Anlamsal Web Servislerinin Bulunması, Eşlenmesi ve Dinamik Çağrımı Üzerine Bir Durum Çalışması

Geylani KARDAŞ[1] Özgür GÜMÜŞ[2] Oğuz DİKENELLİ[3]

[1]Uluslararası Bilgisayar Enstitüsü, Ege Üniversitesi, 35100, Bornova, İzmir
[2, 3]Bilgisayar Mühendisliği Bölümü, Mühendislik Fakültesi, Ege Üniversitesi, Bornova, İzmir

[1]e-posta: kardas@merkez.ube.ege.edu.tr [2]e-posta: gumus@staff.ege.edu.tr
[3]e-posta: oguzd@staff.ege.edu.tr

Özet

Bu bildiride; anlamsal web servislerinin anlamsal eşleme sonucunda bulunması ve dinamik çağrımı üzerine yapılan çalışmaya ait tasarım ve uygulama seviyesinde elde edilen deneyimler ve bulgular yer almaktadır. Çalışma kapsamında, atomik yapıdaki web servislerinin hazırlanması, bu servislerin kendi yeteneklerini tanıtan ilgili DAML-S (DARPA Agent Markup Language-Service - DARPA Etmen Biçimleme Dili-Servis) profil dokümanlarına göre eşleme motoruna kaydı, istemcilerin uygun istek profilleri ile anlamsal web servislerini araması ve bu servislerin uygun servis çağrım altyapısına göre çağırılması ve kullanılması gerçekleştirilmiştir.

Abstract

In this paper, design and implementation experiences, gained in a case study concerning discovery and dynamic invocation of semantic web services in accordance with a semantic matchmaking process, are discussed. Related study includes development of atomic web services, registration of those services via their DAML-S (DARPA Agent Markup Language-Service) profile documents in which service capabilities are described. Discovery and invocation of those semantic web services by clients with use of request profiles and appropriate service groundings are also realized.

1. Giriş

İkinci jenerasyon Internet olarak da tanıtılmakta [1] olan *Anlamsal Web*, web sayfalarının anlam ifade eden içeriğine bir yapı getirmeyi; otonom yapıların –ki bu yapılar için en güçlü adayın yazılım etmenleri olduğu düşünülmektedir- sayfa sayfa dolaşarak, temsil ettikleri kullanıcılarının yerine karmaşık işlemler gerçekleştirebildikleri bir ortam oluşturmayı hedeflemektedir. Anlamsal Web alanında yapılan çalışmalar son zamanlarda hız kazanmakla beraber oldukça yenidir ve şu an kullanımda olan Internet'in gerçekten yazılımların okuyabileceği ve *anlayabileceği* bir forma dönüştürülmesine yönelik araştırmalarda çözülmesi gereken ciddi problemlerle karşılaşılmaktadır.

Söz konusu araştırmalardan önemli bir kısmı, var olan veya geliştirilecek olan web servislerinin bu yeni anlamsal ortama nasıl entegre edileceği ya da buna uygun olarak nasıl tasarlanacağı ve kullanıma geçirilebileceği üzerine yürütülmektedir. Tüm bunlar yapılırken aynı zamanda web servisi olarak adlandırılan bu yazılım bileşenlerinin dil ve platform bağımsız ara yüzlerinin korunmasının da sağlanması gerekmektedir.

Günümüzde kullanılmakta olan web servisleri kendilerini temsil eden ve WSDL (Web Services Description Language – Web Servisleri Tanımlama Dili) kullanılarak hazırlanan ara yüzleri sayesinde geliştirildikleri yazılım dili ve/veya ortamına bağlı kalmaksızın yine çok çeşitli ortamlarda çalışan istemci yazılımlar tarafından kullanılabilmektedirler. Örneğin Java programlama dili ile geliştirilen bir döviz kuru bilgilendirme web servisinin, WSDL'ın XML (Extensible Markup Language - Uzatılabilir İşaretleme Dili) ile şekillendirilmiş üst veri tanımları sayesinde C++ kullanılarak hazırlanan bir istemci tarafından bulunup çalıştırılması olasıdır. İstemci program WSDL'i işleyerek, sunulan servise ait operasyonu ve bu operasyon için gerekli girdi – çıktı parametrelerini öğrenir ve servisi çalıştırabilir. Ancak bu şekilde, WSDL kullanılarak tanımlanan bir servis ara yüzünün, yukarıda söz edilen anlamsal web ortamı için yetersiz kalacağı, özellikle ihtiyaç duyulan anlamsal çıkarsama mekanizmaları göz önünde bulundurulduğunda açıktır.

