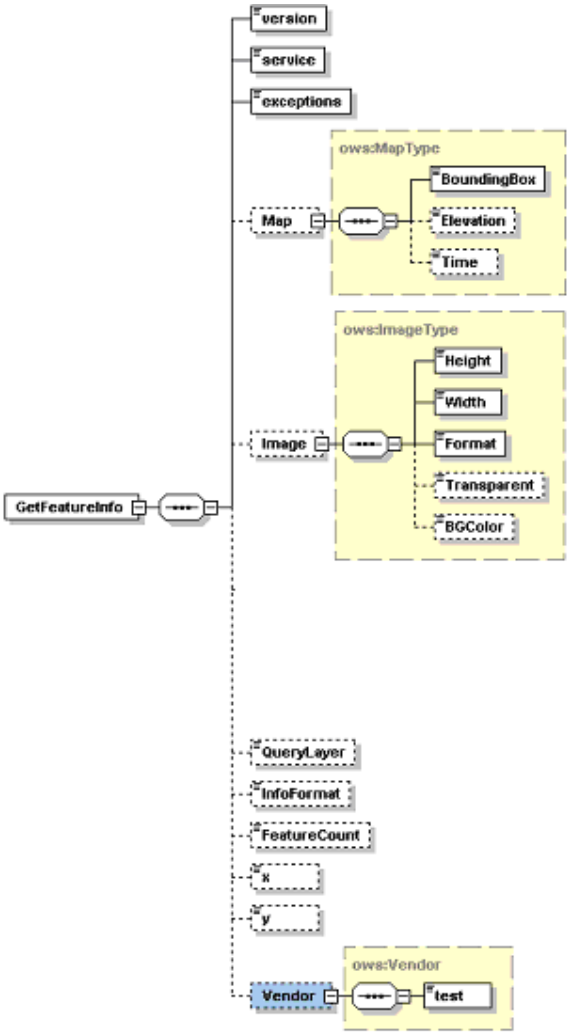


Figure 3.5 GetFeatureInfo istek şeması.

(Kaynak numarası : 8)

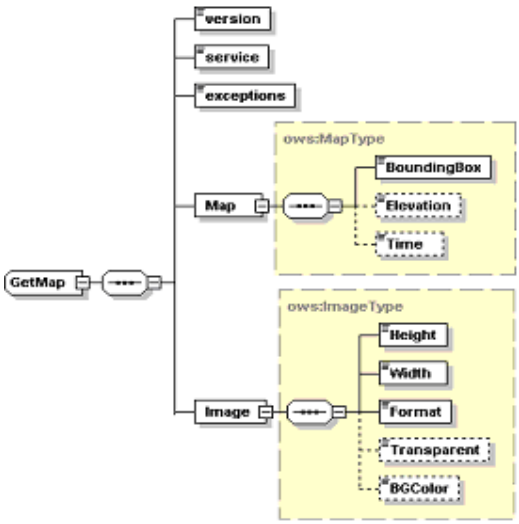
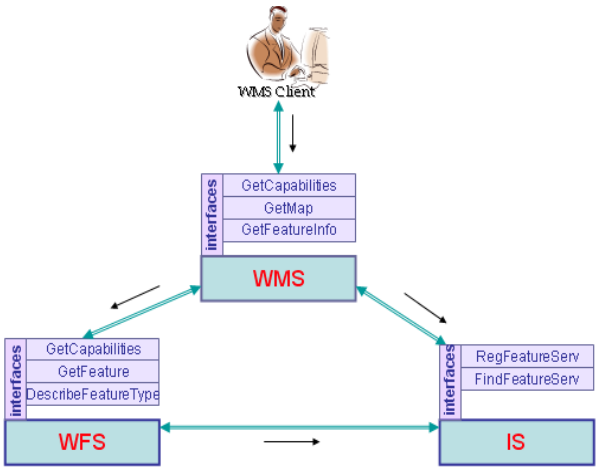


Figure 3.6 GetMap istek şeması.
(Kaynak numarası : 8)

3.5 Görselleme Sistemi İçeren Diğer CBS Bileşenleri

WMS servis uyumlu web servislerimiz, web özellik servislerinin ve IS'in gereken görevleri tamamlamasına dayanmaktadır. Bu bölüm WMS'nin diğer servislerle olan etkileşimini açıklamaktadır. Bu üç servisin aralarındaki etkileşimin genel görüntüsü, Bölüm 3'te Şekil 3.7'de gösterilmiştir. Önceki başlatmalar siyah oklarla gösterilmiştir. Bütün servisler web servisleri olarak uygulanmıştır.



Şekil 3.7 Görselleme sisteminde temel CBS bileşenleri
(Kaynak numarası : 8)

3.6 Web Harita Servisi (WMS) İşlemleri

Bu işlemler için tanımlanan üç işlem; getCapabilities, getMap ve getFeatureInfo'dur. Bu bölüm WMS işlemlerinin HTTP'de kullanımını ve uygulamalarını belirlemektedir.

3.6.1 GetCapabilities (kabiliyetleri alma fonksiyonu) (zorunlu)

3.6.1.1 Genel

Web harita servisinin belirli bir durumunda, getCapabilities isteğinin cevabı servisin kendisi hakkında genel bir bilgidir ve uygun haritalar hakkında belirli bilgilerdir.

3.6.1.2 GetCapabilities İsteklerinin Gözden Geçirilmesi

WMS'den bu isteği yapınca, müşterinin özellikle WMS hakkında bilgi araştırdığı belirtilmelidir. Böylece aşağıdaki Tablo 3.3'te gösterilen servis parametre isteğinin değeri WMS olmalıdır.

İstek Parametreleri	Zorunlu/Opsiyonlu	Tanım
VERSION=version	Opsiyonlu	İstek versiyonu
SERVICE=WMS	Zorunlu	Servis tipi
REQUEST=GetCapabilities	Zorunlu	İstek ismi
UPDATESEQUENCE=number	Opsiyonlu	Hafıza kontrölü için belirli numaralar

Tablo 3.3 GetCapabilities istek url'sinin parametreleri

3.6.1.3 İstek Parametreleri

Versiyon=versiyon

Version=version

WMS versiyon 1.00'da, bu parametrenin adı "WMTVER"idi. Şu anda bu isme karşı çıkılıyor fakat geçmişteki uyumluluklar ve versiyon görüşmeleri için, bir post-1.00 servisi, servis istisnalarını mesele yapmadan bu şekliyle kabul etmelidir. Versiyon ve WMTVER'in birlikte verildiği durumlarda versiyon ifadesi önceliklidir.

Servis=servis-adı

Service=service-name

Servis, hangi uygun servis tiplerinin hangi servis örneklerinde çağrıldığını belirtmektedir. Bu parametre, birçok OGC web servisi için aynı ön ekli url'leri XML kapasitelerine önermeye izin verir. Bu veya sonraki şartnamelerde WMS üzerinden getCapabilities çağrılırken, servis adı değeri "WMS" kullanılmalıdır. Bu işlemi versiyon 10.6 uygularken yapılırsa, servis parametresi müşteriler tarafından kullanılmamalıdır ve sunucu tarafından görmezden gelinebilir.

İstek=Kabiliyetleri Alma

Request=GetCapabilities

GetCapabilities işlemini çağırmak için "GetCapabilities" değeri kullanılmaktadır. WMS'nin 1.00 versiyonunda bu parametrenin adı "capabilities"idi. Şu anda bu isme karşı çıkıldı, fakat servis istisnaları konusu dışında önceki uyumluluklar kabul olmalıdır. Bir müşteri WMS'nin bilmediği versiyonuyla kontak kurarsa o zaman müşteri yeniden elde etmeye hazırlanmalıdır.
Request=GetCapabilities başarısız olursa Request=Capabilities uygulanabilir.

3.6.1.4 Ardışık Güncelleme=numara

updatesequence=number

Updatesequence, uyumlu korumayı sağlamak için isteğe bağlı olarak kullanılan bir parametredir. Bu parametre tahminen bir tarih bilgisidir. Server updatesequence sayısını Kapasite XML'inde içerebilir. Eğer içeriyorsa o zaman bu sayı Kapasite de yapılan değişikliklere göre arttırılmalıdır. Müşteri bu parametreye GetCapabilities isteğinde sahip olabilir. Updatesequence değer tabanlı servis cevabındaki istek ve servis metaverisi Tablo 3.4'e göre olmalıdır.

İstemci İstekleri UpdateSequence Değerleri	**Sunucu Metaveri UpdateSequence Değerleri**	**Sunucu Cevapları**
hiçbiri	herhangi bir değer	En yeni Capabilities XML
herhangi bir değer	hiçbiri	En yeni Capabilities XML
eşit	eşit	Exception:code=CurrentUpdateSequence
çok düşük	çok yüksek	En yeni Capabilities XML
çok yüksek	çok düşük	Exception:code=InvalidUpdateSequence

Tablo 3.4 Ardışık güncelleme parametreleri

3.6.1.5 GetCapabilities (Kabiliyetleri Alma) Cevabı

WMS standardın bu versiyonu ile uyumludur. XML cevabı, XML doküman tip tanımına göre geçerli olmak zorundadır. DTD, gerekli ve isteğe bağlı cevapların

içeriğini ve bu içeriğin nasıl biçim edildiğini belirtir. Bir servis yayımlanmış veya deneysel versiyonlarda uyabilir.

İsimler ve Başlıklar

Birçok eleman <Name> ve <Title> etiketlerine sahiptir. Name, makinadan makinaya iletişim yapılırken kullanılan bir kelimedir. Title ise insanların çıkarlarına göredir. Örnek olarak, bir veri kümesinin adı Maksimum Atmosfer Isısı olabilir ve ATAMAX olarak adlandırılır.

Tabakalar ve Stiller

Bu iki parça WMS kapasitelerinin en kritik iki parçasıdır. Her elverişli harita <Layer> elemanıyla ilan edilir. Tek bir ana tabaka, birçok sayıda ilave edilen tabakaları çevreler ve bu da istenilen hiyerarşik yapıyı sağlar. <Title> her tabaka için gereklidir. Bu insanlar tarafından okunabilen ve sunum için olan bir dizgi'dir. Eğer sadece bir tabaka <Name> etiketine sahipse, o zaman bu bir getMap isteğinin parametresindeki LAYERS ismini kullanarak istenmiş bir harita tabakasıdır. Eğer bir tabakanın Title etiketi var ve Name etiketi yoksa o zaman bu tabaka diğer tüm tabakalar için bir kategori karşılığıdır. İsim elemanı içeren ve tabaka belirten harita servisi getMap isteğindeki tabaka değerini name olarak kabul etmeli ve uyan haritayı döndürmelidir. Bir müşteri Title etiketi olan fakat Name etiketi olmayan bir tabaka talebinde bulunmamalıdır. Eğer geçersiz bir tabaka istenirse, o zaman sunucu code="LayerNotDefined" işlemini yapmalıdır. Tabakalar Kapasite XML'inde hiyerarşik bir şekilde iç içedir. Ana tabaka tanımlanan bazı özellikler, içindeki küçük tabakalardan türetilmiştir. Bu türetilmiş özellikler küçük tabakalarca yeniden tanımlanır ya da eklenir. Bir harita servisi teklif edilen her harita tabakası için en az bir tane <Layer> elemanı içermelidir. Eğer istenirse tabakalar farklı kategorilerde alakalı oldukları zaman tekrar edilebilirler. Kendisini içeren bir tabaka, iç içe olan tabakalardaki Name içerenleri bir kez isteyebilir. Örnek olarak, "Road" isimli ana tabaka "State Highways" (ana caddeler) ve "Interstates" (ara caddeler) adında alt tabakalara sahip olabilir ve kullanıcıların birini veya ikisini birden istemelerine izin

verir. <Abstract> ve <keywordList> parametreleri isteğe bağlıdır fakat tavsiye edilir. Her tabaka en az bir tane <SRS> elemanına sahip olmalıdır veya başka bir ana tabakadan türediği açıkça belirtilmelidir. Kök tabaka elemanı, bütün yardımcı tabakalarda ortak olan SRS'lerin listesini içermelidir. Bu, müşterilerin servislerin SRS isteklerini memnun edemeyeceğini kolayca anlamalarını sağlar. Eğer portal bir SRS yoksa o zaman boş bir SRS listesi kullanılmalıdır. Ayrıca tabakalar global SRS listelerine eklenebilirler, veya listeler ana tabakadan türeyebilirler. Herhangi bir tekrarlama müşteriler tarafından görmezden gelinir. Her tabaka bir ana tabakadan türemiş ya da açıkça belirtilmiş bir <LatLonBoundingBox>'a sahip olmalıdır. <LatLonBoundingBox> özelliği dikdörtgeni çevreleyen enlem ve boylamların köşelerini ondalık olarak gösterir. (SRS'de EPSG:4326 [WGS1984 lat/lon] olarak görülür). LatLonBoundingBox, SRS'nin harita servisini destekleyip desteklemediğine bakılmaksızın sağlanmalıdır, fakat o zaman bu yaklaşık bir değer alır. Bunun amacı arama motorunda koordinat transferi yapmadan coğrafi aramaları kolaylaştırmaktır. Tabakalar sıfır veya daha fazla özel SRS <BoundingBox> elemanına sahiptir. BoundingBox değeri, belirlenen işlemsel referans sistemlerindeki BoundingBox ünitelerinin köşelerini gösterir. İsteğe bağlı resx ve resy değerleri, o ünitelerdeki verilerin işlemsel çözünürlüğünü belirtir. EPSG:4326 için bir BoundingBox, çözünürlük bilgisi yaratmak için sıralı bir şekilde olabilir. Tabakalar, en uygun tabakayı göstermek için minimum ve maksimum elemanlar öneren <ScaleHint> elemanına sahip olabilirler. Çünkü WMS output isteğe göre olan ölçü ve çözünürlük output aygıtları için yazılmıştır. Harita ölçüsü ile gerçek dünya arasındaki ölçü için tanımlanan ölçek tam olarak uygun değildir. İleride tanımlanan ScaleHint tanımı önerilir. BoundingBox, genişlik ve yüksekliği verilmiş kuramsal bir harita düşünülürse haritanın genişlik ve uzunluk değer kümesi olarak tanımlanabilir. Bu haritanın merkez pikseli bazı ölçülere sahiptir ve bu ölçüler güneybatıdan kuzeydoğuya yer mesafesi olarak metre cinsinden tanımlanır. Bu diyagonalde ScaleHint'in iki değeri minimum ve maksimum önerilir. Bu tanımlar yersel olarak kesin değildir fakat aynı zamanda sözleşme hükümleri ileride daha geniş olarak geliştirilecektir. Sıfır ya da daha fazla stil tabaka veya tabakalar koleksiyonu için <style> kullanılarak tanımlanmıştır. Ayrıca bunlar <Name>, <Title> ve diğer elemanları da içerir. Stillerin Name haritada STYLES istek parametresi olarak kullanılmıştır. Eğer sadece tek bir style

kullanılabiliyorsa, o zaman bu style "default" olarak kabul edilir. Kontrol edilmiş bir kelime bilgisi tanımlanmamıştır. O yüzden şu anki tabaka, stil, ad, başlık ve keywordler isteğe göredir.

Tabaka Özellikleri

Bir <Layer> sıfır ya da daha fazla XML özelliklerine sahip olabilir: sorgulanabilme, basamaklı, ışık geçirmez, alt kümesiz, kesin kenar genişliği, kesin kenar yüksekliği. Bütün bu özellikler isteğe bağlıdır ve sıfıra eşitlenmiştir. Özelliklerin her biri türetilebilir yada yardımcı tabakalarda değiştirilebilir. Her özelliğin anlamı Tablo 3.5'te gösterilmiştir.

Özellikler	**Tastiklenmiş Değerler**	**Anlamlar (0 genel değerdir)**
sorgulanabilir	0,1	**0:** sorgulanamaz **1:** sorgulanabilir
kademeli olarak büyütülebilir	0, tamsayı	**0:** Büyütülebilir harita sunucusu tarafından tabaka transfer edilemez **n:** tabaka n-defa transfer edilebilir.
anlaşılmasız	0,1	**0:** vektör özellikli harita verileri büyük olasılıkla tamamen boş alanlarda ifade edilemez **1:** harita verileri çoğunlukla veya tamamen anlaşılmaz (mantıksız) olabilirler
altkümesiz	0,1	**0:** WMS komple koordinat değerlerini altküme olarak barındırabilir ve görselleşebilir. **1:** WMS sadece girilen koordinat değerlerini görselleşebilir.
genel genişlik değeri	0, tamsayı	**0:** WMS sonucunda rast gele verilen genişlik değerlerine göre görüntü değişikliği olabilir. **0 olmayan:** genişlik sabit bir değer alır bu değer WMS tarafından değiştirilmez.
genel yükseklik değeri	0, tamsayı	**0:** WMS sonucunda rast gele verilen genişlik değerlerine göre görüntü değişikliği olabilir. **0 olmayan:** genişlik sabit bir değer alır, bu değer WMS tarafından değiştirilmez.

Tablo 3.5 Tabaka özellikleri

Sorgulanabilir Tabakalar

Bir tabakanın sorgulanabilir olabilmesi için GetFeatureInfo operasyonunun servis tarafından o tabakada desteklenmesi gerekmektedir. Her servis GetFeatureInfo'yu her tabakada desteklemez. Eğer GetFeatureInfo isteği sorgulanabilir değilse servis (code = "LayerNotQueryable") işlemini yapmak zorundadır.

Tabaka Özelliklerinin Türemesi

Tablo 3.6 ana tabakalardan diğer küçük tabakaların nasıl türediğini özetliyor. Bunlar tamimiyle türetilmiş olmayabilir. Küçük tabakalar tarafından yeniden tanımlanıp değiştirilebilirler yada eklenirler. Daha sonraki durumlarda bilgi tekrarı göz ardı edilebilir.

Elemanlar	**Sayılar**	**Kaynaktan bilgi alımı**	**Notlar**
Tabaka	0+	hayır	Opsiyonel
İsim	1	hayır	Tüm haritalı tabaklar zorunludur.
Başlık	1	hayır	Her tabaka için zorunludur.
Özet	0/1	hayır	Tavsiye edilir.
Anahtar kelime listesi	0/1	hayır	Tavsiye edilir
Stil	0+	ekleme	Opsiyonel
SRS	1	ekleme	Liste çok çeşitli değerler içerebilir.
LatLonBoundingBox (koordinatlar)	1	yerini değiştirme	Bir defa için zorunludur, genel değer alır. (default)
BoundingBox	0+	yerini değiştirme	Mevcut SRS için zorunludur, bunun dışında EPSG:4326 şeklinde kaynakta tanımlanır.
Boyut	0+	ekleme	Opsiyonel
Büyültme	0+	yerini değiştirme	Opsiyonel, birden daha fazla boyut değeri tanımlanamaz.

Özellik	0/1	yerini değiştirme	Opsiyonel
Yetki Url	0+	ekleme	Opsiyonel, bir yetkiUrl'ine karşılık bir tanımlayıcı atanır ve atanan kullanır.
Tanımlayıcı	0+	hayır	Opsiyonel
MetaveriUrl	0+	hayır	Opsiyonel
Veri Url	0/1	hayır	Opsiyonel
ÖzellikListeUrl	0/1	hayır	Opsiyonel
Dereceüstgörüntüismi	0/1	yerini değiştirme	Opsiyonel
Tablo 3.5'te özellik listesi	1	yerini değiştirme	Opsiyonel, genel değeri, 0 veya mevcut kaynak değeridir.

Tablo 3.6 Tabaka özelliklerinin türetilmesi

3.6.1.6 Output (çıktı) Biçimleri

Biçim belirteçleri XML kapasitelerinde birçok yerde ortaya çıkabilir. Örneğin işlemler için geçerli output biçimleri, desteklenmiş istisna biçimleri gibi. Biçimleri MIME tipinde imaj/gif ve imaj/png olarak belirtilir. Birçok özel OGC tipi çeşitli tipte XML dokümanlarını ayırt etmek için tanımlanmıştır. Bunlar yukarıda Tablo 3.2'de gösterilmiştir.

3.6.2 Harita Alma (zorunlu)

GetMap - Zorunlu

GetMap operasyonu grafik elemanlar kümesi veya resimli görüntü için tanımlanmış, harita üretmek için yapılan bir işlemdir. Bir harita isteği alırken, Harita Servisi isteği tamamlanmalıdır ya da bir istisna yaratmalıdır.

3.6.2.1 GetMap (Harita Alma) İsteği

Tablo 3.7 harita isteğini tanımlamaktadır. Bu istek WMS'de HTTP/AL işlemi kullanılarak ve url'de şifrelenerek çağırılır. (HTTP YAYINLA yöntemi SLD-uyumlu WMS ile isteğe bağlı olarak yapılabilir).

İstek Parametreleri	**Zorunlu/Opsiyonlu**	**Tanımlar**
VERSION=version	Zorunlu	İstek versiyonu
REQUEST=GetMap	Zorunlu	İstek ismi
LAYERS=layer_list	Zorunlu	Virgülle ayrılmış bir veya daha fazla harita tabaka listesi, SLD parametresi mevcut ise opsiyoneldir.
STYLES=style_list	Zorunlu	Virgülle ayrılmış her bir istek tabakaları listesineden birinin çizilmesi, SLD parametresi mevcut ise opsiyoneldir
SRS=namespace:identifier	Zorunlu	Konumsal kaynak sistemi
BBOX=minX,minY,maxX,maxY	Zorunlu	SRS içindeki köşe noktaları(en küçük sol değer, en büyük sağ değer)
WIDTH=output_width	Zorunlu	Harita görüntüsündeki genişlik değeri
HEIGHT=output_height	Zorunlu	Harita görüntüsündeki yükseklik değeri
FORMAT=output_format	Zorunlu	Haritanın çıktı biçimi.
TRANSPARENT=true/false	Opsiyonlu	Haritanın temel renk çevirimi
BGCOLOR=color_value	Opsiyonlu	Hexadecimal Kırmızı-Yeşil-Mavi renklerin temel renk için kullanılması
EXCEPTIONS=exception_format	Opsiyonlu	Hata raporlarının WMS'den dönmesi
TIME=time	Opsiyonlu	İşlenen tabakanın kullanım zamanı
ELEVATION=elevation	Opsiyonlu	İşlenen tabakanın dikey resmi
Diğer örnek boyutlar	Opsiyonlu	Alınan diğer boyutların

		değerleri
Vendor_specific_parameters	Opsiyonlu	Opsiyonlu deneysel parametreler
WMS servislerinde SLD biçimi destekli parametreler		
SLD=styled_layer_descriptor_URL	Opsiyonlu	SLD görüntü biçimi için url
WFS=web_feature_service_URL	Opsiyonlu	WFS için url

Tablo 3.7 GetMap istek parametreleri

3.6.2.2 İstek Parametreleri

Versiyon = versiyon

Version = version

Bu parametre temel servis elemanları bölümünde tanımlanmıştır. WMS'nin 1.0.0 versiyonunda bu parametrenin adı "WMTVER" idi. Bu isim şimdi kullanılmıyor fakat önceden hiçbir servis istisnası yayınlanmadan kabul ediliyordu.

İstek = Harita Alma

Request = GetMap

Bu parametrenin tanımı temel servis elemanları bölümünde yapılmıştır. GetMap için, "GetMap" değeri kullanılmalıdır. WMS'nin 1.0.0 versiyonunda bu parametrenin değeri "map" idi. Bu değer şimdi kullanılmıyor fakat önceden hiçbir servis istisnası yayınlanmadan kabul ediliyordu.

Tabakalar = tabaka_listesi

Layers = layer_list

Tabaka parametrelerinin değerleri virgülle ayrılmış bir yada daha fazla geçerli tabaka isimleridir. İzin verilen tabaka isimleri XML kapasitelerinde herhangi bir <Layer>

<Name> veri içeriğidir. Bir WMS istenmiş tabakaları en aşağıdan en üst sola kadar çizerek istenilen hale getirmelidir.

SRS = isimuzayı:tanımlayıcı

SRS = namespace:identifier

SRS (spatial reference system [işlemsel referans sistemi]) parametresi temel servis elemanları bölümünde tanımlanmıştır. Bu istek parametresinin değeri istenilen tabakadan türetilen yada tanımlanan <SRS> elemanının karakter veri bölümünde tanımlanmış olmalıdır. Aynı SRS tek bir istekte tüm tabakalara uygulanır. Eğer WMS servisi bir tabaka için SRS = NONE şeklinde bildirirse o zaman o tabaka iyi tanımlanmış işlemsel referans sistemi içermez ve diğer tabakalarla birleşik halde gösterilmemelidir. Müşteri GetMap isteğinde SRS = NONE olarak belirtmelidir.

Koordinat Değerleri

bbox = minx, miny, maxx, maxy

bbox parametresi müşterinin özel BoundingBox istemesine izin verir. BoundingBox temel servis elemanları bölümünde tanımlanmıştır. Eğer bir tabaka altkümesizse o zaman müşteri GetMap'te tanımlanan değerleri tam olarak belirtmelidir.

Genişlik, Yükseklik

Genişlik ve uzunluk parametreleri pozitif tam sayılar olarak tanımlanır. Birlikte alındıkları zaman bir dikdörtgenin (görüntü öğelerini) piksellerini harita olarak sonuçlandırır. Oluşan resimde döndürülen biçime bakılmaksızın aynı genişlik ve yükseklik (görüntü öğelerine) piksellerine sahip olmalıdır. BBOX ve WIDTH/HEIGHT oranının farklı olduğu durumda WMS dönen haritanın boyutlarını genişletmelidir. Böylece BBOX'ın oranına uydurur. Böylelikle pikselleri kare ve diğer şekiller olan output haritaya çevirmek için bu tanımları kullanabiliriz. Eğer bir istek,

gösterme elemanı biçiminde ise, genişilik ve yükseklik değerleri olmalıdır ve servis tarafından yararlı bilgi olarak kullanılabilmelidir. Fakat bu şartname şu anda deneysel olarak kullanılmaktadır. Eğer WMS servisi, Subsettable (Altküme tabloları) ve büyüklüğü resizeable (değiştirilebilir) tabakalar başlığının altında tanımlanan tabakaların genişlik ve yüksekliklerinin sabitlendiğini belirtirse o zaman müşteri getMap isteğindeki genişlik ve yükseklik değerlerini tam olarak tanımlamalıdır.

Hata Raporları

Exceptions = exception_format

WMS bir yada daha fazla hata raporlarını XML kapasitelerindeki <Exceptions> elemanlarının içinde <Formats> elemanlarını ayrı olarak listeler. Geride kalanlar ise MIME tipinde, EXEPTIONS parametresi değeri olarak kullanılır.

Zaman Uzunluğu

TIME = time
Eğer bir tabaka zaman uzunluğu tanımına sahipse, o zaman istekler TIME = value içerebilir.

