EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(DOKTORA TEZİ)

ANLAMSAL WEB ORTAMINDA ÇALIŞAN ÇOK ETMENLİ SİSTEMLERİN MODEL GÜDÜMLÜ GELİŞTİRİLMESİ

Geylani KARDAŞ

Uluslararası Bilgisayar Anabilim Dalı
Bilim Dalı Kodu : 619.02.04
Sunuş Tarihi : 03.07.2008

Tez Danışmanları : Prof. Dr. E. Turhan Tunalı

Prof. Dr. Oğuz Dikenelli

BORNOVA - İZMİR

Geylani KARDAŞ tarafından **DOKTORA TEZİ** olarak sunulan **"Anlamsal Web Ortamında Çalışan Çok Etmenli Sistemlerin Model Güdümlü Geliştirilmesi"** 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 **03.07.2008** tarihinde yapılan tez savunma sınavında aday oybirliği ile başarılı bulunmuştur.

Jüri Üyeleri:		**İmza:**
Jüri Başkanı	: Prof. Dr. E. Turhan TUNALI	.....
Raportör Üye	: Prof. Dr. Oğuz DİKENELLİ	.....
Üye	: Prof. Dr. Bahar KARAOĞLAN	.....
Üye	: Yrd. Doç. Dr. Adil ALPKOÇAK	.....
Üye	: Yrd. Doç. Dr. Pınar YOLUM	.....

ÖZET

ANLAMSAL WEB ORTAMINDA ÇALIŞAN ÇOK ETMENLİ SİSTEMLERİN MODEL GÜDÜMLÜ GELİŞTİRİLMESİ

KARDAŞ, Geylani

Doktora Tezi, Uluslararası Bilgisayar Enstitüsü

Tez Yöneticileri: Prof. Dr. E. Turhan TUNALI

Prof. Dr. Oğuz DİKENELLİ

Temmuz 2008, 205 sayfa

Yazılım etmenleri ve bunların oluşturduğu Çok-etmenli Sistemler, karmaşık yapıdaki dağıtık sistemlerin modellenmesini ve oluşturulmasını sağlayan etkili birer teknoloji olarak ortaya çıkmışlardır. Öte yandan Anlamsal Web, web sayfası içeriklerinin ontolojiler aracılığıyla yorumlanabilmesini ve bu sayede Dünya Geneli Ağ'ın anlamsal seviyeye taşınmasını hedeflemektedir. Söz konusu bu yorumlamanın ve anlam çıkarsamaların özerk etmenler tarafından insanlar adına yerine getirileceği düşünülmektedir. Anlamsal Web ortamının kendine özgü mimari varlıklarının ve farklı bir yapısının olduğu, bu ortam üzerinde çalışacak etmen sistemleri hazırlanırken göz önünde bulundurulmalıdır. Bu tezde, çok-etmenli sistem geliştirme sırasında Anlamsal Web yapılarını ve bunlarla geleneksel etmen sistemi bileşenleri arasındaki etkileşimleri destekleyen yeni bir etmen yazılımı geliştirme süreci tanıtılmaktadır. Önerilen süreç yazılım geliştirmenin odağını koddan modellere çevirmeyi hedefleyen Model Güdümlü Geliştirme yaklaşımına dayanmaktadır.

Model güdümlü yazılım geliştirme, çalışma alanına özgü üstmodellerin tanımlanmasına, bu üstmodellere uyan sistem modellerinin oluşturulmasına, modellerin içerdiği varlıklar arasındaki eşlemelere dayalı olarak modeller arasında dönüşümlerin tanımlanmasına ve uygulanmasına ve son olarak çıktı modellerinden sistem yazılım kodlarının otomatik olarak elde edilmesini sağlayan modelden metne dönüşümlerin tanımlanmasına ve uygulanmasına ihtiyaç duymaktadır. Tezde ortaya konan çalışma Anlamsal Web'de çalışan çok-etmenli sistemlerin geliştirilmesi için model güdümlü yaklaşımın tüm bu ihtiyaçlarını karşılayan bütünleşik bir yazılım geliştirme sürecini sunmaktadır.

Anahtar sözcükler: Çok-etmenli Sistem, Anlamsal Web, Model Güdümlü Geliştirme

ABSTRACT

MODEL DRIVEN DEVELOPMENT OF SEMANTIC WEB ENABLED MULTIAGENT SYSTEMS

KARDAŞ, Geylani

Ph.D. in International Computer Institute

Supervisors: Prof. Dr. E. Turhan TUNALI

Prof. Dr. Oğuz DİKENELLİ

July 2008, 205 pages

Software agents and Multi-agent Systems are emerging technologies which provide efficient composition and modeling of complex distributed systems. On the other hand, Semantic Web aims to improve World Wide Web such that web page contents are interpreted with ontologies. It is apparent that the interpretation in question will be realized by autonomous agents in order to handle the semantic content on behalf of their human users. Surely, Semantic Web environment has specific architectural entities and a different semantic which must be considered to model a multi-agent system within this environment. Hence, this dissertation study introduces an agent software development process which supports the Semantic Web constructs and their interactions with the traditional agent system components during multi-agent system development. The approach of the proposed methodology is based on the Model Driven Development which aims to change the focus of software development from code to models.

Model driven software development requires definition of domain specific metamodels, definition of system models conforming to those metamodels, definition and application of model transformations between those models according to the entity mappings and definition and application of model to text transformation for automatic generation of software codes from output models. The study in here presents a complete software development process that meets all of the above requirements for the model driven development of the Semantic Web enabled multi-agent systems.

Keywords: Multi-agent System, Semantic Web, Model Driven Development

TEŞEKKÜR

Öncelikle bu tez konusu üzerinde bana çalışma imkanı sunan tez danışmanlarım Prof. Dr. Oğuz DİKENELLİ ve Prof. Dr. E. Turhan TUNALI'ya tez çalışmam süresince deneyimleri, bilgileri ve önerileriyle araştırma ve geliştirmeyi yönlendirmeleri ve sağladıkları kaynaklarla destek olmalarından dolayı teşekkürü bir borç bilirim.

Tez izleme toplantılarında jüri olarak görev alan Prof. Dr. Bahar KARAOĞLAN ve Yrd. Doç. Dr. Adil ALPKOÇAK'a değerli önerileri ve çalışmaya olan katkıları için teşekkürlerimi sunarım.

Çalışmalar sırasında bilgi ve görüşlerinden yararlandığım ve gerektiğinde yardımlarını esirgemeyen değerli arkadaşım Arda GÖKNİL ve hocam Prof. Dr. N. Yasemin TOPALOĞLU'na teşekkür ederim. Ayrıca başta Özgür GÜMÜŞ, Önder GÜRCAN ve Erdem Eser EKİNCİ olmak üzere tüm Ege Üniversitesi Etmen Araştırma Grubu üyelerine ve SEAGENT projesinden takım arkadaşlarıma yardımlarından dolayı teşekkür ederim. Ege Üniversitesi Uluslararası Bilgisayar Enstitüsü'nde görevli değerli hocalarım, araştırma görevlisi arkadaşlarım ve idari personele ve tabi bu uzun çalışma süresi boyunca her zaman yanımda olan aileme de manevi desteklerinden ötürü teşekkür ederim.

"Yurtiçi Doktora Başarı Bursu Programı" ile doktora çalışmamı maddi olarak destekleyen Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (TÜBİTAK) Bilim İnsanı Destekleme Daire Başkanlığı'na (BİDEB) teşekkürlerimi sunarım.

Son olarak çalışmanın sunulduğu çeşitli bilimsel konferanslarda ve dergilerde yorumları ve olumlu eleştirileri ile çalışmanın önemli ölçüde geliştirilmesini sağlayan, burada adını sayamadığım yerli ve yabancı birçok bilim insanına teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

ŞEKİLLER DİZİNİ

ŞEKİLLER DİZİNİ (devam)

ÇİZELGELER DİZİNİ

Çizelge Sayfa

KISALTMALAR

ACSM	: Agent Class Superstructure Metamodel (Etmen Sınıf Üstyapısı Üstmodeli)
ADT	: ATL Development Tools (ATL Geliştirme Araçları)
AML	: Agent Modeling Language (Etmen Modelleme Dili)
AOSE	: Agent Oriented Software Engineering (Etmen Yönelimli Yazılım Mühendisliği)
AST	: Abstract Syntax Tree (Soyut Sözdizim Ağacı)
ATL	: Atlas Transformation Language (Atlas Dönüşüm Dili)
AUML	: Agent UML (Etmen Birleşmiş Modelleme Dili)
BDI	: Belief-Desire-Intention (Kanı-İstek-Niyet)
BPEL	: Business Process Execution Language (İş Süreci Yürütme Dili)
CIM	: Computation Independent Model (Programlama Bağımsız Model)
CIMM	: Computation Independent Metamodel (Programlama Bağımsız Üstmodel)
DL	: Description Logic (Tanımlama Mantığı)
EBNF	: Extended Backus-Naur Form (Genişletilmiş Backus-Naur Biçimi)
EMF	: Eclipse Modeling Framework (Eclipse Modelleme Çerçevesi)
FIPA	: Foundation for Intelligent Physical Agents (Akıllı Fiziksel Etmenler Kuruluşu
GEF	: Graphical Editing Framework (Grafiksel Düzenleme Çatısı)
HTML	: HyperText Markup Language (HiperMetin İşaretleme Dili)

KISALTMALAR (devam)

HTN	: Hierarchical Task Network (Hiyerarşik Görev Ağı) HTTP – Internet Inter-ORB Protocol
HTTP-IIOP	: HyperText Transfer Protocol-Internet Inter-Object Request Broker Protokol (HiperMetin Transfer Protokolü – Internet İç-Nesne İstek Aracı Protokolü)
IEEE	: Institute of Electrical and Electronics Engineers (Elektrik ve Elektronik Mühendisleri Enstitüsü)
M2M	: Model-to-Model (Modelden modele)
M2T	: Model-to-Text (Modelden metne)
MAS	: Multi-agent System (Çok-etmenli Sistem)
MDA	: Model Driven Architecture (Model Güdümlü Mimari)
MDD	: Model Driven Development (Model Güdümlü Geliştirme)
MOF	: Meta-Object Facility (Üst-Nesne Binası)
MVC	: Model-View-Controller (Model-Görünüm-Denetleyici)
OCL	: Object Constraint Language (Nesne Kısıt Dili)
OMG	: Object Management Group (Nesne Yönetim Grubu)
OWL	: Web Ontology Language (Web Ontoloji Dili)
OWL-S	: OWL for Services (Servisler için Web Ontoloji Dili)
PIM	: Platform Independent Model (Platform Bağımsız Model)
PIMM	: Platform Independent Metamodel (Platform Bağımsız Üstmodel)
PSM	: Platform Specific Model (Platforma Özgü Model)
PSMM	: Platform Specific Metamodel (Platforma Özgü Üstmodel)
RDF	: Resource Description Framework (Kaynak Tanımlama Çerçevesi)
RDFS	: Resource Description Framework Schema (Kaynak Tanımlama Çerçevesi için Şema)

KISALTMALAR (devam)

SOA	: Service-Oriented Architecture (Servis Yönelimli Mimari)
SOAP	: Simple Object Access Protocol (Basit Nesne Erişim Protokolü)
UML	: Unified Modeling Language (Birleşmiş Modelleme Dili)
URI	: Uniform Resource Identifier (Tekbiçimli Kaynak Tanımlayıcısı)
W3C	: World Wide Web Consortium (Dünya Geneli Ağ Birliği)
WSDL	: Web Services Description Language (Web Servisleri Tanımlama Dili)
WSMO	: Web Service Modeling Ontology (Web Servis Modelleme Ontolojisi)
WWW	: World Wide Web (Dünya Geneli Ağ)
XMI	: XML Metadata Interchange (Genişletilebilir Biçimleme Dili Üstveri Değişimi)
XML	: eXtensible Markup Language (Genişletilebilir Biçimleme Dili)

1 GİRİŞ

Yazılım etmenleri (*"software agents"*) ve bunların oluşturduğu Çok-etmenli Sistemler (*"Multi-agent Systems"*), karmaşık yapıdaki dağıtık sistemlerin modellenmesini ve oluşturulmasını sağlayan etkili birer teknoloji olarak ortaya çıkmışlardır. En temel yapay zeka tanımıyla bir etmen, algılayıcıları (*"sensor"*) yardımıyla içinde bulunduğu ortamı algılayan ve etkileyicileri (*"effector"*) yardımıyla da bu ortam üzerinde eylemlerde bulunan bir sistemdir (Russell and Norvig, 2003). Kullanıcılarının adına bir takım görevleri yerine getirmek üzere davranma yeteneği olan bu özerk (*"autonomous"*) etmenler kendi bilgi ve bireysel yeteneklerini kullanarak çözemedikleri veya etkin bir biçimde çözemeyeceklerini düşündükleri problemlerini çözmek amacıyla bir araya gelerek MAS (*"Multi-agent System"*) adı verilen sistemleri oluşturmaktadırlar. MAS'lar bünyesinde etmenlerin birbirleri ile olan etkileşimleri Sycara'nın (Sycara, 1998)'de belirttiği gibi bencil veya işbirlikçi bir yapıda olabilir. Başka bir deyişle etmenler ortak bir amacı paylaşabilir ya da serbest piyasa ekonomisinde olduğu gibi kendi çıkarlarının takipçisi olabilirler.

MAS'lar üzerinde araştırmalarda bulunan bilim adamları MAS'ların geliştirilmesine katkıda bulunacak çeşitli iletişim dilleri, etkileşim protokolleri ve etmen mimarileri geliştirmektedirler. Etmen yönelimli yazılım geliştirmeye has karakteristikler ve zorluklar güncel yazılım mühendisliği metodolojilerinin sınırlarını zorladığından dolayı araştırmacılar aynı zamanda etmen yönelimli yazılım geliştirme için yeni yazılım metodolojileri ve araçları da önermektedirler (Bergenti et al., 2004).

Bir yazılım geliştirme metodolojisi genelde bir sistem modelinden ve geliştirmeye ait bir yazılım sürecinden ibarettir (Bauer and Odell,

2005). Etmen yönelimli yazılım mühendisliği bakış açısı göz önüne alındığında MAS geliştirme çalışmalarının da bunu dikkate aldığı görülmektedir. Ancak etmen yönelimli yazlım geliştirme özellikle *etmen, amaç*, *rol, içerik* ve *mesajlaşma* gibi yapıları bir sistemin birinci sınıf bileşenleri olarak ele almasından dolayı nesne tabanlı geliştirme tekniklerine göre farklılıklar içermektedir. Bu nedenle örneğin Gaia (Zambonelli et al., 2003), Tropos (Bresciani et al., 2004), MaSE (DeLoach et al., 2001) ve SODA (Omicini, 2000) gibi literatürdeki belli başlı etmen geliştirme metodolojileri MAS geliştiricileri için yukarıda sözü edilen birinci sınıf bileşenleri dikkate alan çeşitli sistem geliştirme süreçlerini önermektedirler.

MAS yazılım modelleri sistem elemanlarının özel bir sözdizim ve anlamsalla (*"semantic"*) ifade edilmesini sağlarken yazılım süreci yazılım geliştirme aktivitelerini ve birbirleri ile olan ilişkilerini tanımlamaktadır. Doğal olarak her sistem geliştirme metodolojisi, içerdiği sürecin ürünlerine ve mimari yapılarına dayanan bir yazılım üstmodelini tanımlamaktadır. Ancak bir üstmodel özel bir metodolojiden bağımsız olabilir ve özel bir uygulama alanına ait mimari bileşenleri ve bunların birbirleri ile olan ilişkilerini tanımlayabilir.

MAS araştırmaları içerisinde yukarıda sözü edilen sistem geliştirme metodolojileri veya süreçleri için MAS üstmodelleri tanımlayan çeşitli çalışmaların (Bernon et al., 2005; Molesini et al., 2005; Odell et al., 2005) yer aldığı görülmektedir. Ancak her ne kadar bu etmen modelleme ve yukarıda sözü edilen MAS geliştirme çalışmalarının ilgili alana önemli katkıları olsa da yakın gelecekte etmenlerin üzerinde çalışacağı düşünülen Anlamsal Web (*"Semantic Web"*) ortamı ve MAS'ların bu ortam üzerinde çalışabilmesi için ihtiyaç duyulan yapıların bu çalışmalarda desteklenmediği gözlenmiştir.

Anlamsal Web (Berners-Lee et al., 2001) evrimi şüphesiz etmen araştırmalarına yeni bir vizyon getirmiştir. Bu ikinci kuşak Web, Dünya Geneli Ağ'ı web sayfası içeriklerinin ontolojiler kullanılarak yorumlanabileceği bir seviyeye taşımayı hedeflemektedir. Söz konusu yorumlamanın ve anlam çıkarsamaların özerk etmenler tarafından insanlar adına yerine getirileceği düşünülmektedir.

Anlamsal Web ortamının kendine özgü mimari varlıklarının ve farklı bir yapısının olduğu, bu ortam üzerinde çalışacak MAS'lar hazırlanırken göz önünde bulundurulmalıdır. Etmen mimarilerinin, modelleme tekniklerinin ve MAS yazılımı geliştirme çerçevelerinin bu yeni ortamı desteklemesi gerekmektedir. İşte bu düşünceden hareketle bu tez çalışmasında MAS geliştirme sırasında Anlamsal Web yapılarını ve bunlarla geleneksel MAS bileşenleri arasındaki etkileşimleri destekleyen yeni bir MAS yazılımı geliştirme süreci tanıtılmaktadır. Böyle bir geliştirme sürecine dayalı olarak hayata geçirilen etmen sistemleri Anlamsal Web yetenekli olacaklar ve bu sistemlerde yer alan yazılım etmenleri de kullanıcıları adına Web içeriğini farklı kaynaklardan elde edebilecek, bilgiyi işleyebilecek ve sonuçları değiş tokuş edebileceklerdir. Ayrıca özerk etmenler bu tip MAS'lar içerisinde anlamsal veriyi değerlendirebilecek ve içerik dilleri vasıtasıyla anlamsal web servisleri gibi anlamsal ortam elemanları ile etkileşimlerde bulunabileceklerdir.

Önerilen sürecin dayandığı yaklaşım yazılım geliştirmenin odağını koddan modellere çevirmeyi hedefleyen ve gün geçtikçe popülerleşen Model Güdümlü Geliştirme'ye (Selic, 2003) dayanmaktadır. MAS'ların tasarımı ve gerçekleştirilmesi özellikle Anlamsal Web gibi yeni etmen çalışma ortamlarının ihtiyaçları ve içerdiği etkileşimler göz önüne alındığında daha karışık ve uygulanması daha zor bir hal almaktadır. Bu tez çalışması gerçekleştirilirken model güdümlü yazılım geliştirmenin bu

tip MAS'ların geliştirmesini kolaylaştıracak bir altyapı sunacağına inanılmıştır. İçsel karmaşıklıklarından ve dağıtık ve açık yapılarından ötürü MAS'ların yazılım kodu detayları ile uğraşmak gün geçtikçe daha zor bir hal almaktadır. Bu nedenle MAS geliştirme sırasında mümkün olan en üst soyut seviyelerde çalışmak kritik bir öneme sahiptir. MAS geliştirmede ihtiyaç duyulan bu en üst soyutlama seviyesini sağlamada model güdümlü geliştirmenin güçlü bir alternatif olduğu görülmektedir.

Model güdümlü yazılım geliştirme, çalışma alanına özgü üstmodellerin tanımlanmasına, bu üstmodellere uyan sistem modellerinin oluşturulmasına, modellerin içerdiği varlıklar arasındaki eşlemelere dayalı olarak modeller arasında dönüşümlerin tanımlanmasına ve uygulanmasına ve son olarak çıktı modellerinden sistem yazılım kodlarının otomatik olarak elde edilmesini sağlayan modelden metne dönüşümlerin tanımlanmasına ve uygulanmasına ihtiyaç duymaktadır. Tezde ortaya konan çalışma Anlamsal Web ortamında çalışan MAS'ların geliştirilmesi için model güdümlü geliştirmenin tüm bu ihtiyaçlarını karşılayan bütünleşik bir yazılım geliştirme sürecini sunmaktadır.

Çalışmanın hedef aldığı iş alanı Anlamsal Web ortamında çalışan etmen sistemleri olduğu için bu tip sistemleri tanımlayan biçimsel bir etmen üstmodeline ihtiyaç duyulmaktadır. Bu üstmodel etmen uygulamalarının geliştirildiği yazılım çatılarından bağımsız bir şekilde Anlamsal Web yetenekli MAS'ların birinci sınıf üstvarlıklarını ve bu varlıkların birbirleri ile olan ilişkilerini modellemelidir. Ancak literatürde yer alan etmen üstmodellerinin çoğu sadece üzerinde çalışılan metodolojinin kavramlarını biçimsel bir şekilde tanıtmayı hedeflemektedirler ve genel amaçlı değildirler (Pavon et al., 2006). Ayrıca üstmodel önerilerinin hiçbirinin ne etmen yeteneklerinin anlamsal temsilini ne de yazılım etmenleri ile anlamsal web servislerinin etkileşimini desteklediği görülmektedir. Tezde ilk olarak bu ihtiyaca

cevap veren ve etmen organizasyonları, etmenler, etmen rolleri ve diğer anlamsal web uzantılarının birbirleri ile olan ilişkileri de dahil olmak üzere modellendiği platform bağımsız bir üstmodel ortaya konmuştur. Daha sonra bu üstmodel ile çeşitli MAS yazılımı geliştirme çerçevelerine ait üstmodeller arası model dönüşümlerinin tanımlanmasını ve hayata geçirilmesini sağlayan model güdümlü geliştirmeye ait süreç adımları tasarlanmış ve uygulanmıştır. Son olarak da model dönüşümleri sonucu elde edilen çıktı modellerinden otomatik olarak yazılım kodunun üretilmesini sağlayan modelden metne dönüşüm süreci tanımlanmış ve uygulamaya geçirilmiştir.

Çalışmada ortaya konan platform bağımsız üstmodel etmen sistemlerinin modellenmesini hedeflemektedir. Amaç yönelimli ve BDI (*"Belief-Desire-Intention"*) (Rao and Georgeff, 1995) prensibine dayanan etmen sistemleri ve bunların anlamsal web servisleri ile olan etkileşimleri söz konusu üstmodel hazırlanırken göz önünde bulundurulmuştur. Etmen sistemlerinin kullandığı ontolojilerin ve buna bağlı yapıların desteklenmesi amacıyla üstmodele yeni üstvarlıklar dahil edilmiştir. Ancak model dönüşümleri sırasında hedef platformların mutlaka bu yapıları desteklemesine gerek yoktur. Kısmi dönüşümlerin de elde edilmesi mümkündür.

Öte yandan çalışmada tanıtılan model dönüşümlerinde çeşitli etmen yazılımı geliştirme çerçevelerinin platforma özgü üstmodelleri kullanılabilmektedir. Ancak dönüşümlerin mümkün olan en üst seviyede gerçekleşebilmesi için yukarıda sözü edilen etmen özelliklerini ve etkileşimlerini bünyesinde bulunduran platforma özgü üstmodeller gerekmektedir.

1.1 Tezin Katkıları

Anlamsal Web ortamında çalışan çok etmenli sistemlerin model güdümlü olarak geliştirilmesini hedef alan tezin katkıları şu şekilde sıralanabilir:

1. Bu tez çalışması ile MAS'ların anlamsal web servisleri gibi Anlamsal Web ortamı yapıları ile etkileşimlerini dikkate alan ve bu etkileşimlere yönelik platform servisi, etmen planlama ve iletişim seviyelerinde görev alan mimari bileşenlerin tanımlandığı ve Anlamsal Web yetenekli etmen sistemlerinin tasarlanması için referans olabilecek katmanlı bir etmen yazılım mimarisi ortaya konmuştur (Kardas et al., 2006).
2. Birinci sınıf üstvarlıkları yukarıda belirtilen katmanlı mimarinin bileşenleri ve fonksiyonları dikkate alınarak türetilen bir platform bağımsız MAS üstmodeli sunulmuştur. Bu üstmodel etmen, rol, grup gibi geleneksel etmen sistemi varlıkları yanında Anlamsal Web ortamında çalışan etmen sistemlerinin ihtiyaçlarına yönelik yeni üstvarlıkları belirlemiş ve bu varlıklar arası ilişkileri modellemiştir (Kardas et al., 2007a).
3. Anlamsal web yetenekli etmen sistemlerinin model dönüşümleri sonrası elde edilmesine yönelik modelden modele dönüşüm süreci tanımlanmıştır ve uygulamaya geçirilmiştir (Kardas et al., 2007b). Bu sürecin değerlendirilmesi sırasında iki adet farklı MAS geliştirme çerçevesinin dikkate alınması ve her ikisi için de çıktı modellerinin otomatik bir şekilde üretilebiliyor olması tanımlanan modelden modele dönüşümün literatürde var olan benzerlerine göre en büyük avantajı olmuştur. Hedef MAS geliştirme çerçevesinin desteklemesine bağımlı olarak Anlamsal Web yapılarının ve etmenlerle bu yapıların etkileşiminin de model

dönüşümü sonrası elde edilebiliyor olması tanımlanan dönüşümlerin sağladığı başka bir faydadır.

4. Yine tez çalışması kapsamında hedef modellerden yazılım kodlarının da otomatik olarak elde edilmesini sağlayan modelden metne dönüşüm süreci tanımlanmıştır ve yine iki ayrı MAS çerçevesi için uygulamaya geçirilmiştir. Üretilen sistem modellerinden şablon seviyesinde yazılım kodlarının elde edilebildiği model güdümlü etmen geliştirme önerilerinin literatürde çok az olması bakımından tezin bu anlamda da ilgili alana katkısının olduğu düşünülmektedir. Buna ek olarak modelden koda dönüşümün hem grafiksel bir ortam kullanılarak hem de direkt model – metin eşlemeleri yerine getirilerek tanımlanıp denenmiş olması farklı açılardan bu modelden kod olan dönüşüm safhasının değerlendirilmesini sağlamıştır.

1.2 Tezden Çıkan Yayınlar

Tez sürecinde ortaya konan çalışma ürünlerinin tanıtıldığı ve savunulan fikirlerin, elde edilen bulguların ve deneyimlerin aktarıldığı bilimsel yayınlar aşağıda sunulmuştur. Tezin ilerleyen bölümlerinde listelenen bu yayınların her birine değinilmektedir:

1. Kardas, G., Goknil, A., Dikenelli, O. and Topaloglu, N. Y. (2007) *"Model Transformation for Model Driven Development of Semantic Web Enabled Multi-Agent Systems"*, In Petta, P., Müller, J. P., Klusch, M. and Georgeff, M. (Eds.): Multiagent System Technologies, Lecture Notes in Artificial Intelligence, Springer-Verlag, Vol. 4687, pp. 13-24.

2. Kardas, G., Goknil, A., Dikenelli, O. and Topaloglu, N. Y. (2007) *"Modeling the Interaction between Semantic Agents and Semantic*

Web Services using MDA Approach", In O'Hare, G. M. P., Ricci, A., O'Grady, M. J. and Dikenelli, O. (Eds.): Engineering Societies in the Agents World VII, Lecture Notes in Artificial Intelligence, Springer-Verlag, Vol. 4457, pp. 209-228.

3. Kardas, G., Goknil, A., Dikenelli, O. and Topaloglu, N. Y. (2006) *"Metamodeling of Semantic Web Enabled Multiagent Systems"*, In proceedings of the Multiagent Systems and Software Architecture (MASSA), Special Track at Net.ObjectDays 2006, Erfurt, Germany, pp. 79-86.

4. Kardaş, G. ve Dikenelli, O. (2006) *"Çok-Etmenli Yazılım Sistemleri için Yürütülen Modelleme Dili Çalışmaları ve Bunların Anlamsal Web Desteği Perspektifinde Değerlendirilmesi"*, Fifteenth Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN 2006), Akyaka, Muğla, Türkiye, ss. 199-206.

5. Kardas, G., Gümüs, Ö. and Dikenelli, O. (2005) *"Applying Semantic Capability Matching into Directory Service Structures of Multi Agent Systems"*, In Yolum, P., Güngör, T., Gürgen, F. and Özturan, C. (Eds.): Computer and Information Sciences - ISCIS 2005, Lecture Notes in Computer Science, Springer-Verlag, Vol. 3733, pp. 452-461.

6. Kardaş, G., Gümüş, Ö. ve Dikenelli, O. (2005) *"Anlamsal Web Servislerinin Bulunması, Eşlenmesi ve Dinamik Çağrımı Üzerine Bir Durum Çalışması"*, İkinci Ulusal Yazılım Mühendisliği Sempozyumu ve Sergisi (UYMS 2005), Ankara, Türkiye, ss. 41-49.

1.3 Tezin Çerçevesi

Tezin ikinci bölümünde çalışma için gerekli alt yapı ve bu alanda daha önce gerçekleştirilmiş çalışmalar hakkında bilgi verilmektedir. Tez kapsamına uygun olarak alt yapı kısmında yazılım etmenleri, anlamsal web ve buna bağlı anlamsal web servisi teknolojisi ve model güdümlü yazılım geliştirme anlatılmıştır. Literatürde bu alanda gerçekleştirilen çalışmalar aktarılırken ilk olarak etmen sistemlerinin modellenmesine yönelik çalışmalar üzerinde durulmuş bunu model güdümlü olarak etmen sistemlerinin geliştirilmesini öneren çalışmaların anlatılması takip etmiştir. Son olarak literatürde yer alan bu çalışmalara ait genel bir değerlendirme gerçekleştirilmiş ve bu tez kapsamında ortaya konan çalışmanın bu çalışmalara göre farkları anlatılmıştır.

Tezin üçüncü bölümünde Anlamsal Web yetenekli etmenler için önerilen katmanlı yazılım mimarisi tanıtılmıştır. Uygulama modellerini oluşturan ve Anlamsal Web ortamında çalışan MAS'ların bileşenleri olan birinci sınıf varlıkların bu tip sistemlerin model güdümlü yaklaşımlarla geliştirilebilmeleri amacıyla tanımlanmış olmaları gerekmektedir. Söz konusu bu bileşenlerin ve görevlerinin tanımlandığı mimari platform bağımsız etmen üstmodellerinin oluşturulmasının önünü açmaktadır.

Anlamsal Web yetenekli etmen sistemlerinin model güdümlü olarak geliştirilmesi için platform bağımsız bir üstmodele ihtiyaç vardır. Tezde, elemanları ve bunlar arasındaki ilişkileri yukarıda sözü edilen etmen yazılım mimarisine bağlı olarak tanımlanmış bir MAS üstmodeli geliştirilmiştir. Bu üstmodel tezin dördüncü bölümünde anlatılmıştır. Tezde önerilen model güdümlü geliştirme sürecinde platform bağımsız bir üstmodel olarak kullanılan modelin çekirdek yapısı ve uzantılarla şekillenmiş son versiyonu anlatılmış bu üstmodele uyan örnek bir MAS modeli de yine aynı bölüm içerisinde anlatılmıştır. Örnek MAS modeli

önerilen model güdümlü sürecin uygulanışını örneklemek amacıyla takip eden tez bölümlerinde de kullanılmaktadır.

Bir model güdümlü yazılım geliştirme yönteminin ana süreci modeller arasındaki dönüşümlerin sağlanmasıdır. Üstmodeller arasında tanımlanan model dönüşümleri sonrasında yazılım modellerinin otomatik olarak elde edilmesi sağlanmaktadır. Tezin önerdiği model güdümlü MAS geliştirme yöntemi de söz konusu bu model dönüşümünü platform bağımlı ve platforma özgü etmen üstmodelleri arasında sağlamaktadır. Tezin 5. bölümünde Anlamsal Web yetenekli MAS'ların model güdümlü geliştirilmesi için gerekli olan bu model dönüşüm mekanizması tanıtılmıştır. Bölümde model dönüşümü için üstmodeller arasında varlık eşlemeleri, bu eşlemelere dayalı olarak türetilen model dönüşüm kuralları ve bu kurallara bağlı olarak uygulanan dönüşümler anlatılmıştır.

Her ne kadar modelden modele dönüşüm MAS geliştirmeyi yüksek bir soyutlama seviyesi sağlayarak kolaylaştırıyor olsa da bu tip sistemlerin gerçek uygulamaları için model dönüşümünün yeterli olmadığı açıktır. Sistem tasarımcılarının kaçınılmaz surette yazılım kodlarını hazırlamaları da gerekmektedir. Bu noktadan hareketle tezde ortaya konan geliştirme süreci platforma özgü etmen sistemi modellerinden yazılım kodlarının otomatik olarak üretildiği bir sistem geliştirme adımını da içermektedir. Sürece ait bu adım tezin 6. bölümünde anlatılmaktadır.

Tezin 7. bölümünde önerilen model güdümlü sürecin değerlendirilmesi gerçekleştirilmiştir. Değerlendirme için seçilen yöntem sürecin başka etmen yazılım geliştirme çerçeveleri için de uygulanabilirliğini göstermeye dayanmaktadır.

Tezin 8. ve son bölümünde ise sonuç ve çalışmayı ileriye götürebilecek öneriler yer almaktadır.

2 ALT YAPI VE İLGİLİ ÇALIŞMALAR

Bu bölümde tez çalışması için gereken alt yapı bilgileri ve daha önce bu konuda gerçekleştirilmiş olan çalışmalar yer almaktadır. Alt yapı alt bölümünde yazılım etmenleri, anlamsal web ve buna bağlı teknolojiler ve model güdümlü yazılım geliştirme hakkında bilgiler yer almaktadır. İkinci alt bölümde ise etmen sistemlerinin modellenmesi ve model güdümlü geliştirilmesi ile ilgili diğer çalışmalar hakkında bilgi verilmiştir.

2.1 Alt Yapı

2.1.1 Yazılım etmenleri

Yazılım etmenleri ve bunların oluşturduğu çok-etmenli sistemler, bilgi ve iletişim teknolojisi iş senaryolarının çok çeşitli alanlarda uygulanması sırasında karşılaşılan zorlukların giderilmesini sağlayan güçlü birer teknoloji olarak ortaya çıkmışlardır. (Russell and Norvig, 2003)'de algılayıcıları (*"sensor"*) yardımıyla ortamı algılayan ve etkileyicileri (*"effector"*) yardımıyla bu ortamı etkileyen bir sistem olarak tanıtılan etmenleri, aynı zamanda özerk (*"autonomous"*) yazılım ortamları da göz önünde bulundurulduğunda, kullanıcısının adına bir takım görevleri yerine getirmek üzere davranma yeteneği olan yazılım bileşenleri olarak kabul etmek yerinde olacaktır. 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 sistemler de MAS ("*Multi-agent System*") adını almaktadır (Weiss, 1999).

Bir etmene ilişkin özellikleri iki grupta incelemek mümkündür: *birincil özellikler* ve *ikincil özellikler* (Erdur, 2001). Bir etmeni klasik donanım veya yazılım sistemlerinden ayıran ve bir etmenin barındırması gereken birincil özellikler şunlardır:

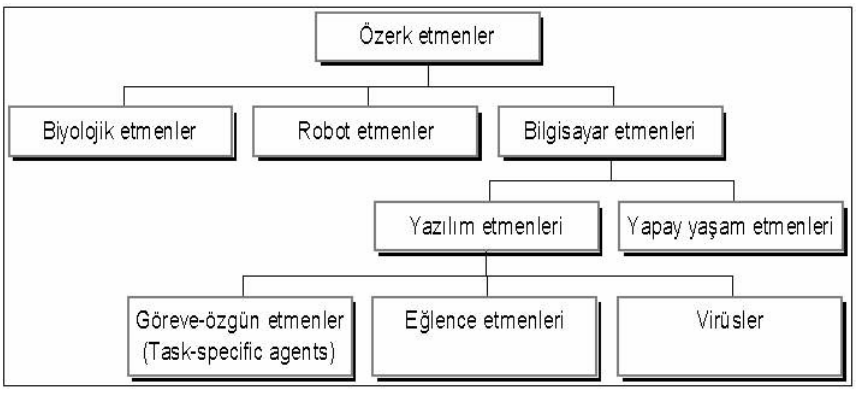
- *Özerklik ("Autonomy")*: Bir etmen insan kullanıcıların, diğer etmenlerin veya sistemlerin doğrudan katılımı olmadan görev başlatabilme ve çalışabilme yeteneğine sahip olmalıdır.
- *Karşıt-eylemlilik ("Reactivity")*: Bir etmen sürekli olarak bulunduğu ortamı algılamalı, ortamdaki değişimlere göre bilgisini, amaçlarını, eylemlerini değiştirebilmelidir.
- *Amaç-yönelimlilik ("Goal-orientedness")*: Bir etmen kendisinden beklenenleri yerine getirmek için planlama yaparak bu planların gerektirdiği eylemleri gerçekleştirmelidir.
- *Sosyal yetenek ("Social ability")*: Bir etmen, planlarına ilişkin görevlerini tamamlayabilmek için diğer etmenler veya insan kullanıcıları ile etkileşimde bulunur.
- *Kalıcı süreklilik ("Permanent continuity")*: Bir etmen, başlattığı görevleri tamamladıktan sonra da etkin olarak kalmalı ve çevresini algılayıp gereken eylemleri gerçekleştirmek için hazır olarak beklemelidir.

Bunların dışında, bir sistemin etmen olma kavramını güçlendiren ve genellikle geliştirilen uygulamaya bağlı olan ikincil özellikler ise şunlardır:

- *Gezicilik ("Mobility")*: Bir etmen örneğin, bir işlemi daha güçlü işlemcili bir bilgisayarda yapıp geri dönmek veya ağ üzerinde bilgi kaynaklarını gezerek bilgi toplamak gibi istekleri yerine getirmek amacıyla ağ üzerinde bir yerden başka bir yere gidebilir.

- *Öğrenme ("Learning")*: Bir etmen kendisini ortama uyarlayabilir, kullanıcı eğilimlerini öğrenerek eylemlerini bu doğrultuda değiştirebilir.
- *Akılcılık ("Rationality")*: Bir etmen kendi amaçları doğrultusunda planlama yapar. Kasıtlı olarak amaçlarına ulaşmayı engelleyen planlar yapmaz.
- *Dürüstlük ("Veracity")*: Bir etmen kasıtlı olarak yanlış bilgi iletişiminde bulunmaz.
- *Olumluluk ("Benevolence")*: Bir etmen, amaçlarına ters düşmedikçe kendisinden beklenenleri yerine getirmek için çalışır.

Literatürde farklı uygulama alanlarına yönelik farklı özellikleri temel alan etmen sınıflandırma önerileri yer almaktadır (Nwana, 1996; Caglayan and Harrison, 1997; Franklin and Graesser, 1997). Ancak bunlardan (Franklin and Graesser, 1997)'de önerilen *Franklin ve Graesser Sınıflandırması*'ndaki sıradüzensel yapının, etmenlerin içsel kontrol yapıları, içsel mimarileri, ikincil özellikleri gibi boyutlar ve bu boyutlara ilişkin özellikler ile desteklendiğinde varolan birçok etmeni sınıflandırabilecek bir yaklaşım olduğu düşünülmektedir (Şekil 2.1).



Şekil 2.1. Franklin ve Graesser etmen sınıflandırması (Franklin and Graesser, 1997)

Franklin ve Graesser Sınıflandırması'nda görülen sıradüzensel yapının altında yer alan göreve-özgün etmenler bu tez kapsamında üzerinde çalışılan etmenlerdir. Bu tip etmenler istenirse, içsel kontrol yapıları (planlama tabanlı, kural tabanlı, bulanık mantık tabanlı, vb.), bulundukları ortam (veritabanı, yerel ağ, İnternet, vb.), gerçekleştirildikleri dil ve uygulama alanlarına göre de alt sınıflara ayrılabilirler.

2.1.1.1 Etmen – Nesne karşılaştırması

Günümüzde, her ne kadar etmenler çoğunlukla nesne tabanlı yazılım geliştirme metodolojileri ve ilgili nesne tabanlı programlama dilleri kullanılarak geliştirilse de etmen ve nesne kavramları arasında farklar bulunmaktadır.

Bir nesne bir etmenin aksine prensipte ne özerktir ne de proaktiftir (*"proactive"*) çünkü kendi içsel aktiviteleri bir başka dış kontrolcüden gelen servis istekleri ile şekillenmektedir. Nesne tabanlı sistemlerde, bir eylemin yapılıp yapılmayacağı (bir yordamın çağrılıp, çağrılmayacağı) çağrımı yapan nesne tarafından kararlaştırılmaktadır. Örneğin, A nesnesi, B nesnesindeki metod1 isimli yordamı çağırmak için B.metod1() biçiminde bir kod işletmelidir. Oysa, etmenlerde bu tür bir çağrım yoktur. Bir A etmeni, B etmeninden metod1 isimli yordamın çağrılmasına karşılık gelen bir eylemde bulunmasını istediğinde bu isteğini bildiren bir *etmen iletişim dili* iletisini B etmenine göndermektedir. Bu gönderim, eylemin mutlaka yapılacağı anlamına gelmemektedir. B etmeni, kendi bilgi tabanında yer alan bilgiler çerçevesinde eylemi yapıp yapmayacağına karar vermektedir.

Öte yandan (Zambonelli and Omicini, 2004)'de belirtildiği gibi kullanımda olan nesne sistemlerinde dış ortamın açık bir modellemesi yoktur. Her şey nesne olarak modellenmiştir ve objeler içinde

bulundukları dünyayı diğer nesnelerin referansları açısından algılayabilirler. Buna ek olarak nesne kaynaklı global bir sistem mimarisinde etmen ortamları için oldukça önemli olan topluluk veya rol gibi kavramlar bir anlam ifade etmezler.

2.1.1.2 FIPA MAS platformu

FIPA[1] (*"Foundation for Intelligent Physical Agents"*) etmenler ve etmen tabanlı sistemler arasında birlikte çalışabilirliği desteklemek amacıyla 1996 yılında kurulmuş kar amacı gütmeyen bir kuruluştur. Etmen tabanlı bilgi işlem alanında standartları belirleyen kurumların başında gelen FIPA, etmen iletişim dilleri, içerik dilleri, etkileşim protokolleri, etmen yönetimi ve etmen mesajlaşma altyapıları konularında standartlar geliştirmektedir. Bu standartlara dayalı olarak tanımlanmış olan FIPA MAS platformu literatürde üzerinde en çok çalışılan ve desteklenen MAS platformudur. Tez kapsamında üzerinde çalışılan MAS geliştirme yazılım çerçevelerinden biri de yine FIPA MAS platformunu desteklediğinden bu platformdan burada kısaca bahsetmek yerinde olacaktır.

FIPA uyumlu bir etmen platformu; ontoloji etmeni, dizin servisi, etmen yönetim sistemi, ileti aktarım servisi ve uygulama alanı bağımlı etmenlerden oluşmaktadır.

Dizin Servisi, platformdaki etmenlerin yeteneklerinin tanıtıldığı ve saklandığı zorunlu bir bileşendir. Dizin servisi genellikle bir etmen olarak tasarlanmakta olup etmen servislerini kaydetme, kayıttan düşme,

---

[1] FIPA, IEEE (*"Institute of Electrical and Electronics Engineers"*) tarafından Haziran 2005'de IEEE'nin 11. standartlaşma komitesi olarak kabul edilmiştir ve bu tarihten sonra IEEE FIPA olarak da adlandırılabilmektedir. Ancak tezde bu organizasyonun daha önceki çalışmalarına değinildiği için isimlendirme olarak yine FIPA tercih edilmiştir.

servislere ait bilgileri değiştirme ve servis arama fonksiyonlarını içermektedir.

Etmen Yönetim Sistemi etmen platformunun işleyişinden sorumlu olan zorunlu bir bileşendir. Etmenlerin ilk oluşturulduğunda isim ve adreslerinin kaydedilmesi, platformdan ayrılan etmenlerin izlenmesi ve platformdaki etmenlerin yaşam döngüsüne ilişkin bilgilerin tutulması gibi görevler bu bileşene aittir.

İleti Aktarım Servisi, platformlar arası iletişim ve platform içindeki etmenlerin iletişiminden sorumludur. Platform içindeki etmenler herhangi bir aktarım protokolü ile iletişimde bulunabilmektedir. Bu servis de zorunlu bileşendir.

Ontoloji Etmeni, FIPA ontoloji servisini gerçekleştiren bir etmendir. MAS'ta yer alan etmenlerin işbirliği yapabilmeleri için birbiriyle iletişimde bulunabilmesi gerekmektedir. Ancak sadece bir etmen iletişim dilinin tasarlanması ve bu dile dayanan protokol kümesinin tanımlanması etmenlerin iletişimi için yeterli değildir. İletilerin içeriklerinde bulunan kelimelerin hangi anlamlarda kullanıldıklarının da bilinmesi gerekmektedir. Aksi durumda, iletilerin hem gönderici hem de alıcı taraflarda aynı biçimde anlaşılması olası olmayacaktır. İletinin hem gönderici hem de alıcı taraflarda aynı biçimde anlaşılması ontolojiler (*"ontology"*) ile sağlanmaktadır. Sisteme ait bu ontolojileri platform etmenlerine sağlayan Ontoloji Etmeni'dir.

FIPA uyumlu bir MAS'ta yer alan diğer etmenler ise uygulama alanına bağlı etmenlerdir (arayüz etmenleri, bilgi etmenleri, vb.). FIPA MAS platformu, soyut etmen mimarisi ve diğer FIPA standartları hakkında ayrıntılı bilgi (FIPA, 2002)'de bulunabilir.

2.1.2 Anlamsal web

Anlamsal Web (Berners-Lee et al., 2001), web sayfalarının anlam ifade eden içeriğine bir yapı getirmeyi; özerk yapıların –ki bu yapılar için en güçlü aday yazılım etmenleridir- sayfa sayfa dolaşarak, temsil ettikleri kullanıcılarının yerine sofistike işlemler gerçekleştirebildikleri bir ortam oluşturmayı hedeflemektedir.

Anlamsal Web şu an kullanımda olan Web'den tamamen farklı değil de onun bir uzantısı olan, bilginin düzgün tanımlanmış bir anlama sahip olduğu ve insanlar ile bilgisayarların beraber çalışabildikleri bir web olarak düşünülmektedir (Berners-Lee et al., 2001). Şu anki Web, veri ve bilgilerin otomatik olarak işlenebildiği bir ortamdan çok insanlar için doküman sağlayan bir medya olacak şekilde geliştirilmiştir. Anlamsal Web ise ilgili otomatik işlemeyi gerçekleştirme amacına sahiptir. Böylelikle Web hem insanlar tarafından *okunabilecek* hem de makineler tarafından *anlaşılabilecektir*.

Anlamsal Web'in düzgün çalışabilmesi için bilgisayarların yapılandırılmış bilgi koleksiyonlarına ve otomatik akıl yürütmeyi (*"automated reasoning"*) sağlayacak çıkarsama (*"inference"*) kuralları kümelerine erişmeleri gerekmektedir.

Yukarıda söz edilen ihtiyaçları karşılamak amacıyla Anlamsal Web'in, hem veriyi hem de veri hakkında akıl yürütmeyi sağlayan kuralların ifade edildiği bir dili sağlaması gerekmektedir. Bu açıdan bakıldığında XML (*"eXtensible Markup Language"*) ve XML tabanlı RDF (*"Resource Description Framework"*) teknolojilerinin Anlamsal Web için önemi ortaya çıkmaktadır. RDF (W3C, 2004) ile her kaynağın bir URI'ye (*"Uniform Resource Identifier"*) sahip olması ve özne-yüklem-nesne (*"subject-predicate-object"*) üçlüsünün yer alması anlamın ifade edilmesini sağlamaktadır. Bir ifade RDF üçlüsü olarak

kodlandığında özne ve yüklemin URI'ler ile ifade edilen kaynaklar olması gerekirken nesne bir kaynak ya da bir hazır bilgi (*"literal"*) elemanı olabilir. İlgili üçlüler de XML etiketleri (*"tag"*) ile ifade edilmektedirler.

Anlamsal Web'in bir başka temel bileşeni ise bilgi koleksiyonları yani *ontoloji*lerdir. Bir ontoloji, kavramlar arasındaki ilişkileri biçimsel (*"formal"*) olarak içeren bir dokümandır. Özellikle aynı kavramı ifade eden farklı tanımlayıcıların belirlenmesi ve otomatik işlemlerin yürütülebilmesi ontolojiler vasıtası ile gerçekleşmektedir. Ontolojilerin bir anlamda üstveri (*"metadata"*) yönetiminde XML şemaları için *isim uzayları*nın ("*namespace"*) gördüğü işi anlamsal boyutta yerine getirdikleri düşünülebilir.

Anlamsal Web'in sunacağı hizmetlerin gerçekleştirilmesinde şu anki Web'e anlam eklenmesini sağlayacak yeni dillerin tanımlanması şarttır. Anlamsal Web için hazırlanacak dillerin iki özelliği içermesi gerekmektedir (Fensel et al., 2003): Öncelikle bu diller, içeriklerinin otomatik olarak işlenebilmesini sağlayacak biçimsel sözdizimlerine (*"syntax"*) ve biçimsel anlamsallara (*"semantic"*) ihtiyaç duymaktadırlar. İkinci olarak bu diller, etmenlerin ve insanların bilgi paylaşımını sağlayacak standartlaşmış sözlüklere ihtiyaç duymaktadırlar. WWW (*"World Wide Web"*) için önerilen katmanlı dil modelinde bildik HTML (*"HyperText Markup Language"*) ve XML dilleri üzerine Anlamsal Web'i destekleyecek RDF'in ve RDFS'in (*"RDF Schema"*) yer aldığı görülmektedir. Ontolojilerin modelleneceği ve hazırlanacağı ontoloji dillerinin RDF tabanlı olacağı görüşü yer almaktadır. Zaten yakın zamanda da W3C (*"World Wide Web Consortium"*) ontolojilerin hazırlanması için RDF tabanlı OWL'ı (*"Web Ontology Language"*) standart olarak kabul etmiştir. Bu dile dayalı olarak çeşitli iş alanları için bilgi tabanı (*"knowledgebase"*) oluşturacak ontolojilerin geliştirilmesi

çalışmaları yürütülmektedir. Bu noktada RDF, RDFS ve OWL arasındaki farklar hakkında kısaca bilgi vermekte yarar vardır. RDF basit ve genel amaçlı bir üstveri dilidir. RDF'in çok sınırlı sözdizim yapıları vardır ve yukarıda bahsedilen üçlüler harici başka yapılar kabul edilmez. RDFS ise RDF için bir tip sistemi tanımlar. Kaynakların sınıflar, özellikler ve değerler ile tanımlanmasını sağlar. İçerdiği sınıf kavramı nesne tabanlı programlama dillerindeki sınıf kavramı ile benzerlik gösterir. Kalıtım ilişkileri RDFS ile tanımlanabilir. Özellikler için kısıtlarının tanımlanmasına imkan verir. Bu özellikleri ile RDFS varolan ontoloji dillerine yaklaşır. Ancak RDFS'in de bazı kısıtları vardır. Sınıflar arası ayrık (*"disjoint"*), geçişli (*"transitive"*) ve tek olma (*"unique"*) gibi ilişkileri ve başka bir özelliğin tersi olan bir özelliğin tanımlanması gibi özellikleri desteklememektedir.

OWL, RDFS'in yukarıdaki eksikliklerini tamamlayan ve ilişkilerdeki anlamın ortaya konmasını sağlayan bir ontoloji dilidir. OWL, RDFS'e ifade etme gücü sağlayan yeni bir katman ekler. Karmaşık kavramsal yapıların tanımlanmasını ve biçimsel olarak Web kaynaklarında kullanılan sınıfların ve özelliklerin anlamlarını DL (*"Description Logic"*) denilen bir biçimsellik ile tanımlamayı sağlar. OWL ile ilgili detaylı bilgi (McGuiness and van Harmelen, 2004)'de bulunabilir.

