KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HARİTA MÜHENDİSLİĞİ ANABİLİM DALI

KONUMSAL VERİ ALTYAPILARINDA SEMANTİK WEB SERVİSİ KOMPOZİSYONU İÇİN BİR MİMARİ

DOKTORA TEZİ

Harita Yük. Müh. Deniztan ULUTAŞ

NİSAN 2015
TRABZON

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HARİTA MÜHENDİSLİĞİ ANABİLİM DALI

KONUMSAL VERİ ALTYAPILARINDA SEMANTİK WEB SERVİSİ KOMPOZİSYONU İÇİN BİR MİMARİ

Deniztan ULUTAŞ

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"DOKTOR (HARİTA MÜHENDİSLİĞİ)"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 06 / 04 /2015
Tezin Savunma Tarihi : 27 / 04 /2015

Tez Danışmanı : Prof. Dr. Çetin CÖMERT

Trabzon 2015

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Deniztan ULUTAŞ Tarafından Hazırlanan

KONUMSAL VERİ ALTYAPILARINDA SEMANTİK WEB SERVİSİ KOMPOZİSYONU İÇİN BİR MİMARİ

başlıklı bu çalışma, Enstitü Yönetim Kurulunun 07 /04 / 2015 gün ve 1597 sayılı kararıyla oluşturulan jüri tarafından yapılan sınavda
DOKTORA TEZİ
olarak kabul edilmiştir.

Jüri Üyeleri

Başkan : Prof. Dr. Çetin CÖMERT .....

Üye : Prof. Dr. Mustafa TÜRKER .....

Üye : Prof. Dr. Haluk ÖZENER .....

Üye : Doç. Dr. Oğuz GÜNGÖR .....

Üye : Yrd. Doç. Dr. Hüseyin PEHLİVAN .....

Prof. Dr. Sadettin KORKMAZ

Enstitü Müdürü

ÖNSÖZ

"Konumsal Veri Altyapılarında Semantik Web Servisi Kompozisyonu için Bir Mimari" adlı bu çalışma, Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Harita Mühendisliği Anabilim Dalında Doktora Tezi olarak hazırlanmıştır.

Tez çalışması süresince ileri görüşlülüğü ile bana yol gösteren, değerli fikirleri ve önerileriyle çalışmalarıma yön veren, ilgi ve yardımlarını hiçbir zaman esirgemeyen değerli hocam Sayın Prof. Dr. Çetin CÖMERT' e tez danışmanlığımı üstlendiği ve çalışmamın her aşamasında yanımda olarak desteğini hiç eksik etmediği için sonsuz minnet ve şükranlarımı sunarım.

Doktora tez çalışmasının yürütülmesinde tez izleme komitesinde görev alarak, bilgi ve tecrübeleriyle beni yönlendiren, desteklerini esirgemeyen değerli hocalarım Doç. Dr. Oğuz GÜNGÖR ve Yrd. Doç. Dr. Hüseyin PEHLİVAN' a teşekkür ederim. Tez çalışmam sırasında karşılaştığım problemler ve yazılım konusunda yardımlarını esirgemeyen Arş. Gör. Deniz YILDIRIM, Aliaksandr AUTAYEU, Jorge Samuel MENDES De JESUS ve Doç. Dr. Halil AKINCI'ya teşekkür ederim.

Tez çalışması süresince bana her konuda yardımcı olan, desteklerini esirgemeyen, Yrd. Doç. Dr. Leyla ÇAKIR, Arş. Gör. Cemre YILMAZ, Yrd. Doç. Dr. Gülten KARA, Arş. Gör. Hayrettin ACAR, Yrd. Doç. Dr. Hasan Tahsin BOSTANCI, Yrd. Doç. Dr. Nazan YILMAZ ve tüm KTÜ Mühendislik Fakültesi Harita Mühendisliği Bölümü akademik ve idari personeline teşekkür ederim.

Ayrıca tez süresi boyunca desteğini hep hissettiğim, bana manevi güç veren dostlarım Ekin Bircan BAYRAM, Nesrin ARSLAN, Ayşe ŞAHİN, Aybeniz BAYRAMOV ve Yrd. Doç. Dr. Meher BAYRAMOV'a teşekkür ederim.

Ayrıca bugünlere gelmemde tüm hayatım boyunca ellerinden gelen her türlü fedakârlığı yapan, sonsuz destekleriyle her zaman yanımda olan annem Nurcan ULUTAŞ, babam Fatih ULUTAŞ ve kardeşlerim Sertan Dağıstan ULUTAŞ ve Nur Birtan ULUTAŞ'a sonsuz teşekkürlerimi sunarım.

Deniztan ULUTAŞ
Trabzon 2015

TEZ ETİK BEYANNAMESİ

Doktora Tezi olarak sunduğum "Konumsal Veri Altyapılarında Semantik Web Servisi Kompozisyonu için Bir Mimari" başlıklı bu çalışmayı baştan sona kadar danışmanım Prof. Dr. Çetin CÖMERT'in sorumluluğunda tamamladığımı, verileri kendim topladığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 27/04/2015

Deniztan ULUTAŞ

İÇİNDEKİLER

Sayfa No

Doktora Tezi

ÖZET

KONUMSAL VERİ ALTYAPILARINDA SEMANTİK WEB SERVİSİ KOMPOZİSYONU İÇİN BİR MİMARİ

Deniztan ULUTAŞ

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Harita Mühendisliği Anabilim Dalı
Danışman: Prof. Dr. Çetin CÖMERT
2015, 114 Sayfa, 9 Ek Sayfa

Gelişen teknoloji ile önemi daha da artan konumsal veri kamu, özel sektör ya da konumsal veri ile iş yapan bütün kesimler için farklı uygulamalarda karar verme aracı olarak kullanılmaktadır. Bu amaçla, son yıllarda Konumsal Veri Altyapılarının (KVA) geliştirilmesi önem kazanmıştır. "Konumsal Veri Altyapıları (KVA)", konumsal veri yönetimine yönelik birlikte işlerlik altyapılarıdır. KVA'ların gerçekleştiriminde kurumlar arası veri entegrasyonu önemli bir gereksinimdir. Gelinen son teknoloji ile literatürde, veri entegrasyonuna kolaylık sağlayan "semantik eşleştirme" teknikleri kullanılmaktadır. Bu amaçla bu tez kapsamında ilk olarak, farklı kurumlara ait konumsal veri tabanı şemalarının semantik eşleştirilmesine yönelik bir gerçekleştirim yapılarak, bu alandaki problemlerin belirlenmesine çalışılmıştır. Bu problemlerden yola çıkarak, son zamanların popüler konularından KVA'ların web servisleri ile oluşturulması boyutuna yönelik, web servislerinin Semantik Web Teknolojilerine (SWT) dayalı olarak kompozisyonu için bir mimari önerilmiştir. Bu mimari, seçilen bir "Yer Seçimi" senaryosuyla tanıtılmıştır. Web Servisi Kompozisyonu (WSK), atomik web servislerinin ihtiyacı karşılamadığı karmaşık uygulamalarda, birden çok web servisinin bir araya getirilerek yeni bir web servisinin oluşturulması demektir. Bu alanda çok çeşitli akademik ve pratik çalışmalar bulunmakla birlikte, Tam Otomatik WSK (TOWSK) henüz çözümlenebilmiş bir konu değildir. TOWSK ile kastedilen, servislerin bir araya getirilmesi ve oluşan yeni servisin koşum işleminin, insan müdahalesi olmadan yürütülmesidir. TOWSK'nın henüz başarılamamış olması, içerdiği problemler nedeniyledir. Bu problemlerin esin kaynağı olduğu bu çalışmada önerilen WSK mimarisi, SWT desteğiyle; WSK geliştiricisinin amacını yüksek bir düzeyde söyleyebilmesi, Konumsal Servis Ontolojisinin (KSO) geliştirilmesi ve web servisi kompozisyonunun oluşturulması aşamalarından oluşmaktadır. Bu aşamaların gerçekleştirimi, geliştiricinin belirttiği üst düzey bir "amaç" tümcesinden, KSO'daki "kompoze servis" tipinin ve WSK içinde kullanılacak atomik web servislerinin çıkarsanmasıyla, iş modeli iş akışı sırasında çalıştırılarak bir "Yer Seçimi" işleminin gerçekleştirilmesi ilkesine dayanmaktadır.

Anahtar Kelimeler: Semantik Eşleştirme, Semantik Web Servisleri, Web Servisi Kompozisyonu, Konumsal Servis Ontolojisi, KVA, Semantik Web.

PhD. Thesis

SUMMARY

AN ARCHITECTURE FOR SEMANTIC WEB SERVICES COMPOSITION IN SPATIAL DATA INFRASTRUCTURES

Deniztan ULUTAŞ

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Geomatics Engineering Graduate Program
Supervisor: Prof. Dr. Çetin CÖMERT
2015, 114 Pages, 9 Appendix Pages

Pressed by technological developments, the increased importance of geospatial data has rendered it to be used in decision making throughout public, private sectors, and all other interested parties. This was indeed the reason of quest for "Spatial Data Infrastructures (SDI)" among spatial data communities for the past two decades. Being interoperability infrastructures, SDIs enable the integration of data from different sources. "Semantic schema matching" is a recent Semantic Web Technology (SWT) for enabling data integration in SDIs an "automatic" manner. For this reason, during first part of this thesis, the database schemas of different Public agencies have been tried to be matched by semantic matching. As a result, it has been found that full automated schema matching by SWT is not possible today. Accordingly, a number of problems have been identified and some proposals for these problems have been made. In the second part of the thesis, another popular problem like semantic matching, Web Services Composition (WSC) has been tackled. The purpose was to employ SWT for the problems of WSC in the context of SDIs. As the result, an architecture for this aim has been proposed. The architecture has been tested for the case study of the thesis, "site selection for waste disposal". A WSC is a combination of a number of Web Services to perform a more complex task than the combined services can do alone. Currently WSC is performed semi-automatically meaning that human intervention is needed. Although there is plenty of research towards, Full Automated WSC (FAWSC) has not been achieved yet. In FAWCS both the combination of services into an "abstract" WSC and the execution of the WSC ("concrete" WSC) are performed by the software without human intervention. To achieve FAWCS a number of problems have to addressed. This has formed the inspiration for this thesis; would SWT have some provisions in dealing with these problems? An architecture has been proposed in this thesis. By the architecture, the user states his "goal" by a natural language sentence. By semantically matching this sentence to a Spatial Services Ontology (SSO), the corresponding "abstract" WSC is "located" within SSO. Discovering the "matching" advertised services semantically again, "concrete" WSC is formed afterwards. Although there are still untackled problems due partly to the scope of the thesis, this study makes a valuable contribution to the area.

Key Words: Semantic matching, Semantic Web Services, Web Services Composition, Spatial Services Ontology, Spatial Data Infrastructures, Semantic Web.

ŞEKİLLER DİZİNİ

Sayfa No

TABLOLAR DİZİNİ

Sayfa No

KISALTMALAR DİZİNİ

CBS	: Coğrafi Bilgi Sistemi
DL	: Description Logic
DOLCE	: Descriptive Ontology for Linguistic and Cognitive Engineering
GML	: Geography Markup Language
GWN	: GeoWordNet
HGK	: Harita Genel Komutanlığı
INSPIRE	: Infrastructure for Spatial Information in Europe
ISO TC 211	: ISO Technical Committee 211
ISO	: International Organization for Standardization
KSO	: Konumsal Servis Ontolojisi
KVA	: Konumsal Veri Altyapısı
OGC	: OpenGIS Consortium
OWL	: Web Ontology Language
OWL-S	: Web Ontology Language for Services
RDF	: Resource Description Framework
RDF-S	: RDF Schema
SPARQL	: SPARQL Protocol and RDF Query Language
STO	: Servis Tanımları Ontolojisi
SWRL	: Semantic Web Rule Language
SWS	: Semantik Web Servisi
SWSK	: Semantik Web Servisi Kompozisyonu
SWT	: Semantik Web Teknolojileri
TOWSK	: Tam Otomatik Web Servisi Kompozisyonu
UAO	: Ulaşım Alan Ontolojisi
URI	: Uniform Resource Identifier
W3C	: World Wide Web Consortium
WFS	: Web Feature Service
WN	: WordNet
WPS	: Web Processing Service
WS	: Web Servisleri

WSDL : Web Service Description Language

WSK : Web Servisi Kompozisyonu

WSML : Web Service Modeling Language

WSMO : Web Service Modeling Ontology

XML : Extensible Markup Language

XSD : XML Schema Definition

1. GENEL BİLGİLER

1.1. Giriş

Günden güne gelişen teknoloji, konumsal verinin önemini daha da artırmıştır. Kamu, özel sektör ya da konumsal veri ile iş yapan bütün kesimler, farklı uygulamalar için karar verme aracı olarak konumsal veriye ihtiyaç duymaktadır. Ancak hangi veri, hangi konumsal işlemlerde kullanılabilir? Bu konumsal işlemler nasıl çalıştırılabilir? Birden fazla konumsal işlemin gerektiği uygulamalarda nasıl bir yol izlenebilir? tarzındaki soruların cevabı, özellikle "yer seçimi" veya "risk analizi" gibi karmaşık ama popüler Coğrafi Bilgi Sistemleri (CBS) uygulamalarında, teknik kullanıcının teorik ve pratik altyapı bakımından donanımlı olmasını gerektirir. Bu da çoğu zaman mümkün olamayan bir durumdur. Bu nedenle, son yıllarda öne çıkan çalışmalarda Semantik Web Teknolojileri (SWT) desteğiyle kullanıcıya yardımcı olmak adına, konumsal veri edinme ve veri entegrasyonuna çözüm sağlayacak, "konumsal veri yönetimine yönelik birlikte işlerlik altyapıları" olarak tanımlanan Konumsal Veri Altyapılarının (KVA) Web Servisleri (WS) ve kompozisyonu ile uygulanabilirliği gündeme gelmiştir.

Web Servisi Kompozisyonu (WSK), WS alanında son yılların en popüler araştırma konularından biri olmuştur. WSK, bir web servisinin ihtiyacı karşılamadığı karmaşık uygulamalarda, birden çok web servisinin bir araya getirilerek yeni bir web servisinin oluşturulması demektir. WSK'nın ana hedeflerinden biri, mevcut web servislerini bir iş akışı içinde kompoze ederek yeniden kullanılabilirliğini sağlamaktır.

Tam Otomatik WSK (TOWSK), servislerin bir araya getirilmesi ve oluşan yeni servisin koşum işleminin, insan müdahalesi olmadan yürütülmesidir. TOWSK'nın en önemli gereksinimi SWT'lerdir. Bu alanda literatürde çok çeşitli akademik ve pratik çalışmalar bulunmakla birlikte, mevcut problemler henüz çözümlenebilmiş bir konu değildir. TOWSK'nın henüz ulaşılamamış bir hedef olması, içerdiği sorunların insan müdahalesi olmadan çözümlenebilmesinin, eğer imkansız değilse, çok zor olmasındandır. Bu çok kolay bir iş değildir ve içerdiği bir dizi problem vardır. Bu tez çalışmasında, bu problemlerden bazıları için, SWT kullanılarak belirli çözüm önerilerinin sunulmasına yönelik çalışmalar yapılacaktır.

1.2. Problemin Tanımı

Bu tezin çıkış noktası, web üzerinden CBS uygulaması gerçekleştirmek isteyen kullanıcıların bu işlerini bir WSK ile gerçekleştirebilmelerine olanak tanıyacak SWT'ye dayalı bir WSK mimarisinin tanımlanması ve kısmen geliştirilmesidir.

Günümüz "uygulama geliştirme" eğilimlerinde en temel faktör olan "hızlı geliştirme", hiç şüphesiz ki önemlidir. Bu bakımdan, KVA'ların SWT desteğinde WS'ler ile icra edilmesi bir süredir gündemdedir. Literatürde WSK'nın SWT ile tam otomatik bir tarzda gerçekleştirilmesine yönelik çok çeşitli akademik ve pratik çalışmalar mevcuttur. Ancak, bu konudaki problemler henüz çözülebilmiş değildir. Bunun nedeni, hala insan müdahalesine gerek duyulmasıdır. Bu gereklilik; hangi veri için hangi WS uygundur? Hangi konumsal işlem için hangi WS kullanılmalıdır? Hangi iş akışı sırasında hangi WS olmalıdır? Hangi WS'ler birbiri ile uyumludur? gibi sorulara cevap verecek olan kullanıcının teknik olarak çok donanımlı olmasını gerektiren bir durumdur. Bu da tasarımcının yükünü artırır. Bu yükü hafifletmek adına çeşitli çalışmalar vardır. Ancak, tam otomatik WSK halen ulaşılabilmiş bir hedef değildir. Çünkü bu, uygulama gerçekleştirecek bir kullanıcının, "insan" olarak yaptığı bütün işlemlerde yüklü olan "semantiğin", uygulama iş akışına "gömülmesi"ni gerektirir. Bu da içerdiği problemler nedeniyle kolay bir iş değildir. Bu tez çalışmasında, bu problemlerden bazıları için, SWT kullanılarak bir WSK mimarisi geliştirilmesi ile belirli çözüm önerilerinin sunulmasına çalışılacaktır.

1.3. Çalışmanın Amacı

Bu tez çalışmasının amacı, KVA'lar için; Dünya genelinde aktif araştırma ve geliştirme alanlarından biri olan konumsal web servisi bulma, servis kompozisyonu ve şema eşleştirme alanındaki problemlerin belirlenmesi, bu problemlerden bazıları için SWT ile belirli çözüm önerilerinin sunulması ve bu alandaki gelecek yönelimlerinin belirlenmesidir.

1.4. Metodoloji

Bu çalışmanın gerçekleştirilmesinde sırasıyla aşağıdaki gibi bir yol izlenmiştir:

- Semantik eşleştirme için gerekli teknolojilerin tanınması ve incelenmesi,
- Konumsal şemaların semantik eşleştirilmesine yönelik bir yol gösterilmesi,
- SWSK için gerekli teknolojilerin tanınması ve uygulama alanlarının belirlenmesi,
- SWSK için hem konumsal hem de genel olarak gerçekleştirilen semantik uygulamalara yönelik uluslararası ve ulusal çalışmalar, geliştirilen standartlar, çok sayıda akademik ve bilimsel çalışma, mevcut yazılım ve araçları ile ilgili dokümanlar, uluslararası ve ulusal düzeyde gerçekleştirilen çalışmaların ayrıntılı olarak incelenmesi,
- SWT ile WSK gerçekleştirimi konusunda mevcut durum analizi,
- SWT ile WSK gerçekleştirim yolunun gösterilmesi ve benzer çalışmalarla karşılaştırılması,
- Gelecek yönelimlerinin belirlenmesi.

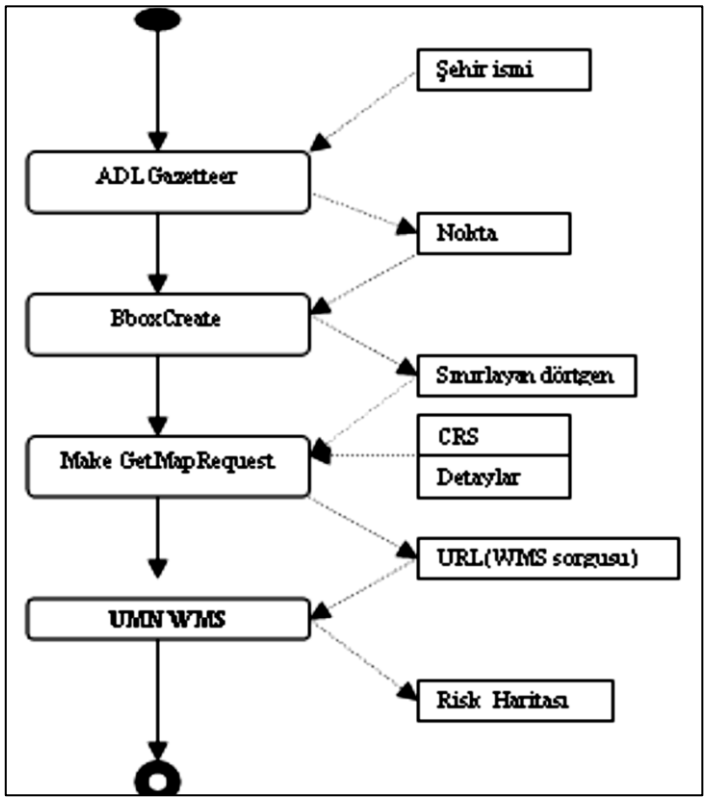
1.5. Benzer Çalışmalar ve Bu Tez Çalışmasının Farkı

Semantik Web Servisi (SWS) bulma ve SWS Kompozisyonu (SWSK) ile ilgili benzer çok sayıda proje ve akademik çalışma yapılmıştır. Bunlardan bazıları izleyen bölümde incelenmiştir ve bu tez çalışmasının farkı ortaya konmuştur.

1.5.1. Dağıtık Konumsal Servislerin Semantik Birlikte İşlerliği

Lemmens (2006) çalışmasında konumsal servisler için semantik birlikte işlerlik çerçevesi geliştirmiştir. Bu yaklaşımda servis bulma, soyut ve somut servis düzenleme ve servis yürütme işlevlerini birleştiren bir yöntem uygulanmıştır. Semantik ve sözdizimsel servis tanımları birleştirilerek servis zinciri (chaining) oluşturulmuştur. Servis eşleştirme, servislerin girdi ve çıktı parametreleri ile yapılmıştır. Yani, servis düzenleme ile oluşturulacak yeni bir servis için, bir "B" servisinin bir "A" servisinden sonra kullanılabilir olması, "B" servisinin girdi parametresi tipinin "A" servisinin çıktı parametresi tipi tarafından kapsanan bir tip olması ilkesi esas alınmıştır (Lemmens vd., 2006).

Çalışma kapsamında risk haritası sunan bir kompoze servis geliştirilmiştir. Risk haritasını oluşturan servis zinciri sırasıyla "Gazeteer", "Bbox", "Make GetMap Request" ve "Web Map Servisi" inden oluşmaktadır. "ADL Gazeteer" servisi girdi parametresi olarak şehir ismi alır ve çıktı parametresi olarak o şehirdeki merkezi bir noktanın koordinatlarını döndürür. "BboxCreate" servisi girdi olarak koordinat bilgisi alır ve çıktı olarak o koordinatları sınırlayan bölgeyi döndürür. "Make GetMap Request" servisi girdi parametresi olarak bir önceki serviste oluşturulan sınırlayıcı dörtgen koordinatları ile birlikte kullanıcının eklediği koordinat sistemi ve detay isimlerini alır, çıktı parametresi olarak WMS'in (Web Map Service) çalışmasını sağlayan sorgu ifadesini temsil eden URL'yi döndürür. Yani, bu operasyon ile WMS'in girdisi olacak parametreler oluşturulur. Son olarak "Web Map Servisi", girdi olarak oluşturulan WMS sorgusunu alır ve çıktı olarak harita döndürür. Böylece istenen herhangi bir yer için, anlık tehlikeli madde potansiyelinin yoğunluklu olduğu bölgeleri gösteren bir risk haritası üretilir (Şekil 1).



Şekil 1. GetRiskMap web servisinin UML aktivite diyagramı (Lemmens vd., 2006)

Çalışma kapsamında kullanılan yazılım bileşenleri “GeoMatchmaker” (servis bulma ve soyut servis düzenleme işlevini görür) ve “Integrated Component designer” (somut servis düzenleme ve servislerin koşturulmasını sağlar) olarak ikiye ayrılmıştır. GeoMatchmaker, ontoloji ve servis zincirini bütünleşik bir yazılımda birlikte oluşturabilmek için geliştirilmiştir. GeoMatchmaker, OWL-S editörü’nün RacerPro (URL-1, 2015) çıkarsamacısı ve bu çıkarsamacı ile iletişim kurmasını sağlayan bir kullanıcı arayüzü (jracer) sağlayarak zenginleştirilmesi ile Eclipse java geliştirme yazılımı ile oluşturulmuştur. Eclipse (URL-2, 2015), eklenti mekanizması ile geliştirilebilen açık kaynak kodlu bir yazılım geliştirme platformudur. Çalışmada ontoloji, Protege Ontoloji Editörü ile oluşturulmuştur. Çıkarsamacı olarak Racerpro kullanılmıştır. RacerPro, OWL dökümanlarını direkt okuyarak DL (Description Logic) bilgi tabanlarında (KnowledgeBase) Tbox (Terminology box) ve Abox (Assertion box) olarak gerçekleştirim yapabilmektedir. Mimari işleyişi şöyledir:

1. Servis sağlayıcı (provider) servis tanımlarını oluşturarak bir web servis kataloğuna (web service repository) kaydeder (publish). Çalışmada “IC Library” yerel servis kataloğu kullanılmıştır.
2. İstemci semantik bir çatıya dayalı olarak sorgusunu GeoMatchmaker’a gönderir.
3. Reasoner, bilgi tabanında istenen servis sınıfına uyan servis örneklerini (instance) ontoloji yardımıyla çıkarsama yaparak bulur. Bulunan servislerin sorguda istenen servisle eşleşme derecesini belirler ve katalog’dan bu servisleri getirerek WSDL tanımlarından adreslerine ulaşır.
4. GeoMatchmaker sonuç olarak soyut servis kompozisyonunu oluşturan uygun servislerin listesini içeren OWL-S dökümanını üretir. Bu listeden “integrated component” (IC) oluşturulur.
5. IC, semantik ve sözdizimsel servis tanımlarını birleştirerek kullanıcının belirlenen servis adaylarından somut servis kompozisyonu oluşturmasını sağlar.
6. Kullanıcı, IC dönüşümü ile oluşan servis kompozisyonunu yürütür. Bu aşamada risk haritası servis kompozisyonunu içeren “integrated component” WSBPEL işlem dökümanına dönüştürülmüş olur.
7. WSBPEL işlem dökümanı Oracle BPEL Process Manager aracılığıyla çalıştırılarak sonuç risk haritası üretilir.

Tez çalışmasının genel amacı; ISO (URL-3, 2015) ve OGC (OpenGIS Consortium) (URL-4, 2015) Coğrafi Bilgi Standartlarına dayanarak, ontolojilerin tasarım ve

uygulanmasıyla dağıtık heterojen konumsal bilgi ve konumsal servislerin semantik tanımlarını oluşturmak için çözüm sağlamaktır. Konumsal operasyon ontolojisi ile coğrafi bilgi ve kavramsal ilişkilerin katmanlı soyutlamasına dayanan, ontoloji tiplerinin ayrımı için bir teori önerilmiştir. Servis tanımları için referans olarak kullanılabilecek konumsal operasyon ontolojisi (OPERA) geliştirilmiştir. SWT kullanılarak semantik servis tanımları yardımıyla konumsal çıkarsama gerçekleştirilmiş, konumsal servis bulma ve servis kompozisyonunda kullanıcıya yardımcı olmak amaçlanmıştır. Tez çalışması kapsamında geliştirilen semantik birlikte işlerlik çerçevesi, konumsal bilgi ve servis tanımlarını oluşturmak ve onlar arasında eşleştirmeyi gerçekleştirmek için kullanılmıştır.

Bu çalışma yarı otomatik servis bulmaya bir örnektir. Çalışmadaki iş akışı bileşenleri, ontoloji kavramları ve WSDL parametreleri arasında ortak bir format yoktur. Bu eksiklik, var olan servis kompozisyonlarının tekrar kullanılmasıyla giderilmeye çalışılmıştır. Ayrıca bu çalışmada servis zincirindeki bileşenler arasında ortak ontolojilerin kullanılması da diğer bir güçlüktür (Lemmens vd., 2006).

Çalışmada servis kompozisyonu, kullanıcı tarafından tek tek bütün servislerin bulunmasıyla oluşturulmaktadır. Bu da servis istemcisinin aradığı her servis için girdi, çıktı, ön koşul ve son koşul parametrelerine göre sorgu oluşturarak servisleri bulması ile gerçekleşmektedir. Yani, kullanıcı her adımda servis kompozisyonuna müdahale etmektedir. Oysaki kullanıcı, oluşturmak istediği uygulama için hangi servislerin hangi operasyonlarını, hangi iş akışı sırasına göre kullanacağını ve bu operasyonları çalıştırabilmek için gerekli parametreleri bilmeyebilir. Bu nedenle, bu tez kapsamında önerilen mimariyle kullanıcının kompozisyon işleyişi hakkında hiçbir şey bilmediği varsayıldığı için, doğrudan hazır bir servis kompozisyonu kullanıcıya buldurulacaktır. Servis kompozisyonu içindeki servisleri çalıştıran parametreler kompozisyon içine gömülerek, bu detaylar kullanıcıdan istenmeden daha üst düzey bir kompozisyon akışı oluşturularak kullanıcının yükü hafifletilecektir.

Çalışma kapsamında çözüm bekleyen konulardan biri yarı otomatik ontoloji eşleştirme problemidir. Ontoloji eşleştirme elle gerçekleştirilmiştir. Semantik tanımlar oluşturulmuş, ancak kural tabanlı dillerle bir sorgulama yapılmamıştır.

1.5.2. OWL-S Tabanlı Dinamik Servis Kompozisyonu

Dong vd., (2006) çalışmasında dinamik WSK gerçekleştirimi için OWL-S üst düzey servis ontolojisini genişleten bir yaklaşım gerçekleştirmiştir. Geleneksel yöntemlerde yaygın olan WSK gerçekleştirimi, kompozisyonların statik (off-line) olarak önceden düzenlenmesiyle oluşturulmaktadır. Bu da eş zamanlı uygulamalarda sorun olmaktadır. Şöyle ki, önceden düzenlenmiş bir WSK çalıştırıldığında içerdiği kompoze servislerden herhangi birine ulaşılamayabilir ya da çalıştırılması için gerekli olan parametrelerde değişiklik olabilir. Bu gibi olumsuzlukların oluştuğu durumlarda, mevcut sistemlerle koşum anında WSK'yı durdurarak yeniden düzenleyip tekrar başlatmak mümkün olamamaktadır. Bu da bütünüyle WSK'nın yeniden düzenlenmesine sebep olmaktadır. Çalışmada eş zamanlı işlemler için dinamik WSK'nın önemi vurgulanarak geliştirilen mimariyle bu problemlerin giderilmesi amaçlanmıştır.

Çalışma kapsamında mevcut OWL-S servis ontolojisi soyut (abstract) WSK düzenleme yeteneğiyle genişletilmiştir. Genişletme için üst düzey OWL-S ontolojisine "Instance" isminde yeni bir sınıf eklenmiştir. "Instance" sınıfı içinde aynı işlemi geçekleştiren birden fazla somut servis adayı ve bilgileri yer almaktadır. Bunlara "Instance Pool" denilmektedir. WSK çalıştırıldığında "Instance Pool" içindeki servis adaylarından biri seçilmektedir. Çalışmada soyut servis hiyerarşisi mevcut olan somut servislerin işlevsel (girdi, çıktı, ön koşul, son koşul) ve işlevsel olmayan (kalite) parametrelerine göre sınıflandırılarak oluşturulmuştur.

Mevcut OWL-S ontolojisinde sanal bir soyut servis temsil edilememektedir. Koşum anında bir OWL-S servisi sadece bir somut web servisi ile ilişkilendirilebildiği için tek kompozisyon işlemine sahiptir. Buna karşın, soyut bir servis genellikle benzer somut servisler kümesinden oluştuğu için çoklu kompozisyon işlemine sahiptir. Örneğin, mevcut OWL-S ile oluşturulmuş, içinde "Gazetteer" (yer adları) servisi bulunan bir kompozisyonun koşum anında "Gazetteer" servisine ulaşılamaması durumunda kompozisyon başarılı bir şekilde çalışmayacaktır. Ancak içinde aynı işlemi yapan "Gazetteer" ve "GeoNames" servislerinin olduğu soyut bir servis kompozisyonu çalışmaya devam edecektir. Çünkü "Gazetteer" servisinde meydana gelen olumsuzluk tespit edildiğinde, yerine kullanılabilecek aday servislerden "GeoNames" somut servisi çalıştırılacaktır. Bu yaklaşımla bir soyut servis içinde çoklu iş akışı mantığı içeren atomik, basit ve kompoze servisler tanımlanabilir.

Çalışmada geliştirilen mimariye göre, kullanıcılar amaçlarını (goal) web tabanlı bir "composer" (Kompozisyon Plancısı) ile belirtir. Varsayılan OWL-S ontolojisi otomatik olarak plancıya yüklenir. Kullanıcılar amaçlarını belirtirken kendi özel ontolojilerini de yükleyebilir. "Composer", mevcut olan soyut servislerin ontolojisi ile kullanıcı amacında belirtilen işlevsel ve işlevsel olmayan servis parametrelerine dayanarak kompozisyon işlem modelini üretir. Bunun için öncelikle kullanıcı amacı RDF'ye dönüştürülür ve genişletilmiş OWL-S servis ontolojisi ile birlikte JTP (Java Theorem Prover) (URL-5, 2015) çıkarsama aracına gönderilir. Çıkarsama sonucunda somut servisler yerine soyut servisler üretilir. Sonuç soyut servislerin her biri olası bütün somut servis alternatifleri için "Instance Pool" a sahiptir. "Composer", sonuç OWL-S tanımlarını üreterek kullanıcıya döndürür. Sonuçta oluşturulan servis yeni bir servis ise, tekrar mevcut servis ontolojisine "Standalone Utility" bileşeni ile eklenebilir ve değişiklik tespit edildiğinde dinamik olarak servis ontolojisi güncellenir. Yeni somut servisler tespit edildiğinde ise "Standalone Utility" bileşeni aday somut servislerden oluşan "Instance Pool" u günceller.

Çalışmada belirlenen problemler aşağıda açıklanmıştır:

- Geliştirilen soyut servis kompozisyonu "Tightly Coupled" olarak oluşturulmuştur. Yani WSK içindeki servisler, çalışma sıraları ve yerlerine kullanılabilecek alternatif servisler belirlidir. Kompozisyon önceden belirlenmiş servislere göre oluşturulmuştur. Çalışma her ne kadar eş zamanlı uygulamalar için dinamik WSK'ya dikkat çekse de, koşum anında WSK'daki herhangi bir servis çalışmadığında zaten WSK içinde önceden belirlenmiş servislerden biri seçilecektir. Dolayısıyla kompozisyona dışarıdan anlık müdahale yapılamayacağı için, kopan servis yerine WSK içinde olmayan ama aynı işlevi gören farklı bir servis dinamik olarak eklenemeyecektir. Koşum anında WSK içindeki bir servise ait alternatif servis adaylarından hiçbiri çalışmazsa ne olacaktır?
- Çalışmada aynı soyut servise ait bütün somut servislerin aynı ara yüze sahip olduğu yani aynı servis parametreleriyle çalıştığı varsayılmıştır. Dolayısıyla sistem aynı işlemi yapan fakat farklı isimli ve farklı sayıda parametresi olan servislerle çalışmayacaktır.
- Geliştirilen mimaride kullanıcı amacını belirtirken "Composer" da tanımlı ontolojileri kullanır. Composer'da olmayan kendi uygulama alanına özgü bir ontoloji kullanmak istiyorsa bu ontolojiyi yükleyip onun kavramlarıyla sorgusunu oluşturur. Yani kullanıcı belirli kavramlarla soru sorabilir. Başka

kavramlara göre sorgu oluşturduğunda ya da sorgusuna orda olmayan bir kalite parametresi eklemek istediğinde sorgu çalışmayacaktır. Çünkü çalışmada anlık semantik eşleştirme gerçekleştirilmemiştir. Var olan ontolojiler arasında önceden belirlenmiş bir ilişki varsa ilgili kavramlar benzer olarak bulunmaktadır. Kullanıcı sorgusunda yeni bir ontoloji eklediği durumda, var olan ontolojilerle arasındaki ilişki daha önceden belirli olmadığı için anlık olarak hesaplanması gerekmektedir. Ama çalışmada bu tarz bir eşleştirme sistemine değinilmemiştir.

- Kullanıcının sorgusunu servis parametrelerine göre belirtmesi, her servisin nasıl ve hangi parametrelerle çalıştığı hakkında donanımlı olmasını gerektirir. Bu da her zaman mümkün olamayan bir durumdur.

Bu çalışma kullanıcı amacından semantik çıkarsama ile SWSK'nın OWL-S servis ontolojisine dayalı olarak otomatik buldurulması, WSK'nın "tightly coupled" olarak oluşturulması yönleriyle bu tez çalışmasında önerilen mimariye benzerdir. Fakat bu tez çalışmasında farklı olarak, WSK oluşturulurken sadece sıralı kompozisyonun yanında seçim, koşul ve iterasyon gibi diğer OWL-S kontrol parametrelerinin de kullanılmasıyla daha çok amaçlı bir kompozisyon önerilmiştir. Ayrıca bu çalışmada WSK'daki her servisin kaç defa çalıştırılacağı önceden bellidir. Fakat bu tez çalışmasında önerilen WSK mimarisinde bir WS'nin kaç defa çalıştırılacağı önceden belli değildir ve kullanıcının girdiği girdi katman sayısına göre koşum anında değişir. Girilen katman sayısı kadar aynı işlem gerçekleştirilir.

1.5.3. Semantik Web Servisleri Kompozisyonunu Bulmak İçin Bir Prototip

Brogi vd., (2006) çalışmasında istenen bir WSK'nın bulunması için tam otomatik bir eşleştirme sistemi geliştirmiştir. Servis eşleştirme için geleneksel çalışmalardan farklı olarak, her servisin farklı ontolojileri kullanabileceği gerçeğinden yola çıkıp, farklı ontolojilerin kullanılmasını örneklendirmiştir. Çalışmanın diğer bir farkı ise, sorgudan bağımsız olan işlemlerin "off-line" olarak önceden gerçekleştirilmesi ile servis bulma aşamasının verimliliğini artırmaktır. Bu işlemler web servisleri ve servislerin kullandığı ontolojiler arasındaki bağımlılıkların ve benzerliklerin hesaplanmasıdır.

Çalışmada kullanılan web servisleri ve ilgili ontolojileri "Hypergraph" veri yapısında temsil edilmektedir. "Hypergraph", nokta (vertice) ve kenarlardan (hyperedges) oluşan bir çizge veri yapısıdır. Noktalar, yayınlanan servis tanımlarının kullandığı ontolojinin

kavramlarını, kenarlar ise bu kavramlar arasındaki ilişkileri temsil etmektedir. Bu ilişkiler; Alt Kavram (SubConceptOf), Eşit Kavram (EquivalentConceptOf) ve Servis İçi Bağımlılığı (Intra-Service Dependency)' dır. "Servis İçi Bağımlılığı'' ndan kasıt, bir servisin girdi ve çıktı parametresi arasında ontolojide bulunan ilişkidir.

Ontoloji kavramları ve servis parametreleri arasındaki benzerlik ilişkileri ile ontolojiler arası mesafe "SemFit" (Semantic Field) algoritması ile hesaplanmaktadır. "SemFit" benzerlik hesabında "name similarity", "data type similarity", "WordNet similarity", "path similarity", "property similarity", "edit distance similarity" gibi farklı eşleştiricilerin kombinasyonunu kullanmaktadır. Eşleştiricilerle bulunan benzerlik değerleri ile ilgili mesafe formülleri kullanılarak ontolojiler arasındaki mesafe hesaplanmaktadır. Mesafesi en az olan ontolojiler birbiri ile en benzer ontolojilerdir. Literatürde kullanılan eşleştiriciler ilerleyen bölümlerde ayrıntılı bir şekilde açıklanacaktır.

Çalışmada geliştirilen servis bulma mimarisi SAM (Service Aggregation Matchmaking) üç ana bileşenden oluşmaktadır. Çalışma kapsamında geliştirilen "System Interface" ile servis istemci hem sorgu yapabilmekte hem de yeni bir servis yayınlayabilmektedir. "AddService" bileşeni yayınlanan web servislerinin kullandığı ontolojilerin "Hypergraph" a kaydedilmesi ve hesaplanan benzerlik değerlerine göre aralarındaki ilişkilerin oluşturulması işlevini görür. Bu işlem sorgu aşamasından önce yapıldığı için eşleştirme işleminin verimliliğini olumsuz etkilemez. Her yeni bir servis ve ontoloji kaydedildiğinde, daha önce kaydedilmiş olanlar ile arasındaki ilişkiler hesaplanır. İstemci aynı ara yüz aracılığıyla daha önce yayınlanmış servislerin kullandığı ontoloji kavramlarını kullanarak istediği servisin parametrelerini belirtir. "QuerySolver" bileşeni WSK'nın girdi ve çıktı parametrelerinden oluşan istemci sorgusunu alır ve kayıtlı servisler içinden benzerlik değeri en yüksek olanları getirir. Bu benzerlik değerleri yukarıda anlatılan "SemFit" eşleştirme algoritması ile daha önceden hesaplanan değerlerdir. Yayınlanan ontolojiler, servis tanımları ve oluşturulan "Hypergraph" ilişkisel bir veritabanında (Database) depolanır.

Çalışmadaki dezavantajlardan biri istemcinin sorgu oluştururken kısıtlanmasıdır. Şöyle ki, istemci sistem arayüzünde sorgu oluştururken sistem içinde var olan ontolojilere dayanarak aradığı WSK için parametreleri belirtir. Oysa istemci bu parametreleri kendisine sunulan ontoloji kavramları arasında bulamadığı zaman, başka kavramlarla sorgu yapmak isteyecektir. O kavramlarla ilgili ontoloji sistemde olmadığı için istediği şekilde sorgu yapamayacaktır.

Çalışmadaki diğer bir dezavantaj ise, yukarıda anılan benzer çalışmalardaki gibi istemcinin sorgusunu servis parametrelerine göre yapmasıdır. Bu da WSK işleyişi hakkında bilgisi olmayan bir kullanıcı için güçlük oluşturacaktır.

1.5.4. DAGIS: Konumsal Web Servislerini Bulma ve Seçme Çatısı

Alam vd., (2007) çalışmasında konumsal SWS'lerin oluşturulması için DAGIS (Discovery of Annotated Geospatial Information Services) (Referanslandırılmış Konumsal Bilgi Servislerinin Bulunması) bütünleşik mimarisini önermiştir. DAGIS, otomatik servis bulma, çalıştırma ve dinamik servis kompozisyonu için konumsal alan ontolojileriyle ilişkilendirilmiş OWL-S servis ontolojisini kullanır. Mimarinin amacı alan ontolojileri kullanılarak otomatik çıkarsama ile insan müdahalesinin en aza indirilmesidir. Çalışmada üretilen algoritmalarla servislerin hem işlevsel (girdi-çıktı parametreleri) hem de işlevsel olmayan parametreleri (kalite parametreleri) semantik olarak eşleştirilebilmektedir.

DAGIS mimarisi; semantik gösterim bileşeni (presentation layer), semantik arayazılım bileşeni (middleware layer), ontoloji bileşeni (ontology layer) olmak üzere üç ana bileşenden oluşmaktadır. Servis arayan bir kullanıcı Portlet sorgu arayüzünden servis isteğini gönderir. DAGIS agent sorguyu alır ve Matchmaker'a (Eşleştirici) gönderir. Eşleştirici, OWL-S API ile alan ontolojilerine ulaşarak sorgu kavramlarının semantik karşılıklarını belirler. Daha sonra OWL-S katoloğunu sorgulayarak bu kavramlarla eşleşen servis tanımlarını belirler.

Çalışmada önerilen mimari işleyişini sağlayan alt bileşenler aşağıda açıklanmıştır:

- DAGIS Sorgu Tarayıcı Portleti (DAGIS Query Browser Portlet), kullanıcıdan gelen sorguyu özel olarak oluşturulmuş bir java portlet arayüzü ile alır.
- DAGIS Aracı (DAGIS Agent), kullanıcıdan alınan sorguyu otomatik olarak OWL-S diline çevirerek DAGIS Eşleştiricisi (Matchmaker)'ne yollar.
- DAGIS Eşleştirici (DAGIS Matchmaker), katalog'da (Registry) bulunan servis tanımları ile OWL-S'e dönüştürülmüş kullanıcı sorgusu arasında hem işlevsel hem de işlevsel olmayan servis parametrelerine göre semantik eşleştirmeyi gerçekleştirir.

- DAGIS Düzenleyicisi (DAGIS Composer), kullanıcı sorgusuyla eşleşen tek bir servis bulunamadığı durumlarda otomatik olarak dinamik servis zincirini oluşturur. Kompoze edilmiş servis URI'sini tekrar Eşleştirici'ye gönderir.
- OWL-S Servis Kataloğu (OWL-S Registry), Oluşturulan semantik web servisleri katalog işlevini gören Registry'de depolanır.
- WSDL Servis Kataloğu (WSDL Registry), herhangi bir standard UDDI ya da herkese açık web servisleri görevini görür.
- WSDL2OWLS Dönüştürücü (Converter), WSDL'de kodlanmış servis tanımlarını OWL-S'e dönüştürür. Bu işlem için gerekli olan XSLT dönüşümleri manuel olarak yapılmıştır.

Çalışma kapsamında konumsal web servislerini tanımlamak için bir konumsal ontoloji ve servis QoS (kalite) parametrelerini seçme işlemi için QoS ontolojisi geliştirilmiştir. Ontolojilerde kullanılan kavramlar OGC Web Servisleri Spesifikasyonu çatısıyla uyumlu olarak geliştirilmiştir.

Çalışmada kullanılan eşleştirici, Pellet OWL-DL çıkarsamacısını kullanarak servislerin yeteneklerine göre çıkarsama yapabilmektedir. Eşleştirici, OWL-S MX Matchmaker (Klusch vd., 2006) aracının QoS kalite parametrelerine dayalı eşleştirme yapabilme yeteneği ile genişletilerek elde edilmiştir. Kullanıcı sorgusu ile eşleşen birden fazla kayıtlı servis bulunduğu durumlarda, QoS tabanlı otomatik servis seçimi sonuçların elenmesinde önemli rol oynar. Şöyle ki, bulunan benzer servis sonuçlarından kalite parametreleri en yüksek olan seçilir.

Çalışmadaki servis kompozisyonu algoritması Renier vd., (2005)' de önerilen "Recursive Back Chaining" algoritmasına dayanır. Servis zincirini oluşturmak için katalogda benzer servisler tekrarlamalı olarak çağırılır. İstemcinin belirttiği çıktı ile eşit çıktısı olan servisler seçilir.

Çalışmadaki problemlerden biri, uygulama istemcisi ile servis sağlayıcının servis tanımlarının aynı ontoloji ile referanslandırıldıktan sonra eşleştirme işleminin yapılmasıdır. Fakat bu tez çalışmasında önerilen mimaride karşılaştırılan sorgu ve servis ontolojisi ortak bir referans çatıya dayanarak oluşturulmamıştır. Ancak semantik eşleştirme aşamasında arka planda bilgi tabanı olarak kullanılan bir konumsal alan ve servis ontolojisinden yararlanılmıştır.

Çalışma WSK'da yer alacak servisleri ontolojiler desteğinde buldurmakla bu tez çalışmasında önerilen WSK mimarisiyle benzerdir. Farklı olarak, WSK'da yer alacak

servisleri eşleştirirken Lemmens (2006)'deki gibi servis parametrelerini kullanmıştır. Bu da WSK işleyiş akışını bilmeyen bir istemci için güçlük yaratacaktır. Bu nedenle bu tez çalışmasındaki WSK modeli istemciye karmaşık gelebilecek ve vakit alabilecek parametre tayinini zaten WSK içinde kendisine sunmakla daha hızlı ve kolay bir işleyiş sağlayacaktır.

1.5.5. Web Servisi Kompozisyonlarının Davranışı Bilinen Şekilde Bulunması

Brogi ve Corfini (2007), çalışmasında WSK'ların bulunabilmesi için hem semantik hem de davranışsal bilgileri kullanan bir eşleştirme sistemi önermiştir. Çalışmada WSK'ların bulunmasını otomatikleştirmek için hem servis bulma doğruluğunu artıran semantik bilginin, hem de bulunan servislerin doğru bir şekilde kompoze edilmesini sağlayan davranışsal bilgilerin dikkate alınması gerektiği vurgulanmıştır. Bu bağlamda, Brogi vd., (2006, 2007)'de WSK bulunmasına yönelik sunulan algoritmalardaki eksiklikler bu çalışmada giderilmeye çalışılmıştır. Bu eksikliklerden biri performanstır. Bunun sebebi, katalogda her servisin davranışını temsil eden bağımlılık grafiğinin (Dependency Graph) sorgu cevaplama aşamasında oluşturulması zorunluluğudur. İkinci eksiklik ise servislerin koşum anında kopması gibi tıkanıklık durumlarında algoritmanın bunu düzeltme özgürlüğüne sahip olmaması ve bulunan servis kompozisyonlarının minimum şeklinin (isteği tam olarak karşılayacak sayıda servis içeren kompozisyonlar) olmamasıdır. Bu çalışmada bu eksiklikleri gideren bir Petri net tabanlı eşleştirme sistemi sunulmuştur. Bu sistem OWL-S servis tanımlarındaki hem semantik hem de davranışsal bilgileri kullanır.

Önerilen eşleştirme sisteminin temel özellikleri şunlardır:

- Servisleri Petri net'ler olarak modellediği için, sistem büyük ölçüde davranışsal bilgiye dayanır. Petri net'lerin gücü, karmaşık servis kompozisyonlarını kolayca modellemeye izin vermesidir.
- Servislerin Petri net tanımları eşleştirme aşamasının performansını etkilemeden, önceden hesaplanabilir ve servis tanımlarıyla birlikte depolanabilir.
- Petri net'lerin kontrol akışı doğrulaması, kompozisyon içindeki servislerin doğru bir şekilde tamamlanıp tamamlanamayacağını belirlemeyi sağlar.
- Oluşturulan kompozisyon, sorguyu gerçekleştirmek için çok zorunlu olmayan servisleri içermez.

