Çankaya University Journal of Science and Engineering
Volume 8 (2011), No. 1, 75–98

OWL-S Tabanlı Atomik Süreçlerin Birleşimi Üzerine Semantik Tabanlı İş Akışı Modeli

Duygu Çelik[1] ve Atilla Elçi[2]

[1] *İstanbul Aydın Üniversitesi, Bilgisayar Mühendisliği Bölümü, 34295, İstanbul, Türkiye*
[2] *Süleyman Demirel Üniversitesi, Eğitim Teknolojileri Bölümü, 32260, Isparta, Türkiye*
Corresponding authors: [1] *duygucelik@aydin.edu.tr,* [2] *atilla.elci@gmail.com*

Özet. Anlamsal Web'in en önemli getirisi olan *ontolojiler* içine gömülen semantik betimlemeler üzerinden, kullanıcı taleplerinin karşılanması ve varolan veriler üzerinden anlamsal ve mantıksal çıkarsamaların otomatik olarak yapılması mümkündür. Ayrıca, Web servislerini ontoloji dosyalarına betimlemek mümkün olup, bu gömülü olan semantik betimlemeler üzerinden Web servislerinin keşfi ve birleştirilmesi işlemlerinde daha akıllı sistemler geliştirilebilir. Servislerin birleştirilmesinde uygun süreçlerin sorgulanması, analizi ve süreçler arası uyum taraması gibi adımları anlamsal boyuta taşımaktadır. Bu makalede, atomik yapıdaki süreçlerin *Çıkarsama-tabanlı Semantik Birleşim Ajanı* (*Inference-based Semantic Composition Agent-SCA*) çerçevesi önerilmektedir. SCA süreçlerde benzerlik eşleşmesi ve çıkarsama tekniklerini içermektedir. SCA'nın görevi varolan süreçleri sentezleyip, kullanıcı tarafından sorgulanan ve aslında varolmayan yeni bir sürecin, *Birleşik/ Kompleks Süreç*'in, otomatik olarak iş akışı planını belirleyebilmektir. Bu nedenle, SCA' nın güçlü bir uyum tarama mekanizmasına sahip olması, ve ayrıca uygun süreçleri bulup, arkasından uygun sırada yerleştirmesiyle, istenilen *Kompleks Süreç*'in üretilmesi sağlanabilir. SCA'nın, sorgulanan *Birleşik/Kompleks Süreç*'e ait, *İş Akışı Planı*'nı yapılandırması esnasında, yeni bir *Semantik Uyum Adımı*'nı (*Semantic Matching Step-SMS*) çalıştırmaktadır. Bunun yanısıra, istenilen *Birleşik/Kompleks Süreç*'in, diğer bir deyişle amaç sürecin, oluşumunda kullanılan çıkarsamalar, *Revize edilmiş Armstrong Aksiyomları*'dır (*Revised Armstrong's Axioms-RAA*). Bu çalışmada, SCA çerçevesinde sunulan katkı, ilk defa *Armstrong'un Çıkarsama Kuralları* revize edilmiş ve Semantik Web uygulamalarında, planlama ve çıkarsama işlevlerinde kullanılmıştır.

Anahtar Kelimeler. Web servisleri, Web servislerin birleşimi, Armstrong'un Çıkarsama Kuralları, Semantik Web servisleri.

Abstract. Due to the insertion of semantics to the Web services, meeting user demands will be possible through logical deductions and achieving resolutions automatically. With the application of semantic description in content of Web services it is easier to discover

Received June 30, 2010; accepted April 25, 2011.
Bu makale, 29-30 Nisan 2010 tarihlerinde Çankaya Üniversitesi'nin Ankara yerleşkesinde yapılmış olan *3. Çankaya Üniversitesi Mühendislik ve Teknoloji Sempozyumu*'nda sunulan ve sadece geniş bildiri özeti bölümü hakem sürecinden geçerek bu sempozyum kitapçığında yayımlanan bir makalenin revize edilmiş şekli olup Sempozyum Değerlendirme Komitesi tarafından yayımlanmak üzere *Çankaya University Journal of Science and Engineering* dergisine gönderilmesi önerilmiş ve derginin bağımsız hakem değerlendirmeleri sonucunda yayıma kabul edilmiştir.

and compose suitable Web services during analyzing, searching, and matching processes. We proposed a framework of an *Inference-based Semantic Composition Agent* (SCA) of atomic processes that employs process similarity matching and inference techniques. The proposed SCA is responsible for the synthesis of new services from existing ones in an automatic fashion. Therefore, it requires a powerful matching mechanism to find fitting tasks in order to attain the required composition by each client. An innovative *Semantic Matching Step-SMS* of SCA helps to find the fitting tasks while constituting workflow to achieve required composition. Additionally, SCA composes available OWL-S atomic processes utilizing *Revised Armstrong's Axioms-RAA* in inferring functional dependencies. Experiments show that the proposed SCA system produces atomic process sequences as a workflow in order to achieve the required composition plan that satisfies user's requirements as a complex task. The novelty of the SCA system is that for the first time Armstrong's Axioms are revised and used for semantic-based planning and inferencing of services.

Keywords. Web services, Web services composition, Armstrong's Axioms, semantic Web services.

1. Giriş

Ontolojiler; belli bir *alan*ın (*domain*) kavramsal modellemesini, yani o alanın biçimsel ve anlamsal betimlemesini sunmaya yarayan bir *Semantik Web* (*Semantic Web-SW*) teknolojisidir. Bilgisayarca-okunabilir ve bilgisayarca-anlaşılabilir olmasını sağlayacak şekilde, alanların (domain) *sınıf*larını (*class*), *altsınıf*larını (*subclass*), *özellik*lerini (property), ve *işlev*lerini (function) betimlemeyi sağlamaktadır. Alanların kavramsal modellemesinde en yaygın olarak kullanılan Semantik Web teknolojisi, *Web Ontoloji Dili*'dir (*Web Ontology Language-OWL*) [1].

Ayrıca ontolojiler, Web servislerininde bilgisayarların anlayabileceği şekilde, yani anlamsal tabanda betimlenmelerinde çok önemli bir rol oynamaktadır. *Semantik Web Servisleri* (*Semantic Web Services-SWSs*) [2], klasik *Web Servisleri*'nin (*Web Services-WS*) bir uzantısı olmakla beraber, WS'nin anlamsal ve biçimsel gösteriminin ontoloji tabanlı diller üzerinden betimlendirilmesiyle oluşturulmaktadır. Bu nedenle, *Servisler için Ontoloji Web Dili* (*Ontology Web Language of Services-OWL-S*) [3], WS'nin anlamsal tabanda betimlenmesi için geliştirilmiş OWL tabanlı bir dildir. Bu makalede önerilen sistemin yapısında kullanılan ontoloji dosyaları, yaygın olarak SW alanında yapılan araştırmalarda kullanılmakta olduğundan İngilizce baz alınmış, bu nedenle de geliştirdiğimiz tüm ontolojiler ve ilgili açıklamalar bazı yerlerde İngilizce olarak yazılmıştır.

OWL-S, WS'nin içerdikleri *süreç*lerin (process), *girdi*/*çıktı*/*önkoşul*/*etki* (input/output/precondition/effect-I/O/P/E) bilgilerini, erişim, adres veya kimlik bilgilerini,

insanlara sunmaktan ziyade, bilgisayarlar tarafından bu bilgilerin okunup işlenebilmesini sağlamak amacıyla tasarlanmış bir ontoloji dilidir. Kısacası, WS'lerine anlamsallık katan en önemli Semantik Web teknolojilerinden biridir.

OWL veya OWL-S gibi SW teknolojilerinin Semantik Web servisleri (SWSs) üzerinde çok önemli bir yere sahip olması, günümüzde bilgisayarca-otomatik olarak *SWSs' nin Birleştirilmesi* (*Composition of Semantic Web Services*) gibi yaygın olarak araştırılan, yenilikçi ve önemli bir konuda bu gibi ontoloji tabanlı dillerin önemli bir rol alabileceğini akla getirmektedir.

WS'lerinin birleştirilmesindeki amaç, *Web Kullanıcıları*'nın (*Web Users*) aradıkları bir sürecin aslında Web ortamında o an bulunamaması ya da var olmaması halinde, elde var olan *Aday Semantik Web Servisleri*'ne (*Candidate Semantic Web Services-CSWSs*) ait süreçlerin birleştirilmesiyle oluşturulan, ve kullanıcının aradığı uygun hizmeti sağlayabilecek yeteneğe sahip, yeni bir *Kompleks Süreç*'in türetilmesini sağlamaktır. Bu teknolojiler sayesinde, dinamik olarak WS'nin birleştirilmesi mümkündür.

Bu makalede, CSWSs'ne ait OWL-S dosyalarında gömülü olan semantik/anlamsal betimlemeler üzerinden, çıkarsama metodu ile, kullanıcı tarafından istenilen Kompleks Süreç'in türetilmesini amaçlayan bir *Çıkarsama-tabanlı Semantik Birleşim Ajanı* (*Inference-based Semantic Composition Agent-SCA*) çerçevesi önerilmektedir. Kompleks Süreç'in türetilmesi için, CSWSs'ne ait süreçlerin fonksiyonel betimlemeleri, bulundukları OWL-S dosyalarından bilgisayarca-okunur ve gerekli çıkarsamalar yapılarak, bu sayede yapılandırılan birleşme için uygun olan süreçler seçilir. Daha sonra mantıklı bir sırada yürütülebilmesi amacıyla, sistemin planlayıcısı tarafından *Birleşim İş Akışı Planı* üretilir. Son zamanlarda, Web süreçlerinin birleştirilmesi hususunda yürütülen araştırmaların bir çoğu, istenilen birleşmenin planlama veya yapılandırılma aşamasında, birleşme zincirine uygun Web süreçlerini birer birer ekleme uslübü ile yürütmeyi önermektedir [4, 5, 26-28]. Uygun süreçlerle yapılandırılan Birleşim İş Akışı Planı'na, süreçler teker teker eklenirken, her eklenen sürecin uygunluğu, o sürecin girdi/çıktı/önkoşul/etki (I/O/P/E) parametrelerine ait konseptlerin anlamsal bağlamda incelenmesini gerektirir.

