İSTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ

YAYINLA/ABONE OL MODELİ İLE KONU TABANLI ETMEN GÖÇÜ

YÜKSEK LİSANS TEZİ

Mustafa Akif KARZAN

İleri Teknolojiler Anabilim Dalı

Bilgisayar Bilimleri Programı

HAZİRAN 2013

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ

YAYINLA/ABONE OL MODELİ İLE KONU TABANLI ETMEN GÖÇÜ

YÜKSEK LİSANS TEZİ

Mustafa Akif KARZAN
(704091025)

İleri Teknolojiler Anabilim Dalı

Bilgisayar Bilimleri Programı

Tez Danışmanı: Prof. Dr. Nadia ERDOĞAN

HAZİRAN 2013

İTÜ, Bilişim Enstitüsü'nün 704091025 numaralı Yüksek Lisans Öğrencisi **Mustafa Akif KARZAN**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"YAYINLA/ABONE OL MODELİ İLE KONU TABANLI ETMEN GÖÇÜ"** başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Prof. Dr. Nadia Erdoğan** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. Coşkun Sönmez** Yıldız Teknik Üniversitesi	.....
	Doç. Dr. Turgay Altılar İstanbul Teknik Üniversitesi	.....

Teslim Tarihi : **03 Mayıs 2013**
Savunma Tarihi : **06 Haziran 2013**

Aileme,

ÖNSÖZ

Çalışmalarıma olan katkılarından dolayı tez danışmanım Prof. Dr. Nadia Erdoğan'a ve yoğun çalışma tempomu anlayışla karşılayıp beni destekleyen aileme teşekkür ederim.

Haziran 2013

Mustafa Akif KARZAN
Mühendis

İÇİNDEKİLER

Sayfa

KISALTMALAR

ACC : Agent Communication Channel (Etmen Haberleşme Kanalı)
ACL : Agent Communication Language (Etmen Haberleşme Dili)
AMS : Agent Management System (Etmen Yönetim Sistemi)
AMSD : Agent Management Service Description (Etmen Yön.Hiz. Tanımı)
EMS : Enterprise Message Service (Kurumsal Mesaj Hizmeti)
FIPA : Foundation of Intelligent Physical Agent (Akıllı Fiziksel Etmen Kur)
JADE : Java Agent Development Framework (Java Etmen Geliş. İskeleleti)
JAM : Java Agency Model (Java Etmen Modeli)
JMS : Java Message Service (Java Mesaj Hizmeti)
MOM : Message Oriented Middleware (Mesaj Tabanlı Ara Katman)

ŞEKİL LİSTESİ

Sayfa

YAYINLA/ABONE OL MODELİ İLE KONU TABANLI ETMEN GÖÇÜ

ÖZET

Gerçek dünya problemlerinin bilgisayar sistemleri ile çözülmek istenmesi, bu problemlerin bilgisayar sistemlerine aktarılması ve çözümün bu alanda yapılması gereğini ortaya koydu. Bu bağlamda bilgisayarlar ile hangi problemlerin çözülebileceği sorusu gündeme geldi ve ilk olarak mühendislik problemlerinin çözülmesi için mühendislik sistemleri ile ilgili problemlerin bilgisayarlara aktarılması ve bu problem uzaylarının bilgisayar sistemleri ile tarif edilmesi söz konusu oldu. Mühendislik sistemleri genel olarak belirli girişlere belirli çıkışlar üreten deterministik (kararlı) sistemlerdir. Bu sistemler, girişlere ilişkin çıkışları üretebilmek için bir takım fonksiyon kümesi ile matematiksel olarak tarif edilirler. Mesela işaret işleme, haberleşme teorisi ve kontrol teorisi alanlarında sistem tarifi matematiksel olarak transfer veya sistem fonksiyonu olarak adlandırılan fonksiyon ile sunulur. Bu fonksiyonun belirlenmesi ile sistem tasarlanmış olur. İşte bu alanlardaki problemleri bilgisayarlar ile çözebilmek için fonksiyonel/prosedürel programlama paradigması ortaya konmuştur. Fonksiyonel programlama ile sistem sahip olduğu fonksiyonlar bütünü ile tarif edilir ve bir ana başvuru noktasından (fonksiyonundan) bu fonksiyonların çağırılması ile bilgisayar alanında sistem gerçeklenmiş olur. Ancak bilgisayar teknolojisindeki hızlı gelişme ile güçlü işlem yapma kapasitesine sahip makinalara daha fazla farklı türde problem çözdürme isteği doğmuştur. Yani mühendislik problemlerinin dışındaki alanlara da bu domende çözüm bulunması gereği ortaya çıkmıştır. Mesela, bankacılık, sigortacılık, telekomünikasyon … gibi alanlarda ortaya konulan problemlerin bilgisayarlarla çözümü. Fakat bu alanlardaki problemlerin çözümü için gereken yazılım uygulamalarının geliştirilmesi fonksiyonel programlama ile mümkün olmamaktadır. Fonksiyonel programlama bu problemlerin bilgisayar dünyasına aktarılmasında çok yetersiz kalmaktadır. Bu amaçla nesneye yönelik programlama paradigması ortaya konmuştur. Nesne teknolojisinde, sistemdeki her şey nesneler olarak tarif edilmektedir. Bu tip sistemler de sistemdeki tüm nesnelerin oluşturulması, sorumluluklarının belirlenmesi (statik) ve bu nesneler arasındaki iletişimin/etkileşimin (dinamik) tarif edilmesi ile tasarlanır. Nesneler ile sistemlerin kaliteli ve hataya dayanıklı olarak tasarlanması, yazılım tasarım kalıp ve prensipleri kullanımı esasına dayanır. Ancak nesneler ile tasarımın dahi yetersiz kalması yazılım bileşenlerinin insan gibi düşünmesinin gereği, mental yazılım bileşenleri ile sistemlerin tasarlanması fikrini ortaya koymuştur. Bu fikir ile yepyeni bir programlama paradigması doğmuş ve adına da etmen odaklı programlama denmiştir.

Etmen odaklı programlamayı daha iyi anlayabilmek için önce etmen kavramını ve teknolojisini anlamak gerekir. Etmen kavramının iyice oturması ile etmenlerle düşünme yeteneği gelişir ve etmen odaklı tasarım gündeme gelir. Bu, insanlar için oldukça yakın bir yöntemdir çünkü insanlarda bu şekilde düşünerek problemlere çözüm ararlar. Etmen, kullanıcısı yada sahibi adına karar veren ve etki eden, otonom olma, işbirliği yapma, tepki verme, …vb özelliklere sahip olan yazılım birimidir. Hareketlilik de seçime bağlı bir etmen özelliğidir. Hareket özelliğine sahip etmenler

ağda konaklar arasında gezinebilme yeteneğine sahip olduklarından dolayı, bu özellik etmen sistemleri için önemlidir.

Etmen sistemlerinde etmen hareketliliği otonom hesaplama gücünün ağ üzerinde gezdirilebilir olmasından dolayı sistem ölçeklenebilirliği ve perfomans açısından önemlidir. Benzer şekideyine sistem performansını arttırmak için kullanılabilecek alternatif çözüm olan süreç göçlerinden farkı ise hareketli etmenlerin otonom ve tepkisel olmasıdır. Yukarıda sayılan gereksinimler göz önünde bulundurularak günümüzde bilgi sistemleri üzerinden taşınan bilgilerin boyutunun çok büyük olmasından ötürü uçlar arasında veri transferi yerine, hesaplama transferi yapmak daha akılcıdır. Böylelikle uzak makinada hesaplama yapıp sonuçları almak yerine yerel makinada hesaplamalar yapılır. Bu, ağ üzerindeki işlem sayısını da azaltmış olur. Yani verilerin uzak makinaya gönderilip hesaplama sonucunun alınmasındansa sadece hesaplama gücünün yerel makinaya göçmesi şekline dönüşür.

Bu çalışma ile çözülmek istenen problem, etmenlerin, bir makinadan başka bir makinaya standartlara uygun bir biçimde yayınla/abone ol modeli alt yapısı üzerinden göç edilebilmesini sağlayan bir tasarım kalıbı oluşturulmasıdır.

Literatürde etmen göçüne önerilen çözümlerin hepsi ya adres bilgisinin önceden verilmesi yoluyla sıkı bağımlılık gerektiren ya da standartlara uygun olmayan sistemlerdir.

Gevşek bağlı sistemler oluşturmak için birimlerin birbirlerini doğrudan tanımamaları gerekmektedir. Bunun için araya bir ara katman konulup bu ara katman aracılığı ile sistemleri haberleştirmek yerinde olacaktır. Bu çözüm yayınla/abone ol modelinde çok tanıdıktır. Bu yüzden gevşek bağlı asenkron haberleşme sağyalan yayınla/abone ol haberleşme alt yapısı çözümümüzde kullandığımız ana prensiplerden biridir.

Ağ yükünü hafifletmek için ağdan veri geçirmek yerine bu veriler üzerinde çalışacak hesaplamayı iletmek amacı ile akıllı, otonom yazılım birimleri olan hareketli etmen kavramı tercih edilmiştir.

Bu çalışmada etmenleri ağ üzerinde bir makinadan diğerine gevşek bağlı ve standartlara uygun bir biçimde taşımak amacı ile yayınla/abone ol modeli ve hareketli etmen kavramı bir araya getirilerek etmenlerin standart etmen haberleşme mesajları içerisinde göç edeceği adresi bilmeden konular üzerinden göçünü sağlayan mimari, bir tasarım kalıbı olarak verilmektedir.

Mimarinin bir tasarım kalıbı olarak sunulması tekrarlayan bu problem için genel bir çözüm oluşturmaktadır.

Bu çalışma ile sunulan bir çoklu etmen ekosistem tasarım modelidir. Böylece ekosistemde hesaplama bilgisi sahibi etmenler kendilerini üreten kaynak konaktan kendilerine ihtiyaç duyan hedef konağa standartlara uygun yollarla yayınla/abone ol düzeni ile göç edip hedef konakta görevlerini tamamlayıp sonlanırlar. Bu işlem neticesinde hedef konak kaynak konaktan hizmet almış olup kendi amacına katkıda bulunur. Bu yönüyle sunulan mimari hizmet odaklı mimarinin bir başka türünü ortaya koymaktadır.

Çalışma ile sunulan özel bir uygulama değil daha genel olarak standartlara uygun etmen göçü problemini çözen uygulama tasarım kalıbıdır.

TOPIC BASED AGENT MIGRATION SCHEME via PUBLISH/SUBSCRIBE PARADIGM

SUMMARY

Demanding to solve real world problems using computer systems requires those problem domain representations to be transferred to computers and their solutions should be handled by these systems. In this context, kind of problems to be solved via computers were questioned. First to solve engineering challenges, problems related to engineering systems needed to be transferred to computers and these problem domains needed to be described by computer systems. Engineering systems are generally systems that produce specific outputs to specific inputs which happen to be deterministic systems. These systems are defined matematically by a set of functions to produce outputs to inputs accordingly. For example, in signal processing, communication theory and control theory fields system definition is given mathematically by a function which is called transfer or system function. After determining the transfer function, system happens to be designed. Here solving problems in these domains by computers functional programming paradigm is introduced. In functional programming, a system is described by functions it has or it must fulfil. Implementing this system requires calling these functions in a main function that brings those functions together in cohorence and determines the control flow of the system. However, with the fast developments in computer technology, demand has emerged to use computers to solve many different domains of problems in which computers have poweful processingcapacity and high storage capacity.In other words, to problems belong to domains other than engineering, solutions should be developed in computer domain. For instance, solving problems defined in fields like banking, insurance, telecommunications …etc. by computers. Though producing solutions by software applications needed to be provided to problems in those domains with functional programming is not that easy, robust, flexible and maintainable. Functional programming is very insufficient to transfer those problem domains to computer world.For that purpose object oriented programming paradigm is introduced. In object technology, everything in the system is described with objects. In this kind of paradigm, systems are designed by producing all of the objects in the system (static view), assigning responsibilities to those objects and defining the interactions between objects (dynamic view). Designing high quality and robust systems by using object technology requires the usage of software design patterns and principles. However, insufficiency of designing with objects and the requirement of intelligent software entities which thinks and acts like humans provided the idea of designing systems with mental software entities. With this idea a brand new programming paradigm has emerged and called agent oriented programming.

Understanding agent concept and technology helps understand agent oriented programming better.Digesting agent concept, it develops thinking in agents capability and agent oriented design comes up. This is a very close method for

humans because humans look for solutions for problems in the same way. Agent is a software entity which has autonomy, collaboration, reactivity, …etc. porperties that acts on its user or owner's behalf. Mobility is an optional property for an agent. Mobile agents are capable of moving in the network and this agent ability is important considering many agent system applications.

In multi agent systems agent mobility is important from scalability and performance aspects due to the ability of moving autonomous computation in the network. Likewise as an alternative solution to increase system performance as a mobile agent, agent migration is not capable of autonomy and reactivity.Considering those requirements mentioned above transferring computation between nodes in the network is more sensible than transferring data to be processed between nodes due to the huge amount of data carried over information systems nowadays.Thus rather than performing computation on a remote host and communicating results back, computation on the local host is performed. This also reduces the number of transactions over the network.Instead of sending data to a remote machine and getting back the computation result, local host receives computation that migrates from remote host to work locally.

This work aims to solve the problem of agent migration from a host to another by adopting publish/subscribe paradigm conforming to industry standards. Main concern of the work is to offer a design pattern that solves the recurring problem for multi agent system application developers using mobile agents defined above.

All of the existing works in this domain contains tightly coupled solutions by providing network location of the hosts identified by address in advance or non-industry standard way which are generally very specific to problem solution.

Forming loosely coupled systems requires that entities should not know each other directly. For this reason there needs to introduce an intermediary layer in between communicating entities and make entities communicate over this middleware. This solution is very familiar in publish/subscribe model. This is the reason why we adopted publish/subscribe communication infrastructure which provides loosely coupled asynchronous messaging as one of the main principles in our solution.

Instead of transmitting data to be processed, transmitting computation is more eligible and mobile agent concept is constructed on this idea. Due to this idea behind mobile agent concept, it is very convenient for our solution architecture and is adopted.

In this work we propose an architecture as a design pattern that enables migrating agents from a host to another in a loosely coupled manner and conforming to industry standards by bringing publish/subscribe scheme and mobile agent concept together and wrapping agents into agent communication language messages as payloads without informing any entity with the address of another entity to migrate to.

Presenting the architecture as a design pattern provides a general solution to the recurring agent migration problem.

This work presents a multi agent ecosystem design model. Thus in the ecosystem information expert agents migrate from source hosts that produce them to target hosts that need their service by publish/subscribe paradigm with conformance to industry standards. After getting started at the target host they work autonomously there and terminate. After this transaction, target host is said to be served by source

host to fulfil its goals. This is an another way of service oriented architecture by using mobile agents.