Web servislerinin anlamsal ortama entegrasyonu için temel yaklaşım, servis yeteneklerinin anlamsal web'in bir başka temel ve vazgeçilmez bileşeni olan bilgi koleksiyonları yani *ontoloji*ler kullanılarak modellenmesi ve ifade edilmesidir. Bir ontoloji, kavramlar arasındaki ilişkileri biçimsel (formal) olarak içeren bir dokümandır. Bu ontolojiler, RDF (Resource Description Framework – Kaynak Tanımlama Çatısı) temelli ontoloji dilleri kullanılarak geliştirilmektedirler. Bu dillere örnek olarak DAML ve W3C tarafından yakın zamanda standart olarak kabul edilen OWL [2] (Ontology Web Language – Ontoloji Web Dili) verilebilir.

Anlamsal web servisi yeteneklerinin anlamsal web ortamında temsil edilmesi ve dinamik olarak bulunup kullanılması için geliştirilen bir web servisi ontolojisi bulunmaktadır. Bu ontoloji, kendisinin geliştirilmesinde kullanılan ontoloji diline bağlı olarak DAML-S veya OWL-S adını almaktadır [3]. Arkalarında yatan tasarım modeli aynı olan iki ontolojiden, bu çalışma kapsamında DAML-S kullanılmıştır.

DAML-S ile atomik veya birleşik (composite) bir web servisi modellendiğinde aynı zamanda servise ait aşağıdaki üç tip anlamsal bilgi de modellenmiş olmaktadır [4]:
1) Servisin yetenekleri veya yapabildikleri nelerdir?
2) Servis nasıl çalışmaktadır?
3) Servis nasıl kullanılmaktadır?

Bu sorulara cevap veren ilgili anlamsal web servisi ontoloji dokümanları sırasıyla service profili (profile), servis süreç modeli (process model) ve servis altyapısıdır (grounding). Bu dokümanlar anlamsal yeteneğe sahip yapılar tarafından servis arama, bulma ve dinamik çağırma aşamalarında çıkarsama amaçlı olarak kullanılmaktadırlar. Bu servis ontolojisi yapıları ile ilgili detaylı bilgi ve kullanım esasları [5 ve 6]'da verilmiştir.

Literatürde yukarıdaki yaklaşımı destekleyen bir çok teorik alt yapı çalışması bulunmasına karşılık bunların gerçek hayatta kullanımına yönelik çalışmalar oldukça az ve yetersizdir. Biz, bu bildiride, DAML-S temelli anlamsal web servislerinin modellenmesinde ve gerçek çalışma ortamlarında kullanılmasına yönelik yapmış olduğumuz yazılım tasarım ve uygulama çalışmalarını ve elde ettiğimiz deneyimleri ortaya koymaktayız. Anlamsal web servislerinin DAML-S modeline göre hazırlanması ve kullanılmasına yönelik uyguladığımız yazılım sürecini ve şu ana kadar karşılaştığımız güçlükleri yine bu bildiride aktarmaya çalıştık.

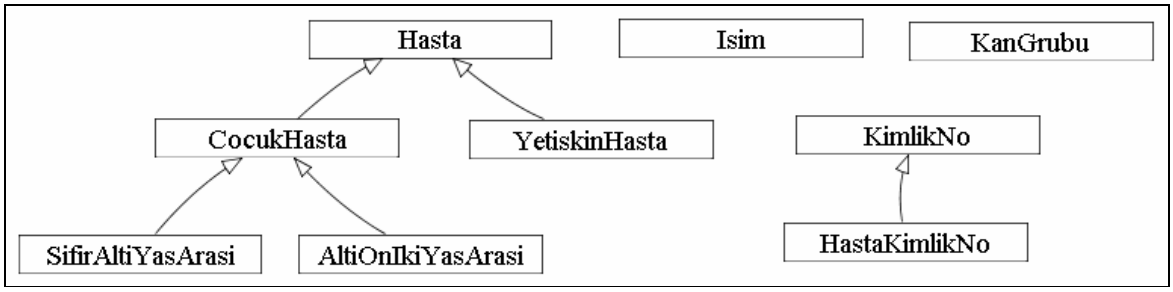
Bildirinin geriye kalan kısmı şu şekilde organize edilmiştir: İkinci bölümde anlamsal web servisleri için üzerinde çalıştığımız problem alanı ve buna uygun web servislerinin tasarlanması ve

uygulanması hakkında bilgi verilmiştir. Bunu izleyen üçüncü bölümde, yetenekleri anlamsal olarak ilan edilmiş web servislerinin bulunması için hayata geçirdiğimiz anlamsal eşleme motoru anlatılmaktadır. Dördüncü bölümde ise DAML-S' e göre modellenmiş ve ilan edilmiş web servislerinin dinamik olarak kullanılmasına yönelik yapılan çalışmalar anlatılmaktadır. Bildirinin beşinci bölümü, sonuçları ve deneyimlerden elde edilen yorumları içermektedir.