2.1.2.1 Anlamsal web servisleri

Web servisleri bir URI ile tanımlanan ve herkesin kullanımına açık olan arayüzlerinin ve bağlantılarının XML ile ifade edildiği yazılım sistemleri olarak tanımlanabilir (Sycara et al., 2003a). Şu an kullanılmakta olan web servisleri kendilerini temsil eden ve WSDL (*"Web Services Description Language"*) kullanılarak hazırlanan ara yüzleri sayesinde geliştirildikleri yazılım dili ve/veya ortamına bağlı

kalmaksızın yine çok çeşitli ortamlarda çalışan istemci yazılımlar tarafından kullanılabilmektedirler. Örneğin Java programlama dili ile geliştirilen bir döviz kuru bilgilendirme web servisinin, WSDL'in XML ile şekillendirilmiş üst veri tanımları sayesinde C++ programlama dili kullanılarak hazırlanan bir istemci tarafından bulunup çalıştırılması mümkündür. İstemci program WSDL'i işleyerek, sunulan servise ait işlemi ve bu işlem için gerekli girdi – çıktı parametrelerini öğrenir ve servisi WSDL gibi yine web servisleri için bir standart olan SOAP (*"Simple Object Access Protocol"*) kullanarak çalıştırabilir. Ancak bu mevcut web servis altyapısı sadece sözdizimsel birlikte işlerliği (*"interoperability"*) göz önünde tutmaktadır ve yine (Sycara et al., 2003a)'da belirtildiği gibi böyle bir yaklaşım ne anlamsal birlikte işlerliği ne de web servislerinin otomatik tümleşimini (*"composition"*) mümkün kılar. Söz konusu birlikte işlerliği ve tümleşimi sağlamak amacıyla web servislerinin yeteneklerinin servis ontolojilerinde tutulması ve bu ontolojiler kullanılarak ihtiyaca en uygun servislerin bulunmasına ve dinamik çağrımının gerçekleştirilmesine çalışılmaktadır.

Servis yeteneklerinin tanımlanması ve servis çalıştırma sürecinin ifade edilmesine yönelik OWL-S (*"OWL for Services"*) (OWL-S Coalition, 2004) ve WSMO (*"Web Service Modeling Ontology"*) (WSMO Group, 2005) gibi çeşitli anlamsal web servis tanımlama ontolojileri literatürde bulunmaktadır. Tanımları ve kullanım yöntemleri bu tip ontolojilere göre verilmiş web servisleri anlamsal web servisleri olarak adlandırılmaktadır. Tezde ortaya konan geliştirme süreci özellikle yazılım etmenleri ile anlamsal web servislerinin etkileşimini desteklediğinden bu bölümde anlamsal web servisleri ile ilgili özet bir bilginin verilmesinde yarar vardır. Hem üzerinde daha fazla çalışılması hem de yine tez çalışması sırasında üzerinde çalışılan MAS geliştirme

ortamlarından birinin desteklemesinden dolayı OWL-S ontolojisi aşağıda kısaca anlatılmaktadır.

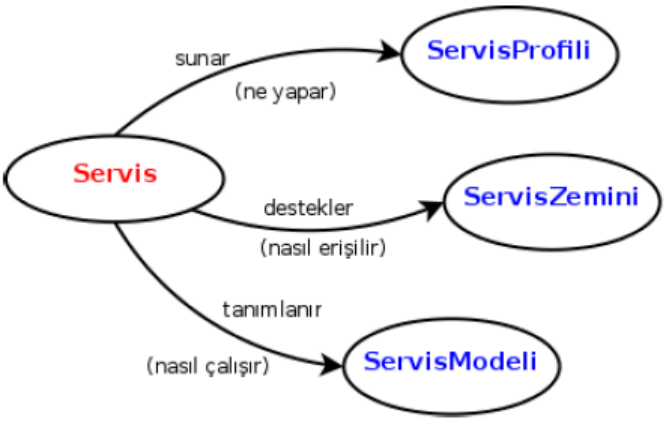
OWL-S (OWL-S Coalition, 2004) ile atomik veya tümleşik bir web servisi modellendiğinde aynı zamanda servise ait aşağıdaki üç tip anlamsal bilgi de modellenmiş olmaktadır:

1) Servisin yetenekleri veya yapabildikleri nelerdir?

2) Servis nasıl çalışmaktadır?

3) Servis nasıl kullanılmaktadır?

Bu sorulara cevap veren ilgili anlamsal web servisi ontoloji dokümanları sırasıyla Servis Profili (*"Profile"*), Servis Süreç Modeli (*"Process Model"*) ve Servis Zemini'dir (*"Grounding"*). Bu dokümanlar anlamsal yeteneğe sahip yapılar tarafından (örneğin etmenler) servis arama, bulma ve dinamik çağırma aşamalarında çıkarsama amaçlı olarak kullanılmaktadırlar (Şekil 2.2).



Şekil 2.2. OWL-S Ontolojisi üst seviyesi (OWL-S Coalition, 2004)

Bir servisin yeteneklerini gösteren Servis Profili, servisi arayan bir etmene servisin onun ihtiyaçlarını karşılayıp karşılamadığını belirlemede yardımcı olur. Servis Süreç Modeli ise servisin yürütülmesi için gereken detaylı girdi ve çıktı bilgilerini tanımlar. Bu model özellikle birçok servisi bir görev için birleştirmede önem kazanmaktadır. Birleştirme eylemi temel olarak servislerin girdilerini ve çıktılarını eşleştirir ve birleştirir. Ayrıca tümleşik servisin çalıştırmasına yönelik önşartlar ve servisin çalıştırılmasından sonra ortamda oluşturduğu etkiler de yine bu ontolojide tanımlanmaktadır. Son olarak Servis Zemini de bir servise nasıl erişileceğinin fiziksel detaylarını tanımlamaktadır. Servise ait iletişim protokolü, mesaj şekilleri ve servise bağlanmak için gereken soket, port numarası gibi özel detayları servis zemini içermektedir. Bir anlamda anlamsal web servis tanımlarıyla aynı servise ait WSDL bilgilerinin eşleştirilmesi Servis Zemini'nde gerçekleşmektedir.

Bir anlamsal web servisine ait yukarıdaki OWL-S tanımlarının kullanılmasıyla servisin otomatik olarak keşfi, çalıştırılması, otomatik olarak başka servislerle tümleşimi ve çalışmasının izlenmesi (*"monitoring"*) mümkün olmaktadır. Bu tez çalışmasının altyapısı oluşturulurken etmen servis etkileşimlerinin belirlenmesi amacıyla anlamsal web servislerinin hazırlanıp kullanılmasına yönelik çalışmalar gerçekleştirilmiştir. Örneğin (Kardaş vd., 2005)'de anlamsal web servislerinin anlamsal eşleme sonucunda bulunması ve dinamik çağrımı üzerine yapılan çalışmalara ait tasarım ve uygulama seviyesinde elde edilen deneyimler ve bulgular yer almaktadır. Anlamsal web servislerinin MAS'lar içerisinde kullanılmasına yönelik (Gümüs et al., 2007; Gürcan et al., 2007)'de önerilen sistem mimarileri hazırlanırken de elde edilen bu deneyim aktarılmıştır.

2.1.3 Model güdümlü geliştirme

Bilgi ve ağ teknolojilerinde görülen ilerlemeler aynı zamanda yazılım sistemlerinin de giderek karmaşıklaşmasına neden olmaktadır. Bu karmaşıklıklarla başa çıkmak için yazılım mühendisliği alanında çalışmalarda bulunan araştırmacılar yeni yazılım geliştirme yaklaşımları, metotları ve teknikleri öne sürmektedirler. Bunlardan biri olan MDD (*"Model Driven Development"*) farklı soyutlama seviyelerindeki modelleri kullanarak yazılım geliştirmedeki karmaşıklığı azaltmayı hedeflemektedir.

MDD (Selic, 2003) bünyesinde üstmodellerin (*"metamodel"*) tanımlanmasını ve bu üstmodeller arasında model dönüşümlerinin sağlanarak gerçek sistem modellerinin soyuttan somuta doğru elde edilmesini barındırır. Model dönüşümü, farklı soyut seviyelerdeki modellerin otomatik eşlenmesi için bir veya daha fazla kaynak modelini girdi olarak alır ve gerekli bir dizi dönüşüm kuralına bağlı olarak yine bir veya daha fazla hedef modelini çıktı olarak ortaya koyar. Bu açıdan bakıldığında model güdümlü yazılım geliştirmenin ana ürünlerinin (*"artifact"*) yazılım modelleri olduğunun belirtilmesinde yarar vardır. Literatürde, MDD'nin derleyicilerden bu yana programlama teknolojisindeki ilk gerçek paradigma değişikliği olduğu öne sürülmektedir (Selic, 2003).

Bu tez çalışmasında Anlamsal Web üzerinde çalışacak etmen sistemlerine ait model güdümlü geliştirme süreci ortaya konurken OMG'nin (*"Object Management Group"*), MDD yaklaşımının bir gerçekleştirimi olarak önerdiği MDA (*"Model Driven Architecture"*) kullanılmıştır.

MDA (OMG, 2003) ile yazılım modellerinin çalıştırılabilir bileşenlere ve uygulamalara dönüştürülerek yazılım sistemlerinin

geliştirilmesi amaçlanmaktadır. Bu nedenle MDA, MOF (*"Meta-Object Facility"*) (OMG, 1997) çerçevesine dayanan birçok model dönüşümünü tanımlamaktadır.

MDA'de modeller geliştirim sürecine model dönüşüm zincirlerinden yazılım kodlarının üretilmesine kadar her safhada entegre olmuş durumdadırlar. Bu entegrasyonu sağlamak amacıyla MDA, modellerin MOF temelli bir dil ile ifade edilmesine ihtiyaç duymaktadır. Bu zorunluluk da modellerin MOF uyumlu depolarda (*"repository"*) saklanabilmesini, yine MOF uyumlu araçlar tarafından incelenebilmesini (*"parsing"*) ve dönüştürülebilmesini ve gerektiğinde de XMI'ya (*"XML Metadata Interchange"*) çevrilerek bir ağ üzerinde iletiminin sağlanmasını garanti eder (OMG, 2003).

MDA kapsamında üç soyutlama seviyesi tanımlanmaktadır: CIM (*"Computation Independent Model"*), PIM (*"Platform Independent Model"*) ve PSM (*"Platform Specific Model"*). Tüm bu soyutlama seviyelerinin en altında da kaynak kod yer almaktadır.

Yazılım geliştiriciler, CIM'leri kullanarak sadece bir iş problemi içerisinde yer alan varlıkları ve bu varlıklar arasındaki ilişkileri modellerler. Bu soyutlama seviyesindeki bir modelde uygulama ile ilgili herhangi bir bilgi yer almamaktadır. Örneğin MAS kullanılarak hayata geçirilecek bir CIM'de etmen yapıları ile ilgili bir bilgi yer almaz.

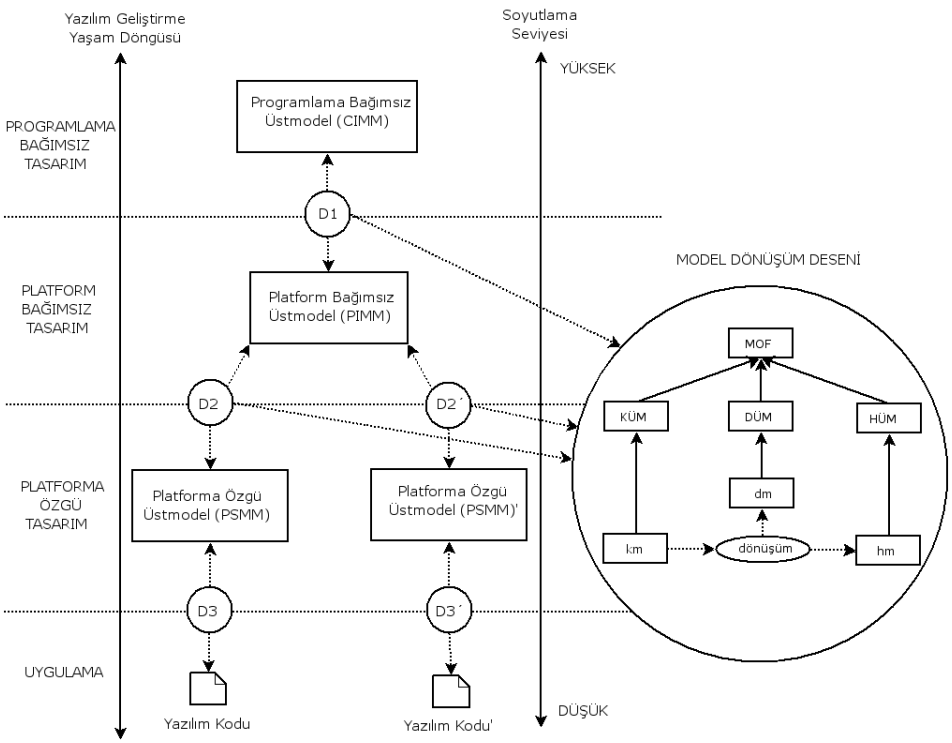
PIM'ler ise herhangi bir platformu temel almamalarına rağmen programlanabilir yapıları içlerinde barındırmaktadırlar. Bir PIM farklı bir çok platform için kullanıma uygun olan bir platform bağımsızlık seviyesi tanımlamaktadır (OMG, 2003). Örneğin Anlamsal Web ortamında çalışan MAS'lar için tanımlanan bir PIM spesifik bir etmen çerçevesine ait olmayan etmen varlıklarını (*"entity"*) ve ilişkilerini tanımlamaktadır.

Öte yandan bir PSM gerçek platform uygulamasının detaylarını barındırmaktadır. Bir PIM'de yer alan platform bağımsız varlıkların ve bu varlıklar arasındaki ilişkilerin model dönüşümleri sonrasında farklı PSM'lerde ifade edilmesi mümkündür. MDA yaklaşımının gücü ve esnekliği de bundan kaynaklanmaktadır. Örneğin yukarıda değinilen Anlamsal Web ortamında çalışan MAS PIM'inin platform bağımsız varlıkları ve ilişkileri model dönüşümleri sonrasında gerçek etmen çerçevelerinin PSM'lerine dayalı olarak ifade edilebilir. Böylece aynı MAS modelinin farklı ortamlarda hayata geçirilmesi mümkün olmaktadır. Anlamsal Web tabanlı etmen varlıklarının MDA kapsamında tanımlanması, CIM, PIM ve PSM ayrışımı ve gereksinimleri hakkında bir çalışma (Göknil vd., 2006)'da yer almaktadır.

Şekil 2.3'te MDA'ya dayalı yazılım geliştirme süreci ve model dönüşümleri verilmiştir. CIM, PIM ve PSM'lerin dayandığı üstmodeller sırasıyla CIMM (*"Computation Independent Metamodel"*), PIMM (*"Platform Independent Metamodel"*) ve PSMM'dir (*"Platform Specific Metamodel"*). CIMM'den PSMM'lere geçişler MOF üstmodeline dayanan model dönüşümleri ile adım adım belirtilmiştir.

Şekil 2.3'te sağ tarafta yer alan model dönüşüm deseninde bir kaynak modelin (km) bir hedef modeline (hm) dönüşümü gösterilmektedir. Söz konusu dönüşüm literatürde çeşitli önerileri (Duddy et al., 2003; Kalnins et al., 2005; Agrawal et al., 2006; Jouault and Kurtev, 2006) yapılan bir model dönüşüm dilinde yazılmış dönüşüm tanımlarına göre yürütülmektedir. Kaynak model, hedef model ve dönüşüm tanımı sırasıyla KÜM (Kaynak Üstmodeli), HÜM (Hedef Üstmodeli) ve DÜM'e (Dönüşüm Üstmodeli) uymaktadır. D1, D2 ve D2' ile ifade edilen MDA dönüşümlerinde kaynak ve hedef modeller yerine göre CIMM, PIMM ve çeşitli PSMM'lerin üstmodellerine uymaktadırlar. Modelden modele dönüşümlerden sonra sıradaki ve son adım

platformlara ait PSMM'leri kullanarak modelden yazılım koduna dönüşümleri gerçekleştirmek (D3 ve D3') ve gerçek sistem uygulamalarını hayata geçirmektir.



Şekil 2.3. MDA'ya dayalı yazılım geliştirme süreci ve model dönüşüm mekanizması

Bu tez kapsamında Anlamsal Web ortamında çalışan MAS'lar için bir PIMM ve çeşitli etmen geliştirme çerçeveleri için PSMM'ler hazırlanmıştır. Sözkonusu bu üstmodelleri kullanan modelden modele ve modelden koda dönüşüm adımları da önerilen MDD süreci için tanımlanmış ve hayata geçirilmiştir. Böylece bir MAS geliştiricisi istediği etmen sisteminin PIMM'e dayanan modelini inşa eder ve tanımlanan MDD sürecine girdi olarak verir. Önceden tanımlanmış olan dönüşümlerin otomatik işletilmesiyle geliştirici tasarladığı sistemin farklı

MAS ortamları için uygulamalarını elde etmektedir. Geliştiricilerin dönüşüm deseni ve dönüşümlerin çalıştırılmasına ait iç mekanizmalar ile uğraşmasına gerek yoktur.

2.2 İlgili Çalışmalar

Bu bölümde yazılım etmenlerinin modellenmesi ve model güdümlü geliştirilmesine yönelik diğer araştırmacılar tarafından gerçekleştirilen çalışmalar hakkında bilgi verilmektedir. İlk altbölümde etmen sistemlerinin modellenmesine yönelik literatürde yer alan belli başlı çalışmalar anlatılırken ikinci alt bölümde MDD kullanılarak etmen sistemlerinin geliştirilmesini öneren çalışmalar hakkında bilgi verilmiştir. Üçüncü altbölümde ise yapılan çalışmalara ait genel bir değerlendirme ve bu tez kapsamında ortaya konan çalışmanın bu çalışmalara göre farkları yer almaktadır.

2.2.1 MAS modelleme çalışmaları

AOSE (*"Agent Oriented Software Engineering"*) araştırmaları bünyesinde bir çok MAS geliştirme metodolojisi ve bunlara ait çeşitli etmen üstmodelleri önerilmiştir. Ancak etmen araştırmacıları tarafından önerilen bu üstmodeller (Pavon et al., 2006)'da belirtildiği gibi sadece üzerinde çalışılan metodolojinin kavramlarını biçimsel bir şekilde tanıtmayı hedeflemektedirler ve genel amaçlı değildirler. Örneğin (Bernon et al., 2005)'de ADELFE (Bernon et al., 2003), GAIA (Zambonelli et al., 2003) ve PASSI (Cossentino and Potts, 2002) MAS metodolojilerinin üstmodelleri tanıtılmıştır. Benzer bir çalışma da SODA metodolojisi (Omicini, 2000) için Molesini et al. (2005) tarafından yerine getirilmiştir. Bu çalışmada SODA metodolojisinin etkileşim ve sosyal bakış açılarının modellenmesi ve bunlara ait bir üstmodelin çıkarılması hedeflenmiştir. Bu üstmodellerin MAS geliştirme araçlarının

hazırlanması sırasında temel alınmasına da ancak yakın zamanda başlanmıştır.

MAS'lar için literatürde yer alan genel amaçlı ilk üstmodelin AALAADIN (Ferber and Gutknecht, 1998) olduğu söylenebilir. Bu üstmodel, MAS yapısını üç ana kavram ile temsil etmektedir: etmen, grup ve rol. Aynı çalışmada bu üstmodel kavramlarının *"MadKit"* adı verilen platform üzerinde gerçekleştirimi yerine getirilmiştir.

MAS modellemeye yönelik literatürde adı en çok geçen üstmodel ve modelleme dili çalışmalarının UML'i (*"Unified Modeling Language"*) temel aldığı ve/veya UML'i uzatan yapılar sunduğu görülmektedir. Depke et al. (2001), UML gösterimine dayalı bir etmen tabanlı modelleme tekniği tanıtmışlardır. Çalışmada sunulan modelleme tekniği çizge dönüşüm kurallarını sistem ihtiyaçlarını tanımlamada ve sistem analizinde kullanarak etmen birlikte çalışılabilirliğini ve özerkliğini modellemeyi hedeflemektedir. Tasarım aşamasında ise bu çizge dönüşüm kuralları etmenlerin lokal faaliyetlerinin etkilerini tanımlamaktadırlar.

UML'i temel alan ve/veya uzatan diğer önemli etmen modelleme çalışmalarının etmen tasarlamaya yönelik tanımladıkları yapılar ve sundukları modelleme uzantıları hakkında aşağıdaki alt başlıklarda bilgi verilmiştir. Daha geniş bir inceleme ise (Kardaş ve Dikenelli, 2006)'da yer almaktadır.

2.2.1.1 AUML

AUML ("*Agent UML*") (Bauer et al., 2001), UML'in paket, şablon, sıra diyagramları, etkileşim diyagramları, aktivite diyagramları ve sınıf diyagramlarına etmen temelli uzantılar getirmiştir. UML model anlamsalları bir üstmodel ile temsil edilmektedir.

AUML etmen etkileşim protokolleri için üç katmanlı bir gösterim şekli sunmaktadır: Protokol bütünü göstermeye yönelik katman, etmenler arasındaki etkileşimleri gösteren katman ve içsel etmen işleyişini gösteren katman.

AUML, protokol bütününü temsil etmek için UML'in paket ve şablon yapılarını kullanmaktadır. Etmenlerin birbirleri ile olan etkileşimlerini modellemek amacıyla da AUML'de, UML'in sıra diyagramlarını, etkileşim diyagramlarını, aktivite diyagramlarını ve durum şemalarını uzatan ("*extend*") yapılar tanımlanmıştır. Etmen iç işleyişinin modellenmesi için yine aktivite diyagramları ve durum şemalarından yararlanılmaktadır.

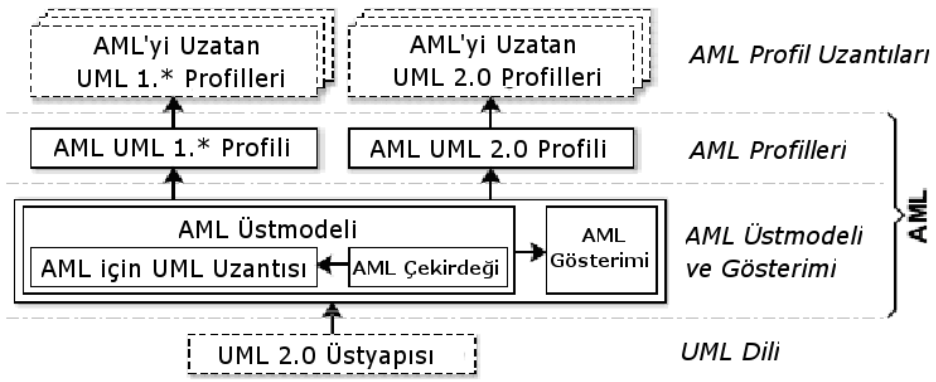
Etmen sistemleri için modelleme dilleri üzerine (Huget, 2005)'de yer alan çalışmada ideal bir modelleme dilinin grafiksel ya da metinsel bir notasyona sahip olması, bir semantiğinin olması, soyut yapılar barındırması, farklı bakış açıları içermesi ve belli başlı araçlara (tasarım, kod geliştirme, vb.) sahip olması gerekliliği vurgulanmıştır. Bu özellikler ele alındığında Agent UML'in aşağıdaki dezavantajlara sahip olduğu görülmektedir:

- Görsel gösterimi eksiktir ve nesne tabanlı sistemlere bağımlıdır.
- Diğer geliştiricilerle bilgi alışverişini sağlayacak metinsel gösterimi bulunmamaktadır.
- Anlam bilgisi yarı-biçimseldir.
- Sunduğu tanımlamaları kullanma adına hiçbir araç şu an için mevcut değildir.

2.2.1.2 AML

Whitestein Technologies şirketi tarafından geliştirilen AML (*"Agent Modeling Language"*) (Cervenka et al., 2005) yarı-biçimsel ve görsel bir modelleme dilidir. UML 2.0'ın bir uzantısıdır. Şekil 2.4'te verilen AML yapısı incelendiğinde AML'nin iki ayrı katmanla tanımlandığı görülür: *AML Üstmodeli ve Gösterimi* ve *AML Profilleri*.

UML 2.0 Üstyapısı'nın (*"superstructure"*) (OMG, 2004) UML'in soyut sözdizim, anlam ve gösterimini tanımlamasından hareketle *UML Dili* katmanı, AML'nin çoklu etmen sistemlerine özel modelleme yapılarını tanımlamasına izin verir.



Şekil 2.4. AML tanımı katmanları (Cervenka et al., 2005)

AML Üstmodeli ve Gösterimi katmanı AML yapısını *AML Çekirdeği* ve *AML için UML Uzantısı* paketleri ile tanımlar. *AML Çekirdeği* paketi AML'ye özel modelleme elemanlarının tanımlandığı çekirdektir. Etmen mimarisi, davranışları ve zihinsel bakış açılarına yönelik yapılar bu çekirdekte yer alır. *AML için UML Uzantısı* paketi standart UML elemanları üzerine üst özellikler ve yapısal kısıtlar ekler.

AML Profilleri katmanında, *AML Üstmodeli ve Gösterimi* katmanı üzerine iki UML profili inşa edilmiştir: UML 1.* sürümlerine dayanan *AML UML 1.* Profili* ve UML 2.0 sürümüne dayanan *AML UML 2.0 Profili*. Bu profiller var olan bilgisayar destekli sistem geliştirme araçları ile AML'nin hem UML 1.* sürümleri hem de UML 2.0 sürümü için uygulanmasını sağlamayı amaçlamaktadırlar.

AML Profilleri üzerine başka kullanıcılar kendi dil uzantılarını geliştirebilir ve AML'i kendi ihtiyaçları doğrultusunda uzatabilirler. Bu uzantılar *AML Profil Uzantıları* adı altında toplanabilir.

(Huget, 2005)'de de belirtildiği gibi AML, AUML'e göre daha yeni bir çalışmadır. Ancak onun da metinsel gösteriminin eksik olduğu ve herhangi bir geliştirme aracı tarafından şu an için desteklenmediği görülmektedir.

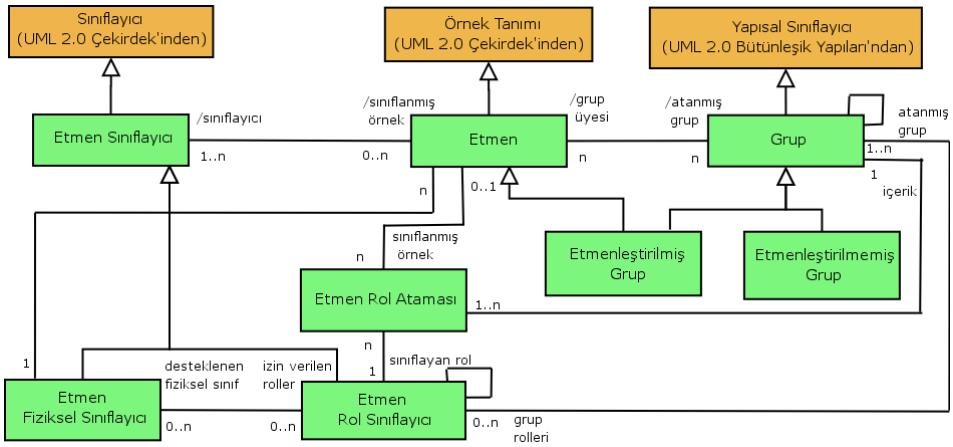
2.2.1.3 FIPA ACSM

FIPA Modelleme Teknik Komitesi ve OMG Etmen Özel İlgi Grubu bir MAS bünyesinde yer alan etmenler, etmen rolleri ve etmen grupları arasındaki ilişkileri belirtmek amacıyla yakın geçmişte bir çalışma yürütmüşlerdir. Bu çalışmanın ürünü olarak FIPA ACSM (*"Agent Class Superstructure Metamodel"*) (FIPA Modeling TC, 2004) adı verilen bir etmen üstmodeli ortaya konmuştur. Etmen organizasyonlarının biçimsel bir gösterimini sunan bu üstmodel hem UML üstyapısına dayanır hem de bu üstyapının bir uzantısıdır.

ACSM, her ne kadar geliştiricilerinin de belirttiği gibi (FIPA Modeling TC, 2004) başlangıç seviyesinde bir etmen ilişki modeli tanımlamış olsa da ACSM'nin etmenleri, rollerini ve gruplarını modellemek amacıyla ihtiyaç duyulan kullanıcı seviyesi yapıları tanımlayan uygun bir üstyapı olduğu görülmüş ve bu tez kapsamında

geliştirilen platform bağımsız MAS üstmodelinin ACSM'de tanımlanan varlıklar ve bu varlıklar arası ilişkileri miras edinmesi sağlanmıştır. Böyle bir miras ilişkisinin tezde önerilen MAS PIMM'ine getirmiş olduğu faydalar tezin bu PIMM'i içeren bölümünde açıklanmaktadır.

ACSM soyut sözdizimi Şekil 2.5'te görülmektedir. Model elemanları ve ilişkileri hakkında bilgi aşağıda verilmiştir. ACSM hakkında daha detaylı bilgiye ise (FIPA Modeling TC, 2004) ve (Odell et al., 2005)'den ulaşılabilir.



Şekil 2.5. FIPA ACSM soyut sözdizimi (FIPA Modeling TC, 2004)

Etmen Sınıflayıcı etmenlerin ne şekilde sınıflandırılacağını tanımlar ve iki altsınıfa sahiptir: *Etmen Fiziksel Sınıflayıcı* ve *Etmen Rol Sınıflayıcı*. Etmen Fiziksel Sınıflayıcı bir etmenin temel ihtiyaçlarını karşılayacak esas (ya da ilkel) sınıfları tanımlar. Bu sınıflar daha çok etmen yapılarının hayata geçirildiği fiziksel platform (örneğin JADE (Bellifemine et al., 2001), Cybele (Intelligent Automation Inc., 2002) veya SEAGENT (Dikenelli et al., 2005)) ile alakalıdır. Doğal olarak etmenler de fiziksel platforma bağlı olarak belirli özellikler ve yetenekler kazanırlar. Bu özellikler etmenin yaşamı boyunca devam eder.

Öte yandan Etmen Rol Sınıflayıcı etmenleri ortamlarda oynayacakları rollere göre sınıflar. Dolaysıyla etmenlerin yeteneklerini ve aktivitelerini içerir. Etmenlerin oynadıkları roller zamanla değişebilir.

Etmen sınıfı bir sistemin bireylerini oluşturan tüm etmen kümelerini tanımlar. Her bir Etmen örneği kendisinin gerekli özelliklerini tanımlayan bir ya da daha fazla Etmen Sınıflayıcı ile ilişkilidir.

Grup'lar ise belli bir amaç için toplanmış etmen sınıflarını temsil eder. Grup içerisinde etmenler kendi rollerine göre birbirleri ile etkileşimde bulunurlar. Bu nedenle bir Grup bir dizi rolle (ya da geçişlilik özelliği düşünülecek olursa etmenle) tanımlanır.

Bir grup, bir etmenin sahip olabileceği tüm özelliklere sahip olabilir. Örneğin mesaj alışverişinde bulunabilir veya direkt olarak bir rol üstlenebilir. Böyle bir grup aynı zamanda bir etmendir ve bu nedenle hem Grup hem de Etmen sınıfının altsınıfıdır. Bu tip gruplara *Etmenleştirilmiş Grup* adı verilmektedir. Öte yandan *Etmenleştirilmemiş Grup*'lar ise modelde birinci sınıf varlıklar olarak bulunan, etmen özellikleri göstermeyen ve daha çok nesneye benzeyen yapılardır.

Etmenler ve bunların Etmen Fiziksel Sınıflayıcı ve Etmen Rol Sınıflayıcı'ları ile aralarındaki ilişki şöyledir: Etmen ile Etmen Fiziksel Sınıflayıcı arasındaki ilişki etmenin temel yeteneklerini ve ihtiyaçlarını gösterir. Her etmen, bir Etmen Fiziksel Sınıflayıcı ile ilişkilendirilmelidir. Etmen ile Etmen Rol Sınıflayıcı arasındaki ilişki ise ilgili etmenin ne tür aktivitelerde bulunacağını göstermektedir. Bir etmenin mutlaka bir Etmen Rol Sınıflayıcı ile ilişkilendirilmesine gerek yoktur ama böyle bir etmen bir Grup içerisine alınamaz.

Etmen Rol Ataması, etmenler, roller ve gruplar arasında üçlü bir ilişki olarak tanımlanabilir. Her bir Etmen Rol Ataması örneği bir rolü, bir grubu ve bir de etmeni birbiri ile ilişkilendirir. Etmen rolleri Etmen

Rol Sınıflayıcı'ları ile temsil edilmektedirler. Her bir Etmen Rol Ataması'nın bir rolü ve grubu olması gerekirken her zaman için ilişkilendirilmiş bir etmeni olmayabilir. Bu tip atamalar *pozisyon* olarak adlandırılır (Odell et al., 2005).

Dikkat edilirse ACSM daha önce de belirtildiği gibi UML'i belli açılardan uzatmakta ama özellikle Sınıflayıcı yapısı ile aynı zamanda UML'i temel almaktadır. Yani bir anlamda UML üstmodelini kendi modeline dahil eder. Bu da etmenleri modellerken nesne tabanlı tasarımdan gelen kısıtların (ya da zorlamaların) ortadan kalkmasını sağlamaktadır.

2.2.2 Model güdümlü MAS geliştirme çalışmaları

Model güdümlü etmen geliştirme yaklaşımı bugünlerde AOSE araştırmacıları için önde gelen araştırma konularından biri olmaya başlamıştır. Geçmişi çok eski olmayan bu AOSE alt alanı için etmen araştırmacılarının genellikle önerileri MAS'lar için yeni MDA'lar tanımlamak, MAS geliştirimi sırasında belli bir sisteme özgü üstmodelleri ve model dönüşümlerini kullanmak veya varolan MAS geliştirme metodolojilerini MDD paradigmasına uygun olarak yeniden organize etmeyi kapsamaktadır. Aşağıdaki alt başlıklarda diğer araştırmacıların model güdümlü MAS geliştirme ile ilgili katkıları bu kapsama gruplarına göre verilmiştir.

2.2.2.1 MDA yaklaşımları

Bauer ve Odell (Bauer and Odell, 2005)'deki çalışmalarında etmen tabanlı sistemlerin hazırlanmasında UML 2.0 ve MDA'in kullanımından söz etmiş; bir MAS'ın hangi bakış açılarının CIM ya da PIM olarak ele alınabileceğini değerlendirmişlerdir. Çalışmada bir etmen CIM'inin, *kullanım durumları* (*"use case"*), *çevre modeli*, *iş alanı* (*"domain"*) /

ontoloji modeli, *rol modeli*, *hedef/görev modeli*, *etkileşim modeli*, *organizasyon/toplum modeli* ve *iş süreci modelleri* adı verilen çeşitli bakış açılarını içermesi önerilmiştir. Bunlardan örneğin kullanım durumları bakış açısında, sistemin işlevsel ihtiyaçlarının belirlenmesi için kullanım durum senaryolarının uygun olduğu savunulmaktadır. Öte yandan çevre modelinde etmenlerin fiziksel ortamının ve iletişiminin dikkate alınması gerektiği belirtilmiştir. Organizasyon modelinde ise etmenlerin çalıştığı ve etkileşimde bulunduğu toplumun modellenmesi öngörülmektedir.

(Bauer and Odell, 2005)'de önerilen platform bağımsız bakış açıları ise *etkileşim protokolü modeli*, *içsel etmen modeli*, *etmen modeli*, *servis / yetenek modeli*, *benzerlik modeli* ve *etmen örnek modeli*dir. Etkileşim protokolü modeli adından da anlaşılacağı üzere çeşitli etmen sınıflarının etmen sınıf örneklerinin ve rollerinin birbirleri ile olan etkileşimlerini tanımlamaktadır. İçsel etmen modeli etmen amaçlarını, inançlarını ve planlarını modellemektedir. Etmen modeli ise etmenlerin ve etmen gruplarının davranışlarını modeller. Benzerlik modeli birlikte çalışılabilme amacıyla bir etmene diğer etmenlerin inançlarını, yeteneklerini ve niyetlerini sağlamayı hedeflemektedir. Çalışmada sözü edilen bu CIM ve PIM bakış açıları oldukça kavramsaldır ve ilgili modeller somutlaştırılmamıştır. Ayrıca bütünleşik bir MDA için gerekli olan PSM tanımları verilmemiştir.

(Gracanin et al., 2005)'de tanıtılan *"Cougaar MDA"* etmen yazılım ürünlerini elde etmek amacıyla tasarım seviyesini, bireysel bileşenlerden model tanımlarına yükselterek etmen sistemleri için üst seviye bir uygulama çerçevesi sunmaktadır. Uygulama platformu olarak kavramsal planlama modelini temel alan bir etmen mimarisi içeren *"Cougaar" ("Cognitive Agent Architecture")* kullanılmıştır. Çalışmada, Cougaar mimarisine dayalı olarak sistem geliştirmenin önemli bir miktarda varlık

eşlemelerine ve dönüşümlerine ihtiyaç duyması nedeniyle zor olduğu belirtilmiş; MDA'in sistem ihtiyaçlarını sistematik bir şekilde yakalamayı ve bu ihtiyaçları PIM'den PSM'e, oradan da kod seviyesine eşlemeyi sağlamasından dolayı Cougaar sistem geliştiricilerinin verimliliğinin arttırılacağı savunulmuştur.

Cougaar MDA (Gracanin et al., 2005), Cougaar'a dayalı olarak yazılım geliştirmeyi kolaylaştırmak amacıyla iki önemli soyutlama katmanı tanımlamaktadır. İlk katman olan *GDAM ("Generic Domain Application Model")* sistem geliştirme için bir PIM'i temsil etmektedir. Bu model genel etmen ve iş alanı bileşenlerini modellemektedir. İkinci katman olan *GCAM ("Generic Cougaar Application Model")* ise PSM'i ve Cougaar mimarisini yansıtmaktadır. Uygulama gereksinimleri iş akışına özgü bir süreç tanımlama dili ile ifade edilmekte ve tanımlanan bir otomatik dönüşümle bu gereksinimlerin GDAM karşılığı elde edilmektedir. GDAM katmanı ihtiyaçları temsil ederken GCAM katmanı sistem tasarımını temsil etmektedir. Platforma özgü GCAM bileşenleri önerilen süreç sonunda Cougaar / Java kodlarına dönüştürülmektedir.

Cougaar MDA tabanlı yaklaşım her ne kadar kavramsal planlama tabanlı etmen sistemleri için bütünleşik bir MDA önerse de çalışmanın kendisinde de belirtildiği gibi şu an için sadece Cougaar mimarisine dayalıdır (Gracanin et al., 2005).

Amor et al. (2005) etmen tabanlı tasarım ve MAS gerçekleştirimi arasındaki boşluğu doldurmak amacıyla MDA tabanlı bir süreç önermişlerdir. Her etmen tabalı sistem geliştirme metodolojisinden (GAIA (Zambonelli et al., 2003), Tropos (Bresciani et al., 2004), vb.) her etmen platformuna (JADE (Bellifemine et al., 2001), Cybele (Intelligent Automation Inc., 2002), vb.) direkt dönüşümün hemen hemen imkansız olmasından hareketle geçişi sağlayacak ara bir nötr etmen platform

mimarisinin bulunması gerektiği savunulmuş ve bu amaçla *"Malaca"* adı verilen bir etmen mimarisi çalışmada tanıtılmıştır.

Malaca etmenleri tanımlanan mimari gereği FIPA uyumlu herhangi bir etmen platformu üzerinde çalışabilmektedir. Böylece etmen tabanlı metodolojiler ile Malaca arasında ve daha sonra Malaca ile farklı etmen platformları arasında eşlemeler tanımlanabilirse ilgili metodolojiler ve platformlar arasında bağlantı kurulabilecektir. Savunulan fikri desteklemek amacıyla çalışmada Tropos (Bresciani et al., 2004) tasarım modeli ile Malaca etmen modeli arasındaki dönüşümün MDA'in uygulanması ile nasıl gerçekleştirilebileceği anlatılmıştır. UML modelleme araçları için yeni şablonlar tanımlanmıştır ve dönüşüm deseninde Tropos tasarım modeli PIM, Malaca modeli ise PSM olarak kabul edilmiştir.

Önerilen yöntemin eksikleri yine (Amor et al., 2005)'de belirtildiği gibi model dönüşümlerinin otomatik olmaması, tasarım safhasında kullanılan diyagramların ve metinlerin dönüşüm sürecinde yorumlanması için standart bir notasyona sahip olması gerekliliği ancak bunun var olan etmen metodolojilerinde bulunmaması ve söz konusu MDA uygulamasını destekleyecek uygun araçların bulunamaması nedeniyle mimarinin tam olarak hayata geçirilememesidir.

2.2.2.2 MAS geliştirme için üstmodel tanımlama ve model dönüşümü çalışmaları

(Jayatilleke et al., 2004)'de tanımlanan iş alanı bağımsız bileşen tiplerine ait kavramsal çerçeve, etmen sistemlerinin temsilini ve ihtiyaç hissedildiğinde yapılarının değiştirilmesini hedeflemektedir. Sunulan bileşen çerçevesinde bileşenlere ait tip tanımları birer XML DTD (*"Document Type Definition"*) ile gösterilmekte; iş alanına özgü bileşen tanımları da bu DTD'lere uyan XML elemanları olarak dönüşüm

modülüne verilmektedir. Dönüşüm modülünde XSLT (*"eXtensible Style Language Translation"*) kuralları ve işletilebilir ikili (*"binary"*) bileşenler kullanılarak iş alanına özgü bileşenlerin işletilebilir etmen kodlarının elde edilmesine çalışılmaktadır. Tanıtılan çerçeveye ait XML tabanlı iş alanı bileşen tanımlama işlemi bir hava durumu uyarı sistemi örneği verilerek yine aynı çalışmada değerlendirilmiştir. Ancak çalışmada, bileşen çerçevesine ait dönüşüm modülü ve dolaysıyla sürecin etmen yazılımı kod üretimi kısmı hayata geçirilmemiştir. Aynı araştırmacıların (Jayatilleke et al., 2007)'deki çalışmasında ise önerdikleri bu bileşen çerçevesinin *"CAFnE"* adı verilen geliştirilmiş ve bir dizi araç takımı ile desteklenmiş halinin bir değerlendirmesi yer almaktadır.

Model dönüşümlerinin etmen tabanlı sistem geliştirme amacıyla kullanılmasına yönelik bir çalışma (Perini and Susi, 2006)'da verilmiştir. MDA'in dönüşüm deseninin uygulandığı bu çalışmada Tropos (Bresciani et al., 2004) yapılarından UML modellerine dönüşümler hazırlanmış ve MOF QVT (*"Query/View/Transformation"*) (OMG, 2007) dili kullanılarak bu dönüşümler hayata geçirilmiştir. Çalışmada özellikle Tropos plan ayrıştırma (*"decomposition"*) yapısının bir UML 2.0 aktivite diyagramına dönüşümü üzerinde durulmuştur. Bir Tropos plan yapısında yer alan planların bir UML aktivite diyagramındaki eylem düğümlerine (*"node"*) eşlenmesi ve ilgili dönüşüm kurallarının yazılması örneklenmiştir. Sözü edilen model dönüşümlerini otomatik olarak uygulamak amacıyla hazırlanan ve *"TAOM4e"* adı verilen bir yazılım aracı da yine (Perini and Susi, 2006)'da açıklanmıştır. TAOM4e'nin grafiksel ara yüzü kullanılarak hazırlanan Tropos sistem modelleri üzerinde yine TAOM4e bünyesinde yer alan dönüşüm motoru ilgili MOF QVT kurallarını işlemekte ve çıktı modellerini elde etmektedir.

(Perini and Susi, 2006)'daki çalışma hem MDA model dönüşüm desenini etmen tabanlı sistemlerin geliştirilmesinde uygulayacak yapıları tanımlaması hem de ilgili model dönüşümlerini gerçekten uygulayacak bir yazılım aracını sunması bakımından kayda değer bir çalışmadır. Ancak dönüşümlerde kullanılan ve çalışma için bir PIMM olarak kabul edilebilecek etmen üstmodelinin sadece Tropos yapılarını içermesi, geliştirme sürecinin çok fazla Tropos metodolojisine bağımlı olması ve örneklemelerin sadece Tropos plan yapılarını dikkate alması çalışmanın dezavantajları olarak sayılabilir.

Etmen sistemlerinin model güdümlü geliştirilmesi sırasında bu sistemlerin SOA'lar (*"Service-Oriented Architecture"*) ile birlikte çalışılabilirliğini dikkate alan çalışmalar da bulunmaktadır. (Zinnikus et al., 2006)'da SOA'ların hızlı bir şekilde modellenmesi için yeni bir çerçeve önerilmektedir. Çerçeve, MDD tekniklerinin uygulanması ile ilgilenen bir modelleme kısmını, web servisleri için esnek bir iletişim platformunu ve SOA'larda pazarlık ve arabuluculuğu sağlayacak olan özerk bir etmen bileşenini içermektedir. Çerçeve, MDA yaklaşımının uygulanması amacıyla SOA'lar için bir PIMM ve web servisleri, BPEL (*"Business Process Execution Language"*) (Andrews et al., 2003) süreçleri ve JACK (AOS, 2006) etmen ortamlarında çalışan BDI (Rao and Georgeff, 1995) etmenleri için PSMM'ler tanımlamaktadır. SOA'lar için önerilen PIMM ve JACK BDI etmenleri için önerilen PSMM, (Hahn et al., 2006)'da detaylandırılmış ve bu üstmodeller arasındaki model dönüşümleri tanıtılmıştır. Tanımlanan model elemanı eşlemelerine uygun olarak SOA'ya ait üstmodeldeki sıralı süreç yapısından etmen üstmodelindeki takım planlarına dönüşümler gerçekleştirilmiştir. (Hahn et al., 2008)'de ise SOA'lar için oluşturulan üstmodele benzer bir şekilde bu sefer MAS'lar için platform bağımsız bir üstmodel önerilmiştir. Söz konusu PIMM, etmen sistemlerinin oluşturulması sırasında MAS, Etmen,

Davranış, Organizasyon, Rol, Etkileşim ve Ortam bakış açılarını göz önüne alan çeşitli alt üstmodellerde tanımlanmış olan varlıkların ve bu varlıklar arasındaki ilişkilerin bir bileşimidir.

2.2.2.3 MAS geliştirme metodolojilerinin MDD'ye uygun olarak yeniden organizasyonu çalışmaları

Bazı MAS geliştirme metodolojilerinin sahipleri MDA yaklaşımının gittikçe popüler olmasını ve MAS geliştirmede sağladığı kolaylıkları göz önüne alarak varolan metodolojilerini MDA'ya uygun olarak yeniden düzenleme çalışmaları gerçekleştirmişlerdir. Örneğin (Pavon et al., 2006)'da INGENIAS (Pavon et al., 2005) etmen geliştirme metodolojisinin MDD paradigmasına uygun olarak yeniden tasarımı anlatılmaktadır.

Her ne kadar INGENIAS'ın eski versiyonunda da MDD'yi destekleyen etmen üstmodel tanımları, modelleme araçları ve modellerin dönüşümleri yer alsa da bu eski versiyonda tanımlanan üstmodellerin yeni uygulama alanlarının ihtiyaçlarını karşılayamadığı ve farklı hedef platformları için model kod dönüşümlerini sağlayamadığı gözlenmiştir. (Pavon et al., 2006)'da savunulan yenileme ile INGENIAS'ta yer alan model oluşturma, tanımlama ve dönüşüm işlemlerinin MAS bağlamında uygunluğunun arttırılması hedeflenmiştir. Bu amaca uygun olarak bir etmen geliştirme ortamı tanıtılmıştır ve bu ortamda bir dönüşüm modülünü geliştirmek için gerekli süreç adımları belirtilmiştir. MDD prensiplerine uygun olarak revize edilmiş INGENIAS etmen geliştirme süreci de yine aynı çalışmada yer almıştır.

(Pavon et al., 2006)'daki sözü edilen INGENIAS üstmodeli ve dönüşümler INGENIAS harici başka etmen geliştirme ortamlarını desteklememektedir (örneğin JADE (Bellifemine et al., 2001)). MDA yaklaşımına ters düşen bu eksiklik çalışma sahiplerince de (Pavon et al.,

2006)'da vurgulanmış ve çözüm için uygulama ihtiyaçlarından türeyebilecek kısmi bir dönüşüm süreci önerilmiştir. Ancak bu önerinin de geliştiriciler için bütün bir çözüm getirmediği ve üzerinde çalışılması gerektiği vurgulanmıştır.

Benzer bir MAS metodoloji revizyonu (Penserini et al., 2006)'da anlatılmaktadır. MDA fikri ve standartları Tropos (Bresciani et al., 2004) metodolojisine ait modelleme sürecinin ve yardımcı araçlarının yenilenmesi amacıyla kullanılmıştır. Tropos bünyesinde tanımlanan ihtiyaç belirleme safhalarından MAS'ların detaylı tasarımı ve uygulanması safhalarına kadar tüm sürecin iyi izlenmesi amacıyla Tropos yetenek tanımları gözden geçirilmiştir. Tropos'un etmen tabanlı kavramsal modeli platform bağımsız model olarak ele alınmış; bu model ile JADE'in (Bellifemine et al., 2001) etmen tabanlı modeli arasında varlık eşlemeleri gerçekleştirilmiştir.

(Rougemaille et al., 2007)'de yer alan çalışmada uyarlamalı (*"adaptive"*) MAS paradigmasına uygun olarak ADELFE (Bernon et al., 2003) metodolojisine bir MDD safhasının eklenmesi hedeflenmiştir. Göz önüne alınan iki uyarlama seviyesi fonksiyonel (*"functional"*) ve işlevsel (*"operational"*) adını almıştır. Fonksiyonel seviye uygulama bağımlı olup etmenlerin karar verme sürecine daha yakındır. İşlevsel seviye ise etmenlerin temel yetenekleri ile ilgilidir. Önerilen yeni metodoloji ve araçları uyarlamalı etmen sistemlerini dikkate almaktadır. Hedeflenen, bir MAS geliştiricinin fonksiyonel uyarlamayı belli bir üstmodele uyan modeller halinde tanımlayabilmesi sonra bu modeller ile işlevsel seviyeye karşılık gelen bir etmen mimarisine ait model arasında eşlemeler tanımlayarak uyarlamalı MAS'lar için gerekli yazılımları elde etmektir. Ancak söz konusu bu seviyelendirmeler ve model dönüşümleri şu an için kavramsaldır ve özellikle dönüşümler uygulamaya geçirilmemiştir.

2.2.3 İlgili çalışmalara ait genel bir değerlendirme

Bölüm 2.2.1 ve 2.2.2'de anlatılan MAS modelleme ve model güdümlü geliştirme çalışmaları AOSE için oldukça yeni olan ilgili araştırma konuları için başlangıç düzeyindeki ilk gayretler olarak nitelenebilir. Şu ana kadar ne tam bir MDD süreci ne de tüm etmen araştırmacılarının üzerinde hemfikir olduğu bir MAS üstmodeli geliştirilebilmiştir. Araştırmacıların etmen sistemlerine farklı bakış açılarından yaklaşması ve farklı ortamlara yönelik varlık etmen bileşeni tanımlamalarından dolayı yakın zamanda da böyle bir genel modelin ve sürecin ortaya konulması zor görünmektedir.

Öte yandan Anlamsal Web (Berners-Lee et al., 2001) ortamı ve MAS'ların bu ortam üzerinde çalışabilmesi için ihtiyaç duyulan yapıların yukarıda değinilen çalışmalarda göz önüne alınmadığı ve özellikle etmen - servis etkileşimlerinin anlamsal boyutta desteklenemediği açıktır. Her ne kadar yukarıdaki bazı çalışmalarda MDD sürecine ait model tanımlama ve dönüşüm safhaları için değerli yaklaşımlar sunulmuş olsa da Anlamsal Web evriminin etmen sistemlerine getirdiği ve gittikçe zorunlu bir hale gelmeye başlayan Anlamsal Web yapıları ile birlikte işlerliğin önerilen süreçlerde yer almadığı gözlenmektedir. Bu tezde ortaya konan MDD sürecinin söz konusu bu eksikliğin giderilmesine yönelik bir katkı sağladığına inanılmaktadır.

Önerilen model güdümlü yaklaşımlarda görülen bir başka eksiklik de öneriyi getiren araştırmacıların zaten üzerinde çalıştıkları MAS geliştirme ortamları için PSMM'ler tanımlamaya çalıştıkları ve bunu model dönüşümlerinde kullandıkları ya da PIMM'den PSMM'lere dönüşümleri kendileri geliştirmemiş olsa dahi sadece tek bir platform modeli için gerçekleştirdikleridir. Oysa genel MDA yaklaşımı dikkate alındığında önerilen bir MDA'in değer kazanması için en az iki platform

için dönüşümlerin tanımlanmasının ve hayata geçirilmesinin gerekliliği açıktır. Böylece önerilen sistemin de farklı platformlar için değerlendirilmesinin ve uygulanabilirliğinin ortaya konması sağlanmış olacaktır. Tezde ortaya konan sürecin farklı iki etmen geliştirme ortamı için denenmiş olması bu açıdan da çalışmanın önceki çalışmalara göre bir artısını ortaya çıkarmaktadır.

