Anlamsal Web için Bilgi Sistemi Altyapısı

M.O. Ünalır, Ö. Öztürk, T. Özacar
Ege Üniversitesi Bilgisayar Müh. Bölümü, İzmir
unalir@bornova.ege.edu.tr,ozturk@staff.ege.edu.tr,ozacar@staff.ege.edu.tr

Özetçe

Anlamsal Web'in gerçekleştirilmesinde kilit bir role sahip olan ontolojilerin geliştirilmesi ve kullanılması için ontoloji geliştirme ve saklama ortamlarına ihtiyaç vardır. Anlamsal Web için geliştirilen ontoloji geliştirme ortamlarının, gelecekte uygulamaların yapı taşı olması beklenen Web servislerini de desteklemesi gerekmektedir. Bu çalışmada AEGONT projesi kapsamında geliştirilmekte olan ontoloji ortamı anlatılmaktadır.

Abstract

Ontologies have a critical role for implementing the semantic Web. It is important to implement enviroments to create and to store ontologies. These environments have to support Web services that is the future of the applications. In this article, we describe the ontology environment that is the subject of the AEGONT project.

1. Giriş

Günümüzde bilginin önemi gittikçe artmaktadır. Buna paralel olarak da paylaşılan bilginin miktarı her geçen gün artmaktadır. Bu durum insanların daha fazla bilgiye ulaşabilmesi açısından faydalı olsa da; insanların, bu bilgiyi verimli bir şekilde kullanabilmesi için, incelemesi gereken bilgide de diğer bir deyişle harcaması gereken zamanda da büyük bir artış olmaktadır.

Bilgi paylaşımının en kolay ve verimli olduğu ortam Web'tir. Web üzerinde bir bilginin yayılma hızı diğer hiç bir ortam ile karşılaştıralamayacak kadar hızlıdır. Ayrıca paylaşılan bilgi ses, görüntü, elektronik kitap gibi çok değişik biçimlerde de olabilir.

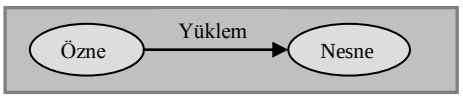
Fakat bu bilgilerin büyük bir kısmı yapılandırılmamış bilgidir. Örneğin, bir alışveriş sitesinde bulunan bir sayfadaki ürün listesinde hangi alanın ürün adı, hangi alanın ürün fiyatı olduğu gibi bilgiler bulunmamaktadır. Bir HTML belgesinde bilgilerin ekranda nasıl gösterilmesi gerektiğinin bilgisi tutulur, diğer bir deyişle insanların bu bilgiyi doğru bir şekilde anlayabilmeleri için gerekli yapılar sağlanmıştır.

Bilgilerin yapılandırılması için W3C tarafından XML[1] dili önerilmiştir. XML dili ile yazılan bir belgenin neresinde hangi bilginin bulunduğu artık belgenin içinde tutulmaktadır. Bilginin ekranda nasıl gösterilmesi gerektiği bilgisi XML belgesinde bulunmaz. Uygulamalar bilginin gösteriminin nasıl olması gerektiğine karar verirler. Diğer bir deyişle XML ile bilginin gösterimi ve yapısı birbirinden ayrılmıştır.

XML belgelerinin yapısının doğruluğu XML Schema[2] dili kullanılarak yazılan belgelere bakılarak yapılır. Aynı XML Schema'yı kullanan yazılımlar birbirlerinin bilgilerini anlayabilmektedir. Fakat bu katmanda da farklı XML Schema'ları kullanan etmenler birbirlerini anlayamamaktadırlar. Çünkü XML belgeleri, gösterdikleri bilginin yapısını içermekle birlikte bu yapının anlamını içermemektedirler.

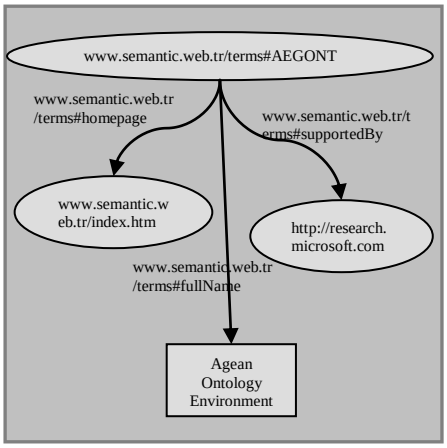
İşte bu yapının anlamını gösterebilmek için ontolojilerden yararlanılmaktadır. Kısaca kavramlaştırmanın tanımlanması olarak nitelendirilen ontolojiler, bilginin gösterimi alanında yapılan araştırmalarda yüzlerce yıldır kullanılmaktadırlar[3]. Felsefenin araştırma konusu olan ontolojiler yapay zeka araştırmalarının yoğunlaşması ile bilgi teknolojisi alanında da kullanılmaya başlanmıştır.
Bilginin yapısının anlamını belirtmek için W3C tarafından RDF (Resource Description Framework), RDF Schema, OWL (Web Ontology Language) dilleri önerilmiştir.

RDF[4] dili kullanılarak Web üzerindeki herhangi bir bilgi, bir kaynak olarak tanımlanabilmektedir. Daha sonra bu kaynak hakkında herhangi bir bilgi de yine RDF üçlüleri kullanılarak tanımlanabilmektedir. RDF üçlülerinin yapısı Şekil 1'de gösterilmektedir.



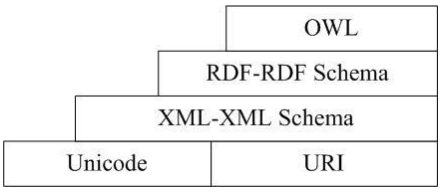
Şekil 1: RDF üçlülerinin yapısı

Şekil 2'de de AEGONT projesini anlatan bir RDF belgesinin çizgesi gösterilmektedir. Bu çizgede, AEGONT projesinin anasayfasına "www.semantic.web.tr/index.htm" adresinden ulaşılabildiği; açılımının "Aegean Ontology Environment" olduğu ve Microsoft Research tarafından desteklendiği anlatılmaktadır.



Şekil 2: AEGONT projesi RDF çizgesi

RDF Schema[5] dili RDF dili ile tanımlanamayan alt-üst sınıf ilişkilerini, niteliklere önalan ve artalan kısıtlarını yapabilmek için önerilmiştir. RDF Schema dili de RDF ile yazılmıştır. OWL[6] dili de RDF ve RDF Schema dillerine yapılmış bir eklentidir ve yine RDF dili kullanılarak tanımlanmıştır. Şekil 3'de Anlamsal Web üzerinde kullanılan diller gösterilmektedir.



