EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(YÜKSEK LİSANS TEZİ)

SERVİS ile ETMEN ENTEGRASYONU için BİR PLANLAMA MODÜLÜ TASARIMI ve GERÇEKLEŞTİRİMİ

Önder GÜRCAN

Bilgisayar Mühendisliği Anabilim Dalı

Bilim Dalı Kodu: 619.01.00

Sunuş Tarihi: 23.07.2007

Tez Danışmanı: Prof. Dr. Oğuz DİKENELLİ

Bornova-İZMİR

Önder GÜRCAN tarafından YÜKSEK LİSANS tezi olarak sunulan "**SERVİS ile ETMEN ENTEGRASYONU için BİR PLANLAMA MODÜLÜ TASARIMI ve GERÇEKLEŞTİRİMİ**" başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve **23/07/2007** tarihinde yapılan tez savunma sınavında aday oy birliği/oy çokluğu ile başarılı bulunmuştur.

Jüri Üyeleri:	**İmza**
Jüri Başkanı: Prof. Dr. Oğuz DİKENELLİ	.....
Raportör Üye: Yrd. Doç. Dr. R. Cenk ERDUR	.....
Üye: Yrd. Doç. Dr. Tuğkan TUĞLULAR	.....

ÖZET

SERVİS ile ETMEN ENTEGRASYONU için BİR PLANLAMA MODÜLÜ TASARIMI ve GERÇEKLEŞTİRİMİ

GÜRCAN, Önder

YÜKSEK LİSANS Tezi, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Prof. Dr. Oğuz DİKENELLİ

23/07/2007, 107 sayfa

Daha esnek ve devingen iş süreci tümleşimine ve otomasyonuna olan küresel ilgi servise-yönelik hesaplama, anlamsal teknoloji ve zeki çok-etmenli sistemler arasında bir yakınlaşmaya sebep olmuştur. Servise-yönelik hesaplama ve anlamsal teknolojinin yakınlaşması anlamsal veb servisleri kavramını ortaya çıkarmıştır. Dağıtık İnternet ortamında çalışan özerk etmenler anlamsal veb servislerini anlamsal tanımları sayesinde otomatik olarak keşfedebilen, yönetebilen ve çalıştırabilen dağıtık uygulamalardır. Ancak anlamsal veb servislerini kullanması kaçınılmaz olan etmenlerin servislerle arasında bazı birlikte işlerlik sorunları bulunmaktadır. Bu tezde, bu birlikte işlerlik sorunlarının çözümüne yönelik bir mimari ortaya konmuş ve bu mimarideki etmenlerin iç yapıları (planlama yapıları) tasarlanmıştır.

Anahtar Sözcükler: Anlamsal servisler, etmenler, çok-etmenli sistemler, planlama.

ABSTRACT

The DESIGN and IMPLEMENTATION of a PLANNING MODULE for SERVICE-AGENT INTEGRATION

GÜRCAN, Önder

MSc. in Computer Engineering
Supervisor: Prof. Dr. Oğuz DİKENELLİ
23/07/2007, 107 pages

The global trend towards more flexible and dynamic business process integration and automation has led to a convergence of interests between service-oriented computing, semantic technology, and intelligent multi-agent systems. The convergence of service-oriented computing and semantic technology deduced the concept of semantic web services. Autonomous agents that reside on the distributed Internet environment are programs which can automatically discover, orchestrate and invoke semantic web services using their semantic descriptions. But there exist some interoperability problems between agents, whose making use of semantic web services is inevitable, and services. In this thesis, an architure is introduced and the internals (planning structures) of the agents within this proposed architecture is designed as a solution for these interoperability problems.

Keywords: Semantic services, agents, multi-agent systems, planning.

TEŞEKKÜR

Herşeyden önce bana araştırmayı sevdiren ve ARGE'ye teşvik eden hocalarım Dr. Oğuz Dikenelli'ye ve Dr. Rıza Cenk Erdur'a teşekkür ederim. Bu tezin ortaya çıkmasının fikirsel temellerini atan, her aşamasında yönlendiren danışmanım Dr. Oğuz Dikenelli'ye ayrıca tekrar teşekkür ederim.

Vakit ayırıp tezin taslağını okuyan, ona son halini vermemde yardımcı olan Özgür Gümüş, Geylani Kardaş'a, geliştirim aşamasında birlikte çalıştığımız proje arkadaşlarım Erdem Eser Ekinci, Ali Murat Tiryaki, İbrahim Çakırlar, Övünç Çetin, A. Burak Eliaçık ve Hüseyin Kır'a, tez şablonu olusturmam için bana kendi fakültesinin lateks tez şablonunu sağlayan Fatih Tekbacak'a, lateks konusunda yardımcı olan Tuğba Özacar ve Övünç Öztürk'e teşekkür ederim.

Beni, ağabeyimi ve kuzenimi küçük yaşta bilgisayar ile tanıştıran ve eğitimim konusunda bana her türlü desteği veren annem K. Ümit Özbek'e, eğitimimin her aşamasında sürekli destek olan rahmetli anneannem H. Ayten Özbek'e ve rahmetli dedem M. İhsan Özbek'e, yıllarca bıkmadan bilgisayar konusundaki her sorumu cevaplayan ve beni sürekli destekleyen ağabeyim Fethi Gürcan'a teşekkürü bir borç bilirim. Siz olmadan bu satırları yazıyor olamazdım. Ayrıca ailemin diğer tüm fertlerine sağladıkları destekten dolayı teşekkür ederim.

Son olarak, bilimin bir adım ilerlemesi için haftalarını ve gecelerini adayan bütün araştırmacılara minnetlerimi sunarım.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ă

ÇİZELGELER DİZİNİ

XVIII

KISALTMALAR

Kısaltma	Açılım
A-VSTD	Anlamsal açıklamalı Veb Servis Tanımlama Dili (*Semantic annotation for WSDL - S-WSDL*)
AVKD	Anlamsal Veb Kural Dili (*Semantic Web Rule Language - SWRL*)
AVSM	Anlamsal Veb Servisler Mimarisi (*Semantic Web Services Architecture*)
BMD	Birleştirilmiş Modelleme Dili (*Unified Modelling Language - UML*)
BNEP	Basit Nesne Erişim Protokolü (*Simple Object Access Protocol - SOAP*)
BT	Bilgi Teknolojileri (*Information Technologies - IT*)
ÇES	Çok-Etmenli Sistem (*Multi-Agent System - MAS*)
DGV	Dünya Geneli Veb (*World Wide Web - WWW*)
DGVK	Dünya Geneli Veb Konsorsiyumu (*World Wide Web Consortium - W3C*)
EEME	Elektrik, Elektronik Mühendisleri Enstitüsü (*Institute of Electrical and Electronics Engineers - IEEE*)
EİK	Etmen İletişim Kanalı (*Agent Communication Channel - ACC*)
GBD	Genişletilebilir Biçimleme Dili (*eXtensible Markup Language - XML*)

Kısaltma	**Açılım**
GÇÖE	Girdi/Çıktı/Önkoşul/Etki (*Input/Output/Precondition/Effect - IOPE*)
HGA	Hiyerarşik Görev Ağı (*Hierarchical Task Network - HTN*)
İİN	İnanç İstek Niyet (*Belief Desire Intetion - BDI*)
KTÇ	Kaynak Tanımlama Çerçevesi (*Resource Description Framework - RDF*)
MTE	Masaçusets Teknoloji Enstitüsü (*Massachusetts Institution of Technology - MIT*)
OE	Ontoloji Etmeni (*Ontology Agent*)
PATD	Planlama Alanı Tanım Dili (*Planning Domain Definition Language - PDDL*)
S-VOD	Servisler için Veb Ontoloji Dili (*OWL for Services - OWL-S*)
SİE	Servis İstemci Etmen (*Service Requester Agent*)
SKE	Servis Kayıtçı Etmen (*Service Registry Agent*)
SSE	Servis Sağlayıcı Etmen (*Servis Provider Agent*)
ÜMAP	Üstün-Metin Aktarım Protokolü (*Hyper-Text Transfer Protocol - HTTP*)
VOD	Veb Ontoloji Dili (*Web Ontology Language - OWL*)
VSMÇ	Veb Servis Modelleme Çerçevesi (*Web Services Modeling Framework - WSMF*)
VSMO	Veb Servis Modelleme Ontolojisi (*Web Services Modeling Ontology - WSMO*)
VSTD	Veb Servis Tanımlama Dili (*Web Services Description Language - WSDL*)

Kısaltma	**Açılım**
YZ	Yapay Zeka (*Artificial Intelligence - AI*)
ZFEK	Zeki Fiziksel Etmenler Kuruluşu (*Foundation for Intelligent Physical Agents - FIPA*)
ZFEK-EİD	ZFEK Etmen İletişim Dili (*FIPA Agent Communication Language - FIPA-ACL*)

XXII

1 GİRİŞ

1.1 Amaç

Daha esnek ve devingen iş süreci tümleşimine ve otomasyonuna olan küresel ilgi servise-yönelik hesaplama, anlamsal teknoloji ve zeki çok-etmenli sistemler arasında bir yakınlaşmaya sebep oldu. Özellikle servise-yönelik hesaplama ve anlamsal teknoloji, sistem mimarilerinin ve süreçlerinin benzerliği, güçlü araçları ve servis kalitesi, güvenlik, güvenilirlik gibi konulara odaklanmaları sebebiyle çok-etmenli sistemler araştırmacıları tarafından büyük ilgi görmüştür. Benzer şekilde çok-etmenli sistemler ve anlamsal teknoloji için geliştirilen teknikler hızla büyüyen servise-yönelik hesaplama teknolojisi için güçlü bir etki vaat etmektedir.

Servise-yönelik hesaplama, dağıtık hesaplama ve e-iş işleme için oturmuş bir paradigma olarak ortaya çıkmıştır. Bu paradigma servisleri, organizasyon içi ve organizasyonlar arası işbirliği yapan iş uygulamalarının çevik ağlarının geliştirimini olanaklı kılmak için temel yapıtaşları olarak kullanmaktadır. Servisler, tanımlanabilen, yayınlanabilen, keşfedilen, yönetilebilen ve İnternet gibi büyük heterojen ağlar üzerinde dağıtık uygulamalar geliştirmek amacı ile dağıtılabilen kendi kendini barındıran, platform-bağımsız yazılım bileşenleridir.

Çok-etmenli sistemler de, daha farklı ama bütünleyici bir perspektifte, dağıtık uygulamalar geliştirmeyi amaçlamaktadır. Servise-yönelik paradigmalar daha çok birlikte işler ve güvenilir altyapıların yaratılması amacıyla yazılım bileşenlerinin, arayüzlerinin, iletişim kanallarının ve ye-

teneklerinin sözdizimsel ve açıklayıcı tanımları üzerine odaklanmıştır. Çok-etmenli sistemler, devingen ve belirsiz ortamlar için esnek ve hata-toleranslı dağıtık sistemleri yaratmak için etkileşim, işbirliği veya pazarlık gibi davranışsal kavramları uygulayan özerk sorun çözücülerin çıkarsama ve planlama yeteneklerinin geliştirimine odaklanmaktadır.

Anlamsal teknoloji etmenler arası otomatik koordinasyonu mümkün kılan makinenin anlayabileceği servis tanımlarını sağlayarak etmenler ve servisler arasındaki etkileşimler için anlamsal bir altyapı ortaya koymaktadır.

Servise-yönelik hesaplama ve anlamsal teknolojinin yakınlaşması servis-tabanlı anlamsal veb kavramını yani anlamsal veb servisleri kavramını ortaya çıkarmıştır. Anlamsal veb servisleri anlamsal tanımları sayesinde otomatik olarak keşfedilebilen, otomatik olarak yönetilebilen ve otomatik olarak çalıştırılabilen dağıtık uygulamalardır. Anlamsal veb servislerini otomatik olarak kullanacak olan yapılar ise dağıtık İnternet ortamında çalışan özerk etmenlerdir. Etmenlerin oluşturduğu çok-etmenli sistemler farklı yetenekteki birçok etmenin bir araya geldiği dağıtık sistemlerdir. Ancak anlamsal veb servislerini kullanması kaçınılmaz olan etmenlerin önünde çalışma şekillerinin farklılığı, kullandıkları iletişim ve tanım standartlarının farklılığı, tasarlanış amaçlarının farklılığı yüzünden birlikte işlerlik sorunları bulunmaktadır. Bunun yanı sıra bu birlikte işlerlik sorunlarını ele alabilecek yetenekteki etmenlerin nasıl bir planlama yaklaşımı kullanması gerektiği, ne tür planlara sahip olması gerektiği, birlikte işlerliği sağlamak için ne gibi çevrimler yapması gerektiği gibi altyapı problemleri de bulunmaktadır.

Bu bölüm şu araştırma sorularıyla bitirilmektedir: (1) servis-tabanlı anlamsal vebde anlamsal veb teknolojileri etmen sistemleri tarafından nasıl kullanılacaktır; (2) etmenlerin anlamsal servisleri kullanabilmeleri için ne gibi içsel ihtiyaçlara (altyapıya) gereksinimleri vardır.

1.2 Yaklaşım ve Önerilen Çözüm

Sayısı son yıllarda dramatik bir artış gösteren veb servislerinin anlamsal servisler haline gelmesini çok-etmenli sistemler sağlayacak böylelikle servisler dinamik olarak bulunup çalıştırılabilecektir. Dağıtık İnternet ortamında çalışan, birbirleriyle haberleşen, birbirlerinin servislerini kullanabilen yazılım bileşenleri olan etmenler servisleri üyesi oldukları plarformun etki alanına göre uyarlayıp kendi yetenekleri yani kendi sundukları servisler olarak ilan edecek böylelikle diğer etmenler bu servisleri özerk bir şekilde kullanabileceklerdir.

Yukarıdaki iddiaları desteklemek için, etmen tabanlı bir anlamsal servis mimarisi olması gerektiği ve bu mimarideki etmenlerin anlamsal servisleri ele alabilmeleri için özel planlara ve planlama yapılarına sahip olmaları gerektiği önerilmektedir. Tezde, bu iddiaları desteklemek ve yazılım etmenleri ve veb servislerinin nasıl uyumlu çalışabileceklerini gösterebilmek için soyut olarak tanımlanmış olan Anlamsal Veb Servisleri Mimarisi'ne (AVSM) (Burstein et al., 2005) dayalı bir anlamsal servis mimarisi, elemanları ve süreçleriyle birlikte ortaya konmuş, sonra bölümümüzde geliştirilmekte olan SEAGENT[1] çok-etmenli çerçevesi altyapısında anlamsal

[1]SEAGENT Projesi Veb Sayfası, http://seagent.ege.edu.tr veya http://etmen.ege.edu.tr, son erişim 25 Haziran 2007.

servis çalıştırma süreci planları ve bu planlardaki ihtiyaçları karşılayan bir planlayıcı geliştirilmiştir. SEAGENT çerçevesi, standart etmen diline, protokollerine ve altyapı bileşenlerine dayalı (ZFEK), ontolojilerin, bilginin ve servislerin gösteriminde anlamsal standartları kullanan (VOD, S-VOD) bir çok-etmenli sistem platformudur.

1.3 Katkılar

Bu çalışmanın iki önemli katkısı olduğu düşünülmektedir. İlk olarak bu tez, AVSM'nin kavramsal modelinin temel ihtiyaçlarının bütün süreçlerini kapsayacak şekilde yerine getiren bir yazılım platformunu tanımlamaktadır. Tanımlanan platformda veb servislerinin tüketicileri olan etmenler açık bir ortamda çalışmaktadırlar. Platformdaki etmenler, iletişimde ve etkileşimde ontoloji tabanlı içerik kullanmaktadırlar. Etmenler, veb servislerini veb servis tanımlarını kullanarak çağırabilmektedir. Platformda hem veb servisleri hem de etmen servisleri anlamsal olarak zengin bir dilde tanımlanmaktadır. Platformda servislerle ilgili etmenler olan kullanıcı etmenleri (istemci etmenler) ve servis sağlayıcı etmenler veb servislerini otomatize etmek ve insanlara daha iyi hizmet etmek için birlikte çalışırlar.

İkincil olarak, bu tezde bu mimarinin süreçlerine yönelik planlama ihtiyaçları doğrultusunda bir etmen planlayıcısı ve her süreç için etmenlerin kullanması gereken plan yapıları tanımlanmaktadır. Bu sayede sözkonusu mimariye dayalı platformlar ileriki geliştirimler ve alan bağımlı uygulamalar için genişletilebilir ve esnek olarak tasarlanabilmektedir. Bu tezde sunulan mimari, kullanıcıların etmenlerine yetenekleri veb servisi olarak eklemelerine ve bu etmenlerin diğer etmenlerle birlikte çalışmasına olanak

sağlamaktadır. Etmenler bu yetenekleri kendi yetenekleri olarak platforma ilan etmektedirler. Bu yeteneklere ihtiyaç duyan etmenler de onları dinamik olarak bulup (etmen servisi bulur gibi) ilgili yeteneğe sahip etmenle etkileşerek kullanmaktadırlar. Burada gerek yeteneği (servisi) sağlayan etmen tarafında, gerekse de yeteneğe ihtiyaç duyan (istemci) etmen tarafında AVSM süreçlerine dayalı öntanımlı etmen planları rol almaktadır.

1.4 Genel Çerçeve

Tezin geri kalanı şu şekilde düzenlenmiştir. İlk önce Bölüm 2'de, anlamsal veb servisleri, etmen teknolojileri, Yapay Zeka (YZ) planlaması ve tezde önerilen mimariye temel teşkil eden soyut Anlamsal Veb Servisleri Mimarisi hakkında temel bilgiler verilmiştir. Bölüm 3, neden etmen - veb servis ilişkisi olması gerektiğini tartışmakta, çözüm bekleyen sorunların bir listesini sunmakta ve bu sorunların çözümüne yönelik somut bir mimarinin elemanlarını ve işleyişini ortaya koymaktadır. Bölüm 4'te ise, bir önceki bölümde ortaya konan mimarideki etmenlere yönelik planlayıcının yapısı ve aynı mimarinin süreçlerine yönelik planlar ve planlama (veya planlayıcı) ihtiyaçları tartışılmaktadır. Son olarak, Bölüm 5, yapılan bütün tezin bir tartışmasını yapmaktadır ve ileriye dönük araştırma olanaklarına değinerek teze son vermektedir.

2 ALTYAPI

Varoluğundan beri Dünya Geneli Veb (DGV), bilgisayarların veb sayfası içeriğini, niyet edilen anlamına eriştirmeye olanak tanımayarak, yalnızca gösterim amaçlı anlamasına izin vermiştir. Günümüzde ise Veb bilgi ve belgelerin yayınlanmasına yarayan bir araç olmaktan çok daha öteye gitmektedir. Veb kaynakları uygulamalara önyüz sağlayan servisler olabilmektedirler. Anlamsal Veb, makine tarafından işlenebilir bilgi ve servisleri olanaklı kılmak için Veb'i makineler tarafından anlaşılır üstveri bir katman ile zenginleştirmeyi amaçlamaktadır. Anlamsal Veb ayrı bir Veb değildir, mevcut olanın bilgi ve servislere iyi-tanımlı anlamlar verilerek genişletilmiş halidir. Böylelikle bilgisayarların ve insanların birlikte çalışması daha olanaklı hale gelmektedir. Endüstri-güdümlü veb servis etkinliklerinin anlamsal veb çabasıyla tümleşimi, Anlamsal Veb'in başarılı olabilmesi için önemlidir. Anlamsal Veb'in yaratılması için DGV Konsorsiyumu (DGVK), KTÇ[1] (Kaynak Tanımlama Çerçevesi) ve VOD[2] (Veb Ontoloji Dili) gibi standartlar ortaya koymuştur. Endüstri ve akademide de anlamsal Veb servisleri için S-VOD[3] (Servisler için VOD), VSMO[4] (Veb

[1]Kaynak Tanımlama Çerçevesi (*Resource Description Framework*), http://www.w3.org/RDF/, son erişim 25 Temmuz 2007.

[2]Veb Ontoloji Dili (*Web Ontology Language*), http://www.w3.org/TR/owl-features/, son erişim 25 Temmuz 2007.

[3]Servisler için Anlamsal Biçimleme (*Semantic Markup for Services*), http://www.daml.org/services/owl-s/, son erişim 25 Temmuz 2007.

[4]Veb Servis Modelleme Ontolojisi (*Web Service Modelling Ontology*), http://www.wsmo.org/, son erişim 25 Temmuz 2007.

Servis Modelleme Ontolojisi), A-VSTD[5] (Anlamsal açıklamalı Veb Servis Tanımlama Dili) gibi açık standartlar üzerinde etkin bir şekilde çalışılmaktadır. Anlamsal servisleri kullanacak yazılımlar olan etmenler Anlamsal Veb'de önemli bir role sahiptirler. Etmenler dağıtık İnternet ortamında çalışan, birbirleriyle haberleşen, birbirlerinin servislerini kullanabilen yazılım bileşenleridirler. Etmenler, davranışlarını belirleyen planlama mekanizmalarına sahiptirler. Dolayısı ile planlamanın da anlamsal veb ile ilişkisi vardır. Bu bölümde bu alanların herbiri bu tezin altyapısı olarak tartışılacaktır.

2.1 Anlamsal Veb

Anlamsal veb, vebdeki veriyi hem insanların okuyabileceği hem de makinelerin anlayabileceği şekilde tanımlar ve bağlar. "İnsanların okuyabileceği"nden kasıt geleneksel metin/resim veb belgelerinin makina tarafından gösterimi ve insan tarafından kullanılmasıdır. "Makinelerin anlayabileceği"nden kasıt ise verinin çıkarsama için hazır olması ve çeşitli uygulamalarda yeniden kullanılabilir olmasıdır. Bilgi gösterimi, bilgi getirme, veritabanları, çok-etmenli sistemler, doğal dil işleme ve makine öğrenimine gibi birçok alanı kapsayan Anlamsal Veb, veb bilgisinin anlamını modeller.

Anlamsal Veb, Veb mimarisine dayanır: kaynakları tanımlamak için tekbiçimli kaynak tanımlayıcılarını (*Uniform Resource Identifier - URI*), ontolojileri birleştirmek için bağlantıları, sözdizim için genişletilebilir biçimleme dilini (GBD, *eXtensible Mark-up Language - XML*), ileti-

[5] Anlamsal VSTD (*Semantic Annotations for WSDL - SAWSDL*), http://www.w3.org/2002/ws/sawsdl/, son erişim 25 Temmuz 2007.

şim için de üstün-metin aktarım protokolünü (ÜMAP, *Hyper-Text Transfer Protokol - HTTP*) kullanmaktadır. Anlamsal Veb'de bilgi, makine tarafından işlenebilir ve anlaşılabilir bir anlama sahiptir. Anlamsal Veb'in temel yapı taşı ontolojilerdir. Ontolojiler kavramları ve ilişkileri tanımlarlar. Ontolojinin eksiksiz bir tanımını yapılmaya çalışılacak olunursa:

> *Ontoloji paylaşılan bir kavramsallaştırmanın biçimsel, belirgin bir tanımıdır.*

Bu tanımda irdelendiğinde *paylaşılan* kelimesi ile anlatılmaya çalışılan ontolojilerin genel kabul görmüş bir anlayışı ifade etmesidir. *Kavramsallaştırma* kelimesi bir alanın kavramsal modelinin ortaya çıkartılmasıdır. *Biçimsel bir tanım* olması sayesinde makine okunabilirliği sağlanır ve *belirgin bir tanım* olması sayesin de ise kesin terminoloji tanımları içerir.

2.2 Veb Servisleri

Veb-tabanlı uygulamalar arasındaki birlikte işlerlik ihtiyacının artmasıyla beraber, veb servisleri hızla genişlemiştir. Veb servisleri İnternet üzerinde bulunan, modüler ve kendini-tanımlayan uygulamalardır. Veb servisleri, kullanıcıların dinamik bağlam-güdümlü veb-tabanlı uygulamaları yapmalarına ve bu uygulamaları birleştirerek daha karmaşık işler yapmalarına olanak sağlar.

Veb servisleri, adından da anlaşılacağı gibi, veb aracılığıyla sunulan servislerdir. DGVK Veb Servisleri grubu veb servisleri için şu tanımı vermektedir[6]:

[6]DGVK Veb Servis Mimarisi Grubu, "Veb Servis Mimarisi",

> *Bir Veb servisi bir ağ üzerinde makiden-makineye birlikte işlerliği desteklemek için tasarlanmış bir yazılım sistemidir. Makine tarafından işlenebilir şekilde hazırlanmış bir arayüze sahiptir (özellikle VSTD). Diğer sistemler veb servisi ile tanımına göre diğer Veb'le ilgili standartlarla birlikte ÜMAP üzerinden taşınan GBD ile serileştirilmiş BNEP mesajları kullanarak etkileşirler.*

Bu tanımda görüldüğü gibi GBD veb servisleri için temel teknolojidir. Veb servislerinin özellikleri şunlardır:

- Birlikte işlerlik: Veb servisleri programlama dillerinden bağımsızdırlar ve sistem sınırları içerisinde çalışırlar;
- Modülerlik: Veb servisleri bileşen nesne modelini kullanır ve yeni servisler yaratmak için yeniden kullanılabilir veya birleştirilebilirler;
- Çok yönlülük: Veb servisleri hem insanlar hem de yazılım etmenleri tarafından kolaylıkla erişilebilecek bir şekilde geliştirilmiştir;
- Mevcudiyet: Veb servisleri mevcut veb altyapısını kullanmaktadır. Bunun bir sonucu olarak, veb servisleri İnternet üzerinde her yerden erişilebilirdirler.

