İSTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ

MOBİL AYGITLAR ÜZERİNDE KİŞİSELLEŞTİRİLMİŞ REKLAM İÇİN ETMEN TABANLI ÇERÇEVE TASARIMI

YÜKSEK LİSANS TEZİ

Resul ŞAHİN

Bilgisayar Bilimleri Anabilim Dalı

Bilgisayar Bilimleri Programı

OCAK 2013

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ

MOBİL AYGITLAR ÜZERİNDE KİŞİSELLEŞTİRİLMİŞ REKLAM İÇİN ETMEN TABANLI ÇERÇEVE TASARIMI

YÜKSEK LİSANS TEZİ

Resul ŞAHİN
(704071015)

Bilgisayar Bilimleri Anabilim Dalı

Bilgisayar Bilimleri Programı

Tez Danışmanı: Prof. Dr. Nadia ERDOĞAN

OCAK 2013

İTÜ, Bilişim Enstitüsü'nün **704071015** numaralı Yüksek Lisans Öğrencisi **Resul ŞAHİN**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"MOBİL AYGITLAR ÜZERİNDE KİŞİSELLEŞTİRİLMİŞ REKLAM İÇİN ETMEN TABANLI ÇERÇEVE TASARIMI"** başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Prof. Dr. Nadia ERDOĞAN** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. A. Çoşkun SÖNMEZ** Yıldız Teknik Üniversitesi	.....
	Doç. Dr. D. Turgay ALTILAR İstanbul Teknik Üniversitesi	.....

Teslim Tarihi : **17 Aralık 2012**
Savunma Tarihi : **24 Ocak 2013**

Aileme,

ÖNSÖZ

Öncelikle tez süresince yardımlarını ve desteklerini esirgemeyen danışmanım, Prof. Dr. Nadia ERDOĞAN'a içten teşekkürlerimi sunarım. Tez süresince karşılaştığım problemlerle sürekli yakından ilgilenerek yönlendirmeleri ile bana yol gösterdi.

Ayrıca tez süresince benden hiçbir yardımlarını esirgemeyen aileme şükranlarımı sunarım.

Ocak 2013

Resul ŞAHİN
(Bilgisayar Mühendisi)

İÇİNDEKİLER

Sayfa

KISALTMALAR

AD	**:** Advertisement
ADS	**:** Advertisements
AID	**:** Agent Identifier
EMAIL	**:** Electronic Mail
FIPA	**:** The Foundation of Intelligent Physical Agents
IMBOX	**:** Intelligent Mail Box
JADE	**:** Java Agent Development Framework
JVM	**:** Java Virtual Machine
LEAP	**:** Lightweight Extensible Agent Platform
LHS	**:** Left Hand Side
OKBC	**:** Open Knowledge Base Connectivity
RHS	**:** Right Hand Side
SMS	**:** Short Message Service
UI	**:** User Interface

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

MOBİL AYGITLAR ÜZERİNDE KİŞİSELLEŞTİRİLMİŞ REKLAM İÇİN ETMEN TABANLI ÇERÇEVE TASARIMI

ÖZET

Günümüz dünyasında mobil aygıtlar günlük hayatın vazgeçilmez parçalarından biri olmuştur. Bu vazgeçilmez cihazlar yardımıyla iletişim gereksinimlerimizin karşılanması yanında günlük hayatımızı kolaylaştıran bir çok özelliğe sahip oluyoruz. Bunlar internete erişim, toplantıların düzenlenmesi, konum bilgisi ve bellirli bir amaça yönelik uygulamalar olarak sıralanabilir.

Bir ürünün veya hizmetin tanıtımı reklamlar aracılığıyla yapılır. Yapılan reklamlar yardımıyla müşteri adayları ürünler veya hizmetler hakkında bilgi sahibi olurlar. Kurumsallaşmanın ve markalaşmanın giderek önem kazandığı günümüzde firmaların reklama daha fazla önem vermelerini doğurmuştur. Fakat, halen reklam geri dönüş oranları düşüktür. Çünkü, reklamların yayınlanmasında kullanılan yaklaşımlar ve araçlar, reklamların hedef kitleye tam anlamıyla ulaştırılmasını sağlamaktan uzaktır. Çoğu zaman firmalar, iletişim bilgilerini aldıkları müşterilerinin tümüne SMS veya e-posta yoluyla reklamlarını ulaştırmaktalar. Bu durum, reklam maliyetini artırdığı gibi müşteriler açısından ise sürekli bir reklam bombardımanına maruz kalmayı doğurmuştur.

Bu araştırmada, mobil aygıtlar üzerinde etmen tabanlı bir çerçeve geliştirilerek kişiselleştirilebilir bir reklam servisi gerçeklenmiştir. Böylece, kişiler sadece ilgilendikleri ürünler veya hizmetler hakkında bilgilendirilecektir. Bu sayede, ilgilenilmeyen reklamların alınması önlenerek kişinin zamanı korunacak ve alınmış olan reklamlara karşı farkındalık ise artırılacaktır. Reklam verenler açısından ise tasarlanan çerçeve ile reklamlar daha düşük maliyet ile hedef kitleye ulaştırılacaktır.

Mobil reklamcılıkta başarı oranını artırmak için kullanıcının tercihlerine uygun reklamların gösterilmesi gerekmektedir. Bu süreç iki ana parçadan oluşmaktadır. İlk olarak, kullanıcı tercihlerini reklam domain bilgilerini(kampanya başlangıç/bitiş tarihi, indirim oranı, vb.) de kullanarak ifade edilmesidir. İkinci olarak, tüm reklamlar arasından kullanıcı tercihlerine uygun olanlarının belirlenmesi gerekmektedir. Fakat, bu süreç oldukça dinamik bir yapı arz etmektedir. Çünkü her bir kullanıcının ilgilendiği reklamlar farklıdır. Ayrıca, reklamlar metin tabanlı olduğu için kullanıcı tarafından ayrı ayrı okunup değerlendirilmelidir. Dolayısıyla bu süreç reklam verenler açısından hedef odaklı olmaktan uzak olduğu gibi, kullanıcılar için ise vakit alan bir aktivitedir. Bu çalışmada kişiye özel reklamların belirlenebilmesi için etmen tabanlı bir çerceve geliştirilmiştir. Böylelikle, reklam verenler için hedef odaklı, kullanıcılar için ise vakit kazandıran bir model sunulmuştur.

Her bir reklam, belirli bir ürüne veya hizmete ait olacağından reklamlar arasında bir standardizasyon yapmak mümkün olmayacaktır. O halde yapılması gereken reklamin ait olduğu ürün/hizmet alanının kavramsal modelini çıkarmaktır. Bir kere çıkartılacak olan bu kavramsal model bu alan(domain) ile ilgili tüm reklamlarda kullanılabilinecektir. Böylece, metin tabanlı reklamlar içerisinde kavramsal model

kullanılarak daha hızlı ve hedef odaklı filitreleme yapılabilinecektir. Çoğu reklam filtreleme çalışmalarında metin tabanlı olan bilgi filtreleme yaklaşımları kullanılmaktadır. Ama bu çalışmada semantic web ile ortaya çıkan ontology yaklaşımı kullanılarak filtreleme yapılacaktır.

Araştırmada tasarlanan çerçeve güncel teknolojiler kullanılarak gerçeklenmiş ve test edilmiştir. Geliştirilen sistem Uzman (Expert) Sistemler ve Etmen (Agent) Sistemlerin yaklaşımlarını bir arada kullanmaktadır. Sistem Android işletim sistemine sahip mobil cihazlar üzerinde test edilmiştir.

Geliştirilen sistem sayesinde mobil cihaz sahipleri, tercihlerini kural tabanlı olarak değerlendirip uygun reklamları kendilerine sunan bir hizmete kavuşmuş oluyorlar. Böylece kullanıcılar kendileri için daha doğru reklamlara hızlı bir şekilde erişebilirler. Ayrıca, sistem kullanıcıların reklamlar arasında kaybolmasını önleyerek hem kullanıcıların zamanını hem de internet bant genişliğini korumaktadır.

AN AGENT-BASED FRAMEWORK FOR PERSONALIZED ADVERTISEMENT ON MOBILE DEVICES

SUMMARY

Mobile devices have become an essential part of our everyday life. We have not only satisfy communication needs but also we have lots of features which simplify our lives. These can be listed as Internet access, meetings organization, location information and applications for a specific purpose.

Advertisements are used for the promotion of a product or service. Customers acquire information about the product or service by advertising. Today, firms give more attention to the advertising. Because, institutionalization and branding have gained importance day by day. However, the rate of return on advertising is still lower than expected. Because, approaches and tools used in the publication of ads are not sufficient for reaching the target audience. Most of the time, A company sends bulk messages to all customers via SMS or email. This situation increases the cost of advertising for ads providers. On the other hand, a customer is exposed to advertising bombardment by the companies in every day.

In this research, An ads service for personalized advertising is implemented on mobile devices which is based on agent based framework. Thus, the person will only be informed about products or services which is interested. This approach saves the person time by preventing receipt of the uninterested ads and provides more awareness to the incoming ads. In addition, the ads providers reach target audiences with lower costs by the developed system.

Ads should be presented according to preferences of the user to improve the rate of return on mobile advertising. This process consists of two main parts. First, user preferences are expressed using advertising domain information such as campaign start/end date, discount rate, product/service properties and so on. Second, the appropriate ones to user preferences must be determined from all the ads. However, this process is of a highly dynamic structure. Because, each user is interested in different ads. In addition, users have to read to evaluate separately each ad because of text-based nature of ads. Therefore, this process is far from being goal-oriented in terms of advertisers, as well as the time-consuming activity for users. In this study, an agent-based framework has been developed in order to identify the most suitable ads that can be interested by the user. Thus, the proposed framework is not only goal-oriented for advertisers but also provides timesaving for users.

A person can be interested in many domains permanently or temporarily. In addition to this, number of domain providers varies greatly from small to large. Therefore, if the person is interested in a domain, he/she needs to register/investigate all providers' sites. This task is so time consuming and requires sharing personal information with each providers if the person wants to be up to date with future ads. Moreover, the person is always exposed to information mass from providers via SMS or email. It is

a psychological point of view. When the person receives more information, each information is gained less attention by the person.

Each ad is for a specific product or service, so it will not be able to make standardization between the ads. Representing a conceptual model of the product/ service that it belongs can be a method of standardization of the domain. Once the conceptual model is defined, all of the ads related with that domain can use the same model. Thus, text-based ads by using the conceptual model can be done faster and more goal-oriented filtering. Most of the studies on the ad filtering used text-based information filtering techniques. However, in this study, an ontology-based technique is used for ad filtering.

Semantic web is a set of paradigms and new technologies about the web. Semantic web does not intend to replace current web, it tries to complete the current web. Semantic web is a new research area so many researchers working about it. Some of the research areas are that; proposing new standards, developing tools, data mining, knowledge managements and web services. Ontology is one of the key paradigms in semantic web. Ontology formally represents knowledge as a set of concepts within a domain, and the relationships between those concepts. It can be used to reason about the entities within that domain and may be used to describe the domain.

Each ad belongs to the domain will be expressed in terms of ontologies in the developed system. The method of determining the ads that meet the user's preferences is also another important point. An ad can be discovered with writing code whether or not it is appropriate the user's preferences. However, writing a piece of code for each ad has negative effects on system flexibility, usability and extensibility. Therefore, an expert system designed to determine the appropriate ads considering the negative effects on system. Expert systems are widely used in industry because of cost-effective and fast responsiveness nature.An expert system can apply a domain if the domain knowledge can be modeled by the means of rules. Modeling domain knowledge requires co-working with domain experts and software engineers/analysts. When domain modeling is completed, it is easy update, addand delete defined rules on the fly.

The system is implemented on JADE(Java Agent Development Framework) which is fully compliant with FIPA specifications and has support for platforms with limited resources, such as mobile devices. To perform pattern matching, a rule-based system Drools is chosen, which is open source Java implementation of Rete algorithm.

Mobile agent is implemented on Android using mobile version of JADE, which is called JADE-Leap. All ad content/ knowledge are represented with ontologies. Client or subscriber fills its interests using ontology-based forms where the forms are dynamically generated from the ontology.

Two type of agents implemented in this work called client agent and server agent. A client agent is owned by specified user and runs on mobile device, which has Android operating system. In general, client agent is an Android application developed with JADE-Leap library. Server agent runs on the server and serves notification service to clients in accordance with the client's preferences. A notification is sent to the client when there is a matching between a product/service and client's preferences in all ads. Sending notification may be made instant or predefined periods.

The developed system was tested on a sample scenario. According to sample scenario, three airline companies announced a set of different advertisements on their websites. The target is that; notify the clients when a campaign is found according to their preferences. Clients can change their preferences over time.

Firstly, the ontologyis developed for flight domain in sufficient depth and features. There are two parts of the system: the client side and the server side. First, an android application that is augmented JADE-Leap libraryis developed on client side. The application has an abilityto generate user interface dynamically from the domain ontology class. Second, server agent is implemented which performs pattern-matching algorithms to determine ads which meets the user's preferences.

Developed system is also extensible with new ads. There are two tasks for adding new domains to the system. First task is defining the new ontology for the target domain. The second task is to write Drools rules, and then adding them to knowledge base of the system. The system offers a flexible, extensible framework using the expert systems and the agent-based technologies.

In conclusion, using a rule-based approach in the developed system, the mobile device owners will have the customizable service, which discovers tailored ads for their preferences. That allows clients to quickly access to the ads, which are more appropriately for themselves. The system also prevents the users from being lost among ads. In addition, saves users time and bandwidth in the ad discovery process.

1. GİRİŞ

Günümüz dünyasında mobil aygıtlar günlük hayatın vazgeçilmez parçalarından biri olmuştur. Bu vazgeçilmez cihazlar yardımıyla iletişim gereksinimlerimizin karşılanması yanında günlük hayatımızı kolaylaştıran bir çok özelliğe sahip oluyoruz. Bunlar internete erişim, toplantıların düzenlenmesi, konum bilgisi ve bellirli bir amaça yönelik uygulamalar olarak sıralanabilir.

Mobil aygıtların sayısının giderek artması mobil reklamcılığın öneminin artmasına sebep olmuştur. Mobil reklamcılıkta başarı oranını artırmak için kullanıcının tercihlerine uygun reklamların gösterilmesi gerekmektedir. Ama kullanıcı tercihlerinin tam olarak bilinememesi, her bir reklamın ilgili olduğu alanın farklı olması ve reklam içeriğinin metin tabanlı olması gibi sebeplerden ötürü kullanıcın tercihlerine göre bir filtreleme mümkün olamamaktadır. Dolayısıyla, günümüzde bir çok firma hedef müşterilerine erişirken ya tüm müşterilerine ya da müşterilerin yaş, cinsiyet veya kategorik tercihlerine göre SMS veya e-posta yardımıyla reklamlarını duyurmaktadırlar. Ama böyle bir reklam yaklaşımı istenilen seviyede bir geri dönüş oranı sağlayamamaktadır. Çünkü, her bir kullanıcının ilgisini çeken reklamlar farklı olduğu gibi bu durum bir kullanıcı açısından zaman ve yere bağlı olarak da değişiklik göstermektedir. Ayrıca, çoğu reklam metin tabanlı olduğu için kullanıcı tarafından ayrı ayrı okunup değerlendirilmesi gerekmektedir. Çoğu zaman kullanıcılara farklı firmalar tarafından çok sayıda mesaj, e-posta gönderilmektedir. Kullanıcılar ise gelen bu mesajları genellikle okumadan silmektedir. Özetle, bu süreç reklam verenler açısından hedef odaklı olmaktan uzak olduğu gibi, kullanıcılar için ise vakit alan bir aktivitedir.

1.1 Tezin Amacı

Bu çalışmada, kullanıcıların sadece ilgilenebileceği reklamlar mevcutsa veya ilerde duyurulduğu zaman bilgilendirileceği bir sistem geliştirilmesi hedeflenmiştir. Böyle bir sistem sayesinde kullanıcılar gerçekten ilgilendikleri reklamları doğru yer ve

zamanda alacaktır.Ayrıca, reklam sağlayıcılar hedef odaklı reklam yayınlama imkanına sahip olacakları gibi kullanıcaların ise zamanı korunmuş olunacaktır.

Belirtilen amacı gerçeklemek için gerekli olan sistem tasarlanarak geliştirildi. Tasarlanan sistemde etmen(agent) tabanlı sistemlerin ve uzman(expert) sistemlerin yaklaşımları kullanılmaktadır. Sistem istemci ve sunucu olmak üzere iki kısımdan oluşmaktadır. Birincisi, kullanıcı isteklerini alıp sunucaya gönderen etmen tabanlı mobil istemci uygulamasıdır. İkinci olarak ise, mobil kullanıcıdan gelen istekler doğrultusunda kullanıcı tercihlerine uygun olan reklamların tespiti için örüntü eşlemesini yapan sunucu uygulamasıdır.

