UML ile Modellenen Coğrafi Verilerin XSLT Yardımıyla OWL'a Dönüştürülmesi

Sermet Önel[1], Murat Komesli[1], Mehmet Cudi Okur[1]

[1] Yaşar Universitesi Yazılım Mühendisliği Bölümü
Üniversite Cad. 35, 35100, Bornova, İzmir, Türkiye
{sermet.onel, murat.komesli, mehmet.okur}@yasar.edu.tr

Özet. Coğrafi Bilgi Sistemleri (CBS), disiplinler arası sorunları çözme amaçlı geliştirilmiş, coğrafi ve sözel veri tabanına sahip, güçlü donanıma ihtiyaç duyan bir yazılım teknolojisidir. CBS, günümüz ihtiyaçları doğrultusunda Anlamsal Web'e doğru yönelmektedir. Anlamsal Web'in gerçekleştirilmesini savunduğu, bilgi sistemler arası veri paylaşımı ve birlikte çalışılabilirlik, mevcut UML (Unified Modelling Language) veri modelleme dili ile sağlanamadığından dolayı, yeni bir modelleme tekniğine ihtiyaç duyulmuştur. Sahip olduğu potansiyelle OWL (Web Ontology Language), bu sorunu giderebilecek bir çözümdür. Bu çalışma, coğrafi verilerin UML'den OWL'a dönüştürülmesini XSLT (XML Style Language Transformation) yardımıyla gerçekleştirmeyi konu almaktadır. Böylelikle, coğrafi verilerin bilgisayarlar tarafından anlaşılabilir ve çıkarsama yapılabilir hale getirilmesi amaçlanmaktadır.

Anahtar Kelimeler: Coğrafi Veri, UML, XML, XSLT, OWL.

1 Giriş

Günümüzde bilgisayar ve yazılım mühendisliği alanlarının ortak olarak üzerinde çalıştığı teknolojilerden birisi Anlamsal Web'dir. Anlamsal Web, kısaca web tabanlı bilginin bilgisayarlar tarafından anlaşılır, paylaşılır ve çıkarsama yapılabilir hale getirilmesini amaçlayan metotları kapsamaktadır [1]. Bu metotların geliştirilmesi amacıyla çeşitli veri modelleme teknikleri kullanılmış ve sözdizimi kuralları geliştirilmiştir. Bunlardan içerik ve fonksiyon olarak en zengini Web Ontoloji Dili (OWL)'dir [2].

CBS, yapısı itibariyle anlamsal web'in getirdiği yeniliklere büyük ihtiyaç duymaktadır. Çünkü, çözmek için geliştirildikleri ekonomik, sosyal ve çevresel sorunlar, zamanla sistem içerisinde büyük bir bilgi birikimine, aynı zamanda bu bilginin etkili ve hızlı bir şekilde analiz edilip işlenmesine gereksinim duymaktadır. Anlamsal web teknolojisi bu noktada görev almakta, insan ve bilgisayar tabanlı çalışma ortamını geliştirmeye hizmet etmektedir. Bunun için, bilgi sistemi içerisinde bir organizasyonun oluşturulması gerekmektedir.

İnsanların yazılım sistemlerini daha iyi kavraması için geliştirilmiş UML modeli, anlamsal web'in gerekliliği olan servislerin bilgisayarlarca anlaşılabilirliğinde ve bilgisayarlar arası kolay veri paylaşımında yetersiz kalmaktadır. Bu sebeple, UML standartlarıyla anlamsal web'in getirdiği anlamları iyi tanımlanmış veriler yaratılamamakta ve anlamsal web'den hedeflenen ideallere ulaşılamamaktadır.

Bu çalışmanın amacı, eksikliği görülen terminolojik birikimi sağlama görevini üstelenebilecek OWL modelini UML sınıf (class) diyagramı tabanlı coğrafi veriler ile bütünleştirilmesinin sağlanmasıdır. Bahsedilen amaç ile coğrafi bilgi sistemlerinde kullanılan UML modellerinin etkili bir biçimde OWL modellerine dönüştürülmesi mümkün olacaktır. UML ile modellenen coğrafi veriler XSLT dili kullanılarak veri ve anlam kaybına yol açmaksızın bire bir OWL modelindeki karşılıklarına dönüştürülmektedir. Bu sayede, insanlar için oldukça yararlı olan UML modelleme dili ile, bilgisayarlar için kullanışlı olan OWL modelleme dilinden birlikte yararlanarak anlamsal web idealine ulaşma şansı oluşacaktır. Çalışmanın bundan sonraki bölümlerinde UML ve OWL modelleme tekniklerinden kısaca bahsedilmiştir. Dördüncü bölümde, farklı modellerden otomatik OWL modeli yaratma işlemi için yapılan önceki çalışmalardan bahsedilmiştir. Beşinci bölümde UML'den OWL'a olabilecek örnek bir dönüştürme gerçekleştirilmiştir. Altıncı bölümde sonuçlar ve gelecek çalışmalar hakkında bilgi verilmiştir.