Bir veb servisi, diğer veb servislerinden bağımsız ve bir sonuç döndürmekten başka bir etkileşimi olmayan tek bir veb-erişimli bilgisayar programının çalıştırılması kadar basit olabilir. Örneğin, posta kodu verilen bir

http://www.w3.org/TR/ws-arch/, son erişim 3 Temmuz 2007.

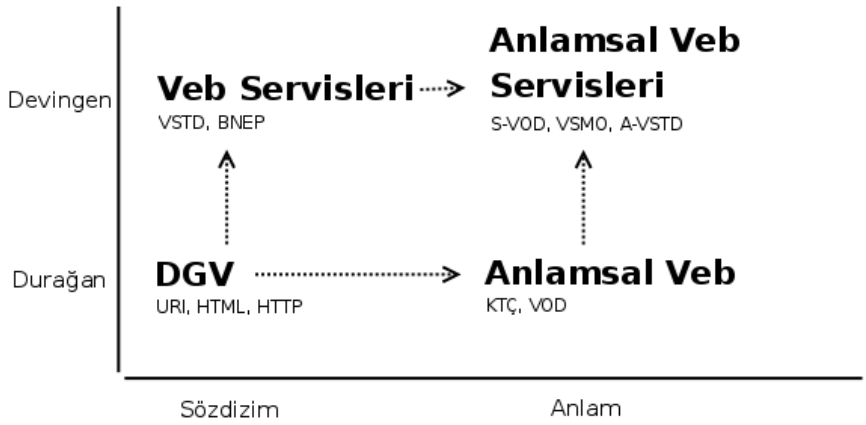
yerin sıcaklığını döndüren bir hava durumu servisi bu kategoride olabilir. Bununla beraber, bir veb servisi, birden fazla veb servisinden oluşmuş ve genellikle servis istemciler ile servis sağlayıcılar arasında daha fazla etkileşimi gerektiren karmaşık bir servis olabilir. Örneğin, bir seyahat ayarlama servisi otel ve havayolu rezervasyon servislerini kullanır, istemcilerinin çoklu seyahat seçenekleri arasından seçim yapmasını ister ve onlara seyahat paketleri önerir.

Yayınlandıktan sonra, veb servis sağlayıcıları sunucu konumuna geçerler ve servis istemcileri tarafından keşfedilip çalıştırılmayı beklerler. Tipik bir senaryoda, bir servis istemci belirli bir adresteki servis sağlayıcıya BNEP protokolünü kullanarak ÜMAP üzerinden bir istek mesajı gönderir. Mesajın alınmasına karşılık, servis sağlayıcı isteği işler ve eğer gerekiyorsa istemciye bir sonuç mesajı gönderir.

2.3 Anlamsal Veb Servisleri

Anlamsal veb'deki servislerin otomasyonu için birçok veb servis ontolojisi ve çeşitli araçlar gerçekleştirilmiştir. Bu ontolojiler içerisinde en bilinenleri S-VOD, VSMO ve A-VSTD ontolojileridir (Şekil 2.1). Tezde, gerek genel bir servis ontolojisi olarak tanımlanması sebebiyle gerekse de planlama formalizmine olan yakınlığı sebebiyle S-VOD ontolojisi tercih edilmiştir. S-VOD ontolojisi kısaca aşağıdaki görevleri başarmayı hedeflemektedir:

- Otomatik servis keşfi: İstenilen kısıtları yerine getiren servisi bulan ontolojisi güçlü bir arama motoru kullanılmaktadır;
- Otomatik servis çalıştırımı: S-VOD, otomatik servis çalıştırmak için



Şekil 2.1. Anlamsal Veb Servisleri

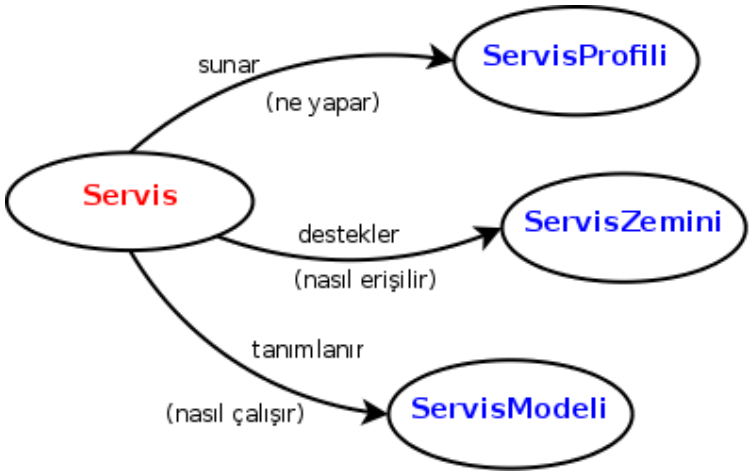
gerekli olan birçok tanımlayıcı ve bilgisayar tarafından yorumlanabilir bir programlama arayüzü sunmaktadır. Bir yazılım etmeni gerekli olan girdiyi, dönecek olan bilgiyi ve servisin nasıl otomatik olarak çalıştırılacağını tanımı yorumlayarak anlayabilir;

- Otomatik servis birleştirimi ve birlikte işlerlik: S-VOD servislerin önkoşullarını ve sonuçlarınının tanımlayıcı tanımlarını sağlamaktadır. Bir yazılım etmeni verilen bir görevi otomatik olarak başarmak için birçok veb servisi arasından seçim ve birleştirim yapabilir;

- Otomatik serviş çalıştırma izlenmesi: S-VOD servislerin çalışması için tanımlayıcılar sağlamayı hedeflemektedir.

S-VOD üç soruyu cevaplayan servis ontolojileri sunmaktadır (Şekil 2.2):

- "Servis ne yapar": Bir servisin yeteneklerini gösteren servis profili, servisi arayan etmene servisin onun ihtiyaçlarını karşılayıp karşılamadığını belirlemede yardımcı olur;



Şekil 2.2. S-VOD Üst Ontolojisi

- "Servis nasıl çalışır": Süreç modeli servisin yürütülmesi için gereken detaylı girdi ve çıktı bilgilerini tanımlar. Servisi arayan etmen süreç modeli üzerinde derinlemesine bir analiz yapar ve servisin onun ihtiyaçlarını karşılayıp karşılamadığına karar verir. Servis modeli ayrıca birçok servisi bir görev için birleştirmek için önemlidir. Birleştirme eylemi temel olarak servislerin girdilerini ve çıktılarını eşleştirir ve birleştirir;

- "Servise nasıl erişilir": Servis zemini bir servise nasıl erişileceğinin detaylarını tanımlamaktadır. Tipik olarak servis zemini bir iletişim protokolü, mesaj şekillerini ve servise bağlanmak için gereken soket port numarası gibi diğer servise-özel detayları içerir.

2.4 Etmenler

Anlamsal veb tarafından sunulan anlamsal-kodlanmış bilginin tüketicisi bu bilgiyi anlayıp kullanacak kadar zeki olmalıdır. Yazılım etmenleri bu role mükemmel bir şekilde uymaktadırlar. Rasıl ve Norvig etmen için şu tanımı vermektedir(Russell and Norvig, 2003):

> *Etmenler bulundukları ortamı algılayıcıları ile algılayan ve ortama karşı etkileyicileri ile eylemde bulunan elemanlardır.*

Bu tanıma göre, etmenler geleneksel YZ-sistemlerine benzemektedir, çünkü her ikisi de ortam ile etkileşir ve eylemlerde bulunur. Fark şu ki, geleneksel YZ-sistemlerinde ortamı gözleyip onu bilgisayara tanımlayan bir insan operatör bulunmaktadır. Bir otonom etmen, diğer yanda, ortamı kendisi gözler ve gözlemlerini tanımlara çevirir. Dahası, etmen hesaplama sonuçlarını yorumlar ve uygun eylemlerde bulunur.

Etmenler genellikle aşağıdaki özelliklerin bazılarını veya hepsini içerir:

- Uyarlanabilir: Ortamdaki değişikliklerle uygun bir şekilde ve zamanında başa çıkabilmelidir;
- Taktiksel: Birçok hedefi ele alabilmelidir ve içinde bulunduğu duruma göre etkin hedeflerini değiştirebilmelidir;
- Çok yönlü: Geniş bir perspektifteki görevleri yerine getirebilmelidir;
- İşbirlikçi: Diğer etmenlerle etkileşebilmelidirler. Hedefler diğer etmenlerle işbirliği yaparak ve rekabet ederek başarılır.

Yazılım etmenleri gerçek dünya ile nasıl ve ne derece etkileştiklerine göre birçok kategoriye ayrılırlar. Temel olarak, bir etmen için iki senaryo bulunmaktadır: ya etmen bulunduğu ortamda yalnızdır ya da ortamda başka etmenler de vardır. Çok-etmenli sistemler genelde heterojen olarak kabul edilirler. Yani etmenler farklı tiplerdedir ve tek bir tasarımcı tarafından belirlenmemişlerdir. Çok-etmenli çerçevelerin bir örneği Etmen Tabanlı Anlamsal Servis Mimarisi'dir(Gürcan et al., 2007a). Servis sağlayıcı etmenler harici servisleri platforma uyarlarak ilan etmektedirler.

Etmenler, davranmalarını sağlayan bir plan mekanizmasına sahiptirler. Plan kısaca ileriki davranışın bir gösterimi olarak tanımlanır. Etmenler planlayıcılarının plan kütüphanesindeki planları uygun zaman ve koşulda tek başına veya diğer planlarla birlikte kullanması vasıtasıyla davranabilmektedir. Dolayısı ile etmen mimarisinde planlayıcı hayati bir önem taşımaktadır.

2.5 Planlama

Belirli bir hedefe ulaşan eylem dizisini oluşturma işine *planlama* denir. Otomatikleşmiş planlama, akıllı davranışın bileşenlerinden birisi olması ve, uzay araç ve robotlarının kontrolünden üretim işletmelerine ve briç oyununa kadar çeşitli uygulamalarda kullanılmaya başlanması sebebiyle oldukça popüler bir konudur. Yapay Zeka (YZ) alanında büyük ilgi uyandıran bu alanın bu kadar ilgi görmesinin sebeplerinin en önemlisi YZ'nin iki temel alanını birleştirmesinden kaynaklanmaktadır: *arama* ve *mantık*(Russell and Norvig, 2003). Bu yüzden, bir planlayıcı hem çözüm arayan bir program hem de bir çözümün varlığını kanıtlayan bir program ola-

rak görülebilir.

Planlama konusuna başlamadan önce *plan* kelimesinin anlamını netleştirmek gerekmektedir. Sözlük[7] anlamına göre plan:

1. Bir düzlem üzerinde çizilen çizim veya diyagram. a) bir nesnenin üstten veya yandan görünümü, b) küçük bir alanın geniş ölçekli haritası.

2. a) Bir sonuca ulaşmak için bir yöntem, b) birşeyi yapmanın alışılagelen yöntemi, c) bir eylem programının detaylı formülü, d) hedef, amaç.

3. Bir tasarım veya hedefin parçalarının sırayla düzenlenmesi.

4. Detaylı bir program (ödemedeki veya bir servisin sağlanmasındaki gibi) <emekli maaşı planı>

YZ açısından plan, sözlükte belirtilen anlamlardan 2 ve 3üncüye karşılık gelmektedir. MTE Bilişsel Bilimler Ansiklopedisi'nde ise plan daha net bir şekilde şöyle tanımlanmaktadır: *bazı etmen veya etmenler tarafından çalıştırılan, genellikle bazıları geçici olan kısıtlara sahip eylemlerin bir kümesinden oluşan ileriki davranışın bir gösterimidir* (Tate, 1999).

Bilgisayar programları insanlara planlama yapma konusunda oldukça yardımcı olmaktadırlar. Bunlara örnek olarak, proje yönetimi (tüketici yazılımı), plan depolanması ve erişimi (üretimde değişken süreç planlaması) ve otomatik takvim yaratılması (çeşitli VEYA ve YZ teknikleri kullanılarak) verilebilir. Bununla beraber, bazı problemlerin çözümü için

[7] http://www.webster.com/, son erişim 4 Temmuz 2007.

planların otomatik olarak yapılmasını isteriz. Bu çok daha zordur, ancak otomatikleştirilmiş planlama araştırmaları sonuç vermeye başlamıştır. Örnek olarak şunları verebiliriz:

- Uzay araştırmaları. Otonom planlama, zamanlama ve kontrol (NASA tarafından geliştirilen JPL ve Ames); Uzak Etmen Denemesi (Remote Agent Expirement RAX - Deep Space 1); Mars Keşif Gezgini (Mars Exploration Rover - MER)

- Üretim. Madeni levha büküm makineleri (Amada Şirketi). Büküm sıralarını planlamak için yazılım(Gupta and Bourne, 1999).

- Oyunlar. Bridge Baron(Smith et al., 1998)

2.5.1 Planlama Çeşitleri

Bilgi-tabanlı, elle-ayarlanabilir planlama sistemleri gerçek-dünya sorunlarını çözmesi en ümit verici planlama sistemleridirler.

Planlama sistemleri genel olarak aşağıdaki gibi üç kategoriye ayrılabilir:

- *Alana-özel planlayıcılar*. Yalnızca bir problem alanında çalışırlar. Alana-özel planlamada, her yeni problem için planlayıcı, o problem için planlama sürecinin etkinliğini artırmaya yönelik özel teknikler kullanılarak en baştan inşa edilir. *ICAPS*(Gupta et al., 1998), *Remote Agent*(Muscettola et al., 1998), *Bridge Baron*(Smith et al., 1998) ve *Mars Rover*(Dias et al., 2003) örnek olarak verilebilir.

- *Tam otomatik alan-bağımsız planlayıcılar* (örneğin, neredeyse bütün klasik planlama sistemleri). Bu gibi planlayıcıları kullanmanın en büyük avantajı, planlama mekanizmasını soyutlayarak, kullanıcının diğer alanlar için fazla çaba gerektirmeden onu yeniden kullanmasını sağlamasıdır. Buna rağmen, (Bylander, 1991; Erol et al., 1995)'te gösterildiği üzere, mantıklı kısıtlamalara rağmen, genel-amaçlı planlamanın zorluk derecesi en az *PSPACE*[8]*-zor*dur. Bu, gerçek-dünya alanlarında genel-amaçlı planlamayı daha az çekici hale getirmektedir. Bu planlama grubunun son bazı örnekleri arasında *LPG*(Gerevini and Serina, 2002), *FF*(Hoffmann and Nebel, 2001) ve *MIPS*(Edelkamp and Helmert, 2001) bulunmaktadır.

- *Elle-ayarlanabilir alan-bağımsız planlayıcılar*. Elle-ayarlanabilir bir planlayıcı genellikle genel-amaçlı bir planlayıcıdır. Buna rağmen, alan tanımının yalnızca planlama operatörü içerdiği otomatik planlayıcılara kıyasla, elle-ayarlanabilir planlayıcılar daha zengin bir alan tanımlama dili sağlarlar, böylece planlayıcının girdisi planın nasıl aranacağı konusunda planlayıcıya tavsiye içerir bir niteliktedir. Elle-ayarlanabilir planlayıcılar bu yüzden kullanıcılarına alan tanımında alana-özel tavsiye belirleme seçeneğini, özellikle alan-bağımsız planlayıcıların esneklik ve yeniden kullanılabilirliklerini alana-özel olanların performansıyla birleştirerek, sunan alan-

[8]*Karmaşıklık teorisine göre PSPACE sınıfı (Savitch teoreminde NPSPACE sınıfına eşittir), polinomial genişlikteki bellek ve sınırsız zaman kullanan deterministik veya deterministik olmayan bir Turing makinesi tarafından çözülebilecek karar problemleri kümesidir.*

bağımsız planlayıcılar olarak tanımlanabilir. En elle-ayarlanabilir planlayıcılar Hiyerarşik Görev Ağı (HGA, *Hierarchical Task Network - HTN*) planlayıcılarıdır; en bilinen örnekleri içinde *Nonlin*(Tate, 1977), *SIPE2*(Wilkins, 1990), *O-Plan*(Currie and Tate, 1991; Tate et al., 1994), *UMCP*(Erol et al., 1994) ve *SHOP2*(Nau et al., 2003) bulunmaktadır. Son zamanlarda, alan bilgisinin kontrol kurallarından oluştuğu elle-ayarlanabilir planlayıcılar inşa edilmiştir; en iyi bilinen örnekler *TLPlan*(Bacchus and Kabanza, 2000) ve *TALplanner*(Kvarnström and Doherty, 2000) planlayıcılarıdır.

HGA Planlamasının ana bilgi yapılarına *metod* denilmektedir. Bunlar üst-seviye görevlerin (görevler gerçekleştirilecek etkinliklerin sembolik gösterimleridirler) daha basit alt-görevlere nasıl ayrıştırılacağını tanımlarlar. Metodlar ayrıca, böyle bir ayrışmanın mümkün olacağı koşullar olan *önkoşul*ları tanımlar.

2.6 HGA Planlaması

2.6.1 Biçimsel Altyapı

Hiyerarşik Görev Ağı (HGA) planlamasında, planlama sistemi *ilkel-olmayan görevler*i (görevler gerçekleştirilecek etkinliklerin sembolik gösterimleridirler) *ilkel görevler*e (örneğin, plan çalıştırıcı tarafından doğrudan çalıştırılabilinecek olan görevler) ulaşılıncaya kadar daha küçük ve daha da küçük altgörevlere *ayrıştırarak* bir plan hazırlar. Temel fikir 70'lerin ortalarında(Sacerdoti, 1975; Tate, 1977) geliştirilmiştir. Formal altyapının gelişmesi biraz daha geç olmuştur, 90'ların ortalarında(Erol et al.,

1996a). HGA-planlaması araştırmaları diğer YZ-planlaması araştırmalarına göre daha uygulama-yönelimli olmuştur ve HGA planlama sistemlerinin bir çoğu birçok uygulama alanında(Wilkins, 1990; Currie and Tate, 1991; Smith et al., 1998; Nau et al., 2005) kullanılmıştır.

Bir HGA planlaması problemi şunlardan oluşur: *ilk durum* (plan çalıştırıcının planı çalıştıracağı sıradaki dünyanın durumunun sembolik gösterimi), *ilk görev ağı* (gerçekleştirilecek görevler kümesi, sağlanması gereken bazı kısıtlarla beraber) ve aşağıdakileri içeren bir *alan tanımı*:

- Çeşitli tiplerdeki, ilkel görevleri yerine getirebilmek için plan çalıştırıcı tarafından doğrudan gerçekleştirilebilen, işleri tanımlayan *planlama operatörleri* kümesi. Bunlar PATD'deki klasik planlama operatörlerine benzer(McDermott, 1998; Fox and Long, 2003).

- İlkel-olmayan görevleri altgörevlere ayrıştırmanın çeşitli mümkün yollarını tanımlayan *metod*lar kümesi. Bunlar, birinin normalde ilkel-olmayan görevleri alanda kullanması için olan "standart işletim yordamları"dırlar. Her bir metod uygulanabilir olmak için sağlanması gereken bir kısıt kümesi içerebilir.

- Seçimsel olarak, yardımcı fonksiyonların tanımları ve dünyanın durumunda açıkça değinilmeyen durumları çıkarsamaya yarayan aksiyomların tanımları gibi diğer çeşitli bilgiler.

Planlama, metodları ilkel-olmayan görevlere, onları altgörevlere ayrıştırmak için uygulayarak ve operatörleri işleri üretmek için ilkel görevlere uygulayarak yapılır. Eğer bu bütün kısıtları sağlayacak bir şekilde yapılırsa,

planlayıcı bir çözüm planı bulur; aksi takdirde planlayıcı geriye dönerek diğer metod ve işleri dener.

2.6.2 Hiyerarşik Görev Ağlarında Birleştirilmiş Bilgi ve Kontrol Akışı

Vilyımsın ve ark. 1996'da yaptıkları bir çalışmada(Williamson et al., 1996) mevcut planlama araştırmalarının birbiriyle ilgili iki konuya (birincisi, bilgi-toplama eylemleri; ikincisi ise, döngüler ve koşullu dallanmalar gibi gelişmiş denetim yapılarına sahip planlar) yoğunlaştığına ancak bu iki araştırma alanının birleştirilmesinin gösterimsel bir probleme yol açtığına dikkati çekmişler; bilgi üreten eylemler, koşullu dallanmalar, periyodik eylemler ve döngülere sahip hiyerarşik planların gösterimi ve çalıştırılması için bir çerçeve ortaya koymuşlardır. İndirgenmiş görevi altgörevleriyle değiştiren diğer bazı HGA planlaması biçimlerinin(Erol et al., 1994) aksine, bu çerçevede görev ağı ağaç benzeri bir yapıda ifade edilmiştir. Ayrıca bu çerçevede *ayrışma* kavramı, yeni altgörevlerin ihtiyaçlarının ve çıktılarının birbirleriyle ve ebeveyn görevleriyle olan ilişkisini tanımlar. Bu altyapıyı baz alan birçok sistem geliştirilmiştir (*RETSINA*, *DECAF*).

Bu mimariye göre HGA, görevleri ifade eden düğümlerden ve iki tip kenar bilgisinden oluşur. *İndirgeme bağlantıları* yüksek-seviye görevlerin ayrışmasını (bir ağaç yapısı) tanımlar. Ebeveyn görevin ayrışmasına ait görevlerin seçiminde kullanılırlar. İkinci tip kenar bilgisi ise görev-düğümleri arasındaki değer (bilgi) geçişini gösteren *ihtiyaç/sonuç bağlantıları*dır. İhtiyaç/sonuç bağlantıları bir görevin sonucunun diğer görevlere nasıl geçtiğini gösterir.

2.6.3 Bilgi Akış İlişkileri

Bir *görev ağı*, eylemler kümesinden ve bu eylemler arasındaki bilgi-akışı ilişkileri kümesinden oluşur. Bilgi akışı üç farklı şekilde olabilir: ebeveyn görevden altgörevlerine (*miras bağlantısı* veya bir başka deyişle ihtiyaç mirası), altgörevlerden ebeveyn göreve (*ters miras bağlantısı* veya bir başka deyişle sonuç ters mirası) ve bir görevden aynı seviyedeki başka bir göreve (*ihtiyaç bağlantısı*).

2.6.3.1 Miras Bağlantısı

Miras, bir görevin altgörevine ihtiyaç bilgisini aktarmasıdır. Bir miras bağlantısı $< \mathrm{T}, \rho_{\mathrm{T}}, S, \rho_S >$ olarak ifade edilir. Bu ifadede S, T'nin bir altgörevi; ρ_{T}, T'nin bir ihtiyacı; ve ρ_S, S'nin bir ihtiyacıdır. Böyle bir bağlantının anlamı ρ_{T}'ye sağlanan her değerin ρ_S'ye geçeceğidir.

2.6.3.2 Ters Miras Bağlantısı

Ters miras, bir görevin ebeveyn görevine sonuç bilgisini aktarmasıdır. Miras bağlantısına benzer bir şekilde, bir ters miras bağlantısı $< \mathrm{S}, \sigma_{\mathrm{S}}, T, \sigma_T >$ olarak ifade edilir. Bu ifadede S, T'nin bir altgörevi; σ_{S}, S'nin bir sonucu; ve σ_{T}, T'nin bir sonucudur. Böyle bir bağlantının anlamı: eğer S'nin çalışması σ_{S} sonucunu üretirse, σ_{S}'nin değeri σ_{T}'ye geçecektir ve T üstgörevi σ_{T} sonucuna sahip olacaktır.

2.6.3.3 İhtiyaç Bağlantıları

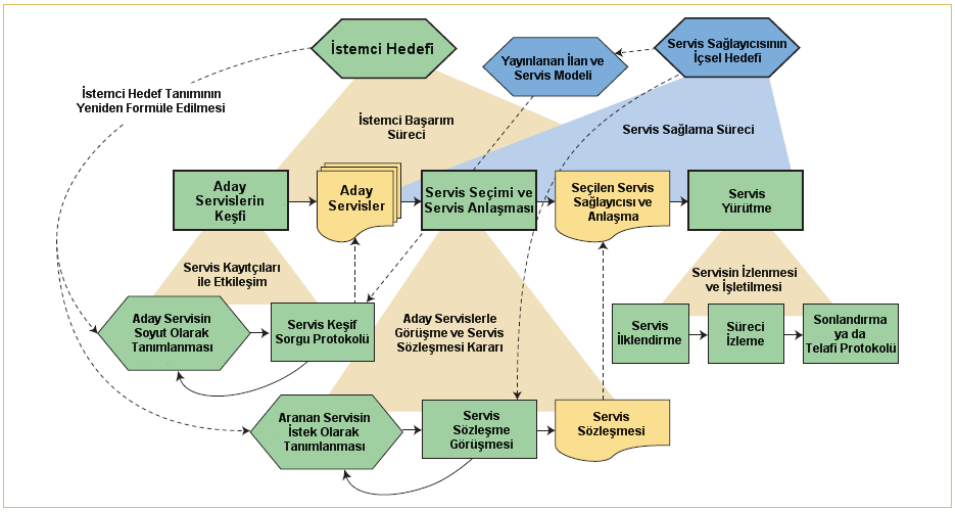
İhtiyaç bağlantıları aynı seviyedeki görevler arasında bilgi akışını sağlar. İki çeşit ihtiyaç bağlantısı vardır: dahili ihtiyaç bağlantısı ve harici ihtiyaç bağlantısı.

