EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(YÜKSEK LİSANS TEZİ)

MOBİL ARAÇLAR İÇİN ETMEN TABANLI BİR ANLAMSAL WEB SERVİS SUNUM PLATFORMU GELİŞTİRME

Gizem OKKALIOĞLU

Bilgisayar Mühendisliği Anabilim Dalı

Bilim Dalı Kodu : 619.01.00

Sunuş Tarihi : 02.06.2008

Tez Danışmanı: Yrd. Doç. Dr. Rıza Cenk ERDUR

Bornova-İZMİR

II

Gizem OKKALIOĞLU tarafından YÜKSEK LİSANS tezi olarak sunulan "MOBİL ARAÇLAR İÇİN ETMEN TABANLI BİR ANLAMSAL WEB SERVİS SUNUM PLATFORMU GELİŞTİRME" 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 02.06.2008 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri: **İmza**

Jüri Başkanı : ....

Raportör Üye: ....

Üye : ....

IV

ÖZET

MOBİL ARAÇLAR İÇİN ETMEN TABANLI BİR ANLAMSAL WEB SERVİS SUNUM PLATFORMU GELİŞTİRME

OKKALIOĞLU, Gizem

Yüksek Lisans Tezi, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Yrd. Doç. Dr. Rıza Cenk Erdur

Haziran 2008, 131 sayfa

Günümüzde ağ üzerinde kullanıma hazır web servisleri büyük bir hızla artmaktadır. Yazılımları otomatik olarak web servisine dönüştüren geliştirme ortamlarının artması, şirketlerin yazılımlarını servis yönelimli olarak geliştirmesini hızlandırmış ve bu gelişmeye paralel olarak servislerin genel amaçlı tasarlanması ve ağa açılması yazılım geliştirme sürecinde temel bir etkinlik haline gelmiştir. Bu yönelim bağlamında yakın gelecekte ağ üzerinde farklı alanlarda yazılımlar tarafından doğrudan ya da birleştirilerek kullanılabilecek çok sayıda servis olacağı açıktır.

Servislerin artmasıyla birlikte servislerin bulunması, seçilmesi, istenen katma değeri yaratacak şekilde birleştirilmesi gibi problemler ortaya çıkmıştır. Anlamsal web standartları ve teknolojileri servislerin işlevlerinin ontolojiler ile tanımlanmasına olanak sağladığından bu problemlerin çözümü için formal bir altyapı sunmaktadır. Bu yönelim anlamsal web servisleri olarak adlandırılan bir araştırma alanının doğmasına yol açmıştır.

Bu tez çalışmasında mobil uygulamaların dış web servislerini kullanabilmesi için etmen tabanlı bir anlamsal web servis platformu geliştirilmiştir. Ayrıca yine tez çalışması kapsamında mobil araçlarda etmen tabanlı yazılım geliştirmek için SeagentLITE mobil etmen platformu geliştirilmiştir. Masaüstü ortamda bulunan SEAGENT platformunda, dış web servisleri servis sunucu etmen olarak isimlendirilen yapı ile anlamsal servislere çevrilerek aracı etmene kaydedilmektedirler. SeagentLITE ortamında çalışan, bulunduğu ortamın kısıtlı kaynaklarına uygun bir etmen uygulamasının kayıtlı servisleri kullanabilmesi için masaüstü ortamda arabulucu etmen adında bir etmen geliştirilmiştir. Arabulucu etmen, aracı etmen ile mobil ortamdaki etmen arasındaki koordinasyonu yönetmekten, anlamsal veri işleme işlevini mobil ortamdaki etmen için yerine getirmekten ve mobil ortamda servis kullanımı için gerekli olan servise özel dinamik arayüz tanımlarını mobil ortamdaki etmene göndermekten sorumludur. Tez çalışmasında son olarak oluşturulan bu platform üzerinde bir durum çalışması gerçekleştirilmiştir.

Anahtar sözcükler: Etmen teknolojileri, çok-etmenli platformlar, mobil araçlar, XML, web servisleri, anlamsal web, anlamsal web servisleri.

ABSTRACT

DEVELOPMENT OF AN AGENT BASED PLATFORM FOR SEMANTIC WEB SERVICE DELIVERY TO MOBILE DEVICES

OKKALIOĞLU, Gizem

MSc Thesis, Department of Computer Engineering

Supervisor: Asst. Prof. Dr. Rıza Cenk Erdur

June 2008, 131 pages

Nowadays, the number of web services that are provided through the net is increasing rapidly. Due to advances in development environments which convert software to web services, companies have begun to develop their software projects using service oriented approach. Based on this improvement, designing general purpose services and advertising them on the net have become a basic activity in software development process. Thus, it is clear that in the future there will be a lot of services on the net that are used either directly or via composition by software applications in several domains.

As a result of increase in the number of services, problems such as discovery, selection and composition of services to create the requested added-value have appeared. Semantic web standards and technologies present a formal infrastructure for the solution of these problems by providing a way for the definition of service properties using ontologies. This inclination has caused rising of a new research area called semantic web services.

In this thesis, an agent based semantic web service platform has been developed for mobile device access to external services. SeagentLITE agent platform, which provides a means for agent oriented software development on mobile devices, has also been developed within thesis study. On SEAGENT platform, which is situated on desktop environment, external web services are converted to semantic services by a component named as service provider agent. Then service provider agents are registered to broker agent. An agent named as mediator agent is developed on desktop environment for providing usage of registered services by a lightweight agent that is situated on SeagentLITE environment. Mediator agent is responsible for managing coordination between broker agent and mobile device agent, performing semantic data processing for mobile device agent, and sending service specific dynamic user interface definitions to mobile device agent for service access on mobile environment. Finally, a case study for the developed platform is given.

Keywords: Agent technologies, multi-agent platforms, mobile devices, XML, web services, semantic web, semantic web services.

TEŞEKKÜR

Tez çalışmam süresince tavsiyeleriyle çalışmama yön vermemdeki katkılarından dolayı Yrd. Doç. Dr. Rıza Cenk Erdur'a ve Prof. Dr. Oğuz Dikenelli'ye teşekkürü bir borç bilirim. Tez danışmanım Yrd. Doç. Dr.Rıza Cenk Erdur'a yüksek lisans eğitimim süresince verdiği destekten ötürü ayrıca teşekkür ederim.

Yüksek lisans eğitimim süresince faydalandığım 2228-Yurt İçi Lisansüstü Burs Programı sayesinde yüksek lisans eğitimimde başarıyı yakalamama büyük katkısı olan TÜBİTAK – BİLİM İNSANI DESTEKLEME DAİRE BAŞKANLIĞI'na ve tez çalışmam süresince yardımını esirgemeyen Hacer Küçükkeleş'e teşekkür ederim.

Tüm eğitim hayatım boyunca her zaman yanımda olan ve beni destekleyen annem Serap Okkalıoğlu, babam İbrahim Okkalıoğlu ve kardeşim Mahmut Okkalıoğlu'na tüm kalbimle teşekkürlerimi sunarım. Son ama aynı derecede önemli olarak, beni çalışmalarımda destekleyen nişanlım Gökhan Özman'a da en derin teşekkürlerimi sunarım.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

XIV

ŞEKİLLER DİZİNİ

Şekil Sayfa

ŞEKILLER DIZINI (devam)

Şekil Sayfa

KISALTMALAR DİZİNİ

Kısaltma	Açılım
ACC	Agent Communication Channel
ACL	Agent Communication Language
AMS	Agent Management System
API	Application Programming Interface
CDC	Connected Device Configuration
CLDC	Connected Limited Device Configuration
CVM	Compact Virtual Machine
DF	Directory Facilitator
FIPA	Foundation For Intelligent Physical Agents
FIPA-ACL	FIPA Agent Communication Language
FP	Foundation Profile
GUI	Graphical User Interface
HTN	Hierarchical Task Network
HTTP	Hyper-Text Transfer Protocol
IMP	Information Module Profile
JADE	Java Agent Development Environment
J2EE	Java 2 Enterprise Edition
J2ME	Java 2 Micro Edition
JRE	Java Runtime Environment

Kısaltma	**Açılım**
J2SE	Java 2 Standard Edition
JVM	Java Virtual Machine
KQML	Knowledge Query Manipulation Language
KVM	Kilobyte Virtual Machine
LEAP	Lightweight Extensible Agent Platform
MIDP	Mobile Information Device Profiles
OWL	Web Ontology Language
OWL-S	OWL for Services
PBP	Personal Basis Profile
PDA	Personal Digital Assistant
PP	Personal Profile
RDF	Resource Description Framework
SOAP	Simple Object Access Protocol
SWSA	SWSI Architecture
SWSI	Semantic Web Services Initiative
TCP	Transmission Control Protocol
UDDI	Universal Description, Discovery and Integration
URI	Uniform Resource Identifier
WSDL	Web Service Description Language
WSFL	Web Services Flow Language

Kısaltma	**Açılım**
WSMO	Web Service Modeling Ontology
XML	Extensible Markup Language

1. GİRİŞ

Mobil araç kullanımı hayatımıza girdiği günden bu yana sürekli artış göstermektedir. Bununla birlikte mobil araçların zaman içinde daha yetenekli araçlar haline gelmesi bu pazara yenilikler getirme imkanını da sunmuştur. Günümüzde masaüstü sistemler için geliştirilen pek çok uygulama mobil araçlara uyarlanabilmektedir. Gerçekleştirilen bu tez çalışmasında mobil araçlardan yazılım etmenleri ile anlamsal web servislerine erişim ve kullanım sağlanmaya çalışılmıştır. Anlamsal servislerin kullanım süreci öncesinde servislerin bulunması ve kullanılmak üzere hazırlanması işlemleri masaüstü sistemde bulunan ara katman bileşenleri ile gerçekleştirilmiştir. Tez çalışması kapsamında mobil araçların anlamsal web teknolojileri, etmen tabanlı yazılım geliştirme teknikleri ve web servisleri ile bütünleşimi üzerine çalışılmıştır.

Anlamsal Web kavramı, bugünkü web'in temelini oluşturan yapıları tasarlayan ve bulan kişi olan Tim BernersLee tarafından öne sürülmüştür. Anlamsal Web'in mevcut web ortamının geliştirilmesi sonucu web'in gelecek adımı olacağı düşünülmektedir. Anlamsal web yeni ve ayrı bir web olmayıp, bilgilere iyi tanımlanmış anlamların verildiği, bilgisayarların ve insanların birlikte çalışmalarına imkan veren bugünkü web'in bir uzantısıdır (Berners-Lee et al., 2001). Anlamsal web'teki temel amaç iyi tanımlanmış ve bağlantılandırılmış olan bilgilerin ve servislerin web ortamında makineler ve insanlar tarafından kolaylıkla okunabilir ve anlaşılabilir olmasını sağlayacak standartların ve teknolojilerin geliştirilmesidir.

Yazılım etmenleri (software agent) teknolojisi; açık, dinamik değişimlerin çok olduğu ve heterojen ortamların yapısına uygun yazılımların geliştirilmesine yönelik bir teknolojidir (Erdur, 2005). Yazılım etmenleri, 1990'lı yıllardan itibaren İnternet üzerine dağılmış,

değişik biçemlerde tutulan, sürekli değişen ve gelişen bilgilerin kullanıcı amaçları doğrultusunda derlenmesi ve işlenmesini gerektiren Web temelli uygulamalarda başarı ile kullanılmıştır. Yazılım etmenleri günümüzde de bilgi teknolojileri alanının en önemli ve en hızlı gelişim gösteren araştırma alanlarından birisi olma özelliğini korumaktadır.

Web servisleri son yıllarda web uygulamalarında sıklıkla kullanılan bir teknolojidir. Web servisleri WSDL[1] ile tanımlanmış ve SOAP[2] gibi standart ağ protokolleri ile erişilebilen yazılım bileşenleridir. Anlamsal web ise bu servislerin otomatik olarak bulunması, seçilmesi, çalıştırılması, karşılıklı işleyebilirliğinin ve izlenmesinin sağlanması için gerekli teknolojileri sunmaktadır. Bu bağlamda; anlamsal web servisleri, ontolojilerin yayınlanan servis tanımlamalarına anlam yüklediği ve bu sayede servis istemcilerinin servis tanımlamalarını yorumlayıp, servisleri çağırabildiği web servisleridir (Burstein et al., 2005). Anlamsal web servislerinin tanımlanmasında kullanılan ontolojiler ile web servislerini tanımlamada kullanılan WSDL'daki yetersizliklerin giderilmesi amaçlanmaktadır. Web servislerinin anlamsal web ile yeniden yapılandırılmasına yönelik çalışmalar da yine günümüzün popüler araştırma konularından biridir.

Gerçekleştirilen tez projesinin önemi de; hem yazılım etmenleri ve anlamsal web servislerini içermesi, hem de bu iki konu için tasarlanan mimarinin yine popüler bir konu olan mobil araçlara yönelik geliştirilmesi olarak ortaya çıkmaktadır.

---

[1] WSDL (Web Service Definition Language) URL : www.w3.org/TR/wsdl, son erişim 16 Şubat 2008.

[2] SOAP (Simple Object Access Protocol) URL : www.w3.org/TR/soap/, son erişim 16 Şubat 2008.

Bu tez projesi kapsamında SeagentLITE mobil etmen platformu geliştirilmiştir. SeagentLITE mobil etmen platformu sayesinde mobil araçtaki etmen veya etmenler SEAGENT (Dikenelli et al, 2005) çok-etmenli sistemine dahil edilebilmektedir. Yapılan çalışmada etmen tabanlı mimarinin geliştirimi tamamlandıktan sonra mobil araçtaki etmen veya etmenlerin çok-etmenli sistem içindeki anlamsal web servislerine erişimini ve web servislerini kullanımını sağlamak için gerekli altyapının oluşturulması amaçlanmıştır.

Gerekli mimari için öncelikle hangi işlemlerin mobil araç içinde gerçekleştirilebileceği ve hangi işlemler için masaüstü sistemin sunucu olarak kullanılması gerektiği belirlenmiştir. İlgili kısımlar belirlendikten sonra mobil araçtaki mimari bileşenlerinin ve sunucu tarafta bir ara katman görevi yapacak mimari bileşeninin gerçekleştiriminin yapılmıştır. Oluşturulan mimaride mobil araç ile servisler arasında etkileşimi sağlayacak bileşen bir arabulucu etmen olarak tasarlanmıştır. Ayrıca çalışma kapsamında mobil araçtaki etmen ve arabulucu etmenin karşılıklı iletişimini sağlacak bir protokol belirlenmiştir. Son olarak, gerçekleştirimi yapılan mimari örnek bir senaryoda sınanmıştır.

Tezin izleyen bölümleri şu şekilde düzenlenmiştir. İkinci bölümde tezin altyapısını oluşturan konu ve kavramlar açıklanmıştır. Bu konular mobil programlama ve mobil programlama kapsamında tez çalışmasında kullanılan J2ME(Java 2 Micro Edition) teknolojisi, etmenler, web servisleri, anlamsal web ve anlamsal web servisleridir. Üçüncü bölümde tez çalışması ile ilgili literatür çalışmalarından bahsedilecektir. Bu bölümde yazılım etmenleri ve web servisleri ile yazılım etmenleri ve anlamsal web servislerinin birlikte kullanımı, son olarak da mobil araçlardan servislerin kullanımını sağlayan literatür çalışmaları açıklanmıştır. Ayrıca tez çalışması literatür bağlamında değerlendirilmiştir. Dördüncü bölümde tez çalışmasının

gerçekleştiriminde kullanılan SEAGENT ve SeagentLITE etmen platformlarından bahsedilecektir. Beşinci bölümde mobil araçların anlamsal web servislerine erişimini sağlamak üzere geliştirilen çalışma tanıtılacaktır. Yine bu bölümde gerçekleştirilen çalışma için yapılan durum çalışmaları anlatılacaktır. Altıncı bölümde yapılan tez çalışmasının değerlendirilmesi yapılmakta ve ileri dönük yapılabilecek çalışmalardan bahsedilmektedir.

2.ALTYAPI

Bu bölümde tez çalışmasının altyapısını oluşturan konular tanıtılacaktır. Mobil programlama, etmenler, web servisleri, anlamsal web, anlamsal web servisleri konuları ile ilgili bilgiler izleyen alt bölümlerde anlatılacaktır.

2.1 Mobil Programlama

İletişimi ve bilgiye erişimi her yerde mümkün kılan mobil araçlar, bireysel hayatı ve kurumları tümüyle yeniden şekillendirmektedir. Günümüzde mobil araçlar hayatımızın vazgeçilmez bir parçası haline gelmiştir. Bu nedenle, masaüstü bilgisayar kullanıcılarına yönelik uygulamaların mobil araç kullanıcıları için uyarlanmasına yönelik projeler günden güne yoğunluk ve önem kazanmaktadır. Mobil araçlar için yazılım geliştirmeyi sağlayan platformlar ve emülatörler yazılım geliştiricilere büyük kolaylıklar sağlamaktadır.

Bu tez çalışmasının gerçekleştiriminde Java yazılımlarını elektronik ve gömülü cihazlara yerleştirebilmek için tasarlanmış olan Java 2 Micro Edition (J2ME)[3] kütüphanesinden faydalanılmıştır. Bu bölümde J2ME teknolojilerinin diğer Java platformları arasındaki yeri, sahip olduğu konfigürasyonlar ve profiller anlatılacaktır.

2.1.1 J2ME ’ye Genel Bakış

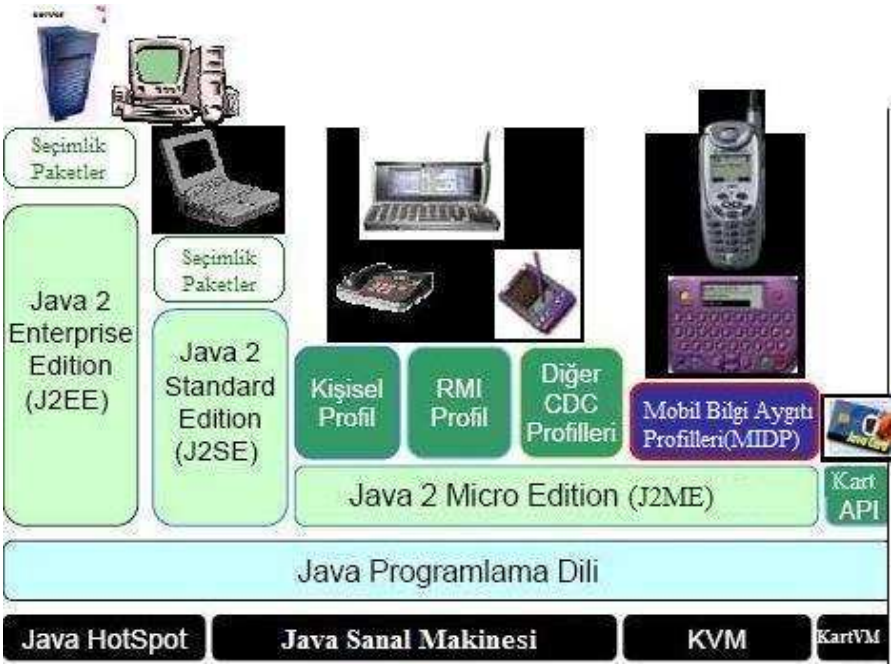
J2ME teknolojileri, Sun Microsystems, Inc. şirketi tarafından 1999 yılında tanıtılmış ve geniş bir tüketici alanının gereksinmelerini karşılamak için özel olarak geliştirilmiş olan JRE(Java Runtime Environment)'yi içermektedir. J2ME teknolojileri, çok geniş bir ürün çeşitliliğiyle son derece küçük cihazlarda kullanılabilmekte ve akıllı

[3] J2ME (Java 2 Micro Edition) URL : java.sun.com/javame/index.jsp, son erişim 29 Nisan 2008.

kartlar, çağrı cihazları, kod çözücüler(decoder) ve diğer küçük aygıtlar için güvenlik ve bağlanabilirlik çözümleri ile yararlı yardımcı programlar üretilmesine olanak sağlamaktadır.

J2ME teknolojileri Java yazılım ürünleri ailesinin yalnızca bir bölümüdür. Bu teknolojiyle bağlantılı diğer Java platformları Java 2 Platform Standard Edition (J2SE platformu) ve Java 2 Platform Enterprise Edition'dır (J2EE platformu). Java teknolojisi aynı zamanda Web hizmetleri oluşturma yöntemleri, XML[4] biçiminde bilgi aktarımları, çeşitli ağ protokolleri, araç setleri ve Java Web Start uygulaması sağlar. Java 2 Platform Standard Edition (J2SE), çekirdek Java sınıflarını ve API'lerini tarayıcılar üzerinde çalışabilecek uygulamalar da dahil standart istemci-sunucu uygulamalarını geliştirme ve çalıştırma olanağı sağlayan temel Java ortamıdır. Java 2 Platform Enterprise Edition (J2EE), J2SE'nin bir üst kümesini oluşturur. Genellikle ölçeklenebilir, işlem tabanlı ve veritabanı merkezli iş programların ve çok katlı, dağıtık uygulamaların geliştirilmesi için kullanılır. J2ME mobil telefonlar, PDA'lar, TV kutuları gibi gömülü sistem araçlar için uygulama geliştirmeye yönelik API'ler ve çalışma zamanı ortamı tanımlar. J2SE ve J2EE'den daha kısıtlı kaynaklara sahiptir. Şekil 2.1'de Java platformunun bileşenleri ve J2ME'nin platform içindeki yeri görülmektedir.

[4] XML (Extensible Markup Language) URL : www.w3.org/XML/, son erişim 16 Şubat 2008.



Şekil 2.1. Java 2 platformuna genel bakış

J2ME platformu temel bileşenleri, tüketici aygıtları ve gömülü aygıtlar pazarına Java çözümleri sunan çeşitli araçların ve teknolojilerin yanı sıra Connected Device Configurations (Bağlantılı Aygıt Yapılandırmaları), Connected Limited Device Configurations (Network Bağlantılı Kısıtlı Aygıt Yapılandırmaları) ve Mobile Information Device Profiles (Mobil Bilgi Aygıtı Profilleri) yazılımlarını içerir. İlerleyen bölümlerde bu yazılımların detaylarından bahsedilmektedir.

2.1.2 J2ME Konfigürasyonları

Konfigürasyon, uygulama geliştiricinin minimum Java kütüphanelerini ve sanal makine yeteneklerini belirtir. J2ME platform konfigürasyonu, bir araç ailesi ile ilgili minimum Java platformunu belirler. J2ME içinde CLDC (Connected Limited Device Configuration-

Limitli Bağlı Araç Konfigürasyonu) ve CDC (Connected Device Configuration-Bağlı Araç Konfigürasyonu) adında iki konfigürasyon bulunmaktadır. CLDC, kaynakları kısıtlı ve ağ bağlanabilirliği sınırlı olan cep telefonu, kişisel bilgisayarlar gibi alt uç (low-end) cihazları destekler. CDC ise, daha az kısıtları olan, yüksek seviyede iletişim olanağına sahip, ileri seviye gömülü cihazları destekler.

CDC konfigürasyonu Java uygulamaları için en az 2MB hafızası olan ve 16/32 bit işlemci ile çalışan araçlar için tanımlanmıştır. CDC araçları genelde sabit güç kaynağına ve geniş bantlı internet bağlantısına sahipler. TV kutusu araçları, görüntülü internet telefonları CDC konfigürasyonuna sahip araçlardır. J2SE sınıflarının bazı bölümlerini içeren CDC konfigürasyonu programları J2SE tabanında çalışabilirler. Bu konfigürasyonun çekirdeğinde ise CVM (Compact Virtual Machine) sanal makinesi bulunmaktadır. Bu konfigürasyona ait Java paketleri;

- java.io
- java.lang
- java.lang.ref
- java.lang.reflect
- java.math, java.net
- java.security
- java.security.cert
- java.text, java.util
- java.util.jar, java.util.zip
- javax.microedition.io

paketlerinden oluşmaktadır. Görüldüğü üzere CDC konfigürasyonu ait olduğu araç özelliklerine de bağlı olarak oldukça geniş programlama özelliklerini sağlamaktadır.

CLDC[5], daha kısıtlı kaynaklara sahip araçlara yönelik bir J2ME konfigürasyonudur. Bu konfigürasyonun çekirdeğinde bir Java sanal makinesi (Java Virtual Machine-JVM) olan KVM (Kilobyte Virtual Machine) vardır. KVM, JVM'nin bazı kısımlarının çıkartılarak daha önemli ve gerekli parçalarını içerecek şekilde tasarlanmış halidir. KVM ismindeki K harfi bu sanal makinenin Megabyte boyutunda hafızayla değil Kilobyte boyutunda hafıza ile çalışabilmesinden gelmektedir. Bu konfigürasyon 512 KB'tan küçük hafızası ve sınırlı ağ bağlantısı olan araçlar için tasarlanmıştır. Bu konfigürasyona ait Java paketleri;

- java.io
- java.lang
- java.lang.ref
- java.util
- javax.microedition.io

paketlerinden oluşmaktadır. Bu konfigürasyon da tasarlandığı araçların kaynak özelliklerine bağlı olarak çok daha az Java paketini içermektedir.

2.1.3 J2ME Profilleri

Konfigürasyonlar, uygulama yaşam döngüsünü yönetmek, kullanıcı arayüzü oluşturmak, araç içinde tutulan veriyi güncellemek ve yönetmek, ağ sunucusunda yer alan güvenlik bilgisine erişmek için gerekli sınıfları içermezler. Bu fonksiyonlar, profiller ya da seçimlik paketler tarafından

[5] CLDC spesifikasyonları, http://java.sun.com/products/cldc/, son erişim 31 Mart 2008

gerçekleştirilir. Profiller, konfigürasyonlar tarafından sağlanan çekirdek sınıf kümelerine, ilgili araca özgü servisler için gerekli sınıfları ekler. Bu sınıflar, alt katmandaki konfigürasyonda yer almayan, ama aracın kendine özgü olan işlevleri yerine getirmesinde ve kullanımında gerekli olan servisleri içerirler.

J2ME platformunda CLDC tabanlı profiller, Mobil Bilgi Aygıtı Profili (Mobile Information Device Profile - MIDP, versiyon 1.0 ve 2.0), Bilgi Birim Profili (Information Module Profile- IMP)' dir. CDC tabanlı üç profil vardır. Bunlar: Foundation Profile (FP), Personal Basis Profile (PBP), Personal Profile (PP) ' dir.

MIDP, J2ME platformu için tanıtılan ilk profildir. Temmuz 2000 yılında CLDC 1.0 konfigürasyonunu temel alan MIDP 1.0 versiyonu ve 2002 yılının sonunda CLDC 1.1 konfigürasyonunu temel alan MIDP 2.0 versiyonu tanıtılmıştır. MIDP profili cep telefonları, avuç içi bilgisayarları ve çağrı araçları gibi küçük araçları hedeflemektedir. Bu araçlar sahip oldukları düşük hafızaları (512kBtan az) ve kablosuz internet bağlantılarının yanı sıra pille (akü) ve düşük işlemci hızında çalışmaktadırlar.

MIDP mevcut profiller arasında ilk ve en yaygın olanıdır ve uygulamanın yaşam döngüsü, kullanıcı grafik arabirimleri, iletişim ağı ve kalıcı depolama ile ilgili kütüphanelerini içerir. MIDP, CLDC konfigürasyonunun üzerinde yer alır ve günümüzde Motorola, Nokia ve Ericsson gibi sektör devleri tarafından desteklenmektedir. MIDP 1.0 profili; grafik kullanıcı ara yüzünün kurumu (GUI - Graphic User Interface), kullanıcının girdiği verileri işlemek, grafik dosyaların görüntülenmesi, zaman sayacının kullanımı, verileri veritabanından okumak/silmek ve verileri kaydetmek, HTTP bağlantılarını kurma gibi imkanlar sunmaktadır. MIDP 2.0 profili ise bu imkanlara ek olarak resim işleme (RGB), gelişmiş çoklu ortam, gelişmiş kullanıcı ara yüzü (GUI),

oyun API (Application Programming Interface - Uygulama Programlama ara yüzü), SSL üzerinden güvenli HTTP bağlantıları, sayısal imza gibi özellikleri de sağlamaktadır.