Veri Uzunluğu

ELEVATION = elevation

Eğer bir veri objesi Elevation uzunluğu tanımına sahipse o zaman o objeleri düzeltmek için yapılan işlem istekleri ELEVATION = value içerebilir.

3.6.2.3 Müşteri Belirli İstekleri

Vendor_Specific Parameters

Burada sadece müşterilerin XML kapasitelerinde karşılaştıkları herhangi bir VSP'yi görmezden gelebileceklerini ve servislerin o VSP'lerdeki isteklere gerek duymayacağını tekrarlıyoruz.

3.6.2.4 Harita İsteme

GetMap==zorunlu

Geçerli bir GetMap isteğine verilen cevap istenilen stilde coğrafi referanslı tabaka istekli harita olmalıdır ve özel işlemli referans sistemine, bounding box'a, ölçüye, biçime ve saydamlığa sahip olmalıdır. Geçersiz bir GetMap isteği, istenen istisna biçiminde hata output vermelidir. (daha karışık durumlarda network protoköl hatası verebilir). HTTP çevresinde MIME tipinde dönen değerlerin içerik tipi dönen değerdeki ile uymalıdır.

3.6.3 Özellikleri İsteme

GetFeatureInfo==isteğe bağlı

GetFeatureInfo isteğe bağlı bir operasyondur. Sadece değerleri "queryable=1" olarak tanımlanmış ve türetilmiş tabakaları destekler. Bir müşteri diğer tabakalar için GetFeatureInfo isteği yayınlamamalıdır. Bir WMS düzenli biçimlenmiş servis istisnalarıyla cevap vermelidir. (application/ vnd.ogc.se_xml) GetFeatureInfo işlemi önceki harita istekleriyle dönen harita kesimlerinin özellikleri hakkında bilgiyle ve WMS ile birlikte müşterilere tasarım sağlamıştır. GetFeatureInfo için kullanım durumunun doğası kullanıcının harita isteğini görmesidir ve bununla birlikte haritadan daha fazla bilgi edinmek için bir nokta seçer. Temel işlem müşteri için hangi (görüntü öğelerinin) piksellerin istendiğini, hangi tabakaların araştırılacağını ve hangi biçimde bilginin döndürüleceğini sağlar. WMS protokölü ifade edilemediği için, GetFeatureInfo isteği WMS'nin hangi harita kullanıcılarının birçok GetMap istek parametrelerini görüntüleyeceğini gösterir. İşlemsel bilgi şartları için (BBOX,WIDTH, HEIGHT), GetMap isteğinde, X, Y'lerle birlikte WMS pozisyon hakkında ek bilgiler

döndürür. Yukarıda açıklanan davranış resim durumuna göre uyarlanmıştır. Grafik eleman durumunda ise GetFeatureInfo'nun anlamsallığı daha az tanımlanmıştır.Eğer listedeki herhangi bir tabaka WMS'nin XML kapasite listesinde yoksa, sonuçlar tanımsızdır ve WMS bir istisna cevabı yaratmalıdır.

3.6.3.1. GetFeatureInfo İstek Tanıtımı

GetFeatureInfo istek için parametreler Tablo3.8'de tanıtılmıştır.

İstek Parametreleri	**Zorunlu/ Opsiyonlu**	**Tanımlar**
VERSION=version	Zorunlu	İstek versiyonu
REQUEST= GetFeatureInfo	Zorunlu	İstek ismi
<map_request_copy>	Zorunlu	Geliştirilen harita bilgilerine bağlı, istek harita parametrelerinin parçalı kopyaları
QUERY_LAYERS= layer_list	Zorunlu	Bir veya daha fazla virgülle ayrılmış tabaka listesinin sorgu ile filtreleşmesi
INFO_FORMAT= output_format	Opsiyonlu	Özellik bilgilerinin sonuç olarak döndürülmesi
FEATURE_COUNT= number	Opsiyonlu	Dönen bilgilere ait özellikler(deafult=1)
X=pixel_column	Zorunlu	X koordinatının piksel değeri (sol üst köşeden 0 ile ölçülmeye başlanır)
Y=pixel_row	Zorunlu	Y koordinatının piksel değeri (sol üst köşeden 0 ile ölçülmeye başlanır)
EXCEPTIONS= exception_format	Opsiyonlu	WMS tarafından tanımlanan ve raporlanan hatalar (deafult=application/vnd.ogc.se_xml)
Vendor-specific parameters	Opsiyonlu	Opsiyonlu deneysel parametreler

Tablo 3.8 GetFeatureInfo istek parametreleri

3.6.3.2. İstek Parametreleri

Prefix (Ek ilave)

Adreslerin tanıtıldığı url linklerinde prefix (ek ilavelerle) temel servis elemanlarının rolleri tariflendirilmiştir. GetCapabilities, GetMap, GetFeatureInfo fonksiyonlarının ek ilaveleri farklı olabilmektedir.

Versiyon

Yayınlanan belirli versiyon numaraları üç tane tamsayı değer içerir ve ondalık nokta biçimi ile "x.y.z" şeklinde ifade edilir. "y" ve "z" numaraları 99 değerini geçemezler. Her OWS bağımsız şekilde numaralandırılmaktadır.

İstek

GetFeatureInfo için GetFeatureInfo'nun değeri kullanılmalıdır.

Sorgu Tabakaları

Sorgu tabakaları aracılığıyla virgül ile ayrılmış bir veya daha fazla harita listesinden, seçilen özelliğe göre tekrar istenen verinin çağrılması için oluşturulan sorguyu içerir. Bu parametre en az bir tabaka ismi içerir, fakat orjinal GetMap istek sonucuna göre birkaç çeşit tabaka da içerebilmektedir. Herhangi bir tabaka WMS'nin Capabilities XML yapısında yer almıyorsa veya sonucu dönmüyorsa, WMS tarafından isteğe karşı cevap olarak hata mesajı dönmektedir.

Bilgi Tipi

Info_Format

Info_Format (Bilgi tipi), özellikli veri döndürülürken hangi biçimin kullanılacağını gösterir. WMS örneklerinde GetFeatureInfo isteğinin desteklenmiş değerleri bir yada daha fazla <Format> elemanları ve kapasite XML elemanları <Request> <FeatureInfo> içinde MIME tipinde dizgi biçiminde <Format> içinde Info_Format parametresi değeri olarak kullanılmıştır. HTTP çevresinde, MIME tipi içerik-tipi bağımsız başlığı kullanılarak döndürülmüş objeye ayarlanmalıdır.

Örnek

Info_Format_application/vnd.ogc.gm parametre isteği özellikli bilginin GML'de biçimlenmesini ister.

X ve Y

X ve Y parametreleri, Harita isteği içine gömülmüş genişlik ve yükseklik parametrelerini sınırlarıyla beraber tek bir noktayla belirlemelidir. Merkez üst sol köşede (0,0) olarak ayarlanmıştır. X sağa doğru ve Y aşağı doğru artar.

Hatalar

WMS'nin GetFeatureInfo özelliği servise gönderdiği isteğe karşı cevap alamıyor ise veya herhangi bir istek/cevap biçiminde dolayı hatalar ile karşılaşılınıyorsa bu işlemin sonucu sistemden exception (hata mesajı) olarak ara yüze yansıtılmaktadır.

3.6.3.3. GetFeatureInfo Cevap

Bilgi tipi aracılığıyla WMS'ye gönderilen isteğin sonucu var ise veya sonucu olmayıp, hata mesajı ile cevaplandırılıyor ise bu şekilde her türlü sonuç GetFeatureInfo istek için cevap niteliğindedir. WMS üreticisinin yapısına bağlı olarak isteğe karşı cevap değerleri dönmektedir ve bu değerler aynı zamanda (X,Y) köşe koordinatlarına yakın değerler olmalıdır.

BÖLÜM 4

İŞLEMSEL VERİ STANDARTLARI VE CBS SERVİSİ BİRLİKTE İŞLERLİĞİ

Coğrafi bilgi sistemi (CBS)) teknolojisi, geleneksel CBS komitesinin ötesinde ve birçok organizasyondaki bilgi altyapısının integral bir parçası olarak gelişmektedir. CBS'nin tek bir entagrasyon kapasiteleri, bu durumun tam resmini yaratmak için farklı veri kümelerine izin vermektedir. CBS teknolojisi, ilişkiler, bağlantılar ve mutlak bir veri kümesinde olmayan kalıplar, daha iyi kararlar almak için ilgili faktörlere dayanan organizasyonlar sağlanması olarak örneklendirilebilir. Organizasyonlar, koordinasyon ve bir organizasyondaki bölümlerle ilgili anahtar fikirlerle bağlantı kurmak veya merkezi işlemsel veri altyapısı olarak CBS'yi farklı organizasyonlar arasında kullanarak, paylaşabilirler. Aynı zamanda CBS teknolojisi internet ve web servisleri yoluyla önemli bilgi paylaşmak için kullanılır. Tam olarak, coğrafi bilginin kapasite ve yararlarını, işlemsel veri ihtiyaçlarının paylaşılmasını, sistemlerin birlikte çalışabilmesini gerçekleştirir. CBS teknolojisi, paylaşılmış işlemsel veri altyapısı ve dağıtılmış mimari için sistemi oluşturmaktadır.

4.1 "Açık" CBS ne demek?

CBS birlikte işlerliğin altı evrede geliştirilmesi ve bunların uygulanması, kavram standart ve teknoloji açısından önemlidir.

1. Veri dönüştürücüleri (DLG, MOSS, GIRAS)
2. Standart değiştirme biçimleri (SDTS, DXF ™, GML)
3. Açık dosya biçimleri (VPF, shapefiles)
4. Direkt okuma uygulama programının arayüzleri (APIs) (ArcSDE API, CAD Reader, ArcSDE CAD Client)
5. Veritabanı işletme sistemindeki ortak özellikler (DBMS) (OGC Simple Feature Specification for SQL ™)
6. CBS web servislerinin standartlaşmış entegrasyonu (WMS, WFS, ArcIMS)

Tüm bu altı yaklaşım ve ilgili teknolojiler önemlidir ve bugün CBS birlikte işlerliği önemli bir rol oynamaya devam etmektedir.

Önceki yıllarda, hesaplama hızı ve maaliyet kısıtlamaları yeteneklerimizi sınırlandırmıştır ve direkt dosya değiştirme gibi pratik çözümlerde odaklanmamıza sebeb olmuştur. Farklı CBS sistemleriyle organizasyonlar arasında veri paylaşımı, veri dönüştürücüleri, transfer standartları ve sonraki açık dosya biçimleri sınırlandırılmıştır. İşlemsel veri paylaşımıyla diğer iş uygulamaları nadiren başarılmıştır. Bugün, birçok CBS ürünleri, az süre gecikmesiyle direkt olarak okunur ve bazen dinamik olarak veri dönüşür. Buradaki nokta, CBS komunitesi uzun yıllardır açık birlikte işlerliği takip etmekte ve bu amacı başarmak için çözümler üretmeye çalışmakta, ve çözümleri yeni teknolojilerin gelişmesiyle değişmektedir. Düşünülmesi gereken diğer faktör hala bir organizasyonda CBS'nin oynadığı rolün gelişmesidir. CBS'nin önceki günlerinde, nadir istisnalarla, bireysel olarak ayrı projelere odaklanılmıştır. Bugün odaklanan ise, işlemsel veri entagrasyonunun ve kritik görev işleri işlemlerinin analizi ve girişim iş akışı, CBS teknolojisinde yatırım dönüşümünün ve veri tabanlarının birlikte işlerliğin gelişmesiyle yükselmesi, karar vermek ve servis teslimidir.

Sonuç olarak, ilk olarak neden coğrafi bilgi sistemi teknolojisini tamamladığımızı hatırlayalım. Eğer, coğrafi verinin işletimi için uzmanlaşmış sorumluluğumuz varsa, CBS'nin kendi içinde son olmadığını hatırlamamız gerekir. Bir CBS mutlaka birçok kullanıcı tarafından paylaşabilecek yararlı bilgiler üretmelidir, aynı zamanda güvenilir veri bütünlüğü için istikrarlı altyapı sağlamalıdır. Teknolojiye bağlanılmaması ve temel prensiplerin unutulmaması çok önemlidir. Birlikte işlerlik veri ile organizasyonlar, çapraz uygulamalar ve endüstriler arasında bütünleşmeye imkan verir.

4.2 CBS Teknoloji Destekleri

4.2.1 Platform-Bağımsız Çözümler

CBS yazılımı büyük çok kullanıcılı çevrelerde artarak kullanılmakta ve işlemsel veriye (RDBMS)'den çeşitli platformlar kullanılarak erişebilmektedir. Erişebilmek için, aynı zamanda farklı servis bileşenlerinden olan CBS platform-bağımsız çözümlerini

desteklemektedir; donanım; network, veritabanları, geliştirme araçları. Masaüstü; web ve mobil müşteriler.

Örneğin ESRI'nin tanımlanması, platformdan bağımsız ürünlerin stratejisi kendi ürünlerinde yansıtılmıştır. Yüksek kapasitede ArcGIS masaüstü ürünleri müşterileri için, güçlü destek ile (ArcView, ArcEditor, ArcInfo) modern windows (2000, NT, XP) işletim sistemleri üzerinde çalışabilmektedir.

Yayılmış hesaplama aletleri için destek web tarayıcaları (HTML aracılığıyla) içeren birçok platformda, Java müşterileri (MapObjects-Java, ArcExplorer ™, ArcSDE-Java API), Tablet PC, Windows CE- ve Pocket PC- based devices (ArcPod), çok ince müşteriler (web servisleri ArcWebSM USA) ve dayalı standartlar (wireless) kablosuz aletleri, mesela kablosuz uygulama protokolleri (WAP), telefonlar, akıllı telefon, ilerlemiş uygulama servisleri sunucuları ve Linux'teki internete dayalı CBS web servisleri, UNIX (HP, IBM AIX ve Solaris ™) ve windows.

Ticari veri tabanı satıcıları için destek (IBM DB2, IBM/Informıx Dynamic Server ™, Microsoft SQL Server ve Oracle)

TCP/IP'ye dayalı LAN, WAN ve kablosuz ağ yapıları için destek, standart geliştirici çevreleri için destek (VB, C++, .NET, Java-J2ME, J2SE, J2EE, ASP/JSP,... vb)

Ayrıca örnek olarak, bizim uygulama durumumuzda açık kaynak ürünü olarak kullandığımız UMN Map Server yazılım araçları ve özelliklerini destekler.
Popüler ve gelişme çevreleri için destek,
PHP, Python, Perl, Ruby, Java ve C#,
Çapraz platform desteği,
Linux, windows, Mac OS X, Solaris,
Çok sayıdaki vektör veri biçimleri,
TIFF/GeoTIFF, EPPL7 ve GDAL aracılığıyla diğerleri,
Raster veri biçimleri (DGN, DWG, gibi..) uydu foto görüntüleri,

ESRI şekil dosyaları (shapefiles), PostGIS, ESRI ArcSDE, Oracle Spatial, MySQL ve OGR aracılığıyla diğerleri,
Açık kaynak kodlu CBS konsorsiyumu (OGC) web şartnameleri,
WMS (imaj-harita servisi) (istemci/sunucu), rapor ve kayıtları göstermeyen WFS (özellik servisi) (istemci/sunucu), WCS (kayıt servisi), Filter Encoding, SLD, GML, SOS
Harita projeksiyonu desteklemesi,
Anında (On-the-fly) işlem görebilen harita projeksiyonu binlerce projeksiyonla proje kütüphanesinde bulunur.

4.3 Coğrafi ilişkisel veritabanı (The Georelational Database)

CBS modelleri, coğrafi ilişkili yapılarda bunlara bağlı değerlerde verilerin ilişkili veri tabanlarında bağlanmış dosya tabanlı işlemsel özelliklerin olduğu yerde gelişmiştir. Fakat, coğrafi ilişki biçim ölçeklenebilirliği kısıtlamıştır ve ikili veri yapısı (ilişkili veritabanında tutulan değerlerle birlikte dosya tabanlı biçimde işlemsel özellikler) gösterdiki CBS bağlantılı veri tabanının yedekleme yenileme gibi özelliklerinden tamamiyle yararlanamamıştır. Buna ek olarak, büyük veri tabakalarının desteklenmesi performansı dengelemeye, ayrıca kompleks yapıların gizli tutulmasını gerektirir ve işlemsel bilgilerin diğer çekirdek işletim uygulamalarıyla paylaşımı hala mümkün değildir.

4.3.1 İşlemsel Elverişli Veritabanı

90'ların ortalarında, yeni teknoloji ilişkili veritabanlarında işlemsel veriyi tutmaya imkan sağlamak için ortaya çıkmıştır. Yeni çağın açılmasıyla ölçeklenebilirlik ve büyük bir destekle devamlı veri tabakalarıyla gizli tutulmadan ortaya çıkmıştır. Müşteri geliştirme çevresiyle çekirdek işletim uygulamalarının içine konabilen ve yeni işlemsel uygun veritabanlarının birleştiği zaman çekirdek işletim uygulamalarıyla işlemsel özelliklerin paylaşılması mümkün olmuştur. Buna ek olarak, bu işlemsel olanaklı veri tabanları, CBS girişimi yönünde ve organizasyonel "işlemsel veri adaları" eleme basamakları olmak için organizasyonlara izin vermektedir. Aslında çakışmadan,

açık CBS hareketi saklayabilme kapasitesi olan bütün bağıntılı modellerin gelişi ve bağıntılı veritabanının organizasyonları standartlaştırdığı zaman sonuçlanmıştır. Örnek olarak OGC, ABD (U.S.). Federal coğrafi veri komitesi, işlemsel veri standartları doğrultusunda veri paylaşımı fikrini ilerletmeye başlamıştır. Bu organizasyonların erken çalışması ilgili veri tabanlarında basit işlemsel özelliklerinin paylaşımına odaklanmıştır, o suretle birlikte çalışabilirlik ve ticari CBS satıcıları arasında imkan sağlanmıştır. OGC (açık kaynak kodlu CBS konsorsiyumu) uluslararası endüstri konsorsiyumu özel şirketler, hükümet acentaları ve üniversiteler, Basit Özellikler Şartnamesi olarak adlandırılan açık işlemsel standartı yayımlanmıştır.

4.3.2 Açık Kaynak Kodlu CBS Veritabanı POSTGIS (GeoDatabase)

PostGIS; PostgreSQL'in çıkarmış olduğu bir eklentidir. ESRI SDE ve Oracle Spatial ile aynı işi görebilecek yapıya sahip bir veri tabanı yönetim sistemidir. CBS çalışmalarımızda kullanabileceğimiz bir yapı barındırmaktadır ve sadece bu amaçla geliştirilmektedir. PostGIS'in gelişimi, OpenGIS tarafından sağlanmaktadır. PostGIS coğrafi verileri bünyesinde barındırabildiğinden ve açık kaynak kodlu bir veritabanı ürünü olduğundan bir çok açık kaynak kodlu CBS araç sağlayıcısı tarafından kullanılmaktadır. UMN MapServer CBS aracının PostGIS desteği olduğu için, biz de uygulamamızda PostGIS veritabanını kullandık.

4.4 XML tabanlı İnternetteki Konumsal Veri Birlikte Çalışabilirliği

Bilgi teknolojisi ve network gelişimiyle, işlemsel veri kazanımı, paylaşma ve analiz her bölümün kararında büyük ölçüde rol oynamaktadır. Tartıştığımız bu uygulama durumları, XML tabanlı işlemsel veri birlikte çalışabilirliğinin metodudur; XML-GML etki alanlarında özelleştirilmiş uygulama standartlarıdır. GML tabanlı bu işlemsel veri birlikte çalışabilirliği sistemi tasarlanmış ve pratikleştirilmiştir. İşlemsel veri saklaması ve standartların iletilmesini almak için GML seçilmiştir; fakat GML haritasını göstermek için genel bir alet yoktur, böylelikle bu HTML biçimi olarak değiştirilebilir ve XSL taşımasıyla harita araştırması veya işlemimizde uyguladığımız Xpath işlemi

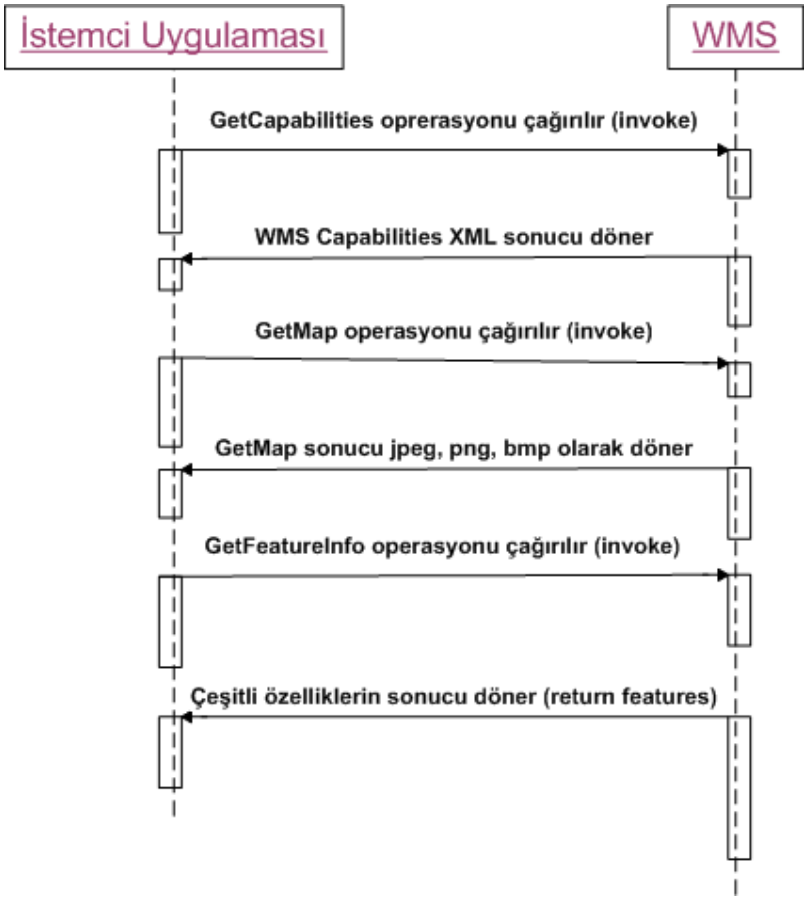
olarak bilgi biçimi olarak gösterilebilir. Uygulama sistemindeki dört tabakalı yapı, veritabanı sunucusu, web servisleri, web uygulama sunucusu ve müşterisi olarak tasarlanmıştır. Bu uygulama sistemi XML ve çevredeki web servisleri teknolojisini destekler. Sonuç memnun edicidir ve teknoloji, XML tabanlı işlemsel veri birlikte çalışabilirliğindeki gelişmiş olasılıkları içerecektir.

Birçok veri tabanı ticari CBS platformlarına dayalı (örneğin önceki bölümlerdeki ArcInfo, PostGIS yazılımı) bütün bölümler tarafından yapılmıştır. Coğrafi veri kümeleri hızla paylaşılmakta, değiştirilmekte ve başka amaçlar için diğer üreticiler dışında tasarlanmış olanlar tarafından kullanılmaktadır. İşlemsel veri paylaşımını ve birlikte çalışabilirliği CBS alanında nasıl yararlı kullanacağımız anahtar problemdir. OGC CBS standartlarında adreslenmiştir ve bu yüzden birçok kaynaklardan veri ve bilgi, gelişmek için çeşitli uygulamalardaki CBS kapasiteleriyle bütünleşebilir. Çalışmalarında veri işlemi için CBS bağlantılı şartname ve arayüzleri tanımlamaktadırlar. Web servisleri açık dağıtılmış sistemler yaratmak için bloklar yaparlar. Web servisleri, tasarlanmış, dağıtılmış uygulamalarda belki evrimsel bir basamaktır, HTTP ve XML kullanarak birleşebilir. XML-GML özelleşmiş etki alanlarında iki çeşit uygulama standartları karşılaştırır.

4.5 Konumsal Veri Birlikte Çalışabilirliği ve XML Tabanlı Görselleme

XML, veri anlatımı ve internetteki veri değişimi için yeni bir standarttır ve aynı zamanda XML'in karakteri web bütünleşmesini mümkün hala getirir. Bundan dolayı yapısı ve bu teknolojileri destekleyen erişilebilir web servisleri bir arada host işletilir. GML işlemsel veri saklaması ve standart iletim için bir XML şifrelemesidir çünkü bu açık ve vendor-neutral (doğal sağlayıcı) referans sistemlerinin tanımı ve coğrafi konumsal bilginin ölçüsü ve coğrafi uygulama şemalı ve objeleri için sistem sağlamaktadır. Coğrafi konumsal özellikleriyle geometri, coğrafi konumsal yayını ve uygun altkümeleri sistemi tanımlayıcı kapasitelerini destekleyen profillere izin vermektedir. Coğrafi uygulama şemaları ve veri kümeleri bağlantılı kreasyonuna ve bakımına olanak sağlamakta ve uygulama şemalarıyla veri kümelerinin saklanmasını ve iletimini desteklemektedir.

XML tabanlı işlemlerde iki metod vardır. Biri, XML tarafından tanımlanmış bütün veri kümlerini değiştirmek, sonra diğer sistemlerle ilişkili şemalarda kaldırılabilir. Diğeri ise gerçek zamanda veri kaldırılırken okunmasıdır. Bu unsur, işlemsel verinin okunması yazılması istenmesi için XML veya SOAP protokolünde gösterilmektedir. Veritabanı işletim sisteminden, işlemsel objelerin birlikte okunması, sistem herkese açık arayüzde XML'le tanımlanmış işlemsel veriyi veri uzantısına çevirir. Sonra diğer sistemler, işlemsel veri isteği gerçek zamanda alabilirler. Bu çevrimiçi veri paylaşımını ve içişlemleri gerçekleştirir. XML'deki arayüze dayalı veri uzantısı platformdaki iç işlemleri kolay anlamak ve gerçekleştirmek için ASCII koddur, farklı OGC ve ISO/TC211, web servise ve XML' e dayalı işlemsel veri birlikte çalışabilirliği için ayrıntıları formüle etmiştir. Örneğin; WMS, WFS, WCS, GML vb.. bunun içinde, WMS coğrafi veri temsil eden, haritaları tanımlayan, coğrafi uzay bilgisi kullanarak harita tasarlamaktadır. Şartname üç farklı işlem tanımlar: "getcapabilities" servise metaveri verir. "GetMap" işlemsel referans ve parametre tanımlı harita döndürür. "GetFeatureInfo" haritalar üzerindeki özellikler hakkında bilgi verir. WMS'nin akışı Şekil 4.1'de gösterilmiştir. İlk önce müşteri getcapabilities işleminden servis metaverisi alır, sonra GetMap arayüzünü çağırır, belirlenen parametrelere değerleri koyar ve servisten haritaları alabilir. Son olarak, müşteri getFeatureInfo işlemini çağırır ve özellik için tanımlanmış bilgiyi alır. Bu sistem başlıca WMS şartnamelerini gerçekleştirir.