Yeni yapılandırılan bir Birleşim İş Akışı Planı için, o an odaklanılan sürecin I/O/P/E parametreleri ile kendisinden önce ve sonra gelen süreçlerin I/O/P/E parametreleri arasındaki anlamsal benzerlik önceden tanımlanmalıdır. Bu nedenle, WS'nin birleştirilmesinde sistematik, güçlü ve skorlama mantığıyla çalışan bir *Semantik Uyum Adımı*'na (*Semantic Matching Step-SMS*) gereksinim vardır. SMS, odaklanılan

süreçlerin I/O/P/E parametrelerine ait konseptlerin, anlamsal ilişkilerini belirler ve derecelendirir. Bu makalede, önerilen çerçevenin en önemli parçalarından biri olan SMS, CSWSs'nin süreçleri arasında benzerlik mesafesini derecelendirerek birer skor değeri hesaplamaktadır. Önerilen SCA çerçevesi, Web kullanıcısının isteğini karşılaması için gerekli olan Kompleks Süreç'in oluşturulmasında, CSWSs'nin içerdikleri atomik süreçleri anlamsal tabanda inceledikten sonra, belirlediği skorlar üzerinden eleme yöntemiyle uygun (skor değeri yüksek) olanları seçer, ve arkasından sistemin planlayıcısı tarafından, uygun süreç sıralaması belirlenir. Böylece, belirlenen süreç sıralaması aslında Birleşim İş Akışı Planı'nı oluşturur. Detaylar gelecek bölümde verilmiştir.

Bu özelliklerinin yanısıra, önerilen SCA, yeni bir *planlama* ve *çıkarsama* tekniğiyle geliştirilmiştir ve bunun için *Revize Armstrong Aksiyomları* (*Revised Armstrong's Axioms-RAA*), çıkarsama kuralları olarak kullanılmaktadır. *Armstrong Aksiyomları* (*Armstrong's Axioms-AA*) bir aksiyomlar/çıkarsama kuralları kümesidir. Bu aksiyomlar sayesinde *İlişiksel Veritabanları*'nda (*Relational Database*) fonksiyonel bağlılıkların çıkarsamaları yapılabilmektedir. Bu aksiyomlar ilk olarak William W. Armstrong'un "Dependency Structures of Data Base Relationships" başlıklı makalesinde yayımlanmıştır [6]. İlk defa bu çalışmada, veritabanları için uygulanan bu aksiyomların, WS'nin birleştirilmesinin planlama ve çıkarsama safhalarında uygulanması önerildi. Bu nedenle doğrulukları kanıtlanmış, yaygın olarak bilinen ve kullanılan Armstrong Aksiyomları'nı, Web servislerinin birleşmesine uygulanabilecek şekilde revize ettik ve Revize Armstrong Aksiyomları olarak adlandırdık.

Sonuç olarak, önerilen SCA, güçlü bir süreç uyum mekanizmasına sahiptir ve bu sayede uygun süreçleri bulup daha sonra uygun sırada yerleştirerek beklenen Kompleks Süreç'in üretilmesini sağlamaktadır. SCA, kullanıcı tarafından istenilen Birleşim İş Akışı Planı'nın yapılandırılması ve Kompleks Süreç'in oluşumu esnasında, kendi Semantik Uyum Adımı'nı yürütmektedir. Bunun yanısıra, Kompleks Süreç'in oluşumunda yürütülen çıkarsamalar, Revize Armstrong Aksiyomları üzerinden yapılandırılmıştır. SCA çerçevesinde sunulan katkı ilk defa Armstrong'un Çıkarsama Kuralları revize edilmiş ve Semantik Web uygulamalarında, planlama ve çıkarsama işlevlerinde kullanılmıştır.

Bu makale aşağıdaki sıra ile sunulacaktır: Bölüm 2'de problem tanımı ve süreçlerin I/O/P/E parametreleri arasındaki uyum taramasının gerekliliği anlatılacaktır. Bölüm 3'te kısaca, önerilen SCA sisteminin mimarisi ve bölümleri anlatılacaktır. Bölüm

4'te, SCA sisteminde kullanılan ontoloji bilgi tabanlarının yapıları, işlevleri, kullanım amaçları hakkında bilgi sunulacaktır. Bölüm 5'te, SCA sisteminin yürüttüğü en önemli işlevlerden biri olan, Semantik Uyum Adımı incelenecektir. Bölüm 6'da, SCA sistemi için revize edilen, Revize Armstrong Aksiyomları incelenecektir. Son olarak, Bölüm 7'de ilgili sonuçlar açıklanacaktır.

2. Problem Tanımı

SWSs'nin betimleme dili olan OWL-S, üç bölümde modellenmiştir: *Profile, Process* ve *Grounding* (Tablo 1, Satır 4-6). Profile bölümü, servislerin sunduğu hizmet ve servis programcısı tarafından yapılandırılmış fonksiyonlar hakkında bilgi içermektedir. Process bölümü, servislerin sunduğu hizmete ait yapılandırılmış bu fonksiyonların nasıl çalıştığı, hangi konseptleri girdi-çıktı olarak aldığı, ve bu konseptlerin parametre türleri, bununla birlikte, o konseptlerin hangi ontoloji dosyaları altında betimlendiği hakkında bilgi içermektedir (Tablo 1, Satır 8-11). Grounding bölümünde ise, servislerin *adres, url* veya o servisi çağırabilmek için gerekli olan *erişim* bilgileri betimlenmiştir.

Tablo 1. 'car_recommendedpriceineuro_service' servisine ait OWL-S dosyası.

```
1 <owl:imports rdf:resource="http://127.0.0.1/ontology/my_ontology.owl" />
2 <owl:imports rdf:resource="http://127.0.0.1/ontology/concept.owl" />

3 <service:Service rdf:ID="CAR_RECOMMENDEDPRICEINEURO_SERVICE">
4   <service:presents rdf:resource="#CAR_RECOMMENDEDPRICEINEURO_PROFILE"/>
5   <service:describedBy rdf:resource="#CAR_RECOMMENDEDPRICEINEURO_PROCESS"/>
6   <service:supports rdf:resource="#CAR_RECOMMENDEDPRICEINEURO_GROUNDING"/>
7 </service:Service>

8    <process:AtomicProcess rdf:ID="CAR_RECOMMENDEDPRICEINEURO_PROCESS">
9       <process:hasInput  rdf:resource="#_CAR"/>
10     <process:hasOutput rdf:resource="#_RECOMMENDEDPRICEINEURO"/>
11  </process:AtomicProcess>

12  <process:Input rdf:ID="_CAR">
13        <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
http://127.0.0.1/ontology/my_ontology.owl#Car</process:parameterType>
14  </process:Input>

15  <process:Output  rdf:ID="_RECOMMENDEDPRICEINEURO">
16        <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
http://127.0.0.1/ontology/concept.owl#RecommendedPriceInEuro</process:parameterType>
17  </process:Output>
```

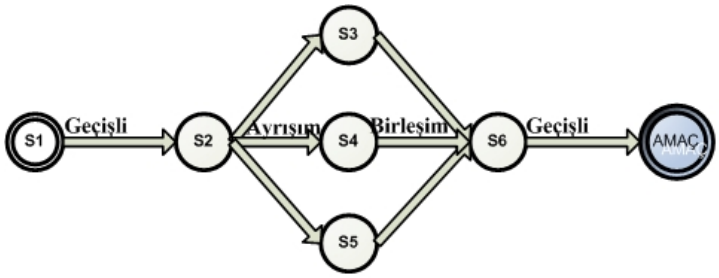
OWL-S'in process bölümünde, her servisin kendine ait, ve daha önceden servis programcısı veya ontolojisti tarafından hazırlanmış olan OWL-S dosyasında betimlenmiş, I/O/P/E parametrelerinin her biri, birer konsepte işaret etmektedir. Her bir konsept de, ait olduğu alana (domain) ait ve önceden ontolojistler tarafından geliştirilmiş o alanın ontoloji dosyasındaki yerine işaret etmektedir.

Girdi (input) ve çıktı (output) parametrelerine ait konseptler, tanımlandıkları sürecin yürütülmesi/çalıştırılması esnasında, gerçekleşen data transferi hakkında bilgi tutmaktadır. OWL-S dosyalarında, girdi bilgisi, o sürecin çalıştırılması esnasında, bir kullanıcıdan/programdan alınan parametrenin konsepti hakkında bilgi tutmaktadır (Tablo 1, Satır 12-14). Çıktı ise o sürecin çalıştırıldıktan sonra geri döndürdüğü sonuç parametrenin ait olduğu konsept hakkında bilgi verir (Tablo 1, Satır 15-17).

Önkoşul ise, o sürecin çalıştırılması esnasında girilen girdilerin üzerinde konulmuş koşullar hakkında bilgi vermektedir. Etki, sürecin çalıştırılmasından sonra elde edilen çıktıların, o sürecin ortamında oluşturduğu etki ve değişimler hakkında bilgi tutmaktadır. Tablo 1'de araba fiyat edinme 'car_recommendedpriceineuro_service' servisine ait bir OWL-S dosyası gösterilmektedir. Bu servisin bir atomik sürece sahip olduğu Tablo 1'de 8. ve 11. satırlar arasında görülmektedir. Bu atomik sürecin sadece bir girdisi bulunmaktadır ve `<process:Input rdf:ID="_CAR">` olarak OWL fişleriyle Satır 12-14'de betimlenmiştir. Ayrıca, bu atomik sürecin sadece bir çıktısı bulunmaktadır ve `<process:Output rdf:ID="_RECOMMENDEDPRICEINEURO">` olarak OWL fişleriyle Satır 15-17'de betimlenmiştir.

