Etmen-Servis Tümleşimi İçin Bir Etmen Altyapısı

Övünç Çetin[1] Beytun Özkan[2] Mustafa Savaşçı[3]
Onur Ulusu[4] Yiğit Çetin[5] Erdem Eser Ekinci[6] Oğuz Dikenelli[7]

[1,2,3,4,5,6,7]Bilgisayar Mühendisliği Bölümü, Ege Üniversitesi, İzmir

[1]e-posta: ovunccetin@gmail.com [2] e-posta: beytunozkan@gmail.com
[3] e-posta: mustafasavasci@gmail.com [4] e-posta: onurulusu@gmail.com
[5] e-posta: yigitcetin@gmail.com [6] e-posta: erdemeserekinci@gmail.com
[7] e-posta: oguzdikenelli@ege.edu.tr

Özetçe

Günümüzde veb servisleri, sayıları hızla artan ve endüstriyel çevrelerce yoğun bir şekilde kullanılan vazgeçilmez bir teknoloji haline gelmiştir. Ayrıca, anlamsal veb teknolojilerindeki gelişmelerle birlikte servislerin daha etkin kullanımı için büyük fırsatlar doğmakta, bu da anlamsal veb servisleri konusundaki araştırmaların önünü açmaktadır. Diğer taraftan, etmenler ve çoklu-etmen sistemleri bilgisayar bilimcileri tarafından büyük ilgi gören ancak henüz endüstriyel olgunluğa ulaşamamış güncel araştırma konularıdır. Bu çalışmada etmen-servis tümleşimini sağlamak amacıyla anlamsal servisleri destekleyebilen bir etmen altyapısı önerilmektedir. Önerilen altyapı SEAGENT çoklu etmen sistemi geliştirme çerçevesi üzerine arttırım yapılarak geliştirilmiştir.

1. Giriş

Gün geçtikçe yaygınlaşan ve gelişen veb servis teknolojisi, heterojen ve az bağımlı yazılım bileşenlerini bir araya getirerek internet üzerinde dağıtık bir ortam oluşturmaktadır. Yeniden kullanılabilir yazılım ürünleri olan veb servisleri sayesinde istemciler, gereksinimlerini internet üzerine yayılmış olan hazır hizmetlerden karşılayabilirler ya da birden çok servisi bir araya getirerek daha karmaşık görevleri yerine getirebilirler. Ancak, veb servisleri büyük bir hızla çoğaldıkça, uygun servislerin bulunması ve bulunan servislerin bir araya getirilmesi oldukça zahmetli bir iş olmaktadır. Dahası, servislerin farklı kurumlarca farklı kavramsal modeller kullanılarak tanımlanması bu işi daha da karmaşık bir hale getirmektedir.

Tüm bu sorunların üstesinden gelmek için *anlamsal veb*[1] ve özellikle onun bir alt dalı olan *anlamsal veb servisleri* alanında önemli gelişmeler sağlanmıştır. OWL-S[1], WSMO[2] gibi anlamsal servis teknolojilerini doğuran bu gelişmeler ile Internet üzerine gelişi güzel dağılmış olan bilgi ve servisin, bilgisayarların anlayabileceği ve yorumlayabileceği şekilde tanımlanması amaçlanmaktadır. Servis içerik ve yeteneğinin bu şekilde tanımlanması ile servisin ne yaptığı, nasıl kullanılması gerektiği gibi çok büyük öneme sahip sorular bilgisayarlar tarafından yorumlanabilecektir. Bu da servislerin insan müdahalesi olmadan keşfedilmesini, birleştirilmesini ve çağrılmasını sağlayacaktır.

Ancak, servislerin anlamsal teknolojiler kullanılarak tanımlanması otomatik servis kullanımı için, bir başka deyişle kullanıcıdan bağımsız servis çağrımı için, yeterli değildir. Bu tanımlamaları kullanarak uygun servisleri keşfedebilecek, duruma göre kullanıcı yerine karar verip servisler arasından seçim yapabilecek ve servis çağrım sürecini yürütüp yönetebilecek *özerk* yazılımlara ihtiyaç vardır. Bu noktada *yazılım etmenleri* ümit verici bir yaklaşım olarak görülmektedir. İçinde bulundukları ortamı algılayabilme ve algıları doğrultusunda kararlar alıp eylemde bulunabilme yetilerine sahip olan etmenler [3], planlama yetenekleri sayesinde kullanıcılarının yerine davranarak anlamsal servis işletim sürecinde önemli roller oynayabilirler.

Bu çalışmada, OWL-S ile tanımlanmış anlamsal veb servislerini çalıştırabilen bir etmen altyapısının gerçekleştirimi üzerinde durulmaktadır. Bu amaçla, veb servislerinin etmen bilgitabanına nasıl dahil edileceğini belirten bir kavramsal model önerilmiştir. *Eylem üst modeli* olarak adlandırılan bu model ile etmenin hedefleri, davranışları ve etkileşimde bulunduğu veb servislerinin etmen eylem modelindeki yeri tanımlanmıştır. Ayrıca, hem etmen davranışlarını hem de OWL-S servislerini bir arada işletebilecek bir planlayıcı tasarımı da bu çalışma kapsamında sunulmuştur. Önerilen planlayıcı mimarisi, etmen planlarıyla birlikte BPEL[4], OWL-S, gibi çeşitli süreç modellerini de destekleyebilecek bir iş-akış modelini baz almaktadır. Böylece, farklı iş-akış dillerini ve HGA[5], STRIPS[6] gibi farklı planlama tekniklerini bir arada destekleyebilecek melez bir altyapı tasarlanmaktadır. Önerilen fikirlerin gerçekleştirimi için araştırma grubumuz tarafından geliştirilen ve anlamsal

[1] http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/

veb tabanlı bir ÇEŞ (Çoklu-Etmen Sistemi) geliştirim çerçevesi olan SEAGENT[7] kullanılmıştır.