1.2 Literatür Araştırması

Kural tabanlı etmen sistemler ile ilgili yapılmış çeşitli araştırmalar vardır. Çalışmaların birinde [1], sağlık domaini için çok etmenli (multi-agent) ve kural tabanlı (rule-based) teknolojileri bir arada kullanılarak esnek ve genişletilebilir iş akışı otomasyon araç takımı geliştirilmiştir. Vogt'ın çalışmasında, sağlık domainin katılımcılarının her biri JADE etmeni olarak gerçeklenmiştir. Domaine ait kurallar ise Jess kural motoru [2]kullanılarak gerçeklenmiştir.

Vogt'ın çalışmasında vaka analizi HCF Hastanesinin laboratuvarlarında uygulanan iş akışları üzerinde yapılmıştır. Hastahaneye ait laboratuvarlar hemotoloji, kimya, mikrobiyoloji gibi çeşitli bölümlerden oluşmaktadır. Laboratuvar analiz sonuçları hazır olduğunda, laboratuvar sorumlusu isteği yapan doktor ile iletişime geçiyor. Her bir bölüm bu süreçi kendi başına yürütüyor. Vogt çalışmasında uygulanan bu süreçin etmen tabanlı olarak otonom yapılmasını sağlamıştır.

İş kuralları değiştikçe kod değişikliği gerektirmeyen bir sistem gerçeklemek için Vogt kural tabanlı teknoloji kullanmıştır. İş kuralları Jess kuralları olarak ifade edildiğinden bu durum sağlanmıştır. Çünkü Jess kuralları Jess motoru tarafından çalışma zamanında yorumlanır. Bunun bir sonuçu olarak daha esnek ve geliştirilebilir bir otonom yapı sağlanmış olmaktadır.

Örneğin, önceden Sağlık Bakanlığı belli özelliklere sahip bir vaka tespiti durumunda Bakanlığın uyarılması gerektiği konusunda hastahaneden bir istekte bulununca süreç tümüyle insan gücü ile yönetilmektedir. Ama Vogt'un çözümünde, böyle bir istek geldiğinde belirtilen özelliklere ait bir vaka Jess kuralı olarak tanımlanıp sisteme

eklenince süreç otomotik olarak yönetilebiliyor olmuştur. Böylece, gerçek hayatta karar adımları belirli olan bir süreç (vaka analizi) kural tabanlı bir yaklaşım ile çözülmüştür.

Filtreleme işleminde uygulanacak olan yönteme karar vermek için önce filtrelenecek veri analiz edilir. Eğer veri yapısal değilse bilgi filtreleme(information filtering) tekniklerinin uygulanması gerekir. Böyle bir filtrelemede filtreleme işlemi için belli anahtar kelimeler(keywords) kullanılır. Ontology ile iyileştirilmiş bilgi filtreleme çalışmasında[3] WordNet [4] ontolojisi kullanılmıştır. Bu çalışma ile Sim, arama moturundan dönen sonuçları WordNet ontolojisi kullanarak yeniden sıralayarak kullanıcılara kendi isteklerine göre web sayfaları arasında gezme imkanı sunuyor.

WordNet İngilizce için hazırlanmış bir sözcüksel veritabanıdır. Bu veritabanında İngilizce sözcüklerin eşanlamlı grupları, kısa yazılışları, genel anlamları ve eşanlamlı gruplar arasındaki anlamsal ilişkiler belirtilmiştir. WordNet'in asıl amacı otomatik text analizi ve yapay zeka uygulamalarında kullanılacak kelime sözlüğünü oluşturmaktır.

Chang ve Huo'nunmobil reklamcılık ile ilgili yapmış oldukları çalışmada [5] ise kullanıcılar için daha efektif reklam hizmeti sunmayı hedeflemişlerdir. Bu çalışmada araştırmacılar başarım için Bağlam (Context), İçerik (Content) ve Kullanıcı Tercihi (User Preference) olmak üzere üç faktöre odaklanıyorlar. Bağlam kullanıcının yer bilgisi, hava durumu, zaman ve hız gibi bilgileri içeriyor. İçerik faktörü ise, marka, fiyat, promosyon gibi bilgileri içeriyor. Son olarak, Kullanıcı Tercihi ise marka, ilgi alanları, kullanıcı aktiviteleri, mobil web gezintileri gibi bilgileri içeriyor. Bu bilgilerin bir arada değerlendirileceği bir platformun Online(Web) reklamcılığından daha etkin sonuçlar doğurabileceğini öngörmektedirler. Bu çalışmada, bilgi alma (information retrieval) teknikleri kullanılarak arama motorundan gelen sonuç kümesi üzerinde web sayfası içeriği ve sayfada bulunan reklam bilgileri çıkartılıyor. Her bir sayfa için web sayfasında adres bilgisi varsa kullanılmakta yok ise rasgele olarak bir adres Google Map API [6] ile üretilmiştir. Bu projede test ortamı yapılan bir uygulama ile simule edilmiştir.

Ben tez çalışmamda mobile reklamcılığın kişinin kendi güdümünde yapılmasını sağlayan bir model geliştirmeyi amaçladığım için reklam filtreleme sürecinde bilgi filtreleme (information filtering) yaklaşımından öte kural tabanlı bir yaklaşım

kullanmayı seçtim. Dolayısıyla mevcut problem bilgi filtreleme modelinden yayıncı/abone (publish/subscribe) modeline dönüşmektedir. Çünkü bir taraftan reklam sağlayıcılar birer veri sağlayıcısı (data provider) yani yayıncı (publisher) olurken, diğer taraftan reklamlar ile ilgilenen birer veri istemcisi yani abone (subscriber) oluyorlar.

Matheson yaptığı kural tabanlı yayıncı abone çalışmasında ölçeklenebilir ifade gücü (expressive) yüksek bir sistem tasarlamıştır [7]. Matheson'un tasarladığı sistem ticari bir sistem olan PADRES'e [8] entegre edilerek sınanmıştır. Bu sistem yayıncı ve abone bilgilerine göre yönlendirme işlemini yöneten bir broker uygulamasıdır. Bu çalışmada kural eşlemesi için Rete algoritmasının [9] bir gerçeklemesi olan Jess kullanılmıştır.

Başka bir çalışmada ise kural tabanlı bir sistem ve etmen tabanlı bir sistemin entegre edilerek uyum sağlayabilen ve akıllı sistemlerin gerçeklenmesi amaçlanmıştır [10]. Belirtilen bu çalışmanın temel felsefesi, sistemin mantık kısmını bir kural motoruna devretmektir. Avrupa Birliği tarafından desteklenen çalışma kapsamında, e-öğrenme sürecinin kişiselleştirilmesi hedeflenmiştir. Böylece kişiler ilgi alanlarına göre öğrenme metaryellerine aracı bir etmen yardımıyla hızlı ve etkin bir şekilde ulaşabileceklerdir.

2. ETMEN SİSTEMLER (AGENT SYSTEMS)

Bir yazılım etmeni (software agent) iş akış otomasyonunu sağlamak ve desteklemek için kullanılan uygun bir yaklaşımdır. İş sürecinde bulunan varlıklar (entities) otonom etmenler olarak temsil edilerek kendilerine verilen görevi başarmak için çevresindeki diğer etmenlerle iş birliği yapar [11]. Bölüm 2.1'de etmenin ne olduğu ve nasıl bir ortamda bulunacağı açıklanıyor. Bölüm 2.2'de etmenlerin sahip olması gereken özellikler anlatılıyor. Bölüm 2.3'teetmen haberleşmesinde merkezi bir rol üstlenen ontolojilerden bahsediliyor. Son olarak Bölüm 2.4'te ise Jade etmen çerçevesi anlatılıyor.

2.1 Etmen Nedir? Akıllı Etmen Ne Demektir?

Etmen'nin çeşitli tanımları mevcuttur.Bilgisayar bilimlerinde genel kabülüyle etmen (agent) özel bir yazılım bileşeni olup, otonomdur, istemcilerine servis verirken bir insan acentesi gibi davranır ve kendi ajandasını uygular [12].

Bu tanımların ortak noktası, etmenlerin otonom olmaları ve sistemdeki diğer etmenlerle birlikte çalışarak ortak bir hedefi gerçekleştirebilmeleridir.

Bu tez çalışmasında her bir kullanıcıyı temsil etmesi için bir etmen tasarlanmıştır. Bu etmen kişinin kendi cep telefonu üzerinde çalışan özel bir uygulamadır.

2.2 Etmenin Özellikleri

Wooldridge ve Jennings'e göre bir etmenin sahip olması gereken dört özellik şunlardır [13]:

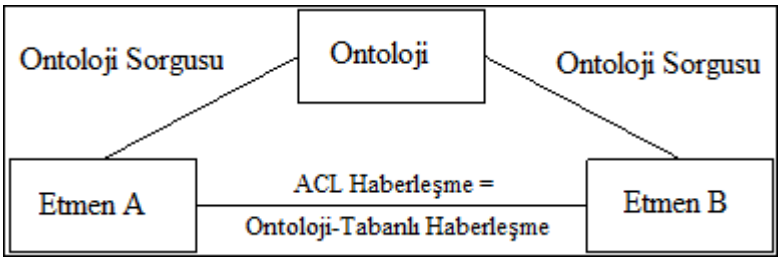
- Otonom(autonomy): Otonom bir etmen insan veya diğer etmenlerin müdahalesi olmadan kendi başına davranış sergileyebilir.
- Sosyal yetenek (social ability): Bir etmen diğer etmenlerle ve/veya insanlar ile haberleşir. Bu haberleşme kabul edilen etmen-haberleşme dilleri aracılığıyla yapılır.

- Reactivite (reactivity): Bir etmen çevresini algılar ve çevresindeki değişikliklere göre yeni davranışlar sergiler.
- Pro-aktiflik (pro-activeness): Etmen çevresinde herhangi bir değişiklik olmadan da belli bir amaç doğrultusunda davranışlara sahip olabilir.

2.3 Ontoloji (Ontology)

Etmenlerin bir birleriyle mesaj alış-verişinde bulunabilmeleri için etmenlerin alıp verdikleri mesajları anlayabiliyor olmaları gereklidir. Yani gönderilecek verilerin hangi formatta gönderileceği, alınan verinin neyi ifade ettiği gibi durumların ortak bir karar sonucu yapılması gereklidir. İşte bu nedenle etmen sistemlerde ortak bir haberleşme dili kullanılır. Bu ortak dil etmen sistemin uygulandığı alanın (domain) ontolojisidir.

Ontoloji [14] bir alan (domain) içerisindeki kavramların ve bu kavramların birbirleri arasındaki ilişkilerinin bilgisinin biçimsel olarak temsilidir. Şekil2.1'de iki etmenin ontoloji kullanarak nasıl haberleştiği temsil edilmiştir.



Şekil 2.1 : Ontoloji-tabanlı haberleşme modeli [15].

Çeşitli ontoloji dilleri geliştirilmiştir. Bunlardan biri ise FIPA-Meta-Ontology olarak anılan, Open Knowledge Base Connectivity'nin (OKBC) çerçeve tabanlı bilgi modelini (frame-based knowledge model) uyarlayan FIPA standardıdır. Bu standarda göre tasarlanan bir ontolojide üç önemli eleman vardır [16]:

- Concept: Concepts (kavramlar) sistemde var olan, etmenlerin konuştukları ve hakkında akıl yürüttüğü varlıklardır. Ör: (Person :name John :age 33)
- Predicate: Mevcut durum hakkında bir şeyler anlatır. Genellikle doğru veya yanlış olarak değerlendirilen ifadelerdir. Ör: (Works-for (Person :name John) (Company :name TILAB))

- AgentAction: Bazı etmenler tarafından yürütülebilecek aksiyonları gösteren ifadelerdir. Ör: (Sell (Book :title “The Lord of the rings”) (Person :name John))

FIPA standardına göre etmenler bir birleriyle ACL (Agent Communication Language) mesajlaşma dili ile mesajlaşırlar. Her bir ACL mesajı ontoloji tabanlı veri içerir. Mesajı alan alıcı etmen ACL mesajını parse ederek kullanılan ontolojiyi saptar. Alıcı etmen, saptanan ontolojiye göre alınan mesajı çözerek uygulama nesnesine dönüştürür.

2.4 Jade

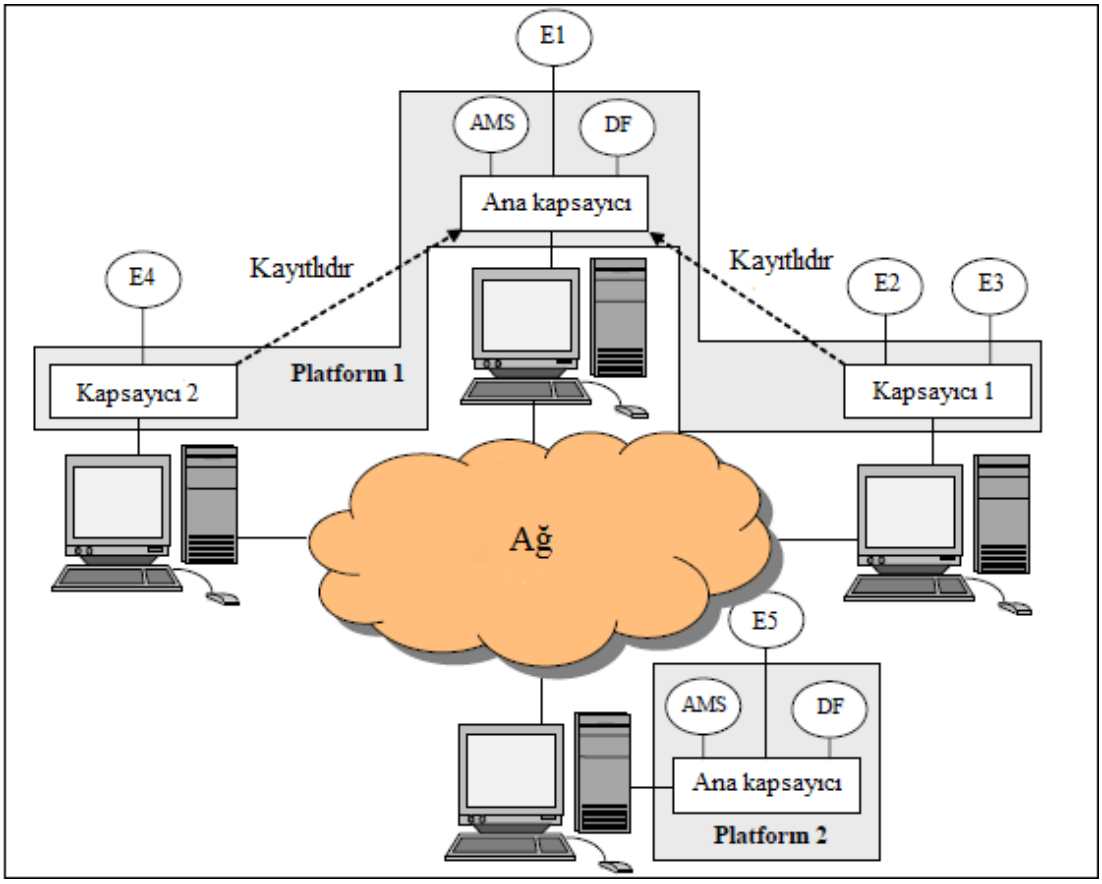
Java Agent Development Framework (JADE) Java ile yazılmış açık kaynak kodlu, tümüyle FIPA uyumlu, etmen sistemler için geliştirilmiş arakatmandır [17]. Bu proje Telecom Italia tarafından 1998 yılında geliştirilmeye başlanmış ve LGPL lisansı ile 2000 yılından itibaren Telecom Italia tarafından dağıtılmaktadır.

JADE belirli bir uygulamaya özel veya alan bağımlı değildir. JADE çok-etmenli paradigmasını baz alan ayrık uygulamalar için temel arakatman fonksiyonları sunar. Bir JADE ortamı aşağıdaki özelliklere sahiptir [12]:

- Etmenler otonom ve proaktiftir: Her bir etmen kendi yürütme ipliğine (thread) sahiptir ve nesnelerinin referanslarını diğer etmenlere göndermezler. Her bir etmen kendi hayat döngülerini kontrol ederler ve yürütülecek bir sonraki aksiyona otonom olarak karar verirler.
- Etmenler bir aksiyonu yürütmeyi reddedebilirler ve az bağımlıdırlar: JADE etmenleri asenkron mesaj tabanlı haberleşmeyi kullanır. Bir gönderici alıcının tekil adını kullanarak alıcıya gönderir. Mesajlar gönderilirken gönderici belirtilmeyebilir. Mesajlar bir group etmene doğrudan gönderilebileceği gibi uygun etmenlere göndermesi için tek bir vekil (proxy) etmene de gönderilebilir. Ek olarak, bir alıcı tercihlerine göre hangi mesajları işleme alacağına ve hangi mesajları göz ardı edeceğine otonom olarak karar verebilir.
- Sistem etmenler arası çalışır: Etmenler bir birlerini doğrudan adreslemek için tekil kimliklere (the AgentIdentifier (AID)) sahiptirler. Her bir etmen host platforma kendi isteklerine göre katılabilirler veya platformdan ayrılabilirler.

Etmenler diğer etmenleri host platform tarafından sunalan servisler (Agent Management System (AMS) ve Directory Facilitator (DF)) aracılığıyla arayabilirler.