Süreçler arasındaki uyum taraması, yeni yapılandırılan bir Birleşim İş Akışı Planı'na eklenen her sürecin uygunluğunu keşfetmek için gerekli bir adımdır. Uyum mekanizması, CSWSs'nin, OWL-S dosyalarında gömülü olan, `process:parameterType` veya `rdf:ID` girdi-çıktı konseptleri arasında olası varolan kavramsal-tabanlı ilişkileri semantik boyutta inceler. Örneğin, bir ontoloji içinde 'ÖnerilenFiyat' (RecommendedPrice) konsepti, 'Fiyat' (Price) konseptinin bir alt sınıfı olarak betimlenmiş, ek olarak, 'DolarCinsindenÖnerilenFiyat' (RecommendedPriceInDollar) ve 'EuroCinsindenÖnerilenFiyat' (RecommendedPriceInEuro) konseptleri de, 'ÖnerilenFiyat' (RecommendedPrice) konseptinin alt sınıfları olarak betimlenmiş olsun. İyi tasarlanmış bir uyum tarama mekanizması, 'EuroCinsindenÖnerilenFiyat' (RecommendedPriceInEuro) konseptinin aslında hem 'ÖnerilenFiyat' (RecommendedPrice) hem de 'Fiyat' (Price) konseptlerinin alt sınıfları olduğunu, iyi tasarlanmış çıkarsama aksiyomlarıyla ele geçirebilir.

Birleşim İş Akışı diagramı genellikle biçimsel olarak akış diagramlama teknikleriyle tanımlanırken, dahil süreçler arasında yönlü veya yönsüz bağlar kurularak gösterilebilir (Şekil 1'de verilen iş akışı örneğine bakınız).



Şekil 1. İstenilen amaç süreci elde etmek için altı farklı sürecin birleşiminden oluşan bir iş akışı plan şeması.

Tek bir iş geçidi veya bir Birleşim İş Akışı Planı'nın temel komponentleri üç parçadan oluşmaktadır;

- Girdi: belli bir noktadaki süreç adımı tamamlamak için gerekli olan bilgi.
- Geçiş kuralları, algoritmaları vs...
- Çıktı: belli bir noktadaki süreç tarafından üretilen ve sıradaki sürece girdi olarak sağlanacak bilgi.

İki sürecin birbirine bağlanabilmesi için bir önceki sürecin çıktı kümesindeki her konseptin, takibindeki sürecin girdi kümesindeki her bir konsept ile bire bir ilişkisi olmalıdır. Ya da, bir önceki sürecin girdi kümesindeki her konseptin, takibindeki sürecin girdi kümesindeki her bir konsept ile bire bir ilişkisi olmalıdır. Bu ilişkiler sözdizimsel olarak incelendiğinde, girdiler ve çıktılar arasında aranılan bire birlik ilişkiler anlam yokluğundan, zincire eklenecek uygun süreçleri taramada kayıplara neden olabilmektedir.

Servis 1.

```
<process:Output  rdf:ID="_RECOMMENDEDPRICEINEURO">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
http://127.0.0.1/ontology/concept.owl#RecommendedPriceInEuro</process:parameterType>
</process:Output>
```

Servis 2.

```
<process:Input rdf:ID="_PRICEINEURO">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
http://127.0.0.1/ontology/concept.owl#Price</process:parameterType>
</process:Input>
```

Yukarıda verilen iki farklı SWS'ne ait OWL-S parçalarında, Servis 1'e ait bir çıktı (output) ve Servis 2'ye ait bir girdi (input) bilgisi verilmiştir. İlaveten, hem girdi hem de çıktı bilgilerinin, parametre türlerine ait konsept bilgileri tanımlanmıştır; Servis 1 için tanımlanan çıktının parametre türü (`parameterType`), `\#Recommended PriceInEuro` konsepti olarak, `http://127.0.0.1/ontology/concept.owl` adresinde bulunan 'concept.owl' adlı alan (domain) ontoloji dosyası altında tanımlanmıştır.

Ayrıca, Servis 2 için tanımlanan girdinin parametre türü (`parameterType`), `#Price` konsepti olarak, yine `http://127.0.0.1/ontology/concept.owl` adresinde bulunan 'concept.owl' adlı aynı alan (domain) ontoloji dosyası altında tanımlanmıştır. Bu verilen süreçler arasındaki girdi-çıktı bilgileri birbirinin altsınıf ilişkisiyle bağlanmaktadır, yani, 'fiyat' (price) konseptinin bir alt sınıfı olan 'EuroCinsindenÖnerilenFiyat' (`RecommendedPriceInEuro`) konsepti, bize ikinci servisin girdi konsepti, ilk servisin çıktı konseptini tükettiğini göstermektedir. Aralarındaki bu ilişki *geçişli* bir ilişkiye işaret eder ve bu iki servisin birleştirilebilir özelliğe sahip olduklarını vurgular.

Servis 1:

```
<process:Input rdf:ID="_CAR">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
http://127.0.0.1/ontology/my_ontology.owl#Car</process:parameterType>
</process:Input>
```

Servis 2:

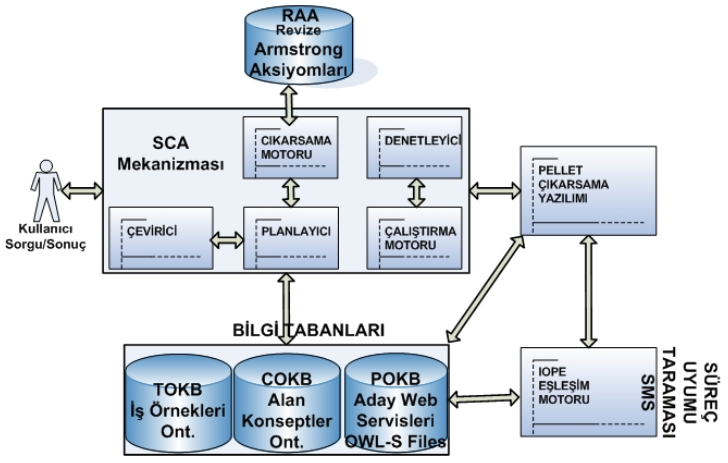
```
<process:Input rdf:ID="_AUTOMOBILE">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
http://127.0.0.1/ontology/my_ontology.owl#Automobile</process:parameterType>
</process:Input>
```

Yukarıda verilen diğer iki OWL-S parçasında, Servis 1'e ve Servis 2'ye ait birer girdi bilgisi verilmiştir. Ayrıca bu girdi bilgilerinin, parametre türüne (`parameterType`) ait konsept bilgileri tanımlanmıştır; Servis 1 için tanımlanan girdinin parametre türü (`parameterType`), `#Car` konsepti olarak, `http://127.0.0.1/ontology/my_ontology.owl` adresinde bulunan 'my_ontology.owl' adlı alan(domain) ontoloji dosyası altında betimlenmiştir. Ayrıca, Servis 2 için tanımlanan girdinin parametre türü (`parameterType`), `#Automobile` konsepti olarak, yine `http://127.0.0.1/ontology/my_ontology.owl` adresinde bulunan aynı 'my_ontology.owl' adlı alan (domain) ontoloji dosyası altında tanımlanmıştır. Bu süreçlerin girdi bilgileri birbirinin 'Eşanlamlısı' (`Synonym`) özelliğiyle ilişkilendirilmiştir, yani, 'Araba' (`Car`) konseptinin eş anlamlısı olan, 'Otomobil' (`Automobile`) konsepti, bize her iki servisin

girdi konseptlerinin, aslında birbirinin aynı anlamda olduğu ve *birleşim* özelliğine sahip olduğunu göstermektedir. Aralarında bulunan birleşim ilişkisi, bu iki servisin birleştirilebilir özelliğe sahip olduklarını ifade eder. Gelecek bölümde, önerilen SCA sisteminin mimarisi ve işlevleri hakkında kısaca bilgi sunulacaktır.

3. Önerilen SCA Mimarisi

Önerilen SCA sistemi, beş ana modülden oluşmaktadır; *Çevirici* (*Translator*), *Planlayıcı* (*Planner*), *Çıkarsama Motoru* (*Inference Engine*), *Çalıştırma Motoru* (*Execution Engine*), ve *Denetleyici* (*Monitoring Agent).* Ek olarak, süreç uyum tarama mekanizması olan Semantik Uyum Adımı'nı (Semantic Matching Step-SMS) içermektedir (bkz. Şekil 2). SMS, Aday Semantik Web Servisleri'ne (Candidate Semantic Web Services-CSWSs) ait tüm süreçler arasındaki anlamsal bağlılıkları tespit edip, bu bağlılıklar için total benzerlik skorlarını hesaplamak için kullanılmaktadır. SMS için gerekli detay bilgi gelecek bölümde verilecektir. Ayrıca SCA, üç farklı ontoloji bilgi tabanına sahiptir: *Konseptler Ontolojisi Bilgi Tabanı* (*Concepts Knowledgebase-COKB*), *İş Örnekleri Ontolojisi Bilgi Tabanı* (*Task Ontology-TOKB*), ve *Süreç Ontolojileri Bilgi Tabanı* (*Processes Knowledgebase-POKB*) ve son olarak bir tane *Çıkarsama Kuralları Bilgi Tabanı*'na (*Rule Knowledgebase-RKB*) sahiptir. RKB, Revize edilmiş Armstrong Aksiyomları'nı (Revised Armstrong's Axioms-RAA) içermektedir. RAA için gerekli detay bilgi Bölüm 6'da verilecektir. SCA mimarisine ait gösterim aşağıda Şekil 2'de gösterilmiştir.



Şekil 2. SCA sisteminin genel mimarisi.