- Dahili İhtiyaç Bağlantısı. Dahili ihtiyaç bağlantıları plan içi bilgi akışını temsil etmek için kullanılır. Dahili ihtiyaç bağlantısı $< \mathrm{T}_{\alpha}, \sigma_{\mathrm{T}_{\alpha}}, T_{\beta}, \rho_{T_{\beta}} >$olarak ifade edilir. Bu ifadede T_{α} ve T_{β} aynı seviyede iki görev; $\sigma_{\mathrm{T}_{\alpha}}$, T_{α}'nın bir sonucu; ve $\rho_{T_{\beta}}$, T_{β}'nın bir ihtiyacıdır. Böyle bir bağlantının anlamı: eğer T_{α}'nın çalışması $\sigma_{T_{\alpha}}$ sonucunu üretirse, $\sigma_{T_{\alpha}}$'nın değeri $\rho_{T_{\beta}}$'ya geçecektir.

- Harici İhtiyaç Bağlantısı. Harici ihtiyaç bağlantıları bir göreve plan dışından değer geçirmede kullanılır. Genellikle plan dışı değerler, etmenin diğer etmenlerle iletişim kurup onlardan bilgi beklemesi ile ortaya çıkar.

2.7 Anlamsal Veb Servisleri Mimarisi (AVSM)

Bu tezde tanıtılan platformun dayandığı soyut mimari (Burstein et al., 2005)'te önerilen Anlamsal Veb Servisleri Mimarisi'dir. Bu mimari S-VOD, Veb Servisi Modelleme Çerçevesi - VSMÇ (*Web Service Modeling Framework - WSMF*) ve DGV Konsorsiyumu (*World Wide Web Concortium - W3C*) Veb Servis Mimarisi çalışma gruplarının bir araya gelerek oluşturdukları komite tarafından ortaya konulmuş olup, anlamsal servis çalıştırılması sürecini birbirini takip eden üç alt sürece ayırmaktadır (Şekil



Şekil 2.3. Servis etkileşim süreci. İstemci ve servis sağlayıcısı hedef tanımları etkileşimin 3 adımını (keşif, anlaşma ve yürütme) gerçekleştirmektedir.

2.3). Bunlardan ilki, bir istemcinin amacına uygun servis(ler)i keşfi (*discovery*) sürecidir. Keşfi takiben istemcinin bulunan servisler içinden seçtiği servis(ler) ile uzlaşma (*engagement*) süreci başlar. Uzlaşma, verilen servisin içeriği ve kalite parametreleri bağlamında bir pazarlık sürecini içerir ve pazarlık sonucunda uzlaşan istemci ve servis karşılıklı bir taahhüt oluştururlar. İstemci ile sunucu servis verilecek hizmet konusunda anlaştıktan sonra, servisin çalıştırılması süreci başlamaktadır. Yürütme (*enactment*) adı verilen bu süreçte, istemci, servisin çalışması için gerekli girdileri sağlar ve servisin görevini başarması ya da başaramaması durumunda ne yapacağını bilir.

Aşağıdaki altbölümlerde yukarıda bahsi geçen AVSM süreçleri ve bu süreçlerin gereksinimleri hakkında bilgi verilmiştir.

2.7.1 Keşif Süreci

Keşif süreci, bir servisin hedef(ler)ini gerçekleştirmek için gereken servisleri keşfetmesine ilişkin işlevsel ve kavramsal mimari gereksinimleri tanımlamaktadır. Keşif süreci içinde üç temel aktör/paydaş rol almaktadır:

- Servis Sunucuları, belli servis(ler)i sunduklarını ilan eden servislerdir.
- Servis İstemcileri, hedeflerini gerçekleştirmek için gereken servisleri arayan servislerdir.
- Eşleyiciler (*Matchmakers*), Servis Sunucular tarafından sunulan servislerin tanımlarını alarak, Servis İstemcilerinin talepleri ile bu tanımları eşlemektedirler.

Keşfetme sürecinin işlevsel gereksinimleri, aktörlerin süreç içinde gerçekleştireceği görevleri tanımlamaktadır ve aşağıda listelenmiştir:

- Sunucu(lar), sunacakları servislerin yeteneklerini ve kısıtlarını tanımlayabilmelidir.
- İstemci(ler), arabulucuda bulunan servis yetenek tanımları ile eşlenebilmesi amacıyla gerek duydukları servislerin yeteneklerini soyut olarak tanımlayabilmelidir.
- Eşleyici(ler), istemciler tarafından gönderilen sorgular ile kayıtlanan servis yetenekleri arasında eşleme yapabilmelidir.
- İstemciler, bulunan aday servislerin tanımlarında belirtilen çalıştırma önkoşullarını sağlayabileceği konusunda karar verebilmelidir.

AVSM kavramsal modelinin öneri dökümanında mimari gereksinimleri olarak, keşif süreci aktörleri arasındaki ilişkileri tanımlayan iletişim protokollerinin üst seviye tanımları bulunmaktadır. Fakat bu protokoller kavramsal düzeyde tanımlanmışlardır ve gerçek bir mimarinin gerçekleştirimi sürecinde içeriklerinin (içerik dili, sorgu dili vb.) netleştirilmesi gerekecektir.

2.7.2 Uzlaşma Süreci

Servisler arası uzlaşma süreci, istemci ile potansiyel sunucu arasındaki iletişimin ilk fazıdır. Bu fazın sonunda istemci ve sunucu arasında belirli bir servisin sunucu tarafından verilmesi konusunda bir anlaımaya varılır. Uzlaşma süreci tanımlanacak etkileşimin karmaşıklığına bağlı olarak değişmesine rağmen, bu sürecin işlevsel gereksinimleri dört temel alana bölünmüştür:

- İstemci, geçerli bir servis isteği oluşturmak veya servis pazarlığı yapabilmek için gereken mesaj ve protokol bilgilerine sahip olmalıdır.
- Potansiyel eşler (istemci ve sunucu servisler), ilgili hedef ve yetenek bilgilerini aralarında değişebilmelidirler.
- Potansiyel eşler, servisin sunacakları konusunda bir anlaşmaya varabilmelidirler.
- Eşler anlaşmanın koşullarını belirleyebilmelidirler.

Uzlaşma sürecinin gereksinimlerini karşılamak için mimari seviyede öncelikle bir anlaşma ile sonuçlanacak pazarlık sürecinin protokollerinin ta-

nımlı olması gerekmektedir. AVSM dökümanında bu protokoller sadece isim düzeyinde tanımlanmıştır.

2.7.3 Yürütme Süreci

İstemci ile sunucu servisler verilecek servis konusunda anlaştıktan sonra, servisin çalıştırılması süreci başlamaktadır. İstemci bu süreci gerçekleştirmek için, sunucunun çalışması için gerekli bilgileri belirleyebilmeli ve sunucunun görevini başarması ya da başaramaması durumunda ne yapacağını bilmelidir. Yürütme sürecinin işlevsel gereksinimleri aşağıda listelenmiştir:

- İstemci, kendi isteğine karşı dönen yanıtları yorumlayabilmelidir.
- İstemci ve sunucu faklı ontolojiler kullanıyorsa, bir ontoloji dönüşümü gerçekleştirilebilmelidir.
- İstemci ve sunucu servisler birleşik servis tanımlarını farklı dillerle yapmışlarsa, bu dilleri birbirine dönüştürecek bir Dönüştürme Servisi'ne gerek vardır.
- İstemci, kendi gereksinimlerini karşılayan tek bir servis bulamadığında, gereksinimlerini karşılamak için birleşik servis oluşturma ve çalıştırma desteğine sahip olmalıdır.
- Sunulan servisin durumunun izlenmesi ve istemcinin servisin anlaşıldığı gibi tamamlandığını denetleyebilmesi gerekmektedir.
- Servis, başarısızlık durumları için açık tanımlamalar sunmalı ve buna bağlı kurtarma protokollerini belirlemelidir.

- Sunucu ve istemci, aralarındaki anlaşmazlıkların çözümü için üçüncü-parti servisleri kullanabilmelidir.
- Tüm eşler aradaki iletişimin güvenliğinden ve birbirlerinin güvenilirliğinden emin olmalıdır.

Yukarıda tanımlanan işlevselliğin sağlanması için mimari seviyede aktörler arasındaki protokollerin ve süreci destekleyecek ontolojilerin tanımlanması gerekmektedir.

Yukarıda anlatılan AVSM incelendiğinde mimarinin kavramsal düzeyde geniş bir perspektifte tanımlandığı, fakat mimariyi gerçekleştirmek için gereken detayların ve mimari elemanlarını gerçekleştirmeye dönük altyapılarının tanımlanmamış olduğu gözükmektedir. Önerdiğimiz platform ise bu mimariye ait alt süreçlerin (keşif, uzlaşma ve yürütme) temel işlevlerinin gerçekleştirilmesini amaçlamaktadır.

3 ETMEN TABANLI ANLAMSAL SERVİS MİMARİSİ

Bu bölümün amacı etmenler ve anlamsal veb servislerinin nasıl birlikte çalışması gerektiğini ve çalışabileceğini belirtmektir.

3.1 Etmenler ve Anlamsal Veb Servisleri

Veb servisleri farklı platformlardaki yazılım bileşenlerinin uygun tanım ve iletişim standartları çerçevesinde iletişimini sağlar. Bu sebeple, internet hesaplama alanındaki endüstriyel kesimlerce çok desteklenmektedirler. Bu yolla, Bilgi Teknolojileri (BT) firmaları İnternet üzerinde müşterileri için daha esnek, uyarlanabilir ve faydalı bilgi sistemleri geliştirebilirler. Keza, yazılım etmenleri diğer etmenlerle etkileşerek kullanıcısının hedeflerini gerçekleştirmek için davranışta bulunan elemanlardır. Herhangi bir etmen mevcut veb servislerini kullanıcısının hedefini gerçekleştirmek için dinamik olarak kullanabilir. Bununla beraber, etmenler veb servislerinden farklı iletişim ve eşgüdüm standartları kullanır ve onları dinamik olarak kullabilmek için bu servisler hakkında bazı anlamsal bilgilere ihtiyaç duyarlar. Bu noktada, anlamsal veb servisleri kavramı bize yardımcı olabilir. Anlamsal veb servisleri, işlevleri ve çalıştırılma detayları ontolojiler kullanılarak ifade edilen veb servisleridir.

3.2 Sorunlar

Etmen - veb servis işbirliğinin başarısını engelleyen bazı problemler ve belirsiz durumlar vardır. Bunlar kısaca:

- Etmenler ve veb servisleri farklı iletişim protokolleri kullanmaktadır.
- Etmenler asenkron, veb servisleri ise senkron çalışmaktadır.
- Etmenler ve veb servisleri farklı (iletişim, tanım) standartlar kullanmaktadır.
- Servisler, etmenlerin aksine, ontolojileri kullanmak ve uzlaştırmak üzere tasarlanmamıştır. Eğer bir servisin istemcisi ve sağlayıcısı farklı ontolojiler kullanmaya kalkarsa servisin çağırılmasının sonucu istemci için anlaşılamaz olur. Etmenler ise bu tür farklılıkları ele alabilirler (Singh and Huhns, 2005).
- Etmenlerin veb servislerinin bütünleşik olarak kullandığı bir ortamdaki etmenlerin planlama ihtiyaçları (planlayıcının gereksinimleri ve süreçleri gerçekleyen planların yapısı) net olarak ortaya konmamıştır.

3.3 İlgili Çalışmalar

Etmen - veb servis işbirliği üzerine birçok çalışma yapılmıştır. Yapılan bu çalışmalar 3 alt başlık altında incelenebilir: (1) doğrudan etmen - veb servis ilişkisi üzerine yapılan çalışmalar, (2) anlamsal veb teknolojileri ve aracılık

(*mediation*) üzerine yapılan çalışmalar, (3) YZ planlaması - veb servisleri ilişkisinin potansiyellerini ve sınırlarını araştıran çalışmalar.

3.3.1 Etmenler ve Veb Servisleri

Veb servislerini ZFEK-uyumlu etmen platformları ile tümleştirmek için az sayıda kısmi gerçekleştirimler yapılmıştır. WSDL2Jade (Varga et al., 2003), harici veb servislerini kullanabilen bir kapsayıcı (*wrapper*) etmen yaratmak için etmen ontolojilerini ve etmen kodlarını VSTD girdi dosyasından türetebilmektedir. WSDL2Agent (Varga et al., 2004) insan yardımıyla VSTD girdi dosyasından Veb Servis Modelleme Çerçevesi (VSMÇ) (Fensel and Bussler, 2002) elemanlarını iskeletini alarak veb servislerinin anlamsal veb servis ortamına geçişini etmen tabanlı bir yöntemle tanımlamaktadır. WSIG (Veb Servisleri Tümleşim Geçidi - *Web Services Integration Gateway*) (Greenwood and Calisti, 2004) veb servisleri ve JADE etmenlerinin iki-yönlü tümleşimini desteklemektedir. WS2JADE (Nguyen and Kowalczyk, 2005) veb servislerinin ZFEK-uyumlu etmenlere vekil (*proxy*) etmenler aracılığıyla görünür olması için veb servislerinin çalışmazamanında JADE etmenlerinin servisleri gibi yayınlanmasını sağlamaktadır. Lopez (Lopes and Botelho, 2007), anlamsal veb servislerinin S-VOD tanımı tabanlı ve bağlam-duyarlı çalıştırılması için Servis Çalıştırma Etmeni'nin (*Service Execution Agent - SEA*) geliştirilmesinden bahsetmektedir.

Ancak bu araçlar yalnızca etmenler ile harici veb servislerinin tümleşimini ele almaktadır ve AVSM çerçevesinin belirttiği tam bir mimariyi ve protokol soyutlamalarını sağlayacak bir mekanizma sunmamaktadır ve

birçoğu anlamsal veb teknolojilerini kullanmamaktadırlar. Şurası açıktır ki sıradan geliştiricilerin AVSM tabanlı yazılım sistemlerini geliştirimini kolaylaştırıcı ortamlar olmalıdır.

3.3.2 Anlamsal Veb Servis Teknolojileri

Veb servislerinin anlamsal veb ortamında çalışması için anlamsal veb servisleri alanında bazı standartlaşma çabaları vardır. En çekicileri S-VOD[1] ve VSMO[2]. S-VOD veb servislerinin tanımlanması için bir ontoloji sistemidir. Servis yeteneklerinin ilanı için olan bir profil (*profile*) ontolojisi, işlevselliği ve birleşimi tanımlayan bir süreç (*process*) ontolojisi ve servise nasıl erişileceğinin detaylarını veren bir zemin (*grounding*) ontolojisinden oluşmaktadır. S-VOD anlamsal veb servisleri kavramı için olan ilk çabadır ama tam bir sistem değildir ve bazı elemanlarını anlamı açıkça tanımlanmamıştır. Dolayısıyla, S-VOD çabasını gerçekleştirmek için eşleştiriciler (*matchmakers*), planlayıcılar ve aracılar (*broker*) gibi birçok araştırma grubu tarafından gerçekleştirilmiş işler vardır. VSMO daha bütünleşik bir çerçevedir ama S-VOD ve AVKD gibi DGVK standartları tabanlı değildir. Ayrıca VOD ontolojileri kullanmamaktadır ve dağıtık ve heterojen bir servis ortamındaki bir işakışı sistemine benzemektedir. Bu çalışmada etmenlerin hedeflerini ve servislerini ve harici veb servislerini tanımlamak için S-VOD tercih edildi fakat ayrıca VSMO'da bahsedilen aracılık (*mediation*) fikri de gerçekleştirildi.

Ayrıca etmen teknolojileri ile anlamsal veb servisleri bütünleştir-

[1] http://www.daml.org/services/owl-s/, son erişim 18 Mayıs 2007.

[2] http://www.wsmo.org/, son erişim 18 Mayıs 2007.

mek için bazı çalışmalar yapılmıştır. Örneğin (Gibbins et al., 2003) ve (Zou, 2004)'deki çalışmalar etmen servislerinin tanımlarını Dizin Kolaylaştırıcı'da (*Directory Facilitator*) ilan etmek için ve bu tanımları EİD (Etmen İletişim Dili, *Agent Communication Language - ACL*) mesajlarında taşımak için S-VOD kullanan etmen ortamları tanımlamaktadır. Dikınsın (Dickinson and Wooldridge, 2005), İnanç İstek Niyet (*Belief Desire Intention - BDI*) etmenleri tarafından veb servis işletimini kontrol etmek için reaktif planlama kullanan bir yaklaşım göstermektedir. Bu çalışmalarda, anlamsal veb kaynakları için olan destek sınırlıdır ve tam sistem mimarileri sağlamamaktadırlar. Sutulberg'de (Stollberg et al., 2005), anlamsal vebde otomatikleşmiş hedef çözümlemesi (*resolution*) için bir etmen çerçevesi tanıtılmıştır. VSMO-tabanlı teknolojiler ve araçlar kullanmaktadır. AVSM çerçevesi ile uyumlu değildir ve VSMO'ya olan sıkı bağlantısından dolayı önerilen mimarinin genişletilmesi zordur.

3.3.3 YZ Planlaması ve Veb Servisleri

Anlamsal veb servisi dilleri servisleri girdi/çıktı/önkoşul/sonkoşul cinsinden tanımladıkları için planlama için de uygundurlar. Bu benzerlikten yararlanılarak, veb servislerine planlama operatörü gibi davranmak mümkündür. Dolayısı ile planlama sistemleri bu bilgileri kullanarak veb servislerini plan hazırlar gibi birleştirip karmaşık servisler oluşturabilirler ve oluşturdukları bu servisleri işletebilirler. Keza şu ana kadar yapılmış olan çalışmalar niyet edilen hedefleri başaran veb servis süreçlerinin türetilmesi için YZ planlama tekniklerinin potansiyellerini ve sınırlarını araştırmıştır yani bir başka deyişle veb servislerinin otomatik birleştirimi probleminin plan-

layıcılar tarafından çözülmesine yönelik olmuştur (Sirin et al., 2004b; Traverso and Pistore, 2004; Pistore et al., 2004; Srivastava and Koehler, 2003; Carman et al., 2003; McIlraith and Son, 2002; McDermott, 2002).

YZ Planlaması ve veb servis birleştirimi üzerine çalışmalar yapılmasına rağmen AVSM gibi bütünleşik bir veb servis işletim ortamındaki etmenlerin planlama modüllerinin ne gibi gereksinimlere sahip olduğu ve AVSM süreçlerini gerçekleyecek planlar üzerine bir çalışma yapılmamıştır.

3.4 Etmen Tabanlı Anlamsal Servis Mimarisi

Çözüm olarak AVSM'nin gerçekleştirilmesi seçildi çünkü S-VOD, VSMÇ (Veb Servis Modelleme Çerçevesi - Web Service Modeling Framework) ve DGVK (W3C) Veb Servis Mimarisi çalışma gruplarının bir araya gelerek oluşturdukları AVSM (Anlamsal Veb Servisleri Mimarisi) komitesi Anlamsal Veb Servis teknolojileri için bir temel oluşturacak bir soyut mimariyi, bu mimaride yer alan temel elemanları ve mimarinin işleyişine ait bir dizi protokolleri ortaya koymuştur(Burstein et al., 2005). Bu mimari çoklu-etmen altyapısına oturtulmuştur çünkü gereksinimler hedefe yönelik yazılım etmenleri ve öntanımlı protokollere dayalı asenkron etkileşimler ile başarılabilir.

AVSM, anlamsal servis çalıştırılması sürecini birbirini takip eden üç alt sürece ayırmaktadır. Bunlardan ilki, bir istemcinin amacına uygun servis(ler)i keşfi (*discovery*) sürecidir. Keşfi takiben istemcinin bulunan servisler içinden seçtiği servis(ler) ile uzlaşma (engagement) süreci başlar. Uzlaşma, verilen servisin içeriği ve kalite parametreleri bağlamında bir

pazarlık sürecini içerir ve pazarlık sonucunda uzlaşan istemci ve servis bir karşılıklı taahhüt oluştururlar. İstemci ile sunucu servis verilecek hizmet konusunda anlaştıktan sonra, servisin çalıştırılması süreci başlamaktadır. Yürütme (*enactment*) adı verilen bu süreçte; istemci servisin çalışması için gerekli girdileri sağlar ve servisin görevini başarması ya da başaramaması durumunda ne yapacağını bilir. AVSM çerçevesi aynı zamanda her safhanın oyuncularını, her safhanın işlevsel gereksinimlerini, bu gereksinimlerin soyut protokoller bazında başarmak için gerekli olan mimari elemanlarını da belirlemektedir. AVSM, ÇES altyapısı ve anlamsal veb standartlarına dayalı detaylı bir kavramsal model tanımlamasına rağmen, bu kavramsal modeli gerçekleştirmek için gerekli olan yazılım mimarisini tanımlamamaktadır ve gereken yazılım mimarisinin teorik ve gerçekleştirim detaylarını içermemektedir.

AVSM kavramsal ve geniş bir perspektifte olduğundan, bu mimariyi akılcı bir yolla gerçekleştirilmesi için bazı kabullenmeler yapılmıştır:

- Platformun çalışma alanını ifade eden bir platform ontolojisi vardır. Bu ontoloji platform yöneticisi tarafından tasarlanmıştır ve platformun ontoloji etmeni tarafından saklanır ve yönetilir.

- Platform kullanıcılarının etmen planlarıyla ulaşmak isteyebileceği hedeflerin öntanımlı şablonları vardır. Bu hedef şablonları platform yöneticisi tarafından anlamsal veb servis tanımları gibi (girdiler, çıktılar, önkoşullar, etkiler) tanımlanırlar ve servis kayıtçı etmen tarafından saklanırlar.

- Etmenlerin servisleri anlamsal tanımları kullanılarak bir servis ka-

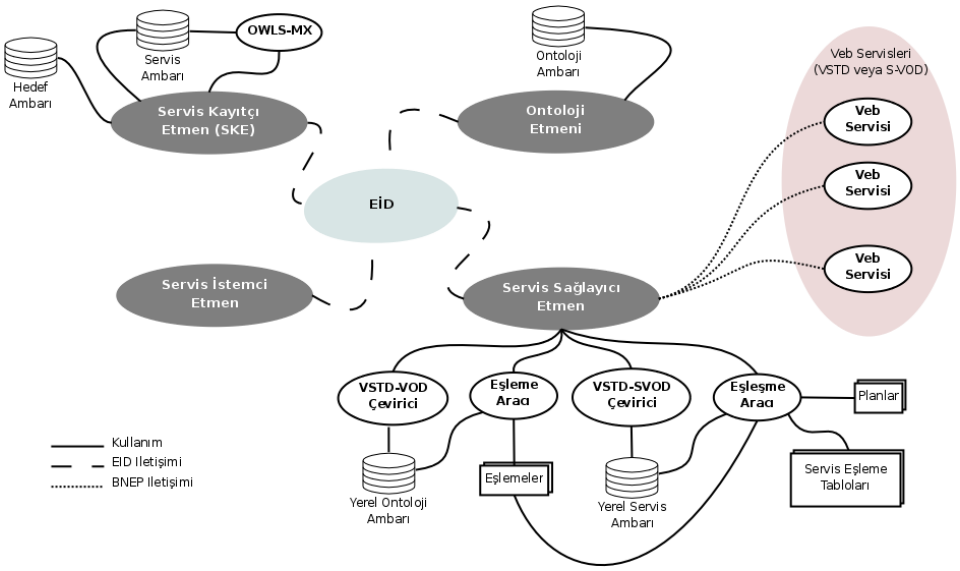
yıtçıya kaydedilir. Bu servisler etmenlerin dahili yetenekleri olabileceği gibi servis sağlayıcı etmenler tarafından platforma dahil edilen harici anlamsal veb servisleri de olabilirler.

- Anlamsal servisin bağımlı olduğu ontoloji ve platform ontolojisi eğer birbirinden farklıysa aralarındaki eşlemeler tanımlanmalıdır. Aksi takdirde bu servis platform etmenleri tarafından kullanılamaz.