This work does not present a specific application but rather presents more generally a design pattern to solve agent migration problem conforming to industry standards.

1. GİRİŞ

Zamanla yazılım teknolojilerindeki gelişmeler sistem mimarilerini kuvvetli bir şekilde etkilemiştir. Merkezi yapılar merkezi olmayan yapılara dönüşmüştür. Burada yazılım çözümleri de yerelleştirilmiş sistemlerden çok dağıtık sistemler şeklinde inşa edilir hale gelmiştir. Ayrıca dağıtık hesaplama mimarisi de istemci sunucu modelinden eşler arası modele dönüşmektedir. Bu değişimin temel nedeni ise sistem ölçeklenebilirliği ve performans gereksinimleridir. Hesaplamayı yüksek kapasiteli tek bir birimden daha düşük kapasiteli çoklu birimlere dağıtmanın çok daha verimli olduğu görülmüştür.

Programlama paradigmasi da fonksiyon merkezli modelden nesne merkezli modele geçmektedir. Günümüzde nesne yönelimli programlamanın daha da değişerek etmen tabanlı programlamaya yol açtığını görmekteyiz. Etmen tabanlı programlama çok populer bir araştırma alanı olan çoklu etmen sistemlerinin gerçeklenmesine olanak sağlamaktadır.

Yazılım etmenleri otonom, işbirlikçi ve tepkisel yazılım birimleridir. Bu yazılım birimleri kullanıcılarının veya sahiplernin adına çalışırlar. Hareketlilik ise çoklu etmen sistemlerinde yazılım etmenleri için isteğe bağlı bir özelliktir. Aslında etmen hareketliliği düşünülmesi gereken önemli bir faktördür. Çünkü otonom ve akıllı hesaplama gücünü yerel olarak çalıştırmayı gerektirecek bir çok uygulama modeli bulunmaktadır. Bunun sağlanması da akıllı hesaplamanın işlenecek yoğun veri ortamına göçmesi ile mümkün olur.

1.1 Tezin Amacı

Literatürde varolan çalışmalar etmen göçünü standart bir yol ile gerçekleştirmedikleri gibi ayrıca sistemde yüksek bağımlılık oluşturmaktadırlar. Bu çalışma bu problemleri çözmek için yapılmıştır. Yanı düşük bağımlılıkla standart bir yol ile konaklar arası etmen göçünün başarılması amaçlanmıştır.

Çalışmamızda, ayrık, kaynak konak ve hareketli etmeni hedef adresi ile bilgilendirmeden konu tabanlı etmen göçü önermekteyiz. Ayrıca platformlar arası etmen hareketliliği de standartlara uygun olarak gerçekleşmektedir. Burada çoklu etmen sistemleri için hizmet tabanlı mimarinin bir başka modeli sunulur.

Hareketli etmenler için ayrık haberleşme sağlamak için hedef konak adresi önceden bilinmeden konu tabanlı göç gerekmektedir. Bu mekanizma yayınla/abone ol modelinde çok tanıdıktır. Bu çalışmada hareketli etmen kavramı ve yayınla/abone ol modelini birbirine yakınsatarak düşük bağımlılıklı haberleşme tabanı ile ölçeklenebilir, eşler arası ve merkezi olmayan sistem üretmekteyiz.

Burada "Bilgi Uzmanı Etmenler Ekosistemi" olarak adlandırdığımız bir ekosistem önermekteyiz. Bu ekosistem, bilgi uzmanı hareketli etmenleri yayınla/abone ol haberleşme modeli üzerinden iletişim kuran kaynak ve hedef konakları içermektedir. Ekosistemde hareketli etmenler ile hizmet üreten konaklar ve hizmet tüketen konaklar bulunmaktadır. Hizmet bilgi uzmanı hareketli etmenler ile sunulmaktadır. Bilgi uzmanı etmenler mesajlaşma ara katmanının konularına yayınlanır. Tüketici konaklar da ilgilendikleri konulara abone olurlar. Üretici konaklar konuya yayın yaptıkları zamanmesajlaşma ara katmanı abone konakları hareketli etmenler ile bildirir. Bu yaklaşım sayesinde hareketli etmen kaynak konaktan, ağ üzerindeki yerini tanımlayan adresini bilmeden hedef konağa göç eder. Yayınla/abone ol paradigmasını uygulamakla sistem kendiliğinden ölçeklenebilir olur. Değişen ağ topolojisindeki çok sayıdaki hedef konak kolaylıkla bildirilir.

1.2 Literatür Araştırması

E-ticaret uygulamalarında hareketli etmenlerin önemini ortaya koyan bir çalışmayı Ametller, Robles ve Borrel yapmıştır.Çalışmalarında etmen sistemlerindeki hareketliliğin eksik olmasını veya kısıtlı ve standartlara uygun olmayan hareketlilik gerçeklemelerin bulunmasını işaret etmişlerdir.Buna çözüm olarak etmenleri FIPA (Foundation of Intelligent Physical Agents) ACL (Agent Communication Language) mesajlarının içerisinde standartlara uygun olarak gezdirmeyi önermişlerdir. Bu öneriye göre etmenler standart etmen haberleşme mesajlarının içerisinde bir konaktan diğerine göç edebilmektedirler.

Malik ve diğerleri ise etmen göçünü tarif eden bir başka çalışma yapmışlardır. Çalışmalarında platformlar arası göç edebilen etmen sistemlerini önermişlerdir. Bunu bir platform etmenini göç edeceği diğer platform etmenine uygun bir biçime çevirme yoluyla gerçekleştirdiklerini anlatırlar.

Etmen göçünün belirtimlerle sabitlenmiş standart bir modeli olmadığından dolayı araştırmacılar verimli bir etmen göçü düzenini başarmak için bu alanda çalışmaya devam etmektedirler.

Hareketli etmenler için tüm varolan çalışmalar göç sürecinde yer alacak olan konakları ve hareketli etmenler ile hedef konakları birbirine sıkı bir biçimde bağlamaktadır. Bu sıkı bağımlılık hedef adresi ile gerçeklenmektedir. Yani hedef makinanın ağ adresi göç işleminden önce bilinmek durumundadır. Eğer adres verilmez ise göç işlemi gerçeklenemez.

Literatürde yayınla/abone ol modeli ve hareketli etmenleri kullanan diğer çalışmalar da bulunmaktadır. Bunlar Paraschiv ve diğ., Shen ve diğ., Şahingöz, Erdoğan çalışmalarını içermektedir. Ancak bu çalışmalar hareketli etmenler ile yayınla /abone ol model mekaniğini ve ara katman birimi tasarımını gerçekleştirmek üzere yapılan çalışmalardır. Önerdiğimiz çalışma ise var olan bir ara katman ile yayınla/abone ol konu tabanlı haberleşme alt yapısı üzerinden hareketli etmenlerin kendilerini bir konaktan diğerine, hedef ağ adresini bilmeden yayınlama üzerinedir.

2. ETMEN SİSTEMLERİ

Ağ tabanlı bilgi kaynaklarındaki büyüme ağa dağıtılabilen bilgi sistemlerini ve onlarla birlikte çalışan diğer sistemleri gerektirmektedir. Geleneksel yazılım teknolojilerinin dağıtım ve birlikte calışma kavramları üzerindeki kısıtlamalarından dolayı bu tip sistemler kolaylıkla gerçeklenemiyor. Etmen-tabanli teknolojiler bu tip dağıtık ve birlikte çalışmayı gerektiren sistemleri gerçekleyebilecek ümit veren bir bir cevap olarak karşımıza çıkmaktadır. Bu teknoloji dağıtım ve birlikte calışma (interoperability) kavramlarının üstesinden gelmek üzere icat edilmiştir.

Yazılım etmenleri (software agent) teknolojisi; açık, dinamik, değişimlerin çok olduğu ve heterojen ortamların yapısına uygun yazılımların geliştirilmesine yönelik bir teknolojidir. Genel olarak, yazılım etmenleri söz edilen türdeki ortamlarda özerk (otonom) bir biçimde davranma özelliğine sahip birimler olarak tanımlanmaktadır.

2.1 Etmen Tanımı

Genel kavram olarak, birlikte veya ayrı ayrı etkisini gösteren ve belli bir sonuca götüren güçlerden, şartlardan, ögelerden her biri, amil, faktör.

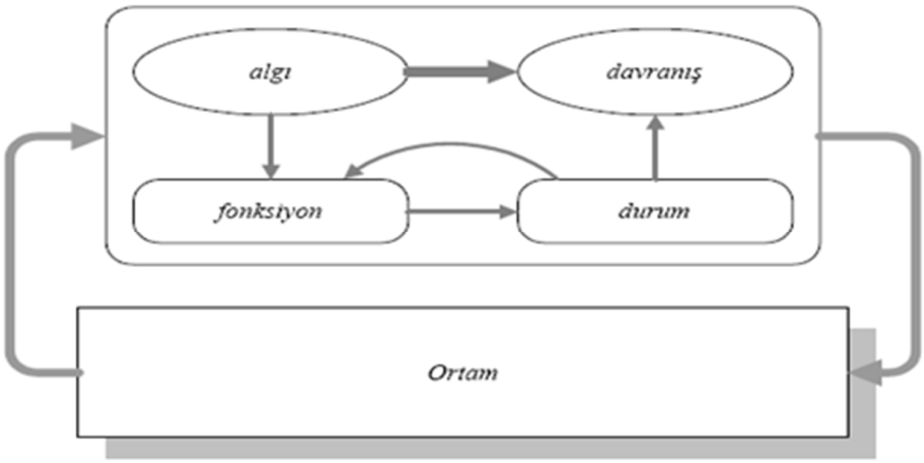
Yazılım etmeni, kullanıcı yada baska bir program adına karar veren (haraket eden) yazilim parcasidir.

“Algılayabilen, muhakeme edebilen, hareket (müdahale) edebilen ve haberleşebilen sürekli (kalıcı) hesaplamaya etmen denir” (Singh & Huhns).

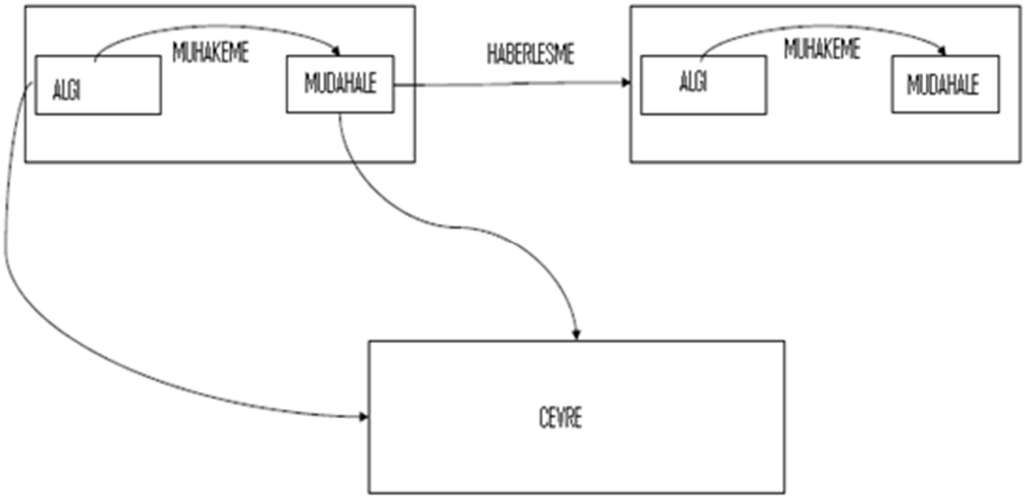
“Etmenler, belli bir ortamda bulunan ve görevlerini gerçeklemek üzere bu ortamda otonom davranış gösterebilen bilgisayar sistemleridir” (Wooldridge ve Jennings, 1995).

“Etmen, karmaşık ve dinamik bir ortamda bulunan işlemsel bir sistemdir. Bu ortamı algılayıp onun üzerinde çeşitli davranışlar sergileyebilir. Bu davranışları yoluyla başarması gereken işler vardır. Bilgi, inanç, niyet ve yükümlülük gibi kavramların yanında duygusallık da etmene yüklenebilecek meziyetler arasındadır” (Maes, 1994).

Etmenler bir birimi (kullanıcı, başka program) temsil ederler. Bu yüzden birden fazla etmen birbiriyle çelişen tercihlere, amaçlara sahip olabilir. Bu problemin üstesinden gelmek için birlikte çalışmaları gerekmektedir. Bu amaçla etmenler birbirleyle haberleşirler ve birbirlerinin ortama olan müdehalelerini koordine ederler ve amaçları üzerine müzakere ederler. Şekil 2.1 ve Şekil 2.2 de sırasıylaetmenlerin çevreleriye ve birbirleriyle olan etkileşimleri görülmektedir.



Şekil 2.1 : Etmen ve ortamla etkileşimi (Sarıel, 2002).



Şekil 2.2 : Etmenlerin birbirleriyle ve çevreleriyle olan etkileşimi.

Bu tanımlara uygun olarak sıklıkla kullanılan iki etmen sistemi örneği verebiliriz. İlk örnek sistemde etmenimiz bizim adımıza internetten bize uygun olan bir tatil paketi bulur. Fiyat, zaman, mekan ve şartlar üzerine tatil yeriyle (web uygulaması olabilir) bizim yerimize bizim isteklerimiz doğrultusunda anlaşma yapar. İkinci örnek sistemimizisetoplantı odası ayırtmak için etmen sistemi kullanımıdır.Burada

etmenimiz bizim adımıza toplantı odasını ayırtmak için diğer kullanıcı etmenleri ve toplantı odası etmeni ile anlaşmaya çalışır.Buradaki tanımlamalardan ve örneklerdende anlaşılacağı üzere etmen, nesneye yönelik programlama kavramına benzer fakat bu kavramın daha gelişmiş bir şeklini yeni bir programlama paradigması olarak ortaya koyar. Etmen, nesnenin daha gelişmiş ve farklılaşmış bir hali olarak karşımıza çıkar.

2.2 Etmen Özellikleri

Başlıca etmen özellikleri aşağıdaki gibi sıralanılır.

Süreklilik: Etmen yazılımı istek üzerine çalıştırılmaz sürekli biçimde çalışır durumdadır ve ne zaman bir etkinlik yapacağına kendi karar verir.

Otonom: Etmenler bağımsız olarak hareket etme yeteneğine sahiptirler. Eylemlerini kendileri seçerler. İletişime geçecekleri diğer etmenlere karar verirler.

Heterojen: Etmenler farklı ekipler (gruplar) tarafından tasarlanıp gerçeklenebilirler. İç özelliklerinin dışarı açılmasına gerek yoktur. Standardlara uygun olarak tasarlanmalıdırlar.

Sosyallik: Etmenler, diğer bileşenlerle haberleşir ve koordinasyon içerisindedir. Bir görevi yerine getirmek için birlikte çalışabilirler.

Reaktiflik: Etmenler içinde bulundukları ortamı algılayıp bu ortamda meydana gelen değişikliklere cevap verirler.

