İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

KULLANICI KESİTLERİYLE YÜZ İFADELERİNİ ANALİZ EDEN BİR ÇOKLU ETMEN SİSTEMİ UYGULAMASI

YÜKSEK LİSANS TEZİ

Müh. Sanem SARIEL

Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ

Programı : BİLGİSAYAR MÜHENDİSLİĞİ

OCAK 2002

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

KULLANICI KESİTLERİYLE YÜZ İFADELERİNİ ANALİZ EDEN BİR ÇOKLU ETMEN SİSTEMİ UYGULAMASI

YÜKSEK LİSANS TEZİ
Müh. Sanem SARIEL
(504991099)

Tezin Enstitüye Verildiği Tarih : 02 Ocak 2002
Tezin Savunulduğu Tarih : 17 Ocak 2002

Tez Danışmanı :	**Doç.Dr. B. Tevfik AKGÜN (Y.T.Ü.)**
Diğer Jüri Üyeleri	**Prof.Dr. A. Emre HARMANCI (İ.T.Ü.)**
	Prof.Dr. Ümit AYGÖLÜ (İ.T.Ü.)

OCAK 2002

ÖNSÖZ

Tez çalışması kapsamında yürüttüğüm çalışmanın mühendislik formatına uygun yönde şekillenmesi ve çalışmamı bu yolda ortaya koymam için destek veren hocam Doç. Dr. B. Tevfik AKGÜN' e, çalışmanın ilk şeklini alması aşamalarında yöntemler konusunda bana yol gösteren Araş. Gör. A. Şima Etaner UYAR' a ve lisans ve yüksek lisans dönemimde bana değerli katkılar sağlayan bölümümüz hocalarına teşekkürlerimi sunarım.

Çalışmalarım esnasında büyük sabır göstererek bana sürekli destek olan annem Necla SARIEL ve kardeşim Selen SARIEL' e teşekkürlerimi sunarım.

Çalışmalarım esnasında desteklerini eksik etmeyen dostlarım Araş. Gör. Tülay DEMİR, Araş. Gör. M. Derin HARMANCI, Araş. Gör. M. Siraç ÖZERDEM, Devrim AKTAŞ ve tüm arkadaşlarıma teşekkürlerimi sunarım.

Yoğun ders programlarına rağmen uygulama programımı kullanarak bana destek veren öğrenci asistanlarımız ve bölümümüz 3. ve 4. sınıf öğrencilerine teşekkür ederim.

Ocak 2002 Sanem SARIEL

İÇİNDEKİLER

KISALTMALAR

ALIVE	: Artificial Life Interactive Video Entertainment
AMR	: Agent Mesage Router
ANS	: Agent Naming Server
AU	: Action Unit
FACS	: Facial Action Coding System
FCM	: Fuzzy c-Means
HCM	: Hard c-Means
JATLite	: Java Agent Template, Lite
JVM	: Java Virtual Machine
KIF	: Knowledge Intercange Format
KQML	: Knowledge Query and Manipulation Language
KSE	: Knowledge-Sharing Effort
MANTA	: Modeling an ANTnest Acitivity
MAS	: Multi Agent Systems
MC	: Monte Carlo Yöntemleri
MICROB	: Making Intelligent Collective Robotics
OpenGL	: Open Graphics Library
TD	: Temporal Difference

TABLO LİSTESİ

ŞEKİL LİSTESİ

SEMBOL LİSTESİ

s_t	:	t anında ortam durumu
S	:	Tüm durumlar kümesi
a_t	:	t anında etmenin davranışı
A	:	s_t durumunda gerçeklenebilecek tüm davranışlar kümesi
r_{t+1}	:	$t+1$ anında etmenin aldığı ödül
R_t	:	Ödül fonksiyonu
$P_{ss'}^{a}$	:	s, a çifti için bir sonraki s' durumunun olasılığı
$R_{ss'}^{a}$	:	s, a çifti ve s' için sonraki ödülün beklenen değeri
$V^{\pi}(s)$	:	s durumundan π yöntemi izlendiğinde beklenen değer
$Q^{\pi}(s,a)$	:	s durumunda a davranışını gerçekleyip, π yönteminin izlenmesi durumunda beklenen değer
c	:	Küme sayısı
X	:	n adet veriden oluşan veri topluluğu
x	:	Tek bir veri örneği
x_j	:	x örneğinin tekil skaler bileşeni
m	:	Özellik uzayı boyutu
A_i	:	i etiketli bölüt (Kesin bölütleme)
$\tilde{A}_i$	:	i etiketli bölüt (Bulanık bölütleme)
A	:	Bölütler topluluğu
v_i	:	i. bölüt merkezi
$\chi_{A_i}(x_k)$	:	x_k verisinin A_i bölütüne ait olup olmadığını belirten değer

M_c	:	Kesin (Hard) bölütleme uzayı
U	:	Verilerin bölütlere üyeliğini gösteren matris
μ_{kj}	:	k verisinin j . bölüte üyelik derecesi
η_{Mc}	:	Kesin bölütlemeler sayısı
$J(U,v)$	:	Hedef foksiyonu (Kesin Bölütleme)
d_{ik}	:	x_k ve v_i arasındaki Euclid uzaklığı ölçüsü
M_{fc}	:	Bulanık (Fuzzy) bölütleme Uzayı
$J_m(\underset{\sim}{U},v)$	:	Hedef foksiyonu (Bulanık Bölütleme)
m'	:	Ağırlık parametresi (Bulanıklık miktarını belirler)
ε_L	:	Tolerans katsayısı
$\rho(U;2)$	:	Ayrılma derecesi
$F(U;c)$	:	Bölütleme katsayısı
i_c	:	Etmen etkilenme katsayısı

ÖZET

İletişimin en önemli araçlarından biri olan yüz ifadelerinin analizi, sosyal bilimlerin en çok ilgi çeken konularından biridir. Bu konu, bilgisayar mühendisliği açısından da oldukça geniş bir çalışma alanı yaratmıştır. Yüz ifadelerinin tanınması ve yeniden oluşturulması gibi uygulamalar için yürütülen analiz çalışmaları oldukça önem taşımaktadır.

Bu tez çalışmasının amacı, çeşitli durumlar için yüz ifadesindeki genel parametrik değerleri belirleyen bir sistem sunmaktır. Bu amacı gerçeklemek üzere toplum kesitine ilişkin yüz ifadelerinin oluşturulabilmesi için bir çoklu etmen sistemi tasarlanmıştır. Çoklu etmen sistemindeki tüm birimler birbirleri ile etkileşim halindedir. Sistem, kullanıcıları temsil eden Kullanıcı Arayüz Birimleri, hazır bir Grafiksel Yüz Animasyon Programı, bir Bölütleme Etmeni ve çoklu etmen sistemini gerçeklemek üzere etmen oluşturma aracı bileşenlerinden oluşur. Etmen oluşturma aracı olarak JATLite ve bileşenleri kullanılmıştır.

Sistem, çalışması sonucunda topluma ilişkin genel bilgiler ile tek tek kullanıcılara ilişkin yerel inançları oluşturulabilmektedir. Sistemde işlenen veriler, yüz ifadelerine ilişkin parametrelerdir. Toplum kesitleri ve yerel etmen inançları oluşturulmak üzere kullanıcıların çizimleri ve diğer kullanıcı çizimlerine verdikleri puanlar değerlendirilmektedir.

Her bir kullanıcı, kendisine atanan Kullanıcı Arayüz Birimi ile etkileşim kurar. Grafiksel Yüz Animasyon Programı, tüm parametrik yüz ifadelerini kullanıcıya gösterebilmek için kullanılmıştır. Kullanıcı Arayüz Birimi, reaktif nesneler ve temel bir birim olarak Kullanıcı Arayüz Etmenini içerir. Kullanıcı Arayüz Etmeni, çoklu etmen sistemi içinde etkileşimi sağlar. Etmenler arası iletişim, KQML mesajlarının alış-verişi ile gerçeklenmektedir.

Etmenlere ilişkin tüm kayıtlar JATLite yönlendirici bileşeni tarafından tutulur. Etmenler, sisteme dahil olabilmek için yönlendiriciye kullanıcı adı ve şifresi ile kaydolurlar. Etmenler, KQML mesaj içerikleri konusunda önceden belirledikleri bir protokole göre el sıkışmış olmalıdır. Bu işlem, sistem ontolojisinin belirlenmesi olarak değerlendirilir.