Gerçekleştirilen tez projesinde CLDC konfigürasyonu ve MIDP 2.0 profili kullanılmıştır. Kullanılan bu yapıda mobil araçların kısıtları, donanımsal başarımları ve gereksinimleri gözönüne alınarak kullanımı uygun olan Java kütüphaneleri J2ME ortamına dahil edilmiş, kimi kütüphanelerin gereksinimine rağmen mobil araçların kaynak kısıtları nedeniyle J2ME ortamına eklenmemiştir. Örneğin ontoloji işlemleri için bir API olmaması nedeniyle tez projesinde mobil araçların ontoloji işlemleri masaüstü sistemler tarafından gerçekleştirilmiştir. İlerleyen bölümlerde bu kısıtlar sonucunda gerçekleştirilen mimari detaylı anlatılacaktır.

2.2 Etmenler

İnternet'in dağınık ve sürekli genişleyen yapısına uygun yazılımların geliştirilmesi isteği, yeni yazılım teknolojileri arayışlarına neden olmuştur. Bu arayışların en önemli sonuçlarından birisi de yazılım etmeni teknolojisidir. Yazılım etmenleri, 1990'lı yıllardan itibaren İnternet üzerine dağılmış, değişik biçemlerde tutulan, sürekli değişen ve gelişen bilgilerin kullanıcı amaçları doğrultusunda derlenmesi ve işlenmesini gerektiren web temelli uygulamalarda başarı ile kullanılmıştır (Sycara et al.,1996). Yazılım etmenleri günümüzde de bilgi teknolojileri alanının en önemli ve en hızlı gelişim gösteren araştırma alanlarından birisi olma özelliğini korumaktadır (Luck et al., 2005). Yazılım etmenleri algılayıcıları yardımıyla ortamı algılayan ve etkileyicileri yardımıyla bu ortamı etkileyen bir sistem olarak tanımlanmaktadır (Russell and Norvig, 2003).

Bir etmene ilişkin özellikler iki grupta incelenmektedir. Her etmende olması gereken özellikler birincil özellikler olarak adlandırılmaktadır. Birincil özellikler, bir etmeni klasik donanım veya yazılım sistemlerinden ayıran özelliklerdir. Bir donanım veya yazılım sisteminin etmen olarak nitelendirilebilmesi için mutlaka birincil özellikleri barındırması gerekmektedir. Bunun dışında, bir sistemin etmen olma kavramını güçlendiren ve genellikle geliştirilen uygulamaya bağlı olan özellikler de bulunmaktadır. Bu özellikler ikincil özellikler olarak adlandırılmaktadır. Gezicilik, öğrenme, dürüstlük, olumluluk etmenlere ait ikincil özelliklerdir. Bir etmene ait birincil özellikler ise şu şekilde özetlenebilir :

- Özerklik : Bir etmenin diğer bir elemana bağımlı olmaksızın kendine ait görevleri başlatabilme ve çalıştırabilme yeteneğidir.
- Karşıt-eylemlilik : Etmenin bulunduğu ortamı algılayıp, ortamdaki değişikliklere göre bilgisini, amaçlarını, eylemlerini değiştirmesi ve gerektiğinde diğer etmenlere ileti göndererek söz konusu değişikliği onlara da bildirebilmesidir.
- Amaç Yönelimlilik : Bir etmenin kendisinden beklenenleri yerine getirmesi için planlama yaparak bu planların gerektirdiği eylemleri gerçekleştirmesidir.
- Sosyal Yetenek : Etmenin planlarına ilişkin görevlerini tamamlayabilmek için diğer etmenler veya insan kullanıcıları ile etkileşimde bulunabilmesidir. Bu etkileşim ancak bir etmenler arası iletişim dili kullanarak sağlanabilecektir. Etmenler arası iletişim, çok-etmenli sistemlerin en önemli konularından birisidir.

- Kalıcı süreklilik : Etmenin başlattığı görevleri tamamladıktan sonra da etkin olarak kalma ve çevresini algılayıp gereken eylemleri gerçekleştirmek için hazır olarak bekleyebilme özelliğidir.

Etmenler davranışlarını kendilerine ait planlar ve bu planların çalışmasını sağlayan plan mekanizmasıyla gerçekleştirirler. Bir etmen bulunduğu ortamda yalnız olabilir ve görevlerini tek başına gerçekleştirebilir. Kimi durumlarda ise etmen bulunduğu ortamda yalnız değildir ve başka etmenlerle etkileşim halinde olabilir. Tek bir etmenin yalnız başına kendi bilgi ve bireysel yeteneklerini kullanarak çözemediği veya etkin bir biçimde çözemeyeceğini düşündüğü problemleri birbiriyle işbirliği yaparak eşgüdümlü bir biçimde çözmek için bir araya gelen etmenlerin oluşturduğu ağ, çok-etmenli sistem olarak tanımlanmaktadır (Durfee et al., 1989).

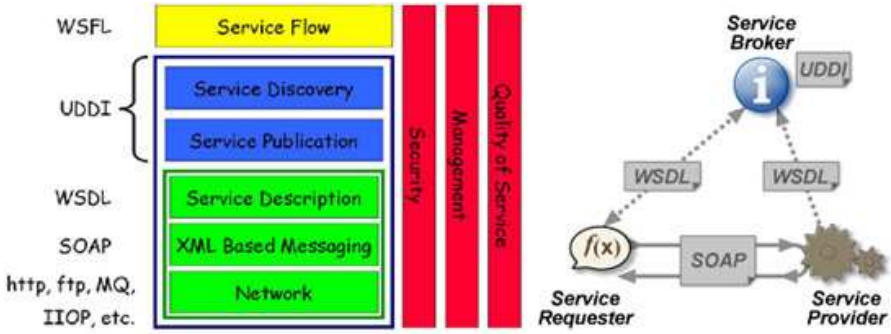
2.3 Web Servisleri

Günümüzde birbiriyle haberleşecek sistemleri gerçeklemek için en çok tercih edilen yöntem web servisleridir. Bu nedenle, web servislerinin kullanımının hızla genişlemesi, web-tabanlı uygulamalar arasındaki birlikte işlerlik ihtiyacının artmasıyla paralellik göstermektedir. Web servisleri, az bağımlı ve dağıtık yazılım bileşenleridir. Bilgi değişimi ve bilginin farklı yollarla bir araya getirilebilmesi için internet ortamını kullanarak bilgisayar ve diğer bilişim cihazlarını birbirlerine bağlamaktadır.

Web servisi, XML tabanlı mesajlaşmayı esas alır. Bu nedenle, haberleşecek sistemlerin birbirlerinin gerçeklenmelerinden haberdar olması veya platformlarının uyumlu olması gerekmez. Örneğin, Java ile geliştirilmiş ve UNIX sistem üzerinde çalışan bir uygulama ile .NET ile geliştirilmiş ve Windows işletim sistemi üzerinde çalışan bir uygulama,

birbirlerinin çalışma ortamlarından bağımsız olarak, XML iletişim standartları aracılığıyla iletişim kurabilir. Web servisinin en büyük faydası budur.

Öte yandan, XML web servisleri, SOAP adı verilen "Simple Object Access Protocol" (Basit Nesne Erişim Protokolü) ile iletişim kurarlar. SOAP Şekil 2.2'de de görüldüğü gibi xml tabanlı mesajlaşmayı sağlayan web servisi erişim standardıdır. SOAP protokolü sayesinde web servisleri, basit ve mesaj tabanlı bir iletişim sağlar.



Şekil 2.2. Web servis standartları katmanlı yapısı[6]

Web servisleri, WSDL (Web Service Description Language) adı verilen bir tanımlama dili ile sundukları servisin tanımını yaparlar. Bununla birlikte, UDDI[7] (Universal Description, Discovery and Integration) kayıt servisi sayesinde kurumlar ihtiyaç duydukları servisleri arayabilmekte veya kendi servislerini farklı kurumlar tarafından bulunabilir hale getirebilmektedir. Birden çok web servisinin

---

[6] URL : http://www.ias.com.tr/enterprise/articles/20070816-web-service.html, son erişim 1 Mayıs 2008.

[7] UDDI (Universal Description, Discovery, and Integration) URL : uddi.xml.org/uddi-org, son erişim 16 Şubat 2008.

kullanılmasını gerektiren durumlarda XML tabanlı bir dil olan WSFL (Web Services Flow Language) web servislerini kullanarak bir iş kompozisyonu yazılabilmesini sağlar.

Şekil 2.2'nin sağ kısmında görülen kısım servis odaklı mimariyi göstermektedir. Bu mimaride servis sağlayıcı (service provider) sahip olduğu servisi bir servis aracısına (service broker) kaydolarak gönderdiği WSDL tanımı ile yayınlar. Servis istemcisi (service requester), servis aracısının kayıtlarını arayarak gerek duyduğu servisi bulur. Servis istemcisi, sunulan servisi kullanmak üzere, servis sağlayıcıya bağlanır (Türker,2001).

2.4 Anlamsal Web

Anlamsal Web kavramı, bugünkü web'in temelini oluşturan yapıları tasarlayan ve bulan kişi olan Tim Berners-Lee tarafından öne sürülmüştür. Anlamsal web'in mevcut web ortamının geliştirilmesi sonucu web'in gelecek adımı olacağı düşünülmektedir. Anlamsal web yeni ve ayrı bir web olmayıp, bilgilere iyi tanımlanmış anlamların verildiği, bilgisayarların ve insanların birlikte çalışmalarına imkan veren bugünkü web'in bir uzantısıdır (Berners-Lee et al., 2001). Anlamsal web'teki temel amaç iyi tanımlanmış ve bağlantılandırılmış olan bilgilerin ve servislerin web ortamında makineler ve insanlar tarafından kolaylıkla okunabilir ve anlaşılabilir olmasını sağlayacak standartların ve teknolojilerin geliştirilmesidir. Bilgi gösterimi, bilgi getirme, veritabanları, çok-etmenli sistemler, doğal dil işleme, makine öğrenimi ve web servisleri anlamsal web ile ilgili alanlardandır.

Anlamsal web'te kaynakların tanımlanması için URI'ler (Uniform Resource Identifier), ontolojilerin birleştirilmesi için bağlantılar, söz dizim için XML, iletişim için de HTTP kullanılmaktadır. Ontolojiler ise Anlamsal web'in temel yapı taşı olarak ortaya çıkmaktadır. Ontolojiler

sayesinde kavramlar daha iyi bir şekilde tanımlanmakta ve kavramlar arası bağlantılar kurulmaktadır. Ontolojiler paylaşılan bir kavramsallaştırmanın biçimsel, belirgin bir tanımı olarak ifade edilmektedir. Tanımda bahsedilen biçimsel tanım ile makine okunabilirliği sağlanır, belirgin tanım ile de kesin terminoloji tanımları yapılır.

2.5 Anlamsal Web Servisleri

Anlamsal web servisleri, anlamsal web ortamının önemli bileşenlerinden biridir. Anlamsal web servisleri ontolojilerin yayınlanan servis tanımlamalarına anlam yüklediği ve bu sayede servis istemcilerinin servis tanımlamalarını yorumlayıp, servisleri çağırabildiği web servisleri olarak tanımlanabilmektedir (Burstein et al., 2005). WSDL kullanılarak tanımlanan bir web servis ara yüzü, özellikle ihtiyaç duyulan anlamsal çıkarsama mekanizmaları göz önünde bulundurulduğunda anlamsal web ortamı için yetersiz kalmaktadır. Anlamsal web servislerinin tanımlanmasında kullanılan ontolojiler ile web servislerini tanımlamada kullanılan WSDL'daki yetersizliklerin giderilmesi amaçlanmaktadır. Anlamsal web servisi yeteneklerinin anlamsal web ortamında temsil edilmesi ve dinamik olarak bulunup kullanılması için geliştirilen standart bir web servis ontolojisi henüz bulunmamakla birlikte, üzerinde çalışılan ve geliştirilmekte olan iki aday ontoloji OWL-S (Web Ontology Language for Services)[8] ve WSMO (Web Servis Modelling Ontology)[9]'dur. Bunlardan OWL-S, WSMO'ya göre daha yaygın kabul görmekte ve anlamsal web'te çalışacak servislerin geliştirilmesinde daha fazla kullanılmaktadır. Anlamsal web servisleri ile

[8] OWL-S (Web Ontology Language for Services) URL : http://www.daml.org/services/owl-s/, son erişim 1 Mart 2008.
[9] WSMO (Web Servis Modelling Ontology) URL : http://www.wsmo.org/, son erişim 1 Mart 2008.

servislerin otomatik olarak bulunması, seçilmesi, çalıştırılması, karşılıklı işleyebilirliğini ve izlenmesini sağlanabilmektedir.

3. İLGİLİ ÇALIŞMALAR

Bu bölümde web servislerinin etmen sistemleri ile birlikte kullanımına yönelik yapılmış çeşitli çalışmalardan bahsedilecektir. Bazı çalışmalar etmen sistemleri ve web servisleri bütünleşimi üzerine, bazıları ise anlamsal web servisleri ile bütünleşim üzerine yoğunlaşmaktadır. Ayrıca mobil araçlardan servislere erişime yönelik çalışmalar da yine bu bölüm kapsamında incelenecektir. Son olarak tez çalışması literatür bağlamında değerlendirilecektir.

3.1 Web Servisleri ve Yazılım Etmenlerinin Bütünleşimi

Literatürde etmenler ve web servislerinin birlikte kullanımını sağlayan çeşitli çalışmalar bulunmaktadır. WSIG (Web Services Integration Gateway) (Greenwood and Calisti, 2004) adlı çalışmada hem bir JADE[10] etmeninin web servisini, hem de bir web servisinin bir JADE etmeninin servisini çağırabileceği bir mimari sunulmaktadır. Bu çalışmada sunulan ara katman ile web servis çağrımı için ve bir web servisinin bir etmen ile iletişiminin sağlanabilmesi için gerekli olan dönüşümler yapılmaktadır. Fakat böyle bir mimari iki taraflı dönüşümler gerektirdiği için işlem yükü çok fazla olan bir çalışma olarak ortaya çıkmaktadır. Önerilen mimarinin anlamsal web ortamını desteklemiyor oluşu ise bir eksiklik olarak göze çarpmaktadır.

Bir diğer çalışmada ise etmenler ve web servislerin birlikte kullanımı kapsayıcı (wrapper) etmenler ile sağlanmaya çalışılmıştır (Varga et al., 2003). Bu çalışmada iki yaklaşımdan bahsedilmektedir. İlk yaklaşımda, tek bir kapsayıcı etmenin web servislerinin çağrımından

---

10 JADE (Java Agent DEvelopment Framework) URL:http://jade.tilab.com, son erişim 16 Şubat 2008.

sorumlu olması önerilmektedir. Diğer etmenler de kapsayıcı etmenle iletişime geçerek web servislerini kullanabilmektedir. Diğer yaklaşımda ise; her bir web servisi ayrı bir kapsayıcı etmen üzerinden kullanılabilmektedir. Yapılan çalışma kullanılmakta olan web servislerinin etmen sistemleriyle bütünleşimi için iki farklı yaklaşımla çözüm sunması açısından faydalı bir çalışma olmakla birlikte yeterli bir çalışma değildir. Bunun nedeni çalışmanın web servislerinin anlamsal web ortamında etmenlerle bütünleşimine yönelik bir fikir sunmamış olmasıdır.

3.2 Anlamsal Web Servisleri ve Yazılım Etmenleri Bütünleşimi

Anlamsal web servisleri, ontolojilerin yayınlanan servis tanımlamalarına anlam yüklediği ve bu sayede servis istemcilerinin servis tanımlamalarını yorumlayıp, servisleri çağırabildiği web servisleridir (Burstein et al., 2005). Anlamsal web ise bu servislerin otomatik olarak bulunması, seçilmesi, çalıştırılması, karşılıklı işleyebilirliğinin ve izlenmesinin sağlanması için kullanılabilecek teknolojileri sunmaktadır. (Sycara et al., 2004) ‘ın yaptığı çalışmada servislerin keşif, anlaşma ve çalıştırma işlemlerinin etmenler tarafından otonom bir şekilde gerçekleştirilmesi için bir mimari sunulmaktadır. Bu mimari çok-etmenli sistemlerin kullanımını önermekte ve anlamsal web servisleriyle etmenlerin etkileşimi için aracı etmen (broker agent) yaklaşımını öne sürmektedir. Bu yaklaşıma göre aracı etmenin görevi; servis istemcisi ve servis sağlayıcısının mesajlarının birbirine çevrimi, istemci için uygun servis aranması, keşfi ve iletişim kurulması olarak tanımlanmıştır. Bu çalışmada oluşturulan mimari masaüstü sistemler için geliştirilmiştir. Fakat mobil araçlar için etmen servis etkileşimine yönelik bir aracı bileşen ile desteklenen bir mimari ortaya konmamıştır.

(Antoniou et al., 2007) ‘ın yaptığı başka bir çalışma da ise; aracılık (brokering) ve eşleştirme (matchmaking) sorunu üzerine çalışılmış ve istemcilerden gelen isteklerin ara etmenler tarafından nasıl ele alınacağına dair bir çözüm sunulmuştur. Sunulan çözümde etmenlerin ve servislerinin tanımlanması için RDF[11], etmen platformu için JADE çok-etmenli sistemi kullanılmıştır. Etmen servislerinin keşfi için ara etmen FIPA[12] soyut mimarisinde önerilmiş olan sarı sayfa dizininde anlamsal sorgulama yapmaktadır. Yine bu mimari de etmen servislerine mobil araçlardan nasıl erişileceğine dair bir çözüm sunmamaktadır.

(Gümüş et al., 2008) ‘ın sunduğu çalışma anlamsal web servislerinin etmen sistemlerde kullanımı için aracı etmen tasarımını tartışmaktadır ve aracı etmenle sistemdeki diğer etmenlerin etkileşimi için genişletilmiş bir protokol sunduğu için diğer çalışmalara göre üstünlük göstermektedir. Fakat yapılan aracı etmen tasarımında servis isteminde bulunan etmenin mobil araçta olması durumunda aracı etmenin mobil araç kısıtlarını da gözönünde bulundurup, servis isteğini nasıl ele alacağı durumu ile ilgili bir çözüm sunulmamaktadır.

3.3 Mobil Araç ve Servis Bütünleşimi

Son yıllarda mobil teknolojilerdeki gelişmelerin günden güne artması ile mobil araçlar için geliştirilen akademik çalışmalarda da artış görülmektedir. Yapılan çalışmalar masaüstü sistemlerin veya web servislerinin mobil araçlarla bütünleşimi üzerinde yoğunlaşmaktadır. Bu çalışmaların herbiri farklı yaklaşımlarla çözümler üretmeye çalışmaktadır. (Mary, 2006) yaptığı çalışmada mobil araçlarda web servisi kullanımını sağlamak için mobil etmen tabanlı bir çatı

---

[11] RDF Web Site : www.w3.org/RDF, son erişim 25 Nisan 2008.

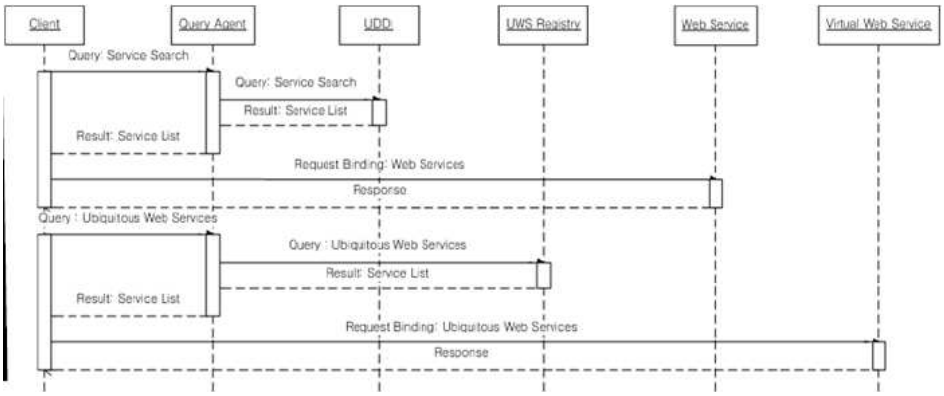
[12] FIPA Web Site:http://www.fipa.org, son erişim 12 Şubat 2008.

sunmaktadır. Sunulan mimaride kullanıcı ile etkileşimde bulunmak, web servislerini web sunucularından çalıştırmak, mesajlaşmayı yönetmek gibi görevler için ayrı ayrı etmenler bulunmaktadır. Bu etmenler kullanıcının mobil aracı ile web sunucu arasında göç etmek suretiyle web servislerinin kullanımını sağlamaktadır. Mimarinin mobil ortamlar için tasarlanmasına rağmen hem mobil etmenlerin mesajlaşması için, hem de web servisinin kullanımı için işlem yükünün oldukça fazla olduğu göze çarpmaktadır. Ayrıca mimari anlamsal web standartlarını desteklememektedir.

(Mahmoud et al., 2007) ‘ın yaptığı bir diğer çalışmada ise, mobil araçlardan web servislerinin kullanılabilmesi için SOAP mesajı hazırlama, XML çözümleme, UDDI kayıtçılarının sorgulanması gibi işlemler, MSM (Mobile Service Manager) adını verdikleri bir ara katman yazılımı ile gerçekleştirilmektedir. Bu çalışmada ayrıca WSDL genişletilerek mobil araca bağlı olarak servisi kullanacak araçta nasıl sunulacağı WSDLE dökümanları şeklinde tanımlanmaya çalışılmıştır. MSM gerçekleştirimi masaüstü sistemde yapılmış ve mobil araç, kukla uçbirim (dummy terminal) olarak tasarlanmıştır. Mimaride etmen tabanlı bir yaklaşım bulunmamakta ve anlamsal web teknolojileri kullanılmamaktadır.

(Hwang et al., 2007) ‘ın mobil araçların hem web servislerini, hem de mobil web servislerini kullanabilmesi için yaptıkları çalışmada; UWS (Ubiquitous Web Service) aracı adı verdikleri bir aracı bileşen ile servis isteği karşılanmaktadır. UWS aracı bileşeni sanal bir web servis yaratarak, UDDI kayıtçısına servisi kaydetmekte ve mobil araçtan gelen servis isteğini bu sanal servisi kullanarak gerçekleştirmektedir. Şekil 3.1’de bu mimarideki servis keşif diyagramı görülmektedir. Mobil istemci sorgu etmenini kullanarak UDDI kayıtçısını sorgulayabilmekte ve sorgu sonucunda dönen web servisini doğrudan kullanabilmetedir.

Fakat gezici bir web servisini sorgulamak istediğinde sorgusunu UWS aracısına göndermekte ve UWS aracısı yarattığı sanal web servisi üzerinden gezici web servis sonucunu mobil istemciye göndermektedir. Yapılan çalışma web servislerinin mobil araçlardan kullanımı sağlamakta yeterlidir. Ancak anlamsal web ortamına bir desteği olmaması bir eksiklik olarak bulunmaktadır.



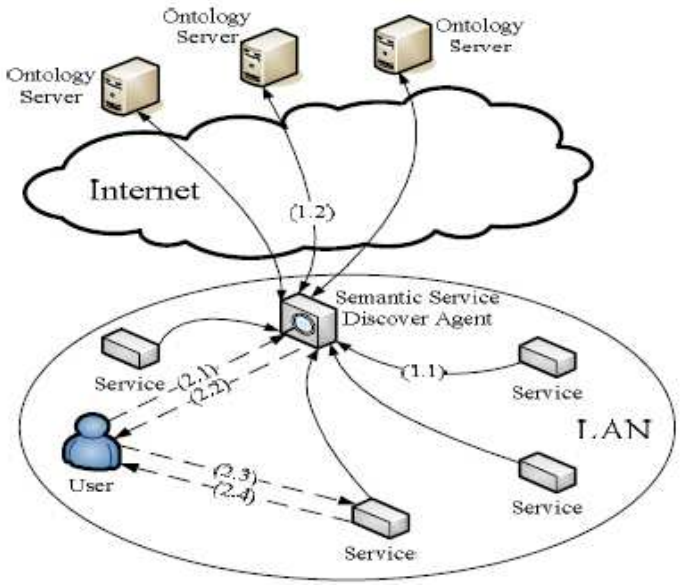
Şekil 3.1. UWS Aracı Sistemi Servis Keşif Diyagramı (Hwang et al., 2007)

(Ratsimor et al., 2004) 'ın yaptığı çalışmada ise Allia adında etmen servislerinin keşfine yönelik etmen tabanlı bir çatı sunulmuştur. Bu sistemde JADE-LEAP[13] kullanılmış ve mobil araçtaki etmen çevresinde bulunan mobil araçlardaki diğer etmenlerin kendisine kayıtlanması sonucu *Alliance* adı verilen kendisine ait çevresini oluşturmaktadır. Mobil araçta bir servis isteği oluştuğu zaman, etmen önce kendi *Alliance*'ında bulunan etmen servislerini sorgulamaktadır. Eğer çevresindeki etmenlerde bu servisi veren bir etmen yoksa, *Alliance*'da

[13] JADE-LEAP URL : http://www.ryerson.ca/~dgrimsha/jade/index.html, son erişim 18 Şubat 2008.

bulunan her bir etmen kendilerine ait *Alliance*'ı sorgulayarak, ilgili servisi bulunmaya çalışılmaktadır. Bu mimaride kullanılmak istenen servisler web servislerini içermemektedir. Yapılan çalışma mobil araçta bulunan etmenlerin sunduğu servislerin etmen tabanlı bir mimari ile kullanılmasını hedeflemektedir. Çalışma anlamsal web ortamını ve web servis kullanımını desteklememektedir.

(Wang et al., 2007) tarafından gerçekleştirilen bir başka çalışmada yine araçların sunduğu servislerin keşfi ve bu servislere erişim sağlanmaya çalışılmıştır. Bu çalışmanın (Ratsimor et al., 2004) tarafından yapılan çalışmadan farklı ve de üstün tarafı ise, servislerin anlamsal olarak tanımlanması olarak ortaya çıkmaktadır. Bu çalışmada mobil istemciler servis isteğini anlamsal servis keşif etmenine iletmektedirler. İstenilen servisin keşfi bu etmen tarafından yapılmakta ve dönen servis ile mobil istemci iletişim kurmaktadır. Çalışmada sunulan mimari Şekil 3.2'de görülmektedir.



Şekil 3.2. Anlamsal Servis Keşif Mimarisi (Wang et al., 2007)

Mobil araçlar için geliştirilen uygulamalarda mobil araçlara özgü kısıtlar gözönüne alınması gerektiği için uygulama geliştirmek masaüstü sistemler için yapılan çalışmalara göre zorluklar içermektedir. (Veijalainen et al,. 2006) ‘ın yaptığı çalışma bu bölümde anlatılan çalışmalara göre farklılık göstermektedir. Çalışmada ASG (Adaptive Service Grid) adı verilen ve anlamsal servis platformları için bir referans mimari sunan projenin mobil araçlarla bütünleşimini sağlayacak mimarinin tasarımında gözönüne alınması gereken kısıtlar tanımlanmış ve geliştirilebilecek olası dört mimari, getireceği avantaj ve dezavantajlarıyla açıklanmıştır. Tanımlanan mimariler aşağıda özetlenmektedir :

- Kukla Uçbirim (Dummy Terminal) : Mobil istemci, masaüstü bir sistem olan anlamsal web servis platformuna mobil araçtaki ince istemci (thin client) katman ile erişebilir.
- Hedef Asistanı içeren Uçbirim : Mobil istemciden gelecek servis isteği mobil araçtaki hedef asistanı tarafından anlamsal bir hedefe dönüştürülmektedir. Anlamsal hedefe göre masaüstü sistemde bulunan anlamsal web servis platformunda servis keşfi yapılmaktadır.
- Çalışma Akışı (Work-Flow) Çalıştırabilen Uçbirim : Bu mimaride anlamsal hedefe uygun olarak bulunan servislerin çalışma akışı anlamsal servis platformu tarafından mobil araca gönderilir. Mobil araçta servisleri kendisine gelen çalıştırma akışına göre kullanır.
- Tam İşlevsel Mobil Anlamsal Web Servis Platformu : Platformdan beklenen servis keşif, anlaşma ve çalıştırma işlemlerinin tümünün mobil araçta gerçekleştirildiği bir

mimaridir. Bu mimarinin gerçekleşmesi günümüzdeki mobil araçların kısıtlı işlem kapasitesi gözönüne alındığında olası görünmemektedir. Bunun nedeni tam işlevsel bir anlamsal servis platformundan beklenen ontoloji yönetimi ve çıkarsama işlemlerinin mobil araçların kapasitesinin üstünde olduğu içindir.