Şekil 2.2'de iki platform ve beş etmenden oluşan örnek bir Jade çoklu etmen sistemi resmedilmiştir.



Şekil 2.2 : Örnek bir Jade etmen sisteminin mimarisi [18].

3. KURAL TABANLI SİSTEMLER (RULE-BASED SYSTEMS)

Kural tabanlı sistemler bellirli bir görev için problem çözen uzmanların bilgisini baz alarak akıllı davranışları modeller [19]. Kural tabanlı sistemler bir çok uygulama alanlarında kullanılmaktadır. Bu alanlara finans, ulaşım ve bilgi teknolojileri örnek olarak verilebilir. Örneğin, ağ konfigürasyonu,müşteri kredi ve risk yönetimi yazılımları kural tabanlı yaklaşımlar kullanılarak daha esnek bir yapıya dönüştürülebilirler.

Jess [2], Prolog [20], CLIPS [21] ve Drools [22] kural tabanlı sistem çerçevelerine birer örnektir. Bu çerçevelerin hepsinde örüntü eşlemeyi sağlayan Rete algoritmasının kendi gerçeklemeleri bulunur.

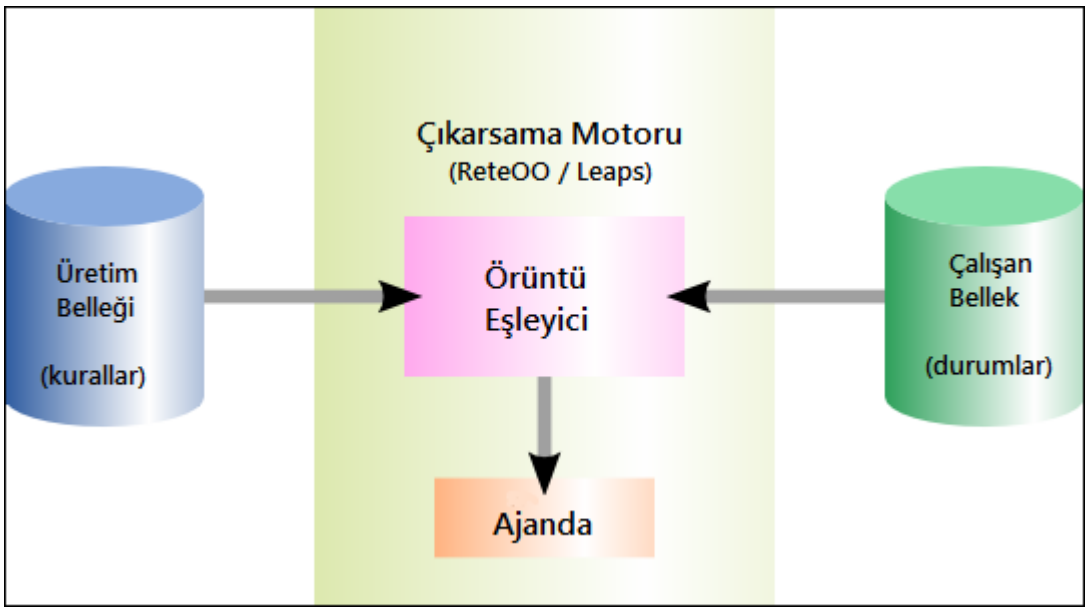
Bir uygulamada iş mantığı (business logic) sık değişiyorsa, böyle bir uygulamaya kural tabanlı bir sistemin entegre edilmesi sisteme esneklik ve performans kazandırır.

Kural tabanlı sistemlerde yazılan kurallar yorumlandığı için kurallar sisteme sistem çalışırken eklenebilir. Yani sistemin kodunun değiştirilip, derlenip, deploy edilip, yeniden çalıştırılması gibi vakit alan bir sürece gerek kalmaz.

3.1 Kural Motoru (Rule Engine)

Bu çalışmada Drools Rule Engine (Drools Kural Motoru) kullanılacaktır. Drools kural tabanlı yaklaşımları kullanarak uzman sistem gerçekleyen bir kural motorudur [23]. Drools gerçek hayattaki sistemlerde kullanılacak kadar gelişmiş ve kendini ispatlamıştır. Drools yüksek sayılarda kuralları (rules) ve durumları (facts) ölçekleyebilen bir Çıkarsama Motoruna (Inference Engine) sahiptir. Çıkarsama Motoru belirtilen kurallar doğrultusunda durumlar ile veriyi eşleyerek kurallarda belirtilen aksiyonu alır. Bu eşleme sürecine örüntü eşleme (pattern matching) denir.

Şekil 3.1'de bir kural motorunun yüzsek düzeyli görünümü verilmiştir.Çıkarsama Motorlarında örüntü eşleme için kullanılan bir dizi algoritma vardır. Drools projesinde, Rete algoritması geliştirilerek gerçeklenmiştir.



Şekil 3.1 : Bir Kural Motorunun yüksek düzeyli görünümü [23].

Drools nesne tabanlı sistemler için Rete algoritmasının daha iyileştirilmiş ve optimize edilmiş kendi gerçeklemesini ReteOO diye adlandırır.

Şekil 3.2'de gösterildiği gibi bir kural Sol Taraf (Left-Hand-Side (LHS)) ve Sağ Taraf (Right-Hand-Side (RHS)) olmak üzere iki taraftan oluşur.Sol taraf kuralın aktive olması için gerekli şartları belirten sırasız örüntülerdir. Sağ taraf ise kural aktive olduğunda yapılacak aksiyonları belirtir.

```
rule "kural-adı"
  when
    LHS     # kuralın aktive olmasi için gerekli şartlar
  then
    RHS     # kural aktive olduğunda yapılacak aksiyonlar
  end
```

Şekil 3.2 : Drools kural şablonu.

3.2 Kural Motorunun Avantajları

Bir uygulamada kural motoru kullanmanın avantajları şunlardır [23]:

- Bildirimli Programlama (Declarative Programming)

  Kural motorları bir şeyin "Nasıl yapılacağını" değil de "Ne yapılacağını" belirlerler. Kural motorlarının anahtar yararlarından biri, zor problemlere kolayca çözümler sunmalarıdır. Ve sunulan çözümler hemen doğrulanabilir. Ayrıca, kuralları okumak kodu okumaktan daha kolaydır.

Kural tabanlı sistemler çok çok zor problemlere de çözüm sunabilirler. Çözüme nasıl ulaştığını ve çözüm süresince alınan kararların açıklamalarını sağlarlar.

- Lojik ve Veri İzolasyonu (Logic and Data Seperation)

  Veri domain nesnelerinde, lojik ise kurallar içerisindedir. Bu temel olarak veri ve lojiğin nesne-yönelimli bağımlılığını engeller. Lojiğin kural dosyalarına izole edilmesi ile ilerde gerçekleşecek değişikliklerde bakımın daha kolay olması sağlanabilir. Eğer lojik bir çok domain nesnesine ve kontrollerine bağlı değilse, lojik bir ve daha fazla farklı kural dosyasında yönetilebilir.

- Hız ve Ölçeklenebilirlik (Speed ve Scalability)

  Rete algoritması ve onun halefi olan Drools ReteOO algoritması, domain nesne verileri ile kural örüntüleri arasında çok etkin bir eşleme sunar. Bu algoritmalar gerçek uygulamalarda kanıtlanmış bulunmaktadır.

- Bilginin Merkezileştirilmesi (Centralization of Knowledge)

  Kurallar ile kural motoru tarafından yürütülebilecek bilgi tabanı (knowledge base) oluşturulur. Bunun anlamı, tüm bilgilerin bir noktada toplanmasıdır. Örnek olarak, tüm iş poliçelerinin bir yerde toplanması verilebilir. Kurallar idealde daha okunabilir olduklarından kurallar dökümantasyon için de sunulabilir.

- Araç Entegrasyonu (Tool Integration)

  Eclipse gibi araçlar (ve gelecekte, Web tabanlı kullanıcı arayüzleri) düzenlemek ve kuralları yönetmek ve anlık geri bildirim, doğrulama ve içerik yardımı almak için yollar sağlar. Ayrıca, denetleme ve hata ayıklama araçları da bulunmaktadır.

- Açıklama İmkanı (Explanation Facility)

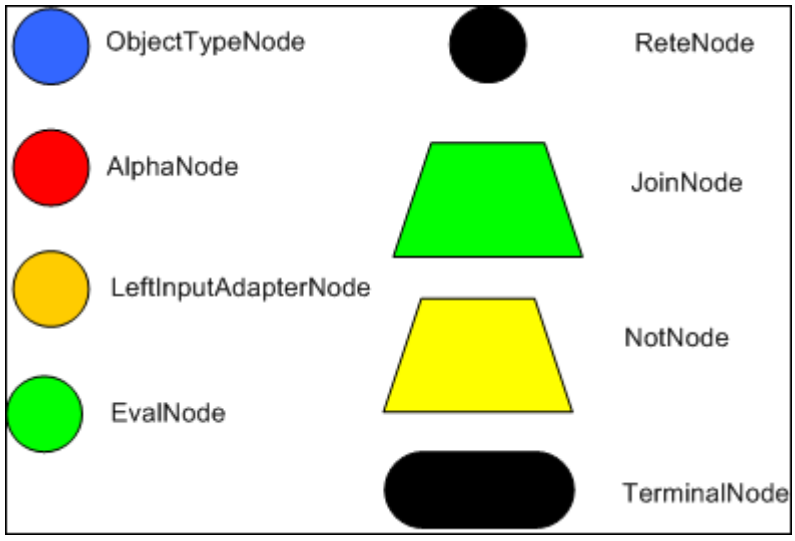
  Kural tabanlı sistemler kural motoru tarafından alınan kararların niçin alındığına dair loglama yaparak, kararların açıklama imkanını etkili bir şekilde sunar. Bu özellik, kural motorlarını kullanan sistemlerin geniş kitlelerce kabulünü kolaylaştırır.

- Anlaşılabilir Kurallar (Understandable Rules)

  Domain Özel Diller (Domain Specific Languages) nesne modellerini oluşturarak problem domainini modeller. Bu diller kullanılarak doğal dillere yakın bir şekilde kişi kendi kurallarını yazabilir. DSL'ler sayesinde domaini bilen teknik olmayan kişiler kendi ifade şekilleriyle domain kurallarını yazar.

3.3 Rete Algoritması

Rete algoritması Dr. Charles Forgy tarafından icat edilmiş ve 1978-79 yıllarında doktora tezinde sunulmuştur [23]. Latince bir sözcük olan "rete", "network" anlamına gelir. Rete algoritması iki parçaya ayrılabilir: kural derlenmesi (rule compilation) ve çalışma zamanı yürütme (runtime execution). Derleme algoritması sistemde bulunan kuralların nasıl verimli bir ayrık ağ (discrimination network) yani Rete Network'ü oluşturulacağını tarif eder. Bir ayrık ağ veriyi filtrelemek için kullanılır. Temel fikir, veri filtreleme işleminin ağ üzerinden yapılmasıdır. Ağın başında bir çok eşleme varken, ağ üzerinden aşağı doğru inildikçe eşleşme sayısı azalır. Ağın en alt düğümleri terminal düğümlerdir. Dr. Forgy dört temel düğümden bahseder: kök, 1-girdi, 2-girdi ve sonlanma. Şekil 3.3'te Rete ağı düğümleri verilmiştir.



Şekil 3.3 : Rete düğümleri.

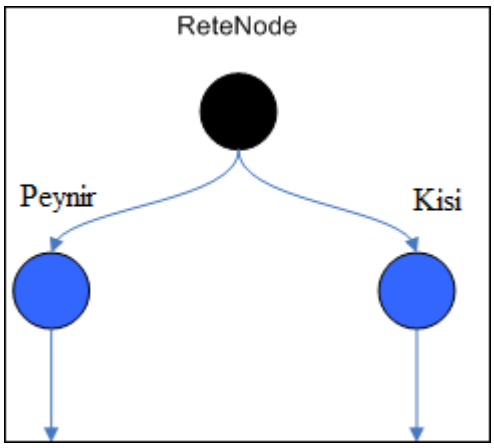
Sistemdeki kuralların tümü Kural Tabanı (Rule Base) oluşturur. Kurallar tarafından işlenen veri Çalışan Bellek (Working Memory) diye adlandırılan global veri içerisinde tutulur. Yorumlayıcı (interpreter) aşağıdaki işlemlere göre sistemi yürütür [23]:

1. Eşleme (Match): Çalışan bellek içeriğinde başarılan durumun saptanması için LHS'in değerlendirilir.
2. Çakışma çözüm (Conflict resolution): Başarılan LHS arasından birinin seçilmesidir. Eğer her hangi bir çakışma çözüm metodu bulunmazsa yorumlayıcı hata verir.
3. Aksiyon (Act): Seçilen kuralın RHS'i yani aksiyonları yürütülür.
4. 1. adıma git.

Rete ağı temel olarak Alpha and Beta networklar olmak üzere iki parçadan oluşmaktadır. Alfa (AlphaNodes) düğümleri iç-eleman koşulunu tanımlayan bir girdiye sahiptir. Beta (BetaNodes) düğümleri iç-eleman koşulunu tanımlayan iki girdiye sahiptir. Rete çapraz çarpım (cross product) hesaplaması için Beta düğümlerine birleştirme (join) uygular.

Rete ağı tüm nesnelerin girmesi gereken ReteNode adı verilen kök düğüm ile başlar. Bu düğüm sonra her bir nesne tipi (ObjectType) için ayrı düğümlere ayrılır. Yani birinci seviye ayrışma nesne tipine göredir. Her bir nesne tipi ayrışmasından sonra, bir veya daha fazla Alfa ayrık düğümlerine sahip oluyoruz. Her bir düğüm bir metinsel kısıtı uygular. Sonra olarak BetaNode kural içerisinde olası çapraz çarpım durumlarını saptar.

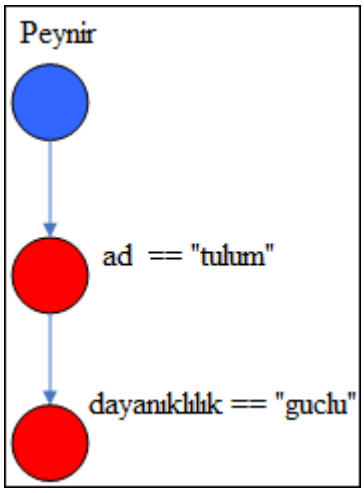
Şekil 3.4'te verilen kök düğüm olan ReteNode kural koşullarında bulunan iki nesne tipine göre ayrılmıştır. Mavi renkli olan iki düğüm de nesne tipinde (ObjectTypeNode) düğümlerdir.



Şekil 3.4 : Nesne tipinde olan düğümler (ObjectTypeNodes) [23].

Rete algoritmasına göre bir ObjectTypeNode'u AlphaNode'ları, LeftInputAdapterNode'ları veya BetaNode'ları takip edebilir [23]. Alfa düğümleri (AlphaNodes) metinsel koşulları değerlendirmek için kullanılır. Örnek olarak, Kisi.ad == "Resul" bir metinsel koşuldur.

Bir kural bir nesne tipi için birden fazla literal koşula sahip ise, koşulların tamamı ayrı ayrı Alfa düğümleri halinde bağlanır. Bunun anlamı, eğer bir uygulama Kisi nesnesini doğrularsa, bir sonraki alfa düğüm koşulunu doğrulamaya çalışır. Böylece Reta ağı üzerinde filtreleme yapılmış olunur.Örnek olarak Şekil 3.5'te Peynir nesnesi için iki adet alfa düğümü verilmiştir. Yani nesne üzerinde iki tane filtre mevcuttur.



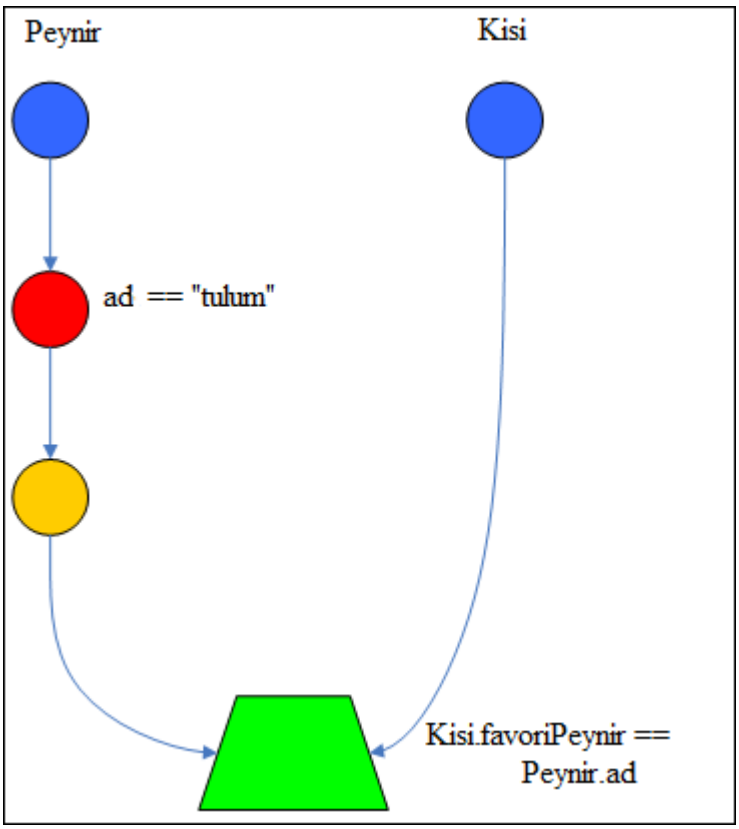
Şekil 3.5 : Alfa düğümleri (AlphaNodes) [23].