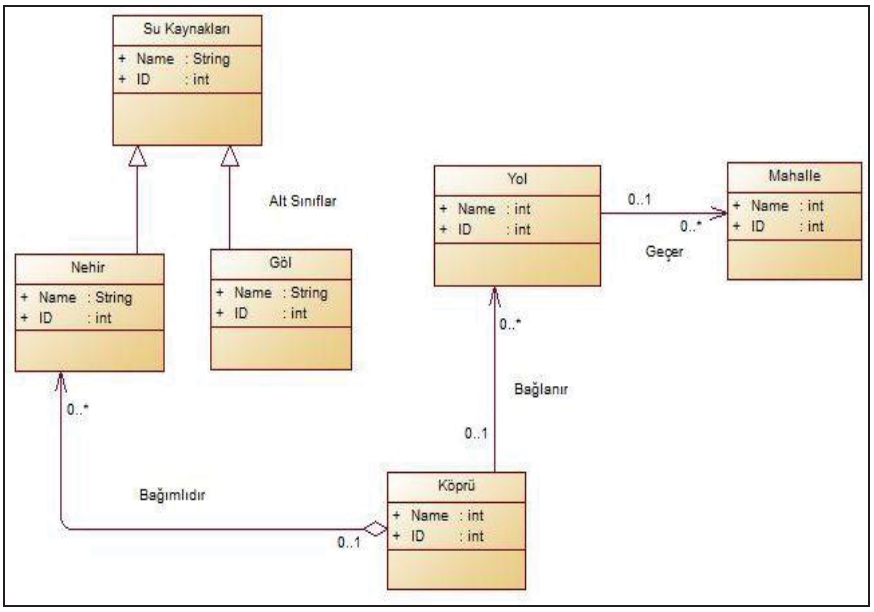
2 UML Veri Modeli

UML veri modeli, günümüzde yazılım sistemlerinin organizasyon ve yapısal şemasını anlatılmasında etkin olan bir modelleme tipidir. Özellikle sahip olduğu grafiksel öğelerle, insanların rahatça sistemi kavramasına olanak sağlamaktadır. Bir UML sınıf diyagramı modelinde dört temel öğe bulunur. Bunlar: Sınıflar (Classes), Sınıf İlişkileri (Relationships), Nesne Örnekleri (Object Instances) ve Paketler (Packages) [3]. Sınıflar, ortak özellikleri ve davranışları olan elemanların gruplandığı bölümdür. Her sistemde sınıflar arası işlemleri ve bağları açıklayan ilişkiler bulunur. Verinin içeriğine ait ilişkiler de kendi arasında belirli gruplara ayrılır. Örneğin, aynı seviye sınıflarda "Association", belli bir sınıftan türeyen alt sınıflar arasında "Inheritance", ve bir sınıfın özellik olarak kapsadığı altkümeler ise "Aggregation" ilişkileri ile tanımlanmaktadır. Sınıflar ve ilişkiler dışındaki önemli bir diğer yapı olan nesneler, sınıfın temelini oluşturan belirli kimlik, durum ve davranışa sahip öğelerdir [4]. Bir UML sınıf diyagramında sınıflar arası erişilebilirlik, paketler sayesinde belirtilmektedir. Aynı paketteki sınıflar birbirleriyle ilişkisi olan sınıflardır.

Coğrafi Bilgi Sistemleri de yapıları itibariyle UML modellemesine elverişlidir. Örneğin, bir bölgenin UML şemasıyla beraber; bölgedeki köprü, yol vb. ulaşım rotalarını, su kaynaklarını veya bölgede bulunan binaları bir organizasyon haline getirebilir, bu sayede coğrafi veri oluşturulur.

Şekil-1'de coğrafi elemanlarıyla bir UML sınıf diyagramı modellemesi örneği görülmektedir. UML modeli bölgenin coğrafik detaylarını göstermektedir. Su

kaynakları sınıfı, göl ve nehri kapsamakta, köprü nehir üzerinde kurulu olacağından aralarındaki ilişki belirtilmekte, köprü yola, yol da mahalleye bağlanarak coğrafi yapıyı tamamlamaktadır. Böylelikle, coğrafi detaylar ve birbirleri ile olan ilişkiler tanımlanmış olmaktadır. Bunun neticesinde, özellikle nesneye dayalı programlar geliştirilmesi sağlanabilmektedir.



Şekil-1 UML ile modellenmiş coğrafi bilgi.