Proaktif olma: Ortamla etkileşimli olmanın yanında zeki bir etmen verilen görevleri de yerine getirebilmelidir. Yani hedef odaklı hareket etmelidir. Bunun için de fırsatları değerlendirmeli, hedefe ulaşmak için gereken tüm koşulları yerine getirmelidir.

Zamanda sürekli olma: Sistemde sürekli canlı bulunma özelliğidir.

Hareketlilik (Mobility): Görevini yerine getirmek üzere diğer yürütme ortamlarına hareket edebilme özelliğidir.

Rasyonellik: Etmenin kendi inançları doğrultusunda kendi amaçlarını gerçeklemek üzere hareket etmesi, amaçları ve inançlarına ters düşen durumlarla karşılaştığında davranış veya hareketlerini kısıtlaması özelliğidir.

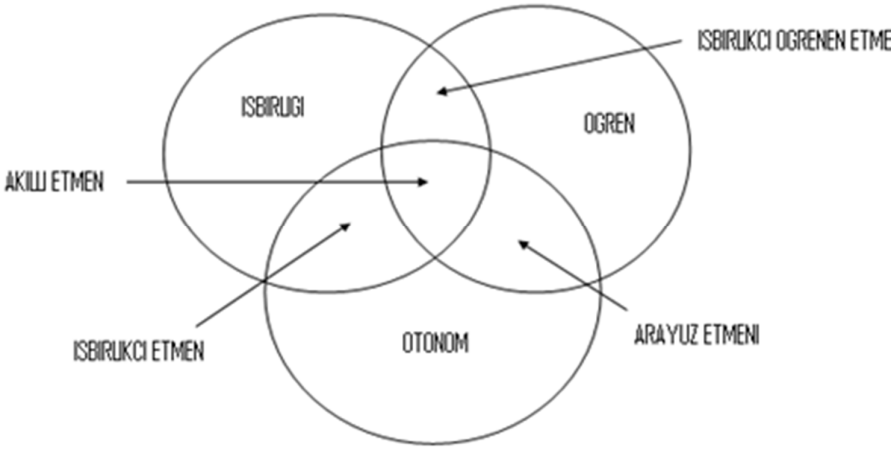
Dürüstlük(Verocity): Diğer etmenlerle iletişim kurduğunda doğru bilgiler aktarma özelliğidir.

Yardımseverlik (Benevolence): Etmenlerin birbirlerine zıt amaçları olmaması ve bir etmenin kendisinden yapılması beklenen işi yapmak üzere çalışması özelliğidir.

Adaptif olma: Etmenler neyin kendileri için iyi ve kullanışlı olduğunu algılayıp davranışlarını zamanla değiştirebilirler.

Menfaatçi (self-interested): Etmenler sahipleri için çalışırlar. Bir şey istenmesi üzerine bir görev yerine getirmezler.

Bu bilgiler doğrultusunda bazı özelliklerine göre etmenler Şekil 2.3'teki gibidir.



Şekil 2.3 : Bazı özelliklerine göre etmenler.

2.3 Etmen Ortamları

Russell ve Norvig'in yaptığı sınıflandırmaya göre etmenlerin bulunduğu ortamlar dört kategoriye ayrılır. Bunları aşağıdaki gibi sıralayabiliriz.

Erişilebilirliğine göre; bir ortamın erişilebilir olması, o ortamın tüm parametrelerinden etmenin sürekli haberdar olması demektir. Gerçek hayatta ise çoğu ortam hakkında etmenlerin kısıtlı bilgisi bulunmaktadır (örneğin fiziksel dünyada bir etmen ortamın yalnızca görüş alanında kalan kısmıyla ilgili fikre sahiptir). Etmen bir ortam hakkında ne kadar bilgi sahibiyse, yani o ortam onun için ne kadar erişilebilirse o etmeni tasarlamak o kadar kolaylaşır.

*Deterministik oluşuna gö*re; etmenin ortama yaptığı her bir etkiye belirli ve tek bir tepki alması durumunda o ortam deterministiktir, her hareketin neyle sonuçlanacağı kesindir. Deterministik olmayan ortamlarda çalışan etmenleri tasarlamak zordur.

Sabitlik veya değişkenliğine göre; etmenin hareketleri ve doğurduğu sonuçlar dışında parametreleri değişmeyen ortamlar sabittir. Fakat fiziksel ortamlar oldukça değişkendir ve etmenin kontrolü dışında çalışan pek çok süreç bulundurmaktadır.

Kesintili veya sürekli oluşuna göre; belirli sayıda adımı ve algısı olan ortamlar kesintilidir. Örneğin bir satranç tahtasında yapılabilecek hamleler belirlidir. Ancak gerçek dünyada bu tarzda yön ve hareket sınırı bulunmaksızın nesneler hareket edebilmektedir.

2.4 Etmen Soyutlaması

Etmen soyutlaması genel olarak iki başlık altında incelenir. Bunlar geleneksel ve etkileşimsel kavramlardan oluşur. Geleneksel soyutlamalar mental kavramlar iken etkileşimsel soyutlamalar etmene özgü doğuştan olan kavramlardır. Geleneksel soyutlamalar, inanç, bilgi, istek, hedef ve niyet kavramlarından oluşur. Bu mental soyutlamalar kısaca aşağıdaki gibi tarif edilir.

Inanç: Etmenin dünya temsili

Bilgi: Dogru inançlar

İstek: Dünyanin tercih edilen durumları

Hedef: Tutarlı istekler

Niyet: Müdahale için benimsenmiş hedefler

Etkileşimsel soyutlamalar ise etmene özgü olup doğuştan gelir. Bunlar etmenin sosyal, organizasyonel, etik ve resmi olmasıdır. Kısaca tanımları aşağıda verildiği gibidir.

Sosyal: Etmen toplulukları ile ilgili

Organizasyonel: Takımlar ve gruplar ile ilgili

Etik: Doğru ve yanlış davranışlar (müdahaleler)

Resmi (yasal): Sözleşme ve uyum ile ilgili

2.5 Etmen Matematik Modeli

Etmenin çevre durum kümesi S ve eylem kümesi A

$$S: \{s_0, s_1, s_2, \dots, s_n\} \quad \textbf{(2.1)}$$

$$A: \{a_0, a_1, a_2, \dots, a_n\} \quad \textbf{(2.2)}$$

olmak üzere *deteministik davranış*: Bir cok duruma gidip yapılacak eylem(ler)in belirlenmesi aşağıdaki gibi tanımlanır.

$$S^* \rightarrow A \quad \textbf{(2.3)}$$

Deterministik olmayan etki: Etmen bir duruma gidip mudahalesini yapinca bir küme duruma gecebilir.

$$S \times A \rightarrow f(S) \quad \textbf{(2.4)}$$

Geçmiş, çevre durumları serisi ve etmen müdahaleleri olarak tanımlanır.

$$h: s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \dots s_n \quad \textbf{(2.5)}$$

n sabit oldugunda etmen nihayetinde sonlanır buna kisa-omurlu hesaplama (short-lived computation) denir.

$n \rightarrow \infty$, *sonsuz döngü*: Bu duruma da uzun-omurlu hesaplama (long-lived computation) denir.

Etmen bulunduğu tüm durumları ve bunlara karşı gelen tüm müdahalelerini hatırlıyorsa buna tüm geçmiş (full-history) denir.

Etmenin algıladığı veri P ve etmen durum kümesi I olmak uzere:

Algı: $S \rightarrow P$

Davranış: $I \rightarrow A$

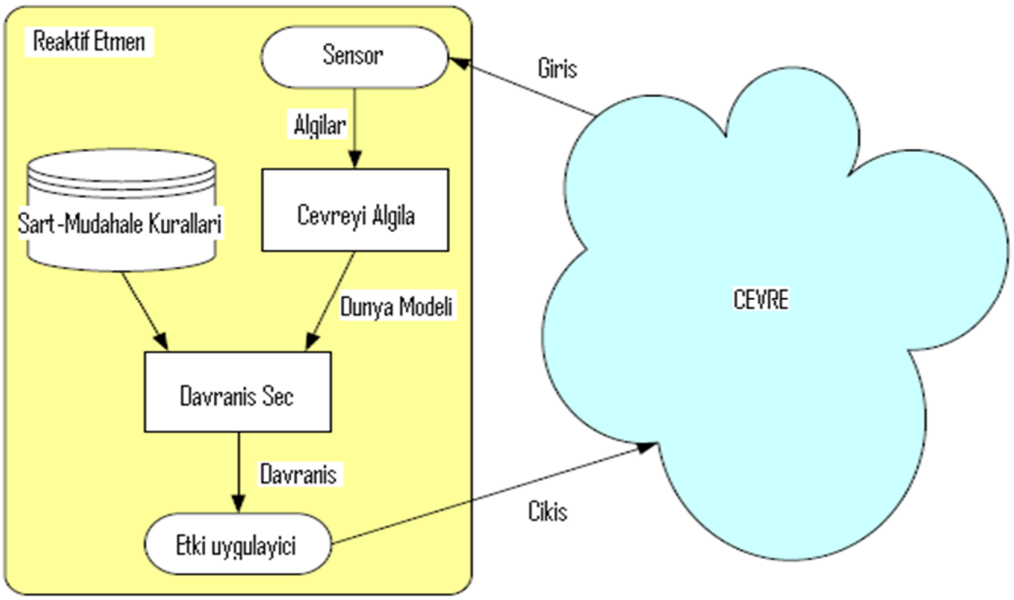
Sonraki durum: $I \times P \rightarrow I$ geçişleri şeklinde tanımlanmaktadır.

Etmen bir başlangıç durumunda çalışmasına başlar. Ortam bilgisini alır. Sonraki durum fonksiyonu, etmenin durumunu günceller. Bu duruma göre etmenin yeni algıya karşılık davranışı seçilir. Bu davranış, etmen tarafından gerçeklendikten sonra etmen bir başka çevrimde algı ile başlayan aynı işlemleri gerçekler. Durum tabanlı etmen modeli güçlü bir yapıya sahip olmasa da etmenlerin çalışma yapılarıaçısından temel bir model oluşturur (Sarıel, 2002).

2.6 Etmen Çeşitleri

Reaktif etmenler:

Durum bilgisi tutmazlar. Davranislari S → A seklindedir yani icinde bulunduklari duruma gore mudahale ederler geçmiş durumları göz önünde bulundurmazlar. Basit bir yapıları vardır. Reaktif etmene termostat örneğini verebiliriz. Termostat etmeni algılama ile sıcaklığı ölçer ve bu değeri bir aralıkla karşılaştırır, karşılaştırma sonucunu kontrol eder sonuç uygun değilse müdahale eder. Şekil 2.4'te görülür.



Şekil 2.4 : Reaktif etmen.

Hedef amaçlı etmenler:

Belirli bir hedefi vardır. Bu anlamda reaktif etmenden farkı ortamda bir değişiklik olmasını beklemeden hedefini gerçekleştirmek üzere calışır. Reaktif etmende böyle bir özellik yoktur. Değişen ortama göre tepki verir. Hedef amaçlı etmenlerde durum (şart) – müdahale kurallari yeterli değildir. Hedef durumu söz konusudur. Hedef gerçeklendiğinde etmen bu duruma geçer. Bulunduğu durumdan hedef durumuna geçmek için bir dizi davranışlar sergiler.

Fayda tabanlı etmenler:

Hedef amaçlı etmenler hedeflerini gerçekleştirmek için bir çok yoldan birini seçerler ancak bu yolu seçerken bu yolun iyi bir yol mu yoksa kötü bir yol mu olduğuna karar vermesi söz konusu değildir. Bunu gerçekleştirmek üzere fayda tabanlı etmenlerden

bahsedebiliriz. Çünkü hedefe ulaşılacak yollardan biri diğerine göre daha kısa daha ucuz…vs. olabilir. Fayda, seçilen yolun niceliksel ölçüsü ve durumlardan gerçek sayılara bir fonksiyon olarak tanımlanır. Birden fazla hedefi olan etmenlerde beklenen faydayı maksimize etmek için her hedefe bir fayda atanır. Etmenlerin çelişen hedefleri olabilir.

Bilgi toplayıcı etmenler:

İnternet gibi bir bilgi ağında kaynaklara etkin şekilde erişip kullanıcı isteklerine göre bilgi toplayan etmenlerdir. Diğer etmenlerle de haberleşerek bilgiye daha çabuk erişmek için gerekli olan bilgileri öğrenirler. Kullanıcı ile sürekli etkileşim kurarak kullanıcısının isteklerini öğrenmek durumundadırlar (Sariel, 2002).

Akıllı etmenler:

Belli bir görevi yerine getirmek üzere çalışan ve öğrenme, çıkarım yapma gibi yeteneklere sahip olan etmenlerdir.

Arayüz etmenleri:

Kullanıcı ile aynı ortamda işbirliği yapan kişisel yardımcılardır. Kullanıcının davranışlarını gözlemler, raporlar ve öğrenme ile yeni yetenekler kazanır. Kullanıcısına gerekirse daha iyi yollar önerebilir. Öğrenme süreci, kullanıcıyı gözlemleyerek veya taklit ederek (supervised öğrenme), kullanıcıdan iyi ya da kötü şekilde geri besleme alarak (destekli öğrenme), kullanıcıdan komutlar alarak veya diğer etmenlerden bilgi sorma ile yardım isteyerek yürütülebilir.

Ortak çalışan etmenler:

Kendi başlarına görevlerini yürütebilmek için özerkliğe sahip olan ve diğer etmenlerle birlikte çalışan etmenlerdir.

Hareketli etmenler:

İnternet gibi geniş-alanlı ağlarda gezebilen yazılım süreçleridir. Başka ortamlarda işlemlerini gerçekleştirdikten sonra kendi ortamlarına geri dönerek elde ettikleri bilgiyi sunarlar. Güvenlik, gizlilik, taşıma mekanizması, doğrulama gibi problemlerin üzerinde çalışılmalıdır.

Hibrit etmenler:

Birden fazla etmen tipi özelliğini bir arada sunabilen etmen türleridir.

2.7 Etmen Tabanlı Sistemler

Tekil etmen sistemi:

Bu tip sistemler merkezi olup bir adet etmene sahiptir ki bu etmen bütün kararları alır. Diğerleri uzak köleler (slave) gibi davranır. Tekil etmen sistemi çoklu varlıklara (isletici (actuator), robot) sahip olabilir ancak herbir varlık algılarını merkezi tekil sürece gonderir ve müdahalelerini yine merkezi tekil süreçten alır. Bu merkezi sürece tekil etmen denir.

Tekil etmen sistemleri çoklu etmen sistemlerine gore daha basitmiş gibi gözüksede aslında sabit, karmaşık bir görevi gerçeklerken bunun tam tersi olduğu ortaya çıkar. Kontrolü çoklu etmenler arasında dağıtmak her bir etmenin daha basit olmasını sağlar. Hiç bir etmen görevin tamamını tek başına tamamlayamaz.