JoinNode ve NotNode olarak adlandırılan iki adet iki-girdi alan düğüm vardır. Bu düğümlerin tipi BetaNode'dır. BetaNode'ları iki objenin alanlarının birbirleriyle karşılaştırılması için kullanılır. Objeler aynı veya farklı tiplerde olabilirler. Şekil 3.6'da Kisi ve Peynir nesnelerinin farklı alanlarının (favoriPeynir ve ad) bir biriyle karşılaştıran bir JoinNode görülmektedir.

Peynir durumunda olduğu gibi ilk objeyi etkinleştirmek için, ağa bir LeftInputNodeAdapter ekliyoruz. Bu adaptör bir objeyi girdi olarak alır ve tek bir obje demeti (tuple) üretir. Bu adaptör düğüm obje karşılaştırmasında kullanılacaktır. Şekil 3.6'da turuncu renkli düğüm bir LeftInputNodeAdapter örneğidir.

Terminal düğümleri tek bir kuralın tüm koşullarının eşleştiğini göstermek için kullanılır. Bu durumda kural tüm eşleşmeye (full match) sahip olur. Bir kural "veya" koşullu ayırıcıya sahipse olası her mantıksal dal için alt-kural oluşturulur. Bu

nedenle, bir kural birden fazla terminal düğüme sahip olabilir. Bir kural oluşturulan alt-kuralların herhangi birinin eşlenmesi ile aktive olabilir.



Şekil 3.6 : Birleşme düğümü (JoinNode) [23].

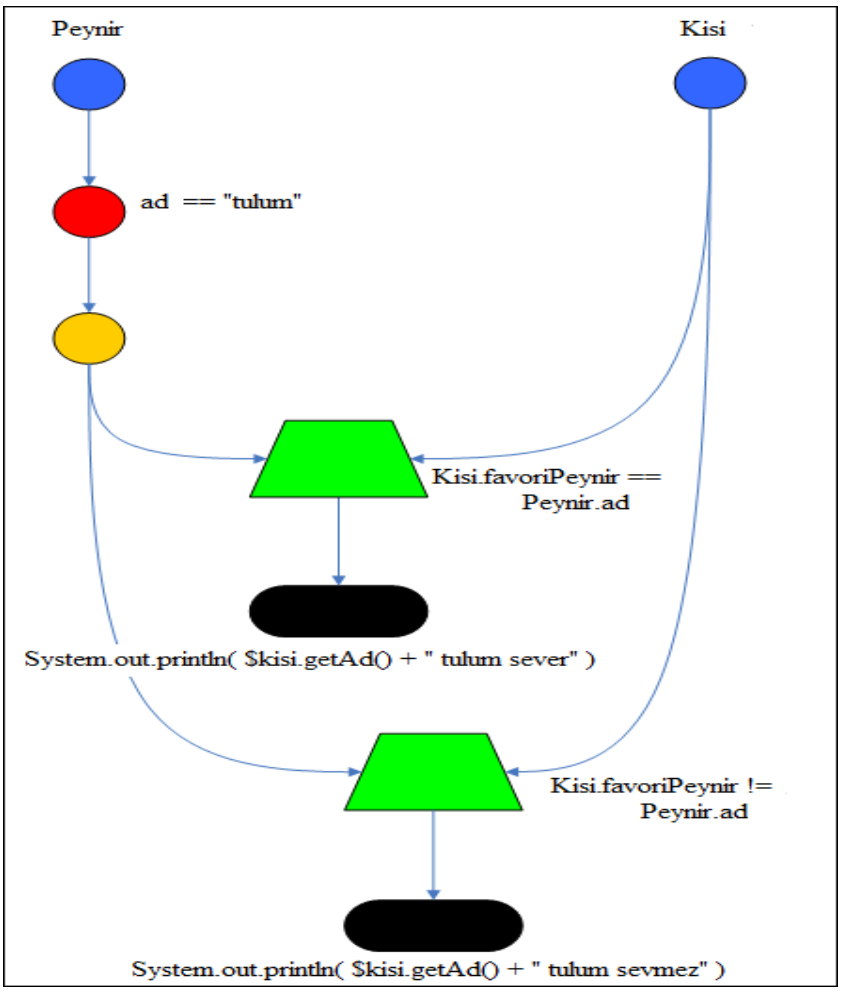
Drools kural motoru düğüm paylaşımı da gerçekleştirir. Bir çok kural aynı örüntüyü tekrarlar ve düğüm paylaşımı tekrarlanan bu örüntüleri daraltmamıza izin verir. Böylece, tekrarlanan örüntüleri her tekil örnek için tekrar değerlendirmek gerekmez. Şekil 3.7'de iki kuralın ilk örüntüleri ortak olduğundan düğüm paylaşımı yapılmıştır. Düğüm paylaşımı yapılarak tekrarlı karşılaştırmalar engellendiği için performans kazanılır.

```
rule
   when
     Peynir( $tulum : ad == "tulum" )
     $kisi : Kisi( favoriPeynir == $tulum )
   then
     System.out.println( $kisi.getAd() + " tulum sever" );
   end

rule
   when
     Peynir( $tulum : ad == "tulum" )
     $kisi : Kisi( favoriPeynir != $tulum )
   then
     System.out.println( $kisi.getAd() + " tulum sevmez" );
   end
```

Şekil 3.7 : İki Drools kuralı [23].

Şekil 3.7'de verilen iki Drools kuralından derlenerek oluşturulmuşRete ağı Şekil 3.8'de gösterilmektedir.



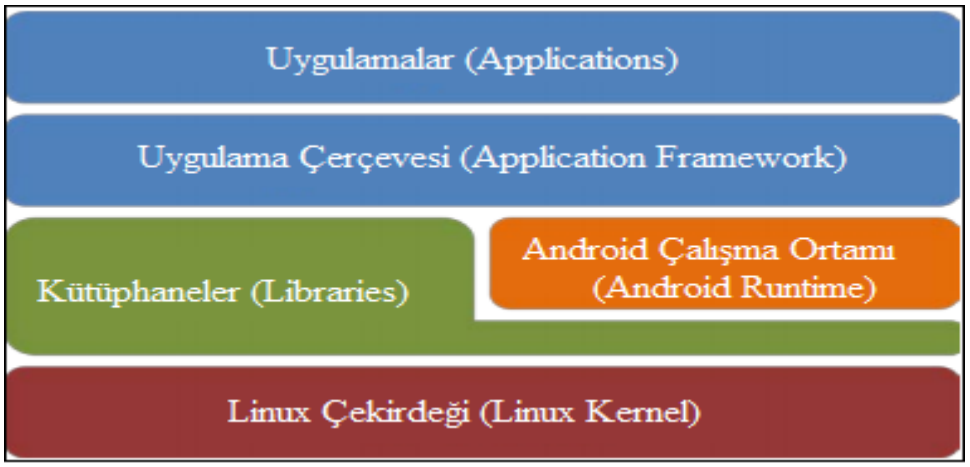
Şekil 3.8 : İki Drools kuralının oluşturduğu Rete ağı [23].

Ağda alfa düğümler paylaşılmış, fakat beta düğümler paylaşılmamıştır. Her bir beta düğüm kendi terminal düğümüne sahiptir. İlk örüntünün (Peynir($tulum : ad == "tulum"))iki kural için de paylaşıldığı görülmektedir.

4. ANDROID İŞLETİM SİSTEMİ

Android öncelikle akıllı telefonlar ve tablet bilgisayarlar gibi dokunmatik ekranlı mobil aygıtlar için tasarlanmış Linux tabanlı işletim sistemidir. Android başlangıçta Google tarafından finansal olarak desteklenen Android, Inc. firması tarafından geliştirildi. Daha sonra bu firma Google tarafından satın alınmıştır. Mobil aygıtlar için açık standartların gelişmesini hedefleyen Open Handset Alliance kurulması ile Android 2007 yılında tanıtıldı. İlk Android kullanılan telefon Ekim 2008'de satıldı [24].

Android açık kaynak kodludur ve Google tarafından Apache Lisansı ile dağıtılmaktadır. Android'in açık kaynak kodlu olması ve liberal bir dağıtım lisansına sahip olmasından ötürü aygıt üreticileri tarafından özgürce değiştirilip dağıtılmasına izin verilir. Ayrıca, Android geniş bir geliştirici topluluğuna da sahiptir. Aygıtın fonksiyonelliğini artıran uygulamalar, Java programlama dilinin özelleştirilmiş bir versiyonunda yazılır. Şekil 4.1'de Android platformunun katmanları görülmektedir.



Şekil 4.1 : Android platformunun mimarisi.

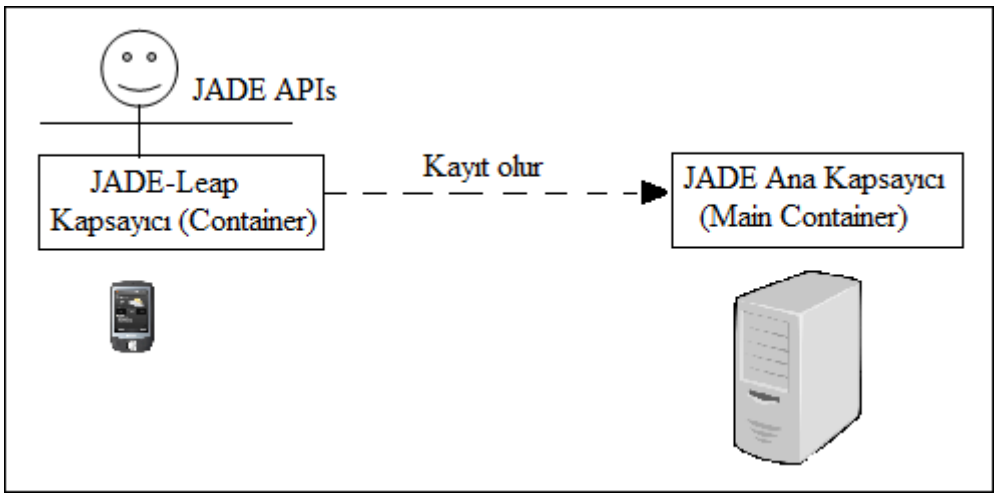
Android dünyanın en çok kullanılan akıllı telefon platformlarından biridir. Android dünya geneli akıllı telefon pazarında 2012 3. çeyreğinde 500 milyon aktive edilmiş aygıtla %75'lik bir paya sahiptir [25].Açık kaynak kodlu projelerin çokluğu ve en yaygın mobil platform olduğu için Android platformunu çalışmamda tercih ettim.

Bu tez çalışmasında kullanıcılar birer etmen yardımıyla sisteme entegre olacaklar. Bu etmen kişinin her an ulaşabileceği, kişi hakkında bilgilere sahip olacağı ve kişinin tercihlerini yöneteceği bir uygulama olacaktır. Özetle, bu etmen kişinin sahip olduğu mobil aygıtı üzerinde koşacak olan bir Android uygulamasıdır.

Geliştirilen uygulamanın bir etmen gibi davranabilmesi için JADE projesinin limitli kaynaklara sahip aygıtlar üzerinde koşan minimize edilmiş versiyonu olan JADE-Leap kütüphanesi kullanılmıştır. Bu kütüphane dağıtık altyapıya entegre olabilmek için gerekli ara katman sınıflarını içermektedir. Bu kütüphane kullanılarak geliştirilen mobil uygulama dağıtık altyapıya katılarak sistem üzerinde bir etmen niteliği taşır.

Android platformunda arayüz geliştirmek için "Activity" sınıfı kullanılmaktadır. Bu sınıf türetilerek istenilen ekranlar oluşturulur. Her bir ekran için bir "Activity" sınıfı kullanmak iyi tasarım alışkanlığı olarak kabul edilir. Tez çalışmasında tasarlanan sistemin olabilirliğini kanıtlamak için gerekli arayüzler geliştirilmiştir.

JADE asenkron haberleşmeyi temel olarak aldığı için mesajlar geri çağırmalar (callbacks) ile uygulamada yakalanır. JADE-Leap mobil platform üzerinde bir kapsayıcı (container) oluşturur. Aslında, bu kapsayıcı bir Android "Service" olarak gerçeklenmiş olup dağıtık altyapı hizmetlerini sunmaktadır. Oluşturulacak etmen tek bir iplik (thread) olarak bu kapsayıcı içerisinde barınır. Bu durum Şekil 4.2'de resmedilmiştir.



Şekil 4.2 : JADE-Leap çalışma görünümü.

5. MOBİL AYGITLAR ÜZERİNDE KİŞİSELLEŞTİRİLMİŞ REKLAM İÇİN ETMEN TABANLI ÇERÇEVE TASARIMI

Bu tez çalışmasının konusu günlük hayatta çoğumuzun karşılaştığı ve karşılaşmakta olduğu bir problemden kaynaklanmıştır.

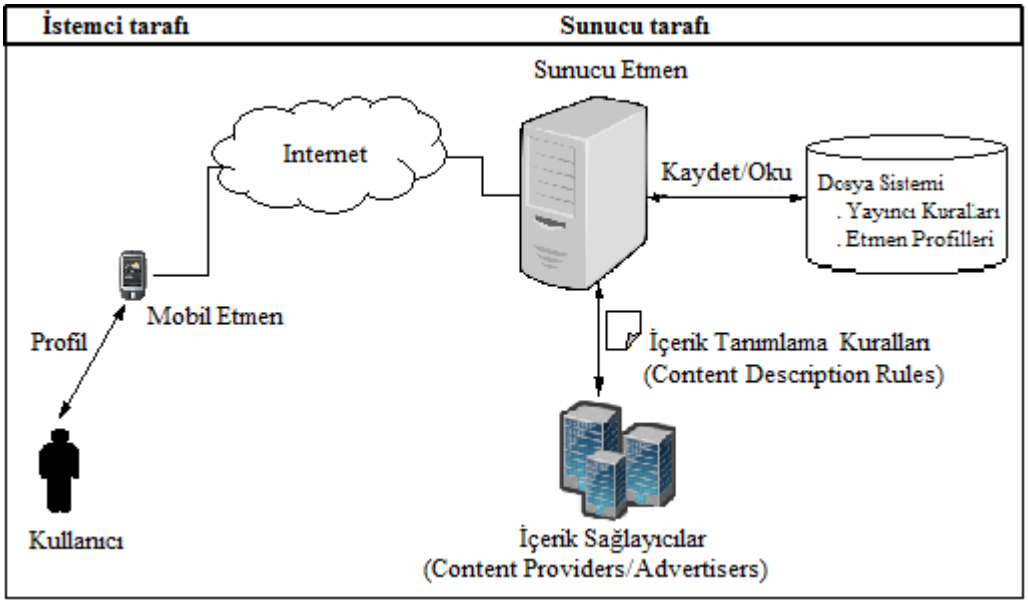
Bu problemi kısaca şöyle tarif edilebilir; internet kullanırken farklı zamanlarda çok sayıda siteye üye olup e-posta adreslerimizi paylaşıyoruz. Site sahipleri sunmuş oldukları ürün ve/veya hizmetlere ait reklam, promosyon gibi durumlarda üyelerine e-postalar gönderiyorlar. Biz kullanıcılar ise bir çok kaynaktan gelen bu e-postaları çoğu zaman okumadan siliyoruz. Bu durumda, gelen reklamlar arasında işimize yarayabilecek ürün ve/veya hizmetlerden haberdar olma şansını kaybedebiliyoruz. Hem posta kutumuzu doldurduğu, hem de vaktimizi aldığı için bu durumdan rahatsız oluyoruz. Diğer taraftan, ilerde faydalanmayı düşünebileceğimiz bir ürün ve/veya hizmet promosyonundan haberdar olmak istediğimiz için posta üyeliğinden çıkmak istemiyoruz. Bunu engellemek için e-postaları filtrelemeyi düşünebiliriz. Fakat, gönderilen e-postalar tümüyle metin tabanlı olduğu için gönderilen bu e-postaları anahtar kelime bazlı filtrelemekten başka bir şansımız olmamaktadır. Anahtar kelime bazlı bir filtreleme ise basit bir filtreleme olacak ve filtreleme işleminden beklediğimiz verimi alamayacağız. İşte bu noktada, nasıl bir reklamcılık modeli tasarlanırsa mevcut durumu iyileştirir. Yani, bir kullanıcı sadece ilgilenebileceği e-postaları alırken, reklam sağlayıcılar ise reklamlarının ilgilenen son kullanıcılara ulaştırıldığından emin olmalılardır.