3 OWL Veri Modeli

Anlamsal web idealine ulaşabilmek için bilgisayarların veriyi anlayabilmesi sağlanmalıdır. Bunun için iyi bir şekilde tanımlanmış ve anlam yüklenmiş verinin yaratabilmesi gerekmektedir. Bu noktada UML modeli yeterli olmamaktadır. Bilgisayarlar, UML modelinden gerekli anlamı çıkaramamakta ve veri paylaşımında sorun yaşanmaktadır. Bu işlemleri kolaylaştırabilmek amacıyla, ortak bir terminoloji tanımlayan ayrı bir modelleme tekniğine ihtiyaç duyulmuştur. Bu yapılar ise, ontolojilerdir. Ontolojiler, kısaca OWL adı verilen Web Ontoloji Dili kullanılarak geliştirilir.

Ontolojiler, belli bir alan üzerine odaklanarak; içerdiği sınıf, nesne, fonksiyon ve diğer öğe tanımlamalarıyla o alan üzerine gelişmiş bir sözlük görevi gören yazılım yapılarıdır. Bu sayede sistem içerisinde gerekli yeniden kullanılabilirlik ve etkili bilgi paylaşımı konularında köprü görevi görürler [5]. Sistemlerin gereksinimlerine göre farklı ontolojiler kullanılabilmekte, hatta ihtiyaca göre birden fazla ontoloji sistem içerisinde hizmet verebilmektedir. Ontolojiler sınıflandırılırken içeriklerine göre sınıflandırılır. Örneğin, coğrafi bilgi sistemleri göz önünde bulundurulduğunda bu

hizmeti sağlayan ontolojiler Coğrafi Ontoloji'lerdir. Coğrafi Ontolojiler, coğrafi terimleri kapsayan ve coğrafi bilgileri gruplayarak yazılım sistemleri için hizmete sunan yapılardır. Haritalar üzerinde coğrafi etiketlemeler yapan web servisler, Coğrafi Ontoloji'leri kullanan servisler örnek olarak sunulabilir [6].

OWL modelindeki yapılar temel olarak üçe ayrılır: Özellikler, sınıflar ve ilişkiler[3]. Sınıflar, aynı UML modelindeki gibi ortak özellik ve davranışa sahip elemanları gruplayan kısımdır. İlişkiler, sınıflar arasındaki bağları açıklar. Özellikler ise, sınıflara ait nesnelerin sahip oldukları özel metodları ya da elementleri betimler. OWL'da özellikler, veri özellikleri ve nesne özellikleri olmak üzere iki ana kısma ayrılmaktadır. Gerektiği takdirde bu özellikleri de niteleyen kısıtlamalar ve ayrıcalıklar ontolojilerde belirtilebilmektedir. Stanford Üniversitesi tarafından geliştirilen bir araç yazılımı olan Protege [7] kullanılarak bu çalışma kapsamında geliştirilen ontolojinin kaynak kodunun bir kısmı aşağıdadır.

```
<?xml version="1.0"?> <rdf:RDF
  xmlns:xsp="http://www.owl-
ontologies.com/2005/08/07/xsp.owl#"
  xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
  xmlns:swrl="http://www.w3.org/2003/11/swrl#"

xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns="http://www.owl-
ontologies.com/Ontology1326573903.owl#"
 xml:base="http://www.owl-
ontologies.com/Ontology1326573903.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Göl">
   <rdfs:subClassOf>
    <owl:Class rdf:ID="Su_Kaynakları"/>
   </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Köprü">
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf:ID="Bağlanır"/>
     </owl:onProperty>
     <owl:someValuesFrom>
      <owl:Class rdf:ID="Yol"/>
     </owl:someValuesFrom>
    </owl:Restriction>
   </rdfs:subClassOf>
```

```
                <rdfs:subClassOf>
                 <owl:Restriction>
                  <owl:onProperty>
                   <owl:ObjectProperty rdf:ID="Bağımlıdır"/>
                  </owl:onProperty>
                  <owl:allValuesFrom>
                   <owl:Class rdf:ID="Nehir"/>
                  </owl:allValuesFrom>
                 </owl:Restriction>
                </rdfs:subClassOf>
                <rdfs:subClassOf
        rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
               </owl:Class>
               <owl:Class rdf:ID="Mahalle"/>
               <owl:Class rdf:about="#Nehir">
                <rdfs:subClassOf rdf:resource="#Su_Kaynakları"/>
               </owl:Class>
               <owl:Class rdf:about="#Yol">
                <rdfs:subClassOf>
                 <owl:Restriction>
                  <owl:allValuesFrom rdf:resource="#Mahalle"/>
                  <owl:onProperty>
                   <owl:ObjectProperty rdf:ID="Geçer"/>
                  </owl:onProperty>
                 </owl:Restriction>
                </rdfs:subClassOf>
                <rdfs:subClassOf
        rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
               </owl:Class>
               <owl:DatatypeProperty rdf:ID="Name">
                <rdfs:range
        rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
               </owl:DatatypeProperty>
               <owl:DatatypeProperty rdf:ID="ID">
                <rdfs:range
        rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
 </owl:DatatypeProperty></rdf:RDF>
```

