İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

JADE DAĞITIK ETMEN SİSTEMİ İÇİN YAYINLA / ABONE OL TABANLI İLETİŞİM EKLENTİSİ

YÜKSEK LİSANS TEZİ

Müh. Kıvanç ÖZBEK

Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ

Programı : BİLGİSAYAR MÜHENDİSLİĞİ

OCAK 2008

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

JADE DAĞITIK ETMEN SİSTEMİ İÇİN YAYINLA / ABONE OL TABANLI İLETİŞİM EKLENTİSİ

YÜKSEK LİSANS TEZİ

Müh. Kıvanç ÖZBEK
(504031549)

Tezin Enstitüye Verildiği Tarih : 18 Aralık 2007
Tezin Savunulduğu Tarih : 28 Ocak 2008

Tez Danışmanı : Prof.Dr. Nadia ERDOĞAN

Diğer Jüri Üyeleri Prof.Dr. Oya KALIPSIZ (Y.T.Ü)

Yrd.Doç.Dr. A. Şima ETANER UYAR

OCAK 2008

ÖNSÖZ

Öncelikle, tüm yüksek lisans öğrenimim süresince anlayışlı tutumu, doğru yönlendirmeleri, çalışma isteği aşılayıcı telkinleri ve yerinde uyarılarıyla verimli bir süreç yaşamama ve zevk aldığım bir çalışma ortaya çıkarmama yardım eden Sayın Prof. Dr. Nadia Erdoğan'a teşekkürü bir borç bilirim.

Ayrıca uzun yıllar boyu destekleri esirgemeyen, uzakta olsalarda varlıklarını hep yanımda hissettiğim, bugünlere gelmemde en büyük paya sahip olan aileme, katkılarından dolayı İTÜ'deki diğer hocalarıma ve desteklerini esirgemeyen yakın arkadaşlarıma teşekkür ederim.

Ocak 2008 Kıvanç Özbek

İÇİNDEKİLER

KISALTMALAR **iv**
TABLO LİSTESİ **v**
ŞEKİL LİSTESİ **vi**
ÖZET **vii**
SUMMARY **viii**

1. GİRİŞ **1**

2. OLAY VE ETMEN SİSTEMLERİ **3**
2.1 Tanım 3
2.2 Olay Modelleri 3
2.2.1 Eşdüzeyde iletişim modeli 4
2.2.2 Aracılı iletişim modeli 4
2.2.3 Dolaylı iletişim modeli 4
2.2.4 İletişim modellerinin karşılaştırılması 4
2.3 Olay Servisleri 5
2.3.1 Topoloji 5
2.3.2 Özellikler 8

3. JADE **12**
3.1 Genel Yapı 12
3.2 Etmenler Arası Haberleşme 16
3.2.1 Konu odaklı iletişim 19
3.3 Olay Sistemi Olarak JADE 19
3.3.1 JADE iletişim modeli 20
3.3.2 JADE olay servisi 20
3.4 Diğer JADE Özellikleri 21

4. YAYINLA ABONEOL TEMELLİ İLETİŞİM **24**
4.1 Tanım 24
4.2 İletişim Akışı 25
4.3 Olay Bildirim Servisi ve Yönlendirmeler 26
4.3.1 Olay taşması yöntemi 26
4.3.2 Kayıt taşması yöntemi 27
4.3.3 Filtrelemeli yönledirme yöntemi 27
4.4 Kayıt Modelleri 28
4.4.1 Konu tabanlı kayıt modeli 28
4.4.2 İçerik tabanlı kayıt modeli 30

4.5 Yayınla Abone Ol Yapısının Getirileri 31
4.5.1 Uzay bağımsızlaşması 31
4.5.2 Zaman bağımsızlaşması 32
4.5.3 Senkronizasyon bağımsızlaşması 32

5. JADE İÇİN İÇERİK TABANLI YAYINLA / ABONEOL EKLENTİSİ 33
5.1 Uygulanan Kod Yazma Standartları 33
5.2 Mesaj Oluşturma Yapısı 35
5.3 Kayıt Modeli 36
5.4 Olay Bildirim Servisi 41
5.5 İletişim Akışı ve Fonksiyon Yorumlama Yapısı 46
5.5.1 Genel mesaj akışı 46
5.5.2 Fonksiyon yorumlama mekanizması 50

6. ÖRNEK UYGULAMA 53
6.1 Grup Yöneticisi Etmen 53
6.2 Proje Yöneticisi Etmen 55

7. SONUÇLAR VE TARTIŞMALAR 58

KAYNAKLAR 60

ÖZGEÇMİŞ 61

KISALTMALAR

JADE **:** Java Agent Development Framework

TILAB **:** Telecom Italia Laboratory

FIPA **:** Foundation for Intelligent Physical Agent

AMS **:** Agent Management System

DF **:** Directory Facility

JVM **:** Java Virtual Machine

ACL **:** Agent Communication Language

TABLO LİSTESİ

Sayfa No

Tablo 5.1 Hisse senedi takibi kayıt örneği …… 40

ŞEKİL LİSTESİ

Sayfa No

Şekil 2.1 : Hiyerarşik varlık yapısı..... 6
Şekil 2.2 : Çevrimsiz varlık yapısı..... 6
Şekil 2.3 : Anti çevrimsiz etmen yapısı..... 7
Şekil 2.4 : Ayrık ara aygulamalı olay sistemleri..... 7
Şekil 2.5 : Tümleşik ara uygulamalı olay sistemleri..... 8
Şekil 2.6 : Servis kalitesi sınıflandırma unsurları..... 9
Şekil 2.7 : Servis kalitesi sınıflandırması..... 10
Şekil 3.1 : JADE etmen sistemi genel yapısı..... 13
Şekil 3.2 : Etmen hayat döngüsü..... 16
Şekil 3.3 : JADE'de asenkron etmen iletişimi..... 18
Şekil 3.4 : JADE sarı sayfalar servisi..... 21
Şekil 3.5 : Dağıtık JADE etmen sistemi yapısı..... 23
Şekil 4.1 : Yayınla-abone ol iletişim akışı..... 26
Şekil 4.2 : Olay taşmalı iletişim akışı..... 27
Şekil 4.3 : Filtrelemeli iletişim akışı..... 28
Şekil 4.4 : Hiyerarşik konu tabanlı iletişim modeli..... 29
Şekil 4.5 : İçerik tabanlı iletişim modeli..... 31
Şekil 5.1 : Kabul metot akışı..... 42
Şekil 5.2 : Parametrik mesaj işleme akışı..... 43
Şekil 5.3 : Genel mesaj akışı..... 47
Şekil 5.4 : Fonksiyon yorumlama akışı..... 52

JADE DAĞITIK ETMEN SİSTEMİ İÇİN YAYINLA / ABONE OL TABANLI İLETİŞİM EKLENTİSİ

ÖZET

Tez çalışmasında JADE dağıtık etmen sistemi üzerine yayınla / abone ol esasına göre çalışan içerik tabanlı bir eklenti gerçeklenmiştir. Eklenti için kayıt mekanizması, olay bildirim servisi yada JADE yapısındaki adıyla konu yönetim servisi ve fonksiyon yorumlama mekanizması üzerinde çalışılmıştır.

Kayıt mekanizması için bir veri saklama modeli tanımlanmıştır. Bu model yardımıyla istemcilerin isteklerini belirli detaylarda iletmeleri sağlanmıştır. Bu detaylar hangi konu ile ilgilendikleri, bu konu üzerindeki ek kısıtları ve ilgili durum oluştuğu takdirde yürütülecek işlemler olarak sıralanabilir.

Konu üzerindeki kısıtları belirtmek üzere bir boole fonksiyon tanımı kullanılmıştır.. Abonelerin kısıtlarını belirten bu fonksiyonları yorumlamak üzere ayrı bir sınıf üzerinde genel bir yapı tanımlanmıştır. Bu genel yapı sayesinde parametreler değişse bile ortak bir yapı üzerinden işlem yapılmaya devam edilebilecektir. Fonksiyon yorumlama yapısının kullanımı, mesaj içeriklerinin kullanılması gerektiği için olay bildirim servisi içine yerleştirilmiştir.

İstemcilerin ilgilendikleri durumlar oluştuğu takdirde yapılmasını istedikleri işlemi tanımlayabilmelerine olanak sağlanarak bir delegasyon mekanizması oluşturulmuştur. Bu sayede etmenlerin başka etmenler adına kayıt olabilmesine olanak sağlanmıştır. Delegasyon yapısı özellikle hiyerarşik yapılarda ve erişimlerin kısıtlı olduğu ortamlarda kullanılabilmek üzere düşünülmüştür. Delegasyon yapısı sayesinde sistemde daha etkin olan bir etmen diğer etmenler adına kayıt olup, onlara yönlendirilmesini istediği mesajları belirleyebilir duruma gelecektir. Böylece belirli bilgiye ve yetkiye sahip etmenler görev dağılımı yapabilme gibi özelliklere sahip olacaklardır.

PUBLISH / SUBSCRIBE BASED COMMUNICATION ADD FOR JADE DISTRIBUTED AGENT SYSTEM

SUMMARY

In this study, an add that is working according to content based publish / subscribe rules for JADE distributed agent system, is implemented. Developments have been take place over subscription mechanism, event notification service or topic management service as the name in the JADE system, and function interpretation mechanism.

A data storage model has been defined for the subscription mechanism. With the help of this model, notification of client demands with some detail has been enabled. These detail are, the topic clients interested in, the constraints on the topic, and the actions to be taken when the conditions have been met.

A Boolean function definition has been used to define the constraints on the topic. To interpret the functions that define the constraints of the subscribers, a generic structure has been defined on a separate class. With this generic structure, it will be possible to continue operation even the parameters are changed. Usage of function interpretation mechanism has been take place inside the event notification service, because the contents of messages have to be used.

By providing the ability to define the actions, to be taken when the conditions met, for the clients, a delegation mechanism has been build. This ability provides subscription in place of another agent. Delegation mechanism has been thought to be used especially in the systems with hierarchical structure and the environments that have limited access. With the help of delegation mechanism, a more active agent in the system will become a condition that it can subscribe on be half of other agents and it can determine the messages that it wants other agents to get. This will bring the abilities like mission distribution, for the agents that have enough knowledge and authority.

1. GİRİŞ

İnternet teknolojilerinin hızla gelişmesi ve dağıtık ortamların giderek daha önemli hale gelmesi dolayısıyla etmen ve olay sistemleri yazılım dünyasında her geçen gün daha da ilgi çekmektedir. Sistemlerin bağımsız çalışan yetenekli parçalar halinde daha verimli kullanılabilmesi sebebiyle etmen sistemleri üzerine araştırma ve geliştirme çalışmaları hızla sürmektedir.

Getirdiği bir çok işlevsel avantajdan ötürü, etmen sistemlerinde yayınla-abone ol iletişim yöntemi daha çok tercih edilmektedir. SIENA, CORBA, JADE gibi sistemler konu tabanlı ve içerik tabanlı yayınla-abone ol mekanizmalarıyla kullanıcılara önemli esneklikler kazandırmaktadır.

Yayınla-abone ol sistemlerinde, sistem içi varlıkların birbirlerinden soyutlanması ve fiziksel olarak birbirlerini tanımak yerine belirli ortak gruplamalar üzerinden iletişim kurulması benimsenmiştir. Bu kapsamda basit olarak konu diye tanımlanan yapılar altında, istemci varlıkların kayıt olması ve bu konuyla ilgili durumlardan haberdar edilmesi sağlanabileceği gibi içerik tabanlı uygulamalarla durumların detaylı olarak filtrelendikten sonra abonelere iletilmesi de sağlanabilmektedir. Bu iki yöntemi kısmen birleştirdiği söylenebilecek bir diğer yöntem ise konu tabanlı iletişimde kullanılan konuları hiyararşik bir yapıda tanımlayarak istemcilerden tercih edenlerin daha detay içerikte konulara kayıt olmasının sağlandığı yöntemdir. Bu yapılarda farklı kayıt modelleri ve ara uygulamalar kullanılmakla beraber temel yaklaşımlar genellikle ortak bir paydada kesişmektedir.

Çalışma kapsamı olarak bir etmen sistemine içerik tabanlı bir yayınla-abone ol mekanizması eklenmiştir. Eklenti, normal bir içerik tabanlı iletişim yapısının tüm gereklerini sağlamaktadır. İstemciler ilgilendikleri konuların yanı sıra, bu konu ile ilgilenme düzeylerini yani ilgilendikleri konu üzerine gelen mesajlara ilişkin filtreleme kısıtlarını da belirtirler. Bu yapı sayesinde ilgili etmen sisteminde, gönderilen mesajların içeriklerinin incelenmesi ve daha önce kayıt olan abonelerin

belirttikleri kısıtlara uyması durumunda, ilgilenen etmenlere ilgili mesajların iletilmesinin yanı sıra bir delegasyon yapısı oluşturma imkanı da sağlanmıştır. Bu sistem içinde, etmenler başka etmenler adına kayıt olabilmekte ve başka etmenler adına uyarılma durumlarına karar verebilmektedir. Böylece etmenler arasında yetkisel farklılıklar oluşturulabilmektedir. Bu eklentinin kullanıldığı bir sistemde iletişimi önceden belirlenmiş etmenlerin kontrol etmesi sağlanabilir. Bu etmenler akıllı uygulamalar olabileceği gibi arka planda sisteme daha hakim ve sistem hakkında daha fazla bilgi sahibi kişilerin yönettiği uygulamalar da olabilir. Her durumda merkezi yada kısmi merkezi bir iletişim kontrol yapısı oluşturulmuş olur. Bu yapıda hangi etmenin hangi konularda mesaj alabileceği ve bu mesajlara hangi filtrelemelerin uygulanacağı yetkili yada etkin etmenler tarafından belirlenir.

İletişimin kontrol edilmesinin yanısıra delegasyon özelliği bir iletişim destek yapısı olarak da kullanılabilir. Mevcut sistemde etmenlerin istekte bulundukları konu ve içeriklerin yanı sıra, bilgilenmelerinin faydalı olacağı hata yada sistem çökmesi gibi uygulama dışı beklenmedik olaylar dolayısıyla üretilen mesajları almaları da sağlanabilir.

Delegasyon yada kontrol yapısı bir dizi ön tanımlı davranış yapısının kayıt sırasında olay yönetim sistemine bildirilmesiyle sağlanır. Bu davnışlar sayesinde bir etmen bir mesajın başka bir etmene de yönlendirilmesini, mesaj göndericisi sanki kendisiymiş gibi başka bir etmene gönderilmesini yada tamamen farklı bir içerikle ve farklı bir etmenden gelmiş gibi yönlendirilmesini sağlayabilir. Tüm davranış yapılarının kullanılması durumda ilgili sistemde kayıt mekanizmasında bir yetkilendirmeye gidilmesi güvenlik açısından daha uygun bir yaklaşım olacaktır.

Eklenti, ilgili uygulamanın eski iletişim yöntemini de desteklemesini sağlamış, bu sayede aynı anda eşdüzeyde, konu tabanlı ve içerik tabanlı yayınla abone ol iletişim yöntemlerini uygulayabilen, farklı beklenti ve ihtiyaçlara karşılık verebilen karma bir yapı oluşturulmuştur.

Tez kitabının ilgili bölümlerinde olay ve etmen sistemlerinden, çalışma kapsamında üzerinde çalışılan ve eklenti hazırlanan JADE sisteminden ve yayınla-abone ol iletişim yapısından ve yapılan çalışmanın detaylarından bahsedilecektir.

2. OLAY VE ETMEN SİSTEMLERİ

2.1 Tanım

Dağıtık yapılarda uygulama geliştirirken baz alınan iki önemli sistem, etmen sistemleri ve olay tabanlı sistemlerdir. Dağıtık bir yapı üzerine kuruldukları için ikisinde de temel iletişim bileşenlerinin iş birimlerine transferini sağlamak amacıyla benzer yaklaşımlar kullanılmaktadır. Olay tabanlı sistemlerde iş birimlerinin haberdar olması gereken unsurlar olaylar iken, etmen sistemlerinde iş birimlerine mesajlar iletilir. İki yapıda iş yapan birimler farklı olmasına rağmen iletişim gereksinimleri ve bu iletişimin kontrol edilmesi gerekliliği sebebiyle modelleme ve servis yapılanmaları birbirine benzer. Bu sebeple etmen sistemlerini modellemek için olay modellerini incelemekte fayda olacaktır. Olay tabanlı sistemlerin sınıflandırılmasında farklı kabuller ve isimlendirmeler olmasına rağmen temelde tüm yapılar aynı unsurları ifade eder.

Olay tabanlı sistemler olay modeli ve olay servisi olmak üzere iki unsur barındırır. Olay servisi, bir olay modelini gerçekleyen, dolayısıyla olay sistemine olay tabanlı iletişim sağlayan özel yazılımdır. Olay modeli ise olaylara dayalı bir iletişimi tanımlayan bir dizi kuralı içerir [10].

2.2 Olay Modelleri

Olay tabanlı sistemlerde olayları yaratan yada kullanan birimlere “varlık” denir. Olayların mesajlara karşı düştüğü düşünülürse, varlıkların da etmenlere karşılık geldiğini düşünebiliriz.

Olay modelleri iletişim şekline göre üç farklı şekilde oluşturulabilir; bunlar, eşdüzeyde, aracı üzerinden ve dolaylı iletişime dayalı modellerdir. Her üç modeli de uygulayan yapılar bulunmakla birlikte fark ihtiyaçlara göre her birinin öne çıktığı durumlar bulunmaktadır.

2.2.1 Eşdüzeyde iletişim modeli

Eşdüzeyde yada uç uça iletişim esasına dayalı modellerde, varlıklar iletişim kuracakları varlıkları tanırlar ve doğrudan birbirleri ile veri alış verişine girerler. Bu yapıda varlıklar birbirlerinin yerini bilir, nasıl iletişime geçeceğini ve iletişim formatının nasıl olacağını bilirler. Küçük çaplı yada çok değişim göstermeyen yapılarda avantaj sağlamaktadır.

2.2.2 Aracılı iletişim modeli

Aracı üzerinden iletişim temelli modellerde, varlıklar aracı bir uygulama vasıtasıyla iletişim kurarlar. Varlıklar birbirlerini yada nasıl iletişim kuracaklarını bilmek zorunda değillerdir. Bu modelle tek bir aracı uygulama yada birden çok aracı uygulama kullanılabilir. Çoklu aracılı modellerde, aracılar benzer fonksiyonları taşıyabildikleri gibi farklı fonksiyonları da olabilir. Aracılar fonksiyonel açıdan eşit olduklarında varlıklar herhangi biri üzerinden işleyişlerini sürdürebilirler. Böyle bir aracı SIENA modelinde "olay sunucu" olarak isimlendirilir [10]. CORBA olay modeli ise her biri farklı olayları yayan birden çok olay kanalı kullanabilir [10].

2.2.3 Dolaylı iletişim modeli

Dolaylı iletişime dayalı modellerde ise olay tipleri iletişimde önemli rol oynar. Tüketici olarak nitelendirilen varlıklar belirli olay tiplerinden olaylar meydana geldiğinde bundan haberdar olmak üzere ilgili olay tiplerine kayıt olurlar. Kayıt bilgileri yerel bir kütüphanede tutulur ve üretici görevini üstlenen varlıklar tarafından olaylar oluşturulduğu zaman, olay servisi tüm kayıtlı tüketicilere onların olay denetimcilerini çağırarak olayları iletir [10].