Çalışmada olası mimarilerin analizinden sonra önerilen gerçekleştimi en uygun mimaride kısıtlı işlevselliği olan bir mimaridir. Önerilen mimaride çıkarsama ve ontoloji işlemleri bulunmamaktadır. (Veijalainen et al,. 2006)'nin yaptığı çalışma gerçekleştirim içermemesine rağmen, yapılması düşünülen bir mobil anlamsal servis platformunun tasarımı için detaylı analiz içermesi nedeniyle bu tür bir sistem geliştirmeyi planlayan araştırmacılar için referans alınabilecek bir çalışmadır.

3.4 Değerlendirme

Bu bölümde tez çalışması ile ilgili literatür çalışmaları incelenmiştir. Bu çalışmalar kullanılan teknolojilere göre üç farklı bölüm içinde tanıtılmıştır. Gerçekleştirilen tez projesi, etmen tabanlı yazılım geliştirme teknikleri ile mobil araçlardan anlamsal web servislerine erişimi sağlamaya çalışarak bu teknolojilerin tek bir çalışma altında kullanılmasını sağlaması açısından incelenen çalışmalara göre farklılık göstermektedir.

(Veijalainen et al,. 2006)'nin de önerdiği gibi tez projesinin mimarisinde mobil araç kısıtlarına uygun olarak mobil araç üzerinde kısıtlı işlevsellik sağlanmıştır. Tez çalışmasında gerçekleştirilen mimaride (Veijalainen et al,. 2006) değerlendirdiği farklı mimari tasarımlarına ek olarak etmen tabanlı bir sistem oluşturulmuştur. Bu sayede; açık, dinamik ve heterojen ortamlarda kullanıcısı adına özerk bir

biçimde bilgi arama ve bütünleştirmeyi sağlayabilen etmen teknolojisinin avantajlarından mobil araç kullanıcılarının da yararlanması mümkün olmaktadır.

Tez çalışmasında gerçekleştirimi yapılan mimari anlamsal web ortamı için tasarlandığından sistemde bulunan etmenlerin anlamsal veri işlemeleri gerekmektedir. Anlamsal veri işleme işlevinin mobil araçlardan gerçekleştirilememesinin bir sonucu olarak bu işlemler masaüstü sistemde bulunan arabulucu etmen tarafından gerçekleştirilmektedir. Bu sayede mimarinin işlevselliği dağıtılarak uygulama mobil araç kısıtlarına uygun hale getirilmiştir.

4. KULLANILAN ETMEN PLATFORMLARININ TANITILMASI

Bu bölümde, yapılan tez çalışmasının gerçekleştiriminde kullanılan etmen platformları tanıtılacaktır. İlk olarak masaüstü sistemler için geliştirilmiş olan SEAGENT çok-etmenli platformu, ardından mobil araçların SEAGENT platformuyla bütünleşimi sağlayan SeagentLITE etmen platformu mimarileri açıklanacaktır.

4.1 SEAGENT Çok-Etmenli Sistem Geliştirme Platformu

Hedefe dayalı çok-etmenli sistem geliştirme platformu olan SEAGENT (Dikenelli et al., 2005a), Ege Üniversitesi Çok-Etmenli Sistemler araştırma grubu tarafından geliştirilmiştir. SEAGENT projesi 2002 yılında geliştirilmeye başlanmış, 2006 yılına kadar olan süreç içinde de yavaş yavaş değişime uğrayarak, bugünkü mimarisine ulaşmıştır. SEAGENT çok-etmenli platformunu benzeri çalışmalardan ayıran özellikleri aşağıda sıralanmaktadır:

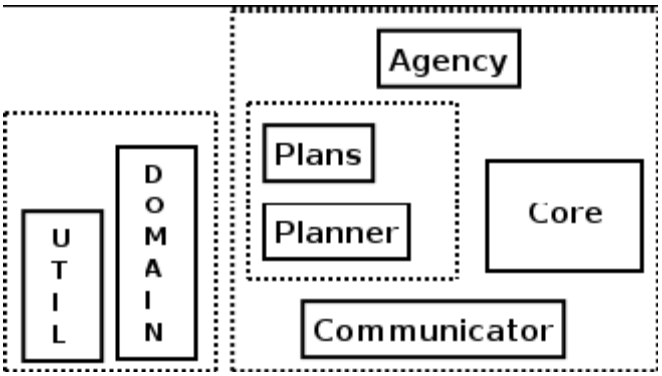
- SEAGENT platformu içerisinde yaratılan etmenler kendilerine ait bilgi tabanlarını anlamsal web standartlarına göre yönetebilmektedirler. Ayrıca; platform, içsel bilginin herhangi bir uygulama programlama arayüzüne (API) bağlı kalmaksızın sorgulanıp, yönetilebilmesi için özel olarak tasarlanmış arayüzler sunar.
- SEAGENT dizin servisleri bilgiyi anlamsal web standartlarına göre saklamakta, aynı zamanda benzer özelliklere sahip etmenleri anlamsal olarak eşleyebilmektedir.

- Etmenler arası iletişimde anlamsal içeriğin taşınması için FIPA-RDF[14] içerik dili kullanılmaktadır.
- SEAGENT platformu ontoloji yönetimi için yeni bir servis sunmaktadır. Bu servis, platform ontolojileri ile dışsal(external) ontolojilerin birbirine eşlenmesini sağlamaktadır.
- SEAGENT platformu anlamsal web servislerinin dinamik olarak keşfini ve çağrımını sağlamaktadır.

Şekil 4.1'de görülen SEAGENT sürüm 0.02 mimarisi genel olarak özetlenecek olursa; mimaride bulunan en alt katman olan *Communicator* paketi platform iletişimini yönetmekten sorumludur. SEAGENT, etmenlerin mesajlaşmasını sağlamak için FIPA'nın tanımlamış olduğu etmen iletişim soyut mimarisini gerçekleştirmiştir. Platformdaki diğer bir katman platformun temel işlevlerini gerçekleştirmesini sağlayan paketleri içermektedir. *Domain* ve *Util* paketleri bu amaçla oluşturulmuş paketlerdir. *Agency* paketinde etmen platformunun ve bir etmenin içsel işlevlerinin gerçekleştirimi sağlanmaktadır. Bir etmenin kendine ait görevlerini yerine getirebilmesi için HTN (Hierarchical Task Network) formalizmine göre tanımlanmış planları kendine ait planlayıcısıyla çalıştırması gerekmektedir. *Core* paketinde ise FIPA soyut mimarisinde bulunan çok-etmenli platform servislerinin gerçekleştirimi yapılmıştır. Bu servislerin gerçekleştiriminde anlamsal web teknolojilerinden faydalanılmıştır. Dizin kolaylaştıcı (Directory Facilitator - DF) servisi, etmen yeteneklerini bir OWL[15] ontolojisinde saklamakta ve sorgulanan yeteneklere sahip benzer etmenleri içerdiği anlamsal eşleme motoruyla

[14] FIPA-RDF URL : http://www.fipa.org/specs/fipa00011/XC00011B.html, son erişim 29 Nisan 2008.
[15] OWL (Web Ontology Language) URL : http://www.w3.org/2004/OWL/, son erişim 30 Mart 2008.

bulmaktadır. Etmen yönetim sistemi (Agent Management System - AMS) ise, etmenlerin tanımlamalarını yine dizin kolaylaştırıcıya benzer şekilde etmen yönetim ontolojisine kaydetmektedir. Platformda bulunan herhangi bir etmene ait bilgiler bu ontolojinin anlamsal olarak sorgulanması ile elde edilmektedir.

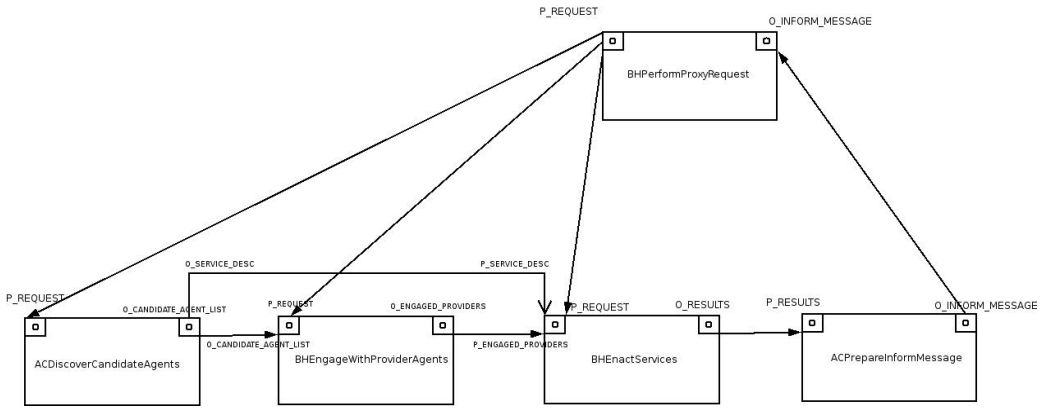


Şekil 4.1. SEAGENT sürüm 0.02 mimarisi

Anlamsal web teknolojilerinin kullanılarak gerçekleştirimi yapılan temel platform servislerinin yanı sıra, anlamsal web servislerinin kullanımını sağlamak için tanımlanmış olan SWSA(Semantic Web Services Initiative Architecture) kavramsal modeli kapsamında SEAGENT platformu için gerekli servislerin bütünleşimi yapılmıştır (Gümüş et al., 2008). SWSA kavramsal modelinin gerçekleştirimi sonucunda SEAGENT çok-etmenli platforma aracı etmen, servis sağlayıcı etmen, ontoloji etmeni ve servis istemci etmenler eklenmiştir:

Aracı(Broker) Etmen, FIPA'da tanımlı sarı sayfa hizmeti veren dizin kolaylaştırıcıya benzer bir işleve sahiptir. Aracı etmen platformdaki anlamsal olarak tanımlanmış servislerin sağlayıcıları ve istemcileri arasında hem keşif hem de aracılık işlevlerini yerine getirir. Bu işlevleri HTN formalizmine göre oluşturulmuş BHPerfomProxyRequest planı ile

gerçekleştirmektedir. Bu plan istemci etmenlerden gelen servis çağırım isteklerine göre uygun servis sağlayıcı etmenleri bulur, onlarla anlaşır ve uygun olanını çalıştırıp, sonucu istemci etmene gönderir. ACDiscoverCandidateAgents, BHEngageWithProviderAgents, BHEnactServices ve ACPrepareInformMessage planları bu planın alt görevilerini oluşturur. Bu plan ve alt görevler arasındaki ilişki Şekil 4.2'de gösterilmektedir.



Şekil 4.2. Aracı etmenin aracılık planı

ACDiscoverCandidateAgents alt görevi gelen istek mesajındaki "ServisDescription" nesnesine göre kendine kayıtlı servis sağlayıcı etmenler içinde anlamsal arama yapar ve uygun servis sağlayıcı etmenleri bulur. BHEngageWithProviderAgents alt görevi bulunan servis sağlayıcı etmenlere anlaşma mesajı gönderir ve bu etmenlerden gelen cevap mesajlarını değerlendirir. BHEnactServices, anlaşılan servis sağlayıcı etmenlere servis kullanma isteğini içeren mesaj gönderir. Son olarak, ACPrepareInformMessage, servis sağlayıcı etmenden gelen sonucu istemci etmene gönderir.

Servis sağlayıcı etmen, kendisine ait servislerin sunumundan sorumludur. Servis sağlayıcı etmenler, kimi zaman kendi içsel yeteneklerini aynı anlamsal web servislerinde olduğu gibi anlamsal tanımlamalar aracılığıyla sunabildikleri gibi, dışsal web servislerinin etmen platformuna dahil edilmesini sağlayan geçit etmeni görevini de yerine getirebilmektedirler.

Ontoloji etmeni, platformda kullanılan ontolojilerin depolandığı bir ontoloji havuzu içerir. Ontoloji etmeni, platformun diğer üyelerinin bu ontolojilere kontrollü bir şekilde erişimini ve sorgulamasını sağlar.

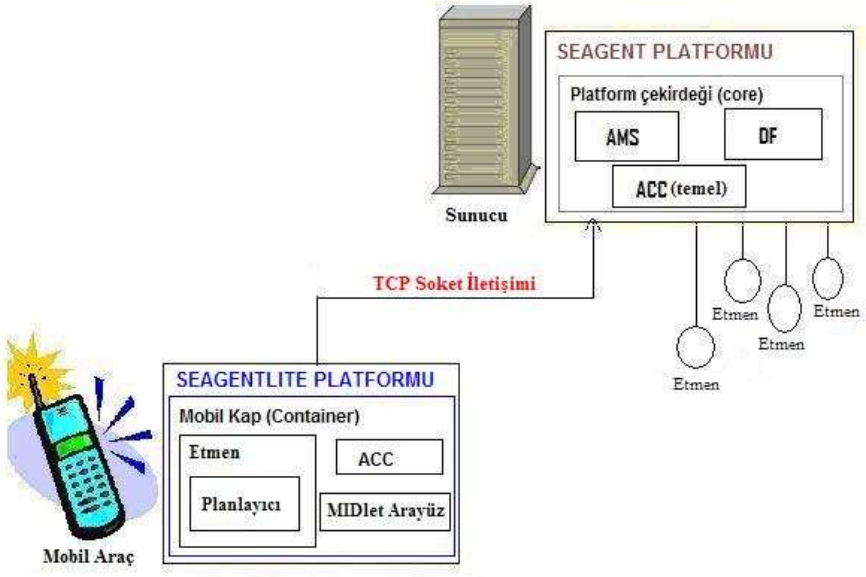
Servis istemci etmen, amaçlarına ulaşmak için diğer etmenlerin servislerini kullanma isteminde bulunan etmendir. Bir servis istemci etmen herhangi bir servise ihtiyaç duyduğunda, öncelikle platformdaki hedef şablonlarından isteğine uygun bir tanesinden örnek hedef yaratarak servis isteğini tanımlar. Daha sonra hedefini başarmak için, bu hedefe uygun servisin/servislerin keşif, seçim ve işletimini girdi parametreleri ile birlikte aracı etmene havale eder. İstemci etmen ve aracı etmen arasında servis kullanım etkileşimi için yapılan mesajlaşmalarda da yine ontolojiler kullanılmaktadır.

SEAGENT çok-etmenli platformda bahsi geçen servis istemci etmen ile aracı etmen arasındaki etkileşimin gerçekleşebilmesi istemci etmenin masaüstü sistemde bulunması gerekmektedir. Mobil araçlarda bulunan etmenler için ontoloji kullanımının gerçekleştirilmesi için gerekli olan işlemlerin yerine getirilebilmesi, günümüz teknolojilerinde özellikle başarım açısından değerlendirildiğinde, mümkün olmamaktadır. Bu nedenle Bölüm 5.1’de mobil araçtaki bir istemci etmenin ilgili servis etkileşiminde bulunabilmesi için SEAGENT platformundaki düzenlemelerden bahsedilecektir.

4.2 SeagentLITE Etmen Platformu

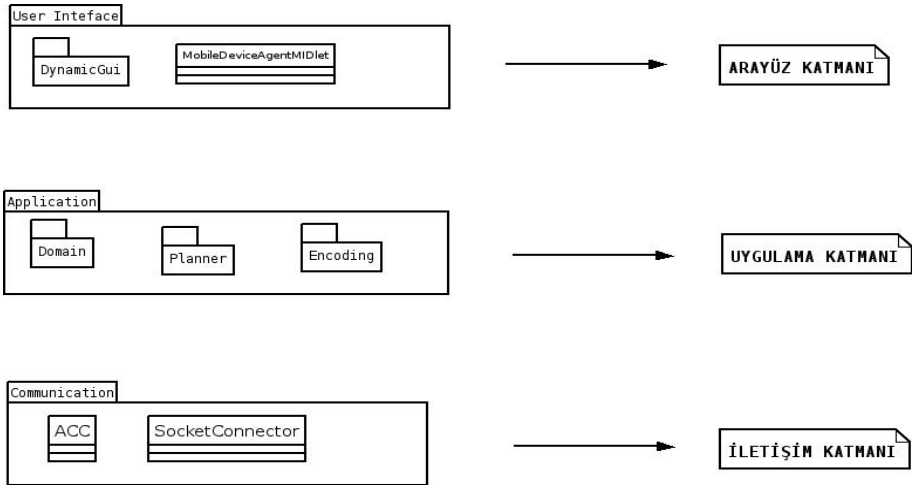
Mobil teknolojilerin giderek yaygınlaşması, mobil araçlar için geliştirilen uygulamaların da artmasını sağlamıştır. Bu tez kapsamında etmen teknolojisinin mobil ortamlara taşınması amacıyla SEAGENT (Dikenelli et al., 2005b) etmen platformuna mobil ortamlarla bütünleşimini sağlamak için gerekli eklentiler yapılarak SeagentLITE mobil etmen platformu geliştirilmiştir. Geliştirilen platform aynı zamanda mobil araçtaki etmenlerin anlamsal web tabanlı çok-etmenli platformla bütünleşimini sağlayan bir mimari sunmaktadır. SeagentLITE projesi bu özelliği ile SEAGENT platformunun mobil araçlarda kullanılmasını sağlayarak SEAGENT platformunun kullanım alanını genişletmektedir.

SEAGENT ve SeagentLITE platformları arasındaki etkileşim Şekil 4.3'de görülmektedir. Şekilde de görüldüğü gibi SEAGENT etmen platformu Etmen Yönetim Sistemi (Agent Management System – AMS), Dizin Kolaylaştırıcı (Directory Facilitator - DF) ve Etmen İletişim Kanalı (Agent Communication Channel-ACC) bileşenlerini içermektedir. AMS platforma kayıtlı tüm etmenlerin bilgilerini tutan dizindir. DF ise platforma kayıtlı etmenlerin sundukları servisleri içeren bir sarı sayfa dizinidir. Etmen İletişim Kanalı ise etmenler arası iletişimi sağlayan bileşendir. Bu üç bileşen SEAGENT platformunun çekirdek mimarisini oluşturmaktadır. Platform içerisinde şekilde de görüldüğü gibi kayıtlı pek çok etmen bulunmaktadır. SeagentLITE platformu etmenler arası iletişimi sağlamak için Etmen İletişim Kanalı bileşeni(ACC), mobil araç kullanıcısı ile etkileşimi sağlamak için MIDlet arayüz ve mobil araç içindeki etmeni ve etmenin çalışmasını sağlayan planlayıcıyı bulunduran Mobil Kap (MobileContainer) bileşenini içermektedir.



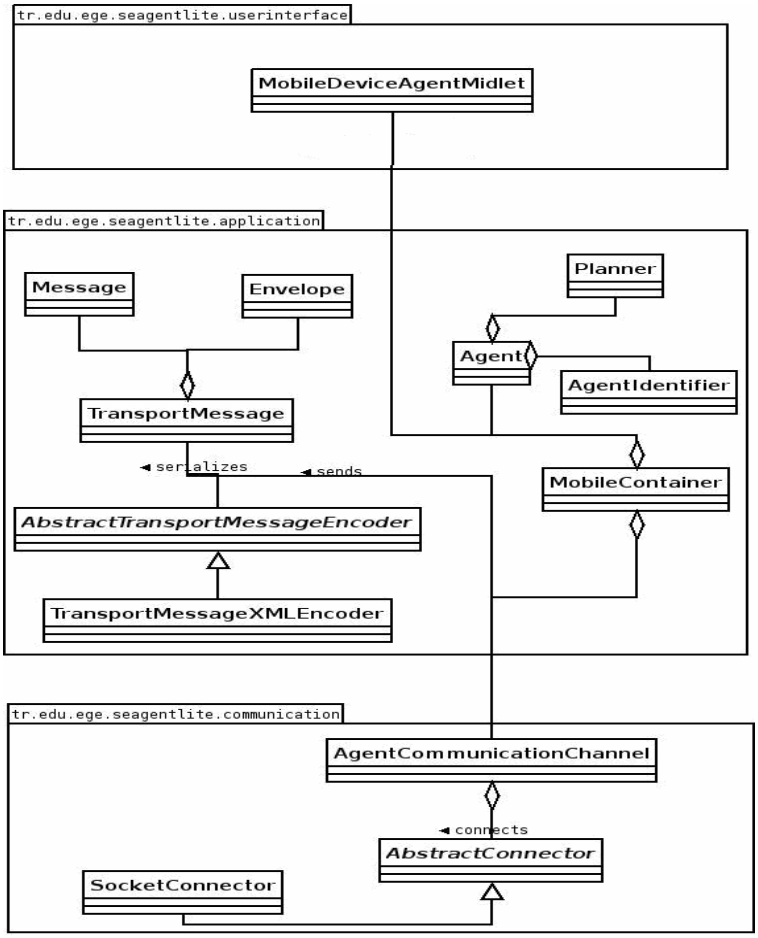
Şekil 4.3. Platformlar arası etkileşim diyagramı

SeagentLITE mimarisinin katmanlı yapısı Şekil 4.4'te görülmektedir. En üst katman olan arayüz katmanı, kullanıcı ile mobil araçtaki etmenin iletişime geçmesini sağlayan dinamik arayüzleri sunar. Orta katman olan uygulama katmanı ise üç alt paketten oluşur. Domain paketi bir etmen platformunda bulunması gereken temel sınıfları içerir. Örneğin; etmenin temel özelliklerini tanımlayan AgentIdentifier sınıfı domain paketi içindedir. Planner paketi, bir etmenin görevlerini yerine getirmesi için oluşturulan planların çalıştırılmasını sağlayacak sınıfları içerir. Encoding paketi, iletişim sırasında aktarılan mesajların serileştirilmesi için gerekli sınıfları içerir. Son katman olan iletişim katmanı, SEAGENT platformuyla mobil araçtaki etmenin anlaşabilmesi için gerekli iletişim altyapısını oluşturur.



Şekil 4.4. SeagentLITE katmanlı mimari

SeagentLITE projesinde katmanlı yapıyı gerçekleştirmek amacıyla her katman için ayrı bir paket oluşturulmuştur. Şekil 4.5'te bu paketlerlerde yer alan bazı sınıflar arasındaki ilişkiler gösterilmektedir. İzleyen bölümlerde her bir katman detaylı bir şekilde anlatılacaktır.



Şekil 4.5. SeagentLITE sınıf diyagramı

4.2.1 İletişim Katmanı

Mobil araçtaki etmenin diğer etmenlerle iletişim kurabilmesi için iletişim katmanı oluşturulmuştur. Etmenler birbirleriyle mesajlaşma yoluyla iletişim kurarlar. Literatürde bulunan en yaygın etmen iletişim dilleri KQML (Finin et al., 1995) ve FIPA ACL[16] etmen iletişim

[16] FIPA Etmen İletişim Dili Tanımları (FIPA Agent Communication Language Specifications), http://www.fipa.org/repository/aclspecs.html, son erişim 12 Şubat 2008.

dilleridir. SeagentLITE etmen platformu FIPA[17] uyumlu bir etmen platformu olduğu için FIPA ACL etmen iletişim dili kullanılmaktadır.

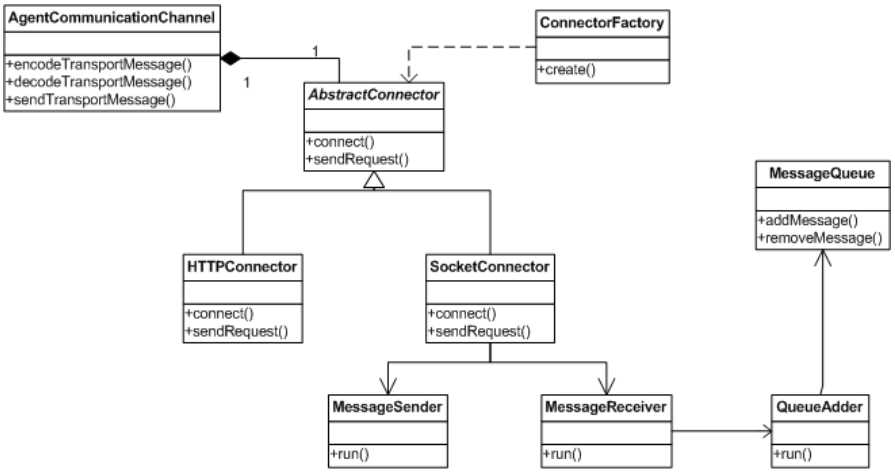
SeagentLITE projesinin iletişim katmanı Şekil 4.6'te görüldüğü gibidir. Bu katmanın mesajların iletimi için farklı iletişim altyapılarını desteklemesi hedeflenmiştir. Bu yapıyı gerçekleştirmek için Factory (Gamma et al., 1995) tasarım deseni kullanılmıştır. Her bir iletişim türü için geliştirilecek olan sınıf AbstractConnector soyut sınıfından türetilmelidir. Seçilen iletişim tipine göre bu sınıfların yaratılmasından ConnectionFactory sınıfı görevlidir. TCP soketleri ile iletişim altyapısını sağlayan SocketConnector sınıfıdır. İletişimde TCP soketleri UDP soketlerine göre daha yavaş iletişim sağlasa da gönderilen mesajların iletimini garantilemesi nedeniyle SeagentLITE platformunda bu iletişim tipi tercih edilmiştir[18].

İletişim katmanında mobil araçtaki etmen iletişim kanalı (Agent Communication Channel – ACC) ile sunucuda bulunan ana etmen iletişim kanalı (Main Agent Communication Channel – MainACC) arasındaki mesaj iletimi sağlanır. TCP soketleriyle iletişim için ana etmen iletişim kanalında bir sunucu soket (Server Socket) açılır. Mobil araçta bulunan etmen iletişim kanalı bu sunucu soketin adresinden haberdardır. Sunucu taraftaki etmen platformuyla iletişim kurulmak istendiğinde mobil araçtaki etmen iletişim kanalı sunucu sokete bağlanmak için SocketConnector sınıfı içinde bir istemci soket oluşturur. Mesajlaşma bu soket üzerinden gerçekleştirilir.

---

[17] FIPA Web Site:http://www.fipa.org, son erişim 12 Şubat 2008.

[18] J2ME Low-Level Network Programming with MIDP 2.0, http://developers.sun.com/mobility/midp/articles/midp2network/, son erişim 30 Mayıs 2008.



Şekil 4.6. İletişim katmanı sınıf diyagramı

İletişim sırasında farklı görevlerin gerçekleştirilmesi için farklı iş parçacıklarına (thread) gereksinim duyulmuştur. Gelen mesajların alınması için MessageReceiver sınıfı, mesaj gönderilmesi için MessageSender sınıfı ve gelen mesajların kuyruğa eklenebilmesi için QueueAdder sınıfı bulunmaktadır. Bu sınıflar görevlerini yerine getirebilmek için iş parçacıklarını kullanırlar.