Petri net'ler (Peterson, 1981; Murata, 1989) "Carl Adam Petri" tarafından dağıtık sistemlerin eş zamanlı davranışlarını modellemek için tanıtılmıştır. Petri net noktaları "Places" ve "Transitions" olarak adlandırılan iki tane ve boş olmayan, ayrık setlere ayrılabilen, yönlü, ağırlıklı bir ikili (bipartite) grafiktir. Yönlü ve ağırlıklı çizgiler (kenarlar) aracılığı ile "Places"lar ve "Transitions"lar birbiri ile ilişkilendirilmiştir. Böylece bir çizgi sadece iki noktayı ilişkilendirebilir. "Places" lar daire ile, "Transitions" lar ise diktörtgenlerle grafiksel olarak temsil edilir. Petri net'ler "Places" lar içinde duran nesneler olan ve noktalarla grafik olarak temsil edilen "Tokens" leri tanıtarak bir sistemin dinamik davranışını simule eder. "Places"lar keyfi sayıda token tutabilir. Petri net'ler içinde bir Token, varlığını ya da eksikliğini temsil ettiği şeyin (koşul, kaynak, sinyal vb.) mevcut olduğunu ya da olmadığını gösteren işaretleyicilerdir. İşaretleme (marking), Token'ler kendi dağılımını değiştirdiğinde değişen Petri net'in durumunu temsil eder.

OWL–S kompoze işlemleri direkt olarak Petri net'lerle temsil edilebilir. OWL–S işlem modelini Petri net'lerle temsil etmek için, atomik işlemler "Transitions" olarak düşünülmüş ve işlemler arasında hem veri akışı hem de kontrol akışı ilişkileri "Petri nets Transition firing" kuralları ile modellenmiştir.

Çalışmada geliştirilen eşleştirme sistemi birbirinden bağımsız olan üç modülden oluşur. Bunlar; OWL–S işlem modellerinden Petri nets'lere dönüşüm sağlayan "Translator", servislerin fonksiyonel özniteliklerine göre servisleri filtreleyen "Functional Analyser" ve seçilen Petri net'ler setini birleştirdikten sonra kompoze Petri net'i aktifleştirerek pozitif eşleştirme için kontrol eden "Behavioural Analyser" modülleridir.

Önerilen sistem mimarisinde depolanan her servisin kendi OWL-S tanımını ve Petri net temsilini içerdiği bir katalog geliştirilmiştir. Kataloğa bir servis eklendiğinde, "Translator" bu servisin OWL-S işlem modelini yükler ve onun Petri net temsilini belirli dönüşüm diyagramlarına göre oluşturur. "Functional Analyser" istemci sorgusunu alır, katalogdaki bütün servislerin OWL-S profilini analiz eder ve sorguyu gerçekleştirmek için kullanılabilecek bütün olası servis setlerinin sıralı listelerini döndürür. Eşleştirme sistemi bu servis setlerinin Petri net temsilini, varsa isteği karşılayan bir tanesini bulana kadar analiz eder. "Behavioural Analyser" bir sorguyu bir servis setinin gerçekten karşılayıp karşılamayacağını tespit etmek için önce aday seti içinde bulunan servislerin Petri net'lerini birleştirerek bir araya getirir. Sonra, global Petri net'in kontrol akışı doğrulaması sayesinde set içindeki servislerin gerçekten bir arada kompoze edilip edilemeyeceğini ve tıkanıklık olmadan sorgu çıktılarını üretip üretemeyeceğini belirler. Eğer sonuç olumlu ise,

“Behavioural Analyser” istemciye pozitif bir eşleşme döndürür, aksi takdirde sıradaki servis setini analiz eder.

1.5.6. Doğal Dil İsteklerine Dayalı Anlık Servis Kompozisyonu

Pop vd., (2009) çalışmasında sınırlandırılmamış doğal dil isteklerini kullanarak interaktif servis kompozisyonu yapan bir yöntem önermiştir. Bu yöntem ile doğal dile dayalı bir servis isteğinin anlamı analiz edilerek istenen kavramlara dayanan kompoze bir servis üretilir. Gerçekleştirim için kullanılan her servis, kullanıcı isteğinin parçası olan bir kelimeye linklenir. Bu sistem ‘Ubiquitous’ programlama alanında, özellikle insan merkezli programlamalarda kullanılmak için tasarlanmıştır. Çalışmadaki dinamik servis kompozisyon sistemi akıllı bir evin içinde akıllı aletlerin fonksiyonel konfigürasyonunu oluşturmak için tasarlanmıştır.

Diğer yaklaşımların aksine, önerilen yaklaşım tek kelime kullanan servisleri bulmamakta ve kontrollü doğal dil alt sınıfını kullanmamaktadır. Sadece kullanıcı isteği ile sunulan potansiyel servisler arasındaki mesafeyi minimize etmeye çalışmaktadır. Bu şekilde, her bir istek için kompoze edilebilen bir servis sağlanmış olur. Genellikle semantik biçimleme dar ve önceden tanımlanmış bir kelime hazinesi kullandığı için, kelime hazinesi bilinen web servislerinin bulunmasını sağlar. Önceden belirlenmiş kelime hazinesinden farklı bir kelime hazinesi kullanan servisler ya da kullanıcı istekleri böyle bir servisi bulmak için uygun değildir. Bu yüzden, çalışmada web servislerinin semantik biçimlemesi için dar kelime hazinesini doğal dil istekleriyle birlikte kullanmanın uygun olmadığı vurgulanmıştır. Ayrıca, çalışmada sözlüksel ağaç yapısının servislere çok fazla semantik bilgi ekleyeceği ve iç içe geçmiş aygıtlar için uygun olmayacağı düşünülerek genel kavramların kullanımı önerilmiştir. Çünkü bir servisin bütünüyle tek bir kavramla tanımlanamadığı, fakat bir kavramın genelleştirilmesi ile tanımlandığında sonsuz sayıda kavramla tanımlanabileceği vurgulanmıştır. Örneğin, Televizyon kavramı TV’yi tanımlamak için kullanılmıştır. Genelleştirme ile hem Televizyon hem de elektronik cihaz kavramlarının aynı aygıtı gösterdiği tespit edilmiştir.

Çalışmada dinamik servis kompozisyonu, servis bulma ve servis orkestrasyonu olmak üzere iki operasyon içerir. Servis bulma, kullanıcı isteğine fonksiyonel olarak en benzer servislerin belirlenmesidir. Bunun için istemcinin cihazlarla etkileşimde bulunduğu ve sınırlandırılmamış bir doğal dil cümlesi ile amacını söylediği bir senaryo tasarlanmıştır.

Amaç cümlesinde belirtilen cihazları bulmak için kavramlar kullanılır. Örneğin, kullanıcı isteğinde belirtilen "I want to use my phone to turn off the light, turn on the TV and play some music on HiFi" cümlesi ihtiyacı karşılayan servislerin bulunması için dilbilimsel işleme modülüne gönderilir. Bu modül, cümle içindeki kelimeleri bölümler (Text Segmentasyon). Alakasız olan bağlaçları (the, to, and) kaldırarak kelimeleri kök haline getirir. Yanlış yazılan ve kökleştirme aşamasında zarar gören kelimeleri düzeltmek için yazım denetimi yapar. Böylece kelime işleme modülünden çıkan kullanıcı isteğindeki kelimeler "want, use, phone, switch, light, turn, tv, play, music, hifi" haline gelir. Elde edilen kelime parçaları servis tanımları ile birlikte kavramsal grafik içindeki noktaları oluşturmak için kullanılır. Bu grafikteki her çizgi bir kelime parçasını her bir servis tanımı ile ilişkilendirir. Kavramlar sözlüksel ağaçta bulunan noktalardır ve genelleştirilerek oluşturulmuştur. Her kenarın ağırlığı kelime parçası ve servis tanımı arasındaki kavramsal mesafeyi temsil eder. Kavramsal mesafe hesaplayan algoritma WordNet (URL-6, 2011) sözlüğünü kavramsal hiyerarşileri oluşturmak için kullanır. Bir kavram hiyerarşisi oluşturulurken önce hiyerarşinin oluşturulacağı kavramı içeren synset[1] bulunur. Synset içindeki her kelime kavram hiyerarşisindeki ağaç için bir kök nokta olur. Her kök nokta için, kök synset'ler ile ilişkili olan synset'ler bulunur. İlişkili synset'deki her kelime hiyerarşide sıradaki alt hiyerarşi için başlangıç kök kavramı, yani noktası olur. Hiyerarşide aynı düzeyde bulunan her kelime için, o kelimenin synset'i ile ilişkili olan synset'ler bulunur ve kelimeleri bulunan synset'ler içine bir sonraki seviyedeki ağaç için nokta olarak eklenir. Bu işlemler hiyerarşi yeterince büyük olana kadar tekrar edilir. Böylece hiyerarşinin oluşturulduğu kavram için genelleştirme, en iyi sonuçları üreterek kabul edilmiş bir doğruluğa karşılık gelir.

Servis bulma aşaması, kullanıcı isteğine en yakın servisleri belirleyerek kavramsal grafikte kavramsal mesafeleri toplamı en küçük olan çiftleri (kelime parçası, servis tanımı) belirleme aşamasıdır. Bu çiftleri bulmak için kavramsal grafiğe iki dönüşüm uygulanır. Birincisi bütün noktaları içeren alt grafiğin hiyerarşi mesafesine en yakın mesafesi olanların bulunmasıdır. Bu dönüşümden sonra her servis tanımı iki kelime parçası ile ilişkilendirilir. İkincisi ise, her üçlü için (servis tanımı, 1.kelime parçası, 2.kelime parçası) maksimum ağırlığın olduğu kenarın kaldırılmasıdır. Çalışmada en küçük ağırlıklı alt grafiği bulmak için Kruskal (1956) algoritması kullanılmıştır. Bu algoritma minimum genişlikteki ağacı (Minimum Spanning Tree) hesaplar. Anılan iki dönüşüm uygulandıktan

[1] WordNet literatüründe "synset", eş anlamlı kelimeler grubunu ifade eder.

sonra oluşmuş servis tanımlarını içeren noktalar, kullanıcı tarafından istenen yüksek düzeydeki servisi orkestra etmek için kullanılan servisleri temsil eder.

Çalışmada geliştirilen tasarım, akıllı cihazlar tarafından sağlanmış servisleri kompoze eden akıllı ev senaryosuna dayanır. Senaryodaki her akıllı cihaz iletişim yeteneğine sahip ve kullanıcıya bir ya da daha çok servis sağlayan bir varlık olarak düşünülmüştür. Akıllı cihazlar akıllı bir evin içinde bilgi alış verişini sağlayan dinamik bir ağ oluşturmak için ilişkilendirilmiştir. Dinamik ağ kavramı cihazların ağ yapısından eş zamanlı olarak ortaya çıkıp tekrar kaybolabileceği, bir cihazın ağa katıldığı zaman otomatik olarak adapte olabileceği ve sonra ağı habersizce terk edebileceğini tanımlar. Bütün bu özellikler aygıtları kontrol etmek için kullanılan bir ara yazılımla sağlanır. Bu ara yazılım adaptasyon için destek sağlayan 'WComp' denilen bir 'ubiquitous' programlama ara yazılımıdır ve 'UPnP' protokolünün kullanımıyla aygıt birlikte işlerliğini sağlar. Akıllı ev sisteminde istemcinin kullanmak istediği cihazlardan biri yoksa veya güncel bir versiyonu ile yer değiştirdiyse, sistem buna adapte olacak ve kullanıcı ihtiyacına en yakın servisi onun yerine monte edecektir.

Çalışmada servis orkestrasyonu, kullanıcı isteğinin gerçekleştirilmesi için bulunan servisleri fonksiyonel bir iş akışı içine linklemek için oluşturulmuştur. Bulunan servisler önceden tanımlanmış 'Aspects of Assembly' (AoA) (Cheung vd., 2007) olarak adlandırılan modellere dayalı olarak kompoze edilir. Çalışmada değişik bileşen setlerinin seçimini sağladığı ve karmaşık etkileşimlerde bulunabildiği için model (template) tabanlı servis kompozisyon sistemi kullanılmıştır. Bu sistem 'ubiquitous' programlama için kullanılan WComp (Cheung vd., 2006) ara yazılımının bir parçasıdır. Model tabanlı olmasının yararı otomatik adaptasyon sağlamasıdır. Bu modeller kullanıcı isteği gerçekleştirilirken ya servis kompozisyon sistemi ya da kendi kendine adapte olan işlem içindeki ortam değişimi ile hareketlenir ve yüksek düzey tanımlamaların mantıksal birleşimi ile kompoze edilir. AoA mimarisi mantıksal birleşim ve adaptasyon için durum yönelimli programlamanın (Aspect Oriented Programming) genişletilmiş bir modelinden oluşur. AoA modeli, kompozisyona katılan bileşenler listesi ve bu alana özgü bir dil ile (domain specific language) kodlanmış adaptasyon tanımı ile oluşturulmuştur. Her UPnP aygıtı yazılım bileşeni gibi davranan bir yazılım proxy'sine sahiptir ve cihazların servislerini meydana çıkarır. UPnP servisine her bir cihaz için semantik bir tanım olan metaveri eklenmiştir. Bu bileşenler arasındaki etkileşim AoA modelleri kullanılarak oluşturulmuştur. Kullanıcıdan gelen istekler bileşenlerin WComp derleyicisiyle alınır ve UPnP protokolü kullanılarak cihazların

konumlandırıldığı durumun tanımı ile birlikte servis tasarımcısına gönderilir. Servis tasarımcısı sadece aygıtları, servis tanımlarını, semantik metaveri için sorguları ve kullanıcı isteğine yakın olan servisleri bulur. İlgili servisleri sağlayan cihaz örnekleri AoA modelleri içinde tanımlanan kurallara dayalı olarak birbirine bağlanır. İstekle eşleşen servisler tespit edildiğinde, bazı durum yönelimli tanımlamalar servisleri birbirine bağlamak için kullanılır. Bu sayede sistemden bir servis ayrıldığında ya da yeni bir servis meydana çıktığında, servis konfigürasyonu buna adapte olur.

Bu çalışma doğal dil isteğiyle tanımlanmış bir amaç cümlesinden ilgili web servislerinin bir kavram hiyerarşisine dayanarak buldurulması ile bu tez çalışmasında önerilen mimariye benzerdir.

1.5.7. Konumsal Detayların WFS ve Konumsal Semanik Web Kullanılarak Mantık Tabanlı Bulunması ve Entegrasyonu

Zhang vd., (2010) çalışmasında konumsal detayların otomatik olarak bulunması ve entegrasyonunu kolaylaştırmak için var olan OGC WFS'leri konumsal semantik web teknolojileriyle genişletmiştir. DL çıkarsamacısı, çıkarsama kuralları ve OGC WFS tanımları için OWL ontolojilerinin kullanılması ile farklı coğrafi bilgi sistemlerinin konumsal detayları semantik olarak paylaşması ve bütünleştirmesi sağlanmıştır. Detay tabanlı bir konumsal veri paylaşım çatısı önerilmiştir.

Çalışmada heterojen veritabanlarında bulunan konumsal detayları Web'de yayınlayan OGC WFS'ler kullanılmıştır. Çalışmada WFS filtre operatörlerinde tanımlanan 'equals' (eşit), 'disjoint' (ayrık), 'intersects' (kesişir), 'touches' (dokunur), 'contains' (içerir), 'crosses' (keser) gibi konumsal ilişkilerin çıkarsaması için bir DL çıkarsamacısı genişletilerek kullanılmıştır. İlişkileri çıkarsayabilmek ve OWL bilgi tabanından başka sorular yapabilmek için bilgi tabanındaki aksiyomlara çeşitli çıkarsama kuralları tanımlanarak uygulanmıştır. Örneğin, bilgi tabanında bulunan "A otobüs durağı B yolu üzerindedir (touches)", "B yolu Waukeshen'dedir (located)" aksiyomundan konumsal ilişki çıkarsamasıyla otobüs durağı A'nın da Waukeshen üzerinde bulunduğu anlaşılır. Genişletilmiş DL yapısı için RCC8 ilişkileri (Randell vd., 1992) kullanılmıştır.

Çalışma kapsamında genel coğrafi detaylar için bir alan ontolojisi, otobüs yolu ve otobüs durakları konumsal verileri için bir uygulama ontolojisi OWL dilinde oluşturulmuştur.

Çalışmada bir detay web servisi bulma aşaması 3 şekilde gerçekleşir:

1. WFS servislerince sunulan her detay, alan ve uygulama ontolojileri ile ilişkilendirilir. İlişkilendirme WFS servislerinin 'Get Capabilities', 'DescribeFeatureType' ve 'GetFeature' operasyonları içindeki her detay tipi ve ilişkilerinin ontolojiler ile eşleştirilmesi ile gerçekleştirilir. Böylece semantik tabanlı servisler oluşturulur. Detaylarla eşleşen ontoloji sınıflarını bulmak için "Breadth-First" arama algoritması kullanılmıştır. "Breadth-First"[2] algoritmasıyla ontoloji ters çevrilerek en alt noktadan en baştaki noktaya doğru arama yapılır. Yani, önce ontolojinin alt sınıfları sonra üst sınıfları aranır. Böylece arama az genel olandan daha genele doğru sürdürüldüğü için, daha hassas şekilde eşleşebilecek sınıflar ilk önce test edilmiş olur.
2. Kullanıcı sorgusu da alan ve uygulama ontolojileri ile ilişkilendirilerek sorgu tanımları üretilir. Servis bulma aracı kullanıcı sorgusunu; detay tipi isimleri, ilişki isimleri, detayların geometri tipleri ve sınırlayıcı dörtgen geometrisi olmak üzere dört parçaya ayırarak değerlendirir.
3. Servis bulma aracı eşleştirme algoritmalarını kullanılarak oluşturulan sorgu tanımları ile servis sağlayıcıların servis tanımlarını eşleştirerek uygun WFS'leri bulur. Bulma aşamasında otomatik konumsal eşleştirme aracı bilgi tabanı olarak DL tabanlı bir çıkarsamacı ve çıkarsama kuralları kullanarak konumsal detayları otomatik olarak bulur. Algoritma önce sorgudaki sınırlayıcı dörtgen bilgisini repository'de bulunan servis sayısını daraltmak için kullanır. Böylece servisler sınırlayıcı dörtgen ile eşleşen geometriye sahip olanlar kalacak şekilde elenir. Daha sonra kalan servisler detayların geometri tiplerine, detay tipi isimlerine ve son olarak da öznitelik isimlerine göre elenir. Sonuçta kalan ilişkili servisler sorgu tanımları ile karşılaştırılarak aralarındaki benzerlik değerleri 'eşit' ya da 'kapsar' şeklinde bulunur.

Kullanıcı amacını tamamen karşılayan WFS'ler bulunamaması durumunda, mevcut kısmi eşleşen WFS'ler belirli bir sıraya göre dizilmek suretiyle birleştirilip servis kompozisyonu oluşturulur. WSK için servis bulma aşaması yukarıda anlatılan tek WFS bulma aşaması ile aynıdır. Servis bulma aracı detay ve geometri tipi sorguda belirtilen detay ve geometri tipi ile eşit ya da ontolojide alt sınıfına karşılık gelen WFS'leri getirir. WFS'leri, konumsal detay ilişkileri sorgudaki ilişkilere eşit ya da ilişkisel olarak alt sınıfı

[2] http://en.wikipedia.org/wiki/Breadth-first_search

(subProperty) olanlar kalacak şekilde filtreler. Filtreleme sonucunda kalan servisleri Sınırlayıcı Dörtgen (Bbox) boyutu sorgudaki Bbox ile kapsananlar kalacak şekilde tekrar filtreler. Son kalan servislerden WSK oluşturulur.

Çalışmadaki problemlerden biri servis bulma aşamasından önce hem WFS'lerce sunulan hem de sorguda belirtilen konumsal detay tipleri ve ilişkilerinin konumsal alan ve uygulama ontolojileriyle manuel olarak ilişkilendirilmesidir. Hem servisler hem de sorgu ortak tek bir semantik çatı altında referanslandırılmıştır. Ayrıca bu işlemin manuel olarak yapılması anlık servis bulma ve anlık WSK oluşturma için dezavantajlı bir durumdur.

Çalışmada WSK oluşumunda kompoze servisler bulunduktan sonra birbiri ile uyumu hakkında bir çalışma yapılmamıştır. Bu da ard arda dizilecek servis bileşenlerinin hangi sırada olacağı ve hangi parametreleri birbirine aktaracağını belirsiz bırakmıştır. Dinamik bir ortamda böyle bir durum da ikinci bir dezavantajdır.

Çalışma WSK oluşumunda konumsal ontolojiler ve OGC filtre operasyonlarından yararlanılmasıyla bu tez çalışmasında önerilen mimariye benzerdir. Fakat bu çalışmada OGC filtre operasyonları ontolojideki konumsal ilişkilere karşılık gelecek şekilde kullanılmıştır. Bu tez çalışmasında ise ontolojideki her bir konumsal operasyon sınıfına karşılık gelmektedir.

Bu çalışma, tek tip servislerin (WFS) bulunması ve kompozisyonuna bir örnektir. Fakat bu tez çalışmasında önerilen mimari ile farklı tipteki servislerin de kompozisyonu amaçlanmıştır. Şöyle ki, ilgili katmanları (temaları) direkt ya da belirli ölçütlere göre filtreleyerek elde etmek için WFS'lerin, bu katmanlara uygulanacak konumsal operasyonlar için WPS (Web Processing Service)' lerin de kompozisyon içine dahil edilmesi sağlanmaktadır.

1.5.8. Taverna'da Semantik İş Akışı Oluşturma: SADI ve BioMoby Eklentileri

Withers vd., (2010) çalışmasında Taverna iş akışı sistemine semantik servis bulma ve kompoze etme işlevlerini sağlayan SADI (Wilkinson vd., 2009) ve BioMoby (The BioMoby Consortium, 2008) eklentilerinin katkısını değerlendirerek karşılaştırma yapmıştır. BioMoby ve SADI platformu, web servisi iş akışları ile son kullanıcılarına bioinformatik destek sağlayan "Genome Canada Bioinformatik Platformu"nun parçasıdır. Platform, güçlü ve anlaşılır arayüzü sayesinde temel son kullanıcı araçlarından biri olarak Taverna istemci uygulamasını seçmiştir. Fakat Taverna arayüzünde web servislerinin fazla

sayıda olması iş akışı oluşturmayı uzman son kullanıcılar için bile zorlaştırmaktadır. Bu yüzden çalışmada son kullanıcılar için hem servis bulma aşamasında hem de servislerin iş akışları içine doğru bir şekilde bağlanmasında BioMoby ve SADI servisleriyle çalışması için Taverna'ya eklentiler oluşturmuşlardır. Çalışmada bu eklentilerin web servisleri işlevlerinin ve amacının daha anlaşılır olmasını sağladığı ve böylece iş akışı oluşturulmasında servis bulmanın kolaylaştırılması için veri yapılarına semantiklik kazandırdığı vurgulanmıştır.

İstenen servislerin ya da iş akışı bileşenlerinin manuel olarak bulunması ve iş akışı içine hem sözdizimsel hem de semantik olarak doğru bir şekilde entegre edilmesindeki zorluklar sebebiyle son kullanıcıların mevcut belirli şablonlar sunulsa bile kullanışlı ve işlevsel iş akışlarını oluşturmada sorun yaşadıkları görülmüştür. Taverna genel amaçlı iş akışı modelleme aracıdır ve herhangi bir araştırma alanıyla ilgili veri akışı yapabilmektedir. Fakat Taverna son kullanıcıları iş akışı oluştururken birçok seçenek arasından istedikleri servisi seçme konusunda sıkıntı yaşamaktadır. Bu yüzden bu çalışmada Taverna'ya sağladığı kataloglarla ilave semantik arama desteği sağlayan BioMoby ve SADI eklentilerinin karşılaştırılması yapılmıştır.

Moby'de servis bulma; servislerin ontoloji tabanlı girdi ve çıktı veri tipi, servis operasyonlarının kontrollü sözlüğü ve servis sağlayıcı kimlik tanımı ile servislerin indekslendiği merkezi bir kataloğu (Moby Central) sorgulamadır. Ancak burada servis girdi ve çıktı parametreleri XML şemaya referans vererek tanımlandığı için verinin altyapısını iyi bir şekilde oluşturamaz. Veri ve servis çıktıları BioMoby veri tipi ontolojisi ile oluşturulduğu için katalog araması belirli bir veri tipini girdi olarak kullanan servisler içindir. Ya da ters yönde iş akışı oluşturuluyorsa, belirli veri tipini çıktı olarak üreten servisler içindir. Aynı zamanda katalog servislerinde belirli veri tipinin ontolojik olarak üst sınıf tipini girdi olarak kullanan servisler de aranarak arama zenginleştirilebilir. Sonuç eşleşmeler servis ismi, servis tipi veya servis sağlayıcısına (URI) göre sıralanır. Servis tipinden kasıt bioinformatik operasyon tiplerini tanımlayan kontrollü sözlük yapısıdır. BioMoby servis operasyonları bioinformatik servis tiplerinin basit hiyerarşisinden oluşmaktadır. İstenen servis seçildiğinde Taverna eklentisi otomatik olarak onu iş akışına bağlar. Biomoby veri tiplemesi sayesinde verinin bir servisten diğer servise veri tipleri ontolojik olarak ilişkili olduğu sürece tam anlamıyla aktarılması sağlanır. BioMoby projesinin başarılarından ve başarısızlıklarından yola çıkarak yeni bir SWS çatısı olan SADI (Semantic Automated Discovery and Integration) geliştirilmiştir.

SADI, web servisleri arasındaki birlikte işlerliği artıran ve kaynak sağlayıcılar tarafından servis sağlamanın karmaşıklığını azaltan bir SWS oluşturma mimarisidir. Java'da geliştirilen SADI, web servisleri semantik tanımlamaları için Perl (URL-7, 2015) dilini kullanır. SADI'de servis bulma, belirli veri ilişkileri seti kullanan servisleri aramaya ve o ilişkilere dayanarak istenen bir ya da daha fazla yeni ilişki üretmeye dayanır. Arama az da olsa belirli çıktı parametresinin ontolojideki sınıfına dayanarak yapılır, fakat çoğunlukla girdi verisi ile ilişkili olan belirli veri ilişki setleri için yapılır.

BioMoby, semantik mesajları temsil etmek için ontolojinin XML'de serileşmiş halini kullanmasına rağmen, SADI ise RDF, OWL ve bunların XML'de serileştirilmiş halini kullanır. SADI'de servis girdi ve çıktıları, OWL-DL sınıfları ve bu sınıfların RDF'de kodlanmış OWL örnekleri (Individual) olarak kullanılmıştır. SADI projesindeki püf nokta girdi ve çıktı RDF noktalarının kimliğinin eşit olması zorunluluğudur. Bunun sonucu olarak her servis, girdi noktasının servisin çalıştırılması sonucu üretilen yeni veri noktalarına etiketlenmiş RDF ilişkileri ile donatıldığı bir referans (bilgi) servisi olur. Buna ilave olarak, servislerin girdi ve çıktı sınıfları OWL-DL'de tanımlandığı için bunların OWL sınıf tanımları arasındaki farkı inceleyerek (bunlar aynı varlıkla (URI) tanımlandığı için) bir girdi noktasına hangi ilişkilerin ekleneceği basitçe belirlenebilir. Bu ilişkiler katalog'da indekslenir ve servis bulma için kullanılır. BioMoby kataloğu sadece veri tipini desteklerken, SADI'de servis tarafından girdi parametresine eklenen ilişkiler katalogda indekslenir. Çünkü bunlar çıktı veri tipi tanımının bir parçasıdır. Böylece SADI'de mevcut servislerin çıktısına dayalı daha ayrıntılı veri aramaları yapılabilir. Örneğin, BioMoby'de girdi veri tipi "gen ismi" olan ve çıktı veri tipi olarak "nükleotit dizisi" üreten bir servis için arama yapılabilir. Fakat bu şekilde arama ile gen ile dizi arasındaki ilişkinin tam olarak ne olduğu belirsiz kalır. Mesela, o bir gen dizisi mi? kodlama dizisi mi? ya da o geni içeren bir DNA serisi mi? Buna karşın, SADI'de gen isimlerinde "KodlamaDizisineSahiptir" ("hasCodingSequence") ilişkisini sağlayan servisler aranabilir. Böylece servisin ne yaptığı hakkındaki belirsizliklerin çoğu giderilmiş olur.

BioMoby ile katalogda anlaşılması güç veri tipi aranırken, bir SADI aramasında kataloğa gönderilen sorgu OWL sınıf ismidir. Sorgu sonucunda geçerli bütün servisler metaverisi (ismi, operasyon tanımı) ve girdi verisine ekleyeceği ilişkiler (property) ile birlikte sergilenir. Burada son kullanıcıya sunulan bu ilişkinin ismidir. SADI kataloğuna erişen arama arayüzü, çıktı verisinin alt sınıfı olan ilişkileri kullanan servisleri bulmak için ilişki kısıtlamalarını geçerli OWL sınıf tanımı içinde kullanabilir. Böylece SADI araması

BioMoby aramasından semantik olarak daha zengin olur. Eşleştirme, eldeki ilişkileri katalogdaki her servisin girdisi olan OWL ilişki kısıtlamaları ile karşılaştıran bir DL çıkarsamacısı tarafından yapılır. Servisler bulunduğu zaman insan müdahalesi olmadan hem sözdizimsel hem de semantik doğrulukla birbiri ile bağlanır. Her SADI servisi, en üstte girdi veri tipi (OWL sınıf ismi), ortada servis ismi, altta ilişkiler listesi ve ilişkilerin alabileceği değerler listesinden oluşur.

Bu çalışmada servis girdi ve çıktı parametrelerinin aynı ontolojilerle tanımlanması, farklı ontoloji kavramlarıyla tanımlanan servislerin bulunamaması nedeniyle problem oluşturacaktır. Ayrıca, SADI ile web servisleri ilişki bazında aranmaktadır. Ancak aynı ilişkiyi farklı işlev gerçekleştirmek için kullanan servisler olabileceği için, bu durum semantik açıdan servisleri ayırt edici bir özellik değildir. Bu da diğer bir problemdir.

Bu çalışma servis bulma ve WSK oluşturma aşamasında belirli bir alana özgü servis operasyonlarının sınıflamasından yararlanması ve Taverna aracını kullanması ile bu tez çalışmasında önerilen mimari ile kesişir. Ancak, bu çalışmada her ne kadar son kullanıcıların zorluklarını gidermek amaçlansa da, eldeki bir veri ile başladıktan sonra "bu tip veriye ne tür operasyonlar uygulanabilir" bilgisinin kullanıcının hangi veriye hangi operasyonları uygulayacağını bilmesini gerektirdiği için, yük oluşturmaktadır. Bu da ancak alanında uzman kullanıcıların yapabileceği bir iştir.

Bu tez çalışmasında ise, son kullanıcıların anılan servis parametrelerini bilmediği durumlar göz önüne alınarak, sadece servis ismine dayalı servis bulma ile akış parametrelerini kompozisyon içine dahil ederek daha kolaylaştırıcı bir çözüm önerilmiştir.

1.5.9. Veri Madenciliği İçin Akıllı Destek

Bu tezde önerilen yaklaşımla önemli benzerlikler gösteren başka önemli bir çalışma Kietz vd., (2014) çalışmasıdır. "e-lico" isimli, AB destekli bir proje olan bu çalışmada birçok yazılım bileşeni geliştirilmiştir. Bunlardan biri "eProPlan" dır. eProPlan, DM (Data Mining) servis sağlayıcılarının servislerini semantik olarak tanımlama aracıdır. Diğer yazılım bileşeni IDA (Intelligent Discovery Assistant) dır. IDA, RapidMiner ve Taverna (Oinn vd., 2004) yeteneklerini geliştirerek, bir dizi semantik servis kompozisyonunun otomatik olarak elde edilmesini sağlar. Bunun için gerekli olan başlangıç verisi, veri seti ve eldeki DM problemidir. IDA'nın bulduğu servis kompozisyonu alternatiflerinden en uygununu seçebilmek için de bir "önerici" bileşen (Serban, 2013) geliştirilmiştir.

Bu tezdeki yaklaşımın en önemli argümanı durumunda olan, kullanıcının doğru iş akışını bulmadaki zorluğunun hafifletilmesi, söz konusu bu çalışmada da en önemli argümandır. Farklı olarak, bu çalışmada "veri madenciliği" bağlamında sorun ele alınmıştır. Bu tez çalışmasındaki yaklaşımda ise, bir KDD iş akışının temel bileşenleri olan veri, algoritmalar ve "amaç" lar ile RapidMiner operasyonlarının temel bileşenleri referanslandırılmıştır. RapidMiner (Mierswa vd., 2006) en çok kullanılan açık kaynak kodlu veri madenciliği yazılımlarından biridir. Çalışmada ayrıca, bir Yapay Zeka Otomatik Planlama (YZOP) Yaklaşımı (Hierarchical Task Network (HTN) Planning) (Erol, 1996) kullanılmıştır.

Bu çalışmada semantik olarak referanslandırılmış olan elemanlar veri, operatörler, modeller, veri madenciliği görevleri (task) ve Bilgi Arama Veri tabanı iş akışlarıdır. Bu işlem için eProPlan modelleme aracı kullanılmıştır. Mimarinin içerdiği ontolojiler DMWF-HTN (Data Mining WorkFlow HTN) ontolojisi, "RapidI generated" ontoloji, "RapidI" ontoloji, DMWF ontolojisi ve "Base" ontolojidir. Bu ontolojiler yazılan hiyerarşik sırada birbirini kullanır ve en üstteki sınıf DMWF-HTN'dir. DMWF, Rapidminer'dan 100'den çok operatör içerir (Kietz, vd., 2012).

1.6. Semantik Web Servisleri İçin Şema Eşleştirme

Bu bölümde sırasıyla Semantik Web Servisleri (SWS) ve Şema eşleştirme konularında bilgiler verilecektir.

1.6.1. Semantik Web Servisleri

Farklı programlama dilleri kullanılarak geliştirilen, ağ üzerinde faklı yerlerde bulunan ve farklı platformlara sahip bilgisayarlar üzerinde koşan uygulamaların, belirli görevleri yerine getirmek için, birlikte işleyebilmelerini sağlayan çeşitli sistemler ve yazılım mimarileri geliştirilmiştir. Bu konuda yaygın olan geleneksel yazılım mimarisi, Servis Yönelimli Mimari (Service OrientedArchitecture) ya da kısaca SYM (SOA) olarak adlandırılmaktadır (Akıncı, 2006). Web servisleri, SYM'yi gerçekleştirmenin en iyi ve en popüler yolu olarak kabul edilmektedir (McGovern vd., 2003; Colan, 2004; Weerawarana vd., 2005). W3C (2002), bir Web servisini, İnternet tabanlı protokoller aracılığıyla XML

tabanlı mesajları kullanarak diğer yazılım uygulamaları ile doğrudan etkileşimleri destekleyen, arayüzleri ve bağlantıları XML tabanlı diller kullanılarak tanımlanabilen ve bulunabilen ve bir URI (Uniform Resource Identifier) tarafından tanımlanan bir yazılım uygulaması olarak tanımlamaktadır. Üst düzey bir görüşle bir Web servisi, belirli bir görevi gerçekleştirmek için internet üzerinden çağrılabilen bir uygulama olarak tanımlanabilir (Akıncı, 2006).

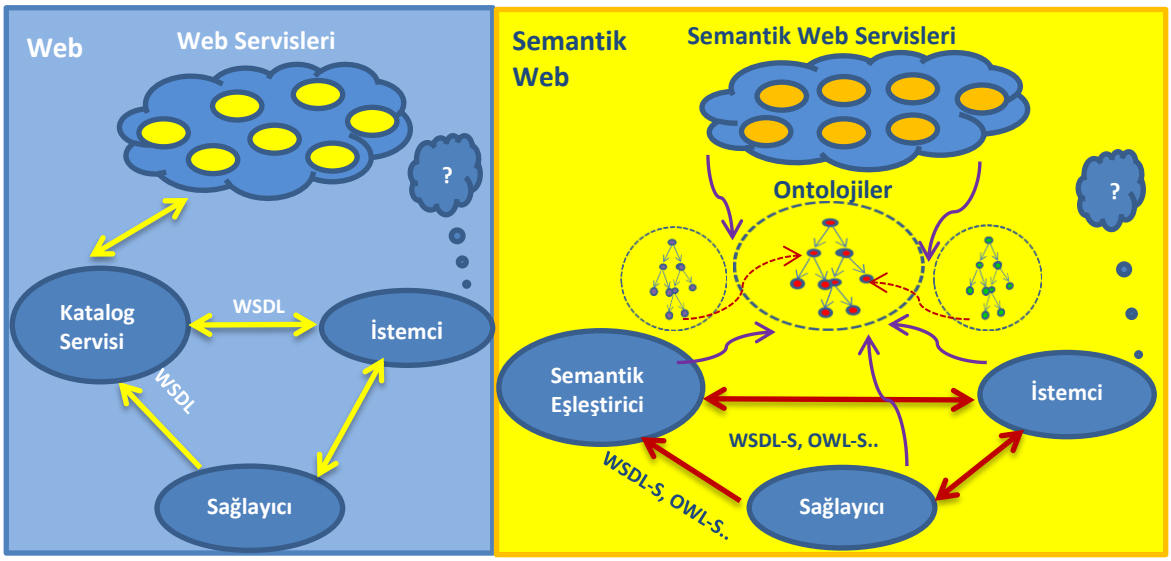
Geleneksel SYM'de, servis sağlayıcılar sahip oldukları Web servislerini katalog servisleri aracılığıyla yayınlarlar. İstemciler ise, katalog servislerinde arama yaparak ihtiyaç duydukları Web servislerini bulurlar. Geleneksel servis katalogları sadece anahtar kelime tabanlı servis aramaya olanak tanımaktadır. Dolayısıyla bir istemci, ihtiyaç duyduğu bir Web servisini bulmak için bir katalog servisine sadece anahtar sözcük tabanlı sorgu gönderebilir. Bu şekilde ancak söz dizimi aynı olan servisler bulunabilmekte, farklı söz dizimde yazılmış metaveri, operasyon ve parametreleri olan servisler aynı işlemi yapsa dahi bulunamamaktadır.

Semantik Web Teknolojileriyle (SWT) geleneksel SYM'nin daha hızlı ve daha doğru bir şekilde başarılması amaçlanmıştır. Ontolojiler desteğiyle semantik referanslandırmalar altında birlikte işlerlik sorunları çözülerek, servis bulma ve çalıştırma, WSK gibi Web servislerine özgü işlemlerin otomatikleştirilmesi amaçlanmaktadır. SWT ile Web'de farklı sunucularda bulunan ve farklı söz dizimde metaveri, operasyon ve parametrelere sahip ancak aynı işlevi gören servisler bulunabilecektir. Semantik Web, bilginin iyi tanımlanmış anlamıyla birlikte verilmesiyle bilgisayarlar ve insanların birlikte çalışmasını kolaylaştıran ve bilgisayarların birbirinin dilinden anlamasını sağlayan şuanki Web'in uzantısıdır. "Semantik Web" in temelinde yatan düşünce, web üzerinde bulunan bilgiye ontolojiler yardımıyla semantik açıklama eklemektir. Böylece web dokümanının içeriği bilgisayarlar tarafından okunabilir ve anlaşılabilir olacaktır (Berners Lee vd., 2001). Ontoloji belirli bir ilgi alanındaki kavramları, bu kavramların özelliklerini, birbirleri ile olan ilişkilerini ve kısıtlamalarını ifade eden bir modeldir. Gruber (1993, 1995)' e göre "Ontoloji, paylaşılan bir kavramsallaşmanın biçimsel ve açıkça belirtimidir".

Son zamanların en aktif konularından biri Semantik Web Servisleri'dir (SWS). Web servislerinin işlevsel ve işlevsel olmayan yeteneklerinin semantik web dilleriyle tanımlanmasıyla SWS'ler oluşturulur. SWS tanımları yapılırken belirli alan ontolojilerine referans verilerek servislerin ilgili bütün parametreleri anlamlandırılır. Servis tanımlarını zenginleştirmek için ontolojileri kullanmak, servislerin anlamlarının yazılım ya da

bilgisayar tarafından yorumlanabilir olmasını ve kullanıcıların kısa ve anlamlı sorgular yapabilmesini sağlar (Lutz, 2005). SWS'ler servis isteği ve servis tanımı parametreleri arasındaki ilişkileri bulmak için mantık tabanlı eşleştirmeye izin vererek servis işlevselliğini daha iyi yakalar ve belirsizlikleri giderir.

Şekil 2'de geleneksel SYM ve SWT ile oluşturulmuş semantik SYM gösterilmektedir. Semantik SYM'de servis istemci ontolojilerle referanslandırılmış sorgusunu semantik servis eşleştiricisine gönderir. Servis sağlayıcılar da ontolojilerle referanslandırılmış servis tanımlarını servis eşleştiricisine gönderir. Eşleştirici bu iki tanım arasındaki benzerlikleri yine ontolojiler desteğinde belirli çıkarsama motorları kullanarak bulup, benzerliği en yüksek olanları istemciye döndürür.



Şekil 2. Geleneksel SYM ve Semantik SYM

1.6.1.1. Semantik Web Servisi Tanımlama Dilleri ve Araçları

Web servislerinin otomatik olarak bulunması servis kullanan yazılım araçlarının servis isteği ve servis tanımlarının anlamını anlaması ile mümkündür. Bu da servis bileşenlerinin sağlam ve mantık tabanlı olarak tanımlanmasına dayanır. Literatürde Web servisi tanımlarına semantiklik kazandıran birçok yaklaşım mevcuttur. Bu bölümde bunlardan en yaygın kullanılan WSDL-S (URL-8, 2015), WSML (URL-9, 2015) ve OWL-S (URL-10, 2015) hakkında bilgiler verilecektir.

1.6.1.1.1. WSDL-S

WSDL (Web Services Description Language) (URL-11, 2015), bir Web servisinin sahip olduğu operasyonları, operasyonların istek ve yanıt mesajlarını, mesajlar aracılığıyla taşınacak olan verileri ve tiplerini, Web servisini çağırmak için kullanılacak olan uygulama protokolünü ve servisin adresini tanımlayan XML tabanlı bir dildir. 15 Mart 2001'de W3C standardı olmuştur. Bir servis sağlayıcısı, sahip olduğu Web servislerini WSDL kullanarak tanımlar.

WSDL-S (Web Service Description Language Semantics) 7 Kasım 2005'de W3C standardı olmuş, SWS'leri tanımlamak için geliştirilmiş bir dildir. OWL-S'den farklı olarak, hazır bir WSDL tanımı üzerinden genişletme yaparak ontolojilere referans verir. Bu şekilde servis girdi, çıktı parametreleri ve operasyon tanımlarına anlam katar. OWL-S'e göre daha basit bir yapısı vardır.

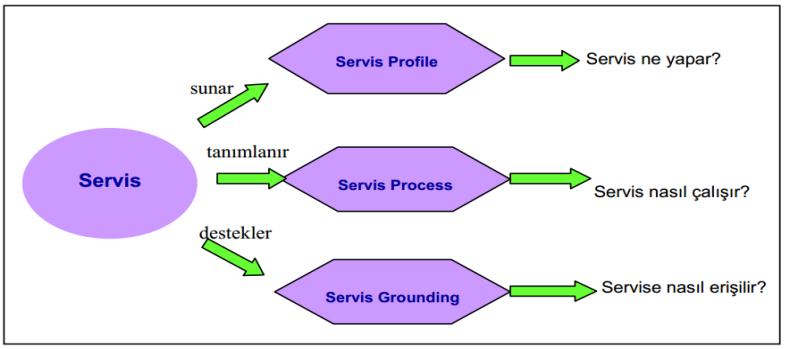
1.6.1.1.2. WSML

WSML (Web Service Modeling Language) SWS'leri tanımlama dilidir. 3 Haziran 2005'de W3C standardı olmuştur. WSML-Core, WSML-DL, WSML-Flight, WSML-Rule ve WSML-Full olmak üzere beş alt modeli vardır. Tanımlama Mantığı (Description Logics), Birincil Mantık (First-Order Logic), Mantıksal Programlama (Logic Programming) tabanında geliştirilmiştir (URL-9, 2015). WSML, SWS'leri tanımlarken WSMO' da tanımlı kavramları kullanılır. WSMO (Web Service Modeling Ontology) (Bruijn vd., 2015) otomatik web servisi bulma, birleştirme ve çalıştırma işlemlerini kolaylaştırmak için web servislerini semantik olarak tanımlayan kavramsal bir çatı ve resmi bir dil sağlar.

1.6.1.1.3. OWL-S

OWL-S (Ontology Web Language (OWL) for Services) , OWL tabanlı Web servisi tanımlama dilidir. 22 Kasım 2004'de W3C standardı olmuştur. Servis bulma, koşturma, düzenleme ve birlikte işlerlik gibi Web servislerine özgü işlevlerin otomatikleştirilmesini sağlar. Bir servis ontolojisi; OWL-S Profili (Profile), OWL-S İşlem Modeli (Process),

OWL-S Referansı (Grounding) olmak üzere üç ana bölümden oluşur. Bu üç ontoloji arasındaki bağlantı "sunar" (present), "destekler" (support) ve "tanımlanır" (described by) ilişkileri ile kurulur (W3C, 2004). Şekil 3'de üst düzey servis ontolojisi gösterilmektedir.



Şekil 3. Üst düzey servis ontolojisi (W3C, 2004)

Servis Profili, bir web servisinin metaverisini sunan kısımdır. Bu kısımda servisin sağlayıcı ve iletişim bilgileri, girdi (input), çıktı (output) parametresi, varsa ön koşul (precondition) ve son koşul (effects) ile servisin kalite parametreleri (puanlama, maksimum cevap verme süresi) gibi bilgiler sunulur. Servis İşlem Modeli, bir servisin nasıl çalışacağını, hangi operasyonları olduğunu, bu operasyonları çalıştıran girdi ve çıktı parametrelerini tanımlar. Bir servis işlem modeli bir çok girdi parametresinden, birçok çıktı parametresine veri dünüşümü (data transformation) yapar. Ayrıca durum değişimi sağlar. Bu da "işlem modeli" nin "ön koşul" (precondition) ve "son koşul" (effects, postcondition) parametreleri ile tanımlanır. Servis Referansı**,** bir web servisinin iletişim protokolü, mesaj formatı, "port" numarası gibi servise erişimi sağlayan parametreleri tanımlar (W3C, 2004).

Semantik işlemciler (processor) bir servisi koşturmak için sözdizimsel bilgilere ihtiyaç duyar. Benzer şekilde, istemci uygulama işlemcisi de istek mesajını nasıl seri hale getireceğini bilmek ister. Bu da "Grounding" olarak adlandırılan semantik ve sözdizimsel tanımlar arasındaki bağlantı ile gerçekleştirilir.

1.6.1.2. Semantik Web Servisi Bulma ve Eşleştirme Araçları

Geleneksel yaklaşımda servis sağlayıcılar web servislerini ebXML (electronic business XML) Registry (OASIS, 2005) ve UDDI (Universal Description Discovery and Integration) (UDDI.org, 2000) gibi standart katalog servisleri aracılığıyla yayınlarlar. UDDI ve ebXML, Web servisleriyle ilgili veri ve metaveri sağlayan servis kataloglarıdır. Web servislerini sınıflandırmak, kataloglamak ve yönetmek için standart bir mekanizma sağlarlar. Böylece, servisler diğer uygulamalar tarafından bulunur ve kullanılır. Kullanıcılar, katalog servislerinde arama yaparak ihtiyaç duydukları Web servislerinin WSDL dokümanlarını elde ederler (Akıncı, 2006).

Bir Web servisinin kullanışlılığı internet ortamında bulunabilirliği ile doğru orantılıdır. Servisin bulunabilirliği ise servis tanımının iyi bir şekilde yapılmış olmasını gerektirir. Servis tanımlarının SWT teknolojileriyle zenginleştirilmesi ile servis bulma işlemi daha doğru ve daha hızlı gerçekleşmektedir. Web servisi bulma, bir istemcinin herhangi bir uygulama alanına yönelik en uygun servisleri Web'de yayınlanmış birçok servis arasından bulma işlemidir. WSK'da servis bulma (eşleştirme olarak da bilinir) ise kompozisyon içindeki her bir operasyon için eşleşen somut bir servisin bulunması aşamasıdır. SWT ile servis bulma istenen bir Web servisinin, servis tanımlarının servis istekleriyle semantik eşleştirilmesi sonucu bulunmasıdır.

Bu bölümde literatürde yaygın kullanılan semantik eşleştiricilerden bir kısmı anlatılacaktır.

1.6.1.2.1. OWL-S UDDI Matchmaker

2002 yılında Carnegie Mellon Üniversitesi (Intelligent Software Agents Group) tarafından geliştirilmiş SWS için kullanılan bütünleşik bir yazılımdır. UDDI'nin genişletilerek Matchmaker (eşleştirici) modülü eklenmiş şeklidir. Bu şekilde UDDI'de servisler yeteneklerine göre aranabilmekte ve UDDI'de servis tanımları semantik yeteneklerine göre yayınlanabilmektedir (publish) (Srinivasan vd., 2004).

Matchmaker, UDDI' nin yayınlama ve sorgulama port'unu kullanır. UDDI'de herhangi bir servis yayınlandığı zaman, UDDI bunu sıradan bir servis tanımı olarak değerlendirir. Ancak, servis tanımı OWL-S profili bilgilerini içeriyorsa, bunu Matchmaker'a gönderir. Matchmaker, servis tanımını içindeki semantik bilgiye göre

sınıflandırır (Srinivasan vd., 2004). Hem yayınlama hem de sorgulama aşamasında OWL-S dili kullanılır. OWL-S'de yapılan servis tanımları UDDI içinde "OWL-S UDDI mapping" fonksiyonu ile UDDI'in veri modeline dönüştürülür. Bu dönüşüm işleminde OWL-S profili ile UDDI veri modelinde bulunan ve bire bir eşleşen parametreler aynen alınır, eşleşmeyen parametreler arasında "t-model" tanımlanır. Böylece yayınlanan her servis tanımı UDDI servis kataloğuna kaydedilir. Matchmaker, SWS bulma aşamasında RacerPro çıkarsamacısını kullanır.