Sistemdeki tüm parametre değerlerini toplayıp analiz eden birim, Bölütleme Etmenidir. Bölütleme Etmeni, sistemin en akıllı birimi olarak hem öğrenme hem de bölütleme süreçlerini gerçekler. Bölütleme süreci öncesinde parametre ağırlıklarının belirlendiği öğrenme süreci gerçeklenir. Bölütleme Etmeni öğrenme sürecinde Q Öğrenmesi yöntemini kullanır. Sistemdeki tüm kullanıcılar, öğrenme sürecine dahil olur ve Bölütleme Etmeni tarafından üretilen yüz ifadeleri için eleştirmen olarak görev yapar. Bölütleme sürecinde bir veri örneği içindeki her bir parametre farklı bir boyut belirtir. Bölütleme süreci için FCM, Complete-link ve Single-link algoritmalarından herhangi biri seçilebilir. Bölütleme sonuçları, tekil yüz ifadeleri veya çeşitli örneklerin aralıklı gösterilmesi ile oluşturulan yüz animasyonları şeklinde ifade edilir.

Bu tez çalışması, Java sınıfları paketlerinden oluşur. Temel birimler, sistemin temel sınıflarını oluşturur. Bu sınıflar etkin çalışabilmek üzere çok sayıda alt sınıfı canlandırır. Hem birim içi sınıflar arasında hem de etmenler arasında etkileşime imkan tanıyan kapsamlı bir sistem gerçeklenmiştir.

A MULTI AGENT SYSTEM APPLICATION ANALYZING FACIAL EXPRESSIONS WITH USER PROFILES

SUMMARY

Analyzing facial expression, an important tool of human communication, is one of the most attractive field of social sciences. Analysis of human face and facial expressions is also important for computational efforts. Many applications have one of the requirements of either the recognition of faces of persons or the reconstruction of faces. Some researchers in the fields of computer graphics and pattern recognition have proposed many works including analyzing and synthesizing facial image sequences, talking facial display generation by analyzing visemes, and facial expression analysis by using FACS coding system.

In this work, a multi agent system is presented to generate society profile about facial expressions. The overall system work is done by cooperating agents living in the system. The aim of the work is to use the designed system to determine some control parameters of an animated face for some situations. The information processed in the system is the parameters related to facial expressions. The parameters are processed for generating general society profile and local agent beliefs according to the facial expressions generated by the users and the grades assigned to the expressions. Consequently, general society profile and some exception situations related to facial expressions are represented by clusters.

The system consists of users, User Interface Units consisting User Interface Agents communicating with users, The Clustering Agent responsible for generating society profile, existing multi agent system library and architecture components. The JATLite is used as an agent construction tool.

The User Interface Agents are constructed as JATLite agents. Each agent has ability to communicate with the Router and the other agents in the system. The User Interface Agent interacts with its user to collect information about the facial expressions related to some questions. It has reactive objects inside it. Each reactive object is responsible for a special muscle parameter. An Agent Server (server of the reactive objects) is implemented in The User Interface Unit to organize behaviors of the reactive objects on the environment. The effects of the reactive objects can be observed from a text file. This text file contains information about facial muscle parameters. The parameters are stored as left and right Zygomatic_Major, left and right Angular_Depressor, left and right Frontalis_Inner, left and right Frontalis_Major, left and right Frontalis_Outer, left and right Labi_Nasi, left and right Inner_Labi_Nasi, left and right Lateral_Corigator, left and right Secondary_Frontalis which are related muscle names. The text file is read at certain time intervals by a Graphical Facial Animation Program. This program generates facial expressions related to the parameters written on the text file. This program, an open source C implementation, was used as a facial generator in the thesis work. Some manipulations were needed on this program for the designed system. The original program uses parameters changed by the user with keyboard instructions. It

uses the Glaux Library which does not support the timer function. In the system, the parameter values should be changed by user interrupts. The text file consisting of facial parameters is updated according to these interrupts. Therefore, new facial parameters should be read at certain time intervals to show the updates on the facial model. Timer function should have been added to implement this work. So the program is converted to use Glut Library consisting the timer function and supporting the texture mapping.

The work of reactive objects and The Agent Server unit plays an important role on generating user profile. The reactive objects and The Agent Server communicate on sockets created by The Agent Server for each reactive object. The communication between the reactive objects and The Agent Server is synchronous. Therefore, a communication subsystem is implemented in The User Interface Agent. The User Interface Agent related to a user communicates with the other agents in the system by means of a AgentJATLite Server unit. Each coming KQML message is processed by this unit. The information transfer process is ensured by this unit. All The User Interface Unit components must work at a organizational level. The information about other users must be shown to the user and the grade assigned by the user must be taken correctly. The AgentJATLite Server unit registers itself to the multi agent environment by a username and a password. After registration, it connects to the system. Every agent in the system has an influence constant which determines how much this agent is influenced by the other agents in the system. After determining the user profile and grading some other facial expressions drawn by the other users, The User Interface Agent can generate its own beliefs. The beliefs of the agents are generated by the own profile and the information of the other users and the related grades. These beliefs are formed as results of The Clustering Process. The clustering process can be implemented by FCM, Complete-link and Single-link algorithms. All of these algorithms are implemented and compared to each other in the proposed system.

The JATLITE Router, the address of which all of the agents registering to the system should know, is alive during the system work. Every User Interface Agent registers and connects to this unit to play a role in the multi agent environment. Communication in the environment can be implemented both synchronously and asynchronously. All messaging processes are implemented by JATLite Router unit. Each AgentJATLite Server is responsible for processing these messages. So handshaking must be implemented about all the message commands. This determines the ontology. In the designed system a messaging protocol is implemented and the messages are processed effectively, and the contents are processed according to the performatives and the commands. The listing of existing registered and connected agents service is provided by the JATLite Router. This facilitates the agents to be informed about the other agents in the system. Therefore direct or indirect (through The Router) connections between agents are implemented.

The Clustering Agent is an intelligent unit in the multi agent environment. This unit collects all the users information and analyzes it. The Clustering Agent clusters all the data samples by using the FCM, Complete-link and Single-link algorithms. The data samples are the arrays of facial parameters having eighteen dimension. Each parameter indicates a value on each dimension. This clustering process is implemented effectively by using weight values for each parameter. A learning process is implemented for these weight values to be calculated. Q Learning method is used for this process.

All the agents play a role in the learning process. The users act as critics for the learner in the environment. The Clustering Agent has ability to generate some facial expressions according to the states of the learning process. The states are modeled as parameter weight arrays. A facial expression is generated related to the new state. Each new facial expression is sent to a User Interface Agent. The user of the current User Interface Unit grades this facial expression. This grade is sent as a reply to The Clustering Agent. The Clustering Agent uses this information and the expected value to generate the reward of the environment. By using this reward value and the previous Q values, the new Q values are calculated. The Q values are updated at each grading step. After many steps, the Q values of some states regularly increase more than the other states. These states correspond to the learned parameter weights. This learning process is performed for each situation. The user society is a dynamic and non-deterministic environment. Because different users assign different grades to the same facial expression, the same state and the action pair does not result from the same reward. To implement the learning task, many trials are needed. After learning parameter weights, The Clustering Agent broadcasts this information to all of the agents in the system. Therefore the consistency of parameter weights is ensured. If an agent can not get the message containing parameter weights, it can request this information from any agent in the system.

The clustering process is implemented on the data samples by using the learned parameter weights. The clustering results are different for different clustering methods. All the clustering methods forms optimum number of clusters of data samples. In the FCM as a partitional algorithm, data samples are assigned to the clusters by membership values. The algorithm runs by making iterations on the cluster assignments. The final results are obtained after some iterations. The final membership values of data samples are ranged between [0,1]. The defuzzification process can be implemented on the results. The final results are cluster centers and the membership matrices of the data samples. These results are stored in text files for further analyses. The clustering process starts with that each data sample is a cluster in the Complete-link and the Single-link algorithms. The algorithms work by merging of clusters effectively. The difference between the Complete-link and The Single-link algorithm is in the calculation of inter-cluster distances. The final dendograms can be cut at the level having optimum cluster numbers. The final membership values are stored in text files. The final clustering results can be shown to user as facial expressions or facial animations (sequential show of facial expression samples) consisting some samples in clusters.

This thesis, is implemented as packages of Java classes. The main units are implemented as the main classes of the system. These classes invoke many subclasses to work effectively. A complex system having interactions among agents and among subclasses of agents is designed, and an effective multi agent system analyzing facial expressions is presented.

1. GİRİŞ

Bir çok bilim dalında, teknoloji geliştirme sürecinde doğa ve doğal süreçler örnek alınmaktadır. Doğada gerçekleşen olaylar, doğal nesneler ve varlıklar, karmaşık yapıları ile oldukça etkin modeller oluşturmaktadır.

Bilgisayar bilimlerinde de çalışmalar, doğal varlıklara yakın makinelerin yapılması yönünde yıllardan beri sürdürülmektedir. Karmaşık modeller sunan insan ve hayvanların yapısı ve davranışları incelenmekte ve bunlardan örnek alınarak bilgi işlemsel sistemler geliştirilmektedir.