Belirtilen problem için çeşitli alanlardan makaleler okunmuştur. Araştırmalar sonucunda, uzman sistemlerin ve etmen tabanlı sistemlerin yaklaşımlarının bir arada kullanılarak tasarlanacak platformun bu probleme bir çözüm sunacağı düşünüldü. Tasarlanan sistem örnek senaryolar kullanılarak test edildi. Bölüm 5.1'de önerilen çözüm yaklaşımı, Bölüm 5.2'de kişisel etmenin nasıl gerçeklendiği, Bölüm 5.3'de ise gerçeklenen sistemin katılımcıları ayrıntılı olarak anlatılacaktır. Son olarak Bölüm 5.4'te ise gerçeklenen sistem üzerinde örnek senaryo kullanılarak yapılan test ve sonuçları anlatılmaktadır.

5.1 Çözüm Yaklaşımı

Öncelikle her bir insanın mobil aygıtı üzerinde kurulmuş bir etmen uygulamasının o kişinin tercihlerini ve davranışlarının alınması/saptanması için geliştirilmiştir. Kullanıcı tercihlerinin alındıktan sonra sunucu etmen üzerinde kullanıcıya uygun reklamların belirlenmesi için bir uzman sistem tasarlanmıştır. Tasarlanan uzman sistem kural tabanlı bir yaklaşım ile kullanıcı tercihlerine uygun reklamları saptayacaktır. Sunucu etmen, uzman sistem tarafından saptanan reklamları kullanıcının mobil etmenine gönderecektir. Sistemde ki tüm haberleşme ontoloji tabanlı yürütülecek olup ve bu sayede reklam domainleri içerisinde normalizasyon yapılmış olunacaktır.

Bu tez çalışmasında belirtilen probleme sunalan çözümün final mimarisi Şekil 5.1'de resmedilmiştir.



Şekil 5.1 : Tasarlanan Sistemin Mimarisi.

Tasarlanan sistemde reklam sağlayıcılarının kuralları ve mobil etmenlerin profilleri disk üzerinden dosya tabanlı olarak tutulur. Sunucu etmen bu dosyalara erişme ve değiştirme de özgürdür.

Reklam sağlayıcılar sistem danışmanlarına metin tabanlı olarak verilmiş reklamı verirler. Sistem danışmanları verilen metin tabanlı reklamı sistemin yorumlayacağı dil olan Drools kural dosyasına bir kereye mahsus çevirirler. Oluşturulan kural dosyasında reklamın ne olduğu, promosyonu yapılan ürün/hizmet, faydalanabilmek için gerekli şartlar, kampanyanın zamanı, yeri, fiyatı vb gibi bilgiler bulunur. Her bir

reklam için metin tabanlı reklamdan kural dosyasının oluşturulması belli bir miktar insan gücü alacaktır. Dolayısıyla, kaybedilen bir miktar zaman vardır. Ama bu vakit, binlerce veya daha fazla insanın e-postalarını gözden geçirirken aynı e-posta için kaybettiği toplam vakit yanında göz ardı edilebilir.

5.2 Kişisel Ajan (Personal Agent)

Tasarlanan sistem bahsedildiği üzere JADE üzerinde geliştirildi. Mobil aygıt üzerinde etmen yapıyı kurmak için de JADE'in limitli kaynaklara sahip aygıtlar için minimize edilmiş versiyonu olan JADE-LEAP [26] kullanıldı. Kural tabanlı sistem olarak da, Java'da geliştirilmiş olması ve Eclipse üzerinde geliştirmeye yardımcı araçlara sahip olmasından ötürü Drools seçildi.

Mobil etmen tez çalışmasında sadece Android işletim sistemi için geliştirildi. Bütün reklamların içeriği ontolojiler kullanılarak temsil edildi. Böylece aynı domaine ait reklamlar arasında normalizasyon yapmak mümkün olmuştur. Kullanıcılar reklam tercihlerini ontolojide bulunan reklam domainin kavramsal modelini temsil eden sınıfdan otomatik olarak üretilmiş arayüzlere Android destekli mobil aygıtı üzerinden girer.

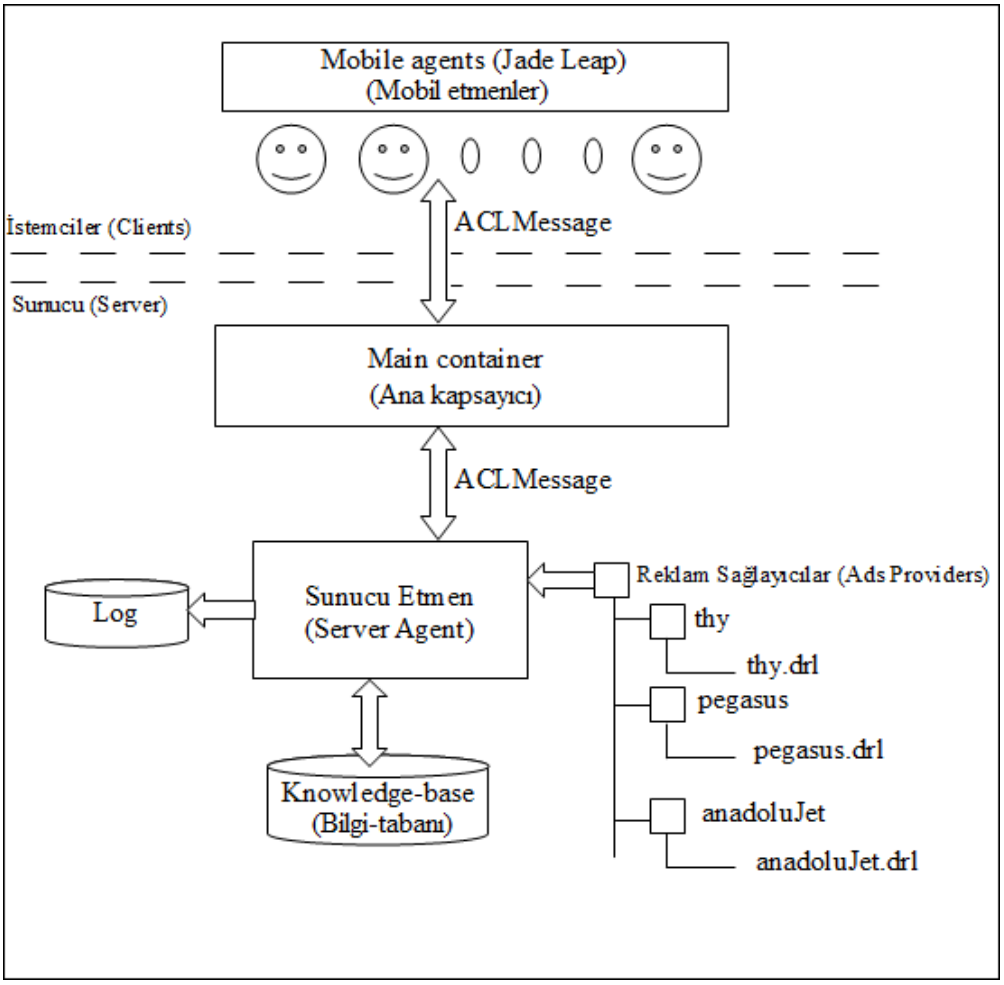
5.3 Sistem Katılımcıları

Bu çalışmada istemci etmen ve sunucu etmen olmak üzere iki tip etmen gerçeklenmiştir. Bir istemci etmen belirli bir kullanıcıya ait olup kullanıcının Android tabanlı mobil aygıtı üzerinde koşar. Genel olarak, istemci etmen JADE-Leap kütüphanesi kullanılarak geliştirilen bir Android uygulamasıdır. Sistemin genel görünümü Şekil 5.2'de verilmiştir.Her bir etmen içingeliştirilen yazılım paketleri ise Şekil 5.3'te gösterilmiştir.

Her bir etmen üç katmandan oluşmaktadır. İstemci etmen, sunum (presentation), iş mantığı (business logic) ve haberleşme (communication) katmanlarından oluşmaktadır. Sunucu etmen ise, haberleşme, iş mantığı ve veritabanı katmanlarından oluşmaktadır.

Ontoloji paketi her iki etmen için ortaktır. Etmenler arası haberleşme ACL mesajları ile yapılır. Her bir ACL mesajı içerisine gönderilmek istenen veri ontoloji belirtilerek yazılır. Böylelikle geliştirilen ontoloji aslında gerçeklenen etmenlerin ortak bir dili

olmuştur. ACL mesajları içerisine ontoloji nesneleri Jade'inkodek sınıfı ile yazılır. Nesneden katar veya katardan nesne dönüşümleri sunulan fonksiyonlar ile yapılır.

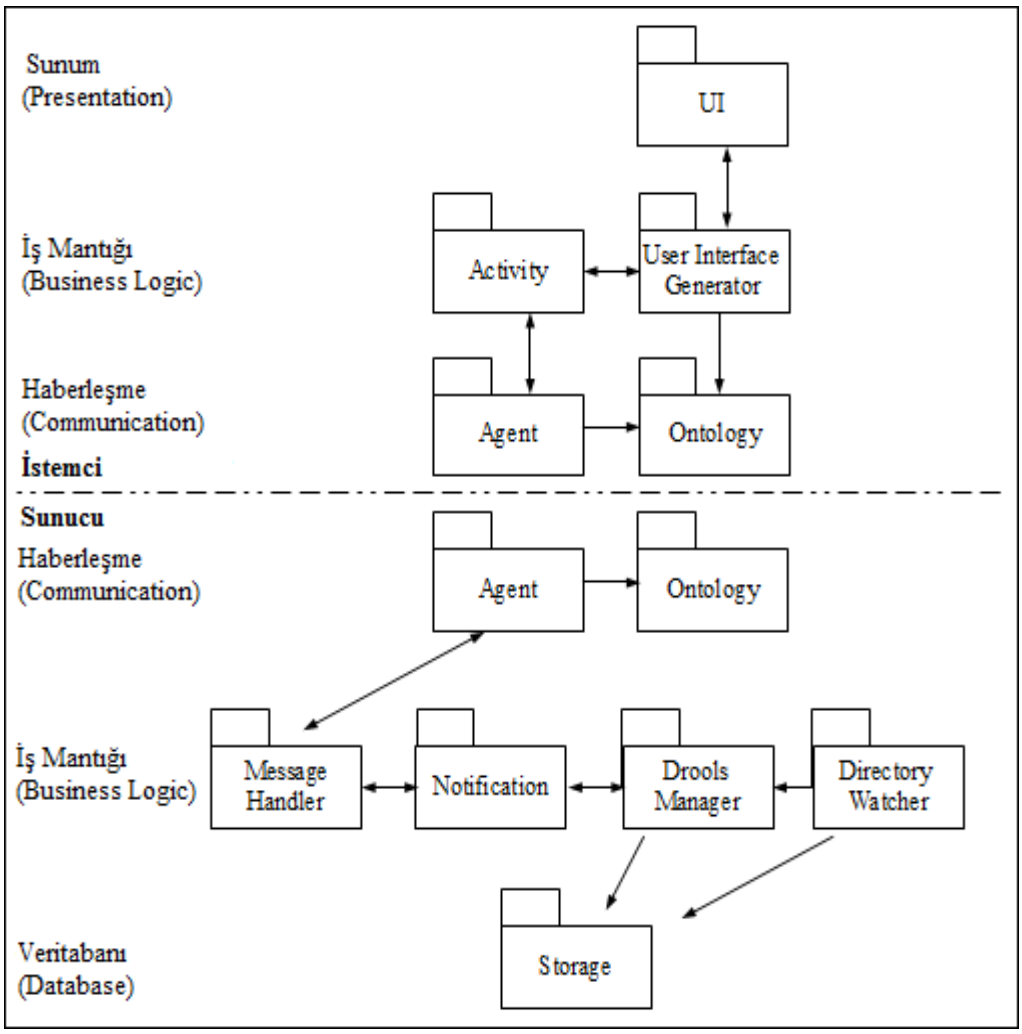


Şekil 5.2 : Sistemin genel görünümü.

Her bir paket yazılımsal işlevsellik olarak bir birinden ayrılmış Java paketlerini temsil eder. Aşağıdaki bölümlerde her bir etmen türü ve paketleri hakkında ayrıntılı bilgiler verilecektir.

İşlevsel olarak birbirinden ayrılan her bir paket bu çalışma kapsamında ayrı ayrı gerçeklenmiştir.İstemci ve sunucu etmen için geliştirilen paketleri ve paketler arasındaki ilişkileri bir arada görebilmek için Şekil 5.3'te tüm paketler tek bir şekilde resmedilmiştir.

Her bir paket için sarfedilen yazılım eforu farklıdır. Bu çalışma kapsamında yapılan başlıca yazılım eforları; JADE dağıtık altyapısının hem sunucu hem istemci tarafında kurulması, Drools kural motorunun sisteme entegrasyonu, Android üzerinde dinamik arayüz oluşturan paketin yazılması ve ontolojilerin geliştirilmesi olarak sıralanabilir.



Şekil 5.3 : Etmenler üzerindeki paketler.

5.3.1 İstemci etmen (Client agent)

İstemci etmen 5 yazılım paketinden oluşmaktadır.

5.3.1.1 Etmen paketi (Agent package)

Bu paket Android OS üzerinde etmen tabanlı altyapının kurulması için gerekli işlemleri içerir. Android uygulaması içerisinde Jade bir servis olarak çalışmaktadır. Uygulamaya açıldığında, istemci kullanıcı adı girerek ana-konteynera (main-container) kayıt olur. Verilen bu kullanıcı adı sistemde tekil bir kullanıcıyı/etmeni temsil edecektir.

Sunucu etmen ile yapılacak tüm haberleşmeler uygun ontology ile ACLMessage'ları kullanılarak yürütülecektir. İstemci etmen sunucu etmen ile yapılacak haberleşmeler için gerekli davranışları (behaviours) oluşturur. Projede kullanılan önemli davranışlar şunlardır;

- CyclicBehaviour, asenkron bir şekilde sunucudan gelecek mesajları bekler ve bir mesaj alındığında alınan mesajı işler.
- OneShotBehaviour, kullanıcının tercihlerine uygun reklamları alması için sunucuya anlık istek göndermek için kullanılır. Bu yaklaşım yoklama (polling) yaklaşımıdır. Çünkü, kullanıcı tercihlerine uygun bir reklam olup olmadığını kendisi sorguluyor. Ayrıca, otomatik olarak sunucuda bir kullanıcının tercihlerine uygun reklam olduğunda ilgili kullanıcıyı uyarma (notification) yaklaşımını da altyapı desteklemektedir.

5.3.1.2 Ontoloji paketi (Ontology package)

Bir çok veri sağlayıcı ve istemcisinin olacağı bir ortamda standardizasyonu sağlamak için bir ortak payda bulunması gereklidir. Bu payda Semantic Web [27] ile giderek önem kazanan ontolojiler yardımıyla sağlanabilir. Ontolojiler her bir domain için önceden tanımlanıp hem servis sağlayıcılar hem de istemciler ile paylaşılabilir. Jade ontolojileri Protege programı ve bu program için geliştirilmiş bir eklenti ile tanımlanabilir [28].

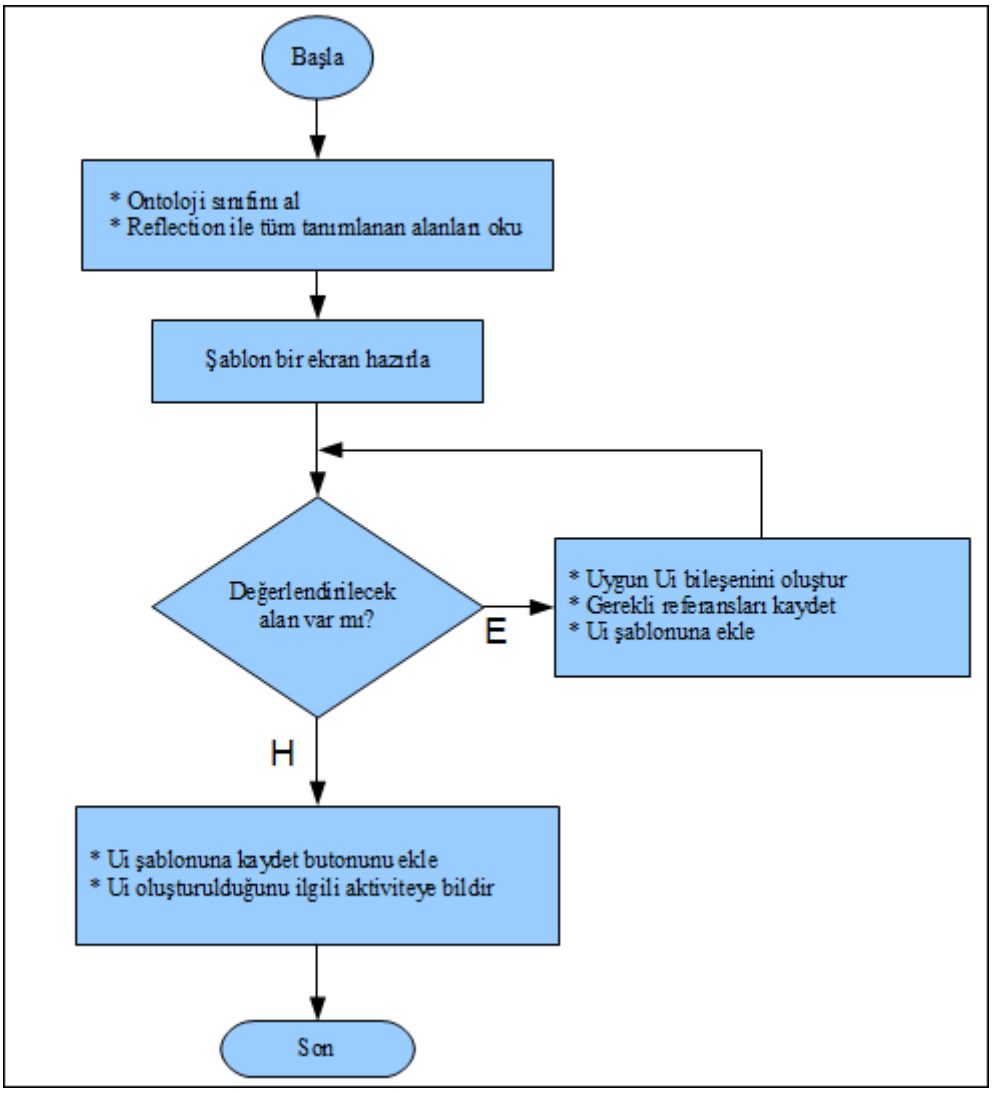
Ontologi geliştirme süreci geliştirilecek ontolojinin alanının (domain) ve kapsamının (scope) belirlenmesi ile başlar. Belirlenen domain ile ilgili istenen kapsamda yani derinlikte bir ontoloji geliştirmek uygun olacaktır. Bu çalışmada, uçuş domaini ile ilgili reklam gereksinimlerini karşılayacak özelliklere sahip "flyOntology" geliştirilmiştir. Yani havayolu firmalarının yaptığı reklamların/promosyanların duyurulması için gerekli ontoloji geliştirilmiştir. Bu ontoloji kullanılarak tasarlanan sistem test edilmiştir.