Önceki bölümlerde sözü edilen çalışmalarla bu tezde ortaya konan çalışmanın bazı benzerlikleri de bulunmaktadır. Örneğin (Hahn et al., 2006)'da tanıtılan dönüşüm mekanizması ile bu tezdeki model dönüşümlerinin uygulanma prensiplerinde benzerlikler görünmektedir. Modelden modele dönüşümlerde aynı dönüşüm dili kullanılmış ve aynı araçlar kullanılarak hayata geçirilmiştir. Uygulama için oldukça popüler olan bu araçların her iki çalışmada da uygun bulunarak kullanılmış olması beklenebilecek bir durumdur. Ayrıca (Zinnikus et al., 2006)'da anlatılan PIMM ve PSMM dönüşümleri ile de bu tezdeki çalışma uygulama açısından benzerdir. Ancak hem (Hahn et al., 2006) hem de (Zinnikus et al., 2006)'da sağlanan dönüşümler etmen bağımsız bir SOA modelinden bir MAS modelinedir ve etmenler ile web servisleri arasındaki birlikte işlerliği sağlamaya çalışmaktadır. Oysa bu tezdeki çalışmada etmenlerin hem platform bağımsız hem de platforma özgü tasarımda birinci sınıf varlıklar olarak yer aldığı üstmodeller arasında dönüşümler sağlanmış ve özerk etmenler ile anlamsal web servisleri arasındaki birlikte işlerlik desteklenmiştir.

Literatürde anlamsal web servislerinin model güdümlü geliştirilmesi üzerine de bazı çalışmaların yürütüldüğü görülmektedir. (Grønmo et al., 2005)'de ortaya konan çalışma MDA tekniklerinin anlamsal web servislerinin geliştirilmesine yönelik kullanılmasına dair iyi bir örnektir. Grønmo et al. (2005), üst seviye grafiksel modellerin anlamsal web servisleri için bir entegrasyon platformu oluşturmasını

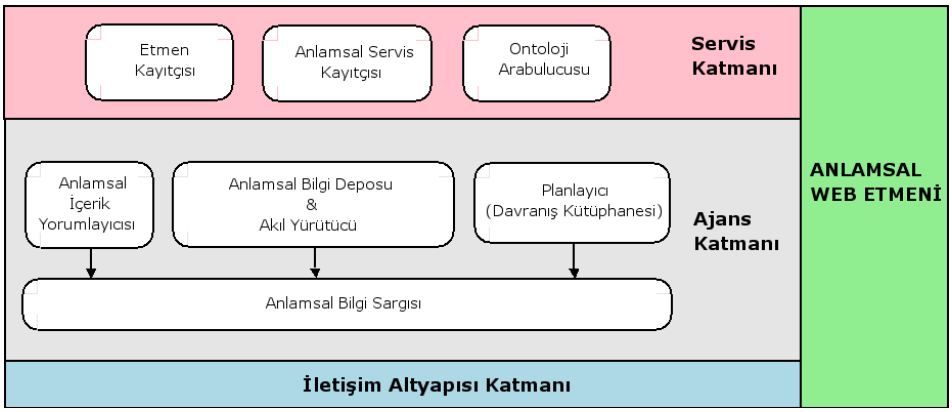
sağlayacak bir UML profili önermişlerdir. Anlamsal web servisleri için önerilen bu UML profili, OWL-S (OWL-S Coalition, 2004) ve WSMO (WSMO Group, 2005) gibi farklı anlamsal web servis ontolojilerine uyan anlamsal web servis dokümanlarının platform bağımsız bir modelden üretilebilmesini hedeflemektedir. Söz konusu bu UML profili ilgili amaç doğrultusunda tezin ilerleyen bölümlerinde de anlatıldığı gibi bu tez çalışmasında ortaya konan PIMM'de kullanılmıştır. Aynı bakış açısında Pahl (2007) MDD çerçevesinde web ontoloji dillerinin kullanımını sağlamak amacıyla katmanlı bir modelleme önermektedir. Önerilen modelleme, yazılım servislerinin oluşturulması için ontoloji dönüşümlerine dayanmaktadır. Adı geçen bu çalışmalar anlamsal web servislerinin model güdümlü olarak geliştirilmesi yönünde önemli katkılar sağlamış olsa da bu tezde direkt olarak desteklenen ve etmenlerin ve anlamsal web servislerinin entegrasyonunu içeren sistemlerin geliştirilmesini dikkate almamışlardır.

3 ANLAMSAL WEB ORTAMINDA ÇALIŞAN MAS'LAR İÇİN BİR YAZILIM MİMARİSİ

Uygulama modellerini oluşturan ve Anlamsal Web ortamında çalışan MAS'ların bileşenleri olan birinci sınıf varlıkların bu tip sistemlerin model güdümlü yaklaşımlarla geliştirilebilmeleri amacıyla tanımlanmış olmaları gerekmektedir. Bu varlıkların Anlamsal Web yetenekli MAS'lara ait bir kavramsal yazılım mimarisinden elde edilebilineceğine inanılmış ve bu tez kapsamında bir MAS yazılım mimarisi hazırlanmıştır. Özerk etmenler böyle bir mimariye uygun olarak geliştirilen bir MAS içerisinde anlamsal veriyi değerlendirebilecek ve içerik dilleri vasıtasıyla anlamsal web servisleri gibi anlamsal ortam elemanları ile etkileşimlerde bulunabileceklerdir.

Şekil 3.1'de resimlenen mimari, yazılım mimarilerini belgelemek için (Clements et al., 2003)'te yer alan bakış tiplerinden (*"viewtype"*) *Modül Bakış Tipi* (*"Module Viewtype"*) göz önüne alınarak hazırlanmıştır ve katmanlı bir yapıya sahiptir. Katmanlar arası ilişki *kullanıma izinli* (*"allowed-to-use"*) adı verilen ilişki gösterim tipindedir. Clements ve ark.'nın (Clements et al., 2003)'te verdikleri tanıma uygun olarak, önerilen mimaride, aralarında bu ilişki olan iki katmandan ilkinde yer alan herhangi bir modül ikinci katmanda yer alan herhangi bir modülü kullanma hakkına sahiptir. İlişkinin yönü aşağıya doğrudur. Bunun anlamı önerilen mimaride sadece üst seviye bir katman alt seviyedeki bir katmanın sunmuş olduğu servis veya hizmetleri kullanabilir. Tersine izin verilmemektedir. Öte yandan önerilen mimari herhangi bir *katman köprüleme*yi (*"layer bridging"*) (Clements et al., 2003) de içermemektedir. Buna göre bir üst katman sadece bir sonraki alt katmanın modüllerini kullanabilir. Mimariyi resimlemek için kullanılan gösterim Şekil 3.1'de görüldüğü üzere *yan eklentili katmanlar*ı (*"layers*

with a sidecar") (Clements et al., 2003) içermektedir. Katmanlar arası kullanıma izinli ilişkisi ise şekilde geometrik komşuluklarla temsil edilmektedir.



Şekil 3.1. Anlamsal Web ortamında çalışan MAS'lar için katmanlı bir mimari

Mimarinin öncül bir versiyonu ilk olarak (Kardas et al., 2006)'daki çalışmada tanıtılmıştır. Söz konusu versiyondaki katmanlar arası ilişki belirsizliği ve biçimsellik eksikliği daha sonra yerine getirilen çalışmalar sonucunda giderilmiştir ve mimari bu tezde anlatılan yapıyı kazanmıştır.

Önerilen mimarinin ana kısmında üç adet katman tanımlanmıştır: *Servis Katmanı*, *Ajans* (*"Agency"*) *Katmanı* ve *İletişim Altyapısı Katmanı*. Anlamsal Web Etmeni ise bu mimari katmanlarında bulunan tüm modülleri (bileşenleri) kullanma hakkına sahip mimarinin eklenti (*"sidecar"*) bileşenidir. Bu mimariye uygun olarak geliştirilen bir MAS'ta bir grup etmen Servis Katmanı'nda tanımlı servisleri sunar. Sistemdeki her etmenin ise Ajans Katmanı'nda tanımlı bir etmen iç yapısı bulunmaktadır. Sistemdeki etmenler birbirleri ile İletişim Altyapısı Katmanı'nda tanımlanmış olan protokollere uygun olarak haberleşirler.

Söz konusu bu mimari katmanları aşağıdaki alt bölümlerde detaylı olarak anlatılmaktadır.

3.1 Servis Katmanı

Servis Katmanı'nda bir MAS'ta yer alan anlamsal web etmenlerinin servisleri (ve/veya rolleri) tanımlanmaktadır. Servis Katmanı'ndaki tüm servisler Ajans Katmanı'nın sunmuş olduğu imkanları kullanırlar. İlgili etmen sisteminin iş alanına özgü etmen servisleri haricinde bu katmanda, etmen sistemine sağlanması gereken sarı sayfa ve arabulucu servisleri yer almaktadır.

Etmen Kayıtçısı, etmen platformunun diğer üyeleri için sistemde yer alan etmenlerin yeteneklerinin anlamsal olarak tanımlandığı ve ilan edildiği bir sistem servisidir. Görevlerinin işletimi sırasında platform etmenleri diğer etmenler tarafından sunulan servislere ihtiyaç duyabilir. Bu nedenle söz konusu bu servis üzerinde sorgular işletirler ve etkileşim için uygun etmenleri belirlerler.

Geleneksel bir MAS, sistemin üyesi olan etmenlerin uygun etmen servislerini bulabilmesi amacıyla sarı sayfa hizmeti sunan bir ya da daha fazla kayıtçıya sahiptir. Söz konusu bu kayıtçılar karmaşık sistem bileşenleri olup genellikle dizin servisleri olarak uygulamaya geçirilmektedirler ve bir platformun bir grup etmeni tarafından sunulmaktadırlar. Örneğin FIPA soyut mimarisi tanımında etmenlerin sunduğu servislerin kayıt olduğu, *Dizin Kolaylaştırıcısı* (DK) adı verilen ve her FIPA uyumlu MAS'ta olması gereken bir etmen çeşidi bulunmaktadır (FIPA, 2002). Bir etmen spesifik bir etmen servisini aradığında DK'dan servisi sağlayan etmenin bilgilerini (örneğin etmenin adı, adresi, vb.) elde etmekte ve görevini tamamlamak üzere ilgili servisi sağlayan etmenle iletişime geçmektedir.

FIPA uyumlu olsun ya da olmasın etkileşim içerisindeki etmenleri ihtiva eden bir MAS'ta yukarıda tarif edilen etmen kayıtçılarının olması gerekliliği açıktır. Ancak etmenlerin talep ettikleri ve sundukları servislerin (bir anlamda etmen yeteneklerinin) eşlenmesi işlemi Anlamsal Web ortamında çalışacak MAS'lar düşünüldüğünde daha karmaşık bir yapıya bürünmektedir ve yeniden tanımlamaya ihtiyaç duymaktadır. Bu tip MAS'larda etmen servislerinin keşfi için servis yeteneklerinin anlamsal eşlenmesine ait kriterlerin tanımlanması ve etmen servis tanımlarının kaydedilme mekanizmalarının (bir anlamda dizin servislerinin) bu kriterlere uygun olarak tasarlanması gerekmektedir. Böylelikle aranan servis özellikleri ve ilan edilen servislerin yetenekleri arasındaki eşleme işlemi sadece özdeş (*"identical"*) servis eşlemeyi göz önünde bulundurmayarak daha etkin bir hale gelmektedir. Buradaki özdeş servis eşleme ile kastedilen standart dizin servislerinde yer alan anahtar kelime bazlı servis arama ve eşleme işlemidir. Oysa yeni yetenek eşleme süreci aranan ve ilan edilen iki servis arasındaki ilişkinin tipini ve derecesini anlamsal olarak belirleyecektir ve bu da etmen ihtiyaçlarını karşılayan en uygun servislerin bulunmasını sağlayacaktır. Tüm bu vizyona uygun olarak tezde önerilen mimari etmen servisleri üzerinde yetenek eşlemesini yerine getirecek bir etmen kayıtçısını Servis Katmanı'nda tanımlamaktadır. Anlamsal yetenek eşlemesi ile ilgili daha detaylı bilgi ve FIPA uyumlu etmen sistemleri için örnek bir anlamsal yetenek eşleme mekanizması (Kardas et al, 2005)'te verilmiştir.

Öte yandan bir MAS'ta yer alan Anlamsal Web yetenekli yazılım etmenleri yerine getirmek istedikleri görevlerinin işletimi sırasında eğer ihtiyaç hissederlerse anlamsal web servisleri ile de etkileşime geçebilirler. Bölüm 2.1.2.1'de haklarında bilgi verilen bu tip servislere ait yeteneklerin de tıpkı etmen servisleri için olduğu gibi uygun kayıtçılarda tutulup ilan edilmesi gerekmektedir. Böylelikle bu servisler de etmenler

tarafından dinamik olarak keşfedilecek ve ihtiyaçları doğrultusunda çalıştırılabileceklerdir. Bu amaca uygun olarak mimaride *Anlamsal Servis Kayıtçısı* adı verilen bir Servis Katmanı bileşeni tanımlanmıştır.

Anlamsal Servis Kayıtçısı, ilgili platforma ait anlamsal web servislerinin ara yüzlerini bu servislerin etmenler tarafından keşfedilmesi amacıyla ilan eden bir servis eşleyici olarak modellenebilir. Örneğin OWL-S servisleri (OWL-S Coalition, 2004) göz önüne alınacak olursa, bir etmen ihtiyaç duyduğu anlamsal servisin yeteneklerini belirten OWL-S profilini bu kayıtçıya göndererek bu kayıtçı üzerinde ilgili sorgunun işletilmesini sağlar. Anlamsal Servis Kayıtçısı verilen ihtiyaç profili ve ilan ettiği servis profilleri arasında bir anlamsal yetenek eşleme işlemi gerçekleştirerek etmenin işine yarayacak uygun servisleri ilgili etmene bildirir. Etmen de uygun olan bu servis (ya da servislerle) anlaştıktan sonra görevini tamamlamak üzere ilgili servis (ya da servislerle) etkileşimde bulunabilir. Söz konusu anlaşma ve servis çalıştırma işlemi yine anlamsal olarak tanımlanmış etkileşim protokollerine uygun olarak yerine getirilmektedir. Anlamsal web servisleri için yetenek eşleme hakkında detaylı bilgiye ve alternatif eşleme uygulamalarına (Paolucci et al., 2002) ve (Li and Horrocks, 2003)'ten erişilebilir. Anlamsal web servislerinin MAS'lar bünyesinde kullanımı ve yukarıda sözü edilen Anlamsal Servis Kayıtçı'sının somut uygulamaları ise (Gümüs et al, 2007) ve (Gürcan et al., 2007)'de anlatılmıştır.

Hem Etmen Kayıtçısı hem de Anlamsal Servis Kayıtçısı bir MAS'ta servis eşleyici yerine birer aracı (*"broker"*) olarak da uygulamaya geçirilebilirler. Bu durumda ilgili kayıtçılar sadece servis yetenek eşlemesini gerçekleştirmezler; buna ek olarak servise ihtiyaç duyan etmen adına servisle bizzat etkileşime geçerler ve servis çalıştırma sonucunu servisi talep eden etmene yönlendirirler.

Bir anlamsal web etmeni farklı etmen organizasyonlarındaki etmenlerle etkileşimde bulunma ihtiyacı hissedebilir. Ayrıca bu etmenlerin ve anlamsal web servislerinin farklı bilgi depolarında ya da dağıtık sistemlerde yer alan bilgi kaynaklarını kullanması gerekebilir. MAS'lar gibi açık sistemlerde bu durumlar nedeniyle birden fazla ontoloji yer alabilir ve farklı sistem elemanları farklı ontolojileri kullanabilir. Bu nedenle farklı ontolojilerin kavram dönüşümlerini ve eşlemelerini yerine getirecek servislerin MAS'larda olması gerektiği düşünülmüştür. Şekil 3.1'de görüldüğü gibi kavramsal mimaride bu servisi sağlayan bileşene *Ontoloji Arabulucusu* adı verilmiştir. Bir MAS bünyesinde bir ya da daha fazla Ontoloji Arabulucusu yer alabilir.

Bir Ontoloji Arabulucusu aynı zamanda MAS iş alanına ait ve ilgili platform içerisindeki elemanlarca kullanılan ontolojiler için merkezi bir veri havuzu görevini üstlenebilir ve ontoloji yükleme, ontoloji güncelleme ve ontolojiler üzerinde sorgu gerçekleştirme gibi temel ontoloji yönetim işlemlerini yürütebilir. Ontoloji Arabulucusu servisini sağlayan bir etmen, ontoloji çevrim isteklerini özel bir kullanıcı ara yüzü vasıtası ile önceden tanımlanmış kavram eşleme bilgisine göre karşılamaktadır. Sağlanan bu ontoloji çevrim desteği ile bir etmen kendisinin üyesi olmadığı başka bir MAS'ta yer alan bir etmenle ya da kendi ortamı dışında yer alan bir servis ile farklı ontolojiler kullansalar bile iletişimde bulunabilir.

3.2 Ajans Katmanı

Mimarinin orta katmanı olan Ajans Katmanı anlamsal web etmenlerinin iç yapısını tanımlamaktadır. Özerk ve karşıt eylemli (*"reactive"*) yapıdaki etmenlerin amaçlarına uygun olarak dış ortam bileşenlerini nasıl kullanacağı ve davranışları için nasıl plan yapacağı bu iç yapıya göre belirlenmektedir.

Sistemdeki her etmenin yerel ontolojilerini sakladığı bir *Anlamsal Bilgi Deposu* bulunmaktadır. Bu ontolojiler etmen platformundaki diğer etmenler veya servisler ile etkileşime geçerken kullanılmaktadır. Ontolojilerin değerlendirilmesi ve temel çıkarsama ise *Akıl Yürütücü* modülü tarafından yerine getirilir.

Anlamsal İçerik Yorumlayıcısı etmen iletişimini kontrol eder. Bir anlamsal web etmeni iletişimleri sırasında diğer etmenlerden veya anlamsal servislerden doğal olarak mesajlar alacaktır. Alınan mesaj içeriğinin anlamsal uygunluğunun kontrol edilmesine ve içeriğin etmenin inançlarına ve niyetlerine uygun bir şekilde yorumlanmasına ihtiyaç vardır. Anlamsal İçerik Yorumlayıcısı söz konusu bu içerik uygunluğunun kontrolünü ve yorumlamayı gerçekleştirmektedir.

Ajans Katmanı'nın *Planlayıcı* adı verilen modülü ihtiyaç duyulan yeniden kullanılabilir etmen planlarını ve ilgili davranış kütüphanelerini içermektedir. Yeniden kullanılabilir etmen planları bir etmenin niyetlerine uygun olarak işletilen görevlerinin birleşiminden oluşmaktadır. Planlayıcı, örneğin HTN (*"Hierarchical Task Network"*) (Williamson et al., 1996) gibi bir karşıt eylemli planlama paradigmasını temel almaktadır. Dickinson ve Wooldridge'in (Dickinson and Wooldridge, 2005)'de belirttiği gibi karşıt eylemli planlama için bir etmene önceden tanımlı (belki de sistemin derlenme zamanında tanımlanmış) genel planlardan oluşan bir kütüphane sağlanmaktadır. Etmen ortamdan elde ettiği algılara tepki olarak bu planların birini ya da birkaçını uygulamaya geçirir.

Ajans Katmanı içerisinde bulunan *Anlamsal Bilgi Sargısı* (*"Wrapper"*) yukarıda sözü edilen ontolojilerin Ajans Katmanı'nın üst seviye bileşenleri tarafından kullanılabilmesini sağlamaktadır. Örneğin görev işletimi sırasında bir etmen bir ontoloji varlığının nesne (ya da başka bir programlanabilir yapı) gösterimine ihtiyaç duyabilir. Anlamsal

İçerik Yorumlayıcısı da etmen ontolojileri üzerinde sorgu işleterek bir konu hakkında çıkarsamada bulunmak isteyebilir. Bu tip ihtiyaçları gidermek amacıyla Anlamsal Bilgi Sargısı, Ajans Katmanı'nın çalışma zamanı ortamı içerisinde ilgili ontolojilerin çizge gösterimlerini oluşturabilir. Etmen iç mimarisinde böyle bir sargının kullanılmasına dair bir örnek JENA[2] çerçevesine (JENA, 2003) dayalıdır ve (Dikenelli et al., 2006)'da anlatılmıştır.

3.3 İletişim Altyapısı Katmanı

Mimarinin en alt katmanı mimarinin iletişim altyapısı uygulamasının soyutlanmasından sorumludur. Mimarinin somut bir örneğinde bu katman FIPA Etmen İletişim ve Etmen Mesaj Taşıma protokollerinin bir uygulaması olabilir. Böylece altyapı FIPA Etmen İletişlim Dili'nin ve ilgili protokolün kullanılması ile etmenler arası mesaj transferini gerçekleştirmiş olur. Fiziksel iletim örneğin HTTP-IIOP (*"HTTP – Internet Inter-ORB Protocol"*) üzerinden gerçekleştirilebilir. Burada asıl önem verilmesi gereken mesaj altyapısında kullanılan içerik dilinin yapısı ve zenginliğidir.

[2] Hewlett-Packard Geliştirme Şirketi'nin bir ürünü olan JENA, Anlamsal Web uygulamalarının geliştirilmesini sağlayan ve Java programlama dili kullanılarak hazırlanmış açık kaynak kodlu bir yazılım çerçevesidir. Ontolojilerin kullanılması için programlanabilir bir ortam sağlamaktadır ve kural tabanlı bir çıkarsama motoru içermektedir.

4 ANLAMSAL WEB ORTAMINDA ÇALIŞAN MAS'LAR İÇİN PLATFORM BAĞIMSIZ BİR ÜSTMODEL

Anlamsal Web yetenekli etmen sistemlerinin model güdümlü olarak geliştirilmesi için platform bağımsız bir üstmodele ihtiyaç vardır. Tezde, elemanları ve bunlar arasındaki ilişkileri bir önceki bölümde (Bölüm 3) tanıtılan MAS yazılım mimarisine bağlı olarak tanımlanmış bir MAS üstmodeli geliştirilmiştir. Söz konusu üstmodel Anlamsal Web yetenekli MAS'lar için bir PIMM'dir. Bölüm 2.1.3'te belirtildiği gibi bu PIMM kullanılarak etmen sistemlerinin platform bağımsız tasarımları gerçekleştirilmekte ve daha sonra bu model üzerine uygulanan model dönüşümleri sonrasında da gerçek MAS çerçevelerinde bu tip etmen sistemleri hayata geçirilmektedir.

PIMM ortaya çıkarılırken önce ihtiyaçları karşılayacak çekirdek bir üstmodel oluşturulmuştur. Daha sonra bu üstmodele, modelin MDD'de uygun bir şekilde kullanılmasını sağlayacak yeni varlıklar ve bu varlıklar arasındaki ilişkiler eklenmiştir. PIMM'in son hali Bölüm 2.2.1.3'de anlatılan FIPA ACSM'nin (FIPA Modeling TC, 2004) bir uzantısıdır. PIMM aynı zamanda UML 2.0 Üstyapısı'nı (OMG, 2004) ve Djuric'in (Djuric, 2004)'de tanımladığı Ontoloji UML Profili'ni genişleterek özellikle Anlamsal Web yapıları ile etmenlerin etkileşimlerini modelleyen yeni üstvarlıklar (*"meta-entity"*) ve ilişkiler tanımlamaktadır.

Tezde önerilen model güdümlü geliştirme sürecinde PIMM olarak kullanılan bu üstmodelin çekirdek yapısı ve yukarıda bahsedilen uzantılarla şekillenmiş son versiyonu sırasıyla bölüm 4.1 ve 4.2'de anlatılmaktadır. Bölüm 4.3'te ise bu üstmodelin bir örneği (*"instance"*) olan ve tezin daha sonraki bölümlerinde bahsedilen model dönüşümlerinde kullanılacak bir MAS modeli anlatılmaktadır.

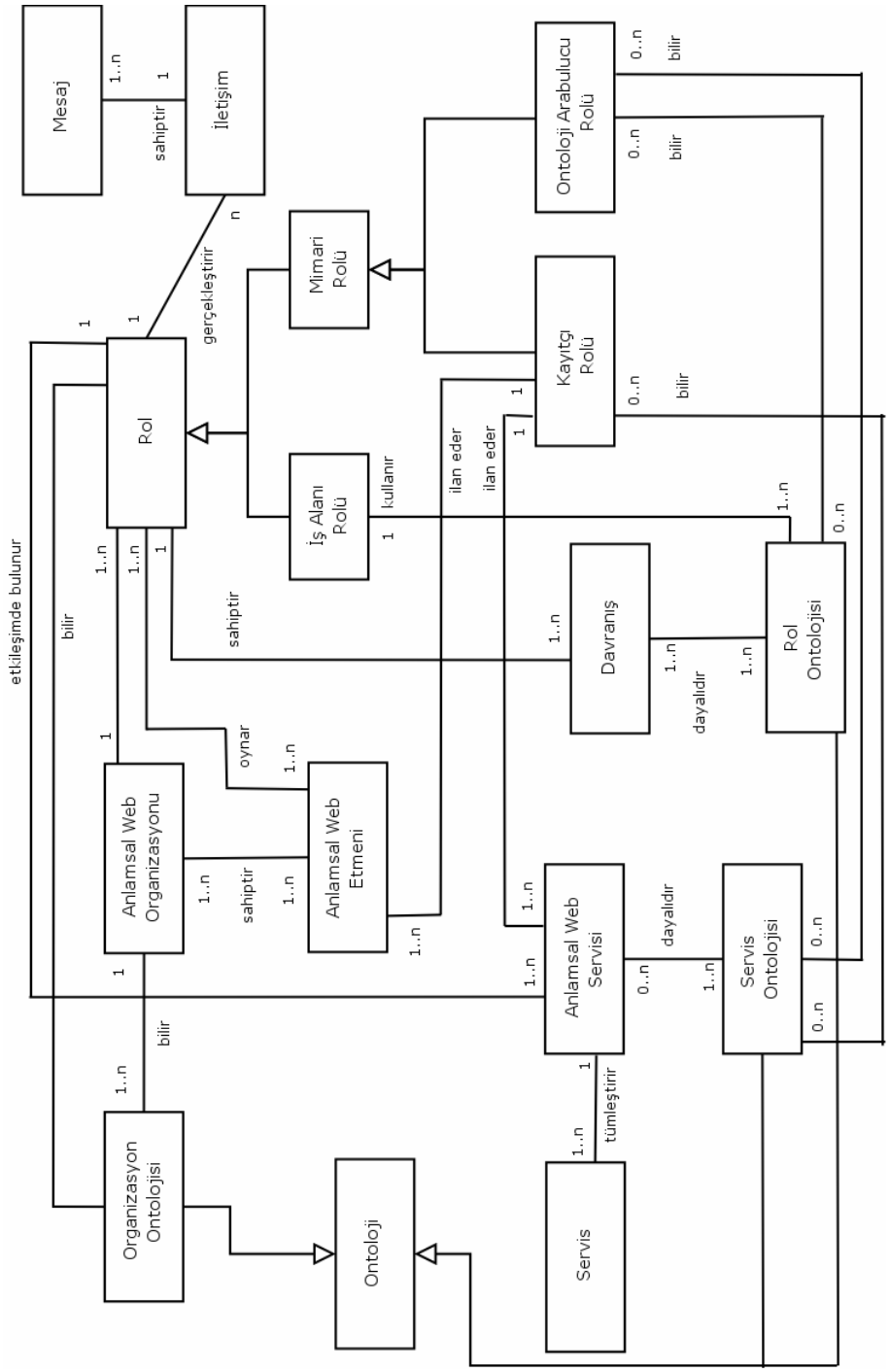
4.1 Çekirdek MAS Üstmodeli

Anlamsal Web ortamında çalışan MAS elemanlarının ve aralarındaki ilişkilerin temsil edildiği çekirdek üstmodel Şekil 4.1'de verilmiştir. Bu üstmodel ilk olarak (Kardas et al., 2006)'daki çalışmada tanıtılmıştır.

Bir *Anlamsal Web Organizasyonu*, *Anlamsal Web Etmenleri*'nin organizasyonel rollerine dayalı olarak oluşturdukları bir yapıdır. Bir *Anlamsal Web Etmeni* ise Bölüm 3'te de anlatıldığı gibi ortamdaki hem diğer etmenler ile hem de anlamsal web servisleri ile etkileşime geçme yeteneğine sahip özerk bir varlık olarak tanımlanmaktadır. Etkileşim süreçleri, önceden tanımlı anlamsal iletişim protokollerine göre işletilmektedir.

Burada dikkat edilmesi gereken organizasyonun sadece etmenlerden oluştuğudur. Etmen harici anlamsal yapıları içermez. Ancak organizasyonel rolleri içerir ve bu roller üye etmenler tarafından oynanırlar. Organizasyonel roller göz önüne alındığında bir Anlamsal Web Etmeni aynı anda birden fazla organizasyonun üyesi olabilir. Bu demektir ki ***bir etmen aynı anda birden fazla Rolü oynayabilir ve bir Rol, Anlamsal Web Organizasyonu içeriğinde ("context") birden fazla Anlamsal Web Etmeni tarafından oynanabilir***.

*Rol*ler etmenlerin etkileşimde bulunduğu etmen sosyal sistemlerine ait yapıtaşlarının ve ihtiyaçların temsil edilmesini sağlamaktadırlar (Odell et al., 2005). Anlamsal Web yetenekli etmen ortamlarında oynanan roller için de aynısı geçerlidir. Ancak oldukça genel bir anlama sahip bu üstmodel varlığının mimari ve iş alanı tabanlı rollerin görev tanımlarına bağlı olarak model içerisinde özelleşmesinin gerektiğine inanılmış ve üstmodelde bu varlığın iki alt varlığı da yer almıştır: *Mimari Rolü* ve *İş Alanı Rolü*.



Şekil 4.1. Çekirdek MAS Üstmodeli

Bir Mimari Rolü, organizasyonun içeriğinden bağımsız olarak platform içerisinde en az bir etmen tarafından üstlenilmesi gereken anlamsal web yetenekli çok-etmenli sistemler için zorunlu bir rolü tanımlamaktadır. Buna karşılık İş Alanı Rolü etmen organizasyonu içeriğine özgü olup tamamıyla spesifik bir iş alanı için oluşturulmuş bir Anlamsal Web Organizasyonu'nun ihtiyaçlarına ve görev tanımlamalarına dayalıdır.

Organizasyonda yer alan bir kısım etmenin Bölüm 3'te tanıtılan kavramsal mimarinin Servis Katmanı'nda tanımlanmış olan servisleri sağlamak amacıyla ortaya konan rolleri oynaması gerekir. Bu nedenle üstmodelde Mimari Rolü'nün iki özelleşmiş alt model elemanı da yer almaktadır: *Kayıtçı Rolü* ve *Ontoloji Arabulucu Rolü*. Kayıtçı Rolleri organizasyondaki etmenler için sarı sayfa servisi sunan bir ya da daha fazla Anlamsal Web Etmeni tarafından oynanmaktadır. Üstlendiği rolün kapsamına göre söz konusu bu etmenler anlamsal web etmenlerinin ya da anlamsal web servislerin yetenek ilanlarını saklayabilirler. Böylelikle, diğer etmenler de bu kayıtçı rolünü oynayan etmenlerle etkileşime geçerek ihtiyaç duydukları servisleri sağlayan uygun sistem elemanlarını bulabilirler. Öte yandan üstmodelde yer alan Ontoloji Arabulucu Rolü Bölüm 3'te de anlatıldığı gibi ontoloji arabulucu etmenlerinin sağlaması gereken ontoloji yönetimi işlevselliğini tanımlamaktadır. Ontoloji Arabulucu Rolü'nü oynayan bir anlamsal web etmeni üyesi bulunduğu anlamsal web organizasyonunun tüm ontoloji bilgi tabanından haberdar olmak zorundadır.

Bir Rol bir ya da daha fazla *Davranış*'ı içermektedir. Bir anlamsal web etmeninin görev tanımları ve ilgili görev işletim süreçleri Davranış varlıkları içerisinde modellenmektedir. Etmen rolleri de bu davranışları içermektedir. Örneğin kullanıcısı adına uygun bir otel odası rezervasyonu gerçekleştirmek isteyen bir etmen oynadığı rol gereği ortamda uygun

rezervasyon servisini bulma, servisle anlaşmaya varma ve servisi çalıştırma gibi görevleri yerine getirecek davranışlar gösterir.

Oynanan rollere bağlı olarak etmenlerin çoğu zaman başka bir etmen ile iletişime geçmesi gerekmektedir. Üstmodeldeki *İletişim* varlığı platformdaki iki etmen arasındaki bir etkileşimi tanımlamaktadır. Etkileşim önceden tanımlı etmen etkileşim protokolüne dayalı olarak gerçekleşir. Bir İletişim varlığı bir ya da daha fazla *Mesaj* varlığını içermektedir. Her bir mesajın içeriği RDF (W3C, 2004) tabanlı bir anlamsal içerik dili ile ifade edilebilmektedir.

Çekirdek üstmodelin önemli birinci sınıf elemanlarından biri *Anlamsal Web Servisi*'dir. Bir Anlamsal Web Servisi üstvarlığı etmen servisi hariç olmakla birlikte yetenekleri ve etkileşimleri mevcut sistem içerisinde anlamsal olarak tanımlanmış herhangi bir servisi temsil etmektedir. Bir Anlamsal Web Servisi bir ya da daha fazla *Servis* varlığını içermektedir. Her servis bir web servisi ya da gerçek hayat uygulamasında önceden tanımlı başka özel bir etkileşim protokolüne sahip bir servis olabilir. Ancak mutlaka ilgili servisin platformun etmenleri tarafından kullanılabilecek bir anlamsal arayüzünün olması gerekmektedir. Burada dikkat edilmesi gereken modelde *Anlamsal Web Etmenleri ile Anlamsal Web Servisleri arasındaki ilişkinin Rol varlığı üzerinden sağlanmış olması*dır. Çünkü ***etmenler organizasyon içerisinde tanımlanmış olan rollerine dayalı olarak anlamsal web servisleri ile etkileşimde bulunurlar***.

Diğer Anlamsal Web ortamları gibi anlamsal web yetenekli çok-etmenli sistemler de ontolojiler olmaksızın düşünülemezler. Bu nedenle önerilen üstmodel bir *Ontoloji* varlığı ve onun ihtiyaç duyulan alt-varlıklarını içermektedir. Bir Ontoloji bir MAS üyesi için herhangi bir bilgi toplama ve akıl yürütme kaynağını temsil eder. Ontolojilerin

koleksiyonu iş alanı muhteviyatını sağlayan çok-etmenli sistem bilgi tabanını oluşturmaktadır.

Ontoloji üstvarlığının *Organizasyon Ontolojisi*, *Servis Ontolojisi* ve *Rol Ontolojisi* adı verilen alt varlıkları ilgili üstmodel varlıkları tarafından kullanılmaktadır. Örneğin servislerin anlamsal arayüzleri ve yetenek tanımları Servis Ontolojisi'ne bağlı olarak oluşturulmaktadır ve bu ontoloji anlamsal web etmenleri tarafından anlamsal web servislerinin bulunması ve çalıştırılması için kullanılmaktadır.

4.2 Genişletilmiş MAS Üstmodeli

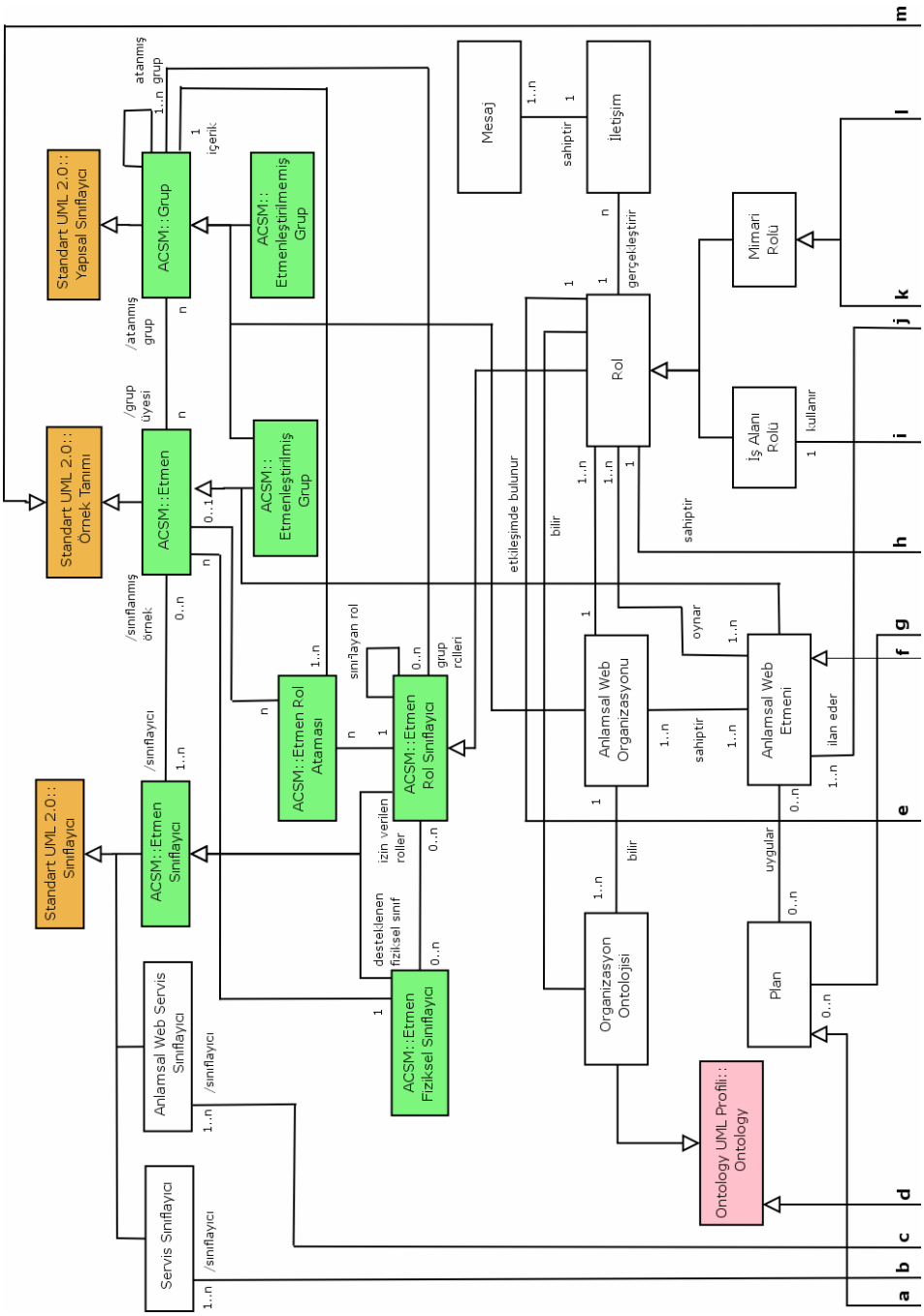
Bir önceki bölümde anlatılan MAS üstmodeli üzerinde çalışıldıkça modelin MDD'de uygun bir şekilde kullanılmasını sağlayacak yeni varlıkların ve ilişkilerin eklenmesi ihtiyacı doğmuştur. Çekirdek üstmodelin özellikle farklı MAS platformları için model dönüşümlerinde kullanılması sırasında görülen soyutlama eksikliklerinin giderilmesine çalışılmıştır. Ayrıca belirgin olmayan biçiminin UML 2.0 Üstyapısı'na dayandırılarak farklı yazılım araçlarınca desteklenmesi sağlanmıştır.

Genişletilmiş üstmodel tezde önerilen MDD sürecinde kullanılan Anlamsal Web yetenekli etmen sistemleri için bir PIMM'dir. Bu PIMM Bölüm 2.2.1.3'de anlatılan FIPA ACSM'nin (FIPA Modeling TC, 2004) bir uzantısıdır. FIPA ACSM'nin etmenleri, rollerini ve gruplarını modellemek amacıyla ihtiyaç duyulan kullanıcı seviyesi yapıları tanımlayan uygun bir üstyapı olduğu görülmüş ve üstmodelin FIPA ACSM'de tanımlanan varlıklar ve bu varlıklar arası ilişkileri miras edinmesi sağlanmıştır. Bu miras edinmenin sağladığı iki önemli fayda şunlardır:

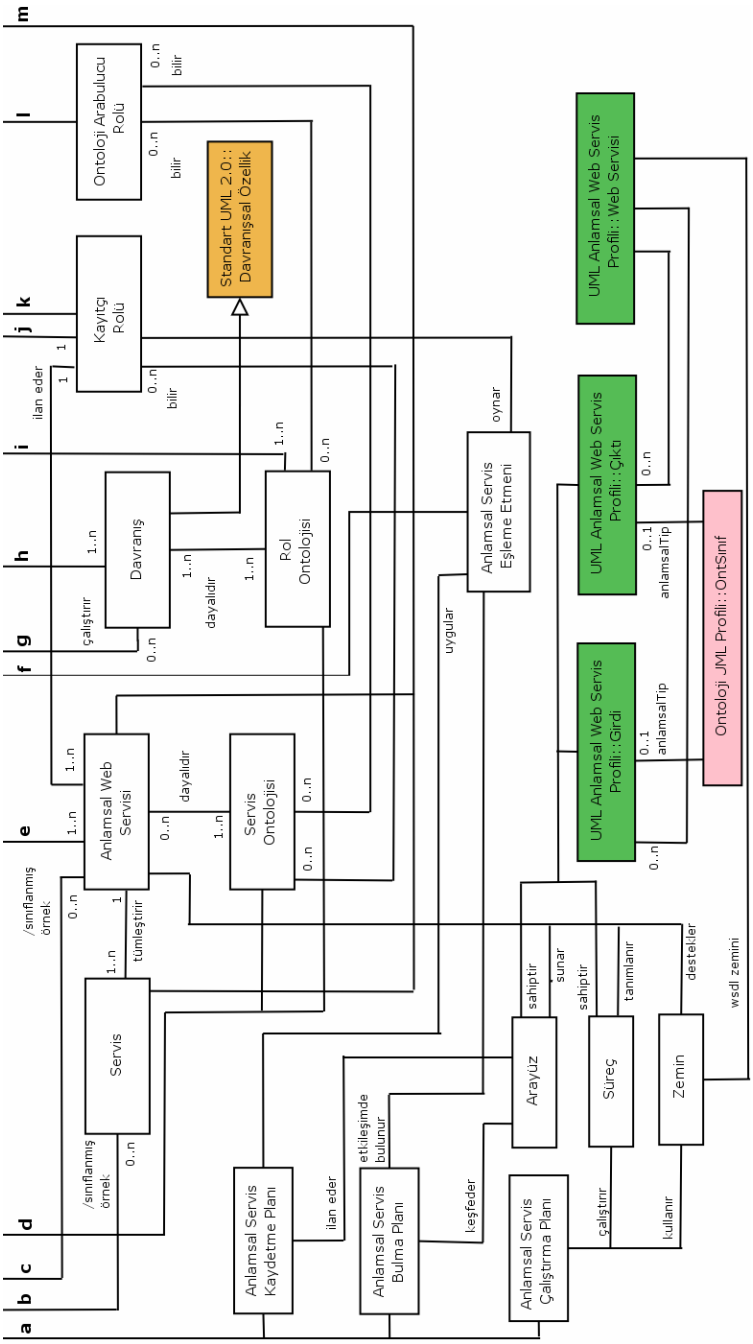
- Geleneksel etmen alanına ait temel varlıklar ve ilişkileri FIPA ACSM'de tanımlanmış olduğundan söz konusu PIMM üzerinde yeniden tanımlanmasına gerek kalmamıştır.
- FIPA ACSM etmenlere rol atama işlemini etmen gruplarının içeriğini de göz önüne alarak modellemektedir. Bu nedenle çekirdek üstmodeldeki Anlamsal Web Organizasyonu, Anlamsal Web Etmeni ve Rol kavramları arasındaki nispeten bulanık kalan ilişkilerin FIPA ACSM'nin Etmen Rol Ataması üstvarlığının çekirdek modele eklenmesi ile netleştirilmesi sağlanmıştır.

Ortaya çıkan yeni PIMM sadece FIPA ACSM'yi değil aynı zamanda UML 2.0 Üstyapısını da uzatmaktadır. Ayrıca Djuric'in (Djuric, 2004)'de tanıttığı Ontoloji UML Profili kullanılarak *Ontoloji* kavramının modelde belirginleştirilmesi yerine getirilmiştir. Genişletilmiş PIMM ilk olarak (Kardas et al., 2007a)'daki çalışmada tanıtılmıştır. Sonradan beliren ihtiyaçlar doğrultusunda kendisine yeni üstvarlıklar eklenmiş ve ilişkileri revize edilmiş PIMM'in son versiyonu ise burada anlatılmaktadır.

40'tan fazla üstvarlığı ve bunların arasındaki 75'ten fazla ilişkiyi içeren PIMM yer kısıtlarından ötürü tezde iki parçaya ayrılarak resmedilmiştir (Şekil 4.2a ve 4.2b). Şekil 4.2a'da koyu harflerle tanımlanmış varlık ilişkilerinin devamı Şekil 4.2b'de yine aynı harflere karşılık gelen ilişki çizgileri ile gösterilmiştir. Bölüm 4.1'de anlatılan çekirdek modelden gelen, yeni eklenen ve diğer üstyapıları (UML 2.0 Üstyapısı, FIPA ACSM ve Ontoloji UML Profili) uzatan üstmodel varlıkları renksiz olarak resmedilirken diğer üstmodellerden miras alınmış varlıklar renklendirilmiştir.



Şekil 4.2a. Anlamsal Web yetenekli MAS'lara ait PIMM (Bölüm 1)



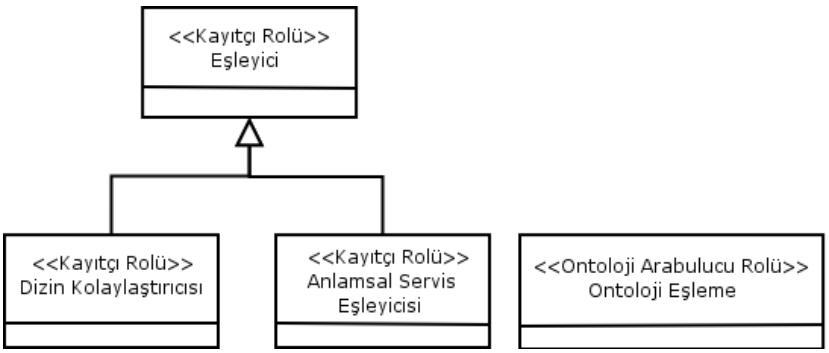
Şekil 4.2b. Anlamsal Web yetenekli MAS'lara ait PIMM (Bölüm 2)

Önceki çekirdek modelde tek başına yer alan *Anlamsal Web Etmeni* yeni modelde FIPA ACSM Etmen sınıfının bir alt sınıfı olarak yer almaktadır. Önceki üstmodeldeki *Rol* kavramı yeni modelde FIPA ACSM Etmen Rol Sınıflayıcı'sının bir uzantısı olarak tasarlanmıştır (Şekil 4.2a). Böylelikle Anlamsal Web Etmeni – Rol ilişkisi FIPA ACSM'deki Etmen – Etmen Rol Sınıflayıcı ilişkisine uygun hale getirilmiştir. Bu ilişkiye bağlı olarak ***Anlamsal Web Etmenleri aynı zamanda birden fazla Rol ile ilişkilendirilebilir (çoklu sınıflama) ve zamanla rollerini değiştirebilirler (dinamik sınıflama)***.

Etmen Rol Sınıflayıcıları bir üstsınıf – altsınıf ayrımına dayalı bir genelleme hiyerarşisi oluşturmaktadırlar. Anlamsal ortamın Rol elemanları için de bu hiyerarşi geçerlidir. Örneğin Şekil 4.3'te SEAGENT MAS çerçevesindeki (Dikenelli et al., 2005) mimari rollerinin tezde önerilen PIMM'e uygun bir hiyerarşisi verilmiştir. SEAGENT, FIPA uyumlu bir MAS geliştirme çerçevesi olduğundan dolayı bünyesinde Dizin Kolaylaştırıcısı adı verilen bir Kayıtçı Rolü tanımlamaktadır. Ayrıca etmenler ile anlamsal web servisleri arasındaki etkileşimin gerçekleştirilmesi amacıyla platform etmenlerinin bir kısmının oynaması gereken Anlamsal Servis Eşleme adlı rolü de içermektedir. Ontoloji Eşleme adlı SEAGENT rolü ise Ontoloji Arabulucu Rolü'nün bir örneğidir.

Etmen organizasyonları bünyelerinde anlamsal web etmenlerini barındırdıklarından dolayı *Anlamsal Web Organizasyonu* üstvarlığı yeni modelde FIPA ACSM'nin Grup elemanının bir uzantısı olacak şekilde yer almıştır (Şekil 4.2a). Ancak burada dikkat edilmesi gereken bir ayrıntı vardır: Bir Anlamsal Web Organizasyonu toplamda bir anlamsal web etmeni gibi davranabilir ya da davranmayabilir. Bu nedenle sadece bir Etmenleştirilmiş Grup ya da Etmenleştirilmemiş Grup olarak

tanımlanmamıştır. Bunların yerine Grup bütünleşik yapısının doğrudan bir uzantısı olarak modelde yer almaktadır.



Şekil 4.3. Bir SEAGENT MAS'ındaki mimari rollerinin hiyerarşisi

Yukarıda yer alan FIPA ACSM uzantıları Anlamsal Web Etmeni, Rol ve Anlamsal Web Organizasyonu arasında daha önceden bahsedilen bulanık ilişkileri FIPA ACSM'nin Etmen, Etmen Rol Sınıflayıcı ve Grup elemanları arasındaki üçlü Etmen Rol Ataması ilişkisinin uygulanması ile ortadan kaldırmaya yardımcı olmaktadırlar.

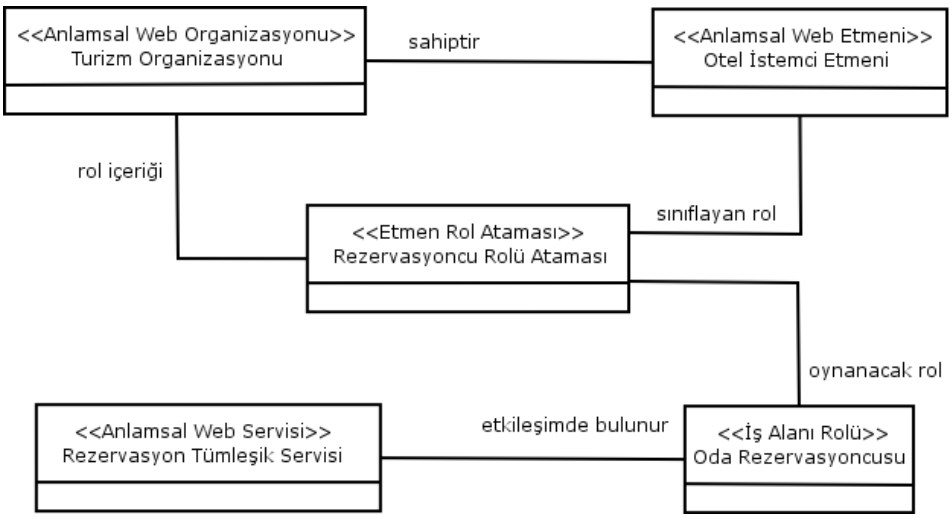
Etmenler gibi anlamsal web varlıklarının da nesne yönelimli paradigmaya uymayacak yetenekleri ve özellikleri bulunmaktadır. Bu nedenle PIMM, Anlamsal Web ortamına ait üst elemanları tanımlamak amacıyla UML 2.0 Üstyapısı'nı genişleten yeni yapılar tanımlamaktadır. Örneğin çekirdek modeldeki *Anlamsal Web Servisi* üstvarlığı modele dahil edilirken öncesinde uygun bir Sınıflayıcı tanımlanmıştır (*Anlamsal Web Servis Sınıflayıcı*). UML 2.0 Üstyapısı'nın soyut bir üstsınıfı olan Sınıflayıcı, sınıf örneklerinin (*"instance"*) içerdikleri özelliklere göre sınıflandırılmasını sağlar (OMG, 2004). Genelleştirme özelliğinin Sınıflayıcı somut alt tiplerinin her birini nasıl etkilediğine dair anlam her alt tipe göre değişkenlik göstermektedir (OMG, 2004). FIPA ACSM'nin UML 2.0 Üstyapsı'nın Sınıflayıcı ve Sınıf ilişkisini etmen ortamları için

nasıl adapte ettiği Bölüm 2.2.1.3'te anlatılmıştı. Benzer bir soyutlamanın anlamsal web servisleri için de PIMM'de yapılması için yukarıda sözü edilen Anlamsal Web Servis Sınıflayıcı adlı UML 2.0 Sınıflayıcı uzantısı tanımlanmıştır. Böylelikle Anlamsal Web Servis üstvarlığı da modelde bu sınıflayıcının "sınıflayıcı örneği" olarak yer almaktadır (Şekil 4.2a ve Şekil 4.2b'deki "c" etiketli ilişki). Aynı ilişki *Servis Sınıflayıcı* ve *Servis* varlıkları arasında da bulunmaktadır (Şekil 4.2a ve Şekil 4.2b'deki "b" etiketli ilişki).