2.2.4 İletişim modellerinin karşılaştırılması

Eşdüzeyde ve aracı ile iletişimde varlıklar, bir olay oluştuğu durumda yada bir mesaj göndermek için ilgili varlık üzerinde yada aracı uygulama üzerinde bulunan metotları tetiklemek durumunda kalırlar. Halbuki dolaylı iletişimde böyle bir zorunluluk olmadan olay tipleri yada konular üzerinden iletişim sağlanabilmektedir.

Dolaylı iletişimin kullanılmadığı modellerde bir diğer problem de iletişim kuracak birimlerin birbirlerini bilme zorunluluğudur. Bu anlık sohbet programlarındaki bazı dosya aktarım yöntemlerinde kullanılan uç uca bağlantıda olduğu gibi iki tarafın

birbirlerinin adreslerini bilmelerine benzer. Aracılı iletişim de ise iletişim kuracak varlıklar özellikle aracılar aynı işlevsellikte olmadığı durumlarda doğru aracılara ulaşmak için onlar hakkında bilgi sahibi olmalıdır. Bu bilgi sahibi olma kavramı, anonimlik derecesi diye de ifade edilir. Birbirlerini bilmeyen varlık yada uygulamalar kendi aralarında anonim olarak adlandırılırlar.

Ancak geliştirmenin kolay olması yada başka sebepler yüzünden de eşdüzeyde iletişim modelleri tercih edilebilir. Örneğin üretici ve tüketiciler arasındaki güven seviyesini tayin etmek isteyen uygulamalar için bu yöntem kullanışlı olabilir [10].

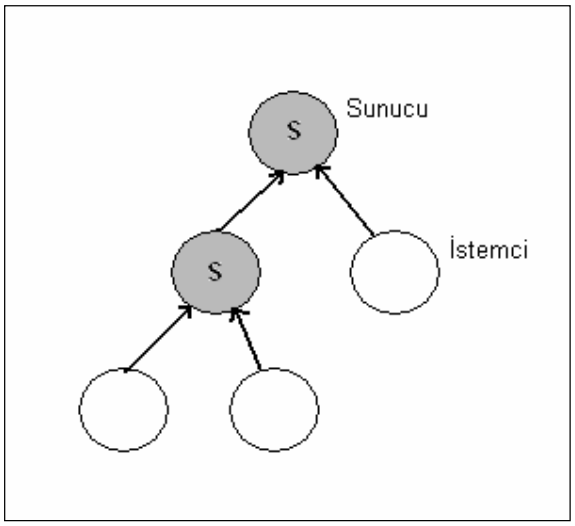
2.3 Olay Servisleri

Olay modelleri ile nasıl bir iletişim kurulacağı belirtildikten sonra, bu modeli gerçekleyecek olay servisi ile de kurulacak sistemde varlıkların ve ara uygulamaların yerleri ve de sistemin sağlayacağı özellikler belirlenir. Yapılan sınıflandırmalarda tanımlar ve isimlendirmeler değişse de temel unsurlar aynıdır.

2.3.1 Topoloji

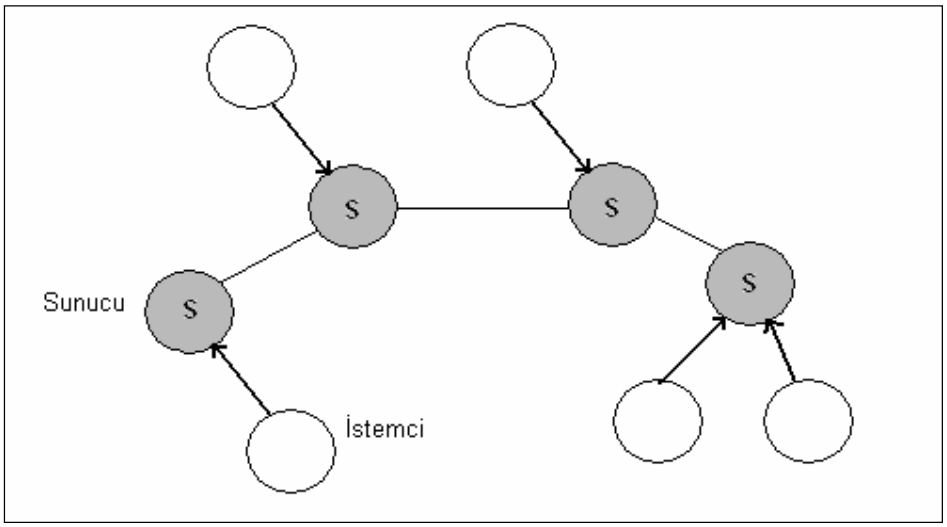
Olay servislerinin sınıflandırmasında topoloji kullanılan ara uygulamanın yerine yada sunucu hizmeti veren uygulamaların bağlanış şekillerine göre yapılabilir. Sunucuların oluşturduğu graflara göre yapılan sınıflandırma, bu uygulamaların iletişim şekillerine dayanır. Ara uygulamaya dayalı sınıflandırmada ise ara uygulama sayısı ve uygulamanın bulunduğu adres uzayı önem kazanmıştır.

Ara uygulamaların bağlanış şekillerine göre hiyerarşik, çevrimsiz, anti çevrimsiz ve hibrit olarak sınıflandırılabilir. Hiyerarşik yapıda her sunucu yada ara uygulama bir diğeriyle anne-çocuk ilişkisi içerisindedir. Ana sunucu tüm istemcilerinden bildiri, yayın yada kayıt isteği alabilir ancak geri sadece bildiri gönderecektir [7]. Yani temel iletişimin yönü varlıklardan ara uygulamalara ve onlarında kendi içlerinde çocuktan anneye doğrudur. Bu yapı ve iletişim yönleri Şekil 2.1'de gösterilmiştir [7].



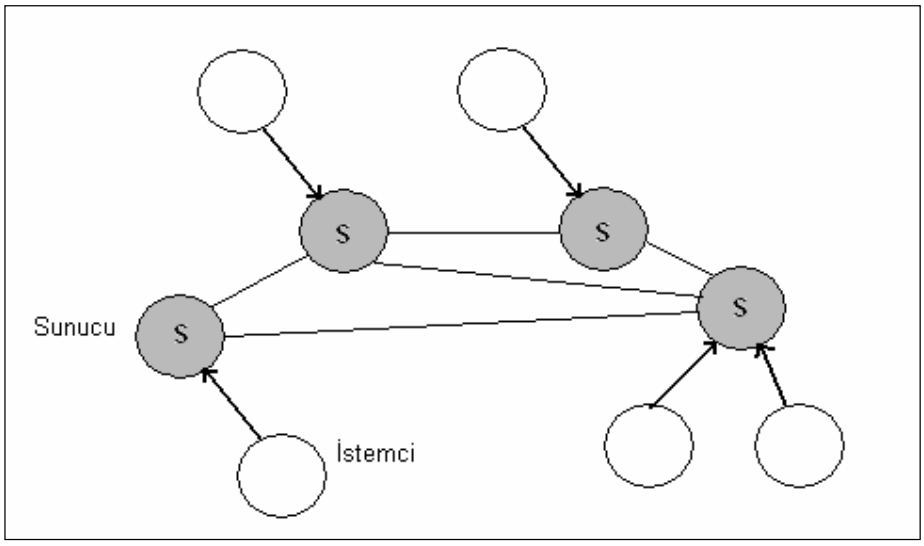
Şekil 2.1 : Hiyerarşik Varlık Yapısı

Çevrimsiz olay servislerinde ara uygulamalar birbirleri ile aynı düzeydedir ve iletişim çift yönlü olarak gerçekleşir. Bu yapıda ara uygulamalar çevrimsiz bir graf oluşturur [7]. Çevrimsiz bir yapıya ilişkin varlık grafı Şekil 2.2'deki gibi gösterilebilir [7].



Şekil 2.2 : Çevrimsiz Varlık Yapısı

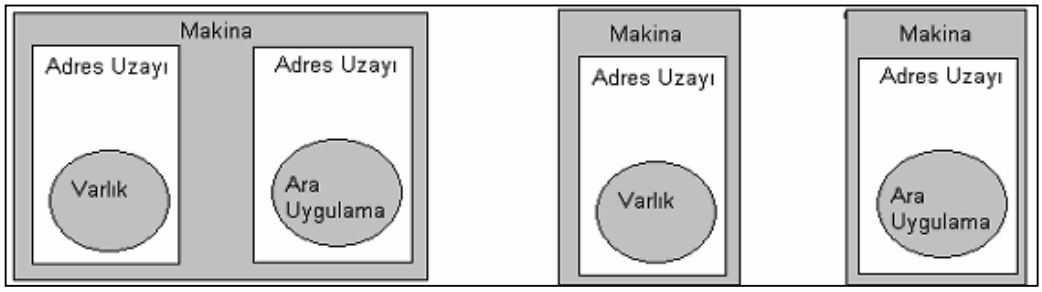
Çevrimsiz olmayan olay servislerinde ara uygulamalar arasında birden fazla bağlantı kurularak iletişim güvenliği sağlanmıştır. Bu ara uygulamalardan birinin çökmesi durumunda sistemin normal işleyişine devam edebilmesi bakımından önemli fayda sağlayacaktır. Muhtemel bir anti çevrimsiz yapıya örnek Şekil 2.3'te verilmiştir [7].



Şekil 2.3 : Anti Çevrimsiz Etmen Yapısı

Hibrit yapılar ise diğer yapıların bir arada kullanılması ile oluşturulacak karmaşık olay servisleridir. SIENA hibrit topolojilere iyi bir örnek sunmaktadır [7].

Ara uygulamaların özelliklerine göre yapılan sınıflandırmalarda ise temel olarak ayrık yada tümleşik olarak iki grup vardır. Ayrık ara uygulama kullanılması durumunda bunun tekil yada çoğul olmasına göre tekrar bölünür. Ayrık ara uygulamalı olay sistemlerinde Şekil 2.4'te görüldüğü üzere, ara uygulama ayrı bir makinede bulunabileceği gibi aynı makinede ama farklı bir adres uzayında da olabilir.



Şekil 2.4 : Ayrık Ara Uygulamalı Olay Sistemleri

Tümleşik ara uygulamalı olay sistemlerinde, bir varlık ile ara uygulama aynı adres uzayını paylaşır. Şekil 2.5'te gösterilen bu yapı nispeten az sayıda varlıktan oluşan küçük ölçekli, grafik arayüzler gibi uygulamalar için kullanılabilir [10].



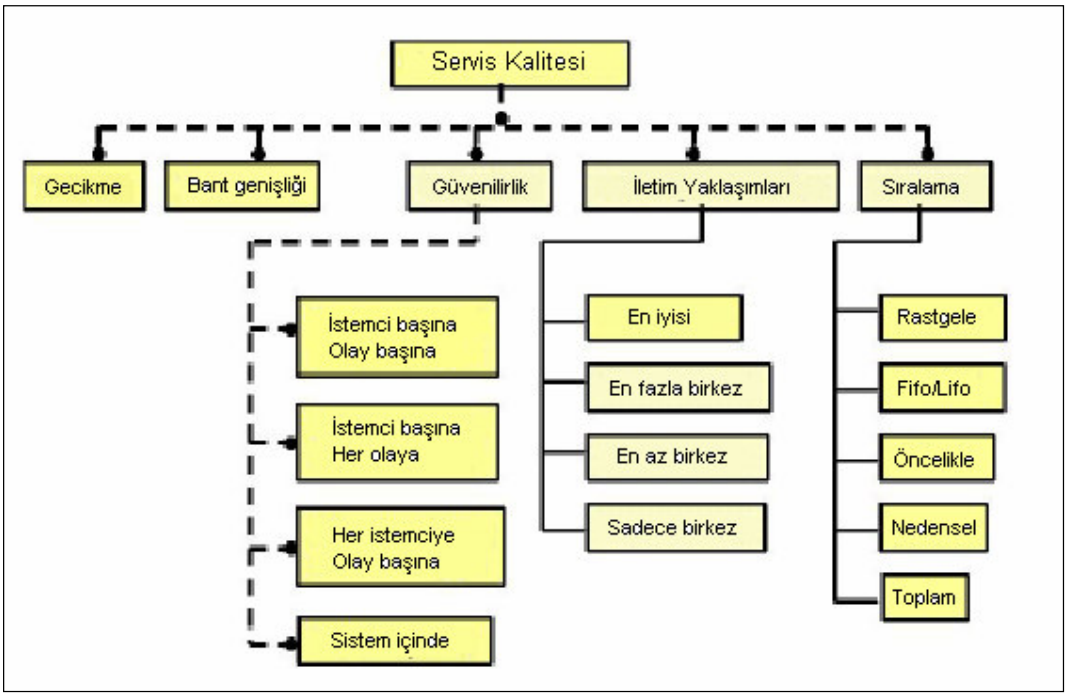
Şekil 2.5 : Tümleşik Ara Uygulamalı Olay Sistemleri

2.3.2 Özellikler

Olay servisleri sağladıkları özelliklere göre çeşitli sınıflandırmalara tabi tutulurlar. Sağladıkları özelliklere göre olay servisleri, fonksiyonel ve fonksiyonel olmayan özellikler olmak üzere ayrılabilir.

Fonksiyonel olmayan özelliklerine göre olay servisleri değerlendirilirken "QoS" yani servis kalitesi çok önemli bir yer tutmaktadır. Bazı "QoS" parametreleri, halihazırda bazı yayınla-abone ol model yada sistemleri tarafından desteklenmektedir [5]. Burada bir iletişim servisi olarak değerlendirildiği için, olay servislerinde de ağ protokollerinde belirtilen servis kalitesi özelliklerine benzer tanımlamalar yapılır. Detay ayrımlar ihtiyaca göre farklı sınıflandırılmakla birlikte önemli unsurlar benzerlik göstermektedir.

İlk olarak iletişimin gerçek zamanlı uygulamalara ne kadar uyacağı değerlendirilir. Burada gerçek zamanlı uygulamaları destekler / desteklemez gibi bir ayrım yapılabileceği gibi matematiksel olarak çeşitli verilerden hesaplanan gecikme süresi yada iletim süresi ve bant genişliği gibi sayısal değerler göz önünde bulundurularak daha detaylı bir ayrıma da gidilebilir. Şekil 2.6'da bunun gibi detaylı unsurlar düşünülerek yapılmış bir sınıflandırmaya ilişkin diyagram görülmektedir [7].

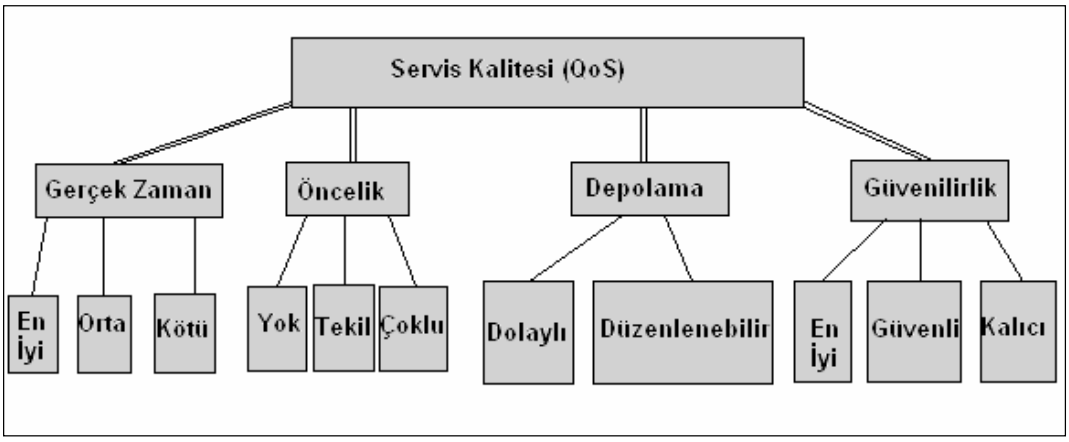


Şekil 2.6 : Servis Kalitesi Sınıflandırma Unsurları

Bunun dışında güvenilirlik ve önceliklendirme de servis kalitesi bakımından ağ sistemlerinde olduğu gibi olay sistemlerinde de servis kalitesi altında incelenen unsurlardır.

Güvenilirlik iletişim sırasında varlıklar arasında gönderilen verilerin iletiminin ne kadar garanti edildiği esasına dayanır. Olay sistemlerinde iletimi garanti edilmesi gereken unsur olay olarak belirlenmişken, etmen sistemlerinde biz bunu mesaj olarak algılayabiliriz. Farklı sınıflamalar esas olsa da güvenli olarak nitelendirilen bir sistemde, bir çökmeden sonraki başlangıçta, üretici ve tüketiciler arasındaki bağlantılar yeniden kayıt olmaya gerek kalmaksızın yeniden kurulurlar ve kalıcı olarak saklanmış olaylar yeniden iletilirler [10].

Önceliklendirme seviyesine göre de olay sistemleri farklı gruplara ayrılır. Önceliklendirme yapılmaması yanı rasgelelik de bir sınıflama olmakla birlikte, her olay bir öncelik verme yada bir olaya farklı durumlara göre birden çok öncelik verme söz konusu olabilir. CORBA bildirim servisi birden çok öncelik sağlar [10]. Şekil 2.7'de önceliklere ilişkin sınıflamayı da içeren bir servis kalitesi detayı gösterilmiştir [10].



Şekil 2.7 : Servis Kalitesi Sınıflandırması

Bunlara ek olarak şekilde olay servisinin çalışırken zaptedeceği hafıza miktarının dinamik yada sabit olmasına veya yeniden düzenlenebilir olmasına göre de bir alt gruplama yer almaktadır.

Servis kalitesi dışında fonksiyonel olmayan özelliklere olay yada mesajların iletim sıralaması, yetkilendirme ve gizlilik gibi güvenlik unsurları da eklenebilir.

Fonksiyonel özelliklere göre sistemi değerlendirirken ise işleyiş sırasındaki davranışlar göz önünde bulundurulur. Olayların iletim şekilleri, desteklenen olay tipleri, olay filtreleme yöntemleri, devingenlik, kayıt şekli gibi özellikler fonksiyonel sınıflamada dikkate alınan unsurlardır.

Temelde iletişim modelleriyle örtüşmesine karşın, olayların iletim şekillerini ayrı olarak düşündüğümüzde istemci yada sunucudan hangisinin tetiklediğinden bağımsız olarak periyodik ve isteğe bağlı olmak üzere iki grup oluşturulabilir. Buna göre olaylar periyodik olarak yada üretici yada tüketiciden birinin isteği doğrultusunda iletilir.

Bir diğer önemli fonksiyonel özellik ise olay filtreleridir. Bunlar hangi olayların hangi varlıklara iletileceğinin belirlenmesinde büyük rol oynar. Olay filtreleri varlıklara yada ara uygulamalar koyulmasına ve filtreleme işleminde kullanılan yönteme alt gruplara ayrılır. Bu gruplar farklı uygulamalarda farklı detaylarda incelenebilir.

Olay servislerinin fonksiyonel özelliklerinden biri de kayıt tipidir. Kayıt olacak varlıklar, bir konu üzerine kayıt olabilecekleri gibi, içerik, olay tipi yada bunların hibriti bir durum için de kayıt olabilirler. Konu üzerine kayıt şeklini benimsemiş sistemlerde, bir üretici bir konuya olay yada mesaj gönderebileceği gibi ara uygulamaya mesajını gönderip konusu da bildirebilir. İçerik üzerine kayıt olunması durumunda ise belirli bir olay içeriğine ilişkin durumlardan haberdar olunmak istenir.