Sonuç olarak ontolojiler her bir domainin kavramsal soyutlamasını yapmak için kullanılmaktadır. Hem istemciler hem de sunucu etmenler arasında bu paket ortaktır.

5.3.1.3 Kullanıcı arayüz üretici paketi (User interface generator package)

Tasarlanan sistemin uygulanabilirliği açısından bu modül oldukça önemlidir. Bu sistemde her bir kullanıcı kendi etmeni üzerinden içerik bazlı filtreleme yapabilmesi için veri girişine imkan sağlayacak ekranların bulunması gerekmektedir. Her bir domain için ayrı bir ekran tasarlamak zaman ve maliyet açısından uygulanabilir değildir. Bunun yerine ontolojiler kullanılarak standart bir şekilde dinamik ekranların oluşturulması hedeflenmiştir.Arayüz üretici sınıfın nasıl çalıştığını gösteren akış

diyagramı Şekil 5.4'te verilmiştir. Bu paket ontoloji sınıfından yansıma (reflection) kullanarak belirtilen ontoloji için nasıl bir ekran oluşturması gerektiğini öğrenecek. Ontoloji sınıfının tanımına göre ontoloji için kullanılacak ekran dinamik olarak üretilecektir.



Şekil 5.4 : Arayüz üretici sınıfın akış diyagramı.

Bu modül domain ile ilgili ontology sınıfını Java reflection [29] yardımıyla analiz (introspection) ederek dinamik olarak çalışma zamanında ekranı oluşturur. Dinamik olarak üretilen ekrandan yine reflection kullanılarak kullanıcı tarafından girilen bilgiler okunur. Bu bilgiler; her bir alana karşılık gelen değer ve bu alan için girilen operatördür.Bu bilgiler ontoloji sınıfının wrapper nesnesine yazılır ve sunucu etmene gönderilir. Ör:Kampanya başlangıç tarihi bir alan olarak düşünülünce değer olarak 8/5/2012 verilebilir. Operatör olarak verilen tarihten sonra anlamına gelen ">" seçilebilir.

5.3.1.4 Kullanıcı arayüz paketi (UI package)

Ontoloji sınıfı kullanılarak dinamik olarak form tabanlı bir ekranın oluşturulacağından bahsetmiştik. Bu paket, formu oluşturan her bir alt bileşen sınıflarının bulunduğu pakettir. Bir form bu paketteki alt bileşenlerin farklı sıra ve sayıdaki örneklerinden (instance) oluşur.

Bir alt bileşen oluşturulurken gerekliyse etiket ve/veya operatör de beraberinde oluşturulur. Operatörler her bir alt bileşen için farklıdır. Bazı alt bileşenler ve her bir alt bileşeniçin geliştirilen operatörler şunlardır;

- UiEditBox: Katar (java.lang.String) ve sayı tipleri (int, float) alanları için kullanılır. Operatörler alan tipine göre değişir. Katar ise: == ve !=.

  Sayı tipleri ise: ==, <, > ve !=

- UiDatePicker: Tarih (java.util.Date) alanları için kullanılır. Operatörler: before, ontime, after.

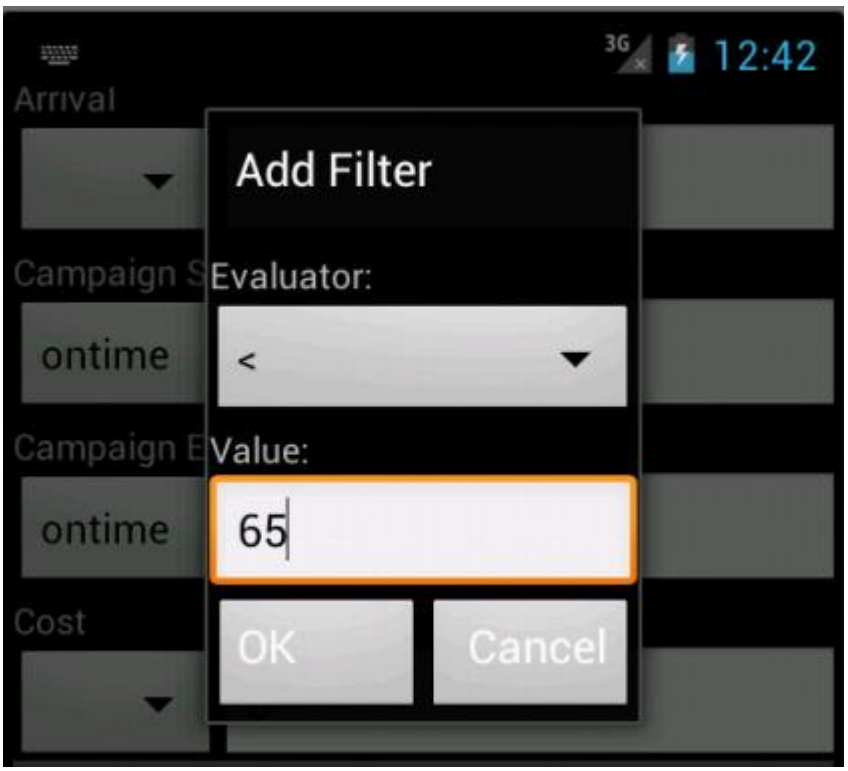
- UiCheckBox: Mantıksal (boolean) alanlar için kullanılır. Operatör: ==

- UiSelectOne: Enum sınıflar için kullanılır. Kullanıcı enum değerleri içerisinden birini seçer. Operatör: ==

Ör: UiEditBox alt bileşeni bir metin alanının form üzerinde etiket, operatör ve değer alanlarının tamamını ekranda oluşturup, alanlara girilen girdilerin işlenmesini (handle) sağlar. Şekil 5.5'te "float" tipinde "cost" alanı için oluşturulan UiEditBox bileşeni verilmiştir. Görüldüğü üzere "Cost" etiketi, operatör listesive değer alanı olmak üzere üç görüntü (view) içermektedir.

Şekil 5.5 : UiEditBox bileşen örneği.

Şekil 5.6'dafloat tipindeki cost alanına çift tıklanınca açılan dialog penceresi verilmiştir. Dialog penceresi sayı operatörlerinin seçilebileceği bir liste (spinner) ve sayının girilebileceği bir metin alanından oluşmaktadır. Bu ekran işlenen alanın tipine göre uygun operatörleri getirmektedir. Değer (value) alanı ise veri tipi sayısal sınıftan (int, float, vb.) ise sadece rakam girdisi kabul etmektedir.



Şekil 5.6 : Float sayı alan bir UiEditBox görünümü.

5.3.1.5 Aktivite paketi (Activity package)

Aktivite sınıfları bir Android uygulamasında kullanıcı arayüz ekranlarına karşılık gelir. Her bir ekran için bir aktivite sınıfı geliştirilmiştir. Uygulamada bulunan domainlerin listelenmesi, uyarıların alınıp gösterilmesi gibi Android aktivitelerinin bulunduğu pakettir.

5.3.2 Sunucu etmen (Server agent)

Sunucuda koşan bu etmen istemcilerin tercihlerine göre bildirim (notification) hizmetini sunacaktır. Tüm reklam sağlayıcılar (providers) içerisinde kullanıcı tercihlerine uygun bir hizmet/ürün olduğu durumda bir bildirim mesajı oluşturularak istemciye gönderilir. Bildirim gönderimi tercihlere uygun bir hizmet/ürün oluştuğunda otomatik olarak gönderilebileceği gibi kullanıcının anlık isteğinde de gönderilebilir.

Her bir modül ve içeriği aşağıda ayrı ayrı anlatılmıştır.

5.3.2.1 Etmen paketi (Agent package)

İstemciler ile haberleşip onların isteklerine cevap verebilmek için sunucu tarafında standart bir JADE etmeni olarak gerçeklenmiştir. Belirli periyotlarla veya her hangi

bir istemciden bildirim isteği gelince, Drools motorunu kullanarak istemci için uygun bir bildirim olup olmadığını kontrol eder.

5.3.2.2 Ontoloji paketi (Ontology package)

Hem istemci etmenler hem de sunucu etmen arasında bu paket ortaktır. Ayrıntılı bilgi için lütfen 5.3.1.2 nolu bölüme bakınız.

5.3.2.3 Drools yönetici paketi (Drools manager package)

Bu paket istemci tercihleri ile reklamlar arasındaki eşleşmeyi kural tabanlı (rule-based) olarak saptayan Drools oturumunu oluşturan ve yöneten modüldür. Tasarlanan sistemde, reklam sağlayıcılarının hazırlamış olduğu reklamlar drools kural dosyalarına (drl files) çevrilir ve sunucu etmenin bulunduğu makinenin dosya sistemine yüklenir. Her bir "drl" dosyası bir reklama karşılık gelecek şekildedir. Drools kural dosyasında iş mantığı (business logic) ilgili domain ontolojisi kullanılarak elle oluşturulur.

Bu modül tekil bir nesne (instance) olarak tüm yayıncıların bilgilerini (drl dosyalarını) okuyarak sistemin bilgi tabanını (knowledge-base) oluşturur. Her bir bildirim isteğine karşılık, istemcinin ilgilendiği domain nesnesini Drools oturumuna bir fact olarak ekler. Drools session koşturulur, oluşan bildirim sonuçları istemci etmene dönülür.

Bu modül çalışırken, çalışan bellek (working memory) ajanda dinleyicileri (agenda listeners) yardımıyla ayrıntılı loglama diske yapılır. Bu loglar, bir bildirimin nasıl oluştuğunu adım adım kayıt eder. Bu loglar yapılan işlemi daha sonra incelemek için kullanılır.

5.3.2.4 Klasör gözlemci paketi (Directory watcher package)

Tüm veri sağlayıcılarının (reklam verenler) bilgileri sunucu etmenin dosya sistemi üzerinde bulunmaktadır. Her bir veri sağlayıcısı (data provider) için ayrı bir klasör bulunmaktadır. Sistem çalışırken sisteme yeni sağlayıcılar eklenebilir veya sistemde bulunan sağlayıcılar sistemden ayrılabilir. Ayrıca, yayıncılar vermiş oldukları reklamlar ile ilgili kurallarını zaman içerisinde değiştirebilirler. Bahsedilen bu değişikliklerin sistemin yeniden başlatılmasına gerek olmadan sistem tarafından saptanıp, bilgi tabanının (knowledge-base) güncellenmesi gerekmektedir. Bunun

için, geliştirilen adanmış bir iplik (thread) dosya sistemini belirli aralıklarla kontrol etmekte ve gerekli olduğu durumda bilgi tabanını güncellemektedir. Böylelikle, reklam sağlayıcılar veya mevcut herhangi bir reklam ile ilgili bir değişiklik söz konusu olduğunda sistemin yeniden başlatılmasına gerek olmamaktadır. Böyle bir tasarım sistemin esnekliği ve geliştirilebilirliği açısından önemlidir.

Özetle bu modül, sistemin bilgi tabanının güncelliğini sağlamak üzere ayrı bir Java ipliği (thread) olarak geliştirilmiştir.

5.3.2.5 Mesaj işleyici paketi (Message handler package)

Sunucu etmene gelen her bir ACL mesajı bu paketteki sınıflar aracılığıyla uygulama objesine (application object) çevrilir ve mesajı işleyecek uygun işleyici (handler) çağrılır. Bir protokol hatası olduğu durumda hata mesajı istemciye dönülür.

5.3.2.6 Bildirim paketi (Notification package)

Bildirim işlemlerini yapan sınıfların olduğu pakettir. Bir istemcinin tercihlerinden oluşmuş domain objesini Drools oturumuna ekler. İlgili oturum (Drools session) koşturulur ve aktive olan kurallar kayıt edilir. Elde edilen sonuç kümesi yeni bir ACLMesajı oluşturularak ilgili istemciye gönderilir. Ayrıca bu paket, Drools oturumunun desen eşleme (pattern matching) işlemini ve adımlarını diske loglar. Bu loglar sayesinde her bir isteğe karşılık üretilen bildirimlerin oluşma sebepleri ayrıntılı olarak daha sonra izlenebilir.

5.4 Örnek Senaryo

Bu bölümde, tasarlanan sistemörnek bir senaryo kullanılarak test edilecek ve test sonuçları incelenecektir. Böylece sistemin uygulanabilirliği değerlendirilecektir.

Örnek senaryoda amaç; istemcilerin ilgilenebilecekleri kampanyalı bir uçuş olduğu durumda bu uçuştan haberdar olmalarıdır. İstemciler tercihlerini zaman içerisinde değişterebilirler.

İlk olarak belirtilen ihtiyaçı karşılayacak bir ontoloji tasarlandı. Ontoloji geliştirirken uygulamanın ihtiyaçları göz önünde bulundurulmalıdır. Hedeflenen sistem için yeterli derinlikte ve özelliklerde uçuş domainini temsil eden ontoloji (FlyOntology) geliştirildi.

Geliştirilen ontolojide kullanıcı arayüz ekranının oluşturulması için kullanılan Fly sınıfı *jade.content.Concept* arayüzünden (interface) türemiştir. Fly sınıfı Şekil 5.7'de verilmiştir.

```
public class Fly implements Concept{

   @UiAttributes(name="Company Name", order = 0)
   private String company;
   @UiAttributes(name="Flying Type", order = 1)
   private FlyType flyingType;
   @UiAttributes(name="Departure", order = 2)
   private String departurePlace;
   @UiAttributes(name="Arrival", order = 3)
   private String arrivalPlace;
   @UiAttributes(name="Campaign Start", order = 4)
   private Date startDate;
   @UiAttributes(name="Campaign End", order = 5)
   private Date endDate;
   @UiAttributes(name="Cost", order = 6)
   private float cost;
   ....
}
```

Şekil 5.7 : Ontolojide geliştirilen Fly kavramı (concept).

Sınıf tanımında kullanılan UiAttributes notu (annotation) arayüz oluşturulması sırasında kullanılacak form bileşeninin alan adı ve alanın formdaki sırasını göstermektedir.

Aslında, sıra (order) bilgisinin belirtilmesine gerek olmayabilirdi. Ama, Android Dalvik VM diye adlandırılan standart Java sanal makinesinin özelleştirilmiş bir halini kullanıyor. Yani, standar JVM'de Java yansıma (reflection) kullanılarak sınıf üyeleri sıralı bir şekilde elde edilebiliyor. Fakat, Dalvik VM'de yansıma ile elde edilen sınıf üyeleri sıralı bir şekilde elde edilemiyor. Bununla ilgili bir hata (bug) Android hata izleme sistemine açılmıştı. Bu tez çalışması sırasında sıra problemini aşmak için annotation'a sıra bilgisini de içeren bir özellik eklemek zorunda kaldım.