Şekil 4.1 Web harita servisi (WMS) iş akışı

XML tabanlı işlemsel veriler işlenebilir ve OGC bileşenleriyle internette url linki olarak gösterilebilir. GML tabanlı veriler SVG veya XSL dönüştürme metoduyla html dosyasını jpeg, png veya başka bir görüntü biçiminde kullanır. Yeni gelişen bir teknoloji olarak XML birçok merkezi adreste kullanılır. Anlamsal tanımlarda XML'in üstünlüğü, veri değiş tokuşu ve bilgi araştırması, birçok farklı sistemden işlemsel veri işlenebilirliğini açığa çıkarmak için CBS alanlarına XML ile uygulanır. GML bir coğrafi bilgi içeren metin dilidir ve her CBS uygulaması görüşünde gerekli olarak kabul edilir. GML obje modelleri uygulamalarında belirtilen tüm sorular için tasarlanmıştır.

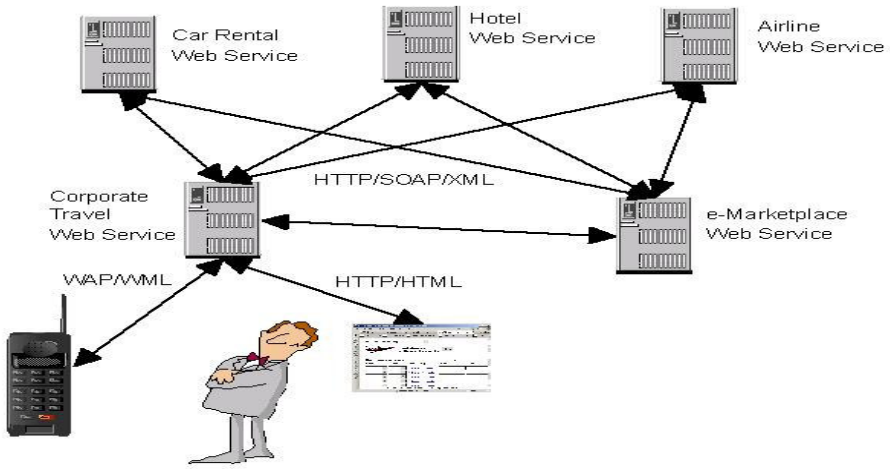
BÖLÜM 5

WEB SERVİSLERİ TANIMLARI VE UYGULAMA İÇERİĞİ

Geçen yıllardan bu yana bir çok alanda internet ve web teknolojileri yeni bir devrim başlatmıştır. Bir çok araştırmacıya göre XML tabanlı web servisleri bu devrimin ikinci adımını oluşturacaktır. Web servisleri web ortamında yayınlanabilen, aranıp bulunabilen ve çağrılarak erişilebilen modüler uygulama fonksiyonlarıdır. Bu fonksiyonlar değişik kurumsal iş süreçlerini gerçekleştireceklerdir. Web servisleri Haziran 2000'de ortaya çıkan bir çok yazılım firması tarafından yoğun bir destek bulan bir modeldir. Web servisleri açık internet standartlarına dayanır. Henüz gelişme ve olgunlaşma aşamasında olan bu modelle ilgili olarak bu aşamada ortaya çıkan ve kullanılan çekirdek standartlar SOAP, WSDL ve UDDI'dır. Web servisleri modelini destekleyen Microsoft, IBM, Sun, HP, Oracle ve daha bir çok firma bu konuda yoğun bir şekilde çalışmakta ve web servisleri yazılım ve uygulama geliştirme araçlarını geliştiricilere sunmaktadırlar. Bu konudaki firmaların yoğun desteğinden dolayı uygulama bütünleştirmesi konusunda ortaya çıkacak hakim ortamın web servisleri modeline dayanacağı yönündedir. Geçen yıllardaki gelişmelerden sonra bugünkü web ortamı olmadan ağ tabanlı bilgi sistemlerinin düşünülmesi çok zordur. Web'in bu kadar başarılı olmasının nedeni basitlik ve yaygınlığıdır. Web ortamındaki gelişmeleri üç safhada inceleyebiliriz:

1. *Belge Web'i (Document Web)* : Belge web'i ile web ilk aşamadaki kullanımı, yani HTTP protokolü ile HTML dilinde biçimlendirilmiş **statik belgelerinin** kullanıcılara sunumunu ifade ediyor.
2. *Uygulama Web'i (Application Web) :* Bu yapı işletmelerin müşterilerine web üzerinden bazı iş süreçlerini yaptırma gereksinimi sonucunda ortaya çıktı. Bu yapıda sunucu tarafında çalışan programlar vasıtasıyla hazırlanan **dinamik HTML belgeleri** ile kullanıcı ve iş uygulaması arasında etkileşim sağlandı.
3. *Servis Web'i (Services Web) :* İşletmelerin diğer işletmelerle olan iş süreçlerini bütünleştirme gereksinimi sonucunda ortaya çıkan ve gelişmekte olan yeni yapıdır. Bu yapının temel taşı web servisleridir. Web

servislerindeki temel amaç işletme bilgi sistemlerindeki **program modüllerinin etkileşimini** sağlamaktır. Web servisleri web ortamında **yayınlanabilen**, **aranıp bulunabilen** ve **çağrılarak erişilebilen** modüler uygulama fonksiyonlarıdır. Bu fonksiyonlar değişik kurumsal iş süreçlerini gerçekleştireceklerdir. Web servisi kavramı destekcilerinden Frank Boss'a göre "İleride internet'te web sayfasından daha çok web servisi bulunacaktır." Şekil 5.1 web servislerini kullanarak bir iş gezisi için gerekli rezervasyonları yapan bir uygulama örneği senaryosunu göstermektedir. Bu örnekte iş gezisi planlayan kişi hava yolu, otel ve araç kiralama rezervasyonları yapacağı var sayılmıştır. İş gezisi organizasyonu için geliştirilen uygulama havayolu, otel, araç kiralama ve e-marketplace firmalarının sağladığı web servislerini çağırarak gerekli işlemleri tek bir uygulama ile gerçekleştirecektir.



Şekil 5.1 – Web servisleri ile etkileşim sağlayan iş gezisi uygulaması

(Kaynak numarası: 62)

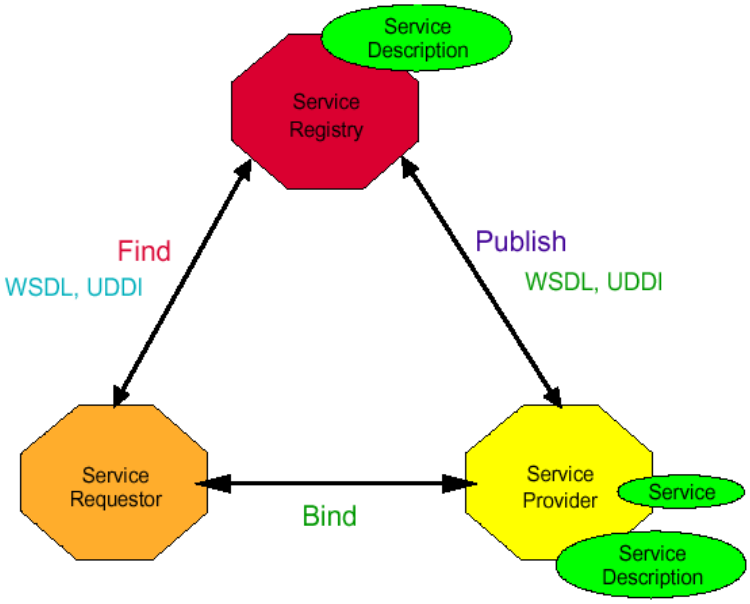
Web servisleri **Haziran 2000'de** ortaya çıkan bir çok yazılım firması tarafından yoğun bir destek bulan kavramdır. Web servisleri yazılım endüstrisinin "El Dorado" su olarak nitelendirilmektedir. "El Dorado" bir zamanlar Amerika da altın madeni bulunduğu var sayılan herkesin aradığı hayali bir şehirdir. Internet üzerinde uygulama bütünleştirmesi için yöntem arayan bir çok yazılım geliştirici firma ve kurum web servisleri modeli konusuna yönelmişlerdir. Bu modeli destekleyen

Microsoft, IBM, Sun, HP, Oracle ve daha bir çok firma bu konuda yoğun bir şekilde çalışmakta ve web servisleri yazılım ve uygulama geliştirme araçları sunmaktadır. Bu konudaki firmaların yoğun desteğinden dolayı uygulama bütünleştirmesi konusunda ortaya çıkacak hakim ortamın web servisleri modeline dayanacağı yönündedir.

Web Servisleri Modeli

Web servisleri modeli üç ana birimin etkileşimine dayanır. Şekil 5.2'de gösterilen bu birimler şunlardır:

- Servis Sağlayıcı (Servis Provider): Servis sağlayıcı istemcilerin sağlayıcıda bulunan servislere erişimini sağlar. Servis sağlayıcı kendi sitesinde bulunan web servisleri tanımını servis kayıt birimine kaydederek bu servisinin nasıl çağrılacağı belirtir.
- Servis İstemcisi (Service Requester) : Servis sağlayıcısında bulunan web servislerini çağırarak kullanan istemci uygulamalardır. Web servisinin nasıl çağrılacağı ve ilgili parametreleri servis kayıt biriminden arayarak bulur ve çağırır.
- Servis Kayıt Birimi (Service Registry) : Servis sağlayıcılarının yayınladıkları web servisi tanımlarını saklar ve aranıp bulunmasını sağlar. Servis sağlayıcıları servis kayıt birimini tarayarak istediği servislere hakkında bilgi alabilir. Servis kayır birimi her servisin nasıl çağırılacağı konusunda tanım bilgileri içerir.

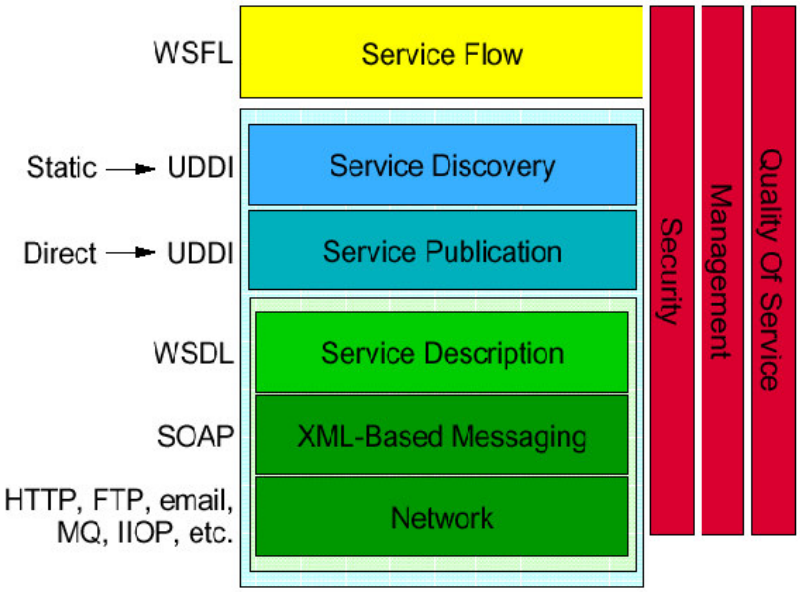


Şekil 5.2 – Web servis modeli (Kaynak numarası: 61)

Web Servisleri Standartları

Web servisleri açık internet standartlarına dayanır. Daha önce belirtildiği gibi web servisi modeli Haziran 2000'de ortaya çıkmıştır. Bu nedenle henüz tamamen olgunlaşmış teknoloji değildir. Şekil 5.3 web servisi mimarisindeki temel katmanları göstermektedir. Bu katmanlarda belirtilen güvenlik, iş akışı, servis kalitesi ve yönetim gibi konulardaki web servisi standartları henüz araştırma aşamasındadır. Bunların yanında bir takım temel çekirdek standartlar oluşmaya başlamıştır. Bunlar şunlardır:

1. SOAP (Simple Object Access Protocol) : Intenet üzerinde web servislerini çalıştırmak için kullanılan protokol
2. WSDL (Web Services Description Language) : Web servislerini tanımlama dili
3. UDDI (Universal Description, Discovery and Integration) : Web servislerinin indekslenip bulunduğu kayıt servisi



Şekil 5.3 – Web servisi mimarisi katmanları (Kaynak numarası: 61)

Bu bölümde web servisleri tanımlamaları, özellikleri hakkında bilgiler verilecektir. Aynı zamanda OGC uyumlu web servislerinin kullanıldığı mimari yapı ile bu mimarinin kullanıldığı uygulama yapısı hakkında tanımlamalar ve açıklamalar yer alacaktır.

5.1 Web Servisleri Tanımlama Dili (WSDL)

Bir uygulamanın bir web servisini kullanması için web servisinin nasıl çağırılacağının, arayüzünün, hangi protokollerin ve kodlama standartlarının belirtilmesi gerekir. WSDL web servisini tanımlayan bir XML belgesidir. Bir anlamda dağıtık programlamada kullanılan IDL'e (Interface Definition Language – Arayüz Tanımlama Dili) benzer. Web servisi tanımı işlemler, giren ve çıkan mesaj biçimleri, ağ ve port adresleri gibi bilgileri tanımlar. Bir web servisi tanım belgesi aşağıdaki temel elemanları içerir:

- *Types:* mesajlarda kullanılacak veri tiplerini belirtir.
- *Message:* İletişimde kullanılacak measjları tanımlar.

- *PortType :* Web servisinin içerdiği işlemleri ve ilgili mesajları tanımlar.
- *Binding :* İşlem ve mesajlarda kullanılacak veri biçimlerini tanımlar.
- *Port:* Binding ve web adresinden oluşan servis noktasını tanımlar. Web adresi servisin çalıştırılacağı URL'dir.
- *Service:* Kullanılan port'lar kümesidir.

Şekil 5.4 Microsoft'un web servisleri aracı olan Microsoft SOAP toolkit'te verilen bir basit aritmetik işlemler yapan bir web servisinin WSDL belgesini göstermektedir. Bu web servisi **EchoString**, **AddNumbers**, and **SubtractNumbers** işlemlerini gerçekleştirmektedir. Şekil'de görüldüğü gibi WSDL belgesi oldukça karmaşıktır. Bu nedenle gerellikle web servisi geliştirme araçları WDSL belgelerini otomatik olarak oluşturan programlar içermektedir.

```
<definitions name ='DocSample1'   targetNamespace = 'http://tempuri.org/wsdl/'
   xmlns:wsdlns='http://tempuri.org/wsdl/'
   xmlns:typens='http://tempuri.org/type'
   xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
   xmlns:stk='http://schemas.microsoft.com/soap-toolkit/wsdl-extension'
   xmlns='http://schemas.xmlsoap.org/wsdl/'>
  <types>
   <schema targetNamespace='http://tempuri.org/type'
     xmlns='http://www.w3.org/2001/XMLSchema'
     xmlns:SOAP-ENC='http://schemas.xmlsoap.org/soap/encoding/'
     xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'>
   </schema>
  </types>
  <message name='Sample1.EchoString'>
   <part name='testString' type='xsd:string'/>
  </message>
  <message name='Sample1.EchoStringResponse'>
```

```
  <part name='Result' type='xsd:string'/>
</message>
<message name='Sample1.AddNumbers'>
  <part name='NumberOne' type='xsd:double'/>
  <part name='NumberTwo' type='xsd:double'/>
</message>
<message name='Sample1.AddNumbersResponse'>
  <part name='Result' type='xsd:double'/>
</message>
<message name='Sample1.SubtractNumbers'>
  <part name='NumberOne' type='xsd:double'/>
  <part name='NumberTwo' type='xsd:double'/>
</message>
<message name='Sample1.SubtractNumbersResponse'>
  <part name='Result' type='xsd:double'/>
</message>
<portType name='Sample1SoapPort'>
  <operation name='EchoString' parameterOrder='testString'>
    <input message='wsdlns:Sample1.EchoString' />
    <output message='wsdlns:Sample1.EchoStringResponse' />
  </operation>
  <operation name='AddNumbers' parameterOrder='NumberOne NumberTwo'>
    <input message='wsdlns:Sample1.AddNumbers' />
    <output message='wsdlns:Sample1.AddNumbersResponse' />
  </operation>
  <operation name='SubtractNumbers' parameterOrder='NumberOne NumberTwo'>
    <input message='wsdlns:Sample1.SubtractNumbers' />
    <output message='wsdlns:Sample1.SubtractNumbersResponse' />
  </operation>
</portType>
<binding name='Sample1SoapBinding' type='wsdlns:Sample1SoapPort' >
  <stk:binding preferredEncoding='UTF-8'/>
```

```
<soap:binding style='rpc' transport='http://schemas.xmlsoap.org/soap/http' />
  <operation name='EchoString' >
    <soap:operation soapAction='http://tempuri.org/action/Sample1.EchoString' />
    <input>
      <soap:body use='encoded' namespace='http://tempuri.org/message/'
      encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
    </input>
    <output>
      <soap:body use='encoded' namespace='http://tempuri.org/message/'
      encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
    </output>
  </operation>
  <operation name='AddNumbers' >
    <soap:operation soapAction='http://tempuri.org/action/Sample1.AddNumbers'/>
    <input>
      <soap:body use='encoded' namespace='http://tempuri.org/message/'
      encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
    </input>
    <output>
      <soap:body use='encoded' namespace='http://tempuri.org/message/'
      encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
    </output>
  </operation>
  <operation name='SubtractNumbers' >
        <soap:operation
        soapAction='http://tempuri.org/action/Sample1.SubtractNumbers' />
    <input>
      <soap:body use='encoded' namespace='http://tempuri.org/message/'
      encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
    </input>
    <output>
      <soap:body use='encoded' namespace='http://tempuri.org/message/'
```

```
      encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
    </output>
   </operation>
  </binding>
  <service name='DocSample1' >
   <port name='Sample1SoapPort' binding='wsdlns:Sample1SoapBinding' >
    <soap:address location='http://localhost/DocSample1/DocSample1.wsdl' />
   </port>
  </service>
</definitions>
```

Şekil 5.4 – Bir WSDL belgesi

5.2 Basit Nesne Erişim Protokolü (SOAP)

SOAP uygulama bütünleştirmesi için geliştirilmiş XML ile biçimlenmiş bilgilerin iletişimini sağlayan bir protokoldür. Uygulama bütünleştirme için *middleware* olarak adlandırılan daha önce bir çok çözümler sunmuştur. RPC (Remote Procedure Call, DCOM, IIOP (Internet Inter-ORB Protocol) ve Java RMI bu çözümlerden bazılarıdır. Bu middleware çözümleri internet ortamında iletişim sağlama konusunda yetersiz kalmaktadır. Bu nedenle XML tabanlı bir protokol olan SOAP giderek yaygınlaşmaktadır. SOAP istemcilerin sunucularda olan nesne yöntemlerini çağırmasını ve sonuçların alınmasını sağlayan basit istek/yanıt protokolüdür. SOAP mevcut internet altyapısında olan router, firewall ve proxy sunucularda herhangi bir değişiklik yapmadan kolayca çalışmaktadır. Bir SOAP uygulaması geliştirmek için istemci ve sunucuya SOAP geliştirme araçları ile birlikte gelen kütüpanelerin yüklenmesi gerekir. Bu kütüpaneler bir XML ayrıştırıcı ve SOAP işlemicisi içerir. İstemci SOAP uygulaması bir SOAP istek mesajı oluşturarak bu isteği SOAP sunucusunda tanımlanmış servis uç noktalarından (end point) birisi tarafından çalıştırılması için gönderir. SOAP sunucu ilgili servisi çalıştırdıktan sonra SOAP yanıt mesajı hazırlar. Hazırlanan SOAP yanıt mesajı istemciye iletilir. SOAP mesajı HTTP POST metodu veri paketinin içinde gönderilir. Bir SOAP mesajı bir SOAP zarfından (SOAP envelope) oluşur. SOAP zarfı opsiyonel bir SOAP başlığı (SOAP

header) ve SOAP gövdesinden (SOAP body) oluşur. SOAP gövdesi çağırılacak metod ve metodun içerdiği parametreleri içerir. Şekil 5.5 ve 5.6 bir istek ve yanıt mesajının iceriğini göstermektedir.

```
POST /StockQuote HTTP/1.1
        Host: www.stockquoteserver.com
        Content-Type: text/xml; charset="utf-8"
        Content-Length: nnnn
        SOAPAction: "http://example.com/stockquote.xsd"

        <SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <SOAP-ENV:Body>
        <m:GetLastTradePrice xmlns:m="http://example.com/stockquote.xsd">
        <symbol>DIS</symbol>
        </m:GetLastTradePrice>
        </SOAP-ENV:Body>
        </SOAP-ENV:Envelope>
```

Şekil 5.5 – Bir SOAP istemci istek (request) mesajı

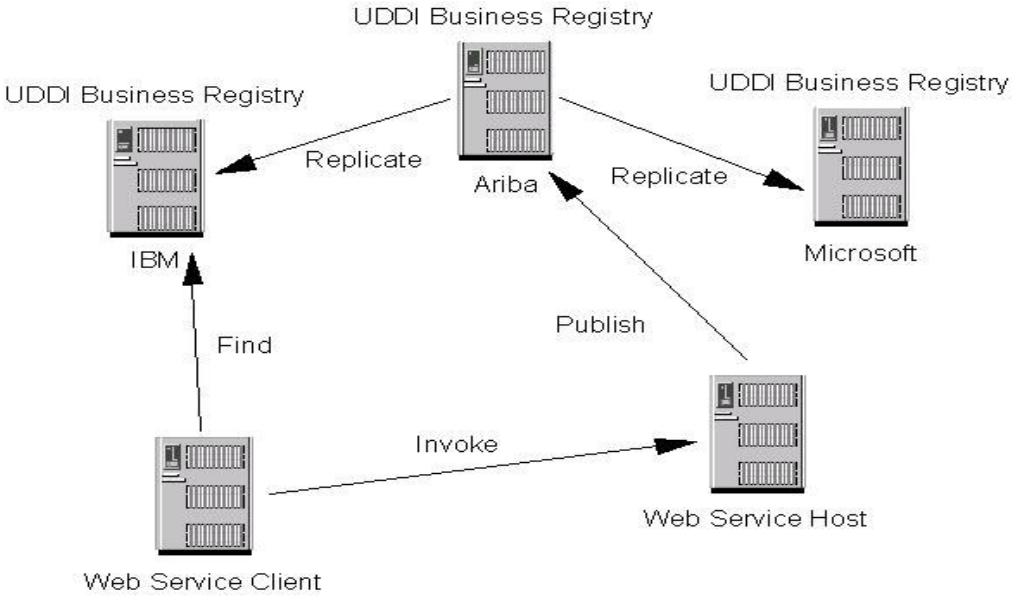
```
        HTTP/1.1 200 OK
        Content-Type: text/xml; charset="utf-8"
        Content-Length: nnnn

        <SOAP-ENV:Envelope
            xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
            SOAP-
        ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
            <SOAP-ENV:Body>
                <m:GetLastTradePriceResponse
        xmlns:m="http://example.com/stockquote.xsd">
```

```
                    <Price>34.5</Price>
                </m:GetLastTradePriceResponse>
            </SOAP-ENV:Body>
        </SOAP-ENV:Envelope>
```

Şekil 5.6 – Bir SOAP yanıt (response) mesajı

5.3 Evrensel Tanımlama, Keşif ve Entegrasyon (UDDI)

Bir web servisini kullanmak için kullanıcının web servisi sağlayan kurumları ve bu kurumların verdikleri web servislerinin neler olduğunu bilmesi gerekir. UDDI kısaltmasında geçen *Evrensel*, *Tanım*, *Buluş* ve *Bütünleştirme* kelimelerinin ifade ettiği gibi UDDI kurumların kendilerini, sağladıkları servisleri yayınlayarak tanımlamalarını, ve bu bilgilerin daha sonra diğer kurumlarca taranıp bulunmasını sağlayan bir standarttır. UDDI Kurum Kayıt Servisi kurum ve web servisleri bilgilerini saklayan sunulardır. Bu sunucular servis sağlayıcılarından gelen bilgilerini kendi veritabanlarına kayıt ederek diğer kurumların erişimine açar. Şu anda aktif olarak çalışan kurum kayıt sunucuları *uddi.microsoft.com* ve *uddi.ibm.com* 'dur. Şekil 5.7'de görüldüğü gibi bu sunucular kendilerine kayıt edilen bilgileri diğer sunucularada kopyalayarak kolayca hızlı bir şekilde erişilmesini sağlarlar. UDDI sunucuları kurum ve servis kayıt, güncelleme ve tarama işlemlerini web servisleri (SOAP mesajları) ile gerçekleştirir.



Şekil 5.7 – UDDI Kurum Kayıt Sunucuları (Kaynak numarası : 63)

Web Servisi ve SOAP Uygulama Geliştirme Araçları

Bir çok firma web servisi ve SOAP uygulaması geliştirmek için araçlar sunmaktadır. Bu araçların listesi www.soapware.org sitesinde listelenmiştir.

- Apache SOAP (Apache project) [Full, Java] 2.2, 2001/05/30
- Web Services Toolkit (IBM, alphaWorks) [Full, Java] 2.3, 2001/05/14
- GLUE[registration required] (Graham Glass)[Full, Java] 3.1, 2001/07/14
- DevelopMentor SOAP (DevelopMentor) [Full, Java] 0.3, 2000/01/24

HP Web Services Platform [registration required] (HP) [Full, Java] ?,

- 2001/08/15
- Soap Toolkit for JBuilder [login required] (Stephen Schaub) [Client, Java | JBuilder 4 | Win] ?, 2001/03/02
- SOAP for BEA WebLogic [no official support] (BEA Systems) [Server?, Java? | BEA WebLogic Server] ?, ?
- XMLBus [registration required] (IONA Technologies) [Full, Java | iPortal AppServer, BEA WebLogic] 1.2.2, 2001/08/22
- Microsoft SOAP toolkit (Microsoft) [Full, VB/C# | WinNT/2K] 2.0, 2001/04/25
- Microsoft SOAP toolkit (aka ROPE) [no support] (Microsoft) [Full, VB/C# | WinNT/2K] 1.0, 2000/12
- Visual Studio .NET (Microsoft) [Full, .NET | WinNT/2K/98/Me] B2, 2001/06/18
- .NET Framework SDK (Microsoft) [Full, .NET | WinNT/XP/2K/98/Me] B2, 2001/06/07
- SOAP::Lite (Paul Kulchenko) [Full, Perl] 0.51, 2001/07/18
- DevelopMentor SOAP (DevelopMentor) [Full, Perl] 0.28, 2000/09/05
- SQLData SOAP Server (SQLData) [Server, C++ | WinNT/2K] 3.01,

2001/01/25

- SOAP toolkit LITE (Lucin) [Full, ?] 2.3.0, 2000/09/18

- Delphi 6 (Borland) [Full, Delphi] ?, ?