3. JADE

3.1 Genel Yapı

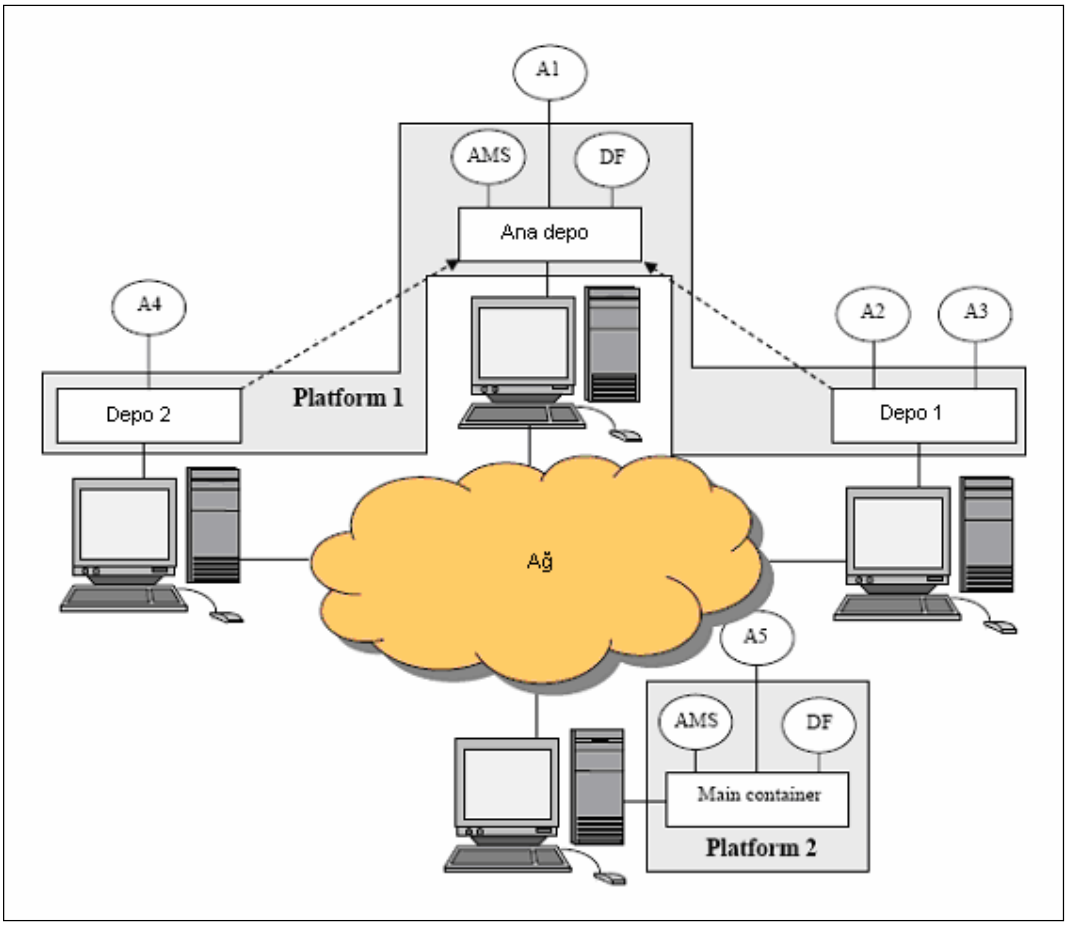
JADE Java programlama dilinde yazılmış ve Java RMI, Java CORBA, Java Serialization ve Java Reflection API gibi gelişmiş özellikleri kullanan açık kaynak kodlu bir etmen geliştirme altyapı yazılımıdır [3]. Temel çalışmaları İtalya Telekom tarafından yürütülmekle birlikte, açık kaynak kodlu olması sebebiyle farklı amaçlar için farklı topluluklar tarafından geliştirilmeye devam etmektedir. Açılımı "Java Agent Based Development Framework" olan yazılım, adından da anlaşılacağı gibi dağıtık sistemlerde etmenlere dayalı bir geliştirme ortamı sağlar.

JADE uygulamasında etmenler, Java appletlerde olduğu gibi doğrudan çalıştırılamazlar [3]. Bunun için özel servisler sağlayan bir yapı kullanılır. Bu yapıyı "Jade.Boot" sınıfı ile elde ederiz. Bu sınıfın içerdiği kod yürütüldüğünde "main container" yani ana etmen havuzu yada deposu denilen yapıyı tetikler. Çalışan her JADE çalışma zamanlı örneği, bir dizi etmen içerebileceği için, bir etmen deposu olarak isimlendirilir [2]. Bu depolar çok-etmenli sistemlerde önemli roller üstlenir. Bu yapıda bir ana depo birbiri ile haberleşen etmenler arasında bir kayıt kütüğü vazifesi görür. Aktif etmen depolarından oluşan yapıya da Platform denir [2].

Ana depo üç adet standart etmen içerir:

- RMA: Görsel arayüzü idare eden "remote management agent"
- AMS: Sistemdeki etmen ve programları takip eden "agent management service"
- DF: Etmenlerin servislerini yayınlayabildiği bir bilgi servisi "directory facility"

Yukarıda bahsedilen depo, ana depo, platform ve standart etmenlerin JADE ile geliştirilmiş bir sistemdeki görünüşü Şekil 3.1'deki gibi gösterilebilir [2]:



Şekil 3.1 : JADE Etmen Sistemi Genel Yapısı

Sistem açılışında bir etmen'i başlatmak için,

#java jade.Boot –gui are:DummyAgent

ifadesi kullanılır. Burada son ifade <Etmen ismi>:<Etmen sınıf ismi> şeklinde yorumlanır. Bir etmeni ortama sonradan eklemek için,

#java jade.Boot –container are2:DummyAgent

ifadesi kullanılır.

Her etmen bir ipliğe atandığı için birbirinden bağımsız ve paralel olarak işlem yapabilirler. Aynı anda çalışan ve birbiri ile haberleşen çok sayıda etmen olabileceği için bunlar arasında kontrolü sağlayan paralel iplikler gerçekleme ihtiyacı doğmaktadır. Buradaki problem, her java ipliğinin bir işletim sistemi ipliğine karşılık gelmesi dolayısıyla, iplikler arası geçişin pek de verimli yapılamamasıdır. Bu

probleme çözüm olarak JADE içinde "behaviour" yani davranış denilen kavram oluşturulmuştur.

Bir davranış bir olaya nasıl tepki verileceğini yada bir durum değişikliğinin nasıl karşılanacağını belirleyen bir "event handler" yani olay yöneticisidir. Daha açık olarak JADE'de davranışlar bir mesaj geldiğinde ne yapılacağını belirler. Davranışlar birer sınıf olarak gerçeklenir ve olay yöneticisi bu sınıflar içindeki "action" ismi verilen metotlardır.

Etmenler gerçekleşen olaylardan sonra bu metotları birbiri ardına çalıştırırlar. Bir etmenin gerçekte yapması için yaratıldığı işler "davranış" denilen yapılar ile yürütülür [2]. Aynen grafik arayüzündeki dinleyiciler gibi etmenlere ilişkin tüm işleyiş durdurulmadan davranışların durdurulması mümkün olmaz. Bir etmen, birçok davranış yürütebilmesine rağmen bunları paralel olarak yürütmez. Paralel yürütülmeyen bu akışlar arasındaki geçişler de yazılımın geliştirilmesi sırasında dikkat edilmesi gereken unsurlardır. Dikkat gerektirmesinin yanında yazılımcıya bırakılan bu düzenleme sayesinde bazı avantajlar elde edilir:

- Her etmenin bir java ipliğine sahip olması sağlanır (bu özellikle cep telefonu gibi sınırlı kaynak bulunan ortamlarda çok önemlidir) [2].

- Davranışlar arası geçiş, java iplikleri arasında geçişten çok daha hızlı olduğu için daha iyi performans sağlar [2].

- Aynı kaynaklara erişen eşzamanlı davranışlar arasındaki senkronizasyon sorunlarını, bütün davranışlar aynı java ipliği tarafından yürütüldüğü için, ortadan kaldırır (bu aynı zamanda performans anlamında bir hızlanmayı da sağlar) [2].

- Davranışlar arası geçiş olduğunda, etmen herhangi bir yığın bilgisi içermez, dolayısıyla o anlık görüntünün alınması imkanı bulunmaktadır. Bu durum önemli gelişmiş özelliklerin, örneğin etmenin o anki görüntüsünü kalıcı bir saklama ortamına kaydetmek (etmen kalıcılığı) yada uzaktan çalıştırma için başka bir depoya aktarmak (etmen devingenliği), gerçeklenmesini olası kılar [2].

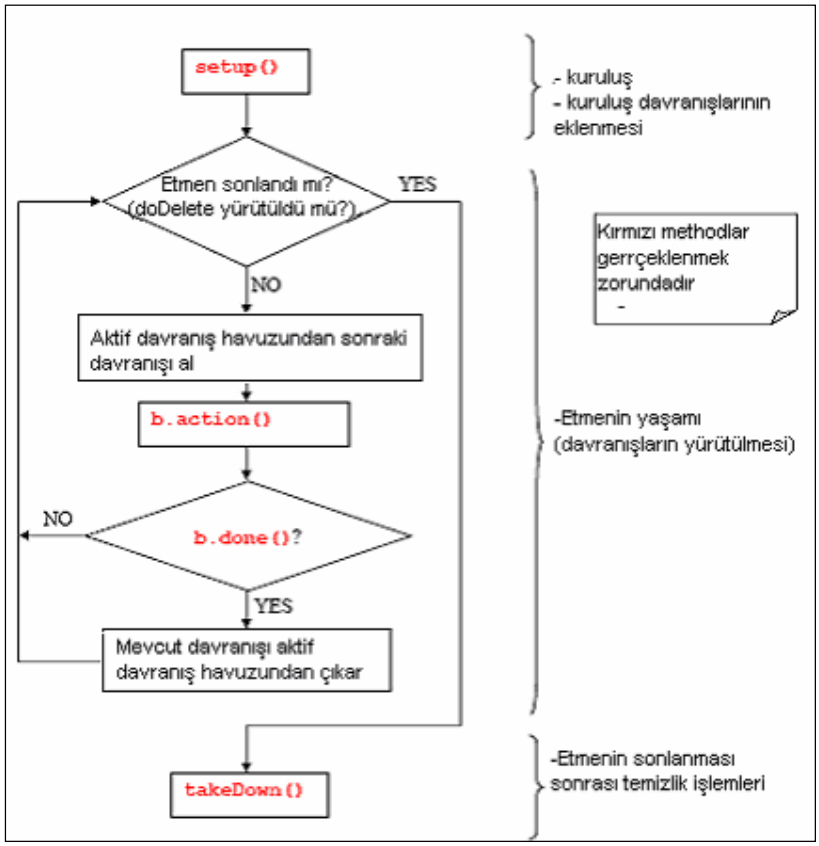
Her davranış tek bir anlık aktiflik fazına karşılık gelir. Bu sebepten dolayı, etmenler arası iletişimde mümkün olduğunca çok davranış olmalı ve doğru sırada tetiklenmeli ve yürütülmelidir. Davranışlar "block(...msec)" adında bir metot yürütürler. Bu ilgili davranış akışının bir sonraki çalışmasına kadar parametre olarak aldığı süre kadar beklemesini sağlar. Bu metot yada "sleep(t)" metodu farklı "action" lar arasında ara vermek için uygun değildir. Bu yüzden davranışları tek bir atomik yapıda gerçeklemek en uygun çözümdür.

Davranışlar ihtiyaca göre çeşitli tiplerde gerçeklenebilecek olmakla beraber JADE bazı temel tipleri kullanılmak üzere içinde barındırmaktadır. Eğer bir etmen bir işi sadece bir defa yapacak ise bu durumda "tek-atış" (one-shot) davranışı kullanılabilir. Eğer yazılımcı bir etmenin bir işi tekrarlı olarak gerçekleştirmesini istiyorsa bu durumda "döngüsel" (cyclic) davranış kullanılabilir. Bir etmenin önceden tanımlı bir karar mekanizması uyarınca belirlenen davranışları gerçeklemesi istenirse, "genel" (generic) davranışın kullanılması faydalı olacaktır. Bu temel davranış tiplerinin dışında ihtiyaçlar gözetilerek bunların karışımından oluşan yeni tipler de kullanılabilir. Yine bazı tanımlı gelişmiş davranış tipleri "ardışıl davranış" (sequential behaviour), "paralel davranış" (parallel behaviour), "Sonlu Durum Makinesi davranışı" (FSM behaviour) olarak gerçeklenmiştir.

Davranışlar kendilerinden önce gelen davranışlar tarafından başlatılabilirler.

```
class Act1 extends SimpleBehaviour{
        [...]
        public void action() {
                [...]
                addBehaviour(new Act2());
                [...]
        }
}
```

Etmen sınıfları aynı zamanda “doDelete()” adında bir metot yürüterek kendisinin tüm davranışlarını sonlandırıp sistemden çıkarılmasını sağlar. Bir etmenin hayat döngüsü Şekil 3.2’deki gibi ifade edilebilir [2]:



Şekil 3.2 : Etmen Hayat Döngüsü

3.2 Etmenler Arası Haberleşme

JADE esnek, etkin ve karmaşık özellikleri olan bir etmenler arası haberleşme platformu sunar. Altyapı yazılımının önceki versiyonlarında etmenler birbirlerine mesajları eşdüzeyde (peer-to-peer) göndermek durumundaydı. Bu sebeple etmenlerin mesajlarını gönderecekleri diğer etmenlerin isimlerini bilmesi gibi bir zorunluluk vardı. TILAB (Telecom Italia Laboratory) tarafından çıkarılan son versiyonda bu problem, haberleşme sistemlerinde “publish-subscribe” yani “yayın-kayıt” yada konu odaklı haberleşme olarak bilinen kavram kullanılarak çözülmüştür.

Kullanıcı tarafında farklı davranış sergilemelerine rağmen, konu odaklı haberleşme ve eşdüzeyde haberleşme aynı mesaj gönderim altyapısını kullanır. Mesaj yapısında

FIPA (Foundation for Intelligent Physical Agent) standartları uygulanmıştır. FIPA etmen standartlarının geliştirilmesinde çok önemli bir rol oynamıştır.[4] JADE mesaj iletim yapısı tam bir FIPA modeli uyarınca gerçeklenmiş olup, mesaj tanımlama, protokol, şifrelemeler ve mesaj iletimi bölümlerinden oluşmaktadır [3]. Mesaj formatı olarak JADE ACL (Agent Communication Language) kullanılmış olup şu temel bölümleri içermektedir:

- Edim: FIPA mesaj tipi (INFORM, QUERY, PROPOSE...)
- Gönderici: Mesajı gönderen etmen belirteci
- Alıcı: Alıcı etmen belirteci
- İçerik: Mesajın temel içeriği (yük)
- İletişimID: Bir iletişimde mesaj bağlantısını sağlamak için kullanılan benzersiz belirteç.

Bir JADE platformunda, bir etmenin mesaj gönderebilmesi için aşağıdaki kodu yürütmesi gerekecektir:

```
ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
//varsayılan bir bilgilendirme mesajı oluşturuldu
// içeriği ve alıcısı boş olduğu için doldurulmalı
msg.setContent("Deneme mesajı.");
msg.addReceiver("hedef_etmen_belirteci");
//sisteme gönderilmeye hazır bir mesaj oluştu
send(msg);
```

Gönderilen mesajları almak için beklemeli yada kuyruk kullanımlı yöntemler kullanılabilir:

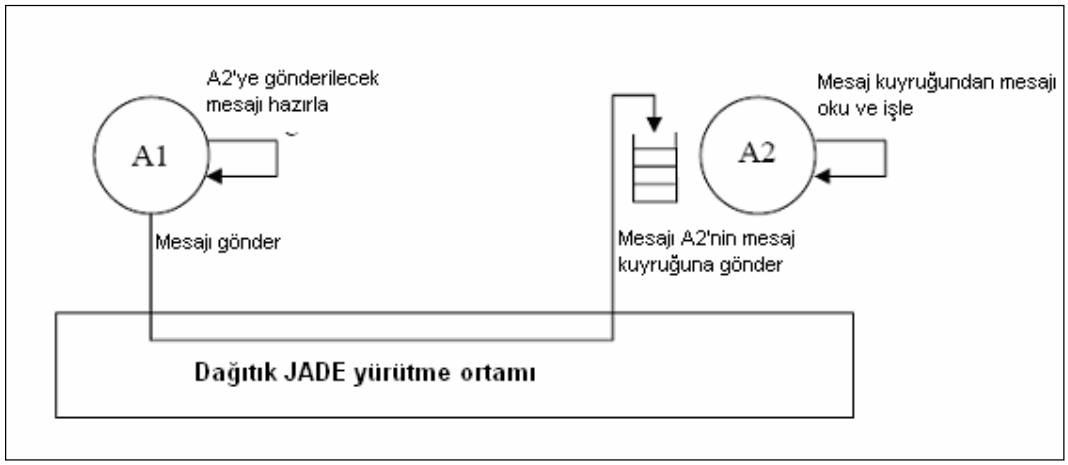
```
//mesaj geldiğinde tüm aktivite bekletilecek
```

```
ACLMessage msg = blockingReceive();

//diğer bir seçenek olarak kuyruk okunabilir

ACLMessage msg = receive();
```

Yukarıdan anlaşıldığı gibi, JADE zaman uyumsuz bir iletişim ortamı sağlar. Zaman uyumsuz iletişimde gönderilmiş ama alınmamış mesajları ise etmen depolarının kontrol ettiği mesaj kuyruklarını kullanarak saklar. Şekil 3.3'te bu zaman uyumsuz iletişim gösterilmektedir [2].



Şekil 3.3 : JADE'de Asenkron Etmen İletişimi

Gönderilmiş bir mesajı cevaplamak için üç olası yöntem bulunmaktadır. İlk yöntem, alıcı tarafından mesajı gönderen etmenin belirtecini öğrenmek için "ACLMessage" sınıfı tarafından tanımlanmış metot "msg.getSender()" şeklinde yürütülür. İkinci yöntem, haberleşen iki etmenin birbirini tanıdığı durumda geçerli olup, ilgili etmenin ismi kullanılarak "new AID(receiver_name)" kodu yürütülür ve elde edilen etmen belirteci cevap mesajına alıcı belirteci olarak eklenir. Üçüncü yöntem ise AMS (Agent Management System) sınıfları tarafından sağlanan sistemdeki etmenlerin listesini veren metotlar kullanılarak, elde edilen liste üzerinde arama ile mesajı gönderene ulaşmaktır.

Etmen belirteçleri "src.jade.core" paketi içindeki "AID.java" sınıfıyla tanımlanmıştır. Etmen belirteçleri için ayrı bir sınıf yapılmasının nedeni, bu belirteçlerin etmen iletişiminde önemli bir rolü olmasından kaynaklanmaktadır. Bir etmen belirtecini oluşturan iki bölüm, "etmen_ismi@platform_ismi" olarak ifade edilir. İlk bölüm

sistem yönetici tarafından ilgili etmen sisteme eklenirken verilen isimdir. İkinci bölüm ise sistem tarafından otomatik olarak eklenir. Etmen ismine eklenen bu uzantı, bir mesaj gönderildiği durumda iletişim servisleri tarafından ilgili etmenin yerini belirlemekte kullanılır. Etmenin yerine göre de iletiyi ulaştırmak için en iyi yöntem belirlenerek, mesaj iletiminin en iyilenmesi sağlanır.