Bu yüzden, AVSM'nin bütün ihtiyaçları gerçekleştirilmedi ama AVSM'nin kavramsal modelinin temel gereksinimlerini sağlayan ve çalışan bir altkümesi gerçekleştirildi. Sağlanan etmen platformu aşağıdaki yeteneklere sahiptir:

- Bilgiyi göstermek ve işlemek için anlamsal veb teknolojilerini kullanmaktadır.
- Servis sağlayıcı etmenler aracılığıyla hem saf veb servisleri hem de anlamsal arayüze sahip veb servisleri platforma dahil edilebilmektedir.
- Platforma eklenen servisin ontolojisi ve platform ontolojisi arasındaki uyumsuzlukları servis sağlayıcı etmenler ele almaktadırlar.
- Platformun hedef şablonları ve platforma eklenen harici servisin süreç modeli arasındaki süreç uyumsuzluklarını servis sağlayıcı etmenin sağladığı araçlar çözmektedir.

Önceki bölümde tartışılan AVSM'nin altsüreçlerinin temel gereksinimlerini yerine getirmek için, anlamsal veb servislerinin otomatik keşfi ve



Şekil 3.1. Etmen Tabanlı Anlamsal Servis Mimarisi

çalıştırılması için etmenlerin dahil edildiği bir yazılım mimarisi önerilmektedir (Şekil 3.1) (Gürcan et al., 2007a). Mimari hem saf veb servislerinin hem de anlamsal arayüze sahip veb servislerinin etmenler aracılığıyla kullanımını temin etmektedir. Mimaride anlamsal veb alanında genel kabul gören VSMO ve S-VOD girişimlerinden S-VOD tercih edildi çünkü Lara ve ark.'ın da karşılaştırmasında söz ettiği gibi (Lara et al., 2004) VSMO e-ticaret ve e-iş gibi daha özel alanlarda daha uygunken S-VOD genel olarak tasarlanmıştır. Mimari, geliştirilen diğer sistemlerle birlikte işlerliği sağlayabilmek amacıyla bir EEME ZFEK uyumlu bir Çok-Etmenli Sistem (ÇES) sunmaktadır. Mevcut ZFEK-uyumlu birçok sistem bulunmaktadır: SEAGENT(Dikenelli et al., 2005a), JADE(Bellifemine et al., 2001), AAP[3], ZEUS(Nwana et al., 1999), AgentAcademy(Mitkas et al., 2002)

[3]Fujitsu Amerika Laboratuarları (*Fujitsu Labs of America*). (2001). April Etmen Plat-

bunlardan bazılarıdır. Mimari için SEAGENT çok-etmenli platformu kullanıldı çünkü hem anlamsal veb desteği bahsedilen diğer çerçevelere göre daha iyidir hem de bölümümüzde geliştirilmektedir. Geliştirilen sistemin üyesi olan etmenler önerilen mimarinin ana bileşenlerini oluşturmaktadır: Servis Kayıtçı Etmen, Ontoloji Etmeni, Servis İstemci Etmen ve Servis Sağlayıcı Etmen. Bu etmenler arasındaki iletişim iyi bilinen Etmen İletişim Dili (*Agent Communication Language*)[4] altyapısı ile gerçekleşmektedir.

3.4.1 İç Mimari

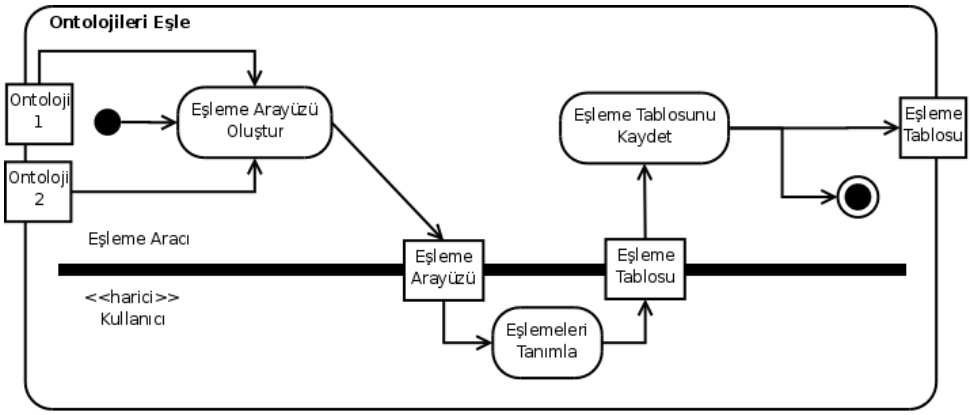
Geliştirilen mimari dört ana bileşenden oluşmaktadır: veb servislerinin platformda kullanımını sağlayan *servis sağlayıcı etmen*, sunulan servisleri dolaylı olarak kullanan *servis istemci etmen*, bütün servislerin kayıtlandığı *servis kayıtçı etmen* ve platformda ontoloji yönetimini sağlayan *ontoloji etmeni*.

3.4.1.1 Servis Sağlayıcı Etmen

Servis Sağlayıcı Etmen (SSE), harici veb servislerini ve anlamsal veb servislerini ÇES'e dahil eder ve etmen-servis etkileşimini sağlar. SSE hem *saf veb servislerini* hem de *anlamsal arayüze sahip veb servislerini* etmen platformuna dahil edebilmektedir. Saf veb servisleri yetenekleri yalnızca VSTD ile tanımlanmış servislerdir. Anlamsal arayüze sahip veb servisleri ise, yetenekleri bir ontoloji temelli bir dil ile tanımlanmış olan servislerdir.

formu proje veb sitesi. http://www.nar.fujitsulabs.com/aap/about.html

[4]FIPA Etmen İletişim Dili Tanımları (*FIPA Agent Communication Language Specifications*), http://www.fipa.org/repository/aclspecs.html, son erişim 16 Mayıs 2007.



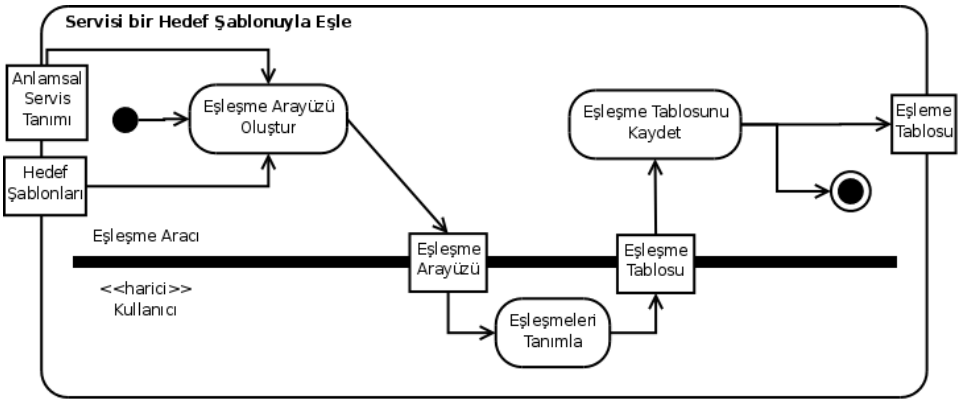
Şekil 3.2. Ontoloji Eşleme Etkinlik Diyagramı

Bu diller S-VOD, VSMO, A-VSTD (Anlamsal VSTD) olabilmektedir.

SSE'nin yöneticisi saf bir veb servisinin eklenmesini istediğinde, servisin VSTD belgesinin adresini SSE'ye verir. VSTD belgesinin adresini alan SSE, o belgede tanımlanan servis(ler)i platformuna eklemek için yönetici yardımıyla gerekli eşleme ve eşleştirmeleri yapar, ilgili planları hazırlar ve servisi platformda yayınlar. Anlamsal arayüze sahip servislerin eklenmesinde ise, çevrim işleri arayüz zaten anlamsal olduğu için esgeçilir ve diğer adımlar saf veb servislerinin eklenmesindeki gibidir. SSE bu işlemleri yapabilmek şu iç bileşenleri kullanır: *VSTD-VOD Çevirici*, *Eşleme Aracı*, *VSTD-SVOD Çevirici* ve *Eşleşme Aracı*.

VSTD-VOD Çevirici, verilen bir VSTD belgesindeki kavramları doğrudan bir VOD belgesine çeviren bir araçtır. VSTD belgesindeki GBD karmaşık tiplerini VOD sınıflarına, GBD elemanlarını da VOD özelliklerine çevirir.

Eşleme Aracı ise, mimari olarak popüler bir ontoloji eşleme aracı olan MAFRA'yı (Maedche et al., 2002) andıran bir ontoloji eşleme ara-



Şekil 3.3. Hedef Eşleşme Etkinlik Diyagramı

cıdır. Kendisine verilen iki ontoloji araşındaki eşlemeleri kullanıcının bir arayüz yardımıyla tanımlasını sağlar ve bir eşleme bilgisi oluşturur (Şekil 3.2).

VSTD-SVOD Çevirici, verilen bir VSTD belgesindeki girdi/çıktı bilgilerini daha önce oluşturulmuş bir kavram ontolojisi belgesi yardımıyla S-VOD servis tanım belgesine çeviren bir araçtır. Benzer bir profil üretme yaklaşımı Şirin ve ark. çalışmasında (Sirin et al., 2004a) bulunabilir.

Eşleşme Aracı, kullanıcı yardımıyla, yayınlanmak istenen bir servisin tanımına uyan bir hedef bulunmasını sağlayan, ikisi arasındaki eşleştirmeleri yapan ve yayınlanmak istenen servisin hedefe uyarlanmasını sağlamak için gerekli etmen planlarının yazılmasına olanak sağlayan bir araçtır (Şekil 3.3).

3.4.1.2 Servis İstemci Etmen

Servis İstemci Etmen (SİE), platformda sunulan servisleri çalıştırmak için istemde bulunan etmendir. SİE, servis bulup çalıştırmak için AVSM'de ta-

nımlanmış olan süreçleri işletmektedir. Bu süreçlere karşılık gelen genel servis çalıştırma planını geliştirici yardımıyla genişleterek kullanır[5]. SİE bir servise ihtiyaç duyduğunda önce SKE'den anlamsal uygunluktaki servisleri ister (keşif), sonra uygun bir servisin sağlayıcı etmeniyle anlaşır (uzlaşma) ve son olarak SSE'den servisi isteğini gerçekleştirmesini ister (yürütme).

3.4.1.3 Servis Kayıtçı Etmen

Servis Kayıtçı Etmen (SKE), EEME ZFEK uyumlu SEAGENT ÇES'inin sarı sayfa servisi olan Dizin Kolaylaştırıcısı'dır (*Directory Facilitator*). SKE etmenler tarafından sağlanan servislerin yeteneklerini ilan eder. Etmenlerin servis ilanlarını S-VOD profili olarak tutan bir *Servis Ambarı* içermektedir. Kendisine bir servis isteği geldiğinde SKE isteğe karşılık gelen en uygun servisleri bulabilmek için istenilen servisin tanımı ile ilan edilmiş olan servislerin tanımları arasında anlamsal bir servis eşleşmesi gerçekleştirir. SKE bu eşleştirmeyi gerçekleştirebilmek için *OWLS-MX* adındaki anlamsal servis eşleyiciyi kullanmaktadır (Klusch et al., 2006). *OWLS-MX*, Kuluş ve ark. tarafından geliştirilmiştir ve S-VOD ile tanımlanmış servisler için hem mantık tabanlı çıkarsama hem de içerik tabanlı bir edinme tekniklerinden faydalanan melez bir anlamsal veb servis eşleştiricidir.

Öte yandan SKE, içerisindeki *Hedef Ambarı*'ında platform hedeflerinin şablonlarını tutmaktadır. Platformda bulunan etmenler yukarıda da bahsedildiği gibi bu hedef şablonlarına uygun servisleri sunabilirler. Do-

[5]Bu planın detayları Bölüm 4.2.1'de anlatılmıştır.

layısıyla servis sunmak isteyen bir etmen öncelikle SKE'den bu şablonları almalı ve sunacağı servisin bunlardan birisine uygunluğunu (gerekirse uyarlanmasını) sağlamalıdır. Benzer şekilde SİE de servis bulup çalıştırabilmek için bu şablonlara ihtiyaç duyar.

3.4.1.4 Ontoloji Etmeni

Ontoloji Etmeni (OE), ontolojileri yönetmekten sorumludur ve platformda kullanılan ontolojileri tutan bir *Ontoloji Ambarı* içermektedir. OE diğer platform üyelerinin platform ontolojilerine kontrollü erişimini ve sorgulamasını sağlamaktadır.

3.4.2 Anlamsal Servis Tümleşim Safhalarının İşleyişi

Mimarinin tasarımı servislerin dört ana safhası etrafında organize edilmiştir: (1) SSE tarafından veb servisin platforma dahil edildiği *ilan (kaydetme) safhası*; (2) SİE tarafından istenilen servisin araştırıldığı *keşif safhası*; (3) SİE ve SSE'nin anlaştığı *uzlaşma safhası*; (4) SİE'nin SSE vasıtasıyla servisi işlettiği *yürütme safhası*.

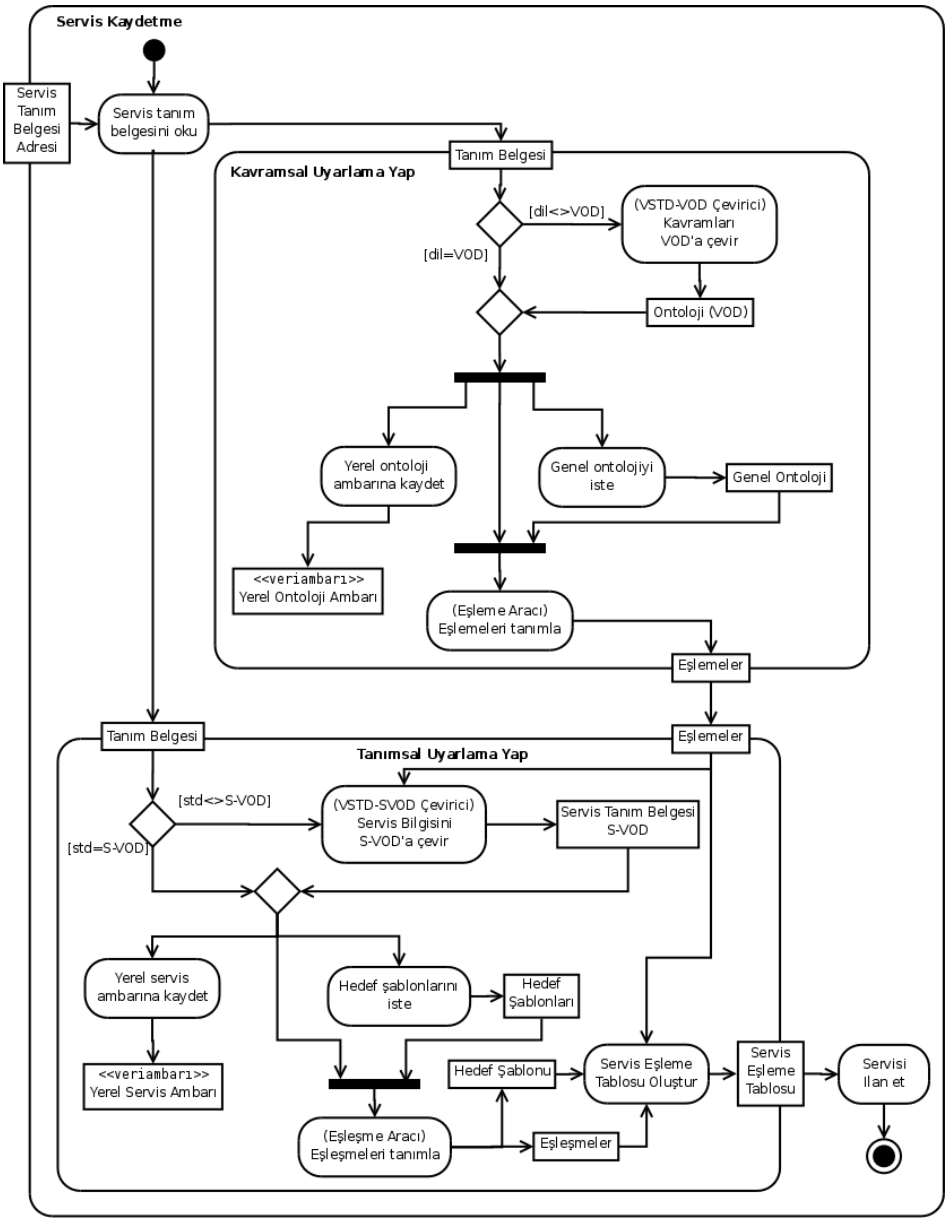
3.4.2.1 İlan (Kaydetme) Safhası

İlan safhası, harici veb servislerinin SSE tarafından platforma dahil edildiği safhadır. EEME ZFEK mimarisinde klasik bir etmen servisinin ilanı, servisi sağlayan etmenin SKE'den sunmak istediği servisin tanımını yayınlaması isteğinde bulunması ve SKE'nin de bu isteği kabul etmesi ile

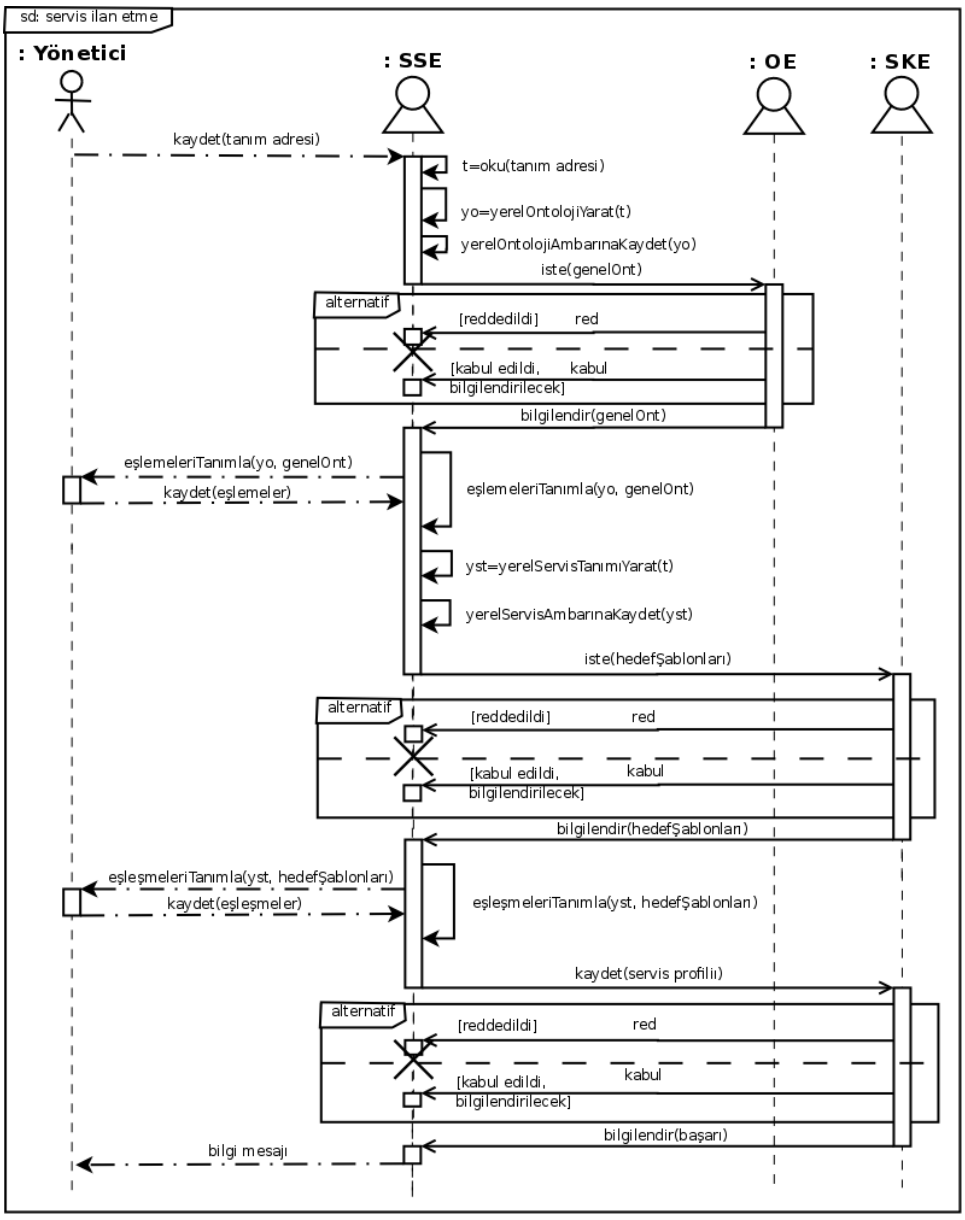
gerçekleşir[6]. Ancak yukarıda da değinildiği gibi etmen her istediği servisi platforma sunamaz, sunmak istediği servisin platform hedefleriyle (hedef şablonlarıyla) örtüşmesi gerekmektedir. Harici veb servislerinin, ister saf veb servisleri olsun ister anlamsal arayüze sahip servisler olsun, ilanı da hemen hemen klasik etmen servislerinin ilanı gibidir. Ancak etmenin bir veb servisini başka etmenlerden gelen isteklere yönelik olarak çağırması ve bu etmenlerin anlayabileceği şekilde onlarla etkileşmesi için sunulacak servisin platforma uyarlanması gerekmektedir. Bu uyarlama sırasında gerek veb servisinin kullandığı kavramlar platformun problem alanına uygun olmalı gerekse de veb servisinin yaptığı iş platform hedefleriyle örtüşmelidir. Aksi takdirde bu veb servisi platforma uyarlanamaz dolayısı ile de sunulamaz.

Şekil 3.4'de servis ilanının BMD etkinlik diyagramı gösterilmiştir. Yukarıda da değinildiği gibi eklenmek istenen servis saf bir veb servisi veya anlamsal arayüze sahip bir veb servisi olabilmektedir. Her iki durumda da SSE'nin yöneticisi harici bir veb servisini eklemek istediğinde, servisin tanım belgesinin adresini sağlayarak SSE'den ekleme işlemini başlatmasını ister. SSE adresteki tanımı yükler ve servisin saf olup olmadığını algılar. Eğer saf bir servis ise SSE, harici veb servisini eklerken o servisin tanımındaki (VSTD belgesindeki) kavramları *VSTD-VOD çevirici* aracı ile anlamsallaştırır ve bir ontoloji belgesi olarak (*yerel ontoloji*) *Yerel Ontoloji Ambarı*'nda saklar. Sonra *Eşleme Aracı* kullanarak yöneticinin de yardımıyla bu *yerel ontoloji*yi, *platform ontolojisi* ile uyumlu hale

[6]FIPA Etmen Yönetim Tanımı (*FIPA Agent Management Specification*), http://www.fipa.org/specs/fipa00023/index.html, son erişim 3 Haziran 2007.



Şekil 3.4. SSE Servis Kayıt Etkinlik Diyagramı



Şekil 3.5. Harici bir Saf Veb Servis İlanı Sıradüzen Diyagramı

getirmesi sağlanır. Bu araç etkinleştiğinde yerel ontoloji ve platform ontolojisini kullanıcıya göstererek birbiriyle ilgili kavramların işaretlenmesini sağlar. Araç, bu işaretleme bittikten sonra oluşan *eşleme bilgisini* bir dosyada saklar. Daha sonra VSTD'de tanımlanmış olan veb servis tanımını (servisin girdi ve çıktıları vd.) *VSTD-SVOD çevirici* aracını kullanarak anlamsal servis tanımına dönüştürür ve onu *Yerel Servis Ambarı*'nda saklar. Bu anlamsal servis tanımları platform servislerinin de tanımlandığı S-VOD ile ifade edilmektedir ve yüksek seviyede girdi ve çıktı tipleri, önkoşullar ve etkiler (GÇÖE) tanımlamaktadır. Bu aşamadan sonra SSE, bu anlamsal servis tanımlarını yöneticinin de yardımıyla platform hedefleri ile örtüştürmeye çalışır. Bunun sebebi platformdaki etmenlerin önceden tanımlanmış genel platform hedeflerine uygun servisler sunması zorunluluğudur. Bunu yapabilmek için SSE önce SKE'den hedef şablonlarını ister ve *Eşleşme Aracı*'nı çalıştırır. Bu araç yöneticinin anlamsal servis tanımı ile ilgili hedef şablonunu örtüştürmesini ve gerekli planları öntanımlı genel servis çalıştırma planlarını[7] kullanarak hazırlamasını sağlar. Bu işlem sonunda Eşleme Aracı bir *Servis Eşleşme Tablosu* oluşturur. Bu tabloda kavramların eşleme bilgisi, yereldeki servisin profili, yereldeki servisin eşleştirildiği platform hedefinin profili ve servisin yürütülmesini sağlayacak olan planı tutmaktadır. Tablo 3.1'de harici bir rezervasyon ekleme servisinin (*RezervasyonEkle*) platforma ilanı aşamasında oluşturulan örnek bir servis eşleme tablosu gösterilmiştir. Tablodaki örnekte harici *RezervasyonEkle* servisinin kavramlarıyla platform kavramları arasındaki eşlemeler *Rezer-*

[7] SSE için öntanımlı genel servis çalıştırma planları Bölüm 4.2.2'de detaylı olarak anlatılmıştır.