Şekil 3: Anlamsal Web dilleri

Tüm bu dillerin önerilmesindeki amaç Web'in gelişimindeki ikinci aşama olan Anlamsal Web'e geçerken gerekli olan altyapıyı sağlamaktır. Bütün bu diller ile Web üzerinde sadece bilgiye değil, bilginin anlamına da ulaşmak mümkün kılar. Bu sayede artık ulaşılabilir durumda olan bu anlamı makinalar; bilgisayarlar, cep telefonları, beyaz eşyalar vs.; işleyebilir duruma gelir. Böylece Anlamsal Web, Web üzerindeki bilgi paylaşımını yalnızca insanlar arasında olmaktan çıkararak makinaları da bu paylaşıma dahil edecektir [7]. Bu insanların günlük hayatını da kolaylaştıracaktır. İnsanlar için çalışan etmenler, tanımlanmış ontolojileri kullanarak bazı görevleri kendi kendine yapabileceklerdir.

Bu çalışmada anlatılan altyapı da, bu dilleri kullanarak ontoloji yaratılması, daha sonra yaratılan ontolojilerin saklanması ve sorgulanmasını gerçekleştirebilmek amacıyla tasarlanmıştır. Anlamsal Web için en önemli araç olan ontolojilerin daha kolay oluşturulabilir ve ulaşılabilir olmasını sağlamak Anlamsal Web'in gerçekleştirilmesine katkı sağlayacağı düşünülmektedir.

İkinci bölümde ontolojiler ve ontoloji tanımlama dilleri anlatılmaktadır, üçüncü bölümde ontoloji araçları ve bunların sınıflandırmasından bahsedilmektedir. Dördüncü bölümde ise AEGONT Ontoloji Ortamı'nın mimarisi ve modülleri anlatılmaktadır. Beşinci ve son bölümde de sonuçlar ve bundan sonra yapılması planlanan çalışmalar sunulmaktadır.

2. Ontoloji ve Ontoloji Tanımlama Dilleri

Yapay zeka üzerinde çalışan araştırmacılar, herhangi bir makinanın zeka ve öğrenme ile ilgili bir davranış sergileyebilmesi için gerekli bilginin, makinaların anlayıp işleyebileceği bir biçimde sunulması gerektiğini farkettiler. Bu nedenle bilgi gösterimi

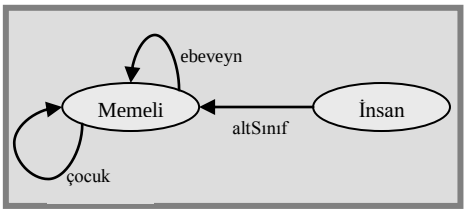
başlı başına bir araştırma alanı haline geldi ve o zamanlardan beri bilgiyi yapısal bir şekilde modelleyebilmek için birçok yöntem geliştirildi.

Bu yöntemlerden biri de ontolojilerdir. Ontoloji terimi için birçok tanımlama yapılmıştır. Bu tanımlar içerisinde en çok bilineni Gruber'in aşağıdaki tanımıdır [8]:

Bir ontoloji, ortak bir kavramsallaştırmanın biçimsel gösterimidir.

Kavramsallaştırma belli bir önalanda, bu önalana ait soyut model oluşturma anlamına gelir. Ontolojinin biçimsel olması, ontolojinin makinalar tarafından anlaşılmasını sağlar. Kavramsallaştırmanın ortak olması, ontolojinin ortak olarak benimsenmiş diğer bir deyişle tek bir nesneye değil de bir gruba özgü bilgiyi modellemesidir.

Bir ontoloji dökümanında ontolojinin içeriğine uygun olarak sınıflar, sınıflara ait öznitelikler, bu sınıflar arası ilişkiler, sınıf örnekleri ve bu sınıf örneklerine ait öznitelik değerleri bulunur. Bir ontoloji oluşturmak için çerçeveler ve birinci basamak mantık, yazılım mühendisliği teknikleri ve ya veritabanı teknikleri gibi yöntemler kullanılabilmesine karşın büyük ve eksiksiz bir ontolojiyi modellerken kavramsal dillerle modelleme tercih edilmektedir.



Şekil 4: Basit bir ontoloji

Kavramsal bir dilin temel elemanları, sınıfları ve bu sınıflara ait öznitelikleri tanımlamaya yarayan yapılardır. Kavramsal dilin ifade gücü, basit sınıf tanımlarını kullanarak karmaşık sınıflar tanımlayabilme yeteneği ile ölçülür.

Şekil 4'te basit bir ontoloji görülmektedir. Bu ontoloji "Memeli" ve "İnsan" sınıflarını içermektedir. "İnsan" sınıfı "Memeli" sınıfının alt sınıfıdır. Sınıflar arası bu ilişki, bir ontoloji biçimleme dili olan RDF ile gerçekleştirilmiştir. "Memeli" sınıfına ait iki öznitelik bulunmaktadır. "çocuk" ve "ebeveyn" özelliklerinin önalanı "Memeli" sınıfıdır. Önalan bu özniteliklerin ait olduğu sınıfı tanımlar. Şekilde okun ucu özniteliğin ardalanını göstermektedir. Buna göre bu iki özniteliğin alabileceği değerlerin yine "Memeli" sınıfına ait örnekler olması gerekmektedir.

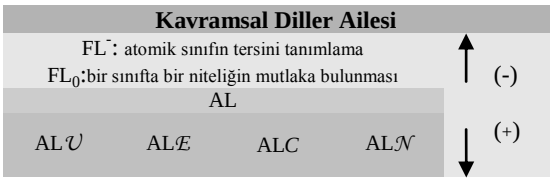
En temel kavramsal dil (AL[1]);

- atomik sınıf tanımlama
- evrensel sınıf tanımlama
- boş sınıf tanımlama
- atomik bir sınıfın tersini tanımlama
  $Soyut = \neg Somut$
- sınıfların kesişimini tanımlama
  $Kadın = \neg Erkek \cap İnsan$
- bir sınıfın bir niteliği üzerinde değer kısıtlaması gerçekleştirebilme
  $İnsan = Canlı \cap \forall ebeveyn.İnsan$
- bir sınıfta bir niteliğin mutlaka bulunmasını sağlama
  $Evli = İnsan \cap \exists Eş$

yeteneklerine sahiptir.

FL^- ve FL_0 dilleri, AL dilinin atalarıdır. Bu diller atomik sınıfın tersini tanımlayamaz ve bir sınıfta bir niteliğin mutlaka bulunmasını sağlayamazlar.

En temel kavramsal dile, bazı özelliklerin eklenmesi ile AL$\mathcal{U}$, AL$\mathcal{E}$, ALC, AL$\mathcal{N}$ gibi kavramsal diller geliştirilmiştir (Şekil 5).



Şekil 5: Kavramsal diller ailesi

[1] AL(attributive language): Kavramsal diller sağladıkları dil yapıları ile değerlendirilirler. Bir kavramsal dilin hangi dil yapılarını sağladığını görmek için bu diller AL dil ailesi ile eşleştirilirler. En temel kavramsal dilin anlatım gücü, AL dil ailesine ait AL [9] dilinin anlatım gücüne eşittir. AL dil ailesine ait diğer diller, AL dilinin genişletilmiş halidir.