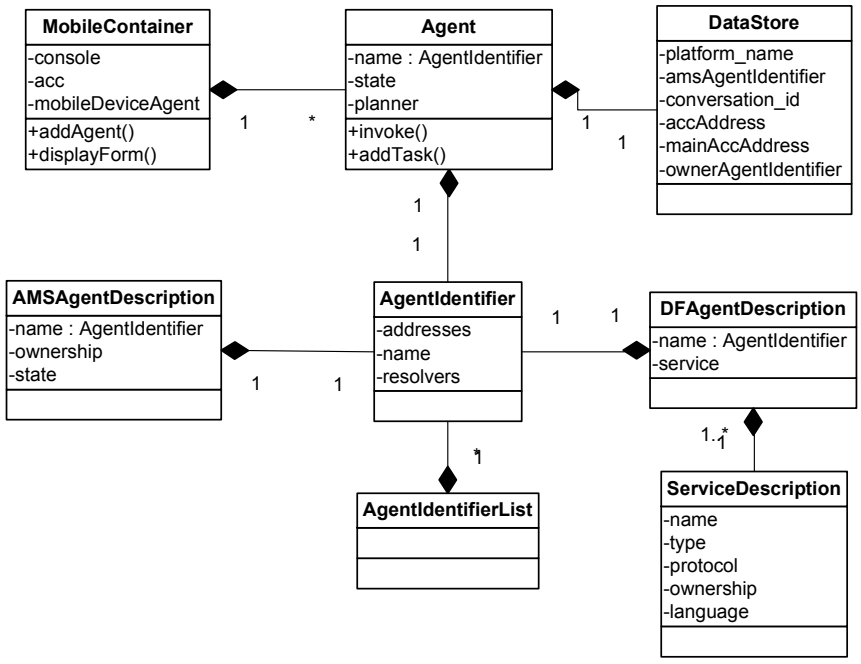
4.2.2 Uygulama Katmanı

Bu katmanda SeagentLite projesinde ihtiyaç duyulan temel uygulama sınıfları bulunmaktadır. Şekil 4.5'te görülen `tr.edu.ege.seagentlite.application` paketi içinde bu katmana ait bazı sınıflar görülmektedir. Bu sınıflar işlevlerine göre alt paketlere bölünmüştür. Domain paketi, planner paketi ve nesne serileştirme paketi uygulama katmanını oluşturmaktadır. Bu bölümde bu katmana ait alt paketler detaylı anlatılacaktır.

Yazılım etmenleri bir ortamda sürekli ve özerk bir biçimde işleyiş gösteren ve çoğunlukla diğer etmenler ile etkileşim içinde çalışan

yazılımlardır. Şekil 4.7' deki Agent sınıfı tanımlanan etmen özelliklerinin somut olarak gerçekleştirimini sağlayan sınıftır. Bu sınıfta etmenin ismi, durumu ve görevlerini yerine getirebilmesi için bir planlayıcı bulunmaktadır. Etmenin planlayıcı proje içinde ayrı bir paket olarak düzenlenmiştir. Bir etmenin durumu yaşam döngüsü boyunca INIATED, ACTIVE, UNKNOWN olabilmektedir. Etmen ismi, AgentIdentifier sınıfından bir nesneyle ifade edilir. AgentIdentifier sınıfı içinde etmeni global olarak tanımlayacak sembolik bir isim ve etmenle iletişime geçilebilecek adresler bulunmaktadır. Ayrıca bu pakette etmene özel bilgileri tutan DataStore sınıfı da bulunmaktadır. Bu sınıf içerisinde bağlanılan SEAGENT platformunun adı, adresi, mobil araçtaki etmenin bilgileri ve etmenin gönderdiği mesajların diyalog tanımlayıcısı bulunmaktadır. DataStore sınıfında bulunan bu bilgiler mobil araçtaki etmenin diğer etmenlerle iletişime geçmek istediğinde gereksinim duyacağı bilgilerdir. Projenin ilerleyen aşamalarında bu sınıfın sunucu tarafında bulunan bir çıkarsama servisi tarafından kullanılan bir bilgi tabanına dönüştürülmesi hedeflenmektedir.

SeagentLITE projesinde arayüz katmanıyla uygulama katmanının birbirinden soyutlanması için MobileContainer sınıfı oluşturulmuştur. Ayrıca mobil araçtaki etmen de MobileContainer içinde bulunmaktadır. Bu etmen diğer etmenlerle iletişim kurabilmek için Şekil 4.4’te en alt katmanda görülen Etmen İletişim Kanalı (AgentCommunicationChannel) bileşenini, mobil araç kullanıcısıyla etkileşimde bulunmak için en üst katman olan arayüz katmanında görülen “AgentMIDlet” arayüz bileşenini ve amaçlarını yerine getirebilmek için bu katmandaki planlayıcı bileşenini kullanmaktadır. Etmenin gereksinim duyduğu tüm bileşenler “MobileContainer” sınıfı içinde bulunmaktadır.



Şekil 4.7. Domain paketi sınıf diyagramı (1)

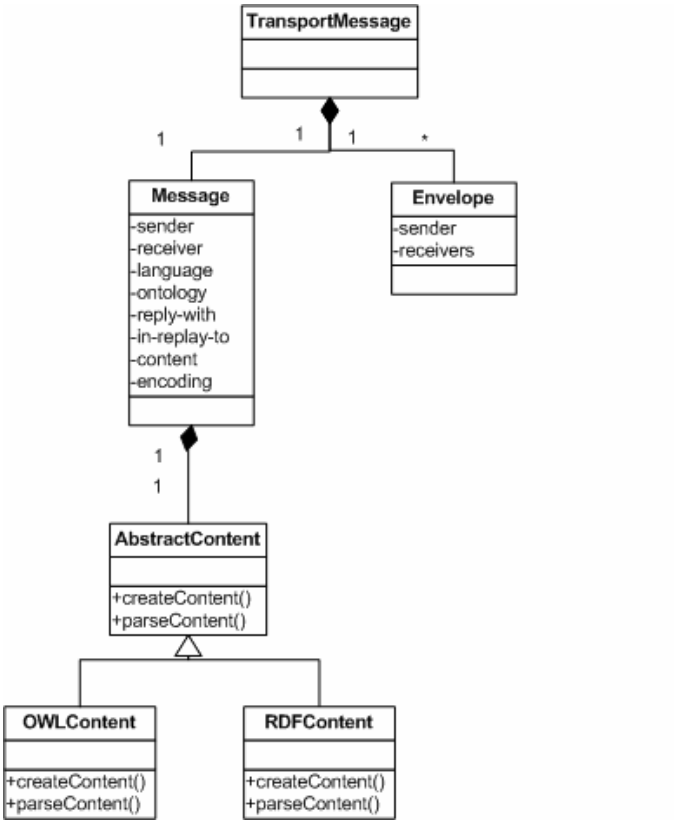
Mobil araçtaki etmenin, sunucu taraftaki SEAGENT çok-etmenli platforma dahil olabilmesi için kendini platformda bulunan etmen yönetim sistemine kaydettirmesi gerekir. Kayıt işlemi için etmen kendini AMSAgentDescription sınıfından yaratılan bir nesne ile tanımlar. Bu nesne içinde etmenin tanımlayıcısı (Agent Identifier), etmenin sahibi ve etmenin durumu bulunur. Mobil Platform içindeki etmenler servis sunuyorlarsa; servislerinin diğer etmenler tarafından kullanılabilmesi için Dizin Kolaylaştırıcıya (Directory Facilitator-DF) servislerini kaydettirmeleri gerekmektedir. Bu işlem için etmen kendini ve verdiği servisleri DFAgentDescription sınıfı ile tanımlamaktadır. Bu tanımlama içinde etmenin sunduğu her servis “ServiceDescription” sınıfı ile ifade edilmektedir.

Etmenler birbirleriyle FIPA ACL mesajlarını kullanarak iletişim kurarlar. Şekil 4.8'de görüldüğü gibi domain paketinde FIPA ACL etmen iletişim diline uygun mesaj yapısını temsil eden sınıflar bulunmaktadır. Etmenler arasında aktarılan mesajlar TransportMessage sınıfı nesneleridir. Bu sınıf içinde, Envelope ve Message sınıflarının nesneleri bulunmaktadır. Envelope sınıfı içindeyse, gönderici ve alıcı adresleri bulunmaktadır. Message sınıfı da gönderici ve alıcı tanımlamalarından, içerik dilinden, kullanılan ontolojilerden ve mesajın içeriğinden oluşur. Mesaj içeriği farklı içerik dilleriyle ifade edilebilir. Bu nedenle soyut bir içerik sınıfı olan AbsractContent sınıfı bulunur. Bu sınıftan türetilecek yeni sınıflarla farklı içerik dilleri desteklenebilir. SeagentLITE platformunda RDF içerik dili[19] desteklenebilmesi için RDFContent sınıfı kullanılmaktadır.

SeagentLite etmen çerçevesinde bir etmen, planlayıcısının plan kütüphanesindeki planları uygun zaman ve koşulda tek başına veya diğer planlarla birlikte kullanarak çalışmaktadır. Şekil 4.9'da görülen SeagentLITE projesinin planlayıcısı, HTN planlama (Erol et al., 1994) formalizmine dayanmaktadır. HTN planlama, planları görev ayrıştırımıyla yaratan bir yapay zeka planlama metodolojisidir. Bu ayrıştırma süreci, planlama sistemi doğrudan çalıştırılabilen ilkel görevler buluncaya kadar devam eder. SeagentLITE planlayıcının temel görevleri, gelen/giden mesajları işlemek, hedefe uygun planı bulmak, HTN görevlerini zamanlamak ve çalıştırmaktır. HTN planlayıcı temel olarak Planlayıcı (Planner), İşletici (Executor), Zamanlayıcı (Scheduler) ve döngüsel sistem planı (BHSystem) işlevsel modüllerinden oluşur.

---

[19] FIPA RDF İçerik dili www.fipa.org/specs/fipa00011/XC00011B.html, son erişim 12 Şubat 2008.



Şekil 4.8. Domain Paketi Sınıf Diyagramı(2)

Planlayıcıda etmenin görevleri soyut Task sınıfı ile temsil edilmektedir. Bu görevler karmaşık görevler veya ilkel görevler olabilmektedir. Karmaşık görevler için Behaviour sınıfı, ilkel görevler içinde Action sınıfı Task sınıfından türetilerek oluşturulmuştur. Action sınıfı ilkel görevlerin çalıştırılmasını sağlayacak metotları içeren sınıftır. Karmaşık bir görev birden çok karmaşık veya ilkel görevden oluşabilir. Behaviour sınıfı bu alt görevler arasındaki ilişkileri tutar.

Planlayıcı çalışırken beş tane kuyruk veri yapısı kullanır. İlki SeagentLITE etmen platformuna gelen ACL mesajlarını tutan gelen

mesaj kuyruğudur. İkincisi, platform dışına gönderilecek olan ACL mesajlarını tutan giden mesaj kuyruğudur. Üçüncüsü ise, etmenin gerçekleştirmesi gereken hedeflerini tutan hedefler kuyruğudur. Dördüncüsü, etmenin hedeflerini yerine getirmek için seçilen planların ve ihtiyaçları karşılanmamış görevlerin bulunduğu bekleyen görev kuyruğudur. Sonuncusu ise çalışmaya hazır eylemlerin yer aldığı hazır eylem kuyruğudur.

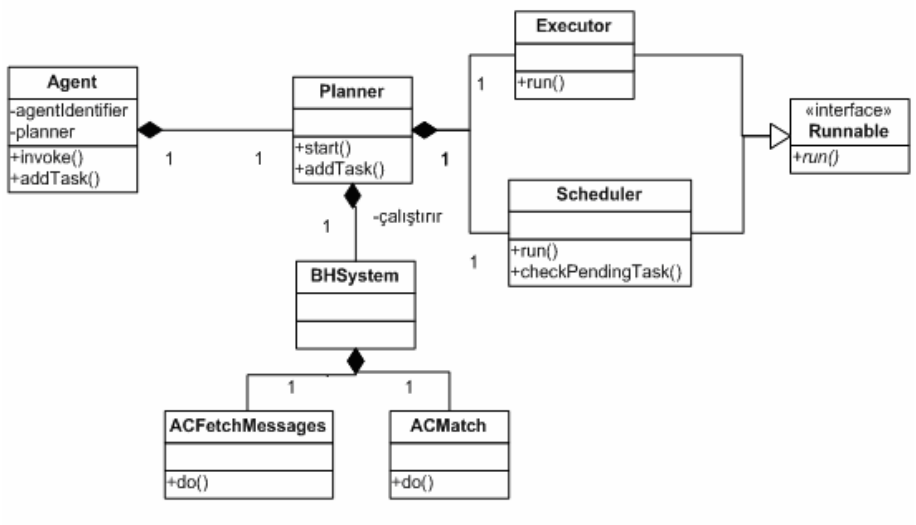
SeagentLITE mimarisinde planlayıcı, etmen içinde gömülü bir bileşen olarak tasarlanmıştır. Etmenin, görevlerini yerine getirebilmesi için Şekil 4.9'de görülen planlayıcı bileşenini kullanması gerekmektedir. Bunun için öncelikle etmen Planner sınıfının start() metodu kullanılarak planlayıcıyı başlatır. Planlayıcı start() metodunda ilk önce eş zamanlı olarak çalışan işletici ve zamanlayıcı bileşenlerini başlatır, daha sonra da döngüsel sistem planını çalıştırır. Planlayıcı başladıktan sonra planlayıcıya görev eklemek için addTask() metodu kullanılır.

İşletici, çalışmaya hazır olan eylemleri çalıştıran, zamanlayıcı ise checkpendingTask() metodu ile görevlerin çalışmaya hazır olup olmadığını kontrol eden ve görevlerin çalışmasını zamanlayan HTN planlayıcı bileşenidir. Bu bileşenler Runnable[20] arayüzünü gerçekleştirdikleri için ayrı Java iş parçacığı olarak çalışırlar.

HTN Planlayıcının son bileşeni olan Şekil 4.9' de görülen BHSystem, etmenin yaşam döngüsü boyunca periyodik olarak çalışan döngüsel bir sistem planıdır ve Behaviour sınıfından türetilmiştir. BHSystem planının Action sınıfından türetilen iki tane alt görevi vardır. İlk alt görevi olan ACFetchMessages basit görevi, etmene gelen mesajın devam eden bir planın parçası olup olmadığını kontrol eder. Eğer devam eden bir planın parçasıysa; bekleyen görev kuyruğunu kontrol edip, ilgili

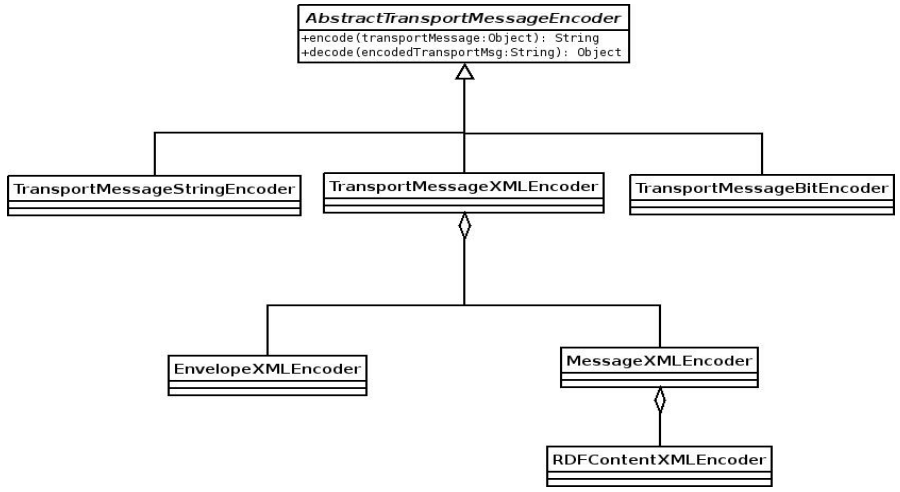
[20] java.util.concurrent.Runnable arayüzü Java'da işparçacağı oluşturmak için kullanılır.

planın çalışması için gerekli olan parametreleri verir. Eğer gelen mesaj devam eden bir planın parçası değilse; gelen isteği karşılamak için yeni bir hedef (Objective) yaratır ve hedefler kuyruğuna ekler. BHSystem planının ikinci alt görevi olan ACMatch, hedef kuyruğunu dinler, hedeflere uygun planları yaratır ve bekleyen görev kuyruğuna ekler.



Şekil 4.9. Planner paketi sınıf diyagramı

Mobil araçlarda TCP soket iletişimi nesne aktarımını desteklemediğinden dolayı nesne serileştirmenin uygulama tarafından yapılması gerekmektedir. Bu nedenle nesnelerin karakter dizisi şeklinde ifade edilmesi ve gelen karakter dizilerinin tekrar uygun nesnelere çevrilmesi için nesne serileştirme paketi oluşturulmuştur. Bu pakete ait sınıf diyagramı Şekil 4.10'da görülmektedir.



Şekil 4.10. Nesne Serileştirme Paketi Sınıf Diyagramı

SeagentLITE projesinde etmenler arasında iletişim serileştirilmiş TransportMessage sınıfı nesnelerinin aktarılmasıyla sağlanır. Bu nesneler karakter dizisi, XML veya bit şeklinde serileştirilebilirler. Farklı yöntemlerin desteklenebilmesi için soyut AbstractTransportMessageEncoder sınıfı bulunmaktadır. XML formatında serileştirme daha uygun bir yöntem olduğu için SeagentLITE projesinde şu anda XML serileştirme desteği verilmektedir.

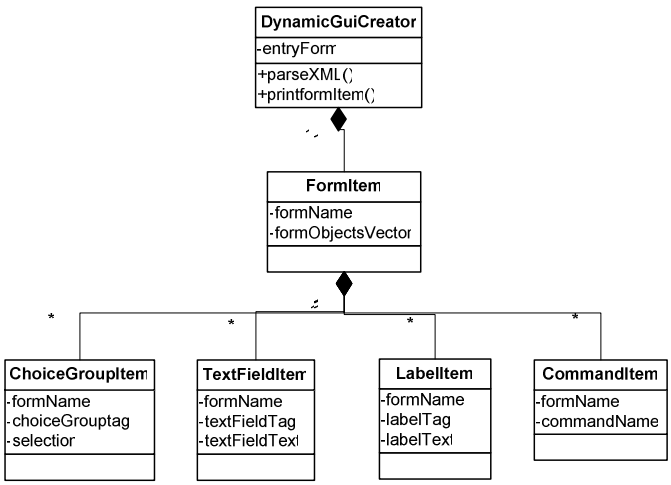
TransportMessage sınıfı, Envelope ve Message nesnelerinden oluşan iki kısım içermektedir. Bu yüzden her iki kısım için de ayrı serileştirme sınıfları bulunmaktadır. RDFContentXMLEncoder sınıfı Message sınıfında bulunan mesaj içeriğini (content) serileştirir.

4.2.3 Arayüz Katmanı

Arayüz katmanı mobil araçtaki etmen ile kullanıcının iletişime geçmesi için gerekli arayüzleri sunar. Örneğin; Etmen Yönetim Sistemine kayıt sırasında kullanıcıdan bilgi alınabilmesi veya Dizin Kolaylaştırıcıya kayıt için sunulmak istenen servislerin seçilebilmesi için

arayüzler oluşturulur. Ayrıca arayüz katmanında, sunucu etmen platformundan servis çağrımında gerekli olan arayüzlerin dinamik olarak yaratılmasından sorumlu sınıflar bulunur. Servise özel XML formatında arayüz tanımlamaları mobil araçtaki etmene mesaj olarak gönderilir. Etmenin bu mesajları ayrıştırmasıyla dinamik olarak arayüzler oluşturulur. Dinamik arayüz yaratılması sayesinde servis kullanmak isteyen etmenden servise özgü alan özellikleri (domain-specific properties) soyutlanmış olmaktadır.

Dinamik arayüz yaratımı için gerekli olan tüm bileşenleri ifade etmek için Şekil 4.11' de görülen sınıflar oluşturulmuştur. Tüm arayüz bileşenleri FormItem sınıfı içinde yer alır. Bu sınıf javax.microedition.lcdui.Form sınıfından türetilmiştir. Etmene gelen XML formatındaki arayüz tanımlamalarını içeren her mesaj için FormItem sınıfının bir nesnesi yaratılır. XML dökümanındaki her bir etiket (tag) için ilgili arayüz bileşeni yaratılır ve FormItem nesnesine eklenir. Bu arayüz bileşenleri CommandItem, ChoiceGroupItem, TextFieldItem, LabelItems sınıflarının nesneleri olabilir.



Şekil 4.11. Arayüz katmanı sınıf diyagramı

Etmen platformunda çeşitli servisler sunulmaktadır. Her servisin çağrılabilmesi için farklı parametrelere gereksinim duyulmaktadır. Bu parametrelerin servis kullanmak isteyen etmenden alınabilmesi için parametrelere özgü farklı arayüzlerin yaratılabilmesi gerekmektedir. Gerekli arayüzler istemci etmene gelen mesajlardaki XML formatındaki arayüz bilgisinden DynamicGuiCreator sınıfı kullanılarak yaratılmaktadır.

Gelen XML dökümanının ayrıştırılarak FormItem nesnesinin yaratılmasından DynamicGuiCreator sınıfı sorumludur. Bu sınıf ayrıca oluşan FormItem nesnesinden dinamik arayüz yaratılmasını ve ekranda gösterilmesini sağlar. MobileDeviceAgentMidlet sınıfı kullanıcı isteklerinin alınabilmesi için yaratılan dinamik arayüzleri sunar.

5. MOBİL ARAÇLAR İÇİN ETMEN TABANLI BİR ANLAMSAL WEB SERVİS SUNUM PLATFORMU GELİŞTİRME

Bu bölümde mobil araçlardaki etmenlerin SEAGENT platformunda sunulan anlamsal web servislerini kullanmalarını sağlamak amacıyla gerçekleştirilen çalışmalar anlatılacaktır. Önceki bölümlerde anlatılan J2ME ortamının özellikleri, mobil araçlara özgü kısıtlar ve gereksinimler çerçevesinde hem SEAGENT platformunda, hem de SeagentLITE platformunda birtakım düzenlemeler yapılmıştır. Öncelikle SEAGENT platformuna yapılan eklentiler, ardından SeagentLITE için yapılan değişiklikler tanıtılacaktır.

5.1 SEAGENT Platformunda Yapılan Çalışmalar

SEAGENT platformunda anlamsal web servislerinin kullanımını sağlamak amacıyla SWSA kavramsal modeli temel alınarak aracılık işlemlerinin gerçekleştirimini sağlamak üzere platform yeniden yapılandırılmıştır. Fakat bu yapı kapsamında; servis hizmeti almak isteyen istemci etmenlerin masaüstü sistemlerde bulunması gerekmektedir. Mobil araçların donanım açısından yeterli olmaması nedeniyle mobil araçtaki bir istemci etmen ontolojilerle ilgili işlemler için yeterli başarım gösterememektedir.

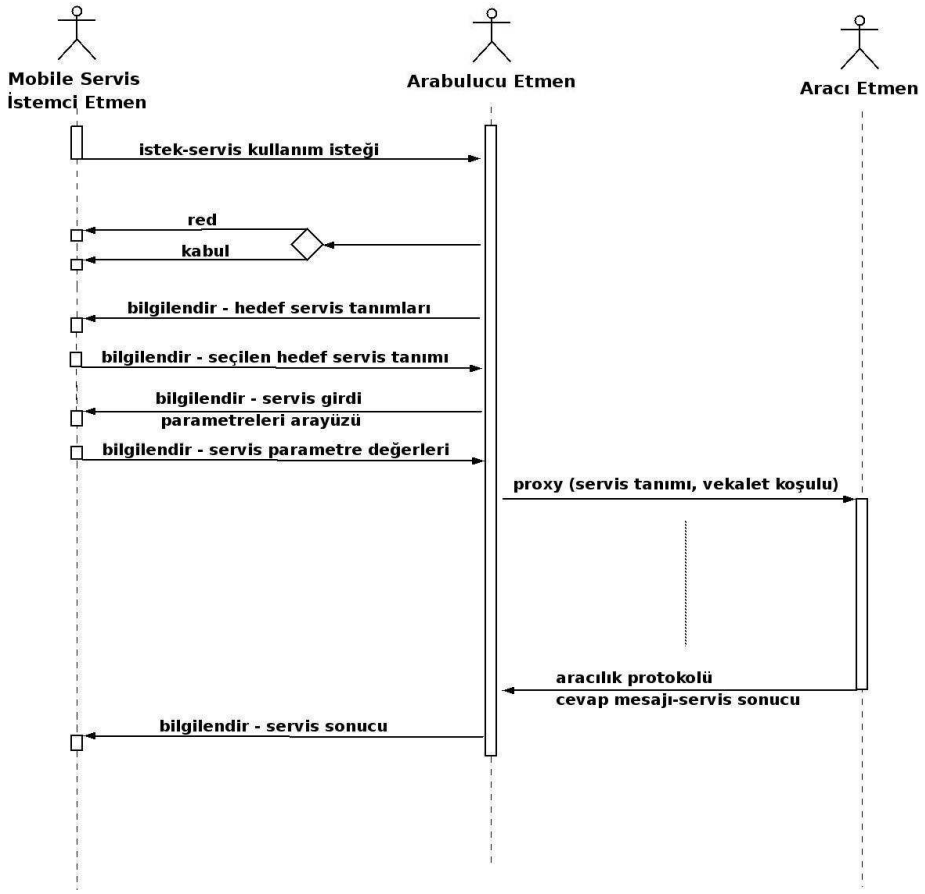
Mobil servis istemci etmen ve aracı etmen arasında kullanılan farklı iletişim protokollerine ve mobil araçların donanım kısıtlarının bir sonucu olarak iki etmen iletişimini sağlayacak ve mobil istemci etmenin seçimlerini yorumlayabilecek farklı bir arabulucu (mediator) etmen ihtiyacı ortaya çıkmıştır. Bu bölümün izleyen alt bölümlerinde öncelikle arabulucu etmenin görevleri ardından arabulucu etmenin bu görevleri yerine getirebilmesi için tanımlanan planları anlatılacaktır.

5.1.1 Arabulucu Etmenin Görevleri

Mobil araçlardan anlamsal web servislerinin kullanımını sağlamak amacıyla mobil araçta gerçekleştirilemeyen işlemleri gerçekleştiren ve mobil araçtaki istemci etmenle aracı etmen arasında iletişimin sağlanabilmesi için gerekli mesaj dönüşümlerini yapan bir arabulucu etmen tanımlanmıştır. Gerçekleştirimi yapılan arabulucu etmen bu tez çalışmasının SEAGENT platformuna yaptığı bir katkı olarak göze çarpmaktadır. Bu arabulucu etmenin görevleri ve her bir görevin açıklaması aşağıda yapılmaktadır:

- *Mobil istemci etmen ve aracı etmenin iletişim kurabilmesi için kullanılan farklı iletişim protokollerinin çevrimini yapmak:*

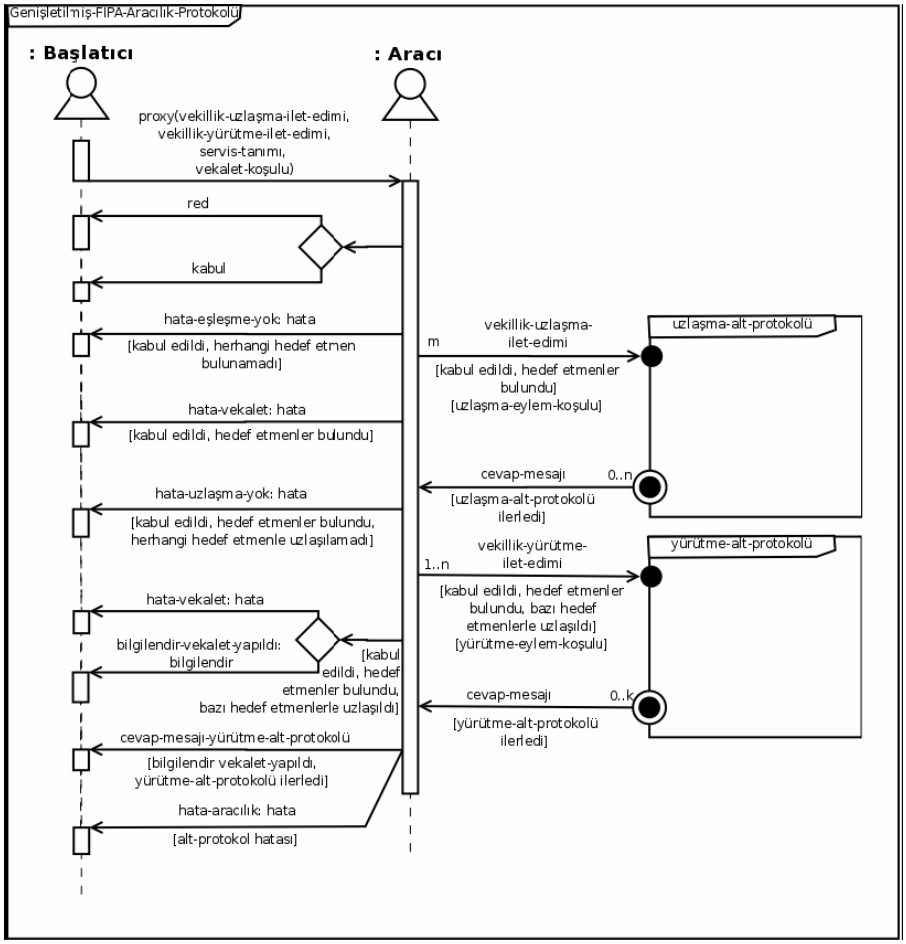
  SEAGENT platformunda mobil araçtaki istemci etmen ve aracı etmen iletişim için farklı protokoller kullanmaktadır. Mobil araçtaki servis istemci etmen ve arabulucu etmen Şekil-5.1'de bu iki etmen arasında görülen iletişim protokolü ile anlaşırlar. Bu iletişim protokolü FIPA istek(request) etkileşim protokolünü destekleyen bir iletişim protokülüdür.