1.6.1.2.2. OWLS-MX Matchmaker

OWLS-MX Matchmaker, Alman araştırma merkezi (German research center for artificial intelligence-DFKI) tarafından 2005 yılında geliştirilmiş açık kaynak kodlu java tabanlı servis bulma ve eşleştirme yazılımıdır. SWS için geliştirilmiş ilk karma (hybrid) servis eşleştiricisidir. Karma ile kastedilen diğer servis eşleştiricilerden farklı olarak, mantık tabanlı (logic based) çıkarsama yanında içerik tabanlı bilgi erişim (information retrieval) tekniklerini (Cohen vd., 2003) kullanarak servislerin girdi ve çıktı parametresi bazında servis eşleştirme yapabilmesidir. OWLS-MX Matchmaker, OWL-S API 1.1 beta versiyonu ile Pellet (www.mindswap.org) OWL DL çıkarsamacısı kullanılarak java'da gerçekleştirilmiştir (Klusch vd., 2006). Matchmaker, servis tanımı ve servis isteği arasındaki eşleştirme derecesini hesaplarken beş farklı filtre kullanır. Bunlardan "exact", "plug-in","subsumes", "subsumed by" sadece mantık tabanlı,"subsumed by" ve "nearest neighbour"ise karma filtrelerdir. Karma filtreler, değişik benzerlik değerlerinin de hesaplanmasıyla içerik tabanlı eşleştirmeyi sağlar. Matchmaker, istemciye sorgu oluştururken eşleştirme aşamasında hangi filtrelerin kullanılacağını seçme olanağı verir. Yayınlanan ve sorgulanan servis tanımları OWL-S dilinde olmalıdır.

1.6.1.2.3. OWL-S Matcher

OWL-S Matcher (URL-12, 2008) 2005 yılında OWL-S servis tanımlarını eşleştirmek için geliştirilmiş bir servis eşleştirme algoritmasının açık kaynak kodlu java gerçekleştirimidir. Önceki versiyonu DAML-S (URL-13, 2015) servis tanımlarını eşleştirmeye yöneliktir. Daha sonra güncellenerek OWL-S uyumlu hale getirilmiştir.

Servis istemcisi ve sağlayıcısından gelen OWL-S tanımlarını OWLJessKb çıkarsamacısını kullanarak karşılaştırıp aralarındaki benzerlik ilişkilerini bulur.

1.6.1.2.4. SPARQLent

SPARQLent (SPARQL Agent) (Sbodio vd., 2010) SWS'lerin bulunması için servis koşul (ön koşul ve son koşul) parametrelerinin doğruluğu ile servis istemcisinin sorgusunu karşılaştırarak benzer servisleri bulur. Servis koşul parametrelerini ve istemci sorgusunu tanımlamak için SPARQL (Simple Protocol and RDF Query Language) (URL-14, 2015) dilini kullanır. SPARQLent aracı ile servis ön koşulunun ve servis çalıştırıldıktan sonra oluşan son koşulun semantik doğruluğunu kontrol etmek ve servisin çalıştırılmasıyla oluşan sonuçların istemci sorgusunu karşılayıp karşılamadığını test etmek için SPARQL sorgu değerlendirmesinin kullanılabileceğini göstermek amaçlanmıştır. Servis ön koşulunun doğruluğu servisin başarılı bir şekilde kullanılabileceğini, son koşul ise servisi kullandıktan sonra neyin doğru olacağını gösterir. Bu şekilde istemci sorgusunun karşılanması, servisin bu amacı gerçekleştirmek için kullanılabileceğini gösterir. Böylece servis bulma problemine bir çözüm sağlanır. SPARQL, bilgi tabanları (Knowledge Base) için en yaygın kullanılan sorgulama dilidir. SPARQLent eşleştiricisinin algoritmaları ve işlevleri SPARQL sorgularının sayısal modelinden yararlanır. SPARQLent, semantik web dillerini değerlendirmek için Jena (Carroll vd., 2003) platformunu, SPARQL'i değerlendirmek için ARQ6 (URL-15, 2015)' ı kullanır. Ayrıca OWL, RDF ve OWL-S ile kullanımı da desteklemektedir.

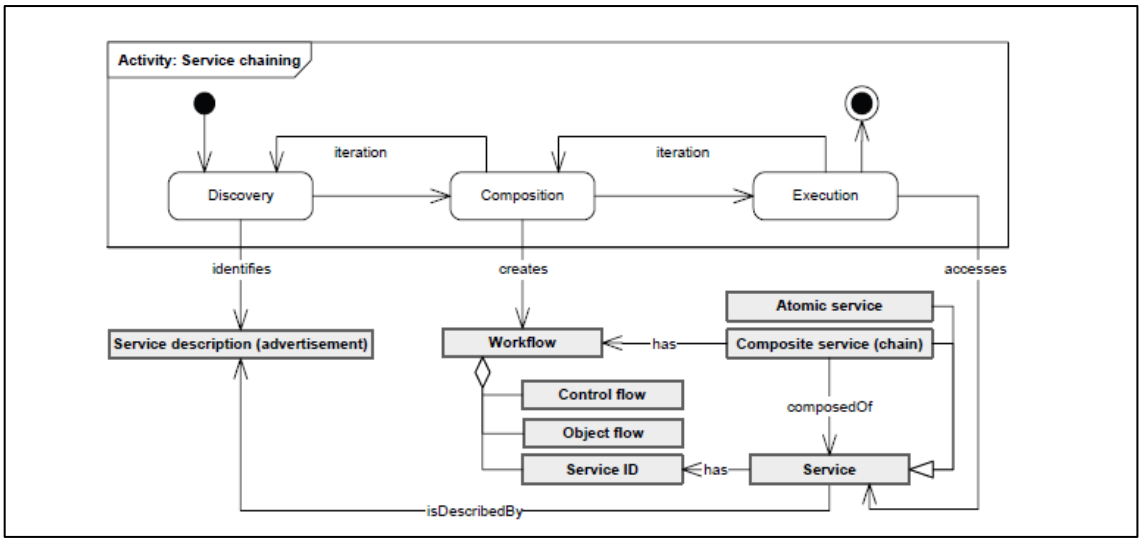
1.6.1.2.5. iSeM

iSeM (İntelligent Service Matchmaker) (Klusch ve Kapahnke, 2012) SWS için karma ve esnek bir eşleştirme aracıdır. Klusch vd., (2006)'da geliştirilen eşleştiricinin zenginleştirilmiş halidir. Fonksiyonel servis tanımlarını mantıksal referanslandırmalarla, ön koşul ve son koşul tanımlamalarıyla kullanır. iSeM, servisleri mantıksal ve mantıksal olmayan semantik eşleştirme ile bulmaktadır. Mantık tabanlı eşleştirme, servis parametrelerinin mutlak ve tahmini kavramsal alt-üst sınıf ilişkilerinin hesaplanmasına dayanır. Mantıksal olmayan eşleştirme ise, farklı sözcük tabanlı benzerlik ölçütleri ile

referanslandırılmış servis kavramlarının ontoloji tabanlı yapısal eşleştirilmesine dayanır. Tahmini eşleştirme ise DL (Description Logic)'in (Baader vd., 2003) alt seti olan SH (Di Noia vd., 2009)'de servis girdi çıktı parametrelerinin mantıksal kavram çıkarsamasına dayanır. Servis ön koşul ve son koşul parametrelerinin mantıksal eşleştirmesi ise SWRL'de yapılır. Bütün servis parametreleri (IOPE) için, anılan farklı eşleştiricilerden gelen eşleştirme sonuçlarının ağırlıklandırılmış toplamı hesaplanır. Bunun için SVM (Support Vector Machine) (URL-16, 2015) tabanlı ikili ilişki sınıflayıcı ve ranklayıcısı kullanılır. SVM aracılığıyla üretilen pozitif ve negatif eğitim sınıfı örneklerinden ranklanmış servis listesi oluşturularak kullanıcıya cevap olarak döndürülür.

1.6.1.3. Semantik Web Servislerinin Kompozisyonu (SWSK)

Birden çok servisi ilgilendiren daha karmaşık uygulamaların gerektiği durumlarda atomik servisler kullanıcı ihtiyaçlarını karşılayamayacaktır. Bu durumda servis kompozisyonu gündeme gelir. Christophi (2003), WSK'yı oldukça basit yapıdaki web servislerinin işlevlerinin daha anlamlı ve karmaşık uygulamalar için kompoze edilmesi tekniği olarak tanımlar. WSK'nın ana hedeflerinden biri mevcut web servislerini bir işlem modeli içinde kompoze ederek tekrar kullanılabilirliğini sağlamaktır. Şekil 4'de bir servis zincirinin (WSK) bileşenleri gösterilmektedir.



Şekil 4. UML'de oluşturulmuş servis zinciri bileşenleri (Lemmens, 2006).

WSK alanındaki problemlerden biri, Web'de web servislerinin sayısının günden güne artması ile istenen servislerin doğru ve hızlı bir şekilde bulunamamasıdır. Diğer bir problem, web servislerinin çalışma durumunda meydana gelebilecek anlık değişiklikler nedeniyle kompozisyon sistemlerinin koşum zamanında güncellenmeye ihtiyaç duymasıdır. Ayrıca Web servislerini tanımlamak için ortak tek bir model olmayışı, farklı kurumlar tarafından geliştirilen servislerin tanımlarında değişik kavram modellerinin kullanılması problemini oluşturmaktadır. Bu konuda iş yükü ve zamansal açıdan önemli olan diğer bir problem, kullanıcıların yeteneklerinin dışında olmasına rağmen kompozisyonu manuel olarak gerçekleştirmesidir. Geleneksel WSK yaklaşımı bu problemlere çözüm sağlayamamaktadır.

SWSK mimarisi ile geleneksel yaklaşımdaki problemlerin çözülmesi amaçlanmıştır. SWSK, WSK' nın SWT ile gerçekleştirilmiş modelidir. SWSK ile makineler ya da yazılım araçları, kullanıcılardan gelen belirli ölçütlere sahip işlemleri yapabilmek için otomatik olarak değişik Web servislerini bulur, ilgili kontrol akışı parametrelerine göre kompoze eder ve çalıştırır. Bu şekilde Web'de gerçekleştirilecek rutin ve karmaşık işlemlerin insan müdahalesi olmadan kompozisyonu ve entegrasyonu ile iş yükü ve zamandan tasarruf sağlanır.

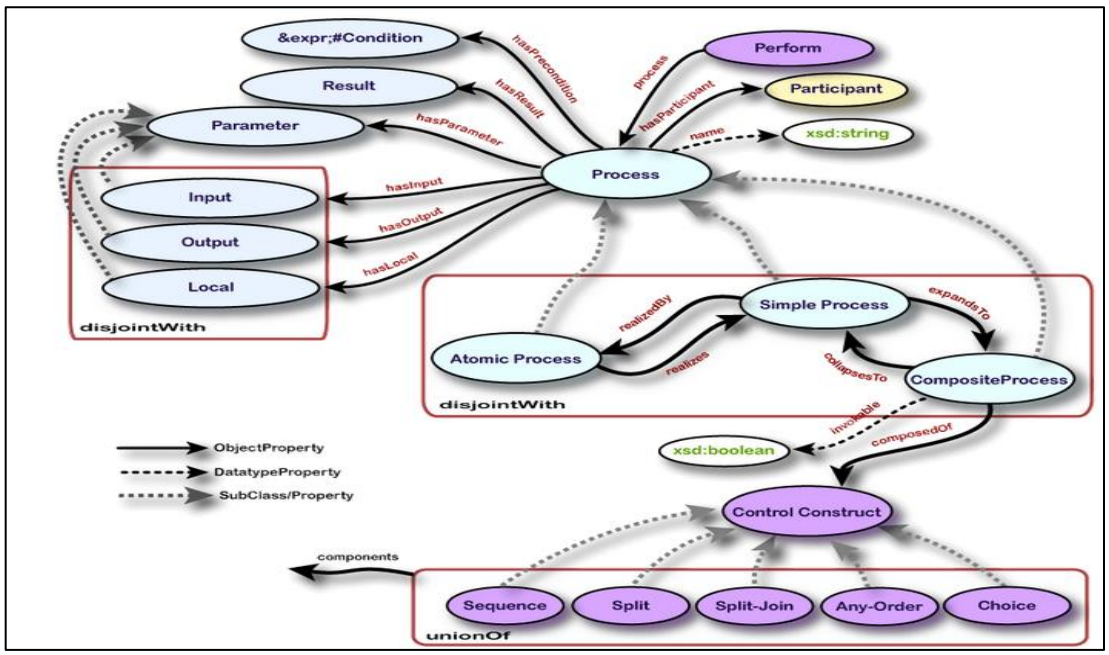
1.6.1.3.1. SWSK Tanımlama Dilleri

Bu bölümde literatürde yaygın olarak kullanılan SWSK tanımlama dillerinden OWL-S kontrol parametreleri ile WSMO kareografi ve orkestrasyon mekanizması anlatılacaktır.

1.6.1.3.1.1. OWL-S Kontrol Parametreleri

OWL-S, içerdiği kontrol parametreleriyle SWSK tanımlaması yapabilmektedir. OWL-S kontrol parametreleri Sequence, Split, Split + Join, Choice, Any-Order, If-Then-Else, Repeat-While ve Repeat-Until`den oluşmaktadır. Sequence, çalıştırılacak operasyonların sırasını ve hangi kontrol parametrelerinin kullanılacağını tanımlar. Split, eş zamanlı (paralel) olarak calıştırılacak operasyonları tanımlar. Split+Join, eş zamanlı olarak calıştırılacak operasyonları ve bunların tekrar senkronize (birleştirilmesi) edilmesini tanımlar. Choice, verilen operasyonlar arasından bir tanesinin çalıştırılmasını tanımlar.

Operasyonlardan herhangi biri seçilebilir. Bu seçim önceden karar verilmemiş, anlık bir seçimdir. Any-Order, verilen operasyonların belli bir sıraya bağlı olmadan eş zamansız olarak çalıştırılmasıdır. İçerilen bütün operasyonlar çalıştırılmak zorundadır. If-Then-Else, verilen bir koşul sağlanıyorsa ilgili operasyonu, sağlanmıyorsa verilen diğer operasyonu çalıştırır. Yani koşul sağlanıyorsa 'Then' parametresi, sağlanmıyorsa 'Else' parametresi çalıştırılır. Repeat-While, verilen bir koşul sağlandığı sürece iterasyonun tekrarlanacağını tanımlar. Repeat-Until, verilen bir koşul sağlanıncaya kadar iterasyonun tekrarlanacağını tanımlar.



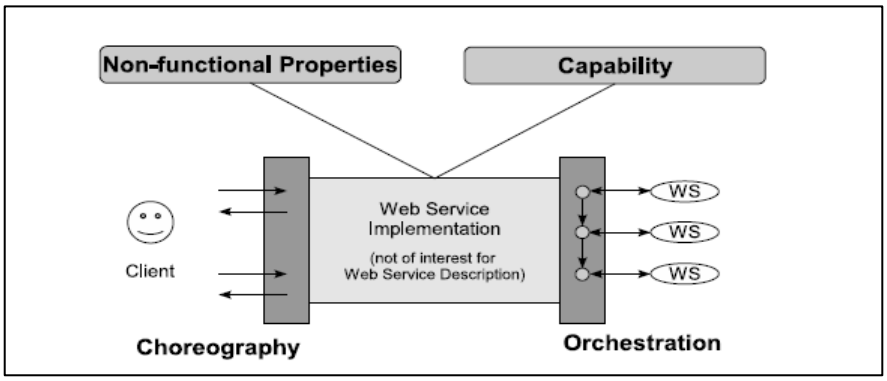
Şekil 5. Üst düzey İşlem (Process) ontolojisi (URL-10, 2015)

Şekil 5'de üst düzey OWL-S işlem ontolojisi ve kontrol parametreleri gösterilmektedir.

1.6.1.3.1.2. WSMO Kareografi – Orkestrasyon

WSMO Kareografi (choreography), bir servisin çalıştığında istemci ile etkileşimde bulunarak bir işlemi nasıl gerçekleştireceğini tanımlar. Yani servisin işlevlerinin kullanılabilmesi için, o servisle istemci arasında kurulacak iletişim bilgilerini tanımlar. Söz konusu istemci başka servisler, uygulamalar ya da insanlar olabilir. WSMO Orkestrasyon

(orchestration) ise, bir servisin diğer servisleri kullanarak bütün işlevleri nasıl gerçekleştireceğini tanımlar. Yani diğer servislerin koordinasyonudur. Servis parametreleri ile kareografi ve orkestrasyon arayüzünden oluşan bir WSMO web servisinin temel yapısı Şekil 6'da gösterilmektedir.



Şekil 6. WSMO Web Servisi genel tanımı (Fensel vd., 2007)

WSMO Kareografi ve Orkestrasyon arayüzlerini tanımlayan durum (state) tabanlı mekanizma ASM (Abstract State Machine) (Börger ve Stark, 2003) metodolojisine dayanır. ASM, bir servisin koşum anında sergileyeceği tavrı soyut olarak tanımlamak için kullanılır. WSMO referans kavramlarıyla WSML dilinde geliştirilen Kareografi ve Orkestrasyonlar WSMX (URL-17, 2015) aracı ile oluşturularak çalıştırılır.

1.6.1.3.2. SWSK Oluşturma Araçları

Literatürde WSK oluşturmak için farklı araçlar mevcuttur. Bu bölümde bunlardan en yaygın kullanılan Protege OWL-S Editörü, OWL-S Composer, OWLS-XPlan ve Taverna hakkında bilgiler verilecektir.

1.6.1.3.2.1. Protege OWL-S Editörü Eklentisi

SRI (Stanford Research Institute) (URL-18, 2015) tarafından Protege Ontoloji Editörü (URL-19, 2015)'ne eklenti olarak geliştirilmiş OWL-S'de servis tanımları yapmayı ve servis koşturma işlemlerini sağlayan açık kaynak kodlu bir yazılımdır. Hem servis

tanımlarının hem de servis tanımlarında kullanılacak alan ontolojilerinin aynı yazılım bileşeni üzerinden oluşturulması ve daha rahat kullanılmasını sağlamak amacıyla geliştirilmiştir. Servis profili, işlem modeli, referansı (grounding) ve kompozisyon işlemlerinin tek bir ara yüz ile tanımlanmasını sağlar. OWL-S editörü kompoze servisleri de modelleyebilmektedir. Tanımlanmış bütün atomik servisleri kompozisyon işlem modeli oluşturarak veri ve kontrol akışı işlemleriyle grafiksel olarak organize etmeyi sağlar. Veri akışından kasıt, WSK içindeki servislerin birbirine hangi parametreleri hangi atomik servisten aktaracağıdır. Kontrol akışından kasıt, WSK içindeki servislerin hangi sıra ve hangi koşullara göre çalıştırılacağıdır. Anılan bütün OWL-S kontrol parametrelerini (bkz. 1.6.1.3.1.1.) desteklemektedir. Ayrıca, "If-Then-Else" gibi koşullar oluşturmak için SWRL (Semantic Web Rule Language) (URL-20, 2015) dilini desteklemektedir. Ancak, Protege OWL-S Editorü eklentisi ile oluşturulan servisler sadece tek olarak (atomik) çalıştırılabilmektedir. Kompoze edilmiş servislerin çalıştırılmasını henüz desteklememektedir.

1.6.1.3.2.2. OWL-S Composer

OWL-S Composer (URL-21, 2015), FORMAS semantik uygulamalar ve biçimlemeler araştırma grubu tarafından UFBA (Federal University of Bahia)'da Eclipse IDE (URL-22, 2015)' ye eklenti olarak geliştirilmiş, açık kaynak kodlu bir WSK oluşturma aracıdır. Görsel diyagramlarla OWL-S'de SWSK oluşturmak, kompozisyonları karşılaştırmak, çalıştırmak ve çalışmasını izlemek için geliştirilmiştir. OWL-S Composer ile java'da bir web servisi oluşturularak, servisin WSDL tanımı elde edilebilmektedir. SWS oluşturmak için servislerin WSDL tanımlarından "WSDLtoOWL-S" dönüşümü ile OWL-S tanımları üretilmektedir. Daha sonra oluşturulan servisler OWL-S kontrol parametrelerine göre ard arda dizilerek SWSK oluşturulmaktadır. Fakat bu composer ile sadece 'sequence', 'split' ve 'any order' kontrol parametreleri kullanılarak SWSK oluşturulabilmektedir. Diğer kontrol parametreleri henüz desteklenmemektedir.

1.6.1.3.2.3. OWLS-XPlan

OWLS-XPlan (Klusch ve Gerber, 2006), OWL-S web servislerini değerlendirip kompozisyon planlamak için kullanılan bir SWSK plancısıdır. Java ve C++' da

geliştirilmiştir. OWL ontolojileri ile ilişkilendirilmiş OWL-S servislerini ve istemcinin oluşturduğu planlama isteğini (goal) girdi olarak alır ve isteğe uygun ilgili OWL-S servislerinin planlama sıralamasını üretir. Bunun için verilen OWL'de kodlanmış bir alan ontolojisi ile OWL-S servis tanımlarını OWLS2PDDL dönüştürücüsünü kullanarak PDDL (Planning Domain Description Language) (URL-23, 2015) ve alan tanımları karşılığına çevirir. Dönüşüm sonucunda oluşan bu tanımlar bütün tipleri, nesneleri, başlangıç ve amaç durumunu kapsar. Bütün bu tanımlar AI (Artificial Intelligence) (Peer, 2015) planner XPlan tarafından PDDL'de verilen problemi belirli bir alanda çözebilen bir plan oluşturmak için kullanılır. Ayrıca OWLS-XPlan, verilen bir planlama problemi için deneyimli kullanıcıların planlama amacını direkt olarak değiştirmesi ve başlangıç durum ontolojisini düzenlemesi için bütünleşik bir PDDXML editörü içerir.

1.6.1.3.2.4. Taverna

Taverna (Hull vd., 2006), belli bir alandan bağımsız java tabanlı açık kaynak kodlu bir iş akışı[3] yönetim sistemidir. Bilimsel iş akışlarını tasarlamak, düzenlemek ve çalıştırmak için myGrid (URL-24, 2015) grubu tarafından geliştirilmiş ve hala BioVeL (URL-25, 2015), SCAPE (URL-26, 2015) ve Wf4Ever (URL-27, 2015) FP7 projeleri tarafından desteklenmektedir (URL-28, 2015). Taverna ile web servisi bulma, çalıştırma ve servis kompozisyonu tasarlayıp çalıştırabilme işlevleri grafiksel olarak yapılmaktadır.

Taverna, farklı alanlarda ve farklı çeşit birçok servis tipini desteklemektedir. Bunlar; WSDL, RESTful (URL-29, 2015), BioMart (URL-30, 2015), BioMoby (URL-31, 2015), SoapLab (URL-32, 2015), R (URL-33, 2015), Beanshell (URL-34, 2015), API Consumer (URL-35, 2015) servisleri ile Excel veya csv tablolarını import eden servislerdir. Taverna bu servislere programatik ara yüzlerle, yani temel olarak web servisleriyle tek bir noktadan erişim sağlar. BioCatalogue (URL-36, 2015) gibi servis kataloglarına erişimi destekler. Taverna, oluşturulan iş akışlarının daha sonra bulunup tekrar farklı uygulamalarda kullanılabilmesi için, iş akışı kataloğu özelliğinde olan myExperiment (URL-37, 2015)'de yayınlanmasını (publish) destekler. İş akışı tasarım anında önceden tanımlanmış akışlar kayıtlı olduğu myExperiment'den import edilerek tekrar kullanılabilir. Her iş akışı kendi

[3] Taverna mimarisinde "kompozisyon" yerine "iş akışı" (workflow) ifadesi kullanılmaktadır.

içinde yazar ve başlık etiketlerinden oluşan metaveri bölümü içerir. Farklı iş akışı metaverisi de eklenebilir.

Taverna, iş akışı içinde yer alan servis parametrelerinin birbiri ile uyumsuz olduğu durumlarda veriye format dönüşümü yapan dönüşüm servisleri sağlar. Taverna servisleri girdi veya çıktı parametreleri için tek değer ya da birden fazla değeri içeren 'list' ve iç içe geçmiş 'list' veri yapısını destekler. 'List' içindeki eleman sayısı 'List Depth' olarak ifade edilir. Yani servisin girdi Port'undaki bir parametrenin alabileceği değer (eleman) sayısını ifade eder. Yeni bir iş akışı ya da servis oluşturulduğunda girdi Port'unun 'List Depth' değerinin belirlenmesi gerekir. 'List Depth'leri uyuşmayan yani 'List' tipinde çıktısı olan bir servis, girdisi tek değerli olan bir servisle ard arda getirildiğinde, Taverna gizli iterasyon yaparak ikinci servisi 'List' içinde bulunan her değer için çalıştırır ve çıktı olarak yeni bir 'List' oluşturur. Taverna, içerdiği 'List Handling' özelliği ile çoklu 'List' leri servislerle ilişkilendirebilmektedir. Yani, girdi parametreleri birden fazla değere sahip olan servisler çalıştırıldığında işlemlerin hangi parametre değerleri arasında karşılıklı olarak yapılacağını ayarlar. Varsayılan olarak her bir değerin birbiri ile işlem gördüğü anlamına gelen çoka çok (all-to-all) iterasyon yapılır. Gerek duyulduğunda 'List Handling' bölümünden bu ayar değiştirilip sadece karşılıklı değerlerin işlem gördüğü 'bire bir' (one-to-one) iterasyon olarak ayarlanabilir.

Web servislerinde bazen ağ bağlantısından kaynaklı kopmalar olabilir. Bu da iş akışı içindeki servislerden bir ya da birkaçının çalışmaması sonucu akışın işlevini tamamlayamamasına sebep olur. Taverna ile böyle geçici kesintiler, iş akışı içindeki servislere 'Retries' (tekrarlamalar) sayısı eklenerek önlenebilir. Bu şekilde, her servis girilen tekrarlama sayısı kadar ard arda çalıştırılır. Taverna ile koşullu servis çalıştırmak için 'Loop' döngüleri ifade edilebilmektedir. Bu döngülerle servislere özel koşullar da eklenebilir. Taverna aynı zamanda, hata ayıklama aşaması için tasarım zamanında iş akışı içindeki servislerin çalıştırılmadan önce çevrim içi (online) olup olmadığını test etmek amacıyla 'validation' bilgisi almayı sağlar. Paralel servislerin çalıştırılmasını destekler. İş akışı çalışmasının izlenmesini sağlayarak koşum anında anlık sonuçları ve akıştaki hataları görüntüleyebilir.

1.6.2. Şema Eşleştirme

SW'nin gereksinimlerinden biri şema eşleştirmedir. Dünyada bu konu ile ilgili çalışmalar son derece hızlı bir şekilde ilerlemektedir. Yeni teknikler bulunmakta, buna ilaveten mevcut tekniklerin zenginleştirilmesi üzerine birçok çalışmalar yapılmaktadır. Konumsal veri altyapılarının (KVA) oluşturulmasında kurumlar arası birlikte işlerliğin sağlanması gereklidir. "Birlikte işlerlik" (interoperability), çok genel olarak donanım ve yazılım olarak farklı sistemlerin birbirleri ile "iletişim kurabilmesi" ya da daha iddialı bir sözcükle, "konuşabilmesi" olarak tanımlanabilir (Cömert ve Akıncı, 2005). Günümüz teknolojisinde mevcut veri sağlayan kurumların kendi başlarına yetersiz olmaları konunun önemini daha da artırmaktadır. Bu sebeple farklı kurum verilerinin ya da aynı kuruma ait farklı verilerin kombine edilerek daha çok amaca hizmet edecek şekilde kullanımı gündeme gelmiştir. Kurumlardaki verilerin birleştirilmesi için öncelikle kurumların sunduğu veritabanı şemalarının eşleştirilmesi gerekmektedir. Şema eşleştirme konusundaki araştırmaların amacı değişik şemalar arasındaki ilişkileri belirlemektir.

Eşleştirme, şemalar çizge (graph) yapısında düşünüldüğünde, iki çizge noktasındaki kavramın birbiri ile semantik (anlamsal) veya sentaktik (sözdizimsel) olarak karşılaştırılarak, aralarındaki benzerliğin belirlenmesi olarak algılanabilir. Şemalar arasındaki ilişkiler hem çeşitli dilsel benzerlik teknikleri kullanılarak sentaktik olarak, hem de WordNet (WN) gibi veri sözlükleri ya da belirli alan ontolojileri kullanılarak semantik olarak bulunur. Eşleştirme sonunda karşılaştırılan şema varlıklarına ve özniteliklerine ait benzerlik değerlerini gösteren rakamsal değerler ya da Tanımlama Mantığı (Description Logic) tabanlı ifadeler üretilmektedir.

Literatürde şema ve örnek (instance) tabanlı sentaktik ya da semantik eşleştirme yapan eşleştiriciler mevcuttur. Şema tabanlı eşleştirme yapan yazılımlara S-Match (Giunchiglia vd., 2004), COMA++ (Aumueller vd., 2005) ve Cupid (Madhavan vd., 2001) örnek verilebilir. Örnek tabanlı eşleştirme yapan yazılımlara ise GLUE (Doan vd., 2003) ve Gsim (Partyka vd., 2010) örnek verilebilir. Literatürde hem şema hem de örnek tabanlı semantik ve sentaktik eşleştiricilere yönelik doğruluk ve performans testi yapan çalışmalar da mevcuttur. Giunchiglia vd., (2004) ve Partyka vd., (2010) bu çalışmalara örnek verilebilir. Ayrıca, Shvaiko ve Euzenat (2005) çalışmasında şema eşleştirme teknikleri konusunda kapsamlı bir sınıflandırma yapılmıştır. Bu tez çalışmasında şema eşleştirme

sentaktik (sözdizimsel) ve semantik (anlamsal) olarak iki ana bölüme ayrılarak incelenecektir.

1.6.2.1. Sentaktik Eşleştirme

Sentaktik eşleştirme, terimler arası yazımsal ya da veri tipi benzerliğini dikkate alarak yapılan eşleştirmedir. Literatürde yaygın olarak kullanılan sentaktik eşleştiriciler "Prefix", "Suffix", "Edit Distance", "NGram", "Text Corpus" eşleştiricileridir. Bu eşleştiriciler hakkında detaylı bilgi ilerleyen bölümlerde verilecektir. Sentaktik eşleştirme sonucu şema elemanları arasında "0-1" aralığında rakamsal değerler üretilir.

1.6.2.2. Semantik Eşleştirme

Semantik eşleştirme, ontolojiler gibi ortak bir referans çatısına göre terimlerin anlamsal olarak karşılaştırılarak aralarındaki benzerliğin bulunmasıdır. Temeli çizge yapısına, kavramların çizgedeki konumuna ve kavramlar arası benzerlik analizlerine dayanır. Semantik eşleştirme teknikleriyle insan müdahalesi en aza indirilerek şemalar arası ilişkiler otomatik olarak bulunmaktadır. Giunchiglia vd., (2005)' e göre semantik şema eşleştirme, şemalar arası semantik ilişkilerin (eşitlik, az genel, çok genel, vb.,) hesaplanarak, şema elemanlarını temsil eden kavramlarda ve şema yapısında kodlanmış anlamların analiz edilmesi ile gerçekleştirilir. Semantik eşleştirme sonucunda nokta kavramları arasında tanımlama mantığı tabanlı ifadeler (eşittir (≡), az geneldir (⊂□), çok geneldir (⊃), ayrıktır (⊥) üretilmektedir.

Semantik eşleştirme, geleneksel anahtar sözcük tabanlı aramayı geride bırakarak, veritabanı şemalarının eşleştirilmesi, ontoloji eşleştirme, veri madenciliği gibi birçok alanın yanında, web servisi bulma alanında da kullanılmaktadır. Semantik eşleştiriciler arka planda ilgili alana özgü tanımlanmış bir veri sözlüğü, alan ontolojisi ya da veritabanı kullanabilir.

Semantik eşleştirme birçok alanda kullanım sahası bulmuştur. Bilgi Temsil Sistemlerinde (Knowledge Representation Systems) ve veri kullanan birçok çalışma alanında kullanılmaktadır. Bunlardan bazıları; kaynak bulma, veri entegrasyonu, sorgu çevirme, "Peer to Peer" (noktadan noktaya) ağlar, agent iletişimi, e-ticaret, semantik web,

şema ve ontoloji birleştirme alanlarıdır. Bu tez çalışmasında semantik şema eşleştirme, konumsal veri altyapılarının oluşturulması kapsamında şema, ontoloji birleştirme ve web servislerinin bulunması bakımından incelenecektir.

Şekil 7. Sentaktik ve semantik eşleştirme

Şekil 7'de iki farklı veritabanı şeması arasındaki semantik ve sentaktik eşleştirme karşılaştırması gösterilmektedir. Aynı kavramla ilgili olan bu iki veritabanı farklı detay tanımlarına sahiptir. Birinci şemadaki yol tablosunun "sınıf", "uzunluk", "genişlik" öznitelikleri ve bu özniteliklerin aldığı "kara yolu", "yaya yolu", "1000 km" ve "20 m" değerleri vardır. İkinci şemadaki yol tablosu "tip", "uzunluk", "en" öznitelikleri ve birinci tablodaki aynı verilere sahiptir. Sentaktik eşleştirme yapıldığında şema özniteliklerinden sadece yazılışları aynı olanlar eşleşmektedir (uzunluk-uzunluk). Ancak semantik eşleştirme yapıldığında bunun yanında yazılışları farklı ama anlamları aynı olan sözcükler de eşleşmektedir (sınıf-tip, genişlik-en).

Literatürde semantik eşleştirme alanında genel ve konumsal birçok akademik çalışma ve proje mevcuttur. Schade (2009), Partyka vd., (2010), Cruz vd., (2005), Lemmens (2006) ve Nathalie (2009) konumsal alandaki akademik çalışmalara örnektir. S-Match[4] ve COMA[5] genel alandaki projelere ve araçlara örnektir. COMA, kullanılan farklı eşleştirme algoritmalarını birleştirmek için kompoze bir eşleştirme yaklaşımıdır. Schade (2009)'da konumsal veri setlerinin bilgisayarca işlenebilen şekilde dönüşümü için, bir semantik referans çatısı ve semantik datumlar için bir cebirsel teori birleştirilmiştir. Partyka vd., (2010), şema özniteliklerinin eşleştirilmesi için örnek kümelemesini (instance clustering)

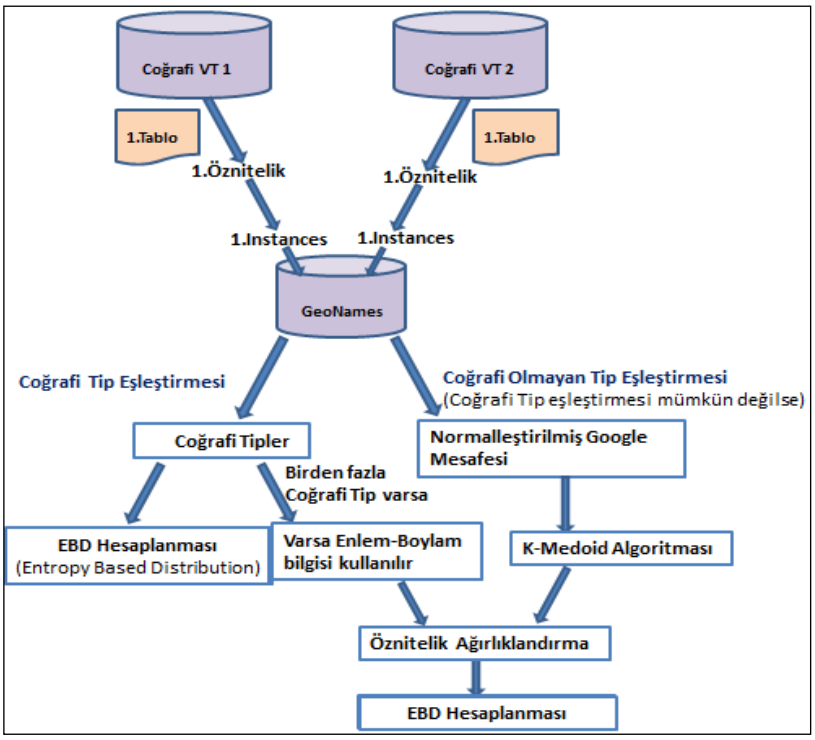
[4] S-Match, http://semanticmatching.org/s-match.html
[5] COMA,http://dbs.uni-leipzig.de/Research/coma.html

uygulamıştır. Shamdasani vd., (2011) çalışmasında tıp alanına özgü bir arka plan ontolojisini (UMLS) S-Match'de kullanarak şema eşleştirme gerçekleştirmiştir. Lemmens (2006), konumsal web servisleri arasında semantik eşleştirme gerçekleştirmiş ve bulunan servislerden servis zinciri oluşturmuştur. Nathalie (2009), sadece şema sınıflarının semantik eşleştirme işleminde dikkate alınmasının yeterli olmadığını ve bu nedenle sınıf özniteliklerinin de karşılaştırılması gerektiğini vurgulamıştır. Cruz vd., (2005), semantik olarak heterojen konumsal verinin entegrasyonu için merkezleştirilmiş bir entegre system ve "peer-to-peer" entegre sistem içinde bir ontoloji eşleştirme yaklaşımı tanımlamıştır. Ivanova (2010), "e-öğrenme" alanında WN heterojen kaynağını kullanan semantik bir kısmi ontoloji eşleştirme metodu önermiştir.

1.6.2.2.1. Semantik Eşleştirme Araçları

1.6.2.2.1.1. GSim Eşleştirme Algoritması

GSim (Partyka vd., 2010), konumsal şemalar arasında örnek tabanlı öznitelik eşleştirmesi gerçekleştirmektedir. Örnek, öznitelik verilerinin aldığı değerlerdir. Algoritma, girdi parametresi olarak karşılaştırılacak her iki coğrafi veritabanındaki tablolarda bulunan iki öznitelik verisine ait örnekleri ve bir yer adları sözlüğü alır, çıktı parametresi olarak öznitelikler arasındaki semantik benzerlik değerini üretir. Çalışmada yer adları sözlüğü olarak "GeoNames" kullanılmıştır. GeoNames, 10 milyonun üzerinde coğrafi isim, 5.5 milyon alternatif isim, 2.8 milyon popüler yer isimleri içeren ve bütün ülkeleri kapsayan bir coğrafi veritabanıdır. Birçok dilde yer isimleri, yükseklik, nüfus ve benzeri diğer coğrafi verileri birçok kaynaktan alarak birleştirir. İçerdiği bütün detaylar "feature class" tipinde depolanmaktadır (URL 38, 2011).



Şekil 8. GSim eşleştirme akış diyagramı

Eşleştirme yapılırken farklı iki coğrafi veritabanı tablolarındaki her bir öznitelik sırasıyla birbiriyle karşılaştırılır. Şekil 8'de görüldüğü gibi "veritabanı 1" ve "veritabanı 2" tablolarında bulunan alanlardan öznitelik verileri alınır ve her bir öznitelik verisine ait örnekler GeoNames yer adları sözlüğüne gönderilir. GeoNames veritabanında gönderilen örneklere ait tanımlı coğrafi tipler (kavramlar) varsa benzerlik değeri bu coğrafi tipler baz alınarak hesaplanır. Birden fazla coğrafi tip bulunması durumunda, varsa GeoNames veritabanında enlem ve boylam koordinatları ve ilgili coğrafi veritabanında o örneğe ait enlem boylam bilgisi karşılaştırılarak örtüşen coğrafi tipler baz alınır. Böylece bulunan coğrafi tiplerin sayısı 1'e düşürülür. GeoNames yer adları sözlüğünde gönderilen örneklere ait coğrafi tipler bulunamazsa ikinci yönteme geçilir. İkinci yöntem, coğrafi olmayan tip eşleştirmesidir. Bu yöntemde özniteliklere ait örnekler arasında "Normalleştirilmiş Google mesafesi" değeri her bir ikili için hesaplanır. Hesaplanan değerlere rastgele kümeleme algoritması olan "K-Medoid" algoritması iterasyonlarla uygulanır ve birbirine en yakın örnek değerler bulunur. Sonuçta karşılaştırılan özniteliklerin benzersizliğini derecelendirmek için ağırlıklandırma yapılır. Benzerlik değeri en yüksek olan öznitelikler yüksek puanla ağırlıklandırılır ve böylece en çok ilgili olan öznitelikler belirlenir. Bulunan sonuçlara göre semantik benzerlik değerini ifade eden EBD (Entropy Based Distribution)

hesaplanır. Tablolar arasındaki en son benzerlik değeri öznitelik eşleştirmelerinin ortalaması alınarak hesaplanır.

Normalleştirilmiş Google Mesafesi, Google arama motoru tarafından indekslenmiş web sayfalarına dayanır. Eşitlik (1)'de Google mesafe (Google Distance) değerinin hesaplanmasında kullanılan formül gösterilmiştir. F(x), Google'da bir "x" terimi için bulunan sonuçların sayısı, F(y) Google'da bir "y" terimi için bulunan sonuçların sayısı, f(x,y) "xy" kelimelerinin Google'da birlikte bulunduğu sonuç sayısı ve "M" Google tarafından indekslenen web sayfalarının sayısını ifade etmektedir. Sonuçta bulunan GD (x,y) değeri, x ve y kelimelerinin birlikte aynı sayfalarda bulunma olasılığını ifade eder. Başka bir ifadeyle, bulunan bu değer iki örneği temsil eden kelime arasındaki anlam mesafesidir. Karşılaştırılan her özniteliğe ait örnekler bir listede birleştirilir. Listedeki her bir kelimenin diğer kelime ile olan yakınlık mesafesi "GD" (Partyka vd., 2010) formülü;

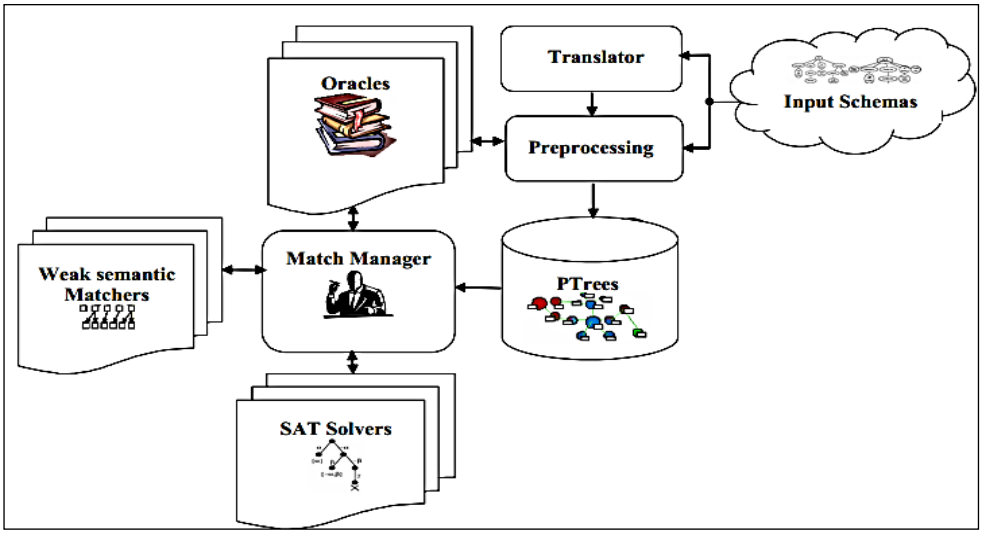
$$GD(x,y) = \frac{\max\{\log f(x), \log f(y)\} - \log f(x,y)}{\log M - \min\{\log f(x), \log f(y)\}} \quad (1)$$

şeklinde hesaplanır .

K-Medoid Kümeleme Algoritmasında GD formülüne göre hesaplanan örnekler arası mesafeler baz alınır. Daha sonra iterasyonlar yapılarak birbirine en yakın örnekler bulunur. Bulunan sonuçlara göre karşılaştırılan öznitelikler arasındaki benzerlik değeri belirlenmiş olur. Algoritma kapsamında öncelikle karşılaştırılan toplam örneklerin sayısına bağlı olarak küme sayısı belirlenir. Küme sayısı, karşılaştırılan iki öznitelik için toplam örneklerin sayısının uygulamanın ihtiyacına göre belirli yüzdesinin alınmasıyla belirlenir. Uygulamada kullanılan veri setiyle en uygun sonuç veren yüzdelik değer alınır. Küme sayısı belirlendikten sonra her küme için 'medoid' adı verilen başlangıç temsili değerler seçilir. Medoid'ler uygulamaya bağlı olarak, birbirine mesafesi (GD değeri) en küçük olan örnekler baz alınarak seçilir. Yani küme içinde diğer kelimelere uzaklığı en küçük olan örnek, medoid olarak seçilir. Geri kalan örnekler, kümelerin medoid değerleri ile karşılaştırılır ve medoid değeri ile arasında GD değeri en küçük olan kümeye dahil edilir. Bu şekilde sonuçlar aynı oluncaya kadar iterasyonlara devam edilir. Sonuçta oluşan her bir küme farklı bir tipi temsil eder ve benzer anlamda olan örnekler aynı kümede gruplandırılır. EBD değeri oluşan her küme için, yani her tip için hesaplanır.

1.6.2.2.1.2. S-Match

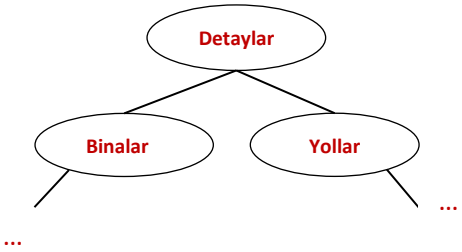
S-Match (URL-39, 2011) açık kaynak kodlu genel bir şema eşleştirme aracıdır. XML, RDF, TXT ya da OWL de kodlanmış iki şema arasındaki semantik ilişkileri harici bir sözlüksel veritabanı (WordNet) kullanarak bulur. S-Match, farklı şemalar arasında hem semantik hem de sentaktik eşleştirme yapabilmektedir. Eşleştirme yaparken şema noktalarının isimlerini çeşitli dilsel benzerlik teknikleriyle karşılaştırarak hem sentaktik eşleştirme hem de şema noktalarını temsil eden kavramları hesaplayarak anlamsal eşleştirme yapmaktadır. S-Match harici kaynak olarak WordNet ve GeoWordNet sözlüklerini kullanmaktadır. Kullanıcıya kendi uygulamasındaki ihtiyaca göre yazılımın konfigürasyonunu değiştirerek farklı bir dış kaynak (ontoloji, veritabanı, sözlük, vb.,) kullanabilme olanağı sunmaktadır. Bu kaynaklar ortak referans çatısı olarak kullanılarak karşılaştırılan iki farklı şema arasındaki ilişkiler saptanmaktadır.



Şekil 9. S-Match mimarisi (Giunchiglia vd., 2004)

Şekil 9'da S-Match mimarisi gösterilmektedir. S-Match, karşılaştırılacak şemaları çizge (ağaç) yapısında değerlendirir. Her girdi veritabanı şeması S-Match`in veri yapısına dönüştürülür. Şemadaki her terim, çizgede bir noktada temsil edilir. Noktaları birbirine bağlayan kenarlar alt, üst sınıf hiyerarşisini temsil eder. Çizgedeki her terim "nokta ismi" (label of node) olarak adlandırılır. Nokta isimlerinin kavramları ise "nokta kavramı" (concepts of node) olarak adlandırılır. Şekil 10'da örnek bir sınıf hiyerarşisinin S-Match'de

çizge yapısı gösterilmektedir. Bu şekil üzerinden S-Match kavram eşleştiricisinin çalışma adımları aşağıda açıklanmıştır:



Şekil 10. Örnek sınıf hiyerarşisi

S-Match eşleştirme adımları sırasıyla nokta isimlerinin kavramlarının oluşturulması, nokta kavramlarının oluşturulması, nokta isimleri arasındaki ilişkilerin hesaplanması ve nokta kavramları arasındaki ilişkilerin hesaplanması olarak dört ana adımdan oluşmaktadır. Nokta isimleri ve nokta kavramlarının oluşturulması eşleştirici çalıştırılmadan (off-line) önce gerçekleştirilen ön işlem aşaması (Preprocessing) olarak adlandırılmaktadır.

1. Ön İşlem Aşaması:
   - Tokenization: Çizgedeki her bir nokta ismi bir ayrıştırıcı (tokenizer) aracılığıyla noktalama işaretleri, boşluk, rakam vb., içeriyorsa ayrıştırılarak en küçük anlamlı parçasına (token) dönüştürülür.
   - Lemmatization: Birinci aşamada elde edilen her bir anlamlı kelime parçası (token) morfolojik analiz edilerek kelimenin temel kök hali (lemma) elde edilir. Örneğin, “Yol” ve “Bina” elde edilir.

1.1. Nokta isimlerinin kavramlarının oluşturulması:

Önceki adımda elde edilen kelime köklerinin kavramları WN sözlüğü kullanılarak elde edilir. Bir ismin kavramı o ismin WN`de bulunan anlamlarının tamamı ile ifade edilir. Tanımlama mantığında (Description Logic) bir ismin kavramı, o ismin WN`de bulunan bütün anlamlarının birleştirilmesi (Union) ile ifade edilir. Her bir isim için WN`de arama yapılır ve anlamları bulunur. Bulunan her bir anlam mantıksal birleştirme operatörü (“U”) ile birleştirilerek mantıksal formüller oluşturulur. Bu işleme nokta isimlerinin kavramsallaştırılması denir. Örneğin, Şekil 10’daki hiyerarşide “Detay” (Feature) sözcüğünün WN`de sekiz anlamı, “Yol”`un (Road) ise, “ulaşım için kullanılan kamuya açık yol” ve “birşeyi gerçekleştirmek için izlenen yol” olmak üzere iki anlamı vardır.

“Detay” sözcüğünün kavramı, WN’de bulunan sekiz anlamının birleştirilmesini ifade eden “C_{Detay} = < Detay, {anlamlar$_{wn}$ #8 >” ifadesiyle belirtilir. “Yol” sözcüğünün kavramı ise, “C_{Yol} = < Yol, {anlamlar$_{wn}$#2}>” ifadesiyle belirtilir. Bu şekilde çizgedeki her nokta isminin atomik kavramları elde edilir.

Çizgedeki bir noktanın isminin birden fazla kelimeden oluştuğu durumda, aradaki bağlaç, noktalama işaretleri vb., de mantıksal ifadelere dönüştürülür. Nokta isimlerinin bir önceki adımda elde edilen atomik kavramları bu bağlaçlarla mantıksal olarak birleştirilir. Bu şekilde nokta isimlerinin karmaşık (complex) kavramları oluşturulur.