AL$\mathcal{U}$, sınıfların birleşimini de tanımlayabilir.

$İnsan = Kadın \cup Erkek$

AL$\mathcal{E}$, bir sınıfta belli bir değer kısıtlamasına sahip bir niteliğin mutlaka bulunmasını sağlar.

$ErkekÇocuğuOlanlar = İnsan \cap \exists çocuk.Erkek$

ALC, yalnız basit atomik sınıfların değil bu sınıflar kullanılarak türetilmiş sınıflarında tersini tanımlayabilir.

AL$\mathcal{N}$, bir sınıfa ait öznitelikler üzerinde sayı kısıtlaması yapabilmeyi sağlar.

$ÇokÇocuğuOlanlar = İnsan \cap \exists^{>2} çocuk$

Anlamsal Web üzerindeki çalışmalar yoğunlaştıkça, ontoloji biçimleme dilleri (XML tabanlı) geliştirilmeye başlandı. Bu dillere örnek olarak RDF, RDFS, OIL[10], DAML+OIL[11], ve OWL gösterilebilir.

Ontoloji biçimleme dillerinin anlatım gücü kriterlerine göre değerlendirilmesi Tablo 1'de sunulmuştur[12].

Tabloda "+" simgesi dilin bu özelliği desteklediğini, "-" simgesi desteklemediğini, "±" simgesi ise dilin bu özelliği doğrudan desteklemediğini fakat dolaylı olarak destekleyebilir hale getirilebileceğini belirtmektedir.

Örnek öznitelikleri, her dilde tanımlanabilmesine rağmen sınıf öznitelikleri için aynı durum söz konusu değildir. Örneğin OWL'da sınıfa ait bir niteliği tanımlayabilmek için öncelikle bir nitelik tanımlanır bu niteliğin önalanına ait olduğu sınıfı atayarak sınıfa ait bir nitelik dolaylı yoldan tanımlanmış olur.

Projenin desteklediği dillerin büyük bir çoğunluğunun, öznitelikler üzerinde değer ve eşlenme derecesi kısıtlamaları gerçekleştirebildiği görülmektedir.

Kavramların tasnifi, kavramların hiyerarşik olarak yapılandırılması anlamına gelir. Alt sınıf tanımlayabilme bütün dillerde gerçekleştirilebilmektedir. Tüketici parçalama, ayrık parçalama, bölümleme gibi daha karmaşık işlevleri sağlayan yapılar ise kısmen desteklenmektedir.

Tablo 1: Ontoloji biçimleme dillerinin değerlendirilmesi

	RDF(S)	OIL	DAML+OIL	OWL
Kavramlar				
Öznitelikler				
Örnek öznitelikleri	+	+	+	+
Sınıf öznitelikleri	±	-	±	±
Öznitelik Kısıtları				
Tip kısıtlamaları	+	+	+	+
Eşlenme derecesi kısıtları	-	+	+	+
Kavramların Tasnifi				
Alt sınıf tanımlayabilme	+	+	+	+
Ayrık parçalama	-	+	+	+
Tüketici parçalama	-	±	±	±
Bölümler	-	+	+	±
İlişkiler				
İkili ilişkiler	+	+	+	+
n-li ilişkiler	±	±	±	±
İlişki hiyerarşileri	+	+	+	+
Bütünlük kısıtları	-	-	-	-
Fonksiyonlar				
İkili fonksiyonlar	-	+	+	+
n-li fonksiyonlar	-	-	-	-
Diğer Bileşenler				
Kurallar	-	-	-	-

Tüketici parçalamada alt sınıfların birleşimleri üst sınıfı oluşturur. Örnek olarak, "Kadın" ve "Erkek" sınıfı "İnsan" sınıfını oluşturur. "İnsan" sınıfına ait bütün örnekler "Kadın" veya "Erkek" sınıfının örneği olmak zorundadırlar.
Ayrık parçalamada ise alt sınıfların birleşimi üst sınıfı oluşturmak zorunda değildir. Üst sınıfın bu alt sınıfların örneği olmayan örnekleri de bulunabilir. "Kuşlar" ve "Sürüngenler", "Omurgalılar" sınıfının alt sınıfı olmalarına rağmen "Omurgalılar" sınıfının kuş ve sürüngen örneği olmayan örnekleri de bulunmaktadır (örneğin "Memeli" sınıfına ait örnekler).

Bölümler, herbiri ayrık; birleşimleri ise üst sınıf olan sınıflardır. Tüketici parçalamaya gösterdiğimiz örnek bölümler içinde geçerlidir. "Kadın" ve "Erkek", "İnsan" sınıfının bölümleridir. Tüketici parçalama ile bölümler arasındaki fark, tüketici parçalamada üst sınıfın parçalandığı alt sınıfların kesişimi boş küme olmak zorunda değildir. Bölümlerin kesişimi ise boş kümedir.

Biçimsel ontoloji dillerinin anlatım gücünü gösterebilmek için Şekil 4'teki örnek ontoloji OWL diliyle modellenmektedir.

```
<owl:class rdf:ID = "Memeli"/>
```

İlk olarak "Memeli" sınıfı tanımlandı. Biçimsel dillerin hiyerarşisini gösteren Şekil 3 incelenirse OWL dilinin RDF ve RDFS dillerine ait dil yapılarını kullanabildiği görülür.

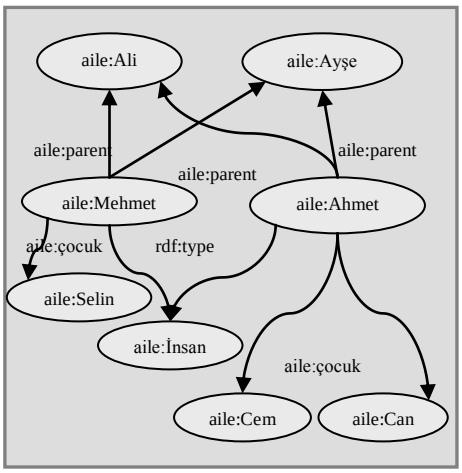
```
<owl:class rdf:ID="İnsan">
    <rdfs:subClassOf rdf:resource=#Memeli">
</owl:class>
```

İkinci adımda, "İnsan" sınıfı tanımlandı ve bu sınıfın "Memeli" sınıfının alt sınıfı olduğu belirtildi. Bundan sonraki adım "Memeli" sınıfına ait "çocuk" ve "ebeveyn" özniteliklerinin[2] tanımlanmasıdır. Burada OWL dilinin eşlenme derecesi kısıtı tanımlayabilme yeteneğini gösterebilmek için "ebeveyn" özniteliği üzerine bir eşlenme derecesi kısıtı tanımlanmaktadır. Buna göre bir "Memeli" örneğinin anne ve baba olmak üzere iki tane "ebeveyn" özniteliği vardır.

```
<owl:ObjectProperty rdf:ID="çocuk">
    <rdfs:domain rdf:resource="#Memeli"/>
    <rdfs:range rdf:resource="#Memeli"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="ebeveyn">
    <rdfs:domain rdf:resource="#Memeli"/>
    <rdfs:range rdf:resource="#Memeli"/>
    <owl:cardinality
    rdf:datatype ="&xsd;nonNegativeInteger
    > 2 </owl:cardinality>
</owl:ObjectProperty>
```

[2] Owl dili, öznitelikleri "ObjectProperty" ve "DatatypeProperty" olarak iki gruba ayırır. "ObjectProperty" önalan ve ardalanı bir sınıf olan öznitelikleri belirtir. "DatatypeProperty" ise önalanı bir sınıf ardalanı ise bir sınıf örneği olan özniteliklerdir.