Örnekler, doğadan birebir olarak alınmayabilir. Uçakların kanat çırpmadığı gibi insan veya hayvan davranışlarını örnek alan bilgi işlemsel birimler de tamamıyla yaşayan varlıklar olarak ortaya çıkmayabilir. Davranış bilimleri ve bilgisayar bilimleri konusunda uzman olan kişiler, insanlara benzer davranışlar gösteren yazılım veya donanım birimlerinin yapılıp yapılamayacağı konusunda henüz bir uzlaşmaya varamamışlardır.

Doğal sistemleri örnek alan sistemler, karmaşık bilgi-işlem gereksinimlerini karşılamak üzere uygun araçlar sunarlar. Son dönemde yeni bir çalışma alanı olarak sunulan etmenler ve çoklu etmen sistemleri bu türden doğaya yakın sistemlere örnek gösterilebilir. Bu sistemleri gerçeklemek üzere, etmenlere, insan veya hayvanların tepkisel özelliklerine sahip akıllı birimler olma yetenekleri yüklenmiştir. Bu birimler, bulundukları ortam ile ve bu ortamdaki diğer varlıklar ile haberleşerek kendi varlıklarını devam ettirmekte ve yürütmeleri gereken görevleri yerine getirmektedir. Etmenler ve etmen sistemleri yapay zeka ve bilgisayar bilimleri açısından önemli bir yere sahip olmuştur. Donanımsal ve yazılımsal olarak etmenler, karmaşık sistemlerin oluşturulmasında önemli rol oynamaktadır.

Bu tez çalışmasında, etmen mimarileri incelenmiş ve bir çoklu etmen yapısı oluşturularak uygulama geliştirilmiştir. Bu çalışma, çoklu etmen sistemleri kullanılarak gerçeklenebilecek daha geniş kapsamlı çalışmalar için bir adım niteliği taşımaktadır. Bu tür sistemlerle gerçeklenebilecek ve insan beynine yakın

karmaşıklıkta çalışan sistemler kurabilmek için çalışmaların sürdürülmesi gerekmektedir.

Çocuklar üzerinde denenmesi amaçlanan bu uygulamada, kullanıcılardan bir yüz animasyon programını kullanarak çeşitli sorulara ilişkin yüz ifadeleri oluşturmaları istenmektedir. Kullanıcı kesitlerini oluşturmak, birçok çalışmada çeşitli kazançlar sağlamaktadır. Bu çalışmada da her bir çocuğa ilişkin kesitler belirlendikten sonra sistem, bu kesitler yardımıyla toplumun çeşitli sorulara ilişkin belirlediği yüz ifadelerini değerlendirmekte, toplum kesitini ve tek tek etmenlerin genel inançlarını oluşturmaktadır. Etmen inançlarını oluşturmak üzere, her bir etmenin ilişkide bulunduğu kullanıcıya diğer kullanıcının çizimleri sunulmakta ve bunlara karşılık çeşitli puanlar alınmaktadır. Toplum kesitini oluşturmak için sistemdeki tüm veriler değerlendirilerek bilinen bölütleme algoritmaları ile bu kullanıcı bilgileri bölütlenmektedir. Bölütleme Sürecini etkin olarak gerçeklemek üzere sistemin oluşturduğu yüz ifadelerine, kullanıcıların puan vermesi sağlanarak, belli parametrelerin sistem tarafından öğrenilmesi sağlanmaktadır (destekli öğrenme). Böylelikle öğrenme, bölütleme ve sistem içinde etkileşme süreçlerini içeren kapsamlı bir sistem oluşturulmuştur.

Bu sistemi gerçeklemek üzere çoklu etmen sistemleri, bölütleme algoritmaları ve öğrenme algoritmaları incelenmiş ve uygun yöntemler kullanılarak yüz ifadelerinin analizini yapan bir sistem oluşturulmuştur. Sistem yazılımı, JATLite Kütüphane bileşenleri, bir Grafiksel Yüz Animasyon Programı ve Java dilinde tasarlanan sistem bileşenlerden oluşmaktadır.

II. Bölümde etmenler ve çoklu etmen sistemleri üzerine genel bilgiler verilmektedir. III. Bölümde çoklu etmen sistemlerinde öğrenme ve özellikle destekli öğrenme yöntemine değinilmektedir. IV. Bölümde Bölüt Analizinden bahsedilmekte ve bu işlemi gerçeklemeye yönelik algoritmalar tanıtılmaktadır. V. Bölümde yüz animasyonu ile ilgili yapılan çalışmalar tanıtılmaktadır. VI. Bölümde tez kapsamındaki yazılım sisteminin oluşturulması için kullanılan Java programlama dilinin sunduğu hizmetlerden kısaca bahsedilmektedir. VII. Bölümde tez kapsamında yürütülen çalışma ve gerçeklenen sistem tanıtılmaktadır. VIII. Bölümde gerçeklenen sistemin yazılım mimarisi ve sınıf hiyerarşisi anlatılmaktadır. IX. Bölümde program sonuçları ve programın kullanıcı arayüzü tanıtılmaktadır. X. Bölüm sonuç ve tartışma bölümüdür.

2. ETMENLER VE ÇOKLU ETMEN SİSTEMLERİ

2.1 Giriş

Bu bölümde etmenler, etmenlerin özellikleri ve modelleri ve etmen uygulamaları tanıtılmıştır. Etmenlerin oluşturduğu topluluklar olan çoklu etmen sistemleri tasarımında önemli noktalar verilmiştir. Etmen oluşturma araçları incelenmiş ve tez kapsamında gerçeklenen sistemde kullanılan etmen oluşturma aracı ayrıntılarıyla tanıtılmıştır.

2.2 Etmen

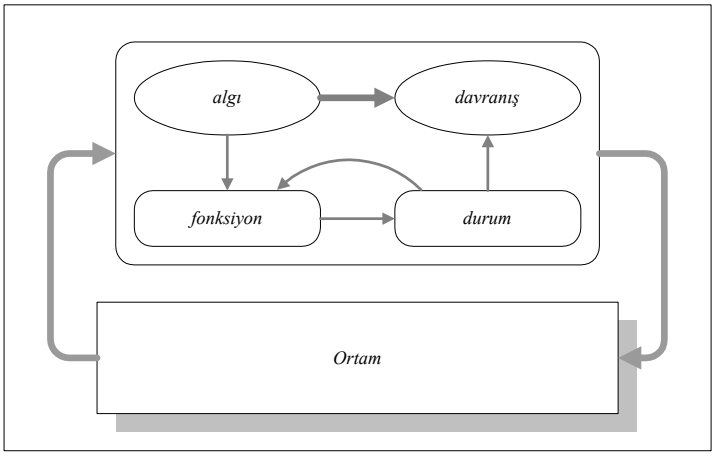
Etmenler, bir çok uygulama ve sistemde kullanılmalarına rağmen henüz tanımları üzerine bir uzlaşmaya varılamamıştır. Etmen sistemleri ve mimarileri üzerinde çalışan çeşitli üniversiteler ve kuruluşlar, geliştirdikleri uygulamalar ve sistemler için çeşitli tanımlar sunmuşlardır. Tanımların farklı olması, tasarlanan etmenlerin yapılarının, tasarım amaçlarının ve yaşam modellerinin farklı olmasından kaynaklanmaktadır.

Genel bir tanıma ulaşmak için, etmenlerin çıkış noktası olan basit veya karmaşık canlı sistemleri göz önüne alınabilir. Etmenler, canlıların içinde bulundukları ortamlardan duyu organları yoluyla algıladıkları verileri değerlendirip bilgi ve inançları doğrultusunda bir davranış sergilemesi modelini örnek alarak tasarlanmış birimlerdir. Etmenlerin karmaşıklığı, gerçeklenen uygulamaya göre sadece tepkisel olanından son derece karmaşık yeteneklere sahip ve işlemsel gücü oldukça yüksek olanına kadar çeşitli derecelerde olabilir. En çok benimsenen etmen tanımları şu şekilde verilebilir.

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

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

Etmen, içinde bulunduğu ortam üzerinde tam veya kısmi bir kontrole sahip olabilir ve ortamını bu dereceye göre etkiler. Etmenlerin farklı anlarda gösterdiği tepkiler, ortamı farklı şekilde etkileyebilir veya istenilen etkiyi yaratmayabilir. Etmen için en önemli problem, amaçlarını gerçeklemek üzere davranış repertuarından hangi davranışları seçmesi gerektiğine karar vermesidir. Şekil 2.1'de etmen yapısı ve ortamı ile etkileşim dinamiği görülmektedir.