Şekil 4.4'te Turizm iş alanında faaliyet gösteren bir çok-etmenli sistem için rol atamanın gerçekleştirildiği örnek model yer almaktadır. Örnek modelde "Otel İstemci Etmeni" bir Anlamsal Web Etmeni örneğidir ve kullanıcıları adına otel odası rezervasyonu gerçekleştirmeye çalışmaktadır. Bu etmenin üyesi olduğu Anlamsal Web Organizasyonu ise "Turizm Organizasyonu" adını taşımaktadır. Bu organizasyon bünyesinde tanımlı olan ve adı "Oda Rezervasyoncusu" olan bir İş Alanı Rolü vardır. Bu rolün ilgili etmene atanması ise bir Etmen Rol Ataması örneği olan "Rezervasyoncu Rolü Ataması" üzerinden gerçekleşmektedir. "Rezervasyoncu Rolü Ataması" adlı sınıf "Otel İstemci Etmeni", "Oda Rezervasyoncusu" ve "Turizm Organizasyonu" arasındaki üçlü ilişkiyi temsil etmektedir. Oda Rezervasyoncusu rolünü oynayacak olan etmenler bir Anlamsal Web Servis olan "Rezervasyon Tümleşik Servisi" ile etkileşimde bulunurlar. Bu etkileşim sonucunda etmenler uygun otelde uygun rezervasyonları kullanıcıları adına gerçekleştirebilirler. İlgili anlamsal web servisi bütünleşik bir servis arayüzüne sahip olup otel odası rezervasyonu için keşif, anlaşma ve çalıştırma alt-servislerinin bir bileşimi olabilir.

Üstmodelde ontoloji varlıkları (*Organizasyon Ontolojisi*, *Rol Ontolojisi* ve *Servis Ontolojisi*), (Djuric, 2004)'te tanımlanan Ontoloji UML Profili'nin *Ontoloji* elemanının uzantıları olarak yer almaktadırlar

(Şekil 4.2a ve Şekil 4.2b'deki "d" etiketli ilişki). Ontoloji UML Profili ontoloji kavramlarını, özelliklerini ve kavramlar arası ilişkilerini modellemekte ve modellerde anlamsal tip olarak kullanılabilecek bir dizi UML elemanını sağlamaktadır. Anlamsal kavramların Ontoloji UML Profili'ne dayalı olarak türetilmesiyle bu amaca yönelik üstmodelde önceden tanımlı UML elemanlarının kullanılması mümkün olmuştur.



Şekil 4.4. Turizm iş alanında faaliyet gösteren bir çok-etmenli sistem için rol atamanın gerçekleştirildiği örnek model

Davranış varlığı üstmodelde UML 2.0 *Davranışsal Özelliği*'nin bir uzantısı olarak tanımlanmıştır (Şekil 4.2b). UML 2.0 Özellikleri ("Feature"), sınıf ya da sınıflayıcıların davranışsal veya yapısal özelliklerini tanımlarlar. Bir davranışsal özellik bir model elemanının dinamik bir özelliğine (örneğin bir işlem ya da bir metot) işaret ederken yapısal bir özellik ise bir model elemanının statik özelliklerine işaret eder. Buna göre PIMM'de de etmen davranışları UML 2.0'ın davranışsal özelliğinin uzantıları olarak modellenmiştir çünkü bir davranış bir

anlamsal web etmeninin dinamik bir özelliğidir (örneğin bir anlamsal web etmeninin başka etmenlerle etkileşimini gerçekleştiren bir görev).

Sistem geliştirme sırasında model dönüşümlerinin gerçekleştirilebilmesi için anlamsal etmenler ve dış servisler arasındaki etkileşimin varlık bakış açısından detaylandırılması gerekmektedir. Bu amaca uygun olarak PIMM yeni üstvarlıkları ve varlık ilişkilerini içermektedir.

Anlamsal web servisi modelleme dillerinde (örneğin OWL-S (OWL-S Coalition, 2004)) çoğunlukla servisler üç anlamsal doküman tarafından temsil edilmektedir: Servis Arayüzü, Süreç Modeli ve Fiziksel Zemin. Servis Arayüzü servislerin girdi, çıktı ve diğer gerekli servis tanımlarının listelendiği, ilgili servise ait bir yetenek temsilidir. Süreç Modeli ise servisin içsel bütününü ve çalıştırma dinamiklerini tanımlamaktadır. Fiziksel Altyapı ise web servisinin çağrılma protokolünü tanımlamaktadır. Bu anlamsal web servisi bileşenleri modelde *Arayüz*, *Süreç* ve *Zemin* üstvarlıkları ile verilmiştir (Şekil 4.2b'de sol alt kısım). Bu bileşenlerin kullandığı anlamsal girdi, çıktı ve web servisi tanımlamaları ise (Grønmo et al., 2005)'de tanıtılan UML Anlamsal Web Servis Profili'nden ithal edilmiştir. Bu profil anlamsal web servis dokümanlarının platform bağımsız bir modelden üretilebilmesini hedeflemektedir. Profile göre her anlamsal girdi ve çıktının tipi iş alanına özgü bir Ontoloji sınıfıdır. Bu ontoloji sınıfları Ontoloji UML Profili'nin (Djuric, 2004) *OntSınıf* üstvarlığı kullanılarak UML Anlamsal Web Servis Profili'nde gösterilmektedir. Bu nedenle UML 2.0'a dayanan PIMM de OntSınıf üstvarlığını bünyesinde bulundurmaktadır (Şekil 4.2b'de sağ alt kısım).

Anlamsal web etmenleri üstlendikleri görevleri yerine getirmek amacıyla *Plan*'lar uygulamaktadırlar. PIMM'de anlamsal web servislerini dinamik olarak keşfetmeye ve çalıştırmaya yönelik olarak Plan

üstvarlığının iki uzantısı da (Şekil 4.2a ve Şekil 4.2b'deki "a" etiketli ilişki) ayrıca tanımlanmıştır: *Anlamsal Servis Bulma Planı* ve *Anlamsal Servis Çalıştırma Planı*. Anlamsal web etmenlerinin anlamsal web servisleri ile etkileşime geçmek için ardışık olarak çalıştırdıkları bu iki plandan birincisinde etmenin isteklerine cevap verebilecek aday servislerin bulunması yerine getirilmektedir. Bu plan işletilirken etmen ilgili platformun servis eşleyicisi ile iletişime geçer. İkinci planda ise etmen uygun servisin süreç modelini ve fiziksel bağlantı altyapısını kullanarak servisi çalıştırır.

Yukarıda tanımlanan anlamsal servis çalıştırma mekanizmasında uygun servislerin bulunması amacıyla etmenlerin bir servis kayıtçısı ile iletişime geçmesi gerekliliği açıktır. Bu nedenle PIMM, özelleşmiş bir anlamsal web etmeni olan *Anlamsal Servis Eşleme Etmeni*'ni içermektedir (Şekil 4.2b). Bu üstvarlık bir MAS içerisinde anlamsal web servislerinin yetenek ilanlarını depolayan ve diğer etmenlerden gelen servis isteklerini ilan edilen servis yetenekleri ile karşılaştırıp uygun servis ya da servisleri belirleyen eşleme etmenlerini modelde temsil etmektedir. Bu etmenlerin uyguladıkları planlar ise PIMM'de yine bir Plan altsınıfı olan *Anlamsal Servis Kaydetme Planı* ile gösterilmiştir.

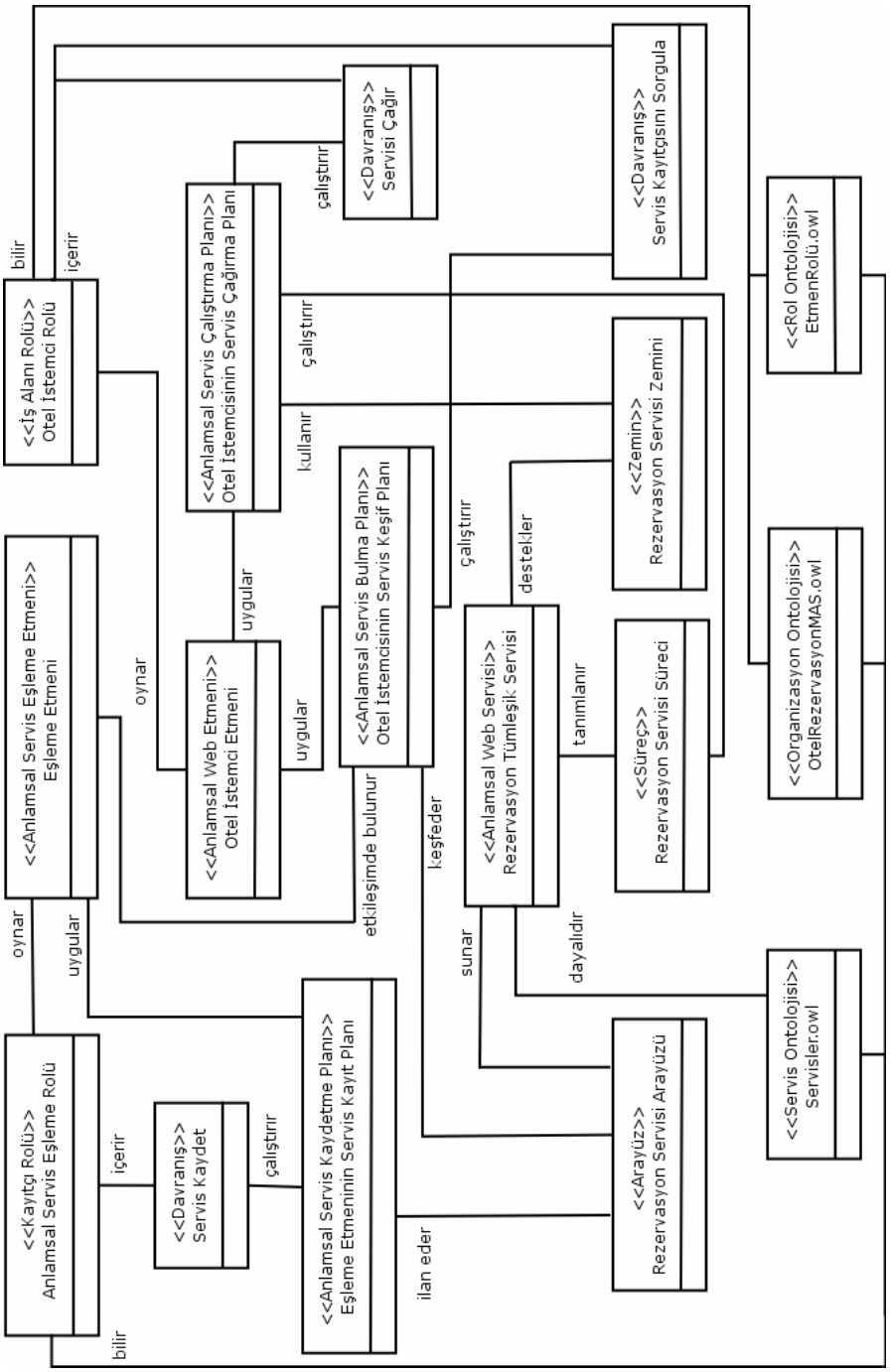
4.3 Turizm İş Alanında Çalışan MAS'lar için Platform Bağımsız bir Model

Bölüm 4.2'de tanıtılan Anlamsal Web yetenekli MAS'lara ait PIMM'i üstmodel olarak kullanan bir MAS modeli bu bölümde anlatılmaktadır. Kullanıcıları adına otel odası rezervasyonu gerçekleştiren yazılım etmenlerini ve bulundukları çevreyi modelleyen bu platform bağımsız model tezin ilerleyen bölümlerinde de model dönüşümlerine girdi olarak verilecek örnek bir kaynak modeldir.

Şekil 4.5'te turizm iş alanında çalışan yazılım etmenleri ve görevleri doğrultusunda etkileşimde bulundukları servislere ait PIM verilmiştir. Kullanıcıları adına otel odası rezervasyonunda bulunan yazılım etmenleri "Otel İstemci Etmeni" adını almıştır. Bu etmenler PIMM'deki anlamsal web etmeni üstvarlığının birer örneğidir. Bir "Otel İstemci Etmeni" görev işletimi sırasında "Rezervasyon Tümleşik Servisi" adlı bir Anlamsal Web Servisi ile etkileşimde bulunur. Ortamda yer alan her otelin bu tip bir servisi vardır ve bu servis üzerinden ilgili otelde oda rezervasyonu gerçekleştirilmektedir.

Bir "Otel İstemci Etmeni", "Otel İstemci Rolü"'nü oynar ve söz konusu anlamsal web servisi ile etkileşimi sağlayan iki etmen planını uygular. Bu planlar "Otel İstemcisinin Servis Keşif Planı" ve "Otel İstemcisinin Servis Çağırma Planı" adlarına sahiptirler ve sırasıyla PIMM Anlamsal Servis Bulma Planı ve Anlamsal Servis Çalıştırma Planı üstvarlıklarının modeldeki örnekleridirler.

"Eşleme Etmeni" ilgili etmen platformunun servis eşleyicisidir ve PIMM Anlamsal Servis Eşleme Etmeni tipindedir. Bu etmen anlamsal web servislerine ait yetenek bilgilerini depolamaktadır ve "Otel İstemci Etmen"lerinden gelen servis istekleri ile bu yeteneklerin eşlemesini sağlamaktadır. Oynadığı kayıtçı rolü "Anlamsal Servis Eşleme" rolüdür. "Eşleme Etmeni" servislerin kaydedilmesi işlemlerini bir Anlamsal Servis Kaydetme Planı örneği olan "Eşleme Etmeninin Servis Kayıt Planı"nı uygulayarak gerçekleştirmektedir. Bu işlem için gösterdiği davranış ise "Servis Kaydet" adını almıştır. Bu etmen anlamsal servislerin arayüzlerini kendi kayıtçısında depolar ve diğer etmenlere ilan eder. Söz konusu arayüzler her bir anlamsal web servisinin "Rezervasyon Servisi Arayüzü" adı verilen ve PIMM Arayüz tipindeki yapılarıdır. Her bir "Rezervasyon Servisi Arayüzü"'nde ilgili anlamsal web servisinin ontolojik girdi/çıktı ve diğer özellikleri tanımlanmıştır.



Şekil 4.5. Turizm iş alanında çalışan bir etmen sistemine ait platform bağımsız model

“Eşleme Etmeni”’nin yetenek bilgilerini tuttuğu servisler birer “Rezervasyon Tümleşik Servisi”’dir. Bir “Otel İstemci Etmeni”, “Eşleme Etmeni”’ne aradığı anlamsal servis özelliklerini servis keşif planını uygulayarak iletir. Bu sırada gösterdiği etmen davranışı “Servis Kayıtçısını Sorgula”’dır. “Eşleme Etmeni” uygun rezervasyon servislerini istemci etmene bildirir. İstemci etmen de “Servisi Çağır” davranışı ile tanımlanmış işlemleri yerine getirerek uygun servisi çalıştırır. “Otel İstemcisinin Servis Çağırma Planı” dahilinde bu iş için öncelikle ilgili anlamsal web servisinin tanımladığı süreci (“Rezervasyon Servisi Süreci”) çalıştırır ve fiziksel etkileşimi sağlamak ve arka plandaki web servisini çalıştırmak için de yine aynı anlamsal web servisine ait zemini (“Rezervasyon Servisi Zemini”) kullanır.

İlgili platforma ait kavramların tanımlandığı ontolojiler de modelde yer almaktadır. Bir servis ontolojisi olan “Servisler.owl” platformdaki etmenler ve anlamsal web servisleri tarafından bilinmektedir. Anlamsal web servislerinin kavram tanımları bu ontolojiden gelmektedir. Dolaysıyla etmenler de servis ihtiyaçlarını belirtirken bu ontolojide tanımlı kavramları kullanırlar. Etmen rol tanımlarının ve organizasyonun bilgi deposu olarak da sırasıyla “OtelRezervasyonMAS.owl” ve “EtmenRolü.owl” ontolojileri modelde yer almaktadır. Örnek modelde daha çok etmen – servis etkileşiminin PIMM’e dayalı olarak modellenmesi dikkate alındığından ve örnek modelin mümkün olduğunca anlaşılabilir olması hedeflendiğinden PIMM’de yer alan diğer üstvarlıklar (örneğin mesaj, organizasyon, vb.) ve ilişkilere ait somut yapılar bu modele dahil edilmemiştir.

5 ANLAMSAL WEB YETENEKLİ MAS'LARIN MODEL GÜDÜMLÜ GELİŞTİRİLMESİ İÇİN MODEL DÖNÜŞÜMÜ

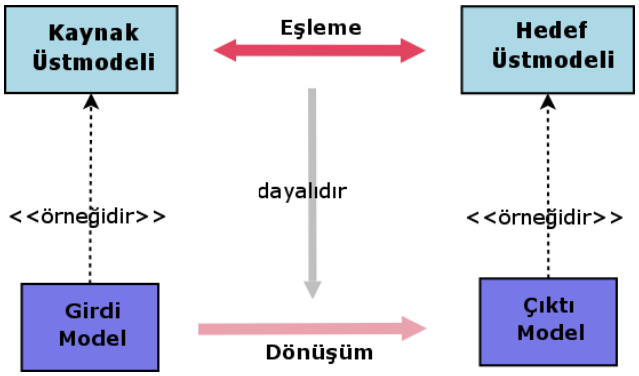
Sendall ve Kozaczynski (2003) model dönüşümünü model güdümlü yazılım geliştirmenin kalbi ve ruhu olarak nitelemektedirler. Gerçekten de üstmodellerin tanımlanması MDD için gereklidir ancak yeterli değildir. MDD sürecinin asıl ürünleri olan hedef modellerinin elde edilmesi için bu üstmodeller arasında dönüşümlerin tanımlanması ve uygulanması gerekmektedir.

Bir *model dönüşümü* bir dizi modelin başka bir dizi modele ya da kendi üzerlerine eşlenmesi olarak tanımlanabilir. Öte yandan *model eşleme* ise kaynak ve hedef üstmodellerinin elemanları arasındaki uygunluğu veya benzerliği tanımlar. Model elemanları arasındaki uygunluk birebir, n'e bir, bire n veya n'e n şekillerinde olabilir.

Şekil 5.1'de görüldüğü gibi model eşleme bir dönüşüme ait girdi ve çıktı modellerinin üstmodelleri arasında tanımlanır ve MDD süreçlerinin "tasarım aşaması"'nda gerçekleştirilir. Üstmodel varlıkları arasındaki eşlemeler model dönüşümü için kullanılan kuralları tanımlamaktadırlar. Bu kurallara dayalı olarak da model dönüşümleri "çalışma zamanı"'nda gerçekleştirilir.

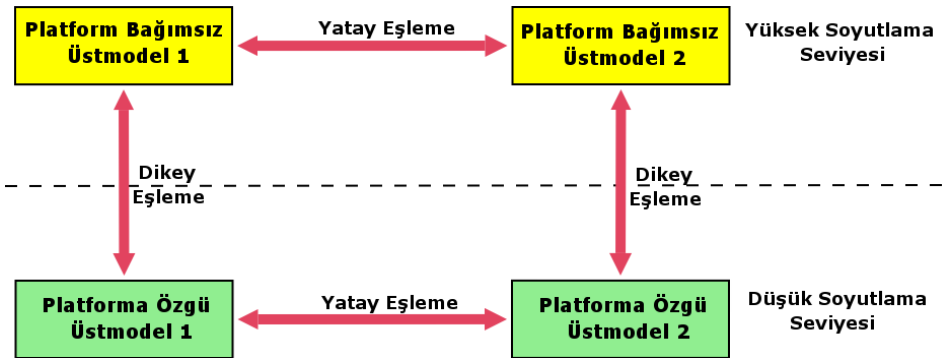
a ve *b* ile adlandırılan iki farklı ortama ait üstmodel ve modeller sırasıyla MM_a, MM_b, M_a ve M_b olmak üzere a ve b arasında tanımlanan bir eşleme fonksiyonu (*f*), MM_a, MM_b ve M_a 'yı kullanarak M_b'yi üretir ve aşağıdaki gösterime sahiptir:

$M_b \leftarrow f(MM_a, MM_b, M_a)$



Şekil 5.1. Model eşleme ve model dönüşümü

Modellerin yer aldığı soyutlama seviyelerinin birbirine göre konumu bu modeller arasındaki eşlemelerin niteliğini belirlemektedir (Mens and Van Gorp, 2006). Aynı soyutlama seviyesindeki modeller arasında yatay eşlemeler gerçekleştirilebilirken farklı soyutlama seviyelerindeki modeller arasında ise dikey eşlemeler gerçekleştirilir (Şekil 5.2). Tez çalışmasında tasarlanan ve gerçekleştirilen model dönüşümleri farklı soyutlama seviyelerinde yer alan platform bağımsız ve platforma özgü etmen model elemanları arasındaki dikey eşlemelere dayalıdır ve temelleri Bölüm 2.1.3'te anlatılmıştır.



Şekil 5.2. Farklı soyutlama seviyelerinde model eşlemeleri

Şekil 2.3'teki MDA'ya dayalı yazılım geliştirme süreci ve model dönüşüm mekanizması göz önüne alınacak olursa Anlamsal Web ortamında çalışan MAS'ların platforma özgü modellerini elde etmek amacıyla Bölüm 4.2'de tanıtılan PIMM ile çeşitli MAS platformlarının üstmodelleri arasında model dönüşümlerinin tanımlanması ve uygulanması gerekmektedir (Kardas et al., 2007b). Dönüşümlerin kaynak üstmodeli bu PIMM iken hedef üstmodeller ise çeşitli yazılım geliştirme çerçevelerinin PSMM'leridir.

Farklı her etmen platformu için izlenen yöntem önce bu platformun üstmodel elemanları ile PIMM elemanları arasındaki eşlemelerin gerçekleştirilmesini ve sonra bu eşlemelere dayalı olarak model dönüşüm kurallarının hazırlanıp girdi modeller üzerinde uygulanmasını içermektedir.

Tez çalışması kapsamında yukarıda tarif edilen model dönüşümünün SEAGENT (Dikenelli et al., 2005) ve NUIN (Dickinson and Wooldridge, 2003) adı verilen iki farklı MAS geliştirme çatısı için tasarımı ve gerçekleştirimi yerine getirilmiştir. Durum çalışması olarak da Bölüm 4.3'te anlatılan turizm MAS'ının bu etmen çatılarına uygun platforma özgü modellerinin dönüşümler sonrası elde edilmesi göz önüne alınmıştır.

SEAGENT çerçevesi için model dönüşümlerinin uygulanarak Anlamsal Web yetenekli MAS'ların çalışan modellerinin elde edilmesi aşağıdaki altbölümlerde anlatılmıştır. İlk altbölümde SEAGENT çerçevesi ve etmen üstmodeli hakkında bilgi verilmiş ve PIMM ile SEAGENT PSMM'i arasındaki varlık eşlemeleri gösterilmiştir. Bu eşlemelere dayalı olarak model dönüşüm kurallarının yazılması ve örnek model üzerinde uygulanması ise ikinci alt bölümde yer almaktadır. NUIN çerçevesi için uygulanan model dönüşümü ise tezin izleyen bölümlerinde ele alınmaktadır.

5.1 Model Dönüşümü için Varlık Eşlemelerinin Tanımlanması

Yazılım modelleri arasındaki dönüşüm, üstmodellerdeki varlıkların eşlenmesi sonucunda ortaya çıkan dönüşüm kurallarının örnek modeller üzerinde uygulanması ile gerçekleştirilmektedir. Anlamsal Web ortamında çalışan MAS'ların model dönüşümleri sonrası elde edilmesi için de tezde önerilen PIMM ve hedef platformların PSMM'lerinin üstvarlıkları arasındaki eşlemelerin yerine getirilmesi gerekmektedir. Eşlemeler daha önce de belirtildiği gibi bir ya da birden fazla kaynak üstmodeli varlığı ile yine bir ya da birden fazla hedef üstmodeli varlığı arasında olabilir.

Tezde öne sürülen model dönüşüm mekanizmasının ilk değerlendirmesi SEAGENT hedef platformu kullanılarak gerçekleştirilmiştir. PIMM ve SEAGENT PSMM'i arasındaki varlık eşlemeleri sunulmadan önce SEAGENT MAS geliştirme çerçevesi hakkında bilgi vermek yerinde olacaktır.

SEAGENT (Dikenelli et al., 2005) özellikle Anlamsal Web tabanlı MAS'ların geliştirilmesini sağlayan bir etmen yazılımı çerçevesi ve platformudur. SEAGENT'in iletişim ve plan çalıştırma altyapısı JADE (Bellifemine et al., 2001), DECAF (Graham et al., 2003) ve RETSINA (Sycara et al., 2003b) gibi mevcut diğer etmen geliştirme çerçeveleri ile benzerlikler göstermektedir. Ancak (Dikenelli et al., 2006)'da da belirtildiği gibi SEAGENT, Anlamsal Web tabanlı MAS geliştirmeyi desteklemek ve kolaylaştırmak için diğer etmen geliştirme çerçevelerinin sahip olmadığı ve aşağıda listelenen yerleşik (*"built-in"*) özelliklere sahiptir:

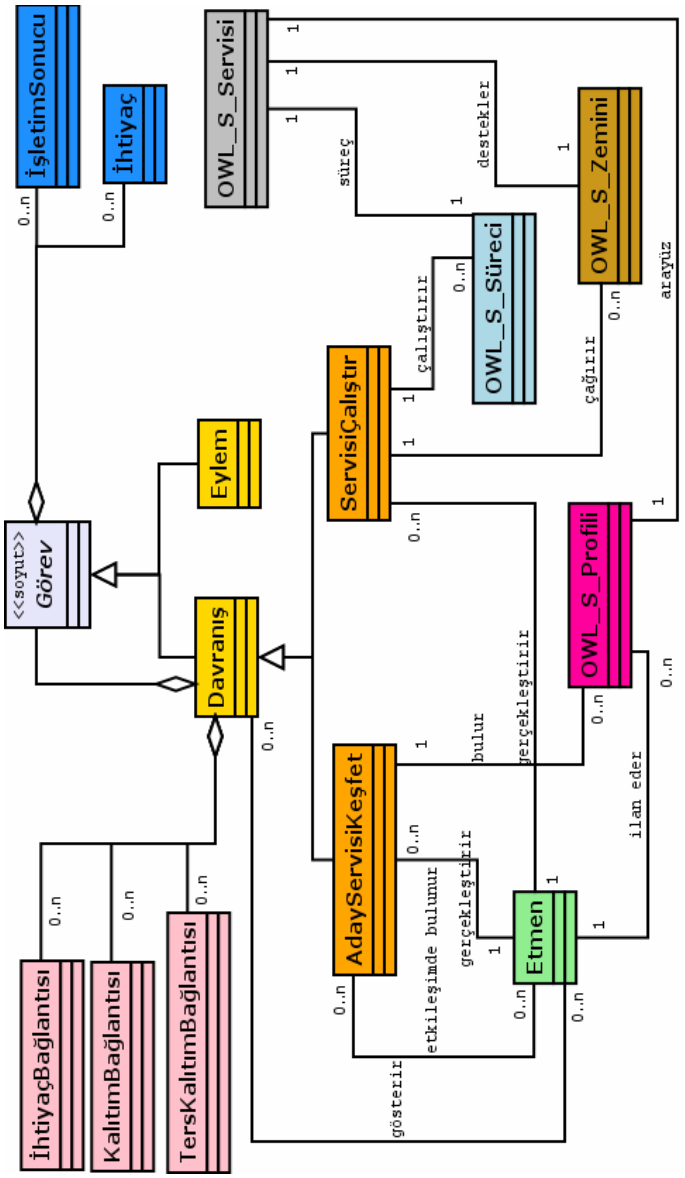
- SEAGENT, bir etmenin içmimarisinde OWL'a (McGuiness and van Harmelen, 2004) dayalı bilgi depolarını kullanmasına imkan vermektedir.

- SEAGENT'a dayalı olarak geliştirilen etmen dizin servisleri OWL tabanlı ontolojiler kullanılarak hazırlanmış etmen yetenek ilanlarını depolayabilirler ve platform etmenlerinin aradıkları etmen yeteneklerinden anlamsal olarak en uygunlarının getirilmesini sağlayan anlamsal eşleme motorlarına sahiptirler.
- SEAGENT bünyesinde FIPA-RDF'e (FIPA, 2002) dayalı SEAGENT İçerik Dili adı verilen bir içerik dili etmen mesaj iletişimlerinde anlamsal içeriğin transfer edilmesi için tanımlanmıştır.
- SEAGENT, FIPA uyumlu diğer etmen çerçevelerinin sunduğu servislere ek olarak Ontoloji Yönetim Servisi adı verilen yeni bir etmen servisi tanıtmaktadır. Bu servisin en önemli özelliği bir etmen platformuna ait ontolojiler ile harici ontolojiler arasındaki eşlemeyi tanımlamasıdır. Böylelikle bu eşlemelere dayalı olarak etmenler için bir ontoloji dönüşüm servisi sunulmaktadır.
- SEAGENT anlamsal web servislerinin etmenler tarafından keşfini ve dinamik olarak çalıştırılmasını desteklemektedir. SEAGENT bünyesinde anlamsal servis keşfi için ayrı bir platform servisi tanımlanmıştır. Keşfedilen servislerin dinamik çağrılması için de SEAGENT plan kütüphanesi hazır ve yeniden kullanılabilir etmen davranışlarını içermektedir.

SEAGENT[3] Java programlama dili kullanılarak gerçekleştirilmiştir ve Anlamsal Web tabanlı MAS'ların geliştirilmesi için yine Java'da hazırlanmış yazılım kütüphaneleri sunmaktadır. SEAGENT yazılım çerçevesinin HTN tabanlı etmen planlayıcısı ve etmen – servis

[3] SEAGENT MAS yazılımı geliştirme çerçevesinin güncel sürümüne ve dokümantasyonuna SEAGENT projesi web sayfasından (http://seagent.ege.edu.tr) erişilebilinir.

etkileşimini dikkate alan üstmodeli tez çalışması sırasında hazırlanmış ve model dönüşümlerinde PSMM olarak kullanılmıştır. SEAGENT çerçevesine ait bu üstmodel (PSMM) Şekil 5.3'te görülmektedir.



Şekil 5.3. SEAGENT MAS çerçevesine ait üstmodel

SEAGENT etmen planlama sistemine ait PSMM varlıkları Şekil 5.3'te üst kısımda gösterilmiştir. SEAGENT platformunda etmenler *Görev*lerini (*"Task"*) HTN (Williamson et al., 1996) karşıt eylemli planlama paradigmasına göre yerine getirirler. Bir yapay zeka planlama çeşidi olarak HTN planlama etmen planlarının görev parçalama (*"task decomposition"*) yöntemi ile işletilmesini sağlamaktadır. Bu görev parçalama planlama sisteminin doğrudan işletilebilecek basit görevleri belirlemesine kadar devam etmektedir. Örneğin SEAGENT'ta bir yazılım etmeninin bir anlamsal web servisi ile olan etkileşimi servis keşfi, servisle anlaşma ve servisi çalıştırma görevlerinden oluşan yeniden kullanılabilir bir plan olarak modellenebilir. Bu plana ait söz konusu bu alt görevleri sırasıyla çalıştıran bir etmen ilgili anlamsal web servisini kullanabilir.

HTN'in bir ihtiyacı olarak etmen görevleri karmaşık veya ilkel yapıda olabilir. SEAGENT'ta karmaşık görevler *Davranış* (*"Behaviour"*) adını alırken ilkel görevler ise *Eylem* (*"Action"*) olarak çağrılmaktadır. PSMM'de de bu üstvarlıklar Görev üstvarlığının altsınıfları olarak gösterilmiştir. Her bir etmen planı karmaşık bir kök görevden oluşmaktadır. Öntanımlı bir amaca ulaşmak için de bu kök görev bir ya da birden fazla altgörevi içerir. Davranışlar "indirgeme şeması" (*"reduction schema"*) adı verilen bir bilgiyi tutarlar. İndirgeme şeması karmaşık görevlerin altgörevlere parçalanmasını ve bu altgörevler ile ana görev arasındaki bilgi akışını tanımlamaktadırlar. SEAGENT'ta bu bilgi akışı mekanizması şu şekilde işlemektedir: Her görev çalışması için ihtiyaç duyduğu bilgiyi bir ihtiyaç (*"provision"*) kümesi ile temsil eder. Görevin çalışması sonucunda da işletim sonuçları (*"outcome"*) üretilir. Bir altgörevin işletim sonuçları ile bir sonraki altgörevin ihtiyaçları arasında bağlantılar mevcuttur ve bu bağlantılar görevler arası bilgi akışını sağlamaktadırlar. Söz konusu bağlantılar üç çeşittir: İşletim

sonuçlarını ihtiyaçlara bağlayan *İhtiyaç Bağlantısı* (*"Provision Link"*), ana görevdeki bir ihtiyacı bu ana görevin altgörevlerinden birine bağlayan *Kalıtım Bağlantısı* (*"Inheritance Link"*) ve bir altgörevin işletim sonucunu ilgili altgörevin ana görevine bağlayan *Ters Kalıtım Bağlantısı* (*"Disinheritance Link"*).

Eylemler etmen planlayıcılarının doğrudan işletebildiği ilkel görevlerdir. Her görev çalıştırıldıktan sonra etmen için bir sonuç durumu (*"outcome state"*) oluşturmaktadır. Bu durumlar bilgi akışının görevler arasında yönlendirilmesini yerine getirirler. SEAGENT MAS çerçevesinde her görevin ne yaptığını belirten bir adı vardır. Bunun dışında bilgi ihtiyaçlarını gösteren *İhtiyaçlar* (*"Provisions"*) ve çalıştırma sonuçlarını temsil eden *İşletim Sonuçları* (*"Outcomes"*) için bilgi kümelerine sahiptir. İhtiyaç bilgisi plan işletimi sırasında dinamik olarak sağlanmaktadır. Buna göre planı oluşturan görev ya da görevler ilgili her bir ihtiyaç sahası için değerlerin belli olmasından sonra çalışmaya hazır hale geçmektedirler. SEAGENT plan yapısı ile ilgili daha detaylı bilgi (Gürcan et al., 2006)'da bulunabilir.

Şekil 5.3'te görüldüğü gibi *Etmen*lerin anlamsal web servisleri ile etkileşimini sağlayacak öntanımlı iki Davranış üstvarlığı uzantısı da SEAGENT PSMM'inde yer almaktadır. Bu uzantılar *Aday Servisi Keşfet* (*"Discover Candidate Service"*) ve *Servisi Çalıştır* (*"Enact Service"*) adlı görevlerdir. Aday Servisi Keşfet görevinde bir etmen kendi ihtiyaçlarını karşılayabilecek en uygun anlamsal web servisini bulmaya çalışır. Servisi Çalıştır görevinde ise bir etmen, ihtiyacını karşılayacak servisi bulduktan sonra servisin çalışma mekanizmasına uygun bir şekilde servisi çalıştırır.

SEAGENT platformlarında anlamsal web servisleri OWL-S Servisleri (OWL-S Coalition, 2004) olarak gerçekleştirilirler ve kullanılırlar. Bu nedenle SEAGENT modellerinde her anlamsal web

servisi bir *OWL-S Servis* (*"OWL-S Service"*) örneği olarak tanımlanır ve servisin platform etmenleri tarafından kullanılmasını sağlamak amacıyla her servisin *OWL-S Profili* (*"OWL-S Profile"*), *OWL-S Süreci* (*"OWL-S Process"*) ve *OWL-S Zemini* (*"OWL-S Grouding"*) adı verilen yapıları bulunmaktadır. Bu OWL-S yapıları ile ilgili bilgi tezin 2.1.2.1. bölümünde verilmiştir.

SEAGENT MAS yazılımı geliştirme çerçevesinin yukarıda anlatılan üstmodeli tez kapsamında kullanılan PSMM'lerden biridir ve Anlamsal Web yetenekli MAS PIMM'i ile bu PSMM arasında model dönüşümleri gerçekleştirilmiştir.

Model dönüşüm sürecinin en önemli kısmı dönüşüm kurallarının önceden tanımlı bir model dönüşüm diline uygun bir şekilde hazırlanmasıdır. Daha önce de belirtildiği gibi dönüşüm kuralları model dönüşümünde yer alan kaynak ve hedef üstmodelleri arasındaki varlık eşlemelerine dayanmaktadır. Kurallar aynı zamanda dönüşüm sırasında uygulanacak eşleme kısıtlarının da biçimsel gösterimini içerirler. Buradaki durum için de PIMM ve SEAGENT PSMM'i arasındaki varlık eşlemelerinin tanımlanması gerekmiştir. Tablo 5.1'de bu varlık eşlemeleri görülmektedir.

PIMM ve SEAGENT PSMM'i arasındaki eşlemeler her zaman birebir yapıda olmamaktadır. Örneğin SEAGENT çerçevesi anlamsal servis eşleme ile ilgili özel bir etmen çeşidi tanımlamaz. Ancak PIM'lerde sözkonusu görevi üstlenen etmenler olabilir. Bu nedenle PIMM'in Anlamsal Web Etmeni ve Anlamsal Servis Eşleme Etmeni varlıkları SEAGENT'taki Etmen sınıfına eşlenmiştir.

Tablo 5.1. PIMM ve SEAGENT PSMM'i arasındaki varlık eşlemeleri

PIMM Varlığı	SEAGENT Varlığı	Açıklama
Anlamsal Web Etmeni, Anlamsal Servis Eşleme Etmeni	Etmen	SEAGENT'taki Etmen kavramı PIMM'deki Rol ve etmen soyutlamalarına karşılık gelmektedir.
Plan	Görev	PIMM Etmen planları SEAGENT çerçevesindeki HTN görevlerine eşlenirler.
Anlamsal Servis Kaydetme Planı	Davranış	Servis eşleme etmeninin kaydetme planı SEAGENT modelinde bir davranış ile hayata geçirilebilir.
Anlamsal Servis Bulma Planı	Aday Servisi Keşfet	Anlamsal web etmenlerinin servis bulma planı SEAGENT'ta yerleşik olan ve Davranış sınıfının bir altsınıfı olan Aday Servisi Keşfet görevine eşlenmektedir.
Anlamsal Servis Çalıştırma Planı	Servisi Çalıştır	Anlamsal web etmenlerinin servis çalıştırma planı SEAGENT'ta yerleşik olan ve Davranış sınıfının bir altsınıfı olan Servisi Çalıştır görevine eşlenmektedir.
Davranış	Eylem	PIMM'de tanımlanan etmen davranışları SEAGENT platformlarında birer Eylem örneği olarak hayata geçirilebilir.
Anlamsal Web Servisi Arayüz Süreç Zemin	OWL-S Servisi OWL-S Profili OWL-S Süreci OWL-S Zemini	SEAGENT'ta anlamsal web servislerinin yetenekleri ve süreç modelelleri OWL-S işaretleme dili kullanılarak tanımlanmaktadır. PIMM Anlamsal web servisi yapıları da OWL-S bileşenleri kullanılarak SEAGENT platformlarında hayata geçirilebilir.

Öte yandan SEAGENT üstmodelindeki Görev varlığı PIMM'deki soyut Planlara karşılık gelirken SEAGENT planlama kütüphanesinin öntanımlı davranış sınıfları da PIMM'in Plan uzantılarına karşılık gelmektedir. Örneğin SEAGENT plan kütüphanesi daha önce de belirtildiği gibi anlamsal web servislerinin yetenek ilanlarına göre bulunmasını sağlayan Aday Servisi Keşfet adlı bir davranış sınıfını içermektedir. Bir etmen bu davranışa göre eylemde bulunduğunda bir anlamda PIMM'de tanımlı Anlamsal Servis Bulma Planı'nı uygulamaktadır. Bu nedenle SEAGENT PSMM'indeki Aday Servisi Keşfet sınıfı PIMM'in Anlamsal Servis Bulma Planı üstvarlığının platforma özgü karşılıklarından biridir.

SEAGENT'ta anlamsal web servislerine ait yetenekler ve çalışma süreçleri OWL-S işaretleme dili kullanılarak tanımlandığı için PIMM'de yer alan servis üstmodel varlıkları (Arayüz, Süreç, Zemin, vb.) ve SEAGENT OWL-S bileşenleri arasında Tablo 5.1'de gösterilen eşlemeler gerçekleştirilerek ilgili model dönüşümleri sağlanmıştır.

5.2 Model Dönüşümü için Dönüşüm Kurallarının Hazırlanması ve Uygulanması

Model güdümlü diğer yazılım geliştirme süreçlerinde olduğu gibi tezde önerilen MDD'ye dayalı Anlamsal Web yetenekli MAS geliştirme sürecinin model dönüşümü safhasında da farklı soyutlama seviyelerindeki etmen üstmodeli varlık eşlemeleri yerine getirildikten sonra bu eşlemeler ve ilgili dönüşüm kısıtlarının biçimsel bir gösteriminin yapılması ve otomatik olarak model örnekleri üzerinde uygulanması yerine getirilmelidir.

MDD için model dönüşümlerinin yazılım ortamlarında biçimsel gösterimini gerçekleştirmek ve uygulanmasını sağlamak amacıyla model dönüşüm dillerinin kullanılmasına ihtiyaç duyulmaktadır. Literatürde bu

amaca yönelik bir çok model dönüşüm dili (Duddy et al., 2003; Kalnins et al., 2005; Agrawal et al., 2006; Jouault and Kurtev, 2006) ve bunları destekleyen yazılım araçları mevcuttur. Bu dönüşüm dillerinin sözdizim ve anlamsallığına uygun olarak hazırlanan model dönüşüm kuralları yazılım ortamlarında girdi modelleri üzerinde çalıştırılmakta ve çıktı modelleri elde edilmektedir. Tez çalışması kapsamında da model dönüşümü kurallarının hazırlanması ve uygulanması MDA ve MOF'u destekleyen ATL (*"Atlas Transformation Language"*) kullanılarak yerine getirilmiştir.

ATL (Jouault and Kurtev, 2006), ATLAS INRIA & LINA araştırma grubunun geliştirdiği ve MDA çalışmalarında oldukça sık kullanılan bir model dönüşüm dilidir. Özel olarak yazılım etmenlerinin model güdümlü geliştirilmesinde ATL'in kullanılmasına yönelik çalışmalar (Hahn et al., 2006; Rougemaille et al., 2007) da literatürde bulunmaktadır.

ATL hem bir dönüşüm üstmodeli olarak hem de metinsel somut bir sözdizim olarak tanımlanmıştır. MDA çerçevesinde genel dönüşümlerin gerçekleştirilmesini sağlamaktadır. Ayrıca dünyada şu an için en çok kullanılan yazılım geliştirme ortamlarından biri olan Eclipse Açık Geliştirme Platformu (Eclipse Community, 2003) için ATL bir eklenti (*"plug-in"*) sunmaktadır. Bu eklenti ATL kurallarının yazılmasını ve örnek modeller üzerinde otomatik işletimini sağlayan bir geliştirme ortamıdır. ATL'in sağladığı bu avantajlar tezde önerilen model dönüşümlerinin farklı PSMM'ler için gerçekleştirilmesinde ATL'in seçilmesinin başlıca nedenlerini oluşturmaktadırlar.

Aşağıdaki altbölümler tez kapsamında ATL kullanılarak gerçekleştirilen model dönüşümlerine ait genel bilgiyi vermekte, SEAGENT PSMM için hazırlanan model dönüşüm kurallarını

detaylandırmakta ve dönüşümlerin örnek model üzerinde uygulanmasını anlatmaktadır.

5.2.1 ATL kullanılarak gerçekleştirilen model dönüşümlerinin genel görünümü

Bir ATL[4] dönüşüm yapısı aşağıdaki elemanların birleşiminden oluşmaktadır (ATLAS Group, 2006):

- Dönüşümle ile ilgili bazı özelliklerin tanımlandığı bir başlık kısmı
- Varolan bazı ATL kütüphanelerinin kullanılmasını sağlayan seçimlik bir ithal kısmı
- Java programlama dili yordamlarının ATL eşleniği olarak kabul edilebilecek bir dizi yardımcı kural
- Kaynak modellerden hedef modellerin nasıl elde edileceğini tanımlayan bir dizi dönüşüm kuralı

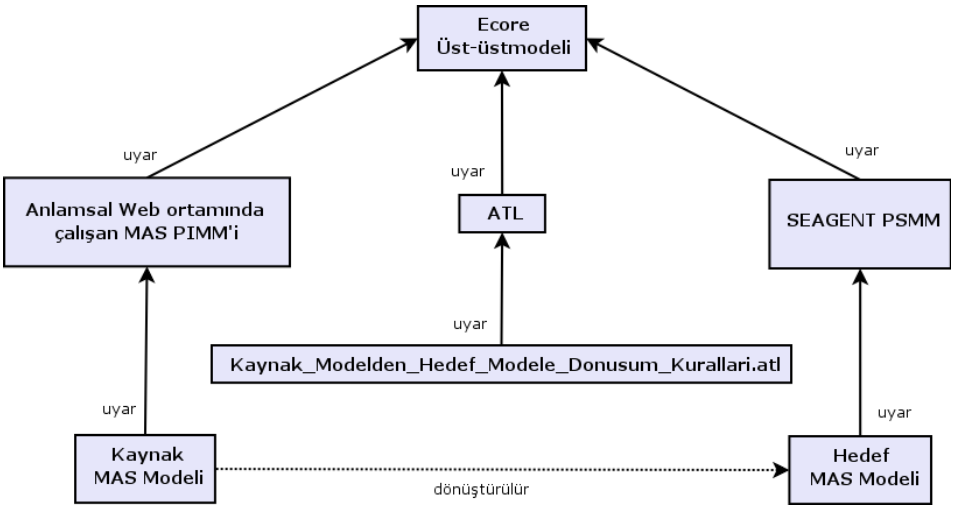
ATL yardımcı kuralları bir ATL dönüşümünün farklı noktalarından çağrılabilecek ATL kodlarının tanımlanmasını mümkün kılmaktadırlar. Öte yandan asıl ATL kuralları gerektiğinde bu yardımcı kuralları da kullanarak kaynak model elemanından hedef model eleman(lar)ının üretilmesini gerçekleştirirler. Her ATL kuralının *"from"* adı verilen kısmında hangi kaynak model elemanının dönüşümde yer alacağı belirtilirken *"to"* kısmında bu kaynak model elemanının eşlendiği ve

---

[4] Şu an için literatürde yer alan iki ATL derleyicisi vardır: ATL 2004 ve ATL 2006. ATL 2004 kuralların sorgu kısmında bir desen elemanına izin vermekte ve buna bağlı olarak tek bir kuralda birebir veya bire n dönüşümlerin tanımlanmasını sağlamaktadır. Öte yandan ATL 2006 birden fazla kaynak elemanına izin vermektedir. Tez çalışmasında model dönüşümleri gerçekleştirilirken ATL 2004 versiyonu kullanılmıştır. Çünkü çalışma hazırlandığı sırada ATL 2004'ün daha fazla dokümantasyona ve daha stabil bir çalışma ortamına sahip olduğu gözlenmiştir. Burada ve tüm tez boyunca ATL kelimesi ATL 2004'ü temsil etmektedir.

dönüşüm sonucunda ilgili örneklerinin oluşturulacağı hedef üstmodeli elemanı belirtilmektedir. İlgili dönüşüm kısıtları da göz önüne alınarak kaynak model elemanından hedef model elemanının oluşturulması sağlanmaktadır. ATL dil özellikleri hakkında daha detaylı bilgi (Jouault and Kurtev, 2006) ve (ATLAS Group, 2006)'da bulunmaktadır.

Anlamsal Web tabanlı MAS'lara ait PIMM ile SEAGENT çatısının PSMM'i arasındaki model dönüşümünün ATL kullanılarak gerçekleştirimi Şekil 5.4'te resmedilmiştir. Dönüşümlerin Bölüm 2.1.3'te anlatılan MOF temelli MDA model dönüşüm deseni ile birebir örtüştüğü ve bu desenin etmen sistemleri için gerçek bir örneğini oluşturduğu görülmektedir.



Şekil 5.4. ATL kullanılarak gerçekleştirilen modelden modele dönüşümler

Model dönüşümlerinin uygulanması sırasında kullanılan tüm üstmodellerin üst-üstmodeli (*"meta-metamodel"*) bir MOF varyantı olan Ecore'dur (Eclipse Community, 2004). Dönüşümlerin üstmodeli ATL'dir ve bu üstmodele uyan dönüşüm kuralları hazırlanmaktadır. Model

dönüşümlerine girecek kaynak MAS modelleri tezin 4.2. bölümünde anlatılan MAS PIMM'inin birer örnekleridirler ve bu PIMM'e uyarlar. Buradaki model dönüşümünün hedef üstmodelini ise SEAGENT MAS çerçevesinin Bölüm 5.1'de anlatılan yazılım üstmodeli oluşturmaktadır. Bu iki üstmodel arasındaki varlık eşlemelerine dayalı olarak hazırlanan dönüşüm kurallarının örnek kaynak modelleri üzerinde işletilmesiyle bu modellerin SEAGENT platformuna özgü eşleniklerinin elde edilmesi sağlanmaktadır. Süreci örneklemek amacıyla daha önce de belirtildiği gibi tezin 4.3. bölümünde anlatılan platform bağımsız turizm MAS modeli kaynak model olarak dönüşümlerde kullanılmıştır ve ilerleyen bölümlerde bu modelin SEAGENT eşleniğinin nasıl elde edildiği açıklanmaktadır.