Tasarlanan sistemi test etmek için kullanılan senaryoda sistemde üç farklı firma ve bu firmalar tarafından duyurulan birer kampanya oluşturulmuştur. Bu üç farklı firma ve duyurdukları kampanyaların şartları şu şekilde verilmiştir:

- AnadoluJet: 5/6/2012 – 25/6/2012 tarihleri arasında İstanbul'dan Van, Samsun, Hatay illerine yapılan uçuşların 70 TL'den başlayan fiyatlarla yapılacağı duyurulmuştur.
- İzair: 5/6/2012 – 5/7/2012 tarihleri arasında İstanbul-İzmir ve İzmir-İstanbul uçuşlarının 55 TL'den başlayan fiyatlarla yapılacağı duyurulmuştur.
- Pegasus: 1/6/2012 – 1/7/2012 tarihleri arasında tüm yurtiçi uçuşların 60 TL'den başlayan fiyatlarla yapılacağı duyurulmuştur.

Görsel reklamlar ile duyurulan bu kampanyalı uçuşların tasarlanan sistemde yorumlanabilmesi için bu kampanyaların Drools kuralları olarak ifade edilmesi gereklidir. Drools kuralları analistler veya programcılar tarafından yazılabilir. Verilen kampanyalara karşılık geliştirilen Drools kuralları ayrı ayrı aşağıda verilmiştir.

AnadoluJet firması için geliştirilen Drools kural dosyası Şekil 5.8'de verilmiştir.

```
package client.drl;

#list any import classes here.
import com.thesis.common.drools.Consequence
import com.thesis.common.drools.RunResult
import com.thesis.ontology.Response
import com.thesis.ontology.fly.Fly
import com.thesis.ontology.fly.FlyType
import java.lang.String
import java.util.Date


rule rule_AnadoluJet
	when
		$fly : Fly(departurePlace == "Istanbul",
		(arrivalPlace == "Van" || arrivalPlace == "Samsun" || arrivalPlace == "Hatay"),
		eval(startDate  != null && startDate.after(new Date(2012,5,5))),
		eval(endDate != null && endDate.before(new Date(2012,5,25))), cost >= 70)
		$run : RunResult(  )
	then
		System.out.println("rule_AnadoluJet");
		Consequence $con = new Consequence();
		$con.setData("İstanbul'dan Samsun, Van, Hatay'a ucuslar 70 TL!");
		$con.setPublisherName("Anadolu Jet");
		$con.setReferenceCode("1007");
		$run.addConsequence($con);
end
# file ends
```

Şekil 5.8 : AnadoluJet Drools kural dosyası (anadoluJet.drl).

AnadoluJet firmasının kural dosyası incelendiğinde metin tabanlı olarak verilmiş olan kampanyanın sistemin yorumlayabileceği bir formata çevrilmiş hali görülmektedir. İşte bu çevrim her bir reklam/kampanya için bir kereye mahsus olmak üzere sistem analistleri veya programcıları tarafından yapılır. Kural dosyasında kampanyanın şartları programatik bir şekilde ifade edilir. Daha karmaşık işlerin yapılması için "eval" fonksiyonu kullanılabilir.

Her bir kuralın sağ tarafında (RHS)yani kural başarılı olduğunda kural ve kampanya ile ilgili kullanıcıya dönülecek kural adı, kampanya kodu, vb.ek bilgiler bulunur.İzair firması için geliştirilen Drools kural dosyası Şekil 5.9'da verilmiştir.

```
package client.drl;

#list any import classes here.
import com.thesis.common.drools.Consequence
import com.thesis.common.drools.RunResult
import com.thesis.ontology.Response
import com.thesis.ontology.fly.Fly
import java.lang.String
import java.util.Date


rule rule_izair
	when
		$fly : Fly((departurePlace == "Istanbul" && arrivalPlace == "Izmir") ||
		(departurePlace == "Izmir" && arrivalPlace == "Istanbul"),
		eval(startDate  != null && startDate.after(new Date(2012,5,5))),
		eval(endDate != null && endDate.before(new Date(2012,6, 5 ))), cost >= 55 )
		$run : RunResult(  )
	then
		System.out.println("rule_izair");
		Consequence $con = new Consequence();
		$con.setData("istanbul - izmir  55 TL");
		$con.setPublisherName("Izair");
		$con.setReferenceCode("1003");
		$run.addConsequence($con);
end
# file ends
```

Şekil 5.9 : İzair Drools kural dosyası (izair.drl).

Pegasus firması için geliştirilen Drools kural dosyası Şekil 5.10'da verilmiştir.

```
package client.drl;

#list any import classes here.
import com.thesis.common.drools.Consequence
import com.thesis.common.drools.RunResult
import com.thesis.ontology.Response
import com.thesis.ontology.fly.Fly
import com.thesis.ontology.fly.FlyType
import java.lang.String
import java.util.Date


rule rule_pegasus
	when
		$fly : Fly( flyingType == FlyType.DOMESTIC,
		eval(startDate  != null && startDate.after(new Date(2012,5,1))),
		eval(endDate != null && endDate.before(new Date(2012,6,1 ))), cost >= 60)
		$run : RunResult(  )
	then
		System.out.println("rule_pegasus");
		Consequence $con = new Consequence();
		$con.setData("Tum Yurtici Ucuslar 60 TL");
		$con.setPublisherName("Pegasus");
		$con.setReferenceCode("1002");
		$run.addConsequence($con);
end
# file ends
```

Şekil 5.10 : Pegasus Drools kural dosyası (pegasus.drl).

Pegasus firması tüm yurtiçi uçuşlarının 60 TL'den başlayan fiyatlarla yapılacağını duyurmuştu. Bu kuralda diğer iki kuralda olduğu gibi kalkış ve varış yerleri alanları belirtilmiyor. Çünkü, uçuş tipini yurtiçi (domestic) seçilmesi yeterli olmaktadır.

Yukarıdaki kurallar yazılırken açıklayıcı (expressive) olması tercih edilmiştir. Çünkü verilen örnekte tarih karşılaştırmaları *java.util.Date*sınıfı ve *eval()* fonksiyonu kullanılarak yapılmaktadır. Bu aslında eşleme algoritmasının (Rete) performansını düşürür. Gerçek bir uygulamada performans göz önünde bulundurularak tarih karşılaştırmaları milisaniye çevrilerek tamsayı karşılaştırması yapılması daha uygun olur.

Kurallar içerisinde Date sınıfından objeler oluşturulmaktadır. Date sınıfı ayları sıfırdan başlayarak temsil eder. Dolayısıyla sıfır Ocak ayına, onbir ise Aralık ayına denk gelir. Örneğin, kurallar yazılırken Haziran ayı için 6 değilde 5 yazılması bu sebepten kaynaklanmaktadır.

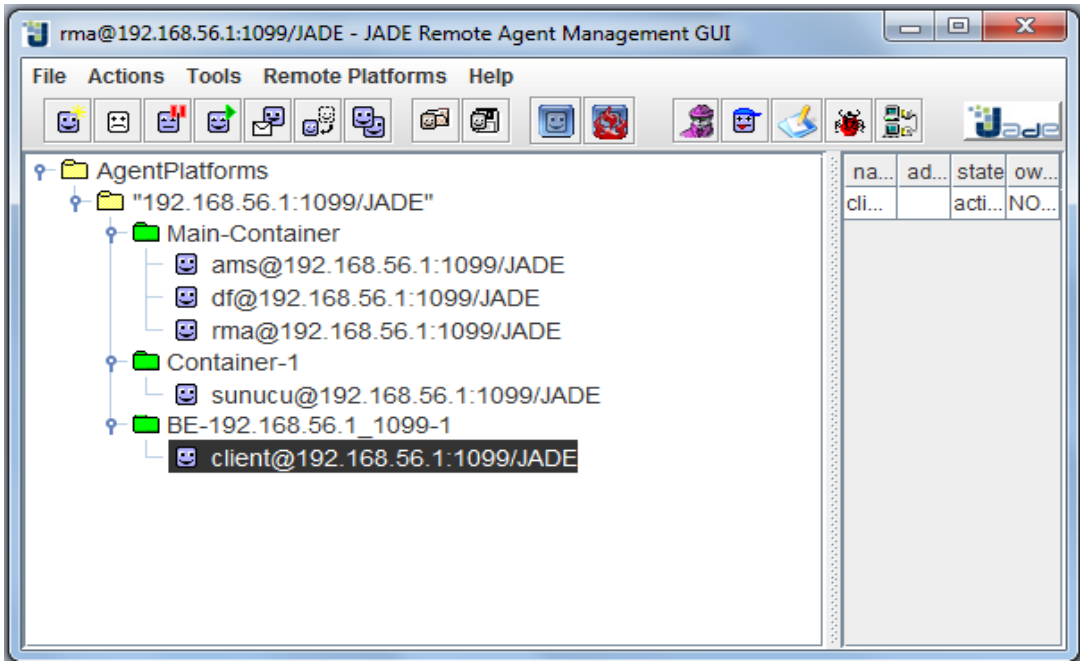
5.4.1 Test senaryosunun çalıştırılması

Test senaryosunun çalıştırılması için aşağıdaki adımlar izlenmelidir:

- Önce etmen tabanlı alt yapının kurulması için Jade altyapısının kurulması gereklidir. Bunun için komut satırından aşağıdaki komut çalıştırılarak Jadeplatformu oluşturulur.
    - cmd > java jade.Boot -gui
- İkinci olarak sunucu etmenin oluşturulması için Eclipse üzerinden veya komut satırından aşağıdaki komut çalıştırılır. Bu komuta göre etmenin adı "sunucu"olarak verilmiştir.
    - java jade.Boot -host localhost -container sunucu:com.thesis.server.agent.ServerAgent
- Son olarak JADE-LEAP kullanılarak istemci etmen olarak gerçekleştirilen Android uygulaması Eclipse emülatoru üzerinden veya bir mobil aygıta yüklenerek çalıştırılır.

Android uygulaması başlatılınca kullanıcı adı girilerek etmen tabanlı platforma katılınmış olunur. Senaryomuzda kullanıcı etmenin takma adı "client" olarak verilmiştir. Şekil 5.11'de etmen platformuna katılmış sunucu ve istemci etmenler görülmektedir.

JADE sistemi her bir etmene tekil bir etmen id (AID) atar. Bu id haberleşmelerde kullanılacaktır. Mevcut senaryoda ana kapsayıcı, sunucu ve bir istemci etmen vardır.



Şekil 5.11 : Test senaryosunun Jade etmenleri görünümü.

JADE çerçevesi uygulama geliştirirken geliştiricilerin ihtiyaç duyacağı araçlarıda içerir. JADE, dinleyici etmen (sniffer agent) ve iç gözetim etmeni (introspector agent) gibi kullanışlı araçlarıda sunarak geliştirme yapmayı kolaylaştırmaktadır.

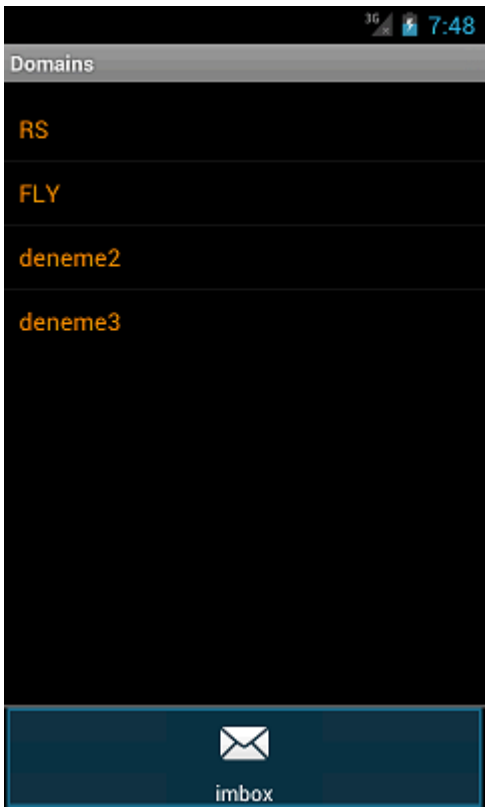
Kullanıcı Android uygulamasına girince önce etmen üzerinde tanımlanmış ontolojilerin listesini görür. Bu ontolojilerin her biri farklı bir alan (domain) için geliştirilmiştir. Test amaçlı geliştirilen ontolojiler ve Fly ontolojisi Şekil 5.12'degörülmektedir.

Görülen alan (domain) listesi çalışma kapsamında yazılımsal olarak kodda oluşturulmuştur. Gerçek bir uygulamada domain listesi bir ontoloji sunucusundan alınabilir.

Kullanıcı menü olarak akıllı posta kutusunu (intelligent mail box) görür. Kullanıcı "imbox" menüsüne tıklayarak kayıt edilmiş tercihlerine uyan kampanyaları alır.

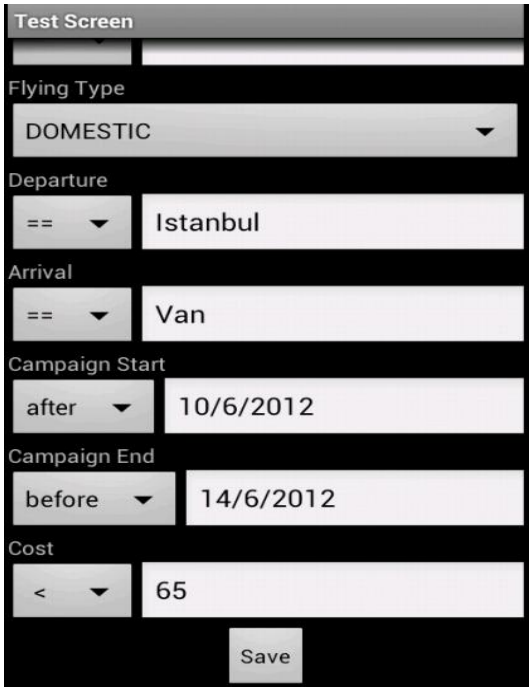
Etmen tabanlı android uygulamasında kullanıcı uçuş alanı (Fly) ile ilgili reklamlarla ilgileniyor. Kullanıcı ilgilendiği domain olan FLY satırına tıklar. Uygulama Fly alanı (domain) için veri girişi sağlayacak olan ekranı oluşturur ve açar.

Şekil 5.13'de kullanıcının mobil aygıtı üzerinde Fly alanı için gördüğü arayüz verilmiştir. Bu arayüz yukarıda belirtilen *com.thesis.ontology.fly.Fly* sınıfından otomatik olarak üretilmiştir. Kullanıcı bu arayüzden uçuş domaini ile ilgili tercihlerini girer.



Şekil 5.12 : Sistemde bulunan domainlerinlistesi.

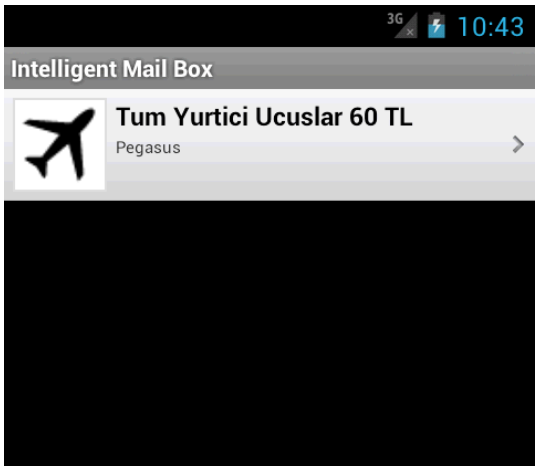
İstemci bu ekranda ilgi duyduğu veya duyabileceği uçuş özelliklerini belirleyen kısıtları ekrandaki alanlaragirer. Bu kısıtlar sistemde bulunan kural motoru tarafından bilgi filtrelemek için kullanılacaktır.



Şekil 5.13 : Uçuş domaini için üretilen kullanıcı arayüzü.

İstemci arayüzden ilgilendiği uçuşlar ile ilgili kısıtları yazabilir. Yukarıdaki örnekte, kullanıcı 10/6/2012 ile 14/6/2012 tarihleri arasında fiyatı maksimum 65 TL olan İstanbul-Van uçuşları ile ilgilenmektedir. Bu şartları sağlayan bir kampanya bulunuyorsa veya ileride oluştuğu durumda ilgili kampanyalar kullanıcıya bilgilendirme mesajı olarak iletilecektir.

Belirtilen kısatlar dahilinde sistemde bu kısıtları sağlayan sadece Pegasus firmasına ait bir kampanya bulunmaktadır. Kullanıcı bu kampanyadan bir bildirim mesajıyla haberdar edilir. Kullanıcıya gelen bildirim mesaji kullanıcının mobil aygıtında Şekil 5.14'teresmedilmiştir.



Şekil 5.14 : Kullanıcının akıllı e-posta kutusu.

Örneğe dikkat edilecek olunursa, İzair firmasının Van'a uçuşu bulunmamaktadır. AnadoluJet firmasının Van'a uçuşu bulunmakla birlikte belirtilen tarihler arasında uçuşların 70 TL'den başlayan fiyatlarla olduğu duyurulmuştu. Bu durumda kullanıcının fiyat kısıtı (65 TL) AnadoluJet kuralının başarılı olmasını engellemektedir. Dolayısıyla, sistemde kullanıcının kısıtlarını sağlayan tek kampanya Pegasus firmasına aittir.

Akıllı posta kutusuna gelen e-postalara tıklanınca ilgili kampanyaya ait ayrıntılı bilgilerin gösterilmesi planlanmaktadır. Ama bu kısım şu an için gerçeklenmemiştir.