Şekil 5.1. Etmenler arası kullanılan farklı iletişim protokolleri

Arabulucu etmen ve aracı etmen ise Şekil 5.1'de görülen etkileşimin genişletilmiş hali olan ve Şekil 5.2'de görülen genişletilmiş aracılık protokolü ile anlaşır. Bu protokol SWSA kavramsal mimarisindeki anlamsal web servislerinin keşif, uzlaşma ve anlaşma evrelerinin yürütülmesini sağlamaktadır. Genişletilmiş aracılık protokolünün işleyişinde ilk olarak aracı etmen vekil olmayı kabul ederse, vekalet mesajında yer alan anlamsal web servis tanımını kullanarak aday etmenleri bulur ve vekalet koşullarına göre bulduğu etmenleri değerlendirerek uzlaşma

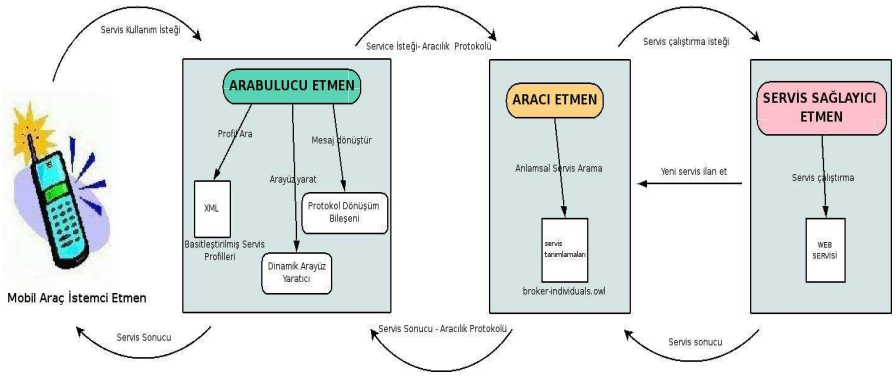
yapılacakları seçer. Bunların sonucunda keşfedilen etmenler listesinde m adet etmen olduğu düşünülürse, aracı etmen bu m etmenle, uzlaşma alt-protokolünü kullanarak uzlaşma etkileşimine başlar. Bu etkileşimler sonucunda, aracı etmen ya hiçbir etmenle anlaşamaz ya da n adet etmenle anlaşır. Eğer anlaşılmış bir servis sağlayıcı etmen varsa, aracı etmen yürütme alt-protokolünü kullanarak 0 ya da daha fazla etmenle servis yürütme etkileşimine başlar. Yürütme alt-protokolü tamamlandığında gelen cevap mesajlarını başlatıcıya iletir ve aracılık etkileşim protokolü sonlanır. Bu arada istenilen servisi sunan hiçbir etmen bulunamazsa, uzlaşma ya da yürütme alt-protokolünden hata mesajı gelirse, hiçbir etmen ile anlaşma sağlanamazsa ya da alt protokollerden kaynaklanmayan bir hata meydana gelirse, aracı etmen başlatıcıya hata mesajı göndererek aracılık etkileşim protokolünü bitirir (Gümüş et al., 2008). Şekil 5.2'deki Başlatıcı aktörü servis isteğinde bulunan aktördür ve mobil araçtaki etmen için başlatıcı rolünü arabulucu etmen üstlenmektedir.



Şekil 5.2. Anlamsal web servisleri için genişletilmiş aracılık etkileşim protokolü (Gümüş, 2008)

Mobil servis istemci etmen ve aracı etmenin iletişime geçebilmesi için protokol dönüşümü yapılması gerekmektedir. Bu protokol dönüşümü arabulucu etmenin ilk görevi olarak tanımlanabilir ve Şekil 5.3'teki protokol dönüşüm bileşeni tarafından sağlanır. Bu bileşen ile aracı etmenden arabulucu etmene gelen mesajlar yorumlanıp, mobil araçtaki etmene gönderilmek üzere uygun mesaj formatında hazırlanır, benzer şekilde; mobil araçtan

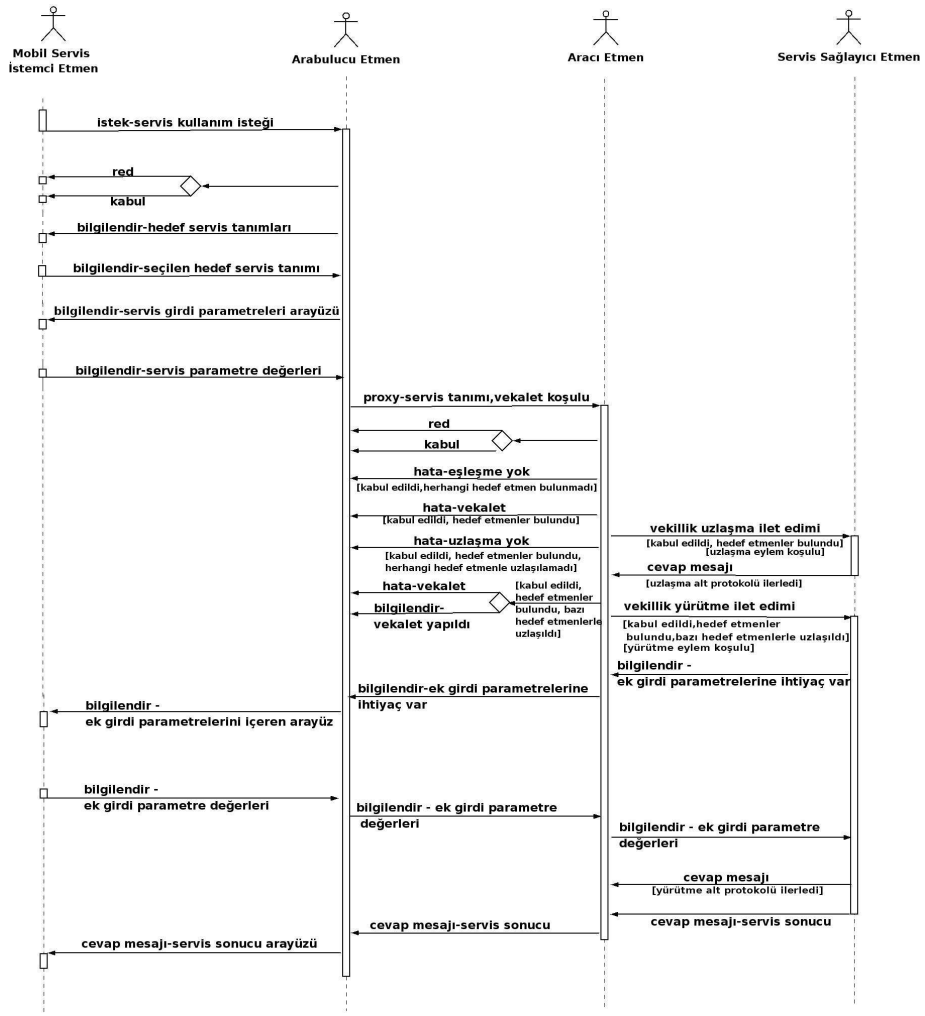
arabulucu etmene gelen mesajlar da yorumlanıp, aracı etmene gönderilmek üzere uygun mesaj formatında hazırlanır. Mobil araçtaki servis istemci etmen kullanmak istediği servise ait girdi parametrelerinin değerlerini içeren mesajı arabulucu etmene gönderdikten sonra arabulucu etmen bu mesaj içindeki bilgileri kullanarak aracı etmene gönderilmek üzere genişletilmiş aracılık etkileşim protokolüne göre vekillik(proxy) mesajını hazırlar. Bu işlem de arabulucu etmenin ACPrepareServiceCallMessage planı içerisinde gerçekleştirilir. Arabulucu etmenin görevlerini yerine getirmesini sağlayan planların açıklaması Bölüm 5.1.2'de yapılacaktır.



Şekil 5.3. Arabulucu etmenin diğer etmenlerle etkileşimi

Servis sağlayıcı etmen, servisi çağırmak için aracı etmendeki profil bilgilerinde olmayan bir bilgiye ihtiyaç duyduğu zaman bu ihtiyacını aracı etmene bildirir. Aracı etmen de istenen bilgilerin mobil araçtaki etmenden alınabilmesi için bu isteği arabulucu etmene gönderir. Şekil 5.4'te bu durumu karşılayan genişletilmiş protokol tasarımı gösterilmektedir. Bu tasarıma göre bir servis sağlayıcı etmenle uzlaşma sağlandıktan sonra istenilen servisin

kullanılması için ek girdi parametrelerine ihtiyaç duyulması halinde Şekil 5.2’den farklı olarak servis sağlayıcı etmen ek girdi parametresi talebinde bulunabilmektedir. Bu isteği sırasıyla aracı etmen arabulucu etmene ve arabulucu etmen de mobil istemci etmene iletmektedir. Mobil istemci etmen de ek girdi parametrelerin değerlerini arabulucu etmene göndermektedir. Genişletilen bu protokol sayesinde servis sağlayıcı etmenin gereksinim duyabileceği ek bilgiler sağlanmaktadır.



Şekil 5.4. Ek bilgi ihtiyacına yönelik genişletilen protokol

- *Aracı etmenin yaptığı yeni servis kayıtlarına göre kendine ait kayıtları güncel tutmak :*

  Aracı etmene yeni bir servis sağlayıcı etmen kayıt yaptırdığı zaman aracı etmen yapılan yeni servis sağlayıcı etmen kaydını arabulucu etmene mesaj göndererek haber verir. Gönderilen bu mesaj tez çalışmasında aracı etmene yapılan eklentilerden biridir.

Aracı etmen gönderdiği mesaj içine de bu yeni kaydın OWL-S dökumanını ekler. Arabulucu etmen bu mesajı aldığında BHUpdateServiceProfiles planını çalıştırır. Bu planda gelen OWL-S dökümanından basitleştirilmiş servis profili elde edilir ve Şekil 5.3'te basitleştirilmiş servis profilleri xml dosyası şeklinde gösterilen "available_service_definitions.xml" dosyasına eklenir. Gelen OWL-S dökümanı SEKMA[21](SEAGENT Knowledge Management Architecture) bileşenleri ile ayrıştırılmaktadır. Ayrıştırma sonucu elde edilen servis tanımlaması, servise ait girdi çıktı parametreleri gibi bilgiler basit xml formatında "available_service_definitions.xml" dosyasına kaydedilir. Bu sayede mobil araçtaki etmen platforma yeni eklenen servislerden dinamik olarak haberdar olabilmektedir. Aşağıda platformda bulunan "available_service_definitions.xml" dosyasının görünümü bulunmaktadır. Bu dosya içinde servisin tanımlayıcısı ve servisi kullanmak için gereken girdi, çıktı parametrelerinin isimleri ve tipleri yer alır. Dosyanın XML ile ifadesi:

```
<goals>
<goal name="HIRE ESTATE" goalName=
                  "hire_Estate_Service">
<inputs>
<input name="estateName" value=""
                  type="XMLType.XSD_STRING"/>
</inputs>
<outputs>
<output name="hireEstateReturn" value=""
                  type="XMLType.XSD_STRING"/>
</outputs>
</goal>

<goal name="FIND ESTATES"
            goalName="find_Estates_Service">
```

[21] SEKMA Web Site : http://seagent.ege.edu.tr/wiki/index.php?title=SeKMA&redirect=no, son erişim 12 Nisan 2008.

```
<inputs>
<input name="estateType" value=""
                  type="XMLType.XSD_STRING"/>
<input name="estatePrice2" value=""
type="XMLType.XSD_STRING"/>
<input name="estatePrice1" value=""
type="XMLType.XSD_STRING"/>
<input name="estateRegion" value=""
type="XMLType.XSD_STRING"/>
</inputs>
<outputs><output name="findEstatesReturn" value=""
      type="XMLType.XSD_ANY"/>
</outputs>
</goal>

<goal name="FIND HOTELS"
goalName="find_Hotels_Service">
<inputs>
<input name="hotelLocation" value=""
               type="XMLType.XSD_STRING"/>
<input name="hotelStar" value=""
               type="XMLType.XSD_STRING"/>
<input name="hotelPrice2" value=""
               type="XMLType.XSD_STRING"/>
<input name="hotelPrice1" value=""
               type="XMLType.XSD_STRING"/>
</inputs>
<outputs>
<output name="findHotelsReturn" value=""
               type="XMLType.XSD_ANY"/>
</outputs>
</goal>

<goal name="BOOK HOTEL"
               goalName="book_Hotel_Service">
<inputs>
<input name="rezervationDate" value=""
               type="XMLType.XSD_STRING"/>
<input name="hotel" value=""
               type="XMLType.XSD_STRING"/>
<input name="personName" value=""
               type="XMLType.XSD_STRING"/>
<input name="numOfPeople" value=""
               type="XMLType.XSD_STRING"/>
</inputs>
<outputs>
<output name="bookHotelReturn" value=""
               type="XMLType.XSD_STRING"/>
```

```
</outputs>
</goal>

</goals>
```

şeklindedir. Bu dosyada <goals> </goals> etiketi arasında platformda bulunan anlamsal servislerin hedef şablonları <goal></goal> etiketiyle tek tek tanımlanmaktadır. Her bir servis için <goal> etiketi altındaki <inputs> etiketinde servise ait girdi parametreleri ayrı ayrı <input> etiketiyle açıklanır. <input> etiketi içinde ise girdi parametresinin adı, değeri ve tipi tanımlanmaktadır. Servise ait sonuç değeri ve değerleri ise <outputs> etiketi altında herbir sonuç değeri için <output> etiketi ile açıklanmaktadır. <input> etiketine benzer şekilde <output> etiketinde de sonuç değerinin ismi, değeri ve tipi bulunmaktadır. Arabulucu etmenin gerçekleştirdiği bu dönüşüm işlemi ile ilgili olarak örnek bir OWL-S dosyası, dönüşüm işleminden sonra basitleştirilmiş tanımı ve bu işlemi gerçekleştiren planın Java kodları EK 3'te bulunmaktadır.

- *Anlamsal web servis isteğinde bulunulması için gerekli olan anlamsal tanımlamaları aracı etmene göndermek, gelen anlamsal servis tanımlamalarından mobil araçtaki istemci etmenin anlayacağı biçimde basitleştirilmiş xml tanımlamalarını oluşturmak :*

  SEAGENT platformunda servis sağlayıcı etmenler kendilerini aracı etmene OWL-S dökümanlarındaki tanımlamalara göre kaydettirmektedirler. Aracı etmende servis sağlayıcı etmenleri OWL-S dökümanlarındaki anlamsal tanımlamalara göre aramaktadır. Herhangi bir servis istemci etmen, aracı etmenden servis isteğinde bulunacağı zaman istediği servise ait anlamsal tanımlamaları göndermek zorundadır. Fakat mobil araçtaki etmen

bu anlamsal tanımlamaları yapamaz. Bu nedenle anlamsal tanımlamalar arabulucu etmen tarafından yapılır. Mobil araçtaki etmen, arabulucu etmen tarafından oluşturulmuş olan basitleştirilmiş servis tanımlamaları dosyasındaki servis tanımlarını kullanarak servis isteğinde bulunabilir. Şekil 5.3'te arabulucu etmen içinde basitleştirilmiş servis profilleri olarak tanımlanmış olan “available_service_definitions.xml” dosyası bu tanımlarını içerir. Arabulucu etmen, mobil araçtan gelen servis isteklerini servis ile ilgili gerekli dönüşümleri yapıp, servisin anlamsal tanımlamalarını oluşturarak aracı etmene gönderir. Arabulucu etmen dönüşüm işlemini ACPrepareServiceCallMessage planı içerisinde gerçekleştirmektedir. Bu plan içerisinde mobil araçtaki istemci etmenden gelen servis kullanım isteğini bildiren mesaj bir ihtiyaç(provision) olarak alınmaktadır. Mobil araçtaki istemci etmenin gönderdiği mesajda kullanmak istediği servis ve servise ait girdi parametrelerinin değerleri bulunmaktadır. Arabulucu etmen bu mesajdan aracı etmene göndermesi gereken aracılık mesajı için gerekli bilgileri elde eder ve kullanılmak istenilen servise ait değerleri ve owl-s dökümanını aracı etmene göndereceği mesaja ekler. Benzer şekilde aracı etmen servis sonucunu arabulucu etmene gönderdiğinde gelen mesajdan servis sonucunu ayrıştırır ve mobil araçtaki istemci etmene xml ile tanımlanan uygun dinamik arayüz formatında gönderir. Bu işlemi de ACAnalyseInformMessage planı içerisinde gerçekleştirir.

- *Mobil araçtaki etmenin servis girdi parametrelerine istediği değerleri verebilmesi amacıyla, servis tanımına özel dinamik arayüz tanımlamaları oluşturmak ve bu tanımlamaları mobil araçtaki etmene göndermek:*

  Mobil araçtaki etmen servis isteğinde bulunacağı zaman servise ait parametrelerin neler olduğunu bilemez. Bu nedenle parametreleri öğrenmek için arabulucu etmene istekte bulunur. Arabulucu etmen servis tanımlamalarının bulunduğu dosyayı okur ve dinamik olarak gerekli parametreleri içeren xml formatında ifade edilmiş arayüzü Şekil 5.3'teki dinamik arayüz yaratıcı bileşeni ile oluşturur. Bu arayüzü mobil araçtaki etmene gönderir. Mobil araçtaki etmen de bu arayüz aracılığıyla gerekli parametrelerin değerlerini girer. Aşağıda emlak arama servisini kullanmak isteyen bir istemci etmene gönderilen FIPA mesajının argument kısmında yeralan arayüz tanımlaması bulunmaktadır :

```
<fipa:argument>
 <guiString>
 <Screen> xsi='"http://www.w3.org/2001/XMLSchema-
      instance'"xsi:SchemaLocation='"C://SEAGENT//SEAG
      ENT//schema//schema.xsd'">
<Form>
<FormName>Find Estates</FormName>
<TextField>  <TextFieldTag>estateType</TextFieldTag>
             <TextFieldText>-</TextFieldText>
</TextField>
<TextField>  <TextFieldTag>estatePrice2</TextFieldTag>
             <TextFieldText>1</TextFieldText>
</TextField>
<TextField>  <TextFieldTag>estatePrice1</TextFieldTag>
             <TextFieldText>1</TextFieldText>
```

```
</TextField>
<TextField>  <TextFieldTag>estateRegion</TextFieldTag>
             <TextFieldText>-</TextFieldText>
</TextField>
<Command>
<commandName>Call Service</commandName>
</Command>
<Command>    <commandName>BackMainForm</commandName>
</Command>
</Form>
</Screen>
</guiString>
</fipa:argument>
```

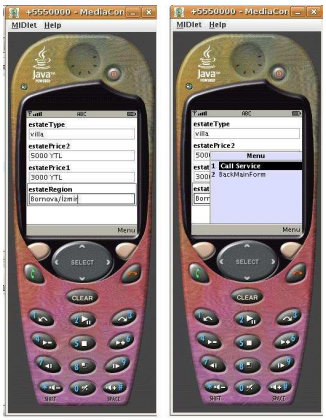
Gönderilen bu arayüz tanımlamasında emlak arama kriterlerinin belirtilebilmesi dört tane metin kutusu ve servisin çağrılması için buton bulunmaktadır. Metin kutularına istenen emlağın tipi(apartman dairesi, villa vb.), fiyat aralığı ve emlağın bulunacağı bölge değerleri verilmektedir. Yukarıda etmenler arası iletişim için kullanılan bir mesajın yalnızca örnek arayüz tanımlaması bulunmaktadır. EK 2'de etmenler arası iletişimde kullanılan mesajların tam içerikleri incelenebilir.

Arabulucu etmen tarafından mesaj için tanımlaması yapılarak mobil araçtaki istemci etmene gönderilen arayüzler "available_service_definitions.xml" dosyasında ilgili servis için bulunan bilgilere göre yapılmaktadır. Yukarıdaki arayüz için "available_service_definitions.xml" dosyasında bulunan bilgiler aşağıdaki gibidir:

```
<goal name="FIND ESTATES"
    goalName="find_Estates_Service">
<inputs>
<input name="estateType" value=""
    type="XMLType.XSD_STRING"/>
```

```
<input name="estatePrice2" value=""
     type="XMLType.XSD_STRING"/>
<input name="estatePrice1" value=""
     type="XMLType.XSD_STRING"/>
<input name="estateRegion" value=""
     type="XMLType.XSD_STRING"/>
</inputs>
<outputs>
<output name="findEstatesReturn" value=""
     type="XMLType.XSD_ANY"/>
</outputs>
</goal>
```

Arabulucu etmen istenen servis için girdi parametrelerini içeren arayüzü oluşturmak için yukarıda görülen basit servis tanımındaki <inputs> etiketi içinde yeralan ve her bir girdi parametresini tanımlayan <input> etiketi içindeki değerlerden yararlanarak uygun arayüzü oluşturmaktadır. Arabulucu etmenin gönderdiği arayüz tanımlaması sonucu oluşan arayüz Şekil 5.5'teki gibidir. Şekil 5.5'teki arayüzde servis için istenen dört adet girdi parametresi bulunmaktadır. Girdi parametrelerinin isimleri(estateType, estatePrice2, estatePrice1, estateRegion) ve tipleri "available_service_definitions.xml" dosyasında bulunan servisle ilgili yukarıda da görülmekte olan bilgiler doğrultusunda oluşturulmuştur.

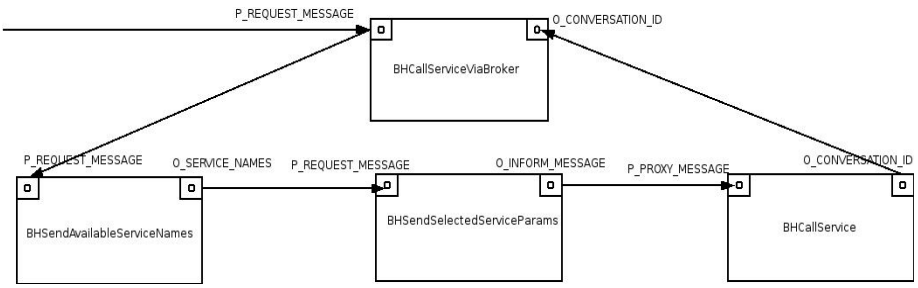


Şekil 5.5. Dinamik Arayüz Örneği

5.1.2 Arabulucu Etmenin Planları

SEAGENT platformunda yaratılan arabulucu etmenin görevleri açıklandıktan sonra bu etmenin görevlerini yerine getirebilmesi için HTN formalizmine göre oluşturulmuş olan planlar anlatılacaktır. Arabulucu etmenin servis çağrım süreci için kullandığı plan pek çok alt plandan oluşan ve Şekil 5.6'da görülen BHCallServiceViaBroker planıdır. Bu plan mobil araçtan gelen servis isteğini karşılar. En üst seviyede üç alt plandan oluşmaktadır. Bu alt planlar aşağıda açıklanmaktadır:

- BHSendAvailableServiceNames: Mobil araçtaki etmene etmen platformunda bulunan genel servis tanımlarını gönderir.
- BHSendSelectedServiceParameters: Mobil etmenin kullanmak istediği servise ait girdi paremetrelerini "available_service_definitions.xml" dosyasından elde eder ve mobil araçtaki etmene gönderir.
- BHCallService: Bu plan aracı etmene servis çağırma isteğini gönderir ve dönen sonucu da mobil araçtaki istemci etmene gönderir.

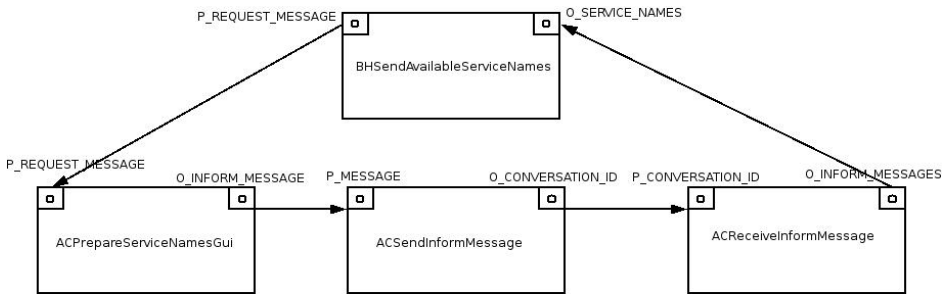


Şekil 5.6. BHCallServiceViaBroker görevinin ayrıştırılmış görünümü

BHCallServiceViaBroker planına ait alt görevlerin de kendilerine ait pek çok alt görevleri bulunmaktadır. Arabulucu etmenin görevlerinin

anlaşılabilirliğinin artması açısından herbir alt görev çalıştırılma sıralarına göre açıklanacaktır. İlk olarak BHSendAvailableServiceNames alt görevi çalıştırılmaktadır(Şekil 5.7) ve bu görev de karmaşık bir alt görevdir. Bu görevde çok-etmenli platformda bulunan servislerin isimlerinden oluşan XML ile ifade edilmiş arayüzü içeren mesaj, arabulucu etmenden mobil araçtaki istemci etmene gönderilir. Bu görevin alt görevleri aşağıda açıklanmaktadır:

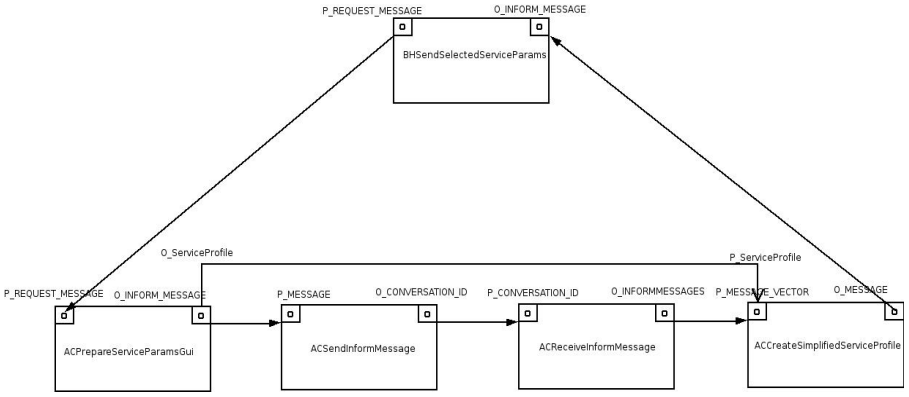
- ACPrepareServiceNamesGui: "available_service_definitions.xml" dosyasını okur. Bu dosyada sistemde bulunan servislerin profilini içeren basitleştirilmiş xml yapıları bulunmaktadır. Arabulucu etmen bu dosyadaki servis isimlerini ayrıştırarak xml formatında ifade edilmiş bir arayüz hazırlar.
- ACSendInfomMessage: Hazırlanan bilgilendirme(inform) mesajını mobil araçtaki etmene gönderir.
- ACReceiveInformMessage: Mobil araçtaki etmenin kullanmak istediği servisin tanımlayıcısının bulunduğu bilgilendirme mesajını alır.