Çevirici (translator), ilgili TOKB ontolojisi altında önceden betimlenmiş olan, ilgili atomik iş örneklerine ait I/O/P/E parametrelerine, ayrıca POKB altında bulunan ilgili Aday Semantik Web Servisleri'nin OWL-S dosyalarında betimlenmiş olan, ilgili atomik süreçlerin I/O/P/E parametrelerine ve son olarak istenilen Kompleks Süreç'in I/O/P/E parametrelerine ait, konsept bilgilerinin bulundukları ontoloji dosyalarından ele geçirilmesinden sorumludur. I/O/P/E modellemesine göre, TOKB veya POKB bilgi tabanlarında, betimlenmiş atomik iş örnekleri veya süreçlerin (A_i), $A_i \equiv I_i \rightarrow O_i$ formuna dönüştürülmesi Çevirici (translator) tarafından yapılmaktadır. Odaklanılan atomik iş örneğinin veya bir sürecin içerdiği girdi konseptleri $I_i \equiv I_{i1} \wedge I_{i2} \wedge \cdots \wedge I_{ik}$ şeklinde ifade edilmiştir. Ayrıca, yine odaklanılan atomik iş örneğinin veya bir sürecin içerdiği çıktı konseptleri $O_i \equiv O_{i1} \wedge O_{i2} \wedge \cdots \wedge O_{im}$ şeklinde ifade edilmiştir. $A_i \equiv I_i \rightarrow O_i$ mantıksal ifadesi, I_i parametrelerine ait konseptleri var olduğunda, O_i parametrelerine erişilebilinir anlamını içermektedir. Doğrusal ilişki $(\rightarrow)$, I_i parametrelerinin tüketildiğini ve O_i parametrelerinin üretildiğini göstermektedir. Kompleks Süreç'in mantıksal formu, $G \equiv I_G \rightarrow O_G$ biçimindedir ve içerdiği girdi konseptleri $I_G \equiv I_{G1} \wedge IG2 \wedge \cdots \wedge I_{Gj}$ şeklinde, ayrıca içerdiği çıktı konseptleri $O_G \equiv O_{G1} \wedge O_{G2} \wedge \cdots \wedge O_{Gh}$ şeklinde ifade edilmiştir. Odaklanılan iş örnekleri, süreçler ve Kompleks Süreç, bu mantıksal forma dönüştürüldükten sonra, Planlayıcı'ya (planner) istenilen Birleşim İş Akışı Planı'nın yapılandırılması için gönderilir.

Planlayıcı (planner), SCA sisteminin merkezinde bulunmaktadır. Planlayıcı'nın amacı istenilen Birleşim İş Akışı Planı'nın yapılandırılmasını sağlamaktır. Bunun için odaklanılan iş örnekleri ve süreçler arasında anlamsal benzerlikler üzerinden Kompleks Süreç'e en yakın Birleşim İş Akışı Planı'nı üretir. Üretilen planın süreçleri arasındaki uyumu taramak için SMS'in ürettiği benzerlik skorlarından faydalanır. En büyük skor, o sürecin o anda zincire eklenecek en uygun süreç olduğunu ifade eder.

Çıkarsama Motoru (inference engine), Revize edilmiş Armstrong Aksiyomları'nı kullanarak, Kompleks Süreç'in türetilmesinde rol alır, ve bunun için odaklanılan aday süreçler arasındaki geçişli, ayrışım veya birleşim gibi ilişkileri tespit eder. Her adımda, çıkarsama motoru, iki farklı süreci birleştirebilmek için, uygun olan çıkarsama kuralını seçer ve bu süreçlere uygulayıp yeni bir süreç elde eder. Bu işlem Kompleks Süreç'e ulaşılana kadar yürütülür. Her döngüde üretilen her yeni süreç, sistemin planlayıcısı tarafından, aranılan Kompleks Süreç'e benzerliği bakımından kıyaslanır. Bu nedenle, Planlayıcı ve Çıkarsama Motoru, Kompleks Süreç'in türetilmesinde koordineli olarak çalışmaktadır. Özet olarak, SCA'nın planlayıcı modülünden üretilen

Birleşim İş Akışı Planı (P) ve Kompleks Süreç ($G \equiv I_G \rightarrow O_G$) aynı anlama gelmelidir (mantıksal gösterimde: $P \models G$).

Çalıştırma Motoru (execution engine), planın içinde yer alan Aday Web Servisleri'ne ait süreçlerin, üretilen Birleşim İş Akışı Planı'nda belirlenen sırayla yürütülmesini sağlar.

Son olarak, Denetleyici (monitoring agent), plandaki süreçlerin doğru olarak yürütülmesini sağlamak amacıyla planın yürütülmesi esnasında denetim yapar. Kısacası, son iki adımda, Planlayıcı tarafından üretilen bir Birleşim İş Akışı Planı'ndaki (P) süreçlerin belirlenen sırada doğru parametreler ile yürütülmesini ve denetleyici tarafından da gerekli denetimlerin yapılması sağlanmaktadır. Birleşim İş Akışı Planı'ndaki (P) süreçlerin yürütülmesi sağlandıktan sonra elde edilen sonuç Web kullanıcısına sonuç olarak sunulur. Gelecek bölümde, önerilen SCA sistemde kullanılan ontoloji tabanlarından bahsedilmektedir.

4. Kullanılan Ontoloji Bilgi Tabanları (Ontology Knowledge Bases)

SCA'nın süreç benzerliği değerlendirmesi, üç farklı kategoriden oluşmuştur; süreçlerde tanımlanan, girdi-çıktı konseptlerindeki (`rdf:ID` (input-output)) benzerlik, parametre türlerindeki (`process:parameterType`) ve özellik-boyutunda benzerlikler araştırılır.

Özellik-boyutunda benzerlikler iki özelliğe bakılarak değerlendirmeye alınır; 'Eşanlamlı' (`hasSynonym`) ve 'Birörneği' (`hasIs_a`)'dir. Gelecek bölümde gerekli detaylar sunulmuştur. Sonuç olarak, önerdiğimiz sistemin SMS adımında, yukarıda tanımladığımız benzerlik değerlendirilmeleri, RAA çıkarsama kuralları eşliğinde incelenmektedir, bu sayede sistemin güvenirliği, etkinliği ve doğruluk payı arttırılmıştır. SCA sisteminde üç farklı ontoloji bilgi tabanı kullanılmaktadır: Konseptler Ontolojisi Bilgi Tabanı (Concepts Knowledgebase COKB), İş Örnekleri Ontolojisi Bilgi Tabanı (Tasks Ontology Knowledgebase-TOKB) ve Süreç Ontolojileri Bilgi Tabanı (Processes Knowledgebase-POKB).

4.1. Konseptler Ontolojisi Bilgi Tabanı (Concepts Knowledgebase-COKB).

Tablo 2'de, araçlar (vehicle) alan ontolojisinin bir bölümü gösterilmektedir. Sistem, bu ontoloji dosyalarına girip, terimler arasındaki akrabalıkları/benzerlikleri kolayca tanımlayabilir. Aşağıdaki araçlar alan ontolojisinin ana sınıfı (class) 'Araçlar'

(`Vehicle`) terimidir. Bu sınıfa ait iki tür data çeşidi (`DataType Property`) tanımlanmıştır: Eşanlamlı (`Synonym`) ve bir_örneği (`Is_a`)' dır (Satır 3-4). 'Araba' (`Car`) terimi, 'Araçlar' (`Vehicle`) teriminin bir alt sınıfı (subclass) olduğu ontolojimizde 6. satırda gösterilmektedir. 'Araba' (`Car`) konseptinin eşanlamlı konseptleri olan 'Otomobil' (`Automobile`) ve 'Motorluaraba' (`Motorcar`) terimleri 7-8. satırlarda gösterilmiştir. Sistem, tüm alan (domain) ontoloji dosyalarına bu veritabanından ulaşabilmektedir.

Tablo 2. Araçlar (Vehicle) ontolojisi.

```
1 <owl: imports rdf:resource="http://127.0.0.1/ontology/concept.owl"/>
2  <owl:Class rdf:ID="Vehicle"/>
3   <owl:DatatypeProperty rdf:ID="Is_a"/>
4   <owl:DatatypeProperty rdf:ID="Synonym"/>
5      <Vehicle rdf:ID="Car">
6         <Is_a rdf:datatype="&Concept;Vehicle">Vehicle</Is_a>
7          <Synonym rdf:datatype="&Concept;Auto">Automobile</Synonym>
8          <Synonym rdf:datatype="&Concept;Car">Motorcar</Synonym>
9     </Vehicle>
10     <Vehicle rdf:ID="Ship">
11       <Is_a rdf:datatype="&Concept;Vehicle">Vehicle</Is_a>
12   </Vehicle>
13   <Vehicle rdf:ID="Bicycle">
14      <Is_a rdf:datatype="&Concept;Vehicle">Vehicle</Is_a>
15      <Synonym rdf:datatype="&Concept; Bicycle">Bike</Synonym>
16   </Vehicle>
17   <Vehicle rdf:ID="ThreeWheeledCar">
18         <Is_a rdf:datatype="&Concept;Car">Car</Is_a>
19         <Synonym rdf:datatype="&Concept; ThreeWheeledCar">Tricar</Synonym>
21   </Vehicle>
17   <Vehicle rdf:ID="TwoPersonBicycle">
18         <Is_a rdf:datatype="&Concept;Bicycle">Bicycle</Is_a>
19         <Synonym rdf:datatype="&Concept;TwoPersonBicycle">TandemBicycle</Synonym>
20         <Synonym rdf:datatype="&Concept;TwoPersonBicycle">TwinBicycle</Synonym>
21   </Vehicle>
```

4.2. İş Örnekleri Ontolojisi Bilgi Tabanı (Tasks Ontology Knowledgebase-TOKB). İş Örnekleri Ontolojisi Bilgi Tabanı (Task Ontology-TOKB), Web servisleri tarafından sunulan iş veya hizmet türlerini ve bu hizmetlerin fonksiyonel tanımlamalarını, bir ontoloji altında iş örnekleri betimlemeleri gibi toplamayı önerdik. OWL-S dili sayesinde, bir alana (domain) ait olan tüm Web servislerin yaygın olarak sunduğu iş türlerini birer sınıf şeklinde TOKB ontolojisi altında tanımlanması mümkündür. Dahası, SCA, bu TOKB kullanarak, iş örnekleri veya süreçler arasındaki fonksiyonel benzerliği veya farklılığı ayırt etmeyi başarabilir. Örneğin,

Araba_Satış (ArabaModel, Fiyat) ve Araba_Alış (ArabaModel, Fiyat). Verilen örnekte görüldüğü gibi, her iki iş tanımı da aynı girdi ve çıktı konseptlerine sahiptir. İstenilen iş akışının yapılandırılmasında ve Kompleks Süreç'in oluşumunda, aday olan bu tür iş tanımlamalarını ayırt edilmesi kritik bir noktadır. Bu tür iş tanımları arasındaki fonksiyonel benzerlikleri ve farklılıkları seçebilmek için ontoloji tabanlı iş örnekleri ontolojisi bilgi tabanı, önerilen SCA sisteminin doğru çalışabilmesi için gerekli görülmüştür.