2. Anlamsal Web Servislerinin Hazırlanması

Çalışma kapsamında hazırlanan servislerin çalıştığı otonom servis ortamı sağlık alanını esas almaktadır. Sistemde ilgili parametrelere (hasta kimlik numarası, kan grubu, vb.) göre hasta bilgileri elde etmede kullanılan web servisleri yer almaktadır. Geliştirilen web servisleri günümüzde kullanılan şekilleriyle yani WSDL kullanılarak çağırılabildikleri gibi bu çalışma kapsamında tanımlanan DAML-S temelli anlamsal ara yüzleri sayesinde uygun istek profillerine göre anlamsal olarak eşlenip çağırılmaları da mümkündür. İlgili anlamsal web servislerinin çalıştırılabilmeleri için her bir servise ait DAML-S profil, süreç modeli ve altyapı ontoloji dokümanları hazırlanmıştır. Muhakeme (reasoning) yetenekleri [5]' de anlatılan çalışmaya dayanan bir eşleme motoru sistem içerisinde hayata geçirilmiştir ve bu motor, servis istemcilerinden gelen istek profillerine göre anlamsal servis yeteneği eşlemesini yerine getirmektedir.

Anlamsal web servislerinin çalıştırılabilmesi için anlamsal servislerin ve bu servisleri çağıran istemci yazılımlarının aynı ontolojiyi kullanmaları gerekmektedir. Bu nedenle servislerin kullandığı, sağlık alanı için basit bir ontoloji tasarlanmış ve DAML kullanılarak yazılmıştır. Bu ontolojinin tasarımı sırasında Protege [7] ontoloji editöründen faydalanılmıştır. Şekil 1'de tasarlanan ontoloji modeli ve içerdiği kavramlar görülmektedir:



Şekil 1. Sağlık alanına özel kavramların bulunduğu basit ontoloji modeli

Altyapı seviyesinde fiziksel servis çağırma için bu ontolojiye karşılık gelen yazılım paketinin hazırlanması da gerekmiştir. Java'da hazırlanan paket "JavaBean" [8] tasarım desenine uygun olarak gerçekleştirilmiştir ve geliştirilen ontolojideki kavramlara karşılık gelen sınıfları barındırmaktadır. Bu noktada DAML' dan Java'ya otomatik dönüşüm için şu ana kadar yapılan çalışmalar ve yazılım araçları incelenmiştir. SWEDE (Semantic Web Development Environment – Anlamsal Web Geliştirme Ortamı) içerisinde yer alan DAML2JAVA dönüştürücüsü ontolojiden Java sınıflarını ürettiğini iddia etmektedir [9]. Ancak indirilen kaynak kodu uygun Java platformu sağlanmasına karşılık derleme zamanında kaynak kodda hatalar verdiğinden bu çalışma içerisinde kullanılamamıştır. Ayrıca düzgün çalışması sağlansa bile ontoloji sınıflarının uygun desene (pattern) göre üretilmiş olması gerekir. Çünkü sınıfların SOAP (Simple Object Access Protocol – Basit Nesne Erişim Protokolü) üzerinden seri olarak transferi için fasulye eşlemelerinin (bean mapping) yer alması yani uygun "*serializer/deserialize*"'ların ya yeniden yazılması ya da uygun

şartlar altında otomatik üretilmesi gereklidir. Bu nedenle ontolojiye karşılık gelen Java paketi bir dönüşüm aracı kullanılmadan JavaBean desenine göre kodlanmıştır.

Sistem içerisinde hayata geçirilen servisler, temelde, aldıkları argümanlara göre sorgu gerçekleştiren ve sorgu sonucu döndüren birer Java programıdır. Ancak bunların web servis olarak hizmet görebilmesi için genel (public) arayüzlerinin ve bağlarının (binding) XML kullanılarak tanımlanması ve tarif edilmesi gerekmektedir.

Hazırladığımız servisler Java RMI' ya (Remote Method Invocation – Uzak Metot Çağrımı) uygun olarak *Remote*'u uzatan (extend) bir ara yüzü uygulamaktadırlar. Bunun bir avantajı hazırlanan servis bileşenlerinin uzak nesne olarak, genel web servis ortamı yanı sıra sadece RMI protokolüne uygun olarak da hizmet verebilmesidir.