Tekil etmen sistemindeki bir etmen, kendini, çevresini ve etkileşimlerini kendisi modeller. Tekil etmenlerin kendi hedefleri, müdahaleleri ve bilgileri vardır. Tekil etmen sistemlerinde buna benzer başka hiç bir varlık etmenler tarafından tanınmaz. Böylelikle ortamda başka etmenler olsa bile bunlar hedeflere sahip olarak modellenmez. Bunlar çevrenin birer parcası olarak algılanırlar. Önemli bir nokta olarak etmenler de ortamın birer parcasıdır ancak kendilerine ait hedefleri, müdahaleleri ve bilgileri vardır ve bunlar açıkça modellenir.

Çoklu etmen sistemi:

Tek bir etmenin yalnız başına kendi bilgi ve bireysel yeteneklerini kullanarak çözemediği veya etkin bir biçimde çözemeyeceğini düşündüğü problemleri birbiriyle işbirliği yaparak eşgüdümlü bir biçimde çözmek için bir araya gelen etmenlerin oluşturduğu ağ, çoklu etmen sistemi (multi-agent systems-MAS) olarak adlandırılmaktadır (Durfee ve diğ. 1989).

Çoklu etmen sistemleri kendi hedefleri ve müdahalaleri olan birçok etmeni barındırmaları farkıyla tekil etmen sistemlerinden ayrılırlar. Etmenler arasında doğrudan etkileşim olabilir (haberleşme). Bu etkileşim çevresel bir uyarı olarak

görülebilmesine de rağmen etmenler arası etkileşim çevreden farklı bir olgu olarak ele alınır.

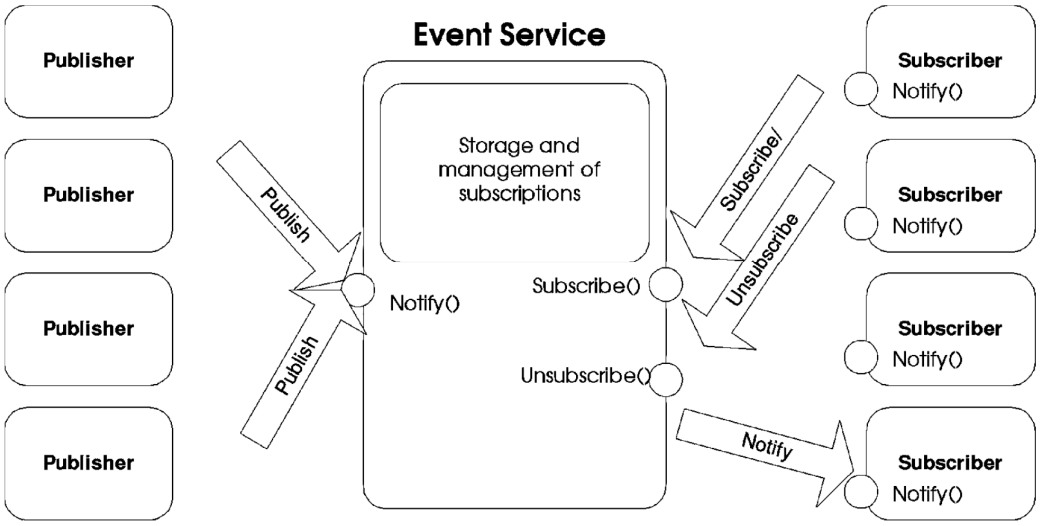
Çoklu etmen sistemleri, çevrenin dinamiklerinin diğer etmenler tarafından belirlenmesi açısından tekil etmen sistemlerinden önemli bir biçimde ayrılır. Çevrenin yapısından dolayı bulunan belirsizliklere diğer etmenlerin kasten çevreyi öngörülemeyen biçimde etkilemeleri de eklenir. Böylece tüm çoklu etmen sistemleri dinamik bir çevreye sahip olur.

3. YAYINLA/ABONE OL MODELİ

Yayınla/abone ol modeli, ölçeklenebilir ve gevşek bağlı sistemlerin kurulmasına ortam sağlayan bir haberleşme/mesajlaşma kalıbıdır. Üretici ve tüketici uygulamalarını birleştiren (entegre eden) bir etkileşim paradigmasıdır. Bu model, ölçeklenebilirlik ve dinamik ağ topolojisi desteği sunar. Sisteme bağlanan uygulamalar uçtan uca ağ topolojisinden haberdar değildir ve ayrıca sisteme yeni bir birim bağlanması veya varolan birimlerin sistemden ayrılması kendilerine tebliğ edilmez. Yayınla/abone ol paradigması mesajlaşma ara katmanını (olay hizmeti), üretici ve tüketici uygulamalarını gerektirir. Bu haberleşme sistemi tüketici uygulamalara ilgilendikeleri olay veya olayları açıklama, bunlara kayıt olma ve sonrasında üretici uygulamaların bu olayları ürettiklerinde kendilerine bildirilmesi yeteneğini kazandırır. Bir başka deyişle, üretici uygulamalar yazılım veriyoluna bilgi yazarlar/yayınlarlar ve tüketici uygulamalar da bu veriyolundan almak istedikleri bilgilere kayıt olurlar. Bu bilgi genel olarak olay ve teslim etme eylemide bilgilendirme, tebliğ etme olarak adlandırılır.

Yayınla/abone ol etkileşimi için temel sistem modeli, olay bildirim hizmetine dayanır. Bu olay bildirim hizmeti, bilgi/olay saklama, abonelik yönetimi ve etkin bir biçimde olaylarin iletimi işlevlerini yerine getirir. Böyle bir olay hizmeti, olay üreticiliği görevi gören yayınlayıcılar ile olay tüketiciliği görevi gören aboneler arasında tarafsız bir aracıyı temsil eder.Şekil 3.1'de sistem modeli görülmektedir.

Aboneler ilgi duydukları olaylara, olay hizmeti (ara katmadaki) üzerinde subscribe() işlemini çağırarak ilgi duydukları olaylara kayıt olurlar. Burada aboneler bu olayların gerçek kaynaklarını bilmezler. Bu kayıt bigisi olay hizmeti üzerinde saklı kalır ve yayınlayıcılara iletilmez. Simetrik unsubscribe() işlemi de aboneliği sonlandır.Bir olay üretmek için ise yayınlayıcı publish() işlemini çağırır. Olay hizmeti, olayı tüm ilgili abonelere çoğaltıp yayar. Böylelikle olay hizmeti abonelerin vekili gibi görülür. Dikkat edilmesi gereken husus, tüm aboneler ilgilerine uyan tüm olaylar ile bildirilirler. Ayrıca yayınlayıcılar da gelecekte yayınlayacakları olayların ilanını advertise() işlemi üzerinden verebilirler. Bu bilgi olay hizmetine kendisini yeni olay



Şekil 3.1 : Yayınla/abone ol modeli (Eugster ve diğ., 1976).

akışlarına hazırlaması için ve abonelerin de yani bilgi tipini öğrenmesi açısından faydalı olur. Olay hizmetinin sunduğu yayınlayıcılar ve aboneler arasındaki bağlaşımın koparılması üç boyuta ayrıştırılır. Bunlar uzay ayrışımı, zaman ayrışımı ve senkronizasyon ayrışımıdır.

Bilgi üretimininin ve tüketiminin ayrıştırılması, etkileşen taraflar arasındaki açık bağımlılıkları ortadan kaldırarak ölçeklenebilirliği arttırır. Aslında bu bağımlılıkların ortadan kaldırılması farklı birimler arasındaki doğrudan koordinasyonu ve böylelikle eşzamanlamayı azaltarak ortaya çıkan haberşeşme alt yapısını, doğası gereği asenkron olan dağıtık ortamlara uygun hale getirir.

3.1 Yayınlayıcı ve Abonelerin Birbirlerinden Ayrışımı

3.1.1 Uzay ayrışımı

Etkileşen taraflar birbirlerini bilmek zorunda değildir. Yayınlayıcılar, olayları olay hizmeti üzerinden yayınlarlar ve aboneler de bu olayları olay hizmeti üzerinden dolaylı olarak alırlar. Yayınlayıcılar abonelere ait referans tutmazlar ve etkileşime kaç tane aboneninkatıldığını da bilmezler. Benzer şekilde aboneler de yayınlayıcılara ait referans tutmazlar ve etkileşime kaç tane yayınlayıcının katıldığını bilmezler.

3.1.2 Zaman ayrışımı

Etkileşen taraflar, etkileşime aynı anda aktif olarak katılmak zorunda değildir. Özellikle yayınlayıcı, abone sisteme bağlı değilken bazı olayları yayınlayabilir. Diğer

taraftan abone, olayların yayınlayıcısı sisteme bağlı değilken de yayınlanmış olaylar ile bildirilebilir.

3.1.3 Senkronizasyon ayrışımı

Yayınlayıcılar, olayları üretirken bloke olmazlar ve aboneler de aynı zamanda başka faaliyeler gösterirken yayınlanan olayla asenkron olarak bildirilebilirler. Olayların üretimi ve tüketimi yayınlayıcılar ve abonelerin ana akış kontrolünde olmaz. Böylece yayınla/abone ol senktron tutumda gerçekleşmez.

3.2 Yayınla/Abone Ol Tipleri

Aboneler genel olarak özel olaylarla yada olay kalıplarıyla ilgilenirler tüm olaylarla ilgilenmezler. İlgilenilen olayların çeşitli yollarla belirtirmesi birbirinden farklı abonelik tiplerinin oluşmasına yol açmıştır. Bunlardan en popüler ve sıklıkla kullanılan iki tanesi konu tabanlı ve içerik tablanlı yayınla/abone ol modelleridir.

3.2.1 Konu tabanlı yayınla/abone ol modeli

İlk yayınla/abone ol modeli konu fikri üzerine kurulmuştur ve birçok endüstriel çözümde gerçeklenmiştir. Bu fikir, haberleşen tarafları olay içeriklerinin sınıflandırılmasına göre bir araya getiren kanal kavramının genişletilmiş halidir. Katılımcılar olay yayınlayabilirler veya anahtar kelimelerle tanımlanan özgün konulara abone olabilirler. Konular gruplama fikrine kuvvetlice benzemektedir. Bu benzerlik şaşırtıcı değildir çünkü yayınla/abone ol etkileşimini sunan ilk sistemler gruplama mantığına dayanıyordu. Sonuç olarak K konusuna abone olmak K grubununun bir üyesi olmak gibi görülebilir. K konusuna bir olay yazmakta, K grubu üyelerine yayın yapmak olarak düşünülebilir. Konular ve gruplar benzer soytlamalar olmasına rağmen farklı uygulama alanlarıyla ilişkilidirler. Gruplar yerel alan ağlarında kritik bileşenlerin kopyaları arasında kuvvetli uyumluluğu sürdürmek için kullanılırken, konular geniş ölçekli dağıtık etkileşimleri modellemek için kullanılırlar.

Pratikte konu tabanlı yayınla abone ol sistemleri özgün konuları farklı haberleşme kanallarına eşleyen bir programlama soyutlması tanıtır. Bu sistemler olay hizmeti arayüzü sunarlar ve konu ismi genel olarak başlangıç noktası olarak belirlenir. Tek

bir isimle tanımlanan her konu, yayınla() ve aboneOl() işlemlerini sunan bir arayüz ile kendi kendisinin olay hizmeti olarak görülür.

Konu soyutlaması anlaşılması kolay bir kavramdır ve olay uzayını anahtar olarak karakter dizileri ile ayırmaya dayanan birlikte çalışan platformları gerektirir. Bir çok sistem tarafından konu tabanlı plana eklemeler önerilmiştir. En yayarlı ilerleme konuların düzenlenmesinde hiyerarşilerin kullanımı olmuştur. Grup tabanlı sistemler, ki burada gruplar bağlantısız olay uzaylarını temsil eder, düz adresleme sunarken neredeyse tüm konu tabanlı sistemler hiyerarşik adresleme sunarlar. Hiyerarşik adresleme, programcılara konuları içerme ilişkisine göre organize etmeye izin verir. Hiyerarşi içindeki bir düğüme abone olma üstü kapalı bir biçimde o düğümün tüm alt konulara da üye olmayı içerir. Konu isimleri URL benzeri gösterimle temsil edilir ve hiyerarşik bir yapı tanıtır. Bir çok sistem konu isimlerinin özel semboller içermesine izin verir çünkü bu verilen ahahtar kelime kümesine uyan, hiyerarşide özel bir seviye veya tüm altağacı kapsayacak şekilde bir çok konu ismine birden yayınlama yada abone olma imkânı sağlar.

3.2.2 İçerik tabanlı yayınla/abone ol modeli

Hiyerarşik adresleme özelliği ve özel sembol kullanımı gibi geliştirmelere karşın konu tabanlı yayınla/abone ol modeli sınırlı açıklayıcılığı olan durağan bir planı temsil eder. İçerik (özellik) tabanlı yayınla/abone ol modeli, dikkate alınan olayların içeriğine dayalı abone olma planını tanıtarak konuları geliştirir. Başka bir deyişle, olaylar önceden belirlenmiş bir dış kritere (konu adı) göre değil olayların kendi özelliklerine göre sınıflandırılırlar. Bu özellikler olayları taşıyan veri yapılarının iç nitelikleri olabilir.

Tüketiciler, abonelik dili kullanarak belirledikleri filtreler ile seçmeli olaylara abone olurlar. Bu filtreler özelliklerin isim-değer çiftleri ve karşılaştırma operatörleri ile kısıtlamalar tanımlar. Böylece geçerli olayları belirlenir. Kompleks abonelik kalıplarını oluşturabilmek için kısıtlamalar mantıksal olarak birleştirilebilir (mantıksal ve, mantksal veya). Abonelik kalıpları verilen bir abonenin ilgilendiği olayarı belirlemek için kullanılır ve buna bağlı olarak olaylar yayılır. Abone olmak için olay hizmeti tarafından abonelik kalıbını temsil eden ek bir argüman ile aboneOl() işleminin bir değişik türü sunulur. Bu kalıpları temsil etmenin çeşitli yolları vardır:

Karakter dizisi: Abonelik kalıpları genel olarak karakter dizileri ile ifade edilirler. Filtreler, SQL, XPath…vs gibi abonelik dilbilgisine benzemek zorundadır. Daha sonra karakter dizileri bir motor taradından çözümlenir.

Şablon nesne: Abone olurken bir katılımcı bir ş nesnesi sunar. Böylece katılımcı tipi ş ile uyumlu olan ve özellikleri ş'nin özellikleri ile eşleşen tüm olaylarla ilgilenir.

Çalıştırılabilir kod: Aboneler çalışma zamanında olayları filtreleyebilmek için önceden yüklenmiş nesneler sunarlar. Bu nesnenin gerçeklemesi genel olarak ugulama geliştiriciye bırakılmıştır. Çalıştırılabilir kod pratikte pek yaygın kullanılmamaktadır çünkü sonuç olarak elde edilen filtreler çok zor optimize edilir.