Şekil 2.1 Etmen Yapısı ve Ortamı İle Etkileşim Dinamiği (Weiss, 2000)

2.2.1 Etmen Ortamı

Etmen tanımında bahsi geçen ortam, erişilebilme ya da tümüyle erişilememe, determinist ya da determinist olmama, bağımlı ya da ayrık olaylar içerme, statik ya da dinamik olma, ayrık ya da sürekli olma özelliklerine sahip olabilir (Weiss, 2000).

- Erişilebilen / Erişilemeyen: Erişilebilen bir ortamda etmen, ortam durumu hakkında tümüyle doğru ve güncel bilgiye sahiptir. Karmaşık ortamlar (örneğin canlıların içinde bulundukları ortam, internet vb.) genellikle tümüyle erişilebilir değildir.

- Determinist / Determinist Olmayan: Etmenin bir hareketinin ortam üzerindeki etkisi (gerçek dünyada olduğu gibi) her zaman aynı sonucu doğurmayabilir. Bu durumda ortam determinist değildir.

- Bağımlı / Ayrık Olaylar İçerme: Ortam ayrık olaylardan oluşuyorsa etmenin başarımı bu olaylara bağlıdır. Bu olaylar arasında bir bağlantı yoktur (Örn: e-posta sıralama sistemi). Karar için sadece o anki durum değerlendirilir. Olaylar arasında bağlantı kurmaya gerek yoktur.

- Statik / Dinamik: Ortam, statik ise etmenin etkileri olmadığında sabit kalan bir ortamdır. Ortam, dinamik olduğunda ortam üzerinde başka etmenler veya süreçler çalışmaktadır ve değişimler etmen kontrolü dışındadır.

- Ayrık / Sürekli: Belirli sabit sayıda davranış ve algı söz konusu ise ortam ayrıktır. Örneğin satranç oyununda etmen, ayrık bir ortamda, taksi şoförlüğünde ise sürekli bir ortamda bulunmaktadır.

Gerçeklenmesi en zor ortamlar, tümüyle erişilemeyen, determinist olmayan, bağımlı olaylardan oluşan, dinamik ve sürekli ortamlardır (Weiss, 2000).

2.2.2 Etmen Modeli

Durum tabanlı etmenin çalışma modelinde tanımlı değişkenler:

I: Etmenin durumlar kümesi, P: Etmenin algıladığı veri, S: Ortam durumu

A: Etmen davranışı olmak üzere,

Algı: S→ P

Davranış: I→ A

Sonraki durum : I X P → I geçişleri şeklinde tanımlanmaktadır.

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

2.2.3 Etmen Özellikleri

- Otonom olma: İnsanlar ve diğer uzman sistemler ile doğrudan iletişim kurmadan iş yapabilme ve karar verebilme yeteneğidir.

- Sosyal olma: Ortamdaki etmenler ve kullanıcılar ile iletişim kurabilme yeteneğidir. Etmen iletişim dillerini kullanırlar.

- Tepkisel davranma: Fiziksel dünya, grafik kullanıcı arayüzü ve diğer etmenlerden oluşan bir topluluk, internet veya bunların bir kombinasyonu olan ortamdan aldıkları bilgiye göre ortama tepki verirler.

- Amaca yönelik çalışma (Pro-activeness): Tasarımları esnasında veya yürütme anında belirlenen görevlerini yerine getirirler. Tepkisel davranma özelliği ile dengelenmesi gereken bir özelliktir.

- Zamanda sürekli olma: Sistemde sürekli canlı bulunma özelliğidir.

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

- Öğrenebilme yeteneği: Etmenin daha önce sahip olmadığı özellikleri ve yetenekleri veya tasarım amaçlarını gerçeklemek üzere işine yarayan bilgileri öğrenebilmesi özelliğidir.

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

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

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

Bu özelliklerden ilk dördü, etmenler için temel olan özelliklerdir. Diğer özellikler. tasarımcı tarafından uygulamaya göre eklenir. Etmenler, ortamı üzerinde bilgiye ait çeşitli davranışlar sergileyen bilgi işlemsel birimler olarak istek, niyet, yükümlülükler, seçimler gibi içsel duygulara da sahip olabilir. Bilgiye ait davranışlar etmenin ortamına ait sahip olduğu düşüncelerdir. İçsel duygular, etmenin ortam üzerindeki davranışlarını etkiler.

2.2.4 Etmen-Nesne Ayrımı

Nesneler, çeşitli durumlarda olabilen, durumlara ilişkin metotları olan ve mesaj aktarma ile haberleşen bilgi işlemsel birimlerdir. Etmenleri nesnelerden ayıran en önemli özellikler:

- Etmenlerin ve nesnelerin özerklik dereceleri farklıdır: Nesnelerin metotları “public”, “protected” veya “private” olarak tanımlanabilmesine karşın diğer nesneler tarafından çağrılabilir. Kararı, çağıran nesne verir. Etmenlerde ise istekte bulunulan etmen isterse isteği reddedebilir. Kararı, hizmete sahip olan etmen verir.

- Nesneler, tepki gösterme, sosyal olma gibi özelliklere sahip değildir.

- Nesneler, tek başına denetime sahip değildir ve içinde bulundukları sistemin denetimi altındadır. Nesneler arası mesajların anlamı nesneden nesneye değişmesine rağmen etmenler arasında değişmez.

2.2.5 Etmen Türleri

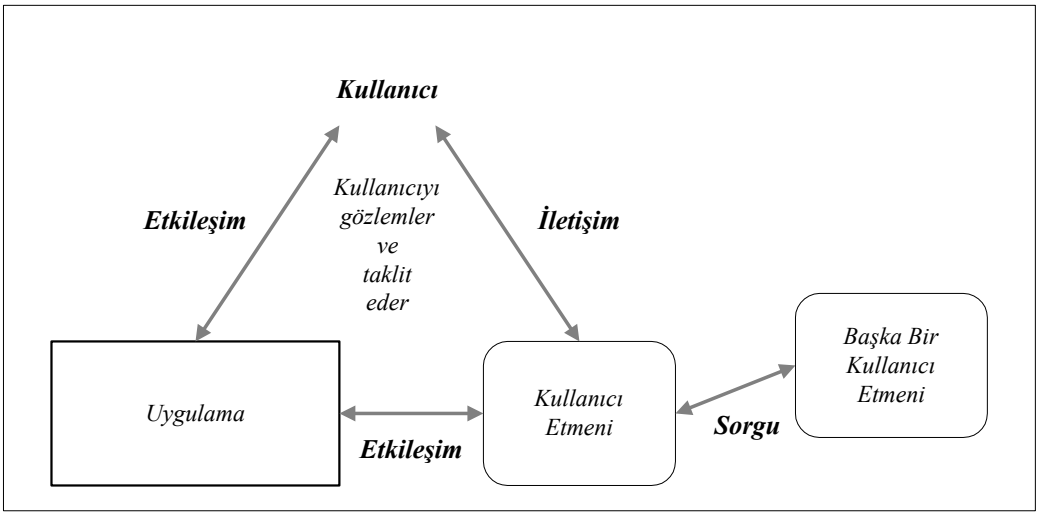
- Özerk Etmenler: Karmaşık ve dinamik ortamlarda kendi kendilerine belli bir görevi yerine getiren etmenlerdir.

- Simülasyon Etmenleri: Çeşitli ortamların veya sistemlerin simülasyonu için kullanılan etmenlerdir. Eğlence amaçlı oyunlarda kullanılan etmenler de bu sınıfa girer.

- Bilgi Toplayıcı Etmenler: İnternet gibi bir bilgi ağında kaynaklara etkin şekilde erişip kullanıcı isteklerine göre bilgi toplayan etmenlerdir. Diğer etmenlerle de haberleşerek bilgiye daha çabuk erişmek için gerekli olan bilgileri öğrenirler. Kullanıcı ile sürekli etkileşim kurarak kullanıcısının isteklerini öğrenmek durumundadırlar.

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

- Arayüz Etmenleri: Kullanıcı ile aynı ortamda işbirliği yapan kişisel yardımcılardır. Kullanıcının davranışlarını gözlemler, raporlar ve öğrenme ile yeni yetenekler kazanır. Kullanıcısına gerekirse daha iyi yollar önerebilir. Öğrenme süreci, kullanıcıyı gözlemleyerek veya taklit ederek (supervised öğrenme), kullanıcıdan iyi ya da kötü şekilde geri besleme alarak (destekli öğrenme), kullanıcıdan komutlar alarak veya diğer etmenlerden bilgi sorma ile yardım isteyerek yürütülebilir. Şekil 2.2’de arayüz etmeni etkileşim dinamiği görülmektedir.



Şekil 2.2 Kullanıcı ile Etkileşim Kuran Arayüz Etmeni (Maes, 1994)

- Ortak Çalışan Etmenler: Kendi başlarına görevlerini yürütebilmek için özerkliğe sahip olan ve diğer etmenlerle birlikte çalışan etmenlerdir.