Şekil 6: Aile ontolojisine ait sınıf örnekleri

Ontolojilerde sınıf örneklerinin de bulunduğu belirtilmişti. Aile ontolojisine, Şekil 6 doğrultusunda eklenen sınıf örnekleri aşağıda gösterilmektedir.

```
<İnsan rdf:ID="Ahmet">
    <parent rdf:resource="#Ali"/>
    <parent rdf:resource="#Ayşe"/>
    <child rdf:resource="#İpek"/>
</İnsan>
<İnsan rdf:ID="Mehmet">
    <parent rdf:resource="#Ali"/>
    <parent rdf:resource="#Ayşe"/>
    <child rdf:resouce="#Cem"/>
    <child rdf:resouce="#Zeynep"/>
</İnsan>
```

Üzerinde durulması gereken bir diğer nokta da "Ali" ve "Ayşe"nin "Memeli" sınıfına ait örnekler olduğunu açıkça belirtmeye gerek olmamasıdır. Çünkü, "ebeveyn" niteliğinin ardalanı "Memeli" sınıfı olarak belirtilmişti. Bu durumda "Ali" ve "Ayşe", "Memeli" sınıfına ait örneklerdir. Benzer şekilde "İpek", "Cem" ve "Zeynep", "Memeli" sınıfına ait örneklerdir. Çünkü, "çocuk" niteliğinin ardalanı "Memeli" sınıfı olarak belirtilmiştir.

AEGONT Projesi kapsamında, RDF(S), OIL, DAML+OIL ve OWL dilleri ile yazılan ontolojiler için bir ontoloji ortamı geliştirilmektedir. Ontoloji ortamı, ontolojilerin yaratılmasına, görüntülenmesine ve sorgulanmasına olanak sağlayacaktır. Bu bağlamda, yaratılan ontolojilerin geçerlik

denetiminin gerçekleştirilmesi ve sorgulara doğru ve eksiksiz yanıtlar döndürülmesi için ontolojilerde açıkça belirtilmiş bilgiyi ve ontoloji dilinin mantıksal kurallarını kullanarak yeni bilgiler elde edilmesi gerekmektedir.

Bu gereksinimlerin karşılanabilmesi açısından proje kapsamında belirtilen dillerle çalışabilecek bir çıkarsama motoru geliştirilmektedir. Kullanılan ontoloji dilinin anlatım gücü yükseldikçe, çıkarsama düzeneği zorlaşmaktadır. Desteklenmesi planlanan dillerin, anlatım gücü incelendiğinde kullanılacak mekanizmanın karmaşık bir yapıda olacağı anlaşılmaktadır.

3. Ontoloji Araçları

Bir ontoloji yaratmak, zor ve zaman alıcı bir iştir ayrıca ontolojinin yaratılacağı dili iyi bilmeyi gerektirir. Bu işlemi kolaylaştırmak için 1990'lı yılların ortalarına doğru ontoloji geliştirme araçları ortaya çıktı. Bu araçlar kullanıcılara ontoloji yaratılması, bütünlük testlerinin gerçekleştirilmesi ve dökümantasyon aşamalarında yardımcı olmak amacıyla geliştirildi. Daha sonraları bu araçlar gelişti ve çeşitlendi. Gomez-Perez'in sınıflandırmasına göre bu araçlar aşağıdaki gibi gruplanabilir [13]:

- Ontoloji geliştirme araçları, sıfırdan bir ontoloji yaratmanın yanında varolan ontolojiler üzerinde değişiklik yapma, farklı dillerde ve biçimdeki ontolojileri içeri/dışarı aktarma, ontolojileri grafiksel olarak gösterme gibi işlevler sunar.

- Ontoloji değerlendirme araçları, ontolojilerin diğer bilişim sistemleriyle uyumluluğunu sağlar.

- Ontoloji birleştirme araçları, aynı önalan üzerindeki farklı ontolojilerin birleştirilmesine olanak tanır.
- Ontoloji tabanlı ek açıklama araçları ile Web sayfaları içindeki ontoloji ek açıklamaları ontolojilere bağlı olarak güncellenir. Ek açıklama araçlarının çoğu anlamsal Web fikrinin popülerleşmesiyle ortaya çıkmıştır.

- Ontoloji sorgulama araçları ve çıkarsama motorları olarak değerlendirilen araçlar, ontolojilerin doğru ve eksiksiz olarak sorgulanmasını sağlar. Ayrıca bu araçlar, ontolojilerin görüntülenmesi ve yaratılması aşamasında, ontolojilerde açıkça belirtilmiş bilginin yanısıra çıkarsama yoluyla bulunabilecek bilgileri de görüntüleyerek kullanıcıya yardımcı olur.

- Ontoloji öğrenme araçları, doğal dildeki metinlerin yanısıra veritabanları ve yarı-biçimsel kaynakları ontolojilere çeviren araçlardır.

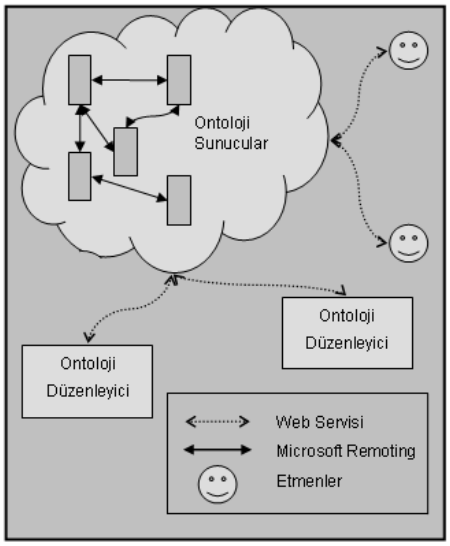
Anlamsal Web'in gerçekleşebilmesi için ontolojilerin sıradan kullanıcılar tarafından rahatça yaratılabilmesi ve ontolojik sorgulamaların Web üzerinde gerçekleştirilebilir olması gerekmektedir. Ayrıca ontoloji tabanlı ek açıklama araçlarının gelişmesi ve tarayıcılara eklenmesi ile birlikte Web, bilgiye daha kolay ulaşılabilen bir ortam olacaktır.