İlgili web servisinin hizmet verebilmesi için bir web servis sunucusuna yerleştirilmesi gerekmektedir. Çalışmalar sırasında bu amaçla kullanılan ortam, üzerinde Axis SOAP motoru bulunan Jakarta Tomcat web sunucusudur. Axis'in burada kullanılma amacı dinamik web servisi çağrımlarımızda SOAP temelli JAX-RPC'yi (Java API for XML based Remote Procedure Call – XML temelli Uzak Yordam Çağrımı JAVA API'si) kullanılabilmemize imkan sağlamasıdır.

Hazırlanan web servisin yerleştirilmesi (deploy) aşamasında üzerinde durulması gereken en önemli konu otomatik fasulye serializer/deserializer'ların devreye girebilmesi için fasulye eşlemelerinin tanımlanmasıdır. Bunun için Axis WSDD (Web Service Deployment Descriptor – Web Servisi Yerleştirme Tanımlayıcısı) formatına uygun olarak hazırlanan yerleştirme dosyasında servis giriş-çıkış argümanlarının kullanıcı tanımlı olanlarının fasulye eşlemelerinin yer alması gerekmektedir. Aşağıda, sistemimizdeki web servislerden biri olan, girdi olarak hasta kan grubu alıp bu kan grubuna sahip hastaların listesini döndüren "*KanGrubunaGoreHastaBul*" adlı atomik web servisi için hazırlanan örnek bir WSDD dokümanı içeriği görülmektedir:

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/" xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
 <service name="urn:SaglikAlani" provider="java:RPC">
   <parameter name="className" value="tr.edu.ege.sws.saglikAlani.SaglikAlaniImpl"/>
   <parameter name="allowedMethods" value="KanGrubunaGoreHastaBul"/>
   <beanMapping qname="myNS:KanGrubu" xmlns:myNS="urn:tr.edu.ege.sws.saglikAlani.ontoloji"
    languageSpecificType="java:tr.edu.ege.sws.saglikAlani.ontoloji.KanGrubu"/>
   <beanMapping qname="myNS:KimlikNo" xmlns:myNS="urn:tr.edu.ege.sws.saglikAlani.ontoloji"
    languageSpecificType="java:tr.edu.ege.sws.saglikAlani.ontoloji.KimlikNo"/>
   <beanMapping qname="myNS:HastaKimlikNo" xmlns:myNS="urn:tr.edu.ege.sws.saglikAlani.ontoloji"
    languageSpecificType="java:tr.edu.ege.sws.saglikAlani.ontoloji.HastaKimlikNo"/>
   <beanMapping qname="myNS:Isim" xmlns:myNS="urn:tr.edu.ege.sws.saglikAlani.ontoloji"
    languageSpecificType="java:tr.edu.ege.sws.saglikAlani.ontoloji.Isim"/>
   <beanMapping qname="myNS:Hasta" xmlns:myNS="urn:tr.edu.ege.sws.saglikAlani.ontoloji"
    languageSpecificType="java:tr.edu.ege.sws.saglikAlani.ontoloji.Hasta"/>
   <operation name=" KanGrubunaGoreHastaBul ">
         <parameter name="kanGrubu" mode="IN"/>
   </operation>
 </service>
</deployment>
```

Yine bir XML dokümanı olan WSDD dosyasının yukarıdaki örneğinde hem giriş hem de çıkış argümanları kullanıcı tanımlı olduğundan fasulye eşlemeleri de dokümanda yer almıştır. Örneğin girdi parametresi olan kan grubu için yapılan eşlemede, bu parametrenin, sistem isim uzayında

"KanGrubu" adlı kavramın tipinde olduğu ve bu kavramı temsil eden Java sınıfının "tr.edu.ege.sws.saglikAlani.ontoloji.KanGrubu" olduğu belirtilmiştir. Böylelikle yukarıdaki WSDD tanımlamasını girdi olarak alan Axis SOAP motoru, buna karşılık gelen WSDL dokümanını otomatik olarak üretmekte ve servis istemcilerinin kullanımına sunmaktadır. Bu servisin, altyapı seviyesinde dinamik çağrımı sırasında istemciler hangi tip nesneleri servise seri halde göndermeleri gerektiğini ve hangilerini seri halde protokol üzerinden servis çıktısı olarak alacaklarını böylece anlayabilmektedirler.

Burada karşılaştığımız en büyük zorluk sadece servis giriş ve çıkış tiplerine ait ontoloji sınıflarının değil bu sınıflar içerisinde yer alan özelliklerden yine kullanıcı tanımlı tiplerde olanları varsa onlara ait sınıfların da fasulye eşlemelerinin yapılması gerekmektedir. Örneğin yukarıdaki tanımlamada servis dönüş değeri *Hasta* nesnesi koleksiyonudur. Ancak *Hasta* da kendi içerisinde *HastaKimlikNo*, *Isim*, vb. ontoloji kavram türlerini özellik olarak barındırdığından bu sınıfların da eşlemelerinin yapılması gerekmektedir. Hatta bu sınıfların uzattığı kullanıcı tanımlı sınıflar varsa (örneğin *HastaKimlikNo – KimlikNo* ilişkisinde olduğu gibi) süper sınıfların da (burada *KimlikNo*) eşlemeleri yapılmalıdır. Aksi takdirde servis çağrımı esnasında eşleme hatası oluşmaktadır.

Daha önce belirtildiği gibi tüm ontoloji sınıflarını JavaBean desenine uygun olarak hazırladığımızdan ek "serializer/deserializer" yazmamıza gerek kalmamıştır. Ancak tüm sınıfların eşlemelerinin yapılması zorunluluğu, yerleştirme aşamasında servis sağlayıcı için gözlenen ve bizimde bu çalışmada yaşadığımız en önemli zorluğu teşkil etmektedir.

Web Servislerinin Anlamsal Arayüzünün Hazırlanması