Tablo 3'te TOKB ontolojisinden bir parça sunulmuştur. SCA, SMS adımındaki çıkarsama işleminde, TOKB içindeki önceden tanımlı olan iş örneklerini baz alarak, Aday Web Servisleri'nin fonksiyonel tanımlarıyla karşılaştırıp eşleşme işlemini yürütür. TOKB ontolojisinde tanımlanan anlamsal iş içerikleri, OWL dilindeki `<owl:class>`, `<rdfs:subClassOf>`, `<owl:DatatypeProperty>` gibi özellikler yardımıyla betimlenmiştir.

Tablo 3. TOKB ontolojisinden kesit.

```
1 <owl: imports rdf:resource="http://127.0.0.1/ontology/concept.owl"/>
2 <owl: imports rdf:resource="http://127.0.0.1/ontology/vehicle.owl"/>

<!-Datatype Properties for I/O/P/E Parameters of Tasks -->

3  <owl:DatatypeProperty rdf:ID="Input_Parameter"/>
4  <owl:DatatypeProperty rdf:ID="Output_Parameter"/>
5  <owl:DatatypeProperty rdf:ID="Effect_Parameter"/>
6  <owl:DatatypeProperty rdf:ID="Precondition_Parameter"/>
7     <owl:Class rdf:ID="Tasks" />
8     <owl:Class rdf:ID="e_commerce_Service">
9              <rdfs:subClassOf rdf:resource="#Tasks"/>
10    </owl:Class>
11     <owl:Class rdf:ID="Information_Service">
12             <rdfs:subClassOf rdf:resource="#e_commerce_Service"/>
13   </owl:Class>
14   <owl:Class rdf:ID="Buying_Service">
15             <rdfs:subClassOf rdf:resource="#e_commerce_Service"/>
16   </owl:Class>
17   <owl:Class rdf:ID="Selling_Service">
18             <rdfs:subClassOf rdf:resource="#e_commerce_Service"/>
19   </owl:Class>

20   <owl:Class rdf:ID="Car_Price">
21      <rdfs:subClassOf rdf:resource="#Information_Service"/>
22       <Input_Parameter rdf:datatype="&Vehicle;Car">Car</Input_Parameter>
23          <Output_Parameter rdf:datatype="&Concept;Store">Car_Store</Output_Parameter>
24          <Output_Parameter rdf:datatype="&Concept;Price">Price</Output_Parameter>
25          <Output_Parameter rdf:datatype="&Concept;Currency">Result_Currency</Output_Parameter>
26    </owl:Class>
```

Örneğin, Tablo 3'teki anlamsal iş tanımları, 'İşler' (`Tasks`), 'E_Ticaret_Servisi' (`E_commerce_Service`), 'Bilgi_Servisi' (`Information_Service`), 'Alış_Servisi' (`Buying_Service`), 'Satış_Servisi' (`Selling_Service`), ve 'Araba_Fiyatı' (`Car_Price`) gibi iş örneklerini içermektedir. TOKB altında tanımlanan iş tanımları, OWL sınıfları (`<owl:class>`) üzerinden tanımlanarak, E_ticaret alanında sıklıkla kullanılan iş örneklerinin tanımlarını içermektedir.

Tablo 3'ün 7. satırında, 'İşler' (`Tasks`) sınıfı `<owl:class>` fişiyle betimlenmiştir. Tablo 3'ün 7. satırında, 'E_Ticaret_Servisi' (`E_commerce_Service`) tanımlanmış, bununla birlikte Satır 9'da `<rdfs:subClassOf>` fişiyle 'İşler' (`Tasks`) sınıfının altsınıfı gibi ifade edilmiştir. Diğer iş örneklerinin sınıfları da aynı yöntemle betimlenmiştir. TOKB'de betimlenmiş olan iş örnekleri, aday Semantik Web Servisleri'ndeki gibi I/O/P/E parametrelerine sahiptir. Her parametre kendi alan (domain) ontolojisindeki veya Konseptler Ontolojisi'ndeki (Concept.owl) bir konsepte işaret etmektedir, bu nedenle, TOKB ilk satırlarında, gerekli olan alan (domain) ontolojileri veya Konseptler Ontolojisi import edilmelidir (Tablo 3, Satır 1-2). İş örneklerini anlamsal tabanda betimlemek için, 'Girdi_Parametresi' (`Input_Parameter`), 'Çıktı_Parametresi' (`Output_Parameter`), 'Önkoşul_Parametresi' (`Precondition_Parameter`) ve 'Etki_Parametresi' (`Effect_Parameter`) gibi parametre tanımlamalarını, OWL data tür özelliği (`<owl:DatatypeProperty>`) kullanılarak betimlenmiştir (Tablo 3, Satır 3-6).

Tablo 3'ün 20-26. satırlarında, 'Araba_Fiyatı' (`Car_Price`) adında bir iş örneği tanımlanmıştır. Bu iş örneği, kullanıcı tarafından seçilen bir araba modeli bilgisine karşın, 'Fiyat' (`Price`), 'Dükkan' (`Store`) ve 'Sonuç_Parabirimi' (`Result_Currency`) gibi bilgileri geri döndürmektedir. Tablo 3'ün 22. Satırında tanımlanan, 'Girdi_Parametresi' (`Input_Parameter`) , 'Araba_Fiyatı' (`Car_Price`) servisiyle ilişkilendirilmiş, bunun yanısıra, tanımlanan bu 'Girdi_Parametresi' (`Input_Parameter`) ise bir OWL `<owl:DatatypeProperty>` özelliğidir sonucuna varılabilmektedir. Bu ilişkilendirme üzerinden, 'Araba_Fiyatı' (`Car_Price`) iş örneği, sadece bir tane 'Araba' (`Car`) adında 'Girdi_Parametresi' (`Input_Parameter`) bulunmaktadır ve bu girdi parametresinin `ParameterType` konsepti de yine 'Araba' (Car) terimidir. Diğer 'Çıktı_Parametresi' (`Output_Parameter`), 'Önkoşul_Parametresi' (`Precondition_Parameter`), ve 'Etki_Parametresi' (`Effect_Parameter`) gibi parametre tanımlamaları da, aynı biçimde tanımlanmıştır. Verilen 'Araba_Fiyatı' (`Car_Price`) iş örneği sadece bir çıktı parametresi almış olup, herhangi bir önkoşul ve etki parametreleri

içermemektedir (Tablo 3, Satır 23-25). Bu method sayesinde, çeşitli iş örnekleri, anlamsal iş içerikleriyle tanımlanabilir.

4.3. Süreç Ontolojileri Bilgi Tabanı (Processes Knowledgebase-POKB). Aday Web servisleri'ne (Candidate Semantic Web Services-CSWSs) ait atomik süreçlerin betimlendiği OWL-S dosyaları, POKB bilgi tabanında tutulmaktadır. POKB, ilgili alana (domain) ait Aday Web Servisleri'nin OWL-S dosyalarının saklanıldığı bir depo gibidir. SCA prototipinin hayata geçirilmesinde, OWL-S tabanlı atomik süreçleri içeren ve yaygın olarak da Semantik Web alanındaki araştırmalarda kullanılan bir test koleksiyonu (OWLS-TC3.zip) kullanılması tasarlanmıştır. OWL-S test koleksiyonu, SemWebCentral Websitesinde[1] yer almaktadır. Bu bölümde, uygun Aday Web Servisleri'nin keşif işleminin birleşim işlemine başlamadan önce yapıldığı farzedilmiştir. Bunun için daha önceki çalışmalarımızdan uygun servisi arama ve keşifi işlemlerini anlamsal tabanda yapabilen ajan tabanlı bir sistemin, birleşim planına uygun servislerin keşif işlemini tamamladığını ve POKB içinde tuttuğunu farzediyoruz [10-16].

5. Semantik Uyum Adımı (Semantic Matching Step-SMS)

SCA, Birleşim İş Akışı Planı'nını yapılandırırken, Semantik Uyum Adımı'nı çalıştırır, ve Aday Web Servisleri'nin atomik süreçlerinin betimlendiği OWL-S dosyalarına girer. Birleşim İş Akışı Planı'nı yapılandırırken, birleştirilecek süreçler arasındaki uyumu anlamak için, süreçler arasında anlamsal tabanda kıyaslama işlemi gereklidir. Aday Semantik Web Servisleri'nin, OWL-S dosyalarında betimlenmiş olan atomik süreçlerin girdi/çıktı bilgileri SCA'nın Çevirici'si tarafından okunur. SMS'deki en önemli nokta, anlamsal olarak yakın terimleri/konseptleri ıskalamamaktır. Burada OWL tabanlı alan (domain) ontolojileri (COKB) devreye girer. Birleşim İş Akışı Planı'ndaki ya da zincirindeki, en son odaklanılan sürece ait girdi/çıktı konseptleri ile diğer tüm ilgili servislerinin süreçlerine ait girdi/çıktı konseptleri kıyaslanırken akrabalık derecelerine göre bir benzerlik değeri hesaplanır. Bu nedenle, SCA, yeni yapılandırdığı Birleşim İş Akışı Planı'ndaki en son sürece ait girdi/çıktı bilgilerini hafızada tutar. Daha sonra tüm Aday Web Servisleri'nin girdi/çıktı bilgilerini ve bir önceki sürece ait girdi/çıktı bilgilerini, Semantik Uyum Adımı algoritmasını (bkz. Şekil-3) kullanarak uygun OWL alan (domain) ontolojileri üzerinden anlamsal tabanda karşılaştırır. Her servisin atomik süreci için, aşağıdaki verilen Formül 1

[1] http://www.semWebcentral.org

üzerinden bir benzerlik skoru hesaplanmaktadır:

$$\text{Benzerlik}_{\text{skoru}(C_I,A_I)} \equiv \frac{\sum_{j=1}^{\max(n_{C_I},n_{A_I})} \left(\max_{j,i} \left\{ \prod_{i=1}^{\max(n_{C_I},n_{A_I})} \frac{\text{SMS}_{\text{skoru}(C_i,A_j)} * d_{\text{ağırlık}(C_i,A_j)}}{\text{uzaklık}(C_i,A_j)} \right\} \right)}{n_{C_I}/n_{A_I}}. \quad (1)$$

Burada, n_C =girdiSayısı(C_I) ve n_A =girdiSayısı(A_I); ya da, n_C =girdiSayısı(C_0) ve n_A =girdiSayısı(A_0); uzaklık(C_i, A_j), alan ontolojisindeki C_i ve A_j konseptleri arasındaki level sayısıdır.