Görüldüğü üzere, OWL sınıfları, aynı UML sınıflarındaki gibi tanımlanmış, aradaki ilişkiler de nesne özellikleri ile belirtilmiştir. Bunun yanında, veri özellikleri de sınıflar içerisindeki isim ve "id" gibi nitelikleri belirtmek üzere tanımlanmıştır. Bu haliyle UML modeli OWL'a çevrilmiş ve bilgisayarlar tarafından paylaşılabilir ve anlaşılabilir bir hale getirilmiştir.

4 Önceki Çalışmalar

CBS, sistemler arası veri paylaşımı ve birlikte çalışabilirlik gibi ihtiyaçları itibariyle Anlamsal Web'e doğru kaymaktadır. Yapısı itibariyle UML modelleri, bu ihtiyacı karşılamaktan yoksun oldukları için, çözüm olarak OWL modeli sunulmaktadır. CBS disiplinler arası bir organizasyona sahip olduğundan, doğrudan OWL modelini yaratma işlemi ontoloji uyarlaması veya disipliner bir paylaşım oluşturulmasında çeşitli engeller sunabilmektedir [8]. Bu sebeple, OWL modelini elde edebilmek için otomatik bir çevirme işlemi ihtiyacı görülmüş ve bu konu hakkında birçok çalışma yapılmıştır.

Bu çalışmalar 3 grupta sınıflandırılabilir: Birinci grup, modeller arası veri yapılarını doğrudan çevirmeyi baz almaktadır. Gasevic ve çalışma arkadaşları, UML ve OWL modellerini birbirine dönüştürmüştür [9]. Pivk, HTML tabanlı web yapılarını otomatik bir biçimde OWL modeline dönüştürmek için Java tabanlı yazılım kullanmıştır [10]. Bohring ve Auer ise, XML ile OWL modeli arasındaki eşleme sorunu üzerine çalışmıştır [11].

İkinci grup, veri madenciliği bazlı dönüştürme işlemini konu alan çalışmalardır. Genellikle çalışmalar metinler üzerinden OWL modeli oluşturmaya yöneliktir. [12,13]

Son gruptakiler ise, harici bilgi sistemlerini baz alan dönüştürme sistemleridir. Moldovan ve Girju'nun doğal dil işleme alanında WORDNET'i kullanarak OWL modeli yaratma işlemi [14], Kietz ve çalışma arkadaşlarının Intranet üzerinden OWL modeli yaratma çalışmaları [15] ve Agirre ve çalışma arkadaşlarının WWW'i kullanarak büyük ontoloji üretme çalışmaları [16] bu alanda verilebilecek örneklerdendir.

5 Dönüştürme İşlemi

Daha önceki kısımlarda bahsedilen UML modelinden OWL modelini elde etmek için bir dönüştürme işlemi yapmak gerekmektedir. Bunu yapabilmek için gerekli adımlar bu kısımda anlatılmıştır.