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

- Reaktif (Tepkisel) Etmenler: Bireysel olarak çok yetenekli olmayan ve sadece bulundukları ortama tepki gösterebilen etmenlerdir. Ortam hakkında bilgi tutmazlar. Doğal çoklu etmen sistemleri oluşturma, yapay yaşam (Artificial Life) sistemleri için uygun bir araçtırlar. Bireysel olarak akıllı değildirler. Fakat oluşturdukları topluluk, yararlı işler yapmak üzere çalışabilir. Hayvan kolonilerinin çalışma modelleri örnek alınmıştır.

- Hibrit Etmenler: Birden fazla etmen tipi özelliğini bir arada sunabilen etmen türleridir.

2.2.6 Etmen Uygulamaları

Etmenler, oldukça geniş bir uygulama alanında kullanıcı ve sistem gereksinimlerini karşılamak üzere hizmet verirler.

- Kullanıcıya yardımcı olan uygulamalar: Kullanıcı ile etkileşim kurarak çalışmalarını yürütürler. Kullanıcıların yaratıcılığının artırılmasına yardımcı olurlar. Günlük yönetimi, e-posta yönetimi, bellek kullanımında yardımcılık

türünden işler yapan uygulamalardır. Diğer etmenlerle haberleşerek bilgi toplarlar. Standart kullanıcı arayüzlerinden farklıdırlar. Yarı-özerk olarak çalışabilirler. Kullanıcı kesitlerini öğrenen sistemler, kişisel sayısal yardımcılar (sayısal telefon sekreteri, randevu düzenleme, e-posta yanıtlama uygulamaları, e-posta süzgeçleri) türünden uygulamalar bu sınıfa girer. Kamera aracılığı ile ortamdan ve kullanıcısından aldığı bilgiyi bir simülasyon ortamına yerleştiren uygulamalar, etmenlerin kullanıcısı ile etkileşim kurmasını sağlar. Kullanıcıya yardımcı olan etmenler, kullanıcının çalışması yanında görevlerini yerine getirebilirler, kullanıcıları eğitebilirler, farklı kullanıcıların ortak çalışmasını sağlayabilirler, olayları görüntülerler. Kullanıcısından sürekli olarak kullanıcı kesitine ilişkin yeni bilgiler öğrenebilen etmenler, bu öğrenme sürecinde kullanıcıdan doğrudan veya dolaylı olarak geri besleme alabilirler (Maes, 1994). Örnek bir uygulama olarak ALIVE (Artificial Life Interactive Video Entertainment) projesinde oluşturulan sanal bir ortamda kullanıcı ve yapay karakteri (etmen) biraraya getirilerek, yapay karakterin davranışlarının, kullanıcının yeri, el ve vücut hareketlerinden etkilenmesi sağlanmıştır. Kullanıcının isteklerini ve buna bağlı hareketlerini algılayan karakter, istenen davranışı yerine getirir (Maes ve diğ., 1995).

- Bilgi Toplama Sistemleri: Kullanıcının bilgiye en kolay ve en hızlı bir şekilde ulaşması için hizmet verirler. Bunun için etmen toplulukları ortak çalışma yürütür. Dizin Hizmetleri, dağıtılmış veri tabanı sorgulama sistemleri, bilgi komisyonculuğu, medya uygulamalarında sıralama ve sayısal kütüphaneler bu türden uygulamalardır.

- Simülasyon Uygulamaları: Oyun, film, etmenlerin rol aldığı tiyatro oyunları, video, reklam üretiminde bilgisayar animasyonu kullanan uygulamalar bu sınıfa girer. Yapay yaşam uygulamaları hem kullanıcıya yardımcı uygulamalar hem de simülasyon uygulamalarına dahil edilebilir. Bu tür uygulamalarda biyolojiden özel olarak etolojiden örnek alınmıştır. Örneğin modellenen toplu davranışlar çeşitli filmlerde animasyon sahneleri olmak üzere gerçekçi çekimler oluşturmaktadır (Maes ve diğ., 1995). Balıkları modelleyen bir uygulamada yapay balık etmeninin ortamı algılaması ve ortamda hareket etmesini söz konusudur. Etmen, ilk başta bırakıldığı ortamda nasıl hareket edeceğini bilmemektedir. Ufak hareketlerinin sonuçlarını görerek ileriye gidebilmek için nasıl davranması gerektiğine karar verir (Bireysel veya topluca öğrenen bir sistem örneğidir.). Tekil olarak yapay bir balığın davranışının modellenmesi yanında balık kolonilerinin

modellenmesi de söz konusudur (Terzopoulos ve diğ., 1994). MANTA (Modelling an ANTnest Acitivity) karınca kolonilerinin davranışlarını modellemek üzere tasarlanan bir sistemdir. Bu şekilde bir modellemenin ve karınca kolonilerindeki karmaşıklığı anlamanın daha gelişmiş sosyal organizasyonlardaki ortak davranışın temelini oluşturmak ve dağıtılmış bilgi işlemede yeni ve pratik yöntemler sunmak üzere iyi bir temel oluşturacağı düşünülmüştür. Karıncalar tepkisel etmenler olarak modellenmiş ve üç tipte etmen tanımı yapılmıştır (Drogoul ve diğ., 1993).

- Hizmet Yönetimi Uygulamaları: Kullanıcılara gereken hizmetin doğru zamanda, doğru maliyetlerle, gerekli olan güvenlik koşullarına uygun şekilde aktarılmasını sağlar. Çoklu Ortam hizmetleri, alım/satım hizmetleri, akıllı ağ yönetim hizmetleri, gezi planlama ve yardımcılık hizmetleri bu sınıfa girer. Kullanıcının isteklerine göre yayın hizmeti sunan bir sistem, her bir kullanıcıya ayrı hizmet vermek yerine kümeleme yöntemlerini kullanarak kullanıcı grupları oluşturur ve benzer isteklere sahip kullanıcıları bir grup içine yerleştirir ve etkin yayın hizmeti sağlar (David ve Kraus, 1999).

- Ticari Uygulamalar: Parasal hizmetler, elektronik ticaret ve veri ambarlığı, iş akışı yönetimi, ofis otomasyonu, İletişim uygulamaları bu sınıfa girer.

- Üretim Yönetimi Uygulamaları: Endüstriyel ortamlardaki işlerin yönetimini gerçekler. Uygulamadaki süreçler endüstriyel robotların yazılım arayüzleri ve makineler ile kontrolünü gerçekler. Endüstriyel robot, fabrika otomasyonu, sanal fabrika yönetimi, yük dengeleme uygulamaları bu sınıfa girer.

- Robot Uygulamaları: Ortam, canlıların üzerinde yaşadığı dünya olunca etmenler de robot olmak zorundadır. Ofislerdeki ve küçük ortamlardaki uygulamalar bu sınıfa girer. Ofis içi kağıt dağıtımı, ev temizleme türünden işleri gerçekleştiren robotlar tasarlanır. Örnek uygulama olarak robotlar arası futbol karşılaşmaları verilebilir. Bu karşılaşmalar, her araştırma grubunun kendi robot takımını her yıl düzenlenen bir turnuvada temsil etmesine olanak tanımaktadır. Robot futbol takımı bir çoklu etmen sistemi olarak modellenmekte ve etmenler, hem yazılım (takım yöneticisi) hem de donanımsal (futbolcu robotlar) olarak gerçeklenmektedir. Böyle bir sistem tasarlamak için grup içindeki bireylerin (etmenler) organizasyon içindeki üstlendikleri görevlerinin iyi belirlenmesi gerekmektedir. MICROB (Making Intelligent Collective Robotics) projesinde robot grupları içinde ortak çalışma ve organizasyon oluşturma çalışması

yapılmıştır. Takım içindeki oyuncular farklı adımlarda farklı roller üstlenmektedir. Fonksiyonel ve temel davranışlar, bağıntılı davranışlar ve bütünsel (organizasyona ait) davranışların belirlenmesi gerekmiştir (Collinot ve diğ.,1996). Son dönemlerde araştırma grupları, bu türden çoklu etmen sistemleri geliştirerek robot takımları oluşturmakta ve kurdukları robot takımları ile uluslararası düzeyde karşılaşmalar düzenlenmektedir. Sadece robotlar arası karşılaşmalar yanında robot-insan karşılaşmaları da son dönemin popüler çalışmaları arasındadır (RoboCup 2002).