İki süreç arasında benzerlik skoru hesaplanırken, bu iki sürecin girdi/çıktı konseptleri arasında anlamsal tabanda kıyaslama, alan (domain) ontolojisinde betimlenmiş alt veya üst konsept ilişkileri tespit edilerek yapılır. Örneğin, Konsept A ve Konsept Z arasındaki ontolojik mesafe için $d_{\text{ağırlık}}(A, Z)$ aşağıdaki gibi hesaplanır [21]:

$$d_{\text{ağırlık}(A,Z)} = d_{\text{ağırlık}(A,B)} * d_{\text{ağırlık}(B,C)} * \cdots * d_{\text{ağırlık}(W,Y)} * d_{\text{ağırlık}(Y,Z)}. \quad (2)$$

Örneğin, zincirdeki en son sürece bağlanacak en uygun Web servisin sürecini aramak için girdi teriminin 'Araba' (`Car`) ve çıktı teriminin 'Fiyat' (`Price`) olduğunu düşünelim. Yani zincire eklenecek yeni süreç, araba satışı yapan bir servis olsun. Bu durumda OWL-S dosyalarında, sadece 'Araba' (`Car`) kelimesini girdi bilgisi olarak içeren Aday Semantik Web Servisleri'ni incelemek yanlış olacaktır. Çünkü, diğer elenen Web servislerine ait OWL-S dosyalarında 'Araba' (`Car`) girdisi yerine, 'Araba' (`Car`) konseptinin altkümesi veya üst kümesi olan başka konseptler olabileceğinden, bu Web servisleri ıskalanacaktır. Örneğin, 'Mercedes', 'Otomobil' (`Automobile`) veya 'Araç' (`Vehicle`) terimleri, 'Araba' (`Car`) terimi ile aynı alanda yer almaktadır ve bunları içeren Aday Semantik Web Servisleri, istenilen Birleşim İş Akışı Planı'nı oluştururken, SCA'nın aradığı niteliklerde bir Web servis olabilmektedir.

Semantik Uyum Adımı, Massimo Paolucci ve grubu [17-19] tarafından keşfedilen ve daha sonra diğer araştırmacılar tarafından yaygın olarak çalışılan [4,7,8,14,20-24] Uyum Algoritması'ndan türetilerek uygulanmıştır. Massimo Paolucci ve grubu bu çalışmayı UDDI'ya bir eklenti şeklinde tasarladılar. Fakat UDDI deposu bir standart olduğu için onun yapısını değiştirmek yerine, SCA gibi Web ortamında çalışan bir sistem üzerine yerleştirmek daha doğru olacaktı. Kullanılan uyum görevi geliştirildikçe veya değiştirildiğinde sisteme entegrasyonu daha kolay olacaktır.

Sistem, iş planındaki en son Web servisine ait girdi/çıktı terimler ile diğer servislere ait girdi/çıktı terimleri arasında bazı ilişkiler belirleyebilmektedir demiştik. Örneğin;

AYNI, KAPSANIR, KAPSAR ve BASARISIZ. Sırayla ilişkiler ve değerleri hesaplarken kullanılan katsayılar (Formül 1'de $\text{SMS}_{\text{skoru}(C,A)}$ olarak gösterilmiştir) şöyledir:

- AYNI - (Exact: 1 puan)
- KAPSANMA - (Plug in: 0.75 puan)
- KAPSAR - (Subsume: 0.5 puan)
- BASARISIZ - (Fail: 0 puan)

`hasSyn(Y)`≡`Z`: orada bir konsept vardır ki, o konsept `Z` olsun, `Y` konseptinin eşanlamlısıdır; 'Araba' (`Car`) konseptinin eşanlamlısı 'Otomobil' (`Automobile`) ya da 'MotorluAraba' (`Motorcar`)'dır.

`hasIs_A(Y)`: orada bir konsept vardır ki, o konsept `Y` konseptinin bir_örneğidir; '3TekerlekliAraba' (`3WheeledCar`), bir 'Araba' (`Car`)'dır.

İş planını oluştururken, gelecek düğümdeki uygun Web servise ait sürecin, bir önceki düğümdeki Web servisin sürecinin sahip olduğu konseptlerle karşılaştırarak seçmektedir. Aşağıdaki SMS'in çalışma prensibini anlatmaktadır. Eğer süreçleri kıyaslanan konseptleri birbirinin aynısı ise, sistem ilişkiyi AYNI olarak nitelendirecektir ve bu ilişkiye en büyük değeri atayacaktır yani $\text{SMS}_{\text{skoru}(C,A)}$ değerini '1' olarak hesaplayacaktır.

Eğer kıyaslanan terimlerden, bir önceki düğümdeki girdi/çıktı terimi, gelecek düğüm için seçilen Web servisinin girdi/çıktı teriminin altkümelerinden biri ise, sistem ilişkiyi KAPSANIR olarak nitelendirecektir, ve bu ilişkiye ikinci büyük değeri atayacaktır yani 0.75 olarak hesaplayacaktır (bkz. Formül 1'de $\text{SMS}_{\text{skoru}(C,A)}$ olarak gösterilmiştir). Diğer ilişkiler sırayla Şekil 3'teki algoritma üzerinde gösterilmektedir. Örneğin, kullanıcı girdi terimi olarak 'Araba' (`Car`) terimini yazmış olsun. Web servisinin OWL-S dosyasında, process modülündeki bilgilerinde, saklı olan girdi terimi ise 'Araç' (`Vehicle`) olmuş olsun. Bu durumda 'Araba' (`Car`) terimi 'Araç' (`Vehicle`) teriminin bir alt kümesidir, yani 'Araba' (`Car`) terimi 'Araç' (`Vehicle`) terimi tarafından kapsanmaktadır. Sistem, $\text{Benzerlik}_{\text{skoru}(C,A)}$ hesaplarken, benzerlik derecesi için KAPSANIR ilişkisini tanımlayacak ve ardından benzerlik yüzdesi hesaplarken, $\text{SMS}_{\text{skoru}(C,A)}$ değerini 0.75 alacaktır.

Eğer, kıyaslanan süreçlerin daha fazla girdi/çıktı terimi var ise, bu durumda, toplam değer puanları ve toplam terim sayısı hesaba katılarak bir yüzdelik derecesi hesaplanmaktadır. Eğer, benzerlik yüzdesi %85-%100 arasında ise, bu süreç Birleşim İş Akışı Planı'nına eklenecek yeni sürecin kendisi ya da çok benzeri nitelikte bir süreç olma ihtimali yüksektir.

Algoritma 1 SMS (Konsept C_i, Konsept A_i)
1 Eğer ((Ci≡Ai) **yada** (hasSyn(Ci)≡Ai) **yada** (Ci≡hasSyn(Ai))) **öyleyse** return rel= **AYNI**;
2 Eğer ((Ci⊂Ai) **yada** (hasSyn(Ci)⊂Ai) **yada** (Ci⊂hasSyn(Ai)) **yada** (hasIs_a(Ci)≡Ai)) **öyleyse** return rel= **KAPSANMA**;
3 Eğer ((Ci⊃Ai) **yada** (hasSyn(Ci)⊃Ai) **yada** (Ci⊃hasSyn(Ai)) **yada** (Ci≡hasIs_a(Ai))) **öyleyse** return rel= **KAPSAR**;
4 Eğer ((Ci≠Ai) **yada** (hasSyn(Ci)≠Ai) **yada** (Ci≠hasSyn(Ai))) **öyleyse** return rel= **BASARISIZ**;

Şekil 3. Semantik Uyum Adımı (Semantic Matching Step-SMS).

C_i & A_i anlamı, uyum taraması yapılan iki süreç'e (C ve A) ait i. girdi (ya da çıktı) konseptlerini göstermektedir

6. Revize Armstrong Aksiyomları (Revised Armstrong's Axioms-RAA)

'Armstrong Aksiyomları' (Armstrong's Axioms-AA) bir aksiyomlar/çıkarsama kuralları kümesidir. AA'ları, 'İlişiksel Veritabanları'nda (Relational Database) rastlanan 'Fonksiyonel Bağlılıkları' (Functional Dependencies-FDs) çıkarsamak için kullanılmaktadır. Tablo 4'te görüldüğü üzere yedi farklı çıkarsama kuralı vardır: Reflexivity, Augmentation, Transitivity, Pseudo Transitivity, Additivity, Accumulation, ve Projectivity.