3.2.1 Konu odaklı iletişim

Konu odaklı iletişimin temeli daha önce eşdüzey iletişim esas alınarak anlatılan yapı ile alıcının bir konu olması dışında aynıdır. Konu da aslında kavramsal olarak bir farklılık yaratmakla birlikte aslında bir etmen belirtecidir. Ancak bu belirteci oluşturmak için yazılımcıların "TopicManagementHelper" sınıfı içindeki "createTopic(Konu_ismi)" metodunu kullanması gerekir. Konu odaklı davranış, eşdüzeyli iletişimden farklı davranış şekilleri sergilediği için JADE aslında bir konu olan etmen belirteçlerini ayırt etmek için basit bir işaretleme yöntemi kullanır. Konu yaratmak için kullanılan createTopic(Konu_ismi)" metodunda verilen parametrenin sonuna "TOPIC_" eki eklenir. Bu ek daha sonra yine "TopicManagementHelper" sınıfı içinde tanımlı "isTopic()" metodu tarafından alıcı etmen belirtecinin bir konu olup olmadığının anlaşılmasında kullanılacaktır. Alıcı tarafında da durum pek farklı değildir. Mesaj alma yöntemleri aynı olmakla birlikte, bir etmenin bir konuda mesaj alabilmesi için önce kayıt olması gerekmektedir.

Kayıt tutmak, konu odaklı iletişimi yöneten servislerin sistemde hangi etmenin hangi konular ile ilgilendiğini ayırt edebilmesi için gereklidir. Sistem bu kayıt bilgileri etmen-konu çiftleri şeklinde saklar. Kayıt olan her etmenin saklandığı bu tablo, bir etmen tarafından bir konu üzerine bir mesaj gönderildiğinde, konu odaklı iletişim yönetim servisleri tarafından taranarak ilgilenen etmenler tespit edilir ve bu etmenlere ilgili mesajın kopyaları gönderilir.

3.3 Olay Sistemi Olarak JADE

Olay sistemlerinin incelendiği bölümde etmen sistemlerinin ve olay sistemlerinin benzerliklerine dikkat çekilmişti. Etmen sistemlerinin kategorize edilmesinde olay sistemlerinin sınıflandırma kriterleri bu benzerlik sebebiyle büyük oranda

kullanılabilir olacaktır. JADE'in sunduğu etmen sistemini bu gözle incelersek, olay sistemi olarak bir ifadeye erişebiliriz.

3.3.1 JADE iletişim modeli

İletişim modeline göre JADE'i tanımlamak istersek, son versiyonla birlikte sistemde eşdüzeyde iletişimin yanısıra dolaylı iletişiminde mümkün olduğu görülmektedir. JADE ile etmenleri birbirini tanıyan ve doğrudan haberleşen bir yapı kurulabileceği gibi bununla beraber yada ayrı olarak konu tabanlı bir iletişim de sağlanabilir.

3.3.2 JADE olay servisi

Olay servisi olarak baktığımızda varlıklara etmenler ve olaylara da etmenlerin gönderdiği ve aldığı mesajlar karşı düşmektedir. Organizasyon yada topoloji olarak JADE etmen sistemini kategorize etmemiz gerekirse dağıtık tümleşik bir yapı ile karşılaşmaktayız. Bu dağıtık yapıda mesajlaşma servislerini ara uygulamalar olarak değerlendirirsek elimize çevrimsiz olmayan bir graf geçmektedir.

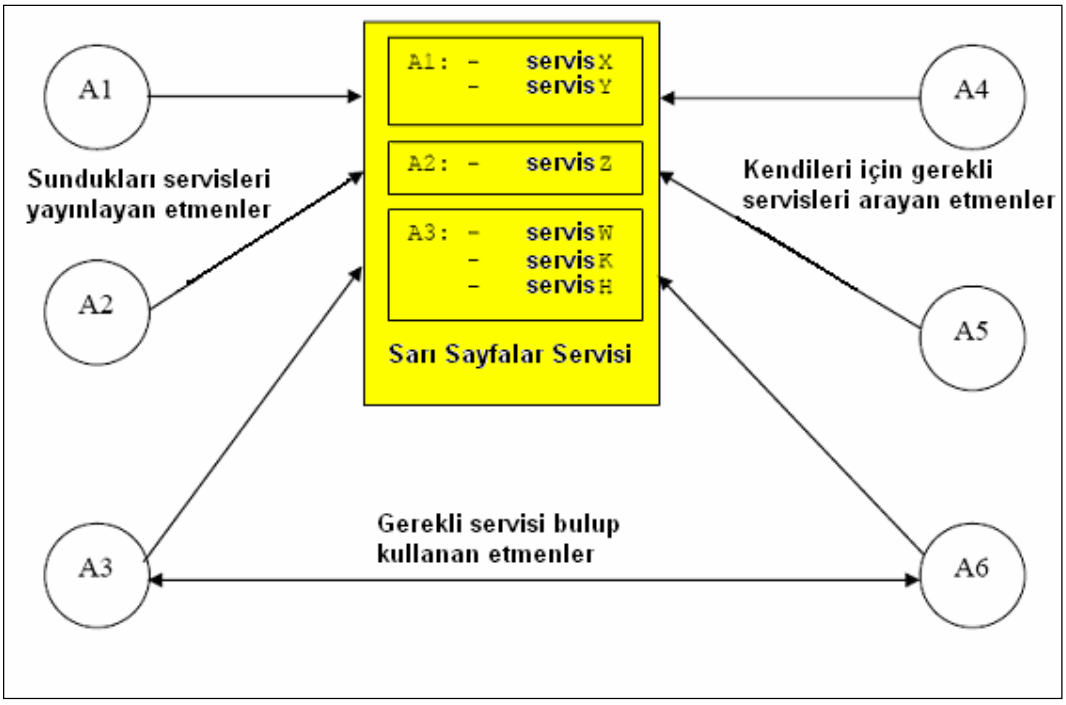
Olay servisinin özelliklerini inceleyecek olursak aslında sistemsel olarak bir çok özelliğin desteklendiğini görüyoruz. Fonksiyonel olmayan özellikleri incelersek, servis kalitesi olarak gerçek zaman desteğinin fiziksel yapı elverdiği durumlarda mümkün olduğunca yüksek seviyede desteklenebilecek olmakla birlikte genellikle orta seviyede olacağı düşünülebilir. Mesajlar için önceliklendirme sadece etmen depoları tarafından ilgili etmenin kuyruğu yönetilirken söz konusu olmakta ve bu da kuyruk mantığı uyarınca FIFO esasına dayanmaktadır. Mesaj iletiminin güvenilirliği yine ek uygulamalar ile sağlanabilmektedir Bu kapsamda bir çökme durumunda dahi mesaj iletimine olanak sağlayan kalıcı yapıya erişmek dokümanlardan aşılabildiği üzere mümkündür.

Fonksiyonel özellikleri incelediğimizde ise varsayılan mesaj tipleri bulunduğunu, mesaj filtreleme işleminin mesajlaşma servislerinde yapıldığını, etmenlerin işlerini davranış denen yapılar üzerinden sürdürmesi sebebiyle devingenlik sağlanıp, bir etmenin bir durumda dondurularak başka bir depoya taşındıktan sonra tekrar kaldığı yerden çalışmasının sağlanabileceği görülmektedir.

3.4 Diğer JADE Özellikleri

Konu odaklı yada eşdüzeyli iletişimin temel unsurları dışında JADE tarafından yazılımcı tarafından kullanılmak üzere yada yazılımcının bilmesi gerekmeyen ama işini kolaylaştıran bazı özellikler bulunmaktadır.

Bu ek özelliklerden biri "sarı sayfalar" servisidir. Ana etmen deposunda bulunan temel etmen haberleşmesine ek özellikler sağlayan ve konu odaklı iletişim gibi kayıt gerektiren bu yapı DF (Directory Facility) tarafından sağlanır. Bu yapı özellikle etmenlerin birbirleri hakkında pek bilgi sahibi olmadıkları ortamlarda iletişime önemli katkılar sağlar. Bir "sarı sayfalar" servisi etmenlerin sunduğu bir yada daha fazla servisi yayınlamasına ve diğer etmenlerin bu servisleri birbiri ardına kendi amaçları doğrultusunda kullanmasına olanak sağlar [2]. Bu yapı Şekil 3.4'te ifade edilmiştir [2].



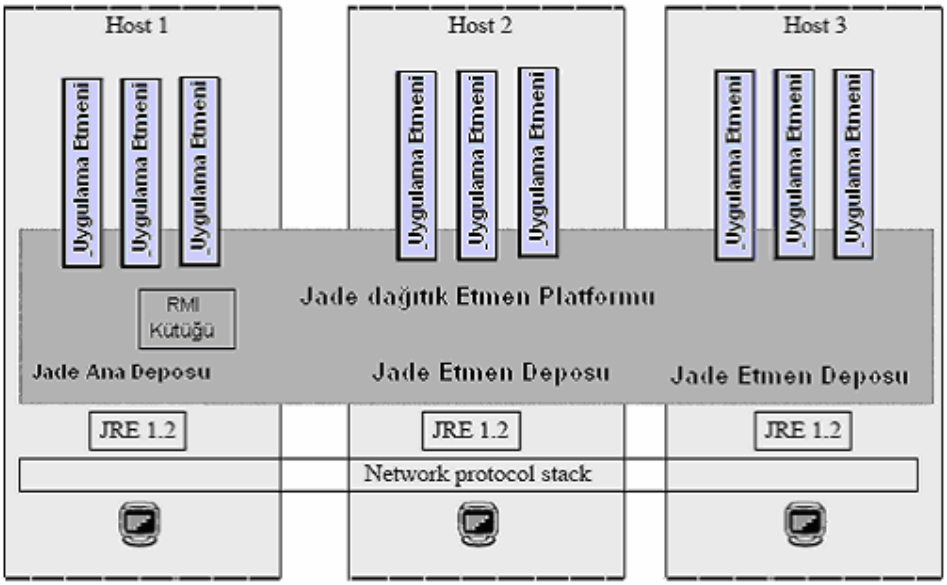
Şekil 3.4 : JADE Sarı Sayfalar Servisi

JADE'in sağladığı özelliklerden biri de, etmen belirtecindeki bilgiyi kullanarak belirlenen etmenin yerine göre mesajı göndermenin en iyi yolunu tespit etmesidir. Eğer haberleşen iki etmen aynı depo içindeyse, gönderilen mesaj herhangi bir dönüşüme tabi tutulmadan doğrudan bir nesne gibi gönderilecektir. Eğer haberleşen

etmenler aynı platformda ama farklı depolarda ise, gönderilen mesaj Java'nın sağladığı RMI (Remote Method Invocation) özelliği ile aynı depodaymış gibi iletilecektir. Eğer etmenler farklı platformlarda ise, gönderilen mesaj IIOP (Internet Inter-Orb Protocol) protokolü uyarınca bir karakter katarına dönüştürülerek gönderilecek ve alıcı platform tarafında da tekrar dönüştürülerek mesaj elde edilecektir.

Bir altyapı yazılımı olan JADE kullanılarak geliştirilen ortamlarda, etmen iletişimi birbirine Java RMI ile bağlanmış birden fazla JVM (Java Virtual Machine) üzerinden sağlanır. Ancak bu durum sadece dağıtık sistemlerde çalışılmasının getirdiği bir durum olup aslında bir zorunluluk değildir. Tek bir JVM kullanarak da bir sistem oluşturulabilir ve burada iletişim olay imleri ile gerçekleşir. Buradan da anlaşıldığı gibi JVM'ler çoklu etmenlerin paralel olarak çalışmasına imkan veren bir ortam sağladıkları için yapının önemli bir bölümünü oluştururlar.

JADE aynı zamanda birden fazla JVM'in aynı makine üzerinde kullanılmasına da olanak sağlar ancak bu yöntem pek tercih edilmez. Her makinede bir JVM ve biri etmenlerin yönetimini ve platform için de bir geçit sağlayan ana depo olan etmen depoları bulunmalıdır. Bu sistemsel bir zorunluluk değil, uygulama içerisinde bir de depoların kontrolü için bir ek yapı oluşturulmasına gerek kalmaması için genel olarak benimsenen kabuldür. Dağıtık bir sistemdeki JADE etmen sistemi genel olarak Şekil 3.5'teki gibi gösterilebilir [1].



Şekil 3.5 : Dağıtık JADE Etmen Sistemi Yapısı

Ana depolarında dışında sistemde bulunan etmen depolarının her biri aslında bir RMI sunucu nesnesidir. Bu depolar, etmenlerin yaratılmasını, etmenlerin askıya alınmasını, etmenlerin uyandırılmasını ve etmenlerin sonlandırılması işlemlerini kontrol ederler. Bunun dışında bu depolar, gelen mesajdaki alıcı bilgisine göre ilgili etmene ait mesaj kuyruğuna mesajı kaydederek, mesaj kuyruklarının idaresini sağlarlar. Gönderilen mesajlar için de alıcı etmenin yerini belirlerler ve en iyi iletim yöntemini tespit ederler.

4. YAYINLA ABONE OL TEMELLİ İLETİŞİM

4.1 Tanım

Yayınla-abone ol, dağıtık sistemlerin bağımsızlaşmış doğasına uyan asenkron mesajlaşma için çok yaygın bir iletişim örneğidir [6]. Yayınla-abone ol temelli iletişimde temel unsurlar varlıkların belirli durumlardan haberdar olmak için kullandıkları kayıt kütüğü, kayıt kütüğü üzerinde işlem yapan ve mesajların ilgili alıcılara iletilmesini sağlayan olay bildirim servisi, mesajları üreten yayıncı varlıklar, mesajları alan abone varlıklar ve gönderilen mesajların ilgilenen alıcılara ulaştırılması için kullanılan ve aslında bir nevi mesaj gurubu işlevi gören konular yada olay şablonlarıdır. Büyük ölçekli uygulamalardaki dağıtık etkileşimin gevşek bağlı doğasına iyi adapte olmuş yayınla/abone ol iletişim paradigması, şu sıralarda artarak dikkatleri üzerine çekmektedir [11]. Bunun nedeni özellikle büyük ölçekli uygulamalara önemli esneklikler getirmesidir. Normalde kullanılan eşdüzeyde iletişime dayalı istemci sunucu mimarilerindeki sorunları belirlersek, yayınla/abone ol temelli iletişimin getirdiği faydaları daha açık görebiliriz, bu sorunlar:

- Etkileşim iki varlıkla sınırlıdır (birebir) [9]
- Her varlık iletişimde partnerini nasıl adresleyeceğini bilmek zorundadır [9]
- İletişim kurabilmeleri için iki varlık da aynı anda kullanılabilir olmalıdır [9]
- İletişim doğası gereği senkrondur [9]
- İletişim istek temellidir [9]

şeklinde sıralanabilir.

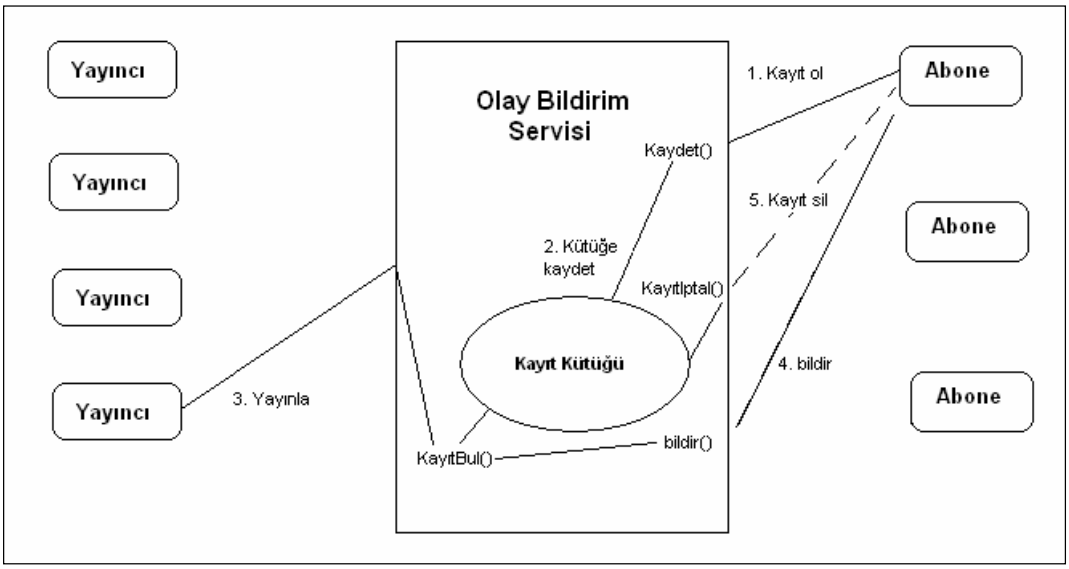
4.2 İletişim Akışı

Yayınla/Abone ol temelli iletişimde işleyişin bir yayıncı bir de abone tarafı vardır. Olay bildirim servisi ise bu iki tarafın tam ortasında bulunur ve entegrasyonu sağlar.

Abone tarafında süreç ilgili varlığın olay bildirim servisin kayıt olma işlevini kullanarak, haberdar olmak istediği bir olaya yada mesaj konusuna abone olmasıyla başlar. Bu aşamadan sonra abone olunan durumla ilgili bir olay gerçekleştiğinde yada bir mesaj gönderildiğinde ilgili aboneye durum olay bildirim servisi tarafından bildirilir. Bu süreçten çıkmak için ise yine olay bildirim servisinin kayıt iptali işlevini kullanarak aboneliğini iptal eder ve bu noktadan itibaren ilgili olay yada mesajlar bu varlığa iletilmez.

Yayıncı tarafında ise süreç aboneden farklı olarak ilgili varlık için atanmış göreve bağlıdır. Yayıncı varlık tarafından yürütülen kod eğer sadece bir konu hakkında mesaj üretmek üzerine ise, süreç varlığın sisteme dahil olması ile başlar ve sistemden çıkması ile sonlanır. Yayıncı varlık tarafında mesaj yada olay oluşumları dinamik olarak değişkenlik gösteriyor ise bu durumda süreç sadece ilgi olay yada mesajın hazırlanması ve gönderilmesi sırasında işlemiş olacaktır.

Olay bildirim servisi yayıncılardan gelen mesajları yada olayları aldıktan sonra kayıt kütüğü aracılığı ile bu mesaj ile ilgilenen aboneleri tespit eder. Kendisine gelen mesajın bir kopyalarını oluşturarak ilgilenen tüm abonelere gönderir. Anlatılan akış Şekil 4.1'de gösterilmiştir.