Nokta isimlerinin kavramlarının oluşturulması çizgedeki diğer noktalardan bağımsız olarak yapılan ve o ismin ait olduğu noktanın konumu dikkate alınmadığı için “element” düzeyinde bir işlemdir.

1.2. Noktaların kavramlarının oluşturulması:

Karşılaştırılacak çizgelerdeki her noktanın kavramı, bulunduğu hiyerarşinin en üst kök noktasından başlayarak kendisine kadar ve kendisi de dahil olmak üzere, bir önceki adımda oluşturulmuş olan nokta isimlerinin kavramlarının mantıksal ifadelerle birleştirilmesi (conjunction) sonucu elde edilir. Yani, noktaların çizgedeki konumunun anlamı analiz edilir. Örneğin, şekil 10’daki hiyerarşide “Yol” nokta isminin bulunduğu noktanın kavramı “$C_{Yol} \cap C_{Detay}$” mantıksal formülü ile ifade edilir. Daha sonra anlam filtreleme fonksiyonu ile her nokta isminin bulundugu hiyerarşi dikkate alınarak filtreleme yapılır ve birbiri ile ilişkili olmayan anlamlar elenir. “Yol” için, sadece “detay” ile ilişkili olan “yol” anlamı saklanır, diğer anlamlar elenir. Ancak aynı hiyerarşi üzerinde bulunan bu kavramlar arasında hiçbir WN ilişkisi yoksa filtreleme yapılmadan, bulunan bütün anlamlar dikkate alınır.

Noktaların kavramlarının oluşturulması, her bir noktanın çizgedeki konumu ve hiyerarşide kendinden üstte bulunan noktaların da kavramlarının dikkate alınması nedeniyle “yapısal” (structure) düzeyde bir işlemdir.

Ön işlem aşamasında çizge semantik bilgi ile zenginleştirilmiş olur. Yani, hiyerarşi ağacının en üst kök noktasından başlayarak, aşağı doğru ve soldan sağa, sırayla bütün nokta isimlerinin ve noktaların kavramları hesaplanır. Bu şekilde noktaların kavramları oluşturulur. Sonuçta S-Match, hiyerarşi ağacında kodlanmış ilişkilerden noktaların kavramlarını bulmuş olur.

2. Eşleştirme Aşaması (Run Time)

2.1. Nokta isimleri arasındaki ilişkilerin hesaplanması

Bu aşamada eşleştirme yapılacak her iki şemanın ön işlem aşamasında elde edilen nokta isimlerinin kavramları arasındaki anlamsal ilişkiler bulunur. Bu ilişkilerin hesaplanmasında sırasıyla aşağıdaki eşleştiriciler kullanılır:

1. WordNet Eşleştiricisi: Anlam tabanlı bir eşleştiricidir. Karşılaştırılacak her iki şemadaki nokta isimlerinin WN'de eş anlamlılarını ve aralarındaki WN ilişkilerini bulur. Eşleştirici, WN'de herhangi iki A ve B kavramı arasında bulunan ilişkileri aşağıdaki kurallara göre S-Match semantik ilişkilerine dönüştürür:
   - WN'de A ve B kavramı arasında "synonymy" (eş anlamlı) ilişkisi varsa ya da iki kavram da aynı eş anlamlı sınıfında ise A, B'ye "eşittir" (equivalent) (A ≡ B),
   - WN'de A kavramı B kavramına, "hypernym" (üst kavram) veya "holonym" (parçası, üyesi) ilişkisi ile ilişkilendirilmiş ise A, B'den "çok geneldir" (more general) (A ⊃ B),
   - WN'de A kavramı B kavramına, "hyponym" (alt kavram, tipinde), "meronym" (parçası) veya "troponym" (tipinde, çeşidi) ilişkisi ile ilişkilendirilmiş ise, A, B'den "az geneldir" (less general) (A ⊂ B),
   - WN'de A ve B kavramı arasında "antonymy" (zıt anlamlı) ilişkisi varsa ya da aynı sınıfın farklı iki kavramı iseler A, B'den "ayrıktır" (disjoint) (A ⊥ B) ilişkileri üretilir.

WN eşleştiricisi, iki kavram arasında WN'de herhangi bir ilişki bulamazsa "Idk" (I don't know) cevabını döndürür. Bunun sonucu olarak aşağıdaki eşleştiriciler sırasıyla yürütülür.

2. Prefix: Karakter tabanlı eşleştiricilerdir. Karşılaştırılan sözcüklerden birinin diğeri ile başlayıp başlamadığını kontrol eder. Sözcüklerden biri başlangıç kısmında ikinci sözcüğü içeriyorsa "benzerdir" sonucu üretilir.
3. Suffix: Karakter tabanlı eşleştiricilerdir. Karşılaştırılan sözcüklerden birinin diğeri ile bitip bitmediğini kontrol eder. İlk sözcük son kısmında ikinci sözcüğü içeriyorsa "benzerdir" sonucu üretilir.
4. Edit distance: Karakter tabanlı eşleştiricilerdir. Karşılaştırılan iki sözcükten birinin diğer sözcüğe dönüştürülebilmesi için gerekli olan düzenleme (edit) operasyonlarının sayısını hesaplar. Bu operasyonlar ekleme, silme ya da yer değiştirme yapılacak harflerin sayısıdır.

5. Ngram: Karakter tabanlı eşleştiricilerdir. Karşılaştırılan iki sözcük arasında belirtilen sayı kadar yanyana gelen ortak harf sayısını hesaplar. Ortak harf sayısının fazlalığı iki sözcük arasındaki benzerliğin fazla olduğunu gösterir.
6. Hierarchy distance: Anlam tabanlı eşleştiricidir. Verilen bir hiyerarşide iki sözcük arasındaki mesafeyi ölçer. Bu mesafe, birinci sözcükten ikinci sözcüğe giderken noktalar arasındaki çizgilerin (arc) sayısıdır. Bulunan mesafe değeri verilen bir eşik değerden küçük ise "eşittir", büyük ise "Idk" ilişkisi döndürülür.
7. WordNet Gloss: Sözcük tanım kümesi (gloss) tabanlı bir eşleştiricidir. Karşılaştırılan iki sözcüğün WN'deki sözcük tanımlarını (cümle, tanım) alır ve ilk cümledeki sözcüklerin ikinci cümle içinde kaç defa bulunduğunu hesaplar.
8. Extended WordNet Gloss: Sözcük tanım kümesi tabanlı bir eşleştiricidir. İki sözcük için, ilkinin sözcük tanım kümesindeki sözcüklerin ikinci sözcüğün sözcük tanım kümesindeki sözcüklerin genişletilmiş tanım kümesi ile karşılaştırmasını yapar. Bu genişletişmiş sözcük tanımı, ilgili sözcük anlamının üst sınıflarının "is-a", "part-of" WN hiyerarşisindeki tanımlarından elde edilir.
9. Gloss Comparison: Sözcük tanım kümesi (gloss) tabanlı eşleştiricidir. Karşılaştırılan iki sözcüğün sözcük tanımı içinde bulunan aynı kelimelerin sayısını hesaplayarak benzerlik değeri bulur.
10. Extended Gloss Comparison: Sözcük tanım kümesi tabanlı eşleştiricidir. İki sözcüğün genişletilmiş sözcük tanım kümesini karşılaştırır. Birinci sözcük kümesi ikincisinin genişletilmiş sözcük kümesinde bulunan birçok ortak kelimeye sahipse, birinci sözcük ikinciden "çok geneldir" ilişkisi üretilir. Aksi takdirde "az geneldir" ilişkisi üretilir. Bulunan ortak sözcük sayısı belirlenen eşik değerin üzerinde ise "eşittir" ilişkisi üretilir.
11. Semantic Gloss Comparison: Sözcük tanım kümesi tabanlı eşleştiricidir. Karşılaştırılan iki sözcüğün sözcük tanım kümelerindeki sadece ortak kelimeleri değil, WN'de "is-a" (part-of) ilişkileri ile ilişkilendirilmiş olan sözcükleri de belirleyerek benzerlik değeri hesaplar.
12. Extended semantic gloss comparison: Sözcük tanım kümesi (gloss) tabanlı eşleştiricidir. Karşılaştırılan iki sözcükten birinci kelimenin sözcük tanımı ile ikinci kelimenin genişletilmiş sözcük tanımı içinde aynı olan kelimelerin ve "eşittir", "az geneldir" ve "çok geneldir" ilişkileri ile ilişkilendirilmiş kelimelerin de sayısını hesaplayarak benzerlik belirler.

13. Text Corpus: Karakter tabanlı eşleştiricilerdir. Bir sözcük öbeği içindeki ilk kelimenin ikinci kelimeye komşu alanda bulunma durumunu kontrol eder. Yeterli sayıda kelime bulunursa "eşittir" ilişkisi döndürülür.

2.2. Noktalar arasındaki ilişkilerin hesaplanması:

Yapısal (structure) düzey semantik eşleştirme aşamasıdır. Karşılaştırılan iki noktanın kavramları arasındaki semantik ilişkinin geçerliliği, arka planda bir teori (context) varsayılarak test edilir. Bu teori, bir önceki bölümde karşılaştırılan nokta isimleri kavramlarının arasında bulunan ilişkilerden çıkarsanarak oluşturulur ve arka plan bilgi tabanı olarak kullanılır. Karşılaştırılan şema noktaları arasındaki semantik ilişkilerin bulunması için yukarıdaki bölümde bulunan bütün benzerlik ilişkileri "propositional connectives" lere (birleştirici) dönüştürülür ve sonra oluşan "Context → rel(Ci, Cj)" formülünün geçerliliği ispat (prove) edilir. Ci, 1.çizgedeki "i" noktasının kavramı, Cj ise 2.çizgedeki "j" noktasının kavramıdır. "Rel", "Ci**"** ve "Cj**"** arasında bulunan ispat edilmek istenen semantik ilişkidir. "Context", "Ci**"** ve "Cj" arasında bulunan bütün nokta isimlerinin kavramları arasındaki ilişkilerin mantıksal kesişimidir (conjunction). Oluşan mantıksal formüller her semantik ilişki tipi (az genel, çok genel, ayrık) için test edilir. Bu geçerlilik testi teorem ispatlayıcısı olan "SAT"[6] (Propositional Satisfiability) ile gerçekleştirilir. SAT, oluşturulan mantıksal formüllerin tersini (negation) alır ve ispat etmeye başlar. Sonuç olumlu ise, yani formülün tersi ispat edilemiyorsa, mantıksal formül geçerlidir ve test edilen ilişki kesin sonuç olarak değerlendirilerek kullanıcıya döndürülür. Sonuç olumsuz ise test edilen ilişki artık geçerli değildir, dikkate alınmaz. Böylece eşleştirme problemi mantıksal geçerlilik (validity) problemine dönüştürülerek sonuç semantik ilişkiler bulunur.

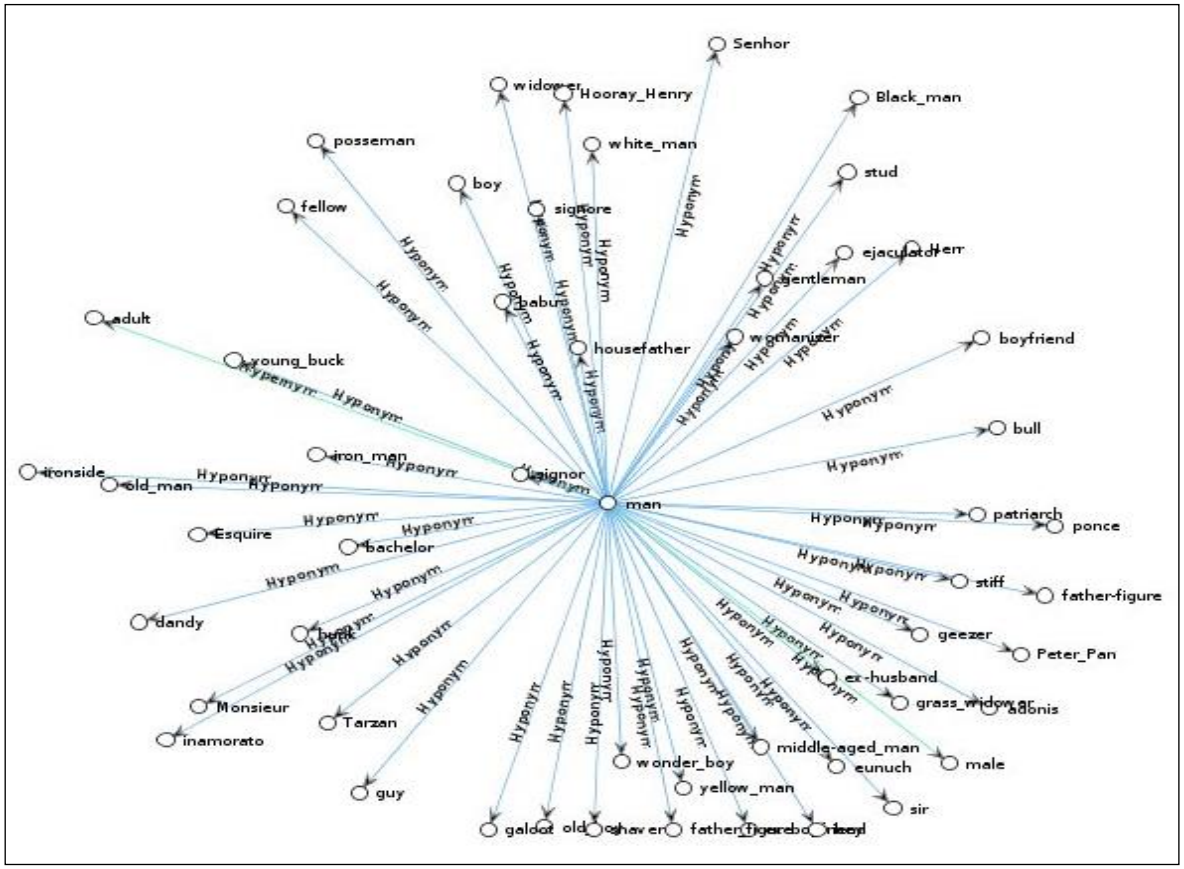
S-Match semantik eşleştirme işlemlerinde varsayılan arka plan bilgi tabanı olarak WN sözlüğünü kullanır. Sıradaki bölümde WN hakkında detaylı bilgi verilecektir.

1.6.2.2.1.2.1. WordNet

WordNet (Miller, 1990) toplam 117 bin eş anlamlı sözcük grubuna sahip bir ingilizce sözlüksel veritabanıdır. Birçok semantik uygulamada arka planda bilgi tabanı olarak kullanılmaktadır. WN yapısındaki her isim, fiil, sıfat ve zarf eş anlamlı gruplar (synsets)

[6] http://en.wikipedia.org/wiki/Satisfiability#Propositional_satisfiability

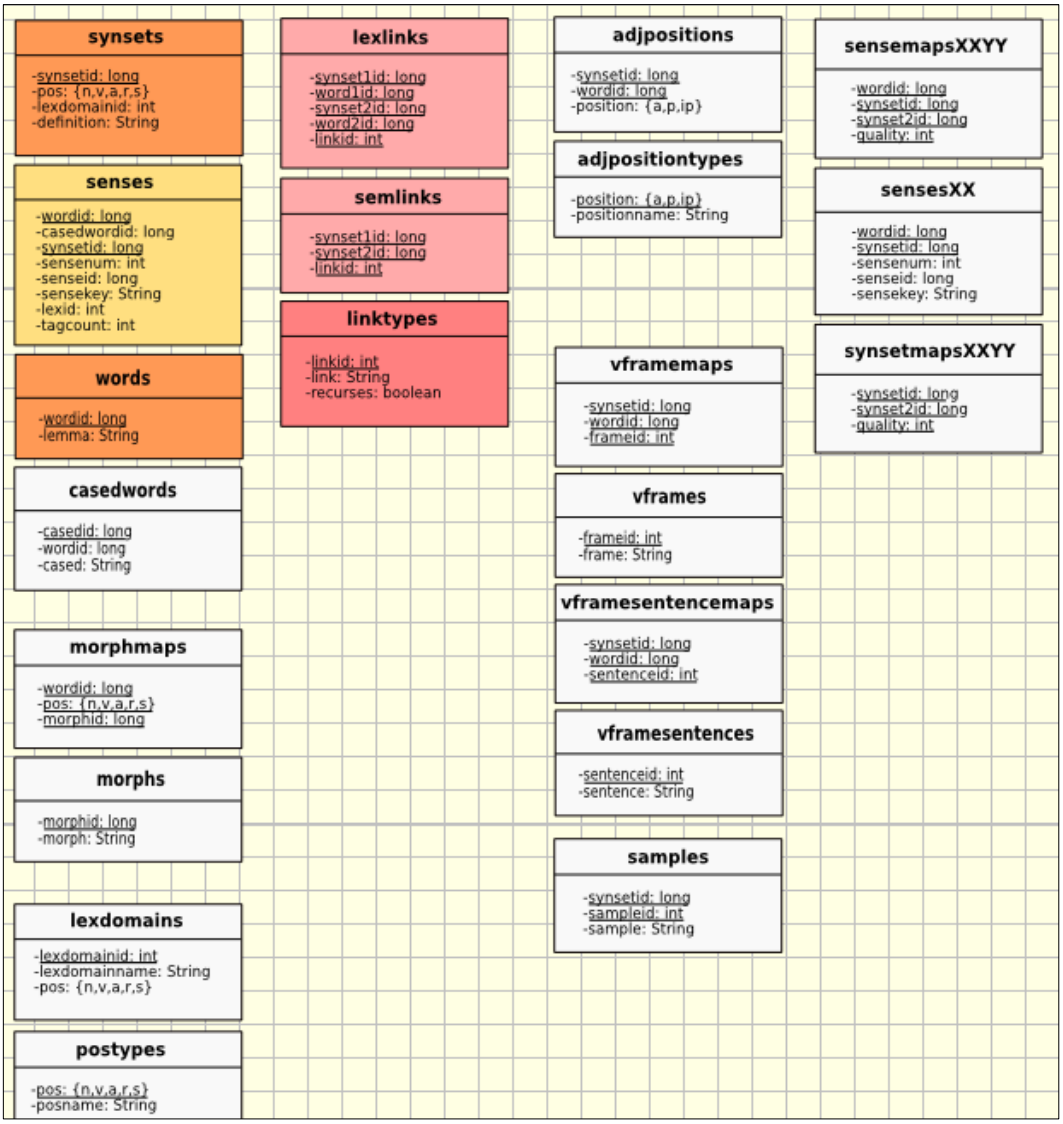
halinde hiyerarşik bir ağaç yapısında tutulmaktadır. Her bir eş anlamlı grup farklı bir anlamı ifade etmektedir. Eş anlamlı gruplar anlamsal (kavramsal) ve sözlüksel ilişkilerle birbirine linklenmiştir. WN hiyerarşisindeki anlamsal ilişkiler; hyponym (tipinde), meronym (parçası), troponym (çeşit), hypernym (üst sınıf), holonym (parçası, üyesi), synonymy (eşanlamlı), antonymy (zıt anlamlı) gibi ilişkilerdir. "Participle", "also see" ve "pertainym" (ilişkili) ise sözlüksel ilişkilerdir. Her eş anlamlı grup, "gloss" adı verilen kısa bir tanım kümesi ya da ilgili kelimeleri içeren bir ya da birden fazla örnek cümle içerir (URL 40, 2011).



Şekil 11. WN veritabanının çizge şeklinde gösterimi (URL 41, 2011)

Şekil 11'de WN veritabanının çizge yapısı gösterilmektedir. WN standard veritabanı "ASCII" formatındadır. Veritabanı dosyaları "index.noun", "data.noun", "index.verb", "data.verb", "index.adj", "data.adj", "index.adv", "data.adv" dosyalarıdır. İndeks dosyaları, WN içindeki bütün kelime formlarının alfabetik listesidir ve bütün verileri ilişkisel "pointer" lar aracılığıyla içerir. Bu "pointer"lar ilgili kelimelerin veri dosyasındaki

adreslerini belirtir. Şekil 12'de WN'nin ilişkisel temsilini idafe eden MySQL veritabanı şeması gösterilmektedir.



Şekil 12. MySQL WN veritabanı şeması (URL-42, 2015).

WN'nin içeriğinin çok genel olması, konumsal alana özgü sözcüklerde sınırlı olması ve konumsal uygulamalarda gerekli olan enlem, boylam koordinatları gibi birçok önemli bilgiyi içermemesi nedeniyle GeoWordNet (GWN) (Giunchiglia vd., 2009) oluşturulmuştur. GWN; WN, GeoNames ve MultiWordNet'in (URL-43, 2015) italyanca kısımlarının birleştirilmesi ile oluşturulmuş çok dilli bir konumsal ontolojidir. İlişkisel, sözlüksel ve RDF formatında paketleri oluşturulmuştur (URL 44, 2011).

2. YAPILAN ÇALIŞMALAR

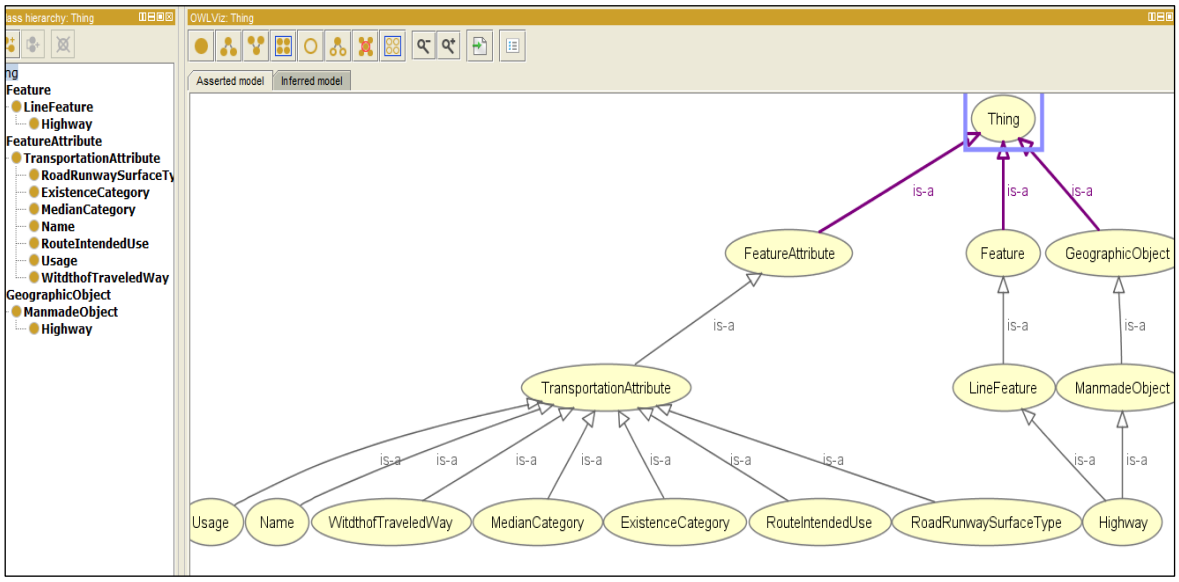
Bu tez çalışmasında konumsal veritabanı şemalarının semantik eşleştirilmesine yönelik bir gerçekleştirim yapılmış ve web servislerinin Semantik Web Teknolojilerine (SWT) dayalı olarak kompozisyonu için bir mimari önerilmiştir. İlerleyen bölümlerde bu mimari ve mimarinin temel bileşenleri açıklanacaktır.

2.1. Semantik Eşleştirme Gerçekleştirimi

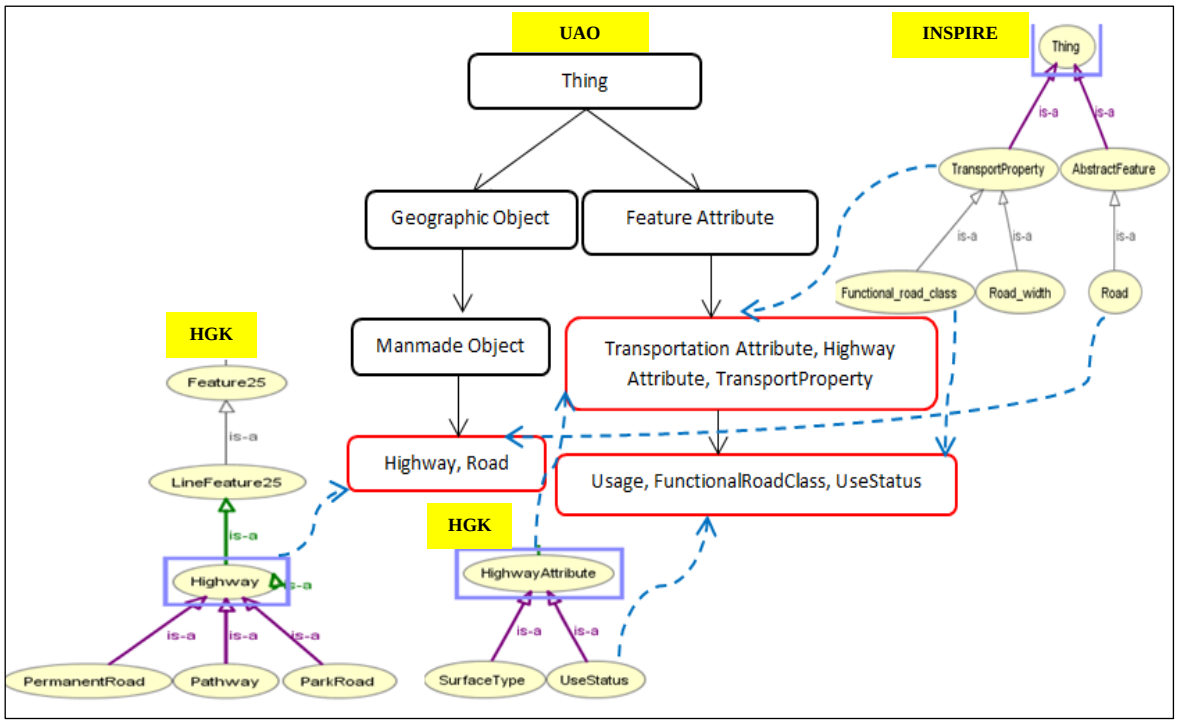
Tez çalışması kapsamında farklı kurumlar tarafından sunulan konumsal veritabanı şemalarının aralarındaki benzerliklerin semantik eşleştirme ile bulunmasına yönelik bir gerçekleştirim yapılmıştır. Bu gerçekleştirim kapsamında, Harita Genel Komutanlığı (HGK) (URL-45, 2015) Karayolu Şeması ve INSPIRE (Infrastructure for Spatial Information in Europe) Road Transport Network (RTN) GML Uygulama şeması (URL-46, 2011) seçilmiş ve aralarında semantik eşleştirme yapılmıştır. Bu şemalardan HGK ve INSPIRE RTN ontolojilerinin üretilmesi Kara vd., (2012) çalışmasında gerçekleştirilmiştir. Üretilen ontolojiler bu tez çalışmasındaki eşleştirme gerçekleştiriminde kullanılmıştır.

Semantik eşleştirmenin başarılı bir şekilde gerçekleştirilebilmesi için karşılaştırılacak kaynak ve hedef ontolojilerini uygun bir şekilde kapsayan arka plan bilgi kaynaklarına ihtiyaç vardır. Literatürde WN, DOLCE (URL-47, 2013), SUMO (URL-48, 2012), Cyc (URL-49, 2013) gibi genel içerikli ve geniş çaplı kaynaklar mevcuttur. Ancak bu kaynaklar konumsal alan gibi özel alanları semantik eşleştirme için gerekli derinlikte kapsamamaktadır. Bu yüzden, çalışma kapsamında konumsal alana özgü kavramları içeren ve eşleştirme işlemine yetecek derinlikte bir Ulaşım Alan Ontolojisi (UAO) oluşturulmuştur (Şekil 13). Oluşturulan UAO, semantik eşleştirme işlemi için S-Match yazılımında varsayılan arka plan bilgi tabanı olan WN yerine yapılandırılarak kullanılmıştır. UAO oluşumunda konumsal detayların referanslandırılması için VMap2 ulaşım temasının yol tanımları, öznitelikleri ve Klien (2008) "Geographic Object" sınıflandırması kullanılmıştır. HGK şemasının katman tabanlı, yani geometri tabanlı olarak tasarımından dolayı UAO'da "Feature" ve "Line Feature" sınıfları tanımlanmıştır. UAO, S-Match'de arka plan bilgi tabanı olarak kullanılmak amacıyla WN yapısına

dönüştürülmüştür. Çalışmada UAO'dan WN oluşturmak için extJWNL (extended Java WordNet Library) yazılımı kullanılmıştır. Açık kaynak kodlu "java api"si olarak tanımlanan ExtJWNL, içinde WN ve benzeri yapılara erişmek ve onları düzenlemek için ewn (edit wordnet) komut satırı aracını içerir (URL-50, 2012). UAO WN'de, INSPIRE RTN ontolojisi ve HGK ontolojisinde bulunan kavramlar ve öznitelikler arasındaki ilişkiler WN ilişkileri (bkz. 1.6.2.2.1.2.1.) kullanılarak oluşturulmuştur. Örnek bir sınıfın UAO WN'deki yapısı ve HGK ile INSPIRE ontoloji sınıfları ile ilişkilendirilmesi şekil 14'de gösterilmektedir. Sınıflar arasındaki mavi renkli ilişkiler WN "eş anlamlı" ilişkisini temsil etmektedir.

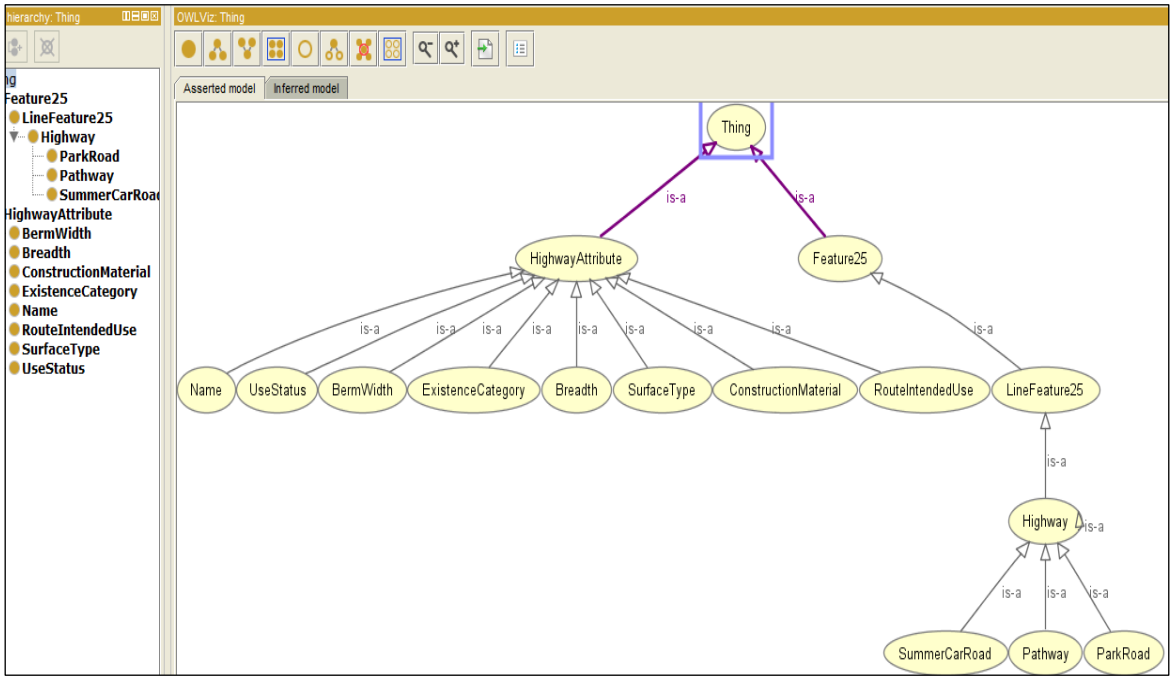


Şekil 13. Oluşturulan Ulaşım Alan ontolojisinin bir kısmı

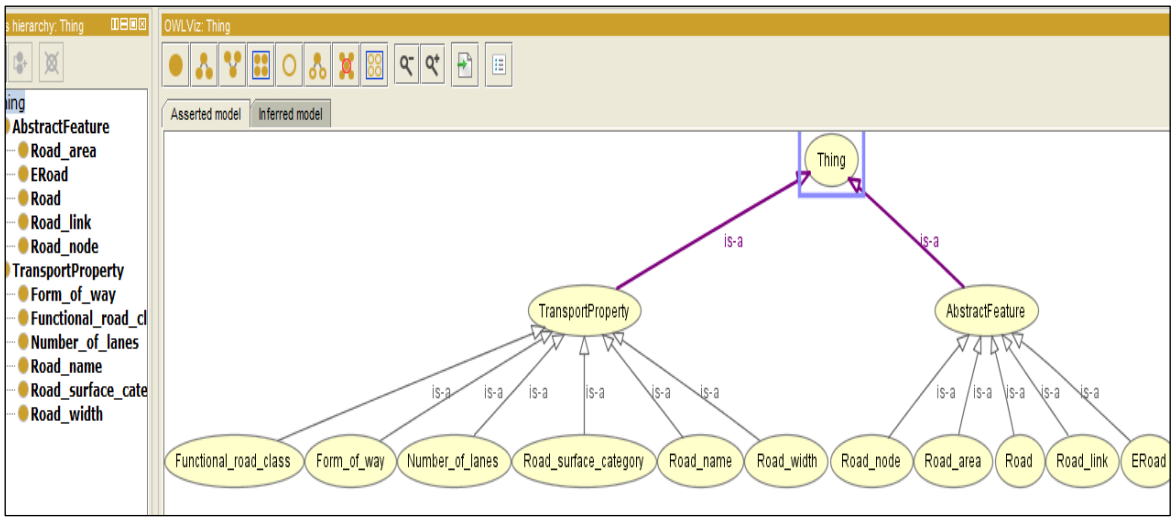


Şekil 14. UAO'da HGK ve INSPIRE ontolojilerinin ilişkilendirilmesi

Şekil 15 ve 16'da ulaşım ve yol alanında farklı kavramsallaştırmalara sahip bu iki ontolojinin bir kısmı Protege OWLViz eklentisinde gösterilmektedir.



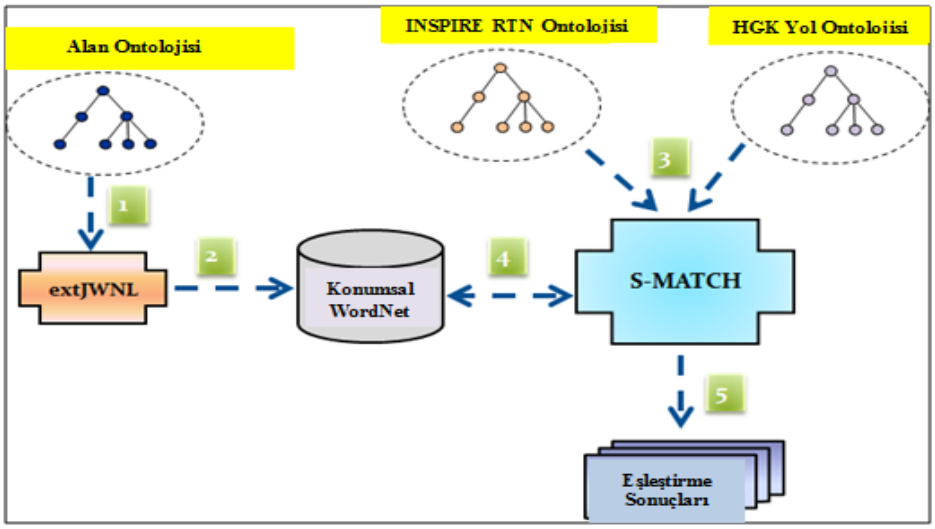
Şekil 15. HGK yol ontolojisi



Şekil 16. INSPIRE RTN Ontolojisi

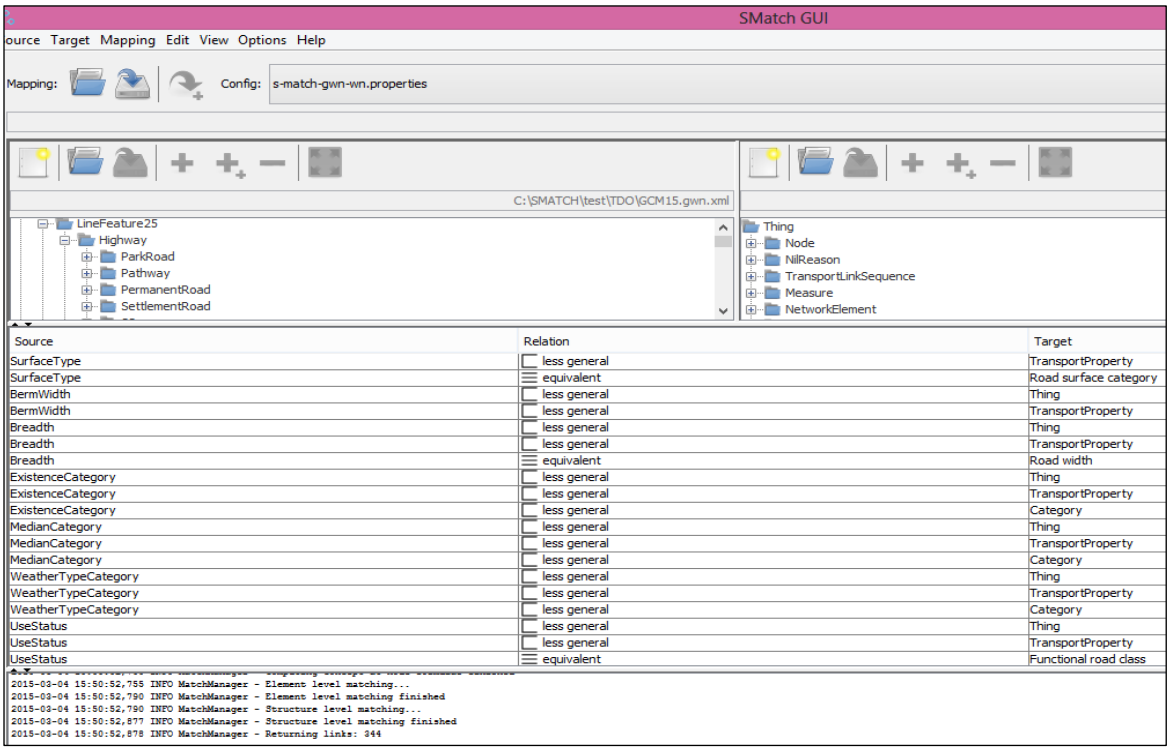
Protégé, açık kaynak kodlu ontoloji editörü ve bilgi tabanı (knowledgebase) dır. Stanford Üniversitesi tarafından geliştirilmiştir (URL-51, 2008). Ontoloji oluşturmak için ontoloji editörü ihtiyacını gidermek amacıyla geliştirilmiştir. Ontoloji oluşturma Protégé' nin eklentisi olan OWL Plugin' i ile gerçekleştirilmektedir (Knublauch vd., 2004). Protege, ontoloji oluşturma, görselleştirme ve işleme yapabilen ve bunları değişik formatlarda sunabilen (clips, rdf, xml, OWL ve ilişkisel veri tabanları) birçok eklenti (plug-in) ile genişletilebilen, kullanıcı dostu (user friendly) bir yazılımdır. Kendi içinde bilgi tabanı ve çıkarsama araçları ile (reasoners) iletişimi sağlayan DIG ara yüzüne sahiptir (URL-51, 2008).

Şekil 17'de, gerçekleştirilen semantik eşleştirme mimarisinin temel adımları gösterilmektedir. Birinci adım, oluşturulan UAO'nın extJWNL aracı ile WN yapısına dönüştürülmesidir. İkinci adım, INSPIRE RTN ve HGK (GCM) ontolojilerinin S-Match'e girdi olarak yüklenmesidir. S-Match, yüklenen iki ontolojiyi kendi iç temsili olan ağaç yapısına dönüştürür. Üçüncü adım, oluşturulan UAO yardımıyla S-Match'in bu iki ontolojinin içerdiği her nokta çiftini karşılıklı olarak, sırayla karşılaştırarak aralarındaki semantik benzerlikleri hesaplamasıdır. S-Match yapısı ve eşleştirme algoritmaları hakkında detaylı bilgi genel bilgiler bölümünde (bkz. 1.6.2.2.1.2.) verilmiştir.



Şekil 17. Semantik eşleştirme mimarisi

Şekil 18' de, S-Match ile sol pencerede HGK ontolojisi, sağ pencerede INSPIRE ontolojisi ve alt pencerede bu iki ontoloji arasında bulunan semantik benzerlik ilişkileri gösterilmektedir.



Şekil 18. S-Match'de eşleştirme gerçekleştirimi

Tablo 1. UAO kullanıldığında eşleştirme sonuçları

Inspire TN Ont. / GCM ont.	AbstractFeature	Road	TransportProperty	Road name	Road width	Category	Road Surface Category	Functional road class
Feature25	≡	⊃	-	⊃	⊃	-	⊃	⊃
LineFeature25	⊂		-	-	-	-	-	-
Highway	⊂	⊂	-	-	-	-	-	-
Pathway	⊂	⊂	-	-	-	-	-	-
Summer CarRoad	⊂	⊂	-	-	-	-	-	-
Settlement Road	⊂	⊂	-	-	-	-	-	-
Permanent Road	⊂	⊂	-	-	-	-	-	-
Park Road	⊂	⊂	-	-	-	-	-	-
HighwayAttribute	-	-	≡	⊃	⊃	-	⊃	⊃
SurfaceType	-	-	⊂	-	-	-	≡	-
Breadth	-	-	⊂	-	≡	-	-	-
BermWidth	-	-	⊂	-	-	-	-	-
ExistenceCategory	-	-	⊂	-	-	⊂	-	-
MedianCategory	-	-	⊂	-	-	⊂	-	-
WeatherTypeCategory	-	-	⊂	-	-	⊂	-	-
UseStatus	-	-	⊂	-	-	-	-	≡
Route Intended Use	-	-		-	-	-	-	-
Name	-	-	⊂	≡	-	-	-	-

Tablo 1'de S-Match yazılımı arka plan bilgi tabanı olarak çalışmada oluşturulan UAO WN ile kullanıldığında yol sınıfı ve öznitelikleri için eşleştirme sonuçları daha ayrıntılı bir şekilde gösterilmektedir. Bulunan "çok geneldir" (⊃), "az geneldir" (⊂), "eşittir" (≡) semantik ilişkileri doğru sonuçlardır. Tablo 2, S-Match yazılımı varsayılan arka plan bilgi tabanı olarak WN ile kullanıldığında HGK ve INSPIRE ontolojileri arasında bulunan eşleşme sonuçlarını göstermektedir. Tablo 3, S-Match yazılımı arka plan bilgi tabanı olarak GWN ile kullanıldığında HGK ve INSPIRE TN ontolojileri arasında bulunan eşleşme sonuçlarını göstermektedir. Üç tablodaki sonuçlara bakıldığında "yol" ve öznitelikleri için en çok ve en doğru eşleşme sonucunun UAO kullanıldığında elde edilen sonuçlar olduğu görülmektedir. Örneğin, "Highway Attribute" ve "Transport Property" kavramları, "Use Status" ve "Functional Road Class" kavramları, "Breadth" ve "Road Width" kavramları, "Surface Type ve "Road Surface Category" ikili kavramları arasında WN ve GWN kullanıldığında hiçbir ilişki bulunamazken, UAO kullanıldığında doğru ilişkiler bulunmuştur. Bunun nedeni çalışmada oluşturulan UAO WN'de HGK ve INSPIRE ontolojilerinin WN ilişkileri ile yukarıda da anıldığı üzere ilişkilendirilmiş olmasıdır. Ayrıca, WN kullanıldığında Tablo 2'de görüldüğü üzere "LineFeature25" ile "AbstractFeature" ve "Road" kavramları arasında bulunan "eşittir" (≡) ilişkisi yanlış bulunan sonuçlardandır. Tüm sonuçlara bakıldığında, şema eşleştirme işleminde mevcut genel kavramları içeren kaynaklar yerine, arka plan bilgi tabanı olarak karşılaştırılacak

şemaları kapsayan, uygulama alanına özgü konumsal bir ontolojinin kullanılması gerektiği anlaşılmaktadır.

Tablo 2. WN kullanıldığında eşleştirme sonuçları

Inspire TN Ont. / GCM ont.	AbstractFeature	Road	TransportProperty	Road name	Road width	Category	Road Surface Category	Functional road class
Feature25	≡	⊃	-	⊃	⊃	-	⊃	⊃
LineFeature25	≡	≡	-	⊃	⊃	-	⊃	⊃
Highway	⊂	⊂	-	-	-	-	-	-
Pathway	⊂	⊂	-	-	-	-	-	-
Summer CarRoad	⊂	⊂	-	-	-	-	-	-
Settlement Road	⊂	⊂	-	-	-	-	-	-
Permanent Road	⊂	⊂	-	-	-	-	-	-
Park Road	⊂	⊂	-	-	-	-	-	-
HighwayAttribute	-	-	-	-	-	-	-	-
SurfaceType	-	-	-	-	-	⊂	-	-
Breadth	-	-	-	-	-	-	-	-
BermWidth	-	-	-	-	-	-	-	-
ExistenceCategory	-	-	-	-	-	⊂	-	-
MedianCategory	-	-	-	-	-	⊂	-	-
WeatherTypeCategory	-	-	-	-	-	⊂	-	-
UseStatus	-	-	-	-	-	-	-	-
Route Intended Use	-	-	-	-	-	-	-	-
Name	-	-	-	-	-	-	-	-

Tablo 3. GWN kullanıldığında eşleştirme sonuçları

Inspire TN Ont. / GCM ont.	AbstractFeature	Road	TransportProperty	Road name	Road width	Category	Road Surface Category	Functional road class
Feature25	≡	⊃	-	⊃	⊃	-	⊃	⊃
LineFeature25	⊂	-	-	-	-	-	-	-
Highway	⊂	-	-	-	-	-	-	-
Pathway	⊂	-	-	-	-	-	-	-
Summer CarRoad	⊂	⊂	-	-	-	-	-	-
Settlement Road	⊂	⊂	-	-	-	-	-	-
Permanent Road	⊂	⊂	-	-	-	-	-	-
Park Road	⊂	⊂	-	-	-	-	-	-
HighwayAttribute	-	-	-	-	-	-	-	-
SurfaceType	-	-	-	-	-	-	-	-
Breadth	-	-	-	-	-	-	-	-
BermWidth	-	-	-	-	-	-	-	-
ExistenceCategory	-	-	-	-	-	⊂	-	-
MedianCategory	-	-	-	-	-	⊂	-	-
WeatherTypeCategory	-	-	-	-	-	⊂	-	-
UseStatus	-	-	-	-	-	-	-	-
Route Intended Use	-	-	-	-	-	-	-	-
Name	-	-	-	-	-	-	-	-

INSPIRE yol şemasının HGK karayolu şemasından daha genel olması gibi, aynı veriyi tanımlayan farklı konumsal veritabanı şemalarının farklı derinliklerde oluşturulması, eşleştirme sonucunda bazı kavramların daha özel sınıflar yerine çoğunlukla daha genel sınıflarla eşleşmesine neden olmaktadır. Ayrıca HGK yol şemasında olduğu gibi, şemadaki

kavram isimlerinde bulunan kısaltmalar, rakamsal değerler gibi sadece şemayı tanımlayan kişilerin ya da o alanda uzman kişilerin anlayabileceği tarzda özel ifadeler kullanılması da başka bir problemdir. Bu nedenle literatürde mevcut genel içerikli kaynaklar yerine, uygulama alanına özgü oluşturulmuş UAO gibi bir kaynağın arka plan bilgi tabanı olarak kullanılması bu problemlerin azaltılması bakımından önemlidir.

2.2. KSO Desteğinde WSK Geliştirme Mimarisi

Web Servisi Kompozisyonu (WSK) , birden çok Web servisinin, bir araya getirilerek yeni bir web servisinin oluşturulması demektir. WSK, Web Servisleri (WS) alanında son on yılın en aktif araştırma konularından biri olmuştur. Bu alanda çok çeşitli akademik ve pratik çalışmalar bulunmakla birlikte, Tam Otomatik WSK (TOWSK) henüz çözümlenebilmiş bir konu değildir. TOWSK ile kastedilen, servislerin bir araya getirilmesi ve oluşan yeni servisin koşum işleminin, insan müdahalesi olmadan yürütülmesidir.

TOWSK'nın henüz ulaşılamamış bir hedef olması, içerdiği bir dizi sorunun insan müdahalesi olmadan çözümlenebilmesinin, eğer imkansız değilse, çok zor olmasındandır. Çünkü burada söz konusu olan, elindeki bir CBS uygulamasını, ister masaüstü, isterse web CBS tarzında gerçekleştirecek bir kullanıcının, "insan" olarak yaptığı bütün müdahalelerde yüklü olan "semantiğin", yarı otomatik ya da tam otomatik tarzdaki iş akışına "gömülmesi"dir. Bu çok kolay bir iş değildir ve içerdiği bir dizi problem vardır. Bu tez çalışmasının amacı, bu problemlerden bazıları için, SWT kullanılarak belirli çözüm önerilerinin sunulmasıdır. Söz konusu problemler, aşağıda açıklanmaktadır.

1. WSK'da iş akış sırasının belirtilmesi problemi:

Burada "amaç" (goal) ile kastedilen, WSK sonucunda elde edilecek "kompoze" servisin yapacağı iştir. Örnek "amaç" lar olarak şunlar verilebilir: "Trabzon'da katı atık depolama alanı için uygun yerleri bul", "Bodrum'da kültür balık çiftlikleri için uygun alanları bul", "Kandahar'a 500 mil uzaklıkta, C5A uçaklarının inişine uygun hava sahalarını göster", "İzmir"de "metrekareye 10 cm^3" yağış sonucunda oluşabilecek taşkın alanlarını bul", "İstanbul'da Büyükşehir belediyesine gelen bir yapı ruhsatı isteğinin, kentin siluetine aykırı olup olmadığı bul".