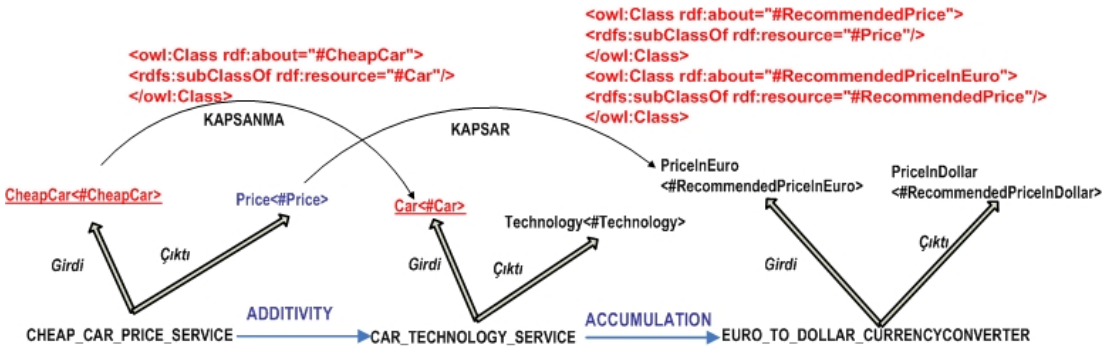
SCA'nın, CSWSs'lerini doğru olarak birleştirebilmesi için, Reflexivity ve Augmentation çıkarsama kurallarını kombine ettik, böylece Pseudo Factorization aksiyomunu meydana getirdik. Bu aksiyomu geliştirmemizdeki amaç, Reflexivity ve Augmentation aksiyomlarının, yeni süreçlerin türetimini daha kompleks hale getirmesini, ilaveten uyumsuz süreç türetimlerini engellemekti. Ayrıca, Projectivity aksiyomunu, Dissection şeklinde yeniden adlandırdık. 'Revize Armstrong Aksiyomları' (Revised Armstrong's Axioms-RAA), AA aksiyomlarının eşdeğeri dönüşümlerle yapılandırılmış ve bu dönüşümler esnasında AA aksiyomlarının doğruluğu ve bütünlüğü bozulmamıştır. Tablo 4'ün ilk sütununda, AA aksiyomları, ikinci sütununda da RAA aksiyomları verilmiştir.

Farz edelim ki, X, Y, Z, W, ve T, elde var olan bir süreç kümesine ait tüm girdi/çıktı parametrelerinin konseptleri olsun, ve girdi/çıktı parametrelerinin ortasında kullanılan '$\rightarrow$' sembolü ise, doğrusal dönüşümü ifade etsin. Tablo 4'te gösterilen $X \| Y \rightarrow Z$ formu, bir sürecin mantıksal ifadesidir, burada Konsept X ve Konsept Y eşzamanlı ('$\|$'- eşzamanlı sembolü) girdi parametrelerini göstermektedir. Sürecin yürütülmesi esnasında girdi konseptlerinin, X ve Y, eş zamanlı alınmasından ötürü sıranın öneminin olmadığı anlamına gelmektedir. Ek olarak, tek bir çıktı parametresi vardır, Z konseptidir.

TABLO 4. Armstrong Aksiyomları ve Revize Armstrong Aksiyomları.

AA	**RAA**
1. Reflexivity Kuralı: Y, X'in bir alt kümesiyse $X \to Y$ fonksiyonel bağımlılığı vardır: Biçimsel ifadeyle, $X \to Y$ if $Y \subseteq X$. 2. Augmentation Kuralı: Eğer $X \to Y$ ise , $X\|Z \to Y\|Z$ fonksiyonel bağımlılığı vardır. Biçimsel ifadeyle, eğer $X \to Y$ öyleyse $X\|Z \to Y\|Z$ herhangi Z için.	1.Pseudo factorization Kuralı: Eğer $\{(X\|Y \to Z), (T\|Z \to W)\}$ ve eğer $(Z \subset T\|Z$ ve $T \equiv X$ or $T \equiv Y)$ öyleyse $X\|Y \to W$.
3. Transitivity Kuralı: Eğer $X \to Y$ ve $Y \to Z$ ise $X \to Z$ fonksiyonel bağımlılığı vardır. Biçimsel ifadeyle, eğer $\{(X \to Y)$ ve $(Y \to Z)\}$, öyleyse $X \to Z$.	2.Transitivity Kuralı: Eğer $\{(X \to Y)$ ve $(Y \to Z)\}$ öyleyse $(X \to Z)$.
4. Pseudo transitivity: Eğer $X \to Y$ ve $Y\|Z \to W$ ise $X\|Z \to W$ fonksiyonel bağımlılığı vardır. Biçimsel ifadeyle, eğer $\{(X \to Y)$ ve $(Y\|Z \to W)\}$ öyleyse $(X\|Z \to W)$.	3.Pseudo transitivity Kuralı: Eğer $\{(X \to Y)$ ve $(Y\|Z \to W)\}$ öyleyse $(X\|Z \to W)$.
5. Additivity Kuralı: $X \to Y$ ve $X \to Z$ ise $X \to Y\|Z$ fonksiyonel bağımlılığı vardır. Biçimsel ifadeyle, eğer $\{(X \to Y)$ ve $(X \to Z)\}$ öyleyse $(X \to Y\|Z)$.	4. Additivity Kuralı: Eğer $\{(X \to Y)$ ve $(X \to Z)\}$ öyleyse $(X \to Y\|Z)$.
6. Accumulation Kuralı: $X \to Y\|Z$ ve $Z \to C\|W$ ise $X \to Y\|C\|W$ fonksiyonel bağımlılığı vardır. Biçimsel ifadeyle, eğer $\{(X \to Y\|Z)$ ve $(Z \to C\|W)\}$ öyleyse $(X \to Y\|C\|W)$.	5. Accumulation Kuralı: Eğer $\{(X \to Y\|Z)$ ve $(Z \to T\|W)\}$ öyleyse $(X \to Y\|T\|W)$.
7. Projectivity Kuralı: Eğer $X \to Y\|Z$ ise $X \to Y$ ve $X \to Z$ fonksiyonel bağımlılığı vardır. Biçimsel ifadeyle, eğer $(X \to Y\|Z)$ öyleyse $X \to Y$ ve $X \to Z$.	6.Dissection Kuralı: Eğer $(X \to Y\|Z)$ öyleyse $X \to Y$ ve $X \to Z$.

Olası bir senaryoya göre; bir Web kullanıcısı ucuz fiyatta bir araba aramaktadır, ilaveten bu araba için en son teknoloji bilgilerini de öğrenmek istemektedir. Sonuç olarak dönecek fiyat bilgisinin de United States Dollar cinsinden olmasını istemektedir. Bu durumda, SCA sistemi, bu konseptleri girdi/çıktı olarak içeren, POKB deposunda elde var olan süreç kümesinden, eşdeğer bir süreç tarayacaktır. Eğer bu süreç bulunamazsa yada tanımlanamazsa, elde var olan süreçlerin birleşiminden istenilen Kompleks Süreç'i oluşturmaya başlıyacaktır.



ŞEKİL 4. Birleşim İş Akışı Planı'nın yapılandırılmasında RAA'nın kullanımını gösteren basit örnek.

Farzedelim ki istenilen Kompleks Süreç, yani `G`, aşağıdaki şekilde SCA'nın Çevirici'si tarafından biçimlendirilmiş olsun:

`G≡(Car[#Car]≡Technology[#Technology]||PriceOfCar[#RecommendedPriceInDollar])`

'[]' içinde verilen terim, o sürecin OWL-S betimlemesinde tanımlanmış Parametre Türü'ne (`parameterType`) ait konsepti göstermektedir. '[]' dışında kalan terim ise, o süreç'in OWL-S betimlemesinde tanımlanmış `rdf:ID` (girdi-çıktı) konseptlerini göstermektedir.

Diyelim ki, POKB altında yer alan on farklı OWL-S tabanlı Aday Web Servisleri'ne ait atomik süreçler olsun. Şekil 4'te, SCA'nın Planlayıcı'sı ve Çıkarsama Motoru tarafından, Kompleks Süreç'in türetilmesinde bu on farklı süreçlerin birleşiminden bir çok çözüm yolu bulabilirler, bunlardan biri, verilen sıra içinde; Cheap_Car_Price.owl, Car_Technology.owl, ve EuroToDollarCurrencyConverter.owl süreçlerinin birleşimi olabilir. Aşağıda birleşmenin biçimsel gösterimi yapılmıştır:

`Servis1≡Cheap_Car_Price[CheapCar<#CheapCar>] [Price<#Price>]`

`Servis9≡Car_Technology[Car<#Car>][Technology<#Technology>]`

`Servis7≡EuroToDollarCurrencyConvertor≡[PriceInEuro<#RecommendedPriceInEuro>]→`
`[PriceInDollar<#RecommendedPriceInDollar>]`

```
Üretilen Yeni Süreç(P)≡[[Cheap_Car_Price.owl○Car_Technology.owl]○
EuroToDollarCurrencyConvertor.owl]]
≡{[ CheapCar<#CheapCar>]→[Price <#Price>]}○{[Car<#Car>]→
[Technology#Technology>]}

{[ CheapCar<#CheapCar>]→[Technology<#Technology>]||[Price <#Price>]}
{[ CheapCar<#CheapCar>]→[Technology<#Technology>]||[Price <#Price>]}○
EuroToDollarCurrencyConvertor.owl
≡{[CheapCar<#CheapCar>]→[Technology<#Technology>] || [Price <#Price>]}○
{[PriceInEuro<#RecommendedPriceInEuro>]→
[PriceInDollar<#RecommendedPriceInDollar>]}
P≡{[CheapCar<#CheapCar>]→[Technology<#Technology>]||
[PriceInDollar<#RecommendedPriceInDollar>]}
```

Süreçler arasında kullanılan '○' sembolü, o süreçlerin arasında RAA'lardan birinin uygulanabileceği anlamında, yani senkronizasyonun varlığını tanımlamaktadır. İki sürecin birbiriyle senkronize olması için; bir önceki sürecin çıktı kümesindeki her konsept, takibindeki sürecin girdi kümesindeki her bir konsept ile bire bir anlamsal tabanlı ilişkisi olmalıdır. Ya da, bir önceki sürecin girdi kümesindeki her konsept, takibindeki sürecin girdi kümesindeki her bir konsept ile bire bir anlamsal tabanlı ilişkisi olmalıdır. Eğer biz önce Additivity ve daha sonrada Accumulation aksiyomunu uygularsak, olası süreç zinciri aşağıdaki şekilde olacaktır;

```
Accumulation [Additivity [Cheap_Car_Price○ Car_Technology]○
EuroToDollarCurrencyConvertor]]
```

Additivity aksiyomunu Servis 1 ve Servis 9'un üzerinde uygulamadan önce, SMS tarafından bu servislerin süreçlerinin girdi konseptleri üzerinde, KAPSANMA (Plug in: 0.75 puan) ilişkisi ele geçirilecektir. Çünkü SMS tarfından, 'UcuzAraba' (`CheapCar`) konsepti 'Araba' (`Car`) konseptinin alt kümesi olarak tespit edilmiştir (bkz. Şekil 4). Bunun yanısıra, yeni türetilen bileşik süreç (Servis ve Servis 9), ve Servis 7'nin atomik süreci ile, Accumulation kuralına göre birleştiğinde, SMS tarafından, KAPSAR (Subsume: 0.5 puan) ilişkisi, 'Fiyat' (`Price`) ve 'EuroCinsindenÖnerilenFiyat' (`RecommendedPriceInEuro`) konseptleri arasında tespit edilecektir. Sonuç olarak türetilen yeni süreç, yani `P`, SCA'nın Planlayıcı'sı ve Çıkarsama Motoru tarafından aşağıdaki şekilde tanımlanacaktır:

```
P≡{[CheapCar<#CheapCar>]→[Technology<#Technology>]||
[PriceInDollar<#RecommendedPriceInDollar>]}.
```

Sonuç olarak `P`⊨`G`. Böylece servislerin süreçleri arasındaki bağlılıkları anlamsal betimlemeler yolu ile ontoloji dosyalarına tanımlayıp, bu tanımlar üzerinden çıkarsama

yoluyla yeni bir süreç oluşturmak için gerekli birleşim yapılarak, kullanıcının aradığı servis türünü yakalamak mümkün olabilmektedir.