4. AEGONT Ontoloji Ortamı

AEGONT Ontoloji Ortamı, Anlamsal Web için gereken temel bileşen olan ontolojilerin yaratılması, saklanması ve sorgulanması ile ilgili tüm işlevler için bir çözüm sunmayı hedeflemektedir.

AEGONT Ontoloji Ortamı'nda ana bileşen Ontoloji Sunucu'dur. Tüm işlemler Ontoloji Sunucu üzerinde gerçekleştirilir. Ontoloji sunucular kendi aralarında eşler arası bir ağ kurarlar. Bu ağ üzerindeki her ontoloji sunucu, diğer ontoloji sunuculara ve ontoloji ile ilgili hizmet almak isteyen etmenlere servis sunar.

AEGONT Ontoloji Sunucusu'nun verdiği tüm hizmetlere Web servisleri yolu ile ulaşılır. Web servisleri bu hizmetlerin her platformdan ulaşılabilir olmasını sağlar. Örneğin, Pocket PC işletim sistemini kullanan bir avuç içi bilgisayar ya da Solaris işletim sistemini kullanan bir iş istasyonundan da aynı Web servisi kullanılır. Bu sayede ontoloji geliştirme araçları, Web servislerini kullanan birer arayüz olur. Buna örnek olarak AEGONT Ontoloji Düzenleyici'si gösterilebilir. AEGONT Ontoloji Ortamı'nda Ontoloji Düzenleyici'nin üç farklı arayüzü vardır. Bunlardan birincisi bir Windows uygulamasıdır. İkincisi bir ASP.NET uygulaması ve üçüncüsü de bir Visual Studio .NET eklentisidir.



Şekil 7: AEGONT Ontoloji Ortamı

Daha sonra geliştirilecek olan uygulamalar da, aynı şekilde Web servislerinin bileşen olarak kullanılması yolu ile hızlı bir şekilde geliştirilebilir.

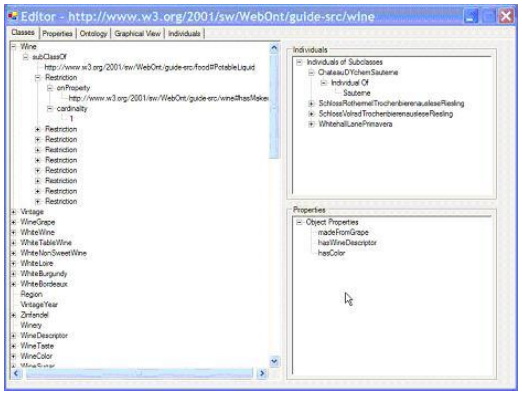
4.1. Ontoloji Düzenleyici

AEGONT Ontoloji Ortamı'nda yeni ontoloji geliştirmek ve varolan ontolojileri düzenlemek için Ontoloji Düzenleyici'si kullanılır (Şekil 8). Ontoloji Düzenleyici, Ontoloji Sunucu tarafından sunulan Web servislerini kullanır. Diğer bir deyişle Ontoloji Düzenleyici, bu servisleri kullanmak için yaratılan bir arayüzdür.

Kullanıcılar, Ontoloji Düzenleyici yardımıyla aşağıdaki işlemleri yapabilirler:

- *Ontoloji Yaratma :* Kullanıcılar yeni bir ontoloji yaratabilirler. Yaratılan ontoloji, ontoloji sunucuya kaydedilene kadar yerel olarak saklanır.
- *Ontoloji Değiştirme :* Kullanıcılar, yerel bir kopya üzerinde ya da doğrudan sunucu üzerinde değişiklik yapabilirler. Bu değişikliklere örnek olarak bir sınıf ekleme, silme ya da tanımını değiştirme verilebilir.
- *Ontoloji Silme :* Artık kullanılmayan bir ontoloji, sunucu üzerinden silinebilir.

Bütün bu işlemleri yaparken kullanıcıların bu haklara sahip olup olmadığı kontrol edilir.



Şekil 8: Ontoloji Düzenleyici

4.2. Ontoloji Sunucu

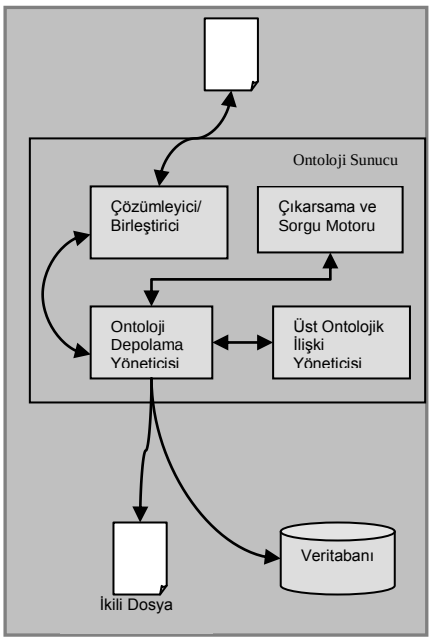
Ontoloji Sunucu, ontolojilerin saklandığı, yönetilebildiği, sorgulanabildiği bir bilgi tabanıdır. Ontoloji Sunucu'da oturum açan kullanıcılar, izinleri doğrultusunda sunucudaki ontolojiler üzerinde değişiklik yapabilirler.

Ontoloji sunucularda, ontolojiler ve aralarındaki ilişkiler, ilişkisel veri tabanında saklanır. Bu veritabanı, iki şekilde oluşturulabilir. Birinci yolda, ontolojiler veritabanında RDF üçlüleri şeklinde tutulurlar. Bu yolda ontolojilerin içindeki kavramların tipleri veritabanının yapısına yansımaz. İkinci yolda ise ontolojiler veritabanında bir şema kullanılarak tutulurlar. Eğer bir kavram sınıfsa, sınıflar tablosunda tutulurken; bu sınıfa ait olan örnekler de ayrı bir tabloda tutulurlar.

Ontoloji Sunucu'nun dört adet temel bileşeni bulunur:

- RDF Çözümleyici/Birleştirici
- Ontoloji Depolama Yöneticisi
- Üst Ontolojik İlişki Yöneticisi
- Çıkarsama ve Sorgu Motoru

Bu bileşenlerin dışında diğer etmenlere ve ontoloji sunuculara hizmet vermek için iki tip arayüz bulunur. Bunlardan birincisi "Microsoft Remoting" teknolojilerini kullanarak servis sunarken, ikincisi Web servisleri ile bu hizmetleri sağlar.