Yeni nesil web uygulamaları web servisleri modeline dayanacaktır. Web servisleri ile, kurumlar işbirliği yaptığı satıcı, müşteri ve banka gibi diğer kurumlardaki iş süreçlerini birleştirerek daha iyi işlevsellik ve daha az maliyet gibi bir çok avantajlar sağlayacaklardır. Internet'te iki site www.xmethods.com ve www.salcentral.com değişik kurumlar ve kişilerce geliştirilen basit web servislerini yayınlamaktadır. Web servisleri modeli XML, SOAP, WSDL ve UDDI gibi açık internet standartlarına dayanır. Bu konudaki firmaların yoğun desteğinden dolayı uygulama bütünleştirmesi konusunda ortaya çıkacak hakim ortamın web servisleri modeline dayanacağı yönündedir. Web servisleri modelini ciddi olarak benimseyen Microsoft, IBM, Sun, HP, Oracle ve daha bir çok firma bu konuda yoğun bir şekilde çalışmakta ve web servisleri yazılım ve uygulama geliştirme araçlarını geliştiricilere sunmaktadırlar. Web servisleri modeli Haziran 2000'de ortaya çıkmış ve yoğun bir ilgi ile karşılaşmıştır. Mevcut web servisleri modeli uygulama bütünleştirme konusunda bir takım temel yapıları içermekle birlikte bir takım eksiklikleri vardır. Bunlar güvenlik, iş akışı, servis sürekliği, servis kalitesi ve yönetim gibi konulardadır. Web servislerinin başarılı ve yaygın olarak kullanılması için mevcut modelin geliştirilerek bu konuların çözüme kavuşturulması gerekmektedir.

5.4 Genişletilebilir İşaretleme Dili - XML

Bilgi teknolojilerindeki hızlı değişim, yazılım ve donanım modellerini etkilediği gibi iş modellerini de önemli ölçüde etkiledi ve radikal değişikliklere yol açtı. Bilgiyi birçok farklı medyalarda ve biçimde yayınlama yada paylaşma ihtiyacı giderek arttı. İnternet'in ilk yıllarında bir kurtarıcı olarak görülen, platformdan bağımsız, kolay

kullanılabilir ve öğrenilebilir yapısı ile HTML, bu yeni ihtiyaçlara statik metin yapısı yüzünden cevap veremez hale geldi. HTML'in metin üzerindeki basit tanımlamaları web sayfalarını görsel açıdan tatmin etse de, veriler için aynı şeyi söylemek zordu. HTML tabanlı bir dokümanın standart PC'lerde yorumlanması, değişiklik yapılmadan PDA'ler için de kullanılması mümkünken, yeni nesil telefonlar yada akıllı ev araçları için bir veri kaynağı olmaktan uzaktı. XML'li popüler yapan unsurların başında sektörün devlerinin vaatleri de yer almaktaydı. Microsoft'un Java'dan desteğini çekip .NET platformunu tamamen XML üzerine kurması bu vaatlerin sonuçlarından biriydi. XML, Java yada Microsoft gibi endüstri devleri tarafından değil, HTTP gibi İnternet standartlarını koymuş W3C tarafından standartlaştırıldı ve sektöre sunuldu. Belirli bir ticari grubun XML'i oluşturmaması da popülariteyi arttırdı.

XML Nedir?

XML'i, belirli bir etiket yapısı ve bu yapının sağladığı tanımlama yeteneği ile hiyerarşik olarak verilerin depolanabildiği bir platform olarak ifade edebiliriz. Bu ifadeyi XML'in açılımını irdeleyerek daha net olarak şekillendirelim. Açılımı Extensible Markup Language olan XML, Markup kısmını HTML'deki gibi metni etiketlerle tanımlama yetisinden alsa da Extensible özelliği ile HTML'den ayrılmaktadır. Kendi etiketlerinizi tanımlayabilmeniz genişletilebilirlik özelliğini XML'de mümkün kılmıştır. HTML ve XML örnekleri üzerinde bir karşılaştırma yapalım.

Bu karşılaştırma için aşağıdaki örnekten faydalanalım.

Metin Biçimi

Meyveler,

Elma,

Çilek,

Kayısı,

Erik,

Kiraz.

HTML biçimi

< H1> Meyveler,</H1>

< P> Elma,< /P>

< P> Çilek,< /P>

< P> Kayısı, </P>

< P> Erik, </P>

< P> Kiraz.</P>

XML biçimi

< MEYVESEPETİ>

< TITLE> Meyveler, </TITLE>

< PART>

< LINE> Elma, </LINE>

< LINE> Çilek, </LINE>

< LINE> Kayısı, </LINE>

< LINE> Erik, </LINE>

< LINE> Kiraz. </LINE>

</PART></MEYVESEPETİ>

Örnekten de anlaşılabileceği gibi verinin yapılandırılması günümüzün popüler veritabanlarının kullanmakta olduğu ilişkisel bağlantının aksine hiyerarşik olarak yapılmaktadır. Bu üç farklı biçimden birincisi, yani metin biçimi, metnin kağıt üzerinde veya bir metin dosyasında görülen biçimidir. Herhangi bir özel biçimlemeye ihtiyaç duymadan belli programlar üzerinden bu tür bir veriye ulaşmak kolaylıkla mümkündür. HTML biçiminde ise kullanılan etiketler yardımı ile metne bir görünüm biçimi verilmiştir. Metnin başlığının büyük puntolarla görünebilmesi için <H1> başlık etiketi kullanılmıştır. Bu biçim ile örnek metin, bir tarayıcı üzerinde metin halindeki görüntüsü ile belirecektir. XML biçiminde ise bu basit örnek dahi birçok unsuru sergilemektedir. Örneğin, bu verinin Meyveleri tanımlayan metin olduğu MEYVESEPETİ etiketi ile ifade edilmiştir. Ayrıca bu metnin, anlamı PART etiketi ile belirlenmiştir. İçeriği tanımlayan ancak sunuma ait herhangi bir bilgiyi içermeyen bu XML dokümanı, sunum için herhangi bir platforma ve istenilen bir biçimde XML'in alt unsurları kullanılarak gönderilebilir. XML hakkında daha ayrıntılı bilgiye;

5.5 Uygulama İçeriği

Bu bölümde tez konusunun bir uygulama üzerinden nasıl test edildiği ve uygulamanın nasıl oluşturulduğu, aynı zamanda uygulama geliştirilirken hangi tipte açık kaynak kodların ve araçların kullanılıdığından bahsedilecektir. Tez konusunun uygulama test içeriğinde, UMN MapServer olarak bilinen ve Minnesota Üniversitesi tarafından geliştirilen Coğrafi Bilgi Sistemi yayınlayıcısı kullanılmıştır.Bu araç açık kaynak kodlu olduğu için uygulamada kullanılmıştır. Aynı zamand OGC web servisleri uyumluluğuda vardır. Bu araç temelde PHP kaynakları ile geliştirilmiştir. Ancak Java, C#, Perl, Python programlarına desteği de vardır. UMN MapServer olarak bilinen ve uygulama testi için kullanılan, CBS yayınlayıcısını imaj görüntüleme servisi ve mekensal veri işleçleri için kullanılmıştır. Aynı zamanda web servisleri özellik olarak, veri aktarımını istemci den sunucuya veya sunucudan istemciye geçiş esnasında da kullanılmıştır. Bu web servisleri PHP kaynak kodları ve PHP SOAP desteği sağlayan NuSOAP açık kaynak kodlu sınıfları ile desteklenerek yazılımları geliştirilmiştir. Geliştirilen web servisleri WMS ve WFS olarak adlandırılır. Bir diğer unsur bu servislerin OGC uyumlu olmalarıdır. İstemci tarafında ise istemci sınıflarını ve web servisleri fonksiyonlarını kullanan Java kodları yazılmıştır ve bunlar yine, açık kaynak kodlu editör olan Eclipse platformunda geliştirilmiştir. Bundan sonraki aşamalarda, UMN MapSerevr, PHP, NuSOAP ve Web servisleri tanımlama dili WSDL dokümanının, Java ile yazılan fonksiyonlarda nasıl kullanıldığına dair ayrıntılı bilgiler verilecektir.

5.5.1 UMN Harita Sunucusu (MapServer)

UMN MapServer açık kaynak kodlu bir yazılım ürünü olarak Coğrafi Bilgi Sistemleri platfromunda konumsal verilerin yayınlanması ve internet uygulamalarında gösterimi için rahatlıkla kullanılabilecek bri araçtır. UMN MapServer orjinal anlamda Minnesota Üniversitesi

tarafından geliştirilmiştir. NASA ve Minnesota Doğal Kaynaklar Araştırma Bölümü ile çeşitli projelerde destek amaçlı olarak geliştirilip kullanılmıştır.Minnesota Doğal Kaynaklar Araştırma Bölümü (MNDNR) ve Minnesota Toprak Kaynakları Yönetim Bilgi Merkezi (LMIC) araştırmaları ve kullandıkları uygulamalara göre UMN MapServer tam özellikli bir CBS sistemi değildir, fakat web uygulamaları için raster dosya biçimlerini, aynı zamanda CBS dosya biçimleri olan GIF ve TIFF dosyalarını ve de JPEG, PNG gibi imaj biçimleri için görüntüleri yeterli düzeyde üretebilmektedir.

5.5.2 PHP

PHP, özellikle HTML içerikli web uygulamalarında açık kaynak kodlu fonksiyon özelliği içeren bir kaynak kod türüdür. PHP'nin ana özelliği sunucu odaklı metin olmasıdır. Bu özelliğide web uygulamalarında istemci sunucu arasındaki transferlerde rahatlıkla kullanılabilir anlamı katmaktadır. PHP 4'ten PHP 5'e geçişte bir takım temel özelliklerde geldi. PHP 5, XML tabanlı işlemlere daha iyi destek veren özelliklere sahiptir. Tam anlamıyla XML özellikleri aynı zamanda yeni SOAP modulüdür de denebilir. PHP 4'ün SOAP içeriğindeki eksiklikler, PHP 5'te giderilmiştir. PEAR özelliği PHP'nin açık kaynak kodlu olan ve, kullanıcıların kendi oluşturdukları sınıf metodlarında rahatlıkla değişiklikler yapmalarını sağlayan ve bu değişiklikleri istemci sunucu arasında transfer eden özelliklere sahiptir. PEAR paketleme yöntemi ile özelliklerini kullanabilir. PEAR özelliği bize bu sayede yüksek kalitede garanti sonuç ve sağlamlık getirmektedir. Bu özellikle birlikte her bir oluşturulan fonksiyon paketi için standard doküman sistemi de gerekmektedir. PHP fonksiyonları web uygulamalarında session denilen değişken içeriği taşıyıcı özelliği kullanmaktadır. Bu özelliğin yönetimi web uygulamalarında, özellikle güvenlik ve yetki denetimlerinde çok önemlidir. Bu özellikte her ir değişken içeriği

session id denilen aktarım elemanları ile bilgileri tutmakta ve taşımaktadır. Bu değişken içerikleri istemci tarafında cookie denilen işletim sistemine gömülü dosyalarda veya url içerisinde saklanabilmektedir.Bu session nesnelerinin avantajı, ard arda gelen isteklere karşı, değişkenleri tutan session id'lerin sürekli bünyesinde bulundurduğu değerleri koruyabilmesidir.PHP 5 de halihazırda var olan SOAP kütüphanelerine göre özellikle hız konusunda avantaj sunulmuştur. Bunun yanında SOAP extention zamanla daha istikrarlı olacağı düşünülürse halen tecrübe aşamasındadır.

5.5.3 Java

Java programlama dili, öğrenmesi kolay, temiz ve sade bir yapıya sahiptir. Doğrudan nesne-yönelimli olması, gelişmiş hata yakalama mekanizmasına sahip olması, gelişmiş koleksiyon ve liste ilerlemesi, küme veri yapısı, yansıma özelliği, otomatik veri tipi çevrimi gibi özellikleri onu diğer programlama dillerinden ayırır. Nesne Yönelimli programlama, gerçek hayattaki nesnelerin, programlamaya uygulanması metodudur, örneğin bir ticari uygulama için, hayatımızdaki haliyle Müşteri, Ürün, Kasa gibi kavramlar, çeşitli özellikleri ve yaptıkları eylemlerle uygulamaya aktarılır. Java derleyicisi, bazı istisnai durum oluşturabilecek işler için, hata yakalama ve işlemeyi zorunlu tutar, böylece bazen programcının gözünden kaçabilen hata yakalama işini programcıya hatırlatır. Ayrıca, Java ile yazılan uygulamalarda, program JVM tarafından kontrollü bir biçimde çalıştırıldığından, kendi bellek alanına, dışarıdan başka bir uygulama ulaşamaz ve bellek yönetimi konusu, programcının üzerinden büyük ölçüde kalkmıştır. Java'da yansıma özelliği ile, uygulamanın her biriminden dinamik olarak çalışma anında bilgi edinme olanağı bulunur.

5.5.4 PHP ve SOAP İşlerliği

Bu bölümde PHP 5 için yeni SOAP extension kullanımını anlatılmaktadır. Burada amaç PHP geliştiricilerinin kendi Web Servis Sunucularını yazmasını ve var olan Web Servis uygulamalarından faydalanmasını sağlamaktır. Web Servisleri modern ve çok popüler bir teknoloji olması nedeniyle webin geleceği için bir devrim niteliğindedir. Web servisleri alt yapısı XML,SOAP,WSDL ve UDDI standartları üzerine inşa edilmiştir. Bu nedenle işletim sisteminden ve programla dilinden bağımsız olarak geniş bir uygulama alanı bulmuştur. SOAP (Simple Object Access Protocol - Basit Nesne Erişim Protokolü), önceden yapısı ve içeriği belirlenmiş bilginin kullanıcılar arasında XML yapısı kullanılarak dağıtılmasını sağlamaktadır.

PHP 5'in *SOAP extension'ı* SOAP protokolünün PHP için C deki ilk uygulama denemesi olmuştur. PHP de halihazırda var olan SOAP kütüphanelerine göre özellikle hız konusunda avantaj sunmuştur. Bunun yanında *SOAP extention* zamanla daha istikrarlı olacağı düşünülürse halen tecrübe aşamasındadır. PHP 4 de SOAP kullananlar muhtemelen NuSOAP (http://dietrich.ganx4.com/nusoap) kütüphanesinden faydalanmışlardır. Bu kütüphane başlı başına PHP kullanıcılarına hem SOAP sunucusu hem de SOAP istemcisi yazmada çok büyük kolaylıklar sağlamıştır. Şimdi bu kullanıcılar PHP 5 deki yeni *SOAP extension* ile çalışırken çeşitli kısaltmalarla daha kolaylık sağlandığını göreceklerdir.

Şimdi Kendi SOAP uygulamalarımızı yazabiliriz.

Özellikler Windows kullanıcıları PHP 5 sürümlerinde soap classlarını kullanırken aşağıdaki hatayı alacaklardır:

"**Fatal error**: Class 'SoapClient' not found in **c:\program files\multiservis\apache\htdocs\soap\test.php** on line **3"**

Bu hata php.ini dosyasında "***extension=php_soap.dll***" satırının unutulmasından kaynaklanmaktadır. Uygulamalara başlamadan önce bu satırı php.ini dosyası içerisinde 'Dynamic Extensions' bölümüne eklemeliyiz. Böylece yeni SOAP extension sorunsuz çalışacaktır. Ayrıca php.ini dosyasının SOAP bölümündeki değerleri aşağıdakilerle yer değiştirmeliyiz.

```
Soap]
soap.wsdl_cache_enabled = "1"
; enables or disables WSDL caching feature
soap.wsdl_cache_dir = "/tmp"
; sets the directory name where SOAP extension will put cache
files
soap.wsdl_cache_ttl = "86400"
; (time to live) sets the number of second while cached file
will be used
; instead of original one
```

Tez konumuzun uygulama yapısı için gerekli olan SOAP sunucusunu hazırlayabilir, istemci olarak bu sunucudan veri çekebiliriz. Bu SOAP sunucu yapısının oluşumunu özetleyen örnek, aşağıda yer alan SOAP sunucu oluşturma alt başlığındaki ifadelerde örnekler verilerek de açıklanmaktadır.

SOAP sunucu oluşturma;

Öncelikle bir SOAP sunucu oluşturmamız gerekiyor. Bu kod veritabanından verileri alarak istemcilere dağıtacak. Birinci adım olarak istediğimiz verileri MySQL den alacak bir fonksiyon oluşturmamız gerekiyor.

```
function UrunBilgisi($urun) {
    mysql_connect('localhost','root');
    mysql_select_db('soap');
    $query = "SELECT urun_fiyati FROM depo WHERE urun_ismi = '$urun'";
    $result = mysql_query($query);
    $row = mysql_fetch_assoc($result);
    return $row['urun_fiyati'];
}
```

Yukarıdaki fonksiyonda **$urun** değişkenine atadığımız değere karşılık veritabanından o değere (ürüne) ait fiyat bilgisi geri dönecektir.

```
ini_set("soap.wsdl_cache_enabled", "0"); // WSDL cache özelliği kaldırılıyor.
$server = new SoapServer("depo.wsdl");
$server->addFunction("UrunBilgisi");
$server->handle();
```

Not: Normalde WSDL cache özelliği açık olmalıdır. Ama WSDL dosyasını geliştirirken bu özelliği kapalı olmasına dikkat etmeliyiz.

$server = **new** SoapServer("depo.wsdl");

Yukarıdaki satır WSDL modunda SoapServer objesi oluşturmamızı sağlıyor. Burada belirttiğimiz depo.wsdl dosyasını, istemcinin isteğini anlayacak şekilde oluşturmamız gerekiyor.

Mesaj (**message**) bölümünde iki mesaj tanımlaması yapıyoruz. Birincisi **UrunBilgisi** mesajını postalayan ve dizgi olarak belirttiğimiz **urun** parametresini alan **UrunBilgisiRequest,** diğeri ise **UrunBilgisi** mesajının cevabı olan **UrunBilgisiResponse** mesajıdır. Bu mesaj integer değer taşıyan **Sonuc** cevabını gönderir.

PortType bölümünde, mesaj bölümünde(**message section**) listelenen **UrunBilgisi** mesajını tanımlayacağız. Böylece istek (**request**) ve cevap (**response**) akışını sağlamış olur.

Binding bölümünde mesajın nasıl gönderileceğini ve şifreleneceğini belirtiyoruz. Burada SOAP şifrelemesi ile HTTP üzerinden bir RPC isteği olarak mesajı şifrelinir. Son olarak servis(**service**) bölümünde servisin çalıştığı son nokta URL'sini tanımlanır.

ÖRNEK 1 (depo.wsdl): Örnek 1 de WSDL dokümanının nasıl olduğu gösterilmektedir.

```
<?xml version ='1.0' encoding ='UTF-8' ?>
<definitions name='UrunBilgisi'
  targetNamespace='http://ornek.org/UrunBilgisi'
  xmlns:tns=' http://ornek.org/UrunBilgisi '
```

```
  xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
  xmlns:xsd='http://www.w3.org/2001/XMLSchema'
  xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/'
  xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
  xmlns='http://schemas.xmlsoap.org/wsdl/'>
<message name='UrunBilgisiRequest'>
  <part name='urun' type='xsd:string'/>
</message><message name='UrunBilgisiResponse'>
  <part name='Sonuc' type='xsd:integer/>
</message><portType name='UrunBilgisiPortType'>
  <operation name='UrunBilgisi'>
    <input message='tns:UrunBilgisiRequest'/>
    <output message='tns:UrunBilgisiResponse'/>
  </operation></portType><binding name='UrunBilgisiBinding'
type='tns:UrunBilgisiPortType'>
  <soap:binding style='rpc' transport='http://schemas.xmlsoap.org/soap/http'/>
  <operation name='UrunBilgisi'>
    <soap:operation soapAction='urn:xmethods-delayed-urun#UrunBilgisi'/>
    <input><soap:body use='encoded' namespace='urn:xmethods-delayed-urun'
       encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
    </input><output><soap:body use='encoded' namespace='urn:xmethods-
delayed-urun' encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
</output> </operation></binding><service name='UrunBilgisiServisi'>
  <port name='UrunBilgisiPort' binding='UrunBilgisiBinding'><soap:address
location='http://[tam adresi girin]/server1.php'/>  </port></service>
</definitions>
```

Şimdi oluşturduğumuz SoapServer objesine **SoapServer::addFunction()** fonksiyonu kullanarak bağlanırız.

```
$server->addFunction("UrunBilgisi");
```

Son olarak **SoapServer::handle** SOAP istekleri alırız.

```
$server->handle();
```

ÖRNEK 2 (server1.php):

```
<?php
function UrunBilgisi($urun) {
   mysql_connect('localhost','root');
   mysql_select_db('soap');
   $query = "SELECT urun_fiyati FROM depo WHERE urun_ismi = '$urun'";
   $result = mysql_query($query);
   $row = mysql_fetch_assoc($result);
   return $row['urun_fiyati'];
}

ini_set("soap.wsdl_cache_enabled", "0"); //  WSDL cache özelliği kaldırılıyor.
$server = new SoapServer("depo.wsdl");
$server->addFunction("UrunBilgisi");
$server->handle();
?>
```

Şimdi bu serverdan veri alacak olan istemciyi yazabiliriz.

ÖRNEK 3 (istemci1.php):

```
<?php
  $client = new SoapClient("depo.wsdl");
  print($client->UrunBilgisi("AMD 2000"));
  echo "<br>";
  print($client->UrunBilgisi("INTEL P4 3.0"));
?>
```

Kodu çalıştırdığımızda “AMD 2000” ve “INTEL P4 3.0” adlı ürünlerin fiyat bilgileri ekrana basılacaktır. Ama bu ürünlerden birisi SOAP serverında yok ise isteklerimiz tanımsız olacaktır. Bu aşamada **SoapFault()** fonksiyonu ile hatayı

kontrol edebiliriz. Daha önce depo.wsdl olarak kaydettiğimiz dosyanın içeriği yeni bir dosyaya kopyalayalım ve ismini depo2.wsdl yapalım. Dosyayı açıp 43ncü satırdaki soap:address karşılığına gelen yeri server2.php olarak değiştirelim.

Şimdi yeni server ve istemcimizi yazalım.

ÖRNEK 4 (server2.php):

```
<?php
class UrunServisi {
        function UrunBilgisi($urun) {
            mysql_connect('localhost','root');
            mysql_select_db('soap');
            $query = "SELECT urun_miktari,urun_fiyati FROM depo WHERE
urun_ismi = '$urun'";
            $result = mysql_query($query);
            $row = mysql_fetch_assoc($result);
            $urunfiyati=$row['urun_fiyati'];
            if (isset($urunfiyati)) {
            return $urunfiyati;
            } else {
            throw new SoapFault("Server","Aradığınız '$urun' urunu veritabanında
bulunmuyor.");
            }}}
ini_set("soap.wsdl_cache_enabled", "0"); // WSDL cache özelliği kaldırılıyor.
$server = new SoapServer("depo2.wsdl");
$server->setClass("UrunServisi");
$server->handle();
?>
```

Gördüğünüz gibi UrunServisi class'ı ile SoapServer'a bağlanmak için **SoapServer::setClass()** kullanırız.

ÖRNEK 5 (istemci2.php):

```
<?php
  $client = new SoapClient("depo2.wsdl");
  try {
    echo "<pre>\n";
    print($client->UrunBilgisi("SEMPRON 2800"));
    echo "\n";
    print($client->UrunBilgisi("CENTRINO 1500"));
    echo "\n</pre>\n";
  } catch (SoapFault $exception) {
    echo $exception;
  }
?>
```

Bu örnekte birinci ürünümüzün fiyatı sorunsuz gelecektir. İkinci ürün ise veritabanında bulunmadığı için server bu ürün için bize şu uyarıyı verecektir.

```
SoapFault exception: [SOAP-ENV:Server] Aradığınız 'CENTRINO 1500' urunu
veritabanında bulunmuyor. in c:\program
files\multiservis\apache\htdocs\soap\depo\istemci2.php:7
Stack trace:
#0 c:\program files\multiservis\apache\htdocs\soap\depo\istemci2.php(7):
SoapClient->UrunBilgisi('UrunBilgisi', Array)
#1 {main}
```

Soap istemcisi yazarken kullandığımız `SoapClient() yapısı` bize birçok imkan sunuyor. Eğer yazdığınız kodlarda SOAP mesajlarının nasıl olduğu merak ediyorsak bu örneğe dikkat etmeliyiz.

ÖRNEK 6 (istemci3.php):

```
<?php
  $client = new SoapClient("depo2.wsdl",array(
    "trace"      => 1,
    "exceptions" => 0));
  $client->UrunBilgisi("INTEL P4 2.8");
  print "<pre>\n";
  print "İstek :\n".htmlspecialchars($client->__getLastRequest()) ."\n";
  print "Sonuç:\n".htmlspecialchars($client->__getLastResponse())."\n";
  print "</pre>";
?>
```

Burada SoapClient() yapısında:

- trace: İstemcinin SOAP isteğini ve cevabını kaydeder(varsayılan olarak bu özellik kapalıdır)
- exceptions: İstemcinin istisnai mekanizmaları kontrol etmesini sağlar(varsayılan olarak bu özellik açıktır)

SoapClient::__getLastRequest: Son SOAP isteğini döndürür. Bu fonksiyon sadece trace ile oluşturulmuş SoapClient ile çalışır.

SoapClient::__getLastResponse: Son SOAP cevabını döndürür.

Örnek 6 çalıştırıldığında şöyle bir sayfa karşımıza çıkacaktır. Ben görüntüyü daha anlaşılır kılmak için biraz düzenlenmiştir.

İstek :

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:ns1="urn:xmethods-delayed-urun"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
```

```
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
    <ns1:UrunBilgisi>
        <urun xsi:type="xsd:string">INTEL P4 2.8</urun>
    </ns1:UrunBilgisi>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Sonuç (Cevap):

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="urn:xmethods-delayed-urun"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body><ns1:UrunBilgisiResponse>
        <Sonuc xsi:type="xsd:integer">350</Sonuc>
</ns1:UrunBilgisiResponse>
```

Artık PHP 5'in yeni SOAP extension ile kendi SOAP sunucunuzu oluşturabilir ve kullanıcılarınızın bu hizmetten faydalanmasını sağlayabiliriz. Eğer böyle bir uygulamayı kullanıcı adı ve şifresi ile yetkilendirmek isteseydik SoapClient() yapısının imkanlarını kullanmamız yeterli olurdu.