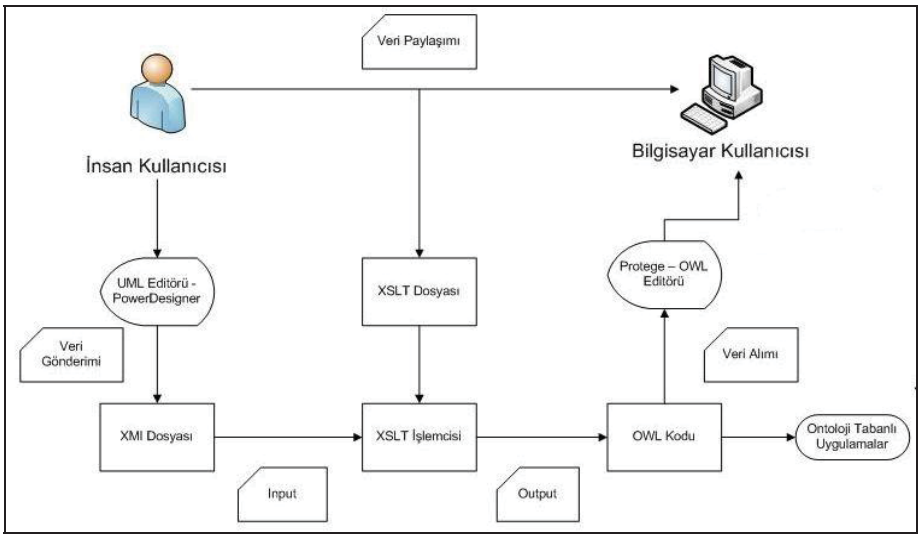
Dönüştürme işleminin ilk basamağı, hazırlanan UML modelinin "XML Metadata Interchange" (XMI) formatında saklanmasıdır [9]. Bu yapı Nesne Yönetim Grubu (Object Management Group – OMG) standardı olarak kabul edilmektedir. XML tabanlı farklı alanlar üzerine yoğunlaşmış sözdizimlerinden biri olan XMI, (bir başka örnek olarak coğrafi bilgi üzerine yoğunlaşan XML'den geliştirilmiş GML'i örnek gösterebiliriz) UML tabanlı yapıları tanımlamada ve diğer heterojen sistemlerle paylaşmada görev almaktadır. Şekil-1'deki örneği geliştirirken Sybase firmasının PowerDesigner [17] isimli UML aracı kullanılmıştır. Bu maksatla Sınıf diyagramını PowerDesigner'ın menüsünde File->Export->XMI File seçilerek XMI dosyası elde edilmiştir. Örnek olarak geliştirilen XMI dosyasının bir bölümü aşağıdadır.

```
<UML:Class name="Yol" isLeaf="false"
xmi.id="{B8EB50A6-E335-4B4E-9707-91E8B426CBA6}"isAbstract="false"
visibility="public">
                    <UML:ModelElement.taggedValue>
                      <UML:TaggedValue tag="strictfp" value="false"/>
                    </UML:ModelElement.taggedValue>
                    <UML:Classifier.feature>
                      <UML:Attribute name="Name"
xmi.id="{3C3F922E-C04A-4B5D-A107-AF05CCB33361}"
ownerScope="instance" visibility="public" changeability="changeable">
                        <UML:StructuralFeature.type>
                          <UML:Classifier xmi.idref="Dttp0"/>
                        </UML:StructuralFeature.type>
                      </UML:Attribute>
                      <UML:Attribute name="ID"
xmi.id="{DDEB45A6-D09E-439E-9A06-2A4047499DB3}"
ownerScope="instance" visibility="public" changeability="changeable">
                        <UML:StructuralFeature.type>
                          <UML:Classifier xmi.idref="Dttp0"/>
                        </UML:StructuralFeature.type>
                      </UML:Attribute>
                    </UML:Classifier.feature>
                  </UML:Class>
```

Yaratılan XMI dosyası içerisinde Yol isimli sınıf ve onun sahip olduğu veri özellikleri tanımlanmaktadır. UML etiketleriyle beraber hangi modelleme tekniği altında gerçekleştirildiği de görülmekte ve UML modeli aynı XML benzeri bir formata dönüşerek XSLT işlemcilerine uygun hale getirilmiştir. Bu sayede, Şekil-2'de gösterildiği gibi XSLT ve XMI dosyası işlemci üzerinden OWL dokümanına dönüştürülebilmektedir [18] [19]. Böylelikle, OWL ile modellenmiş coğrafi veri, anlamsal web uygulamalarının kullanımına hazır hale getirilmiştir.

XMI dosyasını XSLT aracılığıyla OWL dosyasına çevirirken çeşitli sorunlarla karşılaşma riski doğmaktadır. UML modelinin XMI'a çevrilirken her parçanın çevrilmesi sonucu elde edilen bilgi kirliliği, veya eldeki diyagram sayısına göre oluşturulan XMI dosyasının ortaya birleşik bir sonuç çıkarması ve orjinal dosyaya göre farklılaşması, veya oluşturulan modelin doğruluğunun kontrolü gibi farklı etmenler göz önünde bulundurulmalıdır [20].

XSLT dönüştürmesinde temel olarak yapılacak işlem UML modeli içerisinde tanımlanmış olan yapıların OWL'daki karşılıkları ve içerdiği elementlerin bire bir OWL modeline aktarılmasıdır [21]. Bu sebeple, XSLT dosyası ile genel yapısı göz önüne alınarak temel bir dönüştürme işlemi yapılır. Bu fonksiyonları tanımlarken en ufak detaylar olan "link"ler, göz önünde bulundurulmalıdır. Örneklendirmek maksadıyla XSLT kodundaki bazı kısımlar aşağıda verilmiştir:



Şekil-2 Coğrafi Verinin Dönüştürülmesi.

```
    <xsl:template match="/">
            <xsl:apply-templates select="XMI"/>
        </xsl:template>
        <xsl:template match="XMI">
            <xsl:apply-templates select="XMI.content/UML:Model"/>
        </xsl:template>
        <xsl:template match="XMI.content/UML:Model">
            <xsl:apply-templates select="UML:Namespace.ownedElement"/>
        </xsl:template>
        <xsl:template match="UML:Namespace.ownedElement">
           <rdf:RDF>
        <xsl:attribute                                name                                =
"xml:base">http://owl.protege.stanford.edu</xsl:attribute>
        <xsl:apply-templates
select="UML:Package/UML:Namespace.ownedElement"/>
           </rdf:RDF>
        </xsl:template>
        <xsl:template match="UML:Package/UML:Namespace.ownedElement">
        <xsl:if test="(key('ID', UML:ModelElement/UML /@xmi.idref)/@name =
'ontology') and (key('ID', UML:ModelElement /UML /@xmi.idref)/UML: baseClass
= 'Package')">
            <xsl:apply-templates select="UML:Class"/>
                <xsl:apply-templates select="UML:Object"/>
        </xsl:if>
          </xsl:template>
```

[22]

Örnek kodda UML modelinin XMI formatında tanımlanmış dokümanındaki örnek bir nesnenin dönüştürüleceği ontoloji yapısındaki yeri, bağlantılı olduğu yapıları XSLT kodunda gösterilmiştir. Bu sayede, XSLT kodunun çalıştırılacağı XMI formatlı UML modellemesi, gerekli bağlantı ayarları yapılarak OWL formatında dönüştürülmüştür.

```
<xsl:template match="UML:Class">
    <xsl:variable name="param1">
        <xsl:value-of
select="UML:ModelElement.stereotype/UML:@xmi.idref"/>
    </xsl:variable>
    <xsl:variable name="id">
      <xsl:value-of select="@xmi.id"/>
    </xsl:variable>
    <xsl:variable name="name">
    <xsl:value-of select="@xmi.name"/>
    </xsl:variable>
    <xsl:if test="$classSterotype = 'ObjectProperty'">
       <xsl:call-template name="ObjectProperty"/>
    </xsl:if>
    <xsl:if test="$classSterotype = 'DatatypeProperty'">
      <xsl:call-template name="DatatypeProperty"/>
    </xsl:if>
    <xsl:if test="($classSterotype = 'OntClass') or ($class =
'Restriction')">
<xsl:call-template name="Class"/> </xsl:if> </xsl:template>
```

[22]

Yukarıdaki kısımda ise, üst kısımda belirtilen sınıfın sahip olduğu öğelerin OWL modellemesindeki karşılıklarına dönüştürülmesi işlemi gösterilmiştir. Görüldüğü üzere, UML modellemesindeki öğelerin ve bağlantıların OWL modellemesine dönüştürülürken veri tipi veya nesne özelliklerine ait olup olmadığı kontrol edilmektedir. Ayrıca, UML modellemesinde belirtilen ilişki tiplerine göre OWL modellemesindeki kısıtlama, kesişme ve birleşme gibi detaylandırmalar da üretilecek OWL koduna eklenecektir.

Bir nesne özelliği olduğunda, bu nesne özelliğinin alt sınıf veya üst sınıfları gibi detayları belirtilmelidir. Aksi takdirde geliştirilen ontolojide veri kaybı yaşanabilir[23]. Bu sorun aşağıda geliştirilen kod ile giderilebilmektedir.

```
<xsl:template name="ObjectProperty">
   <xsl:variable name="range">
      <xsl:text>Class</xsl:text>
   </xsl:variable>
 <xsl:variable name="classID" select="@xmi.id"/>
   <xsl:element name="owl:ObjectProperty">
      <xsl:attribute                                    name="rdf:ID"><xsl:value-
ofselect=./@name/></xsl:attribute >
          <xsl:call-template name="classDependency">
              <xsl:text>equivalentProperty</xsl:text>
```

```
</xsl:call-template>
<xsl:call-template name="taggedValues"/>
<xsl:call-template name="generalization">
        <xsl:text>rdfs:subPropertyOf</xsl:text>
</xsl:call-template>
<xsl:call-template name="attributeDomainRange"/>
<xsl:call-template name="associationDomainRange">
        <xsl:with-param name="range" select="$range"/>
</xsl:call-template> </xsl:element> </xsl:template>
```

[24]

Bir nesne özelliğinde tanımlanması gereken özellikler bir alt nesneye sahip olması, bir başka nesne tarafından kapsanması veya "range", "domain" gibi detaylandırmalara sahip olmasıdır. Geliştirilen XSLT kodunda bu dönüştürme işlemini yapacak fonksiyonlar üstteki kodda açıklanmıştır. Veritipi özelliklerinde de aynı işlemler kullanılacaktır.