Şekil 4.1 : Yayınla-Abone Ol İletişim Akışı

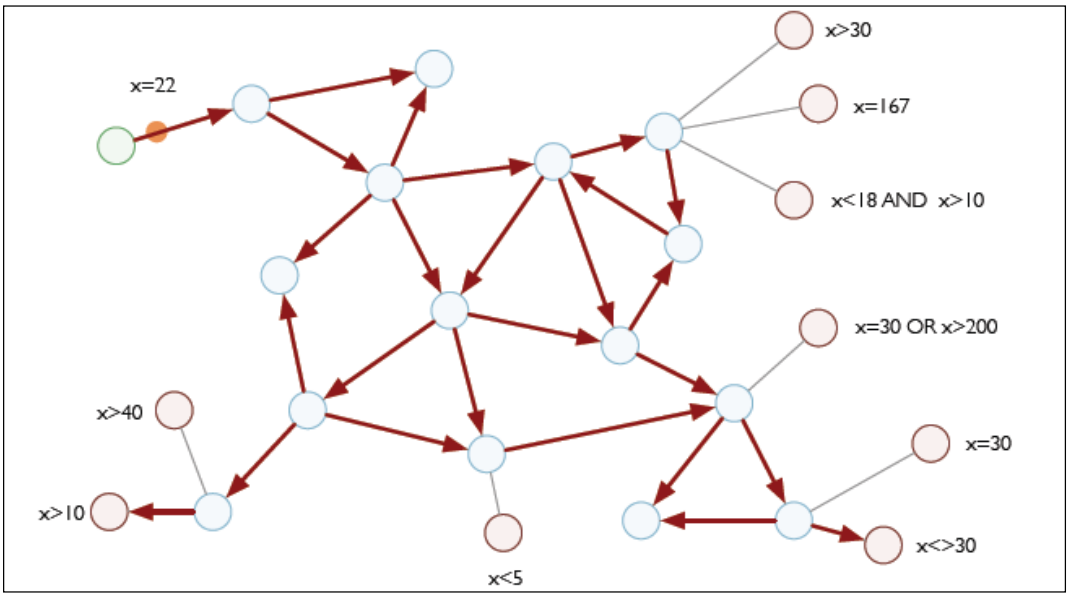
4.3 Olay Bildirim Servisi ve Yönlendirmeler

Olay bildirim servisi bir mesaj geldiğinde, ilgilenen aboneleri bulmak amacıyla kayıt sırasında belirlenmiş kısıtları kullanır. Kısıtlar iletişimin karmaşıklığına göre oluşacak olay yada gönderilecek mesaj için belirlenen parametrelerin detay değerleri içerebileceği gibi sadece beklenen olay yada mesajın konusunu da belirtebilir. Kısıtlara uyan veriler olay bildirim servisi tarafından çoklanarak ilgili varlıklara iletilir.

Olay bildirim servisinin gerçeklenmesi dağıtık olarak yapıldı ise bu durumda iletişim akışında bir de olay bildirim servisi içindeki yönlendirme işlemleri ve algoritmaları devreye girer. Bu kapsamda olay taşması, kayıt taşması, filtrelemeli yönlendirme yöntemleri kullanılabilir.

4.3.1 Olay taşması yöntemi

Olay taşması yönteminde olay bildirim servisinin her elemanı kendisine gelen iletiyi bağlı olduğu düğümlere iletir. Bu yöntemde gerçekleme çok açık olmasına rağmen maliyeti çok yüksektir [9]. Akış Şekil 4.2'de gösterilmiştir [9].



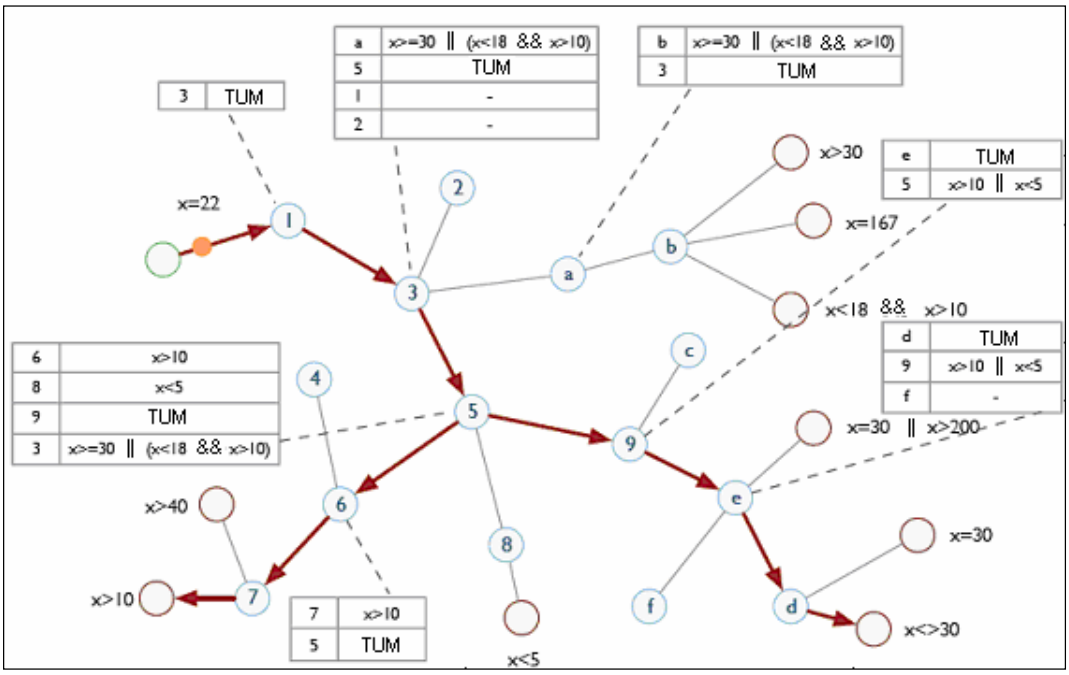
Şekil 4.2 : Olay Taşmalı İletişim Akışı

4.3.2 Kayıt taşması yöntemi

Kayıt taşması yönteminde dağıtık yapıdaki olay bildirim servisi içindeki iletişim minimize edilmeye çalışılmıştır. Bu yapıda her bir düğüm kendisine gelen mesajı abonelerden hangisine ileteceği bilir. Sürekli değişen ortamlarda sistemin sürekli bilgilendirilmek zorunda kalacak olması bu yöntemin kötü tarafıdır.

4.3.3 Filtrelemeli yönledirme yöntemi

Filtrelemeli yönlendirme yönteminde ağ sistemlerine benzer bir yönlendirme mekanizması oluşturulmuştur. Buna göre her düğüm sadece komşu düğümlerinin bilgisine sahiptir ve kendisine bir ileti geldiği durumda bunun bildirimini komşularından uygun olana yapar. Bu yöntemin problemi ise her yeni kayıt işleminde sistemin akışları tekrar oluşturmak üzere düğümler arasında iletişime ihtiyaç duymasıdır. Örnek bir akış Şekil 4.3'te verilmiştir [9].



Şekil 4.3 : Filtrelemeli İletişim Akışı

4.4 Kayıt Modelleri

Yayınla/abone ol esasına dayalı çalışan sistemleri oluşturulurken dikkat edilen bir unsur da kayıt modelleridir. Kayıt modelleri yapılacak filtrelemenin detayını, sistem içindeki iletilerin gruplanma derecesini ve verileri tutacak kayıt kütüğünün yapısını belirlerler. Bir modelin anlam gücü, modeli kullanan abonelerin bekledikleri mesajlardan yada olaylardan ne kadar sapma ile haberdar oldukları ile belirlenir. Bir kayıt dilinin anlam gücü basitçe istemcilerle olan iletişimin esnekliği ile ilintili olmayıp tüm bildirim servisinin gerçeklenmesinde güçlü bir etkisi vardır [8]. Bu kriterlere göre konu tabanlı, hiyararşi tabanlı, içerik tabanlı, tip tabanlı, xml tabanlı gibi farklı kayıt türleri içeren, farklı anlam gücünde yapılar oluşturulabilir. Ancak yaygın kullanım düşünülerek, yayınla-abone ol iletişim sistemleri konu tabanlı ve içerik tabanlı olmak üzere iki ana sınıfa ayrılabilir [6].

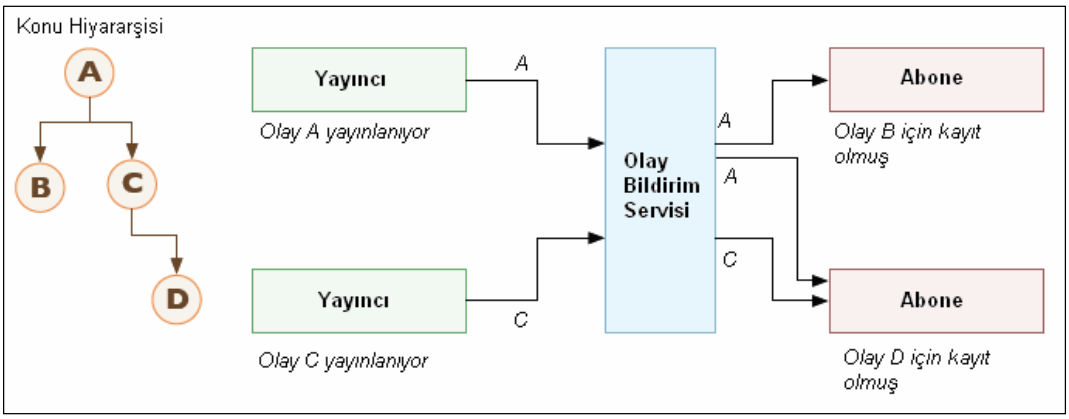
4.4.1 Konu tabanlı kayıt modeli

Konu tabanlı kayıt modellerinde elektronik posta iletişiminde olduğu gibi gönderilen mesajlara yada bildirilen olaylara gönderi içeriğini özetleyen konu bilgisi eklenir. Her konu, her bir olası yayıncıyı tüm ilgilenen abonelere bağlayan mantıksal bir olay

kanalına karşı gelir [8]. Konu bilgileri sistem içinde gereksiz veri transferini engellemek ve varlıklar ilgilenmedikleri olay yada mesajlar ile meşgul etmemek için kullanılır. Sistemde yayıncılar ve aboneler konular hakkında bilgi sahibidir. Bir yayıncı bir ileti oluşturduğu zaman bununla birlikte bu iletinin konusunu da yayınlar. İleti, bildirim servisi tarafından alınarak kayıt kütüğü üzerinde arama yapar ve konu ile ilgilenen varlıklar belirler. Bildirim servisi tarafından belirlenen varlıklara gönderi iletilir.

Konu tabanlı kayıt modelinde aboneler, bildirim servisine kendi kimlik bilgileri ve ilgilendikleri konu ile birlikte kayıt olmalıdır. Bildirim servisi, kayıt işlemini gerçekleştirebilmek için aboneler tarafından tetiklenebilecek kayıt kabul metotları sunar. Kayıt kabul metodu ile abone tarafından belirtilen bilgiler kayıt kütüğüne kaydedilir.

Konu tabanlı kayıt modelleri farklı şekillerde uygulanarak, aynı mantıkta çalışan ancak farklı davranan yapılar oluşturulabilir. Örneğin konular arasında bir hiyerarşik bağlantı tanımlanarak hiyerarşi tabanlı bir kayıt modeli elde edilebilir. Bu yapıda bir abone bir konuya kayıt olduğu zaman, o konuda ve o konunun tanımladığı alt ağaçta yer alan tüm konularda yayınlanan olayları alıyor olacaktır [9]. Şekil 4.4'te verilen bir konu hiyerarşisine göre sistem davranışı gösterilmiştir. Buna göre konu A ile ilgili bir olay yayınlandığında, hiyerarşide bunun altında yer olan B ve D konularına kayıt olmuş abonelere de bu olay iletilmiştir. Aynı durum C konusu için yayınlanan olayın bunun alt ağacında yer alan D konusuna kayıt oluş aboneye iletilmesinde de tekrarlanmıştır [9].



Şekil 4.4 : Hiyerarşik Konu Tabanlı İletişim Modeli

Bu yapı bir elektronik posta uygulaması olan Outlook'taki grup tanımına benzemektedir. Şirketin hiyerarşik yapısına uygun tanımlanan gruplar sayesinde bir gruba mesaj atıldığında bunun alt gruplarında kişiler de mesajı alacaktır.

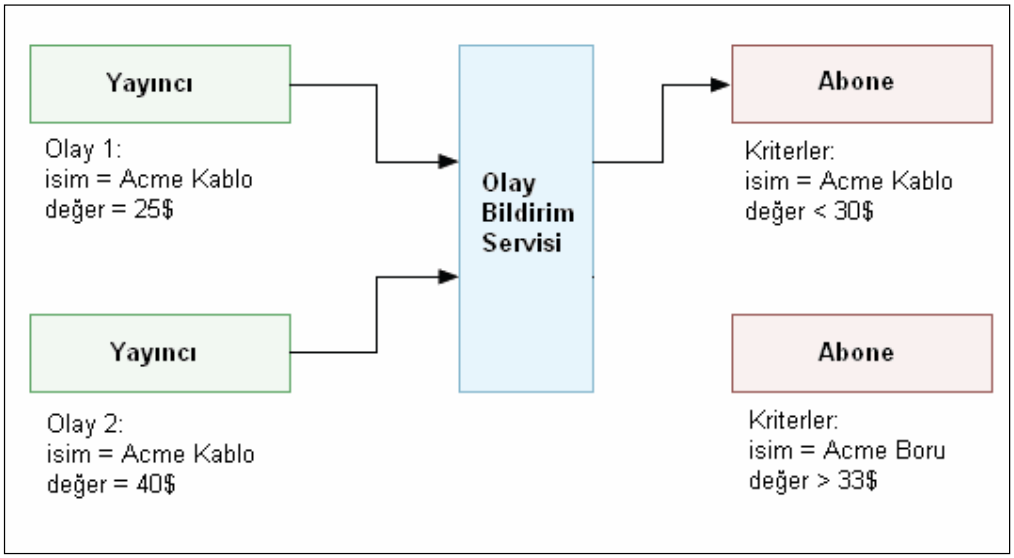
Konu tabanlı modelin en temel sakıncası, abonelere çok kısıtlı bir anlaşılırlık sunmasıdır [8]. Bu sebeple gereksiz iletişimi ancak kısmi olarak engelleyebilmektedir. Bu problem tanımlı konu sayısını arttırarak giderilebilir ancak bu durumda da sistemdeki varlıkların daha fazla detay bilmek zorunda kalması gerekecektir.

4.4.2 İçerik tabanlı kayıt modeli

İçerik tabanlı kayıt modelinde çalışan sistemlerde, aboneler bildirim almak istedikleri olay yada mesajları çok daha ayrıntılı olarak belirtirler. Bunu yaparken bildirim içeriğinde gelmesini bekledikleri değişkenleri ve bunların alabileceği değerleri belirtirler. Aboneler beklentilerini belirtirken genellikle boole cebri kullanırlar. Bu yapı bir mağaza alışveriş yapan bir insanın davranışına benzetilebilir. Alınacak ürünün temel özelliklerinden kişiye uygun olanlar mağaza görevlisine belirtilerek mağaza genelinde yapılacak kapsamlı bir inceleme engellenmiş olur. Örneğin bir kazak alınacak ise temel özellikler olarak belirtilebilecek renk tonu, yaka tipi, kalınlığı, desenli olup olması gibi kriterlerin kişi tarafından belirlenmesi yapıldığında, koyu renk, yuvarlak yakalı, ince, düz bir kazak istendiği anlaşılacak ve mağaza görevlisi de alışveriş yapan kişiye sadece bu kriterlere uygun kazakları gösterecektir.

İçerik tabanlı kayıt modelinde bildirimler önceden tanımlı dış kriterlere (örn., konu ismi) göre sınıflandırılmayıp, bunun yerine her bir farklı bildirimde farklı değerler kabul eden bildirimlerin özelliklerine göre sınıflandırılırlar [8]. Ancak işleyiş olarak bu yapı konu tabanlı yapıdan çok da farklı değildir. Aboneler yine olay bildirim servisinin metotlarını kullanarak kayıt işlemlerini gerçekleştirirler. Bir yayıncı bir olay yayınladığında, olay bildirim servisi bunu alıp kayıt kütüğünden tarama yaparak ilgilenen aboneleri bulur ve olayı hepsine iletir. Fark ilgilenen aboneleri bulmada kullanılan kriterlerdedir. Aboneler tarafından yapılan her kayıt, özellikler üzerinde belirtilen kısıtların bir birleşimi olarak belirtilebilir [9].

Bu yapıda olması muhtemel bir akış Şekil 4.5'te bir örnek üzerine gösterilmiştir [9]. Buna göre abonelerden biri acme kablonun fiyatının 30 dolardan aşağı olduğu durumlarda haberdar olmak isterken diğeri acme borunun fiyatı 33 doların üzerinde olduğu durumlarda haberdar olmak istiyor. Bir yayıncı tarafından gönderilen acme kablonun fiyatının 25 dolar olduğu bilgisi sadece ilk abonenin verdiği kriterleri sağladığı için ona iletiliyor.



Şekil 4.5 : İçerik Tabanlı İletişim Modeli

4.5 Yayınla Abone Ol Yapısının Getirileri

Birçok kullanım şeklinden bahsedilen yayınla / abone ol yapısının ana avantajlarından birisi yayıncı ve aboneleri birkaç düzeyde bağımsızlaştırmasıdır [5]. Bağımsızlaşmanın sağlandığı üç düzey, uzay, zaman ve senkronizasyon düzeyleridir.

4.5.1 Uzay bağımsızlaşması

Varlıkların sistem içindeki iletişimlerini birbirleri üzerinden değil de bir konu, olay yada bunların özellikleri üzerinden sağlamaları sebebiyle birbirlerini tanıma yada yerini bilme gibi zorunlulukların ortadan kaldırması uzay bağımsızlaşmasını getirir. Uzay bağımsızlaşması iletişim kuracak varlıkların birbirlerinin referanslarını tutmaları zorunluluğunu kaldırması nedeni ile varlıkların sistem içindeki taşınmaları sadece olay bildirim sistemine beslemelerle yapılabilir.

4.5.2 Zaman bağımsızlaşması

Yayınla/abone ol yapılarında, iletişim kuran tarafların iletişimde aynı anda aktif olarak yer almasına gerek yoktur [9]. Bu durum zaman bağımsızlaşması olarak ifade edilir. Bu durumun sağlanabilmesine yayıncının olayı yayınlamasından sonra karşı tarafı beklememesi ve iletinin olay bildirim sistemi üzerinden alıcıya iletilmesi olanak sağlamaktadır.

4.5.3 Senkronizasyon bağımsızlaşması

Senkronizasyon bağımsızlaşması aslında zaman bağımsızlaşmasıyla çok ilinti bir kavramdır. Zaman bağımsızlaşmasının oluşmasını sağlayan unsurlar senkronizasyon bağımsızlaşmasının tanımını oluşturur. Bu tanım da, yayıncılar olay üretirken bloke olmazlar ve aboneler de asenkron olarak uyarılabilirler [11]. Bu yolla sistemde olası zaman aşımı problemleri giderileceği gibi yayıncı ve abonelerin paralel işleyebilmelerine olanak sağlandığı için verim arttırılabilecektir.

5. JADE İÇİN İÇERİK TABANLI YAYINLA / ABONEOL EKLENTİSİ

JADE üzerine yapılan eklenti, içerik tabanlı bir yayınla/abone ol yapısının gelişmiş bir halidir. Çalışma ortamı olarak "Eclipse 3.2" kullanılmıştır. Eklenti JADE'in yazıldığı programlama dili olan Java kullanılarak yapılmıştır. Kod yazımında mühendislik esaslarına uygun olarak kod içinde gerekli yerlere açıklamalar eklenmiş, her sınıf ve metot başına içerik hakkında bilgi veren java dokümantasyonu eklenmiştir. Bir akış diyagramı, bir algoritma yada algoritmanın bir parçasının grafik olarak gösterimidir [12]. Anlaşılırlığı kolaylaştırdığı için önemli metotların işlevleri anlatılırken akış diyagramları da kullanılmıştır.

Eklenti için, İtalya telekom tarafından konu tabanlı iletişim eklentisinde olduğu gibi mevcut eşdüzeyde iletişim altyapısı ve filtreleme özellikleri kullanılmıştır. Bu kapsamda gönderilerde kullanılan ACL mesaj yapısı kullanılmış, ancak mesaj içeriği oluşturulmasında farklı bir yapı oluşturularak düz yazı mesaj içerikleri yerine kodlanmış parametrik içerik eklenmiştir.