```
$client = new SoapClient("depo.wsdl", array('login'          => "kullanici_adi",
                                            'password' => "sifre"));
```

5.5.5 NuSOAP kullanarak WSDL dokümanı oluşturma ve programlama

Bu bölümde SOAP istemci/sunucu methodu kullanımında NuSOAP açık kaynak kodlu sınıf nesnelerinin nasıl bütünleştirildiği ve bunun sonucunda oluşturulan SOAP servisinin, WSDL dokümanına nasıl erişildiği tanımlanmaktadır. Bir WSDL dokümanı servis için metadata denilen genel veri dağarcığını üretir. NuSOAP bünyesinde bulundurduğu SOAP servis sınıfları ile servilserin WSDL dokümanı oluşturmalarını sağlar. Biz SOAP servisleri yazdığımızda eğer bu servisleri NuSOAP destekli sınıflarla bütünleştiriyorsak, servislerimizin WSDL dokümanlarına da rahatlıkla ulaşabliriz demektir. Bu bağlamda, servise ait kodlar, WSDL dokümanını belirli bir sıraya göre oluşturur. Servise ait tüm ayrıntılı tariflendirmeler, WSDL dokümanında yer almaktadır. Kullanıcılar WSDL dokükmanlarını inceleyerek web servislerinin, neler yaptıklarını ve ne işe yaradıklarını görebilirler. Aşağıda WSDL oluşturabilen NuSOAP destekli program kodu örneğini göreblirsiniz.

```
<?php
// Pull in the NuSOAP code
require_once('nusoap.php');
// Create the server instance
$server = new soap_server();
// Initialize WSDL support
$server->configureWSDL('hellowsdl', 'urn:hellowsdl');
// Register the method to expose
$server->register('hello',              // method name
    array('name' => 'xsd:string'),      // input parameters
    array('return' => 'xsd:string'),    // output parameters
    'urn:hellowsdl',                    // namespace
    'urn:hellowsdl#hello',              // soapaction
```

```
    'rpc',          // style
    'encoded',    // use
    'Says hello to the caller'         // documentation);
// Define the method as a PHP function
function hello($name) {
        return 'Hello, ' . $name;}
// Use the request to (try to) invoke the service
$HTTP_RAW_POST_DATA = isset($HTTP_RAW_POST_DATA) ?
$HTTP_RAW_POST_DATA : '';
$server->service($HTTP_RAW_POST_DATA);
?>
```

Yukarıda yer alan PHP kodu, XML tabanlı, XML web servisi oluşturabilmektedir. Bu servisin WSDL linkine ise, servisin yer aldığı url'nin sonuna ? işareti ile wsdl yazdığımızda, WSDL dokümanına ulaşabiliriz. Bu senaryo PHP destekli NuSOAP sınıf metodlarında parametre olarak tariflendirilmektedir. (ör:http://localhost/phpservis/hellowsdl.php?wsdl).
Aşağıda oluşturulan WSDL dokümanı içeriği yer almaktadır.

```
<?xml version="1.0"?>
<definitions xmlns:SOAPENV="http://schemas.xmlsoap.org/
soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:si="http://soapinterop.org/xsd"
 xmlns:tns="urn:hellowsdl"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns="http://schemas.xmlsoap.org/wsdl/"
 targetNamespace="urn:hellowsdl">
```

```
 <types>
 <xsd:schema targetNamespace="urn:hellowsdl">
 <xsd:import
namespace="http://schemas.xmlsoap.org/soap/encoding/" />
 <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/" />
 </xsd:schema>
 </types>
 <message name="helloRequest">
  <part name="name" type="xsd:string" />
  </message>
  <message name="helloResponse">
  <part name="return" type="xsd:string" />
  </message>
  <portType name="hellowsdlPortType">
  <operation name="hello">
  <documentation>Says hello to the caller</documentation>
  <input message="tns:helloRequest"/>
  <output message="tns:helloResponse"/>
</operation>
</portType>
<binding    name="hellowsdlBinding"
type="tns:hellowsdlPortType">
<soap:binding     style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
<operation name="hello">
<soap:operation
soapAction="urn:hellowsdl#hello" style="rpc"/>
<input>
<soap:body use="encoded"    namespace="urn:hellowsdl"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
</input>
<output>
```

```
<soap:body use="encoded"    namespace="urn:hellowsdl"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
</output></operation></binding><service name="hellowsdl">
      <port name="hellowsdlPort" binding="tns:hellowsdlBinding">
        <soap:address
        location="http://localhost/phphack/hellowsdl.php"/>
      </port>
 </service>
</definitions>
```

5.5.6 Uygulama Testi için Web Servis Sistem Mimarisi

Bu bölümde uygulama test içeriğini oluşturan web servis sistem mimarisi, araçlar, servisi oluşturan algoritmik safhalar ve bu süreç işlenirken hangi araçlar, hangi yazılım geliştirme ortamları ve hangi açık kaynak kodlu yazılım ürünleri kullanılmıştır, belirtilmektedir. Uygulama yapısında WMS web harita görüntüleme servisinde ve onun WSDL dokümanı için SOAP istek/yanıt süreci ele alınmıştır. Bu bağlamda, WSDL dokümanı üreten SOAP servisleri, NuSOAP aracı ile geliştirilmiştir. NuSOAP daha önce de belirtildiği gibi WSDL dokümanına sahip SOAP servisleri üretmeye yarayan PHP tabanlı bir sınıf aracıdır. Uygulamanın istemci tarafında Java yazılım ürünü kullanılmıştır. Java'nın (API) Uygulama Programlama Arayüz safhası olarak belirtilen (Java Web Services Developer Pack 1.6.) Java Web Servisleri Geliştirici Paketi istemci tarafında web servislerinin uygulama fonksiyonlarına erişmek ve bu erişimin sonuçlarını arayüz platformunda göstermek için kullanılmıştır. Burada yazılım geliştirme platformu olarak, yine açık kaynak kodlu ürün olan Eclipse platformu kullanılmıştır. Aynı zamanda web sunucusu olarak, hem UMN MapServer hemde Java uygulamalarına destek vermesi sebebiyle Apache Tomcat Server kullanılmıştır. UMN MapServer CBS sunucusu ve yayınlayıcısı olarak, internet uygulamalarında rahatlıkla kullanıldığından ve bizim için en önemli özelliği olan (OGC) Açık Kaynak Kodlu CBS konsorsuyumuna uyumlu olan araç olduğu için kullanılmıştır. Aşağıdaki analatımlarda açık kaynak

kodlu araçlar, uygulama programlama arayüzleri (API), ve yayınlayıcı özelliklerinin açıklamaları hem istemci tarafı hem sunucu tarafı için açıklanmıştır.

5.5.6.1 Java Web Servis Geliştirici Paketi (Java Web Service Developer Package)

Java WSDP yüklendiği zaman, sistemde otomatikman, bu paketin içerdiği araçları ve özellikleri bulunduran dosyalar oluşur. Bu dosyalar içeriğinde Web servislerini ve özelliklerini test edebilecek, kullanabilecek, hazır ortamları bünyesinde bulundurur. Burada en önemli araç olarak Java XML Pack, denilen XML verilerine ulaşımı sağlayan araçlardır. Bu araçlar Java (API) uygulama programlama arayüzü olan XML kayıtlarını kullanan JAX araçlarıdır.

XML İşleçleri için Java Uygulama Programlama: JAXP, W3C uyumlu XML (API) uygulama programlama arayüzleri olan (SAX, DOM, XSLT) ürünleri kullanan araçtır. Bu uygulama programlama arayüzü XML ayrıştırıcı kümeleri için standart arayüzler üretir. Bunlara örnek olarak, (Apache Software Foundation) Apache Yazılım Fonunun, Xerces 2 ürünü en önemli XML ayrıştırıcı örneğidir denilebilir. Aynı zamanda bu ürün Java 2 Standart platformunuda desteklemektedir.

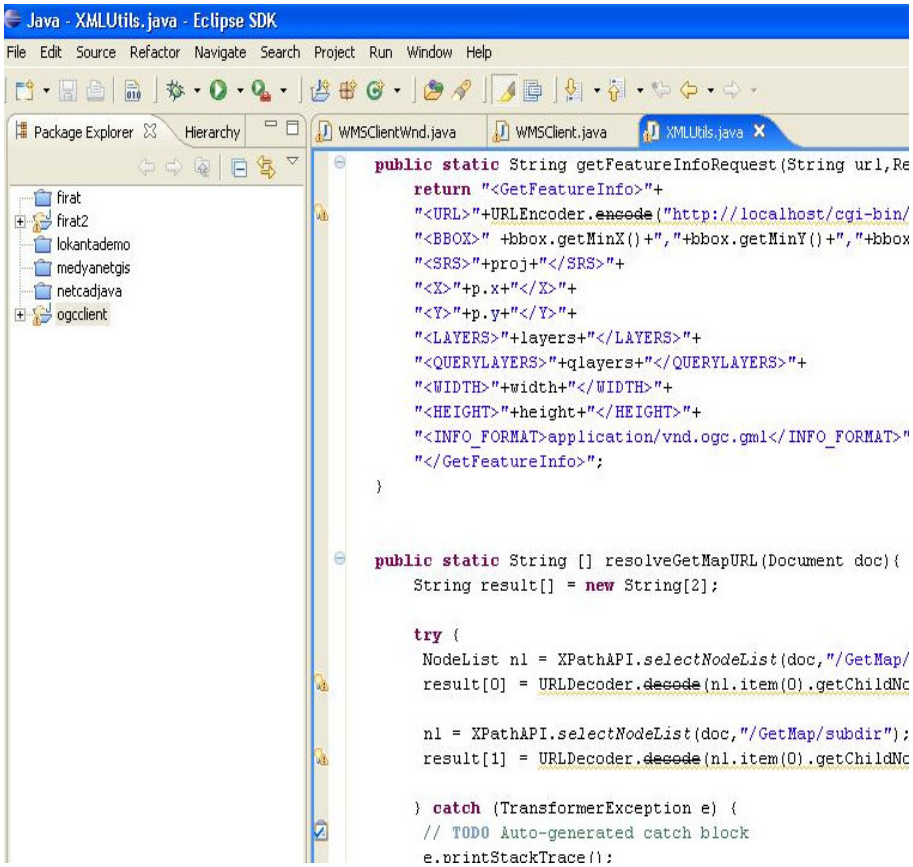
XML Mesajlı veriler için Java Uygulama Programlama Arayüzü: JAXM uygulama programlama ürünü, XML mesajları içeren dokümanları Java tabanlı platfromlarda, mesajların gönderilmesi ve alınması gibi süreçleri XML yoluyla yaparken kullanılan üründür. EbXML gibi, yüksek aşamada standart mesaj protoköllerini, SOAP mesaj protoköl desteği ile üreten yapılarda kullanılır. SAAJ spesifikasyonları, SOAP bağlantılarını eş zamanlı içeren ve JAXM tarafından genelde kullanılan ürünler olarak örnek verilebilir.Aynı zamanda şu an itibari ile bu ürün eş zamanlı olmayan mesaj protoköllerini de içerebilen özelliklere sahiptir. JAXM eş zamanlı olarak kullanılırsa, mesaj üreticisinin içeriğine ve mesajın içeriğine göre kullanılmaktadır. JAXM uygulama arayüz ürünü, ebXML Transport,

Routing, ve Packaging spesifikasyonlarına göre versiyon 1.0 kontrolü ile uygulamalarda kullanılır.

XML Kayıtları için Java Uygulama Programlama Arayüzü: Web servisleri yayınlanması ve servisler hakkında bilgileri saklamak için kullanılan XML kayıtlarını bünyesinde tutabilen üründür. UDDI kayıtlarında yer alan web servislerinin bilgilerine XML dokümanları ile ulaşabilmek için kullanılan uygulama programlama arayüzüdür.

5.5.6.2 Eclipse

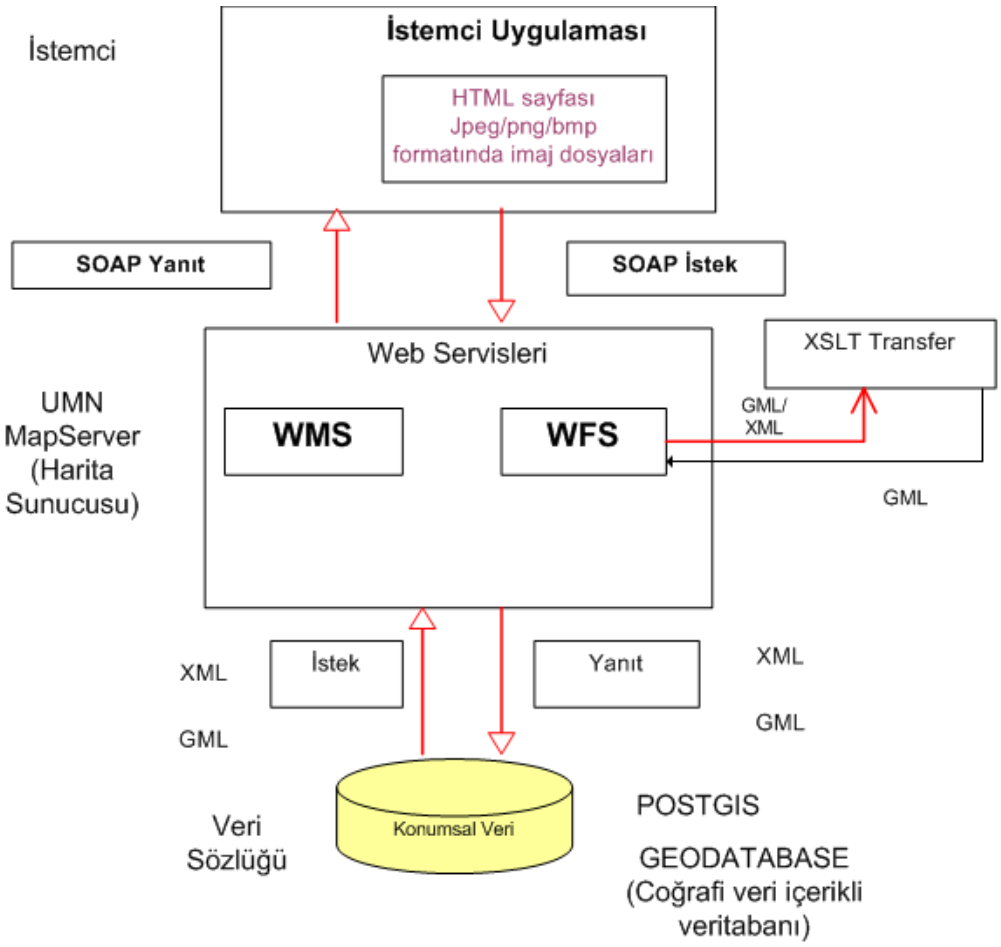
Eclipse açık kaynak kodlu bir komite olarak geliştirilmiştir. Zmanında IBM'in geliştirdiği ve halen IBM WebSphere Studio'nun temeli olarak kullandığı Eclipse, tamamen özgür ve açık seçik kaliteli bir Java uygulama geliştirme ortamıdır. Form tasarımı ve web uygulamaları desteği, eklentiler ile sağlanabilir. http://www.eclipse.org/ Uygulama geliştirme platformunda Java kodlarının geliştirilmesi, yazılması ve Web servis fonksiyonlarının istemci tarafından özelliklerine ulaşılabilmesi için sınıfların yazıldığı ortam olarak kullanılmıştır.



Şekil 5.8 Eclipse Platformu

5.5.6.3 Web Servis Mimarisi

Bu bölümde, uygulama safhalarında web servislerinin nasıl geliştiğini ve nasıl kullanıldığını, aynı zamanda Java WSDP ile Eclipse platformunda istemci tarafının nasıl geliştirldiğini şekil tabanlı gösterimi, ve geliştirilen algotirmanın aşama aşama anlatımını içeren süreç tarif edilmektedir. Şekil 5.9'da WMS WFS servislerinin uygulamadaki yerleri ve nasıl çalıştırıldıkları gösterilmektedir.



Şekil 5.9 Uygulama için Web Servis Sistem Mimarisi

Açık Kaynak Kodlu CBS uyumlu WMS ve WFS servislerinin nasıl işlerliğini ifade eden figüre yukarıda gösterilmektedir. Bundan sonra bu figürü işleyen bir uygulamanın istemci ve sunucu arasındaki iletişiminin nasıl olduğu ve algoritma çalışma safhalarının nasıl işlediği anlatılabilir.

1- İstemci tarafında sunucu veri seçerek XML sorgu içeriğini oluşturur.

2- İstemci tarafından kullanıcı girdi verisini servise gönderir.

3- Sunucu tarafı, istemciden gelen girdi bilgilerine SOAP üzerinden yanıt verir.

4- Aynı zamanda sunucu, client tarafından gelen verilere web servisinden sonuç döndürür.

5- Aynı zamanda bu sonucun WSDL dokümanıda oluşturulur.

6- And then server sends output XML to the client part.

7- Bundan sonra sunucu üzerindeki metodlarla, sonuç bilgilerini istemciye gönderir.

8- İstemci tarafı, gelen verileri parse eden fonksiyonlarla, son kullanıcının görebileceği hale getirir.

9- İstemci tarafındaki XML ayrıştırıcı metodları ile veriler XML'de parametrik url metodlarına dönüştürülür.

10- Yine istemci tarafında sınıflarla url'ler görüntüleme fonksiyonları ile html biçiminde jpeg/png/bmp biçiminde görüntülenirler.

Bu bölümde anlatılan, ilişkili XML, GML, WSDL dokümanları, İmaj görüntüleri ve Kaynak Kodları Ek A, Ek B ve Ek C'de gösterilmektedir.

BÖLÜM 6

GELECEKTE YAPILACAKLAR VE SONUÇ

6.1 Gelecekte Yapılacaklar

Bu tez araştırmasında, OGC (Açık kaynak kodlu CBS konsorsuyum) uyumlu web servislerini tanımladık. OGC servis sistem mimarisi uyumlu WMS ve WFS servislerini ve özelliklerini tanımladık. İlerideki araştırmalarımızda uygulamamıza destek ve genişletmek amacı ile, OGC uyumlu diğer servisler olan, WCS (Web Coverage Service), CPS (Coverage Portrayal Service ve SLD (Styled Layer Descriptor) servislerinin tanımlamaları ve özelliklerini uygulama üzeinde çalıştırarak görüntülemektir. Bütün bu servisler OGC özelliklerine uyumludur. Aynı zamanda hepsinin WSDL dokümanları oluşturulabilmektedir. Uyugulamaya bu servisleri ekleme işlemini aşama aşama yaparak servislerin özelliklerini ve oluşabilecek hataları rahatlıkla ayıklayabiliriz. İlk olarak tüm servisleri bir uygulama üzerinde gösterdiğimiz zaman, bu servisleri arasında birlikte işlerlilik sorunları var ise bunları çözmek için araştırmalar yapabiliriz. Burada diğer önemli yapılacak iş olarak, WCS servisinin UDDI kayıtları üzerinden servislerin, XML sorgular ile otomatikman bulundurulması ve imaj servisi ile etkinleştirerek görüntüsünün alınması planlanmıştır. Böylelikle son kullanıcı arayüz üzerinde bir XML sorgu oluşturup, servislerin bulunduğu UDDI kayıtlarına gönderdiğinde, UDDI'ın gönder-bul-yayınla metodolijisi devreye girerek, otomatikman servis bulundurulup, servisten sonuç dönerek kullanıcıya iletilebilecektir. Burada planlanan diğer iş olarak, uygulamada POSTGIS mekansal veritabanı kullanılmıştır. Amaç, UMN MapServer, bir çok veritabanı yapısını desteklediği için,

burada platform bağımsız bir veritabanı kullanımı sağlamakta en önemli işlerden bir tanesidir.

6.2 Sonuç

Konumsal verilerde platformlar ve bölümler arası veri paylaşımına ve birlikte işlerliliğe ihtiyaç vardır. Mekansal verilerin birlikte işlerlilik ve veri paylaşım kabiliyetleri, ISO/TC211 ve OGC standart tanımlamaları olarak kabul edilir. Buradan yola çıkarak, web servisleri standartlarında, CBS servisleri ile diğer servis platformlarının adaptasyonu sağlanabilmektedir. Bu raporda OGC uyumlu CBS servislerini, web servis teknolojisi ve OGC özellikleri kullanarak tariflendirmek ve uygulama üzerinde göstermekteyiz. Herhangi bir işlemeyen kullanıcı özniteliği için, herhangi bir kurulum, ödenek veya öğrenim gerektirmeden, web servis modelleri CBS servisleri için kullanıcılar üretebilmektedir. Coğrafi ve fiziksel uygulamalar, coğrafi veri üreteçleri ve işleçlerinin olduğu çeşitli şekillerde özellikler içerirler. CBS servis sistem mimarisi kullanarak, oluşturduğumuz coğrafi ve fiziksel uygulamaları, yeni sunucumuza entegre etmiş oluruz. Web servisleri bu bağlamda, platform bağımsız, işletim sistemi bağımsız, dil bağımsız ve kolayca genişletebilinen sistemlerdir diyebiliriz.

Uyugulamamızda OGC özelliklerini kullanırken, web servislerimizin genişletilebilme yapısındaki performans aksaklıklarını düşünerek hareket ettik. Çünkü bu aksaklıklar bize daha sonraki servisleri genişletme sürecinde problemler yaratabilirdi. Web servis modellerimizden WMS içeriğinde yer alan imaj ve kabiliyet dokümanlarının çok geniş olabilme ihtimallerini göz önüne alarak, araştırmalarımızda WMS performansını güçlü kılacak teknikleri araştırdık. Görüntüleme yapısı veri yoğunluğu sebebiyle yavaş, üst üste bindirme yavaşlığı veya geniş alana yayılmadan dolayı yavaşlık gibi sorunlarlada karşılaşabilmekteydi. Komplike

haritalar aynı zamanda zorunlu geniş kabiliyet dosyaları ve çekilen verileri gösterme özelliklerinden dolayı şişebilmektedirler. Harita sunucumuz üzerinde araştırmalarımızda dikkat ettiğimiz unsurlar, performans ve uygunluk gibi faktörlerdir.

Web servis tekniklerinde bir diğer unsurda kısayollar ile verilere ulaşabilmektir. Web servisleri XML tabanlı mesajları, yeniden kodlayarak veya değiştirerek, HTTP üzerinden transfer edebilirler, OGC spesifikasyonlarında bu işlem URL tabanlıda gelişebilmektedir. Web servisleri özellikleri sayesinde, içiçe kullanılan nesne teknikleri, farklı platformlar ve dillerin kullanıldığı ve içiçe kullanılan sistemlerin birlikte işlerlilik ürünlerine de rahatlıkla ulaşılabilmektedir. Ve bu sistem İnternet ortamında da kullanılabilmektedir. Web servisleri uygulamarına, web oratmında uzaktan erişmekte mümkündür. Web servisleri İnternet ortamında XML, SOAP, WSDL gibi özellikler ile oluşum senaryolarını meydana getirmektedir. Basit işlerli bir web servisinin, başka içiçe servislerin olduğu platforma aktarımı veya komplike bir servisin platform bağımsız şekle dönüşümü sağlanabilmektedir.

CBS servis tabanlı servisler ile başka platformlardaki CBS servisleri bütünleştirilerek, Web servisi haline dönüşebilmektedir. Araştırmalarımızda, OGC serislerini web servisi haline dönüştürmenin kolay bir süreç olmadığını gördük. Açık kaynak kodlu CBS konsorsuyumu spesifik parametrelere sahiptir, her bir parametre özelliği WSDL dokümanında farklı bir anlam ifade etmektedir. Kullanıcılar istemci tarafında kendi sınıf parçalarını oluşturarak bu parametrelerin özelliklerine erişebilirler. WSDL dokümanlarının parametre özellikleri her defasında sunucu tarafında değişirse, istemci kullanıcıları her defasında farklı servis tanımlamalarına ulaşırlar. Bundan da anlaşılacağı gibi, sunucu tarafında servislerin tek WSDL tanımları olmalı ve farklı kullanıcılar bunlara ulaşırken ortak sunucuyu kullanarak sonuca ulaşmalıdırlar. Dolaysıyla OGC kabiliyet özelliklerinin sunucu tarafında sabit ifadelere sahip olması gerekmektedir.

Sonuç olarak bu tez araştırmasında, OGC uyumlu web servislerinin nasıl kullanıldığı, ne gibi özelliklere sahip olduğu, bir uygulama üzerinde nasıl test edileceği gibi kavramlar araştırılmıştır. Burada geliştirilen uygulamada açık kaynak kodlu araçlar, uygulama programlama arayüzleri, açık kaynak kodlu yazılım geliştirme

ortamları gibi ürünlerin kullanılmasına önem verilmiştir. CBS yayınlayıcısı olarak kullanılan UMN MapServer açık kaynak kodlu ve OGC destekli bir ürün olduğundan, uygulama için bu ürün kullanılmıştır. Ancak araştırmalar ve incelemeler esnasında UMN MapServer'ın OGC uyumlu web servislerinin WSDL dokümanını oluşturamadığı gözlenmiştir. Uygulamda geliştirme amacı olarak, hem OGC uyumlu web servislerinin yazılımını geliştirmek, hemde kullanılan UMN MapServer CBS görüntü yayınlayıcı aracının eksikliği olan WSDL dokümanı oluşturamamasını çözmek belirlenmiştir. Geliştirilen çözüm sayesinde OGC uyumlu açık kaynak kodlu UMN MapServer harita yayınlayıcısının WSDL dokümanı oluşturması ve uygulamda WMS ile WFS servislerinin kullanılması sağlanarak Coğrafi Bilgi Sistemleri ile Servis Bağımlı Mimari yapısının geliştirilmesi sağlanmıştır.

KAYNAKLAR

[1]. OGC (Open Geospatial Consortium) official web site http://www.opengeospatial.org/

[2]. GIS Research at Community Grids Lab, Project Web Site: http://www.crisisgrid.org.

[3]. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D. "Web Service Architecture." W3C Working Group Note, 11 February 2004. Available from http://www.w3c.org/TR/ws-arch.

[4]. Jeff De La Beaujardiere, OpenGIS Consortium Web Mapping Server Implementation Specification 1.3, OGC Document #04-024, August 2002.

[5]. Kris Kolodziej, OGC OpenGIS consortium, OpenGIS Web Map Server Cookbook 1.0.1, OGC Document #03-050r1, August 2003.

[6]. Lalonde, W. (ed.), Styled Layer Descriptor(SLD) Implementation Specification 1.0.0, OGC Document #02-070, August 2002

[7]. Vretanos, P. (ed.), Web Feature Service Implementation Specification (WFS) 1.0.0, OGC Document #02-058, September 2003.

[8]. Ahmet Sayar, Marlon Pierce, Geoffrey Fox OGC Compatible Geographical Information Services Technical Report (Mar 2005), Indiana Computer Science Report TR610

[9]. Simon Cox , Paul Daisey, Ron Lake, Clemens Portele, Arliss Whiteside, Geography Language (GML) specification 3.0, Document #02-023r4., January 2003.

[10]. Galip Aydin, Marlon Pierce, Geoffrey Fox, Mehmet Aktas and Ahmet Sayar "Implementing GIS Grid Services for the International Solid Earth Research Virtual Observatory". Submitted to Journal of Pure and Applied Geophysics.

[11]. Mehmet Aktas, Galip Aydin, Andrea Donnellan, Geoffrey Fox, Robert Granat, Lisa Grant, Greg Lyzenga, Dennis McLeod, Shrideep Pallickara, Jay Parker, Marlon Pierce, John Rundle, Ahmet Sayar,

and Terry Tullis “iSERVO: Implementing the International Solid Earth Research Virtual Observatory by Integrating Computational Grid and Geographical Information Web Services” Technical Report December 2004, to be published in Special Issue for Beijing ACES Meeting July 2004.

[12]. John D. Evans, OGC Web Coverage Service (WCS) Specifications 1.0.0, Document #03-065r6 August 2003

[13]. Jérôme Sonnet, Charles Savage. OGC Web Service Soap Experiment Report 0.8 Document#03-014, Jan 2003.

[14]. Douglas Nebert, Arliss Whiteside, OpenGIS Consortium Catalogue Services Specifications 2.0. OGC Document# 04-021r2, May 2004.

[15]. Fran Berman, Geoffrey C, Fox, Anthony J. G. Hey., Grid Computing: Making the Global Infrastructure a Reality. John Wiley, 2003.

[16]. Don Box, David Ehnebuske, Gobal Kakivaya, Andrew Layman, Dave Winer., Simple Object Access Protocol (SOAP) Version 1.1, May 2000,.

[17]. Christiensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, Web Service Description Language (WSDL) Version 1.1, March 2001.

[18]. Ferraiolo, Dean Jackson, Scalable Vector Graphics (SVG) Sprcification 1.1., January 2003.

[19]. Mark Little, Eric Newcomer, Greg Pavlik., OASIS Web Services Context Specifications (WS-Context) 0.8. November 2004.

[20]. Jeff Lansing., OWS1 Covarage Portrayal Service (CPS) Specifications 1.0.0, Document #02-019r1 February 2002.

[21]. A Note on Distributed Computing, S. C. Kendall, J. Waldo, A. Wollrath, G. Wyant, A Note on Distributed Computing, Sun Microsystems Technical Report TR-94-29, November 1994. Available from http://research.sun.com/techrep/1994/abstract-29.html.

[22]. Web Services Technologies http://www.w3.org/2002/ws/.

[23]. Foster, I. and Kesselman, C., (eds.) The Grid 2: Blueprint for a new Computing Infrastructure, Morgan Kaufmann (2004).

Message based middleware project at Community Grids Lab, Project Web Site: http://www.naradabrokering.org/

[24]. Project OnEarth at NASA JPL (Jet Propulsion Lab) http://onearth.jpl.nasa.gov/

[25]. W3C XSL Web Site : http://www.w3.org/Style/XSL/

[25]. Aydin G., SERVOGrid WFS implementation web page: http://www.crisisgrid.org/html/wfs.html

[26]. Aktas M., SERVOGrid Information Services Web Site, http://grids.ucs.indiana.edu/~maktas/fthpis

[27]. Bellwood, T., Clement, L., and von Riegen, C. (eds) (2003), UDDI Version 3.0.1: UDDI Spec Technical Committee Specification. Available from http://uddi.org/pubs/uddi-v3.0.1-20031014.htm

[28]. Marlon Pierce, Choonhan Yoon and Geoffrey Fox: Interacting Data Services for Distributed Earthquake Modeling. ACES Workshop at ICCS June 2003 Australia.

[29]. Tiampo, K. F., Rundle, J. B., McGinnis, S. A., & Klein, W. Pattern dynamics and forecast methods in seismically active regions. Pure Ap. Geophys. 159, 2429-2467 (2002).

[30]. Geoffrey Fox, et al, Complexity Computational Environnent (CCE) Architecture. Technical Report available from http://grids.ucs.indiana.edu/ptliupages/publications/CCE%20Architecture.doc

[31]. Alameh N., Chaining Geographic Information Web Services, IEEE Internet Computing, Sept-Oct 2003, 22-29.

[32]. Open GIS Consortium Inc. OWS-1 Registry Service.
2002-07-26.

[33]. Castor http://castor.exolab.org

[34]. XMLBeans (http://xml.apache.org/xmlbeans)

[35]. Philippe Duschene, Jerome Sonnet, WMS Change Request: Support for WSDL and SOAP, OGC Document #04-050r1, April 2005.Available at http://portal.opengeospatial.org/files/index.php?artifact_id=9541
George Percivall, OpenGIS Consortium Reference Model 0.1.3, OGC Document #04-040, September 2003.
[36]. Roel Nicolai, The OpenGIS® Abstract Specification, Topic 2: Spatial referencing by coordinates 2.0.0. Document #03-073r1, October 2003.
[37]. SERVOGrid IS implementation. Available at http://grids.ucs.indiana.edu/~maktas/fthpis/index.html
[38]. Jeff Lansing., OWS1 Covarage Portrayal Service (CPS) Specifications 1.0.0, Document #02-019r1 February 2002.
[39]. A Note on Distributed Computing, S. C. Kendall, J. Waldo, A. Wollrath, G. Wyant, A Note on Distributed Computing, Sun Microsystems Technical Report TR-94-29, November 1994. Available from http://research.sun.com/techrep/1994/abstract-29.html
[40]. Foster, I. and Kesselman, C., (eds.) The Grid 2: Blueprint for a new Computing Infrastructure, Morgan Kaufmann (2004)
[41]. Project OnEarth at NASA JPL (Jet Propulsion Lab) http://onearth.jpl.nasa.gov/
[42]. Panagiotis A. Vretanos, OpenGIS Filter Encoding Implementation Specification 1.0.0. OGC Document #02-059, September 2001. Available at http://www.opengeospatial.org/specs/?page=specs
[43]. SERVOGrid WMS implementation. Available at http://complexity.ucs.indiana.edu/~asayar/wms
[43]. SERVOGrid WFS implementation. Available at http://www.crisisgrid.org/html/wfs.html
[44]. Galip Aydin, Marlon Pierce, Geoffrey Fox, “High Performance Web Feature Service Implementation for GIS Grid and Web Service

Architectures", Submitted to GML And Geo-Spatial Web Services Conference 2005.
[45]. Ahmet Sayar, Mehmet S. Aktas, Galip Aydin, Geoffrey Fox and Marlon Pierce, "Developing a Web Service-Compatible Map Server for Geophysical Applications", Submitted to ACM-GIS Conference 2005.
[46]. Mehmet Aktas, Galip Aydin, Andrea Donnellan, Geoffrey Fox, Robert Granat , Lisa Grant, Greg Lyzenga, Dennis McLeod, Shrideep Pallickara, Jay Parker, Marlon Pierce, John Rundle, Ahmet Sayar, and Terry Tullis, " iSERVO: Implementing the International Solid Earth Research Virtual Observatory by Integrating Computational Grid and Geographical Information Web Services", Technical Report December 2004, To be published in Special Issue of Pure and Applied Geophysics (PAGEOPH) for Beijing ACES Meeting July 2004.
[47]. Apache Axis Project Web Site : http://ws.apache.org/axis/.
[48]. Geoffrey Fox, Shrideep Pallickara and Savas Parastatidis Towards Flexible Messaging for SOAP Based Services Proceedings of the IEEE/ACM Supercomputing Conference November 2004. Pittsburgh, PA.
[49]. Geoffrey Fox, Shrideep Pallickara, Marlon Pierce, Harshawardhan Gadgil, Building Messaging Substrates for Web and Grid Applications to be published in special Issue on *Scientific Applications of Grid Computing* in Philosophical Transactions of the Royal Society of London 2005.
[49]. Message based middleware project at Community Grids Lab, Project Web Site: http://www.naradabrokering.org/
[50]. Global Multimedia Collaboration System. http://www.globalmmcs.org/
[51]. Wenjun Wu, Geoffrey Fox, Hasan Bulut, Ahmet Uyar, Harun Altay "Design and Implementation of A Collaboration Web-services

system", Journal of Neural, Parallel & Scientific Computations (NPSC), Volume 12, 2004.

[52]. The Access Grid Project. http://www.accessgrid.org

[53]. University of Minnesota web site and UMN MapServer of CGL laborotories. http://mapserver.gis.umn.edu/

[54]. Eclipse IDE for Java classes. http://www.eclipse.org/

[55]. Universal Description, Discovery and Integration web page. http://www.uddi.org/

[56]. W3C World Wide Web Consortuim web side. http://www.w3.org/

[57]. Ahmet Sayar's Home page in Minnesota University. http://complexity.ucs.indiana.edu/~asayar/gisgrids/index.html

[58]. For XML definitions and tutorials.http://www.xml.com/

[59]. http://www.w3schools.com/xml/ for XML tutorials.

[60]. http://www.mutasyon.net/makaleoku.asp?id=253 by Ali Mercan.

[61]. http://www-306.ibm.com/software/solutions/webservices/pdf/WSCA.pdf

[62]. http://www128.ibm.com/developerworks/websphere/library/techarticles/0307_ryman/ryman.html

[63]. http://www-128.ibm.com/developerworks/webservices/library/ws-featuddi/

EKLER

EK A

A.1 WMS (Web harita servisi) için WSDL dokümanı içeriği

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<definitions xmlns:SOAPENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://schemas.xmlsoap.org/wsdl/">
<types>
<xsd:schema targetNamespace="http://schemas.xmlsoap.org/wsdl/">
<xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
<xsd:import namespace="http://schemas.xmlsoap.org/wsdl/" />
</xsd:schema>
</types>
<message name="GetCapabilitiesRequest">
  <part name="request" type="xsd:string" /></message>
<message name="GetCapabilitiesResponse">
  <part name="return" type="xsd:string" /></message>
<message name="GetMapRequest">
  <part name="request" type="xsd:string" /></message>
<message name="GetMapResponse">
  <part name="return" type="xsd:string" /></message>
<message name="GetFeatureInfoRequest">
  <part name="request" type="xsd:string" /></message>
<message name="GetFeatureInfoResponse">
  <part name="return" type="xsd:string" /></message>
<portType name="OGCWebServicePortType">
  <operation name="GetCapabilities">
    <input message="tns:GetCapabilitiesRequest"/>
    <output message="tns:GetCapabilitiesResponse"/>
  </operation>
  <operation name="GetMap">
    <input message="tns:GetMapRequest"/>
    <output message="tns:GetMapResponse"/>
  </operation>
  <operation name="GetFeatureInfo">
```

```
    <input message="tns:GetFeatureInfoRequest"/>
    <output message="tns:GetFeatureInfoResponse"/>
  </operation>
</portType>
<binding name="OGCWebServiceBinding" type="tns:OGCWebServicePortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="GetCapabilities">
<soap:operation
soapAction="http://localhost/wmsclient/testserver.php/GetCapabilities"
style="rpc"/>
<input><soap:body use="encoded" namespace="http://schemas.xmlsoap.org/wsdl/"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input><output><so
ap:body use="encoded" namespace="http://schemas.xmlsoap.org/wsdl/"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="GetMap">
    <soap:operation soapAction="http://localhost/wmsclient/testserver.php/GetMap"
style="rpc"/>
<input><soap:body use="encoded" namespace="http://schemas.xmlsoap.org/wsdl/"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input><output><so
ap:body use="encoded" namespace="http://schemas.xmlsoap.org/wsdl/"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output>
  </operation>
  <operation name="GetFeatureInfo">
    <soap:operation
soapAction="http://localhost/wmsclient/testserver.php/GetFeatureInfo" style="rpc"/>
<input><soap:body use="encoded" namespace="http://schemas.xmlsoap.org/wsdl/"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></input><output><so
ap:body use="encoded" namespace="http://schemas.xmlsoap.org/wsdl/"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/></output></operation
>
</binding>
<service name="OGCWebService">
  <port name="OGCWebServicePort" binding="tns:OGCWebServiceBinding">
    <soap:address location="http://localhost/wmsclient/testserver.php"/>
  </port>
</service>
</definitions>
```