- Araştırma Uygulamaları: Çoklu etmen sistemleri gerçeklenerek yürütülen akademik çalışmalar bu sınıfa girer. Görüntü işleme uygulamaları, öğrenen ve adaptif sistemler, ses işleme uygulamaları, dağıtılmış bilgi tabanlı sistemler, insan-bilgisayar arayüzü sistemleri bu sınıfa girer. Arı ve karınca kolonileri ve davranışları örnek alınarak çoklu etmen sistemleri oluşturulmaktadır. Koloni içine giren bir yabancıya gösterilen tepki modellenmekte ve bilgisayar sistemlerinde virüs testi türünden aykırı durum taramaları bu şekilde yapılabilmektedir.

2.3 Çoklu Etmen Sistemleri

Birbirlerine gevşek bağlı problem çözücülerin, kendi başlarına çözemeyecekleri problemleri ortak çalışarak çözmek üzere aynı ortamda bulunmaları sonucu çoklu etmen sistemleri (MAS, Multi-Agent Systems) ortaya çıkar.

Tek başlarına belirli görevleri olan etmenler birlikte çalışarak sistem bütününde sistemden beklenen işi gerçekler. Çoklu etmen sistemleri içinde bulunan etmenler aynı tip olmayabilir (Heterojen sistemler). Etmenler, sistem içinde tek başlarına akıllı birimler olmayabilir. Bir sistemin çoklu etmen sistemi olabilmesi için, etmenlerin ortaklaşa çalışarak bütünde belli bir görevi yürütüyor olması gerekir. Çoklu etmen sistemleri, insan beyninin sahip olduğu özelliklere benzer davranışlar sergiler. İnsan beynindeki sinir hücrelerinin (nöron) tek başlarına fonksiyonu, ancak sistem bütününde diğer sinir hücreleri ile haberleşmesi ile anlamlı olur. Burada tek bir sinir hücresi sadece sinir iletimini gerçekleştirmektedir. Tek bir etmenin işi çok küçük olsa da bütünde gerçeklenen asıl iş için hizmet vermektedir. Tek bir sinir hücresinin zarar görmesi veya sistemden ayrılması sistemin bütününde çok büyük bir değişime sebep olmaz. Dağıtılmış sistemlerdeki ölçeklenebilirlik özelliği bu gereksinimi karşılamak zorundadır. Çoklu etmen sistemlerinde de tek bir etmenin sistemden istemsizce ayrılmasının sistemi ölümcül kilitlenmelere götürmesi ya da sistemin

çökmesine sebep olması söz konusu olamaz. Bu amaçla bu tür önlemlerin çoklu etmen sistemi tasarım aşamasında alınması gerekir.

Çoklu etmen sistemlerinde çözülmesi gereken problemler, etmenler arası iletişimin ne şekilde gerçekleneceği, etmenlerin birbirleri ile etkileşiminin ne şekilde yapılacağı, sistem tutarlılığı ve etmenler arası koordinasyonun nasıl sağlanacağı gibi problemler olup, uygulama geliştirilmesi öncesinde mimari tasarım aşamasında belirlenmesi gereken konulardır (Deepika, 1997).

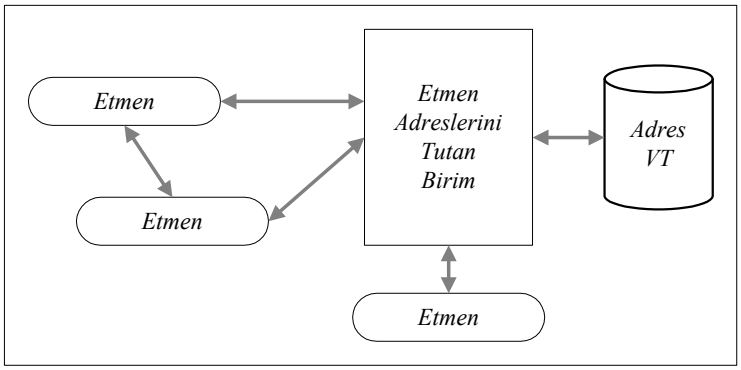
Etmenler, bir çoklu etmen sistemi içinde sistem bütününde bir işi gerçekleyebilmek için diğer etmenlerle haberleşmek, bilgi veya tecrübe aktarımı yapmak zorundadır. Etmenlerin ortak problemler üzerinde paralel çalışabilmeleri gerekmektedir. Etmenler, kurdukları iletişim sayesinde kendilerini yenileyebilmelidirler. İletişim ve etkileşim esnasında sistem tutarlılığı ve koordinasyon problemleri çözülmüş olmalıdır. Tüm bu alt yapı sağlandıktan sonra sistemin hataya dayanıklı olması sağlanmalıdır.

Çoklu etmen sistemi tasarımında, bireysel etmen seviyesinde, hangi tür mimari kullanılacağı, etmenin bilgi seviyesi, yetenek, yetki ve bunların nasıl paylaşılacağı; etmenler arası etkileşim seviyesinde, iletişimin nasıl sağlanacağı ve etmenlerin birbirlerine olan etkileri; organizasyon seviyesinde, paylaşılan amaç, planlar ve organizasyonun yapısı konularında çalışma yapılmalıdır.

2.3.1 İletişim

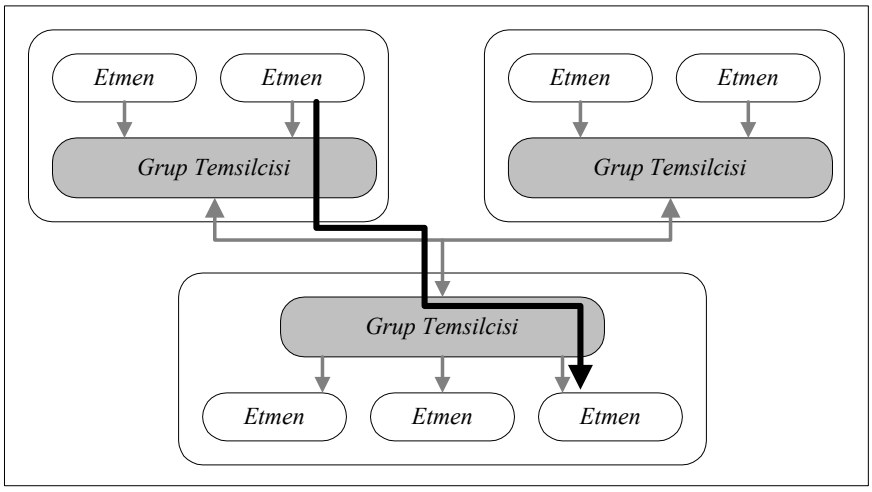
Etmenler topluca daha iyi işler yapabilmek için, diğerlerinin davranışları ile koordinasyon sağlamak için, sistem tutarlılığını (dağıtılmış sistemlerde) sağlamak için ve bilgi aktarımı yapmak için birbirleri ile iletişim kurarlar.

- Doğrudan İletişim: Etmenler TCP/IP türünden bir protokol ile doğrudan iletişim kurarlar. Bu yöntemde, etmenler haberleşecekleri diğer etmenlerin adreslerini bilmek durumundadır. Merkezi bir Adlandırma Hizmetlisi, sistemde bulunan tüm etmenlerin adres bilgilerini tutar (Şekil 2.3).



Şekil 2.3 Doğrudan İletişim

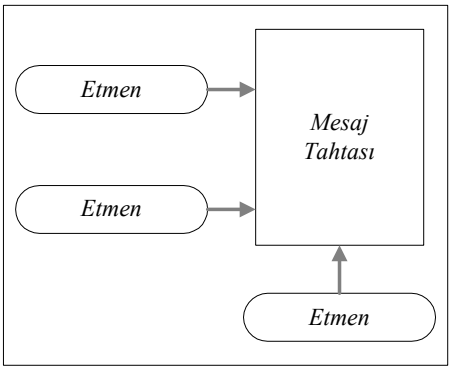
- Birleşik (Federated) Sistemlerde İletişim: Sistemdeki etmen sayısı fazla olduğu zaman doğrudan iletişim yükünü azaltmak için kullanılan etkin bir yöntemdir. Etmenler doğrudan haberleşmek yerine grup temsilcileri (facilitator) ile haberleşirler. Grup temsilcileri, grubunda bulunan etmenlerin ihtiyaçları ve yetenekleri hakkında bilgiye sahiptir (Şekil 2.4).



Şekil 2.4 Birleşik Sistemlerde İletişim

- Mesaj Tahtası (Blackboard) yoluyla İletişim: Etmenler çözüm için paylaşılan ortak bir mesaj tahtası kullanırlar. Bir etmen bir çözüm bulduğunda sonucunu mesaj tahtasına yazar. Böylelikle sistem tutarlılığını sağlamak için ek bir mekanizma kurmaya gerek kalmaz (Şekil 2.5).

- Yayın iletişimi: Mesajlar sistemdeki tüm birimlere yollanır. Etmenlerin sisteme bağlanmaları ve sistemden ayrılmaları açısından esnek bir yapı sunar.