Şekil 9: Ontoloji Sunucu Bileşenleri

4.2.1. RDF Çözümleyici/Birleştirici

RDF çözümleyici/birleştirici, ontolojilerin sisteme dahil edilmesi ve taşınması sırasında ontolojileri istenen biçime dönüştürür. Bir kullanıcı bir ontolojiyi açmak istediğinde, dosyadan okunan ontolojinin RDF üçlülerinden oluşan bir çizgeye dönüştürülmesi ve bu çizgeden tekrar desteklenen bir biçime çevrilmesi işini çözümleyici/birleştirici yapar.

Çözümleyici XML ve N3 biçimlerini destekler [14]. Ayrıca bu iki biçimden başka N3'ün bir alt kümesi olan ve RDF ayrıştırıcılarını test ederken doğru sonuçların gösterimi için kullanılan NTriple biçimi de desteklenir

XML biçimi, Web üzerinde ontolojilerin taşınması sırasında kullanılan standart gösterim biçimidir. Aynı zamanda, XML biçimini kullanan diğer yazılımlarla da uyumluluğu sağlar.

N3 dili, RDF/XML sözdizimine göre daha okunabilir ve daha kısa bir seçenektir dahası bu dilin anlatım gücü daha yüksektir. Veri ve mantık bir dilde ifade edilebilir, kurallar RDF ile ifade edilememesine karşın N3 dili ile ifade edilebilir.

4.2.2. Ontoloji Depolama Yöneticisi

Ontoloji Depolama Yöneticisi, ontolojilerin daha sonra kullanılmak üzere veritabanına ya da bir dosyaya yazılması görevini yapar.

Ontolojilerin, dosyadan okunup çözümlenmesinden ya da kullanıcı tarafından yaratılmasından sonra bellekte duran ontoloji çizgesi Ontoloji Depolama Yöneticisi tarafından kullanıcının seçmiş olduğu uygun ortama aktarılır.

Veritabanında, ontolojiler iki tabloda saklanır. Birinci tabloda kaynakların URI'leri, ikinci tabloda da bu URI'lerin oluşturduğu üçlüler bulunur. Bu üçlülerin çıkarsama ve sorgu motoru tarafından değerlendirilmesiyle sorgular cevaplanır.

Kullanıcı, veritabanında bulunan bir ontolojiyi dosyaya kaydetmek istediğinde de; ontoloji, veritabanından alınarak RDF çözümleyici yardımıyla istenen RDF biçimine dönüştürüldükten sonra dosyaya yazılır.

4.2.3. Üst Ontolojik İlişki Yöneticisi

Web'in dağıtık ve değişken bir ortam olması nedeniyle Web ontolojileri geleneksel ontolojilerden farklıdırlar.

Bir kullanıcı Web ontolojisi yaratırken sıfırdan bir ontoloji yaratmaz. Varolan bir ontolojiyi temel alıp, ona ihtiyaç duyulan kavramları ekleyerek, diğer bir deyişle ontolojiyi genişleterek; yeni ontolojiler yaratır [15]. Ontolojiler bu şekilde yaratıldığında, ontoloji geliştirme süresi tekrar kullanım sayesinde kısalır.

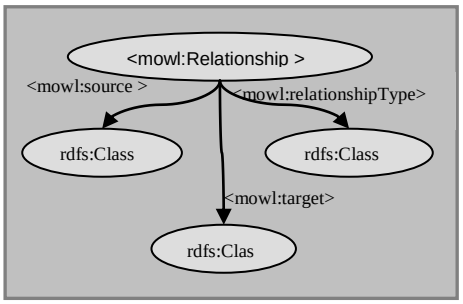
Web ontolojilerinin gücü tekrar kullanım sayesinde açığa çıkar. Bu da ontolojiler arasında bağlantılar kurulmasını gerektirmektedir. Anlamsal Web ontolojileri, tıpkı günümüzdeki Web sayfaları gibi aralarındaki bağlantılar olmadan düşünülemez [16].

Bu bağlantıların bir üst seviyede gösterilmesi ve saklanması ontoloji yaratım sürecinde ve daha sonra ontolojilerin kullanımı sırasında faydalı olur. Örnek olarak, bir kullanıcı belirli bir alandaki ontolojiler arasında en çok yeniden kullanılan ontolojiyi bulmak istediğinde bu verilerden faydalanır. Ontolojiler arası bağlantılar, bir veritabanında saklanır.

4.2.3.1. Üst Ontoloji Dili (MOWL)

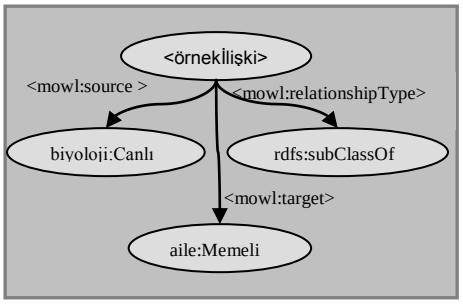
Proje kapsamında geliştirilmekte olan üst ontoloji dili olan MOWL(Meta Web Ontology Language), birbiriyle ilişkili ontolojiler arasındaki bağlantıları tutar. Böylece ontolojiler arasındaki bağlantılar da işlenebilir bir hale gelir. Ontolojilerin yaratılması ve bakımı sırasında da bu bağlantılar kullanılır.

MOWL'da ontolojik ilişkiler "`Relationship`" yapısı ile tanımlanır. Herhangi bir "`Relationship`" örneğinin "`domain`" niteliği ilişkideki etken tarafı, "`range`" niteliği ilişkideki edilgen tarafı, "`relationshipType`" niteliği de ilişkinin cinsini belirtir. Şekil 10'da "`Relationship`" yapısının çizgesi gösterilmektedir.



Şekil 10: mowl:relationship yapısı

Örneğin, aile ontolojisindeki "`Memeli`" sınıfı, biyoloji sınıfındaki "`Canlı`" sınıfının alt sınıfıdır ilişkisini saklamak için yazılan "`mowl:Relationship`" örneğinin çizgesi Şekil 11'de gösterilmektedir.



Şekil 11: Mowl örneği

4.2.4. Çıkarsama ve Sorgu Motoru

Bir ontoloji dökümanında açıkça belirtilmiş bilgiyi kullanarak yeni bilgilere ulaşmaya çıkarsama denmektedir. Çıkarsama yaparken açıkça belirtilmiş bilgi ve bu bilgilerden nasıl çıkarsama yapılacağını gösteren kurallar kullanılır.

Kurallar, $C \Rightarrow D$ şeklinde tanımlanan mantıksal cümlelerdir. Kural cümlelerinde, $\Rightarrow$ sembolünde önceki kısım kuralın sol el tarafı olarak adlandırılır ve burada özne, yüklem ve nesneden oluşan bir atomik üçlü bulunur. Kuralın sağ el tarafında ise bir atomik üçlü veya birçok atomik üçlüye $\wedge$ operatörünün uygulanmasından oluşan bir yapı bulunur. Kurallar, kullanılan ontoloji dilinin dil yapılarıyla ilgili olabildikleri gibi kullanıcıların ontolojilere dolaylı yoldan bilgi eklemesini de sağlayabilirler.