Şekil 5.7. BHSendAvailableServiceNames görevinin ayrıştırılmış görünümü

BHSendAvailableServiceNames görevinin çalıştırılmasından sonra mobil araçtaki istemci etmen seçtiği servisi arabulucu etmene göndermiş durumdadır. Gönderilen mesajda bulunan seçilen servis bilgisine göre arabulucu etmen BHSendSelectedServiceParameters(Şekil 5.8) görevini çalıştırarak ilgili servisin bilgilerinden servisin girdi parametrelerini içeren arayüzü mobil araçtaki istemci etmene gönderir. Bu görevin alt görevleri aşağıda açıklanmaktadır:

- ACPrepareServiceParamsGui: Mobil etmenin kullanmak istediği servise ait girdi paremetrelerini "available_service_definitions.xml" dosyasından elde eder ve bu parametreleri içeren xml formatında bir arayüz oluşturur.
- ACSendInformMessage: Oluşturulan arayüzü mobil araçtaki istemci etmene gönderir.
- ACReceiveInformMessage: Mobil araçtan gönderilen servise ait girdi parametrelerinin değerlerini içeren mesajı alır.
- ACCreateSimplifiedServiceProfile: Mesaj içinde bulunan girdi parametrelerinin değerlerinden SimplifiedServiceProfile nesnesi yaratılır. SimplifiedServiceProfile sınıfı sayesinde platformda bulunan anlamsal web servisleri kendilerine ait OWL-S dosyalarına göre basit olarak tanımlanabilmektedir.

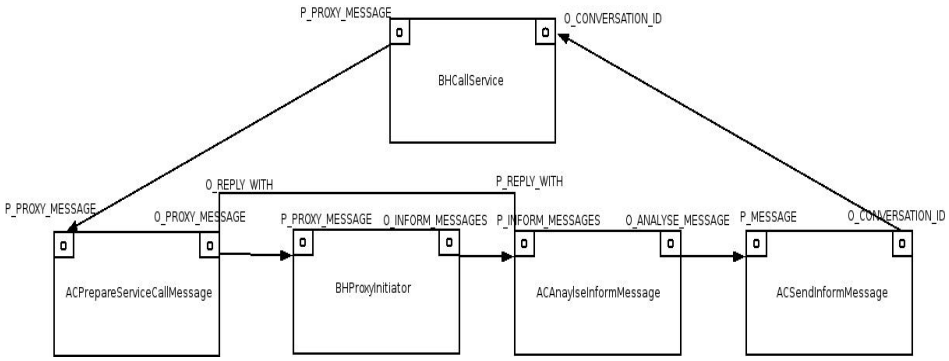


Şekil 5.8. BHSendSelectedServiceParameters görevinin ayrıştırılmış görünümü

Mobil araçtaki istemci etmen girdi parametrelerinin değerlerini içeren mesajı arabulucu etmene göndedikten sonra, arabulucu etmen servis çağırmak için BHCallService(Şekil 5.9) görevini çalıştırır. Bu görev aracı etmenin istenen servisi çalıştırabilmesi için gerekli parametreleri içeren mesajı hazırlar ve aracı etmene gönderir. Aracı etmenden gelen servis sonucunu ayrıştırır ve mobil araçtaki istemci etmene uygun arayüz tanımı içinde gönderir. Bu göreve ait alt görevler aşağıda açıklanmaktadır:

- ACPrepareServiceCallMessage: Aracı etmenin web servis çağırması için gerekli olan parametreleri içeren ServiceDescription nesnesi oluşturulur ve aracı etmene gönderilmek üzere bir mesaj içine konur.
- BHProxyInitiator: Hazırlanan mesajı aracılık etkileşim protokolüne göre aracı etmene gönderir ve aracı etmenden gelecek olan cevap mesajını bekler.

- ACAnalyseInformMessage: Aracı etmenden gelen servis sonucunu içeren vekalet(proxy) mesajını ayrıştırır ve mobil araçtaki etmene gönderilmek üzere servis sonuç mesajını hazırlar.
- ACSendInformMessage: Hazırlanmış olan servis sonuç mesajı mobil araçtaki etmene gönderilir.



Şekil 5.9. BHCallService görevinin ayrıştırılmış görünümü

Arabulucu etmen yukarı açıklanan planlar sayesinde aracı etmenle, mobil istemci etmen arasındaki etkileşimi sağlamaktadır. Mesajların birbirine çevrimi, ontoloji işlemleri ve servise özel dinamik arayüz oluşturma gibi görevlerini bu planlar ile gerçekleştirmektedir. BHCallServiceViaBroker planının tüm alt görevleriyle ilişkisini gösteren şekil ve BHCallServiceViaBroker planının Java kodları EK 1'de incelenebilir. Bir sonraki bölümde SeagentLITE platformundaki mobil istemci etmen için yapılan çalışmalardan bahsedilecektir.

5.2 SeagentLite Platformunda Yapılan Çalışmalar

Mobil araçlardan anlamsal web servislerine erişim amacıyla SEAGENT platfomunda yapılan çalışmalar önceki bölümde anlatılmıştır. Önceki bölümde anlatılan çalışmalarda mobil araçlarda yapılamayan

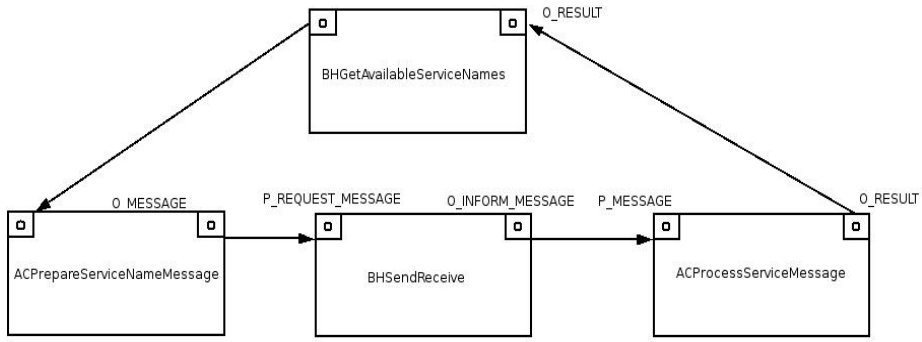
işlemlerin masaüstü sistemdeki arabulucu etmen tarafından yapılması sağlanmıştır. Bu bölümde ise, mobil araçtaki etmenin anlamsal web servislerini kullanabilmesi için yaptığı işlemlerden ve bu işlemleri yerine getirebilmesi için kullandığı HTN formalizmine göre yazılmış planlardan bahsedilecektir.

SEAGENT platformunda gerçekleştirimi yapılmış SWSA kavramsal modeline mobil araçtaki istemci etmenin de dahil edilebilmesi için isteklerini daha az anlamsallık içeren bir ifade ile arabulucu etmene iletebilmesi gerekmektedir. Bu nedenle arabulucu etmenle mobil istemci etmen arasında FIPA istek etkileşim protokolüne bağlı basit bir protokol geliştirilmiştir. Mobil araçtaki istemci etmenin de bu protokole ayak uydurabilmesi gerekmektedir. Bu protokol ile iletişimi sahip olduğu planlar ile yapabilmektedir. Diğer yandan, arabulucu etmen mobil istemci etmenin anlamsal web servislerini kullanabilmesi için ilgili anlamsal web servisine göre oluşturulan XML formatındaki dinamik arayüz tanımlamalarını mobil istemci etmene göndermektedir. Bu arayüz tanımlamalarının mobil araçta oluşturulma görevi yine mobil araçtaki istemci etmenin yerine getirmesi gereken görevlerinden birisidir. Mobil araçlarda ontoloji işlemlerinin yerine getirilmesi donanımsal açıdan mümkün olmamasına rağmen, basit XML dosyaları kXML[22] xml ayrıştırıcı kütüphanesi ile ayrıştırılabilmektedir. Bu sayede dinamik arayüz tanımlamalarını içeren XML dosyaları ayrıştırılabilmektedir.

Mobil istemci etmenin anlamsal web servislerini kullanabilmesi için HTN formalizmine göre oluşturulmuş üç plan bulunmaktadır. Bu planların ilki olan BHGetAvailableServiceNames planı, arabulucu etmenden etmen platformunda bulunan genel servis tanımlarını ister. Bu işlemi gerçekleştirebilmek için üç alt görevi bulunmaktadır. İlk alt görevi

[22] kXML Web Site : kxml.sourceforge.net, son erişim 13 Nisan 2008.

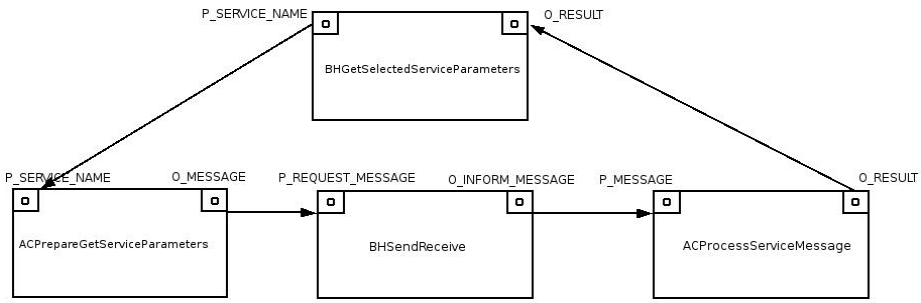
ACPrepareServiceNameMessage ile arabulucu etmene gönderilecek olan istek mesajını hazırlar. BHSendReceive alt görevi ile hazırlanan istek mesajını gönderir ve arabulucu etmenden gelen mesajı alır. Son olarak ACProcessServiceMessage alt görevi gelen mesajdaki genel servis tanımlamalarını içeren xml arayüz dökümanını ayrıştırır ve bu tanıma göre arayüzü oluşturur. BHGetAvailableServiceNames planının alt görevleri arası ilişki Şekil 5.10'da görülmektedir.



Şekil 5.10. BHGetAvailableServiceNames görevinin ayrıştırılmış görünümü

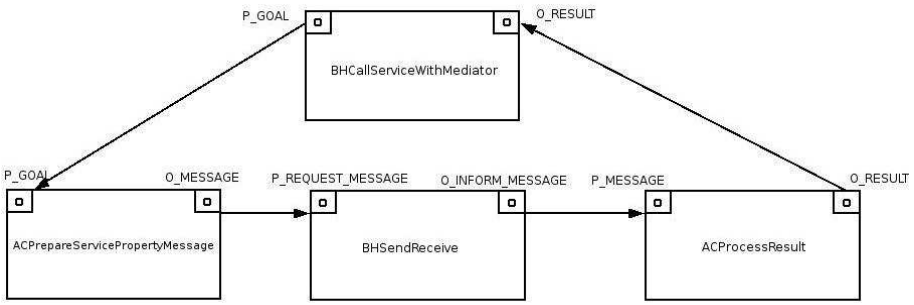
Kullanıcının arayüzden hedef servisi seçmesinden sonra Şekil 5.11'de ayrıştırılmış görünümü bulunan BHGetSelectedServiceParameters planı başlar. Bu plan arabulucu etmenden seçilen servise ait girdi parametrelerini ister. ACPrepareGetServiceParameters, BHSendReceive ve ACProcessServiceMessage alt görevleridir. ACPrepareGetServiceParameters alt görevi, arabulucu etmene gönderilecek olan istek mesajını hazırlar. Bu istek mesajında seçilen servis ismi gönderilecek mesaja eklenir. ACProcessServiceMessage ise gelen mesajdaki servis parametrelerini içeren xml arayüz dökümanını

ayrıştırır ve dinamik olarak arayüz oluşturur. Bir önceki bölümde arabulucu etmenin gönderdiği arayüz tanımına göre emlak arama servisi kullanmak isteyen mobil istemci etmenin yarattığı arayüz Şekil 5.5'te görülmektedir. Hedef servisin çağrılması için Şekil 5.5'te görülen arayüzde girdi parametrelerine istenen değerler verildikten sonra servis çağrım isteğinde bulunulmak üzere CallService butonu kullanılmaktadır.



Şekil 5.11. BHGetSelectedServiceParameters görevinin ayrıştırılmış görünümü

Servis girdi parametrelerine mobil araçtaki arayüz ekranından değer verilmesinin ardından, Şekil 5.12'de görülen BHCallServiceWithMediator planı başlar. Bu plan anlamsal web servis kullanım sürecinde mobil araçtaki etmenin çalıştırdığı son plandır. Arabulucu etmenden amaca uygun servisin çağrılmasını ister. ACPrepareServicePropertyMessage, BHSendReceive ve ACProcessResult alt görevleridir. ACPrepareServicePropertyMessage, arabulucu etmene gönderilecek olan istek mesajını hazırlar. Bu istek mesajında seçilen servis ismi ve parametre değerleri SimplifiedServiceProfile sınıfından oluşturulmuş bir nesne ile ifade edilir. ACProcessResult alt görevi ise, gelen mesajdaki servis sonucunun gösterilmesi için bir arayüz oluşturur.



Şekil 5.12. BHCallServiceWithMediator görevinin ayrıştırılmış görünümü

5.3 Anlamsal Web Servislerinin Kullanımında Etmenler Arası Etkileşimin Açıklanması

Bu bölümde mobil araçlar için anlamsal servis kullanımının sağlanmasını amaçlayan mimaride rol alan etmenler arasındaki etkileşimin nasıl ve hangi sırada gerçekleştiği açıklanacaktır. Mobil araçtaki etmenden gelebilecek anlamsal servis isteği sonrasında oluşacak etkileşim Şekil 5.13'deki gibidir. Bu şekilde gerçekleştirilen her bir adım aşağıda açıklanmaktadır:

1 – Mobil araçtaki etmen, arabulucu etmenden servis isteğinde bulunur.

2 - Arabulucu etmen, mobil araçtaki etmene etmen platformunda bulunan genel servis tanımlarını gönderir.

3 – Mobil araçtaki etmen seçtiği servis ismini arabulucu etmene gönderir.

4 - Arabulucu etmen, mobil etmenin kullanmak istediği servise ait girdi parametrelerini "available_service_definitions.xml" dosyasından elde eder ve mobil araçtaki etmene gönderir.

5 – Mobil araçtaki etmen seçilen servis ismi ve parametre değerlerini SimplifiedServiceProfile sınıfından oluşturulmuş bir nesne ile arabulucu etmene gönderir.

6 - Arabulucu etmen, mobil araçtan gelen servis isteğini aracı etmen için uygun formata dönüştürür.

7 - Arabulucu etmen, servis isteğini aracı etmene gönderir.

8 - Aracı etmen, istenen özellikteki servisleri bulmak için anlamsal arama yapar.

9 – Aracı etmen, uygun servisleri sunan servis sağlayıcı etmenlerin herbirine anlaşma mesajı gönderir.

10 – Mesaj gönderilen servis sağlayıcı etmenler cevap mesajı gönderirler. Kendisine mesaj gelen servis sağlayıcı etmen cevap mesajında servis kullanım isteğini kabul edebilir (AGREE) ya da reddedebilir(REFUSE).

11 - Aracı etmen, anlaşmayı kabul eden servis sağlayıcı etmenlere servis kullanma isteğini içeren mesaj gönderir.

12 – Servis sağlayıcı etmen servisin OWL-S dökümanını kullanarak web servisini çalıştırır.

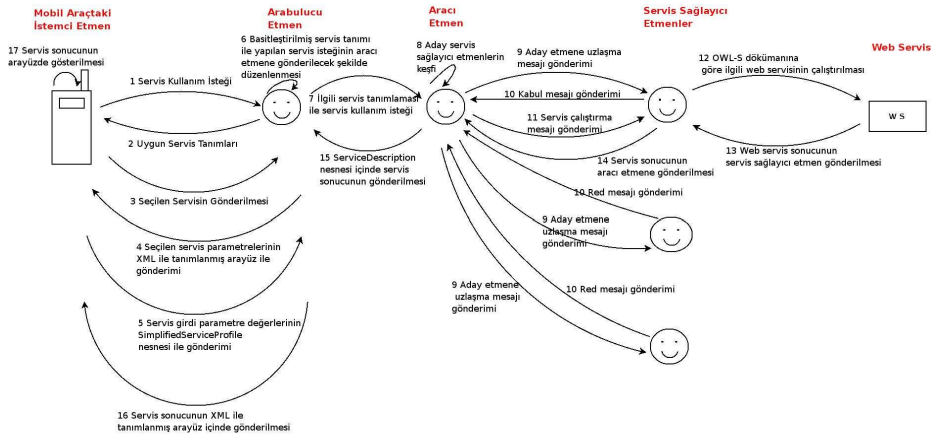
13 – Web servisi sonucu servis sağlayıcı etmene gönderir.

14 – Servis sağlayıcı etmen sonucu aracı etmene gönderir.

15 - Aracı etmen sonucu arabulucu etmene gönderir.

16 - Arabulucu etmen sonucu mobil araçtaki etmene uygun formata çevirerek gönderir.

17 – Mobil araçtaki etmen sonucu arayüzde gösterir.



Şekil 5.13. Anlamsal web servis kullanım sürecinde gerçekleştirilen işlemler

5.4 Durum Çalışmaları

Bu bölümde önceki bölümde anlatılan etmenler arası etkileşim örnek durum çalışması ile somutlaştırılacaktır. Gerçekleştirilen durum çalışmasıyla yeni eklenen servislerin mimariye kolayca eklenebildiği ve mobil araçtaki istemci etmenin de güncellemelerden çalışma zamanında haberdar olduğu görülecektir. Durum çalışmasıyla birlikte mimarinin kalite özniteliği olan sistemin esnek yapısı ortaya çıkmaktadır.

Yapılan durum çalışması mobil araçta gerçekleştirilmek istenen otel arama-rezervasyon işlemi ve emlak arama-kiralama işlemi için örneklenmiştir. Mobil araçtaki etmenin anlamsal servisleri kullanabilmesi için öncelikle SEAGENT platformunun hazır olması gerekmektedir. SEAGENT platformunun başlatılması için gerekli olan kod örneği aşağıdadır.

```
platformCore = PlatformFactory.createPlatformCore(
platformName,ConnectionPointFactory.TCPSocket_CONNECTION,
"2226" );
```

SEAGENT platformunun başlatılmasıyla etmen yönetim sistemi ve etmen iletişim kanalı gibi temel etmen platformu bileşenleri başlatılır. Ardından servis kullanımını sağlayan aracı etmen ve arabulucu etmen yaratılır, etmen yönetim sistemine kaydedilir. Son olarak servis sağlayıcı etmenler servislerini aracı etmene kaydettirirler. Aracı etmen kendine yapılan yeni kayıtları arabulucu etmene bildirir ve arabulucu etmen de kendi kayıtlarını günceller. Yapılan bu işlemler sonucunda SEAGENT platformu anlamsal servis platformu olarak hazır hale gelmiş olur.

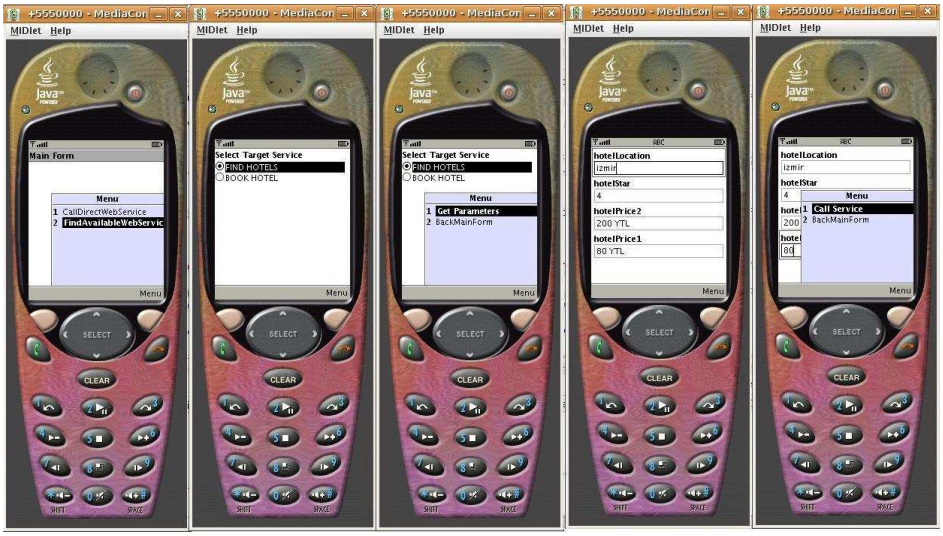
SEAGENT platformu hazırlandıktan sonra, mobil araçtaki etmen yaratılır ve bağlanmak istediği platforma kayıt olmak için istek mesajı gönderir. Mobil araçta etmen yaratılması için kullanıcının etmen adını ve bağlanılmak istenen platform adını tanımlaması gerekmektedir. Kullanıcıdan bu verilerin alındığı arayüz Şekil 5.14(a)'da görülmektedir.

(a) (b)

Şekil 5.14. Mobil araçtaki etmenin yaratılması

Mobil araçtaki etmenin SEAGENT platformunda kayıt işleminin başarılı bir şekilde yapılmasından sonra SEAGENT platformundaki etmen yönetim sistemi mobil araçtaki etmene kayıt işleminin başarılı sonucunu bilgilendirme mesajı ile gönderir. Bu mesaj alındığında ekranda kaydın başarılı olup olmadığını bildiren arayüz görülür (Şekil5.14(b)). Bu aşamada eğer arayüzde kayıt işleminin başarılı olduğu görülüyorsa mobil araçtaki etmen çok-etmenli platforma dahil edilmiştir. Arayüzden çok-etmenli platform aracılığı ile anlamsal web servislerine erişimi sağlayabilecektir.

Tez çalışmasında oluşturulan mobil araçların anlamsal web servislerine erişimi sağlayan mimarinin sınanması amacıyla bu bölümde mobil araçtaki etmen önce otel arama-rezervasyon işlemi gerçekleştirecektir. Ardından platforma yeni servisler eklenecektir. Eklenen servislere göre arabulucu etmen kendi kayıtlarını güncelleyecek ve mobil araçtaki istemci etmen çalışma zamanında tekrar platformdaki servisleri sorguladığında yeni eklenen servisleri görebilecektir. Durum çalışmasında emlak arama ve kiralama servisleri platforma eklenmiştir. Mobil araçtaki istemci etmen bu sefer emlak arama-kiralama işlemi gerçekleştirmek amacıyla bu servisleri kullanacaktır.



(a) (b) (c) (d) (e)

Şekil 5.15. Otel arama-rezervasyon işlemi arayüzleri-1

Mobil araçtaki istemci etmen, çok-etmenli platforma dahil olduktan sonra anlamsal web servislerini kullanabilmek için öncelikle mobil araçtaki arayüzden etmen platformundaki uygun servisleri görüntüleyebilmek için istekte bulunur. Şekil 5.15 (a)' da bu işlem görülmektedir. Daha sonra sistemde bulunan servisler arasından amacına uygun servisi seçer ve seçilen servisin girdi parametrelerini ister (Şekil5.15(b)(c)). Arabulucu etmen servis çağrımı için gerekli olan girdi parametrelerini içeren arayüzü mobil araçtaki etmene gönderir. Mobil araçtaki etmen gelen arayüzü ekranda gösterir (Şekil 5.15(d)). Mobil araçta otel arama kriterleri arayüzde belirtildikten sonra Şekil 5.15(e)'de görülen CallService butonuna basılması ile arabulucu etmene otel arama servisinin çağrılması için istek mesajı gönderilir. Arabulucu etmen, aracı etmenden gelen uygun otellerin listesinin gösterilebilmesi için bir arayüz hazırlar. Bu arayüz mobil araçtaki ekranda gösterilir (Şekil 5.16(a)). Rezervasyon yapılmak istenen otel seçilir ve HotelRezervation butonuna basılarak rezervasyon isteğinde bulunulur (Şekil 5.16(b)). Bu isteğin

ardından rezervasyon işlemi için gerekli olan kişi sayısı, tarih, rezervasyonu yaptıracak kişinin ismi gibi değerlerin girilebilmesi için ikinci bir dinamik arayüz ekrana gelir (Şekil 5.16(c)). Bu arayüz arabulucu etmen tarafından gönderilen otel listesini içeren mesajda bulunmaktadır. Otel listesinin görülmesinin ardından rezervasyon isteğinin oluşması olasılığı yüksek olduğu için otel listesini içeren arayüz mesaj içinde gönderilirken rezervasyon işlemi için gerekecek arayüz de yine bu mesaj içinde gönderilmektedir. Mesajın arayüz tanımlamalarını içeren kısmı aşağıdaki gibidir:

```
<fipa:argument>
<guiString><Screen> xsi='"http://www.w3.org/2001/XMLSchema-instance'"
xsi:SchemaLocation='"C://SEAGENT//SEAGENT//schema//schema.xsd'">
<Form>
<FormName>Hotel List</FormName>
     <ChoiceGroup>
            <ChoiceGroupName>Select Hotel</ChoiceGroupName>
            <Selection>1</Selection>
            <Element>Hotel Name: Princess-Star: 4-Price: 80 -Location:  izmir</Element>
            <Element>Hotel Name: Hilton-Star: 4-Price: 200 -Location:  izmir</Element>
            <Element>Hotel Name: Efes-Star: 4-Price: 80 - Location:  izmir</Element>
     </ChoiceGroup>
     <Command>
      <commandName>Hotel Rezervation</commandName>
     </Command>
     <Command>
      <commandName>BackMainForm</commandName>
```

```
     </Command>
</Form>
<Form>
<FormName>Rezervation Details</FormName>
     <TextField>
      <TextFieldTag>hotel</TextFieldTag>
      <TextFieldText></TextFieldText>
     </TextField>
     <TextField>
      <TextFieldTag>personName</TextFieldTag>
      <TextFieldText></TextFieldText>
     </TextField>
     <TextField>
      <TextFieldTag>numOfPeople</TextFieldTag>
      <TextFieldText></TextFieldText>
     </TextField>
     <TextField>
      <TextFieldTag>rezervationDate</TextFieldTag>
      <TextFieldText></TextFieldText>
     </TextField>
     <Command><commandName>Book Hotel</commandName>
     </Command>
     <Command><commandName>BackMainForm</commandName>
     </Command>
</Form>
</Screen>
</guiString>
</fipa:argument>
```

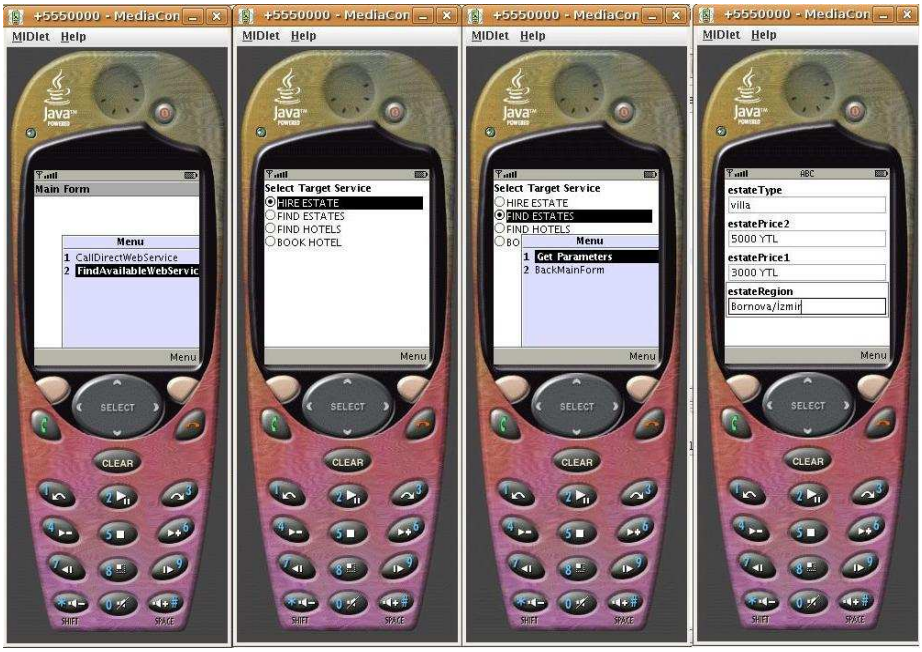
Yukarıda arayüz tanımlamalarını içeren mesajda otel arama-rezervasyon işlemi için iki arayüz gönderilmiştir. Bu sayede mobil araçtaki istemci etmenin rezervasyon işlemini gerçekleştirmek için arabulucu etmene yeniden mesaj göndermesi engellenerek rezervasyon işlemi için oluşacak fazladan mesajlaşmanın önüne geçilmiştir. Gönderilen iki arayüzden birincisi otel arama sonuçlarını içermekte, ikincisi ise otel sonuçlarından bir otel seçildikten sonra rezervasyon işlemi için gerekli bilgilerin alınmasını sağlamaktadır. Şekil 5.16(d)'de görülen mesaj içinde gönderilen ikinci arayüzde rezervasyon işleminin tamamlanması için gerekli bilgilerin girilmesinden sonra BookHotel butonuna basılır. Böylelikle arabulucu etmene rezervasyon bilgilerinin detayları gönderilir ve servis isteğinde bulunulur. Rezervasyon servisinin kullanımından sonra aracı etmenden arabulucu etmene gelen mesajda rezervasyon sonuç bilgileri bulunur. Arabulucu etmen bu bilgileri uygun arayüz tanımlaması içinde mobil araçtaki etmene gönderir. Son olarak rezervasyon sonucu mobil araçtaki ekranda gösterilir (Şekil 5.16(e)).