Burada geleneksel yaklaşım, iş akışının kullanıcı yani insan tarafından belirlenmesidir. Geliştirici ya da kullanıcının kompoze servisi oluşturan bütün bileşen servislerin girdi ve çıktı parametrelerinin koşum sırasında eşleştirilmesidir (Klien vd., 2006 ; Cömert vd.,

2010; Brogi vd., 2006; Lemmens, 2006; Brogi ve Corfini, 2007; Alam vd., 2007, Dong vd., 2006). Ama bu çoğu zaman çok kolay değildir. Çünkü hangi servislere, hangi koşum sırasında ihtiyaç duyulduğu, bu servislerin birbiri ile uyumlu olup olmadığına karar verecek kullanıcının teknik olarak çok donanımlı olmasını gerektiren bir durumdur. Öte yandan bu durum, günümüz "uygulama geliştirme" eğilimlerinde en temel faktör olan "hızlı geliştirme" bakımından da kabul edilebilir bir tarz olmayacaktır.

Bu tezde önerilen mimarinin yaklaşımı ise, kullanıcının amacını anılan geleneksel çalışmalardan çok daha üst düzeyde, fakat bir sonraki adımda amaç tümcesinin bileşenlerine ayrılmasına yardımcı olacak bir formda, belirtmesidir. O nedenle burada önerilen yöntem, geleneksel yönteme göre değerlidir.

2. Önerilen mimaride kullanıcının sağlaması gereken bilgi:

Önerilen mimarinin gereklerinden biri "Konumsal Servis Ontolojisi (KSO)" dir. Bu mimarinin dayandığı yaklaşımın esası, kullanıcının "bir form"da belirttiği "amaç" cümlesinden, KSO'daki "kompoze servis"in[7] bulunması ilkesidir; KSO, kullanıcı amacına karşılık gelen servisin hangi bileşen servislerden hangi koşum sırası seçenekleri dahilinde oluştuğunu gösterecektir. Bundan sonra kullanıcı, bileşen servis meta verilerine uyan servisleri web üzerinden tek tek bulacak ve koşuma hazır kompoze web servisini oluşturmuş olacaktır. Doğal olarak böyle bir mimari bir dizi yazılım aracına ihtiyaç duyacaktır. Bu araçların her birinin geliştirilmesi bu tezin kapsamını aşan bir konudur. O nedenle bu konu, gelecek çalışmalara bırakılmıştır.

Bu tez çalışmasında önerilen mimaride hedeflenen kullanıcının sadece üst düzeyde bir "amaç" (goal) cümlesi vermesidir. WSK için gerekli bilgiler bu cümleden çıkarsanacaktır. Bu işin bir boyutu, amaç tümcesinin nasıl karakterize edileceğidir. Diğer boyutu ise, kullanıcı "amaç" cümlesinden "servis meta verisi" nin çıkarsanmasıdır. Söz konusu "meta veri" nin ne olacağı, başka bir çalışmanın konusu olabilecek boyutta kapsamlı bir konudur. Burada yalnızca genel olarak hangi bileşenlerden oluşacağı belirtilmek istenirse şöyle söylenebilir:

- Servis tipi: Servisin KSO'daki tip ismidir. Örneğin "yer seçimi"

[7] KSO'daki bu servis tanımına bazı kaynaklar "soyut (abstract)" servis demektedir (Dong vd., 2006). Bu niteleme Nesne Yönelimli (Object Oriented) tasarım bağlamında doğru değildir. Ancak WSK için tasarım anındaki WSK yı ifade etmek üzere kullanılmaktadır. Nesne Yönelimli tasarımda "soyut sınıf (abstract class)", bir "örneği" yaratılamayan obje tipi demektir ve tasarım amaçları için gereklidir. Oysa burada, yani servis ontolojisi bağlamında, kompoze servis bir defa tanımlandığında ve oluşturulup bir servis kataloğunda yayımlandığında, artık kendi tipinde bir örneği bir yerlerde bulunan bir servis tipini tanımlıyor durumda olacaktır.

- Servis öznitelikleri: Servisin OWL-S tanımında yer alan özniteliklerdir. Örneğin, servisin içerdiği "seçim koşulu".

3. WSK'da servis parametrelerinin birbiri ile eşleştirilmesi (şema eşleştirme) :

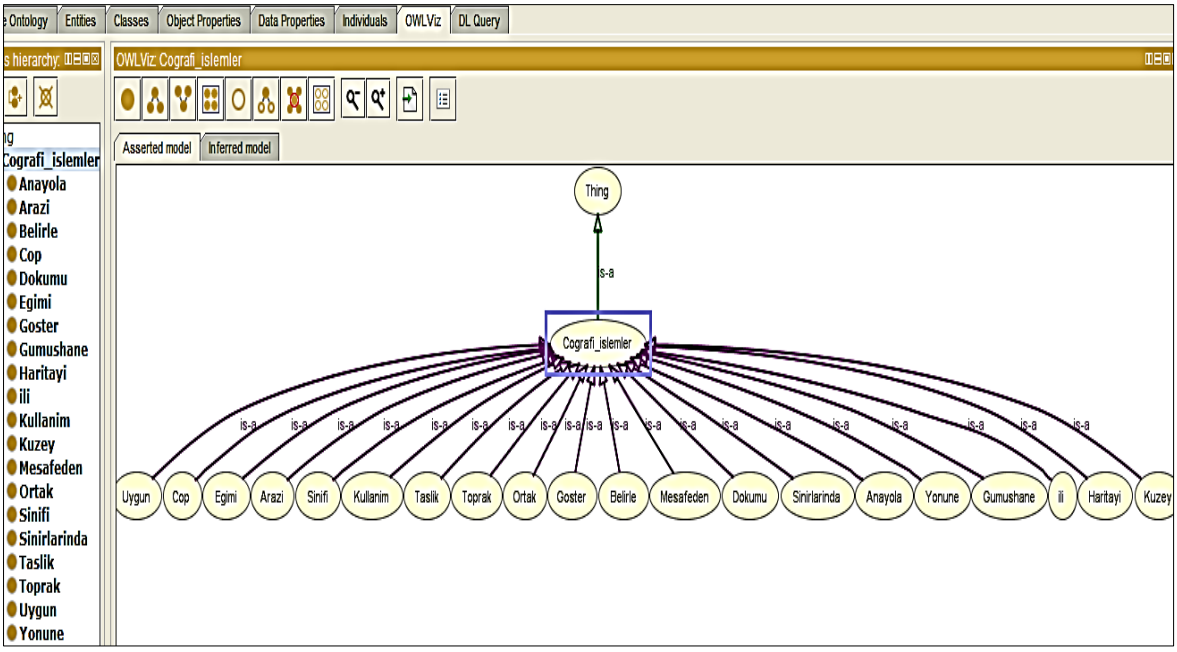
Web servisi kompozisyonunda en önemli problemlerden bir diğeri, kompozisyonu oluşturan servisler arasında verinin bir servisten diğerine "uygun bir şekilde" aktarılmasıdır. Öyle ki, bir servisin birden çok çıktı parametresi olması durumunda, bu parametreleri girdi olarak alacak bir servis için, parametrelerin ne şekilde birbirine karşılık geldiği bilinmelidir. Diğer yandan, birbirine karşılık gelen parametrelerin "tip" leri de eşleşmeli ya da uyumlu olmalıdır. Geleneksel olarak her iki uyum da servis kompozisyonu sırasında insan tarafından sağlanır. Bu da bu işi yapacak geliştiricinin veri şemalarını çok iyi bilmesini gerektirir ki bu da yukarıda anılan dezavantajı içeren bir durumdur. Bu tezde önerilen mimaride parametrelerin eşleştirilmesi şuanda manuel'dir ancak semantik şema eşleştirme ile çözülmesi öngörülmektedir. Bu çözüm S-Match yazılımının WSK içine entegrasyonunun yapılmasıyla sağlanacaktır.

2.2.1. WSK Geliştiricisinin Amacını Yüksek Bir Düzeyde Söyleyebilmesi

Önerilen mimarinin bileşenlerinden biri WSK geliştiricisinin amacını yüksek bir düzeyde söyleyebilmesidir. Bunun için genellikle üst düzeyde parametreleri belirlenmiş bir form ara yüzüne (Hochmair ve Fu, 2006) ya da geliştiricinin kurduğu üst düzey cümleyi doğrudan değerlendirip çizge formatına dönüştüren bir yazılıma (URL-52, 2015; URL-53, 2015) ihtiyaç vardır. Bu çalışmada ikinci yaklaşım esas alınmıştır.

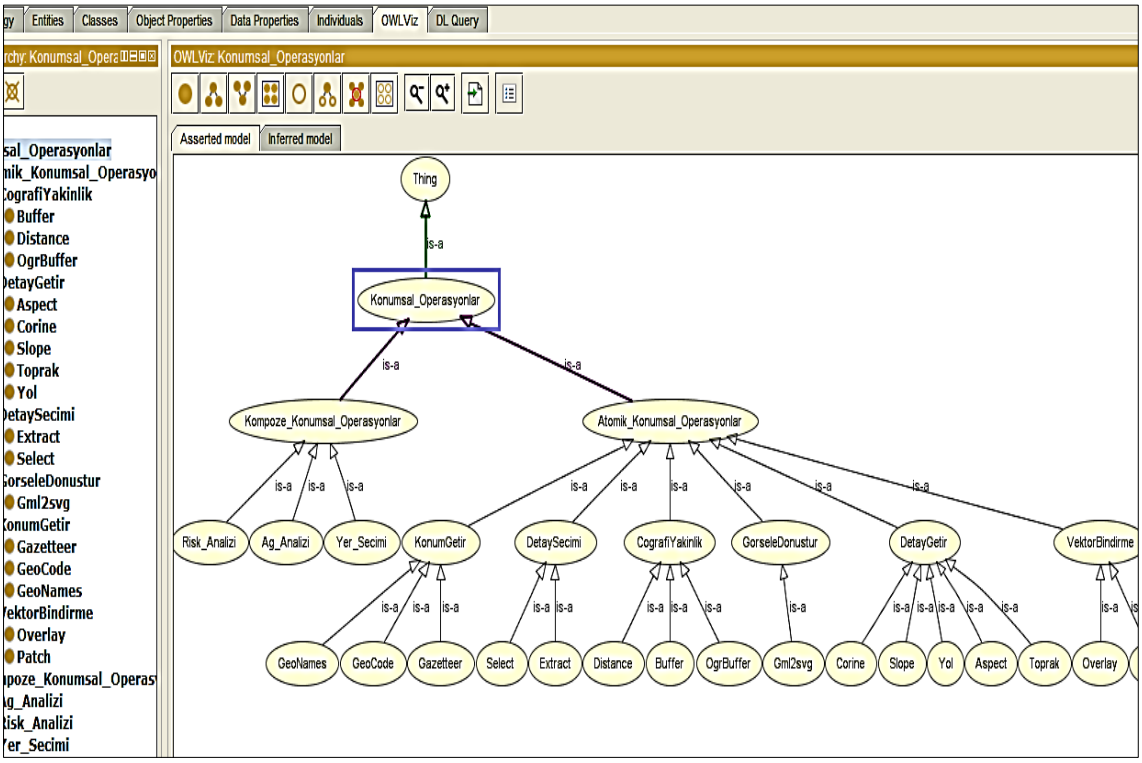
Yer seçimi ("site-selection") işlemi, Coğrafi Bilgi Sistemleri (CBS) alanında en popüler konumsal analizlerden biridir. Çeşitli amaçlar için yaygın olarak kullanılmaktadır. Bu tezde, uygulama senaryosu olarak "katı atık depo alanı yer seçimi" işlemi seçilmiştir. Senaryoya göre servis istemci bir yer seçimi uygulaması için kullanılacak servisleri, servislerin operasyonlarını ve bu operasyonları nasıl çalıştıracağına dair girdi-çıktı parametrelerini bilmemektedir. İstemciden istenen sadece gerçekleştirmek istediği uygulamayı ifade edecek bir "amaç" (goal) tümcesidir. Bu senaryoya göre; yer seçimi yapmak isteyen bir kullanıcı, aradığı kriterlere göre sorgusunu sınırlandırılmamış doğal bir dille, üst düzey bir şekilde yöneltir. Örneğin, "Gümüşhane ili sınırlarında, çöp dökümü için uygun; anayola 1 km mesafeden fazla, eğimi %20'den az, kuzey yönüne bakan, toprak kalitesi VI, VII, ya da VIII olan ve arazi kullanım sınıfı taşlık veya çalılık olan yerlerin

ortak alanını belirle ve sonuç haritayı göster" şeklindeki bir amaç cümlesi içinde istenen uygulama için gereken operasyonlar, bu operasyonlar için gerekli katmanlar ve bu katmanlardan gerekli verilerin hangi seçim koşuluna göre seçileceği bilgisi yer almaktadır. Böyle bir amaç cümlesinden anılan bilgileri çıkartabilmek için öncelikle bu cümle aradaki boşluklar ve noktalama işaretlerine göre parçalanır. Daha sonra kalan cümlede anlam ifade etmeyen bağlaçlar, birimler, rakamlar (ve, 1, km, %, veya gibi) kaldırılır. Kalan amaç cümlesinin son halinin Protege Ontoloji Editörü ile OWL tanımı elde edilir. Bu işlem uygulamada manuel olarak gerçekleştirilmiş olmakla birlikte, ileride buna yönelik bir yazılım bileşeni geliştirilebilir. Bunun için çeşitli yazılımlar (URL-52, 2015; URL-53, 2015) zaten mevcuttur yalnızca bunların entegrasyonu yapılmamıştır. Şekil 19'da geliştirici amacının çizge yapısının Protege OWLViz eklentisindeki temsili gösterilmektedir. Protege, oluşturulan ontolojileri OntoGraf (URL-54, 2015) ve OWLViz (URL-55, 2015) eklentileri ile görsel olarak çizge yapısında gösterebilmektedir. Şekilde görüldüğü üzere, çizgedeki her nokta bir sınıfı yani sorguda geçen sözcükleri, her çizgi ise aradaki ilişkiyi temsil etmektedir. Çalışmada sözcükler çizgeye oluşturulan "Coğrafi işlemler" sınıfının alt sınıfı olarak eklenmiştir.



Şekil 19. Geliştirici amacının çizge yapısı

Çalışma kapsamında kullanıcı amacına karşılık gelen kompoze ve atomik servislerin bulunması için Servis Tanımları Ontolojisi (STO) (service advertisement) oluşturulmuştur. Kullanıcı sorgusu bu ontoloji ile karşılaştırılarak ilgili servisler bulunmuştur. Bu ontolojideki bazı alt sınıfların hiyerarşisi ArcGIS (URL-56, 2015) ve GrassGIS (URL-57, 2015) coğrafi yazılımlarının konumsal operatörlerinin sınıflamasından esinlenerek oluşturulmuştur. Uygulama alanı ile ilgili web`de çeşitli servis kataloglarında yayınlanmış web servisleri oluşturulan bu ontolojiye göre OWL dilinde tanımlanarak eşleştirme aracında yayınlanır. Şekil 20'de ilgili web servislerinin yayınlandığı STO'nun Protege OWLViz eklentisindeki temsili gösterilmektedir.

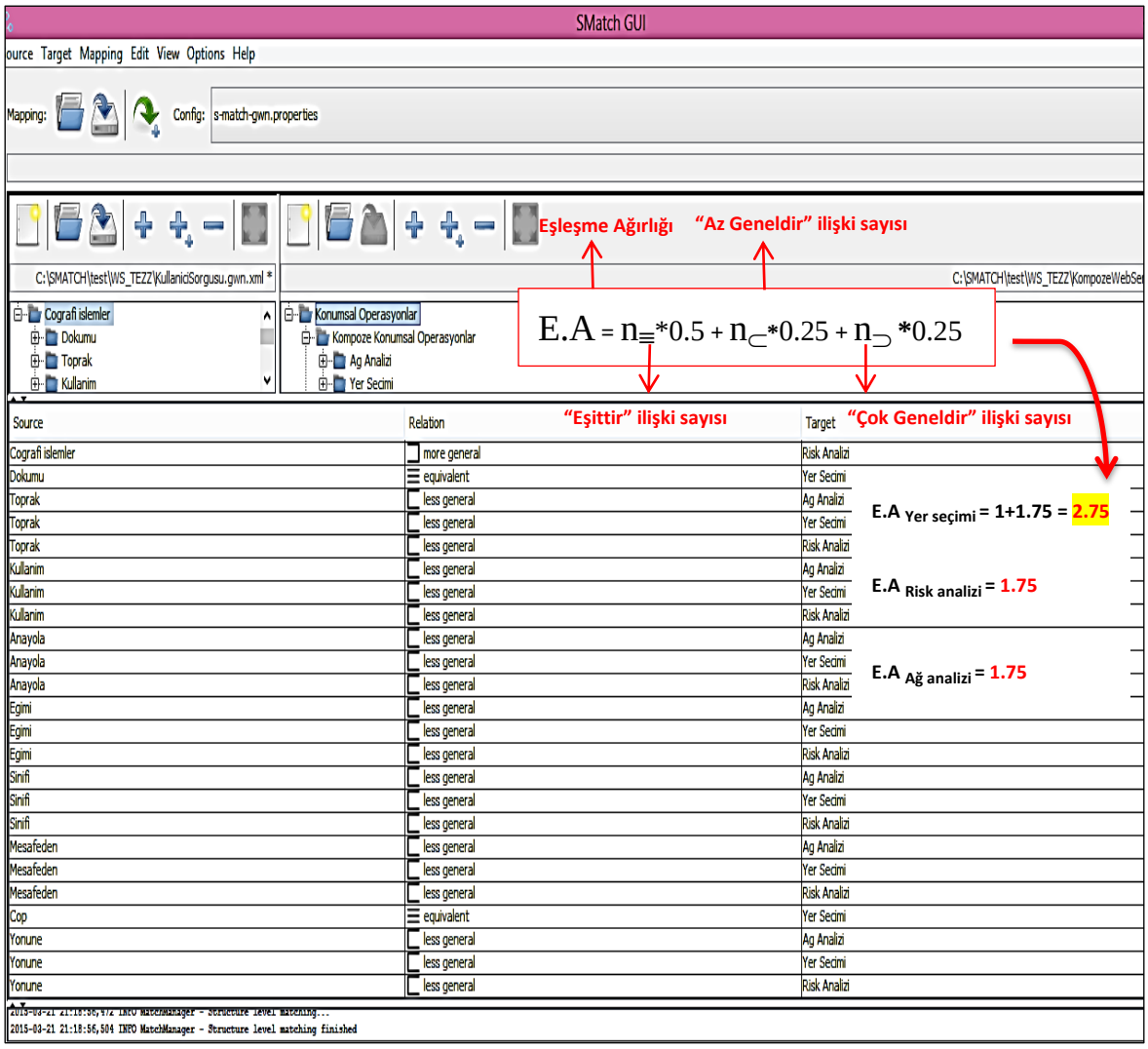


Şekil 20. Servis tanımları ontolojisi

İlk aşama kullanıcı amacından belirli çıkarsamalar yaparak istenen kompoze servis tipinin ve ilgili atomik web servislerinin belirlenmesidir. Bu amaçla kullanıcı sorgusu, servis tanımları ontolojisinin ayrı ayrı iki sınıfı ile ("Kompoze Konumsal Operasyonlar", "Atomik Konumsal Operasyonlar") eşleştirme yazılımında karşılaştırılır ve eşleşen kompoze ve atomik servisler bulunur.

Farklı konudaki (context) çeşitli konumsal uygulamalarda aynı servisler kullanılabilir. Örneğin, katman adı “nehir” ve seçim koşulu: “nehire uzaklık 10 km’den az olmasın” tarzındaki bir sorgu cümlesinde “nehir” katmanı sunan bir servis ve bu katmanın bir seçim koşuluna göre getirilmesi istenir. Ancak “nehir” katmanı risk analizi ya da yer seçimi kompoze servislerinin her ikisi için de kullanılabilir. Örneğin, her iki kompoze servis için de nehir katmanına bir “buffer” gerekebilir. Bu iki kompoze servis, girdi seçim koşulu ile ayırt edilebilir.

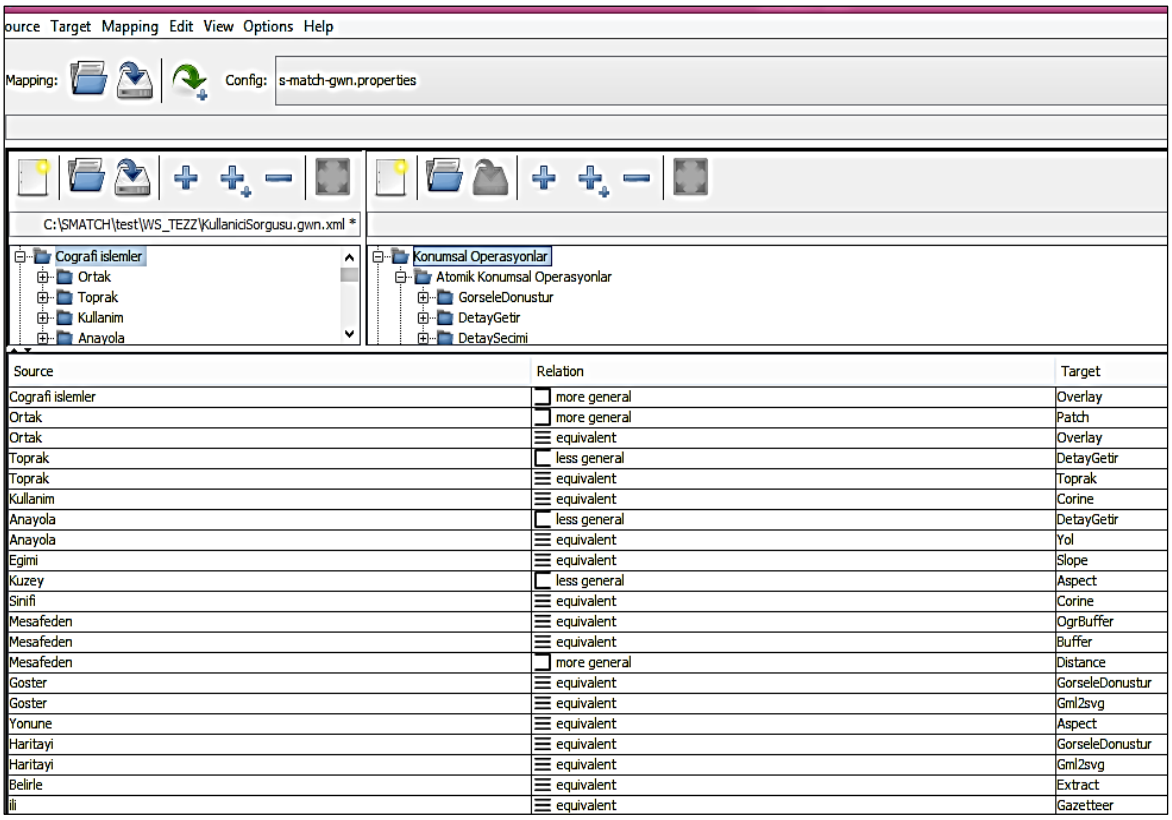
Bu problemin çözümü için, servis tanımları ontolojisinin ”Kompoze Konumsal Operayonlar” bölümü ve geliştirici sorgusu OWL tanımları, S-Match ile karşılaştırılır. S-Match, bu iki ontoloji arasındaki semantik benzerlik ilişkilerini arka plandaki bir alan ontolojisini veri sözlüğü (Background KnowledgeBase) olarak kullanarak belirler. Bir başka ifadeyle, S-Match arka planında, Konumsal Servis Ontolojisini (KSO) (bkz. 2.2.2.) kullanır. S-Match, geliştirici sorgusundaki sözcüklerin KSO yardımı ile “Servis Tanımları” ontolojisinde hangi kompoze işleme benzediğini belirler. Şekil 21’de S-Match’de bulunan benzerlik sonuçları gösterilmektedir. Sonuçlara göre; sorgu amaç cümlesindeki sözcüklerin çoğunun “Yer Seçimi”, “Ağ Analizi”, “Risk Analizi” kompoze operasyonlarının bir alt tipine karşılık geldiği bilgisi “az geneldir” (less general) ilişkisi ile anlaşılmaktadır. Kullanıcı sorgusunun en çok hangi kompoze servis tipine benzediği aradaki “az geneldir” ve “eşittir” (equivalent) ilişkilerinin sayılarak ağırlıklandırılması ile belirlenir. Üç kompoze servisin de eşleşme ağırlığı şekil 21’deki formüle göre hesaplanarak ağırlığı en yüksek bulunan servis seçilir. Sonuçlardan görüldüğü üzere, örneğin “Anayol” sözcüğü ile “Yer Seçimi”, “Ağ Analizi” ve “Risk Analizi” kompoze operasyonları arasında “az geneldir” ilişkisi bulunmuştur. Bu durumda “Anayol” sözcüğünün “Yol” katmanı ile ilişkili olduğu ve “Yol” katmanının bu üç kompoze servis için de kullanılabileceği bilgisi arka planda kullanılan KSO’dan öğrenilmiştir. Bulunan diğer “az geneldir” ilişkileri için de aynı mantık söz konusudur. Ancak, “çöp dökümü” sözcükleri ile diğer kompoze servisler arasında hiçbir ilişki bulunamazken, “yer secimi” kompoze servisi ile arasında “eşittir” ilişkisi bulunmuştur. Çünkü, KSO’da “çöp dökümü” kavramı “katı atık yer seçimi” tipindeki servisin eş anlamlı sınıfındadır. Bu şekilde sonuçlarda çıkan ilişki tiplerinin sayısı ve ağırlığına göre kullanıcı sorgusunda istenen kompoze servis tipinin “Yer Seçimi” olduğu anlaşılır.



Şekil 21. Geliştirici sorgusuna karşılık gelen kompoze servisin S-Match ile bulunması

İkinci aşama, kullanıcı amacından belirli çıkarsamalar yaparak istenen katmanları sunan servislerin ve bu katmanlara uygulanacak konumsal işlemlerin belirlenmesidir. Bunun için OWL dilinde kodlanmış geliştirici sorgusu ve servis tanımları ontolojisinin "Atomik Konumsal Operasyonlar" bölümü S-Match şema ve ontoloji eşleştirme aracına yüklenir. S-Match, KSO'dan istemci şemasında bulunan her kavramın "servis tanımları" ontolojisindeki her servis ile anlamsal olarak bir ilişkisi olup olmadığını öğrenir. Şekil 22'de S-Match yazılımında geliştirici sorgusu (solda) ve servis tanımları (sağda) şemaları ile alt kısımda bu iki şema arasında eşleştirici tarafından bulunan benzerlik ilişkileri gösterilmektedir. Karşılıklı olarak semantik ilişkili bulunan şema kavramları aralarında bulunan benzerlik değerine göre gösterilmektedir. Şekilden görüldüğü üzere "Kuzey" sözcüğünün "yön" sözcüğünün tipinde olduğu bilgisi arka planda kullanılan KSO

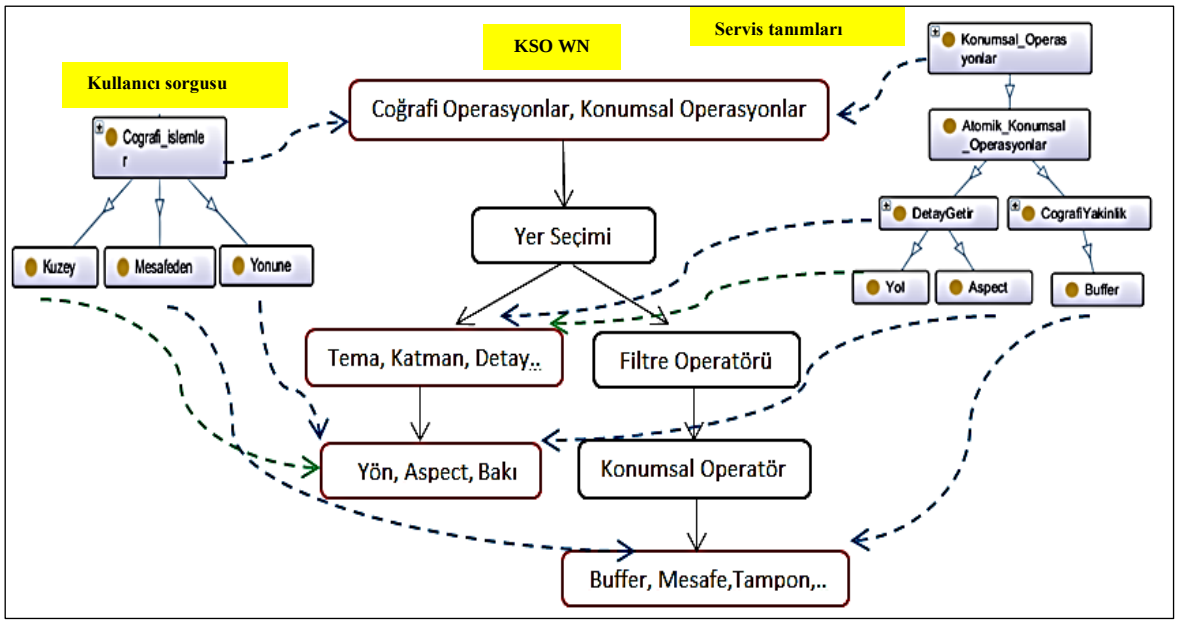
modelinden çıkarsanarak “az geneldir” (⊂) ilişkisi ile ve “yön” sözcüğü ile “Aspect” sözcüğünün eş anlamlı olduğu bilgisi arka planda kullanılan KSO modelinden çıkarsanarak “eşittir” (≡) ilişkisi ile kullanıcıya döndürülmüştür (şekil 22). “Mesafe” sözcüğü KSO’da “Buffer” operasyonu ile aynı eş anlamlı sınıfta olduğu için “mesafeden” sözcüğü ile farklı ama aynı işlevi gören iki servis olan “Buffer” ve “OgrBuffer” servisleri “eşittir” ilişkisi ile döndürülmüştür. “ili” sözcüğü KSO’da yer ismi alıp koordinat döndüren bir operasyon olan “Gazetteer” ile ilişkili olduğu için arada “eşittir” ilişkisi döndürülmüştür. “Eğim” ve “Slope” sözcükleri KSO’da aynı eş anlamlı sınıfın bir üyesi olduğu için “Eğimi” ve “Slope” kavramları arasında “eşittir” ilişkisi bulunmuştur. “Toprak” sözcüğünün KSO’da “Tema” sınıfının alt tipi olarak tanımlanmış olması ve “DetayGetir” operasyonunun da “Tema” sınıfı ile ilişkili olması nedeniyle “Toprak” sözcüğünün de bir detay tipi olduğu anlaşılmış ve “Detay Getir” servisi ile arasında “az geneldir” ilişkisi bulunmuştur. Diğer ilişkiler de benzer şekilde bulunmuştur. Bundan sonra istemci, bulunan servisler arasından en yüksek benzerlik değeri olanları seçer ve istediği ölçütleri sağlayan bir yer seçimi kompozisyonu için gerekli atomik servisleri bulmuş olur. En yüksek benzerlik değerinden kasıt, varsa ilk olarak “eşittir” ilişkisi ile bulunanların seçilmesi daha sonra sırayla “az geneldir” ve “çok geneldir” ilişkisi ile bulunanların seçilmesidir. Bu şekilde S-Match, sorguda bulunan her kelimeden bir anlam çıkarmaya çalışır ve karşılık gelen kavramları belirler. Anlam çıkaramadığı kelimeler de işe yaramayan, anlamsız (den, ve, için, olan vb.) olan kelimelerdir.



Source	Relation	Target
Cografi islemler	more general	Overlay
Ortak	more general	Patch
Ortak	equivalent	Overlay
Toprak	less general	DetayGetir
Toprak	equivalent	Toprak
Kullanim	equivalent	Corine
Anayola	less general	DetayGetir
Anayola	equivalent	Yol
Egimi	equivalent	Slope
Kuzey	less general	Aspect
Sinifi	equivalent	Corine
Mesafeden	equivalent	OgrBuffer
Mesafeden	equivalent	Buffer
Mesafeden	more general	Distance
Goster	equivalent	GorseleDonustur
Goster	equivalent	Gml2svg
Yonune	equivalent	Aspect
Haritayi	equivalent	GorseleDonustur
Haritayi	equivalent	Gml2svg
Belirle	equivalent	Extract
ili	equivalent	Gazetteer

Şekil 22. Geliştirici sorgusuna karşılık gelen atomik servislerin S-Match ile bulunması

Şekil 23'de kullanıcı sorgusu (solda) ile servis tanımları ontolojisindeki (sağda) bazı sınıfların arka planda kullanılan (ortada) KSO WN'de ilişkili olduğu eş anlamlı kavram grupları gösterilmiştir. Yeşil renkli oklar "hyponym" (tipinde) ilişkisini, mavi renkli oklar "synonym" (eş anlamlı) ilişkisini temsil etmektedir. Benzer şekilde KSO'daki her tema ilgili olduğu operasyonlarla ilişkilendirilmiştir.

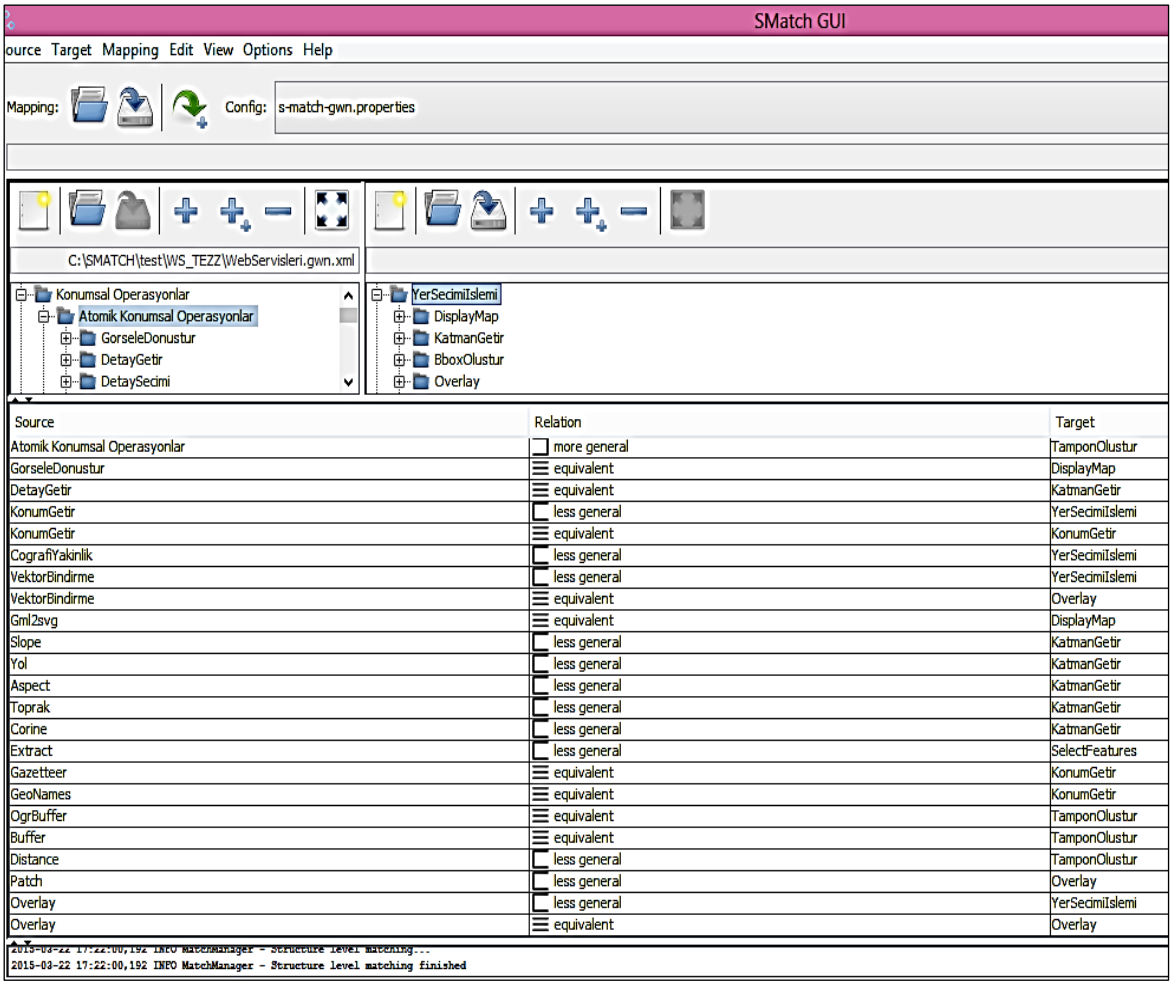


Şekil 23. KSO WN'de kavramların ilişkilendirilmesi

Herhangi bir kullanıcı aynı sorguyu başka kelimeler kullanarak yönelttiğinde amaç cümlesinin içeriği değişecektir. Böyle bir durumda birinci alternatif, kullanıcının sorgusuna başlamadan önce kendisine sunulan bir sözcük grubu ile kısıtlanmasıdır. Örneğin kullanıcının, şekil 23 itibariyle, KSO WN'de tanımsız gözüken "uzaklık" sözcüğü yerine, "eş anlamlı" ilişkisi ile tanımlanmış bulunan "mesafe" sözcüğünü kullanmaya kısıtlanması gibi. Bununla birlikte, eş anlamlı sözcük grubunun (synset) "uzaklık" sözcüğünü de kapsayacak şekilde genişletilmesi, bir sorun değildir. Eklenecek sözcük için "eş anlamlı" ilişkisi kurulacak sınıf mevcut değilse, WN ilişkilerinden uygun olan biri ("üst sınıf", "alt sınıf", "tipinde", "parçası" vb.,) üzerinden ilişki kurulur. Eklenen bu sözcükler de KSO WN'de bulunan servis tipleri ile ilişkilendirilir. Bu şekilde KSO WN'de yapılan eklemelerle bu sorun çözülür. Burada önemli olan, genişletmenin nereye kadar götürülmesi gerektiğidir. Bu problemin çözümü ileriki çalışmalara bırakılmıştır.

Üçüncü aşama, ikinci aşamada bulunan atomik web servislerinin birinci aşamada bulunan kompoze servis tipinin "iş akış şablonu" (OWL-S "işlem modeli") itibariyle hangi tipte servislere karşılık geldiğinin bulunmasıdır. Bunun için STO (servis tanımları ontolojisinin) "Atomik Konumsal Operasyonlar" bölümü ve KSO'daki "yer seçimi işlemi" sınıfları, S-Match'te eşleştirilir. Şekil 24'de eşleştirme sonucu gösterilmektedir. Buna göre; "Gazetteer" sözcüğünün KSO "Yer Seçimi" işlem modelindeki "KonumGetir" işlemine eş değer olduğu "eşittir" ilişkisi ile anlaşılmaktadır. "Toprak" ve "Yol" sözcüklerinin yine

aynı işlem modelinde “KatmanGetir” işlemine ait olduğu, bulunan “az geneldir” ilişkisi ile anlaşılmaktadır. Benzer şekilde, “Buffer” servisinin de yine aynı işlem modelinde “TamponOluştur” servisine eşdeğer olduğu görülmektedir. Keza, STO’daki “Extract” servisinin aynı işlem modelinde “SelectFeatures” servisinin bir alt tipi olduğu “az geneldir” ilişkisi ile anlaşılmaktadır. Diğer servisler ve kompoze işlem modelinde ilişkili olduğu operasyonlar da benzer şekilde bulunmuştur.



Şekil 24. Web servislerine karşılık gelen kompoze işlem modelindeki servislerin bulunması

Yukarıda bahsedilen bütün aşamalardan sonra kullanıcı, amacına uyan kompoze web servisi tipini, bu kompozisyon için gerekli web’de yayınlanmış atomik web servislerini ve bu web servislerinin kompoze web servisi işlem modelindeki hangi servislere karşılık geldiğini bulmuş olur.

2.2.2. Konumsal Servis Ontolojisinin (KSO) Geliştirilmesi

CBS işlemlerinin sınıflandırılması konusunda literatürde çok sayıda çalışma mevcuttur (Berry, 1987; Burrough ve McDonnell, 1998; Albrecht, 1995). Bunların her biri diğerinden farklılık göstermektedir. Örneğin "eğim ve bakı haritası oluşturma", Burrough ve McDonnell (1998)'de "konumsal işlemler / konumsal filtreleme/ yüzey türevlerinin hesaplanması" tipinde, Berry (1987) de ise, "civar (neighborhood)" bazlı işlemler / "civarın karakterize edilmesi ("characterizing neighborhood") tipinde işlemler olarak sınıflandırılmıştır. Diğer yandan, ISO TC211/19119 (ISO, 2001) konumsal servis sınıflandırması da ayrıca farklılık göstermektedir.

Peki iyi bir sınıflandırma nasıl olmalıdır? İlk olarak sınıflar "karşılıklı birbirini dışlayan" (mutually exclusive) olmalıdır. Yani, bir sınıfa dahil edilebilen bir işlem "diğer bir sınıfa da ait olabilir" olmamalıdır. Ancak, burada daha önemli olan sınıflandırmanın hangi amaca hizmet edeceğidir[8]. Bu tez açısından önemli olan da budur. Bu noktadan bakıldığında bu tezdeki ihtiyacı karşılayacak bir sınıflandırma literatürde yoktur. Çünkü konu henüz çözümlenebilmiş değildir ve bu da böyle bir sınıflandırma ne temelinde olmalı ya da hangi sorunu çözmeye yönelik olmalı boyutuna bir düzey getirilememiş olması nedeniyledir. Bununla birlikte, Albrecht (1995)'in, "amaç", "iş (task)", "işlev (function)" ve "veri (data)" bazındaki bakış açısı, en azından anılan isimlendirme bakımından benzerdir. Tümüyle böyle bir sınıflandırma da bu tezin konusu değildir. Bu tezde yalnızca temel prensipleri sunulacak bir sınıflandırma ile, tezin ana katkı noktasını ispata yetecek kadar bir "mini" sınıflandırmanın sunulması uygun olacaktır. Başka çalışmalarla, ilgili prensipler bazında bu sınıflandırma genişletilebilir.

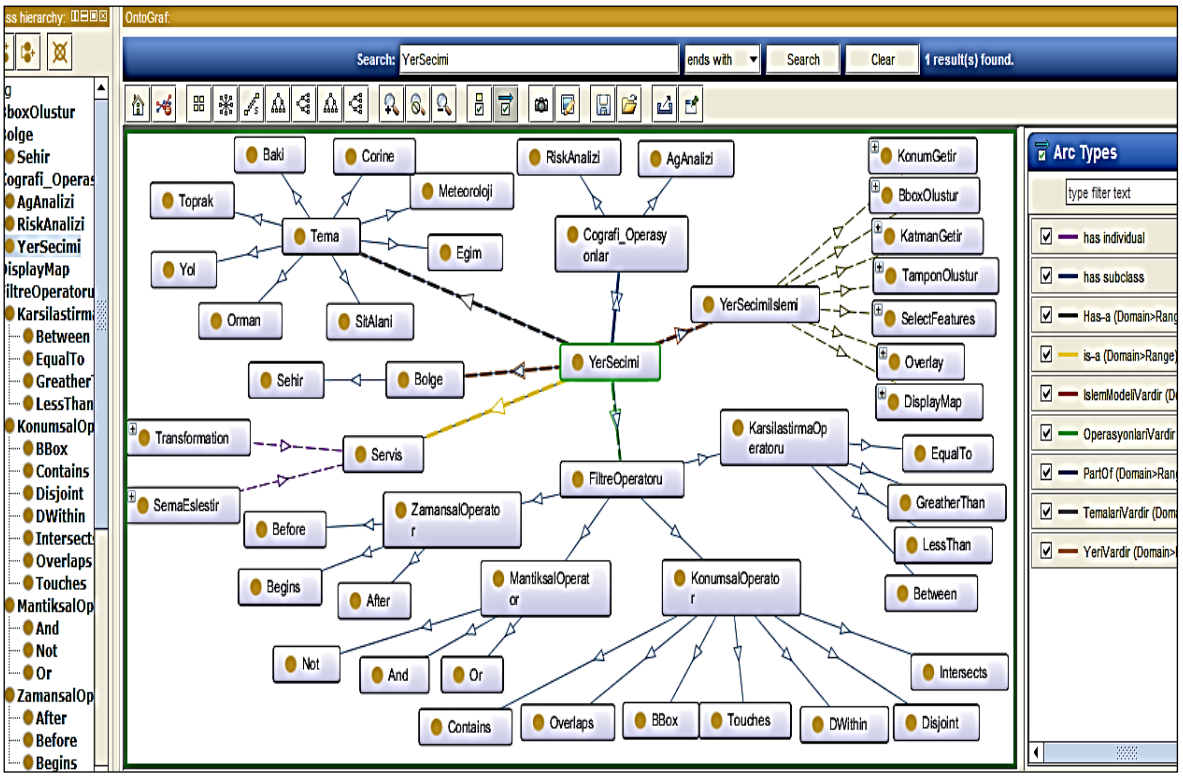
KSO nasıl tanımlanabilir? Ontoloji tasarımında "pattern" (Gangemi, 2005; Hu vd., 2013; Maynard vd., 2009; Martínez vd., 2012) yaklaşımından yararlanılabilir. Pattern yaklaşımının hedefi, veri ontolojisi bağlamında veri tanımlamalarının (şema) eşleştirilmesini kolaylaştırmaktır ki, bu işin tam otomatik olarak yapılması da SWT alanında henüz çözümlenebilmiş bir problem değildir. Ontoloji tasarım pattern'leri farklı alanlarda ortaya çıkan, farklı işlevleri çözen ortak kavramsal pattern'lerden üretilmiştir. Bu pattern'ler bilgi mühendisleri ve alan uzmanları arasındaki ortak kavramsallaştırmaları

[8] Örneğin Berry (1987)'nin, "reclass" işlemlerine bir başka açıdan bakıldığında bunlara "dönüştürme" (transformasyon) işlemleri denilebilir.

belirleyerek, daha karmaşık ontolojilerin tasarımı için yapı taşları oluşturur. Bu esnek yapı taşları, alan uygulamalarında bilgiyi modellemek için kullanılır (Hu vd., 2013).

KSO tasarımı açısından en önemli konulardan biri, tasarlanan "iş akışlarının" her durum için kullanılabilir olmasını temin etmektir**.** Örneğin bir "yer seçimi" işi akışında belirli katmanlara yeniden sınıflandırma ("reclass") işleminin uygulanma ya da uygulanmama durumu, KSO'da ifade edilebilecek midir? Yani "verinin belirli özellikleri varsa ve buradan şöyle bir işlemle devam edilecek ise "reclass" uygulanmalı" gibi bir bilgi KSO da ifade edilebilir mi? Withers vd., (2010) ya da Kietz vd., (2014) çalışmalarında eldeki veriye ne tip işlemler uygulanabilir? diye değerlendirmeleri anılan problemin çözümü içindir. Ancak bu durumda da iş akışı tasarımcısının çok teknik birisi olması gereklidir. Mesela, ArcGIS model builder'da "add data" veya "add tool" işlemlerinde yazılım kullanıcıya herhangi bir yardım yapmamaktadır. Bu durumda kullanıcı "ne" ekleyeceğini bilemeyebilir, oysa söz edilen yayınlarda sistem semantik çıkarsama ile belli tipteki veriye hangi işlemler uygulanabileceğini söylemektedir. Fakat böyle bir durumda bile yine kullanıcının her bir işlemin ne olduğunu bilmesi gerektiği için, bu tez çalışmasında önerilen "iş akışının kullanıcı amaç tümcesinden çıkarsanması" yaklaşımı değerlidir.

Çalışma kapsamında KSO'nun "katık atık depolama alanı yeri seçimi" örnek senaryosu ile ilgili olan "yer seçimi" iş akışı kısmı tanımlanmıştır. Bunun için OWL-S ontolojisi kullanılmıştır. KSO, "Yer seçimi" tipindeki bir servisin kullandığı veri temaları ve operasyonları tanımlamaktadır. Bu temalara uygulanacak operasyonlar da ontolojide ayrı sınıflarda tanımlanmıştır.



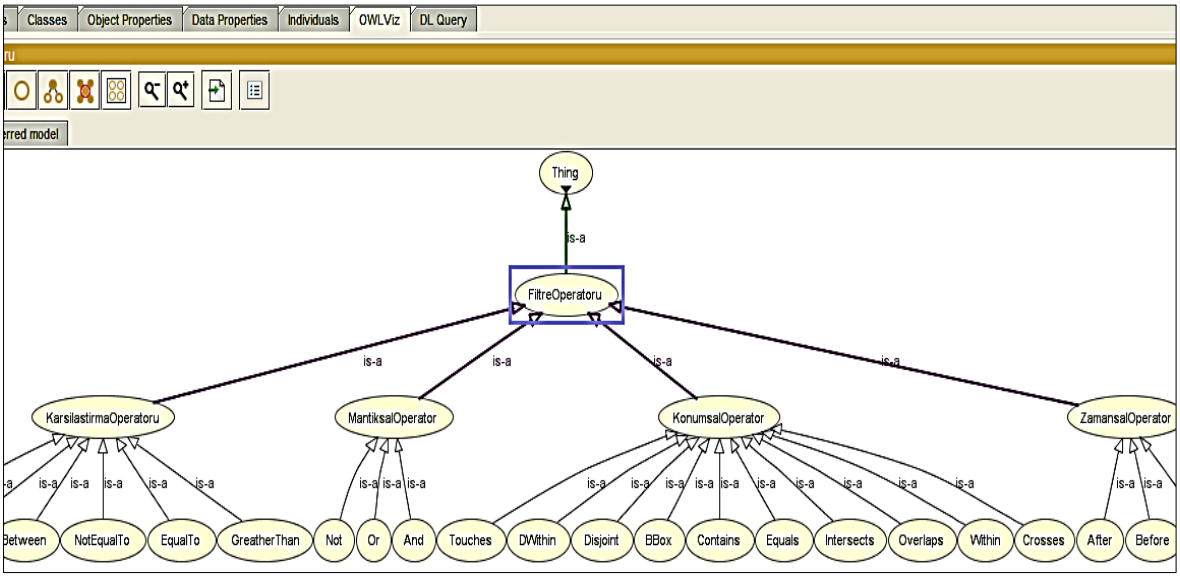
Şekil 25. Önerilen KSO`nun bir kesiti. (Farklı renkler farklı ilişki tiplerini temsil etmektedir) (Arc Types)

KSO, protege ontoloji editörü'nde OWL'de kodlanarak oluşturulmuştur. Şekil 25'de KSO`nun temel yapısı Protege OntoGraf eklentisinde gösterilmektedir. Şekilde görüldüğü üzere oluşturulan KSO'da "yer seçimi", "risk analizi", "ağ analizi" olmak üzere üç çeşit coğrafi operasyon sınıfı tanımlanmıştır. Bunlardan "Yer seçimi" sınıfı ve bileşenleri şekilde gösterilmektedir. Şekle göre; KSO'da merkezi "yer seçimi" sınıfı ile ilişkili beş temel bileşen vardır.