Birinci kümedeki kurallar, aynı dille yazılmış bütün ontolojiler için standarttır. Buna örnek olarak, alt sınıf tanımlayan dil yapısı ile ilgili bir kural aşağıda gösterilmiştir.

$$\forall A \forall B \; subClassOf(A, B) \wedge subClassOf(B, A) \Rightarrow equivalentClass(A, B)$$

Ontoloji biçimleme dilleri ile kural tanımlanamamaktadır (Tablo 1). Bu nedenle AEGONT projesinde kural tanımlamak için N3 dili kullanılmaktadır. Yukarıdaki kural N3 dili ile ifade edilirse:

```
{? A rdfs : subClassOf  ? B.
? B rdfs : subClassOf  ? A}
⇒ {? A owl : equivalentClass  ? B}.
```

İkinci kümedeki kurallar, kullanıcıların ontolojilere dolaylı yoldan bilgi eklemesini sağlar. Böylelikle farklı kullanıcılar aynı ontolojileri farklı amaçlarla kullanabilir. Kullanıcılar, bu ortak ontolojileri kendi amaçlarına uygun hale getirmek için ontoloji üzerinde doğrudan değişiklik yapmak yerine kurallar yardımıyla dolaylı yoldan değişiklik yapar. Aynı ontoloji, aynı sorgulara kurallar farklı olduğunda değişik yanıtlar döndürür. Çünkü aynı ontoloji üzerinde farklı kurallarla çalışan çıkarsama

mekanizması farklı bilgiler üretir. Böylelikle kullanıcılar ontolojiler üzerinde dolaylı yoldan değişiklik yapmış olurlar.

Örnek olarak, Aile ontolojisi, aile ilişkileri ile ilgili kurallar eklenirse kullanıcının amacına göre genişletilmiş olur ve sorgu motoru, aile ilişkileri ile ilgili sorgulara da yanıt döndürür. N3 diliyle ifade edilmiş, aile ilişkilerini gösteren kurallar aşağıda görülmektedir.

```
{? A aile : ebeveyn  ? B.
? C aile : ebeveyn  ? B}
⇒ {? A aile : kardeş  ? C}

{? X  aile : kardeş  ? Y.
? X  aile : çocuk  ? Z.
? Y aile : çocuk  ? W}
⇒ {? Z  aile : kuzen  ? W}.
```

Bu kuralların tanımlanmasından sonra aile ontolojisine "`kardeş`" ve "`kuzen`" ilişkileri eklenmiş olur. Aşağıda AEGONT çıkarsama motorunun "`aile:?X aile:kuzen aile:Selin`" - "Selin'in kuzenleri kimdir?", sorgusuna döndürdüğü yanıt görülmektedir. Görüldüğü gibi AEGONT projesinde, çıkarsama motoruna yöneltilen sorgular da N3 diliyle ifade edilir.

```
Yes
? X   Cem
? X   Can
```

Projede çıkarsama yapabilmek için iki temel mekanizma olan "geriye doğru zincirleme" ve "ileriye doğru zincirleme" kullanılmıştır. Ontolojideki bütün cümleler ve bütün kurallar kullanılarak elde edilebilecek bütün çıkarsamaların yapılmasına "ileriye doğru zincirleme" denir. Buna alternatif olarak ispatlanması gereken bir cümlenin gerektirdiği bütün çıkarsamaların yapılmasına "geriye doğru zincirleme" denir. İleriye doğru zincirleme genellikle ontolojiye yeni bir bilgi girildiğinde bu bilginin çıkarsama sonuçlarının ontolojiye eklenmesi amacıyla kullanılır. Geriye doğru zincirleme ise bir sorgunun yanıtlanması amacıyla kullanılır.

Klasik veritabanı sistemleri, "kapalı dünya" sistemlerdir. Kapalı dünya bir sistemde, sistemde varolmayan bilgilerin yanlış olduğu kabul edilir. Örneğin sistemde, "Murat'ın Cansu ve Poyraz adlı çocukları vardır" bilgisi varsa, Murat'ın sisteme girilmemiş çocukları yok sayılır ve "Murat'ın iki çocuğu vardır." çıkarsamasına varılır.

Ontolojiler ise "açık dünya" sistemlerdir. Açık dünya bir sistemde, sisteme girilmemiş bilgilerin yanlış olduğu sonucuna varılmaz. Bu bilgiler eksik veya yanlıştır. Bir önceki örnekte, sistem açık dünya ise ve sistemde Murat'ın çocukları ile ilgili bir sayı kısıtı yoksa "Murat'ın iki çocuğu vardır" sonucuna varılamaz.

Bu açıdan, iki sistem üzerinde çalışan çıkarsama düzeneklerinin, farklı olacağı açıktır. Kavramsal diller üzerinde gerçekleştirilecek çıkarsama mekanizmasının, doğru ve eksiksiz bir biçimde çalışabilmesi için klasik "geriye doğru zincirleme" ve "ileriye doğru zincirleme" tekniklerine birtakım ek yeteneklerin eklenmesi gerekmektedir. Bu ek yeteneklerin ne olduğu ve örnekleri aşağıda alt başlıklar halinde sunulmuştur.

4.2.4.1. Durum Analizi

Durum analizinin ne olduğunu ve neden gerektiğini anlatabilmek için Yunan mitolojisine ait bir hikayeden alıntı yapılmaktadır.

Bu hikayede, Oedipus babasını öldürüp, annesiyle evlenir ve Polyneikes adında bir çocukları olur. Daha sonra Polyneikes'ın Thersandros isimli bir çocuğu olur. Oedipus baba katilidir, Thersandros'un ise baba katili olmadığı bilinmektedir. Sisteme bu bilgiler girildikten sonra "*Iokaste'nin, kendisi baba katili olan fakat çocuğu baba katili olmayan bir çocuğu*" olup olmadığı sorgusu yöneltiliyor.

```
çocuk(Iokaste, Oedipus)
çocuk(Oedipus, Polyneikes)
babaKatili(Oedipus)
çocuk(Iokaste, Polyneikes)
çocuk(Polyneikes, Thersandros)
¬babaKatili(Thersandros)
```

Kapalı dünya bir sistemde çıkarsamanın nasıl yapıldığına bakılırsa, sistemde Iokaste'nin iki çocuğu vardır. Birinci çocuk, Oedipus baba katilidir.

Oedipus'un Polyneikes adında bir çocuğu vardır. Fakat Polyneikes'ın baba katili olmadığı söylenmemiştir. Bu durumda aranan çocuk Oedipus değildir. İkinci çocuk, Polyneikes'ın ise baba katili olduğu bilgisi olmadığı için, Polyneikes' ta aranan çocuk değildir. Bu durumda sistem olumsuz sonuç döndürür.