Literatürde, anlamsal servisleri etmen sistemlerine dahil etmeye çalışan ve anlamsal servis tümleşimlerini planlayıcı araçları ile gerçekleştirmeye odaklanan birçok yaklaşım vardır. [8] ve [9]'da Dikenelli vd., SWSA [1] kavramsal modelini destekleyen anlamsal servisler için bir çoklu etmen altyapısı önermektedirler. [10]'da Gibbins vd., bir insani yardım senaryosuna ait DAML-S [2] servislerini bir çoklu-etmen altyapısı kullanarak gerçekleştirmişlerdir. Diğer taraftan Şirin vd., [11]'deki çalışmalarında DAML-S süreç modellerini HGA planlarına çevirmişler ve otomatik servis birleştirimini sağlamak için SHOP2[12] planlayıcısından yararlanmışlardır. Benzer şekilde, [13]'deki çalışmalarında Klusch vd., PDDL[14] diline dayanan bir planlama yaklaşımı sunmuşlardır. Öte yandan, bu çalışmada önerilen planlayıcı mimarisi, planlama teknikleri ile veb servis süreçlerini tek bir etmen altyapısında birleştirmeyi öngörmektedir. Ayrıca altyapı, genişleyebilir yapısı sayesinde OWL-S dışındaki farklı süreç modellerini de destekleyebilecek gizilgüce sahiptir.

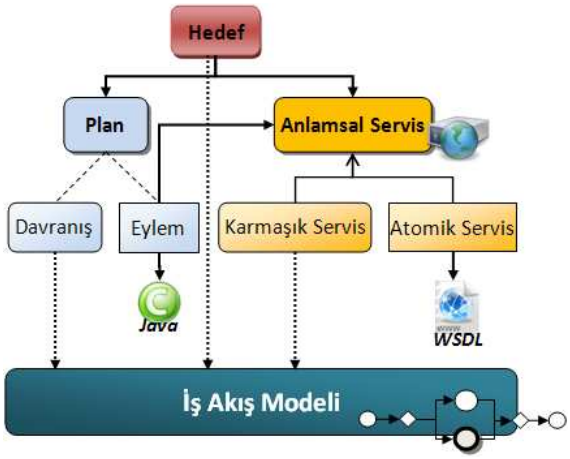
Bildirinin geri kalanı şu şekilde düzenlenmiştir: 2. bölümde önerilen eylem üst modeli ve bileşenleri ayrıntılı bir şekilde tanımlanmıştır. Bölüm 3 ve 4 sırasıyla, planlayıcı mimarisini ve OWL-S süreçlerinin bahsedilen iş-akış modeline dönüşümünü sağlayan indirgeme algoritmasını açıklamaktadır. Bölüm 5, eylem üst modelinde tanımlanan kavramlara ait ontolojilerin hızlı bir şekilde üretilmesi için hazırlanan geliştirim ortamı tanıtılmıştır. Bölüm 6'da önerilen mimarinin hedef uygunluğu bir durum çalışması ile sınanmıştır. Bölüm 7'de ise ileriye dönük araştırma olanaklarına değinilerek bildiri sonlandırılmaktadır.

2. Etmen-Anlamsal Servis Tümleşimi İçin Eylem Üst Modeli

SEAGENT çoklu etmen platformu etmenlerin rol tabanlı olarak geliştirilebildiği bir ortam sağlar. Buna göre bir etmen yaşam döngüsü boyunca bulunduğu organizasyon içinde çeşitli roller üstlenerek hedeflerini yerine getirir. Bu amaçla, verilen hedefe ulaşmak isteyen etmen eylemde bulunur ya da başka bir deyişle onu amacına götürecek davranışı sergiler. Bu davranış planlanmış bir eylem dizisi olabileceği gibi basit ya da karmaşık bir veb servisi de olabilir. Şekil 1'de SEAGENT eylem üst modeli ve bu modelde anlamsal servislerin yeri gösterilmiştir. Bölümün geri kalan kısmında modeldeki temel bileşenler tanımlanacak ve anlamsal servislerin bu modele nasıl dahil olduğu üzerinde durulucaktır.