Yayınla/abone ol sistemlerini tasarlamada çeşitli varyasyonlar vardır. Bunlar farklı derecelerde açıklayıcılık ve performans yükü sunarlar. Konu tabanlı yayınla/abone ol oldukça durağan ve ilkel olmasına karşın çok verimli bir şekilde gerçeklenir. Öte yandan içerik tabanlı yayınla/abone ol oldukça açıklayıcıdır ancak çalışma zamanında bi hayli yük getiren karmaşık protokolleri gerektirir. Bundan dolayı ana özellik olası ayrık değerlerin sınırlı bir kümesi olduğunda statik plan tercih edilir.

.

4. HAREKETLİ ETMEN KAVRAMI

Hareketli etmen, ağ üzerinde hareket edebilen akıllı hesaplamadır. Başka bir deyişle hareketli etmen, kullanıcısının veyabir birimin adına bir makinadan başka bir makinaya göç eder. Otonomluk, sosyal yetenek, öğrenme, reaktif olma ve hareketli olma özelliklerine sahiptir. Hareketli etmen, kodunu ve verisini (durumunu) hedef makinaya taşır ve burada kaldığı durumdan çalışmaya devam eder. Hareketli etmen kavramı, otonomluk ve reaktiflik özellikleriyle süreç göçü sistemlerinden farklılık gösterir. Hareketli etmenler farklı konaklarda konağın durumuna (verisine) göre farklı davranışlar sergilerler.

Hareketli etmen kavramı, istemci/sunumcu paradigmasını yerdeğiştirebilir hesaplamaya dönüştürür. Böylece, özellikle işlenecek çok fazla veri varsa ağ trafiğini etkin bir biçimde düşürür.Bu kavram, merkezi sistemler yerine dağıtık, noktadan noktaya etkileşen sistemler sunar. Farklı konaklarda asenktron yürütme elde edilir. Hedef konak, göç eden etmen kendi görevlerini yürütürken, başka görevler yürütebilir. Üstelik hareketli etmenler yürütme ortamına dinamik bir biçimde adapte olurlar. Özellikle gerekmedikçe farklı konaklarda aynı adımları atmazlar. Hedef konağın durumuna göre gerekli adımları atarlar. Hareketli etmen kavramı ayrıca konaklar arası bağlantıya dayalı olan dinamik ağ topolojisi üzerinde belli bir seviyeye kadar hata hoşgörüsü sunar. Şöyleki etmenin göç edeceği konak eğer o anda ağdan düşmüş ise etmen o konak ağa tekrar bağlandığında göç eder. Böylece konak bir süre ağdan düşmesine rağmen akıllı hesaplamayı kaybetmemiş olur. Buna ek olarak, etmenin kaynağını değiştirmek, hareketli hesaplamada değişiklik yapmak için yeterli olur. Hedef makinalarda herhangi bir konfigürasyon değişikliği yapmaya gerek yoktur. Bu güncelleme işlemlerine esneklik kazandırır. Uzaktaki makine ile haberleşip buna göre yürütmenin gerçekleşmesi ve tekrar sonuçların uzaktaki makinaya gönderilip değerlendirilmesindense yerel makine üzerinde gerekli verilerle yürütme, gerçek zamanlı işleme kabiliyetlerini arttırır.

Hareketli etmenler ve çoklu etmen sistemleri doğaları gereği dağıtık ve noktadan noktaya sistemleri tasarlamaya çok uygundur. Bu çalışmada da hareketli etmenler, bu

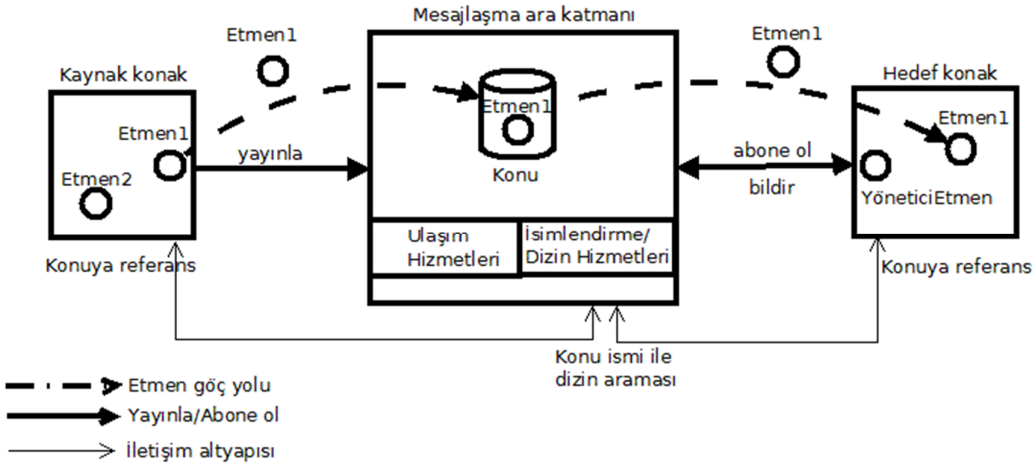
özelliklerinden ve hareketli otonom hesaplamaya olan ihtiyaçtan tercih edilmiştir. Özellikle e-ticaret uygulamalarına baktığımızda işlenecek verinin çok fazla olmasından dolayı ve ağ yükü, gerçek zamanlı işleme gibi verimlilik konuları göz önünde bulundurulduğunda yerel hesaplama gerekmektedir. Otonomluk, reaktiflik ve hareketlilik özellikleri fark edildiğinde hareketli etmenler bu tip problemlere en iyi çözümü sunacağı görülür.

Özelliklerine baktığımızda yayınla/abone ol pardigmasi ve hareketli etmen kavramı çok uyumludur. Her ikisi de üretkenliği, ölçeklenebilirliği ve dinamik ağ topolojisine sundukları esneklik ile dağıtık ve noktadan noktaya sistemlere çok iyi uyar. Bu çalışmada otonom, reaktif ve hareketli yazılım birimleri ile ayrıştırılmış, ölçeklenebilir ve esnek sistemler üretmek için yayınla/abone ol paradigması ve hareketli etmen kavramı bir araya getirilip yeni bir iletişim alt yapısı önerilir.

5. YAYINLA ABONE OL MODELİ İLE KONU TABANLI ETMEN GÖÇÜ

5.1 Sistem Mimarisi

Gerçeklenen sistem, etmenlerin konaklar arasında yayınla/abone ol iletişim mekanizması ile göç etmesini sağlar. Özel olarak etmenler kaynak konaktan hedef konağa ACL (Etmen Haberleşme Dili) mesajları biçimindetaşınırlar. Kullanılan haberleşme paradigmasından dolayı kaynak ve hedef konaklar tamamen birbirinden ayrışmıştır. Konu tabanlı yayınla abone ol düzeni kullanıldığından dolayı hedef ve kaynak konaklar önceden birbirlerini tanımak durumunda değildirler. Konakların sadece aralarındaki aracıyı bilmeleri gerekli ve yeterlidir. Yayınla/Abone Ol tasarım kalıbı sisteme uzay, zaman ve eşzamanlama bakımlarından bağımsızlık getirir. Ayrıca sistemölçeklenebilirliği artar ve dinamik ağ topolojisine karşı esneklik kazanır. Yani sisteme düzensiz bir biçimde uç nokta eklenip çıkması sistemin çalışmasını etkilemez. Konaklar arasındaki işlemleri istek cevap çiftleri halinde göndermek yerine etmenlerin kendilerini (hesaplamayı) konaklar arasında göndermek ağ trafiğini azaltırken sistem performansını da arttırır. Böylelikle yeni bir hareketli etmen kavramı ortaya çıkar. Sistem mimarisi Şekil 5.1 de görüldüğü gibidir.

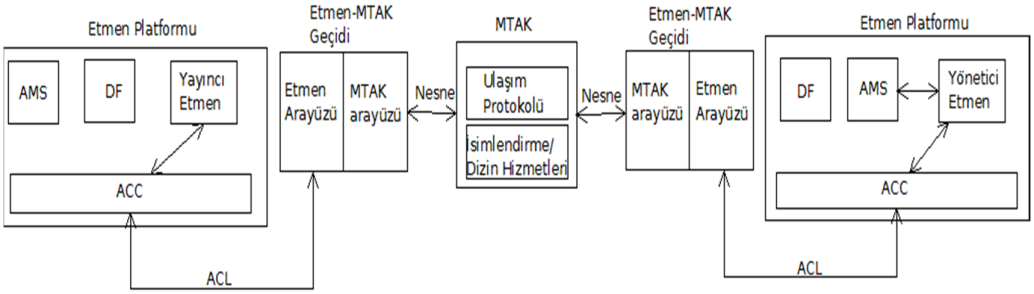


Şekil 5.1 : Ana sistem mimarisi.

Kaynak konak hizmet üretici olarak davranır ve belirgin hizmetler verecek olan etmenleri üretme yeteneğindedir. Hizmet sunan etmen kaynak konak üzerinde çalışmaya başladıktan sonra kendini mesajlaşma ara katmanı üzerinde önceden tanımlanmış olan konuya yazar. Hedef konak hizmet tüketici olarak davranır ve kendi amaçlarını yerine getirmek üzere hizmet sunan etmenin yardımına ihtiyaç duyar. Kaynak konağın mesajlaşma ara katmanı tarafından, hizmet sunan etmenler ile bidirilebilmesi için mesajlaşma katmanında tanımlı konu veya konulara abone olması gerekir.Abone olma ve bildirimleri işleyebilme görevleri hedef konak adına yerel bir etmene atanır. Bu etmene sistemde YöneticiEtmen denir.YöneticiEtmen'nin konağının niyetlerinden haberdar olması beklenir ki konağın ilgi duyduğu konulara abone olabilsin. Abone olunan konulardan birine hizmet sunan etmen yayınlandığında, mesajlaşma ara katmanı YöneticiEtmen'i ACL mesajınının gövdesine özel bir biçimde taşınan hizmet sunan etmen ile bildirir. YöneticiEtmen hizmet sunan etmeni ACL mesajından çıkarır ve bu etmeni yerel konakta etkin hale getirecek işlemleri uygular. Bu noktadan sonra hizmet sunan etmen yeni ortamın (konağın) şartlarını ve durumunu değerlendirir, bu yeni yerel konağa özgün hizmet sağlar ve işi bitince sonlanır. Böylece etmen bir konaktan diğerine özel bir mesaj biçiminde hedef konağın kimliğini (ağ adresini) önceden bilmeksizin seyahat eder. Kaynak ve hedef konaklar bir aracı yardımı ile seçilen konular üzerinden birbirine bağlanır. Başka bir deyişle aracı üzerindeki bir konu hizmet üreten ve hizmet tüketen konakların buluşma noktasıdır. Böylece kaynak ve hedef konaklar birbirinden ayrıştırılmış olur. Konaklar mesajlaşma ara katmanı üzerinde konuları anahtar olarak kullanarak dizin araması gercekleştirirler. Burada hizmet üreten etmen yayınla işleminden önce YöneticiEtmen ise abone olma işlemi sırasında dizin aramasını yapar. Mesajlaşma ara katmanı konaklara ilgilendikleri konular ile ilgili referans geri döner. Böylece konaklar arasındaki buluşma noktası inşa edilmiş olur. Şekil 5.2'de sistemin detaylı yapısı görülmektedir.

Bu mimari ağ üzerinde farklı makinalar arasında transfer olabilecek kodu destekleyen programlama dilleri ile daha kolay gerçeklenir. Sunulan mimari aynı tipteki farklı etmen platformları arasında konu tabanlı etmen göçünü doğrudan başarır. Mesela iki farklı JADE (Java Agent Development Environment) platformu arasında etmen göçü sunulan mimari ile direk olarak gerçekleşir. Aynı dili destekleyen iki farklı etmen platformu arasında ise platformların sunduğu uygulama

programlama arayüzü değişikliklerinin uygulama tasarımında yönetilmesi ile ufak düzenlemeler ile kolayca etmen göçü sağlanır.

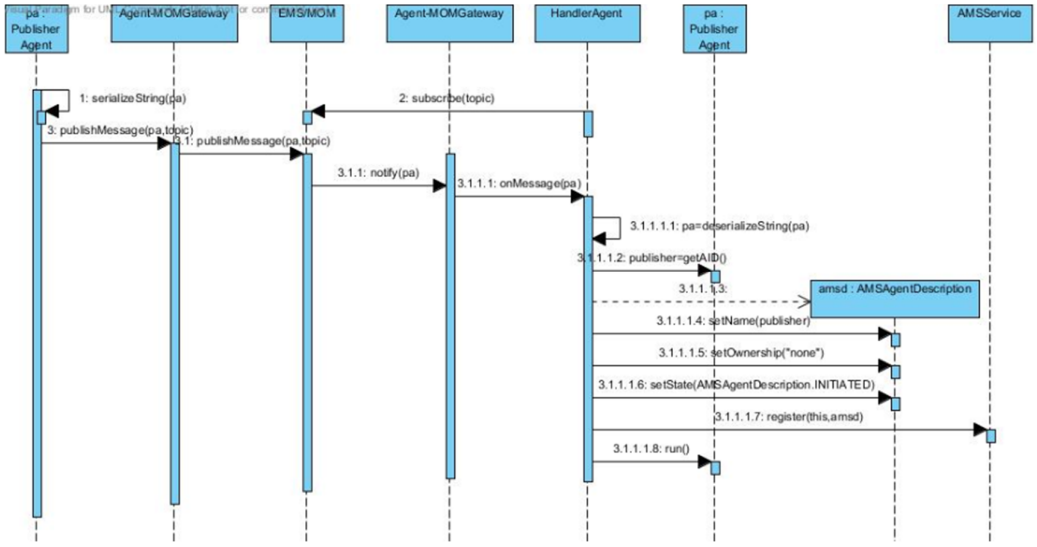


Şekil 5.2 : Sistem detayları.

Mesela JADE – JAM (Java Agency Model) gibi iki farklı platform arasında etmen göçünü sağlamak üzere sadece platformların sunduğu etmen yönetimi arayüzlerine uygun uygulama tasarımı yapılması yeterlidir. Burada JADE ile JAM java tabanlı iki farklı etmen platformudur.