Fakat doğru yanıt bu değildir. İki durum göz önüne alınmalıdır. Birinci durumda Polyneikes'ın baba katili olduğu kabullenilir. İkinci durumda ise Polyneikes'ın baba katili olmadığı varsayılır. Birinci durumda aranan çocuk Polyneikes'tır çünkü Polyneikes baba katilidir ve Thersandros adında baba katili olmayan bir çocuğu vardır. İkinci durumda sonuç Oedipus'tur çünkü Oedipus baba katilidir ve Polyneikes adında baba katili olmayan bir çocuğu vardır. Her iki durumda da (sonuçlar farklı çıksa da) sistem böyle bir çocuk olduğu sonucunu bulur ve olumlu yanıt döndürür.

Açık dünya sistemlerde, sistemde belirtilmemiş durumlar için durum analizi yapan çıkarsama mekanizmalarına gereksinim vardır. Bu nedenle ontolojiler üzerinde çıkarsama yapmak, klasik veritabanları üzerinde sorgu yanıtlamaktan daha karmaşıktır.

4.2.4.2. Tekdüze Olmayan Çıkarsama

Çıkarsama düzeneğinin tekdüze olmayan çıkarsama durumlarını nasıl ele alacağı üzerinde durulması gereken bir diğer noktadır. Tekdüze olmayan çıkarsamaya sıkça gösterilen bir örnek, bütün kuşların uçabilmesine ve devekuşunun da bir kuş olmasına rağmen devekuşunun uçamamasıdır. Sisteme bütün kuşların uçabildiğine dair bir kural ve devekuşunun bir kuş olduğu ve uçamadığı bilgileri girildiğinde oluşan çelişkinin nasıl ele alınması gerektiği planlanmalıdır.

4.2.4.3. Genelleştirme

Çıkarsama mekanizmasına eklenmesi düşünülen bir diğer yetenek ise genelleştirmedir. Sisteme girilen verilerde benzerlikler ve desenler olabilir, bu durumda sistemin bu benzerliklerden yola çıkarak genel kurallar üretmesi planlanmaktadır. Bir örnekle özetlenirse, sisteme girilen insanların belirli bir oranı iki gözlü ise sistem insanların iki gözlü olduğu sonucuna varmalıdır. Genelleştirme yapılabilmesi için veriler arasındaki benzerlik oranın ne kadar olması gerektiği, sisteme ne kadar veri girilmesi gerektiği, dahası genelleştirmenin sistemin döndürdüğü yanıtların doğruluğunu ne kadar arttırdığı testlerle incelenecektir.

5. Sonuç

Bu makalede, anlamsal web ontolojilerini yaratmak ve kullanmak için gerekli servisleri sağlayan AEGONT Ontoloji Ortamı anlatılmaktadır. AEGONT Ontoloji Ortamı'nın diğer benzer uygulamalardan en önemli farkı web servislerini temel almasıdır. Aynı zamanda bu servislerin ontolojik tanımlamaları da yapılacaktır. Böylelikle bu ortamın anlamsal web servislerini kullanan, etmen çerçeveleri ile bütünleştirilmesi kolaylaşacaktır.

AEGONT projesi kapsamında gerçekleştirilen çıkarsama motoru "ileriye doğru çıkarsama" ve "geriye doğru çıkarsama" tekniklerini kullanarak klasik çıkarsama işlemini gerçekleştirmektedir. Çıkarsama motoru ontolojinin bütünlüğünü test edebilmektedir. Bu konudaki tek eksik çıkarsama motorunun tip kısıtlamalarını kontrol edememesidir. Durum analizi, monoton olmayan çıkarsama ve genelleştirme üzerinde çalışmalar devam etmektedir.

Bu projenin amacı, varolan ontoloji ortamlarına alternatif olabilecek bir ontoloji ortamı yaratmaktır. Bu sayede, ileride ortaya çıkabilecek ihtiyaçlara daha çabuk cevap verebilme imkanı olacaktır. Anlamsal Web, çok disiplinli bir araştırma alanı olduğundan bu konuyla ilgili alanlarda çalışan gruplarla ve akademisyenlerle ortak çalışmalar yapmak hedeflenmektedir.

6. Teşekkür

AEGONT Ontoloji Ortamı, Microsoft Research Cambridge araştırma laboratuvarı tarafından 2003-176 numaralı proje kapsamında desteklenmektedir.

Kaynakça

[1] Tim Bray, Jean Paoli, C.M.Sperberg-McQueen, Eve Maler, Francois Yergeau, and John Cowan, 5-11-2003. Extensible Markup Language (XML) 1.1 W3C Proposed Recommendation.

[2] David C.Fallside, 2003. XML Schema W3C Recommendation.

[3] Sowa, John F., 2000. Knowledge representation: logical, philosophical and computational foundations, Brooks/Cole Publishing Co.

[4] Frank Manola and Eric Miller, 15-12-2003. RDF Primer W3C Proposed Recommendation.

[5] Brickley, Dan and Guha, R. V., 15-12-2003. RDF Vocabulary Description Language 1.0: RDF Schema.

[6] Smith, Michael K., Welty, Chris, and McGuinness, Deborah L., 15-12-2003. OWL Web Ontology Language Guide W3C Proposed Recommendation.

[7] Berners-Lee, Tim, 1999. Weaving the Web, Harper, New York.

[8] Gruber, Thomas R., 1993. A translation approach to portable ontology specifications, Knowledge Acquisition, 5, 2.

[9] Manfred Schmidt-Schauss and G.Smolka, 1991. Attributive Concept Descriptions with Complements, Artificial Intelligence, 48.

[10] Fensel, D., Harmelen, F. van, Horrocks, I., McGuinness, D., and P.Patel-Schneider, 2001. OIL: An ontology infrastructure for the Semantic Web.

[11] Horrocks, Ian, Harmelen, Frank van, Patel-Schneider, Peter, Berners-Lee, Tim, Brickley, Dan, Connolly, Dan, Dean, Mike, Decker, Stefan, Fensel, Dieter, Fikes, Richard, Hayes, Pat, Heflin, Jeff, Hendler, Jim, Lassila, Ora, McGuinness, Deb, and Stein, Lynn Andrea, 2001. DAML+OIL.

[12] Gomez-Perez, A. and Corcho, Oscar, 2002. Ontology Specification Languages for the Semantic Web, IEEE Intelligent Systems, 17, 1.

[13] Gomez-Perez, A., Mariano Fernandez-Lopez, and Corcho, Oscar, 2003. Ontological Engineering, Springer.

[14] 2003. Semantic Web Application Platform - SWAP, World Wide Web Consortium.

[15] Staab, Steffen, 2002. Ontologies'KISSES in Standardization, IEEE Intelligent Systems, 17, 2.

[16] Hendler, James, 2001. Agents and the Semantic Web, IEEE Intelligent Systems, 16, 2.