[1] http://www.daml.org/services/swsa/

[2] http://www.daml.org/services/daml-s/0.9/



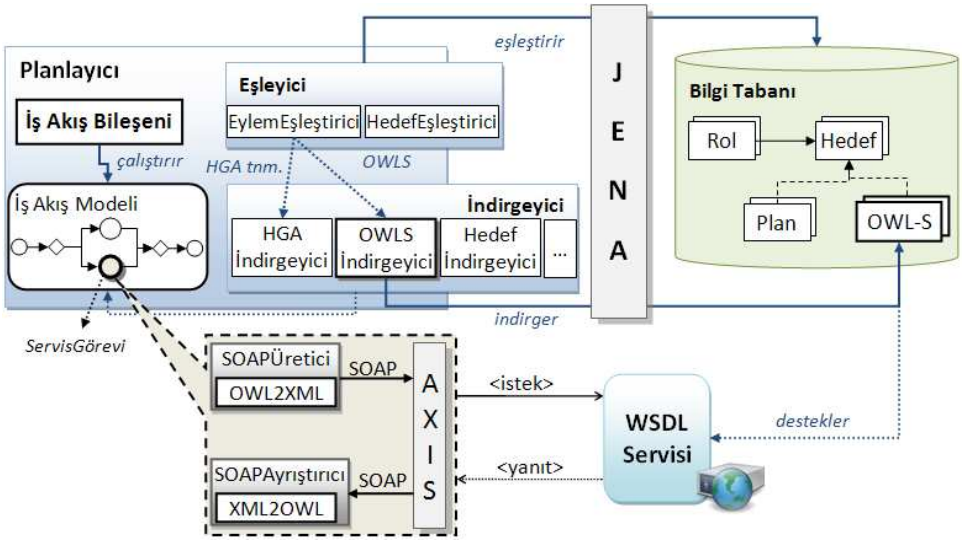
Şekil 1: SEAGENT Eylem Üst Modeli

Hedef. *Hedef* tanımı etmenin ya da etmenin üstlendiği bir rolün organizasyon içinde ulaşmak istediği genel hedefleri belirtir. Bir *hedef* tanımı eyleme geçebilmek için sağlanması gereken önkoşulları, eylem girdi ve çıktılarını ve hedefe ulaştıktan sonra oluşacak etkileri tanımlar. Etmen, *ilkel* (atomik) *hedeflere* sahip olabileceği gibi diğer alt-hedeflerden oluşan *karmaşık hedefleri* de gerçekleştirmek isteyebilir. Karmaşık hedefler doğrudan eyleme geçirilebilir hedefler değildir. Bu nedenle öncelikle kendilerini oluşturan ilkel hedeflere ayrıştırılmalıdırlar. İlkel hedefler ise, karmaşık hedeflerden farklı olarak, çalışma zamanında uygun bir eylem örneği (etmen planı ya da veb servisi) ile bağlanarak doğrudan gerçeklenebilirler. Eylem örneğinin bulunması, hedef tanımına göre etmen bilgi tabanının sorgulanması ya da uygun servis sağlayıcıların aranması şeklinde olur.

Plan. Belirli bir hedefe ulaşan eylem dizisini oluşturma işine *planlama* denir. Plan ise planlama süreci sonunda oluşan çıktıdır. Bu bağlamda SEAGENT eylem üst modelinde belirtilen *Plan* ontolojisi, etmenin verilen bir hedefe ulaşmasını sağlayacak olan, başka bir deyişle hedef tanımı ile eşleşen, eylemler dizisini ifade eder.

SEAGENT'ta etmen planları HGA (Hiyerarşik Görev Ağı) yöntemi kullanılarak tanımlanmıştır. HGA planlamasında kilit nokta karmaşık görevlerin ya da *davranışların* kendini oluşturan ilkel görevlere ya da *eylemlere* ayrıştırılmasıdır. Böylece karmaşık bir HGA görevi, etmeni hedefine ulaştıran eylem dizisine dönüştürülmüş olur. Öte yandan ilkel görevler olan HGA eylemleri ise işletim birimi tarafından doğrudan çalıştırılabilir. Bir eylem, özel olarak üretilmiş bir Java sınıfı ile gerçeklenebileceği gibi çalışma zamanında bir veb servisine de bağlanabilir. Böylece planlanmış bir eylem dizisi içinden veb servisleri çağrılarak esnek ve etkin bir yapı sağlanmış olur. Bu yapı sayesinde etmen, hedefine ulaşmak için varolan servisleri planlanmış bir düzen içinde kullanabilir.

Anlamsal Servis. SEAGENT etmenleri OWL-S (Servisler için Veb Ontoloji Dili) ontoloji grubu ile tanımlanmış anlamsal servisleri işletebilme yeteneğine sahiptir. OWL-S, servis yeteneklerinin istemcilere tanıtımını amaçlayan *profil* ontolojisi, servisin nasıl çalıştırılacağını ve servisi oluşturan süreçler arasındaki iş akışını tanımlayan *süreç modeli* ontolojisi ve servis erişim detaylarını veren *erişim zemini* ontolojisinden oluşmaktadır.



Şekil 2: *SEAGENT Planlayıcı Mimarisi*

Şekil 1'de görüldüğü gibi etmen ya da etmenin oynadığı bir rol, verilen hedefe ulaşmak için bir anlamsal servis örneğini işletebilir. Bu durumda etmen, hedefine uygun olarak organizasyondaki servis sağlayıcılarını sorgulayarak ihtiyaç duyduğu servisi bulmaya çalışır. Daha sonra eşleşen servis işletim birimi tarafından çalıştırılarak hedef gerçekleştirilir. Daha önce de bahsedildiği gibi servislerin sisteme dahil oldukları bir diğer nokta ise HGA eylemleridir. Plan işletimi sırasında etmen, atomik bir eylemi gerçekleştirmek için bir Java sınıfını kullanabileceği gibi eylem tanımı ile eşleşen bir veb servisini de çağırabilir. Bu yolla etmen, planlarını gerçekleştirmek için varolan veb servislerinden yararlanabilir.

İş Akış Modeli: Çizge tabanlı bir iş akış yapısı sağlayan bu model SEAGENT etmenleri tarafından işletilebilen tüm süreç modelleri için ortak bir altyapı sağlar. [15]'te Sadiq vd. tarafından önerilen soyut tanımlamaları baz alan iş akış modeli *görev düğümleri, düzenleyici düğümler* ve *akışlardan* oluşur.

Görev düğümleri, iş akışı tarafından yapılacak olan işin alt parçalarını yerine getirmekle yükümlüdür. Bir başka deyişle her görev düğümü içinde bulunduğu iş akışının amacına katkıda bulunur. *Atomik* ve *bileşik* olmak üzere iki tip görev düğümü tanımlanmıştır. Atomik görevler doğrudan işletilebilir birimlerdir. Bileşik görevler ise atomik görevlerden ve diğer bileşik görevlerden oluşan karmaşık yapılar olup doğrudan çalıştırılamazlar. Ancak, ilişkili görevleri bir araya getirdikleri için basit fakat etkin bir soyutlama sağlamış olurlar.

Akışlar, iş akışını oluşturan görevler arasındaki kontrol ve veri geçişini sağlar. Bu amaçla iki tip akış nesnesi tanımlanmıştır: *kontrol akışı* ve *veri akışı*. Bir kontrol akışı iki görev düğümü arasındaki basit işletim sırasını belirlerken, veri akışları görevler arasındaki veri geçişlerini sağlar.

Son olarak düzenleyici düğümler -dallanma, eş zamanlı işletim, döngü- gibi karmaşık iş akış desenlerini oluşturmak için kullanılır. Dallanma ve döngü yapılarını sağlamak için *seçim* (choice) ve *birleşim* (merge) düğümleri kullanılırken eş zamanlı işletim yapıları oluşturmak için *çatal* (fork) ve *eşzamanlama* (synchronizer) düğümleri kullanılır.