A.2 WMS (Web harita servisi) için istek XML biçimi

GetCapabilities Request XML

```
<GetCapabilities></GetCapabilities>
```

GetFeatureInfo Request XML

```
<GetFeatureInfo>
        <URL></URL>
        <BBOX></BBOX>
        <SRS></SRS>
        <X></X>
        <Y></Y>
        <LAYERS></LAYERS>
        <QUERYLAYERS></QUERYLAYERS>
        <WIDTH></WIDTH>
        <HEIGHT></HEIGHT>
        <INFO_FORMAT> </INFO_FORMAT>
</GetFeatureInfo>
```

GetMap Request XML

```
<GetMap>
        <URL></URL>
        <BBOX></BBOX>
        <LAYERS></LAYERS>
        <SRS></SRS>
        <WIDTH></WIDTH>
        <HEIGHT></HEIGHT>
        <FORMAT> </FORMAT></GetMap>
```

A.3 WMS (Web harita servisi) için yanıt XML biçimi

UMN harita sunucusu CBS uygulamasından, GetCapabilities Cevap XML'i:

```
<?xml version='1.0' encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE WMT_MS_Capabilities SYSTEM
"http://schemas.opengeospatial.net/wms/1.1.1/WMS_MS_Capabilities.dtd"
 [
 <!ELEMENT VendorSpecificCapabilities EMPTY>
 ]>  <!-- end of DOCTYPE declaration -->

<WMT_MS_Capabilities version="1.1.1">

<!-- MapServer version 4.8.3 OUTPUT=GIF OUTPUT=PNG OUTPUT=JPEG
OUTPUT=WBMP OUTPUT=PDF OUTPUT=SWF OUTPUT=SVG
SUPPORTS=PROJ SUPPORTS=FREETYPE SUPPORTS=WMS_SERVER
SUPPORTS=WMS_CLIENT SUPPORTS=WFS_SERVER
SUPPORTS=WFS_CLIENT SUPPORTS=WCS_SERVER
SUPPORTS=THREADS SUPPORTS=GEOS INPUT=JPEG INPUT=POSTGIS
INPUT=OGR INPUT=GDAL INPUT=SHAPEFILE DEBUG=MSDEBUG -->

<Service>
  <Name>OGC:WMS</Name>
<!-- WARNING: Mandatory metadata '..._title' was missing in this context. -->
  <Title>DEMO</Title>
  <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="http://localhost/cgi-bin/ogcdemo?"/>
</Service>

<Capability>
  <Request>
    <GetCapabilities>
      <Format>application/vnd.ogc.wms_xml</Format>
      <DCPType>
        <HTTP>
          <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="http://localhost/cgi-bin/ogcdemo?"/></Get>
          <Post><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="http://localhost/cgi-bin/ogcdemo?"/></Post>
        </HTTP>
      </DCPType>
    </GetCapabilities>
    <GetMap>
      <Format>image/png</Format>
      <Format>image/gif</Format>
      <Format>image/png; mode=24bit</Format>
      <Format>image/jpeg</Format>
```

```
      <Format>image/wbmp</Format>
      <Format>image/tiff</Format>
      <DCPType>
        <HTTP>
          <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="http://localhost/cgi-bin/ogcdemo?"/></Get>
          <Post><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="http://localhost/cgi-bin/ogcdemo?"/></Post>
        </HTTP>
      </DCPType>
    </GetMap>
    <GetFeatureInfo>
      <Format>text/plain</Format>
      <Format>application/vnd.ogc.gml</Format>
      <DCPType>
        <HTTP>
          <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="http://localhost/cgi-bin/ogcdemo?"/></Get>
          <Post><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="http://localhost/cgi-bin/ogcdemo?"/></Post>
        </HTTP>
      </DCPType>
    </GetFeatureInfo>
    <DescribeLayer>
      <Format>text/xml</Format>
      <DCPType>
        <HTTP>
          <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="http://localhost/cgi-bin/ogcdemo?"/></Get>
          <Post><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="http://localhost/cgi-bin/ogcdemo?"/></Post>
        </HTTP>
      </DCPType>
    </DescribeLayer>
    <GetLegendGraphic>
      <Format>image/png</Format>
      <Format>image/gif</Format>
      <Format>image/png; mode=24bit</Format>
      <Format>image/jpeg</Format>
      <Format>image/wbmp</Format>
      <DCPType>
        <HTTP>
          <Get><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="http://localhost/cgi-bin/ogcdemo?"/></Get>
          <Post><OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="http://localhost/cgi-bin/ogcdemo?"/></Post>
        </HTTP>
```

```
    </DCPType>
   </GetLegendGraphic>
  </Request>
  <Exception>
   <Format>application/vnd.ogc.se_xml</Format>
   <Format>application/vnd.ogc.se_inimage</Format>
   <Format>application/vnd.ogc.se_blank</Format>
  </Exception>
  <VendorSpecificCapabilities />
  <UserDefinedSymbolization SupportSLD="1" UserLayer="0" UserStyle="1"
RemoteWFS="0"/>
  <Layer>
   <Name>DEMO</Name>
<!-- WARNING: Mandatory metadata '..._title' was missing in this context. -->
   <Title>DEMO</Title>
   <SRS>EPSG:42304</SRS>
   <LatLonBoundingBox minx="-172.367" miny="35.6673" maxx="-11.5624"
maxy="83.8293" />
   <BoundingBox SRS="EPSG:42304"
            minx="-2.2e+006" miny="-712631" maxx="3.0728e+006"
maxy="3.84e+006" />
   <ScaleHint min="997.805696859274" max="24945.1424214819" />
   <Layer queryable="0" opaque="0" cascaded="0">
    <Name>bathymetry</Name>
<!-- WARNING: Mandatory metadata '..._title' was missing in this context. -->
    <Title>bathymetry</Title>
<!-- WARNING: Mandatory mapfile parameter '(at least one of)
MAP.PROJECTION, LAYER.PROJECTION or wms_srs metadata' was missing in
this context. -->
   </Layer>
   <Layer queryable="0" opaque="0" cascaded="0">
    <Name>land_fn</Name>
<!-- WARNING: Mandatory metadata '..._title' was missing in this context. -->
    <Title>land_fn</Title>
<!-- WARNING: Mandatory mapfile parameter '(at least one of)
MAP.PROJECTION, LAYER.PROJECTION or wms_srs metadata' was missing in
this context. -->
    <LatLonBoundingBox minx="-178.838" miny="31.8844" maxx="179.94"
maxy="89.8254" />
    <BoundingBox SRS="EPSG:42304"
             minx="-2.75056e+006" miny="-936639" maxx="3.58387e+006"
maxy="4.67313e+006" />
    <Style>
     <Name>default</Name>
     <Title>default</Title>
     <LegendURL width="18" height="12">
      <Format>image/png</Format>
```

```
          <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://localhost/cgi-
bin/ogcdemo?version=1.1.1&amp;service=WMS&amp;request=GetLegendGraphic
&amp;layer=land_fn&amp;format=image/png"/>
        </LegendURL>
      </Style>
    </Layer>
    <Layer queryable="0" opaque="0" cascaded="0">
      <Name>drain_fn</Name>
<!-- WARNING: Mandatory metadata '..._title' was missing in this context. -->
      <Title>drain_fn</Title>
<!-- WARNING: Mandatory mapfile parameter '(at least one of)
MAP.PROJECTION, LAYER.PROJECTION or wms_srs metadata' was missing in
this context. -->
      <LatLonBoundingBox minx="-179.973" miny="35.2464" maxx="179.92"
maxy="88.06" />
      <BoundingBox SRS="EPSG:42304"
            minx="-2.75056e+006" miny="-936639" maxx="2.75882e+006"
maxy="4.36727e+006" />
      <Style>
        <Name>default</Name>
        <Title>default</Title>
        <LegendURL width="18" height="12">
          <Format>image/png</Format>
          <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://localhost/cgi-
bin/ogcdemo?version=1.1.1&amp;service=WMS&amp;request=GetLegendGraphic
&amp;layer=drain_fn&amp;format=image/png"/>
        </LegendURL>
      </Style>
    </Layer>
    <Layer queryable="0" opaque="0" cascaded="0">
      <Name>drainage</Name>
<!-- WARNING: Mandatory metadata '..._title' was missing in this context. -->
      <Title>drainage</Title>
<!-- WARNING: Mandatory mapfile parameter '(at least one of)
MAP.PROJECTION, LAYER.PROJECTION or wms_srs metadata' was missing in
this context. -->
      <LatLonBoundingBox minx="-169.629" miny="39.2232" maxx="-15.1085"
maxy="83.0129" />
      <BoundingBox SRS="EPSG:42304"
            minx="-2.1694e+006" miny="-386968" maxx="2.79747e+006"
maxy="3.74336e+006" />
      <Style>
        <Name>default</Name>
        <Title>default</Title>
        <LegendURL width="18" height="12">
```

```
          <Format>image/png</Format>
          <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://localhost/cgi-
bin/ogcdemo?version=1.1.1&amp;service=WMS&amp;request=GetLegendGraphic
&amp;layer=drainage&amp;format=image/png"/>
        </LegendURL>
      </Style>
    </Layer>
    <Layer queryable="0" opaque="0" cascaded="0">
      <Name>prov_bound</Name>
<!-- WARNING: Mandatory metadata '..._title' was missing in this context. -->
      <Title>prov_bound</Title>
<!-- WARNING: Mandatory mapfile parameter '(at least one of)
MAP.PROJECTION, LAYER.PROJECTION or wms_srs metadata' was missing in
this context. -->
      <LatLonBoundingBox minx="-173.537" miny="35.8775" maxx="-11.9603"
maxy="83.8009" />
      <BoundingBox SRS="EPSG:42304"
            minx="-2.3406e+006" miny="-719746" maxx="3.00943e+006"
maxy="3.83661e+006" />
      <Style>
        <Name>default</Name>
        <Title>default</Title>
        <LegendURL width="18" height="12">
          <Format>image/png</Format>
          <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://localhost/cgi-
bin/ogcdemo?version=1.1.1&amp;service=WMS&amp;request=GetLegendGraphic
&amp;layer=prov_bound&amp;format=image/png"/>
        </LegendURL>
      </Style>
    </Layer>
    <Layer queryable="0" opaque="0" cascaded="0">
      <Name>fedlimit</Name>
<!-- WARNING: Mandatory metadata '..._title' was missing in this context. -->
      <Title>fedlimit</Title>
<!-- WARNING: Mandatory mapfile parameter '(at least one of)
MAP.PROJECTION, LAYER.PROJECTION or wms_srs metadata' was missing in
this context. -->
      <LatLonBoundingBox minx="-179.96" miny="34.2409" maxx="178.833"
maxy="89.9051" />
      <BoundingBox SRS="EPSG:42304"
            minx="-2.69358e+006" miny="-724162" maxx="3.38519e+006"
maxy="4.6545e+006" />
      <Style>
        <Name>default</Name>
        <Title>default</Title>
```

```
        <LegendURL width="18" height="12">
          <Format>image/png</Format>
          <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://localhost/cgi-
bin/ogcdemo?version=1.1.1&amp;service=WMS&amp;request=GetLegendGraphic
&amp;layer=fedlimit&amp;format=image/png"/>
        </LegendURL>
      </Style>
    </Layer>
    <Layer queryable="0" opaque="0" cascaded="0">
      <Name>rail</Name>
<!-- WARNING: Mandatory metadata '..._title' was missing in this context. -->
      <Title>rail</Title>
<!-- WARNING: Mandatory mapfile parameter '(at least one of)
MAP.PROJECTION, LAYER.PROJECTION or wms_srs metadata' was missing in
this context. -->
      <LatLonBoundingBox minx="-137.447" miny="37.7146" maxx="-46.3201"
maxy="66.7201" />
      <BoundingBox SRS="EPSG:42304"
                minx="-2.14572e+006" miny="-680853" maxx="2.61606e+006"
maxy="1.93097e+006" />
      <Style>
        <Name>default</Name>
        <Title>default</Title>
        <LegendURL width="18" height="12">
          <Format>image/png</Format>
          <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://localhost/cgi-
bin/ogcdemo?version=1.1.1&amp;service=WMS&amp;request=GetLegendGraphic
&amp;layer=rail&amp;format=image/png"/>
        </LegendURL>
      </Style>
    </Layer>
    <Layer queryable="1" opaque="0" cascaded="0">
      <Name>road</Name>
<!-- WARNING: Mandatory metadata '..._title' was missing in this context. -->
      <Title>road</Title>
<!-- WARNING: Mandatory mapfile parameter '(at least one of)
MAP.PROJECTION, LAYER.PROJECTION or wms_srs metadata' was missing in
this context. -->
      <LatLonBoundingBox minx="-148.059" miny="35.882" maxx="-33.7745"
maxy="72.5503" />
      <BoundingBox SRS="EPSG:42304"
                minx="-2.30369e+006" miny="-724118" maxx="3.00053e+006"
maxy="2.56497e+006" />
      <Style>
        <Name>default</Name>
```

```
        <Title>default</Title>
        <LegendURL width="18" height="12">
          <Format>image/png</Format>
          <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://localhost/cgi-
bin/ogcdemo?version=1.1.1&amp;service=WMS&amp;request=GetLegendGraphic
&amp;layer=road&amp;format=image/png"/>
        </LegendURL>
      </Style>
    </Layer>
    <Layer queryable="1" opaque="0" cascaded="0">
      <Name>popplace</Name>
<!-- WARNING: Mandatory metadata '..._title' was missing in this context. -->
      <Title>popplace</Title>
<!-- WARNING: Mandatory mapfile parameter '(at least one of)
MAP.PROJECTION, LAYER.PROJECTION or wms_srs metadata' was missing in
this context. -->
      <LatLonBoundingBox minx="-172.301" miny="36.3541" maxx="-12.9698"
maxy="83.4832" />
      <BoundingBox SRS="EPSG:42304"
            minx="-2.30386e+006" miny="-681503" maxx="2.96177e+006"
maxy="3.79886e+006" />
      <Style>
        <Name>default</Name>
        <Title>default</Title>
        <LegendURL width="18" height="12">
          <Format>image/png</Format>
          <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://localhost/cgi-
bin/ogcdemo?version=1.1.1&amp;service=WMS&amp;request=GetLegendGraphic
&amp;layer=popplace&amp;format=image/png"/>
        </LegendURL>
      </Style>
    </Layer>
    <Layer queryable="0" opaque="0" cascaded="0">
      <Name>grid</Name>
<!-- WARNING: Mandatory metadata '..._title' was missing in this context. -->
      <Title>grid</Title>
<!-- WARNING: Mandatory mapfile parameter '(at least one of)
MAP.PROJECTION, LAYER.PROJECTION or wms_srs metadata' was missing in
this context. -->
      <LatLonBoundingBox minx="-178.838" miny="31.8844" maxx="179.94"
maxy="89.8254" />
      <BoundingBox SRS="EPSG:42304"
            minx="-2.75056e+006" miny="-936639" maxx="3.58387e+006"
maxy="4.67313e+006" />
      <Style>
```

```
        <Name>default</Name>
        <Title>default</Title>
        <LegendURL width="18" height="12">
          <Format>image/png</Format>
          <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="http://localhost/cgi-
bin/ogcdemo?version=1.1.1&amp;amp;service=WMS&amp;amp;request=GetLegendGraphic
&amp;amp;layer=grid&amp;amp;format=image/png"/>
        </LegendURL>
      </Style>
    </Layer>
  </Layer>
</Capability></WMT_MS_Capabilities>
```