5.4.2 Test senaryosunun performansı

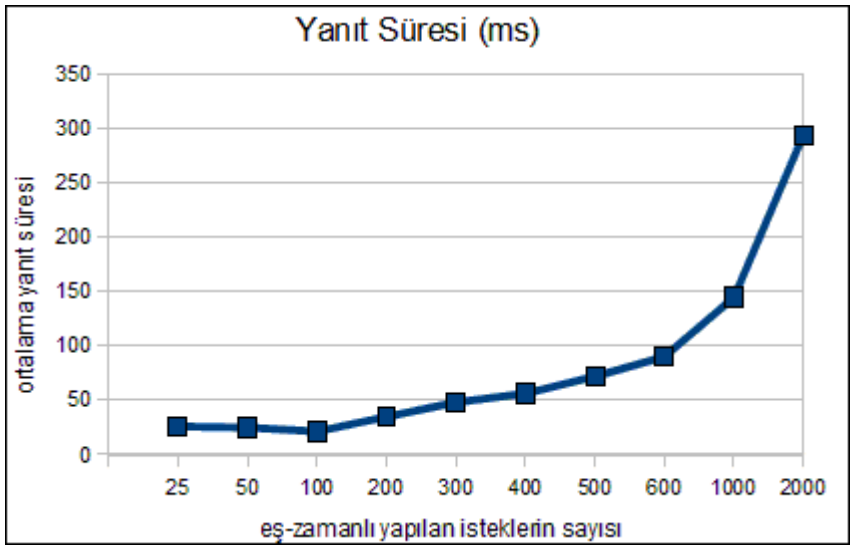
Tasarlanan sistemin performansı örnek senaryo kullanılarak test edilmiştir. Aşağıda farklı istemci sayılarında sistemin ortalama cevaplama süresi verilmiştir. Tüm ölçümler 2.27 GHz double core Intel Core i5 CPU, 4 GB RAM, 64-bit Windows 7 OS özelliklerine sahip kişisel bilgisayarda yapılmıştır.

Yapılan performans testinde her bir istemci eş zamanlı olarak sistemden istekte bulunuyor. Her bir istek kendi Jade konteynerına sahip test etmeni tarafından gönderiliyor. Çizelge 5.1'de farklı istemci sayılarında test senaryosunun yanıt süreleri verilmiştir.

Çizelge 5.1 : Farklı istemci sayılarında test senaryosunun yanıt süreleri.

İstemci Sayısı	Ortalama Zaman	Toplam Zaman
25	26.16	654
50	25.14	1257
100	21.52	2152
200	35.07	7013
300	48.31	14492
400	56.50	22600
500	72.18	36091
600	90.44	54264
1000	144.75	144749
2000	293.79	587573

Şekil 5.15'deki grafikte eş zamanlı istek sayısı artınca cevaplama süresininde artığı gözlemleniyor. Kabul edilebilir cevaplama süresini yaklaşık 50ms olarak alınırsa test ortamı için maximum eş zamanlı istek sayısı 400 olmalıdır. Sistemin yükü arttıkça yavaşlaması artan thread sayısının sebep olduğu bağlam değişikliği (context-switching) maliyetinden kaynaklanmaktadır. Dolayısıyla, bu durumu aşmak için sistem, uygulama sunucularında olduğu gibi iplik havuzu (thread-pool) oluşturarak maximum eş zamanlı isteği sınırlayacak şekilde geliştirilebilir.

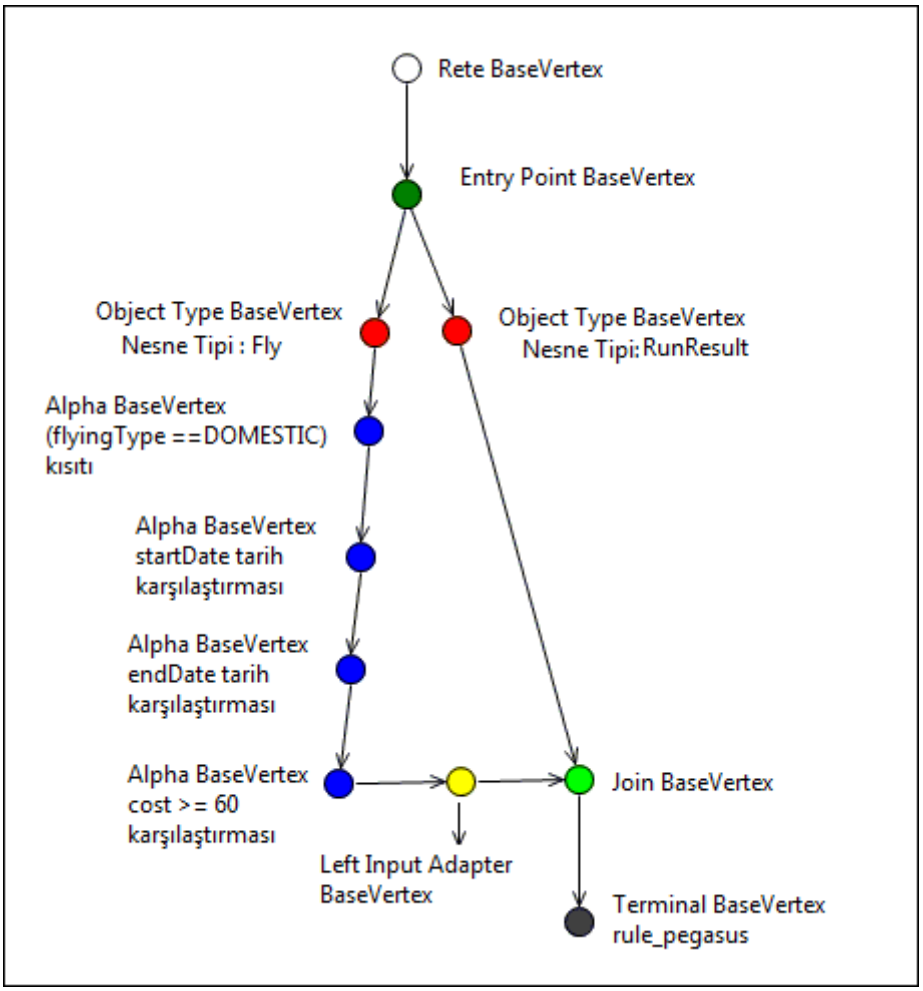


Şekil 5.15 : Yük testi sonuçları.

Yukarıda test senaryosunun nasıl performansının ölçüldüğünden bahsedildi. Bahsedilen yük testi sırasında tüm istemci etmenler programatik bir şekilde aynı kişisel bilgisayar üzerinde oluşturuldu. Dolayısıyla, aynı bilgisayar üzerinde çok sayıda ipliğin (thread)oluşmuş olmasından ötürü böyle bir sistemin performansı hakkında kesin bilgiler veremeyiz.

Performans testi için kullanılan etmen *com.thesis.agents.PerformanceTest*sınıfıdır. Bu sınıf verilen etmen sayısı kadar konteyner oluşturararak her bir konteynera *com.thesis.agents.TestAgent*test etmenini ekler ve her bir test etmenini aktive eder. Sunucu etmen gelen isteklere cevap vermeye çalışarak gerekli analizleri toplar.

Şekil 5.16'da test senaryosunda başarılı olan tek kural olan Pegasus firmasına ait kural dosyasının Rete ağını inceleyebiliriz. Drools oturumunda Fly ve RunResult nesnelerinin olması gereklidir. İkisi de varsa artık eşlemeye Fly nesnesinden devam edilir.



Şekil 5.16 : pegasus.drl dosyasının Rete ağı.

Görüldüğü üzere oluşturulan Rete ağında Fly nesnesi için dört kısıt vardır. Bu kısıtlar; uçuş tipi (flyingType), başlangıç tarihi (startDate), bitiş tarihi (endDate) ve fiyat (cost). Kısıtlardan uçuş tipi ve fiyat ilkel (primitive) bir karşılaştırma ile hesaplanmaktadır. Fakat, başlangıç ve bitiş tarihleri karşılaştırması ise bir değerlendirme sonrası (eval()) hesaplanıyor. Değerlendirme (eval()) içeren düğümler ilkel karşılaştırma içeren düğümlerden daha fazla vakit alacaktır.

Özetle, sistemin performansına etki eden ana bileşen Drools örüntü eşleme işleminin performansıdır.Yani, Rete algoritmasının performansıdır. Bu algoritmanın performansı ise doğrudan geliştirilen kuralların sayısına, kuralların karmaşıklığına (complexity), kuralların yazım şekline bağlıdır. Test senaryosunda daha okunaklı görünmesi için tarih karşılaştırmaları yapıldı. Fakat, Java yaklaşımı şeklinde değil de belirtilen tarihler milisaniyeye çevrilerek tamsayı karşılaştırması yapılabilirdi. Bu şekilde performans iyileştirmeleri, nihai uygulamanın performans ihtiyaçlarına göre sonraki aşamalarda yapılabilir.

6. SONUÇ VE ÖNERİLER

Bu çalışmada mobil aygıtlar üzerinde kişiselleştirilmiş reklam için etmen tabanlı çerçeve tasarımı yapılmıştır. Geliştirilen altyapının farklı domainlerde de çalışabilmesi için iki iş vardır. Birincisi, ihtiyaç duyulan domain için ontolojinin tanımlanması ve sisteme eklenmesidir. İkinci olarak, hedef domain ile ilgili reklamların kurallarının oluşturulması ve sisteme eklenmesidir. Sistem etmen teknolojilerin ve uzman sistemlerin yaklaşımlarını kullanarak esnek, geliştirilebilir bir çerçeve sunmaktadır. Sistem JADE üzerinde geliştirilmiş olduğu için tümüyle FIPA uyumludur.

Geliştirilen sistem sayesinde mobil cihaz sahiplerinin tercihlerini kural tabanlı olarak değerlendiren ve belirtilen tercihlere uygun reklamları kendilerine sunan bir hizmete kavuşmuş olmaktadırlar. Böylece kullanıcılar kendileri için daha doğru reklamlara hızlı bir şekilde erişebilirler. Ayrıca, sistem kullanıcıların reklamlar arasında kaybolmasını önleyerek hem kullanıcıların zamanını hem de internet bant genişliğini korumaktadır.

İleriki araştırmalarda, sistemin kullacının anlık bilgilerini(yer bilgisi, hız vb) de değerlendirme için kullanılarak daha akıllı sonuçlar üretecek şekilde geliştirilmesi planlanmaktadır.

Geliştirilen sistem ile kural tabanlı sistemler ile etmen tabanlı sistemlerin bir arada kullanılarak kişiselleştirilmiş reklamcılık problemine çözüm sunabileceği görülmüştür.Önerilen çözümün, yapılan örnek domain ve senaryo testi ile bazı domainlere uygulanabileceği ispatlanmıştır.

Sonuç olarak, geliştirilen sistem kullanıcının güdümünde kural tabanlı kişisel bir kampanya araştırma araçı sunarak mevcut kampanya öneri sistemlerinden ayrılmaktadır.

KAYNAKLAR

[1] **Vogt, J.** (2008). A rule-based solution supporting intelligent and adaptive agents, *Master Thesis*, Software Engineering Group, Department of Informatics, University of Fribourg, Switzerland.

[2] **Hill, E. F.** (t.y.). *the Rule Engine for the Java Platform*. Alındığı tarih: 06.12.2012, adres: http://www.jessrules.com/jess/index.shtml

[3] **Sim, K. M.** (2004). Toward an Ontology-enhanced Information Filtering Agent. *SIGMOD Record.* Cilt **33**, sayı. 1, Sf. 95-100.

[4] **Miller, G. A.** (1990). Wordnet: an on-line lexical database. *Int. J. Lexicography.* Cilt **3**, sayı. 4, Sf.235-312.

[5] **Chang, C. H. ve Huo, K. H.** (2011). Mobile advertising: triple-win for consumers, advertisers and telecom carriers, *ACM SIGIR 2011 Workshop on Internet Advertising*, Beijing, China, 28 Temmuz.

[6] **Google Maps API.** (2012). Alındığı tarih: 04.12.2012, adres: https://developers.google.com/maps/

[7] **Matheson, D.** (2004). A rule-based approach to publish/subscribe, *Master Thesis*, Department of Electrical and Computer Engineering, University of Toronto.

[8] **Fidler, E.** (2006). PADRES: A distributed content-based publish/subscribe system, *Master Thesis*, Department of Electrical and Computer Engineering, University of Toronto.

[9] **Forgy, C. L.** (1982): Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence.* Cilt **19**, sayı. 1, Sf. 17-37.

[10] **Beneventi, A., Poggi, A., Tomaiuolo, M. ve Turci, P.** (2004). Integrating rule and agent-based programming to realize complex systems. *WSEAS Transactions on Information Science and Applications.* Cilt **1**, sayı. 1, Sf. 422-427.

[11] **Wooldridge, M.** (2002). An Introduction to MultiAgent Systems. John Wiley & Sons, England.

[12] **Bellifemine, F., Caire, G. ve Greenwood, D.** (2007). Developing Multi-Agent Systems with JADE. John Wiley & Sons, England.

[13] **Wooldridge, M. ve Jennings, N.R.** (1995). Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review.* Cilt **10**, sayı. 2, Sf. 115-152.

[14] **Ontology (information science)**. (2012). In *Wikipedia.* Alındığı tarih: 08.12.2012, adres: http://en.wikipedia.org/wiki/Ontology_(information_science)

[15] **FIPA**. (2001). *FIPA Ontology Service Specification*, alındığı tarih: 06.12.2012, adres: http://www.fipa.org/specs/fipa00086/XC00086D.html

[16] **Caire, G. ve Cabanillas, D.** (2010). Jade Tutorial Application-defined Content Languages and Ontologies, alındığı tarih: 06.05.2012, adres: http://jade.tilab.com/doc/tutorials/CLOntoSupport.pdf

[17] **JADE Home Page** <http://jade.tilab.com>, alındığı tarih: 08.12.2012.

[18] **Caire, G.** (2009). Jade Tutorial Jade Programming For Beginners, alındığı tarih: 06.05.2012, adres:http://jade.tilab.com/doc/tutorials/JADEProgramming-Tutorial-for-beginners.pdf

[19] **Roth, H. F.** (1985). Rule-based systems. *Communications of the ACM.* Cilt **28**, sayı. 9, Sf. 921-932.

[20] **Prolog.** (2012). In *Wikipedia.* Alındığı tarih: 06.12.2012, adres: http://en.wikipedia.org/wiki/Prolog

[21] **CLIPS.** (2012). In *Wikipedia.* Alındığı tarih: 19.08.2012, adres: http://en.wikipedia.org/w/index.php?title=CLIPS

[22] **Drools, The Business Logic integration Platform**. (2012). *Red Hat.* Alındığı tarih: 06.04.2012, adres: http://www.jboss.org/drools

[23] **Drools Expert User Guide** <*http://docs.jboss.org/drools/release/5.3.0.Final/drools-expert-docs/html_single/index.html*>, alındığı tarih: 06.04.2012.

[24] **Android (operating system)**. (2012). In *Wikipedia.* Alındığı tarih: 04.12.2012, adres:http://en.wikipedia.org/wiki/Android_(operating_system)

[25] **IDC**. (2012). *Android Marks Fourth Anniversary Since Launch with 75.0% Market Share in Third Quarter*, alındığı tarih: 03.11.2012, adres:http://www.idc.com/getdoc.jsp?containerId=prUS23771812#.UM3jq29mhDQ

[26] **Moreno, A., Valls, A. ve Viejo, A.** (t.y.). Using JADE-LEAP to implement agents in mobile devices, Multi-Agent Systems Group, University Rovira i Virgili, Tarragona, Spain.

[27] **Lee, T. B., Hendler, J. ve Lassila, O.** (2001). The semantic web. *Scientific American.* Cilt **284**, sayı. 5, Sf. 35–43.

[28] **Van Aart, C. J.** (2008). Ontologybeangenerator, alındığı tarih: 06.05.2012, adres:http://protege.cim3.net/cgi-bin/wiki.pl?OntologyBeanGenerator

[29] **The Reflection API** <http://docs.oracle.com/javase/tutorial/reflect/index.html>, alındığı tarih: 06.12.2012.

ÖZGEÇMİŞ

Ad Soyad: Resul ŞAHİN

Doğum Yeri ve Tarihi: Elazığ / 11.05.1984

Adres: Çamlık Mah. Fatih Cad. No: 28/4 Çekmeköy / İstanbul

E-Posta: resulsahin@gmail.com

Lisans: İstanbul Teknik Üniversitesi / Bilgisayar Mühendisliği

Mesleki Deneyim ve Ödüller:

2007 – 2008 Nortel Networks Netaş

2008 – 2009 Tübitak MAM Bilişim Teknolojileri Enstitüsü

2011 – 2012 ACM Yazılım Çözümleri

TEZDEN TÜRETİLEN YAYINLAR/SUNUMLAR

- **Sahin R., and Erdogan N.,** 2012: An Agent-based Framework for Personalized Advertisement on Mobile Devices, *The Second IEEE International Conference on Parallel, Distributed and Grid Computing*, December 6-8, 2012 Waknaghat, India.