Bir "yer seçimi" servisinin yapılacağı coğrafi bölge vardır. "Bölge" sınıfı birinci temel bileşendir ve çalışma kapsamında oluşturulan "coğrafi yer" veri ontolojisini import eder. "Coğrafi yer" ontolojisinde Türkiye içinde yer alan bölgeler ve içerdiği şehirlerin bilgisi tanımlanmıştır.

Bir "yer seçimi" servisinin kullandığı temalar vardır. "Tema" sınıfı ikinci temel bileşendir ve çalışma kapsamında oluşturulan "tema" veri ontolojisini import eder. Tema veri ontolojisindeki sınıflar yer seçimi uygulamalarında çoğunlukla kullanılan temalar baz alınarak oluşturulmuştur. Bu ontoloji çalışma kapsamında kullanılan veri alanını (domain) oluşturur.

Bir “yer seçimi” servisinin kullandığı operasyonlar vardır. “Filtre Operatörü” sınıfı üçüncü temel bileşendir. “Filtre Operatörü” sınıfı OGC’nin standard olarak tanımladığı filtre şemasından (OGC, 2014) elde edilen “Filtre” ontolojisinin import edilmesiyle oluşturulmuştur. Çalışmada “Filtre” ontolojisi filter (xsd) XML şemasının Top Braid Composer (URL-58, 2015) aracı ile OWL’ye dönüştürülmesi ile oluşturulmuştur. Bu şekilde OGC filter operatörlerinin kullanılmasıyla KSO’daki operasyonların sınıflaması standarda dayandırılmıştır. OGC Filtre operatörü daha detaylı bir şekilde Protege OWLViz eklentisinde şekil 26’da gösterilmektedir. Aradaki “is-a” ilişkisi okun başladığı sınıfın, okun bittiği sınıfın tipinde ya da o sınıfa ait bir örnek olduğunu tanımlar.

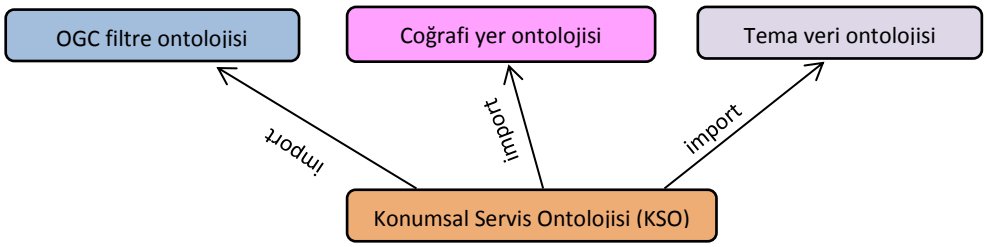


Şekil 26. OGC Filtre Operatörlerinin bir kısmı (OGC, 2014)

Bir yer seçimi kompoze servisinin “Şema eşleştir” ve “Transformation” servisleri vardır. Servis parametrelerinin eşleştirilmesi “Şema eşleştir” servisi ile otomatik olarak gerçekleştirildikten sonra, format farklılığı olduğu durumlarda “Transformation” servisi çalışacak ve veri formatlarını eşitleyecektir. Bu iki servis KSO’da tanımlanmıştır ancak gerçekleştirimi WSK içinde yapılmamıştır.

Bir yer seçimi servisinin kompoze “işlem modeli” vardır ve KSO’da ayrı bir sınıfta tanımlanmıştır. Kompoze servisi oluşturan bileşen servisler (atomik) de alt sınıflarda tanımlanmıştır. “Yer seçimi işlemi” KSO’nun son temel bileşenidir ve yer seçimi kompozisyonunun içerdiği operasyonların iş akışı bilgisine sahiptir. Bu iş akışı herhangi

bir yer seçimi uygulamasında kullanılabilecek operasyonları, operasyonların sırasını ve çalıştırılabilmesi için gerekli olan koşulları ve parametreleri tanımlar.



Şekil 27. KSO'nun import ettiği ontolojiler

Geliştirici amaç cümlesinin farklı tema isimleri veya yer isimleri kullanılarak yöneltilmesi durumunda, bu sözcüklerin ilgili veri ontolojilerine eklenmesiyle bu model genişletilebilir. Bu ontolojiler şekil 27'de görülen KSO'nun import ederek kullandığı ontolojilerdir. Bu şekilde KSO'nun her uygulama alanı için ayrı ayrı oluşturulması engellenerek, farklı uygulama alanlarında da kullanımı mümkün kılınarak genelleştirilmesi sağlanır. Örneğin, KSO'da bulunmayan farklı bir tema ve farklı bir şehir için yer seçimi yapmak isteyen bir kullanıcı olabilir. Bu durumda yeni temaların oluşturulan "tema" veri ontolojisine eklenmesi ve il isminin oluşturulan "coğrafi yer" ontolojisine eklenmesiyle KSO'nun yapısı (içerdiği bileşenler ve aralarındaki ilişkiler) bozulmadan çözüm sağlanır. Çünkü, KSO sadece bu veri ontolojilerini import edecektir. KSO içeriği süper sınıflar bazında zaten birbiri ile ilişkilendirilmiş olduğu için, import edilen veri ontolojilerinden gelen yeni veriler de otomatik olarak ilgili ontoloji sınıfları ile ilişkilendirilmiş olacaktır. Böylece bu sistem farklı bir alanda gerçekleştirilecek yer seçimi işlemine de uyarlanabilir.

Çalışmada çeşitli semantik eşleştirme işleri için S-Match (Giunchiglia vd., 2004) eşleştirici yazılımı kullanılmıştır. S-Match farklı iki şemaya ait çizge noktalarını karşılaştırırken arka planda varsayılan bilgi tabanı olarak WordNet (WN) sözlüksel veritabanını kullanır. WN'nin içeriğinin genel olması, konumsal uygulama alanlarını içermemesi nedeniyle, uygulama kapsamında geliştirilen KSO arka planda bilgi tabanı olarak WN yerine kullanılmıştır. Bu nedenle oluşturulan KSO, S-Match'de kullanılmak üzere WN sözlüksel yapısına extJWNL (Extended Java WordNet Library) yazılımı kullanılarak dönüştürülmüştür.

2.2.3. Önerilen Mimaride WSK

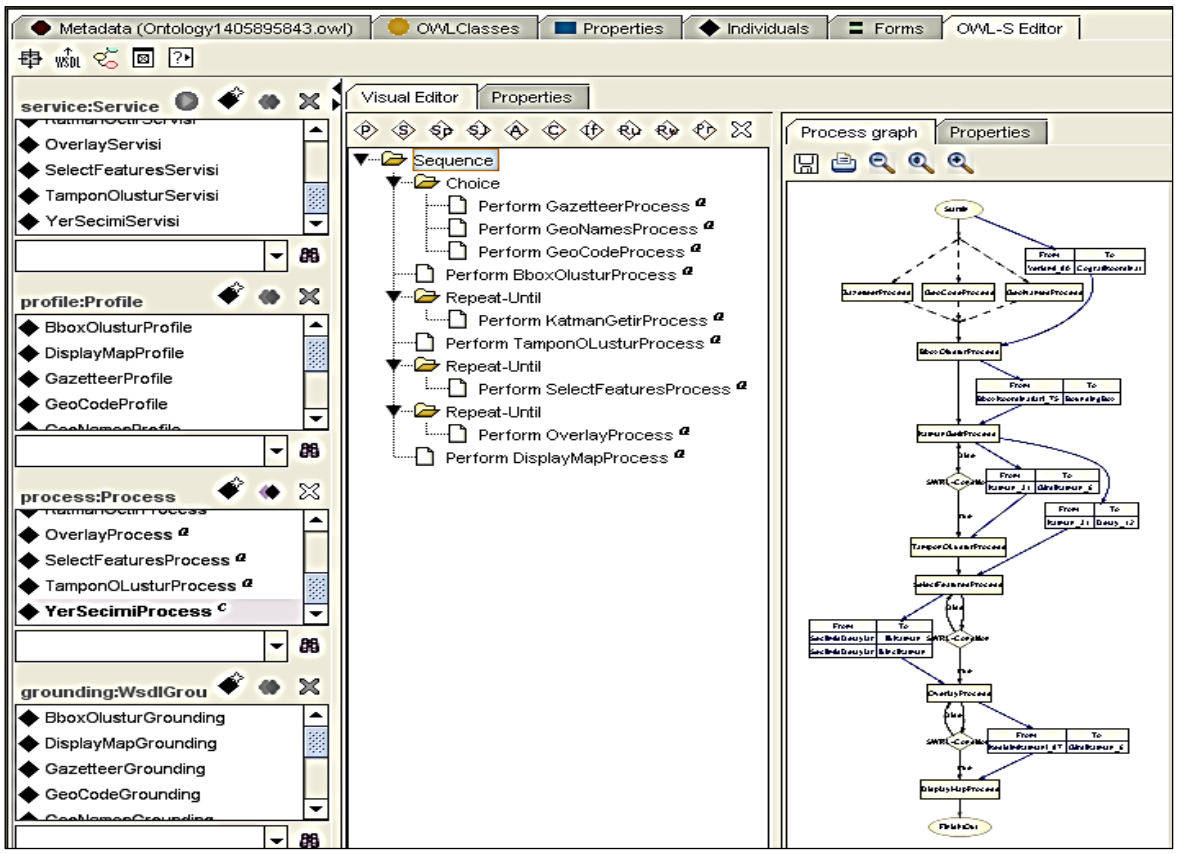
Bu bölümde, örnek bir yer seçimi senaryosunun SWT ile WSK kullanılarak gerçekleştirilmesine çalışılacaktır. Söz konusu uygulama gerçekleştirimi için son kullanıcı olan istemci hangi coğrafi operasyon için hangi web servisini kullanacağını, bu servise nasıl erişeceğini ve eriştikten sonra nasıl çalıştıracağını bilmemektedir. Bu nedenle, bu çalışmada kullanıcının uygulama işleyişi hakkında hiçbir şey bilmediği varsayıldığı için, KSO iş akışı ontolojisinde tanımlı, "soyut" bir web servisi kompozisyonu kullanıcıya buldurulacaktır. Servis kompozisyonu içindeki servislerin girdi, çıktı, ön koşul ve son koşul parametreleri kompozisyon içine gömülerek, bu detaylar kullanıcıdan istenmeden daha üst düzey bir kompozisyon akışı oluşturularak kullanıcının yükü hafifletilecektir.

Bir web servisini doğru bir şekilde koşturabilmek için servisin çalışmasını sağlayan gerekli girdi parametrelerinin verilmesi gerekmektedir. Web'de yayınlanmış çeşitli coğrafi operasyonlar yapan ve farklı veriler sunan birçok web servisi arasından gerekli web servislerini bulmak oldukça zordur. Çünkü aynı işlemi yapan web servisleri farklı girdi ve çıktı parametre isimlerine sahip olabileceği gibi, farklı işlemleri yapan web servisleri de aynı girdi ya da çıktı isimlerine sahip olabilir. Bu zorlukların web servislerinin SWT ile tanımlanması, yayınlanması ve buldurulması ile aşılması amaçlanmıştır. Bu bölümde yukarıda anılan bölümde (bkz. 2.2.1.) kullanıcı amaç cümlesinden çıkarsanarak semantik eşleştirme yazılımı ile KSO desteğinde bulunan uygun web servislerinin soyut ve somut WSK modelinin oluşturulması ve somut WSK'nın çalıştırılarak istenen yer seçiminin gerçekleştirilmesine çalışılacaktır.

2.2.3.1. Soyut WSK Oluşturulması

Bu bölümde, bölüm 2.2.1'de kullanıcı amaç cümlesinden çıkarsanarak KSO'da bulunan OWL-S kompoze servis akışının yapısı hakkında detaylı bilgi verilecektir. Bu servis kompozisyonu katı atık yer seçimini gerçekleştiren atomik web servislerinin belirli bir iş akışı sırasında arka arkaya dizilmesiyle oluşturulan servisler zinciridir. Geliştirici, bulunan servisleri hangi sırada hangi parametre ve koşullara göre çalıştıracağını KSO`daki bu servis iş akışı bileşeninden yani soyut WSK modelinden öğrenir. Daha sonra uygulama geliştiricisi bu modele bakarak ilgili web servislerini bulup somut WSK'yı oluşturur. Çalışma kapsamında oluşturulan soyut WSK modeli OWL-S servis tanımlama dilinde

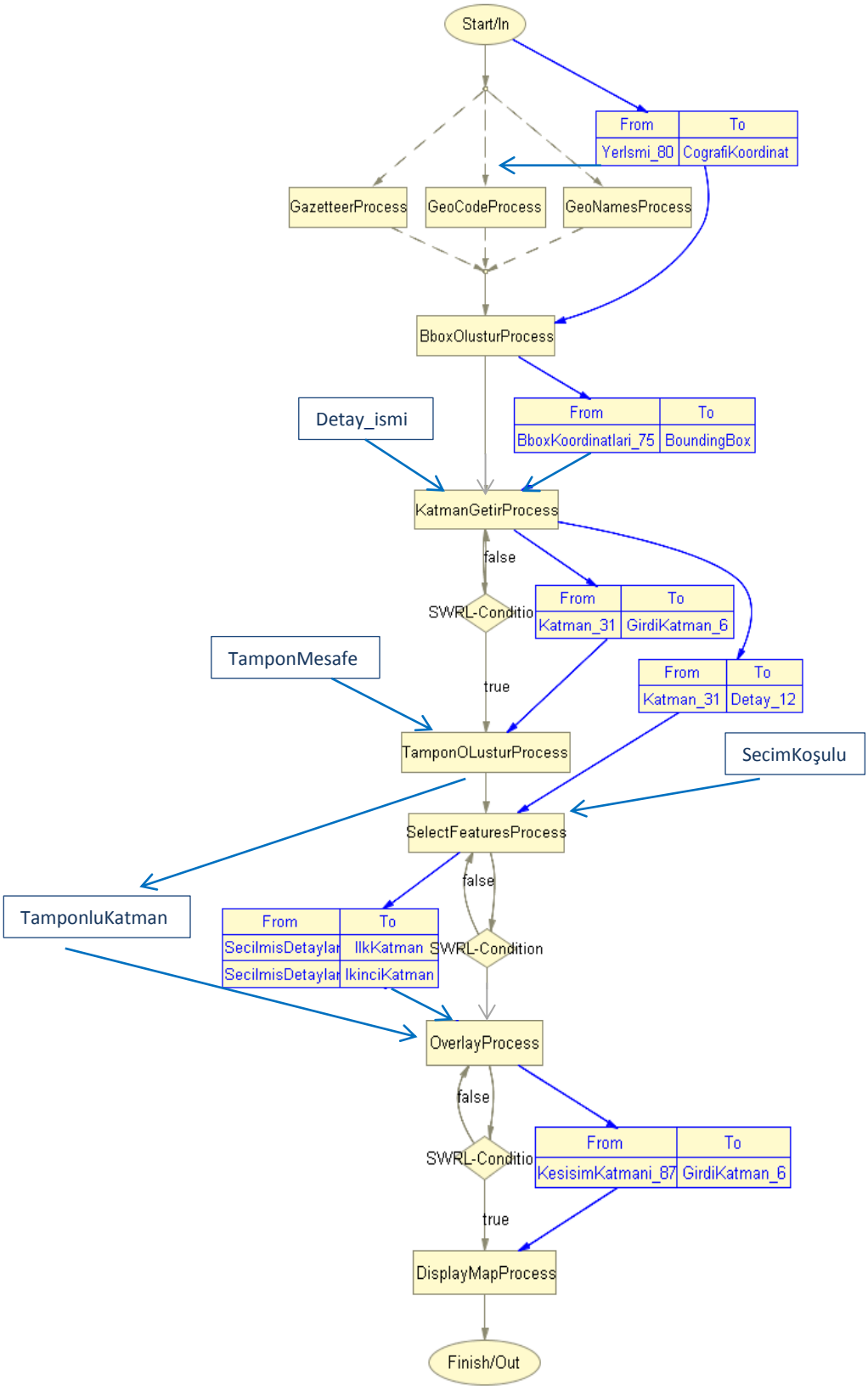
kodlanarak Protege OWL-S editörü eklentisinde tanımlanmıştır (Şekil 28). Bu kompozisyondaki servislerin her birinin profil (profile), process (işlem modeli) ve referanslandırma (grounding) olmak üzere servis tanımları OWL-S'de oluşturulmuştur. Kompozisyon iş akışı ise OWL-S kontrol parametreleriyle tanımlanmıştır.



Şekil 28. Yer seçimi soyut servis kompozisyonunun oluşturulması

Şekil 28'de görüldüğü üzere, oluşturulan soyut WSK içindeki bütün atomik servisler "sequence" (sırasıyla) yapısındadır. Yani kompozisyon çalıştırıldıktan sonra bütün servisler ard arda, sırasıyla çalışır. "Sequence" parametresinin "Choice" (seçim) ve "Repeat Until" (koşullu döngü) alt parametreleri vardır. İlk sıradaki "Choice" kontrol parametresi Gazetteer, GeoNames ve GeoCode servislerinin bir tanesinin kullanıcı isteğine göre seçilip çalıştırılacağını tanımlar. Girdi parametresi "yer ismi", çıktı parametresi ise "koordinat" bilgisidir. Sıradaki BboxOluştur servisi kendinden önceki servisten gelen çıktı parametresi "koordinat" verisini girdi şeklinde alarak çalışır. Sıradaki "RepeatUntil" kontrol parametresi KatmanGetir servisinin belli bir koşul sağlanıncaya kadar çalıştırılacağını tanımlar. Burada koşul, kullanıcı tarafından girilen katman sayısıdır.

Katman sayısı sıfir oluncaya kadar aynı işlem tekrarlanır. KatmanGetir servisi kendinden önceki BboxOluştur servisi ile üretilen “BBox Koordinatları” bilgisini ve kullanıcının gireceği detay ismini girdi olarak alır ve çıktı olarak istenen detayların girilen sınırlayıcı alan içinde kalan kısımlarını döndürür. Sıradaki TamponOluştur servisi KatmanGetir servisinden gelen yol katmanına kullanıcının gireceği mesafe kadar tampon uygular. Sıradaki “RepeatUntil” kontrol parametresi SelectFeatures servisinin girilen katman sayısı kadar tekrarlayarak çalıştırılacağını tanımlar. SelectFeatures servisi KatmanGetir servisinden gelen katmanları ve kullanıcının gireceği seçim koşulunu alarak, her katman için istenen koşullara göre seçilmiş detayları döndürür. Sıradaki “RepeatUntil” kontrol parametresi içindeki “Overlay” servisi, bir önceki servisten gelen seçilmiş katmanları alır ve kesişim katmanını oluşturur. Overlay servisi kendisine gelen girdi katmanı sayısı kadar tekrarlayarak çalışır. Son sıradaki “DisplayMap” servisi ise, bir önceki servisten gelen bindirilmiş sonuç katmanın görsel olarak görüntülenmesini sağlar. Şekil 29’da yer seçimi servis kompozisyonunun OWL-S`de iş akışı ve veri akışı gösterilmektedir. İş akışı içindeki koşul döngüleri SWRL kural tanımlama dili ile Protege OWL-S editörü eklentisinde oluşturulmuştur. Oluşturulan soyut WSK OWL-S tanımının bir kısmı Ek 1.’de yer almaktadır.



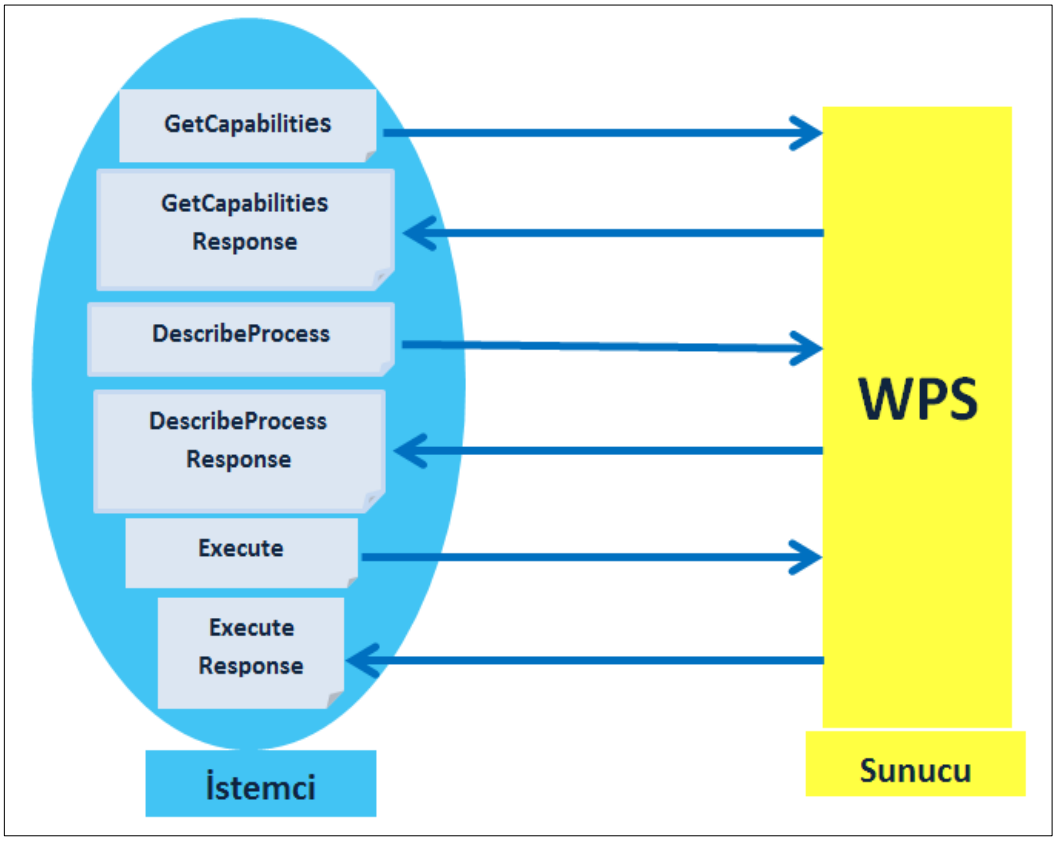
Şekil 29. Yer seçimi servis kompozisyonunun OWL-S`de iş akışı

2.2.3.2. Somut WSK Oluşturulması ve Çalıştırılması

Bundan sonra yapılacak olan, web de yayınlanmış olan STO servislerinin bulunması ve iş modeli iş akışı sırasında koşturulmasıdır. Bunun için Taverna iş akış motoru (bkz. 1.6.1.3.2.4.) kullanılmıştır. Bu tez çalışmasının ilgi alanı gereği, servislerin Web de tek tek aranması yerine, yayınlanmış servislerin Taverna'ya "import" edilmesi yoluna gidilmiştir. Kullanıcı yukarıda anlatılan şekilde bulunan atomik servis isimlerine göre ilgili servisleri buradan seçerek koşturulmaya hazır WSK'ya dahil eder. Daha sonra Taverna iş akışı aracında seçim koşulunu ilgili servise kendi girer. Bulduğu servisleri hangi sırada ve parametrelerle çalıştıracağını Yer seçimi servisinin OWL-S'de tanımlanmış (bkz. 2.2.3.1.) işlem modelinden öğrenir.

Somut WSK gerçekleştirimi için somut, çalıştırılabilir web servisleri gereklidir. Bu çalışma kapsamında konumsal işlemleri elde etmek için OGC (The Open Geospatial Consortium) servislerinden WPS (Web Processing Service) servisi kullanılmıştır. OGC Web servisleri, konumsal veri ve işlemleri sunmak için geliştirilmiş "insan-odaklı" servislerdir. Bir OGC Web servisi tarafından sunulan konumsal veri ve işlemleri elde etmek isteyen bir kullanıcının, ilgili servis tarafından gerçekleştirilen bir dizi operasyonu kullanması gerekmektedir. WPS standardı OGC tarafından 2005 yılında konumsal analizlerin web servisleri tarafından gerçekleştirilmesi için yayınlanmıştır ve 2007 yılında güncellenmiştir. WPS konumsal işlemlerin yayınlanması, bulunması ve kullanılması için standard bir arayüz sağlar. WPS işlemlerinden kasıt, konumsal veri üzerinde çalışan herhangi bir algoritma, hesaplama ya da modeldir (OGC, 2007). Bu standard, tüm WPS'ler tarafından desteklenmesi zorunlu üç operasyon tanımlamıştır. Bunlar; GetCapabilities, DescribeProcess ve Execute operasyonlarıdır. Şekil 30'da çalışma prensibi görüldüğü üzere, kullanıcı "GetCapabilities" isteğini WPS'e gönderir. WPS, "GetCapabilities Response" mesajını kullanıcıya XML formatında döndürür. "GetCapabilities" cevabı ile servis metaverisi, servisin sunduğu her operasyonun ismi ve tanımları öğrenilir. Daha sonra "DescribeProcess" operasyonu ile kullanıcı, WPS sunucusunun sunduğu işlemleri kullanabilmesi için gerekli tüm girdi verileri (inputs), gerçekleştirilecek işlemin desteklediği veri formatları, işlem sonucunda üretilecek çıktı verisi ve formatı hakkında ayrıntılı bilgi alır (OGC, 2007). Bu operasyon, hangi işlem hakkında bilgi alınacaksa, o işlemin tanımlayıcısı (identifier) kullanılarak gerçekleştirilir. Kullanıcı bu iki XML dökümanını inceleyip son olarak "Execute"' isteğini WPS'e gönderir. Execute operasyonu,

WPS sunucusu tarafından sunulan bir işlemin girdi verilerinin tanımlandıktan sonra çalıştırılması ve işlem sonucunda çıktı verilerinin kullanıcıya sunulmasını sağlayan operasyondur. Kullanıcı, istediği bir işlemin tanımlayıcısını belirterek veri üzerinde işlem yapabilmek için Execute operasyonunu kullanır.



Şekil 30. WPS'nin çalışma prensibi

Bu çalışmada WSK gerçekleştirimi için Taverna iş akışı aracı kullanılmıştır. Taverna ile WSDL tanımına doğrudan ulaşılabilen servisler kompoze edilip çalıştırılabilmektedir. Fakat çalışma kapsamında kullanılan web servisleri çoğunlukla OGC WPS servisleridir. OGC servislerinin çalışma mekanizması yukarıda anlatıldığı üzere farklıdır. İstemci ile sunucu arasında karşılıklı gönderilen istek ve yanıt mesajları ile üç aşamada bir OGC servisinden veri elde edilebilmektedir. Bu yüzden OGC servislerinin Taverna'da WSK içinde doğrudan kullanılabilmesi için WSDL tanımlarına ihtiyaç vardır. OGC bu tanımları direkt olarak yayınlamamaktadır. Bu amaçla çalışmada kullanılmak istenen WPS servislerinin sahip olduğu operasyonların WSDL tanımlarını elde etme yoluna gidilmiştir. Literatürde WPS standardını sağlayan 52 North WPS (URL-59, 2015), Deegree WPS

(URL-60, 2015), Geoserver WPS (URL-61, 2015) ve PyWPS (URL-62, 2015) gibi açık kaynak kodlu yazılımlar mevcuttur. Bu çalışmadaki ogrbuffer ve gml2svg servislerinin WSDL tanımları PyWPS yazılımı kullanılarak elde edilmiş ve bu servisler WSK içine dahil edilmiştir. WSK'daki diğer servisler INTAMAR projesi tarafından sunulan WPS[9] sunucusunun sağladığı servislerdir. Bütün bu servisler GRASS GIS yazılımının konumsal işlemlerini kullanmaktadır. PyWPS (Python Web Processing Service), OGC WPS servisinin Python'da (URL-63, 2015) gerçekleştirimidir. DBU (Deutsche Bundesstiftung Umwelt) (URL-64, 2015) tarafından desteklenen bir proje olarak Mayıs 2006'da başlatılmıştır ve 2009 yılından beri çoğunlukla HS-RS (Help Service-Remote Sensing) (URL-65, 2015) tarafından finanse edilmektedir. PyWPS, varsayılan olarak GRASS-GIS fonksiyonlarına doğrudan destek sağlayacak şekilde yazılmıştır. Aynı zamanda istenilen özel bir konumsal fonksiyonun ya da modelin programlanmasını ve herkes tarafından erişilebilecek şekilde yayınlanmasını sağlayan bir ortam sağlamaktadır. Bu çalışmada OGC WPS servisleri çalışma mekanizması nedeniyle WSK içine doğrudan dahil edilemediği için, PyWPS ile bu servislerin sunduğu konumsal işlemlerin WSDL tanımları elde edilerek WSK içinde kullanılmıştır. PyWPS aracılığı ile WPS'den gelen operasyon tanımları WSDL'ye dönüştürülebilmektedir. Üretilen bu WSDL'ler Taverna'da import edilerek WSK içinde kullanılmıştır.

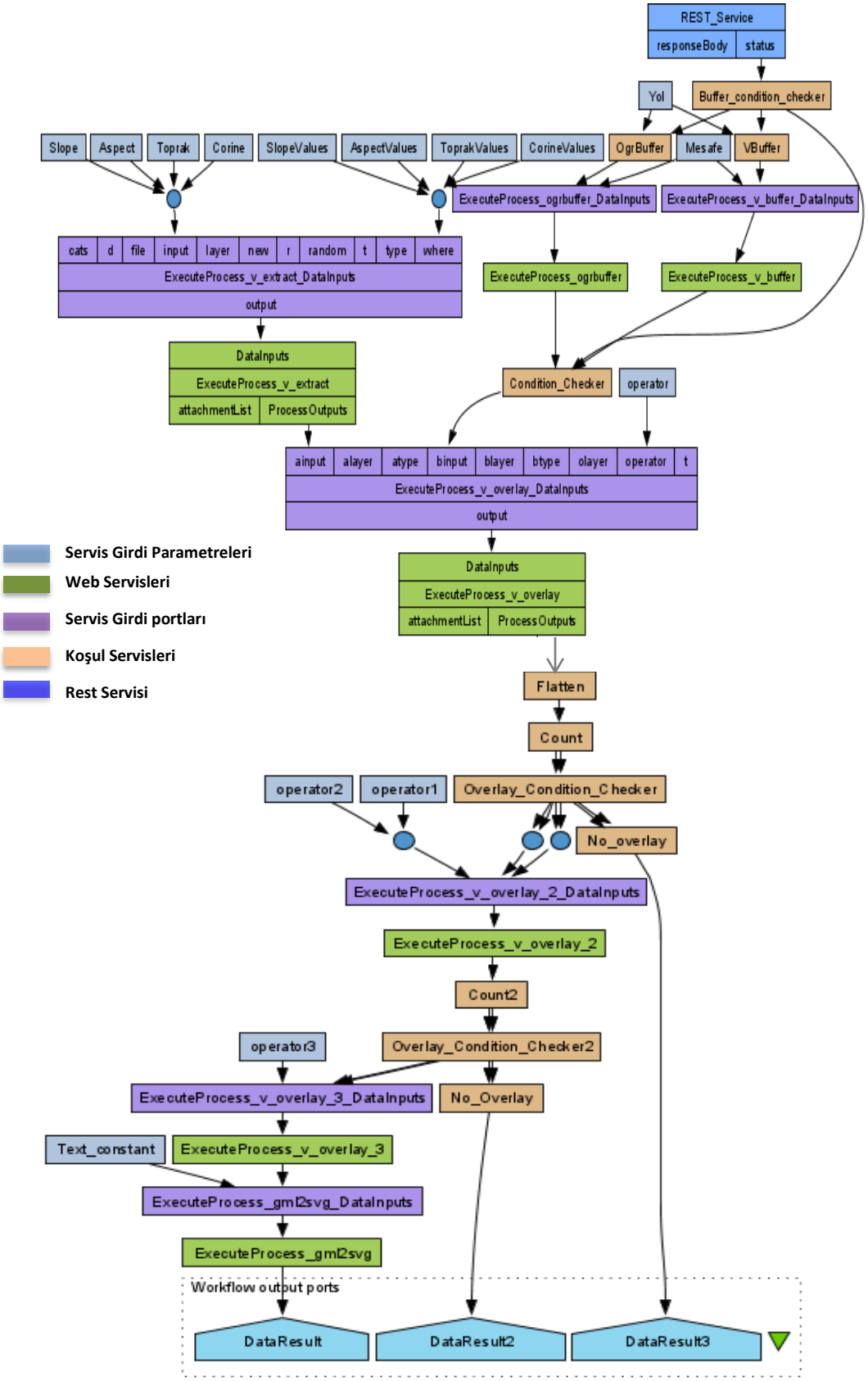
Somut WSK gerçekleştirimi için, eşleştirici aracılığı ile geliştiricinin amaç cümlesinden çıkarsanarak KSO'dan bulunan soyut WSK'ya bakılır ve iş akışı içinde yer alacak servisler öğrenilir. Geliştirici, Web'den yine eşleştirici aracılığı ile bu servislerin somut gerçekleştirimlerini bularak OWL-S işlem modelindeki kontrol akışına göre sıralar. Oluşturulan somut WSK, geliştiricinin amaç cümlesinden çıkarsanan verilere göre çalıştırılır. Oluşturulan somut WSK modeli, girilen GML veri tipindeki "yol", "eğim", "bakı", "toprak" ve "corine" girdi verilerini alır ve geliştirici isteğindeki seçim koşullarını ve operasyonlarını uygular, uygun katı atık depolama alanlarını bulur ve sonucu yine GML veri formatında döndürür.

Şekil 31'de görüldüğü üzere, WSK'nın ilk sırasını iki ayrı sunuculardan gelen buffer servisi oluşturur. Buffer servisi, verilen yol katmanını girdi olarak alır, girilen mesafe kadar tampon uygular ve sonuç yol katmanını üretir. İki tane buffer servisi olmasının sebebi, koşturum anında servislerden birine ulaşılamaması durumunda diğerinin çalıştırılabilmesidir. Buffer servislerinin koşum anında çalışabilir olup olmadığını test

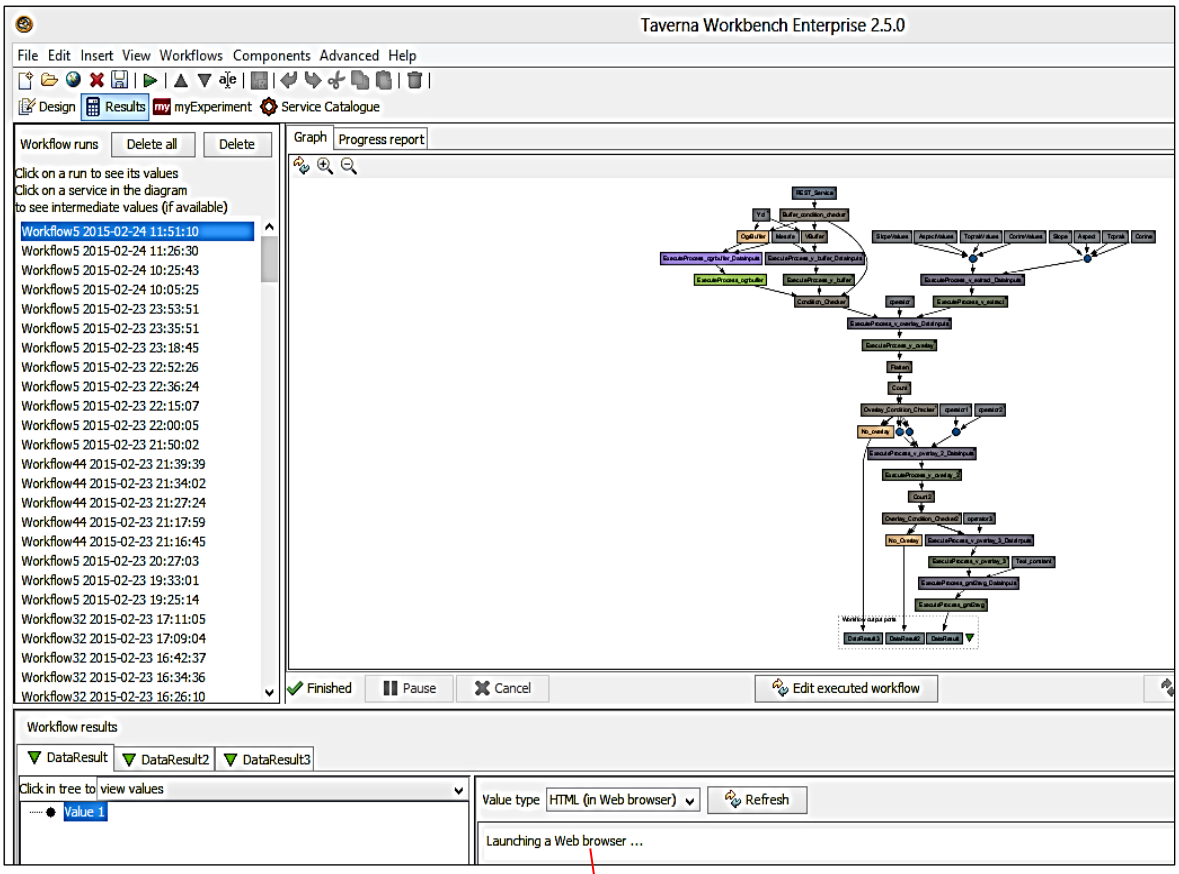
[9] http://www.earthserver.eu/, http://www.pml.ac.uk/

etmek için WSK içinde “Rest” servisi mevcuttur. Koşullu servis çalıştırma hakkında ayrıntılı bilgi aşağıda verilecektir. Sıradaki servis extract servisidir. Extract servisi, girilen katman verilerinden istenen seçim koşuluna göre seçim yaparak seçilmiş katman verilerini döndürür. Burada kullanıcı amaç cümlesinden çıkarsanarak belirlenen slope, aspect, toprak ve corine katmanlarına “where” girdi parametresi ile verilen “value” değerleriyle seçim yaptırılır. Hangi seçim koşulunun hangi katmana ait olduğu bilgisi, Taverna yapısında bulunan “List Handling” mekanizmasındaki işlem tipinin “bire bir” olarak ayarlanması ile sağlanır. Böylece extract servisi, sırasıyla her katmana karşılığındaki parametre değerlerini bire bir uygulayarak seçilmiş öznitelikleri döndürür. Sıradaki servis overlay servisidir. Overlay servisi, extract servisinden gelen seçilmiş katmanları ve buffer servisinden gelen buffer uygulanmış yol katmanını alır ve bindirme işlemi yapar. Overlay servisinin “operatör” girdi parametresi coğrafi kesişim (intersect) operasyonuna karşılık gelen “and”, coğrafi birleşim (union) operasyonuna karşılık gelen “or”, kesişimi dışında (disjoint) anlamına gelen “not” ve “xor” olmak üzere dört değer alabilir. Çalışmada buffer uygulanmış yol katmanı dışındaki alanlarla girilen diğer katmanların kesişimi söz konusu olduğu için “not” operatörü kullanılmıştır. Sıradaki operasyonlar “condition-checker” operasyonlarıdır. Bu koşul servisleri overlay operasyonu sonucunda oluşan katman sayısını kontrol eder ve tek bir katman kalıncaya kadar overlay operasyonunu tekrarlayarak çalıştırır. Eğer girdi katman sayısı “1” ise, overlay operasyonuna gerek kalmadığı için sonucu NoOverlay operasyonuna gönderir ve üretilen bu katman çıktı katmanı olarak görüntülenir. Eğer girdi katmanı sayısı “1”den fazla ise, yine overlay operasyonu çalışmaya devam eder. En sonunda elde edilen GML tipindeki kesişim katmanı “gml2svg” servisine gönderilir ve sonuç katman görselleştirilerek tarayıcıda açılır (Şekil 33). Oluşturulan sonuç katmanın GML tanımının bir kısmı Ek.2’de verilmiştir. Şekil 32’de kompozisyon içindeki servislerin hepsi çalıştıktan sonra WSK’nın görünümü gösterilmektedir. WSK içindeki bütün girdi ve çıktı katmanlar GML veri tipindedir.

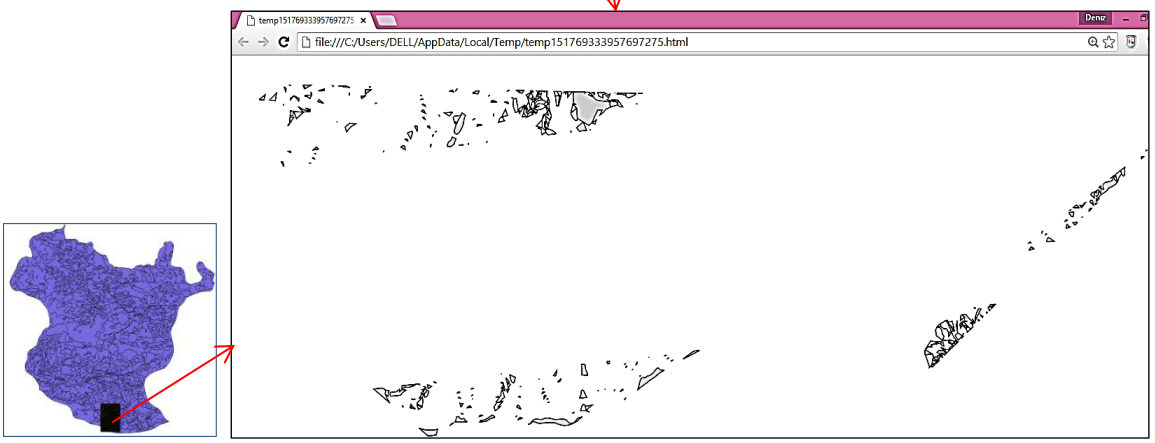
Uygulama kapsamında kullanılan girdi veriler Gümüşhane ilinin bir kısmına ait eğim, bakı, toprak ve corine katmanlarıdır. Bu katmanlar shapefile formatından GML formatına dönüştürüldükten sonra WSK içinde operasyon girdi katmanları olarak kullanılmıştır. Kullanılan veriler hazır olduğu için Gazetteer ve Bbox oluştur servisleri somut WSK içinde kullanılmamıştır.



Şekil 31. Somut WSK gösterimi

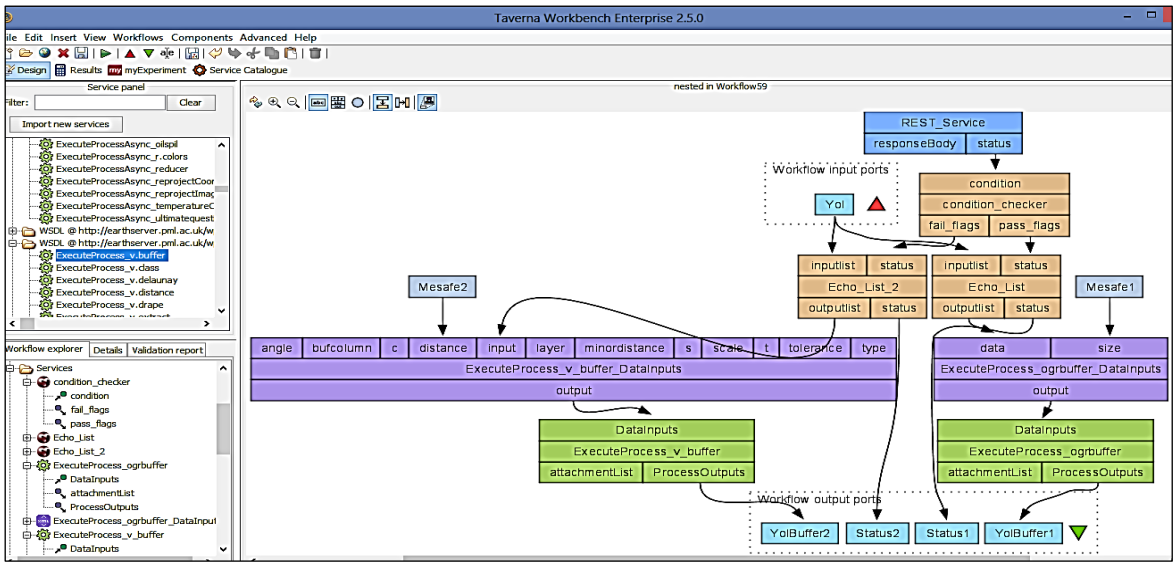


Şekil 32. Sonuç WSK gösterimi

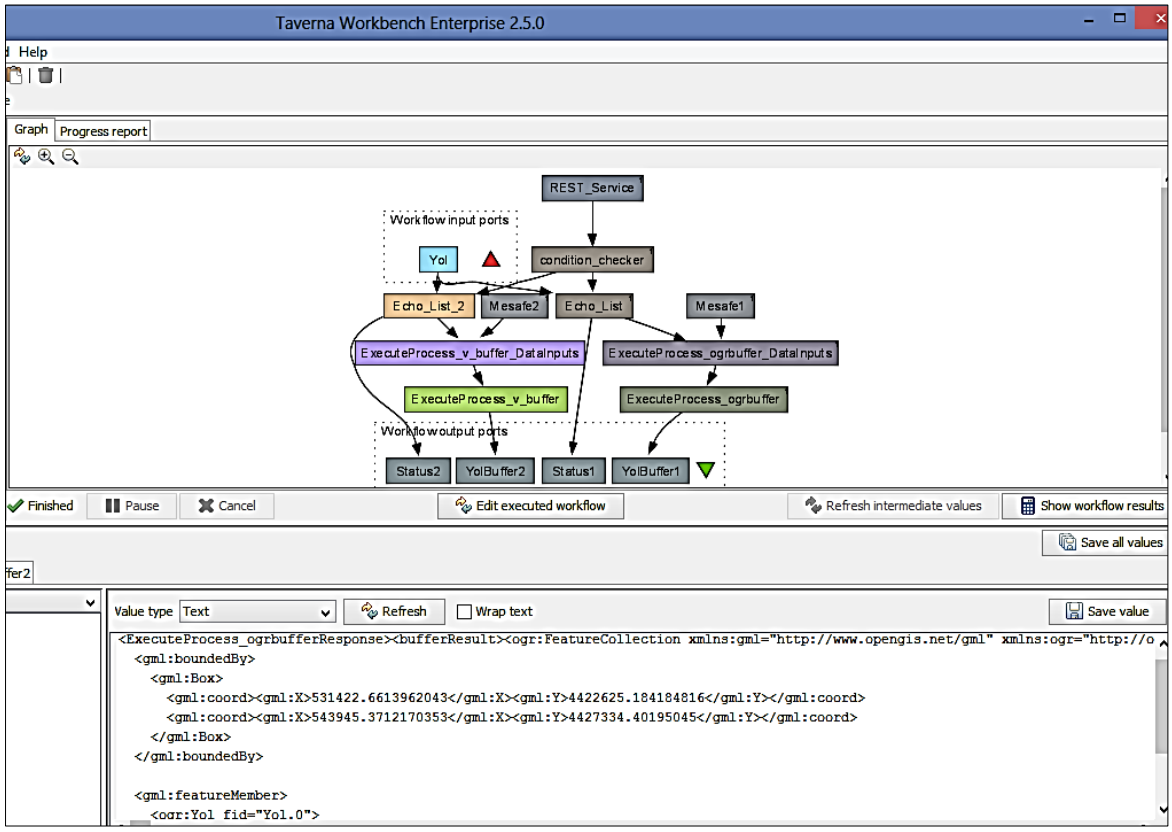


Şekil 33. WSK ile üretilen sonuç görüntü

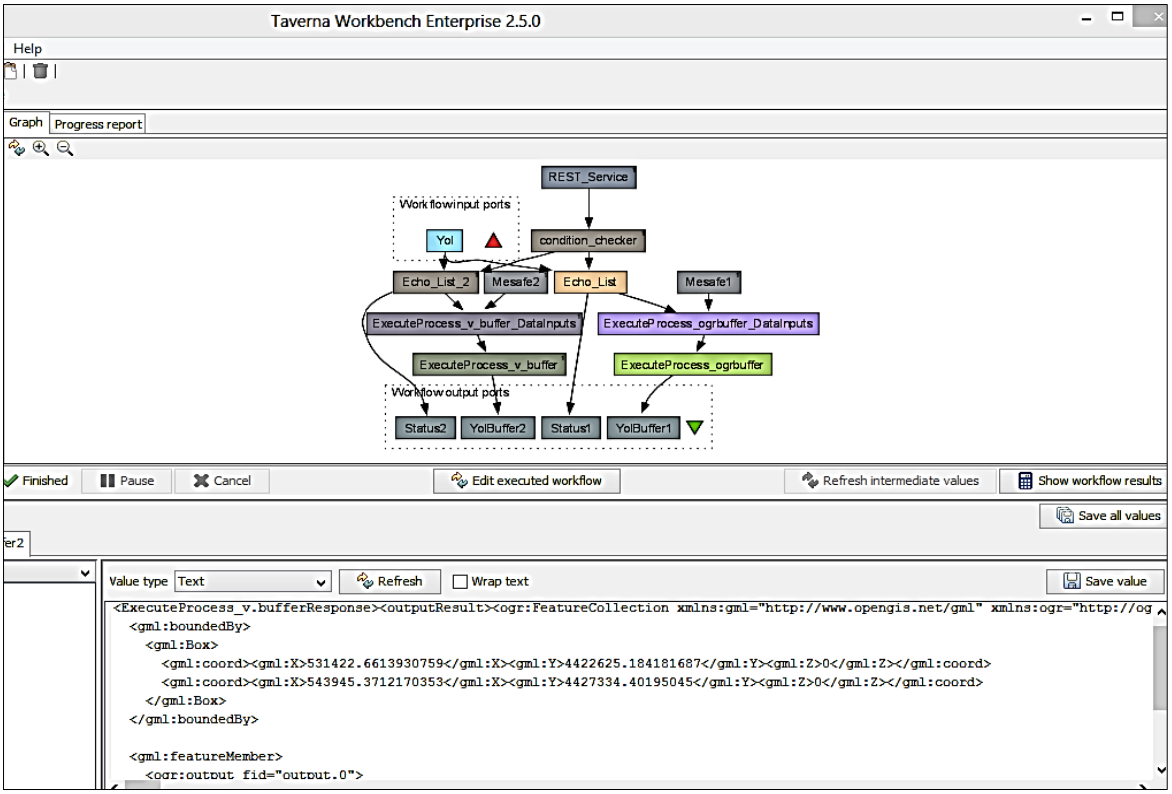
WSK alanındaki problemlerden biri, içerdiği servislerin tasarım (abstract) anında ulaşılabilir ya da çalışır durumda olsa bile, koşum (concrete) anında internette meydana gelebilecek herhangi bir kopukluk durumunda ya da servis sağlayıcının sunduğu servisi kaldırması nedeniyle ilgili servise ulaşılamaması sonucu kompozisyonun başarısız olmasıdır. Bu gibi durumlarda WSK'nın geçerliliğini koruması ve kullanılabilirliği için, söz konusu servise ulaşılamaması durumunda bir çözüm aynı işlevi gerçekleştiren başka bir servisin WSK'ya dahil edilmesidir. Bu tez çalışmasında WSK'nın ilk bileşen servislerinden "Buffer" servisi için bu yol izlenmiştir. Şekil 34'de görüldügü gibi OgrBuffer ve V.Buffer servisleri verilen çizgi, nokta ya da alan tipindeki bir katmana belirtilen mesafe kadar tampon bölge oluşturur. Bu iki servis ayrı sunucularda ayrı sağlayıcılar tarafından sunulmaktadır. Şekildeki yeşil renkler web servislerini, mor renkler servislerin girdi portlarını, mavi renk "http status" servisini, turuncu renkler koşul koymak için oluşturulmuş servisleri, açık mavi renkler kompozisyonun girdi ve çıktı parametrelerini, açık mor renkler ise servislerin girdi parametrelerinin alacağı değerleri göstermektedir. Kompozisyon çalıştırıldığında ilk olarak REST servisi OgrBuffer servisinin sunulduğu adresi kontrol eder. Bunun için sunucuya durum (status) kontrol mesajı yollar ve ilgili protokolden gelen durum cevabını bir sonraki servise gönderir. ConditionChecker servisi REST servisinden gelen durum cevabı olumlu ise bunu OgrBuffer servisine, olumsuz ise V.Buffer servisine yollar ve ilgili servis otomatik olarak çalıştırılır. Cevabın olumlu olması ilgili servisin kullanılabilir olması anlamındadır. Taverna'da servisler sorunsuz çalıştırıldıktan sonra gri renge dönüşür. Şekil 35'de görüldüğü üzere, OgrBuffer servisi kullanılarak kompozisyon çalıştırıldığında sağ taraftaki servisler gri renge dönüşür ve YolBuffer1, Status1 çıktı değerleri üretilir. Şekil 36'da görüldüğü üzere, V.Buffer servisi kullanılarak kompozisyon çalıştırıldığında ise sol taraftaki servisler gri renge dönüşür ve YolBuffer2, Status2 çıktı verileri üretilir.