Tanımlanan bu yapısıyla *iş akış modeli* yukarıda söz edilen etmen hedefleri, HGA planları ve anlamsal servis süreçleri için ortak bir işletim zemini sağlar. Karmaşık hedefler, HGA davranışları ve bileşik OWL-S servisleri bu ortak modele dönüştürülerek bir arada ve melez bir şekilde kullanılabilirler. Dahası bu altyapı, sistemin genişlemesine ve yeni iş süreçlerinin sisteme dahil edilebilmesine olanak verir.

3. Planlayıcı Mimarisi

Şekil 2'de mimari görünümü belirtilen SEAGENT planlayıcı birimi, iş akış modelinin örneklerini işleten *iş akış bileşeni* ile üçüncü taraf bir ontoloji çözümleme

kütüphanesi olan JENA [1] 'yı kullanarak etmen bilgi tabanını sorgulayan *İndirgeyici* ve *Eşleştirici* isimli iki alt birimden oluşmaktadır. İş akış bileşeni, iş akış örneklerini işletmek için ziyaretçi (*token*) temelli bir mekanizma kullanır. Temel olarak ziyaretçi, iş akışındaki düğümleri işletim sırasına göre dolaşarak her bir görev düğümünü çağıran bir veri yapısıdır. İş akış bileşeni ayrıca, iş akış örneklerini doğrulamak için bir *doğrulama birimi* ve örneklerin çalışma zamanında genişleyebilmesini sağlamak için devingen bir *çizge bağlama birimi* sunar. Diğer taraftan, İndirgeyici birimi, HGA ve OWL-S tanımlarını ayrıştırmak ve bunları karşılık gelen iş akış örneklerine indirgemek için *HGAİndirgeyici* ve *OWLSİndirgeyici* gibi çizge indirgeme bileşenlerine sahiptir. Şekil-2'de belirtildiği gibi, İndirgeyici paketi genişleyebilir bir yapıdadır. Bu sayede yeni bir süreç modelini sisteme dahil etmek isteyen bir kullanıcı kendi indirgeyicisini yazabilir. Örneğin, BPEL servisleri bu şekilde sisteme dahil edilebilir. Son olarak, Eşleştirici paketi, etmenin amaçlarına bağlı olarak bilgi tabanından etmen eylemlerini sorgulamak ve uygun eylemi seçmekle yükümlüdür. Bu amaçla ilk olarak HedefEşleyicisi, etmenin yerine getirmek zorunda olduğu varolan görev için bilgi-tabanında uygun hedef tanımını arar. Eğer eşleşme sonucu karmaşık bir hedef bulunmuş ise bulunan hedef tanımı *HedefİndirgeyicisiI* tarafından iş akış çizgesine dönüştürülür. Hedefe ait iş akışının işletilmesi sırasında her atomik hedef için bu hedefin başarıyla tamamlanmasını sağlayacak olan eylemi (HGA planı veya OWL-S servisi) sorgulayan *EylemEşleyicisi* çalıştırılır. EylemEşleyicisi tarafından uygun eylem tanımı döndürüldükten sonra, eylemin çizgeye çevrilmesi için için bir indirgeyici (HGAİndirgeyici veya OWLSİndirgeyici) seçilir. Son olarak, seçilen ve indirgenen eylem tanımına karşılık gelen iş akışı, devingen bağlayıcı bileşeni aracılığıyla çalışma zamanında hedef iş akışına bağlanır. Böylece etmen, kendini hedefine ulaştıracak eylemler dizisini belirlemiş ve işletmiş olur.

Anlamsal servislerin çalıştırılması açısından bakıldığında mimaride iki bileşen ön plana çıkmaktadır: *OWLSİndirgeyici* ve *ServisGörevi*. Yukarıda açıklandığı gibi OWLSİndirgeyici OWL-S bileşik süreç modellerini karşılık gelen iş akış örneklerine dönüştürmekle sorumludur. Bu amaçla Algoritma-1'de gösterilen indirgeme algoritmasını kullanır.

İndirgeme işlemi sırasında OWLSİndirgeyici'si her OWL-S atomik süreci için servis çağrımını yapacak bir ServisGörevi örneği yaratır. ServisGörevi, WSDL ile tanımlanmış herhangi bir veb servisini uyandırabilen, özel olarak gerçekleştirilmiş bir iş akış düğümüdür. Şekil 2'de kesikli çizgilerle belirtilmiş kutu bir ServisGörevi'nin iç yapısını göstermektedir. Buna göre bir ServisGörevi iş akış bileşeni tarafından çalıştırılırken, ilk adımda, *SOAPÜretici* birimi OWL ile tanımlanmış servis girdilerini XML biçimine çevirerek bir SOAP istek iletisi hazırlar. Ardından bu ileti hedeflenen WSDL servisini çağırmak amacıyla, üçüncü taraf bir SOAP gerçekleştirimi olan Axis üzerinden, servis sağlayıcısına gönderilir. Bir sonraki adımda ise sağlayıcı tarafından gönderilen SOAP yanıt iletisi *SOAPAyrıştırıcı*'sı ile çözümlenir ve eğer varsa servis çıktısı XML'den OWL örneğine çevrilir. Son olarak, ServisGörevi sonlanmadan hemen önce, örnek çıktı olarak üretilir ve gerekliyse ileriki görev düğümlerine aktarılır.

4. İndirgeme Algoritması

Bu bölümde OWL-S süreç modeli ile önerilen iş akış modeli arasındaki dönüşümü yapan indirgeme algoritması üzerinde durulmaktadır. Sözü edilen algoritma Tablo 1'de gösterilmiştir.

Tablo 1: OWL-S bileşik süreçlerinin indirgenmesi

Girdi: σ kontrol yapısından oluşan bir bileşik süreç, *cp*
Çıktı: bir iş akış örneği, *wf*

1. *cp* için bir iş akış örneği, *wf*, ilklendir.
2. σ'yı tipine göre bir iş akışına, wf_σ, indirge.
3. wf_σ 'yı, *wf*'ye alt düğüm olarak ekle ve *wf*'nin uç düğümleri arasına (i_{wf}, t_{wf}) kontrol akışları bağla.
4. Girdi-çıktı bağlamalarına (Binding) karşılık gelen veri akışlarını oluştur.