Şekil 2.5 Mesaj Tahtası Yoluyla İletişim

2.3.2 Etkileşim

Etmenler, görevlerini yerine getirebilmek için diğer etmenlerle haberleşirler. Bazı durumlarda etmenlerin kendi işlerini gerçekleştirebilmeleri için diğer etmenlerin bazı işlemlerini tamamlamaları gerekmektedir. Bir etmen, diğer bir etmenden bir işi gerçeklemesini isteyebilir. Bu isteğini diğer etmene bildirebilmek için etkileşim kurması gerekmektedir. Etkileşim katmanı, iletişim katmanının üzerine kurulan bir katman olup, etmenler arası gidip gelen mesajların içeriklerini belirler. Bu katman, çoklu etmen mimarisinden bağımsız bir dil yaratılabilmesi için imkan tanır. Etkileşimde sentaks, iletişim için gerekli olan sembollerin yapısını belirler. Semantik, sembollerin ne belirttiği ile ilgilenir. Yorum, sembollerin yorumu ve anlamı üzerinde durur. Etmenlerin, topluca bir işi gerçeklemek üzere karşılıklı olarak birbirlerini anlamaları gerekmektedir.

2.3.2.1 Speech-Act teorisi

1969'da Searle' ün ortaya koyduğu Speech-Act teorisi insan iletişimini örnek alır. Bu teoriye göre ifade, ifade ile kastedilen anlam, ifadenin belirtilmesi sonucu gerçeklenen olay ve davranış, etmenler arasında hiçbir şüpheye düşülmeyecek şekilde doğru bir şekilde belirlenmelidir. Bu teori, etmenin inançları, istekleri ve niyetlerini belirtmesi için mesaj formatları sunar. Bu mesaj formatları performative olarak adlandırılır (Weiss, 2000).

Etmenler arası iletişim için çeşitli standartlar (KQML, Agent ACL) oluşturulmuştur. KQML, etmenler arasında yüksek seviyede çalışma ve ilişkisel işlemler imkanı tanır. Burada bahsi geçen etmenler, basit programlardan veri tabanlarına, daha karmaşık bilgi tabanlı sistemlere kadar farklı yapıda olabilir.

2.3.2.2 KQML

Süreçler arası iletişimi destekleyen uygulamalar (ftp, www vb.) ilkel iletişim protokollerine dayanırlar. Bunun sebebi süreçler arasında karmaşık iletişimi destekleyen uygun standartların bulunmamasıdır. KQML (Knowledge Query and Manipulation Language), bilgi tabanlı sistemler veya akıllı etmenler için ağ programlamasını destekleyen standart bir dil ve protokoldür. ARPA destekli olarak geliştirilen Knowledge-Sharing Effort (KSE) çalışmasının bir ürünüdür (Finin ve Fritzon, 1992).

Etmenler arası etkileşimde iletişim protokolünün semantiğini (uygulamadan bağımsız olmalıdır) mesaj semantiğinden (uygulamaya bağımlı olabilir) ayırmak gerekir. İletişim protokolü, tüm etmenler tarafından paylaşılmalıdır ve belirli sayıda iletişim ilkeline sahip olmalıdır. İki etmenin ortak temsili dilleri varsa haberleşmeleri veya tercüme ettirilebilir dilleri kullanmaları gereklidir. Ortak veya tercüme ettirilebilir dilleri olsalar dahi etmenlerin birbirlerine yolladıkları mesajı anlayabilmeleri için bir bilgi paketi tanımlanmış olmalıdır. Bu, gerçek anlamda paylaşılan bir semantik değil, paylaşılan bir ontolojidir. Ontoloji, üzerinde anlaşılması gereken ortak kavram tanımlarını belirler. Ontoloji, bir bilgi alanında nesneler, kavramlar ve ilişkilerin belirlenmesidir. Ontoloji, ilişkileri açıklamalıdır. Sınıflar ve ilişkiler ontoloji içinde temsil edilmelidir. Sınıfların üyeleri temsil edilmek zorunda değildir.

Mesajı anlamak için gerekli olan tüm bilgi iletişimin kendisinde saklıdır. Temel mesaj tanımı aşağıdaki yapı ile verilir.

(KQML-performative
:sender < word >
:receiver < word >
:language < word >
:ontology < word >
:content < word >
...)

Sentaks, LISP yapısına benzer bir yapıya sahiptir. KQML, hem uygulama programları hem de etmenler arası haberleşme için geliştirilen bir protokoldür.

KQML katmanı, mesajı etmenin anlayacağı şekilde paketler. Alıcı, mesajı anlayabilmek için dili anlamalı ve ontolojiye erişebilmelidir. KQML mesaj aktarım parametre alanları tanımları şu şekildedir:

- content →mesajın içeriğini oluşturur.
- language→mesajın dilini belirtir (KIF, LISP, PROLOG, üzerinde anlaşılan bir dil).
- ontology →mesajın semantiklerini oluşturur.
- sender→mesajı yollayan etmen adını belirtir.
- receiver→mesajın yollandığı etmen adını belirtir.
- reply-with→karşılık mesaj türünü belirtir.
- in-reply-to → karşılık mesajı ise önceki mesajın referans bilgisini belirtir.

KQML, farklı sistemler arasında bilgiyi dağıtmak için geliştirilen bir metodolojidir. Bu anlamda yapılması gereken ilk iş KIF-Knowledge Intercange Format (bilgiyi temsil etmek için biçimsel bir sentaks) tanımlamaktır. Bu anlamda yapılması gereken ikinci iş genel kavramlar, sıfatlar ve ortam bilgisinin farklı alt kümeleri için ilişkileri tanımlayan ontolojileri belirlemektir. Ontoloji terimlerinin tanımları KIF ile temsil edilen ifadelere anlam verir.

KQML mesajı içindeki dil KIF olmak zorunda değildir. PROLOG, LISP, SQL veya diğer etmen iletişim dilleri olabilir.

KQML ile konuşan etmenler birbirlerine istekçi-hizmetli olarak gözükür. İletişimler asenkron veya senkron olabilir. KQML mesajlarının içerikleri başka KQML mesajları olabilir. Örneğin etmen-1 etmen-2 ile doğrudan iletişim kuramıyorsa etmen-3' ten mesajını etmen-2' ye iletmesini isteyebilir.

KQML mesajları yedi sınıfta toplanabilir.

- Temel sorgulama mesajları (evaluate, ask-one, ask-all,..)
- Çoklu-yanıtlı sorgulama mesajları (stream-in, stream-all,..)
- Yanıt mesajları (reply,sorry,..)
- Genel bilgiye yönelik mesajlar (tell, achive, cancel, untell, unachive,...)
- Yetenek tanımlama mesajları (advertise, subscribe, monitor,...)
- Ağ mesajları (register, unregister, forward, broadcast,...)

Gönderici ve alıcı etmenler kullandıkları etmen iletişim dilini anlamalıdır. Ontolojiler yaratılmış olmalıdır ve iletişim halinde bulunan etmenler tarafından erişilebilir olmalıdır. KQML standardı üzerinde halen çalışılmaktadır.

KIF (Knowledge Interchange Format): Etmenler, gerçek dünyanın açıklamasına ihtiyaç duyarlar. Açıklamalar doğal dillerle ifade edilebilir (Türkçe, İngilizce gibi). Fakat doğal dil ifadesi çoğu zaman farklı yorumlara açıktır. Sembolik mantık, bazı şeyleri açıklamak için genel bir matematiksel araç sunar. Basit mantık, insanlar ve diğer etmenler için ilgi alanı veya yararlı olan herhangi bir şey olabilir. Bu şeyler somut olgular, çıkarım kuralları, kısıtlamalar veya meta-bilgi olabilir. KIF, hem bilgisayar sistemleri hem de insanlar tarafından anlaşılabilir.

Etmenler için tanımlanan KQML protokolü dışında diğer iletişim protokolleri de kullanılabilir. Örneğin bir etmen diğer bir etmeni kamera yoluyla gözlüyor olabilir ve diğer etmenden gelen sonuç görüntülerini kendi davranışlarını şekillendirmek için kullanıyor olabilir.

İletişim ve etkileşim protokolleri konusunda tanımlama yapıldıktan sonra daha üst seviyedeki katmanlar gerçeklenebilir.

2.3.3 Koordinasyon ve Tutarlılık

Etmenlerin davranışları arasındaki bağımlılıklardan ve etmenlerin belirli problemleri çözmek için yeterli bilgiye, yeteneğe veya kaynağa sahip olmamalarından dolayı koordinasyon sağlanmalıdır. Koordinasyon görevlerin yerine getirilmesi sürecini hızlandırır. Bu sayede bir etmenin bulduğu bir sonucu diğer etmen kullanabilir. Sistemde oluşturulan koordinasyon, sistem tutarlılığının sürmesini sağlar. Tüm etmenler en güncel ve doğru bilgiye koordinasyon ve etkileşim ile sahip olur.