ATL'in model dönüşüm motorunu kullanmak amacıyla kaynak ve hedef üstmodellerinin EMF (*"Eclipse Modeling Framework"*) kodlamalarının (*"encoding"*) hazırlanması ve Ecore dosyalarında saklanması gerekmektedir. EMF model ve üstmodel kodlama için ".ecore" adı verilen kendi dosya biçimini sağlamaktadır (Eclipse Community, 2004). Ancak Ecore üstmodellerinin EMF kullanılarak elle düzenlenmesi oldukça güç bir iş olduğundan hem PIMM'in hem de tez çalışmalarında kullanılan PSMM'lerin Ecore kodlamalarının oluşturulması amacıyla ADT'de ("ATL Development Tools") yer alan KM3 ("Kernel MetaMetaModel") üst-üstmodel metinleme notasyonu kullanılmıştır. Jouault ve Bezivin'in (Jouault and Bezivin, 2006)'da tanıttığı KM3, üstmodel düzenlemeyi Java programlama diline benzer bir sözdizimle kolaylaştırmaktadır. Bir KM3 üstmodeli düzenlendikten sonra ADT'ye entegre edilmiş enjektörler kullanılarak Ecore biçimine otomatik olarak çevrilebilir. Böylece model dönüşümleri için kaynak ve hedef üstmodellerinin ilgili yazılım ortamı için hazırlanmaları tamamlanmış olmaktadır. KM3 ve KM3'ün Ecore'a dönüştürülmesi ile ilgili detaylı

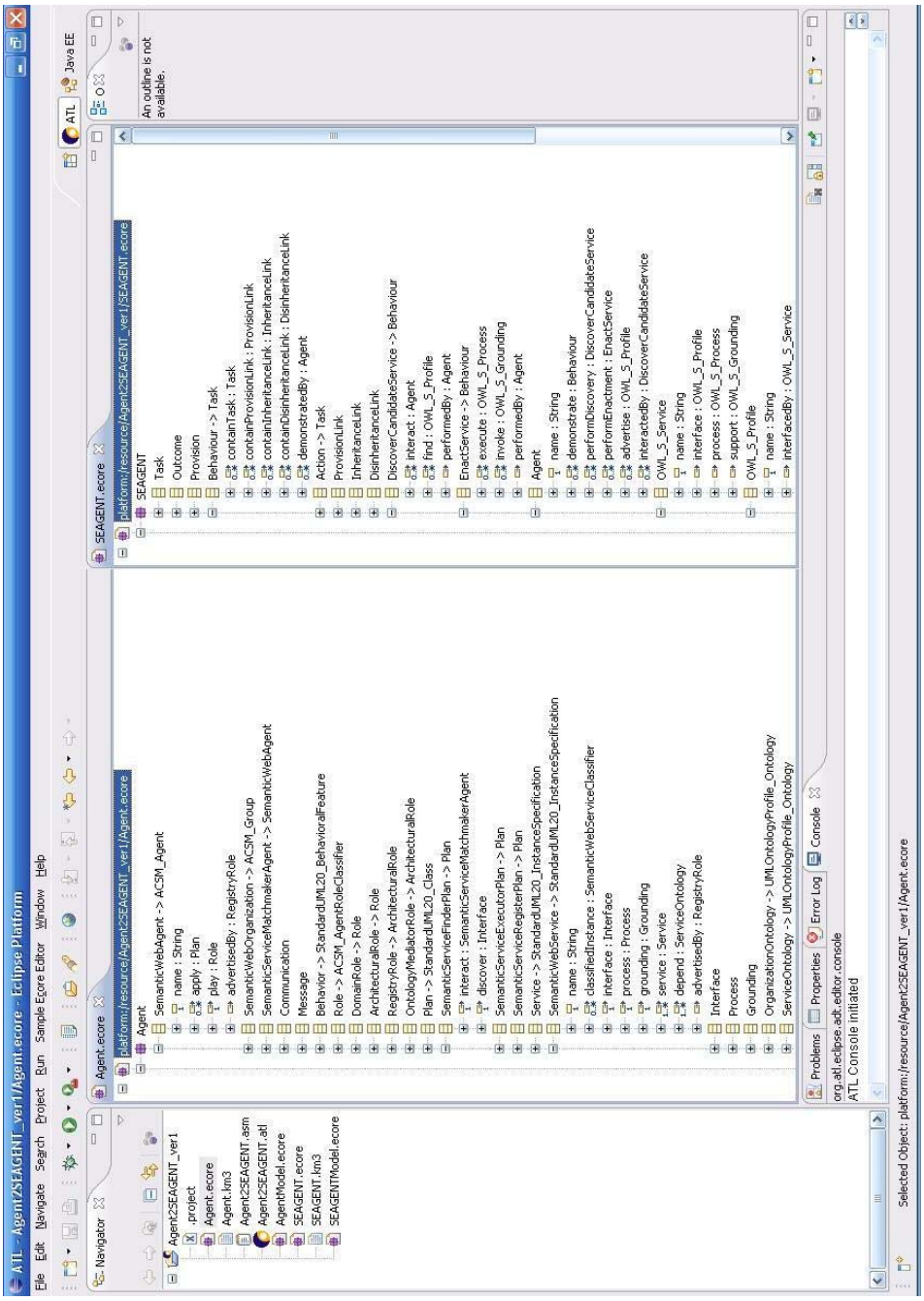
bilgi (Jouault and Kurtev, 2006) ve (Jouault and Bezivin, 2006)'da yer almaktadır.

Anlamsal Web ortamında çalışan MAS PIMM'i ve SEAGENT PSMM'i model dönüşümlerinde kullanılmadan önce KM3 kullanılarak metinsel gösterimleri hazırlanmış ve ADT Ecore enjektörü kullanılarak bu üstmodellerin Ecore gösterimleri elde edilmiştir. Şekil 5.5'te her iki üstmodelin de Ecore çevrimleri sonrasında ATL ortamındaki görsel temsilleri görülmektedir. PIMM ve SEAGENT PSMM'ine ait Ecore gösterimleri tezin Ek 1 ve Ek 2 bölümlerinde verilmiştir.

Her ne kadar PIMM'in UML 2.0 Üstyapısı'ndan ve diğer üstmodellerden miras edindiği üstvarlıklar model dönüşümlerinde doğrudan kullanılmasalar da Şekil 5.5 ve Ek 1'de görüleceği üzere PIMM'in Ecore gösterimine dahil edilmişlerdir. Çünkü bu yapılar temel yapılardır ve PIMM çekirdek üstmodelinin birçok varlığı bu yapılara dayanmaktadır.

PIMM ve SEAGENT PSMM'indeki tüm elemanların Ecore gösterimlerinin burada anlatılması mümkün olmamakla birlikte gösterimleri örneklemek amacıyla iki üstmodeldeki etmen kavramlarının KM3 notasyonunda gösterimleri ve buna karşılık elde edilen Ecore elemanları aşağıda anlatılmıştır.

Bölüm 5.1'de verilen varlık eşlemelerine göre PIMM'deki "Anlamsal Web Etmeni" varlığının SEAGENT platformundaki karşılığı "Etmen"'dir. "Anlamsal Web Etmeni" varlığının KM3 gösterimi şu şekildedir:



Şekil 5.5. ATL ortamında PIMM ve SEAGENT PSMM'inin Ecore sözdizimlerinin oluşturulmasından sonra elde edilen görsel temsilleri

```
class SemanticWebAgent extends ACSM_Agent {
      attribute name: String;
      reference apply[0-*]: Plan oppositeOf appliedBy;
      reference play: Role oppositeOf playedBy;
      reference advertisedBy[0-1]: RegistryRole oppositeOf
                advertiseAgent;
}
```

"Anlamsal Web Etmeni" üstvarlığının diğer üstmodel elemanları ile olan ilişkileri "*reference"* etiketleri ile verilmiştir. Her ilişki için "Anlamsal Web Etmeni"'nin ilgili ilişkide üstlendiği rol adı ve örnek sayısı verilmiştir. Her ilişkideki "*oppositeOf"* etiketinden sonra ilişkinin karşı tarafındaki üstvarlığın üstlendiği rol adı yer almaktadır. Yukarıdaki KM3 biçimindeki üstmodel elemanı tanımı Ecore'a çevrildiğinde aşağıdaki gösterim elde edilmektedir:

```
<eClassifiers xsi:type="ecore:EClass"
   name="SemanticWebAgent" eSuperTypes="/0/ACSM_Agent">
   <eStructuralFeatures xsi:type="ecore:EAttribute"
      name="name" ordered="false" unique="false"
      lowerBound="1" eType="/1/String"/>
   <eStructuralFeatures xsi:type="ecore:EReference"
      name="apply" ordered="false" upperBound="-1"
      eType="/0/Plan" eOpposite="/0/Plan/appliedBy"/>
   <eStructuralFeatures xsi:type="ecore:EReference"
      name="play" ordered="false" lowerBound="1"
      eType="/0/Role" eOpposite="/0/Role/playedBy"/>
   <eStructuralFeatures xsi:type="ecore:EReference"
      name="advertisedBy" ordered="false"
      eType="/0/RegistryRole"
      eOpposite="/0/RegistryRole/advertiseAgent"/>
</eClassifiers>
```

Ecore gösterimlerinde her üstmodel elemanı bir "*eClassifier"* olarak yer almaktadır. Veri tipi ve referans özellikleri "*eStructuralFeatures"* ile gösterilmektedir. Hedef üstmodelinde

(SEAGENT PSMM) “Anlamsal Web Etmeni”’nin eşlendiği “Etmen” üstvarlığının KM3 ve Ecore gösterimleri de sırasıyla aşağıda verilmiştir:

```
class Agent {
     attribute name: String;
     reference demonstrate[0-*]: Behaviour oppositeOf
               demonstratedBy;
     reference performDiscovery[0-*]:
           DiscoverCandidateService oppositeOf performedBy;
     reference performEnactment[0-*]: EnactService
               oppositeOf performedBy;
     reference advertise[0-*]: OWL_S_Profile oppositeOf
               advertisedBy;
     reference interactedBy[0-*]: DiscoverCandidateService
               oppositeOf interact;
}
```

```
<eClassifiers xsi:type="ecore:EClass" name="Agent">
  <eStructuralFeatures xsi:type="ecore:EAttribute"
     name="name" ordered="false" unique="false"
     lowerBound="1" eType="/1/String"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
     name="demonstrate" ordered="false" upperBound="-1"
     eType="/0/Behaviour"
     eOpposite="/0/Behaviour/demonstratedBy"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
     name="performDiscovery" ordered="false" upperBound="-
     1" eType="/0/DiscoverCandidateService"
     eOpposite="/0/DiscoverCandidateService/performedBy"/>
   <eStructuralFeatures xsi:type="ecore:EReference"
     name="performEnactment" ordered="false" upperBound="-
     1" eType="/0/EnactService"
     eOpposite="/0/EnactService/performedBy"/>
   <eStructuralFeatures xsi:type="ecore:EReference"
     name="advertise" ordered="false" upperBound="-1"
     eType="/0/OWL_S_Profile"
     eOpposite="/0/OWL_S_Profile/advertisedBy"/>
   <eStructuralFeatures xsi:type="ecore:EReference"
     name="interactedBy" ordered="false" upperBound="-1"
     eType="/0/DiscoverCandidateService"
     eOpposite="/0/DiscoverCandidateService/interact"/>
</eClassifiers>
```

5.2.2 Model dönüşüm kurallarının ATL kullanılarak hazırlanması

Dönüşümlerde kullanılacak kaynak ve hedef etmen üstmodelleri bir önceki altbölümde anlatıldığı gibi hazırlandıktan sonra bu üstmodeller arasındaki varlık eşlemelerine dayalı olarak bulgusal (*"heuristic"*) dönüşüm kurallarının uygun sözdizimde yazılması gerekmektedir. PIMM ve çeşitli PSMM'ler arasındaki model dönüşüm kuralları tez çalışmasında ATL sözdizim ve anlamsalı kullanılarak hazırlanmıştır.

Dönüşüm için yazılan her ATL kuralı kaynak kısmında bir kaynak üstmodel elemanını tanımlar ve girdi olarak verilecek bir modelde ilgili kaynak desenini sorgulayacak kısıtları içermektedir. Bu kurallar Şekil 5.4'te gösterildiği gibi bir ATL dosyası içerisinde hazırlanır. Model dönüşüm motoru bu kuralları kaynak üstmodeline uyan bir girdi modeli üzerinde uygulayarak hedef üstmodeline uyan bir karşılığını otomatik olarak elde etmektedir.

Anlamsal Web ortamında çalışan MAS PIMM'i ve SEAGENT MAS çerçevesine ait PSMM elemanları arasında Bölüm 5.1'de anlatılan varlık eşlemelerine uygun olarak hazırlanan dönüşüm kuralları ve bu kuralların kullandığı yardımcı kurallar tezin Ek 3 bölümünde verilmiştir. Bu kurallardan PIMM "Anlamsal Web Etmeni" üstvarlıklarını SEAGENT PSMM "Etmen" varlığına ve yine PIMM "Anlamsal Servis Bulma Planı" üstvarlıklarını SEAGENT PSMM "Aday Servisi Keşfet" üstvarlıklarına dönüştüren kurallar örnek olarak seçilmiş ve aşağıda anlatılmıştır.

Bir kaynak modeldeki PIMM "Anlamsal Web Etmeni" örneklerinin SEAGENT karşılıklarının elde edildiği `SemanticWebAgent2Agent` adı verilen dönüşüm kuralı şu şekildedir:

```
1 rule SemanticWebAgent2Agent {
2   from
3     ag: Agent!SemanticWebAgent (
4         ag.partofPatternforWebAgent
5     )
6   to
7     sa: SEAGENT!Agent (
8         name <- ag.name,
9         performDiscovery <- ag.finderPlans,
10        performEnactment <- ag.executorPlans
11     )
12 )
```

Dönüşüm kuralları yazılırken girdi üstmodeli (PIMM) "Agent" tanımlayıcısı ile çıktı üstmodeli de (PSMM) "SEAGENT" tanımlayıcısı ile temsil edilmiştir. Yukarıdaki dönüşüm kuralı kaynak modeldeki her bir "Anlamsal Web Etmeni" örneği için bir SEAGENT "Etmen" örneği oluşturmakta (Satır 7) ve bu örneğin ilgili özellik ve ilişkilerini hazırlamaktadır. Kuralın kaynak kısmında verilen (Satır 3) "Anlamsal Web Etmeni" sınıfı örnekleri kaynak modelde belirlenirken bu sınıf örneklerini ayırt edecek ve ilişkilerini belirleyecek yardımcı kurallar kullanılmaktadır. `SemanticWebAgent2Agent` kuralında da bu tip yardımcı kuralların kullanıldığı görülmektedir (Satır 4, 9 ve 10).

Yardımcı kurallar asıl kuralların kısıtlama (*"constraint"*) kısmını oluşturmaktadırlar. Kısıtlar kaynak modeller sorgulanırken kullanılırlar. ATL'de kısıtlar da OCL (*"Object Constraint Language"*) (Warmer and Kleppe, 2003) kullanılarak hazırlanmaktadır. Aynı yardımcı kurallar ve kısıt tekrarlarının aynı hedef model dönüşümü için ya da başka model dönüşümleri için (örneğin burada SEAGENT dışında başka bir MAS geliştirme çerçevesine ait PSMM) kullanılması gerekebilir. Bu şekilde yardımcı kuralların ana kurallardan ayrıştırılması aynı yardımcı kuralların başka dönüşümlerde de kullanılabilmesini sağlamaktadır.

Etmen sistemlerine ait PIMM – PSMM dönüşümleri için tezde hazırlanan yardımcı dönüşüm kuralları iki tiptir (Kardas et al., 2007b):

- Üzerinde çalışılan model elemanının istenilen desene ait olup olmadığını belirleyen yardımcı kurallar
- Hedef elemanları arasındaki ilişkileri kaynak ve hedef üstmodeli dikkate alınarak oluşturan yardımcı kurallar

Yukarıdaki `SemanticWebAgent2Agent` kuralında her iki tipe de uyan yardımcı kurallar kullanılmıştır. Kuralın 4. satırında çağrılan `partofPatternforWebAgent` yardımcı kuralı ilk tipe örnektir. Bu yardımcı kural ile dönüşüme girdi olarak verilen bir modeldeki Anlamsal Web Etmen'leri belirlenir. Kuralın 9 ve 10. satırlarında çağrılan `finderPlans` ve `executorPlans` yardımcı kuralları ise ikinci tipe örnek oluşturmaktadırlar. Bu yardımcı kurallardan `finderPlans` aşağıda verilmiştir:

```
1 helper context Agent!SemanticWebAgent def:
2  finderPlans : Sequence(Agent!SemanticServiceFinderPlan)
3    = self.apply->select(fp|fp.oclIsTypeOf(
4      Agent!SemanticServiceFinderPlan))->select(fndpln|
5        fndpln.appliedBy->forAll(agnt| not
6          agnt.oclIsTypeOf(
7            Agent!SemanticServiceMatchmakerAgent)
8        and not agnt.play.oclIsTypeOf(Agent!RegistryRole))
9        and fndpln.interact.play.oclIsTypeOf(
10         Agent!RegistryRole)
11       and fndpln.interact.apply->select(p|p.oclIsTypeOf(
12         Agent!SemanticServiceRegisterPlan))->forAll(p|
13         p.advertise->exists(
14           intfc|intfc.discoveredBy=fndpln))
15       and fndpln.discover.advertisedBy.appliedBy->
16         exists(sma|sma.interactedBy=fndpln));
```

`finderPlans` yardımcı kuralı ile bir "Anlamsal Web Etmeni" model elemanına ait "Anlamsal Servis Bulma Planı" Plan örneklerinin

kaynak modelde bulunması yerine getirilmektedir. Böylelikle dönüşümü gerçekleştiren ana kuralda hedef modeldeki ilgili etmen örneklerinin anlamsal servis bulma planları da belirlenmiş ve ilişkileri hazır hale getirilmiş olmaktadır. `finderPlans` yardımcı kuralının 5 ve 16. satırları arasındaki OCL kısıtları, üzerinde çalışılan Plan örneğinin bir "Anlamsal Servis Bulma Planı" olup olmadığını diğer model elemanları ile olan ilişkilerine bakarak belirlemektedir. Belirlenen plan örneğinin ilgili "Anlamsal Web Etmeni" model elemana ait bir "Anlamsal Servis Bulma Planı" olup olmadığı ise yardımcı kuralın 3. ve 4. satırındaki sorgular ile belirlenmektedir. Kısıtlara uyan plan örnekleri bir dizi halinde yardımcı kuralın sorgu sonucu olarak asıl kurala döndürülmektedir. Bölüm 5.1'deki varlık eşlemelerine göre PIMM'deki "Anlamsal Servis Bulma Planı" örneklerinin SEAGENT modellerindeki karşılıkları birer "Aday Servisi Keşfet" planı olduğu için `SemanticWebAgent2Agent` kuralının 9. satırında `finderPlans` yardımcı kuralından dönen plan örneklerinin uygun SEAGENT etmeni için ataması gerçekleştirilmektedir.

PIMM - SEAGENT PSMM dönüşüm kurallarına ait bir diğer örnek ise PIMM "Anlamsal Servis Bulma Planı" üstvarlıklarını SEAGENT PSMM "Aday Servisi Keşfet" üstvarlıklarına dönüştüren kuraldır. `SemanticServiceFinderPlan2DiscoverCandidateService` adı verilen bu kural aşağıda verilmiştir:

```
1 rule SemanticServiceFinderPlan2DiscoverCandidateService {
2   from
3     fndpln: Agent!SemanticServiceFinderPlan (
4             fndpln.partofPatternforFinderPlan
5     )
6   to
7     plnf: SEAGENT!DiscoverCandidateService (
8       name <- fndpln.name,
9       containTask <- Sequence{
10        Agent!Behavior->allInstances()->asSequence()->
11        select(bhv|bhv.execute->exists(pln|pln=fndpln))},
```

```
12      interact <- Sequence {
13        Agent!SemanticServiceMatchmakerAgent->
14        allInstances()->asSequence()},
15      find <- Sequence {
16        Agent!Interface->allInstances()->asSequence()}
17    )
18 }
```

Kuralda SEAGENT "Aday Servisi Keşfet" hedef elemanı oluşturulduktan sonra bu planın içerdiği görevlerin bağlantısı da gerçekleştirilmektedir (9 ve 11. satırlar arası). Kaynak modelde ilgili "Anlamsal Servis Bulma Planı" plan örneğinin içerdiği etmen davranışları belirlenmekte ve bunlara karşılık hedef modelde oluşturulan SEAGENT görevlerinin referansları da yukarıdaki kuralla oluşturulmuş "Aday Servisi Keşfet" model örneğinin ilgili özelliğine atanmaktadır. Benzer şekilde kuralın 12 ve 14. satırları arasında planın etkileşimde bulunacağı servis eşleme etmenleri ve 15 ve 17. satırlar arasında da planın bulmakla yüklü olduğu anlamsal servis arayüzleri ile oluşturulan SEAGENT plan örneğinin bağlantısı gerçekleştirilmektedir. Kuralın 4. satırında ise aşağıda verilen `partofPatternforFinderPlan` yardımcı kuralı çağrılmaktadır:

```
1 helper context Agent!SemanticServiceFinderPlan def:
2  partofPatternforFinderPlan : Boolean =
3   if self.appliedBy->forAll(agnt| not
4    agnt.oclIsTypeOf(Agent!SemanticServiceMatchmakerAgent)
5    and not agnt.play.oclIsTypeOf(Agent!RegistryRole))
6    and self.interact.play.oclIsTypeOf(Agent!RegistryRole)
7    and self.interact.apply-> select(p|p.oclIsTypeOf(
8      Agent!SemanticServiceRegisterPlan))->
9       forAll(p|p.advertise->
10       exists(intfc|intfc.discoveredBy=self))
11   and self.discover.advertisedBy.appliedBy->
       exists(sma|sma.interactedBy=self)
12   then
13     true
14   else
```

```
15      false
16  endif;
```

`partofPatternforFinderPlan` yardımcı kuralı yukarıda sözü edilen yardımcı kural kategorilerinden birincisine girmektedir ve üzerinde çalışılan model elemanının bir "Anlamsal Servis Bulma Planı" örneği olup olmadığının belirlenmesini gerçekleştirir. 3 ve 12. satırlar arasındaki kısıtlar ilgili Plan örneğinin "Anlamsal Servis Eşleme Etmeni", "Kayıtçı Rolü" ve "Arayüz" örnekleri ile olan ilişkilerini kontrol etmektedir. Üzerinde çalışılan model elemanı şartları sağlıyorsa yardımcı kural ana kurala "doğru" değerini döndürür ve bu elemanın ilgili SEAGENT PSMM dönüşümü sağlanır.

5.2.3 Model dönüşümlerinin örnek girdi modeli üzerinde uygulanması

Bölüm 5.2.1'de anlatılan model dönüşümleri için kaynak ve hedef üstmodellerinin Ecore gösterimlerinin kodlanması ve Bölüm 5.2.2'de anlatılan dönüşüm kurallarının yazılması tezde önerilen MDD sürecinin "tasarım aşamasına" ait işlemlerdir. Bu süreci kullanarak platforma özgü MAS modelleri elde etmek isteyen etmen yazılımı geliştiricilerinin bu işlemlerle uğraşmasına, farklı sistem tasarımları için tekrar tekrar üstmodellerin tanımlanmasına ve yukarıda anlatılan ATL kurallarının yazılmasına gerek yoktur. Bir MAS yazılımı geliştiricisinin bu süreç içerisinde yapması gereken sadece Anlamsal Web ortamında çalışan MAS PIMM'ine uyan platform bağımsız sistem modelini tasarlamak ve bu modelin (PIM) Ecore veya XMI gösterimini hazırlayarak otomatik model dönüşümünü bu model üzerinde uygulamaktır. Böylelikle aynı PIM'in farklı etmen platformları için karşılıklarının elde edilmesi (PSM'ler) mümkün olmaktadır ki bu da savunulan MAS geliştirme sürecinin en önemli avantajıdır.

Girdi modelleri üzerinde sözkonusu PIM – SEAGENT PSM dönüşümünü örneklemek amacıyla tezin 4.3. bölümünde tanıtılan platform bağımsız turizm MAS'ının SEAGENT karşılığının elde edilmesi burada anlatılmaktadır.

Model dönüşümünün uygulanması için öncelikle Şekil 4.5'te resmedilen Turizm MAS'ına ait platform bağımsız modelin Ecore gösterimi hazırlanmıştır. PIMM'e dayalı bu kaynak (girdi) modelin Ecore gösterimi tezin Ek 4 bölümünde verilmiştir. Her model elemanı ilgili PIMM üstvarlığının bir örneğidir. Model elemanları arasındaki ilişki gösterimlerinde yer alan referans numaraları modelin Ecore dosyasında model elemanlarının konumlarını belirtmektedir. Örneğin plan dönüşümü göz önünde bulundurulacak olursa kaynak model, "Otel İstemcisinin Servis Keşif Planı" adlı Anlamsal Servis Bulma Planı örneğini içermektedir. Bu model elemanı ve diğer model elemanları ile olan ilişkiler kaynak Ecore modelde aşağıdaki gibi tanımlanmıştır. Model elemanlarının asıl Ecore dosyasındaki konumları sol taraftaki rakamlarla gösterilmiştir. "Otel İstemcisinin Servis Keşif Planı" model elemanı hariç diğer model elemanlarının başka model elemanları ile olan ilişki bağlantıları burada gösterilmemiştir ve "..." ile temsil edilmiştir.

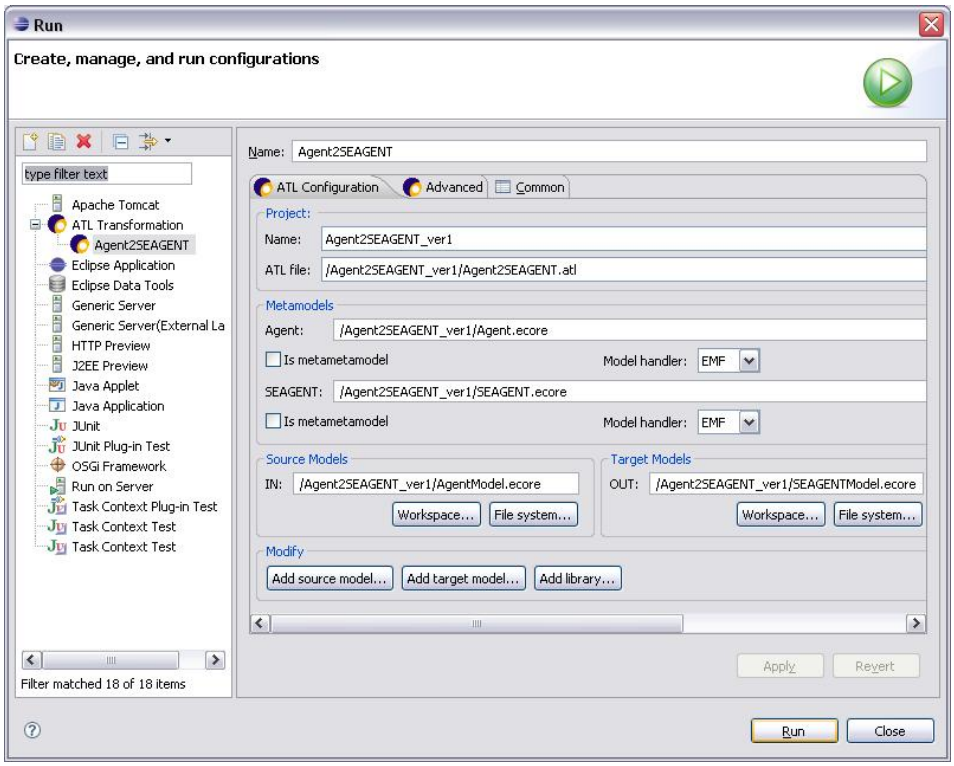
```
<xmi:XMI xmi:version="2.0"
 xmlns:xmi="http://www.omg.org/XMI" xmlns="Agent">
 [4]  <SemanticServiceFinderPlan name="Otel Istemcisinin
          Servis Kesif Plani" appliedBy="#/0"
          executedBy="#/10" interact="#/1" discover="#/7"/>
 [0]  <SemanticWebAgent name="Otel Istemci Etmeni"
          apply="#/4 ..."   play="..."/>
 [10] <Behavior name="Servis Kayitcisini Sorgula"
    includedBy="..." execute="#/4" knowRoleOntology="..."/>
 [1]  <SemanticServiceMatchmakerAgent name="Esleme Etmeni"
          apply="..." play="..." interactedBy="#/4"/>
 [7]  <Interface name="Rezervasyon Servisi Arayuzu"
        owner="..." advertisedBy="..." discoveredBy="#/4"/>
</xmi:XMI>
```

`SemanticServiceFinderPlan2DiscoverCandidateService` adlı, Bölüm 5.2.2'de anlatılan model dönüşüm kuralı bu kaynak model elemanı üzerinde uygulandığında "Otel İstemcisinin Servis Keşif Planı"'nın SEAGENT platformuna özgü hedef (çıktı) model karşılığı otomatik olarak elde edilmektedir. Varlık eşlemesine uygun olarak elde edilen bu model elemanı bir SEAGENT Aday Servisi Keşfet üstvarlığı örneğidir. Dönüşüm sonucu elde edilen bu hedef model elemanının Ecore gösterimi aşağıda verilmiştir. Model elemanlarının asıl Ecore dosyasındaki konumları yine sol taraftaki rakamlarla gösterilmiştir. "Otel İstemcisinin Servis Keşif Planı" model elemanı hariç diğer model elemanlarının başka model elemanları ile olan ilişki bağlantıları da yine burada gösterilmemiştir ve "..." ile temsil edilmiştir.

```
<xmi:XMI xmi:version="2.0"
 xmlns:xmi="http://www.omg.org/XMI"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="SEAGENT">
 [2] <DiscoverCandidateService name="Otel Istemcisinin
       Servis Kesif Plani" interact="/1" find="/6"
       performedBy="/0">
         <containTask xsi:type="Action" name="Servis
          Kayitcisini Sorgula"/>
     </DiscoverCandidateService>
 [0] <Agent name="Otel Istemci Etmeni"
       performDiscovery="/2" performEnactment="..."/>
 [1] <Agent name="Esleme Etmeni" demonstrate="..."
       advertise="..." interactedBy="/2"/>
 [6] <OWL_S_Profile name="Rezervasyon Servisi Arayuzu"
       interfacedBy="..." advertisedBy="..."
       foundedBy="/2"/>
</xmi:XMI>
```

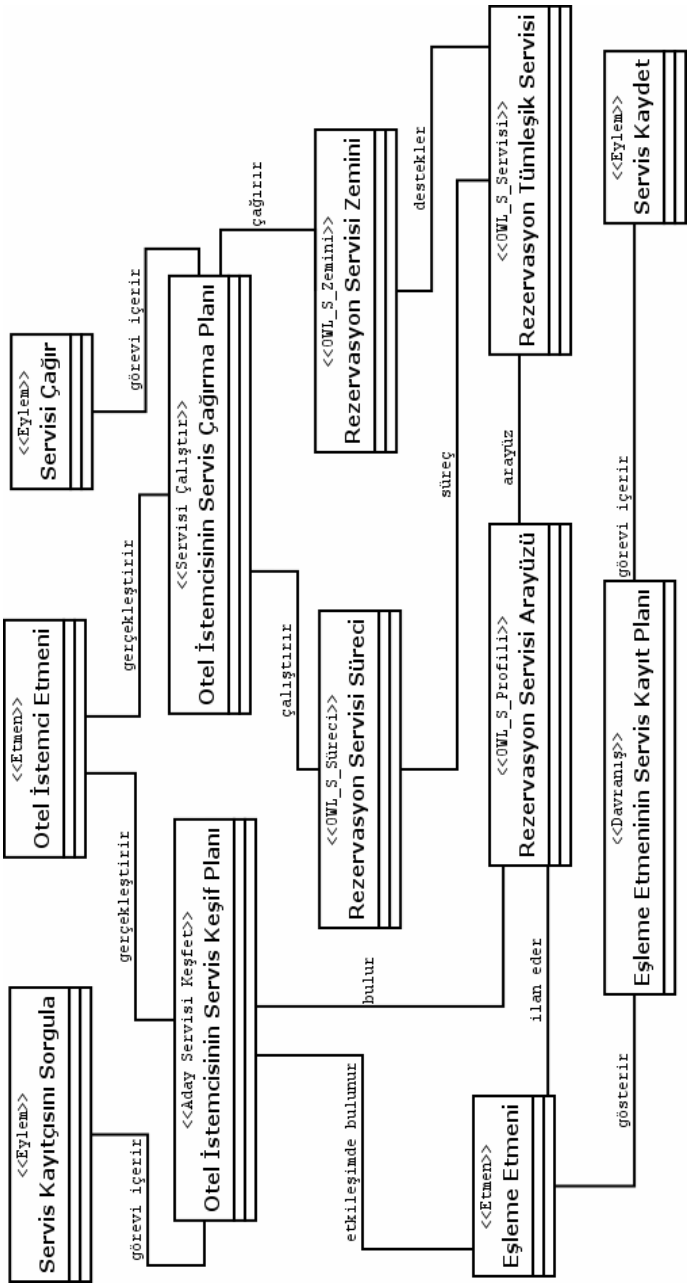
Platform bağımsız MAS modelinin Ecore gösterimi hazırlanıp dönüşüme girdi olarak verilmesinden sonra modelin SEAGENT karşılığı elde edilmektedir. Şekil 5.6'da ATL ortamında model dönüşümü çalışma zamanı ayarlarının nasıl yapıldığı görülmektedir. Dönüşümün kaynak ve

hedef üstmodelleri ve dönüşüme girdi olarak verilecek MAS sistem modeli belirlenmektedir. Çıktı modelinin adı da verildikten sonra (bu örnek için Şekil 5.6'da gösterildiği gibi "SEAGENTModel.ecore" adı verilmiştir) dönüşüm işlemi başlatılır.



Şekil 5.6. Model dönüşümü çalışma ortamı ayarlarının hazırlanması

Dönüşüm kurallarının işletilmesinden sonra elde edilen SEAGENT Turizm MAS modeli tezin Ek 5 bölümünde verilmiştir. Dönüşüm motoru girdi olarak PIM'in Ecore'unu almış çıktı olarak da PSM'in yine Ecore'unu üretmiştir. Ek 5'te verilen çıktı modelin içerdiği model elemanlarının ve ilişkilerinin görsel bir gösterimi Şekil 5.7'de verilmiştir.



Şekil 5.7 Turizm MAS PIM'inin model dönüşümleri sonrası elde edilen SEAGENT platformuna özgü modeli

“Otel İstemci Etmeni" adlı yazılım etmeni platforma özgü modelde bir SEAGENT “Etmen”’i olarak yer almaktadır ve “Otel İstemcisinin Servis Keşif Planı” adlı planı kapsamında otel rezervasyon servislerini anlamsal yetenek arayüzlerini kullanarak bulmaktadır. Bu planın işletimi sırasında sistemin “Eşleme Etmeni” ile iletişimde bulunur. Eşleme etmeninin “Eşleme Etmeninin Servis Kayıt Planı” adlı kayıt planı PSM’de bir SEAGENT Davranış sınıfı örneği ile temsil edilmektedir. ”Otel İstemci Etmeni”’nin servis keşif planı “Aday Servisi Keşfet” davranış sınıfının bir örneğidir. Bu davranış PIMM’deki “Anlamsal Servis Bulma Planı” üstvarlığının SEAGENT PSMM’indeki karşılığıdır. Benzer şekilde “Otel İstemci Etmeni”’nin servis çalıştırma planı (“Otel İstemcisinin Servis Çağırma Planı”) PSM’de bir “Servisi Çalıştır” davranışı örneğidir. “Servisi Çalıştır” üstvarlığı PIMM’deki “Anlamsal Servis Çalıştırma Planı”’nın SEAGENT PSMM’indeki karşılığıdır.

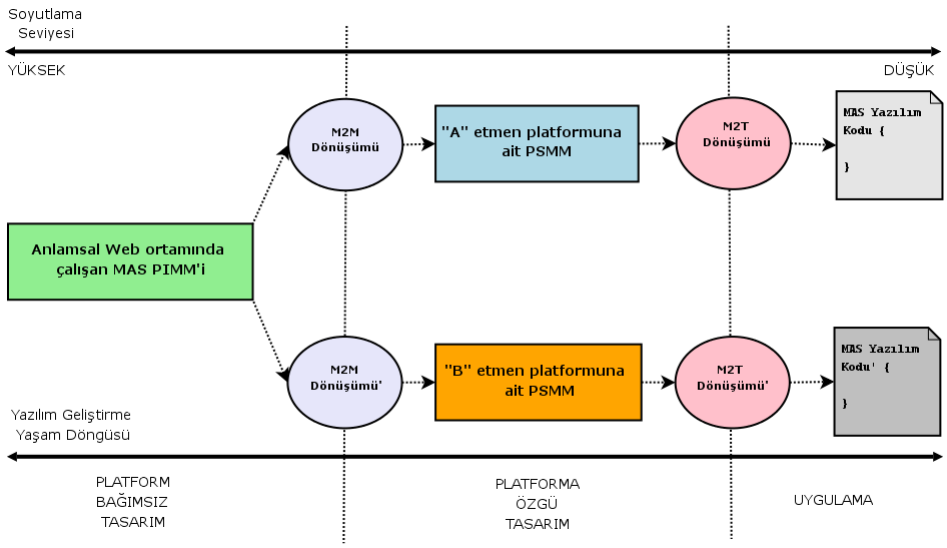
Kaynak modelde tanımlanan etmen davranışları Şekil 5.7’de görüldüğü gibi hedef modelde birer SEAGENT plan eylemi olarak yer almıştır. Öte yandan SEAGENT platformlarında anlamsal web servisleri daha önce de belirtildiği gibi birer OWL-S servisidir. Bu nedenle kaynak modelde yer alan rezervasyon servisi dönüşüm sonrası elde edilen çıktı modelde bir “OWL-S Servisi” örneği ile gösterilmiştir ve beklenen profil, servis çalıştırma ve zemin tanımlamalarına sahiptir.

6 MODELDEN MODELE DÖNÜŞÜMLERİN ÖTESİ: ANLAMSAL WEB ORTAMINDA ÇALIŞAN MAS'LAR İÇİN YAZILIM KODU ÜRETİMİ

Her ne kadar modelden modele dönüşüm MAS geliştirmeyi yüksek bir soyutlama seviyesi sağlayarak kolaylaştırıyor olsa da bu tip sistemlerin gerçek uygulamaları için model dönüşümünün yeterli olmadığı açıktır. Sistem tasarımcılarının kaçınılmaz surette yazılım kodlarını hazırlamaları da gerekmektedir. Model güdümlü olsun ya da olmasın önerilen yazılım geliştirme metodolojilerinin veya süreçlerinin de sistem geliştiricilere yazılım kodu üretiminde yardımcı olacak bir adımı sağlaması önemlidir. Bu noktadan hareketle tezde ortaya konan MDD süreci platforma özgü sistem modellerinden yazılım kodlarının otomatik olarak üretildiği son bir sistem geliştirme adımını içermektedir.

Şekil 6.1'de MDA'ye dayalı olarak tezde önerilen MDD sürecinde Anlamsal Web ortamında çalışan MAS PIMM'i ile çeşitli platformlara ait üstmodeller arasındaki M2M (*"Model-to-Model"*) dönüşümleri ve ilgili platform üstmodellerine uygun olarak yazılım kodu üretimini sağlayan M2T (*"Model-to-Text"*) dönüşümleri gösterilmiştir. M2T dönüşümleri de tıpkı önceki bölümlerde açıklanan M2M dönüşümleri gibi sürecin "tasarım aşamasında" hazırlanmaktadır. Her etmen platformunun PSMM'ini temel alan bu dönüşümler ilgili platformun yazılım geliştirme kütüphanesini kullanarak girdi MAS PSM'leri için şablon seviyesinde sistem kodlarının oluşturulmasını sağlamaktadırlar. Bu seviyede artık platforma özgü yapılar ile çalışıldığından dönüşümlerin tasarlanması ve girdi modeller üzerinde uygulanması bir etmen yazılım çerçevesinden başka bir etmen çerçevesine değişkenlik göstermektedir. En uygun ve arzu edilen yöntem ilgili yazılım çerçevesinin görsel tasarımı destekleyen entegre bir yazılım geliştirme ortamına sahip olması ve sistem

modellerinin burada görsel olarak tasarlanıp şablon seviyede yazılım kodlarının elde edilmesidir. Ancak hem model tabanlı yazılım geliştirmenin yeni bir paradigma olması hem de çoğu MAS yazılımı geliştirme ortamının bu tip entegre yazılım geliştirme ortamlarına sahip olmamasından dolayı birçok durumda PSMM'lerden yazılım kodlarının elde edilmesi işlemini gerçekleştirecek dönüşümlerin de (tıpkı M2M için ATL kullanımı gibi) başka bir dönüşüm üstmodeli ve dili kullanılarak hazırlanması gerekmektedir.



Şekil 6.1. Anlamsal Web ortamında çalışan MAS'ların model güdümlü geliştirilmesine yönelik dönüşüm adımları

Anlamsal web etmen sistemlerinin model güdümlü geliştirilmesi için tezde önerilen sürecin M2T safhalarını somutlaştırmak amacıyla yukarıda sözü edilen her iki yöntem üzerinde de durulmuştur. Hem yazılım araç destekli hem de bir M2T dönüşüm dilinin kullanıldığı yazılım kodu üretme mekanizmaları çalışmalar kapsamında tanımlanmıştır ve uygulamaya geçirilmiştir. Üzerinde çalışılan

SEAGENT platformu için görsel araç destekli, NUIN platformu için ise ayrı bir dönüşüm betikleme dili (*"scripting language"*) kullanılarak M2T dönüşümleri gerçekleştirilmiştir. SEAGENT PSM'lerinden otomatik kod üretilmesi bu bölümde ele alınmıştır. NUIN PSM'lerinden NUIN etmenlerine ait yazılım kodlarının elde edilmesi ise tezin takip eden bölümünde yer almaktadır.

HTN'e (Williamson et al., 1996) dayalı SEAGENT etmen planlarına ait şablon kodların otomatik olarak üretilmesi bir Eclipse (Eclipse Community, 2003) eklentisi kullanılarak gerçekleştirilmektedir. Bu HTN editörü[5] sunduğu grafiksel kullanıcı arayüzü sayesinde SEAGENT kod kütüphanesini kullanarak sistem geliştiricilere yazılım kodu üretmede yardımcı olmaktadır. MAS geliştiricileri etmen plan modellerini bu GUI ortamında hazırlamakta; söz konusu araç da modele uygun bir şekilde etmen planlayıcıları için şablon seviyesinde yazılım kodlarını üretmektedir.

Kod üretimi için kullanılan HTN editörü Eclipse GEF'i ("*Graphical Editing Framework*") Grafiksel Düzenleme Çatısı'nı temel almaktadır. GEF, kullanıcıların var olan bir uygulama modelinden zengin bir grafiksel editör oluşturabilmesine imkan vermektedir (Eclipse Community, 2006). GEF'in içerdiği üç bakış açısı *"Model"*, *"EditParts"* ve *"Figures"* olarak adlandırılmaktadır. Bu bakış açıları popüler yazılım sistem mimarisi desenlerinden biri olan MVC'nin (*"Model-View-Controller"*) (Buschmann et al., 1996) bileşenlerine eşlenmektedir. Şöyle ki; GEF *"Model"*, MVC'nin Model'ine karşılık gelmektedir ve bu model tanımı kullanıcıya bırakılmaktadır. GEF *"EditParts"*, MVC'nin Denetleyici'sine GEF *"Figures"* ise MVC'nin Görünüm'üne karşılık

[5] HTN Editörü açık kaynak kodlu bir Eclipse eklentisi olup güncel sürümüne SEAGENT projesi web sayfasından (http://seagent.ege.edu.tr) erişilebilinir.

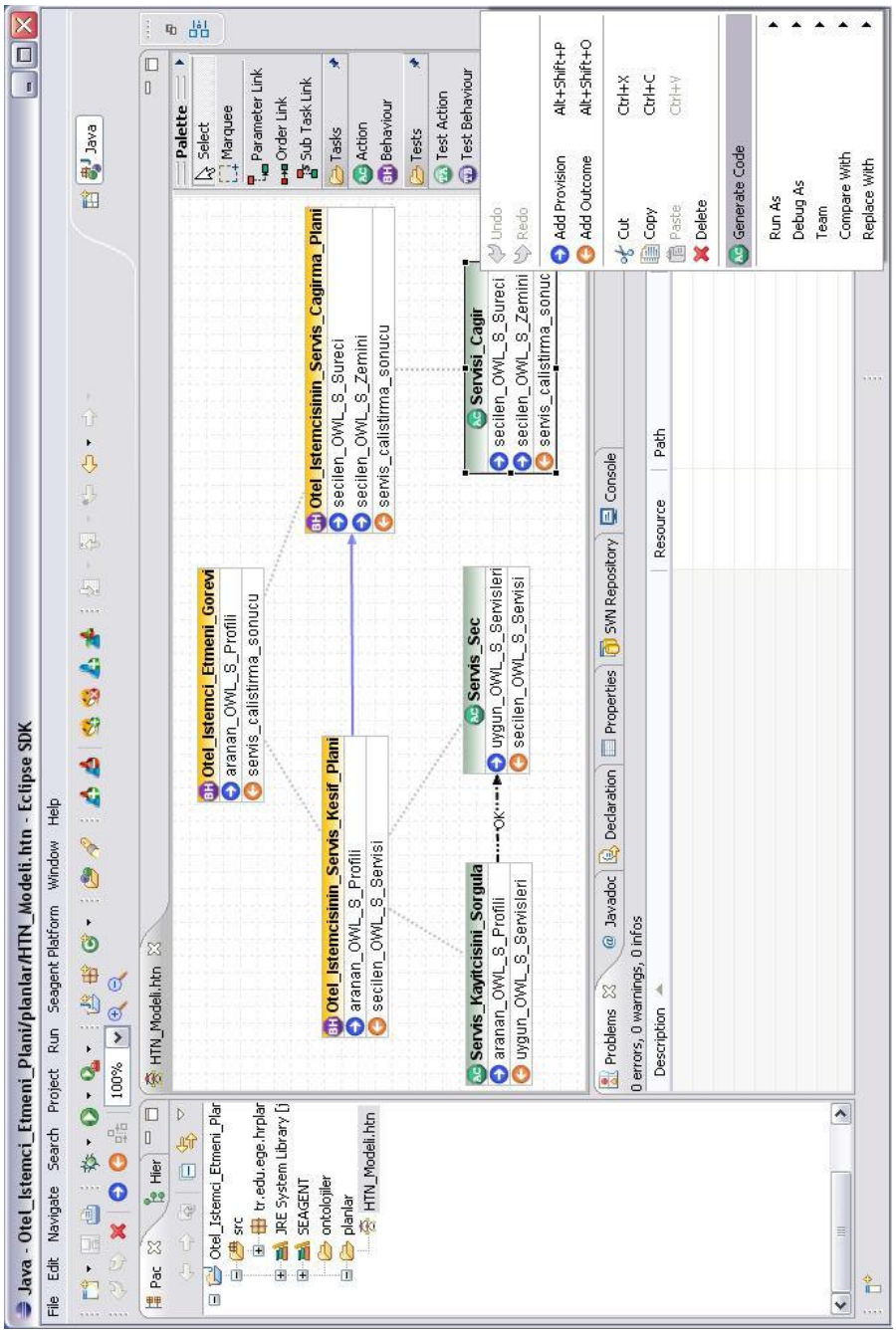
gelmektedir. SEAGENT modelleri için geliştirilen HTN editörü göz önüne alındığında GEF *"Model"*, Davranış, Eylem, İhtiyaç ve İşletim Sonucu gibi SEAGENT planlayıcı üstvarlıklarından oluşmaktadır. GEF *"EditParts"* model ve şekiller arasındaki iletişimi sağlamaktadır. Modelde var olan bir değişiklik *"EditParts"* üzerinden karşılık gelen şekle (*"Figures"*) yansıtılır. Tersi bir durumda da iletişim yine *"EditParts"* üzerinden sağlanır.

Editör AST (*"Abstract Syntax Tree"*) ve Eclipse Java Geliştirme Araçları içerisinde yer alan ilgili ayrıştırıcıyı (*"parser"*) HTN plan kodlarını otomatik olarak üretmek amacıyla kullanmaktadır. Örneğin editörün `ActionCodeGen` adı verilen bileşeni SEAGENT HTN Eylem sınıfları için bir kaynak kodu üreticisidir. Bu üretici bir derleme birimi[6] (*"compilation unit"*) olarak belirli bir kod modelinden yazılım kodu üretimini gerçekleştirmekte ve üretilen kodun bir dosyaya ya da başka bir çıktı akımına (*"output stream"*) yazılmasını sağlamaktadır.

Şekil 6.2'de verilen ekran görüntüsünde editör kullanılarak tezin 5.2.3 bölümünde anlatılan ve model dönüşümleri sonrasında SEAGENT PSMM'e uygun olarak elde edilen Turizm MAS modelinde yer alan "Otel İstemci Etmeni" etmenlerine ait plan kodlarının üretimi gösterilmektedir. Hatırlanacağı üzere kullanıcıları adına otel odası rezervasyonunda bulunmak isteyen ve bu amaçla anlamsal web servisleri ile etkileşimde bulunan etmenlerin hedeflerine yönelik işlettikleri bir plan modelleri vardır. Editör kullanılarak yerine getirilen modelden koda dönüşümler sonrası bu sistemde yer alan etmenler için plan kodları elde edilmektedir.

---

[6] "derleme birimi" bir programın Java programlama dilinde derlenebilecek en küçük parçasını tanımlamaktadır.

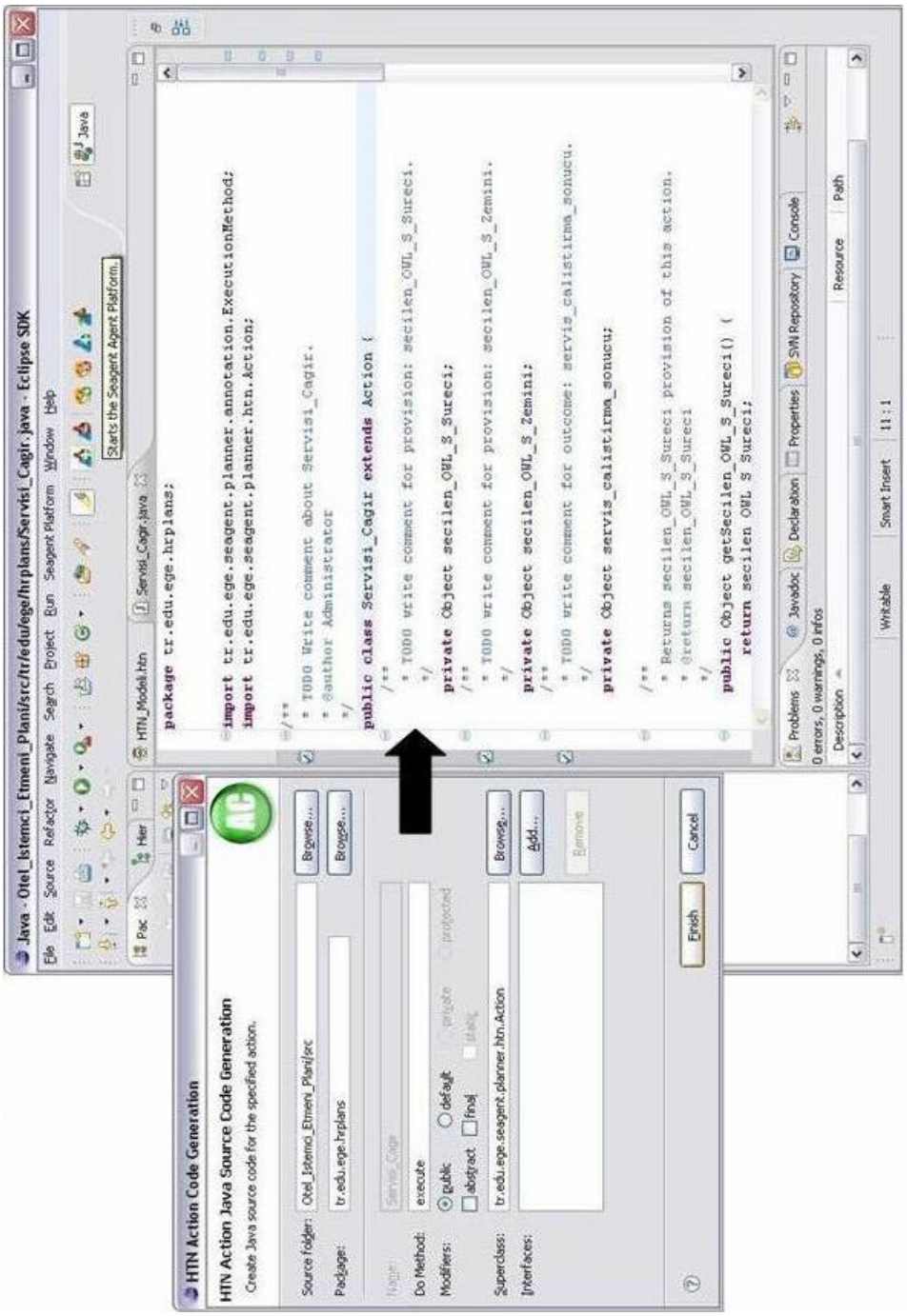


Şekil 6.2. SEAGENT platformlarında çalışan etmenler için plan kodu üretimini sağlayan HTN Planlayıcı Editör'ü

Uygun editör menüleri kullanılarak her Eylem ve Davranış bileşenine karşılık gelen spesifik SEAGENT kodları elde edilebilmektedir. Şekil 6.2'de sağ tarafta görülen Editör paleti kullanılarak Plan bileşenlerinin modifiye edilmesi de (örneğin yeni bir plan ihtiyacının ya da işletim sonucunun eklenmesi, hatta yeni bir plan adımının eklenmesi) mümkündür.