Saha	**Değer**
Eşlemeler	RezervasyonEkle.map
Hedef Şablonu	InsertReservation.owls
Yerel Servis Tanımı	RezervasyonEkle.owls
Yürütme Planı	org.sample.plan.BHEnactRezervasyonEkle

Tablo 3.1. Örnek Servis Eşleme Tablosu

vasyonEkle.map dosyası olarak saklanmaktadır, harici servis platformdaki *InsertReservation* hedef şablonuyla örtüştürülmüştür ve servise ait yerel servis tanımı da *ReservasyonEkle.owls* içerisinde tutulmaktadır. Bütün bu işlemler başarıldıktan sonra SSE, anlamsal servis tanımını SKE'ye yollayarak servisi yayınlaması için istekte bulunur. Tablo 3.2'da bu servisin kaydında kullanılması için hazırlanan örnek servis kayıt tanımı gösterilmiştir. Saf servisin ilanına ilişkin sıradüzen diyagramı Şekil 3.5'de gösterilmiştir[8].

Eklenmek istenen servisin anlamsal bir arayüze sahip olması durumunda arayüz dilinin platform servislerinin tanım dili olan S-VOD ile tanımlanıp tanımlanmayışına göre işleyiş değişmektedir. S-VOD olması durumunda dilin çevirilmesi gerekmez, yalnızca anlamsal uyumun sağlanması gerekir. Dolayısı ile tanım belgesi doğrudan Yerel Ontoloji Ambarı'na konur ve ondan sonraki adımlar aynen devam eder. Ancak tanım dilinin S-VOD olmaması durumunda işleyiş tamamen saf servislerin eklenmesi gibidir. Önce tanım belgesi S-VOD diline çevrilir, sonra anlamsal

[8]Bu diyagram ve daha sonraki sıradüzen diyagramları Etmen UML (*Agent UML - AUML*) gösterimiyle çizilmiştir. Daha fazla bilgi için http://www.auml.org/, son erişim 18 Haziran 2007.

Saha	**Değer**
Name	Deniz_Oteli_Etmeni
Type	Otel_Etmeni
Protocol	FIPA-Request, FIPA-Query
Ontology	Turizm
Language	OWL
Ownership	Ege_Universitesi
Properties	(property :name owlsProfile :value InsertReservation.owls)

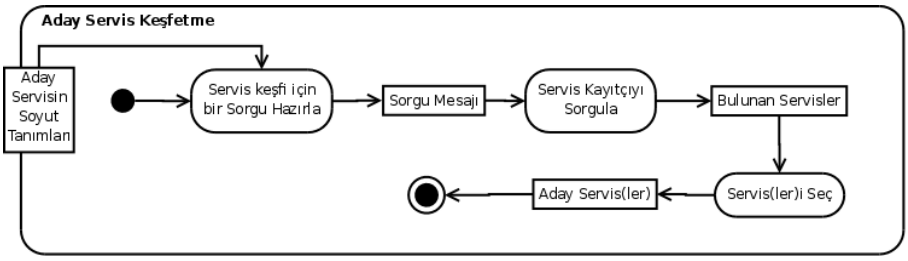
Tablo 3.2. Örnek Servis Kayıt Tanımı

uyum sağlanır.

3.4.2.2 Keşif Safhası

Anlamsal servis keşfini gerçekleştirmek için, platform servisleri eşleştirme yeteneğine sahip bir servis kayıtçıya kaydedilmelidirler. Servis istemcileri bu kayıtçıyı sorgulamalıdır ve dönen servis ilanlarını yorumlayabilmelidirler. Önerilen platformdaki SKE, kayıtlı servislerin yetenek ilanlarını S-VOD profili olarak kendi servis ambarında saklar ve bu servis tanımları sorgulanarak etkin servislerin anlamsal keşfine olanak verir.

İstemci etmenin bütün servis çalıştırma aşaması, AVSM için öntanımlı genel bir anlamsal servis çalıştırma planına dayanan bir plan tarafından ele alınmaktadır. Daha önceki çalışmalarımızda da anlattığımız (Gürcan et al., 2007b; Gürcan et al., 2006) bu genel plan Bölüm 4.2.1'de detaylı olarak anlatılacaktır. Servis İstemci Etmen'in keşif süreci bu planın ilk adımı kullanılarak yürütülmektedir. Şekil 3.6'deki etkinlik diyag-



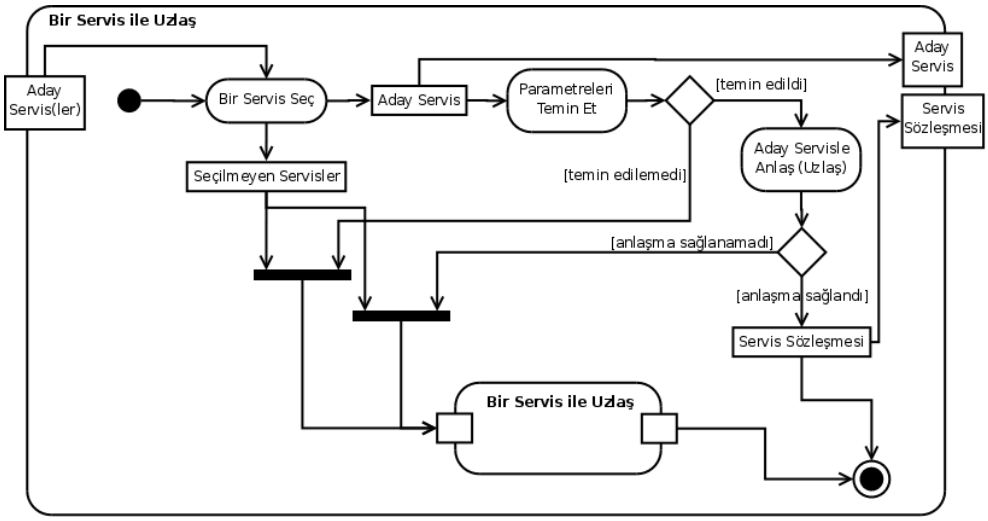
Şekil 3.6. SİE Aday Servis Keşfetme Etkinlik Diyagramı

ramında da görüldüğü gibi SİE bir servise ihtiyaç duyduğunda (hedefini gerçekleştirecek bir servis), keşif safhasında niyet edilen servisin soyut yetenek tanımlarını kullanarak bir sorgu hazırlar. Bu sorgu ayrıca getirilecek uygun servis tanımlarının anlamsal eşleşme derecesini de içerir[9]. Bu yetenek tanımı S-VOD profili olarak ifade edilir ve platformun yetenek tanımlarına (hedef tanımlarına) uyuyorsa geçerlidir. İstemci için uygun iletişim protokolü ve içerik dili S-VOD servisleri için zaten tasarlanmış ve gerçekleştirilmiştir (Dikenelli et al., 2005c). Sonra SKE sorgu isteğini alır ve anlamsal servis eşleştiriciyi ve servis ambarını kullanarak anlamsal uygunluktaki servisleri bulur ve döndürür. Bundan sonra SİE bulunan servisleri alır, içlerinden bir veya daha fazlasını seçer ve sonra bu seçilen servislerin sağlayıcılarıyla uzlaşma sürecini anlamsal servis çalıştırma planının bir sonraki aşaması olarak başlatır.

3.4.2.3 Uzlaşma Safhası

Servis seçim işi bittikten sonra, servis kalitesi ölçütleri üzerine görüşme, anlaşma sağlama ve izlemeyi içeren istemci-sağlayıcı uzlaşma süreci baş-

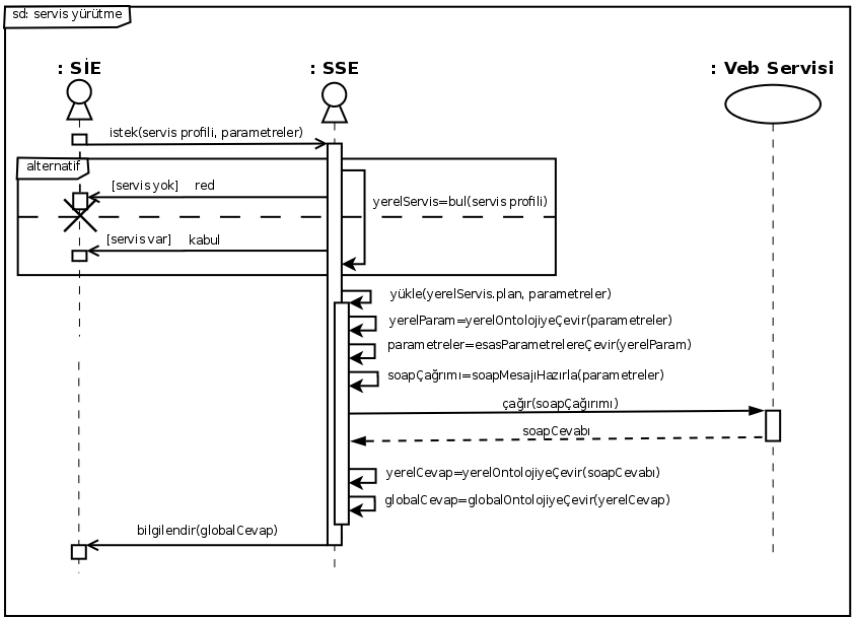
[9] Anlamsal eşleşme Bölüm 3.4.1.3'de anlatılmıştı.



Şekil 3.7. SİE Bir Servis ile Uzlaşma Etkinlik Diyagramı

lar. Önerilen platformda bu aşama basitçe gerçekleştirildi. Sadece çeşitli çalışmalarda (Zeng et al., 2003; Cardoso et al., 2004) tanımlanan bazı kalite parametrelerini (servis çalıştırma maliyeti, çalışma-zamanı, yer, vb.) kullanıldı.

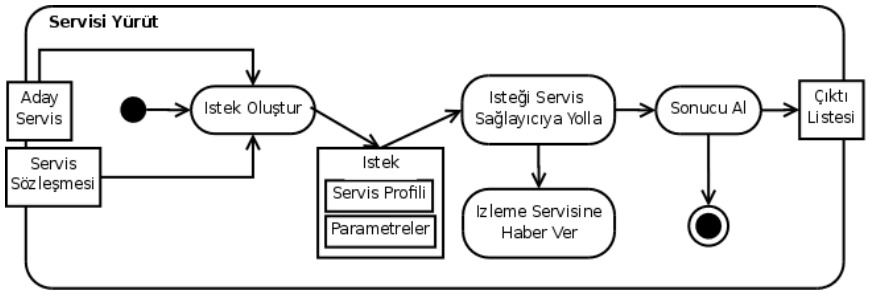
Şekil 3.7'de bir servis ile uzlaşmak için olan sürecin etkinlik diyagramı gösterilmiştir. Şekilde de görüldüğü gibi bu süreçte keşif sürecinde elde edilen aday servislerden birisi seçilerek parametreleri temin edilmeye çalışılır ve parametreleri temin edilebilen servisin sağlayıcısı ile uzlaşılırak bir servis sözleşmesi oluşturulur. Parametrelerin temin edilememesi veya servis ile uzlaşılmaması durumunda keşif sürecinden gelen diğer servislerle aynı süreç tekrarlanır.



Şekil 3.8. Harici Servis Yürütme Sıradüzen Diyagramı

3.4.2.4 Yürütme Safhası

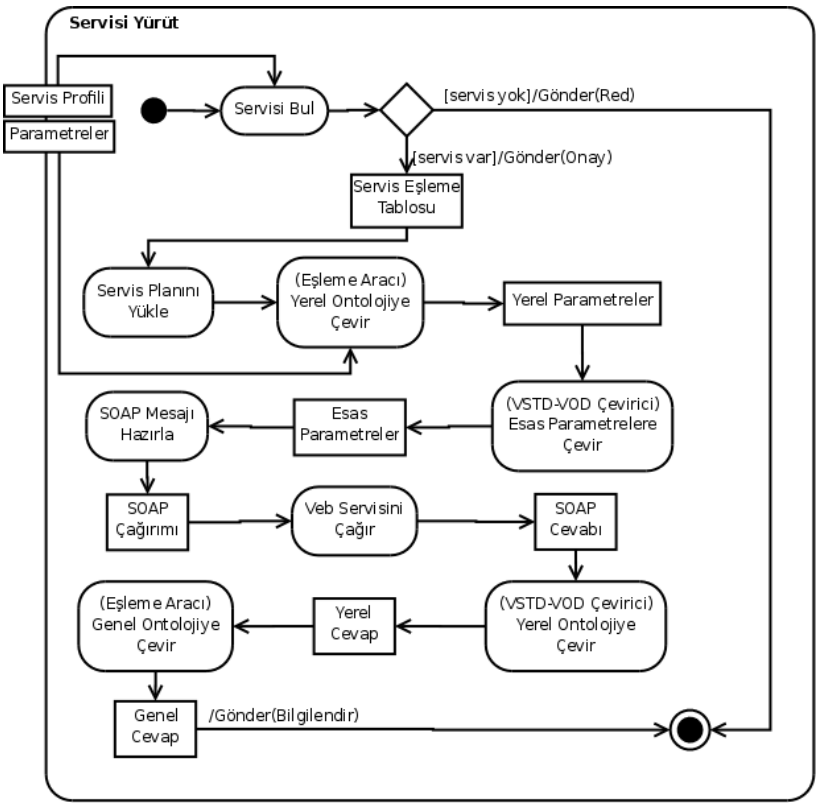
SSE ve SİE anlamsal servisin ölçütleri üzerinde uzlaştıktan sonra, SİE yürütme sürecini başlatır (Şekil 3.8). SİE uygun parametreler ve istediği servisin profilini kullanarak bir istek oluşturur. Sonra bu isteği SSE'ye yollayarak istekte bulunur ve varsa İzleme Servisi'ne süreci izlemesi için haber verir. İsteği alan SSE, ilk önce Eşleşme Aracı'nı kullanarak bu çalıştırılması istenen servisin olup olmadığına bakar. Uygun servisin olması durumunda SİE'ye bir kabul mesajı gönderir ve bu yerel servisi çalıştıran planı yükler. Bu plan yukarıda da bahsedildiği gibi servisin çalıştırılmasını sağlayan plandır ve genel bir servis çalıştırma planından türetilmiştir. Bu aşamadan sonra SSE, öncelikle platform ontolojisinde gelen parametreleri çevirici araçlar yardımıyla önce yerel ontolojiye, daha sonra da esas pa-



Şekil 3.9. SİE Servis Yürütme Etkinlik Diyagramı

rametrelere (VSTD parametrelerine) dönüştürür. Bu dönüştürüm işlemleri sırasında kayıt safhasında oluşturulan eşleşme ve eşleme bilgileri kullanılır. Parametreler VSTD'ye uyumlu hale geldikten sonra bir servis bu parametrelerle çağırılır. Bu çağırım sonucu gelen cevap mesajı ilk önce yerel ontolojiye çevrilir. Daha sonra da global ontolojiye çevirilir ve SİE'ye yollanır.

Şekil 3.9 ve Şekil 3.10'de SİE ve SSE'nin yürütme sürecine yönelik etkinlik diyagramları verilmiştir.



Şekil 3.10. SSE Servis Yürütme Etkinlik Diyagramı

4 ETMEN TABANLI ANLAMSAL SERVİS MİMARİSİNİN PLANLAMA GEREKSİNİMLERİ

Bölüm 4'te bahsedilen Etmen Tabanlı Anlamsal Servis Mimarisi AVSM süreçlerinin gerçekleştirimi için gerekli altyapı gereksinimlerini ve mimari elemanları içermektedir. Böyle bir ortam, geliştiricilerin anlamsal veb servis çalıştırma sürecini basitleştirir. Bu ortamdaki Servis İstemci Etmen(ler) anlamsal servis keşfi, uzlaşması ve çalıştırılmasında kullanılacak planlar sağlamalıdır. Benzer şekilde Servis Sağlayıcı Etmen(ler) de, anlamsal servislerin ilanında (yayınlanması) ve çalıştırılmasında kullanılacak planları sağlamalıdır. Buna ek olarak her iki etmenin planları da platformun ilgi alanına göre özelleşebilmelidirler. Bu sebepten dolayı, bu planlar yeniden kullanılabilir şablon planlar olarak modellendi. Daha sonra bu planların modellenmesi esnasında ortaya çıkan planlama ihtiyaçlarına yönelik bir planlayıcı tasarlandı.

Bu bölümün içerisinde öncelikle SEAGENT çerçevesi için geliştirilen planlayıcı tanıtılacaktır. Daha sonra Bölüm 3'de anlatılan süreçlerin planlarının bu planlayıcı bağlamında modellenmesiyle planlayıcının sahip olması gereken ek gereksinimler saptanacaktır. Son olarak da geliştirilen planlayıcının bu ihtiyaçlara nasıl ve ne kadar cevap verdiği tartışılacaktır.

4.1 SEAGENT Planlayıcı

Bilgi-tabanlı ve elle-ayarlanabilir planlama sistemleri gerçek dünya planlama problemlerini en iyi çözecek planlama sistemleri olarak görüldüğünden[1], SEAGENT adındaki ÇES geliştirme çerçevemizin planlayıcısı, Hiyerarşik Görev Ağı (HGA) planlama formalizmine[2] dayanmaktadır. HGA planlama, planları görev ayrıştırımıyla yaratan bir YZ planlama metodolojisidir. Bu ayrıştırma süreci, planlama sistemi doğrudan çalıştırılabilen ilkel görevler buluncaya kadar devam eder. HGA planlamanın temel fikri 70'lerin ortalarında Sacerdoti(Sacerdoti, 1975) ve Teyt(Tate, 1977) tarafından ortaya atıldı. Biçimsel temelleri ise 90'ların ortalarında Erol(Erol et al., 1996b) tarafından geliştirildi. Bir HGA planlama sisteminde, hedef parçalı-sıralı olarak verilen bir görev kümesini (plan) başarmaktır. Bir plan eğer çalıştırılabilir ve verilen görevleri başarır ise doğrudur. Yani, bir HGA planlayıcının ana odağı, geleneksel planlayıcıların istenilen bir duruma ulaşmaya odaklanmasının aksine, görevleri gerçekleştirmektir.

Planlar ileriki davranışların gösterimleridirler, bu yüzden, geliştirilen platformda etmenlerin bütün davranışları (planlamanın kendisi bile) plandır. Böyle bir mekanizmayı sağlayabilmek için tek görevi çalıştırılabilir elemanları çalıştırmak ve zamanlamak olan bir görev çalıştırma birimi (*task execution unit*) kullanıldı. İlk olarak, bir planlama birimi iki temel mekanizmayı sağlamalıdır: sevk (*dispatching*) ve hedef eşleştirme (*goal matching*). *Sevk* ile kastedilen giden mesajların yollanması, gelen mesajların alınması ve gerektiğinde bu mesajlardan hedefler yaratılmasıdır. He-

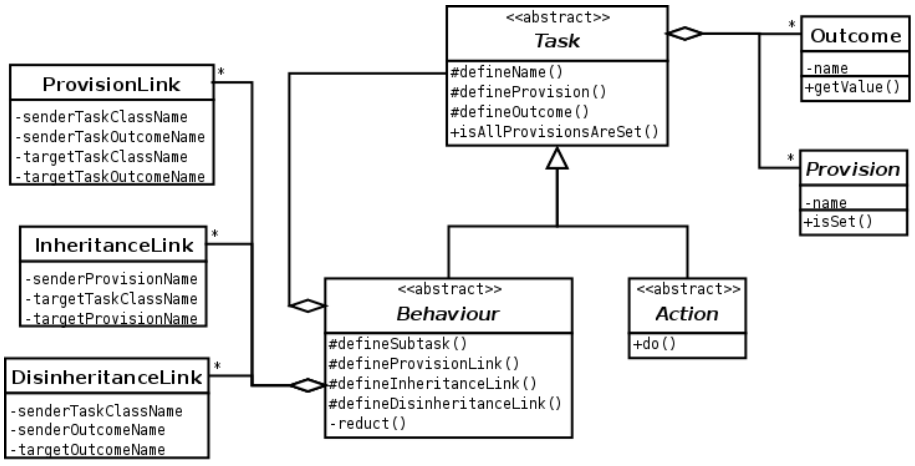
[1] Planlama ile ilgili daha geniş bilgi Bölüm 2.5'de anlatılmıştır.

[2] HGA planlama formalizmi Bölüm 2.6'de detaylı olarak anlatılmıştır.

defler gelen bir istek üzerine oluşturulabileceği gibi içsel olarak da oluşturulabilirler. *Hedef eşleştirme* ile kastedilen ise hedeflerin uygun planlara eşlenmesidir.

4.1.1 Plan Yapısı

Geliştirilen çerçeve Saykara ve ark. (Sycara et al., 1996) tarafından sunulan çerçeveye ve DECAF mimarisine (Graham et al., 2003) benzemektedir. HGA'nın bir gereksinimi olarak, görevler karmaşık görevler (davranış - *behaviour*) ve ilkel görevler (eylem - *action*) olmak üzere iki tiptedir. Her plan, öntanımlı bir görevi başaran altgörevlerden oluşan karmaşık bir kök görevden oluşmaktadır. Davranışlar, karmaşık görevin altgörevlerine ayrışımını ve görevler arası (altgörevler arası ve altgörevler ile ebeveyn görev arası) bilgi akışını tanımlayan bir "indirgeme şeması" (*reduction schema*) tutmaktadır. Bilgi akış mekanizması şu şekilde olmaktadır: her görev bilgi ihtiyacını bir ihtiyaçlar (*provisions*) kümesi ile gösterir ve görevin işletimi sonuçları (*outcomes*) üretir, ve ihtiyaç ve sonuç yuvalarını kullanarak görevler arası bilgi akışını gösteren bağlantılar vardır. Bu bağlantılar sonuçları ihtiyaçlara bağlayan ihtiyaç bağlantısı (*provision link*), ebeveyn görevdeki ihtiyacı altgörevine bağlayan miras bağlantısı (*inheritance link*) ve altgörevdeki sonucu ebeveyn göreve bağlayan ters miras bağlantısı (*disinheritance link*) olmak üzere üç adettir. Eylemler, planlayıcı tarafından doğrudan çalıştırılabilen ilkel görevlerdir. Ayrıca, her görev çalıştıktan sonra bir sonuç durumu (*outcome state*) üretir. Sonuç durumları görevler arası bilgi akışını yönlendirmek için kullanılır. Varsayılan sonuç durumu "OK"dir ve diyagramlarda genellikle açıkça gösterilmez. Ancak



Şekil 4.1. SEAGENT plan yapısının bileşenleri

farklı sonuç durumları oluşacak olursa bunlar bağlantıların üzerine durumu oluşturan göreve yakın bir şekilde yazılarak gösterilirler.

Plan yapısının bileşenleri Şekil 4.1'de gösterilmiştir. Görevler soyut `Task` sınıfı ile temsil edilmektedirler. Görevlerin, yaptıları işi tanımlayan isimleri vardır ve bunlar emir kipinde ifade edilmektedir (Örneğin "Sorgu Gönder"). Bir görevin ismi `defineName()` metodu içinde tanımlanmaktadır. Görevlerin sıfır veya daha çok ihtiyacı (bilgi ihtiyacı) veya sonucu (çalışma ürünü) olabilir. Bu bilgilerin görev içindeki gösterimine ihtiyaç yuvası ve sonuç yuvası denmektedir. İhtiyaç ve sonuç yuvaları sırasıyla `defineProvision()` ve `defineOutcome()` metodlarıyla tanımlanmaktadırlar. Görevler, bütün ihtiyaçları sağlandığında (hepsi için bir değer olduğunda) *hazırdırlar*. Hazır bir görev çalıştırılmak için uygundur ve bu duruma geçtiği anda görev çalıştırıcı tarafından çalışıtırılmaktadır. Görevlerin hazır olup olmadığı `isAllProvisionsAreSet()` metodu ile öğrenilir. Bu metod, görev içindeki bütün ihtiyaçların hazır olup

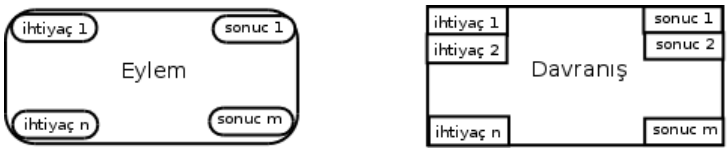
olmadığını onların `isSet()` metodlarını çağırarak belirler. Hiç ihtiyacı olmayan bir görev her zaman hazırdır, dolayısı ile görev ağacı açılır açılmaz çalıştırılabilirler.

Sözü edilen HGA çerçevelerinin aksine (Sycara et al., 1996; Graham et al., 2003), bu planlayıcı plan yapısı dışından gelen bilgileri ifade eden harici ihtiyaçları desteklememektedir. Çünkü, dış iletişimi kontrol eden *Sevk planı* (ilerleyen bölümlerde anlatılıyor) dışarıdan gelen bilgileri bilgitabanına eklemekte, o bilgiye ihtiyacı olan görev de onu bilgitabanından almaktadır. Bu sebepten dolayı bütün ihtiyaç değerleri planın iç işleyişinde atanan dahili ihtiyaçlardır. Yani, bir ihtiyacın değeri bir başka görevin sonucuyla belirlenmektedir.