Görüldüğü gibi algoritma çevrimi yapabilmek için dört adımdan oluşan bir yordam izler. Öncelikle yalnızca uç düğümler ile girdi-çıktılardan oluşan ve *cp*'ye karşılık gelen boş bir iş akışı oluşturulur. Daha sonra bileşik süreci oluşturan kontrol yapısı σ Tablo 2'de gösterilen algoritma tarafından indirgenerek başka bir iş akışı (wf_σ) oluşturulur. Sonraki adımda wf_σ, *cp* için oluşturulan iş akışının uç düğümlerine bağlanarak çocuk düğüm olarak bu akışa eklenir. Son olarak da süreç modeli içinde tanımlanmış olan girdi ve çıktı bağlamaları alt görevler arasındaki veri akışlarına çevrilir.

Adım 2'de bileşik süreç ile ilişkili olan kontrol yapısı, tipine uygun olarak karşılık gelen iş akış örneğine indirgenmektedir. Bir başka deyişle, her kontrol yapısını dönüştürmek için özel bir indirgeme yöntemi uygulanır. Tablo 2'de tüm kontrol yapılarının dönüşümü için genel bir algoritma verilmiştir. Algoritmanın ilk adımında kontrol yapısına karşılık gelen boş bir iş akış nesnesi yaratılır. Daha sonra kontrol yapısının her bir bileşeni yinelemeli olarak indirgenir(2a) ve oluşturulan görev düğümü iş akışına eklenir(2b). Son olarak, adım 2c'de, yaratılan görev düğümü tipine göre iş akışı içerisinde uygun yere yerleştirilir.

[1] http://jena.sourceforge.net/

Sequence kontrol yapısı için, algoritma her görev düğümü τ_i'yi adım 2c'de kendinden bir sonra gelen düğüme bağlar. Örneğin τ_k görevi, τ_{k-1} ile τ_{k+1} görevleri arasına yerleştirilir. Sonuç olarak, başlangıç düğümünden bitiş düğümüne kadar bir biri ardına sıralanmış görevlerden oluşan bir iş-akışı üretilir.

Sequence'in tersine *Choice* yapısının indirgenmesi ek adımlara ihtiyaç duyar. İkinci adımdan önce, bir seçim düğümü (*x*) ve bir birleştirme düğümü (*mr*) yaratılır ve bunlar *wf*'nin çocuklarını tutan listeye eklenir. Bu durumda algoritma, her bileşenin indirgenmesinden elde edilen görev düğümlerini, kontrol akışları kullanarak, *x* ve *mr* düğümleri arasına bağlar. OWL-S Choice yapısı gelişigüzel seçim yaptığı için *x* seçim düğümünün belirsiz bir şekilde seçim yapması sağlanır.

IfThenElse nesnesi için indirgeyici algoritma Choice kontrol yapısında olduğu gibi görev düğümlerini seçim ve birleştirme düğümlerinin arasına yerleştirir. Burada yerleştirilen görev düğümleri, IfThenElse yapısının *then* ve *else* bileşenlerine karşılık gelmektedir. Choice dönüşümünden farklı olarak, seçim düğümü bir sonraki

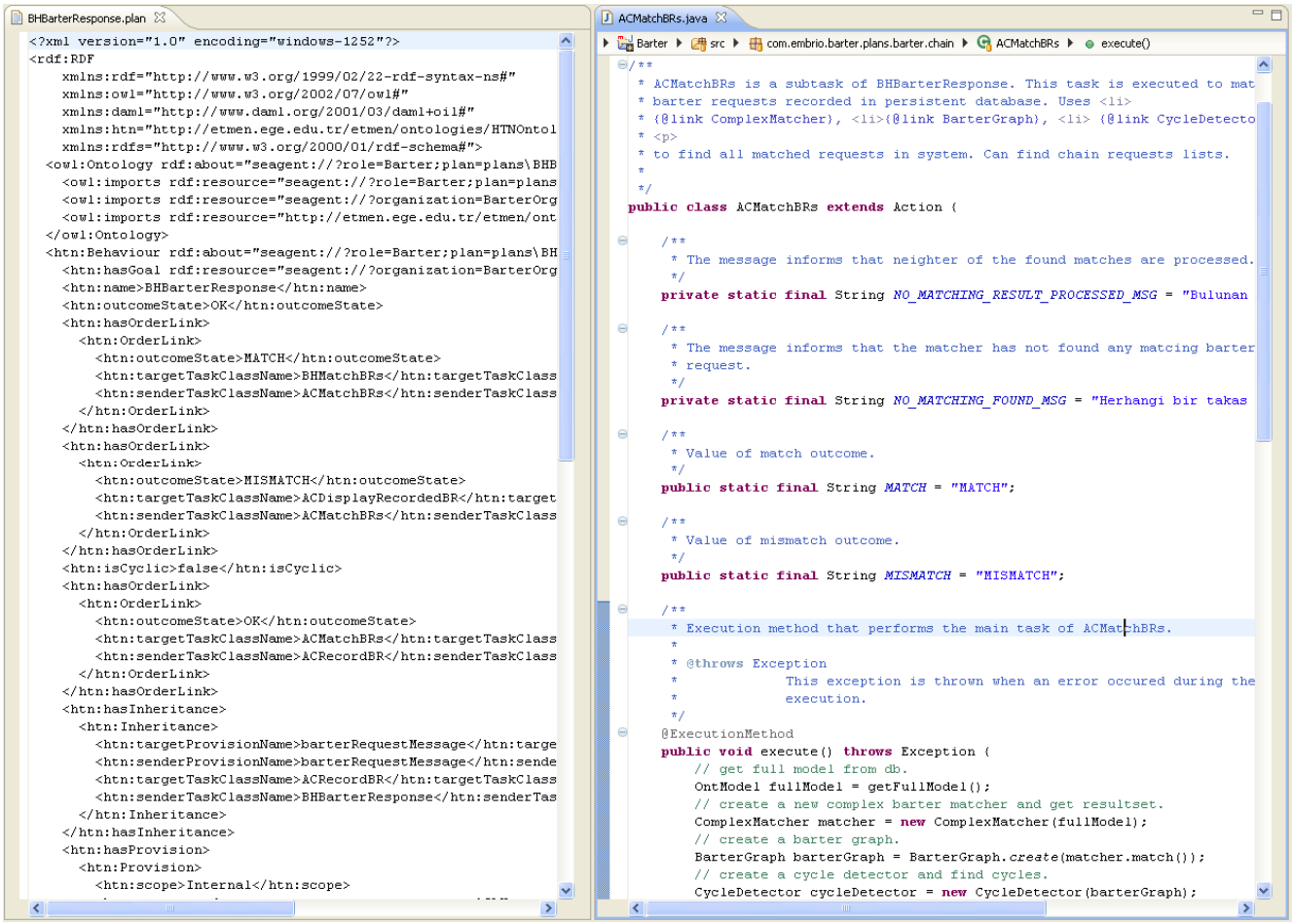
Tablo 2: Bir OWL-S kontrol yapısının indirgenmesi