Hazırlanan web servislerinin anlamsal arayüzünün sağlanabilmesi için servise ait DAML-S dokümanlarının hazırlanması gerekmektedir. Bu çalışma kapsamındaki servisler atomik olarak tasarlanmış ve gerekli DAML-S dokümanları buna göre hazırlanmıştır. Dokümanlar manuel olarak hazırlanabildiği gibi halihazırdaki araçlar sayesinde yarı otomatik olarak da hazırlanabilmektedir. Örnek olarak WSDL2DAMLS [10] verilebilir. Çalışmalar sırasında kullandığımız bu araç bir servise ait WSDL dokümanını girdi olarak almakta ve servise ait servis, profil, süreç ve altyapı DAML-S dokümanlarını hazırlamaktadır. Ancak servisin özelliklerine göre bu çevirim genellikle kısmi olarak gerçekleşmektedir. Altyapı tanımlaması düzgün olarak oluşturulsa da profil ve süreç modeli çoğunlukla kısmi olarak oluşturulmakta ve dönüşüm sonrası kullanıcı müdahalesi gerektirmektedir. Tıpkı bizim servislerimizde olduğu gibi servisin XSD Complex tipinde girdi ve/veya çıktı mesajları varsa dönüşüm işlemi kısmi olarak gerçekleşmektedir. Dokümanlardaki isim uzayı ayarlarının (ontoloji isim uzayı, dokümana göre servis, profil, süreç ve altyapı dokümanlarının URL'leri, vb.) yapılması ve DAML-S'in WSDL'dan daha fazla bilgi içermesinden kaynaklanan ve bu nedenle araç tarafından eksik bırakılan yerlerin dokümanlar incelenerek doldurulması gerekmektedir. Her ne kadar çevirim sonunda yapılması gereken değişiklikler ile ilgili bir açıklama dosyası araç tarafından hazırlansa da bu dokümanda belirtilmeyen bazı değişikliklerin de yine DAML-S dosyalarında yapılması gerektiğinin farkına vardık. Örneğin servise ait *Service DAML-S* dokümanında profil, süreç ve altyapı adlarının değiştirilmesi, servis *Profile DAML-S* dokümanında servis dokümanına olması gereken bağlantının eklenmesi gibi.

3. Anlamsal Web Servisi Eşleme Motoru

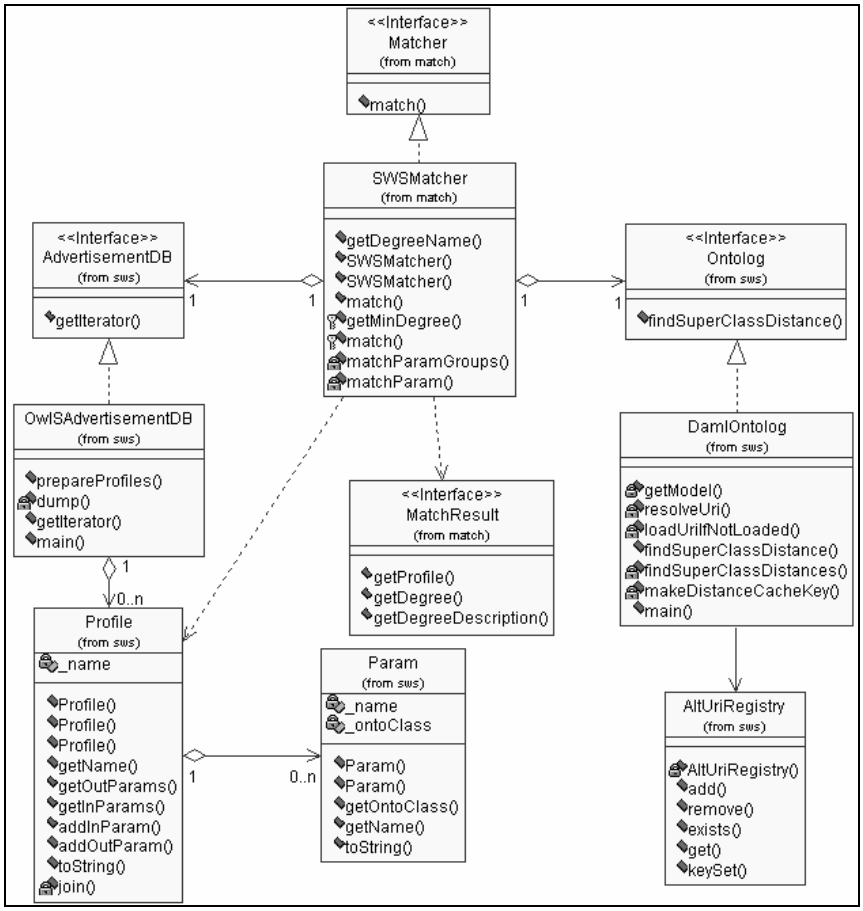
Sistem içerisinde; hazırlanan web servislerinin istemciler tarafından kullanılmak amacıyla bulunmasını ve isteğe göre eşleştirilmelerini yerine getiren bir eşleme motorunun tasarımı ve

gerçekleştirimi yerine getirilmiştir. Eşleme motorunun servis eşlemede kullandığı yetenek eşleme algoritması [5]' de önerilen işleyişe sahiptir. Şöyle ki; istemcinin servis isteğini içeren istek profili R, eşleme motoru veritabanına kayıtlanan anlamsal web servisi ilan profillerinden her biri A ile temsil edilecek olursa A ve R profilleri arasındaki eşleme, öncelikli olarak servis çıktılarının karşılaştırılması ile olmaktadır. Eşleştirme başarılarına göre sıralanmış olarak, başlıca dört eşleme düzeyi bulunmaktadır;

- *Tam (Exact)*: Eğer A bildiriminin çıktıları ile R isteğinin çıktıları birbirine denkse ($A \equiv R$)
- *Uyumlu (Plug-in)*: Eğer R isteği çıktıları A bildirimi çıktılarının alt kümesi ise ($R \subseteq A$)
- *Kapsayan (Subsume)*: Eğer R isteği çıktıları A bildirimi çıktılarını kapsıyorsa ($A \subseteq R$)
- *Başarısız (Fail)*: Eğer A bildirimi çıktıları ile R isteği çıktıları yukarıdaki durumların dışında kalıyorlarsa ($A \cap R = \varnothing$)

Bu esaslara göre profil eşlemeleri sonucunda eşit düzeydeki profiller arasında bu kez girdiler karşılaştırılır ve bunlar arasında *tam (exact)* seviyesinde eşlenenler sıralama da öne alınır. Eşleme düzeyleri için bir başka varyasyon [11]' de bulunabilir.

Şekilde 2' de nesne modeli görülen eşleme motoru, DAML-S profillerinin bildirilip saklandığı *AdvertisementDB*; eşleme işlemi yapan *Matcher* ve ontolojilerden DAML-S *subClassOf* ilişkilerini takip edip servis parametre ilişkilerinin çıkarsamasını yapan *Ontolog* bileşenlerinden oluşmaktadır.



Şekil 2. Anlamsal web servisi yetenek eşleme motorunun nesne modeli

Eşleme motorunun çalışma senaryosu aşağıdaki gibi özetlenebilir:

- Web servisi sunmak isteyen taraflar *AdvertisementDB* bileşenine, sunmak istedikleri servislerin DAML-S servis profillerini kaydederler.
- Bir servise ihtiyacı olan taraf, servis isteğini ifade eden bir istek profili hazırlayıp *Matcher*'a gönderir.
- *Matcher* gelen isteği *AdvertisementDB*'de kaydedilmiş olan profillerle teker teker karşılaştırır ve istenen koşullara uyan servis profillerinin listesini isteği yapana döndürür.
- İsteği yapan taraf dönen sonuçlar arasından istediği servisi seçerek tanımlı altyapı aracılığıyla çalıştırır.