Çalışma kapsamında yeni bir mesaj oluşturma yapısı, yeni bir kayıt modeli ve buna uygun gerçeklemeler, parametrik mesaj yapısına uygun olay bildirim servisi ve bu değişiklikleri desteklemek üzere mevcut JADE altyapısında düzenlemeler yapılmıştır.

5.1 Uygulanan Kod Yazma Standartları

Metot yazımlarında, okunabilirliği arttırmak için tek sayfada görülebilecek kısa kodlar yazılmasına dikkat edilmiştir. Genellikle çok kısa kodlarda açıklama eklenmemiş, ancak ana işleri yapan metotlarda açıklamalarla kodun anlaşılırlığı arttırılmaya çalışılmıştır.

Java dokümantasyonunda standart oluşturulmuş ve tüm kodlarda aynı standart uygulanmıştır. Buna göre bir sınıfa ait standart java dokümantasyonu aşağıda verilmiştir.

```
/**
 * <p>
 * "Sınıf içerisindeki kodun genel amacı ve kullanımı bu bölümde
 * belirtilir."
 * </p>
 * @author "Sınıfı yazan kişinin adı" "Sınıfın oluşturulma tarihi"
 * <p>
 * @history "İlgili değişikliğin yapılma tarihi"
 * "Yapılan değişiklik hakkında bilgi bu bölümde yer alır.
 * </p>
 */
```

Metotlarda kullanılan standart java dokümantasyonu ise aşağıdaki gibidir.

```
/**
 * <p>
 * "Metodun ne işe yaradığı ve ne amaçla kullanıldığı bu bölümde
 * belirtilir."
 * </p>
 * @param "parametrenin adı" "Parametrenin türü"
 * @return "dönüş tipi"
 * @throws "hata tipi"
 */
```

Bu standartlar kullanılarak kodun daha anlaşılır olması amaçlanmıştır.

5.2 Mesaj Oluşturma Yapısı

Yapılan eklentide kullanılan parametrik yapıyı oluşturmak için "ParametricMessage.java" sınıfı kullanılmıştır. Tamamıyla yeni olan bu sınıf içerdiği metotlar ile parametrelerin eklenmesini ve mesaj içeriğine koyulacak olan kodlanmış parametrik içeriğin oluşmasını sağlar. Sınıf içinde tanımlı değişken ve metotlar şunlardır:

- topicParameters : "Hashtable" tipindeki bu değişken parametreleri ve onların değerlerini tutmak için kullanılır.

- aclMessage : "ACLMessage" tipinde olan bu değişken mevcut bir içeriğin parametrik mesaja uyarlanmasında yada yeni oluşturulmuş bir içeriğin ACL mesajı olarak verilmesinde kullanılır.

- keys : "Vector" tipindeki bu değişken parametreleri liste halinde tutmak için kullanılır.

- ParametricMessage(ACLMessage acl) : Bu metot parametre olarak aldığı ACL mesajındaki içeriği kullanarak "topicParameters" ve "keys" değişkenlerini doldurur.

- public void put(String key, String value) : Bu metot kendisine parametre olarak gelen mesaj parametresini "keys" değişkenine ekler, mesaj parametresini ve buna ilişkin düz yazı değeri de "topicParameters" değişkenine ekler.

- public void put(String key, long value) : Bu metot kendisine parametre olarak gelen mesaj parametresini "keys" değişkenine ekler, mesaj parametresini ve buna ilişkin "long" tipinden değeri de "topicParameters" değişkenine ekler.

- public void put(String key, int value) : Bu metot kendisine parametre olarak gelen mesaj parametresini "keys" değişkenine ekler, mesaj parametresini ve buna ilişkin tamsayı değeri de "topicParameters" değişkenine ekler.

- public void put(String key, short value) : Bu metot kendisine parametre olarak gelen mesaj parametresini “keys” değişkenine ekler, mesaj parametresini ve buna ilişkin “short” tipinden değeri de “topicParameters” değişkenine ekler.

- public void put(String key, float value) : Bu metot kendisine parametre olarak gelen mesaj parametresini “keys” değişkenine ekler, mesaj parametresini ve buna ilişkin “float” tipinden değeri de “topicParameters” değişkenine ekler.

- public ACLMessage getACLMessage() : Bu metot bir parametrik mesaj nesnesi üzerinden çağırıldığında bu nesne içerisinde oluşturulmuş ACL mesajını döner.

- public void setACLMessage(ACLMessage acl) : Bu metot üzerinden çağırıldığı objenin ACL mesajını tutan “aclMessage” değişkenine aldığı parametreyi yazar.

- public String getAsString(String key) : Bu metot parametre olarak aldığı anahtarı kullanarak buna ilişkin değeri düz yazı olarak “String” tipinde döner.

- public String getAsLong(String key) : Bu metot parametre olarak aldığı anahtarı kullanarak buna ilişkin değeri “long” tipinde döner.

- public String getAsInteger(String key) : Bu metot parametre olarak aldığı anahtarı kullanarak buna ilişkin değeri tamsayı tipinde döner.

- public String getAsShort(String key) : Bu metot parametre olarak aldığı anahtarı kullanarak buna ilişkin değeri “short” tipinde döner.

- Public Vector getKeys() : Bu metot “keys” değişkenini döner.

Bu “ParametricMessage.java” sınıfı ile oluşturulan mesajlar olay bildirim servisi tarafından alınıp yorumlanır ve ilgili etmenlere iletilir.

5.3 Kayıt Modeli

Bir etmen bir olay hakkında mesaj almak istediği taktirde, ilgili konuyu ve bu konu üzerindeki kısıtlarını belirterek kayıt olur. Kayıt işlemleri için kullanılan sınıf “TopicManagementService.java” içindeki “TopicHelperImpl” iç sınıfıdır. Bu sınıfa ait metotlar “TopicManagementHelper.java” sınıfında tanımlamış olup içerikleri

burada doldurulmuştur. “TopicHelperImpl” sınıfına ait parametrik mesaj kayıt işleminde kullanılan metotlar aşağıdaki gibidir:

- public AID createParametricTopic(String topicName) : Bu metot kendisine gönderilen parametreye bir son ek ekleyerek parametrik konu olduğunu belirten bir etmen belirteci oluşturur. Hakkında mesaj alınmak istenen konu için kayıt olma sırasında konu adı da kullanıldığı için bu metot çağırılmak zorundadır. Ayrıca bu metot sayesinde konuların parametrik olduğunun kolayca anlaşılmasına olanak sağlanmıştır.

- public boolean isParametricTopic(AID id) : Boole bir değer dönen bu metot olay bildirim servisi tarafından gelen mesajın parametrik bir konu ile ilgili olup olmadığının anlaşılmasında kullanılır.

- public void register(TopicParameters params) : Bu metot tüm sistem bölümlerinin bilgisini alarak kayıt işlemini hepsine iletir.

- public void deregister(TopicParameters params) : Bu metot tüm sistem bölümlerinin bilgisini alarak kayıt silme işlemini hepsine iletir.

Bu iç sınıfa ait metotlar üzerinden “TopicManagementService.java” sınıfındaki metotlar çağırılarak kayıt olma ve kayıt silme işlemleri ilgili sistem bölümlerine yani etmen depolarına iletilir. Bu sınıfta kayıt akışında kullanılan metotlar şöyledir:

- private void broadcastRegistration(TopicParameter params, Service.Slice[] slices) : Bu metot kendisine parametre olarak gelen etmen deposu listesine ilgili parametrik konu kayıt nesnesini göndererek onlarında kayıt yapmalarını sağlar..

- private void broadcastDeregistration(TopicParameter params, Service.Slice[] slices) : Bu metot kendisine parametre olarak gelen etmen deposu listesine ilgili parametrik konu kayıt nesnesini göndererek onlarında kayıt silme işlemi yapmasını sağlar.

Tüm etmen depolarına ilgili kayıt olma yada silme işlemi bildirildikten sonra bu depolarda yine “TopicManagementService.java” sınıfının bir iç sınıfı olan “ServiceComponent” sınıfına ait metot. Bu metot şöyle verilebilir:

- public VerticalCommand serve(HorizontalCommand cmd): Bu metot genel olarak sistemdeki bölümlere gönderilen kayıt mesajını alıp, eğer mesaj bir kayıt olma mesajı ise "ParametricTopicTableDM" tipindeki "parametricTopicTable" değişkeninin "register(TopicParameters)" metodunu, eğer mesaj bir kayıt silme mesajı ise "deregister(TopicParameters)" metodunu çağırır.

Bu noktadan sonra "ParametericTopicTableDM.java" sınıfının kayıt ve kayıt silme metotları kullanılır. Bu sınıfa ait değişkenler ve metotlar şöyle verilebilir:

- detailTable : "Vector" tipindeki bu değişken kayıt satırlarından oluşan tabloyu tutar.

- final synchronized void register(TopicParameters params) : Bu metot kendisine parametre olarak gelen bir satırlık bilgiyi temsil eden nesneyi bir özel metod olan "createRow(TopicParameters params)" ile "detailTable" değişkenine yazar.

- final synchronized void deregister(TopicParameters params) : Bu metot kendisine parametre olarak gelen kayıtları silinecek konu ve etmen bilgisini içeren nesneyi kullanarak kayıt tablosu üzerinde arama yapar ve gelen bilgilere uyan "deleteRows(TopicParameters prms)" özel metodunu kullanarak kayıtları kayıt tablosundan siler.

- public Vector getRegistrations(TopicParameters prms) : Bu metot kendisine parametre olarak gelen nesnedeki kayıt bilgilerine bakarak bunlara uyan tüm kayıtları "Vector" tipinde döner.

- public Vector getInterests(AID topic, ParametricMessage msg) : Bu metot parametre olarak gelen etmen belirtecini ve parametrik mesajı kullanarak kayıt tablosundan gelen bu mesaj ile ilgilenen etmenleri tespit etmek için kullanılır.

Bu metotlar kullanılarak yapılan kayıt işlemi sırasında kayıt tablosunu tutan vektöre eklenen ve tablonun bir satırına karşılık gelen konu parametreleri "TopicParameters.java" sınıfı ile tanımlanmış olup bu sınıfa ilişkin değişken ve metotlar şöyledir:

- indexes : “String” tipindeki bu değişken kayıt işleminde kullanılan fonksiyon içerisinde yer alacak olan değişkenlerin indeks bilgilerini ayraçlarla birbirine eklenmiş olarak tutar.

- function : “String” tipindeki bu değişken bir konuya kayıt olan etmenin o konu ile ilgili kısıt bilgisini yani ilgili konunun sahip olduğu değişkenlerden hangilerinin hangi değerlerini etmenin kabul edeceğini tutar.

- keys : “String” tipindeki bu değişken kayıt olan etmenin verdiği fonksiyona ilgili parametrelerin hangi sırayla uygulanacağını yani fonksiyondaki değişken sekansını tutar.

- topic : “AID” tipindeki bu değişken kayıt olan etmenin hangi konu başlığındaki mesajları almak istediğini tutar.

- agentID : “AID” tipindeki bu değişken kayıt olan etmenin belirtecini tutar.

- action : tamsayı tipindeki bu değişken kayıt olan etmenin verdiği kriterlere uyan bir mesaj alındığında ne yapılması gerektiğini tutar. (Bu değişkenin alabileceği değerleri ve bunların ne anlama geldiği bilgisi olay bildirim servisin anlatıldığı bölümde detaylı olarak ele alınacaktır.)

- sender : kayıt olan etmenin isteğine göre uygun durumda yeni bir mesaj gönderilecekse bunun göndereni olarak belirlenecek etmen belirtecini tutar.

- receiver : kayıt olan etmenin isteğine göre uygun durumda yeni bir mesaj gönderilecek yada gelen mesaj yönlendirilecek ise alıcı olacak etmenin belirtecini tutar.

- content : kayıt olan etmenin isteğine göre uygun durumda yeni bir mesaj gönderilecekse bunun içeriğini tutar.

- public boolean match(TopicParameters compared) : Bu metot kendisine parametre olarak gelen konu parametreleri ile metodun üzerinden çağırıldığı nesnenin birbiri ile uyup uymadığını belirten bir boole değer döner. Karşılaştırma yapılırken parametrede değer verilmemiş değişkenler için karşılaştırma yapılmaz.

- public boolean suits(TopicParameters registered, ParametricMessage msg) : Bu metot bir mesaj alındığında kayıt kütüğünde buna uyan bir kayıt olup olmadığını anlamak için kullanılır. Kayıt kütüğündeki her bir kayıt bir parametrik bir konu olarak gelen mesaj ile karşılaştırılır. (Bu metot olay bildirim servisi işleyişi anlatılırken detaylı olarak, karşılaştırmanın nasıl yapıldığı da belirtilerek anlatılacaktır.)

- public void copyTo(TopicParameters target) : Bu metot, üzerinden çağırılan nesnenin parametre içeriklerini parametre olarak gelen nesnenin parametre içeriklerine kopyalar.

Bu sınıfta bulunan diğer metotlar tamamen işlevsel olarak nesnenin değişkenlerini belirleme metotlarıdır.

Hisse senetlerinin takibi ve işlenmesi üzerine çalışan bir sistemde bir etmenin kayıt koşulu, "Acme şirketine ait hisse senetlerinin fiyatı 5.95'in altına indiğinde mevcut talep sayısı da 100'ün altında olduğunda haber almak istiyorum" şeklinde verilmiş olsun. Bu isteğin kayıt satırına dönüştürülmüş hali Tablo 5.1'de verilmiştir.

Tablo 5.1 : Hisse Senedi Takibi Kayıt Örneği

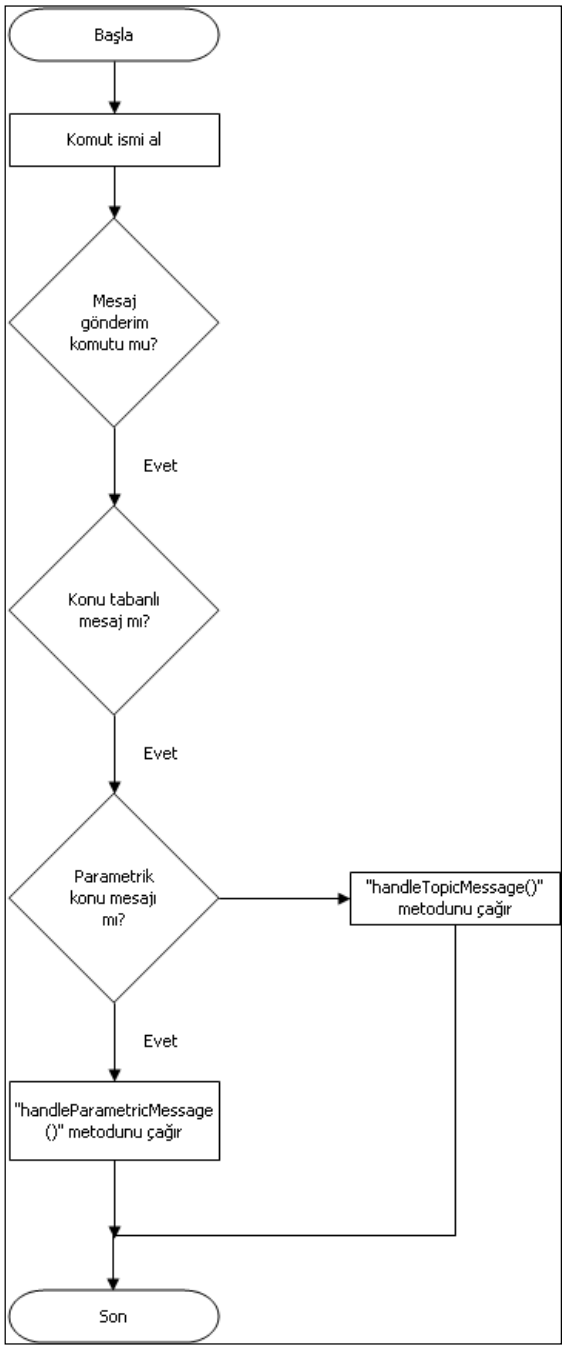
Indexes	Function	Keys	Topic	agentID	Action	Sender	Receiver	Content
1*10*	<5.95 && <100	FIYAT*SAYI	ACME	A1	1			

Kayıt işleminde etmenlerin aynı konu üzerine birden fazla defa kayıt olmalarına izin verilir. Bu yolla mesaj isteme koşulları çok karmaşık olan etmenlerin yorumlanması zor büyük fonksiyonlar oluşturması ve belirtilen davranış tiplerine göre aynı durum için farklı mesaj gönderimlerinin yapılabilmesi hedeflenmiştir.

5.4 Olay Bildirim Servisi

Olay bildirim servisinin işlevini JADE sisteminde "TopicManagementService.java" sınıfı içindeki metotlar tarafından yürütülür. Bir etmen tarafından gönderilen her mesaj bu sınıf içerisindeki "Filter.java" sınıfından türemiş bir iç sınıf olan "CommandOutgoingFilter.java" sınıfında tanımlı metotlar tarafından yürütülür. Bu sınıf içinde değişken tanımı olmayıp metotlar aşağıdaki gibi verilebilir:

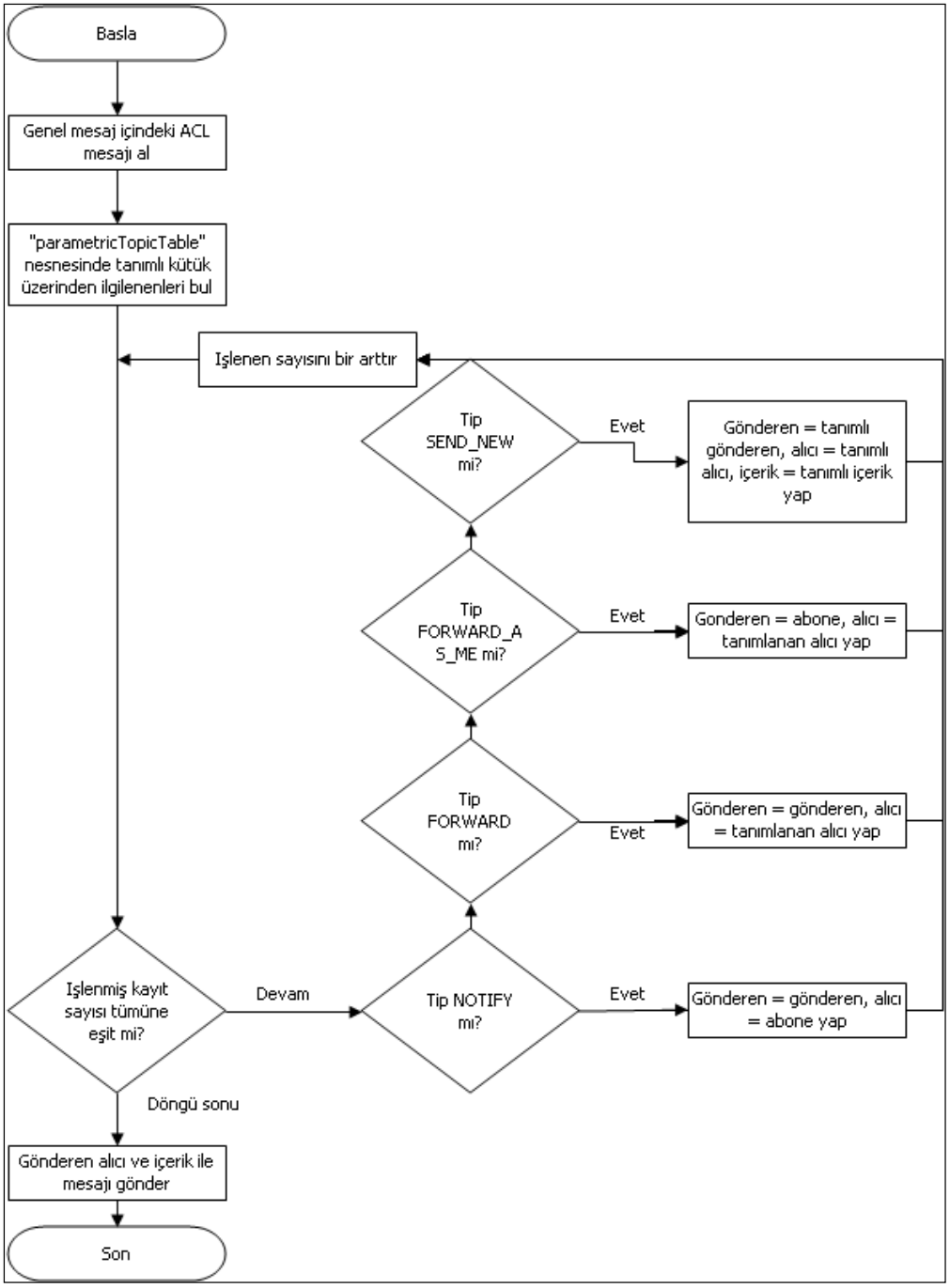
- public final boolean accept(VerticalCommand cmd) : Bu metot asıl filtreleme işlevini gören yapı olup, gönderilen mesajları bir komut şekli olarak algılayıp öncelikle bir mesaj gönderim komutu olup olmadığını sorgular sonrasında ilgili komut bir mesaj gönderim komutu ise bunun bir konu olup olmadığı kontrol edilir. Konu tabanlı gönderilen mesaj ise bir de parametrik konu mesajı olup olmadığına bakılarak ilgili işi yapacak metot çağırılır. Bu metoda ilişkin akış Şekil 5.1'de verilmiştir.