Girdi: bir OWL-S kontrol yapısı σ
Çıktı: bir iş akış örneği, *wf*

1. σ için boş bir iş akış örneği, *wf*, ilklendir.
2. σ'nın her bir σ_i bileşeni için,
   a. Adım 1'e giderek σ_i'yi τ_i görev düğümüne indirge.
   b. τ_i'yi *wf*'nin çocuk listesine ekle.
   c. τ_i'yi σ'nın tipine göre *wf* içinde uygun bir yere yerleştir.

görevi, koşul değerini göz önünde bulundurarak seçer. Bu seçim mekanizmasını kurmak için *then* ve *else* düğümlerine karşılık gelen mantıksal değerler (*doğru*, *yanlış*) sonraki görev ile eşleştirilir. Böylece eğer durum "*doğru*" olarak belirlenmiş ise seçim yapısı kontrolü *then* görevine geçirmekte, eğer '*yanlış*' olarak belirlenmiş ise *else* görevine geçirmektedir.

SplitJoin yapısının indirgenmesi için Adım 2'den önce algoritma bir çatal düğümü *fr* ve bir eşzamanlayıcı düğümü *sy* yaratır. Daha sonra adım 2c'de SplitJoin'in her bileşeninin indirgenmesiyle elde edilen görev düğümleri, çatal ve eşzamanlayıcı düğümlerinin arasına



Şekil 3: HGA Editör tarafından üretilen örnek bir plan ontolojisi (solda) ve eylem kodu (sağda).

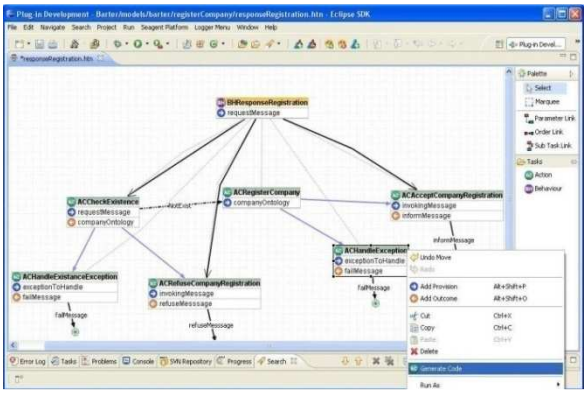
bağlanır.

RepeatWhile nesnesinin indirgenmesi sırasında, öncelikle algoritmanın 2a adımında, kontrol yapısına ait *while* bileşeni karşılık gelen iş akış görevi τ_w'ye indirgenir. Bundan sonraki adımda τ_w, *birleştirme* ve *seçim* düğümlerinin arasına yerleştirilir. Ayrıca döngüyü sağlamak için birleştirme düğümünden seçim düğümüne doğrudan bir bağlantı kurulur. Koşul-düğüm eşlemesine bağlı olarak seçim düğümü, döngü koşul değeri *'doğru'* olduğu sürece iş akışını birleştirme düğümüne geçirir, aksi halde ise bitiş düğümünü etkinleştirir ve böylece döngüyü sonlandırır.

5. Geliştirim Ortamı

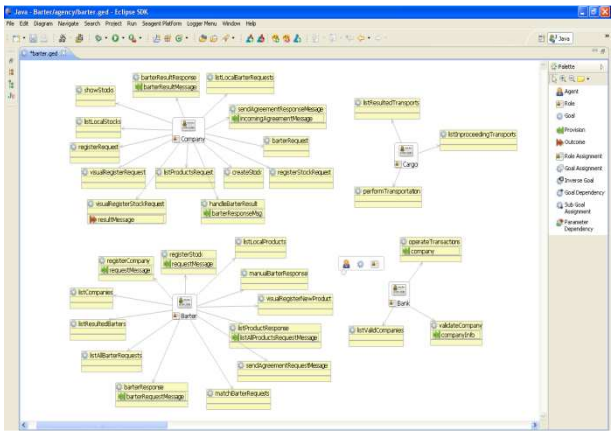
Hedef, plan gibi SEAGENT bilgi-tabanı öğeleri ve OWL-S servisleri OWL dili kullanılarak tanımlandığı için bunların el ile oluşturulması geliştirim aşamasını oldukça zahmetli bir hale getirmektedir. Bu nedenle SEAGENT geliştirim ortamı hedef, HGA planı ve OWL-S servislerinin yaratılması için özelleşmiş görsel editörler sunmaktadır. Dahası bu editörler Eclipse geliştirim ortamında eklenti olarak kullanılabildiği için kullanıcılara esnek ve tümleşik bir ortam sağlanmaktadır. Bu bölümde SEAGENT geliştirim ortamının sunduğu görsel editörler tanıtılacaktır.

- **HGA Editörü:** Etmen planlarının HGA yöntemi kullanılarak hızlıca üretilmesini sağlar. Kullanıcı, editör paleti üzerinden HGA davranış ve eylemlerini sürükleyerek çeşitli planlar hazırlayabilir. HGA eylemlerinin gerçekleştirimini sağlayan Java sınıfları ve hazırlanan plana ait örnek ontolojisi editör tarafından otomatik olarak üretilmektedir. Şekil 4'teki ekran görüntüsünde HGA editörüyle modellenmiş bir etmen planı gösterilmiştir. Şekil 3'te ise bu plan için üretilen OWL ontolojisi ve Şekil 4'te vurgulanan eylem için üretilen Java kodu gösterilmiştir.