Matcher arayüzünün örnek bir implementasyonu olan *SWSMatcher*, istek olarak gelen servisin çıktı ve girdi parametrelerini mevcut servislerin çıktı ve girdi parametreleri ile karşılaştırır. İsteğin çıktı parametrelerini istenen minimum derecede (*tam, uyumlu* veya *kapsayan*) karşılayabilen servisler çalıştırılmaya aday servisler olarak belirlenir. Algoritmada da anlatıldığı gibi girdi parametrelerinin karşılaştırılması, seçilmek için değil seçilen eşit dereceli servisler arasında sıralama yapmak için kullanılmaktadır.

İki parametrenin karşılaştırılması aşamasında ilgili parametrelerin ontoloji sınıflarının birbiri ile olan ilişkisi önemlidir. Bu ilişkiyi bulan bileşen, basit bir *çıkarsayıcı (reasoner)* olarak tasarladığımız *Ontolog*'dur. *Matcher* karşılaştırdığı parametrelerin ontoloji sınıfları arasındaki ilişki uzaklığını *Ontolog* bileşenine sorar. *Ontolog* bileşeni verilen sınıfların bulunduğu ontolojiyi daha önce yüklenmediyse yükleyip istenen sınıfların *subClassOf* ilişkilerini sınıf hiyerarşi ağacında takip eder ve parametreler arasında ilişki varsa aralarındaki uzaklığı bulur. Bu arama sırasında DFS (Depth First Search – Derinlik Öncelikli Arama) uygulayan *Ontolog* aynı zamanda entegre bir önbelleğe (cache) sahiptir. Böylelikle ard arda gelen aynı tip ilişki belirleme istekleri önbellekten karşılanarak performans arttırılmaktadır. Ontoloji dokümanlarının ayrıştırılıp işlenmesinde ve sorgulamalarda, anlamasal web uygulamaları geliştirmede kullanılan JENA [12] çatısı ve buna bağlı RDQL (RDF Query Language – RDF Sorgu Dili) kullanılmıştır.

Matcher, *Ontolog*'dan dönen uzaklık değerine göre parametre sınıfları arasındaki ilişkinin (*tam, uyumlu*, *kapsayan* veya *başarısız*) ne olduğuna karar verir. İki profil arasındaki eşleme sonucunu isteğin en "kötü" eşlenen çıktı parametresi belirler.

4. Anlamsal Web Servislerinin Dinamik Çağrılması

DAML-S'e uygun olarak modelleyip gerçekleştirimini yaptığımız anlamsal web servislerinin dinamik olarak çağrılması için servis istemcilerinde, SOAP protokolünün kullanımını sağlayan JAX-RPC'den yararlandık.

Anlamsal web servislerimizin altyapı (grounding) tanımlamalarında, DAML-S'in *WsdlAtomicProcessGrounding* ontoloji elemanı sayesinde servislerin WSDL dokümanlarına bağlantılarını gösterebildik. Böylelikle SOAP protokolü aracılığıyla dinamik anlamsal web servisi çalıştırmayı başarmamız mümkün oldu.

Servis istemcisi olan bir yazılım, eşleme motorundan dönen ilan profillerinden istediği birini seçip bu profili sunan servisi çalıştırmak istediğinde öncelikle bu profil dokümanı içinde, karşılık gelen altyapı dokümanına ait URI'yi bulmaktadır. Bu altyapı dokümanından da öncelikle bu servisin

hangi gerçek bağlantı protokolünü desteklediğini algılamakta ve eğer altyapı WSDL ise bu servise ait WSDL dokümanına ulaşmaktadır. Burada karşılaştığımız en büyük güçlük anlamsal web servisinin profil dokümanında tanımladığı ve ihtiyaç duyduğu servis girdilerine, asıl servis çağrım zamanında yenilerinin eklenmesi durumunda görülmüştür –ki bu durum özellikle bileşik (composite) yapıdaki anlamsal web servislerinin kullanılmasında sıklıkla görülmektedir. Eğer dinamik zamanda kullanıcıdan istenecek yeni servis parametreleri basit XML şema veri tiplerinde değilse yani nesne tipindeyse bunların dinamik zamanda oluşturulup içlerinin kullanıcıdan alınan verilerle doldurulması oldukça güç bir işlemdir. İnceleme fırsatı bulduğumuz diğer hiçbir çalışmada bu probleme bir çözüm getirilmemiştir. Biz bu çalışma kapsamında, atomik web servisleri için bu problemin giderilmesinde *yansıma* mekanizmasına dayalı izleyen çözümü getirdik: Servisin WSDL dokümanında dinamik zamanda yüklenmesi gereken sınıflara ait URI'ler bulunmaktadır. Bu URI'ler kullanılarak bu sınıflar çalışma zamanında istemci yazılım tarafında yüklenirler. Sınıf yükleme işleminden sonra ihtiyaç duyulan nesneler örneklenir (instantiation) ve ilgili nesne özellikleri (attributes) Java yansıma API'sı vasıtasıyla kullanıcıdan alınan değerlerle doldurulur. Tabi söz konusu nesnenin yine nesne tipinde özellikleri olabilir. Bu durumda yukarıdaki işlemleri yerine getiren metot özyineli (recursive) olarak çalıştırılmaktadır.