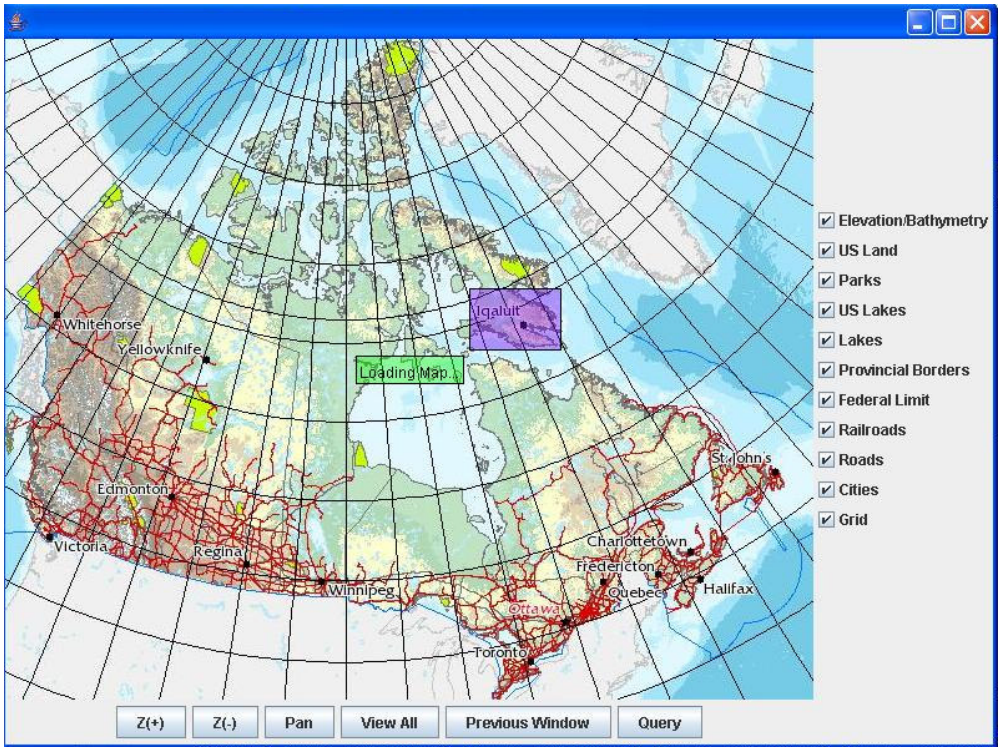
A.4 UMN MapServer (Harita Sunucusu) GIS (CBS) aracı üzerinden yayınlanan GML istek formatı

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<msGMLOutput
        xmlns:gml="http://www.opengis.net/gml"
        xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <popplace_layer>
                <popplace_feature>
                        <gml:boundedBy>
                                <gml:Box srsName="EPSG:42304">
                                        <gml:coordinates>-
1991310.375000,1937023.000000 -
1991310.375000,1937023.000000</gml:coordinates>
                                </gml:Box>
                        </gml:boundedBy>
                        <AREA>0.000</AREA>
                        <PERIMETER>0.000</PERIMETER>
                        <POPPLACE_>467</POPPLACE_>
                        <POPPLACE_I>96</POPPLACE_I>
                        <UNIQUE_KEY>KAHFT</UNIQUE_KEY>
                        <NAME>Whitehorse</NAME>
                        <NAME_E></NAME_E>
                        <NAME_F></NAME_F>
                        <UNIQUE_KEY>KAHFT</UNIQUE_KEY>
                        <UNIQUE_KEY>KAHFT</UNIQUE_KEY>
                        <REG_CODE>60</REG_CODE>
                        <NTS50>105D11</NTS50>
                        <LAT>604300</LAT>
                        <LONG>1350300</LONG>
                        <SGC_CODE>6001009</SGC_CODE>
                        <CAPITAL>3</CAPITAL>
                        <POP_RANGE>3</POP_RANGE>
                </popplace_feature>
        </popplace_layer>
</msGMLOutput>
```

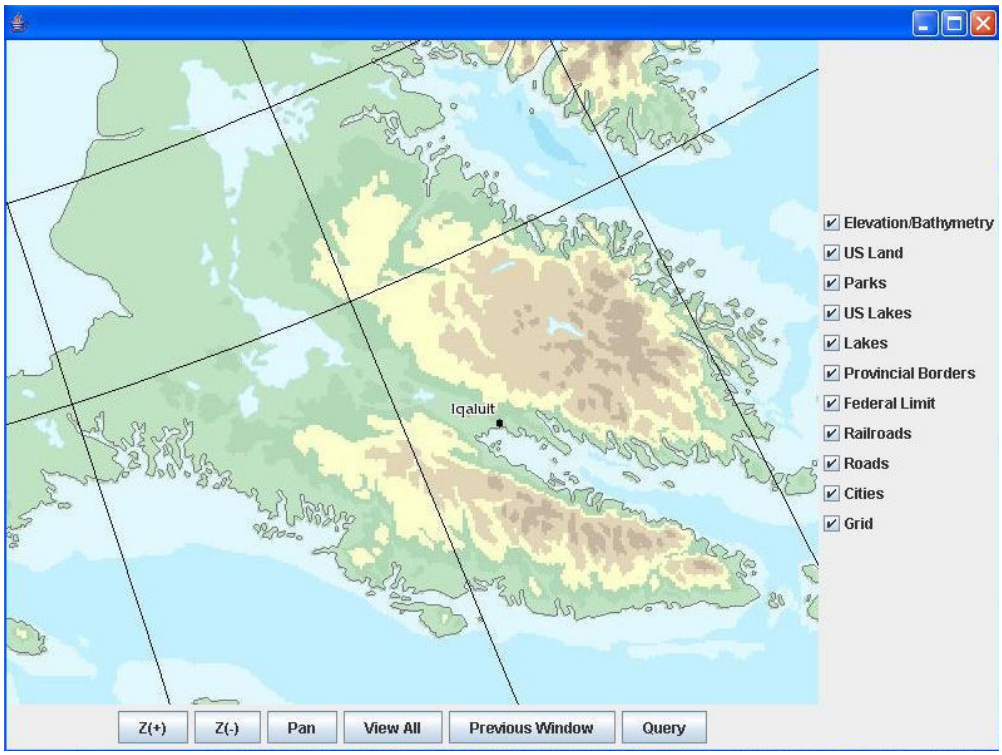
EK B

Java Swing görüntü editörü ile tasarlanmış ekran üzerinden gösterilen harita imajları Kanada'ya ait harita koordinatları üzerinden servis görüntülemeleri yapılmıştır.

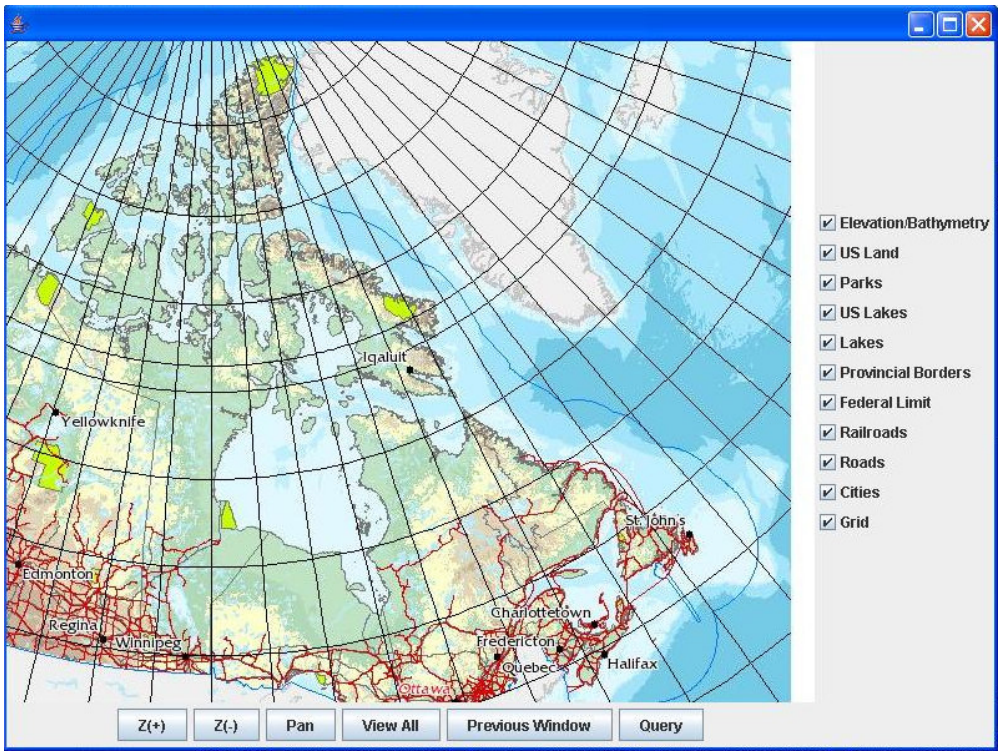
B.1 ZoomIn(+) metodu kullanılmadan önceki ekran görüntüsü:



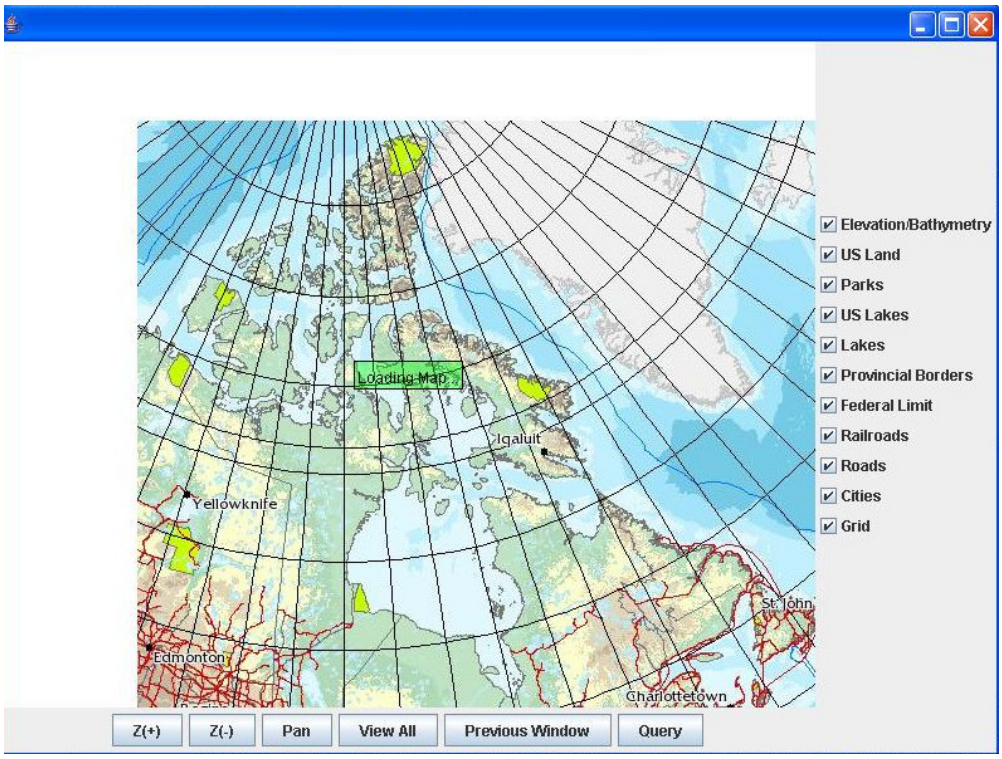
B.2 ZoomIn(+) metodu kullanıldıktan sonra ekran görüntüsü:



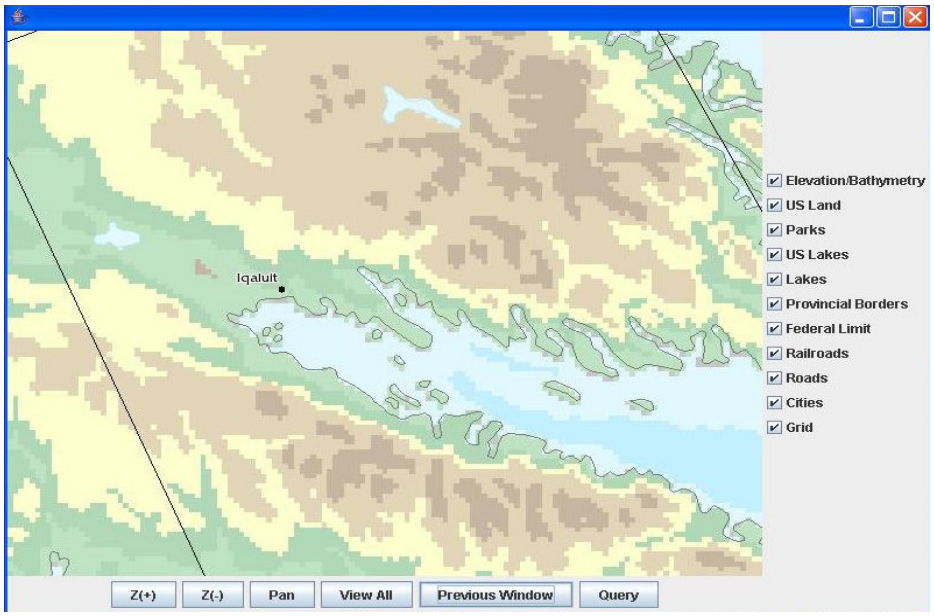
B.3 ZoomOut (-) metodu kullanılınca ekran görüntüsü:



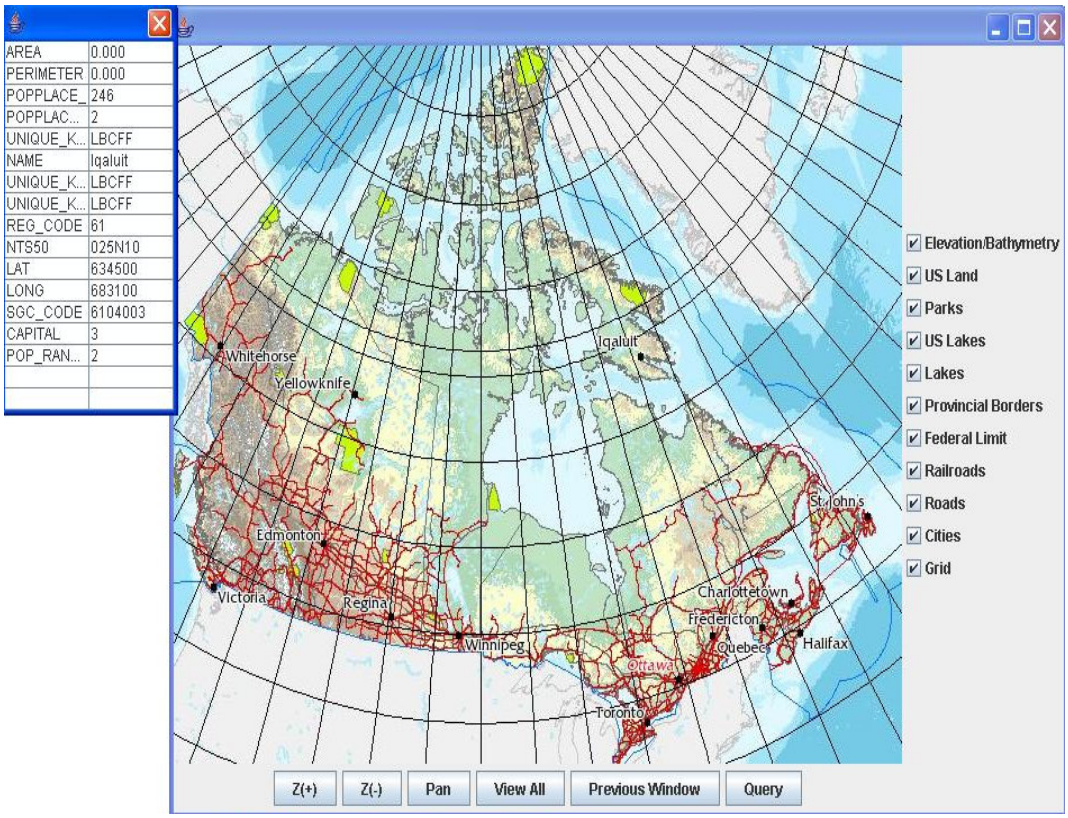
B.4 Pan metodu kullanılınca ekran görüntüsü:



B.5 Önceki Pencere (PreviousWindow) metodu kullanılınca ekran görüntüsü



B.6 Sogu (Query) metodu kullanılınca ekran görüntüsü:



EK C

C.1 Sunucu tarafında yer alan WMS servisine ait prosedürlerin, açık kaynak kod ürünü olan NuSOAP ile entegrasyonunu sağlayan kaynak kodu aşağıda belirtilmektedir. PHP ile geliştirilmiş web servis fonksiyonlarına ait metot scriptleri.

```
<?
dl("php_domxml.dll");

function parseGetInfoRequestXML($dom){

/*
"<GetFeatureInfo>
<URL></URL>
<BBOX>=-2200000%2C-
321678.171906%2C3072800%2C3644523.58595</BBOX>
<SRS>EPSG:42304</SRS>
<X></X>
<Y></Y>
<LAYERS></LAYERS>
<QUERYLAYERS></QUERYLAYERS>
<WIDTH></WIDTH>
<HEIGHT></HEIGHT>
<FORMAT></FORMAT>
</GetFeatureInfo>";
*/

  $xp = "/GetFeatureInfo/URL";
  $ctx = xpath_new_context($dom);
  $xpnode = xpath_eval($ctx,$xp);
  $dtnode = $xpnode->nodeset[0]->first_child();
  $url = $dtnode->content;

//  echo $url;

  $xp = "/GetFeatureInfo/BBOX";
  //$ctx = xpath_new_context($dom);
  $xpnode = xpath_eval($ctx,$xp);
  $dtnode = $xpnode->nodeset[0]->first_child();
  $bbox = $dtnode->content;

  $xp = "/GetFeatureInfo/SRS";
  //$ctx = xpath_new_context($dom);
  $xpnode = xpath_eval($ctx,$xp);
  $dtnode = $xpnode->nodeset[0]->first_child();
```

```
$srs = $dtnode->content;

$xp = "/GetFeatureInfo/X";
//$ctx = xpath_new_context($dom);
$xpnode = xpath_eval($ctx,$xp);
$dtnode = $xpnode->nodeset[0]->first_child();
$x = $dtnode->content;

$xp = "/GetFeatureInfo/Y";
//$ctx = xpath_new_context($dom);
$xpnode = xpath_eval($ctx,$xp);
$dtnode = $xpnode->nodeset[0]->first_child();
$y = $dtnode->content;


$xp = "/GetFeatureInfo/LAYERS";
//$ctx = xpath_new_context($dom);
$xpnode = xpath_eval($ctx,$xp);
$dtnode = $xpnode->nodeset[0]->first_child();
$layers = $dtnode->content;

$xp = "/GetFeatureInfo/QUERYLAYERS";
//$ctx = xpath_new_context($dom);
$xpnode = xpath_eval($ctx,$xp);
$dtnode = $xpnode->nodeset[0]->first_child();
$qlayers = $dtnode->content;

$xp = "/GetFeatureInfo/WIDTH";
//$ctx = xpath_new_context($dom);
$xpnode = xpath_eval($ctx,$xp);
$dtnode = $xpnode->nodeset[0]->first_child();
$width = $dtnode->content;

$xp = "/GetFeatureInfo/HEIGHT";
//$ctx = xpath_new_context($dom);
$xpnode = xpath_eval($ctx,$xp);
$dtnode = $xpnode->nodeset[0]->first_child();
$height = $dtnode->content;

$xp = "/GetFeatureInfo/INFO_FORMAT";
//$ctx = xpath_new_context($dom);
$xpnode = xpath_eval($ctx,$xp);
$dtnode = $xpnode->nodeset[0]->first_child();
$format = $dtnode->content;

/*
echo  $url."VERSION=" . urlencode("1.1.0") . "&" .
```

```
            "REQUEST=GetMap&" .
                "BBOX=" . urlencode($bbox) . "&" .
                            "LAYERS=" . urlencode($layers) . "&" .
                            "SRS=" . urlencode($"srs") . "&" .
                        "WIDTH=" . urlencode($width) . "&" .
                "HEIGHT=" . urlencode($height) . "&" .
                "FORMAT=" . urlencode($format);
                            */

  $url = urldecode($url) . "VERSION=" . urlencode("1.1.1") . "&" .
        "Request=GetFeatureInfo&" .
        "BBOX=" . urlencode($bbox) . "&" .
        "SRS=" . urlencode($srs) . "&" .
                  "X=" . $x . "&" .
                  "Y=" . $y . "&" .
        "LAYERS=" . urlencode($layers) . "&" .
        "QUERY_LAYERS=" . urlencode($qlayers) . "&" .
        "WIDTH=" . urlencode($width) . "&" .
        "HEIGHT=" . urlencode($height) . "&" .
        "INFO_FORMAT=" . urlencode($format);


 return $url;
}

function parseGetMapRequestXML($dom){

/*<GetMap>
   <BBOX>111111 222222 333333 444444</BBOX>
   <LAYERS>Ovecler</LAYERS>
   <SRS>EPSG:5432</SRS>
   <WIDTH>320</WIDTH>
   <HEIGHT>240</HEIGHT>
   <FORMAT>jpeg</FORMAT>
 </GetMap>";
*/

 $xp = "/GetMap/URL";
 $ctx = xpath_new_context($dom);
 $xpnode = xpath_eval($ctx,$xp);
 $dtnode = $xpnode->nodeset[0]->first_child();
 $url = $dtnode->content;
 $url = urldecode($url);

//  echo $url;

 $xp = "/GetMap/BBOX";
```

```
//$ctx = xpath_new_context($dom);
$xpnode = xpath_eval($ctx,$xp);
$dtnode = $xpnode->nodeset[0]->first_child();
$bbox = $dtnode->content;

$xp = "/GetMap/LAYERS";
//$ctx = xpath_new_context($dom);
$xpnode = xpath_eval($ctx,$xp);
$dtnode = $xpnode->nodeset[0]->first_child();
$layers = $dtnode->content;

$xp = "/GetMap/SRS";
//$ctx = xpath_new_context($dom);
$xpnode = xpath_eval($ctx,$xp);
$dtnode = $xpnode->nodeset[0]->first_child();
$srs = $dtnode->content;

$xp = "/GetMap/WIDTH";
//$ctx = xpath_new_context($dom);
$xpnode = xpath_eval($ctx,$xp);
$dtnode = $xpnode->nodeset[0]->first_child();
$width = $dtnode->content;

$xp = "/GetMap/HEIGHT";
//$ctx = xpath_new_context($dom);
$xpnode = xpath_eval($ctx,$xp);
$dtnode = $xpnode->nodeset[0]->first_child();
$height = $dtnode->content;

$xp = "/GetMap/FORMAT";
//$ctx = xpath_new_context($dom);
$xpnode = xpath_eval($ctx,$xp);
$dtnode = $xpnode->nodeset[0]->first_child();
$format = $dtnode->content;

/*
echo  $url."VERSION=" . urlencode("1.1.0") . "&" .
          "REQUEST=GetMap&" .
               "BBOX=" . urlencode($bbox) . "&" .
                         "LAYERS=" . urlencode($layers) . "&" .
                         "SRS=" . urlencode($"srs") . "&" .
                    "WIDTH=" . urlencode($width) . "&" .
               "HEIGHT=" . urlencode($height) . "&" .
               "FORMAT=" . urlencode($format);
                         */

$url   = $url."VERSION=" . urlencode("1.0.0") . "&" ;
```

```
  $subdir =   "REQUEST=GetMap&" . "BBOX=" . $bbox . "&" .
                          "LAYERS=" . $layers . "&".
                          "SRS=" . urlencode($srs) . "&" .
                      "WIDTH=" . urlencode($width) . "&" .
                  "HEIGHT=" . urlencode($height) . "&" .
                  "FORMAT=" . urlencode($format);

  $result =
"<GetMap><mainurl>".urlencode($url)."</mainurl><subdir>".urlencode($subdir)."<
/subdir></GetMap>";

  //$result = "<GetMap>".urlencode($url).urlencode($subdir)</GetMap>";

  return $result;
}

function xml2html ( $str ) {
  #
  # Convert XML characters to HTML entities
  # for display purposes.
  #
  $str = ereg_replace("&","&amp;",$str);
  $str = ereg_replace("<","&lt;",$str);
  $str = ereg_replace(">","&gt;<BR>",$str);
  return $str;
}

function wms_title ( $dom ) {
  #
  # Read the WMS service title and return it as text.
  #
  $xp = "/WMT_MS_Capabilities/Service/Title";
  $ctx = xpath_new_context($dom);
  $xpnode = xpath_eval($ctx,$xp);
  $dtnode = $xpnode->nodeset[0]->first_child();
  return $dtnode->content;
}

function wms_onlineresource ( $dom ) {
  #
  # Read the WMS online resource URL and return it as text.
  #
  $xp = "/WMT_MS_Capabilities/Service/OnlineResource";
  $ctx = xpath_new_context($dom);
  $xpnode = xpath_eval($ctx,$xp);
  return $xpnode->nodeset[0]->get_attribute("href");
}
```