Şekil 6.2'deki ekran görüntüsünde "Servisi_Cagir" adı verilen bir eylem için SEAGENT kodunun üretilmesi örneklenmiştir. Üretilen yazılım kodu dosyası için kaynak dizinin ve paket adının girilmesi gibi kod üretim sürecine ait ayarların bir diyalog penceresi (Şekil 6.3'te solda) aracılığı ile yerine getirilmesinden sonra kod üretici model elemanını ayrıştırmakta, işletim sonucu, ihtiyaç veya daha başka ilişkileri belirlemekte ve sonunda ilgili eylem için Şekil 6.3'te sağ tarafta gösterildiği gibi şablon kodlarını üretmektedir. Yazılım geliştirici örneğin üretilmiş Eylem metotlarının içini doldurarak kodu tamamlamaktadır. Benzer şekilde tüm eylemler için kod üretimi tamamlanır. Hiyerarşiye uygun olarak sıradaki adım, ilgili eylemlerden oluşan etmen davranışlarına ait kodların üretilmesidir. Bir davranışa ait kodların üretilmesinden önce bu davranışın eylemlerine ait kodların üretilmesi şarttır. Aksi takdirde editör kullanıcıyı uyarmaktadır. Sisteme has ayarların yapılmasından sonra davranış kodları ilgili davranışın alt-görevlerinin (eylemler) birbirleri ile olan ihtiyaç – işletim sonucu ilişkileri göz önüne alınarak üretilmektedir. Etmen planına ait daha üst seviyedeki yapılar için de kod üretimi benzer şekildedir.

Yazılım geliştirici, üretilmiş olan etmen plan kodlarının işletimini SEAGENT Eclipse eklentisini kullanarak başlattığı bir SEAGENT MAS platformu içerisinde test edebilmektedir.



Şekil 6.3. “Servisi_Cagir” adlı bir SEAGENT Eylem sınıfı için yazılım kodunun üretilmesi

7 MODEL GÜDÜMLÜ MAS GELİŞTİRME SÜRECİNİN DEĞERLENDİRİLMESİ

Model güdümlü yazılım geliştirme tekniklerinin hem standart değerlendirme kriterlerinin belli olmamasından hem de MDD'nin deneysel (*"empirical"*) çalışmalarında insan faktörünün etkisinin şu an için biçimsel olarak ifade edilememesinden dolayı bu tip yazılım geliştirme yöntemlerinin değerlendirilmesinde net bir metodoloji henüz ortaya konulamamıştır.

Değerlendirmeyi bir bütün olarak gerçekleştirmeyi hedeflemese de özellikle üstmodellemenin ve model dönüşümünün değerlendirilmesini sağlamak amacıyla bazı kalite metriklerinin belirlenmesine yönelik literatürde yakın zamanda gerçekleştirilmiş çalışmalar bulunmaktadır (Solheim and Neple, 2006; Mohagheghi and Aagedal, 2007). Solheim ve Neple (Solheim and Neple, 2006) MDD için iki kalite kriteri önermişlerdir: Dönüşebilirlik ve Değiştirilebilirlik. Dönüşebilirlik ile modellerin başka modellere mümkün olan en uygun seviyede dönüşebilmesi ve mümkünse yazılım kodlarının elde edilmesi gerekliliği ifade edilmektedir. Değiştirilebilirlikten kasıt ise sistem ihtiyaçlarında yer alan değişikliklerin uygun bir biçimde modele ve yazılım koduna yansıtılabilmesidir. Mohagheghi ve Aagedal, modelleme kalitesinin farklı bakış açılarını içermesi gerektiğini savunmuşlardır (Mohagheghi and Aagedal, 2007). Bu bakış açıları dönüşüm dillerinin karmaşıklığı, modellerin dönüşebilirliği ve MDD araçlarının yetenekleri gibi teknik faktörleri, öğrenebilirlik, dönüşüm diline aşina olma ve yorumlama kolaylığı gibi psikolojik faktörleri, kullanılabilirlik ve estetik gibi insan-bilgisayar etkileşimi faktörlerini ve modellemenin iş alanı ve amaçları gibi organizasyonel faktörleri içermektedir. Ancak bu kalite bakış açılarının ölçüm metrikleri ise ilgili çalışmalarda net değildir.

Model güdümlü MAS geliştirme için tezde önerilen yöntemin değerlendirilmesini sağlamak amacıyla başlangıçta yukarıda anlatılan yaklaşımlara benzer bir değerlendirme yönteminin hazırlanıp uygulanması yoluna gidilmiştir. Özellikle kod merkezli geliştirme süreci ile yeni model merkezli geliştirme sürecinin karşılaştırılması ve avantaj ve dezavantajların ortaya konulması düşünülmüştür. Önerilen MDA yaklaşımının MAS geliştirmede nasıl bir kullanılabilirlik ve etkinlik getireceğinin değerlendirilmesi hedeflenmiştir. Bu amaç doğrultusunda SEAGENT çerçevesinin kullanıldığı bir araştırma geliştirme projesinin analiz ve tasarım safhalarında tezde önerilen PIMM ve model dönüşümlerinin uygulama geliştiriciler tarafından kullanılması, yeni yaklaşıma adaptasyon ve öğrenme eğilimlerinin ölçülmesi ve bu kullanıcılardan geri bildirimlerin biçimsel bir biçimde (form doldurma, anket, röportaj vb.) elde edilip değerlendirilmesi düşünülmüştür. Ancak söz konusu kriterleri standart bir biçimde ölçecek metriklerin yukarıda da değinildiği gibi henüz bulunmaması ve buna dayalı olarak elde edilen bulguların biçimsel gösterimlerinin gerçekleştirilememesi bu tip bir değerlendirmenin sağlıklı olarak yerine getirilmesini mümkün kılmamıştır. Öte yandan tezde önerilen MDD sürecinin tanıtıldığı konferanslarda fikirlerini beyan eden bilim adamları ve yine sürecin anlatıldığı çeşitli bilimsel yayınları değerlendiren hakemlerin ortak görüşü bu tip bir model güdümlü MAS geliştirme yönteminin değerlendirilmesi ve özgün değerinin ortaya konulması için yukarıda anlatılan yaklaşım yerine sürecin farklı platformları desteklediğini ve uygulanabilirliğini gösteren deneysel bir çalışmanın daha uygun olacağı yönünde olmuştur. Tüm bunların sonucunda önerilen MDD sürecinin SEAGENT dışında ve başka araştırmacılar tarafından ortaya konan bir MAS geliştirme çatısı için uygulanması ve sürecin deneysel bir biçimde değerlendirilmesi yerine getirilmiştir.

Her ne kadar yukarıda sözü edilen değerlendirme çalışmasının gerçekleştirilmesi kullanıcı geribildirimlerine dayalı değerlendirmeye göre daha zor ve zaman alıcı olsa da özellikle Anlamsal Web ortamında çalışan MAS PIMM'ine dayalı etmen modellerinin farklı platformlar için dönüştürülebiliyor olmasının gösterimi (Solheim and Neple, 2006) tezde önerilen MAS geliştirme yaklaşımının bölüm 2.2.3'te de ifade edildiği gibi literatürdeki benzer çalışmalardan ayırt edilebilmesini sağlayan önemli özelliklerinden biri olmuştur.

Değerlendirmenin yerine getirilmesi amacıyla ilk olarak SEAGENT harici başka bir etmen yazılım çerçevesinin ve bu çerçeveye ait üstmodelin belirlenmesi gerekmiştir. Ancak bu noktada karşılaşılan en büyük güçlük halihazırda SEAGENT gibi yerleşik olarak Anlamsal Web desteğine sahip başka bir MAS geliştirme çerçevesinin bulunamaması olmuştur. Ancak kısmi olarak destek veren başka yazılım çerçeveleri göz önünde bulundurulmuştur. Ortaya çıkan bu ihtiyaç için Dickinson ve Wooldridge'in önerdiği NUIN (Dickinson and Wooldridge, 2003) platformu incelenmiş ve bu platformda çalışan MAS'ların geliştirilmesi amacıyla tezde önerilen MDD süreci uygulanmıştır.

NUIN (Dickinson and Wooldridge, 2003), Anlamsal Web uygulamalarında çalışacak etmenlerin BDI prensiplerine uygun olarak tasarlanıp hayata geçirilmesini hedefleyen esnek bir etmen mimarisidir. NUIN çerçevesi üzerine inşa edilen etmen uygulamaları belki de bilinçli (*"deliberative"*) etmenler için üzerinde en çok çalışılan BDI (Rao and Georgeff, 1995) teorik mimarisine dayanmaktadır. Bir BDI etmeni hem ortama ilişkin bilgiyi modellemekte ve planlama yaparak amaç-yönelimli davranışta bulunabilmekte hem de ortamdan elde ettiği algılar çerçevesinde gerektiğinde yeniden planlama yapabilmektedir. Böylece, etmenlerin amaç-yönelimlilik ve karşıt-eylemlilik özellikleri birleştirilmiş olmaktadır.

Öte yandan NUIN, Anlamsal Web notasyonunu ve temsillerini etmenlerin tasarımının ana unsurları haline getirmeyi hedeflemektedir. (Dickinson, 2006)'da da belirtildiği gibi *anlamsal web etmeni* fikri beraberinde birçok yorumlamayı getirse de NUIN etmenleri bunlardan sadece aşağıda listelenmiş olanları desteklemektedir:

- NUIN etmenleri RDF (W3C, 2004) kullanılarak yapılandırılmaktadır ve prensipte, sorgulara karşılık olarak bu yapılandırmalarını raporlayabilirler.
- NUIN etmenleri anlamsal web bilgi kaynaklarını etmen yapılandırmasında JENA (JENA, 2003) model tanımlarını içermek suretiyle birer etmen kanı merkezi olarak kullanabilir ve harici anlamsal web bilgi kaynaklarını sorgulamak için kısıtlı yetenekleri vardır.
- NUIN dili anlamsal web kavramını tüm sembolik sabitlerin birer URI olması önşartı ile desteklemektedir.

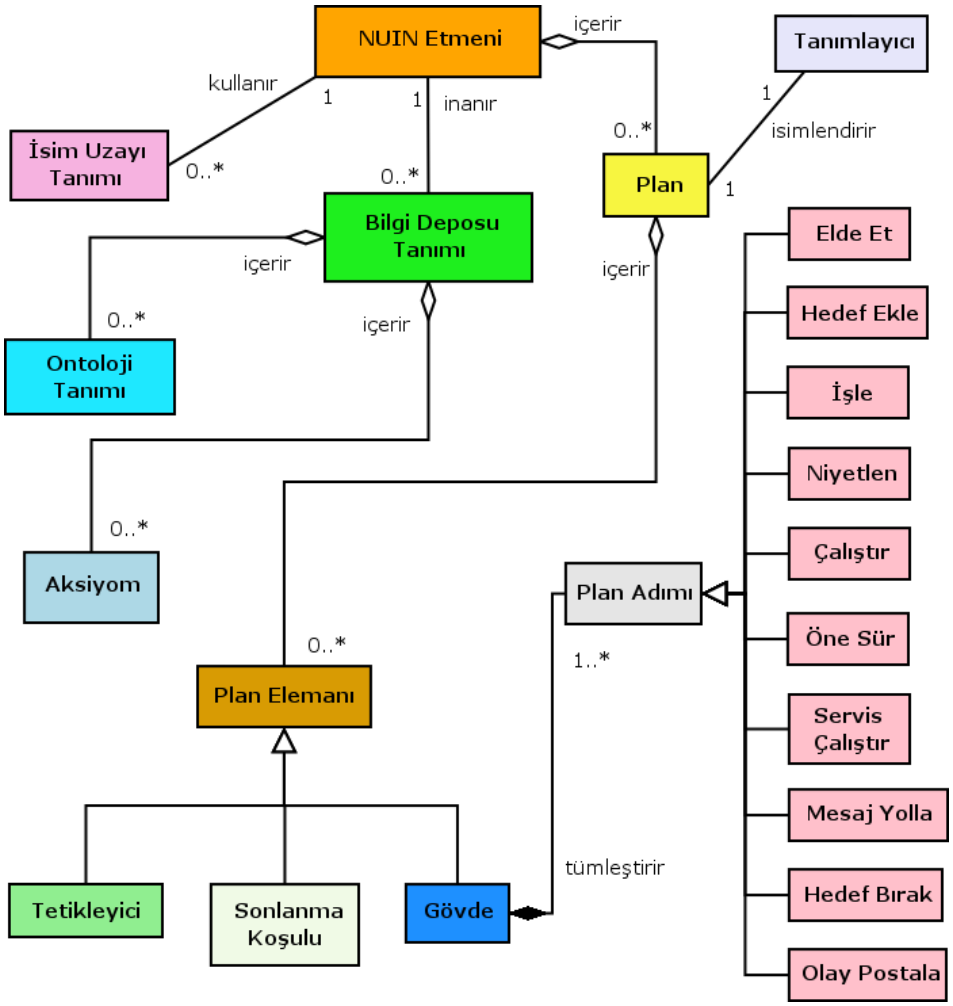
NUIN çerçevesinin yukarıda yer alan tasarım temelleri ve bu tez çalışması ile de ortak "anlamsal web etmeni" vizyonu NUIN'in tezde önerilen MDD sürecinin değerlendirilmesinde bir başka alternatif MAS geliştirme ortamı olarak seçilmesine neden olmuştur. Bu amaçla ilk olarak NUIN platformuna özgü etmen üstmodelinin çıkarılması yerine getirilmiştir. Daha sonra tezde önerilen MAS PIMM'i ile bu üstmodel arasında M2M dönüşümleri tanımlanmış ve daha önce SEAGENT (Dikenelli et al., 2005) çerçevesi için yapıldığı gibi dönüşümlerin ATL (Jouault and Kurtev, 2006) ve ilgili geliştirme ortamı kullanılarak hayata geçirilmesi sağlanmıştır. Modelden modele dönüşümler tamamlandıktan sonra NUIN üstmodeline bağlı olarak NUIN ortamı için yazılım kodu üretimi gerçekleştirilmiştir. Böylece PIMM'e uyan bir MAS modelinin NUIN'de uygulanması MDD süreci sonrasında gerçekleştirilmiştir.

Aşağıdaki alt bölümlerde NUIN'de BDI etmenlerinin MDD ile geliştirilmesi için yerine getirilen çalışmalar detaylandırılmıştır. Sürecin uygulanışını örneklemek amacıyla SEAGENT uygulamasında olduğu gibi yine tezde önerilen PIMM'i üstmodel olarak alan ve Bölüm 4.3'te anlatılan turizm MAS platform bağımsız modeli kullanılmıştır.

7.1 NUIN MAS Geliştirme Çerçevesine ait PSMM'in Oluşturulması:

NUIN, sistem geliştiricilere etmen sistemlerinin oluşturulması için bir Java Uygulama Programlama Arayüzü sunmaktadır. Bu arayüzün içyapısında etmen planları Java nesneleri ile gösterilmektedir ve eylem yapıcılarının (*"constructor"*) direkt çağrılması gibi çeşitli yöntemlerle bu planlar üretilebilmektedir. Ancak (NUIN, 2006)'da da belirtildiği gibi NUIN ortamındaki etmenleri programlamak için en uygun yöntem *Nuinscript* adı verilen bir dilin kullanılmasıdır. Bu dil NUIN yorumlayıcısı tarafından Java nesnelerine ayrıştırılmaktadır. Bu nedenle ilk olarak Nuinscript etmen betik (*"script"*) yazım diline ait (Dickinson, 2006)'daki EBNF (*"Extended Backus-Naur Form"*) tanımına uygun bir NUIN üstmodeli türetilmiştir. Bu üstmodel bir PSMM olarak M2M ve M2T dönüşümlerinde kullanılmıştır. Şekil 7.1'de bu üstmodel görülmektedir.

Şekil 7.1'deki üstmodele dayanan ve ilerleyen bölümlerde anlatılan model dönüşümlerinin ve uygulamalarının anlaşılabilmesi için Nuinscript dilinin üstvarlıkları ve birbirleri arasındaki ilişkiler hakkında aşağıda bilgi verilmiştir. Nuinscript hakkında daha detaylı bilgiye (Dickinson, 2006)'dan ve tam bir betik tanımlamasına da (NUIN, 2006)'dan erişilebilinir.



Şekil 7.1. Nuinscript etmen betik yazım dilinin üstmodeli

Bir Nuinscript betiği bir NUIN Etmeni (*"NUIN Agent"*) konfigürasyonuna ait isim uzayı ve bilgi deposu (*"knowledgestore"*) tanımlarını (*"declaration"*) ve planlarını içermektedir. Bir etmene ait betikte yer alan İsim Uzayı Tanımları'nda (*"Namespace Declaration"*), Anlamsal Web bilgi kaynakları ile kolay birlikte işlerliği sağlamak amacıyla isim uzaylarına ait URI'ler verilmektedir. Öte yandan bir betik, etmenin bilgi tabanı için Bilgi Deposu Tanımları (*"Knowledgestore*

Declaration") içerisinde bir dizi başlangıç verisi koleksiyonunu içerebilmektedir. Bu koleksiyonlar etmenin yaşadığı çevreye ait inanışlarını (kanılarını) oluşturmaktadırlar. Bir bilgi deposu bir veya daha fazla ontoloji ile ilişkilendirilebilir. Her bir Ontoloji Tanımı (*"Ontology Specification"*) bir URI ile tanımlanmaktadır. Ayrıca bir etmene ait bilgi deposu, etmen çalışmaya başlamadan önce kendisine eklenmiş bir dizi başlangıç cümlesini tanımlayabilir. Bu cümleler birer Aksiyom (*"Axiom"*) olarak gösterilmektedir.

Şekil 7.1'de görüldüğü gibi NUIN etmenlerinin gösterdikleri yüksek seviye davranışlar *Plan*'larla ifade edilmektedir. Planlar Tanımlayıcı'larla (*"Identifier"*) isimlendirilirler. Bu tanımlayıcılar bir etmen kapsamında tektirler. Aynı isimde iki plan aynı etmen tanımı içerisinde yer alamaz. Her plan bir ya da daha fazla Plan Elemanı (*"Plan Element"*) içermektedir. Bu elemanlar bir Tetikleyici (*"Trigger"*), Sonlanma Koşulu (*"Post Condition"*) veya Gövde (*"Body"*) olabilir. Bir Tetikleyici etmenin algılayabileceği olaylar üzerine bir deseni ifade eder. Sonlanma Koşulu ise ilgili planın uygulanmasından sonra etmenin içinde bulunması gereken durumu ifade eden bir mantıksal cümledir.

Bir etmen planına ait Gövde, eylemleri temsil eden Plan Adımları'nın (*"Plan Step"*) birleşiminden oluşmaktadır.Bir etmenin yapmak istedikleri (niyeti ya da isteği) etmenin gerçekleştireceği bir dizi eylemle veya yerine getirmeye çalışacağı mantıksal bir amaç ile veya bu ikisinin karışımı ile temsil edilmektedir (NUIN, 2006). NUIN platformuna özgü önceden tanımlı bazı plan adımı örnekleri de (örneğin Hedef Ekle (*"Add Goal"*), Mesaj Yolla (*"Send Message"*), Servis Çağır (*"Invoke Service"*), vb.) yine NUIN yazılım kütüphanesinde yer almaktadır ve bu örnekler üstmodel şeklinin sağ kısmında gösterilmiştir.

7.2 Modelden Modele Dönüşümlerin Tanımlanması ve Uygulanması

Anlamsal Web ortamında çalışan MAS'ların model dönüşümleri sonrası elde edilmesi için tezde önerilen PIMM ve hedef platformların PSMM'lerinin üstvarlıkları arasındaki eşlemelerin yerine getirilmesi gerekmektedir. Bölüm 5.1'de SEAGENT MAS çerçevesi için anlatılan varlık eşlemelerine benzer bir biçimde NUIN için de PIMM ve NUIN PSMM üstvarlıkları arasında eşlemelerin tanımlanması ve bu eşlemelere uygun olarak dönüşüm kurallarının yazılması gerekmiştir. Tablo 7.1'de bu varlık eşlemeleri görülmektedir.

Tablo 7.1. PIMM ve NUIN PSMM'i arasındaki varlık eşlemeleri

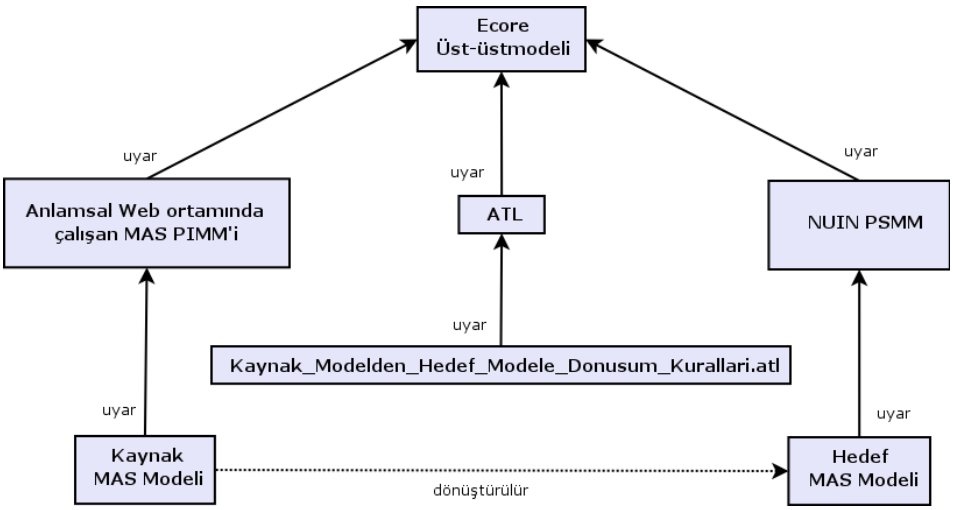
PIMM Varlığı	NUIN Varlığı	Açıklama
Anlamsal Web Etmeni, Anlamsal Servis Eşleme Etmeni	NUIN Etmeni	Bir NUIN Etmeni, anlamsal yetenekli MAS'lara ait PSMM'deki etmen varlıklarına karşılık gelir.
Ontoloji, Rol Ontolojisi, Servis Ontolojisi, Organizasyon Ontolojisi	Ontoloji Tanımı	PIMM'de yer alan Ontoloji ve altsınıfları NUIN platformunda ontolojik bilgi deposu tanımları ile temsil edilir.
Plan, Anlamsal Servis Kaydetme Planı, Anlamsal Servis Bulma Planı, Anlamsal Servis Çalıştırma Planı	Plan Tanımlayıcı Gövde	Anlamsal servis etkileşimine ait etmen planları NUIN Plan'ları ve bunların Tanımlayıcı'ları olarak hayata geçirilebilir. Her plan için planın adımlarını içerecek bir Gövde elemanı oluşturulmaktadır.
Davranış	Plan Adımı	Platform bağımsız modellerde tanımlanan etmen davranışları NUIN platformunda Plan Adım'ları ile gösterilmektedir.

Tablo 7.1'de görüldüğü üzere, PIMM'de yer alan etmen varlıkları NUIN hedef üstmodelindeki NUIN etmenlerine eşlenmiştir –ki bu beklenen bir eşlemedir. NUIN ortamlarında bilgi depoları olarak kullanılan Ontoloji Tanımları PIMM'de tanımlı olan Ontoloji ve ontoloji uzantılarına karşılık gelmektedir. Öte yandan bir platform bağımsız modelde yer alan Plan örneklerinin her biri kavramsal olarak hedef modelde bir NUIN Plan ile temsil edilir. Ancak gerçekte bu elemanlar arasındaki dönüşüm basit bir *Plan'dan Plan'a dönüşüm* değildir. PIM'de yer alan her Plan için karşılık gelen bir NUIN Plan elemanının oluşturulması yanında bu planın Tanımlayıcı'sının ve ilgili plana ait adımlar için bir kap (*"container"*) olan Gövde'sinin de oluşturulması gerekmektedir. Bu nedenle Plan elemanlarının dönüşümü PIMM'in Plan üstvarlığı ile NUIN üstmodelindeki Plan, Tanımlayıcı ve Gövde elemanları arasındaki eşlemeleri içermektedir. PIMM'de yer alan etmen davranışları ise NUIN üstmodelinin Plan Adımı varlıklarına eşlenmektedir.

Varlık eşlemelerine bağlı olarak model dönüşüm kurallarının yazılması SEAGENT PSMM'de olduğu gibi yine ATL (Jouault and Kurtev, 2006) kullanılarak gerçekleştirilmiştir. Model dönüşümlerinin genel görünümü Şekil 7.2'de verilmiştir ve görüldüğü gibi dönüşüm süreci Bölüm 5.2.1'deki anlatılan yapının aynısına sahiptir. Farklı olan tek bileşen üzerinde dönüşüm uygulanacak modellerin üstmodelinin NUIN PSMM olmasıdır.

ATL motorunu kullanmak için PIMM'in yanı sıra NUIN PSMM'inin de EMF kodlamasının (*"ecore encoding"*) hazırlanması gerekmiştir. Bu nedenle NUIN PSMM'inin KM3 (Jouault and Bezivin, 2006) kullanılarak metinsel gösterimi hazırlanmış ve ADT Ecore enjektörü kullanılarak bu üstmodelin de Ecore gösterimi elde edilmiştir. Önceki dönüşüm çalışmalarından PIMM'in ve örnek turizm MAS

modelinin Ecore'ları hazır olduğundan buradaki dönüşümlerde tekrar hazırlanmalarına gerek kalmamıştır.



Şekil 7.2. NUIN PSMM için ATL kullanılarak gerçekleştirilen modelden modele dönüşümler

NUIN PSMM üstvarlıklarına ait Ecore gösterimlerinin tamamı tezin Ek 6 bölümünde verilmiştir. Örnekleme amacıyla burada tezin 5.2.1. bölümünde olduğu gibi PIMM'de Anlamsal Web Etmeni'nin NUIN eşleniği olan "NUIN Etmeni" üstvarlığına ait KM3 ve Ecore gösterimleri verilmiştir. Aşağıda "NUIN Etmeni"'nin KM3'e uygun gösterimi yer almaktadır:

```
class NUINAgent {
      attribute name: String;
      reference use[0-*]: NamespaceDeclaration oppositeOf
                usedBy;
      reference believe[0-*]: KnowledgestoreDeclaration
                oppositeOf believedBy;
      reference contain[0-*] ordered container: Plan
                oppositeOf containedBy;
```

```
    }
```

"NUIN Etmeni"'nin üstmodeldeki diğer elemanlarla olan her ilişkisi için ilgili ilişkide "NUIN Etmeni"'nin üstlendiği rol adı ve örnek sayısı verilmiştir. Her ilişkideki "*oppositeOf"* etiketinden sonra ilişkinin karşı tarafındaki üstvarlığın üstlendiği rol adı yer almaktadır. Yukarıdaki KM3 biçimindeki üstmodel elemanı tanımı Ecore'a çevrildiğinde ise aşağıdaki gösterim elde edilmiştir:

```
<eClassifiers xsi:type="ecore:EClass" name="NUINAgent">
   <eStructuralFeatures xsi:type="ecore:EAttribute"
      name="name" ordered="false" unique="false"
      lowerBound="1" eType="/1/String"/>
   <eStructuralFeatures xsi:type="ecore:EReference"
      name="use" ordered="false" upperBound="-1"
      eType="/0/NamespaceDeclaration"
      eOpposite="/0/NamespaceDeclaration/usedBy"/>
   <eStructuralFeatures xsi:type="ecore:EReference"
      name="believe" ordered="false" upperBound="-1"
      eType="/0/KnowledgestoreDeclaration"
      eOpposite="/0/KnowledgestoreDeclaration/believedBy"/>
   <eStructuralFeatures xsi:type="ecore:EReference"
      name="contain" upperBound="-1" eType="/0/Plan"
      containment="true" eOpposite="/0/Plan/containedBy"/>
</eClassifiers>
```

Model dönüşümlerinde kullanılacak NUIN PSMM'in Ecore'u hazırlandıktan sonra sürecin "tasarım aşaması" işlemlerinden sonuncusu olan varlık eşlemelerine dayalı model dönüşüm kurallarının uygun sözdiziminde yazılması gerçekleştirilmiştir. ATL kullanılarak hazırlanan bu dönüşüm kuralları tezin Ek 7 bölümünde yer almaktadır. Dikkat edilirse Ek 7'de verilen ATL yardımcı kurallarının SEAGENT PSM'leri için hazırlananlarla (Ek 3) aynı olduğu görülmektedir. Çünkü bu kurallar daha önce de belirtildiği gibi PIM'lerdeki model elemanlarının seçimini

ve birbirleri ile olan ilişkilerin belirlenmesini sağlamaktadır ve dönüşümü gerçekleştiren asıl kurallar tarafından kullanılmaya uygun modüler bir yapıya sahiptirler. Böylelikle her bir PIMM için sadece bir kez yazılmaları yeterlidir ve farklı PIMM – PSMM dönüşümleri için tekrar tekrar yazılmalarına gerek yoktur. Bu da önerilen MDD sürecinin önemli avantajlarından birini teşkil etmektedir.

PIMM ve NUIN PSMM üstvarlıkları arasında dönüşümü sağlayan kuralların Bölüm 5.2.2'de anlatılan SEAGENT PSMM dönüşümleri göz önüne alındığında özellikle kurallardaki hedef model elemanı sayısı açısından daha fazla çeşitlilik gösterdikleri görülmektedir. Tezin Ek 7 bölümünde tamamı verilen bu kuralları örneklemek amacıyla burada PIMM "Davranış", "Anlamsal Web Etmeni" ve "Anlamsal Servis Bulma Planı" üstvarlıklarının NUIN PSMM karşılıklarını elde etmeyi sağlayan dönüşüm kuralları verilmiştir.

Tablo 7.1'de görüldüğü gibi platform bağımsız modellerdeki "Davranış" elemanlarının NUIN PSM'lerindeki karşılıkları "Plan Adımı" tipindeki elemanlardır. Bu iki üstvarlık arasındaki dönüşümü sağlayan `Behavior2PlanStep` adlı dönüşüm kuralı aşağıdadır:

```
1 rule Behavior2PlanStep {
2   from
3     bhv: Agent!Behavior(
4          bhv.partofPatternBehavior
5     )
6   to
7     ps: NUIN!PlanStep (
8         name <- bhv.name
9     )
10 }
```

Kural, birebir eşlemeyi gerçekleştiren bir ATL kuralıdır. Kuralın 4. satırında çağrılan `partofPatternBehavior` yardımcı kuralı üzerinde

çalışılan girdi model elemanının bir PIMM "Davranış" elemanı olup olmadığını belirlemektedir. Eğer kuralın üzerinde işletildiği eleman bir "Davranış" elemanı ise bu durumda bir NUIN "Plan Adımı" örneği oluşturulur ve ilgili özellik atama işlemi gerçekleştirilir (6 ve 9. satırlar arası).

Dönüşüm kurallarına ikinci bir örnek ise PIMM "Anlamsal Web Etmeni" varlıklarının NUIN PSMM "NUIN Etmeni" varlıklarına dönüşümünü sağlayan `SemanticWebAgent2NUINAgent` adlı dönüşüm kuralıdır. Bu kural aşağıda verilmiştir:

```
1 rule SemanticWebAgent2NUINAgent {
2  from
3   ag: Agent!SemanticWebAgent (
4       ag.partofPatternforWebAgent
5   )
6  to
7   na: NUIN!NUINAgent (
8     name <- ag.name,
9     contain <- Sequence{ag.executorPlans,ag.finderPlans},
10    believe <- ksdec
11  ),
12  ksdec: NUIN!KnowledgestoreDeclaration
13 }
```

Bir önceki kurala göre daha karmaşık yapıda olan bu kuralda bir kaynak model elemanından iki hedef model elemanının oluşturulması söz konusudur. Girdi modeldeki her bir "Anlamsal Web Etmeni" model elemanına karşılık bir "NUIN Etmeni" elemanı üretirilirken aynı zamanda bu etmene ait "Bilgi Deposu Tanımı" da oluşturulmaktadır. Kuralın 4. satırında çağrılan `partofPatternforWebAgent` yardımcı kuralı Bölüm 5.2.2'de anlatıldığı gibi üzerinde çalışılan kaynak modelde Anlamsal Web Etmenleri'nin belirlenmesini sağlamaktadır. Kuralın 9. satırında çağrılan `finderPlans` ve `executorPlans` yardımcı kuralları

ise ilgili Anlamsal Web Etmeni'nin planlarını girdi model üzerinde belirlemekte ve etmen ile planları arasındaki ilişkinin çıktı modelinde de korunmasını sağlamaktadır. Kuralın 10.satırındaki atama sonucunda ise 12. satırda oluşturulan "Bilgi Deposu Tanımı" referansının etmenin ilgili özelliğine atanması gerçekleştirilmektedir. Böylelikle her bir NUIN Etmeni, inanışlarını temsil eden Bilgi Deposu Tanımı ile birlikte kuralın işletilmesi sonucunda oluşturulmaktadır. Ek 7'de verilen ontoloji dönüşüm kuralı incelendiğinde burada oluşturulan Bilgi Deposu Tanımlarının içeriklerinin söz konusu ontoloji dönüşümleri sonrasında doldurulduğu görülmektedir.

PIMM ve NUIN PSMM arasındaki model dönüşüm kuralları için verilecek bir diğer örnek ise PIMM "Anlamsal Servis Bulma Planı" üstvarlıklarının NUIN PSMM "Plan" karşılıklarını elde etmeyi sağlayan `SemanticServiceFinderPlan2Plan` adlı dönüşüm kuralıdır. Bu kural aşağıda verilmiştir:

```
1 rule SemanticServiceFinderPlan2Plan {
2   from
3    fndpln: Agent!SemanticServiceFinderPlan (
4       fndpln.partofPatternforFinderPlan
5     )
6   to
7     plnf: NUIN!Plan (
8      containedBy <- Agent!SemanticWebAgent->
9      allInstances()->asSequence()->select(agt|agt.apply->
10     exists(pln|pln=fndpln))->first()
11    ),
12    identifier: NUIN!Identifier (
13      id <- fndpln.name,
14      name <- Agent!SemanticServiceFinderPlan->
15      allInstances()->asSequence()->
16      select(pln|pln=fndpln)->first()
17    ),
18    plnfBody: NUIN!Body (
19      name <- 'Finder Plan Body',
20      containedBy <- plnf,
21      compose <- Sequence{Agent!Behavior->
```

```
22      allInstances()->asSequence()->
23      select(bhv|bhv.execute->exists(pln|pln=fndpln))}
24    )
25 }
```

Tablo 7.1'de verilen varlık eşlemelerine göre bu kuralda kaynak bir modeldeki "Anlamsal Servis Bulma Planı" elemanlarına karşılık hedef modelde NUIN "Plan", "Tanımlayıcı" ve "Gövde" elemanlarının oluşturulduğu görülmektedir. Bu açıdan söz konusu kural yukarıda örnek verilen diğer iki kurala ve daha önce tezin 5.2.2. bölümünde üzerinde durulan dönüşüm kurallarına (örneğin bu kuralın SEAGENT PSMM'i için eşleniği olan `SemanticServiceFinderPlan2DiscoverCandidateService` adlı kural) göre daha karışık bir yapıya sahiptir. PIMM Plan'larının NUIN PSMM'inin "Plan", "Tanımlayıcı" ve "Gövde" üstvarlıklarına eşlenmesinden dolayı kuralın bir kaynak kısmı fakat üç hedef kısmı bulunmaktadır. Her " Anlamsal Servis Bulma Planı" için önce bir NUIN Plan ve bunun Tanımlayıcı'sı oluşturulmakta ve ilişki dönüşümleri gerçekleştirilmektedir (7 ve 17. satırlar arası). İlgili planı çalıştıran etmenin belirlenmesi ve çıktı modelde de bu ilişkinin korunması amacıyla kuralın 8 ve 10. satırları arasındaki OCL sorgusu işletilmektedir. Her bir NUIN Planı'nın ismi modelde bir "Tanımlayıcı" ile saklandığından kaynak modeldeki ilgili plan adı bu tanımlayıcıya aktarılmakta (satır 13) ve oluşturulan plan model elemanı ile bu tanımlayıcı model elemanı arasındaki isimlendirme ilişkisi hazırlanmaktadır (14 ve 16. satırlar arası).

Kaynak modeldeki plana ait davranışlara karşılık gelen NUIN plan adımlarını kapsayan bir NUIN Gövdesi yukarıdaki kuralın 18 ve 24. satırları arasında oluşturulmaktadır. Planın içerdiği plan adımlarının belirlenmesini sağlayan 21 ve 23. satırlar arasındaki OCL sorgusu ile kaynak modelde söz konusu "Anlamsal Servis Bulma Planı" model elemanının içerdiği "Davranış"'lar belirlenmektedir. Tablo 7.1'de

görüleceği üzere PIMM "Davranış" üstvarlıklarının NUIN PSMM karşılıkları "Plan Adımları" olduğundan davranışlara karşılık gelen plan adımları ile oluşturulan gövde arasındaki kapsama ilişkisi bu kuralın işletimi sonrasında hazırlanmış olmaktadır.

`SemanticServiceFinderPlan2Plan` kuralının 4. satırında çağrılan `partofPatternforFinderPlan` adlı yardımcı kural tezin 5.2.2. bölümünde anlatıldığı gibi burada da ilgili plan örneklerinin kaynak modelde belirlenmesini sağlayan bir yardımcı kuraldır.

"Davranış", "Anlamsal Web Etmeni" ve "Anlamsal Servis Bulma Planı" varlıklarının dönüşümü gibi ilgili tüm PIMM üstvarlıklarından NUIN PSMM üstvarlıklarına dönüşüm kuralları yazılmış ve hayata geçirilmiştir. Burada dikkat edilmesi gereken tıpkı SEAGENT çalışmasında olduğu gibi üstmodellerin KM3 ve Ecore'larının hazırlanması ve ATL kurallarının yazılması işlemlerinin ilgili MDD sürecinin "tasarım aşaması"'nda yerine getirilmiş olmasıdır. Artık bir MAS geliştirici NUIN ortamı için bu süreci uygulamak istediğinde tek yapması gereken PIMM'e uygun olarak MAS modelini hazırlamak ve dönüşüm işlemine girdi olarak bu modeli vermektir. Dönüşüm kuralları ve dönüşümün iç mekanizması ile uğraşmasına gerek yoktur.

7.2.1 NUIN Model dönüşümlerinin örnek girdi modeli üzerinde uygulanması

Anlamsal Web ortamında çalışan MAS'lara ait PIMM ve NUIN PSMM arasında bir önceki bölümde tanıtılan M2M dönüşümleri tezin 4.3. bölümünde tanıtılan platform bağımsız turizm MAS'ının NUIN karşılığının elde edilmesi amacıyla söz konusu girdi modeli üzerinde uygulanmıştır. Bölüm 5.2.3'te de belirtildiği gibi girdi model üzerinde ATL motorunun dönüşüm kurallarını uygulaması için modelin Ecore

gösterimine ihtiyaç vardır. Tezin Ek 4 bölümünde Ecore biçimi verilen bu MAS modeli doğrudan burada anlatılan uygulamada da kullanılmıştır.

Örneğin kaynak modeldeki “Otel İstemcisinin Servis Keşif Planı” adlı Anlamsal Servis Bulma Planı model elemanı üzerinde bir önceki bölümde tanıtılan `SemanticServiceFinderPlan2Plan` adlı dönüşüm kuralı uygulandığında ilgili planın NUIN çerçevesi için karşılığı elde edilmektedir. Aşağıdaki Ecore gösteriminde de elde edilen bu çıktı model elemanı ilişkileri ile birlikte verilmiştir. Model elemanlarının asıl Ecore dosyasındaki konumları yine sol taraftaki rakamlarla gösterilmiştir. “Otel İstemcisinin Servis Keşif Planı” model elemanı hariç diğer model elemanlarının başka model elemanları ile olan ilişki bağlantıları da yine burada gösterilmemiştir ve “...” ile temsil edilmiştir.

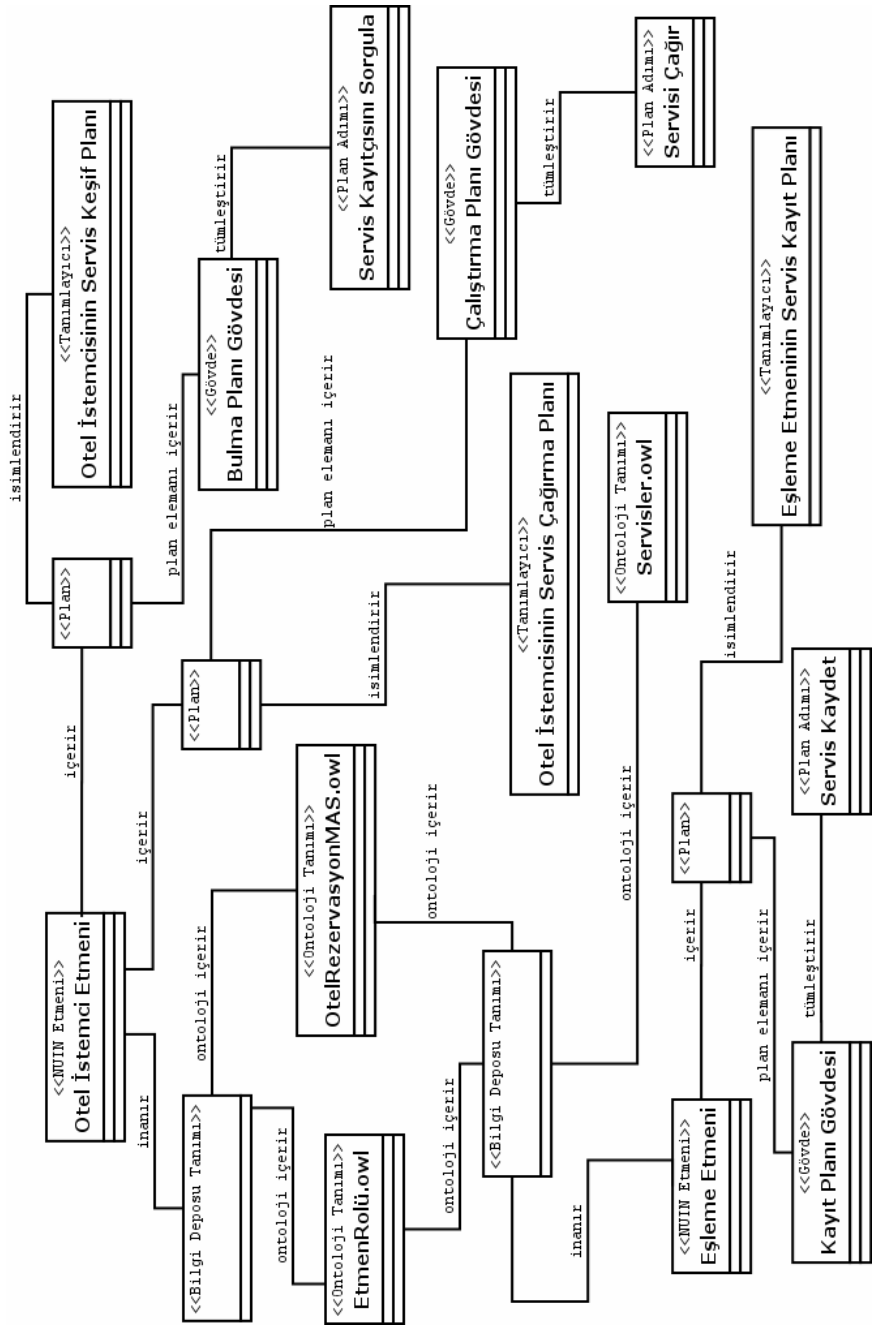
```
<xmi:XMI xmi:version="2.0"
 xmlns:xmi="http://www.omg.org/XMI"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="NUIN">
 [0] <NUINAgent name="Otel Istemci Etmeni" believe="...">
        <contain namedBy="/4">
           <containPlanElement xsi:type="Body"
              name="Finder Plan Body">
             <compose name="Servis Kayitcisini Sorgula"/>
           </containPlanElement>
        </contain>
      </NUINAgent>
  [4] <Identifier id="Otel Istemcisinin Servis Kesif Plani"
         name="/0/@contain.1"/>
</xmi:XMI>
```

Burada dikkat edilmesi gereken ilgili plan ve ihtiyaç duyduğu plan adımları yine otomatik dönüşüm sonucu elde edilmiş bir NUIN Etmeni örneğinin iç elemanları olarak sırası ile “<contain>” ve “<compose>” etiketleri ile gösterilmişlerdir. Planın tanımlayıcısı ise tek başına bir model elemanı olarak gösterilmiştir. Bu tip Ecore eleman gösterimleri de

hem sözdizim hem de anlamsal olarak doğrudur ve ATL motorunun çıktı modeli otomatik olarak üretilirken tercih ettiği bir gösterim şeklidir.

Bölüm 5.2.3'teki Şekil 5.6'da gösterilen biçimde ATL model dönüşümü çalışma zamanı ayarları yapıldıktan sonra girdi model üzerinde hazırlanan dönüşüm kuralları uygulanmıştır ve tezin Ek 8 bölümünde verilen NUIN Turizm MAS modeli otomatik olarak elde edilmiştir. Dönüşüm motoru girdi olarak PIM'in Ecore'unu almış çıktı olarak da PSM'in yine Ecore'unu üretmiştir. Ek 8'de verilen çıktı modelin içerdiği model elemanlarının ve ilişkilerinin görsel bir gösterimi Şekil 7.3'te verilmiştir.

Dönüşümler sonrasında turizm MAS'ında yer alan platform etmenleri "NUIN Etmen"'leri, planları da NUIN "Plan"'ları ile temsil edilmektedir. Her BDI etmeni için etmen inançlarını "Ontoloji Tanımı" örnekleri olarak saklayacak olan birer "Bilgi Deposu Tanımı" oluşturulmuştur. Nuinscript tanımlarına uygun olarak kaynak modelin Plan örnekleri çıktı modelinde tanımlayıcıları ve gövdeleri ile yer almaktadır. Kaynak modeldeki etmen davranışları da ilgili etmenin plan davranışları olarak modelde yer almaktadır. Örneğin PIM'de "Otel İstemci Etmeni"'nin uyguladığı "Otel İstemcisinin Servis Çağırma Planı"'nın içerdiği "Servisi Çağır" davranışı PIMM – NUIN PSMM varlık eşlemelerine uygun olarak çıktı modelde bir "Plan Adımı" örneği olarak yer almaktadır. İlgili planın da bu plan adımını içeren gövdesi dönüşümler sırasında otomatik olarak oluşturulmuştur.



Şekil 7.3. Turizm MAS PIM'inin model dönüşümleri sonrası elde edilen NUIN platformuna özgü modeli

7.2.2 NUIN model dönüşümlerinin değerlendirilmesi

NUIN çerçevesi için hazırlanan modelden modele dönüşümlerin SEAGENT dönüşümlerine göre iki noktada farklılığı gözlenmiştir: Dönüşümlerin *Bütünlüğü* ve *Karmaşıklığı*. Bir PIM'den SEAGENT PSM'lerine olan dönüşümler NUIN PSM'leri için olanlarına göre daha bütündür. PIMM üstvarlıklarının ve ilişkilerinin birçoğunun SEAGENT PSMM'inde uygun veya hemen hemen tam karşılıkları sağlanabilmiştir. Bu nedenle, birçok durumda, tasarlanmış MAS modelinin SEAGENT ortamı için gerçekleştirimleri elde edilebilmiştir. Ancak özellikle anlamsal web servis yapıları göz önüne alındığında aynı modellerin NUIN ortamında tam olarak gerçekleştirimlerinin tamamlanabilmesi için yeni varlık tanımlarının ve Nuinscript varlık uzantılarının sağlanması ihtiyacı doğmuştur. Çünkü NUIN'in Anlamsal Web desteği etmen konfigürasyonları ve bilgi deposu tanımları ile sınırlıdır (Dickinson, 2006). Etmen servis etkileşimlerinin gerçek uygulamalarının sağlanması için şu anki NUIN üstmodelinde bulunmayan harici yazılım yapılarının (örneğin yeni etmen planları, anlamsal web servis bileşenleri, vb.) NUIN için tanımlanması gerekmektedir.

Öte yandan NUIN PSMM'i için hazırlanan model dönüşümleri SEAGENT PSMM'i için hazırlananlara göre daha karmaşıktır ve uygulaması daha zor olmuştur. SEAGENT PSMM için tanımlanan model dönüşümleri genellikle hedef modeldeki varlıklara ait özelliklerin ayarlanması ve kaynak desenindeki ilgili model örneklerinin belirlenmesinden sonra bu kaynak modeli örneklerinin hedef modeli karşılıklarının oluşturulmasından ibarettir. Ancak NUIN PSMM'i için hazırlanan dönüşümler varlık eşlemesinden çok daha fazlasına ihtiyaç duymuştur. NUIN dönüşümleri için yazılan kurallar varlık eşlemeleri dışında çeşitli dinamik model elemanı oluşturma işlemlerini ve hedef ortamdaki ilişkilere uygun olarak önceden türetilmiş model örneklerinin

seçimini sağlayan karmaşık OCL sorgularını içermektedir. NUIN ve SEAGENT PSMM için gerçekleştirilen model dönüşümlerinde gözlenen bu farklılıklar beklenen ve doğal faklılıklardır. Çünkü SEAGENT platformu ve bu tezde tanıtılan PIMM, soyutlamaları, tasarım mekanizmaları ve modelledikleri ortamlar yönünden birbiri ile uyumludur.

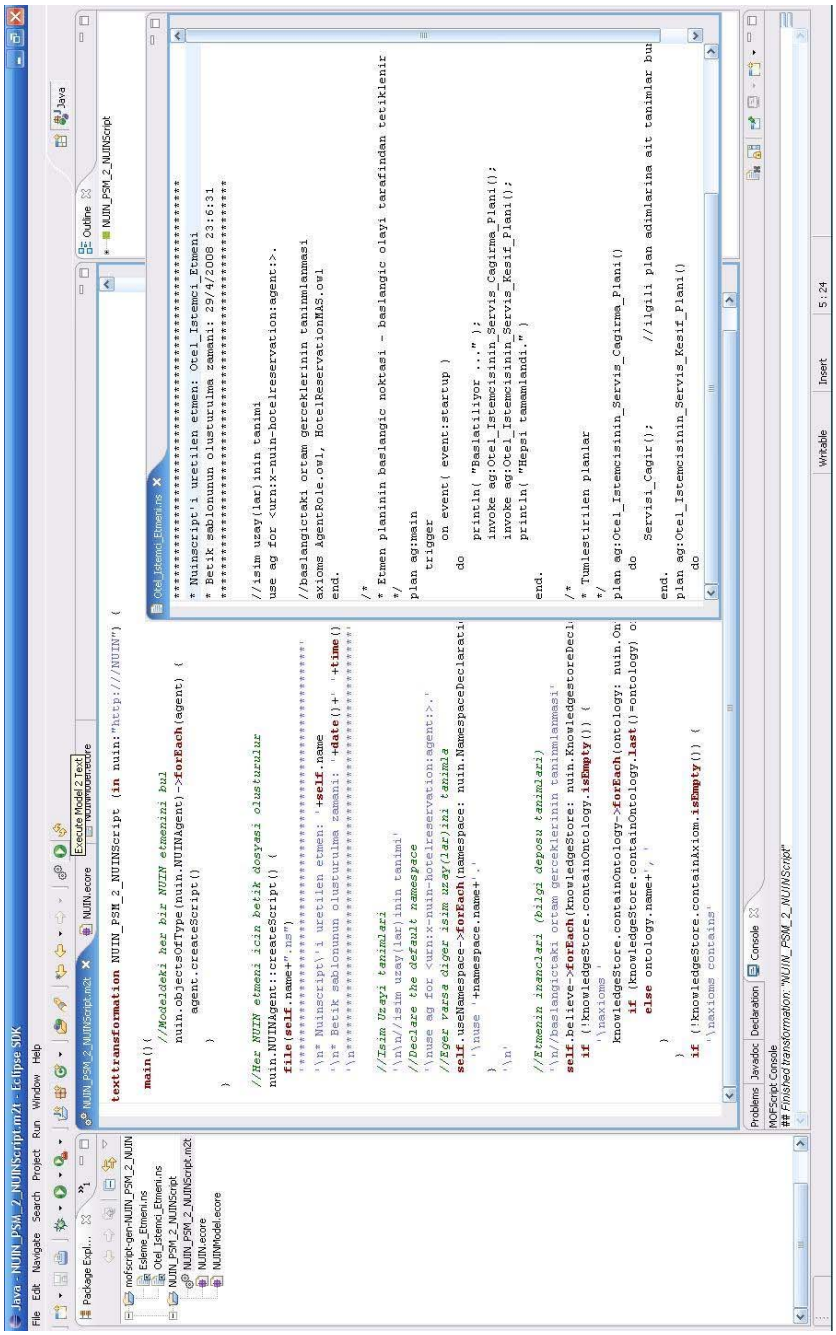
7.3 NUIN Etmen Yazılım Kodlarının Otomatik Üretimi için Modelden Metne Dönüşümlerin Tanımlanması ve Uygulanması

NUIN platformuna yönelik model güdümlü etmen sistemi geliştirme sürecinin son adımı olarak platforma özgü modellerden yazılım kodlarının elde edilmesi çalışması yerine getirilmiştir. NUIN çerçevesi göz önüne alındığında modelden koda dönüşüm NUIN PSMM'ine uyan modellerden Nuinscript'lerin elde edilmesini içermektedir. Tezin 6. bölümünde de belirtildiği gibi birçok MAS geliştirme ortamı model güdümlü geliştirmeye uygun yazılım geliştirme araçları sunmamaktadır. NUIN çerçevesi de sadece bir kod kütüphanesinden ibarettir ve herhangi bir yazılım geliştirme aracına sahip değildir. Bu nedenle yine 6. bölümde belirtildiği gibi NUIN PSMM'inden yazılım kodlarının elde edilmesi işlemini gerçekleştirecek dönüşümler başka bir dönüşüm üstmodeli ve dili kullanılarak hazırlanmıştır.