Şekil 5.1 : Kabul Metot Akışı

- private void handleParametricMessage(AID sender, GenericMessage gMsg, AID receiver) : Bu metot bir konu üzerine gönderilmiş parametrik mesajların işlenmesi işini yapar. Bunun için öncelikle gelen mesajın ACL mesaj içeriğini kullanarak "ParametricMessage" tipinden nesne yaratır. Bu nesneyi kullanarak daha önce mesaj kaydı oluşturma kısmında anlatılan kayıt kütüğüne ait

"getInterests(...)" metodunu çağırır ve vektör tipinde bir liste şeklinde gönderilen mesajla ilgili abonelik kayıtlarına ulaşır. Bu kayıtlar üzerinde dönerek, ilgili davranış tipine göre yeni gönderilecek yada yönlendirilecek mesajları oluşturur. Bu metoda ilişkin akış Şekil 5.2'de verilmiştir.



Şekil 5.2 : Parametrik Mesaj İşleme Akışı

- private final void sendMessage(AID sender, GenericMessage gMsg, AID receiver) : Bu metot parametre olarak aldığı "sender" etmen belirtecindeki gönderici bilgisi ve "receiver" etmen belirtecindeki alıcı bilgisi kullanarak "gMsg" genel mesaj nesnesinin sistemde ilgili yere iletilmesini sağlar.

Olay bildirim servisi tarafından mesaj gönderimleri yapılırken, mesaj gönderim şekilleri daha önce kayıt olmuş etmenin belirttiği davranış tiplerine göre yapılır. Bu davranış tipleri şunlardır:

- NO_ACTION : Bu davranış tipi hiçbir yapılmayacağını belirtmek için kullanılır ancak kayıt yapısında kullanılan bir davranış tipi olmadığı için olay bildirim servisinde bu tip kontrol edilmez. Bu tip boş bir "TopicParameters" nesnesi oluşturmak yada mesaj içeriklerinin kayıtlar ile karşılaştırılması sırasında mesaj içeriğine karşılık gelen davranış tipi olarak kullanılır.

- NOTIFY : Bu davranış tipi mesaj akışında herhangi bir değişiklik yapılmadan gönderici etmenin mesajının aynen eşdüzeyde iletilmiş gibi alıcı etmene yani kayıt olan aboneye iletilmesi gerektiğini belirtir. Bu normal konu yada içeri tabanlı yayınla/abone ol yapılarında kullanılan davranış biçimidir.

- FORWARD : Bu davranış biçimi çalışma kapsamında içerik tabanlı iletişim yapısına getirilen yeniliklerden biridir. Olay bildirim servisi bu tiple bir kayıt aldıysa, buna uyan bir mesaj geldiğinde ilgili abonenin belirttiği etmene bu mesajı gönderir. Bu delegasyon yada yetki dağıtım yapısı kapsamında kullanılan üç tipten biri olup bir başka etmen adına kayıt olma olanağı sağlamıştır. Bunu yaparken gerçekte işlemi yapan etmenin belirtecini kaybetmeyerek işlem kaydı da tutulmuştur.

- FORWARD_AS_ME : Yetki dağıtım yapısı yeniliklerini kullanmak için tanımlanmış bir diğer tip davranış da mesaj gönderen etmenin yerini almaya olanak sağlayan bu tiptir. Bu davranış tanımı sayesinde belirli durumların oluştuğuna dair mesaj alındığında kendisinden mesaj bekleyen etmenlere sanki kendisinden geliyormuş gibi otomatik mesaj gönderimi yapılabilir hale gelmiştir. Bu etmenin kendi üzerindeki otomatize işin bir kısmını olay bildirim servisine devretmesi olarak da yorumlanabilir. Bu sayede normalde bir etmene iletilip onun tarafından yorumlandıktan sonra tekrar olay bildirim servisine gönderim

yapılıp buradan ilgili etmenlere dağıtılması gerekliliği ortadan kaldırılmış ve işlem anlamında kazanç sağlanmış olmaktadır. Ayrıca ilgili etmenlerin bu mesajı alabilmesi için kendilerinin de ayrıca kayıt olması zorunluluğu kalmamıştır. Hem işlem hem kayıt anlamında kazanç sağlayan bu tip içerik tabanlı iletişimde önemli bir unsurdur.

- SEND_NEW : İçerik tabanlı iletişime getirilen bir diğer yenilik olan bu tip, belirli bir durumda gönderici ve alıcı etmen bilgilerine kadar tamamıyla yeni bir mesaj oluşturularak sisteme gönderilmesine olanak sağlanır. Delegasyon yapısındaki bu yenilik, sistem mesaj trafiğinde bir kontrol etmeni oluşturulmasını mümkün hale getirir. Bu sayede tamamen farklı iki etmen adına kayıt ve gönderim işlemleri yürütülmüş olur. Diğer tiplerde olduğu gibi burada da işlemi gerçekte hangi etmenin yaptığı bilgisi tutulmaktadır.

Mesaj gönderimi dışında sistemsel durumlar için de olay bildirim servisi işlevi gören "TopicManagementService" kayıt sistemi üzerinde bir takım işlemler gerçekleştirir. Bu kapsamda yönetilmesi gereken sistemsel durumlar, bir etmenin sonlanması, sisteme yeni bir depo yada bölümün eklenmesi, bir deponun sistem bağlantısının koptuktan sonra yeniden gelmesi durumlarıdır. Bu durumları yöneten metotlar şunlardır:

- private void handleInformKilled(VerticalCommand cmd) : Bu metot bir etmen sonlandığı zaman bu etmenin kayıt olduğu durumları tespit edip bunların kayıt kütüğünden silinmesi için kullanılır. Bunun için tüm bölümler tespit edilip bunlara "broadcastDeregistration(...)" metodu yardımıyla ilgili kayıtların silinmesi için uyarı gönderilir. Bundan sonra kayıt oluşturma ile ilgili bölümde anlatıldığı şekilde kütükten kayıtlar silinir. Bu metot içerisinde parametrik mesaj yapısı özelinde bir şey yapılmamıştır.

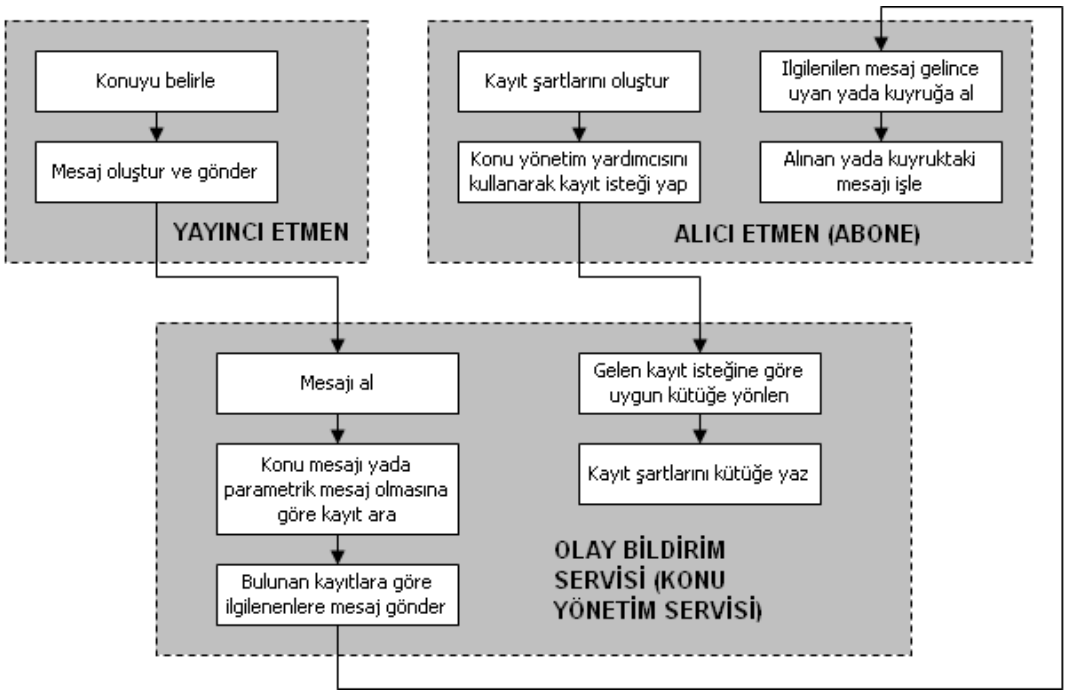
- private void handleNewSlice(VerticalCommand cmd) : Bu metot sisteme yeni bir bölüm eklenmesi durumunda çalışır. Yeni bölümdeki kayıt bilgilerini güncellemek için mevcut kayıt bilgileri alınır ve ilgili bölüme iletilir. Bu kapsamda her iki yapının da desteklenebilmesi için normal konu tabanlı yapı kodlarının yanında yeni parametrik kayıt bilgilerini göndermek üzere kodsal değişiklik yapılmıştır.

private void handleReattached(VerticalCommand cmd) : Bu metot daha önce sistemde olup bir şekilde bağlantısı kopan ve sonra tekrar sisteme dahil olan bir bölümün kayıt bilgilerinin güncellenmesi için kullanılır. Aynen yeni bölüm eklenmesinde olduğu gibi mevcut konu tabanlı kayıt kütüğündeki ve yeni eklenen parametrik kayıt kütüğündeki kayıtlar alınarak yeni bölümün "register(...)" metodu çağırılır.

5.5 İletişim Akışı ve Fonksiyon Yorumlama Yapısı

5.5.1 Genel mesaj akışı

Mesaj akışı bir etmenin kayıt olması ve bir başka etmenin de bu kayda uygun mesaj üretmesi sonucu sistemde yapılan dönüşüm ve iletimlerin tamamıdır. Bu kapsamda kayıt işlemi için etmen, "TopicManagementHelper.java" sınıfı ile tanımlanan metotları kullanarak kayıt kütüğüne ilgilendiği konu ve buna ilişkin kısıtlar ve davranışları kaydettirir. Üretici yada yayıncı etmen de ilgili konu ile ilgili mesajı "ParametricMessage.java" sınıfının yardımıyla oluşturarak gönderir. Bu noktadan sonra olay bildirim servisi devreye girerek mesajı alıp hangi tip mesaj olduğunu belirler. Mesajın bir parametrik mesaj olması durumunda olay bildirim servisi kayıt kütüğünün bulunduğu "ParametricTopicTable" tipindeki nesnenin metotları üzerinden ilgili kayıtları bulur ve bunlarda belirtilmiş davranışlar doğrultusunda mesaj gönderimi yapar. Genel akış Şekil 5.3'teki gibi verilebilir.



Şekil 5.3 : Genel Mesaj Akışı

Mesaj akışı kapsamında bir çok sınıf tarafından gerçeklenen bir sınıf olan "Constants.java" kod kalitesinin arttırılması ve tekrarlanan sabit yazı yada sayıların tek bir sınıf içerisinden müdahale edilerek değişikliklerde hata riskini en aza indirmek için oluşturulmuştur. Sınıf içerisinde metot bulunmayıp sistem kapsamındaki değişkenlerin bulunduğu bir depo olarak düşünülebilir. Bu sınıf kapsamındaki değişkenler şunlardır:

- NOTIFY : "short" tipindeki bu değişken kayıt yapısında kullanılan davranış tipi belirtecidir.
- FORWARD : "short" tipindeki bu değişken kayıt yapısında kullanılan davranış tipi belirtecidir.
- FORWARD_AS_ME : "short" tipindeki bu değişken kayıt yapısında kullanılan davranış tipi belirtecidir.
- SEND_NEW : "short" tipindeki bu değişken kayıt yapısında kullanılan davranış tipi belirtecidir.

- NO_ACTION : “short” tipindeki bu değişken kayıt yapısında kullanılan davranış tipi belirtecidir.

- SEPARATOR : “String” tipindeki bu değişken kayıt işlemlerinde kütüğe birleştirilerek yazılan anahtar ve indeks değerlerinin birbirinden ayrılması için kullanılır.

- KEY_INDICATOR : “String” tipindeki bu değişken fonksiyon yorulama sırasında belirtilen indekslere göre anahtarları yerleştirirken ifadeyi fonksiyondan ayırmak için kullanılan ayraçtır.

- INDEX_KEY_ARRAY_DONOT_MATCH : “String” tipindeki bu değişken mesaj oluşturma sırasında indeks ve anahtarların toplu halde verildiği durumda birbiri ile uyuşmadığını belirten bir hata tanımıdır.

- DATA_TYPE_MISSING : “String” tipindeki bu değişken mesaj oluşturma sırasında eklenen anahtarın veri tipi olmaması durumunda verilen hata tanımıdır.

- INVALID_FUNCTION : “String” tipindeki bu değişken yorumlama sırasında kurallara uymayan bir fonksiyon olması durumunda verilen hata tanımıdır.

- INVALID_OPERATOR : “String” tipindeki bu değişken fonksiyon yorumlama sırasında tanımsız bir operatör ile karşılaşılması durumunda verilen hata tanımıdır.

- SHORT_SUFFIX : “String” tipindeki bu değişken “short” tip belirtecidir.

- LONG_SUFFIX : String” tipindeki bu değişken “long” tip belirtecidir.

- FLOAT_SUFFIX : String” tipindeki bu değişken “float” tip belirtecidir.

- INT_SUFFIX : String” tipindeki bu değişken tamsayı tip belirtecidir.

- STRING_SUFFIX : String” tipindeki bu değişken “String” tip belirtecidir.

- AND_OPERATOR : “String” tipindeki bu değişken “ve” işlemini tanımlamak için kullanılır.

- OR_OPERATOR : “String” tipindeki bu değişken “veya” işlemini tanımlamak için kullanılır.
- GREATER_OPERATOR : “String” tipindeki bu değişken “büyüktür” işlemini tanımlamak için kullanılır.
- LESS_OPERATOR : “String” tipindeki bu değişken “küçüktür” işlemini tanımlamak için kullanılır.
- GREATEREQ_OPERATOR : “String” tipindeki bu değişken “büyük eşit” işlemini tanımlamak için kullanılır.
- LESSEQ_OPERATOR : “String” tipindeki bu değişken “küçük eşit” işlemini tanımlamak için kullanılır.
- EQUAL_OPERATOR : “String” tipindeki bu değişken “eşittir” işlemini tanımlamak için kullanılır.
- NOTEQ_OPERATOR : “String” tipindeki bu değişken “eşit değildir” işlemini tanımlamak için kullanılır.
- PLUS_OPERATOR : “String” tipindeki bu değişken toplama işlemini tanımlamak için kullanılır.
- MINUS_OPERATOR : “String” tipindeki bu değişken çıkarma işlemini tanımlamak için kullanılır.
- MULT_OPERATOR : “String” tipindeki bu değişken çarpma işlemini tanımlamak için kullanılır.
- DIV_OPERATOR : “String” tipindeki bu değişken bölme işlemini tanımlamak için kullanılır.
- OPERATORS : “String” dizisi tipindeki bu değişken tüm işlemleri içinde barındırır.

- BOOLEAN_OPERATORS : “String” tipindeki bu değişken “ve” ve “veya” işlemlerinin birleştirilmiş halini tutar. Fonksiyon yorumlama sırasında sonuçları mutlaka boole ifade olması gereken kısımları ayırmak için kullanılır.

5.5.2 Fonksiyon yorumlama mekanizması

Fonksiyon yorumlama mekanizması etmenlerin kayıt olurken bir konu hakkında belirttikleri kısıtları boole bir sonuç döndürecek şekilde tasarlanmış bir fonksiyon ile ifade etmeleri dolayısıyla olay konu yönetim servisi tarafından kullanılmaktadır. Fonksiyon yorumlama ile ilgili işlemler “FunctionInterpreter.java” isimli sınıf tarafından tanımlanmıştır. Bu sınıf içinde tanımlı değişken de metotlar şunlardır:

- message : “ParametricMessage” tipindeki bu değişken kayıt kütüğündeki fonksiyonlara uygulanacak parametrelerin değerlerini içeren parametrik mesajı saklar.

- public boolean applyFunction(TopicParameters registered) : Bu metot kayıt kütüğündeki kayıt üzerinden anahtar tanımlarını alıp yine kayıt üzerindeki fonksiyona anahtar belirteçleriyle birlikte bunlara ilişkin yer bilgilerini kullanarak ekler. Daha sonra elde edilen anahtarlı fonksiyonu “processFunction(...)” metoduna gönderip bunun dönüş değerini sonuç olarak olay bildirim servisine iletir.

- private boolean processFunction(String fnc) : Bu metot boole bir sonuç elde edilebilecek bir düz fonksiyon oluşturmak amacıyla mevcut fonksiyonu öncelik belirlemedeki tek kıstas olan parantezlere göre bölümleyip işlenebilecek bölümleri “processSub(...)” metoduna göndererek, buradan aldığı sonuçları birleştirir. Son olarak yine elde ettiği boole sonuç verebilecek düz fonksiyonu “processSub(...)” metodu ile işleterek sonucu boole olarak döner.