Şekil 4: HGA Editörü'nden bir görünüm.

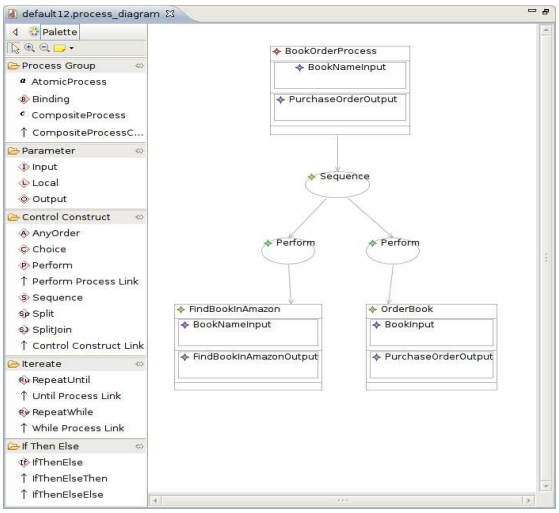
- **Hedef Editörü:** Etmenlerin organizasyon içinde üstlendikleri rollerin ve bu rollere ait hedeflerin tasarlandığı editördür. Bu editör sayesinde kullanıcı, karmaşık hedefleri modelleyebildiği gibi bu hedeflerin uygun rollere atamasını da yapabilir. Editör, çıktı olarak tasarlanan hedef modelinin OWL diliyle ifade edilmiş ontolojisini üretmektedir. Şekil 5'teki ekran görüntüsü Hedef Editörü ile tasarlanan 4 rolü ve bunlara ait hedefleri göstermektedir.



Şekil 5: Hedef Editörü'nden bir görünüm.

- **OWL-S Editörü:** OWL-S işaretleme dili ile tanımlanmış anlamsal servis örneklerinin hızlı bir şekilde üretilmesini sağlayan görsel bir eklentidir. OWL-S Eklentisi iki alt birimden oluşmaktadır. İlk birim, servise ait süreç modelinin tasarlanmasını sağlayan ve Şekil 6'da bir örneği görülen süreç modeli editörüdür. Editör EMF [1] kullanılarak geliştirildiği için bir Ecore örneği olarak tasarlanan servis süreci kolaylıkla OWL-S Süreç Modeli örneğine çevrilebilir. OWL-S eklentisine ait ikinci birim ise, OWL-S erişim zemini ontolojisi ile WSDL arasındaki eşleşmeleri oluşturan görsel bir eşleştirme sihirbazıdır. Bu bileşen sayesinde kullanıcı, WSDL işlemlerini (operation), OWL-S atomik servisleriyle eşleyerek zemin ontolojisi elde edebilir.

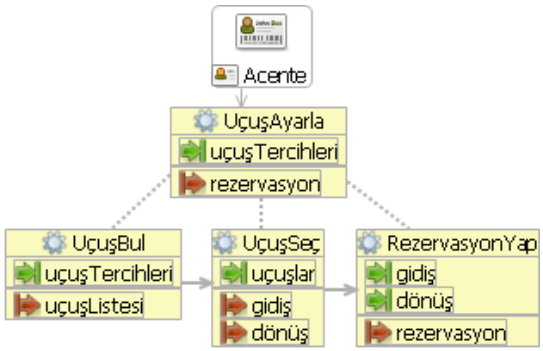
[1] http://www.eclipse.org/modeling/emf/



Şekil 6: OWL-S Süreç Modeli Editörü

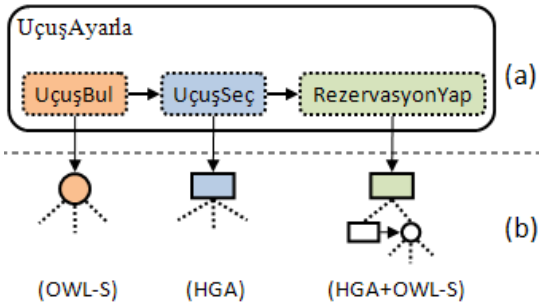
6. Durum Çalışması

Bu bölümde bir uçuş rezervasyon uygulaması durum çalışması olarak gerçekleştirilmiştir. Bu uygulamada basit bir uçuş rezervasyon senaryosu, etmene ait *Acente* rolü tarafından kotarılmaktadır. Şekil 7'de gösterilen hedef tanımına göre Acente rolünü üstlenen etmen istenilen uçuşu ayarlamak için üç aşamalı bir yöntem izlemektedir. İlk aşamada, etmen kendisine verilen uçuş tercihlerini kullanarak uygun uçuşları listelemekte, daha sonra bu uçuşlar arasından uygun gidiş-dönüş (ya da sadece gidiş) uçuş(lar)ı seçmekte ve son aşamada rezervasyon isteğini tamamlamaktadır. Sonuç olarak, etmene rezervasyon bilgisi dönmektedir.



Şekil 7: Uçuş Rezervasyon Hedefi

UçuşAyarla karmaşık hedefini gerçekleştirmek isteyen etmen, hedef tanımını planlayıcıya verir. Planlayıcı, bu hedefin karmaşık bir hedef olduğunu saptayınca HedefİndirgeyiciÕyi (bkz. Şekil 2) kullanarak *UçuşAyarla* hedefini iş-akışına indirger ve sonuçta Şekil 8aÕdaki çizge oluşur. Görüldüğü gibi, hedef modelindeki her atomik hedef, iş-akışında, bu atomik hedefi eşleşen HGA planına ya da OWL-S servisine bağlayabilecek özel bir görev düğümü ile temsil edilmiştir.