NUIN etmenleri için yazılım kodu üretmeye yönelik dönüşümün uygulanması sırasında MOF modellerinden metne dönüşümü sağlayan MOFScript (Oldevik et al., 2005) dili kullanılmıştır. MOFScript literatürde oldukça iyi bilinen ve modellerin metin dosyalarına dönüştürülmesi için tasarlanmış bir dildir. Girdi olarak doğrudan üstmodel tanımları (Ecore dosyaları) ile ilgilenir. Ayrıca MOFScript

dönüşümlerinin doğrudan Eclipse ortamında yazılıp üzerinde çalıştırıldığı bir Eclipse eklentisi (Eclipse Community, 2005) bulunmaktadır. Bu avantajları nedeniyle MOFScript, NUIN platformuna özgü modellerden Nuinscript kodlarının üretilmesi safhasında uygulama dili olarak seçilmiştir.

Şekil 7.4'te arka planda, hazırlanan MOFScript dönüşümünden bir alıntı verilmiştir. Bu metin dönüşümü sürecin önceki adımlarında da kullanılan NUIN PSMM'inin Ecore gösterimini kullanmaktadır. Söz konusu dönüşüm NUIN üstmodeline uyan bir MAS modelini okumakta ve modelde yer alan her NUIN Etmeni için Nuinscript'leri oluşturmaktadır. Ecore yapısında verilmiş bir NUIN MAS modeli üzerinde bu M2T dönüşümü çalıştırıldığında NUIN Etmen örnekleri belirlenmekte ve her biri için ".ns" uzantılı betik dosyaları oluşturulmaktadır (Şekil 7.4'te sağ tarafta). Her etmen için şablon seviyesinde oluşturulan bu betikler Nuinscript sözdizim kurallarına uygun bir biçime sahiptirler. Betikler oluşturulurken etmenin isim uzayı tanımları, modelde bilgi deposu tanımları olarak verilmiş etmen inançları ve yine modelde plan adımlarının bir kompozisyonu şeklinde verilen ve NUIN planlarınca temsil edilmiş etmen niyetleri göz önüne alınmaktadır. Tam bir uygulama için bir sistem geliştirici otomatik olarak üretilmiş bu şablon betiklerini tamamlar. NUIN PSM'lerinden etmenler için şablon seviyede yazılım kodu üretilmesini sağlayan "NUIN_PSM_2_NUINScript.m2t" adlı MOFScript M2T dönüşümünün tamamı tezin Ek 9 bölümünde verilmiştir.



Şekil 7.4. NUIN PSM'lerinden Nuinscript betiklerinin elde edilmesi için hazırlanan MOFScript M2T dönüşümü ve bu dönüşümün turizm MAS PSM'i üzerinde işletimi sonrası "Otel İstemci Etmeni" adlı NUIN Etmeni için üretilen şablon Nuinscript

NUIN BDI etmenlerinin MDD'ye uygun bir biçimde geliştirilmesini örneklemek amacıyla kullanılan turizm MAS'ının platforma özgü modeli üzerinde söz konusu MOFScript dönüşümü çalıştırıldığında modelde yer alan her NUIN Etmeni için şablon Nuinscript betik dosyalarının otomatik olarak elde edilmesi gerçekleştirilmiştir. Örneğin "Otel İstemci Etmeni" adlı etmen için otomatik olarak üretilen şablon Nuinscript betiği Şekil 7.4'te sağ tarafta listelenmiştir.

Yukarıda anlatılan M2T dönüşümüne ait MOFScript hazırlama işlemi de tıpkı M2M için ATL dönüşüm kurallarının yazılması gibi tezde önerilen MDD sürecinin "tasarım aşaması"'na ait bir işlemdir ve bu geliştirme sürecini kullanacak bir etmen geliştiricisinin bu işlemin iç yapısı ile ilgilenmesine gerek yoktur. Böylelikle hem M2M hem de M2T dönüşümleri otomatik bir şekilde yürütülmektedir ve sistem geliştiricinin yine yapması gereken tek iş sadece PIMM'e dayalı olarak üzerinde çalıştığı MAS'ın modelini hazırlamaktır. Bu model M2M sürecine girdi olarak verilerek NUIN PSM karşılığı elde edilmekte; elde edilen PSM de M2T'ye girdi olarak verilerek tasarlanan sistemin NUIN platformuna ait şablon seviyesinde yazılım kodları elde edilmektedir.

8 SONUÇ

Anlamsal Web ortamında çalışan MAS'ların model güdümlü olarak geliştirilmesine yönelik yapılar ve geliştirme süreci adımları bu tez çalışması kapsamında tanımlanmış ve uygulamaya geçirilmiştir. Gerçekleştirilen çalışmalar sonucunda söz konusu etmen sistem yazılımların hazırlanması amacıyla bir referans mimari ve buna bağlı bir platform bağımsız MAS üstmodeli geliştirilmiştir. Anlamsal Web yetenekli etmen sistemlerine ait varlıkları ve bunların birbirleri ile olan ilişkilerini içeren ve bu çalışmada tanıtılan üstmodel, etmen ve Anlamsal Web yapılarının tek bir ortamda modellenebilmesini sağlamaktadır ve bu alanda literatürdeki ilk çalışmadır.

Önerilen geliştirme süreci aynı zamanda bütün bir model dönüşümünü sağlamaktadır. Bu model dönüşümünde kaynak ve hedef etmen üstmodelleri arasındaki eşlemeler ve dönüşümün farklı MAS geliştirme çerçeveleri için uygulamaları yerine getirilmiştir. Sistem gerçekleştirimlerini örneklemek amacıyla iki ayrı MAS yazılımı geliştirme platformu kullanılmıştır. Başka platformlar için de sürecin uygulanması mümkündür. Bunu sağlamak amacıyla yeni platformlara ait üstmodellerin oluşturulması ve bu uygulama platformu üstmodelleri ile bu çalışmada ortaya konan platform bağımsız etmen üstmodeli arasındaki varlık eşlemelerinin yerine getirilmesi ve dönüşüm kurallarının yazılması gereklidir.

Tezde önerilen model güdümlü etmen geliştirme sürecinin SEAGENT (Dikenelli et al., 2005) dışında NUIN (Dickinson & Wooldridge, 2003) BDI etmenlerinin geliştirilmesinde kullanılması önerilen MDD'in farklı platformlar için de uygulanabilir olduğunu göstermiştir. Özellikle NUIN gibi tasarımı ve uygulaması başka etmen araştırmacıları tarafından yerine getirilmiş bir etmen geliştirme çerçevesi

için önerilen MDD'in uygulanabilir olduğunu gözlemlemek özellikle savunulan yöntemin güçlü bir değerlendirmesini ve özgün değerinin ortaya konmasını sağlamıştır. Çünkü bugüne kadar ki model güdümlü etmen geliştirme çalışmalarının Anlamsal Web ortamı ve ilgili anlamsal servis – etmen etkileşimlerini göz önüne almamaları yanında en büyük eksiklikleri önerilen çalışmaların sadece tek bir sistem geliştirme platformunu ve dolaysıyla PSMM'ini dikkate almalarıdır. Söz konusu platform da genellikle yine MDD önerisini getiren araştırmacıların kendilerinin geliştirdiği, tasarım mantığında ve uygulamada modifikasyon haklarının ellerinde olduğu MAS geliştirme ortamlarıdır. Tez kapsamında ortaya konan model güdümlü geliştirme yönteminin ilk versiyonları, bu çalışmalara benzer bir biçimde, tasarımında ve uygulamasında doğrudan görev alındığı için iç yapısı bilinen ve üzerinde değişiklik yapma serbestisine sahip olunan SEAGENT MAS çerçevesi göz önüne alınarak hazırlanmıştır. Ancak SEAGENT'a ek olarak NUIN çerçevesinin de MAS geliştirmede hedef platformlardan biri olarak kullanılması önerilen sürecinin geliştirilmesini ve iyileştirilmesini de sağlamıştır.

Çalışmanın geliştirilmesine yönelik ileride yapılabilecek katkıların başında iş alanları için ihtiyaç mühendisliğinin yerine getirilmesi ve platform bağımsız sistem özelliklerinin programlama bağımsız bir üstmodelden PIMM'e aktarılması gelebilir. Bu nedenle bir Programlama Bağımsız Üstmodel oluşturulmalı ve bu modele göre sistem bileşenlerinin etmen, servis, vb. yapılardan hangileri ile bir MAS'ta temsil edileceği belirlenip ilgili sistemin model dönüşümleri sonucunda PIM'i çıkarılmalıdır. PIM'in elde edilmesinden sonra bu PIM tezde önerilen MDD'ye girdi olarak verilerek arzu edilen platformda ilgili MAS'ın uygulaması yerine getirilebilir.

Bir diğer geliştirme çalışması da tezde önerilen model dönüşüm ve varlık eşlemelerinin iyileştirilmesi yönünde olabilir. Varlık özellikleri (*"attribute"*) seviyesinde eşlemelerin detaylandırılması, anlamsal web servis varlıklarının girdi/çıktı ve ön koşul/etki gösterimlerinin modellerde detaylandırılması bu kapsamda yerine getirilebilecek çalışmalara örnektir.

Öte yandan model dönüşümlerine girdi olması için MAS modellerinin oluşturulması amacıyla bir yazılım aracının hazırlanmasının geliştirme sürecine büyük katkı sağlayacağı düşünülmektedir. Platform bağımsız etmen modellerinin Ecore gösterimlerinin hazırlanması metinsel olmasından dolayı hataya açıktır ve nispeten zahmetlidir. Özellikle model içerisinde elemanların, modele yazılma sıralarına göre birbirlerine referans vermesi gerekliliği ve buna dayalı olarak ilişki gösterimlerinde referans numaralarının verilmesi model hazırlamayı zorlaştırmaktadır. Bunları önlemek için sistem geliştiricilerin kullanımına sunulacak grafiksel ara yüze sahip bir modelleme aracı tezde tanıtılan PIMM'i üstmodel alan MAS modellerinin görsel olarak tasarımına imkan verebilir ve ilgili modellerin Ecore gösterimlerinin otomatik olarak elde edilmesini sağlayabilir.

KAYNAKLAR

Agrawal, A., Karsai, G., Neema, S., Shi, F. and Vizhanyo, A., 2006, The design of a language for model transformation, *Software and Systems Modeling,* 5(3): 261-288.

Amor, M., Fuentes, L. and Vallecillo, A., 2005, Bridging the Gap Between Agent-Oriented Design and Implementation Using MDA, *Lecture Notes in Computer Science,* 3382: 93-108.

Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I. and Weerawarana, S., 2003, Business Process Execution Language for Web Services Version 1.1, http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/ws-bpel.pdf, (son erişim: Nisan, 2008).

AOS, 2006, Agent Oriented Software Group JACK Intelligent Agents, http://www.agent-software.com/, (son erişim: Nisan, 2008).

ATLAS Group, 2006, ATL User Manual, http://www.eclipse.org/m2m/atl/doc/ATL_User_Manual[v0.7].pdf, (son erişim: Nisan, 2008).

Bauer, B., Muller J.P. and Odell, J., 2001, Agent UML: A formalism for specifying multiagent software systems, *International Journal of Software Engineering and Knowledge Engineering,* 11(3): 207-230.

Bauer, B. and Odell, J., 2005, UML 2.0 and agents: how to build agent-based systems with the new UML standard, *Engineering Applications of Artificial Intelligence,* 18(2): 141-157.

Bellifemine, F., Poggi, A. and Rimassa, G., 2001, Developing Multi-agent Systems with a FIPA-compliant Agent Framework, *Software Practice and Experience,* 31: 103-128.

Bergenti, F., Gleizes, M. and Zambonelli, F., 2004, Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook, Kluwer Academic Publishers, USA, 506p.

KAYNAKLAR (devam)

Berners-Lee, T., Hendler, J. and Lassila, O., 2001, The Semantic Web, *Scientific American,* 284(5): 34-43.

Bernon, C., Gleizes, M-P., Peyruqueou, S. and Picard , G., 2003, ADELFE: A methodology for adaptive multi-agent systems engineering. *Lecture Notes in Artificial Intelligence,* 2577: 70-81.

Bernon, C., Cossentino, M., Gleizes, M-P., Turci, P. and Zambonelli, F., 2005, A Study of some Multi-Agent Meta-Models, *Lecture Notes in Computer Science,* 3382: 62-77.

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F. and Mylopoulos, J., 2004, Tropos: An agent-oriented software development methodology, *Autonomous Agents and Multi-Agent Systems,* 8(3): 203-236.

Buschmann F., Meunier R., Rohnert H., Sommerlad P. and Stal M., 1996, Pattern-Oriented Software Architecture, Volume 1: A System of Patterns, John Wiley & Son, USA, 476p.

Caglayan, A.K. and Harrison, C.G., 1997, Agent Sourcebook: A Complete Guide to Desktop, Internet, and Intranet Agents, John Wiley & Sons, USA, 349p.

Cervenka, R., Trencansky, I., Calisti, M. and Greenwood, D., 2005, AML: Agent Modeling Language – Toward Industry-Grade Agent-Based Modeling, *Lecture Notes in Computer Science,* 3382: 31-46.

Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Nord, R. and Stafford, J., 2003, Documenting Software Architectures: Views and Beyond, Pearson Education, USA, 560p.

Cossentino, M. and Potts, C., 2002, A CASE tool supported methodology for the design of multi-agent systems. In proceedings of the 2002 International Conference on Software Engineering Research and Practice (SERP'02), Las Vegas, USA.

DeLoach, S.A., Wood, M.F. and Sparkman, C.H., 2001, Multiagent systems engineering, *International Journal of Software Engineering and Knowledge Engineering,* 11(3): 231-258.

KAYNAKLAR (devam)

Depke, R., Heckel, R. and Küster, J.M., 2001, Agent-Oriented Modeling with Graph Transformations, *Lecture Notes in Computer Science,* 1957: 105-119.

Dickinson, I., 2006, BDI Agents and the Semantic Web: Developing User-Facing Autonomous Applications, Ph.D. Thesis, University of Liverpool, 204p.

Dickinson, I. and Wooldridge, M., 2003, Towards Practical Reasoning Agents for the Semantic Web, In proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003), Melbourne, Australia, ACM Press, pp. 827-834.

Dickinson, I. and Wooldridge, M., 2005, Agents are not (just) web services: considering BDI agents and web services, In proceedings of the workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE 2005), Utrecht, the Netherlands.

Dikenelli, O., Erdur, R.C., Gumus, O., Ekinci, E.E., Gurcan, O., Kardas, G., Seylan, I. and Tiryaki, A.M., 2005, SEAGENT: A Platform for Developing Semantic Web Based Multi Agent Systems, In proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), Utrecht, The Netherlands, ACM Press, pp. 1271-1272.

Dikenelli, O., Erdur, R.C., Kardas, G., Gümüs, O., Seylan, I., Gurcan, O., Tiryaki, A.M. and Ekinci, E.E., 2006, Developing Multi Agent Systems on Semantic Web Environment using SEAGENT Platform, *Lecture Notes in Artificial Intelligence,* 3963: 1-13.

Djuric, D., 2004, MDA-based Ontology Infrastructure, *Computer Science Information Systems,* 1(1): 91-116.

Duddy, K., Gerber A., Lawley, M., Raymond, K. and Steel, J., 2003, Model Transformation: A declarative, reusable patterns approach, In proceedings of Seventh IEEE International Enterprise Distributed Object Computing Conference (EDOC'03), IEEE Computer Society, pp. 174-185.

KAYNAKLAR (devam)

Eclipse Community, 2003, Eclipse Açık Geliştirme Platformu, http://www.eclipse.org, (son erişim: Mayıs, 2008).

Eclipse Community, 2004, Eclipse Modeling Framework, http://www.eclipse.org/emf, (son erişim: Mayıs, 2008).

Eclipse Community, 2005, MOFScript modelden metne dönüşüm dili ve aracı, http://www.eclipse.org/gmt/mofscript/, (son erişim: Mayıs, 2008).

Eclipse Community, 2006, Graphical Editing Framework, http://www.eclipse.org/gef, (son erişim: Mayıs, 2008).

Erdur, R.C., 2001, Yazılım Etmeni Teknolojisinin İnternet Tabanlı Yazılım Yeniden Kullanımına Uygulanması, Doktora Tezi, Ege Üniversitesi Fen Bilimleri Enstitüsü, 213s.

Fensel, D., Hendler, J., Lieberman, H. and Wahlster, W., 2003, Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential, The MIT Press, USA, 479p.

Ferber, J. and Gutknecht, O., 1998, A Meta-Model for the Analysis and Design of Organizations in Multi-Agent Systems, In proceedings of the Third International Conference on Multi-Agent Systems, IEEE Computer Society, pp. 128-135.

FIPA, 2002, FIPA Standartları, http://www.fipa.org, (son erişim: Mart, 2008).

FIPA Modeling TC, 2004, Agent Class Superstructure Metamodel, http://www.omg.org/docs/agent/04-12-02.pdf, (son erişim: Nisan, 2008).

Franklin, S. and Graesser, A., 1997, Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents, *Lecture Notes in Computer Science,* 1193: 21-25.

Göknil, A., Kardaş, G., Topaloğlu, N. Y. ve Dikenelli, O., 2006, Anlamsal Web Tabanlı Etmen Sistemlerinin Geliştirilmesinde Model Tabanlı Yaklaşım, Birinci Ulusal Yazılım Mimarisi Konferansı (UYMK 2006), İstanbul, Türkiye, ss. 177-183.

KAYNAKLAR (devam)

Gracanin, D., Singh, H.L., Bohner, S.A. and Hinchey, M.G., 2005, Model-Driven Architecture for Agent-Based Systems, *Lecture Notes in Artificial Intelligence,* 3228: 249-261.

Graham, J.R., Decker, K.S. and Mersic, M., 2003, DECAF - A Flexible Multi Agent Systems Infrastructure, *Autonomous Agents and Multi-agent Systems,* 7(1-2): 7-27.

Grønmo, R., Jaeger, M. C. and Hoff, H., 2005, Transformations Between UML and OWL-S, *Lecture Notes in Computer Science,* 3748: 269-283.

Gümüs, Ö., Gürcan, Ö., Kardas, G., Ekinci, E. E. and Dikenelli, O., 2007, Engineering an MAS Platform for Semantic Service Integration based on the SWSA, *Lecture Notes in Computer Science,* 4805: 85-94.

Gürcan, Ö., Kardas, G., Gümüs, Ö., Ekinci, E. E. and Dikenelli, O., 2006, A Planner for Implementing Semantic Service Agents based on Semantic Web Services Initiative Architecture, In proceedings of the Fourth European Workshop on Multi-Agent Systems (EUMAS 2006), Lisbon, Portugal, CEUR Workshop Proceedings, 223: 249-259.

Gürcan, Ö., Kardas, G., Gümüs, Ö., Ekinci, E. E. and Dikenelli, O., 2007, An MAS Infrastructure for Implementing SWSA based Semantic Services, *Lecture Notes in Computer Science,* 4504: 118-131.

Hahn, C., Madrigal-Mora, C., Fischer, K., Elvesæter, B., Berre, AJ. and Zinnikus, I., 2006, Meta-models, Models, and Model Transformations: Towards Interoperable Agents, *Lecture Notes in Artificial Intelligence,* 4196: 123-134.

Hahn, C., Madrigal-Mora, C. and Fischer, K., 2008, A platform-independent metamodel for multiagent systems, *Autonomous Agents and Multi-agent Systems,* DOI: http://dx.doi.org/10.1007/s10458-008-9042-0.

Huget, M-P, 2005, Modeling Languages for Multiagent Systems, In proceedings of the 6th International Workshop on Agent-Oriented Software Engineering (AOSE 2005), Utrecht, The Netherlands.

KAYNAKLAR (devam)

Intelligent Automation Inc., 2002, OpenCybele Agent Infrastructure, http://www.opencybele.org, (son erişim: Nisan, 2008).

Jayatilleke, G.B., Padgham, L. and Winikoff, M., 2004, Towards a Component based Development Framework for Agents, *Lecture Notes in Artificial Intelligence,* 3187: 183-197.

Jayatilleke, G.B., Padgham, L. and Winikoff, M., 2007, Evaluating a Model Driven Development Toolkit for Domain Experts to Modify Agent Based Systems, *Lecture Notes in Computer Science,* 4405: 190-207.

JENA, 2003, A Semantic Web Framework for Java, available at: http://jena.sourceforge.net, (son erişim: Mart, 2008).

Jouault, F. and Bezivin, J., 2006, KM3: A DSL for Metamodel Specification, *Lecture Notes in Computer Science,* 4037: 171-185.

Jouault, F. and Kurtev, I., 2006, Transforming Models with ATL, *Lecture Notes in Computer Science,* 3844: 128-138.

Kalnins, A., Barzdins, J. and Celms E., 2005, Model Transformation Language MOLA, *Lecture Notes in Computer Science,* 3599: 62-76.

Kardas, G., Gümüs, Ö. and Dikenelli, O., 2005, Applying Semantic Capability Matching into Directory Service Structures of Multi Agent Systems, *Lecture Notes in Computer Science,* 3733: 452-461.

Kardas, G., Goknil, A., Dikenelli, O. and Topaloglu, N. Y., 2006, Metamodeling of Semantic Web Enabled Multiagent Systems, In proceedings of the Multiagent Systems and Software Architecture (MASSA), Special Track at Net.ObjectDays - NODe 2006, Erfurt, Germany, pp. 79-86.

Kardas, G., Goknil, A., Dikenelli, O. and Topaloglu, N. Y., 2007a, Modeling the Interaction between Semantic Agents and Semantic Web Services using MDA Approach, *Lecture Notes in Artificial Intelligence,* 4457: 209-228.

KAYNAKLAR (devam)

Kardas, G., Goknil, A., Dikenelli, O. and Topaloglu, N. Y., 2007b, Model Transformation for Model Driven Development of Semantic Web Enabled Multi-Agent Systems, *Lecture Notes in Artificial Intelligence,* 4687: 13-24.

Kardaş, G., Gümüş, Ö. ve Dikenelli, O., 2005, Anlamsal Web Servislerinin Bulunması, Eşlenmesi ve Dinamik Çağrımı Üzerine Bir Durum Çalışması, İkinci Ulusal Yazılım Mühendisliği Sempozyumu ve Sergisi (UYMS 2005), Ankara, Türkiye, ss. 41-49.

Kardaş, G. ve Dikenelli, O., 2006, Çok-Etmenli Yazılım Sistemleri için Yürütülen Modelleme Dili Çalışmaları ve Bunların Anlamsal Web Desteği Perspektifinde Değerlendirilmesi, Fifteenth Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN 2006), Akyaka, Muğla, Türkiye, ss. 199-206.

Li, L. and Horrocks, I., 2003, A Software Framework for Matchmaking based on Semantic Web Technology, In proceedings of the World Wide Web Conference 2003, Budapest, Hungary, pp. 331-339.

McGuiness, D.L. and van Harmelen, F., 2004, OWL Web Ontology Language Overview, http://www.w3.org/TR/owl-features/, (son erişim: Mart, 2008).

Mens, T. and Van Gorp, P., 2006, A Taxonomy of Model Transformation, *Electronic Notes in Theoretical Computer Science,* 152: 125-142.

Mohagheghi, P. and Aagedal, J., 2007, Evaluating Quality in Model-Driven Engineering, In proceedings of the International Workshop on Modeling in Software Engineering (MISE'07), Minneapolis, USA.

Molesini, A., Denti, E. and Omicini, A., 2005, MAS Meta-models on Test: UML vs. OPM in the SODA Case Study, *Lecture Notes in Artificial Intelligence,* 3690: 163-172.

NUIN, 2006, NUIN Framework, http://www.nuin.org, (son erişim: Mayıs, 2008).

KAYNAKLAR (devam)

Nwana, H.S., 1996, Software Agents: An overview, *Knowledge Engineering Review,* 11(3): 205-244.

Odell, J., Nodine, M. and Levy, R., 2005, A Metamodel for Agents, Roles and Groups, *Lecture Notes in Computer Science,* 3382: 78-92.

Oldevik, J., Neple, T., Grønmo, R., Aagedal, J. and Berre, A.J., 2005, Toward Standardised Model to Text Transformations, *Lecture Notes in Computer Science,* 3748: 239-253.

OMG, 1997, Meta Object Facility Specification, OMG Document Number: AD/97-08-14, http://www.omg.org/docs/ad/97-08-14.pdf, (son erişim: Mart, 2008).

OMG, 2003, Model Driven Architecture Guide Version 1.0.1, OMG Document Number: omg/2003-06-01, www.omg.org/docs/omg/03-06-01.pdf, (son erişim: Mart, 2008).

OMG, 2004, UML 2.0 Superstructure Specification, http://www.omg.org/technology/documents/formal/uml.htm, (son erişim: Nisan, 2008).

OMG, 2007, MOF QVT Final Adopted Specification, www.omg.org/docs/ptc/05-11-01.pdf, (son erişim: Nisan, 2008).

Omicini, A., 2000, SODA: Societies and Infrastructures in the Analysis and Design of Agent-based Systems, *Lecture Notes in Computer Science,* 1957: 185-193.

OWL-S Coalition, 2004, OWL-S: Semantic Markup for Web Services, http://www.daml.org/services/owl-s/1.1/overview/, (son erişim: Mart, 2008).

Pahl, C., 2007, Semantic model-driven architecting of service-based software systems, *Information and Software Technology,* 49: 838-850.

Paolucci, M., Kawamura, T., Payne, T.R. and Sycara, K., 2002, Semantic Matching of Web Services Capabilities, *Lecture Notes in Computer Science,* 2342: 333-347.

KAYNAKLAR (devam)

Pavon, J., Gomez, J. and Fuentes, R., 2005, The INGENIAS Methodology and Tools, 236-276, Agent-Oriented Methodologies, Henderson-Sellers, B. and Giorgini, P. (Eds.), Idea Group Publishing, USA, 413p.

Pavon, J., Gomez, J. and Fuentes, R., 2006, Model Driven Development of Multi-Agent Systems, *Lecture Notes in Computer Science,* 4066: 284-298.

Penserini, L., Perini, A., Susi, A. and Mylopoulos, J., 2006, From Stakeholder Intentions to Software Agent Implementations, *Lecture Notes in Computer Science,* 4001: 465-479.

Perini, A. and Susi, A., 2006, Automating Model Transformations in Agent-Oriented Modeling, *Lecture Notes in Computer Science,* 3950: 167-178.

Rao, A. and Georgeff, M., 1995, BDI Agents: From Theory to Practice, In proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, USA, pp. 312-319.

Rougemaille, S., Migeon, F., Maurel, C. and Gleizes, M-P., 2007, Model Driven Engineering for Designing Adaptive Multi-Agent Systems, In proceedings of the Eighth Annual International Workshop on Engineering Societies in the Agents World (ESAW 2007), Athens, Greece.

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

Selic, B., 2003, The Pragmatics of Model-Driven Development, *IEEE Software,* 20: 19-25.

Sendall, S. and Kozaczynski, W., 2003, Model Transformation – the Heart and Soul of Model-Driven Software Development, *IEEE Software,* 20: 42-45.

Solheim, I. and Neple, T., 2006, Model Quality in the Context of Model-Driven Development, In proceedings of the 2nd International Workshop on Model-Driven Enterprise Information Systems (MDEIS'06), Paphos, Southern Cyprus Greek Region, pp. 27-35.

KAYNAKLAR (devam)

Sycara, K., 1998, Multiagent Systems, *AI Magazine,* 19(4): 79-92.

Sycara, K., Paolucci, M., Ankolekar, A. and Srinivasan, N., 2003a, Automated discovery, interaction and composition of Semantic Web Services, *Journal of Web Semantics,* 1: 27-46.

Sycara, K., Paolucci, M., Van Velsen, M. and Giampapa, J., 2003b, The RETSINA MAS Infrastructure, *Autonomous Agents and Multi-agent Systems,* 7(1-2): 29-48.

W3C, 2004, World Wide Web Consortium Resource Description Framework (RDF), http://www.w3.org/RDF/, (son erişim: Mart, 2008).

Warmer, J. and Kleppe, A., 2003, Object Constraint Language: The Getting Your Models Ready for MDA, Second Edition, Pearson Education, USA, 240p.

Weiss, G., 1999, Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, MIT Press, USA, 619p.

Williamson, M., Decker, K. and Sycara, K., 1996, Unified Information and Control Flow in Hierarchical Task Networks, In proceedings of the AAAI-96 Workshop, California, USA, pp. 142-150.

WSMO Working Group, 2005, Web Service Modeling Ontology, http://www.wsmo.org/index.html, (son erişim: Mart, 2008).

Zambonelli, F., Jennings, N.R. and Wooldridge, M., 2003, Developing multiagent systems: The Gaia methodology, *ACM Transactions on Software Engineering and Methodologies,* 12(3): 317-370.

Zambonelli, F. and Omicini, A., 2004, Challenges and Research Directions in Agent-Oriented Software Engineering, *Autonomous Agents and Multi-agent Systems,* 9(3): 253-283.

Zinnikus, I., Benguria, G., Elvesæter, B., Fischer, K. and Vayssière, J., 2006, A Model Driven Approach to Agent-Based Service-Oriented Architectures, *Lecture Notes in Artificial Intelligence,* 4196: 110-122.

EKLER

Ek 1 Anlamsal Web Ortamında Çalışan MAS'lara ait PIMM'in Ecore Gösterimi

Ek 2 SEAGENT MAS Geliştirme Çerçevesine ait PSMM'in Ecore Gösterimi

Ek 3 Anlamsal Web Ortamında Çalışan MAS'lara ait PIMM ve SEAGENT MAS Geliştirme Çerçevesine ait PSMM Varlıkları Arasındaki Model Dönüşümlerini Sağlayan ATL Kuralları

Ek 4 Turizm İş Alanında Çalışan MAS'lara ait PIM'in Ecore Gösterimi

Ek 5 Turizm İş Alanında Çalışan MAS'lara ait SEAGENT PSM'in Ecore Gösterimi

Ek 6 NUIN MAS Geliştirme Çerçevesine ait PSMM'in Ecore Gösterimi

Ek 7 Anlamsal Web Ortamında Çalışan MAS'lara ait PIMM ve NUIN MAS Geliştirme Çerçevesine ait PSMM Varlıkları Arasındaki Model Dönüşümlerini Sağlayan ATL Kuralları

Ek 8 Turizm İş Alanında Çalışan MAS'lara ait NUIN PSM'in Ecore Gösterimi

Ek 9 NUIN PSM'lerinden Şablon Nuinscript Kodlarını Üreten MOFScript M2T Dönüşüm Betiği

Ek 10 Türkçe - İngilizce Terimler Sözlüğü

Ek 1 Anlamsal Web Ortamında Çalışan MAS'lara ait PIMM'in Ecore Gösterimi

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore">
  <ecore:EPackage name="Agent">

    <eClassifiers xsi:type="ecore:EClass" name="SemanticWebAgent"
     eSuperTypes="/0/ACSM_Agent">
      <eStructuralFeatures xsi:type="ecore:EAttribute"
       name="name" ordered="false" unique="false" lowerBound="1"
       eType="/1/String"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="apply" ordered="false" upperBound="-1"
       eType="/0/Plan" eOpposite="/0/Plan/appliedBy"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="play" ordered="false" lowerBound="1" eType="/0/Role"
       eOpposite="/0/Role/playedBy"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="advertisedBy" ordered="false" eType="/0/RegistryRole"
       eOpposite="/0/RegistryRole/advertiseAgent"/>
    </eClassifiers>

    <eClassifiers xsi:type="ecore:EClass"
     name="SemanticWebOrganization" eSuperTypes="/0/ACSM_Group">
      <eStructuralFeatures xsi:type="ecore:EAttribute"
       name="name" ordered="false" unique="false" lowerBound="1"
       eType="/1/String"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="knowOrganizationOntology" ordered="false"
       lowerBound="1" upperBound="-1"
       eType="/0/OrganizationOntology"
     eOpposite="/0/OrganizationOntology/knownBySemanticWebOrganiz
ation"/>
    </eClassifiers>

    <eClassifiers xsi:type="ecore:EClass"
     name="SemanticServiceMatchmakerAgent"
     eSuperTypes="/0/SemanticWebAgent">
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="interactedBy" ordered="false" lowerBound="1"
       eType="/0/SemanticServiceFinderPlan"
       eOpposite="/0/SemanticServiceFinderPlan/interact"/>
    </eClassifiers>

    <eClassifiers xsi:type="ecore:EClass" name="Communication">
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="encapsulate" ordered="false" lowerBound="1"
       upperBound="-1" eType="/0/Message"
```

```
   eOpposite="/0/Message/encapsulatedBy"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="realizedBy" ordered="false" eType="/0/Role"
   eOpposite="/0/Role/realize"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass" name="Message">
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="encapsulatedBy" ordered="false"
   eType="/0/Communication"
   eOpposite="/0/Communication/encapsulate"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass" name="Behavior"
 eSuperTypes="/0/StandardUML20_BehavioralFeature">
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="name" ordered="false" unique="false" lowerBound="1"
   eType="/1/String"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="includedBy" ordered="false" lowerBound="1"
   eType="/0/Role" eOpposite="/0/Role/include"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="execute" ordered="false" upperBound="-1"
   eType="/0/Plan" eOpposite="/0/Plan/executedBy"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="knowRoleOntology" ordered="false" upperBound="-1"
   eType="/0/RoleOntology"
   eOpposite="/0/RoleOntology/knownByBehavior"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass" name="Role"
 eSuperTypes="/0/ACSM_AgentRoleClassifier">
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="name" ordered="false" unique="false" lowerBound="1"
   eType="/1/String"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="include" ordered="false" lowerBound="1" upperBound="-
   1" eType="/0/Behavior" eOpposite="/0/Behavior/includedBy"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="realize" ordered="false" upperBound="-1"
   eType="/0/Communication"
   eOpposite="/0/Communication/realizedBy"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="playedBy" ordered="false" lowerBound="1"
   eType="/0/SemanticWebAgent"
   eOpposite="/0/SemanticWebAgent/play"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="knowOrganizationOntology" ordered="false"
   eType="/0/OrganizationOntology"
   eOpposite="/0/OrganizationOntology/knownByRole"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass" name="DomainRole"
```

```
 eSuperTypes="/0/Role">
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="useRoleOntology" ordered="false" lowerBound="1"
   upperBound="-1" eType="/0/RoleOntology"
   eOpposite="/0/RoleOntology/usedByDomainRole"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass"
 name="ArchitecturalRole" eSuperTypes="/0/Role"/>

<eClassifiers xsi:type="ecore:EClass" name="RegistryRole"
 eSuperTypes="/0/ArchitecturalRole">
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="knowServiceOntology" ordered="false" upperBound="-1"
   eType="/0/ServiceOntology"
   eOpposite="/0/ServiceOntology/knownByRegistryRole"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="advertiseAgent" ordered="false" upperBound="-1"
   eType="/0/SemanticWebAgent"
   eOpposite="/0/SemanticWebAgent/advertisedBy"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="advertiseService" ordered="false" upperBound="-1"
   eType="/0/SemanticWebService"
   eOpposite="/0/SemanticWebService/advertisedBy"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass"
 name="OntologyMediatorRole"
 eSuperTypes="/0/ArchitecturalRole">
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="knowRoleOntology" ordered="false" upperBound="-1"
   eType="/0/RoleOntology"
   eOpposite="/0/RoleOntology/knownByOntologyMediatorRole"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="knowServiceOntology" ordered="false" upperBound="-1"
   eType="/0/ServiceOntology"
 eOpposite="/0/ServiceOntology/knownByOntologyMediatorRole"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass" name="Plan"
 eSuperTypes="/0/StandardUML20_Class">
  <eStructuralFeatures xsi:type="ecore:EAttribute"
 name="name" ordered="false" unique="false" lowerBound="1"
 eType="/1/String"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
 name="appliedBy" ordered="false" upperBound="-1"
 eType="/0/SemanticWebAgent"
 eOpposite="/0/SemanticWebAgent/apply"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
 name="executedBy" ordered="false" upperBound="-1"
 eType="/0/Behavior" eOpposite="/0/Behavior/execute"/>
</eClassifiers>
```

```
<eClassifiers xsi:type="ecore:EClass"
 name="SemanticServiceFinderPlan" eSuperTypes="/0/Plan">
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="interact" ordered="false" lowerBound="1"
   eType="/0/SemanticServiceMatchmakerAgent"
 eOpposite="/0/SemanticServiceMatchmakerAgent/interactedBy"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="discover" ordered="false" lowerBound="1"
 eType="/0/Interface" eOpposite="/0/Interface/discoveredBy"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass"
 name="SemanticServiceExecutorPlan" eSuperTypes="/0/Plan">
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="execute" ordered="false" lowerBound="1"
   eType="/0/Process" eOpposite="/0/Process/executedBy"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="use"
   ordered="false" lowerBound="1" eType="/0/Grounding"
   eOpposite="/0/Grounding/usedBy"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass"
 name="SemanticServiceRegisterPlan" eSuperTypes="/0/Plan">
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="advertise" ordered="false" upperBound="-1"
 eType="/0/Interface" eOpposite="/0/Interface/advertisedBy"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass" name="Service"
 eSuperTypes="/0/StandardUML20_InstanceSpecification">
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="name" ordered="false" unique="false" lowerBound="1"
   eType="/1/String"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="classifiedInstance" ordered="false" upperBound="-1"
   eType="/0/ServiceClassifier"
   eOpposite="/0/ServiceClassifier/classifier"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="owner" ordered="false" lowerBound="1"
   eType="/0/SemanticWebService"
   eOpposite="/0/SemanticWebService/service"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass"
 name="SemanticWebService"
 eSuperTypes="/0/StandardUML20_InstanceSpecification">
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="name" ordered="false" unique="false" lowerBound="1"
   eType="/1/String"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="classifiedInstance" ordered="false" upperBound="-1"
   eType="/0/SemanticWebServiceClassifier"
   eOpposite="/0/SemanticWebServiceClassifier/classifier"/>
```

```
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="interface" ordered="false" lowerBound="1"
     eType="/0/Interface" eOpposite="/0/Interface/owner"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="process" ordered="false" lowerBound="1"
     eType="/0/Process" eOpposite="/0/Process/owner"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="grounding" ordered="false" lowerBound="1"
     eType="/0/Grounding" eOpposite="/0/Grounding/owner"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="service" ordered="false" lowerBound="1" upperBound="-
     1" eType="/0/Service" eOpposite="/0/Service/owner"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="depend" ordered="false" lowerBound="1" upperBound="-
     1" eType="/0/ServiceOntology"
     eOpposite="/0/ServiceOntology/dependedBy"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="advertisedBy" ordered="false" eType="/0/RegistryRole"
     eOpposite="/0/RegistryRole/advertiseService"/>
  </eClassifiers>

  <eClassifiers xsi:type="ecore:EClass" name="Interface">
    <eStructuralFeatures xsi:type="ecore:EAttribute"
     name="name" ordered="false" unique="false" lowerBound="1"
     eType="/1/String"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="owner" ordered="false" lowerBound="1"
     eType="/0/SemanticWebService"
     eOpposite="/0/SemanticWebService/interface"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="advertisedBy" ordered="false" lowerBound="1"
     eType="/0/SemanticServiceRegisterPlan"
     eOpposite="/0/SemanticServiceRegisterPlan/advertise"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="discoveredBy" ordered="false" lowerBound="1"
     eType="/0/SemanticServiceFinderPlan"
     eOpposite="/0/SemanticServiceFinderPlan/discover"/>
  </eClassifiers>

  <eClassifiers xsi:type="ecore:EClass" name="Process">
    <eStructuralFeatures xsi:type="ecore:EAttribute"
     name="name" ordered="false" unique="false" lowerBound="1"
     eType="/1/String"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="owner" ordered="false" lowerBound="1"
     eType="/0/SemanticWebService"
     eOpposite="/0/SemanticWebService/process"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="executedBy" ordered="false" lowerBound="1"
     eType="/0/SemanticServiceExecutorPlan"
     eOpposite="/0/SemanticServiceExecutorPlan/execute"/>
  </eClassifiers>
```

```
    <eClassifiers xsi:type="ecore:EClass" name="Grounding">
      <eStructuralFeatures xsi:type="ecore:EAttribute"
       name="name" ordered="false" unique="false" lowerBound="1"
       eType="/1/String"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="owner" ordered="false" lowerBound="1"
       eType="/0/SemanticWebService"
       eOpposite="/0/SemanticWebService/grounding"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="usedBy" ordered="false" lowerBound="1"
       eType="/0/SemanticServiceExecutorPlan"
       eOpposite="/0/SemanticServiceExecutorPlan/use"/>
    </eClassifiers>

    <eClassifiers xsi:type="ecore:EClass"
     name="OrganizationOntology"
     eSuperTypes="/0/UMLOntologyProfile_Ontology">
      <eStructuralFeatures xsi:type="ecore:EAttribute"
       name="name" ordered="false" unique="false" lowerBound="1"
       eType="/1/String"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="knownBySemanticWebOrganization" ordered="false"
       upperBound="-1" eType="/0/SemanticWebOrganization"
eOpposite="/0/SemanticWebOrganization/knowOrganizationOntology"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="knownByRole" ordered="false" upperBound="-1"
       eType="/0/Role"
       eOpposite="/0/Role/knowOrganizationOntology"/>
    </eClassifiers>

    <eClassifiers xsi:type="ecore:EClass" name="ServiceOntology"
     eSuperTypes="/0/UMLOntologyProfile_Ontology">
      <eStructuralFeatures xsi:type="ecore:EAttribute"
       name="name" ordered="false" unique="false" lowerBound="1"
       eType="/1/String"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="dependedBy" ordered="false" upperBound="-1"
       eType="/0/SemanticWebService"
       eOpposite="/0/SemanticWebService/depend"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="knownByRegistryRole" ordered="false" upperBound="-1"
       eType="/0/RegistryRole"
       eOpposite="/0/RegistryRole/knowServiceOntology"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="knownByOntologyMediatorRole" ordered="false"
       upperBound="-1" eType="/0/OntologyMediatorRole"
       eOpposite="/0/OntologyMediatorRole/knowServiceOntology"/>
    </eClassifiers>

    <eClassifiers xsi:type="ecore:EClass" name="RoleOntology"
     eSuperTypes="/0/UMLOntologyProfile_Ontology">
      <eStructuralFeatures xsi:type="ecore:EAttribute"
       name="name" ordered="false" unique="false" lowerBound="1"
```

```
    eType="/1/String"/>
   <eStructuralFeatures xsi:type="ecore:EReference"
    name="knownByBehavior" ordered="false" upperBound="-1"
    eType="/0/Behavior"
    eOpposite="/0/Behavior/knowRoleOntology"/>
   <eStructuralFeatures xsi:type="ecore:EReference"
    name="usedByDomainRole" ordered="false"
    eType="/0/DomainRole"
    eOpposite="/0/DomainRole/useRoleOntology"/>
   <eStructuralFeatures xsi:type="ecore:EReference"
    name="knownByOntologyMediatorRole" ordered="false"
    upperBound="-1" eType="/0/OntologyMediatorRole"
    eOpposite="/0/OntologyMediatorRole/knowRoleOntology"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass"
 name="StandardUML20_Classifier"/>
<eClassifiers xsi:type="ecore:EClass"
 name="StandardUML20_Class"/>
<eClassifiers xsi:type="ecore:EClass"
 name="StandardUML20_InstanceSpecification"/>
<eClassifiers xsi:type="ecore:EClass"
 name="StandardUML20_StructuredClassifier"/>
<eClassifiers xsi:type="ecore:EClass"
 name="StandardUML20_BehavioralFeature"/>
<eClassifiers xsi:type="ecore:EClass"
 name="UMLOntologyProfile_Ontology"/>
<eClassifiers xsi:type="ecore:EClass"
 name="UMLOntologyProfile_OntClass"/>

<eClassifiers xsi:type="ecore:EClass"
 name="UMLSemanticWebServiceProfile_Input">
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="semanticType" ordered="false"
   eType="/0/UMLOntologyProfile_OntClass"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass"
 name="UMLSemanticWebServiceProfile_Output">
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="semanticType" ordered="false"
   eType="/0/UMLOntologyProfile_OntClass"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass"
 name="UMLSemanticWebServiceProfile_WebService">
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="includeInput" ordered="false" upperBound="-1"
   eType="/0/UMLSemanticWebServiceProfile_Input"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="includeOutput" ordered="false" upperBound="-1"
   eType="/0/UMLSemanticWebServiceProfile_Output"/>
</eClassifiers>
```

```
<eClassifiers xsi:type="ecore:EClass"
 name="ACSM_AgentClassifier"
 eSuperTypes="/0/StandardUML20_Classifier">
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="classifier" ordered="false" lowerBound="1"
   upperBound="-1" eType="/0/ACSM_Agent"
   eOpposite="/0/ACSM_Agent/classifiedInstance"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass" name="ACSM_Agent"
 eSuperTypes="/0/StandardUML20_InstanceSpecification">
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="classifiedInstance" ordered="false" upperBound="-1"
   eType="/0/ACSM_AgentClassifier"
   eOpposite="/0/ACSM_AgentClassifier/classifier"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="physicalClassifier" ordered="false"
   eType="/0/ACSM_AgentPhysicalClassifier"
   eOpposite="/0/ACSM_AgentPhysicalClassifier/agent"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="groupMember" ordered="false" upperBound="-1"
   eType="/0/ACSM_Group"
   eOpposite="/0/ACSM_Group/assignedGroup"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="roleAssignment" ordered="false" upperBound="-1"
   eType="/0/ACSM_AgentRoleAssignment"
   eOpposite="/0/ACSM_AgentRoleAssignment/agent"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass" name="ACSM_Group"
 eSuperTypes="/0/StandardUML20_StructuredClassifier">
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="assignedGroup" ordered="false" upperBound="-1"
   eType="/0/ACSM_Agent"
   eOpposite="/0/ACSM_Agent/groupMember"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="self" ordered="false" lowerBound="1"
   eType="/0/ACSM_Group"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="context" ordered="false"
   eType="/0/ACSM_AgentRoleAssignment"
eOpposite="/0/ACSM_AgentRoleAssignment/agentRoleAssignment"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="assignedGroupRole" ordered="false" lowerBound="1"
   upperBound="-1" eType="/0/ACSM_AgentRoleClassifier"
   eOpposite="/0/ACSM_AgentRoleClassifier/groupRoles"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass"
 name="ACSM_AgentifiedGroup"
 eSuperTypes="/0/ACSM_Agent/0/ACSM_Group"/>
<eClassifiers xsi:type="ecore:EClass"
```

```
 name="ACSM_NonAgentifiedGroup" eSuperTypes="/0/ACSM_Group"/>

<eClassifiers xsi:type="ecore:EClass"
 name="ACSM_AgentRoleClassifier"
 eSuperTypes="/0/ACSM_AgentClassifier">
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="permittedRoles" ordered="false" upperBound="-1"
   eType="/0/ACSM_AgentPhysicalClassifier"
   eOpposite="/0/ACSM_AgentPhysicalClassifier/
                              supportingPhysicalClass"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="agentAssignment" ordered="false" upperBound="-1"
   eType="/0/ACSM_AgentRoleAssignment"
   eOpposite="/0/ACSM_AgentRoleAssignment/role"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="classifyingRole" ordered="false" lowerBound="1"
   eType="/0/ACSM_AgentRoleClassifier"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="groupRoles" ordered="false" upperBound="-1"
   eType="/0/ACSM_Group"
   eOpposite="/0/ACSM_Group/assignedGroupRole"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass"
 name="ACSM_AgentPhysicalClassifier"
 eSuperTypes="/0/ACSM_AgentClassifier">
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="supportingPhysicalClass" ordered="false"
   upperBound="-1" eType="/0/ACSM_AgentRoleClassifier"
   eOpposite="/0/ACSM_AgentRoleClassifier/permittedRoles"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="agent" ordered="false" upperBound="-1"
   eType="/0/ACSM_Agent"
   eOpposite="/0/ACSM_Agent/physicalClassifier"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass"
 name="ACSM_AgentRoleAssignment">
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="agent" ordered="false" upperBound="-1"
   eType="/0/ACSM_Agent"
   eOpposite="/0/ACSM_Agent/roleAssignment"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="role" ordered="false"
   eType="/0/ACSM_AgentRoleClassifier"
   eOpposite="/0/ACSM_AgentRoleClassifier/agentAssignment"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="agentRoleAssignment" ordered="false" lowerBound="1"
   upperBound="-1" eType="/0/ACSM_Group"
   eOpposite="/0/ACSM_Group/context"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass"
```

```
      name="ServiceClassifier"
      eSuperTypes="/0/StandardUML20_Classifier">
       <eStructuralFeatures xsi:type="ecore:EReference"
        name="classifier" ordered="false" lowerBound="1"
        upperBound="-1" eType="/0/Service"
        eOpposite="/0/Service/classifiedInstance"/>
     </eClassifiers>

     <eClassifiers xsi:type="ecore:EClass"
      name="SemanticWebServiceClassifier"
      eSuperTypes="/0/StandardUML20_Classifier">
       <eStructuralFeatures xsi:type="ecore:EReference"
        name="classifier" ordered="false" lowerBound="1"
        upperBound="-1" eType="/0/SemanticWebService"
        eOpposite="/0/SemanticWebService/classifiedInstance"/>
     </eClassifiers>