Bir görev çalışması esnasında varsa ihtiyaçlarını tüketir ve çalışma sonunda sonuçlar üretir. Görevler geliştiriciler tarafından periyodik (döngüsel) olarak düzenlenmiş olabilirler. Periyodik bir görev kendisine verilen periyoda göre önceki çalışmasını takiben yeniden çalıştırılmak üzere zamanlanır.

Eylemler soyut `Action` sınıfı ile ifade edilmektedir. Bir eylem tanımlanması için `Action` sınıfı genişletilmeli ve `Do()` metodu içine eylem çalıştığında yapılması istenen kod yazılmalıdır.

Davranışları ise soyut `Behaviour` sınıfı ifade etmektedir. Davranışın altgörevlerini ve altgörevleri arası bilgi akışını tutan indirgeme şeması yine bu sınıf içinde tutulmaktadır. Dolayısı ile `Behaviour` sınıfında altgörevleri ve bilgi akışlarını (miras, ihtiyaç ve ters miras bağlantıları) tanımlamaya yönelik metodlar bulunmaktadır. Bunlar sırasıyla `defineSubtask()`, `defineInheritanceLink()`,



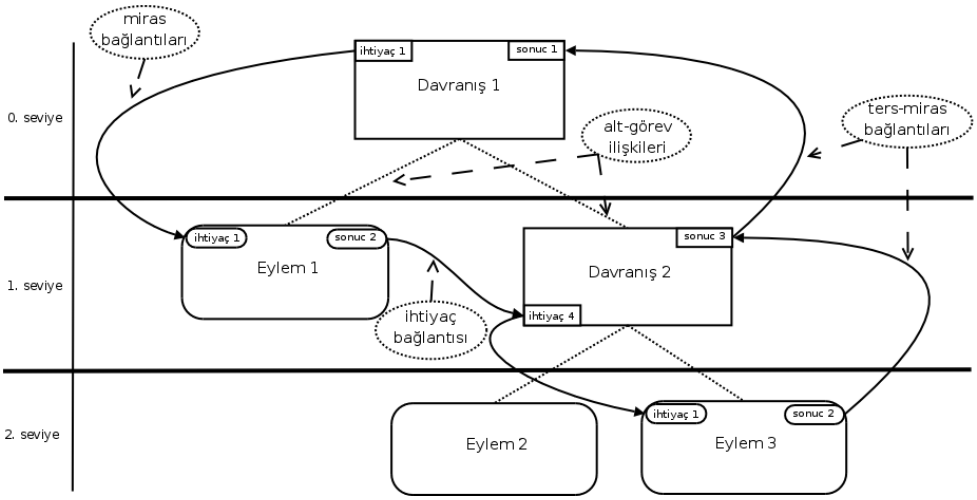
Şekil 4.2. Örnek SEAGENT HGA görevleri

`defineProvisionLink()` ve `defineDisinheritanceLink()` metodlarıdır.

Görev sınıflarının isimlendirilmesi şu şekilde olmaktadır: `Behaviour` sınıfından türeyen davranış sınıfları "BH<davranış ismi>" ve `Action` sınıflarından türeyen eylem sınıfları ise "AC<eylem ismi>" olarak adlandırılır. Örnek olarak "Oda Kiralama" davranışının sınıfı `BHOdaKiralama` olarak adlandırılırken "Oda Kiralama" eyleminin sınıfı ise `ACOdaKiralama` olarak adlandırılır.

HGA görevlerinin şekilsel gösterimi Şekil 4.2'de gösterilmiştir. Görevler ortalarında isimleri yazacak şekilde dörtgenlerle gösterilmektedir. Davranış tipindeki görevler keskin köşeli dikdörtgenlerle gösterilmekte iken eylemler ise yuvarlak köşeli dörtgenlerle gösterilmektedir. İhtiyaçlar görevlerin sol tarafında, sonuçlar ise sağ tarafında gösterilmektedir.

Şekil 4.3'de örnek bir HGA planının görev ağacı gösterilmektedir. "Davranış 1" planın kök görevidir ve kök görevin bulunduğu seviyeye 0. seviye denmektedir. Kök görevin bir seviye altında (1. seviye) "Eylem 1" ve "Davranış 2" bulunmaktadır. Bu görevler "Davranış 1" görevinin altgörevidirler. "Davranış 2" görevinin de altgörevleri olduğu için ağaç bir seviye daha açılabilmektedir. 2. seviyede de "Davranış 2" görevinin altgörevleri olan "Eylem 2" ve "Eylem 3" bulunmaktadır. Bilgi akışları ya aynı

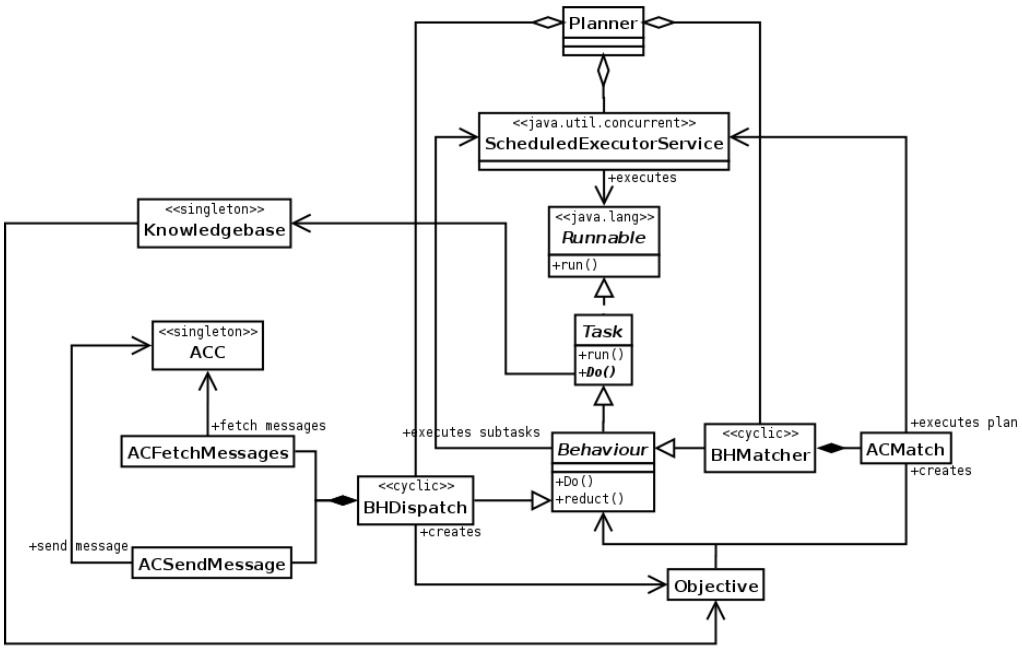


Şekil 4.3. Örnek SEAGENT HGA planı

seviyede olmaktadır (ihtiyaç bağlantısı) ya da komşu iki seviye arasında olmaktadır (miras ve ters miras bağlantıları). İhtiyaç bağlantıları aynı seviyedeki en az iki görevden birisinin sonucunun diğer görevlerine ihtiyacını sağlamasını ifade etmektedir. Miras bağlantısı bir davranışın bir doğrudan altgörevine (1 seviye altındaki altgörevi) bilgi aktarmasıdır. Ters miras bağlantısı ise görevin doğrudan ebeveynine (1 seviye üstündeki ebeveyn) bilgi aktarmasıdır. Şekil 4.3'deki "Davranış 1" görevine ait kod Ek A'da ve "Eylem 1" eylemine ait kod da Ek B'de gösterilmekte ve açıklanmaktadır.

4.1.2 İç Mimari

Planlayıcı, gelen giden mesajları işlemek, hedef eşlemek, HGA görevlerini zamanlamak ve çalıştırmaktan sorumludur. Bir planın çalıştırılması için, karmaşık kök görev "indirgeme şeması" kullanılarak açılır ve bu indirgeme eylemler bulununcaya kadar devam eder. Sonra, çalıştırma birimi bu



Şekil 4.4. SEAGENT Planlayıcısının Sınıf Diyagramı

eylemleri çalıştırır. Planlayıcı temel olarak şu işlevsel modüllerden oluşmaktadır: bir çalıştırma birimi, periyodik bir sevk planı ve periyodik bir hedef eşleme planı. Tasarlanan planlayıcının sınıif mimarisi Şekil 4.4'de gösterilmiştir.

Planlayıcının işleyiş mekanizması şu şekildedir: Yeni ZFEK-EİD mesajları *Sevk planı* (`BHDispacth`) tarafından Etmen İletişim Kanalı'ndan (`ACC` - *Agent Communication Channel*) alınır. Sonra *Sevk planı* bu mesajları çözümler ve onların süregelen bir iletişimin parçası olup olmadığına bakar. Eğer öyleyse, mesajı bilgi-tabanına (`Knowledgebase`) gelen bir mesaj olarak koyar. Aksi takdirde, mesajın süregelen bir iletişimin bir parçası olmadığı durumda, yeni bir hedef yaratır ve bunu

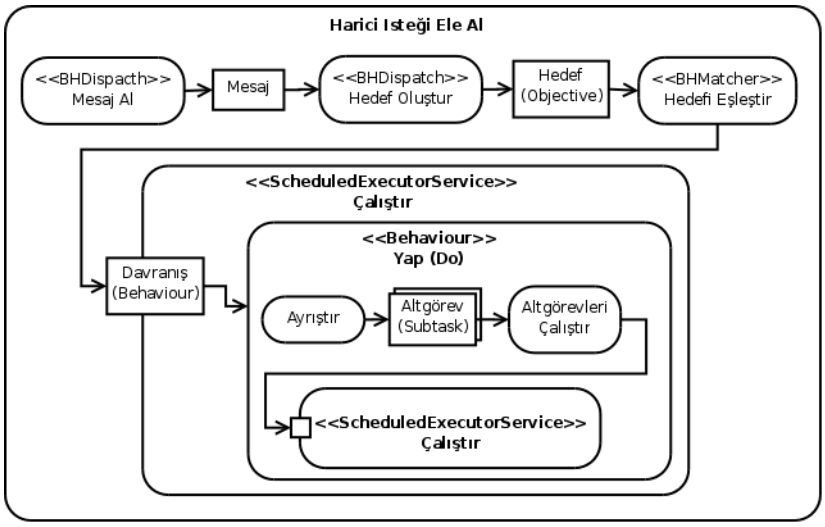
bilgi-tabanına yeni bir hedef bilgisi olarak koyar. *Hedef eşleme planı* (`BHMatcher`) döngüsel olarak bilgi-tabanını yeni bir hedef olup olmadığı yönünde kontrol eder. Bir hedef bulduğunda, *Hedef Eşleme planı* bilgi-tabanındaki plan kütüphanesini sorgulayarak hedefi öntanımlı bir planla eşleştirmeye çalışır. Eğer bu eşleştirme süreci başarılı olursa, bu plan eşleşen planın bir örneğini yaratır ve onu çalıştırılması için çalıştırma birimine verir. Bütün bunlardan sonra, bu yeni planın nasıl yürütüleceği planın kendisi tarafından ele alınır.

Çalıştırma birimi, verilen çalıştırılabilir görevin çalıştırılması ve zamanlanmasından sorumludur. Çalıştırma biriminin gerçekleştirimi için Java 1.5[3] ile gelen `java.util.concurrent` paketindeki `ScheduledExecutorService` arayüzü üzerinden tek işparçacıklı bir çalıştırma birimi kullanılmıştır. `Task` sınıfı da `Runnable`[4] arayüzününü genişleten `Executable` arayüzünü gerçekleştirdiği için çalıştırma birimi tarafından işparçacağı gibi çalıştırılır. Bu sınıf komutları anında veya verilen belirli bir süre geçtikten sonra ya da belirli periyotlarla çalıştırabilmektedir. Görevlerin `execute()` metodunda `Do()` metodu çağırılır. Davranışların `Do()` metodunda `reduct()`[5] metodu çağırılıp altgörevlere ayrıştırma yapılır ve hepsi çalıştırma birimine verilir. Eylemlerin `Do()` metodunda yukarıda da anlatıldığı gibi eylemin kodu çalışır. Eğer bir görev *hazır* ise (bütün ihtiyaçları sağlanmış ise) ve çalışma zamanı gelmişse, çalıştırıcı `execute()` metodunu çağırarak onu çalıştırır.

[3] http://java.sun.com/j2se/1.5.0/, son erişim 27 Haziran 2007.

[4] `java.util.concurrent.Runnable` arayüzü Java'da işparçacağı oluşturmak için kullanılır.

[5] `reduct()` metodu Ek C'de gösterilmekte ve açıklanmaktadır.

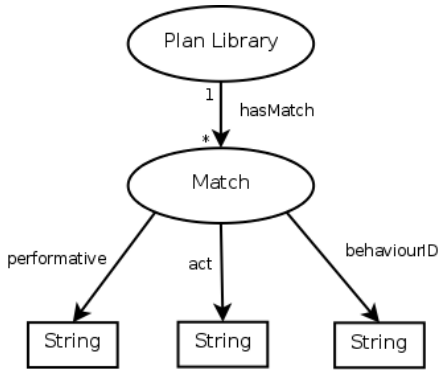


Şekil 4.5. Harici İstek Ele Alınması Etkinlik Diyagramı

Şekil 4.5'de yukarıda anlatılan modüllerin harici bir isteğin ele alınmasını anlatan bir etkinlik diyagramı verilmiştir.

Sevk planı gelen giden mesajlardan ve etmenler için hedef yaratmaktan sorumlu döngüsel bir plandır. Periyodik olarak EİK'yi gelen mesajlar için kontrol eder ve eğer varsa onları getirir (bilgi-tabanına koyarak). Ayrıca periyodik olarak bilgi-tabanını gidecek mesajlar için kontrol eder. Bunun sebebi bir etmenin mesaj atmak istediğinde onu bilgi-tabanına koymasıdır.

Hedef Eşleme planı gelen hedefi, plan ontolojisi ve eşleme ontolojisi kullanılarak yapılandırılmış plan kütüphanesini sorgulayarak öntanımlı bir plana eşlemekle sorumludur. Bu plan periyodik olarak bilgi-tabanını yeni hedef oluşup oluşmadığına dair kontrol etmektedir. Yeni bir hedef olduğunda, plan kütüphanesinden bu hedefe uygun bir plan eşlemeye çalışmaktadır. Eğer eşleme başarılı olursa, Hedef Eşleme planı eşlenen planın

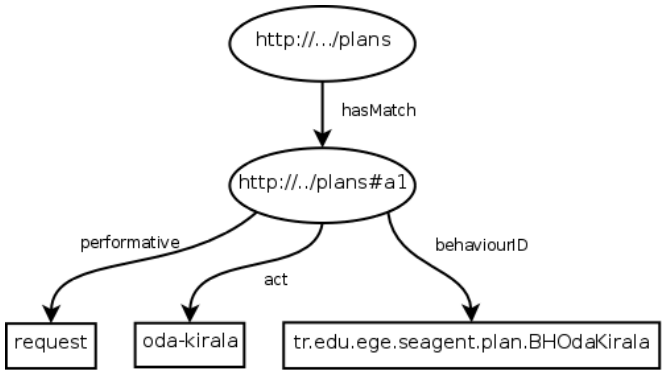


Şekil 4.6. Eşleme ontolojisi

bir örneğini yaratmakta ve onu çalıştırma birimine çalıştırılmak üzere vermektedir.

Eşleme ontoloji hedefleri planlarla eşlemek için kullanılmaktadır ve Şekil 4.6'de gösterilmektedir. Bu ontolojide bir adet plan kütüphanesinde (*Plan Library*) birçok eşleme (*Match*) bulunabilmektedir. Her eşlemede dışarıdan gelen mesajın iletişim edimi (*performative*), mesajda istenen eylem (*act*) ve bunlara uygun olan planın kök görevi (*behaviourID*) bilgisi tutulmaktadır. Şekil 4.7'de örnek bir hedef eşlemesi gösterilmektedir. Bu örneğe göre eğer etmene "oda-kirala" diye bir istek (*request*) gelirse buna karşılık `BHOdaKirala` planı çalıştırılmalıdır. Bu örnek hedef eşleme ontolojisinin kod örneği Ek D'de gösterilmektedir.

Hedefin eşlenmesi aşamasından sonra, eşlenen planın çalışma sorumluluğu planın kök görevine aittir. Çünkü bir davranış bütün altgörevleri gerçekleştirildiğinde başarılmış demektir. Planın yaşam döngüsü şöyledir: Önce kendini altgörevlerine ayrıştırır ve bunları çalıştırma birimine verir. Sonra hepsi çalıştırılıncaya kadar onları kontrol eder.



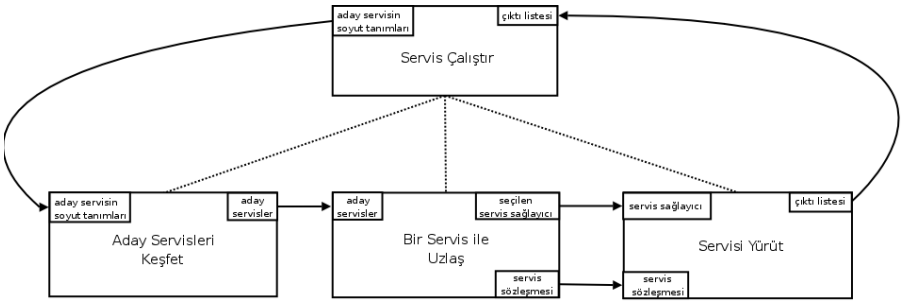
Şekil 4.7. Eşleme ontolojisinde örnek bir hedef eşlemesi

4.2 Anlamsal Servis Mimarisine Yönelik Etmen Planları

Bu bölümde Bölüm 3'te anlatılan Etmen Tabanlı bir Anlamsal Servis Mimarisi'ndeki etmenlerin planlarının nasıl olması gerektiği ve bu planları işletebilmek için etmen planlayıcısının ne gibi gereksinimlere sahip olması gerektiği anlatılacaktır. Planlayıcı mimarinin kalbindedir ve AVSM süreçlerinin iş akışlarını kontrol etmektedir. AVSM süreçlerini temel olarak SİE ve SSE gerçekleştirmektedir. Süreçlerin herbiri yeniden kullanılabilir planlar olarak modellenmiştir. Bu planlar hazırlanırken, Bölüm 3'de ortaya konan etkinlik diyagramlarından faydalanılmıştır. Etkinlik diyagramlarındaki etkinlikler zaman zaman bir ilkel görev olarak zaman zaman ise bir karmaşık görev olarak modellenmiştir.

4.2.1 Servis İstemci Etmen Planları

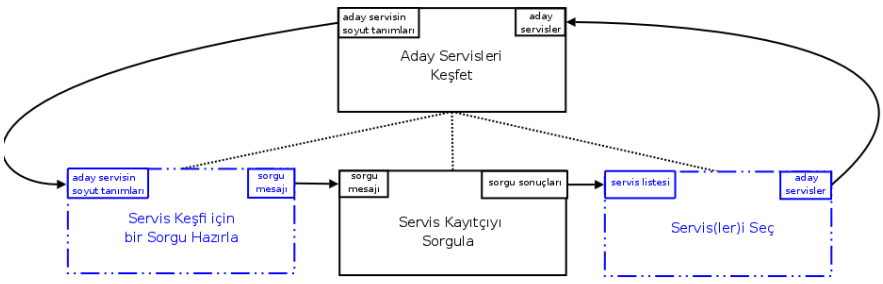
Servis İstemci Etmen (SİE), platformda sunulan servisleri çalıştırmak için istemde bulunan etmendir. SİE, servis bulup çalıştırmak için AVSM'de ta-



Şekil 4.8. Servis İstemci Etmen için AVSM tabanlı Anlamsal Servis Çalıştırma HGA Planı

nımlanmış olan süreçleri işletmektedir. AVSM'ye göre SİE çalıştırmak istediği servisi önce keşfeder, sonra onun sağlayıcısı ile uzlasır ve son olarak da onu yürütür. Bu süreçler birbirini takip eden kesintisiz süreçlerdir. Dolayısı ile SİE AVSM tabanlı servis çalıştırma işini yeniden kullanılabilir tek bir plan şeklinde kullanabilir.

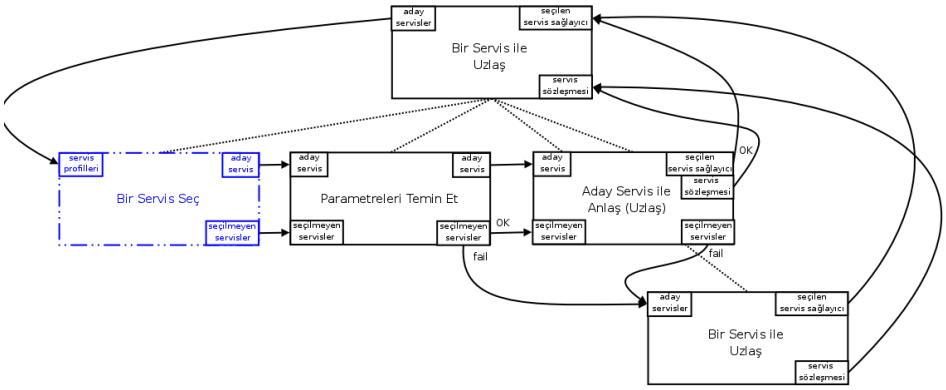
Şekil 4.8'te, SİE için servis çalıştırma planının HGA görev ağacının tek seviye açılmış hali gösterilmiştir. "Servisi Çalıştır" görevi servis AVSM'de önerilen çalıştırma sürecini gösterir ve planın kök görevidir. Bu görev işletilebilmek için aday servislerin soyut bilgilerine ihtiyaç duyar. Bu görev "Aday Servisleri Keşfet", "Bir Servis ile Uzlaş" ve "Servisi Çalıştır" isimli üç altgörevden oluşmaktadır. "Aday Servisleri Keşfet" görevi, aday servislerin soyut bilgilerini ebeveyn görevinden miras alır ve çalışması sonucunda aday servislerin çıktı olarak profillerini oluşturur. Sonra bu servis profilleri "Bir Servis ile Uzlaş" görevine ihtiyaç bağlantısı vasıtasıyla aktarılır ve bu görevin çalışması başlar. "Bir Servis ile Uzlaş" görevi iki çıktı üreterek sonlanır: seçilen servis sağlayıcı (*selected service provider*) ve servis sözleşmesi (*service agreement*). Bu çıktılar "Servisi Çalıştır"



Şekil 4.9. "Aday Servis Keşfi" görevinin ayrıştırılmış görünümü

görevi tarafından planın son aşamasını gerçekleştirmek için kullanılır. Bu son görev, seçilen servisi çalıştırır ve bu çalıştırmanın sonucunu ters miras bağlantısıyla ebeveyn görevine geçirir.

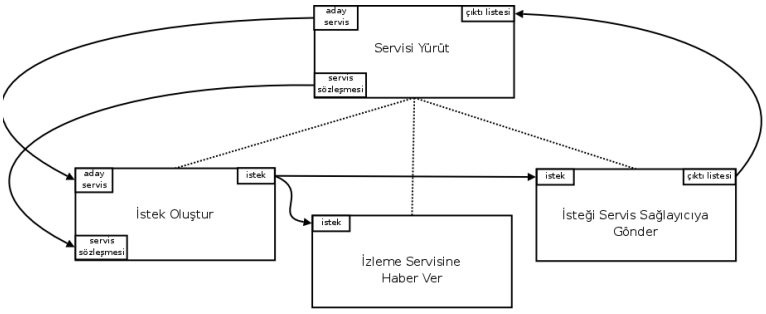
Servis keşif sürecinde, SİE, ilk önce gereksindiği servisin profilini kullanarak bir sorgu hazırlar ve onunla servis kayıtçıyı sorgular, sonra anlamsal uygunluktaki servislerin kayıt profillerini ve son olarak uygun profilleri döndürmek için bunlar üzerinde bir servis seçme politikası uygular (Şekil 3.6). Bu süreç içinde iki aşama soyut özellik göstermektedir. Servis keşfi için olan sorgu, servisi tipine göre hazırlanabileceği gibi servisin girdi/çıktı parametrelerinin tipine göre de hazırlanabilir. Benzer şekilde servis seçimi sırasında yerine göre farklı politikalar uygulanabilir. Dolayısı ile bu sürece yönelik bir planda bu iki aşama *soyut görev yapıları olarak tanımlanabilmeli ve bu soyut yapılar kullanılarak şablon planlar tasarlanabilmelidir. Daha sonra da bu şablonların soyut görevleri özelleştirilerek etmenlerin gerçek planlarının türetilebilmelidir.* Şekil 4.9'de üç altgörevden oluşan "Aday Servis Keşfi" karmaşık görevi gösterilmektedir. Altgörevleri sırasıyla "Servis Keşfi için bir Sorgu Hazırla", "Servis Kayıtçıyı Sorgula" ve "Servis(ler)i Seç" görevleridir. Kenar çizgileri kesikli olarak



Şekil 4.10. ”Bir Servis ile Uzlaş” görevinin ayrıştırılmış görünümü.

gösterilen görevler soyut görevleri, kesiksiz düz olanlar ise somut görevleri ifade etmektedir.