Şekil 8 UçuşAyarla hedefine ilişkin iş-akış çizgesi

UçuşAyarla hedefine ait iş-akış çizgesi oluşturulduktan sonra, planlayıcı, bu çizge üzerinde ilerlemeye başlar. İşletim sırasına göre ilk olarak *UçuşBul* hedefi ele alınır. *EylemEşleyici*, etmen bilgi-tabanında hedef için uygun eylem tanımını sorgular ve bir OWL-S servisi bulur. Bulunan servis tanımı *OWLSİndirgeyici* tarafından çizgeye dönüştürülür ve elde edilen çizge hedefe ait görev düğümüne bağlanır. Daha sonra planlayıcı, işletimi bu çizgeye geçirir ve böylece servis işletimi gerçekleşir. Servis çağrımı tamamlandığında elde edilen servis çıktısı, yani uygun uçuş listesi, *UçuşSeç* hedef düğümüne aktarılır. Bu noktada *EylemEşleştirici* tekrar devreye girerek *UçuşSeç* hedefini gerçekleyecek HGA planını bulur. Bu plan, *HGAİndirgeyici* tarafından çizgeye dönüştürülerek *UçuşSeç* düğümüne bağlanır ve HGA çizgesi planlayıcı tarafından işletilir. Planın çıktısı olan gidiş ve dönüş uçuşları son görev düğümü olan *RezervasyonYap* düğümüne aktarılır ve benzer şekilde bu hedef ile eşleşen HGA planı bulunup çalıştırılır. Bu planın ilk eyleminde kullanıcıya rezervasyonu onaylayıp onaylamadığı sorulur. Daha sonraki eylemde ise, kullanıcı onayı alındığı durumda bir veb servisine bağlanılarak rezervasyon tamamlanır. Aksi takdirde rezervasyon isteği iptal edilir.

7. Sonuç

Bu çalışmada etmen-anlamsal servis tümleşimini sağlamaya yönelik bir eylem üst modeli tanıtılmış ve bu modeldeki kavramları somut nesneler olarak ele alıp işleten bir planlayıcı mimarisi önerilmiştir. Önerilen altyapı OWL-S, BPEL gibi farklı iş süreçlerini destekleyebilecek esnekliktedir. Farklı işletim semantiklerini destekleyebilen böyle bir etmen altyapısı, günümüz etmen geliştirme çerçevelerinin endüstrileşebilmeleri için sahip olmaları gereken bir özelliktir. Bu şekilde veb servisleri, iş-akış süreçleri gibi endüstride yoğun olarak kullanılan teknolojiler etmen sistemlerine dahil edilebilir. Söz edilen temel nitelikleriyle bu çalışma, otomatik servis keşif ve birleştirim çalışmaları için bir alternatif olarak değil, tamamlayıcı bir çalışma olarak görülmelidir.

8. Kaynakça

[1] I. Horrocks and J. A. Hendler, Eds., The Semantic Web - ISWC 2002, First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002, Proceedings, ser. Lecture Notes in Computer Science, vol. 2342. Springer, 2002.

[2] Dumitru Roman and Uwe Keller and Holger Lausen and Jos de Bruijn and Rubén Lara and Michael Stollberg and Axel Polleres and Cristina Feier and Christoph Bussler and Dieter Fensel, "Web Service Modeling Ontology," Applied Ontology, vol. 1, no. 1, pp. 77–106, 2005.

[3] Russell, S. and Norvig, P., Artificial Intelligence: A Modern Approach. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 2003.

[4] Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Le mann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, Sanjiva Weerawarana, "Business Process Execution Language for Web Services V-1.1," W3C, Candidate Recommendation, 2003.

[5] K. Erol, J. Hendler, and D. S. Nau, "Semantics for Hierarchical Task Network Planning," College Park, MD, USA, Tech. Rep., 1994.

[6] R. E. Fikes and N. J. Nilsson, "Strips: A New Approach to the Application of Theorem Proving to Problem Solving," Artificial Intelligence, vol. 2, no. 3-4, pp. 189–208, 1971. [Online]. Available: http://dx.doi.org/10.1016/0004-3702(71)90010-5

[7] E. E. Ekinci, A. M. Tiryaki, Ö. Gürcan, and O. Dikenelli, "A Planner Infrastructure for Semantic Web Enabled Agents," in OTM Workshops (1), 2007, pp. 95–104.

[8] Özgür Gümüş, Önder Gürcan, Geylani Kardaş, Erdem Eser Ekinci, Oğuz Dikenelli. Engineering an MAS Platform for Semantic Service Integration based on SWSA, 3rd International Workshop on Agents and Web Services in Distributed Environments (AWeSOMe'07), Algarve, Portugal, November 25-30, 2007.

[9] Önder Gürcan, Geylani Kardaş, Özgür Gümüş, Oğuz Dikenelli, İbrahim Çakırlar, Övünç Çetin, A. Burak Eliaçık, Hüseyin Kır. Etmen Tabanlı bir Anlamsal Servis Platformu. 3. Ulusal Yazılım Mühendisliği Sempozyumu ve Sergisi - UYMS' 2007), Ankara, Turkey, September 27-30, 2007.

[10] N. Gibbins, S. Harris, N. Shadbolt, Agent-based Semantic Web Services, WWW 2003, Budapest, Hungary, 20–24 May 2003.

[11] E. Sirin, B. Parsia, D. Wu, J. A. Hendler, and D. S. Nau, "Htn Planning for Web Service Composition Using SHOP2," J. Web Sem., vol. 1, no. 4, pp. 377–396, 2004.

[12] D. N. Nau, "Shop2: An HTN Planning System. Journal of Artificial Intelligence Research 20 (2003) 379-404.".

[13] M. Klusch and A. Gerber, "Evaluation of Service Composition Planning with OWLS-Xplan." in IAT Workshops, 2006, pp. 117–120.

[14] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL-The Planning Domain Definition Language," 1998. [Online]. Available: citeseer.ist.psu.edu/article/ghallab98pddl.html

[15] W. Sadiq and M. Orlowska, "Modeling and Verification of Workflow Graphs," in Technical Report No. 386, Department of Computer Science. The University of Queensland, Australia, 1996.