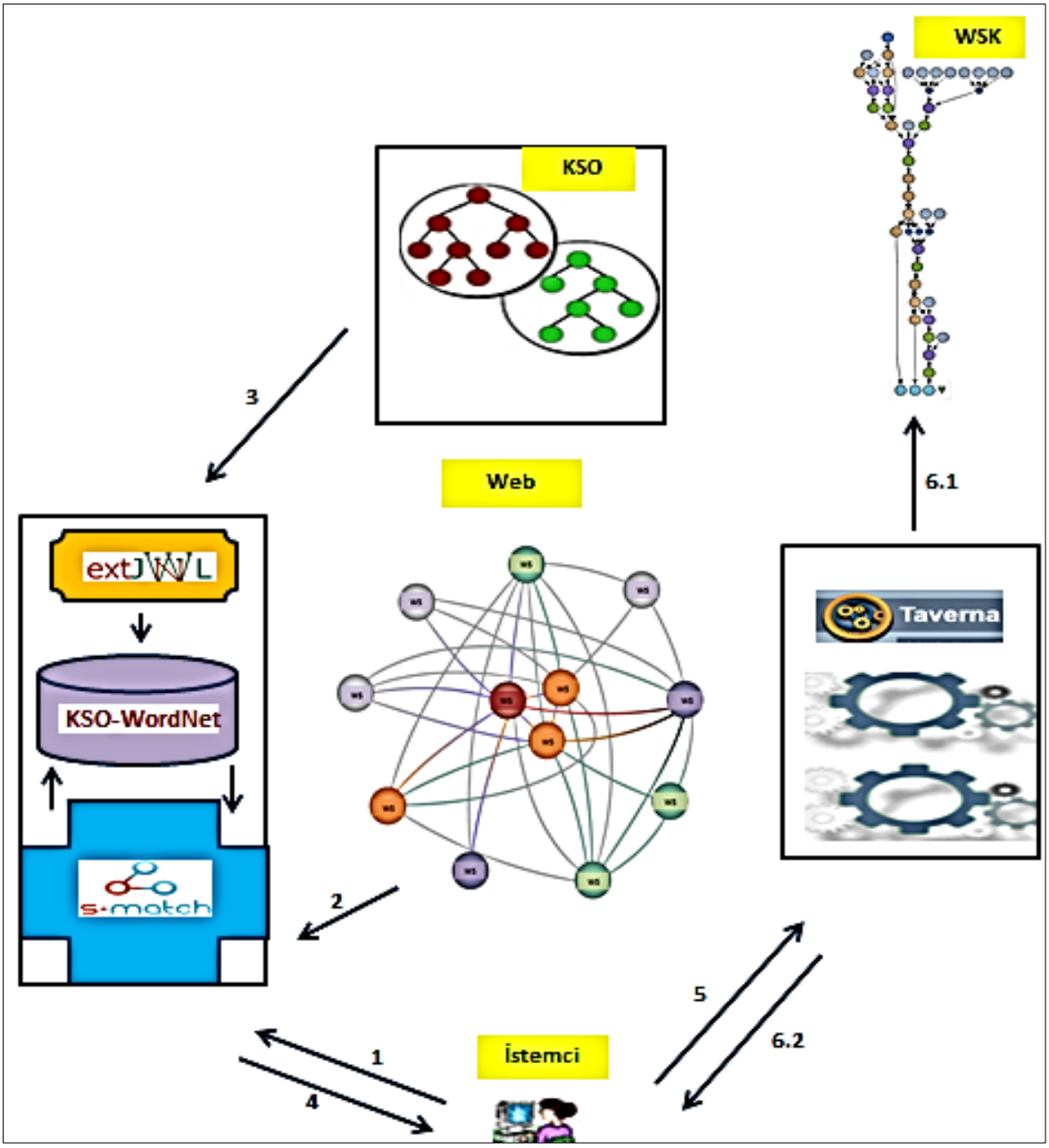
Sekil 34. Buffer servisinin koşullu seçimi



Şekil 35. Koşullu seçilmiş Buffer (ogrbuffer) servisinin Koşturum sonucu 1



Şekil 36. Koşullu seçilmiş Buffer (v.buffer) servisinin koşturum sonucu 2



Şekil 37. WSK Mimarisi

Şekil 37'de tez çalışmasında önerilen WSK mimarisinin ana bileşenlerinin işleyiş yapısı gösterilmektedir. Mimariye göre;

- Servis istemcisi gerçekleştirmek istediği uygulamaya yönelik bir amaç cümlesi üretir ve bu cümle OWL'de kodlanarak S-Match eşleştiricisine gönderilir.
- Web'de bulunan, çeşitli coğrafi operasyonlar yapan web servisleri STO'da yayınlanarak eşleştirme aracına gönderilir.

- Konumsal alana özgü katman ve operasyonları içeren KSO, arka planda bilgi tabanı olarak kullanılmak üzere extJWNL yazılımı ile KSO WN'ye dönüştürülür. S-Match, eşleştirme aşamasında KSO-WordNet'i kullanır.
- S-Match, yayınlanan servisleri temsil eden STO ile istemciden gelen sorgu cümlesini karşılaştırır ve eşleşen servisleri istemciye gönderir. Sonuç servisler arasında istemcinin amacına uyan servisler ve bu servislerle ilişkili olan kompoze işlem modeli mevcuttur. İstemci bulunan servisler arasından en yüksek benzerlik değeri olanları seçer.
- İstemci eşleştirme sonucunda elde ettiği servisleri, bulunan iş akışına göre WSK içine dahil eder. WSK'da ilgili servis parametrelerini ve seçim koşullarını girerek kompozisyonu Taverna'da çalıştırır.

Taverna, WSK koşum (6.1) sonucunda oluşturulan sonuç katmanı ve görüntüyü (6.2) istemciye döndürür.

3. BULGULAR VE İRDELEMELER

Bu bölümde, SWT desteğinde WSK oluşturma ile ilgili 1. bölümde anlatılan benzer çalışmaların bu tez çalışmasında önerilen WSK mimarisi ile farkı, önerilen mimarinin WSK alanındaki problemlerin giderilmesine yönelik çözümü ve mimarinin gereksinimleri irdelenecektir.

3.1. Önerilen Mimarinin Benzer Çalışmalar ile Farkının İrdelenmesi

Lemmens (2006), WSK'yı dağıtık konumsal servislerin semantik birlikte işlerliği için kullanmıştır. Çalışmada servis kompozisyonu, servis istemcisinin aradığı her servis için girdi, çıktı, ön koşul ve son koşul parametrelerine göre sorgu oluşturarak servisleri bulması ile gerçekleşmektedir. Çalışmada bulunan servisleri hangi iş akışı sırasında kullanacağı yine kullanıcının bilmesi gereken bir durumdur. Ancak kullanıcı, oluşturmak istediği uygulama için hangi servislerin hangi operasyonlarını hangi iş akışı sırasına göre kullanacağını ve bu operasyonları çalıştırabilmek için gerekli parametreleri bilmeyebilir. Bu tez kapsamında önerilen mimaride ise, kullanıcının kompozisyon işleyişi hakkında hiçbir şey bilmediği varsayıldığı için, hazır bir servis kompozisyonu kullanıcıya buldurulmakla, kompozisyon içindeki servisleri çalıştıran parametreler kompozisyon içine gömülerek, bu detaylar kullanıcıdan istenmeden daha üst düzey bir kompozisyon akışı oluşturularak kullanıcının yükü hafifletilmektedir.

Alam vd., (2007), konumsal web servislerini bulma ve seçme çatısı kapsamında WSK kullanmıştır. Çalışmada, uygulama istemcisi ile servis sağlayıcının servis tanımları aynı ontoloji ile referanslandırıldıktan sonra eşleştirme işlemi yapılmaktadır. Bu tez çalışmasında önerilen mimaride ise, karşılaştırılan sorgu ve servis ontolojisi ortak bir referans çatıya dayanarak oluşturulmamıştır. Ancak semantik eşleştirme aşamasında arka planda bilgi tabanı olarak kullanılan bir konumsal servis ontolojisinden yararlanılmıştır. Çalışmada WSK'da yer alacak servisler konumsal ontoloji desteğiyle buldurulmaktadır. Bu yönüyle bu tez çalışmasında önerilen WSK mimarisiyle benzerdir. Ancak, WSK'da yer alacak servisleri eşleştirirken Lemmens (2006)'deki gibi servis parametrelerini kullanmasıyla farklılık göstermektedir. Bu parametre tayini, istemciye karmaşık

gelebileceği ve vakit alabileceği için, bu tez çalışmasındaki WSK modeli bu parametre tayinini WSK içinde kullanıcıya sunmakla daha hızlı ve kolay bir işleyiş sağlayacaktır.

Zhang vd., (2010)**,** konumsal detayların WFS ve konumsal semanik web kullanılarak mantık tabanlı bulunması ve entegrasyonu için WSK kullanmıştır. Çalışmada WSK oluşumunda konumsal ontolojiler ve OGC filtre operasyonları kullanılmıştır. Bu yönüyle bu tez çalışmasında önerilen mimariye benzerdir. Ancak bu çalışmada OGC filter operasyonları ontolojideki konumsal ilişkileri temsil etmektedir. Bu tez çalışmasında ise ontolojideki konumsal operasyon sınıflarına karşılık gelmektedir. Bu çalışmada, kompozisyon içinde tek tip servisler (WFS) kullanılmaktadır. Fakat bu tez çalışmasında önerilen mimari ile ilgili katmanları (temaları) doğrudan ya da belirli ölçütlere göre filtreleyerek elde etmek için WFS'lerin, bu katmanlara uygulanacak konumsal operasyonlar için WPS (Web Processing Service)' lerin de kompozisyon içine dahil edilebilmesi ile farklı tipteki servislerin de kompozisyonu amaçlanmıştır.

Dong vd., (2006), OWL-S tabanlı dinamik servis kompozisyonu için WSK kullanmıştır. Çalışmada kullanıcı amacından semantik çıkarsama ile OWL-S servis ontolojisi desteğiyle "tightly coupled" olarak oluşturulmuş bir SWSK otomatik olarak buldurulmaktadır. Bu yönleriyle bu tez çalışmasında önerilen mimariye benzerdir. Çalışmada WSK oluşturulurken sadece sıralı kompozisyon uygulanmıştır. Ayrıca çalışmada oluşturulan WSK'daki her servisin kaç defa çalıştırılacağı önceden bellidir. Ancak bu tez çalışmasında sıralı kompozisyonun yanında seçim, koşul ve iterasyon gibi diğer OWL-S kontrol parametrelerinin de kullanılmasıyla daha çok amaçlı bir kompozisyon önerilmiştir. Ayrıca bu tez çalışmasında önerilen WSK mimarisinde bir WS'nin kaç defa çalıştırılacağı önceden belli değildir ve kullanıcının girdiği katman sayısına göre koşum anında değişerek o sayı kadar tekrar edecektir.

Brogi vd., (2006), SWSK bulmak için bir prototip geliştirmiştir. Çalışmada yukarıda anılan benzer çalışmalardaki gibi, istemci sorgusunu servis parametrelerine göre yapmaktadır. Bu da WSK işleyişi hakkında bilgisi olmayan bir kullanıcı için güçlük oluşturacaktır. Bu tez çalışmasında önerilen mimari daha önce açıklandığı üzere kullanıcıya bu bakımdan önemli bir kolaylık sağlamaktadır.

Pop vd., (2009)**,** çalışmasında doğal dil isteklerine dayalı anlık servis kompozisyonu oluşturmuştur. Bu çalışma doğal dil isteğiyle oluşturulmuş bir amaç tümcesinden ilgili web servislerinin bir kavram hiyerarşisine dayalı olarak çıkarsanması ile bu tez çalışmasında önerilen mimari ile benzerdir.

Withers vd., (2010), SADI ve BioMoby eklentileri ile Taverna'da semantik iş akışı gerçekleştirmiştir. Bu çalışmada servis girdi ve çıktı parametreleri aynı ontolojilerle tanımlanmıştır. Dolayısıyla farklı ontoloji kavramlarıyla tanımlanan servislerin bulunamaması problem oluşturacaktır. Ayrıca, SADI ile web servisleri ilişki bazında aranmaktadır. Ancak aynı ilişkiyi farklı işlev gerçekleştirmek için kullanan servisler olabileceği için, bu durum semantik açıdan servisleri ayırt edici bir özellik değildir. Bu çalışma servis bulma ve WSK oluşturma aşamasında alana özgü bir servis operasyonları sınıflamasından yararlanması ve Taverna aracını kullanması ile bu tez çalışmasında önerilen mimari ile benzerdir. Ancak, bu çalışmada her ne kadar son kullanıcıların zorluklarını gidermek amaçlansa da, eldeki bir veri ile başladıktan sonra "bu tip veriye ne tür operasyonlar uygulanabilir" bilgisinin kullanıcının hangi veriye hangi operasyonları uygulayacağını bilmesini gerektirdiği için, yük oluşturmaktadır. Bu da ancak alanında uzman kullanıcıların yapabileceği bir iştir. Bu tez çalışmasında önerilen mimari daha önce açıklandığı üzere kullanıcıya bu bakımdan önemli bir kolaylık sağlamaktadır.

3.2. Web Servislerini Ayırt Etmek İçin Sadece Parametre Tipi Yeterli Mi dir?

Geleneksel yaklaşımda web servisleri girdi ve çıktı parametre tiplerine göre ayırt edilmektedir. Ancak bu ayırım servisleri tanımak için yeterli değildir. Şöyle ki; farklı iki servisin girdi ve çıktı tipleri aynı olabilir, bu durumda servis zincirlemeyi bir önceki servisin çıktısı bir sonrakinin girdisi olacak mantığına dayanan çalışmalarda, "yanlış" bir servis zincire dahil edilebilir. Girdi ve çıktı parametresi "harita" tipinde olan "kesişim" ve "birleşim" servisleri buna örnek verilebilir. Bu iki servis farklı işlev göstermesine rağmen, girdi ve çıktı tipleri eşit olduğu için geleneksel yaklaşımla ayırt edilemez. Aynı şekilde, girdisi "yer ismi", çıktısı "harita" olan bir "yer seçimi" kompoze servisi ile atomik bir "display" servisinin de çıktı parametresi "harita" dır. Problemin çözümü için Withers vd., (2010) çalışmasında kullanılan yazılım aracı kullanıcıya servis ihtiyacı karşılar mı diye karar vermesi için, aranan ve bulunan bir servisin çıktı tipini göstermektedir. Bu tez çalışmasında önerilen WSK modeli servislerin girdi-çıktı parametrelerine göre servisleri bulmadığı için bu problemleri çözebilmektedir. Kullanıcı bu parametreleri oluşturulan soyut WSK tanımından öğrenecektir. Soyut WSK içinde kompozisyonun ve içeriğindeki her servisin OWL-S tanımı yer almaktadır. OWL-S tanımında bir servise ait metaveri ile bütün parametreler ve koşullar mevcuttur.

3.3. Format Farklılığı Problemi

Kullanıcı sorgusunda belirtilen parametrelerle sorgu sonucu bulunan web servislerinin girdi ve çıktı parametreleri farklı ise, eşleştirme ile çözüm sağlanır. Ancak eşleştirme sonrasında servislerin veri formatları ya da "temsil" leri uymuyorsa ne yapılacaktır? Örneğin, WSK içinde kendinden önce gelen servisin çıktı veri formatı "shapefile" ama kendi girdi veri formatı "GML" olan bir servis olabilir. Böyle durumlarda parametre eşleştirilmesi yeterli değildir. Parametre eşleştirmesinin ardından gerekiyorsa format dönüşümü için ilgili "Transformation" (Dönüşüm) servisleri WSK içine dahil edilerek format dönüşümü gerçekleştirilir. Temsil farklılığı için diğer örnekler projeksiyonları birbirine uymayan katmanlar, ya da birimlerden birinin "inch" diğerinin "metre" olduğu "uzunluk" tipleri olabilir. Bu tez kapsamında incelenen çalışmaların hiç birinde bu problem ele alınmamıştır.

Bu amaçla soyut WSK içine "if then else" OWL-S kontrol parametresi ile SWRL'de tanımlanmış bir "Format Transformasyon" servisi koyulabilir. Somut WSK'da ise bunun karşılığı, koşum anında devreye girecek bir "Transformasyon" koşul servisi olacaktır. Şöyle ki; format kontrolü yapan bu koşul servisi, test edeceği format tiplerini girdi olarak alır ve dönüşüm yaparak aynı formata getirir. Projeksiyon eşitlenmesi gereken durumlarda da benzer şekilde, önerilen WSK modeli içine bir "Projeksiyon Transformasyon" servisinin dahil edilmesi ile çözüm sağlanır.

3.4. Önerilen Mimarinin Tek Bir Sistemde Hızlı ve Yeni Uygulamalar Geliştirmeye Yönelik Boyutu

Bir CBS analiz uygulamasının kullanıcı tarafından gerçekleştirilebilmesi için gerekli iş akışının kullanıcı tarafından bilinmesi problemi bağlamında marketteki yazılımların destekleri bugün itibariyle çok zayıftır. Örneğin, CBS alanında Dünya'da lider konumundaki ArcGIS yazılımının bu amaca yönelik, "model builder" aracına bakılırsa, orada sadece "add tools" (araç ekle) veya "add data" (veri ekle) seçenekleri olduğu görülür. Bunun dışında geliştiriciye yönelik bir yardım, yol gösterme söz konusu değildir. Geliştirici, elindeki uygulama için hangi rutinleri kullanabileceğine tamamen kendi bilgisi ile karar vermek durumundadır. Bu tezde önerilen mimari ile, anılan sorunun çözümü yolunda katkı sunulabilir. Öte yandan, bu tezdeki yaklaşım tek bir sistemden ziyade tek bir

yazılım şirketi içerisinde farklı geliştiricilere hizmet edecek bir yapıya dönüştürülebilirse, o zaman çok daha değerli olacaktır. Öyle ki, şirketin önemli bir ya da birkaç geliştiricisinin işten ayrılması durumunda yaşanabilecek kayıplar bu yaklaşımla azaltılabilecektir.

3.5. Önerilen Mimarinin Farklı Ontolojilerin Kullanılmasına Yönelik Desteği

Literatürdeki çalışmaların çoğunda servis sağlayıcıların ve servis istemcilerin aynı servis ontolojisini kullanması öngörülmektedir. Bu tez çalışmasında önerilen mimaride farklı ontolojiler adapte edilebilmektedir. Örneğin bu tezde kullanılan yayınlanmış servisler GRASS GIS servisleridir. GRASS GIS ve ArcGIS konumsal işlemlerinin sınıflamasına göre yayınlanmıştır. Bu servisler tezde STO ile tanımlanmıştır. Diğer yandan, servis iş akış modelleri KSO ile tanımlanmıştır. KSO ve STO farklı ontolojik tanımlamalara sahiptir.

3.6. Önerilen Mimarinin Farklı Diller Kullanılmasına Yönelik Desteğinin İrdelenmesi

Tez çalışmasında önerilen KSO modeli sadece bir dil için kısıtlı değildir, her dil ile uyarlanarak kullanılabilir. Çalışmada istemcinin sorgusunu Türkçe söylediği durum ele alınmıştır. Ancak istemci amacını farklı dilde söylediğinde de bu model kullanılabilir. Örneğin, amaç tümcesini İspanyolca dilinde oluşturan bir istemci olabilir. Bu durumda KSO WN'de bulunan ilgili kavram sınıflarına "eş anlamlı" ilişkilerle ya da diğer WN ilişkileri ile ilgili sözcüklerin ispanyolca karşılıkları eklenir. Bu şekilde herhangi bir değişikliğe gerek kalmadan ya da yeniden WN oluşturulmadan sadece ispanyolca kavramların eklenmesiyle genişletme yapılır. Bu amaçla MultiWordNet kullanılabilir. MultiWordNet, İngilizce ve İtalyanca kelimeleri içeren çok dilli sözlüksel bir veritabanıdır. WordNet'e bir genişletme olarak Princeton Üniversitesi'nde geliştirilmiştir. MultiWordNet tarayıcısı İspanyol, Portekiz, Romanca, Latin ve İbrani WN'lerine erişime izin vermektedir (URL-43, 2015). Herhangi bir dilde bir WN oluşturulduktan sonra MultiWordNet'e entegre edilebilmektedir.

3.7. Önerilen Mimarinin Amaç Tümcesinin Mevcut "Arka Plan Bilgi Tabanında" Bulunmayan Kavramlar İçermesi Durumunda Sunduğu Çözüm

Çalışma kapsamında kullanıcıdan sorgusunu bir amaç cümlesi ile oluşturması istenir. Kullanıcı "Gümüşhane ili sınırlarında, çöp dökümü için uygun; anayola 1 km mesafeden fazla, eğimi %20'den az, kuzey yönüne bakan, toprak kalitesi VI, VII, ya da VIII olan ve arazi kullanım sınıfı taşlık veya çalılık olan yerlerin ortak alanını belirle ve sonuç haritayı göster" (bkz. 2.2.1.) amaç cümlesini oluşturan kelimeler yerine farklı kelimeler kullanarak da sorgusunu oluşturabilir. Örneğin, "Trabzon'da yola 10 km'den uzak, güneye bakan, çalılık olarak kullanılan yerleri getir" gibi veya benzeri farklı kelimelerle bir amaç tümcesi kurulabilir. Bu durumda da yine S-Match, bu sözcüklere en yakın olan servisleri bulacaktır. İlk cümlede geçen "yön" sözcüğü olmasa bile, eşleştirici "güney" sözcüğü ile "yön" sözcüğünün ilişkili olduğunu KSO'dan anlayarak yine "aspect" servisi ile ilişkilendirecektir. Çünkü, KSO'da "güney" sözcüğü ile "yön" sözcüğü arasında "tipinde" (hyponym) ilişkisi vardır. Ancak bu durumda ilk cümleden farklı olarak, eşleştirici "eşittir" ilişkisi yerine "güney" ile "aspect" arasında "az geneldir" ilişkisini bulur. Çünkü KSO'da "yön" ve "aspect" aynı eş anlamlı sınıfı içinde olmasına rağmen, "Güney", "yön" sözcüğünün KSO'da bir örneğidir. Yine de, sonuçta üretilen "az geneldir" ilişkisi kullanıcıya yapmak istediği işlemler için bir "aspect" servisinin gerekli olduğunu anlatmaya yetecektir.

Amaç cümlesinin, arka plan bilgi tabanında bulunmayan sözcükler içermesi durumunda ise, bu sözcükler KSO WN'de eş anlamlı sınıfları varsa o sınıfa eklenir, yoksa ilişkili olduğu sınıfla WN ilişkileri (üst-alt sınıf, tipinde, parçası vb.,) kurularak eklenir. Bu sözcükler de KSO WN'de bulunan servis tanımları (advertisement) ile ilişkilendirilir. Bu şekilde WN'de yapılan eklemelerle bu sorun çözülür. Örneğin, yukarıdaki cümlede "uzak" sözcüğü "mesafe" sözcüğünün eş anlamlı sözcük grubuna eklenir. "Güney" sözcüğü ise "yön" sınıfının "hyponym" (tipinde, alt kavram) ilişkisi ile bağlantılı olduğu sınıfa eklenir.

3.8. Önerilen Mimaride Somut Servisin "Tam Otomatik" Koşturulmasına Yönelik Yazılım Bileşenleri İhtiyacının İrdelenmesi

Mevcut haliyle önerilen WSK mimarisi yarı otomatiktir. Anılan bütün servisler oluşturulup WSK içine dahil edildikten sonra ve mimari için eksik olan diğer bileşenlerin de eklenmesiyle sistem tam otomatik bir işleyişe sahip olacaktır.

Çalışmada önerilen mimari bileşenleri ayrı ayrı yazılım araçlarında geliştirilmiştir. Semantik servis tanımlama ve KSO oluşumunda Protege ontoloji editörü kullanılmıştır. Semantik eşleştirme işleminde S-Match eşleştirme aracı kullanılmıştır. Soyut WSK oluşumu Protege OWL-S editörü eklentisinde yapılmış, somut WSK ise Taverna iş akışı motorunda tanımlanarak çalıştırılmıştır. Bütün bu araçları bütünleşik bir şekilde birlikte çalıştıran bir yazılıma ihtiyaç vardır.

Çalışmada geliştirici sorgusu manuel olarak kelimeler arasındaki noktalama işaretlerine ve boşluklara göre parçalanarak çizge yapısına dönüştürülmektedir. Bu işlemi otomatik olarak yapan URL-52 (2015), URL-53, (2015) çalışmalarındaki gibi bir araca ihtiyaç vardır. Ayrıca geliştiricinin sorgusunu gireceği bir kullanıcı ara yüzüne ihtiyaç vardır. Ara yüzden gelen sorguyu otomatik olarak parçalayan araç, daha sonra oluşturduğu sorguyu çizge yapısında OWL'de kodlanmış bir şekilde S-Match eşleştirme aracına gönderecektir.

Çalışmada kompoze servis tipinin seçilmesi, belirlenen bir ağırlıklandırma formülüne göre kullanıcı tarafından seçim esnasında hesaplanmaktadır. Bu ağırlıklandırma işlemini otomatik olarak yapıp, sonuçları kullanıcıya döndüren bir araca ihtiyaç vardır.

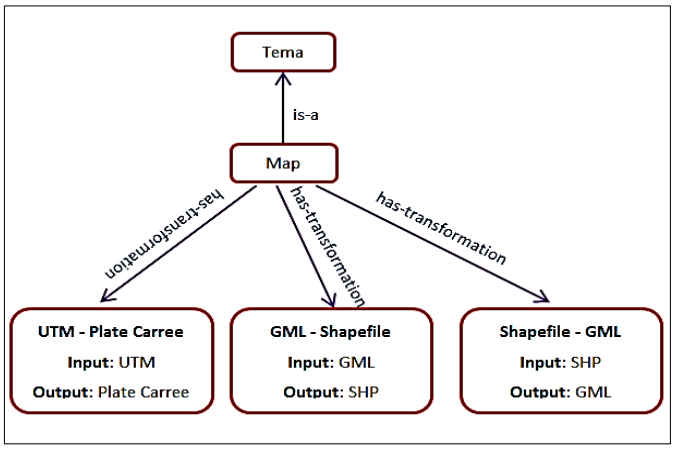
Önerilen mimaride ayrıca S-Match eşleştirme yazılımının entegre edilmesi öngörülmüştür. Mevcut gerçekleştirimde WSK içindeki servislerin girdi-çıktı parametrelerinin eşleştirilmesi, koşturumdan önce kullanıcı tarafından manuel olarak yapılmıştır. S-Match'i entegre edecek bir bileşenle, söz konusu eşleştirme işlemi soyut WSK içinde otomatik olarak yapılabilir. Şöyle ki, WSK'yı çalıştırmadan önce, içerdiği atomik servislerin her birinin kullandığı (bir servisin çıktısı ile kendinden sonra gelen servisin girdisi) girdi-çıktı parametreleri eşleştirilmek üzere S-Match'i kullanan bir "şema eşleştir" servisine gönderilecek ve bu servisten gelen eşleşme sonuçlarına göre servis parametreleri birbiri ile eşleştirilmiş olacaktır.

"Şema eşleştir" servisi KSO'da her servis içinde tanımlı olacaktır. Dolayısıyla, soyut WSK içinde her servis iş akışına göre kendisinden önceki servisin çıktı parametreleri ile kendi "girdi" parametrelerinin eşleşmesini bu yolla yapabilecektir. Ancak bu servisin

somut gerçekleştirimi yapılmamış, bu gerçekleştirim ileriki çalışmalara bırakılmıştır. Mevcut gerçekleştirimde, geliştirici soyut WSK'nın Protege'deki OWL-S tanımına ve Taverna'daki somut tanımına bakarak bu işlemi elle yapmaktadır.

3.9. Önerilen Mimarinin "Temsil Dönüşümü" Sorununa Yönelik Çözümü

Yukarıda belirtildiği üzere parametre eşleştirmenin ardından parametrelerin olası temsil uyuşmazlığı sorununun giderilmesi gerekir. Örneğin "uzunluk birimi" tipinde ifade edilen "uzunluk" özelliğinin "metre" ya da "inch" olarak temsil edilmesi, diğer yandan "harita" veri tipindeki bir özelliğin "GML" ya da "shp" temsilinde olması gibi farklılıkların giderilmesi olabilir. Bunun için öncelikle mevcut KSO'da "dönüşüm servisleri" nin modellenmesi gerekir. Bu da basittir. Bunun için KSO'da tanımlı her bir servis içerisinde, kendi girdi parametreleri sayısınca "dönüşüm servisi ()" bulunacaktır. Burada WSK içerisinde bir önceki servisin çıktı parametreleri ile bir sonraki servisin girdi parametreleri arasında "bire-bir eşleştirme" (URL-39, 2011) olacağı varsayılmaktadır. Dönüşüm servisleri şekil 38'de görüldüğü gibi modellenebilir. Burada her servis "has-transformation" ilişkisi ile kendi girdi tipleri için tanımlı dönüşüm servislerini çağırabilecektir. Böylece her servis girdi parametresinin WSK içinde bir önceki servisten gelen "temsil" ve kendi "temsil" tipini kendi içinde karşılaştıran bir "kural" ile ilgili dönüşüm servisini belirleyecektir.



Şekil 38. KSO'da dönüşüm servislerinin modellenmesi

3.10. Önerilen Mimarinin Sorgu "Seçim Koşulları" nın Yöneltilmesine Yönelik Çözümü

Kullanıcı amaç tümcesinde geçen seçim koşullarının ("kullanım sınıfı taşlık olan, "toprak kalitesi IV olan" gibi) hangi katmanlara uygulanacağı bilgisi WSK'nın çalıştırılmasından önce geliştiricinin ilgili katmanları belirtmesi ve o esnada "extract" servisine seçim koşulunu girmesi ile gerçekleşmektedir. Ancak eklenecek bir "şema import" servisi ilgili katmanı sunan servisin şemasını getirecek ve kullanıcının belirttiği seçim koşulu ile birlikte "şema eşleştir" servisine gönderecektir. "Şema eşleştir" servisi S-Match eşleştime aracına bu iki şemayı gönderecek ve eşleşen sonuçlara göre hangi seçim koşulunun hangi katmana ait olduğunu belirleyerek ilgili seçim koşulunu sıradaki "extract" servisine gönderecektir.

4. SONUÇ VE ÖNERİLER

Bu tez çalışmasında, Konumsal Veri Altyapıları (KVA) için kurumlar arası veri entegrasyonunu kolaylaştırmak amacıyla konumsal veritabanı şemalarının semantik eşleştirilmesine yönelik bir gerçekleştirim yapılmıştır. Ayrıca, KVA'ların web servisleri ile oluşturulması boyutuna yönelik, web servislerinin Semantik Web Teknolojilerine (SWT) dayalı olarak kompozisyonu için bir mimari önerilmiştir. Literatürde bu konularda yapılan çalışmalar mevcuttur. Ancak problemler henüz aşılmış değildir.

Çalışmada farklı kurumlar tarafından sunulan konumsal şemaların aralarındaki benzerliklerin semantik eşleştirme ile bulunmasına yönelik Harita Genel Komutanlığı (HGK) karayolu şeması ile INSPIRE RTN şeması seçilmiş ve aralarında semantik eşleştirme yapılmıştır. Semantik eşleştirmenin başarılı bir şekilde gerçekleştirilmesi için karşılaştırılacak kaynak ve hedef veri tabanı şemalarını uygun bir şekilde kapsayan arka plan bilgi kaynaklarına ihtiyaç vardır. Literatürde bu amaçla kullanılan genel ve geniş çaplı kaynaklar mevcuttur. Fakat bu kaynaklar konumsal alanı gerekli derinlikte kapsamamaktadır. Bu nedenle, çalışmada konumsal alana özgü kavramları içeren semantik eşleştirme işlemine yetecek kadar bir Ulaşım Alan Ontolojisi (UAO) oluşturulmuştur. Oluşturulan UAO, S-Match eşleştiricisinde varsayılan arka plan bilgi tabanı olan WN yerine kullanılarak söz konusu veritabanı şemaları eşleştirilmiştir. Ayrıca, S-Match arka plan bilgi tabanında WN ve GWN kullanılarak eşleştirme gerçekleştirilmiştir. Sonuçlar birbiriyle karşılaştırılmış ve UAO kullanıldığında bulunan sonuçların daha çok ve daha doğru olduğu görülmüştür. Bu alanda literatürde mevcut problemler hala tam anlamıyla çözülebilmiş değildir. Bu problemlerden biri aynı veriyi tanımlayan farklı konumsal veritabanı şemalarının, birinin diğerine göre daha genel olması gibi, farklı derinliklerde oluşturulması nedeniyle eşleştirme sonucunda bazı kavramların daha özel sınıflar yerine çoğunlukla daha genel sınıflarla eşleşmesidir. Diğer bir problem ise, şema kavram isimlerinde bulunan kısaltmalar, rakamsal değerler gibi sadece şemayı tanımlayan kişilerin ya da o alanda uzman kişilerin anlayabileceği tarzda özel ifadeler kullanılmasının eşleştirmede zorluk oluşturmasıdır. Bu nedenle literatürde mevcut genel içerikli kaynaklar yerine, uygulama alanına özgü oluşturulmuş UAO'nun arka plan bilgi tabanı olarak kullanılması bu problemlerin azaltılması bakımından önemlidir. Eşleştirme alanındaki bu

problemlerden yola çıkarak, bu aşamada elde edilen semantik eşleştirme bilgisi tezin ikinci aşamasında kullanılarak WSK alanında daha somut bir sonuç üretme yoluna gidilmiştir.

Web Servisi Kompozisyonu (WSK), birden çok Web Servisinin (WS), bir araya getirilerek yeni bir web servisinin oluşturulması demektir. WSK'nın ana hedeflerinden biri mevcut web servislerini bir iş akışı içinde kompoze ederek yeniden kullanılabilirliğini sağlamaktır. WSK, WS alanında son on yılın en aktif araştırma konularından biri olmuştur. Bu alanda çeşitli akademik ve pratik çalışmalar bulunmaktadır. Ancak, Tam Otomatik WSK (TOWSK) henüz çözümlenememiştir. Bunun nedeni TOWSK'nın içerdiği sorunların insan müdahalesi olmadan çözümlenebilmesinin çok zor olmasıdır. Çünkü bu elindeki bir CBS uygulamasını, masaüstü ya da web CBS tarzında gerçekleştirecek bir kullanıcının, "insan" olarak yaptığı işlemlerin tümünde yüklü olan "semantiğin", yarı-otomatik ya da tam otomatik yapıdaki iş akışına "gömülmesi" demektir. Bu çok kolay bir iş değildir ve çok sayıda problem içermektedir. Bu tez çalışmasının amacı, bu problemlerin bir kısmı için, SWT kullanılarak belirli çözüm önerilerinin sunulmasıdır.

WSK alanındaki birinci problem kompozisyondaki iş akış sırasının belirtilmesi problemidir. Geleneksel yaklaşım, iş akışının kullanıcı yani insan tarafından belirlenmesidir. Geliştirici ya da kullanıcının kompoze servisi oluşturan bütün bileşen servislerin girdi ve çıktı parametrelerinin koşum sırasında eşleştirilmesidir (Klien vd., 2006 ; Cömert vd., 2010; Brogi vd., 2006; Lemmens, 2006; Brogi ve Corfini, 2007; Alam vd., 2007, Dong vd., 2006). Ama bu çoğu zaman çok kolay değildir. Çünkü hangi servislere, hangi koşum sırasında ihtiyaç duyulduğu, bu servislerin birbiri ile uyumlu olup olmadığına karar verecek kullanıcının teknik olarak çok donanımlı olmasını gerektiren bir durumdur. Öte yandan bu durum, günümüz "uygulama geliştirme" eğilimlerinde en temel faktör olan "hızlı geliştirme" bakımından da kabul edilebilir bir tarz olmayacaktır. Bu tezde önerilen mimarinin yaklaşımı ise, kullanıcının amacını anılan geleneksel çalışmalardan çok daha üst düzeyde fakat bir sonraki adımda amaç tümcesinin bileşenlerine ayrılmasına yardımcı olacak bir formda, belirtmesidir. Amaç cümlesinden çıkarsanarak bulunan kompoze servis tipi ile ilgili WSK'nın iş akış sırası da bulunmuş olacaktır. Bu amaçla ilk olarak, belirtilen amaç cümlesinden istenen servisin hangi kompoze servis tipine uyduğu, amaç tümcesi ve STO ontolojilerinin S-Match eşleştirme aracında eşleştirilmesi ile bulunur. Bulunan bu kompoze servis tipinin OWL-S işlem modeli tanımı (soyut WSK) içinde hangi atomik servislerin hangi sırada, hangi kontrol parametreleri ve koşullarla arka arkaya dizildiği

bilgisi mevcuttur. Kullanıcı bu soyut WSK'daki işlem sırasına göre web servislerini kompozisyona dahil ederek çalıştırır.

WSK alanındaki ikinci problem, kompozisyon içindeki servislerin girdi ve çıktı parametrelerine göre bulunarak kompozisyona dahil edilmesidir. Geleneksel yaklaşımda kullanıcı aradığı servisleri girdi ve çıktı parametrelerine göre aramaktadır. Web servislerini ayırt etmek için sadece parametre tipi yeterli değildir. Şöyle ki, aynı parametre tipleri ile çalışan, farklı işlev gerçekleştiren servisler olabilir (girdi ve çıktı parametresi "harita" olan "kesişim" ve "birleşim" servisleri gibi). Withers vd., (2010) ve Kietz vd., (2014) çalışmalarında eldeki veriye ne tip işlemler uygulanabilir? tarzındaki yaklaşımı anılan problemin çözümü içindir. Söz edilen yayınlarda sistem, semantik çıkarsama ile belli tipteki veriye hangi işlemler uygulanabileceğini söylemektedir. Ancak bu durumda da iş akışı tasarımcısının her bir işlemin ne olduğunu bütün detayları ile bilmesi ve iş akışını ona göre yönlendirmesi gerekir. Bu durumun tasarımcıya yüklediği yük oldukça ağırdır. Bu tez çalışmasında önerilen yaklaşımda ise "soyut servis kompozisyonu", "çıkarsama" yoluyla kullanıcıya buldurulduğu için, kullanıcının belirtilen yaklaşımlardaki gibi bir "girdi-çıktı" parametre eşleştirmesi yapması gerekmemektedir. Bu detaylar yerine kullanıcıdan kurduğu amaç cümlesinde kullandığı sözcüklerle çıkarsama yapılabilecek bir konu (context) oluşturması istenmektedir. Daha sonra cümle içindeki sözcükler birbiri ile bağlantılı olarak değerlendirilip konu dışına taşmadan istenen servisler bulunmaktadır. Bu amaçla, kompoze servis tipinin belirlenmesinde ağırlıklandırma hesaplanmakta, bulunan bütün kompoze servis tipleriyle olan semantik ilişkilerin bir bütün olarak değerlendirilerek cümlenin en çok hangi kompoze servis tipine benzediği belirlenmektedir. Bulunan kompoze servis tipi ile aynı zamanda o servisin soyut WSK tanımı da bulunduğu için, WSK içindeki servislerin hangi girdi çıktı parametreleri ile çalıştı öğrenilmektedir. Daha sonra kullanıcı bu parametrelere göre somut WSK'yı çalıştırmaktadır.

WSK alanındaki üçüncü problem, kompozisyonu oluşturan servisler arasında verinin bir servisten diğerine "uygun" bir şekilde aktarılmasıdır. Öyle ki, bir servisin birden çok çıktı parametresi olması durumunda, bu parametreleri girdi olarak alacak bir servis için, hangi "çıktı" parametresinin, hangi "girdi" parametresine karşılık geldiği, yani "semantik" olarak nasıl eşleştikleri bilinmelidir. Diğer yandan, semantik olarak eşleşen parametrelerin "temsilleri" farklı olabilir. Örneğin her ikisi de "harita" tipinde olan ve semantik olarak birbiri ile eşleşen iki parametreden biri "GML" diğeri "shape" temsilinde olabilir. Bu durumda, elde edilecek "somut" WSK'nın çalışabilmesi için, parametre eşleştirmesi

ardından bir de “temsil” dönüşümü uygulanması gerekir. Bu tez kapsamında taranan literatürde her iki problem de ele alınmış değildir. Yukarıda ve ilgili bölümlerde de açıklandığı üzere, WSK iş akışının girdi ve çıktı parametrelerinin veri tipi (ya da “kavramı”) eşleşmesi üzerinden sağlandığı ve tek parametre farzedildiği için (Brogi vd., 2006) buradaki problem söz konusu değildir. Ancak, hem tek parametre olması hem de yukarıda anılan “2” no’lu WSK problemi nedeni ile bu tarz yaklaşımlar zaten sorunludur. Temsil farklılığı sorunu Kietz vd., (2014) gibi yaklaşımlarla aşılabilir görünse de bu tarz yaklaşımlarda da kullanıcı ya da geliştiricinin iş akışını yönetebilme güçlüğü nedeni ile sorunludur. Bu tezde önerilen mimaride kullanıcı amaç tümcesinden çıkarsanarak bulunan soyut WSK tanımında kompozisyon içindeki servislerin birbirine hangi parametreleri aktararak bağlandığı belli olduğu için, kullanıcı koşum anında soyut WSK’ya bakarak servis parametrelerini manuel olarak eşleştirmektedir. Yani geleneksel yaklaşıma göre daha kolay olmaktadır. Ancak gelecekteki çalışmalarda bu sorunun S-Match yazılımının WSK içine entegrasyonunun sağlanması ile semantik şema eşleştirme gerçekleştirilerek çözülmesi öngörülmektedir. Temsil farklılığı sorunun da “dönüşüm servisleri (transformation services)” ile aşılması öngörülmektedir.

Anılan bütün bu problemlerin çözümüne öneri olarak, bu tez çalışması kapsamında örnek bir yer seçimi senaryosunun SWT ile WSK kullanılarak gerçekleştirilmesine çalışılmış ve bu amaçla dört temel aşama uygulanmıştır. Söz konusu aşamaların gerçekleştiriminde son kullanıcı olan istemci hangi coğrafi operasyon için hangi web servisini kullanacağını, bu servise nasıl erişeceğini ve eriştikten sonra nasıl çalıştıracağını bilmesi gerekmemektedir. Kullanıcıdan sadece yapmak istediği uygulamaya yönelik amacını bir cümle ile belirtmesi istenmektedir. İlk aşama kullanıcının doğal dille yönelttiği üst düzey amaç cümlesinden belirli çıkarsamalar yaparak istenen kompoze servis tipinin ve bu serviste yer alacak ilgili atomik web servislerinin belirlenmesidir. Bu amaçla kullanıcı sorgusu, çalışmada oluşturulan servis tanımları ontolojisinin (STO) ayrı ayrı iki sınıfı ile (“Kompoze Konumsal Operasyonlar”, “Atomik Konumsal Operasyonlar”) eşleştirme yazılımında karşılaştırılır ve eşleşen kompoze ve atomik servisler bulunur. Bu aşamada kompoze servis tipinin bulunması ile bu servisin kompoze işlem modeli de bulunmuş olmaktadır. İkinci aşama, bulunan atomik web servislerinin kompoze servis tipinin “iş akış şablonu” (OWL-S “işlem modeli”) itibariyle hangi tipte servislere karşılık geldiğinin belirlenmesidir. Bunun için, bulunan atomik web servisleri ile kompoze servis “işlem modeli” eşleştirici ile karşılaştırılarak KSO’dan atomik web servislerinden her birinin

işlem modelindeki servislerle olan anlamsal ilişkisi çıkarsanır. Üçüncü aşamada kullanıcı, bulunan web'de yayınlanmış STO servislerinin hangi parametre ve koşullara göre çalıştıracağını bu iş modelinden öğrenir ve iş modeli iş akışı sırasında koşturulmaya hazır WSK (somut WSK) ya dahil eder. Son aşama olarak kullanıcı, Taverna iş akışı aracında ilgili servislere seçim koşulunu girerek somut WSK'yı çalıştırır ve koşum sonucunda katı atık yer seçimi için uygun alanları bulmuş olur.

Çalışmada söz konusu aşamaların gerçekleştiriminde en önemli gereksinim, arka planda bilgi kaynağı olarak konumsal tema ve operasyon sınıflamasını temsilen bir ontolojinin kullanılmasıdır. Literatürde CBS işlemlerinin sınıflandırılması konusunda çalışmalar mevcuttur. Ancak, bu tez açısından önemli olan sınıflandırmanın hangi amaca hizmet edeceğidir. Bu noktadan bakıldığında bu tezdeki ihtiyacı karşılayacak bir sınıflandırma literatürde yoktur. Çünkü konu henüz çözümlenebilmiş değildir ve bu da böyle bir sınıflandırma ne temelinde olmalı ya da hangi sorunu çözmeye yönelik olmalı boyutuna bir düzey getirilememiş olması nedeniyledir. Bu tezde yalnızca temel prensipleri sunulacak bir sınıflandırma ile, tezin ana katkı noktasını ispata yetecek kadar bir "mini" sınıflandırma yapılmıştır. Çalışma kapsamında oluşturulan KSO, "Yer seçimi" ve benzeri (Risk analizi vb.,) tipteki bir servisin kullandığı veri temalarını ve operasyonları tanımlamaktadır. Bu temalara uygulanacak operasyonlar da KSO'da ayrı sınıflarda tanımlanmıştır. KSO dört temel bileşenden oluşturulmuştur. "Bölge" sınıfı birinci temel bileşendir ve çalışma kapsamında oluşturulan "coğrafi yer" veri ontolojisini import eder. "Tema" sınıfı ikinci temel bileşendir ve çalışma kapsamında oluşturulan "tema" veri ontolojisini import eder. "Filtre Operatörü" sınıfı üçüncü temel bileşendir ve OGC'nin standard olarak tanımladığı filtre şemasından (OGC, 2014) oluşturulan "Filtre" ontolojisinin import edilmesiyle oluşturulmuştur. "Yer seçimi işlemi" KSO'nun son temel bileşenidir ve yer seçimi kompozisyonunun içerdiği operasyonların iş akışı bilgisine sahiptir.

Sonuç olarak bu çalışmada önerilen WSK mimarisiyle, uygulama geliştirmek isteyen bir kullanıcının kompozisyon işleyişi hakkında hiçbir şey bilmediği varsayıldığı için, doğrudan hazır bir servis kompozisyonu kullanıcıya buldurulmakla, işleyiş detayları kullanıcıdan istenmeden daha üst düzey bir kompozisyon akışı oluşturulmuştur. Böylece kullanıcı yükü hafifletilerek, zamandan da tasarruf sağlanacağı için önerilen yöntem, bu bağlamda, literatürdeki mevcut yaklaşımlara üstünlük sağlamaktadır.

Dolayısıyla, bu çalışma “yarı otomatik WSK oluşturma” ya bir örnektir. Ancak, 6. bölümde belirtilen araçların ve servislerin bu mimariye dahil edilmesiyle otomatikleşme artacaktır. Diğer bir anlatımla, “amaç” tümcesini “yeniden düzenleme” ye yönelik ve semantik eşleştirmeye yönelik yazılım bileşenlerinin eklenmesiyle Tam otomatik WSK’ya giden yolda ilerleme sağlanmış olacaktır. Ancak bu konu, bu tez çalışması sonrasında gerçekleştirilecek çalışmalara bırakılmıştır.