7. Sonuçlar

Bu makalede atomik yapıdaki süreçlerin sözdizimsel yerine anlamsal tabanda birleşimini sağlamak amacıyla gerekli olan semantik iş akışı planının üreten, Çıkarsama-tabanlı Semantik Birleşim Ajanı (Inference-based Semantic Composition Agent-SCA) diye isimlendirdiğimiz bir sistem önerdik. Geliştirilen bu sistemin en önemli modülleri; Çevirici (translator), Planlayıcı (planner), Çıkarsama Motoru (inference engine), Çalıştırma Motoru (execution engine), ve Denetleyici (monitoring agent)'dir. Bunların yanısıra, SCA, süreçler arası uyum tarama mekanizması olan, Semantik Uyum Adımı (Semantic Matching Step-SMS)'nı içermektedir. Semantik Uyum Adımı, Aday Semantik Web Servisleri'ne (Candidate Semantic Web Services-CSWSs) ait tüm süreç'ler arasındaki anlamsal bağlılıkları tespit edip, bu bağlılıklar için total benzerlik skorları hesaplar. Ayrıca SCA, üç farklı ontoloji bilgi tabanı vardır: Konseptler Ontolojisi Bilgi Tabanı (Concepts Knowledgebase-COKB), İş Örnekleri Ontolojisi Bilgi Tabanı (Task Ontology-TOKB), ve son olarak Süreç Ontolojileri Bilgi Tabanı (Processes Knowledgebase-POKB)'dır. SCA, kendi çıkarsama kurallarını kullanmaktadır ve bununların saklandığı Çıkarsama Kuralları Bilgi Tabanı (Rule knowledge base-RKB)'nı vardır. İlk defa bu çalışmada, Armstrong'un Çıkarsama Kuralları revize edilmiş ve Semantik Web uygulamalarında, planlama ve çıkarsama işlevlerinde kullanılmıştır. RKB, bu yüzden kendi 'Revize Armstrong Aksiyomları' (Revised Armstrong's Axioms-RAA)'nı içermektedir. Sistemin en büyük avantajı geliştirilmeye açık olmasıdır. Yani uygun birleşmeyi sağlamak ve Kompleks Süreç'i meydana getirebilmek için doğru süreçleri yakalamak ontolojiler ile mümkündür. SCA'nın SMS modülünde, sadece iki çeşit benzer kelime (eş anlamlı & bir_örneğidir) üzerinde arama yapıldı. Eğer daha fazla kelime benzerlikleri kullanılsa (örneğin bir kelime başka bir kelimenin zıt anlamlısı, sonucu, etkisi veya kısaltması gibi ilişkiler yaratılarak), Birleşim İş Akışı Planı'nı oluşturmada, zincire eklenecek uyumlu süreçleri yakalama oranı/verimlilik daha yüksek olacaktır.

Teşekkür. Bu çalışma, Doğu Akdeniz Üniversitesi A tipi bilimsel araştırma projesi olarak desteklenmiştir. Proje 'Provision of Semantic Web Services through an Intelligent Semantic Web Service Finder' adı altında BAP-A-07-20 numarasıyla kayıtlıdır.

Kaynaklar

[1] W3C. OWL Web Ontology Language Overview. `http://www.w3.org/TR/owl-features/`, 2004. Online; accessed 24-May-2011.

[2] S. A. McIlraith, T. C. Son and H. Zeng, Semantic Web services, *IEEE Intelligent Systems* **16** (2001), 46–53.

[3] W3C. OWL-S: Semantic Markup for Web Services. `http://www.w3.org/submission/owl-s/`, 2004. Online; accessed 24-May-2011.

[4] R. Akkiraju, B. Srivastava, A. Ivan, R. Goodwin and T. Syeda-Mahmood, Semantic matching to achieve Web service discovery and composition, *Proceedings of the 8th IEEE International Conference on E-Commerce Technology and the 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06)* (2006), 70–70.

[5] E. Sirin, B. Parsia and J. Hendler, Composition-driven filtering and selection of semantic Web services, *AAAI Spring Symposium on Semantic Web Services* (2004).

[6] W. W. Armstrong, Dependency structures of data base relationships, *Proceedings of IFIP World Computer Congress* (1974), 580–583.

[7] A. B. Bener, V. Ozadali and E. S. Ilhan, Semantic matchmaker with precondition and effect matching using SWRL, *Expert Systems with Applications* **36** (2009), 9371–9377.

[8] U. Bellur and H. Vadodaria, On extending semantic matchmaking to include preconditions and effects, *IEEE International Conference on Web Services, 2008. ICWS'08* (2008), 120–128.

[9] OWL-S Service Retrieval Test Collection. Version 3.0. `http://projects.semWebcentral.org/projects/owls-tc/`, 2009-11-11. Online; accessed 24-May-2011.

[10] D. Çelik and A. Elçi, Provision of semantic Web services through an intelligent semantic Web service finder, *Multiagent and Grid Systems* **4** (2008), 315–334.

[11] D. Çelik and A. Elçi, Ontology-based QoS model for appropriate selection and composition of Web services, *International Review on Computers and Software* **3** (2008), 176–184.

[12] D. Çelik and A. Elçi, Akıllı anlamsal ağ servisi arayıcısı, *Türkiye Bilişim Vakfı, Bilgisayar Bilimleri ve Mühendisliği Dergisi* **2** (2006), 31–42.

[13] D. Çelik and A. Elçi, A semantic search agent approach: finding appropriate semantic Web services based on user request term(s), *Enabling Technologies for the New Knowledge Society: ITI 3rd International Conference on Information and Communications Technology ICICT'2005* (2005), 675 -687.

[14] D. Çelik and A. Elçi, Searching semantic Web services: An intelligent agent approach using semantic enhancement of client input term(s) and matchmaking step, *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-2 (CIMCA-IAWTIC'05)* (2005), 916–922.

[15] D. Çelik and A. Elçi, A semantic search agent discovers suitable Web services according to an e-learning client demand, *Proceedings of 6th International Educational Technology Conference (IETC 2006) Gazimagusa, TRNC, 19-21 April 2006* **1** (2006), 416–424.

[16] D. Çelik and A. Elçi, Discovery and scoring of semantic Web services based on client requirement(s) through a semantic search agent, *COMPSAC '06. 30th Annual International Computer Software and Applications Conference* **2** (2006), 273–278.

[17] M. Paolucci, T. Kawamura, T. R. Payne and K. P. Sycara, Semantic matching of Web services capabilities, *Proceedings of the First International Semantic Web Conference on The Semantic Web ISWC '02* (2002), 333–347.

[18] M. Paolucci, T. Kawamura, T. R. Payne and K. Sycara, Importing the Semantic Web in UDDI, *Web Services, E-Business, and the Semantic Web, Lecture Notes in Computer Science* **2512** (2003), 815–821.

[19] T. Kawamura, J. A. De Blasio, T. Hasegawa, M. Paolucci and K. Sycara, Preliminary report of public experiment of semantic service matchmaker with UDDI business registry, *International Conference on Service Oriented Computing (ICSOC 2003), Trento, Italy,* (2003), 208–224.

[20] U. Bellur, H. Vadodaria and A. Gupta, Semantic matchmaking algorithms, *Advances in Greedy Algorithms* (2008), 481–502.

[21] E. S. Ilhan, G. B. Akkuş and A. Bener, SAM: Semantic advanced matchmaker, *International Conference on Software Engineering and Knowledge Engineering* (2007), 698–703.

[22] H. Wang and Z. Li, Capability matchmaking of semantic Web services with preconditions and effects, *CSWS 2009: The 3rd Chinese Semantic Web Symposium*, Nanjing, China, 29-31 August 2009.

[23] J. Wu and Z. Wu, Similarity-based Web service matchmaking, *2005 IEEE International Conference on Services Computing* **1** (2005), 287–294.

[24] M. Şenvar and A. Bener, Matchmaking of semantic Web services using semantic-distance information, *Advances in Information Systems, Lecture Notes in Computer Science* **4243** (2006), 177-186.

[25] D. Kang, S. Lee, K. Kim and J. Y. Lee, An OWL-based semantic business process monitoring framework, *Expert Systems with Applications* **36** (2009), 7576–7580.

[26] E. Sirin, B. Parsia, D. Wu, J. Hendler and D. Nau, HTN planning for Web service composition using SHOP2, *Web Semantics: Science, Services and Agents on the World Wide Web* **1** (2004), 377–396.

[27] E. Sirin and B. Parsia, Planning for semantic Web services, *Semantic Web Services Workshop at 3rd International Semantic Web Conference (SWS-ISWC04)* (2004).

[28] O. Aydin, N. K. Cicekli and I. Cicekli, Towards automated Web service composition with the abductive event calculus, *Proceedings of Applications of Logic Programming in the Semantic Web and Semantic Web Services (ALPSWS 2006)* (2004), 103–104.