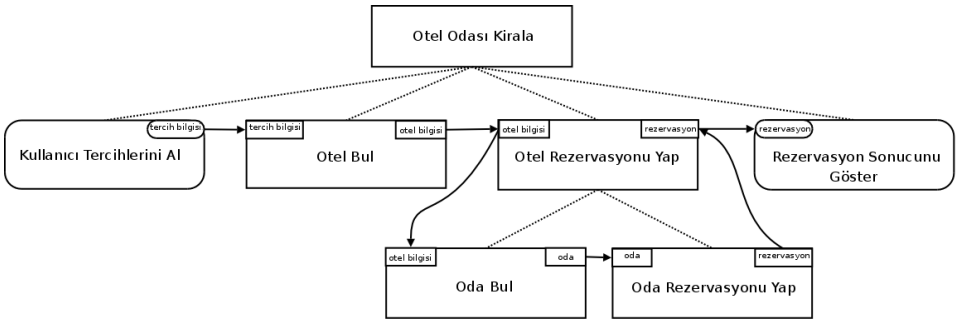
Uzlaşma sürecinde SİE'nin işleyişi Şekil 3.7'deki etkinlik diyagramında anlatılmıştı. Buna göre SİE ilk önce aday servislerden birisini seçer ve sonra bütün servis çağırım parametrelerinin tamlığı garantilemeye çalışır. Garantileme başarılı olursa, SSE ile Servis Kalitesi ölçütleri üzerine bir görüşme yapılır ve anlaşma kararı verilir. Eğer garantileme veya görüşme görevleri başarısız olursa, uzlaşma süreci bir önceki aşamada seçilmeyen servisler için yeniden başlatılır. Sürecin son aşaması, görüldüğü gibi özyineli bir yapı sergilemektedir. Uzlaşma süreci özyineli bir şekilde devam etmektedir ta ki adaylardan birisiyle uzlaşma sağlanana kadar. Bu durumu HGA planlaması açısından ele alacak olursak *özyineleme ile kastedilen bir görevin örneğinin görev ağında aynı görevin bir diğer örneğinin atası olmasıdır. Bunu sağlayabilmek için plan yapısının (planlayıcının) özyineleme desteği olması gerekmektedir. Yani bir görevin kendisini kendi altgörevi olarak barındırabilmesini gerekmektedir.* Şekil 4.10'de bu süreç



Şekil 4.11. ”Servisi Yürüt” görevinin ayrıştırılmış görünümü.

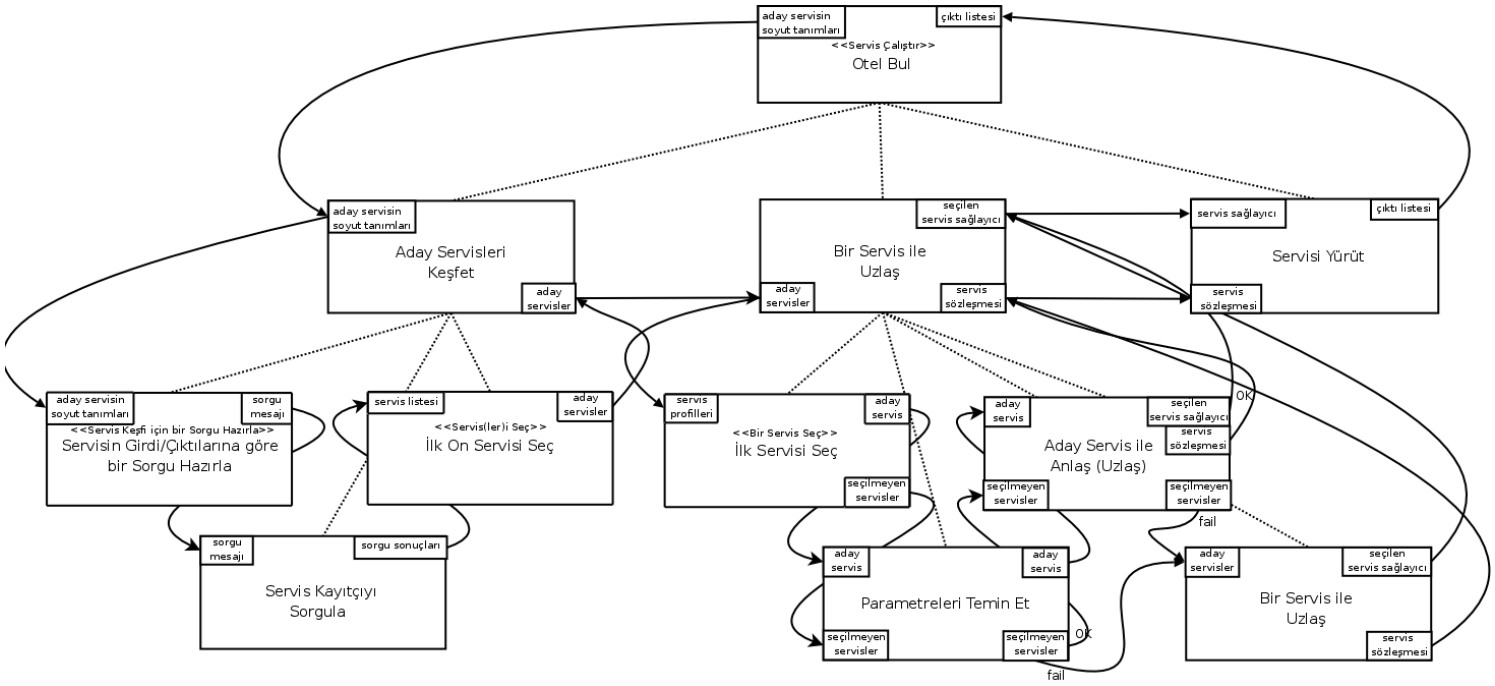
için tasarlanmış olan “Bir Servis ile Uzlaş” görevi tek seviyeli olarak gösterilmektedir. Bu görev servislerden birini seçildiği “Bir Servis Seç”, seçilen servisin parametrelerinin temin edilmeye çalışıldığı “Parametreleri Temin Et”, parametreleri temin edilen servisin sağlayıcısı ile uzlaşılmaya çalışılan “Aday Servis ile Uzlaş” ve servis ile uzlaşılamazsa seçilmeyen servislerde yeniden aynı görevi işleten “Bir Servis ile Uzlaş” altgörevlerinden oluşmaktadır. Servis seçme politikası burada da değişebileceği için “Bir Servis Seç” görevi soyuttur. Ayrıca görüldüğü üzere “Bir Servis ile Uzlaş” görevi kendisini altgörev olarak içermektedir, yani özyineli bir görevdir.

Yürütme süreci diğer süreçlere nazaran daha basit adımlardan oluşmaktadır. Bu süreçte (Şekil 3.9) uzlaşma sürecinden gelen aday servis ve servis sözleşmesi kullanılırak bir istek hazırlanır ve SSE’ye bu istek gönderilerek servisin yürütülmesi istenir. Aynı zamanda izleme servisine de haber verilerek süreci takip etmesi istenir. SSE isteği yerine getirdikten sonra sonuçları SİE’ye gönderir ve “Servisi Yürütme” süreci, dolayısıyla da “Servis Çalıştırma” süreci tamamen bitmiş olur. Şekil 4.11’de bu sürece yönelik tasarlanan yürütme planı gösterilmiştir.



Şekil 4.12. ”Otel Odası Kirala” planı.

Yukarıda altgörevleri de açıklanan “Servis Çalıştır” planı (Şekil 4.8) yeniden kullanılabilir şablon bir plandır. Planın içerisindeki soyut görevler somut görevlerle bağlanarak servise-özel çalıştırma planları hazırlanabilir. Örneğin bir otel odası kiralama senaryosunu ele alalım. Kullanıcı isteklerine uyan bir otel odası kiralanabilmesi için öncelikle kullanıcıdan tercihleri alınır ve bu tercihlere uyan bir otel bulunur. Sonra o otelden bir oda kiralanmaya çalışılır ve son olarak sonuçlar kullanıcıya gösterilir. *Otel bulma*, *oda bulma* ve *oda rezervasyonu yapma* birer anlamsal servis olarak platforma sunulmuş olabilir. Dolayısı ile bu servislerin kullanılması için “Servis Çalıştır” planını özelleştirebilir. Hatta *oda bulma* ve *oda rezervasyonu* yapma servislerini birleştirip *otel odası rezerve etme* adı altında tek bir servismiş gibi planlayabilir. *Yani birleşik servisleri (servisleri birleştirerek kullanmayı) anlamsal servis planlarını kullanarak oluşturabiliriz.* Şekil 4.12’de sözü edilen oda kiralama planı gösterilmiştir. Bu planda, “Otel Bul”, “Oda Bul” ve “Oda Rezervasyonu Yap” altgörevleri “Servis Çalıştır” görevinin somut gerçektirimleridir. İhtiyaç ve çıktı yuvaları ile



Şekil 4.13. ”Otel Bul” planının HGA ağacı.

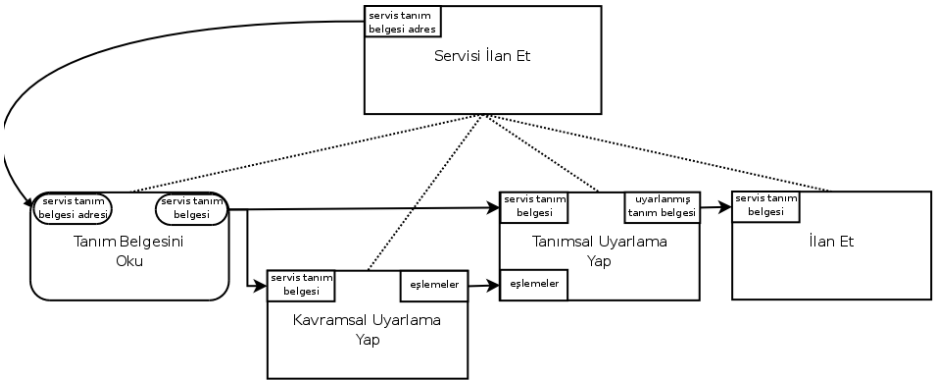
bağlanmışlardır ve alan bağımlı planlar oldukları için hangi girdileri alacaklarını bilmektedirler.

Şekil 4.12'deki planın altgörevlerinden birisi olan "Oda Bul" görevinin nasıl somutlaştırıldığını gösteren HGA ağacı Şekil 4.13'de verilmiştir. Görevlerin hangi görevlerden türediği "<<>>" içerisinde yazılı olarak gösterilmektedir. Örneğin "Servis Çalıştır" görevinden türeyen "Oda Bul" görevinin üzerinde <<Servis Çalıştır>> yazmaktadır. "Oda Bul" görevi "Servis Çalıştır" görevinin somut bir gerçekleştirimidir. Dolayısı ile onun altgörevlerindeki soyut görevlerin de somut görevlerle gerçekleştirilmesi gerekmektedir. Şekilde gösterilen gerçekleştirimlerden de görülebileceği gibi bu örnekte servis keşfi için hazırlanacak olan sorgu girdi/çıktı parametrelerine göre hazırlanacak ve keşif sonunda gelen servislerden ilk on tanesi seçilerek bunlarla uzlaşılmaya çalışılacaktır. Uzlaşma aşamasında da her seferinde aday servislerden ilkiyle anlaşmaya varılmaya çalışılacaktır. Servis yürütme aşamasında soyut görev gerçekleştirimi olmadığı için bir alt seviyesi gösterilmemiştir.

4.2.2 Servis Sağlayıcı Etmen Planları

Servis Sağlayıcı Etmen (SSE), harici veb servislerini ve anlamsal veb servislerini ÇES'e dahil eder ve etmen-servis etkileşimini sağlar. SSE bu işlevleri iki temel süreç halinde gerçekleştirir. Bunlardan birincisi harici servislerin etmen platformuna eklendiği *İlan Süreci*, diğeri de servisin yürütüldüğü *Yürütme Süreci*'dir.

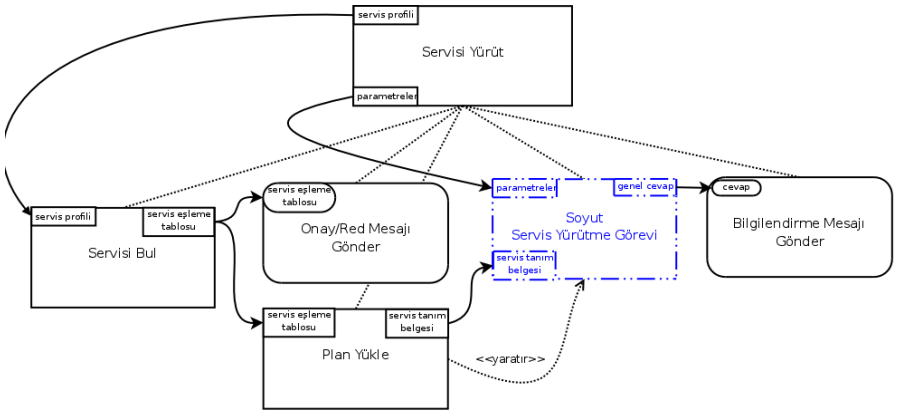
İlan süreci, temel olarak harici veb servisinin etmen platformuna



Şekil 4.14. SSE için AVSM tabanlı Anlamsal Servis İlan Etme HGA Planı

uyarlanması ve platformda ilan edilmesini safhalarını kapsar. Servisin platforma uyarlanmasından kasıt hem verisel açıdan hem de süreç açısından etmenin servise aracılık yapmasının yönteminin tanımlanmasıdır. Etmenin verisel açıdan aracılık yapılabilmesi için servisin kullandığı kavramların veya ontolojinin platform kavramlarıyla olan ilişkisinin tanımlanması gerekmektedir ve ayrıca servis tanımının da bu bağlamda platformun servis tanımına uyumlu olması gerekmektedir. Etmenin süreç açısından aracılık yapabilmesi için ise etmenler ve veb servislerinin işleyiş farklarının etmen planlarıyla alınması gerekmektedir. Bu işleyiş farklarına örnek olarak veb servislerinin senkron çalışması etmenlerin ise asenkron ve protokol bazlı çalışması, veb servisinin etmen platformuna sunulması istenmeyen işleyiş detaylarının (kimlik doğrulama vb.) etmen tarafından ele alınması verilebilir. Şekil 4.14'de, Şekil 3.4'deki etkinlik diyagramından faydalanılarak oluşturulan harici servis ilan planının bir seviye açılmış hali gösterilmektedir.

Yürütme sürecinde SSE, SİE'den gelen istek doğrultusunda bir ser-



Şekil 4.15. SSE için AVSM tabanlı Anlamsal Servis Yürütme HGA Planı

visi çalıştırır ve sonucunu SİE'ye gönderir. SSE, SİE'den gelen platform ontolojisindeki ilgili servis çalıştırma planını bularak servis çalıştırımını başlatır. Bu plan isteği servisin kullandığı ontolojiye (veya kavramlara) çevirir, servisi çalıştırır ve servis sonucunu tekrar platform ontolojisine çevirerek SİE'ye gönderir. Şekil 4.15'de, Şekil 3.10'deki etkinlik diyagramında yararlanılarak modellenen harici servis yürütme planı gösterilmiştir. Şekilde de görüldüğü gibi *servis yürütme planının istenen servise özel planı bulup çalışma-kipinde yaratıp kendisine bağlayabilmesi gerekmektedir.*

4.3 AVSM Planlarına Yönelik Planlayıcı İhtiyaçları

Önceki bölümde Etmen Tabanlı Anlamsal Servis Mimarisi'ndeki etmenlerin planları gösterilmiştir. Bu planlar tasarlanırken mimari elemanlarının etkinlik diyagramlarından yararlanılmıştır. Bu tasarım esnasında bu planları işletecek HGA planlayıcının sahip olması gereken ihtiyaçlar saptanmıştır. Bu ihtiyaçları toparlayacak olursak:

- AVSM'nin altsüreçleri için soyut görev yapıları içeren yeniden kullanılabilir şablon planların tanımlanabilmesi ve bu şablonların soyut görevler özelleştirilerek etmenlerin gerçek planlarının türetilmesi için kullanılması.

- Plan yapısında özyineleme desteği.

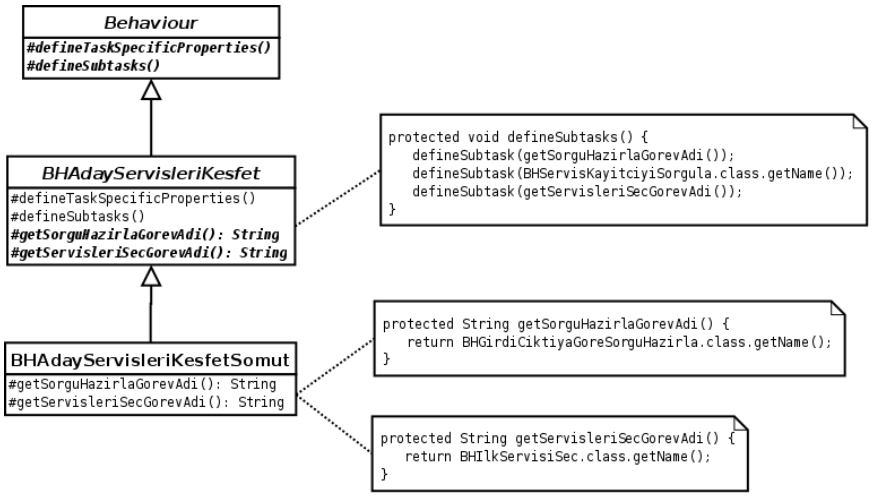
- Gerçek etmen planlarındaki anlamsal servis planlarını kullanarak birleşik servislerin oluşturulması.

- Çalışma-kipinde bir planın başka bir planı yaratıp kendisine bağlayabilmesi.

Tasarlanan planlayıcı bu ihtiyaçların ilk üç tanesini gerçekleştirebilmektedir. Bunu yaparken planlayıcının ve plan yapısını özelliklerinden yararlanıldı. İlerleyen altbölümlerde bu özelliklerin tasarlanan planlayıcıda nasıl gerçekleştirildiği anlatılmaktadır.

4.3.1 Şablon Plan Tanımlanması

Sunulan planlayıcıda şablon planlar yaratmak mümkündür. Temel fikir nesneye yönelimdeki soyut sınıf mantığına dayanmaktadır. Yani, genel bir plan yapılandırmak için, planın ana kısımları tanımlanır, bazı kısımları (görevleri) gerçekleştirilmez ve soyut tanımlanır. Böylelikle gelecekte ileride özelleştirilip kullanılabilirler. Yeniden kullanılabilir şablon planlarda tanımlanan görevler, soyut görevlerin yeniden tanımlanmasıyla, somut planlar olarak genişletilebilirler. Buradaki önemli nokta, genişletilecek olan soyut görevin tanımlarıyla (ihtiyaç ve sonuç yuvaları) genişletilmiş somut



Şekil 4.16. “Aday Servisleri Keşfet” soyut görevi

görevin tanımlarının çakışması gerekmektedir. Şablon plan yazmak için, plan normal bir plan gibi yapılandırılmalı ama soyut görevlerin soyut görev tanımlarıyla tanımlanmalıdır.

SEAGENT çerçevesinde altgörev tanımlamak için `defineSubtask()` metodu altgörevin sınıf adı kendisine parametre verilerek kullanılmaktadır. Eğer bir soyut altgörev tanımlanmak istenirse, yapılması gereken tek şey soyut altgörevin sınıf adının dolaylı olarak verilmesidir. Bu, altgörevin adını döndüren soyut bir metod kullanılıp bu metodun dönüş değeri `defineSubtask()` metoduna verilerek yapılabilir.

Örneğin, Şekil 4.9’deki “Servis(ler)i Seç” soyut görevi “Aday Servisleri Keşfet” ebeveyn görevi altında `getServisleriSecGorevAdi()` soyut metodu kullanılarak tanımlanabilir (`defineSubtask( getServisleriSecGorevAdi()`

)). Bu planın somut gerçekleştirimi bu plan genişletilerek ve soyut metodlar somut görevlerin isimlerini döndürecek şekilde gerçekleştirilerek yapılır. Şekil 4.16'de "Aday Servisleri Keşfet" soyut görevi ve onun somut bir gerçekleştirimi gösterilmektedir.

4.3.2 Özyineleme

Özyineleme ile kastedilen bir görevin örneğinin görev ağında aynı görevin bir diğer örneğinin atası olmasıdır. Bunu sağlayabilmek için plan yapısının (planlayıcının) özyineleme desteği olması gerekmektedir. Yani bir görevin kendisini kendi altgörevi olarak barındırabilmesini gerekmektedir. Tasarlanan planlayıcıda altgörev tanımlamada herhangi bir kısıt yoktur, dolayısı ile bir görev kendisini altgörev olarak kolaylıkla barındırabilmektedir. Ancak buradaki önemli nokta görev açılımlarının, aynı geleneksel programlamadaki özyinelemeli metodlar gibi, kontrollü gitmesidir. Tanıtlan planlayıcıda, bir görevin açılımı ihtiyaçlarına göre belirlenmektedir: bütün ihtiyaçları sağlanan görevler veya hiç ihtiyacı olmayan görevler altgörevlerine ayrıştırılır. Bu yüzden, özyineli bir HGA planında, özyineli görevler en az bir adet ihtiyaç içermelidir ve bu ihtiyacın alacağı değer atasının aldığından farklı olmalıdır. Aksi takdirde etmen göçmesine sebep olabilecek bir sonsuz döngü oluşur.

4.3.3 Birleşik Servis Oluşturulması

SEAGENT çerçevesinin plan kütüphanesinde servis çalıştırmak için gerek servis istemci etmen için gerekse de servis sağlayıcı etmen için hazır planlar bulunmaktadır. Bu planlar önceki bölümlerde anlatılmıştı. SİE'nin bir-

den fazla servisi tek plan içerisinde çalıştırması, dolayısı ile birleşik servis oluşturması mümkündür. Bölüm 4.2.1'de anlatılan "Servis Çalıştır" planı (Şekil 4.8) farklı servisler için özelleştirilebilir ve aynı plan içerisinde kullanılabilir. Şekil 4.12'deki "Otel Odası Kiralama" planı mesela buna bir örnektir.

Öte yandan SSE tarafında da birden fazla servis birleştirilerek tek bir servis olarak platforma sunulabilmeli, ona yönelik bir istek geldiğinde SSE her servisi tek bir plan içerisinde ayrı ayrı çalıştırıp istemciye ilgili sonucu döndürebilmelidir. Ancak bu tez çalışmasında buna yönelik planlar tasarlanmadığı için şu anda yapılamamaktadır.

4.4 Sonuç

Sonuç olarak bu bölümde Bölüm 3'te anlatılan Etmen Tabanlı bir Anlamsal Servis Mimarisi'ndeki SSE ve SİE'nin hazır planlara sahip olması gerektiği ve bu planlardaki bazı gereksinimlerden dolayı böyle bir platformdaki etmen planlayıcının sahip olması gerek özellikler saptanmış, bu amaca yönelik tasarlanan ve geliştirilen etmen planlayıcısı anlatılmıştır.

5 SONUÇ ve TARTIŞMA

5.1 Gözden Geçirme ve Tartışma

Bu tezde anlamsal veb servislerini kullanan etmenlerin içyapısı ele alınmaktadır. Araştırma alanı veb servislerini, anlamsal vebi, çok-etmenli sistemleri ve planlamayı kapsamaktadır. Tezdeki zorluklardan bir tanesi etmen tabanlı anlamsal servis mimarisinin elemanlarının ve işleyişinin tasarlanışıdır. Böyle bir mimaride servisleri etmen platformuna uyarlayan ve sunan servis sağlayıcı etmenler ve servisi kullanan servis istemci etmenler bulunmaktadır. Tezde bu etmenlerin servis kullanımına yönelik yeniden kullanılabilir planlara sahip olması gerektiği belirtilmektedir. Böylelikle sıradan geliştiriciler kolaylıkla anlamsal vebde çalışan yazılımlar geliştirebileceklerdir. Ayrıca bu planları çalıştırabilecek bir planlayıcının sahip olması gereken yetenekler de saptanmaktadır. Mevcut planlayıcılar bu ihtiyaçları tam olarak karşılamadığından tezde bu amaca yönelik HGA tabanlı bir planlayıcı tasarlanmış ve geliştirilmiştir.

5.2 Katkılar

Bu tezin katkıları aşağıda sıralanmıştır:

- AVSM'nin kavramsal modelinin temel ihtiyaçlarının bütün süreçlerini kapsayacak şekilde yerine getiren bir yazılım platformunu tanımlamaktadır.
- Tanımlanan mimarinin süreçlerine yönelik planlar tasarlanmıştır.

- Süreçlere yönelik tasarlanan planlar doğrultusunda planlayıcı ihtiyaçları belirlenmiştir.