Anlamsal web servisinin çalıştırılması için istenen parametreler yukarıdaki gibi hazırlanıp servis çalıştırılmaktadır. Servis çağrımı çıktısının da bir nesne olması durumunda yine yukarıdaki prosedür izlenerek nesne istemci tarafından kullanılabilecek şekilde hazır hale getirilir.

5. Sonuç

Bu bildiride anlatılan çalışmada, anlamsal web ortamında servis bulma ve çağırmaya yönelik gittikçe popülerleşen ve standartlaşmaya doğru giden DAML-S (ve yeni sürümü olarak OWL-S) modeline uygun anlamsal web servisi geliştirme ve kullanma hedeflenmiştir. Atomik yapıdaki anlamsal web servislerinin kendilerine ait ontoloji dokümanlarınca sağladığı çıkarsama olanakları kullanılarak otonom olarak bulunması ve dinamik çağrımını burada anlatılan çalışma kapsamında başarıyla gerçekleştirildi. Ayrıca bu bildiride yetenekleri anlatılan anlamsal servis eşleme motoru fikrine bağlı olarak geliştirilen eşleme motoru; devam etmekte olan anlamsal eşlemeye dayalı çok etmenli sistem geliştirme çatısı [13], mobil ortamlar için anlamsal eşlemeye dayalı bilgi arama sistemi vb. bir çok anlamsal web temelli çalışmalarda kullanılmaktadır.

Ancak elde ettiğimiz deneyimlere dayanarak özellikle bütünleşik anlamsal web servisleri için dinamik çağrımda, yani WSDL veya başka bir alt yapı temelli fiziksel çağrım için servis birleştirmede (composition) şu an için güçlüklerin bulunduğu ve ontolojik tanımlamalardan gerçek çağrım seviyesine inmede daha etkin yapılar getiren ve güncel servis çağrım teknolojilerince de desteklenen anlamsal altyapı tanımlamalarının geliştirilmesi gerektiğine inanmaktayız.

Kaynakça

[1]. Berners-Lee T., Hendler J. ve Lassila O., "The Semantic Web", Scientific American, 284(5), s.34-43, 2001.

[2]. McGuinness D. L. ve van Harmelen F., "OWL Web Ontology Language Overview", W3C Recommendation (URL: http://www.w3.org/TR/owl-features/), 2004.

[3]. The OWL Services Coalition, "OWL-S: Semantic Markup for Web Services" (URL: http://www.daml.org/services /owl-s/1.0/owl-s.html), 2004.

[4]. The DAML Services Coalition, "DAML-S: Semantic Markup for Web Services" (URL: http://www.daml.org/ services /daml-s/0.9/daml-s.html), 2003.

[5]. Sycara K., Paolucci M., Ankolekar A. ve Srinivasan N., "Automated discovery, interaction and composition of Semantic Web Services", Journal of Web Semantics, Elsevier, (1), s.27-46, 2003.

[6]. Paolucci M., Kawamura T., Payne T. R. ve Sycara K., "Semantic Matching of Web Services Capabilities", Lecture Notes In Computer Science; Vol. 2342 Procs of the First International Semantic Web Conference, s.333-347, 2002.

[7]. Noy N. F., Sintek M., Decker S., Crubezy M., Fergerson R. W. ve Musen M. A., "Creating Semantic Web Contents with Protege-2000", IEEE Intelligent Systems, 16(2), s.60-71, 2001.

[8]. JavaBeans Technology (URL: http://java.sun.com/products/javabeans/).

[9]. Johan Lövdahl, The Semantic Web Development Environment (URL: http://www.the-swede-system.org/).

[10]. Paolucci M., Srinivasan N., Sycara K. ve Nishimura T., "Towards a Semantic Choreography of Web Services: from WSDL to DAML-S", International Conference for Web Service, 2003.

[11]. Li L. ve Horrocks I., "A Software Framework for Matchmaking based on Semantic Web Technology", in the proc. of WWW'2003, Budapeşte, Macaristan, s. 331-339, 2003.

[12]. JENA - A Semantic Web Framework for Java (URL: http://jena.sourceforge.net)

[13]. Dikeneli O., Gümüs O., Tiryaki A. M. ve Kardas G., "Engineering a Multi Agent Platform with Dynamic Semantic Service Discovery and Invocation Capability", MATES'05 (Third German Conference on Multiagent System Technologies), Koblenz-Landau Üniversitesi, Koblenz, Almanya, 2005