Hizmet sunan etmen yayıncı etmen olarak adlandırılır çünkü bu isim etmenin rolünü açıkça belirtir. Standardları destekleyerek platformlar arası etmen göçünü veya hatta etmen haberleşmesini sağlamak için FIPA (Foundation of Intelligent Physical Agents) ACC (Agent Communication Channel) kullanılmak durumundadır. FIPA etmen platformundan ya da platformdaki bir konaktan dışarı çıkacak herhangi bir mesajın standardlara uygun olması için etmen haberleşme kanalını kullanması gerektiğini belirtir. Ayrıca platformar arası veya hatta konaklar arası iletişim için bile standart yöntemin ACL (etmen haberleşme dili) mesajlarının kullanılması ön görülmektedir. Bu noktaları göz önünde bulundurarak sunulan mimari yayıncı etmenleri mesaj gövdelerinde barındırılacak şekilde ACL mesajlarına sarmalayarak ve bu etmenleri bu şekilde platformdan ve/veya konaktan etmen haberleşme kanalı (ACC) üzerinden FIPA belirtimlerinde tarif edildiği biçimde dışarı çıkarır. Öte yandan bilgisayar sistemlerini standart bir biçimde ağ üzerinde gevşek bağlı şekilde kurumsal çapta standartlarda haberleştirmek üzere kurumsal mesajlaşma sistemleri (EMS – Enterprise Messaging Systems) kullanılır. Bu sadece etmen sistemleri için değil eskiden beri gelen yazılımsistemleri için de geçerlidir. Böylelikle, mimari etmen göçünü olağanca standart bir şekilde gerçekleştirmek üzere tüm bu endüstri standartlarını birleştirir. Burada Etmen-MTAK (Mesaj Tabanlı Ara Katman) geçiti etmen sistemlerini kurumsal sistemlere entegre etmek üzere göz önünde bulundurulmuştur. Bu geçit basitçe etmenleri ve/veya etmenlerin içinde bulundukları

ACL mesajlarını kurumsal mesajlaşma sistemleri/mesaj tabanlı ara katman (EMS/MOM) sistemleri tarafından anlaşılır nesnelere çevirir. Geçit EMS/MOM ile TCP (transmission control protocol) gibi bir standart bir protokol veya dil özel RMI (remote method invocation) gibi bir protocol ile haberleşir. Bu şekilde ACL mesajına sarmalanan etmenin EMS/MOM sistemine yayınlanması gerçekleşir. Mesaj tabanlı ara katman abone olan birimi konusunda bulunan nesneyi Mesaj Tabanlı Ara Katman-Etmen geçitine TCP, RMI …vs haberleşme protokolu ile göndererek bildirir.Geçit, nesneyi ACL mesajına çevirir ve etmen platformundaki YöneticiEtmen ile etmen haberleşme kanalı üzerinden irtibata geçer. YöneticiEtmen yerel platformun etmen yönetim sistemi ile (AMS – Agent Management System) haberleşerek gelen etmeni yerel konakta başlatır. Bu sorumluluk etmen yönetim sistemine (AMS) de verilebilir ancak bu şekilde mimari platform bağımlı hale gelir. Bu sorumluluğu uygulama tasarımına bırakmak sistem tasarım esnekliğini ve platform tipinden bağımsızlaşmayı arttırır.Sistemin çalışma zamanındaki görüntüsü Şekil 5.3'teki sekans diagramında gösterildiği gibidir.



Şekil 5.3 : Bileşenler arası mesaj akışı.

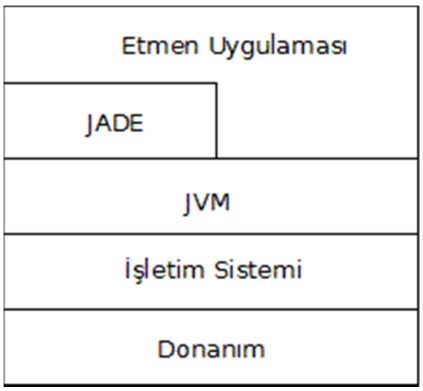
Etmen kavramı gereği etmen görevleri için davranış yapıları oluşturulur. Bu yüzden yayıncı etmenin yayınla abone ol modelindeki yayınla ilkeli ile başa çıkabilmesi için bir davranış tasarlanması gerekir.Bu davranışa YayınlaKomutu denir. YayınlaKomutu davranışı etmenin davranış listesine eklenmelidir. Bu davranışın etmene eklenmesi ile etmen kendini serilize etme ve etmen-ara katman geçidi üzerinden mesaj tabanlı ara katmanı konusuna yayınlama yeteneği

kazanır.YayıncıEtmen (YE) yayınla sorumluluğunu geçide delege eder ki geçit mesajın mesaj tabanlı ara katman tarafından anlaşılır olabilmesi için gerekli protokol çevirmelerini yapsın. Aynı sebepten dolayı YöneticiEtmen de mesaj tabanlı ara katman konularına yine etmen-ara katman geçidi üzerinden abone olur. Mesaj tabanlı ara katman abone olan YöneticiEtmen'i mesaj içine sarılmış YayıncıEtmen ile geçit üzerinden bildirir. YöneticiEtmen gelen etmeni tekrar oluşturmak için deserilize eder ve yerel konakta gerek beyaz sayfa hizmetleri için gerek duyulan verileri etmenden toplar.Beyaz sayfa hizmetleri platformun etmenler için sunduğu etmen yönetim hizmetleridir. Bu bağlamda lazım olan veri de etmen kimliği bilgisidir. YöneticiEtmen yayınlanan etmeni yerel platformda Etmen Yönetim Hizmetine (AMS – Agent Management Service) kayıt edebilmesi için yayıncı etmen ile ilgili tanmlayıcı oluşturur. Etmenlerin platformlarda faaliyet gösterebilmeleri için platformlara kayıtlı olmaları gerekir. YöneticiEtmen göç ederek gelen etmenin kimlik bilgisiyle beraber sahip ve durum bilgilerinden oluşan tanımlayıcısını oluşturur. Bu tanımlayıcıya etmen yönetim servis tanımlayıcısı (amsd - AMS Descriptor) denir. YöneticiEtmen bu tanımlayıcı ile gelen etmeni etmen yönetim hizmetine kayıt eder. Yayıncı etmen yerel platforma kayıt edildikten sonra YöneticiEtmen yayıncı etmeni başlatır. Yayınlanan etmen bu andan itibaren çalışmaya başlar.Yayıncı ve abone etmenler bir etkileşim protokolu olan ve standart etmen haberleşmesi olan FIPA abone protokolü üzerinden haberleşirler. Ancak verilen mimari gereği bu etmenler birbirleri ile direk etkileşime geçemezler. Bu yüzden bu protokolun verilen mimari ile gerçeklenmesi mesaj tabanlı ara katman bileşenleri ve yukarıda anlatılan tüm çevirimleri içerir. Bahsedilen düzen sadece platformlar arası konu tabanlı etmen göçünü başarmakla yetinmeyip ayrıca etmen-kurumsal yazılım haberleşmesini de konu tabanlı olarak standart yolla gerçekleştirir.

5.2 Gerçekleme

Bu çalışmada bir etmenin bir konaktan başka bir konağa, anlatılan mimariye göreve yayınla/abone ol paradigmasi kullanarak nasıl göç edebileceğini görmek üzere bir yazılım sistemi geliştirildi. Hareketli kod üzerindeki avantajlarından dolayı gerçekleme dili olarak java programlama dili tercih edildi. Etmen platformu olarak JADE (Java Agent Development framework) tercih edildi çünkü bu platform esneklik ve yeteneklerinden dolayı son zamanlarda etmenlerle uğraşan

araştırmacıların ilgisini çekmektedir ve ayrıca java tabanlı bir uygulama iskeleti olması da seçilmesinde önemli rol oynar. Şekil 5.4 katmanlı yapıyı gösterir.



Şekil 5.4 : Katmanlı yapı.

Bu yapı gereğince etmen uygulaması etmen yönetim hizmetlerini JADE platformundan alırken tüm başka işlemleri için standart ve/veya özel java kütüphanelerinden yararlanır. Bir mesajlaşma sistemi ile gevşek bağlı haberleşme gerçekleştirebilmek için JMS (Java Messaging Service) java mesajlaşma hizmeti tercih edildi. Esasen JMS bu mesajlaşmanın nasıl yapılacağını tarif eden bir belirtimdir ve doyısıyla bu belirtimin gerçeklenmesine de ihtiyaç duyulur. JMS gerçeklemesi olarak ta Open JMS tercih edildi. Bir başka unsur da JADE ve JMS'i entegre edecek bir geçit uygulamasıdır. Bu kapsamda Currie ve diğerlerinin JMS'i JADE platformuna yeni bir MTP (Message Transport Protocol) mesajlaşma iletişim protokolu olarak entegre ettikleri geçit kullanıldı. Bu geçit JMS kuyruk yapısı için tam bir gerçekleme sunarken yayınla/abone ol modeli için kullanılır durumda değildi. Bu çalışma kapsamında geçit üzerinde yayınla ve abone ol ilkellerinin de gerçeklemesi yapılmıştır. JADE-JMS geçidi Open JMS'i JMS sunucusu olarak desteklemektedir. Bu çalışmada yayınla/abone ol paradigması kullanılarak konu tabanlı etmen göçü düzeni önerilmektedir. Böylelikle etmen gerçekleme ve hizmetleri için JADE platformu, mesajlaşma protokolü olarak JMS, JADE–JMS entegrasyonu için JADE-JMS geçidini içeren çözüm önerilmektedir. Bu çözüm, JADE etmenlerini JMS yayınla/abone ol modeli ile aynı JADE platformundaki konaklar arasında ve farklı JADE platformlarında bulunan konaklar arasında göç ettirmeyi kapsar. Gerçeklemede JADE etmenleri FIPA ACL mesajlarına mesaj gövdeleri olarak sarmalanır.Etmen kendini serilize eder ve kendini serilize mesaj gövdesi olarak ACL mesajına ekler. Etmen taşıyan ACL mesajı Open JMS üzerinde

önceden oluşturulmuş olan konulara JADE-JMS geçiti vasıtası ile yayınlanır. Open JMS abone hedef kaynakları etmen taşıyan ACL mesajı ile bildirir. JMS sunucu konularına abonelik işlemlerini ve bildirim yönetimleri hedef konak üzerindeki YöneticiEtmen bileşeni tarafından düzenlenir.YöneticiEtmen tasarımı JMS sunucusu üzerindeki konulara JADE-JMS geçidi üzerinden abone olma işlemlerini ve göç eden etmenlerin ACL mesajlarından ayıklanıp eğer platformlar arası göç eden bir etmen ise AMS (Agent Management Service) etmen yönetim hizmetine kayıt ettirilmesini ve göç eden etmenlerin çalıştırılması görevlerini içerir. Göç eden etmen başlatıldıktan sonra otonom olarak çalışır ve işi bittiği zaman hedef konakta sonlanır.

Gerçekleme detayları ise şöyledir:

JMS belirtimi ile tarif edilen standart mesajlaşma ortamını etmenler tarafından anlaşılır hale getirmek üzere bir ontoloji tasarlanması gerekir. Çalışmamızda bunu gerçeklemek üzere "`JmsAgentGatewayOntology`" tasarlanmıştır. Bu ontoloji platform olarak kullanılan standart JADE ontolojisi genişletilerek elde edilmiştir ve sınıf tanımlaması;

"public class JmsAgentGatewayOntology extends jade.content.onto.Ontology"

şeklinde verilir. Ontolojiyi oluşturan sözlük ise aşağıdaki gibidir:

ONMESSAGE_MESSAGE = "message"

ONMESSAGE = "OnMessage"

PUBLISHMESSAGE_PROVIDERINFO = "providerInfo"

PUBLISHMESSAGE_MESSAGE = "message"

PUBLISHMESSAGE = "PublishMessage"

UNSUBSCRIBE = "Unsubscribe"

SUBSCRIBE_SUBSCRIPTION = "subscription"

SUBSCRIBE_PROVIDERINFO = "providerInfo"

SUBSCRIBE = "Subscribe"

PROPERTY_PROPVALUE = "propValue"

PROPERTY_PROPNAME = "propName"

PROPERTY = "Property"

SUBSCRIPTION_DURABLE = "durable"

SUBSCRIPTION_DURABLEIDENT = "durableIdent"

SUBSCRIPTION_SELECTOR = "selector"

SUBSCRIPTION_DESTINATION = "destination"

SUBSCRIPTION = "Subscription"

PROVIDERINFO_DESTTYPE = "destType"

PROVIDERINFO_USERNAME = "username"

PROVIDERINFO_PROVIDERICF = "providerICF"

PROVIDERINFO_PROVIDERURL = "providerURL"

PROVIDERINFO_PASSWORD = "password"

PROVIDERINFO = "ProviderInfo"

JMSMESSAGE_PAYLOAD = "payload"

JMSMESSAGE_PRIORITY = "priority"

JMSMESSAGE_PROPERTIES = "properties"

JMSMESSAGE_DELIVERYMODE = "deliveryMode"

JMSMESSAGE_TIMETOLIVE = "timeToLive"

JMSMESSAGE_DESTINATION = "destination"

JMSMESSAGE = "JmsMessage"

Bu sözlükte JMS mekaniğini oluşturan parçalar (yayıncı, abone, mesaj… vs.) detaylı bir biçimde yer alır. Bu sayede bu ontolojiyi kullanan etmenler sözlüğün içerdiği kavramları karşılıklı olarak anlayabilirler.Etmenler kullanacakları bu ontolojiye referans tutarlar:

private Ontology ontology = JmsAgentGatewayOntology.getInstance();

Bu referans ile tutalan ontolojiyi kullanabilmek için etmenlerin ontolojiye kayıt olmaları gerekir. Kayıt olma işlemi etmenin başlangıç zamanında, ilgili davranış eklenmeden önce gerçekleşmelidir. Çünkü etmenlerden yapılması istenen görevler davranışlar olarak ifade edilip etmen başlangıcında etmene eklenir. Bunu ifade eden en basit etmen başlangıcı ise:

```
protected void setup() {
   ...
   // Ontolojinin etmene eklenmesi
getContentManager().registerOntology(ontology);
   lookupAgent();
   // Yayınla etmen davranışı eklenmesi
   addBehaviour(new PublishCommand(this));
   ...
 }
```

Şeklinde verilir. Burada etmen başlangıç kodunda etmen yayınla abone ol ontolojisini öğrenmiş, lookupAgent() yordamı ile JADE-JMS geçidini tanımış ve PublishCommand() yordamı ile de kendini yayınlama yeteneği kazanmış olur. Önce yayıncı etmen nasıl geçit etmeni buluyor onu inceleyelim sonra da kendisini nasıl yayınlıyor ona göz atalım.