Otel arama-rezervasyon servisi kullanımını örnekleyen bu durum çalışmasında mobil araçtaki istemci etmen ve arabulucu etmen arasındaki mesajlaşmalar EK 2'de incelenebilir.

(a) (b) (c) (d) (e)

Şekil 5.16. Otel arama-rezervasyon işlemi arayüzleri-2

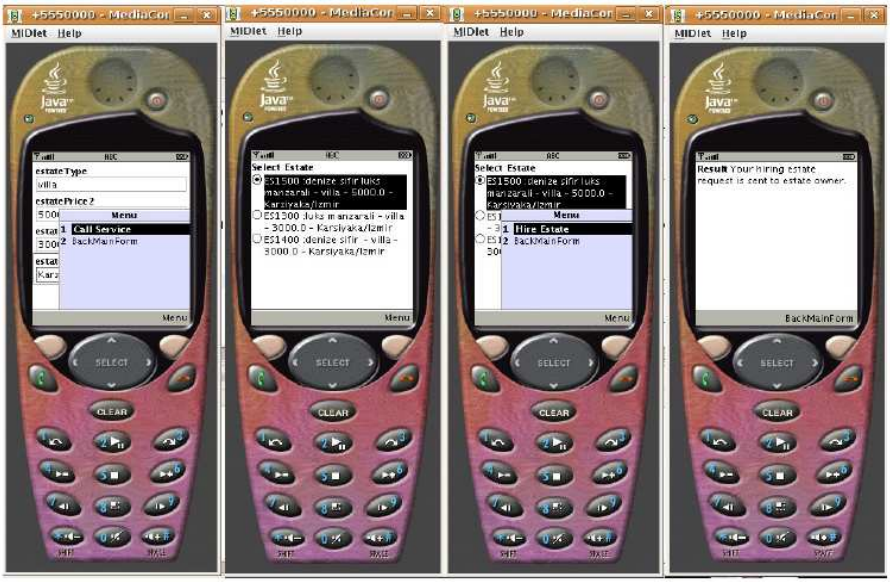
Emlak arama-kiralama işlemi gerçekleştirmek isteyen mobil araçtaki etmen Şekil 5.17 ve Şekil 5.18'de görülen arayüzleri kullanarak ilgili servisleri kullanmaktadır. Mobil araçtaki istemci etmenin otel arama-rezervasyon servisini seçene kadar Şekil 5.17 (a)-(b)-(c) 'deki arayüzleri kullanacaktır. Burada dikkat edilmesi gereken bir nokta etmen bu servisi kullanmadan önce platforma yeni servis sağlayıcı etmenler kayıt olmuştur. Şekil 5.15(b)'den farklı olarak bu sefer platformda dört servis görülmektedir. Arabulucu etmen aracı etmene yapılan yeni servis kayıtlarına göre mobil araçtaki etmeni de çalışma zamanında haberdar etmektedir.



(a) (b) (c) (d)

Şekil 5.17. Emlak arama-kiralama işlemi arayüzleri-1

Mobil araçtaki istemci etmen Şekil 5.17(b)'de görülen emlak arama servisini seçtikten sonra arabulucu etmen servis çağrımı için gerekli olan girdi parametrelerini içeren arayüzü mobil araçtaki etmene gönderir. Mobil araçtaki etmen gelen arayüzü ekranda gösterir (Şekil 5.17(d)). Kullanıcı istediği emlak özelliklerini belirtir (Şekil5.18(a)). Girdi parametrelerini arabulucu etmene gönderir. Arabulucu etmen, aracı etmenden gelen uygun emlak listesi için bir arayüz hazırlar. Bu arayüz mobil araçtaki ekranda gösterilir (Şekil5.18(b)). Kullanıcı kiralamak istediği emlağı seçer ve kiralama isteğinde bulunur (Şekil5.18(c)). Kiralama isteğinin sonucu mobil araçtaki ekranda gösterilir (Şekil5.18(d)).



(a) (b) (c) (d)

Şekil 5.18. Emlak arama-kiralama işlemi arayüzleri-2

Yapılan farklı iki işlemde de farklı özellikleri olan servisler kullanılmıştır. Her iki işlemde de mobil araçta yaratılan arayüzler xml ile tanımlanıp, mobil araçtaki etmene gönderildiğinden dolayı mobil araçtaki etmenin sadece servis çağrımını ve dinamik arayüz yaratımını bilmesi anlamsal web servislerini kullanabilmesi için yeterli olmaktadır. Farklı alanlar için tanımlanan servisler için (bilet satın alma servisi, otel rezervasyon servisi gibi) mobil araçtaki etmenin bu alanlar için düzenlemeler yapması gerekmemektedir. Önceki bölümlerde anlatılan masaüstü sistemlere dağıtılan bazı işlemler ile mobil araçtaki etmenin işlem yükü azaltılmış, mobil araçların kısıtları ve yeteneklerine uygun anlamsal web servis sunum platformu oluşturulmaya çalışılmıştır.

6. SONUÇ

Bu tezde mobil araçlardan anlamsal web servislerine erişimi sağlayan etmen tabanlı bir mimari geliştirilmiştir. Mobil araçlar için yazılım geliştirme, yazılım etmenleri, web servisleri ve anlamsal web tezin araştırma alanını oluşturmaktadır. Tez çalışması sırasında geliştirim aşamasına geçilmeden önce mobil araçlar için anlamsal web servis kullanımını zorlaştıran kısıtlar belirlenmiştir. Bu kısıtlardan biri mobil araçların anlamsal bilgiyi oluşturan ontolojiler üzerinde işlem gerçekleştirememeleridir. Bu nedenle mobil araçtaki servis istemci etmenin yerine masaüstü sistemde bulunan bir etmen gerekli işlemleri gerçekleştirmiştir. Arabulucu etmen olarak isimlendirilen bu etmen mobil araçtaki etmenin SEAGENT platformu üzerinde bulunan anlamsal web servislerine erişimi için aracı etmenle etkileşimini sağlamakla görevlidir.

Tez çalışması sırasında ortaya çıkan diğer bir gereksinim de kullanılmak istenen servis belirlendikten sonra ilgili servis için girdi parametrelerinin mobil araçtan nasıl elde edileceğinin tasarlanmasıdır. Bu gereksinim de mobil araçtaki etmenin planları ile tanımlanarak servise özgü dinamik arayüzlerin yaratılmasıyla giderilmiştir. Mobil araçtaki etmen kendisine gelen xml ile tanımlanmış arayüz ifadeleri ile servisin içeriği hakkında bilgi edinmeye ihtiyaç duymaksızın servis girdi parametreleri için gerekli olan arayüzleri yaratabilmektedir.

Tez çalışması sonucunda SEAGENT çok-etmenli platformda bulunan anlamsal web servislerinin kullanımını sağlamak için tanımlanmış olan SWSA(Semantic Web Services Architecture) kavramsal modeli kapsamında oluşturulmuş aracı etmenle, mobil araçta bulunan servis istemci etmenin anlaşabilmesi için gerekli arabulucuk işlemleri SEAGENT çok-etmenli platformda yaratılan bir arabulucu etmen ile gerçekleştirilmiştir. Arabulucu etmen ve mobil servis istemci etmenin iletişimi belirlenen FIPA belirtimlerine dayalı bir etkileşim

protokolü ile sağlanmıştır. Belirlenen gereksinimler doğrultusunda arabulucu etmenin işlevsel gereksinimleri analiz edilmiş ve bu arabulucu etmenin içsel mimarisi ortaya konulmuştur. Ayrıca tez çalışmasında oluşturulan mimaride çalışma zamanında yeni servislerin kolayca eklenebilmesi ve mobil araçtaki istemci etmenin bu servislerden eş zamanlı haberdar olabilmesi bu çalışmanın katkılarından biridir.

Bu tezde gerçekleştirilen çalışmaların sonucunda gerçekleştirilebilecek ileriye dönük çalışmalar bulunmaktadır. Mobil araç kullanıcısına ait özel bilgilerin saklanarak, bağlam duyarlılığın(context-awareness) sağlanması bu çalışmalardan biridir. Mobil araca ve kullanıcısına ait bilgilerin ve kullanıcı tercihlerinin masaüstü bir sistemde saklanıp, bu verilerden faydalanarak bir çıkarsama motoru ile elde edilen bilgilerden yararlanarak servis seçiminde daha etkin sonuçlar elde edilebilecektir. Günümüz de mobil araçların yetenekleri gözönüne alındığında mobil araç kullanıcılarına ait verilerin masaüstü sistemde saklanıp, çıkarsama motoru ile yorumlanması daha uygun görünüyor olsa da, gelecekte daha yetenekli mobil araçlar sayesinde verilerin saklanması hatta mobil araç içindeki bir çıkarsama motoru ile yorumlanabilmesi olabilirliği yüksek ihtimaller dahilindedir.

Yapılabilecek bir diğer ileriye dönük çalışma da mobil araçtaki servis istemci etmenin birtakım mobil servislerin kullanımını sağlayacak şekilde düzenlenip, mobil servis sağlayıcı etmene dönüştürülebilmesidir. Mobil araçtaki etmenlerin birbirlerinin servislerinden yararlanabileceği bu durumda mobil iletişime uygun farklı iletişim yöntemleri de gözönüne alınmalıdır. Yapılması düşünülen bu çalışma için SeagentLITE platformunda gerekli olabilecek temel Java sınıfları bulunmaktadır. İhtiyaca göre yeni iletişim yöntemleri(bluetooth gibi) SeagentLITE mimarisine kolayca eklenebilecektir. Mobil servis sunum gereksinimlerini giderildikten sonra mobil servis sağlayıcı etmen için

gerekli servis sunum planlarının oluşturulması ile servis sunum işlevi gerçekleştirilebilecektir.

KAYNAKLAR DİZİNİ

Antoniou, G., Skylogiannis, T., Bikakis, A., Doerr, M., Bassiliades, N. (2007). DR-BROKERING: A semantic brokering system. Knowl.-Based Syst. 20(1): 61-72.

Berners-Lee, T., Hendler, J., Lassila, O. (2001). The Semantic Web, Scientific American, 284:34-43.

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.

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, ACM, 1271–1272.

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, Lecture Notes in Computer Science, 3963:1-13.

Durfee E. H., Lesser V. R. and Corkill D. D., 1989, Trends in Cooperative Distributed Problem Solving, IEEE Transactions on Knowledge and Data Engineering, KDE-1:63–83.

Erdur, R.C., 2005, Yazılım Etmenleri: Genel Kavramlar,Mimariler, Standartlar, Araçlar, COMPOTEK 2005 Davetli Bildiri, İzmir.

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

Finin, T., Labrou, Y., and Mayfield, J. (1995) Evaluating of KQML as an agent communication language, ATAL, 347-360.

Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1995, Design Patterns: Elements of Reusable Object-Oriented Software, Boston, Addison-Wesley Professional Computing Series, 395p.

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

KAYNAKLAR DİZİNİ (devam)

Gümüş, Ö. (2008). Anlamsal Veb Servisleri Ortaminda Bir Araci Etmen Tasarimi Ve Gerçekleştirimi. Doktora tezi, Ege Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Ana Bilim Dalı, (Tamamlanmış).

Gümüş, Ö., Gürcan, Ö., Dikenelli, O. "Towards a Broker Agent in the Semantic Services Environment". Service-Oriented Computing and Agent-Based Engineering (SOCASE'2008) held at The Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'2008), Estoril, Portugal, 5006:31-44 (to be appeared).

Hwang, Y., Oh, I., Im, H., Lee, K., Lee, K., Lee, S. (2007). UWS Broker for Ubiquitous Web Services Dynamic Discovery. In: Proceedings of the The 2007 International Conference on Intelligent Pervasive Computing, 204-209.

Küçükkeleş, H. (2008). Mobil Araçlarda Çalışan Etmenler İçin Bir Planlayıcı Tasarımı ve Gerçekleştirimi. Yüksek lisans tezi, Ege Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Ana Bilim Dalı, (Tamamlanmamış).

Luck, M., McBurney P., Shehory, O., Willmott, S. (2005). Agent Technology: Computing as Interaction, A Roadmap for Agent Based Computing", AgentLink III, U.K.

Mahmoud, Q.H., Al-Masri, E. (2007). MSM: A Middleware Architecture for Enhancing Interaction with Mobile Services. Wireless Pervasive Computing, 2007. ISWPC '07.

Mary, F. (2006). Providing Web Services to Mobile Users Using Agent Technology. ICWN 2006: 430-435.

Ratsimor, O., Chakraborty, D., Joshi, A., Finin, T., Yesha, Y. (2007). Service discovery in agent-based pervasive computing environments. Mobile Networks and Applications Volume 9 , Issue 6, 679 – 692.

Russell S. J. and Norvig P., 2003, Artificial Intelligence: A Modern Approach Second Edition, Pearson Education, USA, 1080p.

KAYNAKLAR DİZİNİ (devam)

Sycara, K., Decker, K., Pannu, A., Williamson, M. and Zeng, D. (1996). Distributed Intelligent Agents, IEEE Expert, Intelligent Systems and Their Applications, 11(6):36-45.

Sycara, K., Paolucci, M., Soudry, J., Srinivasan, N. (2004). Dynamic discovery and coordination. of agent-based semantic Web Services. IEEE Internet. Computing, vol. 8, no. 3, 66-73.

Türker, M. (2001). Web Servisleri. INET-TR 2001 7nci Türkiye'de Internet Konferansı, İstanbul, Türkiye.

Varga, L. Z., kos Hajnal and Werner, Z., 2003, Engineering Web Service Invocations from Agent Systems, Lecture Notes in Computer Science, 2691:626-635.

Veijalainen, J., Nikitin, S., Törmälä, V. (2006) Ontology based Semantic Web Service platform in Mobile Environments. In: Proceedings of the 7th International Conference on Mobile Data Management (MDM'06), Nara, Japan.

Wang, R., Chang, Y., Chang, R. (2007). Design Issues of Semantic Service Discovery for Ubiquitous Computing. Multimedia and Ubiquitous Engineering, 2007. MUE '07.

EKLER

Ek 1 Arabulucu Etmen Planları

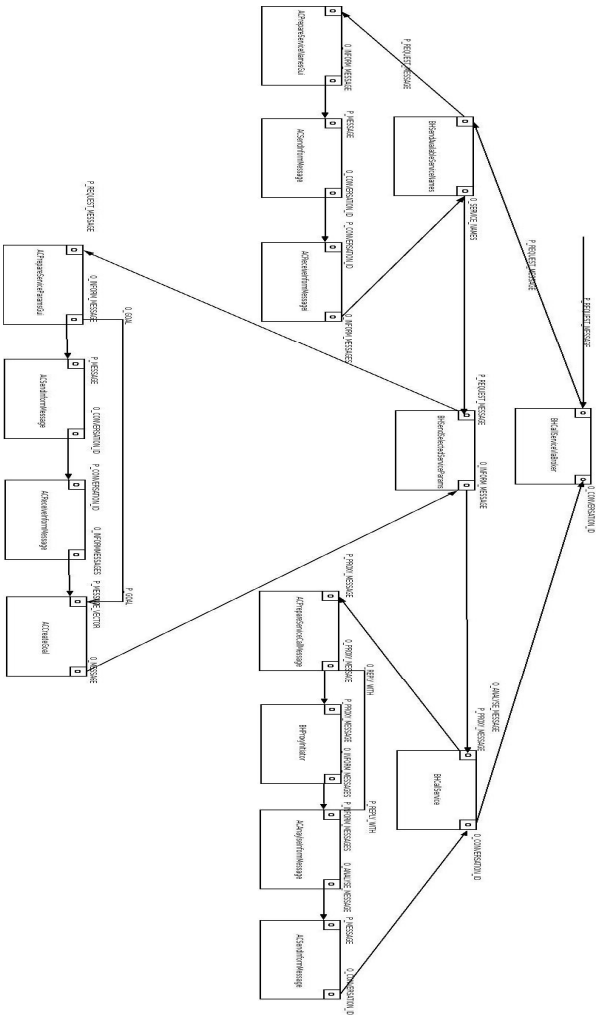
Ek 2 Etmenler Arası Mesaj Örnekleri

Ek 3 Dönüşüm İşlemi Örneği

Ek 4 İngilizce Terimler Sözlüğü

Ek 1 Arabulucu Etmen Planları

Bölüm 5.1.'de açıklanmış olan arabulucu etmenin aracı etmen üzerinden anlamsal servis çağırdığı planının tüm alt planları Şekil A.1'de görülmektedir.



Şekil A.1. Arabulucu etmen anlamsal servis çağırma planı

BHCallServiceViaBroker planı arabulucu etmen trafından kullanılan bir plandır. Bu plan aracı etmen vasıtasıyla anlamsal wev servislerini çağırmayı sağlar. Üç alt sınıftan oluşur:

BHSendAvailableServiceNames, BHSendSelectedServiceParameters, BHCallService.

BHCallServiceViaBroker sınıfı diğer bütün planlar gibi SEAGENT projesinde bulunan Behaviour sınıfından türetilmiştir. Aşağıda BHCallServiceViaBroker sınıfının kodları bulunmaktadır.

```
package mediator.plans.callservice;

import tr.edu.ege.seagent.planner.exception.TaskException;
import tr.edu.ege.seagent.planner.task.Behaviour;
import tr.edu.ege.seagent.planner.task.Disinheritance;
import tr.edu.ege.seagent.planner.task.Inheritance;
import tr.edu.ege.seagent.planner.task.InternalProvision;
import tr.edu.ege.seagent.planner.task.Task;

public class BHCallServiceViaBroker extends Behaviour {

	public static final String P_REQUEST_MESSAGE =
		"requestmessageProv1";

	public static final String O_CONVERSATION_ID =
		"bhcallserviceviabrokerOut";

	public BHCallServiceViaBroker() throws TaskException
		{
		super();
	}

	public BHCallServiceViaBroker(int planNo, Behaviour
		superBehaviour)throws TaskException {
		super(planNo, superBehaviour);

	}

	@Override
	protected void defineLinks() throws TaskException {

	defineInheritance(new Inheritance (P_REQUEST_MESSAGE,
	BHSendAvailableServiceNames.class.getName(),
```

```
BHSendAvailableServiceNames.P_REQUEST_MESSAGE));

defineProvisionLink(BHSendAvailableServiceNames.
      class.getName(),
      BHSendAvailableServiceNames.O_SERVICE_NAME,
      "OK", BHSendSelectedServiceParameters.
      class.getName(),BHSendSelectedServiceParameters
      .P_REQUEST_MESS_GES);

defineProvisionLink(
      BHSendSelectedServiceParameters.class.getName()
      ,
      BHSendSelectedServiceParameters.O_INFORM_MESSAG
      E, "OK",BHCallService.class.getName(),
      BHCallService.P_PROXY_MESSAGE);

defineDisinheritance(
      newDisinheritance(BHCallService.class.getName()
      ,"OK", BHCallService.O_CONVERSATION_ID,
      O_CONVERSATION_ID));
}

@Override
protected void defineSubTasks() {

defineSubTask(BHSendAvailableServiceNames.class.getNa
    me());

defineSubTask(BHSendSelectedServiceParameters.class.g
etName());
      defineSubTask(BHCallService.class.getName());

}

protected Task createTask(String className) {
      try {
            if
            (className.equals(BHSendAvailableServiceN
            ames .class.getName())) {

                  return new
                     BHSendAvailableServiceNames
                     (this.getPlanNo(), this);
            } else if (className.equals
            (BHSendSelectedServiceParameters.class.ge
            tName())) {
```

```
                        return new
                           BHSendSelectedServiceParameters
                           (this.getPlanNo(),this);
                  }else if (className.equals(BHCallService
                  .class.getName()))
                  {
                        return new
                        BHCallService(this.getPlanNo(),
                        this);
                  }
            } catch (TaskException e)
                  {e.printStackTrace();}
            return null;
      }

      @Override
      protected void defineTaskSpecificProperties() {

            defineName(BHCallServiceViaBroker.class.
                  getName());
            defineOutcome(O_CONVERSATION_ID);
            defineProvision(new InternalProvision(
              P_REQUEST_MESSAGE));
      }
}
```

Şekil A.2. BHCallServiceViaBroker kod örneği

Ek 2 Etmenler Arası Mesaj Örnekleri

Bu bölümde Bölüm 5.4'te açıklanan durum çalışmalarında arabulucu etmenle aracı etmen arasındaki mesajlar bulunmaktadır. Aşağıdaki mesajlar otel arama-rezervasyon servisi kullanmak isteyen mobil istemci etmen ve arabulucu etmen arasındaki mesajlardır.

Mobil servis istemci etmenin arabulucu etmene gönderdiği ilk mesaj:

Bu mesajdaki mobil servis istemci etmen arabulucu etmen SEAGENT platformunda bulunan anlamsal web servislerinin tanımlarını istemektedir.

```
<?xml version="1.0"?>
<envelope>
<params index="1">
<to>socket://155.223.24.62:2226</to>
<from>socket://155.223.24.96:5005</from>
<intended-
receiver>socket://155.223.24.62:2226</intended-
receiver>
</params>
</envelope>
$$$(request
        :sender
        (agent-identifier
        :name mobileServiceRequesterAgent@brokering.com
        :addresses socket://155.223.24.96:5005
        )
        :receiver
        (agent-identifier
        :name mediatorAgent@brokering.com
        :addresses socket://155.223.24.62:2226
```

```
        )
        :reply-with 14-replyMessage
        :conversation-id 0
        :content <?xml version="1.0"?>
        <rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
        syntax-ns#"
        xmlns:fipa="http://www.fipa.org/schemas#">
        <fipa:Action rdf:ID="action">
        <fipa:actor>
        <agent-identifier>

  <name>mobileServiceRequesterAgent@brokering.com</name
>
        <addresses>
              <url>socket://155.223.24.96:5005</url>
        </addresses>
        </agent-identifier>
        </fipa:actor>
        <fipa:act>CallServiceViaBroker</fipa:act>
        <fipa:done>false</fipa:done>
</fipa:Action>
</rdf:RDF>
)
```

Arabulucu etmenden gelen cevap mesajı:

Gelen mesajda arabulucu etmen SEAGENT platformunda bulunan anlamsal web servislerini uygun arayüz formatı içinde göndermektedir.

```
<?xml version="1.0"?>
<envelope>
 <params index="1">
 <to>155.223.24.96:5005 </to>
```

```
 <from>155.223.24.62:2226</from>
</params>
</envelope>
 $$$(inform
 :sender
 (agent-identifier
 :name mediatorAgent@brokering.com
 :addresses 155.223.24.62:2226
 )
 :receiver
 (agent-identifier
 :name mobileServiceRequesterAgent@brokering.com
 :addresses 155.223.24.96:5005
 )
 :in-reply-to 14-replyMessage
 :reply-with service-name
 :conversation-id 1
 :content
 <?xml version="1.0"?>
 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#" xmlns:fipa="http://www.fipa.org/schemas#">
 <fipa:Action rdf:ID="action">
 <fipa:actor>
       <agent-identifier>
       <name>mediatorAgent@brokering.com</name>
       <addresses>
             <url>155.223.24.62:2226</url>
       </addresses>
       </agent-identifier>
 </fipa:actor>
```

```
 <fipa:act>inform</fipa:act>
 <fipa:argument>
       <guiString>
       <Screen>
       xsi='"http://www.w3.org/2001/XMLSchema-
instance'"
xsi:SchemaLocation='"C://SEAGENT//SEAGENT//schema//sch
ema.xsd'">
       <Form>
       <FormName>formName</FormName>
       <ChoiceGroup>
             <ChoiceGroupName>
             Select Target Service
             </ChoiceGroupName>
             <Selection>1</Selection>
             <Element>HIRE ESTATE</Element>
             <Element>FIND ESTATES</Element>
             <Element>FIND HOTELS</Element>
             <Element>BOOK HOTEL</Element>
       </ChoiceGroup>
       <Command>
             <commandName>Get Parameters</commandName>
       </Command>
       <Command>
             <commandName>BackMainForm</commandName>
       </Command>
       </Form>
       </Screen>
       </guiString>
</fipa:argument>
<fipa:done>true</fipa:done>
```

```
</fipa:Action>
</rdf:RDF>)
```

Mobil servis istemci etmenin servis seçtikten sonra gönderdiği mesaj:

Mobil servis istemci etmen otel arama servisini seçtikten sonra aşağıdaki mesajı arabulucu etmene gönderir. Mesaj içeriğinde seçilen servis ismi bulunur ve seçilen servisin girdi parametreleri istenir.

```
<?xml version="1.0"?>
<envelope>
 <params index="1">
 <to>socket://155.223.24.62:2226</to>
 <from>socket://155.223.24.96:5005</from>
 <intended-
 receiver>socket://155.223.24.62:2226</intended-
 receiver>
 </params>
</envelope>
$$$(inform
 :sender
 (agent-identifier
 :name mobileServiceRequesterAgent@brokering.com
 :addresses socket://155.223.24.96:5005
 )
 :receiver
 (agent-identifier
 :name mediatorAgent@brokering.com
 :addresses socket://155.223.24.62:2226
 )
 :in-reply-to service-name
 :reply-with 74-replyMessage
```

```
 :conversation-id 11
 :content <?xml version="1.0"?>
       <rdf:RDF
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
       syntax-ns#"
       xmlns:fipa="http://www.fipa.org/schemas#">
       <fipa:Action rdf:ID="action">
       <fipa:actor><agent-identifier>

 <name>mobileServiceRequesterAgent@brokering.com</name
>
       <addresses>
       <url>socket://155.223.24.96:5005</url>
       </addresses>
       </agent-identifier>
       </fipa:actor>

 <fipa:act>getAvailableServiceParameters</fipa:act>
       <fipa:argument>
             <string>FIND HOTELS</string>
       </fipa:argument>
</fipa:Action>
</rdf:RDF>)
```

Arabulucu etmenin gönderdiği servis girdi parametrelerini içeren mesaj:

```
<?xml version="1.0"?>
<envelope>
 <params index="1">
 <to>155.223.24.96:5005 </to>
 <from>155.223.24.62:2226</from>
 </params>
</envelope>
```

```
$$$(inform
 :sender
 (agent-identifier
       :name mediatorAgent@brokering.com
       :addresses 155.223.24.62:2226
 )
 :receiver
 (agent-identifier
       :name mobileServiceRequesterAgent@brokering.com
       :addresses 155.223.24.96:5005
 )
 :in-reply-to 32-replyMessage
 :reply-with aa
 :conversation-id 4
 :content <?xml version="1.0"?>
       <rdf:RDF
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
       syntax-ns#"
       xmlns:fipa="http://www.fipa.org/schemas#">
       <fipa:Action rdf:ID="action">
       <fipa:actor><agent-identifier>
       <name>mediatorAgent@brokering.com</name>
       <addresses>
             <url>155.223.24.62:2226</url>
       </addresses>
       </agent-identifier>
       </fipa:actor>
       <fipa:act>inform</fipa:act>
       <fipa:argument>
       <guiString>
```

```
<Screen>
xsi='"http://www.w3.org/2001/XMLSchema-
instance'"xsi:SchemaLocation='"C://SEAGENT//SEA
GENT//schema//schema.xsd'">
<Form>
<FormName>formName</FormName>
<TextField>

<TextFieldTag>hotelLocation</TextFieldTag>
      <TextFieldText>-</TextFieldText>
</TextField>
<TextField>
      <TextFieldTag>hotelStar</TextFieldTag>
      <TextFieldText>-</TextFieldText>
</TextField>
<TextField>
      <TextFieldTag>hotelPrice2</TextFieldTag>
      <TextFieldText>-</TextFieldText>
</TextField>
<TextField>
      <TextFieldTag>hotelPrice1</TextFieldTag>
      <TextFieldText>-</TextFieldText>
</TextField>
<Command>
      <commandName>Call Service</commandName>
</Command>
<Command>
      <commandName>BackMainForm</commandName>
</Command>
</Form>
</Screen>
```

```
        </guiString>
        </fipa:argument>
        <fipa:done>false</fipa:done>
        <fipa:results></fipa:results>
        </fipa:Action>
        </rdf:RDF>)
```