   </ecore:EPackage>

   <ecore:EPackage name="PrimitiveTypes">
     <eClassifiers xsi:type="ecore:EDataType" name="String"/>
   </ecore:EPackage>
</xmi:XMI>
```

Ek 2 SEAGENT MAS Geliştirme Çerçevesine ait PSMM'in Ecore Gösterimi

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore">
  <ecore:EPackage name="SEAGENT">
    <eClassifiers xsi:type="ecore:EClass" name="Task"
     abstract="true">
      <eStructuralFeatures xsi:type="ecore:EAttribute"
       name="name" ordered="false" unique="false" lowerBound="1"
       eType="/1/String"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="containOutcome" upperBound="-1" eType="/0/Outcome"
       containment="true" eOpposite="/0/Outcome/containedBy"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="containProvision" upperBound="-1"
       eType="/0/Provision" containment="true"
       eOpposite="/0/Provision/containedBy"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="containedBy" ordered="false" eType="/0/Behaviour"
       eOpposite="/0/Behaviour/containTask"/>
    </eClassifiers>

    <eClassifiers xsi:type="ecore:EClass" name="Outcome">
      <eStructuralFeatures xsi:type="ecore:EAttribute"
       name="name" ordered="false" unique="false" lowerBound="1"
       eType="/1/String"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="containedBy" ordered="false" eType="/0/Task"
       eOpposite="/0/Task/containOutcome"/>
    </eClassifiers>

    <eClassifiers xsi:type="ecore:EClass" name="Provision">
      <eStructuralFeatures xsi:type="ecore:EAttribute"
       name="name" ordered="false" unique="false" lowerBound="1"
       eType="/1/String"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="containedBy" ordered="false" eType="/0/Task"
       eOpposite="/0/Task/containProvision"/>
    </eClassifiers>

    <eClassifiers xsi:type="ecore:EClass" name="Behaviour"
     eSuperTypes="/0/Task">
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="containTask" upperBound="-1" eType="/0/Task"
       containment="true" eOpposite="/0/Task/containedBy"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="containProvisionLink" upperBound="-1"
       eType="/0/ProvisionLink" containment="true"
```

```
   eOpposite="/0/ProvisionLink/containedBy"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="containInheritanceLink" upperBound="-1"
   eType="/0/InheritanceLink" containment="true"
   eOpposite="/0/InheritanceLink/containedBy"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="containDisinheritanceLink" upperBound="-1"
   eType="/0/DisinheritanceLink" containment="true"
   eOpposite="/0/DisinheritanceLink/containedBy"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="demonstratedBy" ordered="false" upperBound="-1"
 eType="/0/Agent" eOpposite="/0/Agent/demonstrate"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass" name="Action"
 eSuperTypes="/0/Task"/>
<eClassifiers xsi:type="ecore:EClass" name="ProvisionLink">
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="name" ordered="false" unique="false" lowerBound="1"
   eType="/1/String"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="containedBy" ordered="false" eType="/0/Behaviour"
   eOpposite="/0/Behaviour/containProvisionLink"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass" name="InheritanceLink">
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="name" ordered="false" unique="false" lowerBound="1"
   eType="/1/String"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="containedBy" ordered="false" eType="/0/Behaviour"
   eOpposite="/0/Behaviour/containInheritanceLink"/>
</eClassifiers>
<eClassifiers xsi:type="ecore:EClass"
 name="DisinheritanceLink">
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="name" ordered="false" unique="false" lowerBound="1"
   eType="/1/String"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="containedBy" ordered="false" eType="/0/Behaviour"
   eOpposite="/0/Behaviour/containDisinheritanceLink"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass"
 name="DiscoverCandidateService" eSuperTypes="/0/Behaviour">
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="interact" ordered="false" upperBound="-1"
   eType="/0/Agent" eOpposite="/0/Agent/interactedBy"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="find" ordered="false" upperBound="-1"
   eType="/0/OWL_S_Profile"
   eOpposite="/0/OWL_S_Profile/foundedBy"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="performedBy" ordered="false" eType="/0/Agent"
```

```
    eOpposite="/0/Agent/performDiscovery"/>
  </eClassifiers>

  <eClassifiers xsi:type="ecore:EClass" name="EnactService"
   eSuperTypes="/0/Behaviour">
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="execute" ordered="false" upperBound="-1"
     eType="/0/OWL_S_Process"
     eOpposite="/0/OWL_S_Process/executedBy"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="invoke" ordered="false" upperBound="-1"
     eType="/0/OWL_S_Grounding"
     eOpposite="/0/OWL_S_Grounding/invokedBy"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="performedBy" ordered="false" eType="/0/Agent"
     eOpposite="/0/Agent/performEnactment"/>
  </eClassifiers>

  <eClassifiers xsi:type="ecore:EClass" name="Agent">
    <eStructuralFeatures xsi:type="ecore:EAttribute"
     name="name" ordered="false" unique="false" lowerBound="1"
     eType="/1/String"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="demonstrate" ordered="false" upperBound="-1"
     eType="/0/Behaviour"
     eOpposite="/0/Behaviour/demonstratedBy"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="performDiscovery" ordered="false" upperBound="-1"
     eType="/0/DiscoverCandidateService"
     eOpposite="/0/DiscoverCandidateService/performedBy"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="performEnactment" ordered="false" upperBound="-1"
     eType="/0/EnactService"
     eOpposite="/0/EnactService/performedBy"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="advertise" ordered="false" upperBound="-1"
     eType="/0/OWL_S_Profile"
     eOpposite="/0/OWL_S_Profile/advertisedBy"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="interactedBy" ordered="false" upperBound="-1"
     eType="/0/DiscoverCandidateService"
     eOpposite="/0/DiscoverCandidateService/interact"/>
  </eClassifiers>

  <eClassifiers xsi:type="ecore:EClass" name="OWL_S_Service">
    <eStructuralFeatures xsi:type="ecore:EAttribute"
     name="name" ordered="false" unique="false" lowerBound="1"
     eType="/1/String"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="interface" ordered="false" eType="/0/OWL_S_Profile"
     eOpposite="/0/OWL_S_Profile/interfacedBy"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="process" ordered="false" eType="/0/OWL_S_Process"
```

```
      eOpposite="/0/OWL_S_Process/processedBy"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="support" ordered="false" eType="/0/OWL_S_Grounding"
     eOpposite="/0/OWL_S_Grounding/supportedBy"/>
  </eClassifiers>

  <eClassifiers xsi:type="ecore:EClass" name="OWL_S_Profile">
    <eStructuralFeatures xsi:type="ecore:EAttribute"
     name="name" ordered="false" unique="false" lowerBound="1"
     eType="/1/String"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="interfacedBy" ordered="false"
     eType="/0/OWL_S_Service"
     eOpposite="/0/OWL_S_Service/interface"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="advertisedBy" ordered="false" eType="/0/Agent"
     eOpposite="/0/Agent/advertise"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="foundedBy" ordered="false"
     eType="/0/DiscoverCandidateService"
     eOpposite="/0/DiscoverCandidateService/find"/>
  </eClassifiers>

  <eClassifiers xsi:type="ecore:EClass" name="OWL_S_Process">
    <eStructuralFeatures xsi:type="ecore:EAttribute"
     name="name" ordered="false" unique="false" lowerBound="1"
     eType="/1/String"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="processedBy" ordered="false" eType="/0/OWL_S_Service"
     eOpposite="/0/OWL_S_Service/process"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="executedBy" ordered="false" eType="/0/EnactService"
     eOpposite="/0/EnactService/execute"/>
  </eClassifiers>

  <eClassifiers xsi:type="ecore:EClass" name="OWL_S_Grounding">
    <eStructuralFeatures xsi:type="ecore:EAttribute"
     name="name" ordered="false" unique="false" lowerBound="1"
     eType="/1/String"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="supportedBy" ordered="false" eType="/0/OWL_S_Service"
     eOpposite="/0/OWL_S_Service/support"/>
    <eStructuralFeatures xsi:type="ecore:EReference"
     name="invokedBy" ordered="false" eType="/0/EnactService"
     eOpposite="/0/EnactService/invoke"/>
  </eClassifiers>
</ecore:EPackage>

<ecore:EPackage name="PrimitiveTypes">
  <eClassifiers xsi:type="ecore:EDataType" name="String"/>
</ecore:EPackage>
</xmi:XMI>
```

Ek 3 Anlamsal Web Ortamında Çalışan MAS'lara ait PIMM ve SEAGENT MAS Geliştirme Çerçevesine ait PSMM Varlıkları Arasındaki Model Dönüşümlerini Sağlayan ATL Kuralları

```
module Agent2SEAGENT; -- Module Sablonu
create OUT : SEAGENT from IN : Agent;

-- Asagidaki yardimci kurallar desen eslemede uygun elemanlarin
secilmesi amaciyla asil donusum kurallari tarafindan
kullanilacaklardir.

helper context Agent!SemanticWebAgent def:
  partofPatternforWebAgent : Boolean =
    if not self.oclIsTypeOf(Agent!SemanticServiceMatchmakerAgent)
       and not self.play.oclIsTypeOf(Agent!RegistryRole)
       and self.apply->select
       (p|p.oclIsTypeOf(Agent!SemanticServiceExecutorPlan))->
       forAll(p|p.execute.owner = p.use.owner)
       and self.apply->
       select(p|p.oclIsTypeOf(Agent!SemanticServiceFinderPlan))->
       forAll(p| p.discover.oclIsTypeOf(Agent!Interface))
       then
         true
       else
         false
    endif;

helper context Agent!SemanticServiceMatchmakerAgent def:
  partofPatternforMatchmaker : Boolean =
    if self.oclIsTypeOf(Agent!SemanticServiceMatchmakerAgent)
       and self.play.oclIsTypeOf(Agent!RegistryRole)
       and self.apply->
       select(p|p.oclIsTypeOf(Agent!SemanticServiceRegisterPlan))
       ->forAll(p| p.advertise->
       exists(intfc|intfc.discoveredBy.interact=self))
       then
         true
       else
          false
    endif;

helper context Agent!SemanticServiceFinderPlan def:
  partofPatternforFinderPlan : Boolean =
    if self.appliedBy->forAll(agnt| not
       agnt.oclIsTypeOf(Agent!SemanticServiceMatchmakerAgent)
       and not agnt.play.oclIsTypeOf(Agent!RegistryRole))
       and self.interact.play.oclIsTypeOf(Agent!RegistryRole)
       and self.interact.apply->
     select(p|p.oclIsTypeOf(Agent!SemanticServiceRegisterPlan))->
     forAll(p|p.advertise->exists(intfc|intfc.discoveredBy=self))
```

```
      and self.discover.advertisedBy.appliedBy->
      exists(sma|sma.interactedBy=self)
      then
        true
      else
        false
    endif;

helper context Agent!SemanticServiceExecutorPlan def:
  partofPatternforExecutorPlan : Boolean =
    if self.appliedBy->forAll(agnt|not
       agnt.oclIsTypeOf(Agent!SemanticServiceMatchmakerAgent)
       and not agnt.play.oclIsTypeOf(Agent!RegistryRole))
       and self.execute.owner = self.use.owner
       and self.appliedBy->
 select(agt|agt.oclIsTypeOf(Agent!SemanticServiceMatchmakerAgent)
       and agt.play.oclIsTypeOf(Agent!RegistryRole))->
       forAll(agt|agt.apply->
       select(p|p.oclIsTypeOf(Agent!SemanticServiceFinderPlan))->
       forAll(p| p.interact.advertise->
       exists(intfc|intfc=p.discover)))
       then
         true
       else
         false
    endif;

helper context Agent!SemanticServiceRegisterPlan def:
  partofPatternforRegisterPlan : Boolean =
    if self.appliedBy->forAll(
     agnt|agnt.oclIsTypeOf(Agent!SemanticServiceMatchmakerAgent))
       then
         true
       else
         false
    endif;

helper context Agent!RoleOntology def:
  partofPatternforRoleOntology : Boolean =
    if self.knownByBehavior->
     select(bhvr|bhvr.includedBy.oclIsTypeOf(Agent!DomainRole))->
       forAll(bhavr|bhavr.includedBy=self.usedByDomainRole)
       then
         true
       else
         false
    endif;

helper context Agent!ServiceOntology def:
  partofPatternforServiceOntology : Boolean =
    if self.dependedBy->
       select(sws|sws.oclIsTypeOf(Agent!RegistryRole))->
       forAll(rgrl|rgrl = self.knownByRegistryRole)
```

```
      then
        true
      else
        false
    endif;

helper context Agent!OrganizationOntology def:
  partofPatternforOrganizationOntology : Boolean =
    if self.knownByRole->select(rl|rl.oclIsTypeOf(Agent!Role))->
       forAll(rl|rl.knowOrganizationOntology = self)
       then
         true
       else
         false
    endif;

helper context Agent!Behavior def:
  partofPatternBehavior : Boolean =
    if self.knowRoleOntology->
       forAll(rlont|rlont.usedByDomainRole.include->
       exists(bhvr|bhvr = self))
       or self.includedBy.oclIsTypeOf(Agent!RegistryRole)
       or self.includedBy.oclIsTypeOf(Agent!OntologyMediatorRole)
       then
         true
       else
         false
       endif;

helper context Agent!SemanticWebService def:
  partofPatternService : Boolean =
    if self.advertisedBy.oclIsTypeOf(Agent!RegistryRole)
       and self.interface.oclIsTypeOf(Agent!Interface)
       and self.process.oclIsTypeOf(Agent!Process)
       and self.grounding.oclIsTypeOf(Agent!Grounding)
       then
         true
       else
         false
    endif;

helper context Agent!Interface def:
  partofPatternInterface : Boolean =
    if self.owner.oclIsTypeOf(Agent!SemanticWebService)
       and self.advertisedBy.appliedBy->
       exists(agnt|agnt.play = self.owner.advertisedBy)
       then
         true
       else
         false
    endif;

helper context Agent!Process def:
```

```
  partofPatternProcess : Boolean =
    if self.owner.oclIsTypeOf(Agent!SemanticWebService)
       and self.executedBy.appliedBy->exists(agnt|agnt.apply->
    select(pln|pln.oclIsTypeOf(Agent!SemanticServiceFinderPlan))->
       exists(pln|pln.discover.owner = self.owner))
       then
         true
       else
         false
    endif;

helper context Agent!Grounding def:
  partofPatternGrounding : Boolean =
    if self.owner.oclIsTypeOf(Agent!SemanticWebService)
       and self.usedBy.appliedBy->exists(agnt|agnt.apply->
    select(pln|pln.oclIsTypeOf(Agent!SemanticServiceFinderPlan))->
       exists(pln|pln.discover.owner = self.owner))
       then
         true
       else
         false
     endif;

-- Asagidaki yardimci kurallar hedef elemanlari arasindaki
iliskileri belirlemek amaciyla asil donusum kurallari tarafindan
kullanilacaklardir.

helper context Agent!SemanticWebAgent def:
  executorPlans : Sequence(Agent!SemanticServiceExecutorPlan) =
    self.apply->select(
    excpln|excpln.oclIsTypeOf(Agent!SemanticServiceExecutorPlan)
    and excpln.appliedBy->forAll(agnt|not
    agnt.oclIsTypeOf(Agent!SemanticServiceMatchmakerAgent)
    and not agnt.play.oclIsTypeOf(Agent!RegistryRole)))->
    select(pln|pln.execute.owner = pln.use.owner
    and pln.appliedBy->
 select(agt|agt.oclIsTypeOf(Agent!SemanticServiceMatchmakerAgent)
    and agt.play.oclIsTypeOf(Agent!RegistryRole))->
    forAll(agt|agt.apply->
    select(p|p.oclIsTypeOf(Agent!SemanticServiceFinderPlan))->
    forAll(p| p.interact.advertise->
    exists(intfc|intfc=p.discover))));

helper context Agent!SemanticWebAgent def:
  finderPlans : Sequence(Agent!SemanticServiceFinderPlan) =
    self.apply->select(
    fp|fp.oclIsTypeOf(Agent!SemanticServiceFinderPlan))->
    select(fndpln| fndpln.appliedBy->
    forAll(agnt| not
    agnt.oclIsTypeOf(Agent!SemanticServiceMatchmakerAgent)
    and not agnt.play.oclIsTypeOf(Agent!RegistryRole))
    and fndpln.interact.play.oclIsTypeOf(Agent!RegistryRole)
    and fndpln.interact.apply->
```

```
    select(p|p.oclIsTypeOf(Agent!SemanticServiceRegisterPlan))->
    forAll(p| p.advertise->
    exists(intfc|intfc.discoveredBy=fndpln))
    and fndpln.discover.advertisedBy.appliedBy->
    exists(sma|sma.interactedBy=fndpln));

helper context Agent!SemanticServiceMatchmakerAgent def:
  registerPlans : Sequence(Agent!SemanticServiceRegisterPlan) =
  self.apply->select(
  rg|rg.oclIsTypeOf(Agent!SemanticServiceRegisterPlan))->
  select(regpln| regpln.appliedBy->forAll(agnt|
  agnt.oclIsTypeOf(Agent!SemanticServiceMatchmakerAgent)
  and agnt.play.oclIsTypeOf(Agent!RegistryRole)));

-- Kaynak model elemanlarindan hedef model elemanlarini ureten
asil donusum kurallari

rule SemanticWebAgent2Agent {
     from
       ag: Agent!SemanticWebAgent (
           ag.partofPatternforWebAgent
       )
     to
       sa: SEAGENT!Agent (
           name <- ag.name,
           performDiscovery <- ag.finderPlans,
           performEnactment <- ag.executorPlans
       )
}

rule SemanticMatchmakerAgent2Agent {
     from
       ag: Agent!SemanticServiceMatchmakerAgent (
           ag.partofPatternforMatchmaker
       )
     to
       sam: SEAGENT!Agent (
            name<- ag.name,
            demonstrate <- ag.registerPlans,
            advertise <- Sequence {
                   Agent!Interface->allInstances()->asSequence()}
       )
}

rule SemanticServiceFinderPlan2DiscoverCandidateService {
     from
       fndpln: Agent!SemanticServiceFinderPlan (
               fndpln.partofPatternforFinderPlan
       )
     to
       plnf: SEAGENT!DiscoverCandidateService (
             name <- fndpln.name,
             containTask <- Sequence{
```

```
                Agent!Behavior->allInstances()->asSequence()->
                select(bhv|bhv.execute->exists(pln|pln=fndpln))},
             interact <- Sequence {
                Agent!SemanticServiceMatchmakerAgent->
                allInstances()->asSequence()},
             find <- Sequence {
                Agent!Interface->allInstances()->asSequence()}
      )
}

rule SemanticServiceExecutorPlan2EnactService {
     from
       smtpln: Agent!SemanticServiceExecutorPlan (
               smtpln.partofPatternforExecutorPlan
       )
     to
       plnf: SEAGENT!EnactService (
             name <- smtpln.name,
             containTask <- Sequence{
                Agent!Behavior->allInstances()->asSequence()->
                select(bhv|bhv.execute->exists(pln|pln=smtpln))},
             execute <- Sequence {
                Agent!Process->allInstances()->asSequence()},
              invoke <- Sequence {
                Agent!Grounding->allInstances()->asSequence()}
       )
}

rule SemanticServiceRegisterPlan2Behaviour {
     from
       regpln: Agent!SemanticServiceRegisterPlan(
               regpln.partofPatternforRegisterPlan
       )
     to
       plnr: SEAGENT!Behaviour (
             name <- regpln.name,
             containTask <- Sequence{
                Agent!Behavior->allInstances()->asSequence()->
                select(bhv|bhv.execute->exists(pln|pln=regpln))}
       )
}

rule Behavior2Action {
     from
       bhv: Agent!Behavior(
            bhv.partofPatternBehavior
       )
     to
       act: SEAGENT!Action (
            name <- bhv.name
       )
}
```

```
rule SemanticWebService2OWL_S_Service {
     from
       srv: Agent!SemanticWebService (
            srv.partofPatternService
       )
     to
       ifcOWL: SEAGENT!OWL_S_Service(
                name<-srv.name,
                interface<-srv.interface,
                process<-srv.process,
                support<-srv.grounding
       )
}

rule Interface2OWL_S_Profile {
     from
       ifc: Agent!Interface (
            ifc.partofPatternInterface
       )
     to
       ifcOWL: SEAGENT!OWL_S_Profile(
               name<-ifc.name
       )
}

rule Process2OWL_S_Process {
     from
       prc: Agent!Process (
            prc.partofPatternProcess
       )
     to
       prcOWL: SEAGENT!OWL_S_Process(
               name<-prc.name
       )
}

rule Grounding2OWL_S_Grounding {
     from
       grd: Agent!Grounding (
            grd.partofPatternGrounding
       )
     to
       grdOWL: SEAGENT!OWL_S_Grounding(
               name<-grd.name
       )
}
```

Ek 4 Turizm İş Alanında Çalışan MAS'lara ait PIM'in Ecore Gösterimi

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
    xmlns="Agent">
  <SemanticWebAgent name="Otel Istemci Etmeni" apply="#/4 #/5"
      play="#/2"/>
  <SemanticServiceMatchmakerAgent name="Esleme Etmeni"
      apply="#/16" play="#/3" interactedBy="#/4"/>
  <DomainRole name="Otel Istemci Rolu" playedBy="#/0"
      include="#/10 #/11" knowOrganizationOntology="#/13"
      useRoleOntology="#/12"/>
  <RegistryRole name="Anlamsal Servis Esleme Rolu" playedBy="#/1"
      include="#/14" knowServiceOntology="#/15"
      advertiseService="#/6"/>
  <SemanticServiceFinderPlan name="Otel Istemcisinin Servis Kesif
      Plani" appliedBy="#/0" executedBy="#/10" interact="#/1"
      discover="#/7"/>
  <SemanticServiceExecutorPlan name="Otel Istemcisinin Servis
     Cagirma Plani" appliedBy="#/0" executedBy="#/11"
     execute="#/8" use="#/9"/>
  <SemanticWebService name="Rezervasyon Tumlesik Servisi"
     interface="#/7" process="#/8" grounding="#/9" depend="#/15"
     advertisedBy="#/3"/>
  <Interface name="Rezervasyon Servisi Arayuzu" owner="#/6"
     advertisedBy="#/16" discoveredBy="#/4"/>
  <Process name="Rezervasyon Servisi Sureci" owner="#/6"
     executedBy="#/5"/>
  <Grounding name="Rezervasyon Servisi Zemini" owner="#/6"
     usedBy="#/5"/>
  <Behavior name="Servis Kayitcisini Sorgula" includedBy="#/2"
     execute="#/4" knowRoleOntology="#/12"/>
  <Behavior name="Servisi Cagir" includedBy="#/2" execute="#/5"
     knowRoleOntology="#/12"/>
  <RoleOntology name="EtmenRolu.owl" knownByBehavior="#/10 #/11
     #/14" usedByDomainRole="#/2"/>
  <OrganizationOntology name="OtelRezervasyonMAS.owl"
     knownByRole="#/2"/>
  <Behavior name="Servis Kaydet" includedBy="#/3" execute="#/16"
     knowRoleOntology="#/12"/>
  <ServiceOntology name="Servisler.owl" knownByRegistryRole="#/3"
     dependedBy="#/6"/>
  <SemanticServiceRegisterPlan name="Esleme Etmeninin Servis
     Kayit Plani" appliedBy="#/1" advertise="#/7"
     executedBy="#/14"/>
</xmi:XMI>
```

Ek 5 Turizm İş Alanında Çalışan MAS'lara ait SEAGENT PSM'in Ecore Gösterimi

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="SEAGENT">
  <Agent name="Otel Istemci Etmeni" performDiscovery="/2"
      performEnactment="/3"/>
  <Agent name="Esleme Etmeni" demonstrate="/4" advertise="/6"
      interactedBy="/2"/>
  <DiscoverCandidateService name="Otel Istemcisinin Servis Kesif
      Plani" interact="/1" find="/6" performedBy="/0">
    <containTask xsi:type="Action" name="Servis Kayitcisini
       Sorgula"/>
  </DiscoverCandidateService>
  <EnactService name="Otel Istemcisinin Servis Cagirma Plani"
      execute="/7" invoke="/8" performedBy="/0">
    <containTask xsi:type="Action" name="Servisi Cagir"/>
  </EnactService>
  <Behaviour name="Esleme Etmeninin Servis Kayit Plani"
      demonstratedBy="/1">
    <containTask xsi:type="Action" name="Servis Kaydet"/>
  </Behaviour>
  <OWL_S_Service name="Rezervasyon Tumlesik Servisi"
      interface="/6" process="/7" support="/8"/>
  <OWL_S_Profile name="Rezervasyon Servisi Arayuzu"
      interfacedBy="/5" advertisedBy="/1" foundedBy="/2"/>
  <OWL_S_Process name="Rezervasyon Servisi Sureci"
      processedBy="/5" executedBy="/3"/>
  <OWL_S_Grounding name="Rezervasyon Servisi Zemini"
      supportedBy="/5" invokedBy="/3"/>
</xmi:XMI>
```

Ek 6 NUIN MAS Geliştirme Çerçevesine ait PSMM'in Ecore Gösterimi

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore">
  <ecore:EPackage name="NUIN">

    <eClassifiers xsi:type="ecore:EClass" name="NUINAgent">
      <eStructuralFeatures xsi:type="ecore:EAttribute"
       name="name" ordered="false" unique="false" lowerBound="1"
       eType="/1/String"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="use"
       ordered="false" upperBound="-1"
       eType="/0/NamespaceDeclaration"
       eOpposite="/0/NamespaceDeclaration/usedBy"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="believe" ordered="false" upperBound="-1"
       eType="/0/KnowledgestoreDeclaration"
       eOpposite="/0/KnowledgestoreDeclaration/believedBy"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="contain" upperBound="-1" eType="/0/Plan"
       containment="true" eOpposite="/0/Plan/containedBy"/>
    </eClassifiers>

    <eClassifiers xsi:type="ecore:EClass"
       name="NamespaceDeclaration">
      <eStructuralFeatures xsi:type="ecore:EAttribute"
       name="name" ordered="false" unique="false" lowerBound="1"
       eType="/1/String"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="usedBy" ordered="false" lowerBound="1"
       eType="/0/NUINAgent" eOpposite="/0/NUINAgent/use"/>
    </eClassifiers>

    <eClassifiers xsi:type="ecore:EClass"
       name="KnowledgestoreDeclaration">
      <eStructuralFeatures xsi:type="ecore:EAttribute"
       name="name" ordered="false" unique="false" lowerBound="1"
       eType="/1/String"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="believedBy" ordered="false" lowerBound="1"
       eType="/0/NUINAgent" eOpposite="/0/NUINAgent/believe"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="containOntology" upperBound="-1"
       eType="/0/OntologySpecification" containment="true"
       eOpposite="/0/OntologySpecification/containedBy"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
       name="containAxiom" upperBound="-1" eType="/0/Axiom"
       containment="true" eOpposite="/0/Axiom/containedBy"/>
```

```
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass"
   name="OntologySpecification">
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="name" ordered="false" unique="false" lowerBound="1"
   eType="/1/String"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="containedBy" ordered="false" upperBound="-1"
   eType="/0/KnowledgestoreDeclaration"
   eOpposite="/0/KnowledgestoreDeclaration/containOntology"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass" name="Axiom">
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="name" ordered="false" unique="false" lowerBound="1"
   eType="/1/String"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="containedBy" ordered="false" upperBound="-1"
   eType="/0/KnowledgestoreDeclaration"
   eOpposite="/0/KnowledgestoreDeclaration/containAxiom"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass" name="Plan">
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="id"
   ordered="false" unique="false" lowerBound="1"
   eType="/1/String"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="containedBy" ordered="false" lowerBound="1"
   eType="/0/NUINAgent" eOpposite="/0/NUINAgent/contain"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="namedBy" ordered="false" lowerBound="1"
   eType="/0/Identifier" eOpposite="/0/Identifier/name"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="containPlanElement" upperBound="-1"
   eType="/0/PlanElement" containment="true"
   eOpposite="/0/PlanElement/containedBy"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass" name="Identifier">
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="id"
   ordered="false" unique="false" lowerBound="1"
   eType="/1/String"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
   name="name" ordered="false" lowerBound="1" eType="/0/Plan"
   eOpposite="/0/Plan/namedBy"/>
</eClassifiers>

<eClassifiers xsi:type="ecore:EClass" name="PlanElement">
  <eStructuralFeatures xsi:type="ecore:EAttribute"
   name="name" ordered="false" unique="false" lowerBound="1"
   eType="/1/String"/>
  <eStructuralFeatures xsi:type="ecore:EReference"
```

```
      name="containedBy" ordered="false" lowerBound="1"
      eType="/0/Plan" eOpposite="/0/Plan/containPlanElement"/>
    </eClassifiers>

    <eClassifiers xsi:type="ecore:EClass" name="Trigger"
      eSuperTypes="/0/PlanElement"/>

    <eClassifiers xsi:type="ecore:EClass" name="PostCondition"
      eSuperTypes="/0/PlanElement"/>

    <eClassifiers xsi:type="ecore:EClass" name="Body"
      eSuperTypes="/0/PlanElement">
      <eStructuralFeatures xsi:type="ecore:EReference"
      name="compose" lowerBound="1" upperBound="-1"
      eType="/0/PlanStep" containment="true"
      eOpposite="/0/PlanStep/composedBy"/>
    </eClassifiers>

    <eClassifiers xsi:type="ecore:EClass" name="PlanStep">
      <eStructuralFeatures xsi:type="ecore:EAttribute"
      name="name" ordered="false" unique="false" lowerBound="1"
      eType="/1/String"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
      name="composedBy" ordered="false" lowerBound="1"
      eType="/0/Body" eOpposite="/0/Body/compose"/>
    </eClassifiers>

    <eClassifiers xsi:type="ecore:EClass" name="Achieve"
      eSuperTypes="/0/PlanStep"/>
    <eClassifiers xsi:type="ecore:EClass" name="AddGoal"
      eSuperTypes="/0/PlanStep"/>
    <eClassifiers xsi:type="ecore:EClass" name="Commit"
      eSuperTypes="/0/PlanStep"/>
    <eClassifiers xsi:type="ecore:EClass" name="Intend"
      eSuperTypes="/0/PlanStep"/>
    <eClassifiers xsi:type="ecore:EClass" name="Invoke"
      eSuperTypes="/0/PlanStep"/>
    <eClassifiers xsi:type="ecore:EClass" name="Assert"
      eSuperTypes="/0/PlanStep"/>
    <eClassifiers xsi:type="ecore:EClass" name="InvokeService"
      eSuperTypes="/0/PlanStep"/>
    <eClassifiers xsi:type="ecore:EClass" name="SendMessage"
      eSuperTypes="/0/PlanStep"/>
    <eClassifiers xsi:type="ecore:EClass" name="DropGoal"
      eSuperTypes="/0/PlanStep"/>
    <eClassifiers xsi:type="ecore:EClass" name="PostEvent"
      eSuperTypes="/0/PlanStep"/>
  </ecore:EPackage>

  <ecore:EPackage name="PrimitiveTypes">
    <eClassifiers xsi:type="ecore:EDataType" name="String"/>
  </ecore:EPackage>
</xmi:XMI>
```

Ek 7 Anlamsal Web Ortamında Çalışan MAS'lara ait PIMM ve NUIN MAS Geliştirme Çerçevesine ait PSMM Varlıkları Arasındaki Model Dönüşümlerini Sağlayan ATL Kuralları

```
module Agent2NUIN; -- Module Sablonu
create OUT : NUIN from IN : Agent;

-- Asagidaki yardimci kurallar desen eslemede uygun elemanlarin
secilmesi amaciyla asil donusum kurallari tarafindan
kullanilacaklardir.

helper context Agent!SemanticWebAgent def:
  partofPatternforWebAgent : Boolean =
    if not self.oclIsTypeOf(Agent!SemanticServiceMatchmakerAgent)
       and not self.play.oclIsTypeOf(Agent!RegistryRole)
       and self.apply->select
       (p|p.oclIsTypeOf(Agent!SemanticServiceExecutorPlan))->
       forAll(p|p.execute.owner = p.use.owner)
       and self.apply->
       select(p|p.oclIsTypeOf(Agent!SemanticServiceFinderPlan))->
       forAll(p| p.discover.oclIsTypeOf(Agent!Interface))
       then
         true
       else
         false
    endif;

helper context Agent!SemanticServiceMatchmakerAgent def:
  partofPatternforMatchmaker : Boolean =
    if self.oclIsTypeOf(Agent!SemanticServiceMatchmakerAgent)
       and self.play.oclIsTypeOf(Agent!RegistryRole)
       and self.apply->
       select(p|p.oclIsTypeOf(Agent!SemanticServiceRegisterPlan))
       ->forAll(p| p.advertise->
       exists(intfc|intfc.discoveredBy.interact=self))
       then
         true
       else
          false
    endif;

helper context Agent!SemanticServiceFinderPlan def:
  partofPatternforFinderPlan : Boolean =
    if self.appliedBy->forAll(agnt| not
       agnt.oclIsTypeOf(Agent!SemanticServiceMatchmakerAgent)
       and not agnt.play.oclIsTypeOf(Agent!RegistryRole))
       and self.interact.play.oclIsTypeOf(Agent!RegistryRole)
       and self.interact.apply->
     select(p|p.oclIsTypeOf(Agent!SemanticServiceRegisterPlan))->
     forAll(p|p.advertise->exists(intfc|intfc.discoveredBy=self))
```

```
      and self.discover.advertisedBy.appliedBy->
      exists(sma|sma.interactedBy=self)
      then
        true
      else
        false
    endif;

helper context Agent!SemanticServiceExecutorPlan def:
  partofPatternforExecutorPlan : Boolean =
    if self.appliedBy->forAll(agnt|not
       agnt.oclIsTypeOf(Agent!SemanticServiceMatchmakerAgent)
       and not agnt.play.oclIsTypeOf(Agent!RegistryRole))
       and self.execute.owner = self.use.owner
       and self.appliedBy->
 select(agt|agt.oclIsTypeOf(Agent!SemanticServiceMatchmakerAgent)
       and agt.play.oclIsTypeOf(Agent!RegistryRole))->
       forAll(agt|agt.apply->
       select(p|p.oclIsTypeOf(Agent!SemanticServiceFinderPlan))->
       forAll(p| p.interact.advertise->
       exists(intfc|intfc=p.discover)))
       then
         true
       else
         false
    endif;

helper context Agent!SemanticServiceRegisterPlan def:
  partofPatternforRegisterPlan : Boolean =
    if self.appliedBy->forAll(
     agnt|agnt.oclIsTypeOf(Agent!SemanticServiceMatchmakerAgent))
       then
         true
       else
         false
    endif;

helper context Agent!RoleOntology def:
  partofPatternforRoleOntology : Boolean =
    if self.knownByBehavior->
     select(bhvr|bhvr.includedBy.oclIsTypeOf(Agent!DomainRole))->
       forAll(bhavr|bhavr.includedBy=self.usedByDomainRole)
       then
         true
       else
         false
    endif;

helper context Agent!ServiceOntology def:
  partofPatternforServiceOntology : Boolean =
    if self.dependedBy->
       select(sws|sws.oclIsTypeOf(Agent!RegistryRole))->
       forAll(rgrl|rgrl = self.knownByRegistryRole)
```

```
      then
        true
      else
        false
    endif;

helper context Agent!OrganizationOntology def:
  partofPatternforOrganizationOntology : Boolean =
    if self.knownByRole->select(rl|rl.oclIsTypeOf(Agent!Role))->
       forAll(rl|rl.knowOrganizationOntology = self)
      then
        true
      else
        false
    endif;

helper context Agent!Behavior def:
  partofPatternBehavior : Boolean =
    if self.knowRoleOntology->
       forAll(rlont|rlont.usedByDomainRole.include->
       exists(bhvr|bhvr = self))
       or self.includedBy.oclIsTypeOf(Agent!RegistryRole)
       or self.includedBy.oclIsTypeOf(Agent!OntologyMediatorRole)
       then
         true
       else
         false
       endif;

helper context Agent!SemanticWebService def:
  partofPatternService : Boolean =
    if self.advertisedBy.oclIsTypeOf(Agent!RegistryRole)
       and self.interface.oclIsTypeOf(Agent!Interface)
       and self.process.oclIsTypeOf(Agent!Process)
       and self.grounding.oclIsTypeOf(Agent!Grounding)
       then
         true
       else
         false
    endif;

helper context Agent!Interface def:
  partofPatternInterface : Boolean =
    if self.owner.oclIsTypeOf(Agent!SemanticWebService)
       and self.advertisedBy.appliedBy->
       exists(agnt|agnt.play = self.owner.advertisedBy)
       then
         true
       else
         false
    endif;

helper context Agent!Process def:
```

```
  partofPatternProcess : Boolean =
    if self.owner.oclIsTypeOf(Agent!SemanticWebService)
       and self.executedBy.appliedBy->exists(agnt|agnt.apply->
    select(pln|pln.oclIsTypeOf(Agent!SemanticServiceFinderPlan))->
       exists(pln|pln.discover.owner = self.owner))
       then
         true
       else
         false
    endif;

helper context Agent!Grounding def:
  partofPatternGrounding : Boolean =
    if self.owner.oclIsTypeOf(Agent!SemanticWebService)
       and self.usedBy.appliedBy->exists(agnt|agnt.apply->
    select(pln|pln.oclIsTypeOf(Agent!SemanticServiceFinderPlan))->
       exists(pln|pln.discover.owner = self.owner))
       then
         true
       else
         false
     endif;

-- Asagidaki yardimci kurallar hedef elemanlari arasindaki
iliskileri belirlemek amaciyla asil donusum kurallari tarafindan
kullanilacaklardir.

helper context Agent!SemanticWebAgent def:
  executorPlans : Sequence(Agent!SemanticServiceExecutorPlan) =
    self.apply->select(
    excpln|excpln.oclIsTypeOf(Agent!SemanticServiceExecutorPlan)
    and excpln.appliedBy->forAll(agnt|not
    agnt.oclIsTypeOf(Agent!SemanticServiceMatchmakerAgent)
    and not agnt.play.oclIsTypeOf(Agent!RegistryRole)))->
    select(pln|pln.execute.owner = pln.use.owner
    and pln.appliedBy->
 select(agt|agt.oclIsTypeOf(Agent!SemanticServiceMatchmakerAgent)
    and agt.play.oclIsTypeOf(Agent!RegistryRole))->
    forAll(agt|agt.apply->
    select(p|p.oclIsTypeOf(Agent!SemanticServiceFinderPlan))->
    forAll(p| p.interact.advertise->
    exists(intfc|intfc=p.discover))));

helper context Agent!SemanticWebAgent def:
  finderPlans : Sequence(Agent!SemanticServiceFinderPlan) =
    self.apply->select(
    fp|fp.oclIsTypeOf(Agent!SemanticServiceFinderPlan))->
    select(fndpln| fndpln.appliedBy->
    forAll(agnt| not
    agnt.oclIsTypeOf(Agent!SemanticServiceMatchmakerAgent)
    and not agnt.play.oclIsTypeOf(Agent!RegistryRole))
    and fndpln.interact.play.oclIsTypeOf(Agent!RegistryRole)
    and fndpln.interact.apply->
```

```
    select(p|p.oclIsTypeOf(Agent!SemanticServiceRegisterPlan))->
    forAll(p| p.advertise->
    exists(intfc|intfc.discoveredBy=fndpln))
    and fndpln.discover.advertisedBy.appliedBy->
    exists(sma|sma.interactedBy=fndpln));

helper context Agent!SemanticServiceMatchmakerAgent def:
  registerPlans : Sequence(Agent!SemanticServiceRegisterPlan) =
  self.apply->select(
  rg|rg.oclIsTypeOf(Agent!SemanticServiceRegisterPlan))->
  select(regpln| regpln.appliedBy->forAll(agnt|
  agnt.oclIsTypeOf(Agent!SemanticServiceMatchmakerAgent)
  and agnt.play.oclIsTypeOf(Agent!RegistryRole)));

-- Kaynak model elemanlarindan hedef model elemanlarini ureten
asil donusum kurallari

rule SemanticWebAgent2NUINAgent {
     from
       ag: Agent!SemanticWebAgent (
           ag.partofPatternforWebAgent
       )
     to
       na: NUIN!NUINAgent (
           name <- ag.name,
           contain <- Sequence{ag.executorPlans, ag.finderPlans},
           believe <- ksdec
        ),
        ksdec: NUIN!KnowledgestoreDeclaration
}

rule SemanticMatchmakerAgent2NUINAgent {
     from
       ag: Agent!SemanticServiceMatchmakerAgent (
           ag.partofPatternforMatchmaker
       )
     to
       nam: NUIN!NUINAgent (
            name<- ag.name,
            contain <- Sequence{ag.registerPlans},
            believe <- ksdecm
       ),
       ksdecm:NUIN!KnowledgestoreDeclaration
}

rule SemanticServiceFinderPlan2Plan {
     from
       fndpln: Agent!SemanticServiceFinderPlan (
               fndpln.partofPatternforFinderPlan
       )
     to
       plnf: NUIN!Plan (
             containedBy <- Agent!SemanticWebAgent->
```

```
            allInstances()->asSequence()->select(agt|agt.apply->
            exists(pln|pln=fndpln))->first()
      ),
      identifier: NUIN!Identifier (
                  id <- fndpln.name,
                  name <- Agent!SemanticServiceFinderPlan->
                  allInstances()->asSequence()->
                  select(pln|pln=fndpln)->first()
      ),
      plnfBody: NUIN!Body (
                name <- 'Finder Plan Body',
                containedBy <- plnf,
                compose <- Sequence{Agent!Behavior->
                allInstances()->asSequence()->
                select(bhv|bhv.execute->exists(pln|pln=fndpln))}
      )
}

rule SemanticServiceExecutorPlan2Plan {
     from
      smtpln: Agent!SemanticServiceExecutorPlan (
              smtpln.partofPatternforExecutorPlan
      )
     to
      plne: NUIN!Plan (
            containedBy <- Agent!SemanticWebAgent->
              allInstances()->asSequence()->
              select(agt|agt.apply->
              exists(pln|pln=smtpln))->first()
      ),
      identifier: NUIN!Identifier (
                  id <- smtpln.name,
                  name <- Agent!SemanticServiceExecutorPlan->
                          allInstances()->asSequence()->
                          select(pln|pln=smtpln)->first()
      ),
      plneBody: NUIN!Body (
                name <- 'Executor Plan Body',
                containedBy <- plne,
                compose <- Sequence{Agent!Behavior->
                allInstances()->asSequence()->
                select(bhv|bhv.execute->exists(pln|pln=smtpln))}
      )
}

rule SemanticServiceRegisterPlan2Plan {
     from
      regpln: Agent!SemanticServiceRegisterPlan (
              regpln.partofPatternforRegisterPlan
      )
     to
      plnr: NUIN!Plan (
            containedBy <- Agent!SemanticServiceMatchmakerAgent->
```

```
                allInstances()->asSequence()->
                select(agt|agt.apply->
                exists(pln|pln=regpln))->first()
        ),
        identifier: NUIN!Identifier (
                    id <- regpln.name,
                    name <- Agent!SemanticServiceRegisterPlan->
                      allInstances()->asSequence()->
                      select(pln|pln=regpln)->first()
        ),
        plnrBody: NUIN!Body (
                   name <- 'Register Plan Body',
                   containedBy <- plnr,
                   compose <- Sequence{Agent!Behavior->
                     allInstances()->asSequence()->
                     select(bhv|bhv.execute->
                     exists(pln|pln=regpln))}
        )
}

rule Behavior2PlanStep {
  from
    bhv: Agent!Behavior (
         bhv.partofPatternBehavior
    )
  to
    ps: NUIN!PlanStep (
        name <- bhv.name
    )
}

rule RoleOntology2OntologySpecification {
  from
    ro: Agent!RoleOntology (
        ro.partofPatternforRoleOntology
    )
  to
    ros: NUIN!OntologySpecification (
         name <- ro.name,
         containedBy <- Sequence{
           thisModule.resolveTemp(Agent!SemanticWebAgent->
             allInstances()->asSequence()->first(),'ksdec'),
           thisModule.resolveTemp(
             Agent!SemanticServiceMatchmakerAgent->
             allInstances()->asSequence()->first(),'ksdecm')}
    )
}

rule OrganizationOntology2OntologySpecification {
     from
       oo: Agent!OrganizationOntology (
           oo.partofPatternforOrganizationOntology
       )
```

```
      to
        oos: NUIN!OntologySpecification (
             name <- oo.name,
             containedBy <- Sequence{
                thisModule.resolveTemp(Agent!SemanticWebAgent->
                  allInstances()->asSequence()->first(),'ksdec'),
                thisModule.resolveTemp(
                  Agent!SemanticServiceMatchmakerAgent->
                  allInstances()->asSequence()->first(),'ksdecm')}
         )
}

rule ServiceOntology2OntologySpecification {
      from
        so: Agent!ServiceOntology (
            so.partofPatternforServiceOntology
         )
      to
        sos: NUIN!OntologySpecification (
             name <- so.name,
             containedBy <- thisModule.resolveTemp(
              Agent!SemanticServiceMatchmakerAgent->
              allInstances()->asSequence()->first(),'ksdecm')
        )
}
```

Ek 8 Turizm İş Alanında Çalışan MAS'lara ait NUIN PSM'in Ecore Gösterimi

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="NUIN">
  <NUINAgent name="Otel Istemci Etmeni" believe="/1">
    <contain namedBy="/5">
      <containPlanElement xsi:type="Body"
          name="Executor Plan Body">
        <compose name="Servisi Cagir"/>
      </containPlanElement>
    </contain>
    <contain namedBy="/4">
      <containPlanElement xsi:type="Body"
          name="Finder Plan Body">
        <compose name="Servis Kayitcisini Sorgula"/>
      </containPlanElement>
    </contain>
  </NUINAgent>
  <KnowledgestoreDeclaration believedBy="/0">
    <containOntology name="EtmenRolu.owl"/>
    <containOntology name="OtelRezervasyonMAS.owl"/>
  </KnowledgestoreDeclaration>
  <NUINAgent name="Esleme Etmeni" believe="/3">
    <contain namedBy="/6">
      <containPlanElement xsi:type="Body"
          name="Register Plan Body">
        <compose name="Servis Kaydet"/>
      </containPlanElement>
    </contain>
  </NUINAgent>
  <KnowledgestoreDeclaration believedBy="/2">
    <containOntology name="EtmenRolu.owl"/>
    <containOntology name="OtelRezervasyonMAS.owl"/>
    <containOntology name="Servisler.owl"/>
  </KnowledgestoreDeclaration>
  <Identifier id="Otel Istemcisinin Servis Kesif Plani"
      name="/0/@contain.1"/>
  <Identifier id="Otel Istemcisinin Servis Cagirma Plani"
      name="/0/@contain.0"/>
  <Identifier id="Esleme Etmeninin Servis Kayit Plani"
      name="/2/@contain.0"/>
</xmi:XMI>
```

Ek 9 NUIN PSM'lerinden Şablon Nuinscript Kodlarını Üreten MOFScript M2T Dönüşüm Betiği

```
/**
 * Donusum Adi: NUIN_PSM_2_NUINScript
 * Bu donusum Ecore biciminde verilmis ve NUIN platforumuna ozgu
 * bir MAS modelinden Nuinscript olusturmayi saglar
 * Yazar: Geylani Kardas
 * Tarih: 29 Subat 2008
 */

texttransformation NUIN_PSM_2_NUINScript (in nuin:"http:///NUIN")
{
  main(){
     //Modeldeki her bir NUIN etmenini bul
      nuin.objectsOfType(nuin.NUINAgent)->forEach(agent) {
             agent.createScript()
      }
  }

  //Her NUIN etmeni icin betik dosyasi olusturulur
  nuin.NUINAgent::createScript() {
    file(self.name+".ns")
    '************************************************************'
    '\n* Nuinscript\'i uretilen etmen: '+self.name
    '\n* Betik sablonunun olusturulma zamani: '+date()+' '+time()
    '\n************************************************************'

    //Isim Uzayi tanimlari
    '\n\n//isim uzay(lar)inin tanimi'
    //Varsayilan isim uzayini tanimla
    '\nuse ag for <urn:x-nuin-hotelreservation:agent:>.'
    //Eger varsa diger isim uzay(lar)ini tanimla
    self.useNamespace->
    forEach(namespace: nuin.NamespaceDeclaration) {
      '\nuse '+namespace.name+'.'  }
    '\n'

    //Etmenin inanclari (bilgi deposu tanimlari)
    '\n//baslangictaki ortam gerceklerinin taninmlanmasi'
    self.believe->
    forEach(knowledgeStore: nuin.KnowledgestoreDeclaration) {
      if (!knowledgeStore.containOntology.isEmpty()) {
       '\naxioms '
       knowledgeStore.containOntology->
       forEach(ontology: nuin.OntologySpecification) {
             if (knowledgeStore.containOntology.last()=ontology)
                 ontology.name+'\nend.'
             else ontology.name+', '
       }
      }
```

```
      if (!knowledgeStore.containAxiom.isEmpty()) {
       '\naxioms contains'
       knowledgeStore.containAxiom->forEach(axiom: nuin.Axiom) {
              '\n'+axiom.name+'.'
       }
       '\nend.'
      }
    }

    //Etmenin planlari
    //ilk olarak ana plan tanimi
    '\n\n/*'
    '\n* Etmen planinin baslangic noktasi - baslangic olayi
         tarafindan tetiklenir'
    '\n*/'
    '\nplan ag:main'
    '\n\ttrigger'
      '\n\t\ton event( event:startup )'
    '\n\tdo'
     '\n\t\tprintln( "Baslatiliyor ..." );'
     self.contain->forEach(plan: nuin.Plan) {
       '\n\t\tinvoke ag:'+plan.namedBy.id+'();'
     }
     '\n\t\tprintln( "Hepsi tamamlandi." )'
     '\nend.'

     //Tumlestirilen planlar (eger varsa)
     '\n\n/*'
     '\n* Tumlestirilen planlar'
     '\n*/'
     self.contain->forEach(plan: nuin.Plan) {
       '\nplan ag:'+plan.namedBy.id+'()'
       //olay tetikleyicileri icin sablonu hazirla
       plan.containPlanElement->forEach(trigger: nuin.Trigger) {
                '\n\ttrigger'
                '\n\t\ton event(ag:'+trigger.name')'
       }
       //plan sonlanma kosullari icin sablonu hazirla
       plan.containPlanElement->
        forEach(postCondition: nuin.PostCondition) {
                '\n\tpostcondition'
                '\n\t\tag:'+postCondition.name
       }
       //plan adimlari icin sablonu hazirla
       plan.containPlanElement->forEach(body: nuin.Body) {
                '\n\tdo'
                body.compose->forEach(planStep: nuin.PlanStep) {
                     '\n\t\t'+planStep.name+'(); }
       }
       '\nend.'
    }
  }
}
```

Ek 10 Türkçe - İngilizce Terimler Sözlüğü

ajans	: agency
akıl yürütme	: reasoning
algılayıcı	: sensor
anlamsal	: semantic
anlamsal web	: semantic web
anlamsal web servisi	: semantic web service
ayrıştırma	: decomposition
bakış tipi	: viewtype
betik	: script
betikleme dili	: scripting language
biçimsel	: formal
bilgi deposu	: knowledgestore
bilgi tabanı	: knowledgebase
bilinçli	: deliberative
birlikte işlerlik	: interoperability
çerçeve	: framework
çıkarsama	: inference
çok-etmenli sistem	: multi-agent system
deneysel	: empirical
eklenti	: plug-in
etkileyici	: effector
etmen	: agent
etmenleştirilmemiş	: non-agentified
etmenleştirilmiş	: agentified
gösterim	: notation
içerik	: context
ihtiyaç	: provision

isim uzayı	: namespace
iş alanı	: domain
işletim sonucu	: outcome
karşıt eylemli	: reactive
katman köprüleme	: layer bridging
kullanıma izinli	: allowed-to-use
ontoloji	: ontology
örnek	: instance
özerk	: autonomous
proaktif	: proactive
sınıflayıcı	: classifier
sözdizim	: syntax
sözdizimsel	: syntactic
tümleşim	: composition
uzantı	: extension
uzatmak	: extend
ürün	: artifact
üstmodel	: metamodel
üst-üstmodel	: meta-metamodel
üstvarlık	: meta-entity
üstyapı	: superstructure
üstveri	: metadata
varlık	: entity
yan eklentili katmanlar	: layers with a sidecar
yazılım etmeni	: software agent
yerleşik	: built-in

ÖZGEÇMİŞ

1979 yılında Diyarbakır'da doğan Geylani Kardaş, 1997 yılında İzmir Bornova Anadolu Lisesi'ni bitirdikten sonra Ege Üniversitesi Bilgisayar Mühendisliği Bölümü'nde lisans öğrenimine başlamış ve bu bölümden 2001 yılında mezun olmuştur. Yine 2001 yılında yüksek lisans öğrenimine başladığı Ege Üniversitesi Uluslararası Bilgisayar Enstitüsü'nden 2003 yılında Bilgi Teknolojileri Bilim Dalı'nda yüksek lisans derecesi ile mezun olmuş ve aynı yıl içerisinde bu enstitüde doktora öğrenimine başlamıştır. 2001 yılından bugüne kadar da yine aynı enstitüde araştırma görevlisi olarak görev yapmaktadır.

Bugüne kadarki mesleki ilgi alanlarının özerk etmenler ve çok-etmenli sistemler, yazılım sistemlerinin model güdümlü geliştirilmesi, Anlamsal Web, akıllı kart teknolojileri ve nesneye dayalı yazılım geliştirme süreçleri ve metodolojileri olarak sıralanabileceği Geylani Kardaş'ın ilgili araştırma konularında çeşitli bilimsel dergi ve kitap serilerinde yayınlanan ve uluslararası yayın indekslerince taranan 10'dan fazla makalesi ve çeşitli ulusal ve uluslararası bilimsel konferanslarda sunulan bildirileri bulunmaktadır.