Geçit etmen, yayıncı etmen ile JMS belirtimini sağlayan ara birimi arasında vekil görevi görmektedir ve sistemimizde "JmsProxyAgent"olarak adlandırılmaktadır. JmsProxyAgent, yayıncı etmen ile aynı platformda ayağa kaldırılmalırdır ki yayıncı etmen ile JMS ara birimi arasında vekil görevini sağlayabilsin (benzer şekilde abone olunan platformda da bir vekil etmen ayağa kaldırılmalıdır). Vekil etmen platformda ayağa kalkar kalmakmaz kendisini "JmsProxyAgent" tipinde bir hizmet veren etmen olarak bulunduğu platformun dizin hizmetlerine (Directory Facilitator), ki burası platform için sarı sayfalar hizmeti sağlar, kayıt eder. Bu hizmeti almak isteyen etmen ise platformundaki dizin hizmetleri sağlayıcısına hizmet ismi ile istekte bulunur. lookupAgent() yordamı ile alınan bu hizmeti basit bir şekilde aşağıdaki gibi gerçekleyebiliriz.

```
 private void lookupAgent() {
   // Vekil hizmeti veren etmen platformda kayıtlı ise alınır
   ServiceDescription sd = new ServiceDescription();
```

```
    sd.setType("JmsProxyAgent");
    DFAgentDescription dfd = new DFAgentDescription();
    dfd.addServices(sd);
    try {
      DFAgentDescription[] dfds = DFService.search(this, dfd);
      if (dfds.length > 0) {
        jmsProxyAgent = dfds[0].getName();
      } else {
        System.out.println("Couldn't localize server!");
      }
    } catch (Exception e) {
      e.printStackTrace();
      System.out.println("Failed searching in the DF!");
    }
  }
```

Burada yayıncı etmen platforma kayıtlı vekil etmeni bulmak için dizin hizmetçisine istekte bulunmaktadır. Şimdi de "addBehaviour(new PublishCommand(this));" komutu ile etmene eklenen yayınlama davranışını inceleyelim. Yayınlama davranışı etmenin kendini serilize etmesi ve yayınlaması adımlarını içerir:

```
  class PublishCommand extends OneShotBehaviour {
    Agent client;
    PublishCommand(Agent a) {
      super(a);
      client = a;
    }
    /**
     *  Davranışın yapacağı işlem
```

```
 */
  public void action() {
     ...
        try{
     // Davranış ile etmen önce kendini serilize eder
        String payload = serializeString(client);
     // Sonra da yayınlar
        publishMessage(payload);
        }catch (IOException ex){
                ex.printStackTrace();
        }
     ...
  }
}
```

Burada serilize etme işlemi anlaşır ve apaçıktır ama yayınlama işlemi karmaşıktır bu yüzden bu sorumluluk başka bir yordama verilmiştir. Bu yeni yordam ile yayıncı etmen hangi JMS sunucusuna ve seçilen bu sunucu üzerindeki hangi konuya yayın yapacağını belirleyip mesajı gönderir:

```
 private void publishMessage(String payload) {
  PublishMessage pm = new PublishMessage();
  try {
    // JMS sunucusu üzerindeki hangi konuya yayın yapılacağı bilgisi
pm.setMessage(Util.createMessage("topic1", payload, Util.PERSISTENT));
  } catch (Exception e) {
   System.out.println("Error creating message:" + e.toString());
  }
  // JMS sunucusuna bağlantı
```

```
pm.setProviderInfo(Util.createProviderInfo(Util.TOPIC, "rmi://localhost:1099/",
"org.exolab.jms.jndi.InitialContextFactory"));
   System.out.println("Sending message: " + payload);
sendMessage(ACLMessage.REQUEST, pm);
 }
```

sendMessage() yordamı ile JMS sunucusu bilgilerinden, mesaj gönderme mekaniği tasarım kalitesini arttırmak amacı ile ayrılmıştır.

```
 private void sendMessage(int performative, AgentAction action) {
   ACLMessage msg = new ACLMessage(performative);
   ...
   msg.setOntology(ontology.getName());
   try {
     getContentManager().fillContent(msg, new Action(jmsProxyAgent, action));
     msg.addReceiver(jmsProxyAgent);
     send(msg);  //Standart JADE ACL mesaj gönderme yordamı
         try {
                 AMSService.deregister(this);
                 this.doDelete();
                 System.out.println(this.getAgentState().toString());
         } catch (FIPAException e) {
                 …
                 e.printStackTrace();
         }
   } catch (Exception e) {
     ...
   }
 }
```

sendMessage() yordamı ile mesajın ontolojisi belirlenip vekil etmen aracılığı ACL mesajı halinde yayıncı etmen belirlenen JMS ara katmanı üzerindeki konuya gönderilir. Etmen'nin kaydı yerel platformdan silinir ve en sonunda etmenin platformdaki kopyası silinir.

Abone tarafında etmen başlangıcı için önerilen mimari gerçeklemesi şöyle yapılabilir:

```
protected void setup() {
  ...
  // Ontolojiye kayıt
  getContentManager().registerOntology(ontology);
  lookupAgent();
  // JMS sunucusuna bağlanmak için gerekli bilgiler
  Properties props = new Properties();
  props.put(Context.INITIAL_CONTEXT_FACTORY,
"org.exolab.jms.jndi.InitialContextFactory");
props.put(Context.PROVIDER_URL, "rmi://localhost:1099");
  InitialContext ctx;
  try {
ctx = new InitialContext(props);
    // Konu bağlantısı için hazırlık
    TopicConnectionFactory topicFactory = (TopicConnectionFactory)
ctx.lookup("JmsTopicConnectionFactory");
    TopicConnection topicConnection = topicFactory.createTopicConnection();
    topicConnection.start();
    TopicSession topicSession = topicConnection.createTopicSession(false,
Session.AUTO_ACKNOWLEDGE);
    // İlgilenilen konuya abone olma işlemi
Topic topic = (Topic) ctx.lookup("topic1");
```

```
subscriber = topicSession.createSubscriber(topic);
      subscriber.setMessageListener(this);
      ...
   } catch (NamingException e) {
      ....
      e.printStackTrace();
   } catch (JMSException e) {
     ....
     e.printStackTrace();
   }
...
}
```

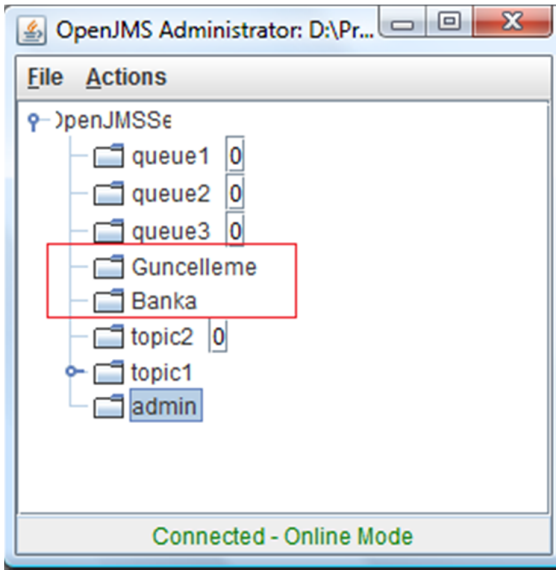
Burada abone etmen, mimari gereği istediği bir JMS sunucusunun istediği konusuna abone olur ve JMS ara katmanından gelecek olan mesajları dinlemek üzere yayınla abone ol düzeni gereği "onMessage()" geri çağırma (callback) yordamı tanımlar JMS ara katmanı abone etmeni bu yordam üzerinden bildirir. Bu yordam da YöneticiEtmen (abone) gelen mesajı deserilize edip etmen haline getirir. Etmenden kimlik bilgisini alıp, etmen tanımlama hizmetine verir. Ayrıca etmen tanımlama hizmetine gelen etmenin, ilgili diğer bilgilerini sunar ve yerel platforma kayıt edip çalışmasını başlatır:

```
public void onMessage(Message msg) {
    ...
    try {
      String msgPayload;
      if (msg instanceof ObjectMessage) {
        System.out.println("ObjectMessage")
        msgPayload = (String) ((ObjectMessage) msg).getObject();
      } else if (msg instanceof TextMessage) {
```

```
        System.out.println("TextMessage");
        msgPayload = ((TextMessage) msg).getText();
      } else {
        System.out.println("Unsupported Message Format");
        msgPayload = "Unsupported Message Format";
      }
      ...
      // Gelen mesaj deserilize edilip etmen formuna dönüştürülür
      Agent clientAgent = (Agent) deserializeString(msgPayload);
      //Gelen etmeni platforma kaydetmek için etmen tanımlayıcı hizmeti kullanılır
      AMSAgentDescription amsd = new AMSAgentDescription();
      //Etmenden kimlik bilgisi alınır
      AID client = clientAgent.getAID();
      //Etmen tanımlayıcı hizmetine gerekli bilgiler verilir
      amsd.setName(client);
      amsd.setOwnership("none");
      amsd.setState(AMSAgentDescription.INITIATED);
      // Yerel platforma kayıt
      AMSService.register(this,amsd);
      //Etmen yerel platformda çalışmaya başlatılır
      clientAgent.run();
    } catch (Exception jmse) {
      System.out.println("Error in JMS Message Extraction: " + jmse.toString());
    }
    ...
}
```

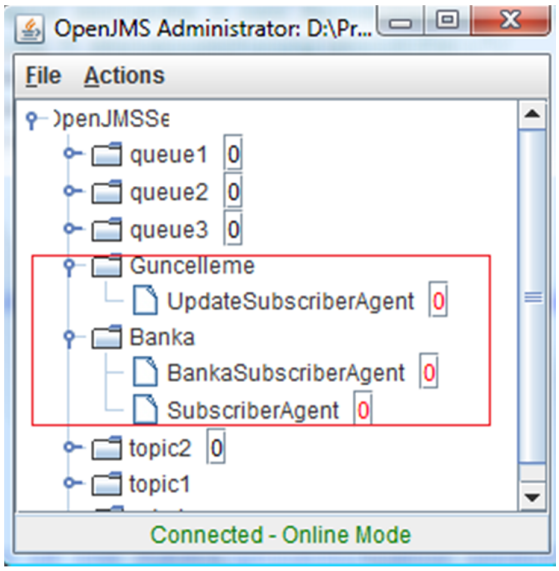
Burada hem yayıncı hemde abone etmen ve ayrıca vekil etmen sistem mimarisi kısmında anlatılan Şekil 5.3 ile gösterilen mesaj akışında belirtim seviyesinde verilen yapının kritik gerçekleme detayları verilmiştir. Şekil 5.3'te verilen mesaj akışı belirtim seviyesinde mimariye sağdık kalınarak çeşitli gerçeklemeler sunulabilir.

Burada anlatılanları somut bir örnek üzerinde inceleyelim. Yayınla abone ol mekaniğinin tün varyasyonlarını içermesi açısından çoklu konu, aynı konuya çoklu yayıncı ve çoklu abone unsurlarını içeren bir örnek problem uzayı tanımlanır. Bunu sağlamak üzere JMS sunucusu olarak kullandığımız OpenJMS ara katmanına "Guncelleme" ve "Banka" adlarında iki konu ekleriz. Şekil 5.5'te ara katman üzerindeki konular görünmektedir.



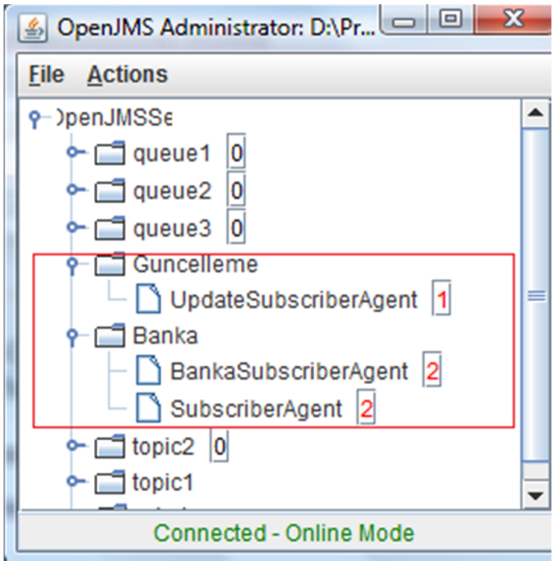
Şekil 5.5 : OpenJMS üzerinde tanıtılan "Guncelleme" ve "Banka" konuları.

Burada gerçekleşecek olan senaryolar, güncelleme ile ilgilenen abonelerin "Guncelleme" ve banka ile ilgilenen abonelrin de "Banka" konularına abone olması ve bu konulara bilgi uzmanı etmenlerin yayınlanması ve yayınlanan etmenlerin abonelere bildirilmesini kapsar. Bu bağlamda "Banka" konusuna "SubscriberAgent" ve "BankaSubscriberAgent" etmenleri, "Guncelleme" konusun da "UpdateSubscriberAgent" etmeni abone olur. Şekil 5.6'da OpenJMS konularına abone olan etmenler görülmektedir. Artık abone etmenler ilgilendikleri konulara yayın yapılmasını beklerler. Bu sırada "ClientAgent" ve "BankPublisherAgent" "Banka" konusuna, "UpdatePublisherAgent" ise "Guncelleme" konusuna yayın yapar. OpenJMS konulara yapılan yayınları ilgili abonelere bildirir.



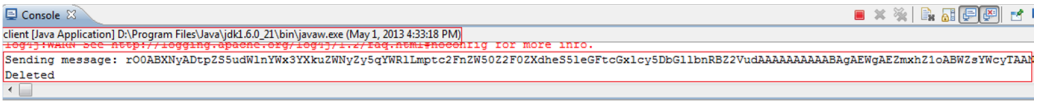
Şekil 5.6 : Abone etmenler.

Yayınlanan etmenler ara katman konularındaki mesaj sayısını arttırır. Şekil 5.7'de görülmektedir.

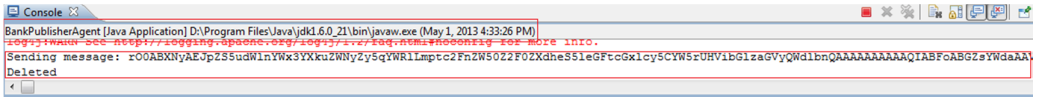


Şekil 5.7 : Etmen yayınından sonra ara katman konularındaki görünüm.

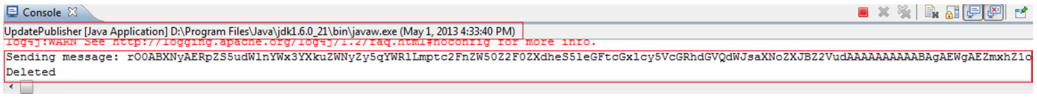
Bildirilen etmenler mesajdan ayıklanıp sistemde ayağa kaldırılırlar ve çalışmaları başlatılır. Böylece etmenler kaynak konaklarında silinip hedef konakta çalışıp sonlanırlar. Şekil 5.8, 5.9 ve 5.10'da etmenlerin kendilerini yayınlamaları ve platformdan silmeleri görülmektedir.



Şekil 5.8 : ClientAgent etmeninin kendini göndermesi.