5.3 İleriye Dönük Araştırma Olanakları

Bu tezde sunulan işi geliştirebilecek ve genişletebilecek birçok farklı araştırma olanağı var:

- **Çalışma-kipinde bir Planın Başka bir Planı Yaratıp Kendisine Bağlayabilmesi** Planlayıcının çalışma-kipinde bir planın başka bir planı yaratıp kendisine bağlayabilmesine olanak sağlaması gerekmektedir.

- **SSE tarafından Servislerin Birleştirilerek İlan Edilmesi** Servis Sağlayıcı Etmen tarafından servislerin birleştirilerek tek bir servis olarak platforma sunulabilmesi için gerekli süreçlerin ve planların tanımlanması gerekmektedir.

- **Farklı Servis Tanım Dil(ler)i Desteği** Bu çalışmada yalnızca VSTD veya S-VOD ile tanımlanan servisler dikkate alınmıştır. VSMO, A-VSTD gibi başka dillerle tanımlanmış olan servislerin de platforma eklenebilmesi gerekmektedir.

- **Plarfotmda Farklı Ontolojileri Destekleme** Tezde sunulan etmen platformunda tasarım platformun tek bir etki alanı kullandığını varsaymaktadır. Ancak etmen platformları birden fazla etki alanına (ve dolayısı ile o etki alanlarının ontolojilerine) sahip olabilmelidir.

- **Protokol Tasarlanması** Anlamsal servis çalıştırılması süreci mevcut çalışmada işleyiş ve akış açısından birbiriyle bağımlı ancak protokol bağlamında birbirinden bağımsız altsüreçlerden oluşmuş bir yapıdadır. Bu sürece bütününe yönelik özel bir protokol tasarlanması sürecin akışının daha iyi akışın daha kolay kontrol edilebilir olması için gereklidir.

- **Uzlaşma Sürecinin Detaylandırılması** Tezde detaylarına girilmeyen uzlaşma sürecinin alt adımlarının ve servis kalitesi ölçütlerinin süreci ve planlamayı nasıl etkileyeceğinin araştırılması gerekmektedir.

KAYNAKLAR DİZİNİ

Bacchus, F. and Kabanza, F. (2000). Using temporal logics to express search control knowledge for planning. *Artif. Intell.*, 116(1-2):123–191.

Bellifemine, F., Poggi, A., and Rimassa, G. (2001). Developing multi-agent systems with a fipa-compliant agent framework. *Softw., Pract. Exper.*, 31(2):103–128.

Burstein, M., Bussler, C., Zaremba, M., Finin, T., Huhns, M., Paolucci, M., Sheth, A., and Williams, S. (2005). A semantic web services architecture. *IEEE Internet Computing*, Volume 9 Issue 5:72 – 81.

Bylander, T. (1991). Complexity results for planning. In *IJCAI*, pages 274–279.

Cardoso, J., Sheth, A. P., Miller, J. A., Arnold, J., and Kochut, K. (2004). Quality of service for workflows and web service processes. *J. Web Sem.*, 1(3):281–308.

Carman, M., Serafini, L., and Traverso, P. (2003). Web service composition as planning. In *Workshop on Planning for Web Services*, Trento, Italy.

Currie, K. and Tate, A. (1991). O-plan: The open planning architecture. *Artif. Intell.*, 52(1):49–86.

Dias, M. B., Lemai, S., and Muscettola, N. (2003). A real-time rover executive based on model-based reactive planning. In *The 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space*.

Dickinson, I. and Wooldridge, M. (2005). Agents are not (just) web servi-

ces: investigating bdi agents and web services. In *AAMAS'05 Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE'2005)*.

Dikenelli, O., Erdur, R. C., Gümüs, Ö., Ekinci, E. E., Gürcan, Ö., Kardas, G., Seylan, I., and Tiryaki, A. M. (2005a). Seagent: A platform for developing semantic web based multi agent systems. In *AAMAS*, pages 1271–1272. ACM.

Dikenelli, O., Erdur, R. C., Kardas, G., Gümüs, Ö., Seylan, I., Gürcan, Ö., Tiryaki, A. M., and Ekinci, E. E. (2005b). Developing multi agent systems on semantic web environment using seagent platform. In Dikenelli, O., Gleizes, M.-P., and Ricci, A., editors, *Proceedings of ESAW'05*, Lecture Notes in Computer Science, Berlin, Germany. Department of Computer Engineering Ege University, Springer Verlag.

Dikenelli, O., Gümüs, Ö., Tiryaki, A., and Kardas, G. (2005c). Engineering a multi agent platform with dynamic semantic service discovery and invocation capability. In Eymann, T., Klügl, F., Lamersdorf, W., Klusch, M., and Huhns, M. N., editors, *MATES*, volume 3550 of *Lecture Notes in Computer Science*, pages 141–152. Springer.

Edelkamp, S. and Helmert, M. (2001). Mips: The model-checking integrated planning system. *AI Magazine*, 22(3):67–72.

Erdur, R. C., Dikenelli, O., Seylan, I., and Gürcan, Ö. (2004a). An infrastructure for the semantic integration of fipa compliant agent platforms. In *AAMAS*, pages 1316–1317.

Erdur, R. C., Dikenelli, O., Seylan, I., and Gürcan, Ö. (2004b). Semantically federating multi-agent organizations. In *ESAW*, pages 74–89.

Erol, K., Hendler, J. A., and Nau, D. S. (1994). Umcp: A sound and complete procedure for hierarchical task-network planning. In *AIPS*, pages 249–254.

Erol, K., Hendler, J. A., and Nau, D. S. (1996a). Complexity results for htn planning. *Ann. Math. Artif. Intell.*, 18(1):69–93.

Erol, K., Hendler, J. A., and Nau, D. S. (1996b). Complexity results for htn planning. *Ann. Math. Artif. Intell.*, 18(1):69–93.

Erol, K., Nau, D. S., and Subrahmanian, V. S. (1995). Complexity, decidability and undecidability results for domain-independent planning. *Artif. Intell.*, 76(1-2):75–88.

Fensel, D. and Bussler, C. (2002). The web service modeling framework wsmf. *Electronic Commerce Research and Applications*, 1(2):113–137.

Fox, M. and Long, D. (2003). Pddl2.1: An extension to pddl for expressing temporal planning domains. *J. Artif. Intell. Res. (JAIR)*, 20:61–124.

Gerevini, A. and Serina, I. (2002). Lpg: A planner based on local search for planning graphs with action costs. In *AIPS*, pages 13–22.

Gibbins, N., Harris, S., and Shadbolt, N. (2003). Agent-based semantic web services. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 710–717, New York, NY, USA. ACM Press.

Graham, J. R., Decker, K., and Mersic, M. (2003). Decaf - a flexible multi agent system architecture. *Autonomous Agents and Multi-Agent Systems*, 7(1-2):7–27.

Greenwood, D. and Calisti, M. (2004). Engineering web service - agent integration. In *SMC (2)*, pages 1918–1925. IEEE.

Gupta, S. and Bourne, D. (1999). Sheet metal bending: Generating shared setups. *ASME Journal of Manufacturing Science and Engineering*, 121:689–694.

Gupta, S., Nau, D., and Regli, W. (1998). Imacs: A case study in real world planning.

Gürcan, O., Kardas, G., Gümüs, O., Dikenelli, O., Cakirlar, I., Cetin, O., Eliacik, A. B., and Kir, H. (2007a). Etmen tabanli bir anlamsal servis mimarisi. In *Ulusal Yazilim Mühendisliği Sempozyumu ve Sergisi (UYMS'07)*. Sunum ve basım için kabul edildi.

Gürcan, O., Kardas, G., Gümüs, O., Ekinci, E. E., and Dikenelli, O. (2006). A Planner for Implementing Semantic Service Agents based on Semantic Web Services Initiative Architecture. In *Fourth European Workshop on Multi-Agent Systems (EUMAS'06)*.

Gürcan, O., Kardas, G., Gümüs, O., Ekinci, E. E., and Dikenelli, O. (2007b). An MAS Infrastructure for Implementing SWSA based Semantic Services. In *Service-Oriented Computing: Agents, Semantics, and Engineering*, number 4504 in Lecture Notes in Computer Science, pages 118 – 131. Springer-Verlag.

Hoffmann, J. and Nebel, B. (2001). The ff planning system: Fast plan generation through heuristic search. *J. Artif. Intell. Res. (JAIR)*, 14:253–302.

Klusch, M., Fries, B., and Sycara, K. (2006). Automated semantic web service discovery with owls-mx. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 915–922, New York, NY, USA. ACM Press.

Kvarnström, J. and Doherty, P. (2000). Talplanner: A temporal logic based

forward chaining planner. *Ann. Math. Artif. Intell.*, 30(1-4):119–169.

Lara, R., Roman, D., Polleres, A., and Fensel, D. (2004). A conceptual comparison of wsmo and owl-s. In *Proceedings of the European Conference on Web Services (ECOWS 2004)*, Erfurt, Germany.

Lopes, A. and Botelho, L. (2007). Executing Semantic Web Services with a Context-Aware Service Execution Agent. In *Service-Oriented Computing: Agents, Semantics, and Engineering*, number 4504 in Lecture Notes in Computer Science, pages 1 – 15. Springer-Verlag.

Maedche, A., Motik, B., Silva, N., and Volz, R. (2002). Mafra - a mapping framework for distributed ontologies. In *EKAW '02: Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*, pages 235–250, London, UK. Springer-Verlag.

McDermott, D. (1998). PDDL, the planning domain definition language. Technical Report 1165, Yale Center for Computational Vision and Control, New Haven, CT.

McDermott, D. (2002). Estimated-regression planning for interactions with web services.

McIlraith, S. and Son, T. (2002). Adapting golog for composition of semantic web services.

Mitkas, P., Symeonidis, A., Kechagias, D., Athanasiadis, I. N., Laleci, G., Kurt, G., Kabak, Y., Acar, A., and Dogac, A. (2002). An agent framework for dynamic agent retraining: Agent Academy. In Stanford-Smith, B., Chiozza, E., and Edin, M., editors, *Challenges and Achievements in e-business and e-work*, pages 757–764, Prague, Czech Republic.

Muscettola, N., Nayak, P. P., Pell, B., and Williams, B. C. (1998). Remote agent: To boldly go where no AI system has gone before. *Artificial Intelligence*, 103(1-2):5–47.

Nau, D. S., Au, T.-C., Ilghami, O., Kuter, U., Muñoz-Avila, H., Murdock, J. W., Wu, D., and Yaman, F. (2005). Applications of shop and shop2. *IEEE Intelligent Systems*, 20(2):34–41.

Nau, D. S., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., and Yaman, F. (2003). Shop2: An htn planning system. *J. Artif. Intell. Res. (JAIR)*, 20:379–404.

Nguyen, T. X. and Kowalczyk, R. (2005). Ws2jade: Integrating web service with jade agents. In *AAMAS'05 Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE'2005)*.

Nwana, H. S., Ndumu, D. T., Lee, L. C., and Collis, J. C. (1999). ZEUS: a toolkit and approach for building distributed multi-agent systems. In Etzioni, O., Müller, J. P., and Bradshaw, J. M., editors, *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 360–361, Seattle, WA, USA. ACM Press.

Pistore, M., Bertoli, P., Barbon, F., Shaparau, D., and Traverso, P. (2004). Planning and monitoring web service composition.

Russell, S. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition.

Sacerdoti, E. D. (1975). The nonlinear nature of plans. In *IJCAI*, pages 206–214.

Singh, M. P. and Huhns, M. N. (2005). *Service-oriented Computing - Semantic, Processes, Agents*. John Wiley & Sons, Ltd.

Sirin, E., Parsia, B., and Hendler, J. (2004a). Filtering and selecting semantic web services with interactive composition techniques. *IEEE Intelligent Systems*, 19(4):42–49.

Sirin, E., Parsia, B., Wu, D., Hendler, J., and Nau, D. (2004b). Htn planning for web service composition using shop.

Smith, S. J. J., Nau, D. S., and Throop, T. A. (1998). Computer bridge - a big win for ai planning. *AI Magazine*, 19(2):93–106.

Srivastava, B. and Koehler, J. (2003). Web service composition — current solutions and open problems. In *ICAPS 2003*.

Stollberg, M., Roman, D., Toma, I., Keller, U., Herzog, R., Zugmann, P., and Fensel, D. (2005). Semantic web fred - automated goal resolution on the semantic web. In *38th Annual Hawaii International Conference*, pages 111c–111c.

Sycara, K., Williamson, M., and Decker, K. (1996). Unified information and control flow in hierarchical task networks. In *Working Notes of the AAAI-96 workshop 'Theories of Action, Planning, and Control'*.

Tate, A. (1977). Generation project networks. In *International Joint Conference on Artificial Intelligence*, pages 888 – 893.

Tate, A. (1999). In *MIT Encyclopedia of Cognitive Science*. MIT Press, Cambridge.

Tate, A., Drabble, B., and Kirby, R. (1994). O-Plan 2. In Fox, M. and Zweben, M., editors, *Knowledge Based Scheduling*. Morgan Kaufmann, San Mateo, California.

Traverso, P. and Pistore, M. (2004). Automated composition of semantic web services into executable processes. pages 380–394.

Varga, L. Z., kos Hajnal, and Werner, Z. (2003). *Engineering Web Service Invocations from Agent Systems*, volume 2691, pages 626 – 635. Lecture Notes in Computer Science.

Varga, L. Z., kos Hajnal, and Werner, Z. (2004). *An Agent Based Approach for Migrating Web Services to Semantic Web Services*, volume 3192, pages 381 – 390. Lecture Notes in Computer Science.

Wilkins, D. E. (1990). Can ai planners solve practical problems? *Computational Intelligence*, 6:232–246.

Williamson, M., Decker, K., and Sycara, K. (1996). Unified information and control flow in hierarchical task networks.

Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., and Sheng, Q. Z. (2003). Quality driven web services composition. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 411–421, New York, NY, USA. ACM Press.

Zou, Y. (2004). *Agent-Based Services for the Semantic Web*. PhD thesis, the Faculty of the Graduate School of the University of Maryland.

EKLER

EkA DAVRANIŞ SINIFI KOD ÖRNEĞİ

Şekil A.1'de Şekil 4.3'deki "Davranış 1" adlı davranışa ait kod örneği gösterilmektedir. Bölüm 4.1.1'de anlatıldığı gibi davranış sınıf haline getirilirken `Behaviour` sınıfını genişletmiştir ve sınıfın ismi de `BHDavranis1` olarak konmuştur. Bu davranışın iki tane ihtiyacı, bir tane de sonucu vardır ve bunlar `defineTaskSpecificProperties()` metodu içerisinde `defineProvision()` ve `defineOutcome()` metodlarıyla tanımlanmaktadırlar.

Dikkat edilecek olunursa ihtiyaç ve sonuçların isimleri genel sabitler olarak tanımlanmaktadır. Bunun sebebi görevler arası bağlantıların kurulabilmesi için bu davranış içerisindeki ihtiyaç ve sonuçlara diğer sınıflar tarafından da erişilebilinmesini sağlamaktadır.

Şekil 4.3'de de görülebileceği gibi "Davranış 1" iki adet altgörevi bulunmaktadır: "Eylem 1" ve "Davranış 2". Bu altsınıflar davranışın `defineSubtasks()` metodunda tanımlanmaktadır.

Son olarak Şekil 4.3'de gösterildiği üzere "Davranış 1" görevinin altgörevleri ile arasında bir adet miras ve bir adet ters miras bağlantısı, ve altgörevleri arasında da bir adet ihtiyaç bağlantısı bulunmaktadır. Bu bağlantı bilgileri davranış içerisinde bir indirgeme şemasında tutulmaktadır ve dolayısı ile bu görev içerisinde `defineLinks()` metodu içerisinde `defineInheritance()`, `defineDisinheritance()` ve `defineProvisionLink()` metodları ile Şekil A.1'de gösterildiği gibi tanımlanmaktadırlar.

```
package tr.edu.ege.seagent.plan.example;
import tr.edu.ege.seagent.planner.exception.TaskException;
import tr.edu.ege.seagent.planner.task.*;
public class BHDavranis1 extends Behaviour {
   public static final String P_IHTIYAC_1 = "ihtiyac 1";
   public static final String P_IHTIYAC_2 = "ihtiyac 2";
   public static final String O_SONUC = "sonuc 1";
   public BHDavranis1() throws TaskException {
      super();
   }
   public BHDavranis1(int planNo, Behaviour superBehaviour) throws
TaskException {
      super(planNo, superBehaviour);
   }
   @Override
   protected void defineTaskSpecificProperties() {
      defineProvision(P_IHTIYAC_1);
      defineProvision(P_IHTIYAC_2);
      defineOutcome(O_RESULT);
   }
   @Override
   protected void defineSubTasks() {
      defineSubTask(ACEylem1.class.getName());
      defineSubTask(BHDavranis2.class.getName());
   }
   @Override
   protected void defineLinks() throws TaskException {
      defineInheritance(P_IHTIYAC_1, ACEylem1.class.getName(),ACEylem1.P_IHTIYAC_1));
      defineProvisionLink(ACEylem1.class.getName(),ACEylem1.O_SONUC_2,
                          BHDavranis2.class.getName(),
BHDavranis2.P_IHTIYA4);
     defineDisinheritance(BHDavranis2.class.getName(),
BHDavranis2.O_SONUC_3,
                                                                  O_SONUC_1));
   }
}
```

Şekil A.1. “Davranış 1” kod örneği

EkB EYLEM SINIFI KOD ÖRNEĞİ

Şekil B.1'de Şekil 4.3'deki "Eylem 1" adlı eyleme ait kod örneği gösterilmektedir. Bölüm 4.1.1'de anlatıldığı gibi eylem sınıf haline getirilirken `Action` sınıfını genişletmiştir ve sınıfın ismi de `ACEylem1` olarak konmuştur. Bu eylemin bir tane ihtiyacı, bir tane de sonucu vardır ve bunlar `makeDefinitions()` metodu içerisinde `defineProvision()` ve `defineOutcome()` metodlarıyla tanımlanmaktadırlar.

Dikkat edilecek olunursa ihtiyaç ve sonuçların isimleri genel sabitler olarak tanımlanmaktadır. Bunun sebebi görevler arası bağlantıların kurulabilmesi için bu eylem içerisindeki ihtiyaç ve sonuçlara diğer sınıflar tarafından da erişilebilinmesini sağlamaktadır.

Bir eylem çalıştırıldığında işletilmesi istenen kod `Do()` metodu içerisine yazılmaktadır. Bu metod içerisinden bir ihtiyacın değerini okumak için `getProvision()` metodu ve bir sonuç üretmek için de `produceOutcome()` metodu Şekil B.1'de gösterildiği gibi kullanılmaktadır.

```
package tr.edu.ege.seagent.plan.example;
import tr.edu.ege.seagent.planner.exception.TaskException;
import tr.edu.ege.seagent.planner.task.*;
public class ACEylem1 extends Action {
   public static final String P_IHTIYAC_1 = "ihtiyac 1";
   public static final String O_SONUC_2 = "sonuc 2";
   public ACEylem1(int planNo, Behaviour superBehaviour) throws
TaskException {
      super(planNo, superBehaviour);
   }
   @Override
   public void Do() throws Exception {
      Provision provision = this.getProvision(P_IHTIYAC_1);
      ... = provision.getValue();
      ...
      this.produceOutcome(O_SONUC_2, ...);
   }
   @Override
   protected void makeDefinitions() throws TaskException {
      defineProvision(P_IHTIYAC_1));
      defineOutcome(O_SONUC_2);
   }
}
```

Şekil B.1. “Eylem 1” kod örneği

EkC İNDİRGEME METODU KODU

Bölüm 4.1.1'de anlatıldığı gibi davranışlar altgörevlerini ve aralarındaki bilgi akışlarını tanımlamak için bir indirgeme şeması tutmaktadırlar. Bir davranışın çalışabilmesi için bu şemayı kullanarak altgörevlerini ayrıştırması ve bilgi akışını kontrol etmesi gerekmektedir.

Şekil C.1'de davranış sınıfındaki (`Behaviour`) indirgemeyi yapan `reduct()` metodu gösterilmektedir. Bu metod çağırıldığında öncelikle altgörevler yaratılmakta ve daha sonra da altgörevlere miras yoluyla giden herhangi bir ihtiyaç olup olmadığına bakılmaktadır. Altgörevlerin yaratılması bütün altgörevlerin tek tek yaratılıp altgörev vektörününe (`subTaskVector`) koyulması şeklinde olmaktadır. Miras ilişkisinin kontrolü için ise yaratılan bütün altgörevler tek tek checkInheritances() metoduna parametre geçilmektedir. Bu metod eğer bu davranış ile altgörevleri arasındaki miras ilişkilerini saptayıp ilgili değer aktarımlarını yapmaktadır.

```
final private void reduct() throws TaskException {
   // first create subtasks
   subTaskVector = new Vector<Task>();
   Iterator<String> taskDefIter;
   taskDefIter = getSubTaskDefinitionVector().iterator();
   while (taskDefIter.hasNext()) {
      String subTaskClassName = taskDefIter.next();
      Task subTask = createTask(subTaskClassName);
      subTask.setOwnerAgent(this.ownerAgentIdentifier);
      subTask.setPlanner(this.getOwnerPlanner());
      subTaskVector.add(subTask);
   }
   // then check inheritances
   Iterator<Task> taskIter = subTaskVector.iterator();
   while (taskIter.hasNext()) {
      Task subTask = (Task) taskIter.next();
      checkInheritances(subTask);
   }
}
```

Şekil C.1. `Behaviour` sınıfındaki `reduct()` metodu

EkD HEDEF ONTOLOJİSİ KOD ÖRNEĞİ

Şekil D.1'de Bölüm 4.1.2'de bahsedilen örnek eşleme ontolojisinin kodu gösterilmektedir.

```
<?xml version="1.0" encoding="WINDOWS-1254"?>
<rdf:RDF..>
  ...
  <Match rdf:ID="a104">
    <performative rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      request
    </performative>
    <act rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      oda-kirala
    </act>
    <behaviorID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      tr.edu.ege.seagent.plan.BHOdaKirala
    </behaviorID>
  </Match>
  ...
</rdf:RDF>
```

Şekil D.1. Örnek plan kütüphanesi

EkE PLANLAYICI KODLARI

Tezde anlatılan SEAGENT Planlayıcı kaynak kodları ve derlenmiş haliyele Java Arşivi (.jar) olarak projenin sunucunda yayınlanmaktadır. Kodun tezde anlatılan haline erişmek için http://seagent.ege.edu.tr/etmen/snapshots/Seagent/SeagentPlanner/20070222s/ adresindeki jar dosyası indirilmesi gerekmektedir.

EkF TÜRKÇE-İNGİLİZCE TERİMLER SÖZLÜĞÜ

ambar *i.*: repository
aktarım *i.*: transfer
bağlam *i.*: context
bağlantı *i.*: link
biçimsel *i.*: formal
bileşen *i.*: component
bilgi getirme *i.*: information retrieval
bilişsel *i.*: cognitive
birlikte işlerlik *i.*: interoperability
çerçeve *i.*: framework
çıktı *i.*: output
çok yönlülük *i.*: versatiliy
devingen *s.*: dynamic
elle-ayarlanabilir *s.*: hand-tailorable
eşgüdüm *i.*: coordination
eşleme *i.*: mapping
eşlemek *e.*: to map
eşleşme *i.*: matching
eşleştirmek *e.*: to match
etmen *i.*: agent
girdi *i.*: input
güdümlü *s.*: driven
hesaplama *i.*: computing
ilan *i.*: advertisement
ilgi alanı *i.*: domain
ilkel *s.*: primitive
keşif *i.*: discovery
miras *i.*: inheritance
profil *i.*: profile
sevk *i.*: dispatching
sıradüzen *i.*: sequence
süreç *i.*: process
ters miras *i.*: disinheritance
uyarlamak *e.*: to adapt
uzlaşma *i.*: engagement
üstün-metin *i.*: hyper-text
veb *i.*: web
vekil *i.*: proxy
yürütme *i.*: enactment
zemin *i.*: grounding

ÖZGEÇMİŞ

Önder Gürcan 3 Mart 1982 tarihinde İzmir'de doğdu. Ege Üniversitesi Bilgisayar Mühendisliği Bölümü'nde "Servis ile Etmen Entegrasyonu için bir Planlama Modülü Tasarımı ve Gerçekleştirimi" konusu üzerine yüksek lisans yapmaktadır (2004-). Bilgisayar Mühendisliği Lisans derecesini Ege Üniversitesi'nden almıştır (2000-2004). Aralık 2004'ten beri Ege Üniversitesi Bilgisayar Mühendisliği Bölümü'nde araştırma görevlisi olarak çalışmaktadır. Dergi ve konferans kitapçıklarından oluşan toplam altı tane bilimsel yayını vardır (Erdur et al., 2004b; Erdur et al., 2004a; Dikenelli et al., 2005b; Gürcan et al., 2007b; Gürcan et al., 2006; Gürcan et al., 2007a). Bir adet uluslararası çalıştayda organizasyon komitesinde yer almıştır (ESAW'05). Çok-etmenli sistemler, servise-yönelik mimariler, anlamsal veb ve yazılım sınanması gibi geniş bir akademik ilgi alanına sahiptir.