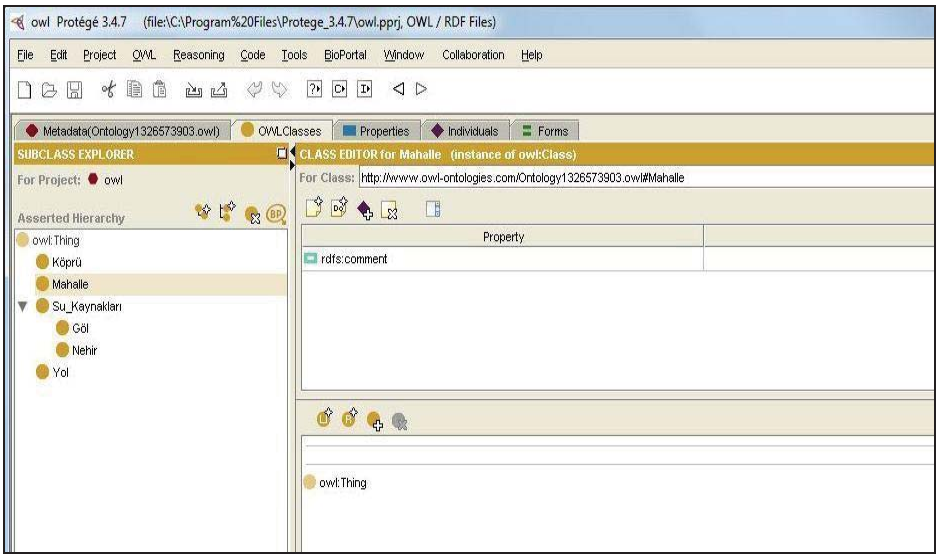
Şekil-2'deki algoritmayı test etmek maksadıyla, UML ve OWL kısımlarında gösterilen örnek kod kullanılmıştır. Sybase firmasının PowerDesigner aracıyla, UML modellemesini ve XMI formatlı dosya üretim süreci tamamlanır. Sonraki aşamada XML Spy 12 [25] isimli araç yazılımıyla, XMI dosyası ve XSLT çevirici dosyası ortak işleme sokulur. Sonunda elde edilen OWL uzantılı ontoloji test amaçlı olarak Protege 3.4.7 araç yazılımıyla açılır ve Şekil-3'deki görüntü elde edilir [26].

6 Sonuç

UML sınıf diyagramı dosyasının XSLT ile OWL' a dönüştürülmesi neticesinde geliştirilen ontoloji, anlamca daha zengin bir hale gelmiştir. Dolayısıyla, anlamsal web uygulamalarına uygundur. Konunun anlaşılabilmesi maksadıyla örnek kod mümkün olduğunca basitleştirilmiştir. Sadece belirli nesne ve sınıflar ele alınmış, fazla ilişki ve kısıtlardan kaçınılmıştır. Böylelikle, OWL'daki karşılıkları belirli olan UML yapıları, sorunsuz olarak bir ontolojiye dönüştürülmüştür. Daha karmaşık yapılar, benzer şekilde dönüştürülebilinecektir. Böylece, çalışma kapsamında geliştirilen çevirici XSLT dosyası, oluşturulan modellemenin doğruluğunu kontrol etmeye çalışacağından daha iyi bir deneme süreci elde edilmiş olunur.

Günümüzde, kullanıma açık UML modellemeleri arasında bazı farklar mevcuttur. Çalışmada kullanılan PowerDesigner ile Poseidon [27] araçları için geçerli olan bu dönüştürme algoritması, farklı UML araçlarının kullanabileceği sözdizimleri ile farklı sonuçlar verebilir. Bu çalışmada, önceki çalışmalardan farklı olarak, UML modellemesinde PowerDesigner aracı kullanılmıştır. Dönüştürme işleminin gerçekleşmesi için gerekli olan XSLT çeviricisi, hazır XSLT kütüphanesinden temin edilmiştir ve PowerDesigner'dan alınan XMI dosyasıyla uyumluluk gösterdiği görülmüştür. UML modelinin bilgisayar kullanıcıları için yetersiz kalmasının nedenlerinden biri olan bu uyumluluk konusu hakkında, yapılması gereken daha

kapsamlı bir yelpazede dönüştürme işlemi yapabilen daha genel bir XSLT dosyası üretmek olacaktır. Sonraki çalışmalarda bu sorun ele alınacaktır.



Şekil-3 OWL ile modellenmiş coğrafi veri.

Temel olarak çalışmanın geleceğin web ortamı olan anlamsal web'de hem insan hem de bilgisayar kullanıcıları için yararlı olduğu söylenebilir. İnsanların verimini artırmak için gerekli olan UML ve bilgisayarların verimini artırmak için gerekli olan OWL dosyalarının bir sistem içerisinde dönüştürülerek ortak kullanımı, sistemin genel performansını ve kullanımını da geliştirecektir. Daha geniş veri tabanlarıyla çalışan ve çok modüllü sistemler için bu çalışma, faydalı bir dönüştürme işlemi sunmaktadır.