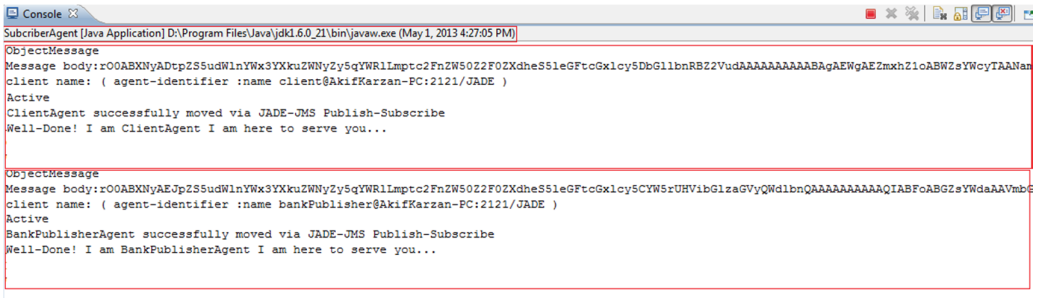


Şekil 5.9 : BankPublisherAgent etmeninin kendini göndermesi.



Şekil 5.10 : UpdatePublisherAgent etmeninin kendini göndermesi.

Şekil 5.11'da "SubscriberAgent" ta bildirilen "ClientAgent" ve "BankPublisherAgent" görülmektedir. "SubscriberAgent" "Banka" konusuna abone olduğundan dolayı bu konuya yayın yapan iki etmen de "SubscriberAgent" a ara katman tarafında bildirilmiştir ve "SubscriberAgent" mesaj içerisinde gelen etmenleri mesajdan ayıklayıp kendi ortamında başlatmıştır.

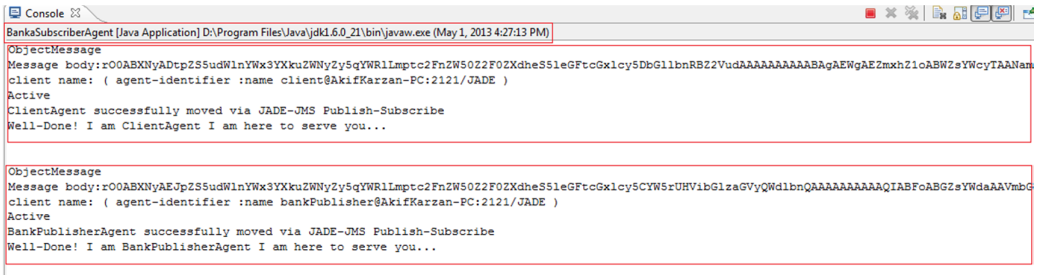


Şekil 5.11 : SubscriberAgent'ın konağındaki görünüm.

Şekil 5.12'de de benzer şekilde aynı konuya abone olan "BankaSubscriberAgent" için benzer işlemler görülmektedir.

Şekil 5.13'de ise "Guncelleme" konusuna abone olan "UpdateSubscriberAgent"ına bildirilen "UpdatePublisherAgent" görülmektedir. "UpdatePublisherAgent" etmeni yeni konağında ayağa kaldırıp çalışmaya başlar ve işlerini bitirince sonlanır.

Böylece yayınla abone ol mekaniğinin gerçekleşebilicek tüm senaryoları uygulamamızdaki çoklu etmen sisteminde gösterilmiştir.



Şekil 5.12 : BankaSubscriberAgent'ın konağındaki görünüm.

Şekil 5.13 : UpdateSubscriberAgent'ın konağındaki görünüm.

5.2.1 Fipa – akıllı fiziksel etmenler kurumu

Fipa.org'a göre, FIPA (Foundation of Intelligent Physical Agents) –Akıllı fiziksel etmenler kurumu- etmen tabanlı teknoloji ve kendi standartlarını diğer teknolojiler ile birlikte çalışabilmesini sağlayan bir IEEE Bilgisayar Topluluğu (Computer Society) standartlar organizasyonudur.

FIPA etmenler ve çoklu etmen sistemleri standartlar organizasyonu IEEE tarafından 2005 yılında onbirinci standartlar komitesi olarak kabul edilmiştir.

FIPA 1996 yılında heterojen ve etkileşen etmenler ile etmen tabanlı sistemlerin yazılım standart belirtimeri için üretilmiş İsviçre kökenli bir organizasyondur.Kurulduğundan beri FIPA etmen standartlarının gelişmesinde hayati bir rol oynamıştır. Etmen teknolojisini gerektiren bir çok etkinlik ve girişim düzenlemiştir. Ayrıca FIPA tarafından oluşturulan ve geliştirilen bir çok fikir şimdi yeni nesil Ağ/İnternet teknolojileri ve ilintili belirtimlere keskin bir şekilde girmektedir.

5.2.2 Jade – java etmen geliştirme iskeleti

Jade.tilab.com'a göre, JADE (Java Agent DEvelopment framework) tamamen Java dili ile gerçeklenmiş bir yazılım iskeletidir. JADE bir ara katman üzerinden çoklu etmen sistemlerinin gerçekleştirilmesini kolaylaştırır. FIPA belirtimleri ile uyumludur ve hata ayıklama ve kurulum fazlarını desteklemek üzere grafik araçlar

bulundurur. Etmen platformu işletim sistemleri dahi aynı olmayan makinalar arasında dağıtılabilir ve konfigürasyonlar grafik arayüz ile kontrol edilebilir. Konfigürasyon, gerektiğinde etmenleri bir makinadan diğerine geçirmek yoluyla çalışma zamanında da değiştirilebilir. JADE tamamen java dili ile gerçeklenmiştir ve minimum sistem gereksinimi JAVA 1.4 versiyonudur (çalışma zamanı ortamı veya JDK).

JADE ücretsiz bir yazılımdır ve telif hakkı Telecom Italia'da bulunmaktadır. LGPL (Lesser General Public Licence Version 2) lisansı ile korunmaktadır.

5.2.3 Jms – java mesajlaşma hizmeti

Wikipedia.org'a göre JMS (Java Message Service) uygulama programlama arayüzü, iki veya daha fazla istemci arasında mesaj göndermek için Java mesaj tabanlı ara katman arayüzüdür. JMS Java platform kurumsal tipinin bir parçasıdır ve Java topluluk süreci (JSR 914) tarafından geliştirilen bir belirtim ile tanımlanmıştır. JMS Java kurumsal tipine (JEE) dayalı uygulama bileşenleri arasında mesaj oluşturma, gönderme, alma ve okumaya izin veren mesajlaşma standardıdır. Dağıtılmış uygulamalarda farklı bileşenler arasındaki haberleşmenin gevşek bağlı, güvenilir ve asenkron olmasına izin verir.

6. SONUÇ VE ÖNERİLER

Bu çalışmada önerilen ölçeklenebilir, gevşek bağlı, verimli ve esnek bir çoklu etmen ekosistemidir. Bu ekosistem konu tabanlı etmen göçünü başaran haraketli etmenlerle oluşturulmuş bir hizmet tabanlı mimari modeli sunar.Bu sistem "Bilgi Uzmanı Ekosistemi" olarak adlandırılır. Bilgi uzmanı ekosistemi, bilgi uzmanı etmenler ve bunlara çalışma ortamı sağlayan çevrelerinin (konakların) karşılıklı ilişkileri ile meydana gelen ve süreklilik arz eden sistemdir.

Bilgi Uzmanı Etmen: Belirli bir konuda bilgisi olan ve bu konuya ilişkin problemleri çözebilen bu konuda danışmanlık yapabilen/yapan etmendir.

Kaynak Konak: Bilgi uzmanı etmenleri üreten konaktır ve/veya alt sistemdir.

Hedef Konak: Belirli bir konuda bilgi uzmanı etmenin yardımına ihtiyacı olan konaktır ve/veya alt sistemdir.

Bilgi uzmanı ekosistemi, bilgi uzmanı, iş yapabilen etmenler, bunları üreten/yetiştiren konaklar ve/veya alt sistemler ve belirli bir konuda danışmanlık hizmeti almak, iş yaptırmak (problemlerini çözdürmek) isteyen konaklar ve/veya alt sistemlerden oluşur. Burada bahsedilen ekosistem, bir konuda yardım almak isteyen alt sistemler ile bu konularda yardım edebilecek bu hizmeti veren alt sistemleri buluşturan bir sistemdir. Burada amaçlanan birbirlerine ihtiyaçları olan bu alt sistemlerin birbirlerini tanımadan buluşabilmesidir. Buna örnek olarak bankayı verebiliriz. Bankalar soyut anlamda, borç verenler ile borç almak isteyenlerin buluşma noktasıdır. Bankalar kimin parasının kime verildiği bilinmeden bu ihtiyacın giderildiği buluşma ortamlarıdır. Bilgi uzmanı ekosistemi de bir iş yaptırmak, danışmanlık hizmeti almak isteyenlerin bu hizmetlere ulaşabilecekleri güvenli bir ortamdır.

Bu çalışmada varolan etmen göçü düzenlerinden farklı olarak konu tabanlı yayınla/abone ol modeli benimsenmiştir. Sunulan mimari ile standartlara uygun ve platformlar arası hareket edebilen etmen sistemi başarılır.

6.1 Çalışmanın Uygulama Alanı

KOBİ bankacılığının yazılım sistemleri ile gerçeklenmesi bilgi uzmanı ekosistemi için çok uygun bir uygulama alanı oluşturabilir. Burada bankalar KOBİ'lerin yanında olup onları büyütürken ihtiyaçlarına en doğru ve en hızlı şekilde yanıt vermeyi amaçlarlar. Bu hizmeti hemen hemen tüm banklar vermektedir. Burada KOBİ'ler ihtiyaç duyduklarında bu danışmanlık hizmetini almak üzere bankalara başvururlar. Sistemde bankalar danışmanlık hizmetlerini kendi birikimlerine göre gerçeklerler. KOBİ'ler ise bankaların sunabilecekleri bu hizmetleri değerlendirip kendilerine en uygun olanı sunandan almak isterler. Bu sistem bilgi uzmanı ekosistemi ile yazılımsal olarak gerçeklenebilir. KOBİ (yazılım) alt sistemleri ihtiyaç duydukları danışma hizmetini (yazılım) banka alt sistemlerinden alırlar. Burada danışmanlık hizmetini bilgi uzmanı etmen sağlar. Bu hizmeti ortam şartlarını değerlendirip ihtiyaca uygun olacak şekilde sunar.

Seyahat acentalığı da sistem için uygun bir kullanım alanıdır. Bu alanda seyahat firmaları, gidilecek yerdeki konaklama hizmetleri veren firmalar, gidilecek yerdeki etkinlik düzenleyen firmalar, kalınan yer ile etkinlik yeri arasında ulaşım hizmeti veren firmalar ve müşteriler önerilen platform üzerinde buluşurlar. Böylece bir yerden başka bir yere bir etkinlik ve/veya tatil için gitmek isteyen insanlar bu hizmet sayesinde tüm bu işleri önceden planlayabilirler. Ayrıca bu işlerden para kazanan farklı işler yapan firmalarda birbirleri ile sistem üzerinden haberleşerek müşteri potansiyellerini arttırırlar. Önerilen sistem ile oluşturulabilecek paketlerde hem müşteriler hem de hizmet veren firmalar karlı çıkar.

Güncelleme süreçlerini de sistemin bir başka uygulama alanı olarak söyleyebiliriz. Bu süreçleri önerilen mimariye uygun olarak gerçekleyen sistemler, güncelleme işlerindeki kullanıcı tarafa işlikin yüklerini hafifletirken, kullanıcılar açısından da kendi istedikleri zamanda ve istedikleri üründe güncelleme alma imkanı sunar.

Yukarıda bahsedilen uygulama alanları dışında herhangi başka konularda danışmanlık hizmeti veya iş yapma hizmeti de verebilir bilgi uzmanı etmenler. Uygulamaya göre ihtiyaç duyan bir konakta yapılması gereken bir hesaplama sürecini gerçekleyebilir.

KAYNAKLAR

Ametller, J., Robles, S. ve Borrel, J. (2003). Agent Migration over FIPA ACLMessages. Mobile Agents for Telecommunication Applications Lecture Notes in Computer Science Volume 288, 210-219.

Currie, E., Chambers, D.ve Lyons, G. (2003). A JMS Message Transport Protocol for the JADE Platform. *IEEE International Conference on Intelligent Agent Technology*. IAT, 2003, IEEE/WIC 596-600.

Eugster, P., Felbel, P., Guerraoui, R. ve Kermarrec, A.(1976). The many faces of Publish/Subscribe. ACM Computing Surveys, Vol. 35, No. 2, Haziran, 114-131.

Maes, P. (1994). Agents that Reduce Work and Information Overload, *Communications of the ACM*, 37,7, 31-40.

Malik, S., Qureshi, N., Ali, A., Ahmad, H., Suguri, H., Snigaroff, R., (2005). Inter Platform Mobility in FIPA Compliant Multi-Agent Systems. *IEEE International Conference on Active Media Technology*. AMT 2005. Proceedings of the 2005. 393-396.

Paraschiv, M., Stefanescu, A.ve Almasi, A.(2011). Agent Based Implementation of a P2P Publish/Subscribe System. Computer Science Master Research.

Pham, V., ve Karmouch, A. (1998). Mobile Software Agents: an Overview. IEEE Communications Magazine.

Sariel, S. (2002). Kullanıcı Kesitleriyle Yüz İfadelerini Analiz Eden Bir Çoklu Etmen Sistemi Uygulaması, Yüksek Lisans Tezi İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü.

Shen, Z., Li, R. ve Luo, J. (2009). Mobile Agent Based Middleware using Publish/Subscribe Mechanism in Wireless Sensor Networks. International Conference on Communication Software and Networks.

Singh, M. P. ve Huhns, M. N. (1998).Readings in Agents, Morgan Kauffmann Publishers, Inc., San Francisco, California.

Şahingöz, O. K. ve Erdoğan, N. (2004). AGVENT: Agent Based Distributed Event System. Proceedings of 30th The Conferenceon Current Trends in Theory and Practice of Computer Science (SOFSEM 2004). Prag, Czech Republic.

Weiss, G.(2000). Multi-Agent Systems: A Modern Approach to Distributed Artificial Intelligence, Massachusetts Institute of Technology.

Wooldridge, M., ve Jennings, N.R. (1995). Intelligent Agents: Theory and Practice, The Knowledge Engineering Review, 10,2, 115-152.

Url-1 <*http://www.fipa.org*>, alındığı tarih: 21.03.2013.

Url-2 <*http://www.tilab.jade.org*>,alındığı tarih: 21.03.2013.

Url-3<*http://en.wikipedia.org/wiki/Java_Message_Service*>,alındığı tarih: 21.03.2013

ÖZGEÇMİŞ

Ad Soyad: Mustafa Akif Karzan

Doğum Yeri ve Tarihi: İstanbul, 1983

Adres: Yasemen sokak No:14/2 Yeniköy-SARIYER, İSTANBUL

E-Posta: akifkarzan@yahoo.com

Lisans: İstanbul Teknik Üniversitesi

TEZDEN TÜRETİLEN YAYINLAR/SUNUMLAR

- Karzan, M. A., Erdoğan N., 2013: Topic Based Agent Migration Scheme via Publish/Subscribe Paradigm. *International Conference on Compter Engineering and Technology (ICCET 2013)–Science and Engineering Institute*, April 13-14, 2013 Vancouver, Canada.