- private String processSub(String sub) : Bu metot kendisine parantezlerden arınmış olarak gelen fonksiyon parçalarını işler. Öncelikle “startsWithOperator(...)” metodunu kullanarak kendisine gelen parçanın bir operatör ile başlayıp başlamadığını kontrol eder. Eğer parça operatör ile başlıyorsa bu durumda doğrudan birleştirmek üzere geri gönderilir. Eğer parça operatör ile başlamıyorsa tekrar bir bölme işlemine tabi tutulur. Bu bölmede

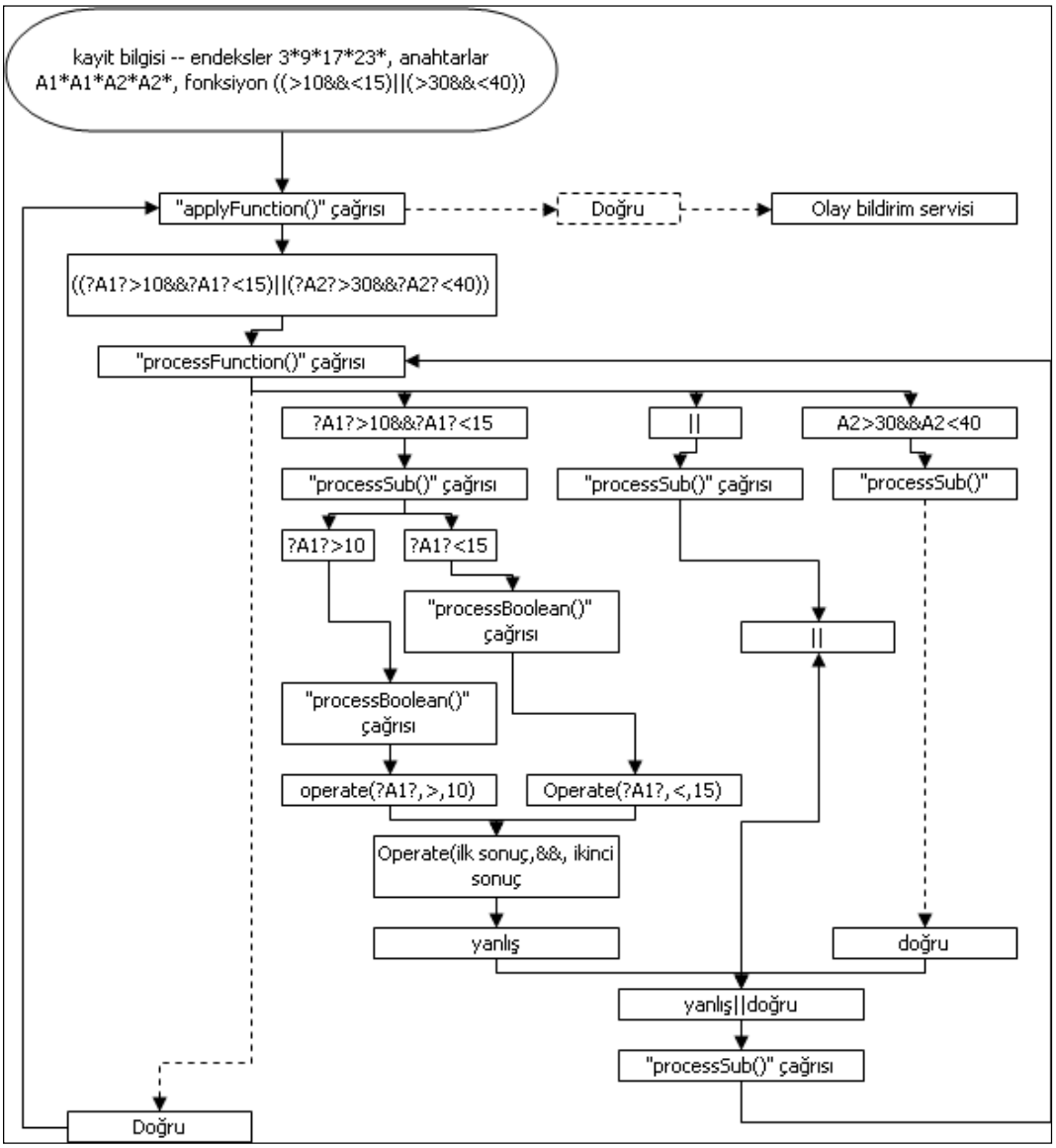
boole sonuç dönmesi gereken ana operatörler olan "ve" ve "veya" operatörlerinin yerleşimi esas alınır. Buna göre bölünmüş parçalar artık işlenecek değer, işlem ve ikinci işlenecek değer sırasıyla işlenebilir durumdadır. Bu parçaların sonuçları her zaman bir boole değer olacaktır çünkü "ve" ve "veya" operatörleri boole olmayan değerler ile işlem yapamaz. Metot her bir bölümü bir önceki bölümün işlenmesinden elde edilen sonucu kullanarak "operate(...)" metodu yardımıyla işler. Bölümlerin sonuçlarını "operate(...)" metoduna göndermeden önce "processBoolean(...)" metodu ile bulur.

- private String processBoolean(String opList, String subfunction) : Bu metot kendisine gönderilen alt fonksiyonu yine kendisine gönderilen operatör listesine göre bölüp işlenen ve işlemleri belirleyip "operate(...)" metodunu kullanarak bir boole değer elde eder. Bu değeri "String" olarak ilgili metoda döner.

- private String operate(String operand1, String operator, String operand2) : Bu metot tanımlı işlemlerin nasıl işleneceğini bilip ona göre bir işleme ait sonucu bulup dönen metottur. Öncelikle işlemin bir sayısal işlem mi yoksa boole işlem mi olduğu tespitini yapar. Bu kullanılan operatörlerin farklı işlemlerde farklı davranmasından kaynaklanmaktadır. Bir de kendisine işlenmek üzere gönderilen değerler içinde anahtar var ise nesne içindeki "message" değişkeninden bunun değerini alıp yerine yazma işlemini yapar.

- private boolean startsWithOperator(String statement) : Bu metot operatör dizisi üzerinde dönerek kendisine verilen ifadenin bunlardan biri ile başlayıp başlamadığını tespit eder.

Olay bildirim merkezi bu sınıftan bir nesne üretip parametre ve değerleri içeren mesajı bu nesneye vererek ilgili kayıt fonksiyonunun yorumlanmasını sağlar. Sonuca göre de ilgilenenler listesini belirler. Fonksiyon yorumlamanın genel akışı örnek bir fonksiyon üzerinden Şekil 5.4'teki gibi verilebilir.



Şekil 5.4 : Fonksiyon Yorumlama Akışı

6. ÖRNEK UYGULAMA

Kurulan sistemin hatalarını bulmak ve parametrik konu tabanlı iletişim eklentisinin çalışabilirliğini test etmek amacıyla basit bir proje yönetim uygulaması tasarlanmıştır. Bu örnek uygulama ile delegasyon yapısını da kullanmak amaçlanmıştır.

Buna göre sistemde bir proje yöneticisini temsil eden etmen, bir projedeki işi yapacak grubun yöneticisi temsil eden etmen, iki projedeki işi yapacak grup elemanını temsil eden etmen ve bir de grup yöneticisinin yöneticisi olan birim yöneticisini temsil eden etmen bulunacaktır. Birim yönetici etmen ve grup elemanı etmenleri iletişinde rol almayacak sadece kayıt ve mesaj gönderim işlemlerinin doğru yapıldığını izlemek için kullanılacaktır.

6.1 Grup Yöneticisi Etmen

Grup yöneticisi etmen projelerde kendi grubuna ilişkin bir iş olan web servis hazırlama konusunda bir ihtiyaç olursa kendi belirlediği süre ve bütçeler dahilinde olduğu takdirde yine kendi yetkisi dahilinde tespit ettiği kişilere ilgili işi vermekle yükümlüdür.

İş dağıtımında grup yöneticisi, ilk grup elemanına daha az tecrübeli olduğu için süresi 10 günü geçmeyen yada 4000YTL'den az bütçe ayrılması gereken küçük işlerin verilmesi gerektiğini düşünmektedir. Bu durumu oluşturmak için kayıt kütüğüne aşağıdaki gibi gönderici ve içerik bilgileri boş olan bir giriş yapılmasını sağlayacaktır.

- Fonksiyon : (= = "WEB"||= ="ARAYÜZ")&&<10&&<4000
- Endeksler : 2*11*24*29*
- Anahtarlar : İŞ*İŞ*SÜRE*BÜTÇE

- Davranış : “FORWARD_AS_ME” (benden gelir gibi yönlendir)
- Alıcı : GE1 (Grup elemanı 1)

Grup yöneticisi ikinci eleman için ise, daha tecrübeli olduğu için süresi 10 ila 20 gün yada ayrılması gerekli bütçesi 4000YTL ila 9000YTL olan işleri uygun görmektedir. Bu durumu oluşturmak için kayıt kütüğüne yine gönderici ve içerik bilgileri boş olan aşağıdaki gibi bir giriş yapılmasını sağlayacaktır.

- Fonksiyon : (= = “WEB”||= =”ARAYÜZ”) && ((>=10&&<20) ||(>=4000&&<9000))
- Endeksler : 2*11*26*32*39*47*
- Anahtarlar : İŞ*İŞ*SÜRE*SÜRE*BÜTÇE*BÜTÇE*
- Davranış : “FORWARD_AS_ME” (benden gelir gibi yönlendir)
- Alıcı : GE1 (Grup elemanı 1)

Grup yönetici herhangi bir şekilde kendisine 20 günden fazla sürecek ve birde 9000YTL’den fazla bütçe ayrılması gerekecek bir iş gelirse bunu yöneticisine yönlendirmek istemektedir. Bunu sağlamak için aşağıdaki gibi kayıtlar oluşturacaktır.

- Fonksiyon : (= = “WEB”||= =”ARAYÜZ”) &&>=20&&>=9000
- Endeksler : 2*11*26*32*39*47*
- Anahtarlar : İŞ*İŞ*SÜRE*SÜRE*BÜTÇE*BÜTÇE*
- Davranış : “FORWARD” (yönlendir)
- Alıcı : BRY (Birim yöneticisi)

Bu belirtilen kayıtları oluşturmak için grup yöneticisi etmen “setup()” metodu içinde her kayıt için içeriği değişmekle birlikte aşağıdaki kodu yürütecektir:

//grup lideri bölüm adı olan web ve internet çözümlerini ifade eden

```
//WISP konusunu kayıt için yaratır

final AID topic = topicHelper.createParametricTopic("WISP");

//ilk eleman için belirlediği kısıtları oluşturur

TopicParameters tp = new TopicParameters();

tp.setFuction("(==\"WEB\"||==\"ARAYUZ\")&&<10&&<4000");

tp.addParameter(1, "IS".concat(STRING_SUFFIX));

tp.addParameter(10, "IS".concat(STRING_SUFFIX));

tp.addParameter(23, "SURE".concat(INT_SUFFIX));

tp.addParameter(28, "BUTCE".concat(INT_SUFFIX));

tp.setAction(FORWARD_AS_ME);

tp.setMinRegistrationInfo(topic, this.getAID());

tp.setMessageCreationInfo(null, new AID("emp1",false), null);

//kayıt işlemini tamamlar

topicHelper.register(tp);
```

Bu kodu diğer kayıtlar için de yürüttükten sonra gelecek mesajları dinlemeye başlar.

6.2 Proje Yöneticisi Etmen

Proje yöneticisi etmen örnek uygulama dahilinde üç adet örnek mesaj üretecektir. Bunlardan biri ilk grup elemanının şartlarına uyan bir mesaj, diğeri ikinci grup elemanının şartlarına uyan bir mesaj ve son olarak da şartlara uymayan birim yöneticisine gönderilecek olan mesaj olacaktır.

İlk mesajın içeriği şöyledir :

- İŞ : WEB

- SÜRE : 5
- BÜTÇE : 2000

İkinci mesajın içeriği şöyledir:

- İŞ : ARAYÜZ
- SÜRE : 15
- BÜTÇE : 8500

Üçüncü mesaj içeriği ise şöyle olacaktır:

- İŞ : WEB
- SÜRE : 32
- BÜTÇE : 12000

Bu içeriklere sahip mesajları göndermek için proje yöneticisi etmen içeriğe göre değişmek üzere "action()" metodunda aşağıdaki kodu yürütecektir:

```
ParametricMessage dnm = ParametricMessage.newInstance();

dnm.put("IS", "WEB");

dnm.put("SURE", 5);

dnm.put("BUTCE", 2000);

ACLMessage msg = dnm.getACLMessage();

msg.addReceiver(topic);

msg.setSender(myAgent.getAID());

myAgent.send(msg);
```

Uygulama içinde takibi kolaylaştırmak amacıyla mesajların gönderimi sırayla ve arada kısa bir bekleme süresi ile yapılmıştır. Her mesajın gönderimi sonunda alıcı

etmenler yerel dosyaya aldıkları mesaj detaylarını ve buna istinaden gerekli işi yaptıklarına dair bir yazıyı kaydetmektedirler. Bu sayede hem proje yöneticisi tarafından gönderilen mesajların içerikleri hem de alıcı tarafında mesajın hangi etmen tarafından alındığı ve alındığı zamanki içeriği takip edilebilir.

7. SONUÇLAR VE TARTIŞMALAR

Bu tez çalışmasında konu ve içerik tabanlı iletişim şekilleri, JADE örneği üzerinden etmen sistemleri tanıtılmış ve JADE için hem konu tabanlı hem de içerik tabanlı iletişimi destekleyebilen, her iki yönteme de bazı ek özellikler sağlayan bir eklenti oluşturulmuştur.

Çalışma sırasında ana amaç olan eklentinin hazırlanması işi yapılırken yan unsurlar olarak yazılan uygulamanın anlaşılırlığı ve standartlara uygunluğu da önemsenmiştir. Bu kapsamda yazılan kodların java dokümantasyonunda ilgili sınıf yada metot hakkında kısa bir açıklamaya yer verilmiştir. Ayrıca sınıf java dokümantasyonunda bir tarihçe yapısı tutularak geçmiş değişiklikler takip edilmeye çalışılmış ve olası bir hata durumunda bunlarında yardımıyla hatanın başka bir yazılımcı tarafından da tespit edilebilmesi kolaylaştırılmıştır. Metot java dokümantasyonunda ise, metoda ilişkin giriş / çıkış parametreleri belirtilerek yine başka bir yazılımcının metodu detaylı incelemesine gerek kalmadan kullanabilmesi sağlanmıştır. Bu detaylarla, açık kaynak kodlu bu yazılımın yayınlanması durumunda, yazılım üzerinde yeni geliştirmeler yapacak kişilere kolaylık sağlanmıştır.

Çalışma sırasında ana amaca yönelik düzenlemelerden biri olarak fonksiyon yorumlama mekanizması oluşturulmuştur. Bu sayede sistemde karakter katarları üzerinden yapılan mesaj alış verişleri ile içerik tabanlı iletişim mümkün kılınmıştır. Bu kapsamda istemciler tarafında, ilgilendikleri konularla ilgili kısıtları belirtirken tanımlı operatörleri kullanan mantıksal ifadeleri günlük hayatta insanların ifade ettiği gibi düz yazı şeklinde tanımlamaları sağlanmıştır. Ayrıca her konu için farklı bir nesne tanımı yapılması ve nesne sınıflarında tanımlı değişkenlerin ihtiyaçlar doğrultusunda değişmesi durumunda istemci dışında bir yerde kod düzeltmesi yapılması engellenmiştir. Her konu için farklı bir ayrıştırma yöntemine ihtiyaç duyulmadan tek bir yorumlama mekanizması üzerinden sonuca ulaşılması da kullanılan yöntemin önemli bir getirisidir.

Kapsam olarak çalışmada, konu ve içerik tabanlı iletişimlerle ilgili kaynaklarda anlatılan işleyiş şekilleriyle kısıtlı kalmayarak incelenen kaynaklarda uygulamasına rastlanmayan delegasyon yani yetki dağıtım yapısı da oluşturulmaya çalışılmıştır. Daha çok hiyerarşik yapılarda uygulanması mümkün olan bu işleyiş ile, mesaj akışına sadece yetkin etmenlerin müdahale etmesi istenen sistemler ile erişim yetkilerinin kısıtlı olduğu alt yapılarda sürece katkı sağlanmaya çalışılmıştır.

İleriki çalışmalar yapıdaki önemli unsurlar olan yetki dağıtım mekanizması ve fonksiyon tabanlı işleyiş üzerine olabilir. Bu kapsamda hiyerarşik yapılarda uygulanmaya daha yatkın yetki dağıtım mekanizmasında bir takım roller tanımlanıp bu rollere ilişkin yetkiler belirlenebilir. Kayıt modelinin belirlenen yetki ve roller üzerinden işlemesi sağlanıp, kimlik denetleme ve yetkilendirme standartları tanımlanabilir. Bu sayede sistem yönetimi tarafından hangi etmenin ne tür konularda ne kadar yetkisi olacağının belirlenmesi gibi daha etkin bir etmen idaresi sağlanabilir. Fonksiyon yapısında da yeni operatörler tanımlanarak daha kısa ifadeler oluşturulması sağlanabilir.

Bunlara ek olarak kayıt saklama yapısında da geliştirmeye gidilebilir. Bu kapsamda ana ve detay olmak üzere iki tablo tutulup her ana kayıt için detay tabloda birden fazla davranış tanımlanmasına imkan sağlanabilir.

KAYNAKLAR

[1] **Bellifemine, F. and Caire, G. and Trucco, T. and Rimassa, G.,** 2007. JADE Programmer's Guide, *Telecom Italia,* Italy.

[2] **Caire G.,** 2003. JADE Programming for Beginners, *Telecom Italia,* Italy.

[3] http://jade.tilab.com/ / Ocak 2007.

[4] http://www.fipa.org/ / Ocak 2007.

[5] **Araújo, F. and Rodrigues, L.,** 2002. On QoS-Aware Publish-Subscribe, *Universidade de Lisboa,* Portugal.

[6] **Baldoni, R. and Contenti, M. and Piergiovanni, T.S. and Virgillito, A.,** 2001. Modelling Publish/Subscribe Communication Systems: Towards a Formal Approach, *Università di Roma,* Italy.

[7] **Mahambre, P.S. and Kumar, M. and Bellur, U.,** 2006. A Taxonomy and Classification of Adaptive Event Based Middleware with Support for Service Guarantees, *IIT Bombay,* India.

[8] **Virgillito, A.,** 2003. Publish/Subscribe Communication Systems: from Models to Applications, *PhD Thesis,* Universitá di Roma, Roma.

[9] **Querzoni, L.,** 2007. Publish/Subscribe Communication Infrastructures, *Seminari di Sistemi Distribuiti,* Universitá di Roma, Italy.

[10] **Meier, R. and Cahill, V.,** 2005. Taxonomy of Distributed Event-Based Programming Systems, *University of Dublin,* Ireland.

[11] **Eugster, P. and Felber, A.P. and Guerraoui, R. and Kermarrec, M.A.,** 2003 The Many Faces of Publish/Subscribe, *ACM Computing Serveys,* **35(2)**, s.114-131.

[12] http://www.tenouk.com/ / Ocak 2007.

ÖZGEÇMİŞ

Manisa Akhisar doğumlu olan Kıvanç Özbek, ilköğrenimini Akhisar Misak-ı Milli ilkokulunda tamamlamıştır. Ortaöğrenimini 1995 yılında Akhisar Anadolu Lisesi'inde tamamlamış ve lise seviyesindeki eğitimine Eskişehir Fatih Fen Lisesi'nde devam etmiştir. Liseyi 1998 yılında İzmir'de Bornova Mustafa Kemal Lisesi'inde tamamladıktan sonra lisans eğitimine İstanbul Teknik Üniversitesi'nde başlamıştır. Lisans eğitimini 2003 yılında tamamlamış olup halen İstanbul Teknik Üniversitesi'nde yüksek lisans eğitimine devam etmektedir.