Kaynaklar

1. Berners-Lee T., Hendler J., Lassila O.: "The semantic web", In: Scientific American, Vol.284(5), pp.34–43, 2001.
2. OWL Web Ontology Language Overview, http://www.w3.org/TR/owl-features/, Erişim Tarihi: 3 Nisan 2014
3. Zhang C., Peng Z.R., Zhao T., Li W.: "Transformation of Transportation Data Models From Unified Modeling Language to Web Ontology Language", In: Proceedings of the Transportation Research Board (TRB) Annual Meeting, vol.2064, pp.81-8, 2008.
4. Kocabıçak Ü., 2012. "Nesneye Dayalı Programlama ve UML", http://web.sakarya.edu.tr/~umit/panel/dosya/hafta1a.pdf, Erişim Tarihi: 10 Nisan 2014
5. Gruber T. R.: A Translation Approach to Portable Ontology Specifications, In: Knowledge Acquisition, no: 5(2), pp. 199-220, 1993.
6. Geonames Ontology, http://www.geonames.org/ontology/documentation.html, Erişim Tarihi: 7 Nisan 2014.
7. Protégé Web Sitesi, http://protege.stanford.edu, Erişim Tarihi: 15 Nisan 2014

8. Ratsch E., Schultz J., Saric J., Lavin P. C., Wittig U.: “Developing a Protein-interactions Ontology”. Comparative and Functional Genomics, Vol.4, pp. 85–89, 2003
9. Gasevic D., Djuric D., Devedzic V., Damjanovic V.: “Converting UML to OWL ontologies”, In: WWW (Alternate Track Papers & Posters), pp.488-489, 2004
10. Pivk A.: “Automatic ontology generation from Web tabular structures”. In: AI Communications, Vol.19, pp. 83–85, 2006
11. Bohring H., Auer S.: “Mapping XML to OWL Ontologies”. In: Leipziger Informatik-Tage, Vol.72 of LNI. GI, pp. 147–156, 2005
12. Hu H., Liu D.: “Learning OWL Ontologies from Free Texts”. In: Proc. International Conference on Machine Learning and Cybernetics (ICMLC 04), pp. 1233–1237, 2004
13. Biebow B., Szulman S.: “TERMINAE: a linguistics-based tool for the building of a domain ontology”. In: Proc. 11th European Workshop on Knowledge Acquisition, Modeling, and Management (EKAW 99), pp. 49–66, 1999
14. Moldovan D., Girju R.: “Domain-specific knowledge acquisition and classification using Wordnet”. In: Proc. 13th International Florida Artificial Intelligence Research Society Conference (IFAIRSC 00), pp. 224–228, 2000
15. Kietz J., Maedche A., Volz R.: “A method for semi-automatic ontology acquisition from a corporate Intranet”. In: Proc. 12nd European Workshop on Knowledge Acquisition, Modeling and Management (EKAW 00), 2000
16. Agirre E., Ansa O., Hovy E, Martinez D.: “Enriching very large ontologies using the WWW”. In: Proc. Ontology Learning Workshop (OL 00), 2000
17. SAP Sybase Power Designer, http://www.sybase.com/products/modelingdevelopment/powerdesigner, Erişim tarihi: 20 Şubat 2014
18. Otegem M.V., 2002. “Teach Yourself XSLT in 21 Days”, SAMS
19. Tidwell D., 2008. “XSLT Second Edition”, O’Reilly Media
20. Gasevic D., Djuric D., Devedzic V.: “Model Driven Engineering and Ontology Development (2. ed.)”, In: Springer 2009: I-XXI, pp. 1-378, 2009
21. Mangano S., 2005. “ XSLT Cookbook 2nd Edition”, O'Reilly Media
22. UMLtoOWL Project, http://www.sfu.ca/~dgasevic/projects/UMLtoOWL/
23. Viademonte, S., Chui, Z.: “Deriving OWL Ontologies from UML Models: an Enterprise Modelling Approach”, Academia.edu, 2011
24. Gasevic D., Djuric D., Devedzic V.: “MDA-based Automatic OWL Ontology Development”, In: International Journal on Software Tools for Technology Transfer, Vol. 9, No. 2, pp. 103-117, 2007
25. XML SPY XML Editor, http://www.altova.com/xmlspy.html, Erişim Tarihi: 18 Nisan 2014
26. Protege OWL Tutorial, http://130.88.198.11/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP3_v1_0.pdf, Erişim Tarihi: 19 Nisan 2014
27. Poseidon for UML, http://www.gentleware.com/new-poseidon-for-uml-8-0.html, Erişim Tarihi: 20 Nisan 2014