Mobil servis istemci etmenin servis girdi parametre değerlerini gönderdiği mesaj:

```
<?xml version="1.0"?>
<envelope>
 <params index="1">
 <to>socket://155.223.24.62:2226</to>
 <from>socket://155.223.24.96:5005</from>
 <intended-
 receiver>socket://155.223.24.62:2226</intended-
 receiver>
 </params>
</envelope>
$$$(inform
 :sender
 (agent-identifier
       :name mobileServiceRequesterAgent@brokering.com
       :addresses socket://155.223.24.96:5005
 )
 :receiver
 (agent-identifier
       :name mediatorAgent@brokering.com
       :addresses socket://155.223.24.62:2226
 )
 :in-reply-to aa
```

```
 :reply-with 38-replyMessage
 :conversation-id 4
 :content <?xml version="1.0"?>
       <rdf:RDF
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
       syntax-ns#"
       xmlns:fipa="http://www.fipa.org/schemas#">
       <fipa:Action rdf:ID="action">
       <fipa:actor><agent-identifier>

 <name>mobileServiceRequesterAgent@brokering.com</name
>
       <addresses>
             <url>socket://155.223.24.96:5005</url>
       </addresses>
       </agent-identifier>
       </fipa:actor>
       <fipa:act>callWebService</fipa:act>
       <fipa:argument>
             <goal>
             <goalName>find hotel</goalName>
             <inputs>
             <input>
                   <name>hotelLocation</name>
                   <value>izmir</value>
                   <type>string</type>
             </input>
             <input>
                   <name>hotelStar</name>
                   <value>4</value>
                   <type>string</type>
             </input>
```

```
                <input>
                      <name>hotelPrice2</name>
                      <value>200 YTL</value>
                      <type>string</type>
                </input>
                <input>
                      <name>hotelPrice1</name>
                      <value>80 YTL</value>
                      <type>string</type>
                </input>
                </inputs>
                </goal>
          </fipa:argument>
          </fipa:Action>
          </rdf:RDF>)
```

Arabulucu etmenin uygun otelleri içeren arayüzü gönderdiği mesaj:

```
<?xml version="1.0"?>
<envelope>
      <params index="1">
      <to>155.223.24.96:5005 </to>
      <from>155.223.24.62:2226</from>
      </params>
</envelope>
$$$(inform
:sender
(agent-identifier
      :name mediatorAgent@brokering.com
      :addresses 155.223.24.62:2226
)
:receiver
```

```
(agent-identifier
      :name mobileServiceRequesterAgent@brokering.com
      :addresses 155.223.24.96:5005
)
:in-reply-to 44-replyMessage
:conversation-id 8
:content <?xml version="1.0"?>
       <rdf:RDF
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
       syntax-ns#"
       xmlns:fipa="http://www.fipa.org/schemas#">
      <fipa:Action rdf:ID="action">
      <fipa:actor><agent-identifier>
      <name>mediatorAgent@brokering.com</name>
      <addresses>
            <url>155.223.24.62:2226</url>
      </addresses>
      </agent-identifier>
      </fipa:actor>
      <fipa:act>inform</fipa:act>
      <fipa:argument>
      <guiString>
       <Screen>
       xsi='"http://www.w3.org/2001/XMLSchema-
       instance'"xsi:SchemaLocation='"C://SEAGENT//S
       EAGENT//schema//schema.xsd'">
      <Form>
      <FormName>formName</FormName>
      <ChoiceGroup>
            <ChoiceGroupName>Select
Hotel</ChoiceGroupName>
            <Selection>1</Selection>
```

```
                    <Element>Hotel Name: Princess-Star:
                    4-Price: 80 -Location:
                    izmir</Element>
                    <Element>Hotel Name: Hilton-Star:
                    4-Price: 200 -Location:
                    izmir</Element>
                    <Element>Hotel Name: Efes-Star: 4-
                    Price: 80 -Location:
                    izmir</Element>
        </ChoiceGroup>
        <Command>
              <commandName>Hotel
Rezervation</commandName>
        </Command>
        <Command>
              <commandName>BackMainForm</commandName>
        </Command>
        </Form>
        </Screen>
         <Screen>
         xsi='"http://www.w3.org/2001/XMLSchema-
         instance'"xsi:SchemaLocation='"C://SEAGENT//S
         EAGENT//schema//schema.xsd'">
        <Form>
        <FormName>formName</FormName>
        <TextField>
              <TextFieldTag>hotel</TextFieldTag>
              <TextFieldText></TextFieldText>
        </TextField>
        <TextField>
              <TextFieldTag>personName</TextFieldTag>
              <TextFieldText></TextFieldText>
        </TextField>
```

```
        <TextField>
              <TextFieldTag>numOfPeople</TextFieldTag>
              <TextFieldText></TextFieldText>
        </TextField>
        <TextField>

  <TextFieldTag>rezervationDate</TextFieldTag>
              <TextFieldText></TextFieldText>
        </TextField>
        <Command>
              <commandName>Book Hotel</commandName>
        </Command>
        <Command>
              <commandName>BackMainForm</commandName>
        </Command>
        </Form>
        </Screen>
        </guiString>
        </fipa:argument>
   </fipa:Action>
   </rdf:RDF>)
```

Bu mesajdan sonra mobil araçtaki ekranda bulunan uygun oteller görüntülenir. İstenen otel seçildikten sonra ekrana gelen ikinci arayüz sayesinde rezervasyon isteği için gerekli olan bilgiler girilir. Ardından mobil araçtaki istemci etmen rezervasyon servisi kullanmak için arabulucu etmene mesaj gönderir.

Mobil istemci etmenin gönderdiği rezervasyon istek mesajı:

```
<?xml version="1.0"?>
<envelope>
 <params index="1">
```

```
 <to>socket://155.223.24.62:2226</to>
 <from>socket://155.223.24.96:5005</from>
 <intended-
receiver>socket://155.223.24.62:2226</intended-
receiver>
 </params>
</envelope>
$$$(request
 :sender
 (agent-identifier
       :name mobileServiceRequesterAgent@brokering.com
       :addresses socket://155.223.24.96:5005
 )
 :receiver
 (agent-identifier
       :name mediatorAgent@brokering.com
       :addresses socket://155.223.24.62:2226
 )
 :reply-with 32-replyMessage
 :conversation-id 4
 :content <?xml version="1.0"?>
       <rdf:RDF
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
       syntax-ns#"
       xmlns:fipa="http://www.fipa.org/schemas#">
       <fipa:Action rdf:ID="action">
       <fipa:actor><agent-identifier>
       <name>mediatorAgent@brokering.com</name>
       <addresses>
             <url>socket://155.223.24.62:2226</url>
       </addresses>
       </agent-identifier>
```

```
</fipa:actor>
<fipa:act>callWebService</fipa:act>
<fipa:argument>
<goal>
<goalName>Book Hotel</goalName>
<inputs>
<input>
      <name>rezervationDate</name>
      <value>02.08.08</value>
      <type>string</type>
</input>
<input>
      <name>hotel</name>
      <value> Princess</value>
      <type>string</type>
</input>
<input>
      <name>personName</name>
      <value>Gizem Okkalıoğlu</value>
      <type>string</type>
</input>
<input>
      <name>numOfPeople</name>
      <value>2</value>
      <type>string</type>
</input>
</inputs>
</goal>
</fipa:argument>
<fipa:done>false</fipa:done>
```

```
<fipa:results></fipa:results>
</fipa:Action>
</rdf:RDF>)
```

Arabulucu etmenden gelen rezervasyon sunuç mesajı:

```
<?xml version="1.0"?>
<envelope>
 <params index="1">
 <to>155.223.24.96:5005 </to>
 <from>155.223.24.62:2226</from>
 </params>
</envelope>
$$$(inform
 :sender
 (agent-identifier
       :name mediatorAgent@brokering.com
       :addresses 155.223.24.62:2226
 )
 :receiver
 (agent-identifier
       :name mobileServiceRequesterAgent@brokering.com
       :addresses 155.223.24.96:5005
 )
 :in-reply-to 32-replyMessage
 :conversation-id 6
 :content <?xml version="1.0"?>
       <rdf:RDF
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
       syntax-ns#"
       xmlns:fipa="http://www.fipa.org/schemas#">
       <fipa:Action rdf:ID="action">
       <fipa:actor><agent-identifier>
```

```
<name>mediatorAgent@brokering.com</name>
<addresses>
      <url>155.223.24.62:2226</url>
</addresses>
</agent-identifier>
</fipa:actor>
<fipa:act>inform</fipa:act>
<fipa:argument>
<guiString>
<Screen>
xsi='"http://www.w3.org/2001/XMLSchema-
instance'"xsi:SchemaLocation='"C://SEAGENT//SEA
GENT//schema//schema.xsd'">
<Form>
<FormName>formName</FormName>
<stringItem>
<stringItemLabel>Result</stringItemLabel>
<stringItemText>Rezervation is completed.
       Hotel name :  Princess
      personName : Gizem Okkalioglu
      rezervation date : 02.08.08
       room number : 365</stringItemText>
</stringItem>
<Command>
      <commandName>BackMainForm</commandName>
</Command>
</Form>
</Screen>
</guiString>
</fipa:argument>
<fipa:done>false</fipa:done>
```

```
            <fipa:results></fipa:results>
            </fipa:Action>
      </rdf:RDF>)
```

Ek 3 Dönüşüm İşlemi Örneği

Örnek OWL-S Dosyası:

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<rdf:RDF    xmlns:owl = http://www.w3.org/2002/07/owl#
            xmlns:rdfs = http://www.w3.org/2000/01/rdf-
      schema#
            xmlns:rdf = http://www.w3.org/1999/02/22-rdf-
      syntax-ns#
            xmlns:service =
            "http://www.daml.org/services/owls/
            1.1/Service.owl#"
            xmlns:process =
            "http://www.daml.org/services/owl-s/
            1.1/Process.owl#"
            xmlns:profile =
            "http://www.daml.org/services/owl-s/
            1.1/Profile.owl#"
            xmlns:grounding =
            "http://www.daml.org/services/owl-s/
            1.1/Grounding.owl#"
      xml:base = "http://etmen.ege.edu.tr/services/1.1/
      find_Hotels_Service.owls">
      <owl:Ontology rdf:about="">
      <owl:imports
        rdf:resource="http://etmen.ege.edu.tr/ontology/Serv
        ice.owl" />
      <owl:imports
        rdf:resource="http://etmen.ege.edu.tr/ontology/Proc
        ess.owl" />
      <owl:imports
        rdf:resource="http://etmen.ege.edu.tr/ontology/Prof
        ile.owl" />
      <owl:imports
        rdf:resource="http://etmen.ege.edu.tr/ontology/Grou
        nding.owl" />
      </owl:Ontology>

      <service:Service rdf:ID="findHotelsService">
      <service:presents rdf:resource="#findHotelsProfile"/>
      <service:describedBy
      rdf:resource="#findHotelsProcessModel"/>
      <service:supports
      rdf:resource="#findHotelsGrounding"/>
      </service:Service>
```

```
<profile:Profile rdf:ID="findHotelsProfile">
      <service:isPresentedBy
      rdf:resource="#findHotelsService"/>
      <profile:serviceName xml:lang="en">
      findHotelsService
      </profile:serviceName>
      <profile:textDescription xml:lang="en">
      This is a recommended service to know about
      the best destination of the sports.
      </profile:textDescription>

      <profile:hasInput  rdf:resource="#hotelLocation"/>
      <profile:hasInput  rdf:resource="#hotelPrice1"/>
      <profile:hasInput  rdf:resource="#hotelPrice2"/>
      <profile:hasInput  rdf:resource="#hotelStar"/>
      <profile:hasOutput rdf:resource="#findHotelsReturn"/>

      <profile:has_process
rdf:resource="findHotelsProcess"/> </profile:Profile>

<process:ProcessModel rdf:ID="findHotelsProcessModel">
      <service:describes
      rdf:resource="#findHotelsService"/>
      <process:hasProcess
      rdf:resource="#findHotelsProcess"/>
      </process:ProcessModel>

      <process:AtomicProcess rdf:ID="findHotelsProcess">
      <process:hasInput  rdf:resource="#hotelLocation"/>
      <process:hasInput  rdf:resource="#hotelPrice1"/>
      <process:hasInput  rdf:resource="#hotelPrice2"/>
      <process:hasInput  rdf:resource="#hotelStar"/>
      <process:hasOutput rdf:resource="#findHotelsReturn"/>
      </process:AtomicProcess>

      <process:Input rdf:ID="hotelLocation">
        <rdfs:label>hotelLocation</rdfs:label>
       <process:parameterType
       rdf:datatype="http://www.w3.org/2001
       /XMLSchema#anyURI">http://www.w3.org/2001/XMLSchema
       #string
       </process:parameterType>
      </process:Input>
       <process:Input rdf:ID="hotelPrice1">
        <rdfs:label>hotelPrice1</rdfs:label>
       <process:parameterType
       rdf:datatype="http://www.w3.org/2001
       /XMLSchema#anyURI">
```

```
        http://www.w3.org/2001/XMLSchema#string
        </process:parameterType>
      </process:Input>
      <process:Input rdf:ID="hotelPrice2">
        <rdfs:label>hotelPrice2</rdfs:label>
         <process:parameterType
        rdf:datatype="http://www.w3.org/2001
        /XMLSchema#anyURI">http://www.w3.org/2001/XMLSchema
        #string
        </process:parameterType>
      </process:Input>
       <process:Input rdf:ID="hotelStar">
        <rdfs:label>hotelStar</rdfs:label>
         <process:parameterType
        rdf:datatype="http://www.w3.org/2001
        /XMLSchema#anyURI">
        http://www.w3.org/2001/XMLSchema#string
        </process:parameterType>
      </process:Input>
      <process:Output rdf:ID="findHotelsReturn">
        <rdfs:label>findHotelsReturn</rdfs:label>
        <process:parameterType
        rdf:datatype="http://www.w3.org/2001
        /XMLSchema#anyURI">
        http://www.w3.org/2002/07/owl#Thing
        </process:parameterType>
      </process:Output>

<grounding:WsdlGrounding rdf:ID="findHotelsGrounding">
 <grounding:hasAtomicProcessGrounding>
      <grounding:WsdlAtomicProcessGrounding
         rdf:ID="findHotelsAtomicProcessGrounding">
      <grounding:wsdlDocument
         rdf:datatype="http://www.w3.org/2001/
         XMLSchema#anyURI">http://155.223.24.62:8080/axis/s
         ervices/
         HotelService?wsdl</grounding:wsdlDocument>
      <grounding:owlsProcess>
          <process:AtomicProcess
rdf:ID="findHotelsProcess">
            <process:hasOutput
rdf:resource="#findHotelsReturn"/>
            <process:hasInput
rdf:resource="#hotelLocation"/>
            <process:hasInput
rdf:resource="#hotelPrice1"/>
            <process:hasInput
rdf:resource="#hotelPrice2"/>
```

```
            <process:hasInput  rdf:resource="#hotelStar"/>
            <service:describes
rdf:resource="#findHotelsService"/>
            <rdfs:label
            rdf:datatype="http://www.w3.org/2001
            /XMLSchema#string"
            >findHotelsProcess</rdfs:label>
          </process:AtomicProcess>
        </grounding:owlsProcess>
         <grounding:wsdlInput>
            <grounding:WsdlInputMessageMap>
              <grounding:owlsParameter
rdf:resource="#hotelStar"/>
               <grounding:wsdlMessagePart
                   rdf:datatype="http://
                   www.w3.org/2001/XMLSchema#anyURI">
                   http://155.223.24.62:8080/axis/services/
                   HotelService?wsdl#hotelStar
                   </grounding:wsdlMessagePart>
            </grounding:WsdlInputMessageMap>
          </grounding:wsdlInput>
          <grounding:wsdlInput>
            <grounding:WsdlInputMessageMap>
              <grounding:owlsParameter
rdf:resource="#hotelPrice2"/>
                   <grounding:wsdlMessagePart
                   rdf:datatype="http://
                   www.w3.org/2001/XMLSchema#anyURI">
                   http://155.223.24.62:8080/axis/services/
                   HotelService?wsdl#hotelPrice2
                   </grounding:wsdlMessagePart>
            </grounding:WsdlInputMessageMap>
          </grounding:wsdlInput>
          <grounding:wsdlInput>
            <grounding:WsdlInputMessageMap>
              <grounding:owlsParameter
rdf:resource="#hotelPrice1"/>
              <grounding:wsdlMessagePart
              rdf:datatype="http://
              www.w3.org/2001/XMLSchema#anyURI">
              http://155.223.24.62:8080/axis/services/
              HotelService?wsdl#hotelPrice1
            </grounding:wsdlMessagePart>
            </grounding:WsdlInputMessageMap>
          </grounding:wsdlInput>
          <grounding:wsdlInput>
            <grounding:WsdlInputMessageMap>
```

```
              <grounding:owlsParameter
df:resource="#hotelLocation"/>
              <grounding:wsdlMessagePart
              rdf:datatype="http://
              www.w3.org/2001/XMLSchema#anyURI">
              http://155.223.24.62:8080/axis/services/
              HotelService?wsdl#hotelLocation
              </grounding:wsdlMessagePart>
            </grounding:WsdlInputMessageMap>
          </grounding:wsdlInput>
         <grounding:wsdlOperation>
            <grounding:WsdlOperationRef>
              <grounding:operation
              rdf:datatype="http://www.w3.org
              /2001/XMLSchema#anyURI">
              http://155.223.24.62:8080/axis/services/
              HotelService?wsdl#findHotels
              </grounding:operation>
              <grounding:portType
              rdf:datatype="http://www.w3.org
              /2001/XMLSchema#anyURI">
              http://155.223.24.62:8080/axis/services/
              HotelService?wsdl#HotelService
              </grounding:portType>
            </grounding:WsdlOperationRef>
          </grounding:wsdlOperation>
       <grounding:wsdlOutput>
            <grounding:WsdlOutputMessageMap>
              <grounding:owlsParameter
rdf:resource="#findHotelsReturn"/>
                  <grounding:wsdlMessagePart
                  rdf:datatype="http://
                  www.w3.org/2001/XMLSchema#anyURI">
                  http://155.223.24.62:8080/axis/services/
                  HotelService?wsdl#findHotelsReturn
                  </grounding:wsdlMessagePart>
            </grounding:WsdlOutputMessageMap>
          </grounding:wsdlOutput>

         <grounding:wsdlInputMessage
        rdf:datatype="http://www.w3.org
        /2001/XMLSchema#anyURI">
        http://localhost:8080/axis/services/
        HotelService#findHotelsRequest
        </grounding:wsdlInputMessage>
        <grounding:wsdlOutputMessage
        rdf:datatype="http://www.w3.org/
        2001/XMLSchema#anyURI">
```

```
        http://localhost:8080/axis/services
        /HotelService#findHotelsResponse
        </grounding:wsdlOutputMessage>
      </grounding:WsdlAtomicProcessGrounding>
    </grounding:hasAtomicProcessGrounding>
<service:supportedBy rdf:resource="#findHotelsService"/>
</grounding:WsdlGrounding>

</rdf:RDF>
```

Örnek OWL-S dosyasının Arabulucu Etmen Tarafından Dönüştürülmüş Hali:

```
<goals>
      <goal name="FIND HOTELS"
      goalName="find_Hotels_Service">
            <inputs>
                  <input      name="hotelLocation"
                              value=""
                              type="XMLType.XSD_STRING"/>
                  <input      name="hotelStar"
                              value=""
                              type="XMLType.XSD_STRING"/>
                  <input      name="hotelPrice2"
                              value=""
                              type="XMLType.XSD_STRING"/>
                  <input      name="hotelPrice1"
                              value=""
                              type="XMLType.XSD_STRING"/>
            </inputs>
            <outputs>
                  <output     name="findHotelsReturn"
                              value=""
                              type="XMLType.XSD_ANY"/>
            </outputs>
      </goal>

</goals>
```

ACUpdateServiceProfiles sınıfı kodları:

```
/**
 * Bu sınıf basitleştirilmiş servis profillerini oluşturan
plandır.
 *
 * @author Gizem Okkalıoğlu
 *
 */
public class ACUpdateServiceProfiles extends Action {

	private SimplifiedServiceProfile serviceProfile = new
SimplifiedServiceProfile();


	protected ACUpdateServiceProfiles() throws
TaskException {
		super();

	}

	@Override
	public void Do() throws Exception {

	// aracı etmenin gönderdiği kaydedilen servis
	// sağlayıcı etmenin bilgilerinin bulunduğu mesaj
	Message message = (Message) internalProvision.
	getValue();

	// gönderilen mesaj içeriği
	ActionContent actionContent = (ActionContent)
	message.getContent();

	// servis sağlayıcı etmeni tanımlayan nesne
	DFAgentDescription agentDescription =
	(DFAgentDescription) actionContent.getArgument(0);

	// servis sağlayıcı etmenin servisleri alnıyor
	List<ServiceDescription> services =
	agentDescription.getServices();

	Iterator<ServiceDescription> iterator =
	services.iterator();

	// servis sağlayıcı etmenin her servisi
```

```
// dolaşılıyor
      while (iterator.hasNext()) {

            ServiceDescription serviceDescription =
            iterator.next();

            // her servisin owl-s dosyası //okunuyor
            ve basitleştirilmiş servis //profilleri
            elde ediliyor
            String fileName=serviceDescription.
            getPropertyName();

            JenaModelLoader jenaModelLoader = new
            JenaModelLoader();

            KnowledgeManager knowledgeManager
            =jenaModelLoader.load ("file://" +
            file.getAbsolutePath(), null);

            serviceProfile.setInputHashTable(
                 createServiceParameter(knowledgeMana
                 ger,"http://www.daml.org/services/ow
                 l-/1.1/Profile.owl# asInput"));

            serviceProfile.setOutputHashtable(
                  createServiceParameter(knowledgeMa
                  nager,"http://www.daml.org/service
                  s/owl-s/1.1/Profile.owl#
                  asOutput"));
            }

            SimplifiedServiceProfileEncoder
                  serviceProfileEncoder = new
                  SimplifiedServiceProfileEncoder();

            String fileStr=serviceProfileEncoder.
            readFile("available_service_definitions.x
            ml);

            // basitleştirilmiş servis profilleri
            //dosyası güncellenir
            if
            (fileStr.indexOf(serviceProfile.getServiceName()
            ) < 0)
            {
                  int index =fileStr.indexOf
                  ("<goals>");
```

```
                        String substring1=fileStr.
                        substring(0, index + 7);

                        String subString2=fileStr.
                        substring(index + 7);

                        String updatedFileStr= substring1
                        "\n"+ serviceProfileEncoder.
                        EncodeServiceProfile
                        (serviceProfile) + "\n"+
                        subString2;


                        File file = new File ("./files/
                              available_service_definitions.x
                              ml");
                        file.createNewFile();

                        FileWriter fileWriter = new
                        FileWriter(file);

                        fileWriter.append (updatedFileStr);

                        fileWriter.flush();
                        fileWriter.close();
                  }
            }
      }

      @Override
      protected void makeDefinitions() throws TaskException
{

      defineName(ACUpdateServiceProfiles. class.getName());

      defineProvision(new InternalProvision
      (P_DF_AGENT_DESC));

      }

      /**
      * her servisin owl-s dosyası okunur ve girdi,
      * çıktı parametreleri belirlenir.
      */
      public Hashtable
      createServiceParameter(KnowledgeManager
      knowledgeManager, String uri)
      {
```

```
Hashtable parameterHashtable = new Hashtable();

TripleList triples =
knowledgeManager.listTriples(null,

knowledgeManager.createResource (uri), null);

String serviceName = "";
String parameterName = "";
ServiceParameter serviceParameter = new
ServiceParameter();
while (triples.hasNext()) {
      Triple triple = triples.getNext();

      parameterName=triple.getObject().
      toString();

      serviceName=parameterName;
      int index=serviceName. lastIndexOf(".");

      int lastIndex=serviceName.
      lastIndexOf("/");

      serviceName=serviceName.substring
      (lastIndex + 1, index);

      serviceProfile.setGoalName (serviceName);

      serviceProfile.setServiceName
      (serviceName);

      serviceParameter.setParamaterValue ("");
            serviceParameter.setParameterName
    (parameterName.substring(parameterName.ind
    exOf("#")+1));

      TripleList triplesInput= knowledgeManager.
      listTriples(knowledgeManager.createResource(p
      arameterName),knowledgeManager.createResource
      ("http://www.daml.org/services/owls/1.1/Proce
      ss.owl#parameterType"), null);

      while (triplesInput.hasNext())
      {
            Triple tripleInput =
            triplesInput.getNext();
```

```
                        String inputType=tripleInput.
                        getObject().toString();

                        int indexInput=inputType.
                        indexOf("#");

                        int lastIndexInput=inputType.
                        indexOf("^");

                        serviceParameter.setParameterType(
                              inputType.substring(indexInput + 1,
                              lastIndexInput));
                  }

                  parameterHashtable.put(serviceParameter.getParame
                  terName(),serviceParameter);
            }
            return parameterHashtable;
      }
}
```

Ek 4 İngilizce Terimler Sözlüğü

anlaşma	: agreement
arabulucu	: mediator
aracı	: broker
aracılık	: brokerage
arayüz	: interface
bağlam duyarlılık	: context-awareness
başlatıcı	: initiator
bilgilendirme	: inform
bütünleşme	: integration
çalışma akışı	: work-flow
çalışma zamanı	: runtime
çerçeve	: framework
çıktı	: output
dönüşüm	: conversion
eşleme	: matching
etkileşim	: interaction
etmen	: agent
girdi	: input
hedef	: goal
içerik	: content
istek	: request
istemci	: requester
iş parçacığı	: thread
işletici	: executor
kabul	: agree

keşif	: discovery
kukla uçbirim	: dummy terminal
özerk	: autonomous
profil	: profile
red	: refuse
sağlayıcı	: provider
serileştirilmiş	: serializable
servis	: service
süreç	: process
şablon	: template
uzlaşma	: engagement
vekalet	: proxy
yürütme	: enactment
zamanlayıcı	: schedular

ÖZGEÇMİŞ

Kişisel Bilgiler

Adı Soyadı **:** Gizem Okkalıoğlu

Uyruğu **:** T.C

Doğum Yeri **:** Denizli

Doğum Tarihi **:** 10/03/1984

Medeni Durum **:** Bekar

Eğitim Durumu

- 2006 – 2008 Yüsek Lisans, EÜ Bilgisayar Mühendisliği

  Not Ortalaması : 94/100

  Yüksek lisans eğitimi süresince TUBİTAK BİDEB tarafından verilen Lisansüstü burs programından yararlanılmıştır.

- 2002 – 2006 Lisans, EÜ Bilgisayar Mühendisliği

  Diploma Notu : 79/100

- 1999 – 2002 Menemen Anadolu Lisesi

  Diploma Notu : 5.00/5.00

İş Deneyimi

- 08/2005 – 09/2005 Aselsan – Ankara (Stajyer)
- 07/2005 – 08/2005 Schneider Electric – Izmir (Stajyer)
- 07/2004 – 08/2004 Petkim A.S. –Izmir (Stajyer)