Amaç tümcesinin yeniden düzenlenmesi konusu biraz daha detaylandırılacak olursa, mevcut mimaride, “amaç” (goal) tümcesini bazı yapısal bileşenlere indirgeyecek bir bileşene ihtiyaç vardır. Burada kastedilen, amaç tümcesinden kullanıcının ne yapmak istediğinin anlaşılmasıdır. Çünkü kullanıcı da neyi, nasıl söylemesi gerektiğini bilemeyebilir; Elindeki problemin, bir “yer seçimi (site-selection)” mi yoksa “konuma dayalı seçim” (select by location) mi olduğunu bilemeyebilir. Şöyle ki, KSO da bir sınıf olarak gözüken her bir işlem tipi, kullanıcının nihai amacı olarak belirtilebilir. Bu durumda çok sayıda sınıf olacağı için kullanıcı hangisini seçecektir? Amaç tümcesini parçalamaya yönelik detaylı çalışmalar yapılması gereklidir. Bu amaçla “doğal dil işleme” (Natural Language Processing) tekniklerinden yararlanılmalıdır. Ayrıca, hangi coğrafi uygulamalarda çoğunlukla hangi operasyonların ve katmanların kullanıldığına yönelik bir araştırma yapılması gereklidir. Bu bağlamda, Partyka vd., (2010) çalışmasında kullanılan “Google mesafe” hesabı gibi hangi coğrafi sözcüklerin web’de en çok yan yana kullanıldığının hesaplanması ve “kümeleme” (“clustering”) algoritmalarından yararlanılabilir. Ayrıca, web’de en çok aranan konumsal işlem ve katmanlar (Renier ve Gilberto, 2005) bilgisinden yararlanılabilir. Böylece her coğrafi uygulama için bir operasyon ve veri “pattern”i geliştirilmesi gereklidir. Bu konuda çözüm olarak “Metin Madenciliği” (“Text Mining”) tekniklerinin kullanılması önerilmektedir. Gelecek çalışmalar buna yönelik olarak planlanmıştır.

5. KAYNAKLAR

Akıncı, H., 2006. Konumsal Veri Altyapılarının Web Servisleri ile Gerçekleştirilmesi: Mevcut Durum Analizi ve Gelecek Yönelimlerinin Belirlenmesi, Doktora Tezi, Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Trabzon.

Alam, A., Subbiah, G., Khan, L. ve Thuraisingham B., 2007. DAGIS: A Geospatial Semantic Web Services Discovery and Selection Framework, GeoS 2007, LNCS 4853, pp. 268–277.

Albrecht, J., 1995. Semantic Net of Universal Elementary GIS Functions. Proceedings, Twelfth International Symposium on Computer-Assisted Cartography (Auto-Carto 12), Charlotte, NC. GI Forum 2008, 232-241.

Aumueller, D., Hai Do, H., Massmann, S. ve Rahm, E., 2005. Schema and Ontology Matching with COMA++, SIGMOD 2005, Baltimore, Maryland, USA, June 14–16.

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

Berners-Lee, T., Hendler, J. ve Lassila, O., 2001. The Semantic Web, Scientific American Magazine, 284,5, 34-43.

Berry, J. K., 1987. Fundamental operations in computer-assisted map analysis, International Journal of Geographical Information Science, cilt 1, sayı 2, 119–136.

Börger, E. ve Stark, R. F., 2003. Abstract State Machines-A Method for High-Level System Design and Analysis. Springer-Verlag.

Brogi, A., Corfini, S., Aldana, J. F. ve Navas, I., 2006. A Prototype for Discovering Compositions of Semantic Web Services, In SWAP.

Brogi, A. ve Corfini, S., 2007. Behaviour-aware discovery of Web service compositions. International Journal of Web Services Research, 4, 3, 1-25.

Bruijn, J de vd., Web service modeling ontology (WSMO). http://www.w3.org/Submission/WSMO/, 28.03.2015.

Burrough, P. A. ve McDonnell, R. A., 1998. Principles of Geographical Information Systems (Spatial Information Systems), Oxford University Press, 2nd edition.

Carroll, J. J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A. ve Wilkinson, K., 2003. Jena: Implementing the Semantic Web Recommendations, Tech. Rep. HPL-2003-146, HP Laboratories Bristol.

Cheung-Foo-Wo, D., Tigli, J. Y., Lavirotte, S. ve Riveill, M., 2006. Wcomp: a multi-design approach for prototyping applications using heterogeneous resources, In 17th IEEE Intern. Workshop on Rapid Syst. Prototyping, pag 119125, Crete.

Cheung-Foo-Wo, D., Tigli, J. Y., Lavirotte, S. ve Riveill, M., 2007. Self-adaptation of event- driven component-oriented Middleware using Aspects of Assembly, In 5th International Workshop on Midd leware for Pervasive and Ad-Hoc Computing (MPAC), California, USA, Nov.

Christophi, C., 2003. Introduction to Web Services, University of Cyprus.

Cohen, W., Ravikumar, P., ve Fienberg, S., 2003. A comparison of string distance metrics for name-matching tasks. In Proc. IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03). DBLP at http://dblp.uni-trier.de.

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

Cömert, Ç. ve Akıncı, H., 2005. Ulusal Konumsal Veri Altyapısı ve e-Türkiye için önemi, TMMOB Harita ve Kadastro Mühendisleri Odası 10. Türkiye Harita Bilimsel ve Teknik Kurultayı, Mart - Nisan, Ankara, 1-9.

Cömert, Ç., Ulutaş, D., Akıncı, H. ve Kara, G., 2010. Semantic web services for implementing national spatial data infrastructures, Scientific Research and Essays, 5-7, 4, 685-692.

Cruz, I. F., Sunna, W. G. ve Ayloo, K., 2005. Concept Level Matching of Geospatial Ontologies, In GIS Planet Second Conference and Exhibition on Geographic Information.

Di Noia, T., Di Sciascio, E. ve Donini, F. M., 2009. A Tableauxbased Calculus for Abduction in Expressive Description Logics: Preliminary Results. Proceedings of 22nd International Workshop on Description Logics (DL).

Doan, A., Madhavan, J., Domingos, P. ve Halevy, A., 2003. Learning to map ontologies on the semantic web. In Proceedings of the International World Wide Web Conference (WWW) pages 662–673.

Dong, J., Sun, Y., Yang, S. ve Zhang, K., 2006. Dynamic web service composition based on OWL-S, Science in China Series F: Information Sciences, 49-6, 843-863.

Erol, K., 1996. Hierarchical task network planning: formalization, analysis, and implementation. PhD thesis, University of Maryland at College Park, College Park, MD, USA, UMI Order No. GAX96-22054.

Fensel, D., Lausen, H., Polleres, A., de Bruijn, J., Stollberg, M., Roman, D. ve Domingue, J., 2007. The Concepts of WSMO. Enabling Semantic Web Services, The Web Service Modeling Ontology book, chapter 6, XIV, 188p, Hardcover

Gangemi, A., 2005. Ontology Design Patterns for Semantic Web Content, ISWC 2005, LNCS 3729, 262–276, Springer-Verlag Berlin Heidelberg.

Gruber, T., 1993. A Translation Approach to Portable Ontology Specifications, Knowledge Acquisition, 5,2, 199-220.

Gruber, T., 1995. Toward Principles for The Design of Ontologies Used for Knowledge Sharing, International Journal of Human-Computer Studies, 43, 5-6, 907-928.

Giunchiglia, F., Shvaiko, P. ve Yatskevich, M., 2004. S-MATCH: An Algorithm and An Implementation of Semantic Matching, Technical report # DIT-04-015, February

Giunchiglia, F., Shvaiko, P. ve Yatskevich, M., 2005. Semantic Schema Matching. In Proceedings of CoopIS, 347-365.

Giunchiglia, F., Vincenzo, M., Farazi, F. ve Dutta, B., 2009. GEOWordNet: A Resource For Geo-Spatial Applications, Technical report # DISI-09-071, Trento (Italy).

Hochmair, H., H. ve Fu, J., 2006. User Interface Design for Semantic Query Expansion in GeodataRepositories, In 18. AGIT-Symposium, Salzburg: Heidelberg, Wichmann.

Hu, Y., Janowicz, K., Carral, D., Scheider, S., Kuhn, W., Berg-Cross, G., Hitzler, P., Dean, M. ve Kolas, D., 2013. A Geo-ontology Design Pattern for Semantic Trajectories, COSIT 2013, LNCS 8116, Springer International Publishing Switzerland, 438–456.

Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, R. M., Li, P. ve Oinn, T., 2006. Taverna: a tool for building and running workflows of services. Nucleic Acids Research, 34, 729-732.

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

Ivanova, T., 2010. A Semantic Ontology Alignment Method, Conference ICL2010, September 15-17, Hasselt, Belgium.

Kara, G., Ulutaş, D. ve Cömert, Ç., 2012. Semantic Definition and Matching for National Spatial Data Infrastructure, In INSPIRE Conference, Istanbul, Turkey.

Kietz, J. U., Serban, F., Fischer, S. ve Bernstein, A., 2014. "Semantics Inside!" But let's not tell the Data Miners: Intelligent Support for Data Mining, Lecture Notes in Computer Science, 8465, 706-720.

Kietz, J., Serban, F. ve Bernstein, A., 2012. Designing kdd-workflows via htn-planning. In J. Vanschoren, P. Brazdil, and J.-U. Kietz, editors, 4rd Planning to Learn Workshop at ECAI 2012, 950 of CEUR Workshop Proceedings.

Klien, E., Lutz, M. and Kuhn, W., 2006. Ontology-Based Discovery of Geographic Information Services - An Application in Disaster Management. Computers, Environment and UrbanSystems, 30, 1, 102-123.

Klien, E., 2008. Semantic Annotation of Geographic Information, PhD Thesis, University of Münster, Institute for Geoinformatics, Münster, Germany.

Klusch, M. ve Gerber, A., 2006. Fast Composition Planning of OWL-S Services and Application, Proceedings of the European Conference on Web Services (ECOWS'06).

Klusch, M., Fries, B. ve Sycara, K., 2006. Automated Semantic Web Service Discovery with OWLS-MX. In: AAMAS. Proceedings of 5th International Conference on AutonomousAgents and Multi-Agent Systems, Hakodate, Japan, ACM Press, New York.

Klusch, M. ve Kapahnke, P., 2012. The iSeM Matchmaker: A Flexible Approach For Adaptive Hybrid Semantic Service Selection. Web Semantics: Science, Services and Agents on the World Wide Web, Elsevier, 15, 1-14.

Knublauch, H., Fergerson, R., Noy, N. ve Musen, M., 2004. The Protégé-OWL Plugin: An Open Development Environment for Semantic Web Applications, Third International Semantic Web Conference (ISWC 2004), Hiroshima, Japan, Bildiriler Kitabı: 3298, 229-243.

Kruskal, J. B., 1956. On the shortest spanning subtree of a graph and the traveling salesman problem, Proc. Amer. Math. Soc., 7.

Lemmens, R.L.G., 2006. Semantic Interoperability of Distributed Geoservices, PhD Thesis, ITC Dissertation, Delft University of Technology, Delft, The Netherlands.

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

Lutz, M., 2005. Ontology-Based Service Discovery in Spatial Data Infrastructures, GIR'05, 4 Kasım, Bremen, Germany.

Madhavan, J., Bernstein, P. ve Rahm, E., 2001. Generic schema matching with Cupid. In Proceedings of the Very Large Data Bases Conference (VLDB), 49–58.

Martínez, D. C., Janowicz, K. ve Hitzler, P., 2012. A Logical Geo-Ontology Design Pattern for Quantifying over Types, ACM GIS 2012 Redondo Beach, CA, USA.

Maynard, D., Funk, A. ve Peters, W., 2009. SPRAT: a tool for automatic semantic pattern-based ontology population, In International conference for digital libraries and the semantic web, Trento, Italy.

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

Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M. ve Euler, T., 2006. Yale: Rapid prototyping for complex data mining tasks. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, 935-940.ACM.

Miller, G. A., 1990. WordNet: An on-line lexical database. International Journal of Lexicography, 3, 4, 235–312. Special Issue.

Murata, T., 1989. Petri Nets: Properties, Analysis and Applications, Proceedings of the IEEE, 77, 4, 541–580.

Nathalie, A., 2009. Schema Matching Based on Attribute Values and Background Ontology, 12th AGILE International Conference on Geographic Information Science, Leibniz Universitat Hannover, Germany.

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

OGC, 2014. Filter Encoding 2.0 Encoding Standard, OGC 09-026r2, version 2.0.2, OGC Implementation Specification.

OGC, 2007. Web Processing Service (WPS) Implementation Specification, Version 1.0.0, OGC 05-007r7, OGC Implementation Specification.

Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M. R. ve Wipat, A., 2004. Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics, 20, 17, 3045-3054.

Partyka, J., Parveen, P., Khan, L., Thuraisingham, B. ve Shekhar, S., 2010. Enhanced geographically typed semantic schema matching, Web Semantics: Science, Services and Agents on the World Wide Web, 9, 1, 52-70.

Peer, J., Web Service Composition as AI Planning: A Survey. Technical Report, University of St. Gallen, Switzerland http://elektra.mcm.unisg.ch/pbwsc/docs/pfwsc.pdf, 20.03.2015.

Peterson, J. L., 1981. Petri Net Theory and the Modeling of Systems. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Pop, F. C., Cremene, M., Riveill, M. ve Vaida, M., 2009. On-demand service composition based on natural language requests, In: Wireless On-Demand Network Systems and Services, WONS 2009, Sixth International Conference on. IEEE, 45-48.

Randell, D. A., Cui, Z. ve Cohn, A. G., 1992. A spatial logic based on regions and connections. In:Proceedings of the international conference on principles of knowledge representation and reasoning (KR'92), Cambridge, MA: Morgan Kaufmann Publishers.

Renier, G. F. ve Gilberto, R., 2005. GeoDiscover – a Specialized Search Engine to Discover Geospatial Data in the Web. In: 7th Brazilian Symposium on GeoInformatics.

Sbodio, M. L., Martin, D. ve Moulin, C., 2010. Discovering Semantic Web services using SPARQL and intelligent agents, Web Semantics: Science, Services and Agents on the World Wide Web, 8, 4, 310-328.

Schade, S., 2009. Ontology-Driven Translation of Geospatial Data, PhD Thesis, Institute for GeoInformatics, University of Münster, April, Münster, Germany.

Serban, F., 2013. Towards Effective Support for Data Mining using Intelligent Discovery Assistance. PhD thesis, University of Zurich, Department of Informatics.

Shamdasani, J., Bloodsworth, P., Munir, K., Rahmouni, H. B. ve McClatchey, R., 2011. MedMatch-Towards Domain Specific Semantic Matching, S. Andrews et al. (Eds.) ICCS 2011, LNAI 6828, Springer-Verlag Berlin Heidelberg, 375–382.

Shvaiko, P. ve Euzenat, J., 2005. A Survey of Schema-based Matching Approaches, Journal on Data Semantic, 4, 146-171.

Srinivasan, N., Paolucci, M. ve Sycara, K., 2004. An efficient algorithm for OWL-S based semantic search in UDDI. In: Proceedings of the 1st International Workshop on Semantic Web Services and Web Process Composition (SWSWPC) at the 2nd International Conference on Web Services (ICWS), San Diego, CA, USA, July, 96-110.

The BioMoby Consortium, 2008. Interoperability with Moby 1.0 - It's better than sharing your toothbrush! Briefings in Bioinformatics, 9, 3, 220-231.

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

URL-1, http://franz.com/agraph/racer/ RacerPro. 03.04.2015.

URL-2, http://www.eclipse.org/eclipse. 10.01.2015.

URL-3, http://www.iso.org/iso/home.html/ ISO. 03.04.2015.

URL-4, http://www.opengeospatial.org/ OGC. 02.02.2015.

URL-5, http://www.ksl.stanford.edu/ Stanford K S L. JTP. 03.01.2015.

URL-6, http://wordnet.princeton.edu/ What is WordNet?. 22.12.2011.

URL-7, http://www.perl.org/ The Perl Programming Language, 04.01.2015.

URL-8, http://www.w3.org/Submission/WSDL-S/, 03.03.2015.

URL-9, http://www.w3.org/Submission/WSML/, 23.01.2015.

URL-10, http://www.w3.org/Submission/OWL-S/, 04.01.2015.

URL-11, http://www.w3.org/TR/wsdl, 04.01.2015.

URL-12, http://owlsm.projects.semwebcentral.org/, 21 Şubat 2008.

URL-13, http://www.daml.org/services/owl-s/, 01.04.2015.

URL-14, http://www.w3.org/TR/rdf-sparql-query/, 03.02.2015.

URL-15, http://jena.sourceforge.net/ARQ/, 03.02.2015

URL-16, http://www.support-vector-machines.org/,03.02.2015

URL-17, http://www.wsmx.org/, 03.02.2015

URL-18, http://www.sri.com/, 03.02.2015

URL-19, http://protege.stanford.edu/, 04.02.2015

URL-20, http://www.w3.org/Submission/SWRL/, 04.02.2015

URL-21, http://owl-scomposer.sourceforge.net/, 04.02.2015

URL-22, https://eclipse.org/ide/, 04.02.2015

URL-23, http://en.wikipedia.org/wiki/Planning Domain Definition Language, 05.02.2015

URL-24, http://www.mygrid.org.uk/, 07.02.2015

URL-25, http://www.biovel.eu/, 07.02.2015

URL-26, http://www.scape-project.eu/, 07.02.2015

URL-27, http://www.wf4ever-project.org/, 07.02.2015.

URL-28, http://www.taverna.org.uk/, 07.02.2015.

URL-29, http://en.wikipedia.org/wiki/Representational state transfer, 07.02.2015.

URL-30, http://www.biomart.org/, 07.02.2015

URL-31, http://dev.mygrid.org.uk/wiki/display/tav250/BioMoby, 07.02.2015

URL-32, http://soaplab.sourceforge.net/soaplab1/, 07.02.2015

URL-33, http://www.r-project.org/, 07.02.2015

URL-34, http://www.beanshell.org/, 07.02.2015

URL-35, http://dev.mygrid.org.uk/wiki/display/tav250/API+Consumer, 07.02.2015

URL-36, https://www.biocatalogue.org/, 07.02.2015

URL-37, http://www.myexperiment.org/, 07.02.2015

URL-38, http://www.geonames.org/ , 23.12.2011

URL-39, http://semanticmatching.org/s-match.html, 20.12.2011

URL-40, http://wordnet.princeton.edu/, 22.12.2011

URL-41, http://njames.trevize.net/wiki/projects:wnxplorer, 22.12.2011

URL-42, http://wnsql.sourceforge.net/, 26.03.2015

URL-43, http://multiwordnet.fbk.eu/english/creation.php, 16 Mart 2015

URL-44, http://s-match.org/background-knowledge-datasets.html, 23.12.2011

URL-45, http://www.hgk.msb.gov.tr/ Harita Genel Komutanlığı (HGK), 03.04.2015

URL-46, http://inspire.jrc.ec.europa.eu/index.cfm/pageid/541/downloadid/1698 INSPIRE. 06.06.2011.

URL-47, http://www.loa.istc.cnr.it/DOLCE.html DOLCE: a Descriptive Ontology for Linguistic and Cognitive Engineering. 01.05.2013.

URL-48, www.ontologyportal.org/index.html SUMO: Suggested Upper Merged Ontology. 09.10.2012.

URL-49, http://www.cyc.com/platform/opencyc OpenCyc. 01.05.2013.

URL-50, http://extjwnl.sourceforge.net/ extJWNL. 21.10.2012

URL-51, http://protege.stanford.edu/overview/ What is Protégé? 10.01.2008.

URL-52, http://www.nactem.ac.uk/software/termine/webservice/, 10.03.2015

URL-53, http://www.e-lico.eu/text-mining-ws.html, 10.03.2015

URL-54, http://protegewiki.stanford.edu/wiki/OntoGraf, 15.03.2015

URL-55, http://protegewiki.stanford.edu/wiki/OWLViz, 15.03.2015

URL-56, http://www.esri.com/software/arcgis, 03.04.2015

URL-57, http://grass.osgeo.org/, 03.04.2015

URL-58, http://www.topquadrant.com/tools/IDE-topbraid-composer-maestro-edition/, TopBraid Composer. 25.03.2015

URL-59, http://52north.org/communities/geoprocessing/wps/, 01.03.2015

URL-60, http://www.deegree.org/, 01.03.2015

URL-61, http://docs.geoserver.org/stable/en/user/extensions/wps/index.html, 01.03.2015

URL-62, http://pywps.wald.intevation.org/, 01.03.2015

URL-63, https://www.python.org/, 01.03.2015

URL-64, https://www.dbu.de/, 01.03.2015

URL-65, http://www.bnhelp.cz/, 01.03.2015

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

Wilkinson, M. D., Vandervalk, B. ve McCarthy, L., 2009. SADI Semantic Web Services-cause you can't always GET what you want! Services Computing Conference, APSCC 2009. IEEE Asia-Pacific, 13-18.

Withers, D., Kawas, E., McCarthy, L., Vandervalk, B. ve Wilkinson, M., 2010. Semantically-guided Workflow Construction in Taverna: The SADI and BioMoby Plug-ins, In Leveraging Applications of Formal Methods, Verification, and Validation, Springer Berlin Heidelberg, 301-312.

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

W3C, OWL-S: Semantic Markup for Web Services, W3C Member Submission, http://www.w3.org/Submission/OWL-S/, 15 Şubat 2008.

Zhang, C., Zhao, T., Li, W. ve Osleeb, J. P., 2010. Towards logic-based geospatial feature discovery and integration using web feature service and geospatial semantic web, International Journal of Geographical Information Science, 24, 6, 903-923, DOI:10.1080/13658810903240687 2010.

6. EKLER

Ek 1

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:service="http://www.daml.org/services/owl-s/1.2/Service.owl#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:time="http://www.isi.edu/~pan/damltime/time-entry.owl#"
    xmlns:list="http://www.daml.org/services/owl-s/1.2/generic/ObjectList.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:expr="http://www.daml.org/services/owl-s/1.2/generic/Expression.owl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:profile="http://www.daml.org/services/owl-s/1.2/Profile.owl#"
    xmlns:process="http://www.daml.org/services/owl-s/1.2/Process.owl#"
    xmlns:grounding="http://www.daml.org/services/owl-s/1.2/Grounding.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns="http://www.owl-ontologies.com/Ontology1405895843.owl#"
  xml:base="http://www.owl-ontologies.com/Ontology1405895843.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.w3.org/2003/11/swrl"/>
    <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.2/Grounding.owl"/>
    <owl:imports rdf:resource="http://www.daml.org/services/owl-s/1.2/Profile.owl"/>
    <owl:imports rdf:resource="http://www.w3.org/2003/11/swrlb"/>
  </owl:Ontology>
  <owl:Class rdf:ID="Katman_ismi">
    <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl#Input"/>
  </owl:Class>
  <owl:Class rdf:ID="BboxKoordinatlari">
    <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl#Output"/>
  </owl:Class>
  <owl:Class rdf:ID="SiteMap">
    <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl#Output"/>
  </owl:Class>
  <owl:Class rdf:ID="Katman">
    <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl#Output"/>
  </owl:Class>
  <owl:Class rdf:ID="Koordinat">
    <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl#Input"/>
  </owl:Class>
  <owl:Class rdf:ID="Coordinates">
    <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl#Output"/>
  </owl:Class>
  <owl:Class rdf:ID="SecilenDetaylar">
    <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl#Output"/>
  </owl:Class>
  <owl:Class rdf:ID="NumberOfLayers">
    <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl#Local"/>
  </owl:Class>
  <owl:Class rdf:ID="Detay">
    <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl#Input"/>
  </owl:Class>
```

```
  <owl:Class rdf:ID="Yer_ismi">
    <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl#Input"/>
  </owl:Class>
  <owl:Class rdf:ID="IntersectLayer">
    <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl#Output"/>
  </owl:Class>
  <owl:Class rdf:ID="BoundingBoxKoordinatlari">
    <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl#Input"/>
  </owl:Class>
  <owl:Class rdf:ID="SecimKosulu">
    <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl#Input"/>
  </owl:Class>
  <owl:Class rdf:ID="TamponMesafe">
    <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl#Input"/>
  </owl:Class>
  <owl:Class rdf:ID="KesisimKatmani">
    <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl#Input"/>
  </owl:Class>
  <owl:Class rdf:ID="Katman2">
    <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl#Input"/>
  </owl:Class>
  <owl:Class rdf:ID="AttributeValue">
    <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl#Input"/>
  </owl:Class>
  <owl:Class rdf:ID="TamponluKatman">
    <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl#Output"/>
  </owl:Class>
  <owl:Class rdf:ID="GirdiKatman">
    <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl#Input"/>
  </owl:Class>
  <owl:Class rdf:ID="Katman1">
    <rdfs:subClassOf rdf:resource="http://www.daml.org/services/owl-s/1.2/Process.owl#Input"/>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="HasProjectionSystem"/>
  <SiteMap rdf:ID="Harita"/>
  <process:ValueOf rdf:ID="ValueOf_145">
    <process:fromProcess rdf:resource="http://www.daml.org/services/owl-
s/1.2/Process.owl#TheParentPerform"/>
  </process:ValueOf>
  <process:Repeat-Until rdf:ID="Repeat-Until_SelectFeatures">
    <process:untilProcess>
      <process:Perform rdf:ID="Perform_SelectFeaturesProcess">
        <process:process>
          <process:AtomicProcess rdf:ID="SelectFeaturesProcess">
            <process:hasInput>
              <Detay rdf:ID="Detay_12"/>
            </process:hasInput>
            <process:hasInput>
              <SecimKosulu rdf:ID="SecimKosulu_13">
                <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
                >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
              </SecimKosulu>
            </process:hasInput>
            <process:hasOutput>
              <SecilenDetaylar rdf:ID="SecilmisDetaylar"/>
            </process:hasOutput>
          </process:AtomicProcess>
        </process:process>
        <process:hasDataFrom>
```

```
        <process:InputBinding rdf:ID="InputBinding_30">
          <process:toParam rdf:resource="#Detay_12"/>
          <process:valueSource>
            <process:ValueOf rdf:ID="ValueOf_31">
              <process:theVar>
                <Katman rdf:ID="Katman_31"/>
              </process:theVar>
              <process:fromProcess>
                <process:Perform rdf:ID="Perform_KatmanGetirProcess">
                  <process:hasDataFrom>
                    <process:InputBinding rdf:ID="InputBinding_11">
                      <process:toParam>
                        <BoundingBoxKoordinatlari rdf:ID="BoundingBox">
                          <process:parameterType rdf:datatype=
                          "http://www.w3.org/2001/XMLSchema#anyURI"
                          >http://www.w3.org/2001/XMLSchema#int</process:parameterType>
                        </BoundingBoxKoordinatlari>
                      </process:toParam>
                      <process:valueSource>
                        <process:ValueOf rdf:ID="ValueOf_12">
                          <process:theVar>
                            <BboxKoordinatlari rdf:ID="BboxKoordinatlari_75">
                              <process:parameterType rdf:datatype=
                              "http://www.w3.org/2001/XMLSchema#anyURI"
                              >http://www.w3.org/2001/XMLSchema#int</process:parameterType>
                            </BboxKoordinatlari>
                          </process:theVar>
                          <process:fromProcess>
                            <process:Perform rdf:ID="Perform_BboxOLusturProcess">
                              <process:process>
                                <process:AtomicProcess rdf:ID="BboxOlusturProcess">
                                  <process:hasOutput rdf:resource="#BboxKoordinatlari_75"/>
                                  <process:hasInput>
                                    <Koordinat rdf:ID="CografiKoordinat">
                                      <process:parameterType
                                       rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
>http://www.w3.org/2001/XMLSchema#int</process:parameterType>
                                    </Koordinat>
                                  </process:hasInput>
                                </process:AtomicProcess>
                              </process:process>
                              <process:hasDataFrom>
                                <process:InputBinding rdf:ID="InputBinding_9">
                                  <process:valueSource>
                                    <process:ValueOf rdf:ID="ValueOf_10">
                                      <process:theVar>
                                        <Yer_ismi rdf:ID="YerIsmi_80">
                                          <process:parameterType
                                          rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
>http://www.w3.org/2001/XMLSchema#string</process:parameterType>
                                        </Yer_ismi>
                                      </process:theVar>
                                      <process:fromProcess rdf:resource="http://www.daml.org/services/owl-
s/1.2/Process.owl#TheParentPerform"/>
                                    </process:ValueOf>
                                  </process:valueSource>
                                  <process:toParam rdf:resource="#CografiKoordinat"/>
                                </process:InputBinding>
                              </process:hasDataFrom>
```

```
                    </process:Perform>
                   </process:fromProcess>
                  </process:ValueOf>
                 </process:valueSource>
                </process:InputBinding>
               </process:hasDataFrom>
               <process:process>
                <process:AtomicProcess rdf:ID="KatmanGetirProcess">
                 <process:hasOutput rdf:resource="#Katman_31"/>
                 <process:hasInput>
                  <Katman_ismi rdf:ID="Detay_ismi_28">
                   <process:parameterType rdf:datatype=
                    "http://www.w3.org/2001/XMLSchema#anyURI"
                   >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
                  </Katman_ismi>
                 </process:hasInput>
                 <process:hasInput rdf:resource="#BoundingBox"/>
                </process:AtomicProcess>
               </process:process>
              </process:Perform>
             </process:fromProcess>
            </process:ValueOf>
           </process:valueSource>
          </process:InputBinding>
         </process:hasDataFrom>
        </process:Perform>
       </process:untilProcess>
       <process:untilCondition>
        <expr:SWRL-Condition rdf:ID="SWRL-Condition">
         <expr:expressionObject>
          <swrl:AtomList>
           <rdf:rest>
            <swrl:AtomList>
             <rdf:first>
              <swrl:BuiltinAtom>
               <swrl:builtin rdf:resource="http://www.w3.org/2003/11/swrlb#equal"/>
               <swrl:arguments>
                <rdf:List>
                 <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                  >0</rdf:first>
                 <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                </rdf:List>
               </swrl:arguments>
              </swrl:BuiltinAtom>
             </rdf:first>
             <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
            </swrl:AtomList>
           </rdf:rest>
           <rdf:first>
            <swrl:ClassAtom>
             <swrl:argument1>
              <swrl:Variable rdf:ID="x"/>
             </swrl:argument1>
             <swrl:classPredicate rdf:resource="#NumberOfLayers"/>
```

Ek 2

<ExecuteProcess_v.overlayResponse><outputResult><ogr:FeatureCollection
xmlns:gml="http://www.opengis.net/gml" xmlns:ogr="http://ogr.maptools.org/"
xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ogr.maptools.org/ output_0C8ZXrY.xsd">
 <gml:boundedBy>
  <gml:Box>

<gml:coord><gml:X>534308.0980555454</gml:X><gml:Y>4422528.950974972</gml:Y><gml:Z>0</gml:Z></gml:coord>

<gml:coord><gml:X>542280.6</gml:X><gml:Y>4426788.273112839</gml:Y><gml:Z>0</gml:Z></gml:coord>
  </gml:Box>
 </gml:boundedBy>

 <gml:featureMember>
  <ogr:output fid="output.0">

<ogr:geometryProperty><gml:Polygon><gml:outerBoundaryIs><gml:LinearRing><gml:coordinates>536729.9,4426641.57,0 536742.85,4426579.71,0 536737.39,4426571.22,0 536724.84,4426557.09,0
536705.45,4426603.91,0
536729.9,4426641.57,0</gml:coordinates></gml:LinearRing></gml:outerBoundaryIs></gml:Polygon></ogr:geometryProperty>
    <ogr:cat>1</ogr:cat>
    <ogr:a_cat>4</ogr:a_cat>
    <ogr:a_fid>output.3</ogr:a_fid>
    <ogr:a_cat_>4</ogr:a_cat_>
    <ogr:a_a_cat>1283</ogr:a_a_cat>
    <ogr:a_a_fid>output.1231</ogr:a_a_fid>
    <ogr:a_a_cat_>1283</ogr:a_a_cat_>
    <ogr:a_a_a_cat>3493</ogr:a_a_a_cat>
    <ogr:a_a_a_fid>slope_clip2.5557</ogr:a_a_a_fid>
    <ogr:a_a_a_cat_>5558</ogr:a_a_a_cat_>
    <ogr:a_a_a_SlopeCode>5</ogr:a_a_a_SlopeCode>
    <ogr:a_b_cat>227</ogr:a_b_cat>
    <ogr:a_b_fid>output.226</ogr:a_b_fid>
    <ogr:a_b_cat_>227</ogr:a_b_cat_>
    <ogr:a_b_a_cat>3819</ogr:a_b_a_cat>
    <ogr:a_b_a_fid>aspect_clip2.8963</ogr:a_b_a_fid>
    <ogr:a_b_a_cat_>8964</ogr:a_b_a_cat_>
    <ogr:a_b_a_AspectCode>8</ogr:a_b_a_AspectCode>
    <ogr:b_cat>64</ogr:b_cat>
    <ogr:b_fid>output.63</ogr:b_fid>
    <ogr:b_cat_>64</ogr:b_cat_>
    <ogr:b_a_cat>11</ogr:b_a_cat>
    <ogr:b_a_fid>output.10</ogr:b_a_fid>
    <ogr:b_a_cat_>11</ogr:b_a_cat_>
    <ogr:b_a_a_cat>13</ogr:b_a_a_cat>
    <ogr:b_a_a_fid>toprak_clip2.12</ogr:b_a_a_fid>
    <ogr:b_a_a_cat_>13</ogr:b_a_a_cat_>
    <ogr:b_a_a_BTG>B</ogr:b_a_a_BTG>
    <ogr:b_a_a_TOK>24</ogr:b_a_a_TOK>
    <ogr:b_a_a_ERZ>3</ogr:b_a_a_ERZ>
    <ogr:b_a_a_SAK>M</ogr:b_a_a_SAK>
    <ogr:b_a_a_SAK2>M</ogr:b_a_a_SAK2>

```
    <ogr:b_a_a_AKK>VII</ogr:b_a_a_AKK>
    <ogr:b_a_a_ATS>es</ogr:b_a_a_ATS>
    <ogr:b_a_a_BTG_TOK>B_24</ogr:b_a_a_BTG_TOK>
    <ogr:b_a_a_EGM>6</ogr:b_a_a_EGM>
    <ogr:b_a_a_DER>D</ogr:b_a_a_DER>
    <ogr:b_b_cat>33</ogr:b_b_cat>
    <ogr:b_b_fid>output.32</ogr:b_b_fid>
    <ogr:b_b_cat_>33</ogr:b_b_cat_>
    <ogr:b_b_a_cat>11</ogr:b_b_a_cat>
    <ogr:b_b_a_fid>corine_clip2.43</ogr:b_b_a_fid>
    <ogr:b_b_a_cat_>44</ogr:b_b_a_cat_>
    <ogr:b_b_a_Code2006>321</ogr:b_b_a_Code2006>
    <ogr:b_b_a_SHAPE_Leng>86978.6341462</ogr:b_b_a_SHAPE_Leng>
    <ogr:b_b_a_SHAPE_Area>57278248.0056</ogr:b_b_a_SHAPE_Area>
    <ogr:b_a_a_DTO>r</ogr:b_a_a_DTO>
    <ogr:b_a_a_AZT/>
   </ogr:output>
  </gml:featureMember>
  <gml:featureMember>
   <ogr:output fid="output.1">

<ogr:geometryProperty><gml:Polygon><gml:outerBoundaryIs><gml:LinearRing><gml:coordinates>53729
4.83,4426459.12,0 537267.48,4426436.75,0 537217.2,4426466.77,0 537164.18,4426496.7,0
537157.800146008492447,4426582.579322713427246,0 537124.54,4426620.84,0
537155.090073004132137,4426619.059661356732249,0 537130.29,4426625.71,0 537131.31,4426634.04,0
537126.33,4426672.76,0 537128.383728838991374,4426726.651208209805191,0
537227.973902385216206,4426724.246702435426414,0 537231.99,4426717.35,0
537234.401597830234095,4426724.091512115672231,0
537270.989179318072274,4426723.208141319453716,0 537275.17,4426721.4,0 537266.87,4426699.74,0
537309.57,4426681.48,0 537332.81,4426657.69,0 537377.91,4426649.56,0 537417.65,4426645.45,0
537459.13,4426638.57,0 537457.26,4426615.11,0 537462.99,4426596.02,0 537421.57,4426576.41,0
537395.73,4426547.19,0 537365.46,4426525.0,0 537386.42,4426570.45,0 537346.6,4426568.01,0
537319.48,4426513.11,0
537294.83,4426459.12,0</gml:coordinates></gml:LinearRing></gml:outerBoundaryIs></gml:Polygon></o
gr:geometryProperty>
    <ogr:cat>2</ogr:cat>
    <ogr:a_cat>33</ogr:a_cat>
    <ogr:a_fid>output.32</ogr:a_fid>
    <ogr:a_cat_>33</ogr:a_cat_>
    <ogr:a_a_cat>129</ogr:a_a_cat>
    <ogr:a_a_fid>output.128</ogr:a_a_fid>
    <ogr:a_a_cat_>129</ogr:a_a_cat_>
    <ogr:a_a_a_cat>83</ogr:a_a_a_cat>
    <ogr:a_a_a_fid>slope_clip2.3</ogr:a_a_a_fid>
    <ogr:a_a_a_cat_>4</ogr:a_a_a_cat_>
    <ogr:a_a_a_SlopeCode>6</ogr:a_a_a_SlopeCode>
    <ogr:a_b_cat>282</ogr:a_b_cat>
    <ogr:a_b_fid>output.281</ogr:a_b_fid>
    <ogr:a_b_cat_>282</ogr:a_b_cat_>
    <ogr:a_b_a_cat>135</ogr:a_b_a_cat>
    <ogr:a_b_a_fid>aspect_clip2.374</ogr:a_b_a_fid>
    <ogr:a_b_a_cat_>375</ogr:a_b_a_cat_>
    <ogr:a_b_a_AspectCode>7</ogr:a_b_a_AspectCode>
    <ogr:b_cat>64</ogr:b_cat>
    <ogr:b_fid>output.63</ogr:b_fid>
    <ogr:b_cat_>64</ogr:b_cat_>
    <ogr:b_a_cat>11</ogr:b_a_cat>
    <ogr:b_a_fid>output.10</ogr:b_a_fid>
    <ogr:b_a_cat_>11</ogr:b_a_cat_>
```

```
    <ogr:b_a_a_cat>13</ogr:b_a_a_cat>
    <ogr:b_a_a_fid>toprak_clip2.12</ogr:b_a_a_fid>
    <ogr:b_a_a_cat_>13</ogr:b_a_a_cat_>
    <ogr:b_a_a_BTG>B</ogr:b_a_a_BTG>
    <ogr:b_a_a_TOK>24</ogr:b_a_a_TOK>
    <ogr:b_a_a_ERZ>3</ogr:b_a_a_ERZ>
    <ogr:b_a_a_SAK>M</ogr:b_a_a_SAK>
    <ogr:b_a_a_SAK2>M</ogr:b_a_a_SAK2>
    <ogr:b_a_a_AKK>VII</ogr:b_a_a_AKK>
    <ogr:b_a_a_ATS>es</ogr:b_a_a_ATS>
    <ogr:b_a_a_BTG_TOK>B_24</ogr:b_a_a_BTG_TOK>
    <ogr:b_a_a_EGM>6</ogr:b_a_a_EGM>
    <ogr:b_a_a_DER>D</ogr:b_a_a_DER>
    <ogr:b_b_cat>33</ogr:b_b_cat>
    <ogr:b_b_fid>output.32</ogr:b_b_fid>
    <ogr:b_b_cat_>33</ogr:b_b_cat_>
    <ogr:b_b_a_cat>11</ogr:b_b_a_cat>
    <ogr:b_b_a_fid>corine_clip2.43</ogr:b_b_a_fid>
    <ogr:b_b_a_cat_>44</ogr:b_b_a_cat_>
    <ogr:b_b_a_Code2006>321</ogr:b_b_a_Code2006>
    <ogr:b_b_a_SHAPE_Leng>86978.6341462</ogr:b_b_a_SHAPE_Leng>
    <ogr:b_b_a_SHAPE_Area>57278248.0056</ogr:b_b_a_SHAPE_Area>
    <ogr:b_a_a_DTO>r</ogr:b_a_a_DTO>
    <ogr:b_a_a_AZT/>
   </ogr:output>
  </gml:featureMember>
  <gml:featureMember>
   <ogr:output fid="output.2">

<ogr:geometryProperty><gml:Polygon><gml:outerBoundaryIs><gml:LinearRing><gml:coordinates>53659
0.47,4426663.72,0 536616.72,4426670.77,0 536620.98,4426634.58,0 536629.97,4426603.96,0
536630.18,4426574.92,0 536637.05,4426555.33,0 536608.19,4426548.78,0 536589.67,4426597.45,0
536590.47,4426663.72,0</gml:coordinates></gml:LinearRing></gml:outerBoundaryIs></gml:Polygon></o
gr:geometryProperty>
    <ogr:cat>3</ogr:cat>
    <ogr:a_cat>41</ogr:a_cat>
    <ogr:a_fid>output.40</ogr:a_fid>
    <ogr:a_cat_>41</ogr:a_cat_>
    <ogr:a_a_cat>217</ogr:a_a_cat>
    <ogr:a_a_fid>output.214</ogr:a_a_fid>
    <ogr:a_a_cat_>217</ogr:a_a_cat_>
    <ogr:a_a_a_cat>289</ogr:a_a_a_cat>
    <ogr:a_a_a_fid>slope_clip2.302</ogr:a_a_a_fid>
    <ogr:a_a_a_cat_>303</ogr:a_a_a_cat_>
    <ogr:a_a_a_SlopeCode>5</ogr:a_a_a_SlopeCode>
    <ogr:a_b_cat>22</ogr:a_b_cat>
    <ogr:a_b_fid>output.21</ogr:a_b_fid>
    <ogr:a_b_cat_>22</ogr:a_b_cat_>
    <ogr:a_b_a_cat>116</ogr:a_b_a_cat>
    <ogr:a_b_a_fid>aspect_clip2.330</ogr:a_b_a_fid>
    <ogr:a_b_a_cat_>331</ogr:a_b_a_cat_>
    <ogr:a_b_a_AspectCode>7</ogr:a_b_a_AspectCode>
    <ogr:b_cat>64</ogr:b_cat>
    <ogr:b_fid>output.63</ogr:b_fid>
    <ogr:b_cat_>64</ogr:b_cat_>
    <ogr:b_a_cat>11</ogr:b_a_cat>
    <ogr:b_a_fid>output.10</ogr:b_a_fid>
    <ogr:b_a_cat_>11</ogr:b_a_cat_>
    <ogr:b_a_a_cat>13</ogr:b_a_a_cat>
```

```
    <ogr:b_a_a_fid>toprak_clip2.12</ogr:b_a_a_fid>
    <ogr:b_a_a_cat_>13</ogr:b_a_a_cat_>
    <ogr:b_a_a_BTG>B</ogr:b_a_a_BTG>
    <ogr:b_a_a_TOK>24</ogr:b_a_a_TOK>
    <ogr:b_a_a_ERZ>3</ogr:b_a_a_ERZ>
    <ogr:b_a_a_SAK>M</ogr:b_a_a_SAK>
    <ogr:b_a_a_SAK2>M</ogr:b_a_a_SAK2>
    <ogr:b_a_a_AKK>VII</ogr:b_a_a_AKK>
    <ogr:b_a_a_ATS>es</ogr:b_a_a_ATS>
    <ogr:b_a_a_BTG_TOK>B_24</ogr:b_a_a_BTG_TOK>
    <ogr:b_a_a_EGM>6</ogr:b_a_a_EGM>
    <ogr:b_a_a_DER>D</ogr:b_a_a_DER>
    <ogr:b_b_cat>33</ogr:b_b_cat>
    <ogr:b_b_fid>output.32</ogr:b_b_fid>
    <ogr:b_b_cat_>33</ogr:b_b_cat_>
    <ogr:b_b_a_cat>11</ogr:b_b_a_cat>
    <ogr:b_b_a_fid>corine_clip2.43</ogr:b_b_a_fid>
    <ogr:b_b_a_cat_>44</ogr:b_b_a_cat_>
    <ogr:b_b_a_Code2006>321</ogr:b_b_a_Code2006>
    <ogr:b_b_a_SHAPE_Leng>86978.6341462</ogr:b_b_a_SHAPE_Leng>
    <ogr:b_b_a_SHAPE_Area>57278248.0056</ogr:b_b_a_SHAPE_Area>
    <ogr:b_a_a_DTO>r</ogr:b_a_a_DTO>
    <ogr:b_a_a_AZT/>
   </ogr:output>
  </gml:featureMember>
  <gml:featureMember>
   <ogr:output fid="output.3">

<ogr:geometryProperty><gml:Polygon><gml:outerBoundaryIs><gml:LinearRing><gml:coordinates>53561
7.33,4426255.37,0 535619.66,4426226.73,0 535580.64,4426249.78,0 535598.03,4426262.1,0
535607.11,4426262.48,0
535617.33,4426255.37,0</gml:coordinates></gml:LinearRing></gml:outerBoundaryIs></gml:Polygon></o
gr:geometryProperty>
    <ogr:cat>4</ogr:cat>
    <ogr:a_cat>203</ogr:a_cat>
    <ogr:a_fid>output.193</ogr:a_fid>
    <ogr:a_cat_>203</ogr:a_cat_>
    <ogr:a_a_cat>14</ogr:a_a_cat>
    <ogr:a_a_fid>output.13</ogr:a_a_fid>
    <ogr:a_a_cat_>14</ogr:a_a_cat_>
    <ogr:a_a_a_cat>88</ogr:a_a_a_cat>
    <ogr:a_a_a_fid>slope_clip2.4</ogr:a_a_a_fid>
    <ogr:a_a_a_cat_>5</ogr:a_a_a_cat_>
    <ogr:a_a_a_SlopeCode>1</ogr:a_a_a_SlopeCode>
    <ogr:a_b_cat>1</ogr:a_b_cat>
    <ogr:a_b_fid>output.0</ogr:a_b_fid>
    <ogr:a_b_cat_>1</ogr:a_b_cat_>
    <ogr:a_b_a_cat>1</ogr:a_b_a_cat>
    <ogr:a_b_a_fid>aspect_clip2.2</ogr:a_b_a_fid>
    <ogr:a_b_a_cat_>3</ogr:a_b_a_cat_>
    <ogr:a_b_a_AspectCode>-1</ogr:a_b_a_AspectCode>
    <ogr:b_cat>78</ogr:b_cat>
    <ogr:b_fid>output.77</ogr:b_fid>
    <ogr:b_cat_>78</ogr:b_cat_>
    <ogr:b_a_cat>35</ogr:b_a_cat>
    <ogr:b_a_fid>output.34</ogr:b_a_fid>
    <ogr:b_a_cat_>35</ogr:b_a_cat_>
    <ogr:b_a_a_cat>24</ogr:b_a_a_cat>
    <ogr:b_a_a_fid>toprak_clip2.29</ogr:b_a_a_fid>
```

```
    <ogr:b_a_a_cat_>30</ogr:b_a_a_cat_>
    <ogr:b_a_a_BTG>B</ogr:b_a_a_BTG>
    <ogr:b_a_a_TOK>24</ogr:b_a_a_TOK>
    <ogr:b_a_a_ERZ>3</ogr:b_a_a_ERZ>
    <ogr:b_a_a_SAK>O</ogr:b_a_a_SAK>
    <ogr:b_a_a_SAK2>O</ogr:b_a_a_SAK2>
    <ogr:b_a_a_AKK>VII</ogr:b_a_a_AKK>
    <ogr:b_a_a_ATS>es</ogr:b_a_a_ATS>
    <ogr:b_a_a_BTG_TOK>B_24</ogr:b_a_a_BTG_TOK>
    <ogr:b_a_a_EGM>6</ogr:b_a_a_EGM>
    <ogr:b_a_a_DER>D</ogr:b_a_a_DER>
    <ogr:b_b_cat>37</ogr:b_b_cat>
    <ogr:b_b_fid>output.36</ogr:b_b_fid>
    <ogr:b_b_cat_>37</ogr:b_b_cat_>
    <ogr:b_b_a_cat>33</ogr:b_b_a_cat>
    <ogr:b_b_a_fid>corine_clip2.78</ogr:b_b_a_fid>
    <ogr:b_b_a_cat_>79</ogr:b_b_a_cat_>
    <ogr:b_b_a_Code2006>333</ogr:b_b_a_Code2006>
    <ogr:b_b_a_SHAPE_Leng>75498.3814264</ogr:b_b_a_SHAPE_Leng>
    <ogr:b_b_a_SHAPE_Area>27771956.1229</ogr:b_b_a_SHAPE_Area>
    <ogr:b_a_a_DTO/>
    <ogr:b_a_a_AZT/>
  </ogr:output>
</gml:featureMember>
```

ÖZGEÇMİŞ

21.09.1981 tarihinde Gümüşhane ilinde doğdu. İlk öğrenimine Rize'de başladı ve Giresun ilinin Tirebolu ilçesinde tamamladı. Orta ve lise öğrenimini Giresun HamdiBozbağ Anadolu lisesinde tamamladı. 2000 yılında başladığı Yıldız Teknik Üniversitesi Jeodezi ve Fotogrametri Mühendisliği bölümünü 2005 yılında tamamladı. Aynı yıl içinde Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsüne bağlı Mühendislik Fakültesi Jeodezi ve Fotogrametri Mühendisliği Yüksek Lisans programına girerek Araştırma Görevlisi olarak çalışmaya başladı. 2008 yılında Kartoğrafya Bilim Dalı Tezli Yüksek Lisans programından mezun olarak "Harita Yüksek Mühendisi" unvanını aldı ve aynı Bilim Dalında doktora eğitimine başladı. 2012-2013 yıllarında 1 senelik TÜBİTAK yurt dışı doktora araştırma bursu ile Amerika Birleşik Devletleri UTD (University Of Texas At Dallas) Semantik Web Laboratuvarında araştırmalarda bulundu. Bildiği yabancı dil İngilizcedir.