```
function wms_formats ( $dom ) {
 #
 # Read the WMS image formats and return them as an array.
 #
 $xp = "/WMT_MS_Capabilities/Capability/Request/GetMap/Format";
 $ctx = xpath_new_context($dom);
 $xpnode = xpath_eval($ctx,$xp);
 $arr = array();
 for( $i = 0; $i < sizeof($xpnode->nodeset); $i++ ) {
  $dtnode = $xpnode->nodeset[0]->first_child();
  array_push($arr,$dtnode->content);
 }
 return $arr;
}

function wms_exceptions ( $dom ) {
 #
 # Read the WMS exception formats and return them as an array.
 #
 $xp = "/WMT_MS_Capabilities/Capability/Exception/Format";
 $ctx = xpath_new_context($dom);
 $xpnode = xpath_eval($ctx,$xp);
 $arr = array();
 for( $i = 0; $i < sizeof($xpnode->nodeset); $i++ ) {
  $dtnode = $xpnode->nodeset[0]->first_child();
  array_push($arr,$dtnode->content);
 }
 return $arr;
}

function wms_layers ( $dom ) {
 #
 # Read the WMS first level layers and return an
 # array of nodes.
 #
 $xp = "/WMT_MS_Capabilities/Capability/Layer/Layer";
 $ctx = xpath_new_context($dom);
 $xpnode = xpath_eval($ctx,$xp);
 return $xpnode->nodeset;
}

function wms_xpnode2content( $xp_node ) {
 #
 # Read the content child node of an element tag
 # node.
 #
```

```
  $content = "";
  if( $xp_node->nodeset[0] ) {
    $node = $xp_node->nodeset[0]->first_child();
    $content = $node->content;
  }
  return $content;
}

function wms_srs( $dom ) {
  #
  # Read the SRS from a WMS capabilities.
  #
  $ctx = xpath_new_context($dom);
  $xp = "/WMT_MS_Capabilities/Capability/Layer/BoundingBox";
  $xpnode = xpath_eval($ctx,$xp);
  if( $xpnode->nodeset[0] ) {
    #
    # If there is a BoundingBox, we are using it for default
    # extent, so get that SRS.
    #
    $node = $xpnode->nodeset[0];
    $srs = $node->get_attribute("SRS");
  }
  else {
    #
    # Otherwise we are using WMS standard Lat/Lon for
    # default extent so use WMS standard Lat/Lon.
    #
    $srs = "EPSG:4326";
  }
  return strtoupper($srs);
}

function wms_bbox( $dom ) {
  #
  # Read the default extent from a WMS capabilities.
  #
  $ctx = xpath_new_context($dom);
  $xp = "/WMT_MS_Capabilities/Capability/Layer/BoundingBox";
  $xpnode = xpath_eval($ctx,$xp);
  if( ! $xpnode->nodeset[0] ) {
    #
    # No node for planar BoundingBox, so we will get the
    # LatLonBoundingBox instead. There is parallel behavior in
    # wms_srs()
    #
    $xp = "/WMT_MS_Capabilities/Capability/Layer/LatLonBoundingBox";
```

```
    $xpnode = xpath_eval($ctx,$xp);
  }
  $node = $xpnode->nodeset[0];
  return wms_bbox2txt($node);
}

function wms_bbox2txt( $node ) {
  #
  # Convert a BoundingBox node into a text string.
  #
  if( $node ) {
    $txt .= 1 * $node->get_attribute("minx");
    $txt .= ",";
    $txt .= 1 * $node->get_attribute("miny");
    $txt .= ",";
    $txt .= 1 * $node->get_attribute("maxx");
    $txt .= ",";
    $txt .= 1 * $node->get_attribute("maxy");
  }
  else {
    $txt = "-180,-90,180,90";
  }
  return $txt;
}

function wms_layer2html( $node ) {
  #
  # Convert a Layer node into an HTML representation.
  #
  $ctx = xpath_new_context($node);

  $xp_title  = xpath_eval($ctx,"/Title");
  $xp_name   = xpath_eval($ctx,"/Name");
  $xp_srs    = xpath_eval($ctx,"/SRS");
  $xp_llbbox = xpath_eval($ctx,"/LatLonBoundingBox");
  $xp_bbox   = xpath_eval($ctx,"/BoundingBox");

  $txt_title = wms_xpnode2content($xp_title);
  $txt_name  = wms_xpnode2content($xp_name);
  $txt_srs   = strtoupper(wms_xpnode2content($xp_srs));

  $node_llbbox = $xp_llbbox->nodeset[0];
  $node_bbox   = $xp_bbox->nodeset[0];

  $queryable = 0;
  if ( $node->get_attribute("queryable") ) {
    $queryable = 1;
```

```
  }
  $opaque = 0;
  if ( $node->get_attribute("opaque") ) {
    $opaque = 1;
  }

  $html  = "<INPUT TYPE='checkbox' NAME='checks' VALUE='$txt_name'
onClick='toggle(event)'>";
  $html .= "  ";
  $html .= $txt_title . "\n";
  $html .= wms_hidden("bbox_$txt_name", wms_bbox2txt($node_bbox));
  $html .= wms_hidden("llbox_$txt_name", wms_bbox2txt($node_llbbox));
  $html .= wms_hidden("srs_$txt_name", $txt_srs);
  $html .= wms_hidden("query_$txt_name", $queryable );
  $html .= wms_hidden("opaque_$txt_name", $opaque );
  $html .= "<BR>";

  return $html;

}

function wms_fatal ( $err ) {
  print "<P>$err</P>";
  exit;
}

function wms_hidden ( $name, $value ) {
  return "<INPUT TYPE='hidden' NAME='$name' VALUE='$value'>\n";
}

function wms_pan ( $bbox, $size, $click ) {
  #
  # Calculate a new extent based on an image size and image
  # click coordinates. Move extent linearly so that the
  # click becomes the new spatial center.
  #
  $sw = $bbox[2] - $bbox[0];
  $sh = $bbox[3] - $bbox[1];
  $iw = $size[0];
  $ih = $size[1];
  $x = $click[0];
  $y = $ih - $click[1] - 1;
  $mx = $iw / 2;
  $my = $ih / 2;

  $bbox[0] = $bbox[0] - $sw * ($mx - $x) / $iw;
  $bbox[1] = $bbox[1] - $sh * ($my - $y) / $ih;
```

```
  $bbox[2] = $bbox[2] - $sw * ($mx - $x) / $iw;
  $bbox[3] = $bbox[3] - $sh * ($my - $y) / $ih;

  return $bbox;
}

function wms_zoom ( $bbox, $factor ) {
  #
  # Create a new resized extent based on a spatial extent and
  # zoom factor.
  #
  $sw = $bbox[2] - $bbox[0];
  $sh = $bbox[3] - $bbox[1];
  $mx = ($bbox[2] + $bbox[0]) / 2;
  $my = ($bbox[3] + $bbox[1]) / 2;
  $bbox[0] = $mx - $factor * $sw / 2;
  $bbox[1] = $my - $factor * $sh / 2;
  $bbox[2] = $mx + $factor * $sw / 2;
  $bbox[3] = $my + $factor * $sh / 2;
  return $bbox;
}

function wms_scale ( $bbox, $size ) {
  #
  # Resize a spatial extent to have the same aspect ratio
  # as an image size.
  #
  # Spatial width is always preserved and spatial height is
  # altered to match the new aspect ratio. There are other
  # heuristics for this transformation which might be
  # more aesthetically pleasing.
  #
  $sw = $bbox[2] - $bbox[0];
  $sh = $bbox[3] - $bbox[1];
  $iw = $size[0];
  $ih = $size[1];
  $sh = $sw * $ih / $iw;
  $bbox[1] = (($bbox[3] + $bbox[1]) / 2) - ($sh / 2);
  $bbox[3] = (($bbox[3] + $bbox[1]) / 2) + ($sh / 2);
  return $bbox;
}

?>
```

C.2 Istemci sınıfları ve metodlarında web servislerinin kullanıldığı kaynak kodlar.

```
package net.firat.ogc;
import java.io.IOException;
import java.rmi.RemoteException;
import java.util.Iterator;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.FactoryConfigurationError;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.rpc.ServiceException;
import javax.xml.rpc.Stub;
import org.w3c.dom.Document;
import org.xml.sax.SAXException;

public class WMSClient{

        private OGCWebServicePortType service;

        public Object getMap(String xml) throws RemoteException{
                return service.getMap(xml);
        }

        public String getCapabilities(String xml)
        throws RemoteException{
                return service.getCapabilities(xml);
        }

        public String getInfo(String xml)
        throws RemoteException{
                return service.getFeatureInfo(xml);
        }

        public WMSClient(String sWSDL_URL)
        throws ServiceException{
                service = null;
                //Iterator ports = new
                WMSServicesService_Impl().getWMSServices()
                //if(ports.hasNext()){
                //Stub stub = (Stub)ports.next();
        Stub stub = (Stub)(new  OGCWebService_Impl().getOGCWebServicePort());
        service = (OGCWebServicePortType)stub;
        //}
          }
                /**
                * @param args
```

```
                * @throws ServiceException
                */



        public static void main(String[] args) {
        // TODO Auto-generated method stub
                try {
                        WMSClient client = new WMSClient("");
                        String xml = "<?xml version=\"1.0\" encoding=\"UTF-
8\"?><GetCapabilities
xmlns=\"http://www.opengis.net/ows\"><version>1.1.1</version>
<service>wms</service><exceptions>application_vnd_ogc_se_xml
</exceptions><style>full</style></GetCapabilities>";
String result = client.getCapabilities(xml);
System.out.println(result);

Document document = XMLUtils.parseXml(result,false);
//XMLUtils.getLayers(document);
//XMLUtils.getBBOX(document);
} catch (ServiceException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (RemoteException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (FactoryConfigurationError e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
}
```

C.3 İmaj görüntüleme fonksiyonları ile istemci tarafında web servis metodlarını çağıran sınıfların kullanıldığı kaynak kodlar.

```
package net.firat.ogc;

import java.awt.Canvas;
import java.awt.Color;
import java.awt.Component;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
```

```
import java.awt.Image;
import java.awt.Insets;
import java.awt.Point;
import java.awt.Rectangle;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.image.BufferedImage;
import java.awt.image.ImageProducer;
import java.io.DataInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;
import java.net.URLEncoder;
import java.rmi.RemoteException;
import java.util.ArrayList;
import java.util.Iterator;
import javax.imageio.ImageIO;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTable;
import javax.xml.rpc.ServiceException;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

public class WMSClientWnd extends JFrame implements IMapLoader,IGetInfo {
        private WMSClient client;
        private Rectangle bbox,rLimit;
        private ArrayList alLayer;
        private Image imgMap;
        private static final Insets insets = new Insets(0,0,0,0);
        private JPanel pLayer;
        private MapPanel pMap;
        private IMapDrawer mapDrawer;
        private IMapInfo   mapInfo;
        private ICommandListener mapCmd;
        private Point pQuery;
```

```
    private String onlineResource;

    public WMSClientWnd() throws ServiceException, RemoteException{
    super();

    pQuery = new Point();

    client = new WMSClient("");
    String xml="<?xml version=\"1.0\"
    encoding=\"UTF-8\"?>
    <GetCapabilities   xmlns=\"http://www.opengis.net/ows\"><version>1.1.1
    </version>
    <service>wms</service><exceptions>application_vnd_ogc_se_xml</excepti
    ons><style>full</style></GetCapabilities>";
    String result = client.getCapabilities(xml);

 Document document = XMLUtils.parseXml(result,false);
 alLayer = XMLUtils.getLayers(document);
 bbox = XMLUtils.getBBOX(document);
 onlineResource = XMLUtils.getOnlineResource(document);

 rLimit = new Rectangle();
 rLimit.setBounds(bbox);

    setSize(800,600);
    //new Window

    addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent event) {
    }
    });
    Container content = getContentPane();
    content.setLayout(new GridBagLayout());

    GridBagConstraints constraints = null;

 pMap = new MapPanel(this,this);

 mapDrawer = (IMapDrawer)pMap;
 mapInfo   = (IMapInfo)pMap;
 mapCmd    = (ICommandListener)pMap;

pMap.setBackground(Color.WHITE);
constraints = new GridBagConstraints(0, 0, // gridx, gridy
      1, 1, // gridwidth, gridheight
      1.0, 1.0, // weightx, weighty
      GridBagConstraints.NORTHWEST, // anchor
```

```
        GridBagConstraints.BOTH, // fill
        // GridBagConstraints.NONE, // fill
        // GridBagConstraints.VERTICAL, // fill
        // GridBagConstraints.HORIZONTAL, // fill
        insets, // insets
        0, // ipadx
        0); // ipady

content.add(pMap, constraints);

pLayer = new JPanel(false);
Iterator rs = alLayer.iterator();

pLayer.setLayout(new GridBagLayout());

GridBagConstraints c = new GridBagConstraints();

c.fill = GridBagConstraints.VERTICAL;
c.weightx = 0.5;
c.anchor = GridBagConstraints.LINE_START;

      int layerCount = 0;
      rs = alLayer.iterator();
      while(rs.hasNext()){
      OGCLayer layer =(OGCLayer)rs.next();
      JCheckBox cbLayer = new JCheckBox(layer.displayName);
      cbLayer.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
      repaint();
      loadMap();
      }
      });
 c.gridx = 0;
 c.gridy = layerCount++;
 pLayer.add(cbLayer,c);
      }

constraints = new GridBagConstraints(
        1, 0, // gridx, gridy
        1, 1, // gridwidth, gridheight
        0.0, 1.0, // weightx, weighty
        GridBagConstraints.NORTHEAST, // anchor
        GridBagConstraints.VERTICAL, // fill
        // GridBagConstraints.NONE, // fill
        // GridBagConstraints.VERTICAL, // fill
        // GridBagConstraints.HORIZONTAL, // fill
        insets, // insets
```

```
            0, // ipadx
            0); // ipady
        content.add(pLayer, constraints);

        JPanel pCmd = new JPanel();

        pCmd.setLayout(new FlowLayout());

        JButton bZoomIn = new JButton("Z(+)");
        bZoomIn.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
        setMode(ICommandListener.MODE_ZOOM_IN_RECT);
     }
    });

        JButton bZoomOut = new JButton("Z(-)");
        bZoomOut.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
        setMode(ICommandListener.MODE_ZOOM_OUT_POINT);
            }
    });

        JButton bPan = new JButton("Pan");
        bPan.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
        setMode(ICommandListener.MODE_PAN);
            }
            });

        JButton bLimit = new JButton("View All");
        bLimit.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
        GLMapScale.setInfo(new NCWorld(rLimit.getMinX(),          rLimit.getMinY(),
            rLimit.getMaxX(), rLimit.getMaxY()),

            mapInfo.getMapHeight(),mapInfo.getMapWidth(),mapInfo.getMapOffsetY(),
            mapInfo.getMapOffsetX());
                                    loadMap();
                            }
                    });

          JButton bPrevWindow = new JButton("Previous Window");
          bPrevWindow.addActionListener(new ActionListener() {
                            public void actionPerformed(ActionEvent e) {
                                    if (GLMapScale.getPrevWindow()) loadMap();
                            }
                    });
```

```
JButton bQuery = new JButton("Query");
bQuery.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                            setMode(ICommandListener.MODE_QUERY);

                    }
            });

pCmd.add(bZoomIn);
pCmd.add(bZoomOut);
pCmd.add(bPan);
pCmd.add(bLimit);
pCmd.add(bPrevWindow);
pCmd.add(bQuery);


constraints =
  new GridBagConstraints(
    0, 1, // gridx, gridy
    1, 1, // gridwidth, gridheight
    0.0, 0.0, // weightx, weighty
    GridBagConstraints.SOUTH, // anchor
    GridBagConstraints.HORIZONTAL, // fill
    // GridBagConstraints.NONE, // fill
    // GridBagConstraints.VERTICAL, // fill
    // GridBagConstraints.HORIZONTAL, // fill
    insets, // insets
    0, // ipadx
    0); // ipady
content.add(pCmd, constraints);

      setVisible(true);

pMap.initDoubleBuffering();

GLMapScale.init();
   GLMapScale.setInfo(new     NCWorld(bbox.getMinX(),     bbox.getMinY(),
   bbox.getMaxX(), bbox.getMaxY()),

   mapInfo.getMapHeight(),mapInfo.getMapWidth(),mapInfo.getMapOffsetY(),
   mapInfo.getMapOffsetX());
   loadMap();
   }

   public void setMode(int mode){
   mapCmd.setMode(mode);
```

```
        }

        protected String getActiveLayers(){
          String result = "";
          Iterator rs = alLayer.iterator();
          for(int i=0;i<pLayer.getComponentCount();i++){
                  Component c = pLayer.getComponent(i);
                  if( c   instanceof JCheckBox ) {
                    JCheckBox cbLyr = (JCheckBox)c;
                       OGCLayer ol = (OGCLayer) rs.next();

                    if ( cbLyr.isSelected() ) {
                            result += ","+ol.name;
                    }
                  }

          }
          if (result == "") return result;
          return result.substring(1);
        }
        /**
         * @param args
         * @throws ServiceException
         * @throws RemoteException
         */
        public   static   void   main(String[]   args)   throws   ServiceException,
        RemoteException {
        new WMSClientWnd();
        }

        private synchronized Image downloadImage (String url) {
        Image img = null;
                try {
         //System.err.println ("Fetching " + url + " ...");
         try {
            java.net.URL u = new java.net.URL (url);
         img = Toolkit.getDefaultToolkit ().createImage (u);
         }
                catch (Exception ee) {
         // filename
         img = Toolkit.getDefaultToolkit ().createImage (url);
         }
java.awt.MediaTracker tracker = new java.awt.MediaTracker(this);
         tracker.addImage(img, 0);
         try {
                        tracker.waitForAll();
               } catch (Exception e) { }
```

```
        ////System.err.println ("w = " + img.getWidth (this));
                ////System.err.println ("h = " + img.getHeight (this));
    } catch (Exception ex) {
        ////System.err.println(ex);
    }
if (img==null) return null;
    else return img;
 }

        public void loadMap() {
                String sLayers = getActiveLayers();
                  try {
                        mapDrawer.mapLoading();
                        String  sGetMap        =
        XMLUtils.getMapRequest(onlineResource,mncursor.getWorld(GLMapScale.
        getCurrentWindow()),sLayers,"EPSG:42304",mapInfo.getMapWidth(),mapIn
        fo.getMapHeight());
        Object sMapURL = client.getMap(sGetMap);
        String url[]=
        XMLUtils.resolveGetMapURL(XMLUtils.parseXml(sMapURL.toString(),fal
        se));
        String t = url[0]+url[1];
        t = t.trim();
        URL _url = new URL(t);
        imgMap = Toolkit.getDefaultToolkit().createImage(_url);
        if (imgMap!=null)
        mapDrawer.drawMap(imgMap);
        } catch (RemoteException e1) {
        e1.printStackTrace();
        } catch (MalformedURLException e) {
        e.printStackTrace();
        }
        }
        public void setBbox(Rectangle r) {
                }
        public void setQueryPoint(Point p) {
                pQuery.setLocation(p);
        }
        public Node getGMLNode(Node node){
          if (node.getNodeName()=="gml:boundedBy"){
        System.out.println("name:"+node.getNodeName()+"value:
        "+node.getNodeValue());
        printAttribute(node);
        return node;
          }

        NodeList list=node.getChildNodes();
```

```
            for(int i=0;i<list.getLength();i++){
                    Node aNode=list.item(i);
                    getGMLNode(aNode);
            }
            return null;
          }

          public void printAttribute(Node node){
          NamedNodeMap map=node.getAttributes();
          if(map!=null){
          for(int i=0;i<map.getLength();i++){
          Node aNode=map.item(i);
          String name=aNode.getNodeName();
          String value=aNode.getNodeValue();
          System.out.println("Attribute: "+name+" : "+value);
                          }
                        }
                      }

          public void queryCallback() {
          String sLayers = getActiveLayers();
          String  result  =              XMLUtils.getFeatureInfoRequest(onlineResource,
mncursor.getWorld(GLMapScale.getCurrentWindow())
,"EPSG:42304",pQuery,sLayers,sLayers,mapInfo.getMapWidth(),mapInfo.getMapH
eight());
           if(result!=null){
                    if(result!=""){
                            try {
                                    result = client.getInfo(result);
                                            System.out.println(result);

final Document doc = XMLUtils.parseXml(result,false);
new QueryResultWnd(this,doc);
} catch (Exception e) {
e.printStackTrace();
    }
   }
  }
 }
}
```

C.4 XML Utilities sınıf metodu ile XML istek/yanıt servisini istemci tarafına aktaran ve kullanan kaynak kodlar.

```
package net.firat.ogc;

import java.awt.Component;
import java.awt.Image;
import java.awt.MediaTracker;
import java.awt.Point;
import java.awt.Rectangle;
import java.awt.Toolkit;
import java.awt.geom.Arc2D.Double;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URLDecoder;
import java.net.URLEncoder;
import java.util.ArrayList;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.TransformerException;
import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import com.sun.org.apache.xpath.internal.XPathAPI;

public class XMLUtils {

public static Document parseXml(String xmlData, boolean validating) {
            try {
// Create a builder factory
DocumentBuilderFactory      factory DocumentBuilderFactory.newInstance();
factory.setValidating(validating);

// Create the builder and parse the file
InputStreamReader inputStreamReader = null;

İnputStreamReader    =         new     InputStreamReader(new
ByteArrayInputStream(
xmlData.getBytes()), "ISO8859_9");

InputSource inputSource = new InputSource(inputStreamReader);
Document doc = factory.newDocumentBuilder().parse(inputSource);
return doc;
```

```
} catch (SAXException e) {
// A parsing error occurred; the xml input is not valid
} catch (ParserConfigurationException e) {

} catch (IOException e) {
}
return null;
            }

public static ArrayList getLayers(Document doc){
try {
NodeList        nl        =        XPathAPI.selectNodeList(doc,
"/WMT_MS_Capabilities/Capability/Layer/Layer");

ArrayList al = new ArrayList();

for(int i=0;i<nl.getLength();i++){
Node nLayer = nl.item(i);
Node nlLayer = XPathAPI.selectSingleNode(nLayer,"Title");
String sLayerTitle=nlLayer.getChildNodes().item(0).getNodeValue();
nlLayer = XPathAPI.selectSingleNode(nLayer,"Name");
String sLayerName    = nlLayer.getChildNodes().item(0).getNodeValue();

OGCLayer ol = new OGCLayer();
ol.name = sLayerName;
ol.displayName = sLayerTitle;
al.add(ol);
nLayer = null;
}
return al;
}
catch(TransformerException te){
return null;
}
}

public static Rectangle getBBOX(Document doc){
try {
NodeList        nl        =
XPathAPI.selectNodeList(doc,"/WMT_MS_Capabilities/Capability/Layer/Bounding
Box");
Rectangle r = new Rectangle();
String sMinX  = nl.item(0).getAttributes().getNamedItem("minx").getNodeValue();
String  sMinY = nl.item(0).getAttributes().getNamedItem("miny").getNodeValue();
String  sMaxX= nl.item(0).getAttributes().getNamedItem("maxx").getNodeValue();
String  sMaxY= nl.item(0).getAttributes().getNamedItem("maxy").getNodeValue();
```

```
double clly = java.lang.Double.parseDouble(sMinX);
double cllx = java.lang.Double.parseDouble(sMinY);
double cury = java.lang.Double.parseDouble(sMaxX);
double curx = java.lang.Double.parseDouble(sMaxY);

r.setFrameFromDiagonal(clly,cllx,cury,curx);
return r;
            }
catch (TransformerException e) {
return null;
            }
  }

public static String getCapabilitiesRequest(){
return "<GetCapabilities></GetCapabilities>";
  }

public static String getMapRequest(String url,Rectangle bbox,String layers,String
proj,int width,int height){
String result =
"<GetMap>
        <URL>"+URLEncoder.encode(url)+"</URL>
        <BBOX>"+bbox.getMinX()+","+bbox.getMinY()+","+bbox.getMaxX()+","+
        bbox.getMaxY()+"</BBOX>
        <LAYERS>"+layers+"</LAYERS>
        <SRS>"+proj+"</SRS>
        <WIDTH>"+width+"</WIDTH>
        <HEIGHT>"+height+"</HEIGHT>
        <FORMAT>image/jpeg</FORMAT>
        </GetMap>";
        return result;
  }

  public static String getOnlineResource(Document doc){
            try {
NodeList nl    =
XPathAPI.selectNodeList(doc,"/WMT_MS_Capabilities/Service/OnlineResource");
String  result  =
nl.item(0).getAttributes().getNamedItem("xlink:href").getNodeValue();
return result;
}
catch (TransformerException e) {
return "";
}
}
```

```
public static String getFeatureInfoRequest(String url,Rectangle bbox,String
proj,Point p,String layers,String qlayers,int width,int height){
return
"<GetFeatureInfo>"+
"<URL>"+URLEncoder.encode("http://localhost/cgi-
bin/mapserv.exe?map=/ms4w/apps/maplab-
2.2.1/tutorial/tutorial.map&")+"</URL>"+
"<BBOX>"
+bbox.getMinX()+","+bbox.getMinY()+","+bbox.getMaxX()+","+bbox.getMaxY()+
"</BBOX>"+
"<SRS>"+proj+"</SRS>"+
"<X>"+p.x+"</X>"+
"<Y>"+p.y+"</Y>"+
"<LAYERS>"+layers+"</LAYERS>"+
"<QUERYLAYERS>"+qlayers+"</QUERYLAYERS>"+
"<WIDTH>"+width+"</WIDTH>"+
"<HEIGHT>"+height+"</HEIGHT>"+
"<INFO_FORMAT>application/vnd.ogc.gml</INFO_FORMAT>"+
"</GetFeatureInfo>";
  }
public static String [] resolveGetMapURL(Document doc){
String result[] = new String[2];

try {
        NodeList nl = XPathAPI.selectNodeList(doc,"/GetMap/mainurl");
result[0]=URLDecoder.decode(nl.item(0).getChildNodes().item(0).getNodeValue());

nl = XPathAPI.selectNodeList(doc,"/GetMap/subdir");
result[1] =
URLDecoder.decode(nl.item(0).getChildNodes().item(0).getNodeValue());

} catch (TransformerException e) {e.printStackTrace();}
return result;}
public static Image loadImageResource(Component parent,
InputStream is)
throws IOException {
    Image ret = null;
    if (is != null) {
      byte[] buffer=null;// = new byte[];
      byte[] tmpbuf = new byte[1024];
      boolean started = false;
      int numRead = 0;

      String s = "";

      while (true) {
       try{
```

```
            int len = is.read(tmpbuf);
            if (len <= 0) {
                if (!started)
                    return null;
                break;
            } else {
                started = true;
                numRead += len;
            }
          } catch(Exception e){e.printStackTrace();
          }
        }
    // create image
    ret = Toolkit.getDefaultToolkit().createImage(s.getBytes());
      }
      MediaTracker mt = new MediaTracker(parent);
      mt.addImage(ret, 0);
      try {
         // wait until the image is loaded
         mt.waitForAll();
         if (mt.isErrorAny()) {
         }
      }catch (InterruptedException e) {
      }return ret;
  }
}
```