2.4 Etmen Oluşturma Araçları

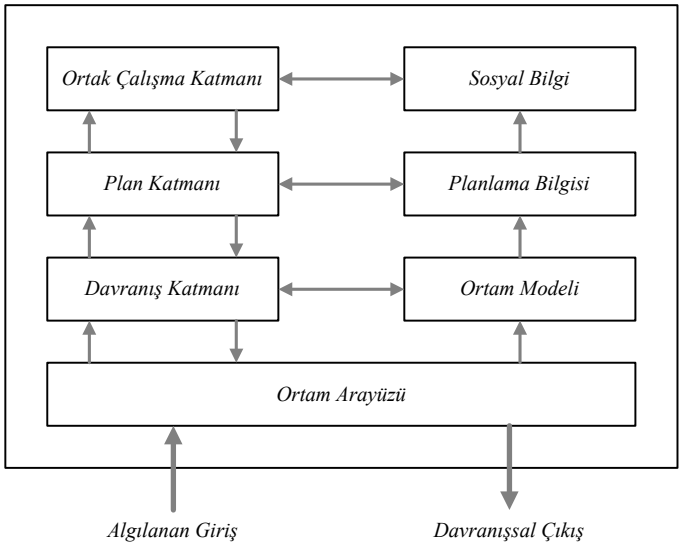
IBM Aglets Workbench: Agent ve Applets isimlerinden türetilen IBM Aggletleri ortak veri ve diğer bilgileri yönetmek, bunlara erişmek ve aramak için hareketli etmenleri kullanan ağ-tabanlı uygulamalar geliştirmek için görsel bir ortam sağlar. IBM Aggletleri hareketli Java programlarıdır ve farklı ortamlara taşınabilir. Ağ içinde özelleştirilmiş düğümlerde yürütülebilir (Aglets).

Agglets Workbench güvenli hareketli etmen-tabanlı uygulamalar oluşturabilmek için güzel bir araç sunar. Koordinasyon, ortak çalışma ve tutarlılığı gerçeklemek üzere

uygulama geliştirmek zordur. IBM Agglet' leri Sadece TCP/IP protokolünü kullanarak doğrudan iletişim kurarlar. Speech-act teorisine uygun mesajlaşma gerçekleşmez.

Voyager: Object-Space Inc. tarafından geliştirilmiştir. Recursion Software Inc. tarafından sunulmaktadır (Voyager). Hem etmen-tabanlı hem de dağıtılmış nesne sistemleri için geliştirme ortamı sunar. Nesne hareketliliği için destek sağlar. Güvenlik için özel bir önlem alınmamıştır. Etmenlerin sosyal davranışları, yayın iletişimi ve speech-act türünden mesajlaşmayı desteklemez.

Interrap: Boyuna katmanlı iki-geçişli bir etmen mimarisidir (Şekil 2.6). TCP/IP protokolünü kullanır. Sadece Unix işletim sistemi üzerinde çalışıp Oz dilinde yazılmış olmasına rağmen iyi bir etmen mimarisi sunar. Etmen mimarisindeki her bir katmanın belli bir görevi (sosyal özellik) vardır. Etmen davranışı, ortam bilgisinin tüm katmanlardan geçerek değerlendirilmesi sonucu oluşur (Weiss, 2000).



Şekil 2.6 InteRRap Mimarisi

JAFMAS: Lisp dilinde yazılan COOL mimarisi üzerine kurulmuştur (Deepika,1997). Java programlama dilinde gerçeklenmiştir. İletişim, doğrudan, birleşik ya da yayın şeklinde gerçeklenebilir. Merkezi olmayan bir anaçatı sunar. Hataya dayanıklıdır. Sistemde tutarlılık ve koordinasyon sağlanmaktadır. Tutarlılık için Petri-Net' ler kullanılır.

JATLite: Stanford Üniversitesi Center Design Research tarafından geliştirilmiştir (JATLITE). Etmenlerin sistemdeki varlıklarından haberdar olan Adlandırma Hizmetlisi (mesaj yönlendiricisi) türünden bir etmen, sistemde mesajların güvenli bir

şekilde iletilmesini sağlar. Sistemde Speech-act türü mesajlaşma KQML katmanı ile sağlanabilmektedir.

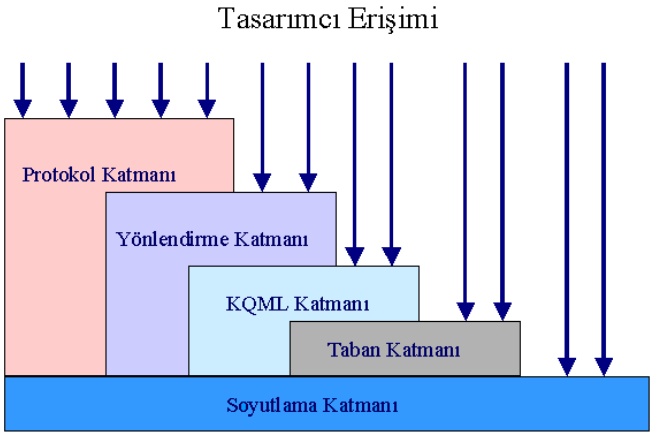
2.5 Bir Etmen Oluşturma Aracı Olarak JATLite

JATLite (Java Agent Template, Lite), Stanford Üniversitesi Center Design Research grubu tarafından geliştirilen ve etmen oluşturmak ve etkin şekilde haberleşmelerini sağlamak üzere Java paketleri sunan bir araçtır. TCP/IP üzerine kurulu temel iletişim araçları sunar. İletişim altyapısı ile etmenlerin birbirleri arasında KQML mesajları ile haberleşmesini sağlar.

JATLite, sistemdeki etmenlerin kullanıcı adı ve şifre ile kayıt oldukları bir Etmen Mesaj Yönlendiricisi (AMR-Agent Mesage Router) sunar. Bu hizmetli hem adlandırma hizmetlisi hem de mesajlaşma hizmetlisi olarak görev yapmaktadır. AMR, tüm etmen mesajlarını etmenlerin en son tanımlanan IP adreslerine yollar. Böylelikle etmenlerin ortam içinde hareket etmesine olanak tanıyan bir yapı sunulmaktadır. AMR tüm mesajları karşı taraftan mesajı aldığına ilişkin bir işaret gelene (delete mesajı ile) dek saklar. Dolayısıyla etmenler birbirlerine çevrim-dışı olarak da mesaj gönderebilir.

2.5.1 JATLite Mimarisi

JATLite mimarisi tasarımcıların kendi sistemlerini tasarlaması aşamasında uygulamaya uygun katmanları seçebileceği bir hiyerarşi sağlar (Şekil 2.7). Örneğin KQML Katmanını kullanmayıp sadece TCP/IP iletişimi yapmak isteyen tasarımcının sadece Soyutlama ve Taban Katmanlarını seçmesine olanak tanır.



Şekil 2.7 JATLite Katmanları

Soyutlama Katmanı: JATLite için gerekli temel sınıfları sunar. JATLite tüm iletişimi TCP/IP üzerinden tanımlamış olmasına rağmen bu katman genişletilerek UDP protokolü gibi farklı protokoller de kullanılabilir.

Taban Katmanı: TCP/IP' yi ve Soyutlama Katmanına dayanan temel iletişimi sağlar. Mesaj dili ve protokol konusunda bir kısıtlama yoktur. Bu katman, etmenler çoklu mesaj portlarından okuyacak şekilde ya da soketlerden giriş alacak ve dosyalara çıkış oluşturacak şekilde genişletilebilir.

KQML Katmanı: KQML mesajlarının saklanması ve çözümlenmesini sağlar.

Yönlendirme Katmanı: İsim kaydı ve mesaj yönlendirmesi ve kuyruğa atılması gibi hizmetleri sağlar. Tüm etmenler, Yönlendirici yardımıyla mesajlarını yollar ve alır.

Protokol Katmanı: Uygulamalar ve apletler için SMTP, FTP, POP3 ve HTTP gibi hizmetleri destekler.

2.5.2 AMR Etmen Mesaj Yönlendiricisi

AMR Etmen Mesaj Yönlendiricisi, kayıtlı bir etmenden mesaj alıp bu mesajı gerçek alıcısına ulaştırır. Alınan mesajlar dosya sisteminde kuyruğa atılır. Hem uygulamalar arası hem de uygulama-apletler arası mesajlaşmaya imkan tanır. Çevrim-dışı destek üretmek üzere asenkron iletişim söz konusudur. Etmenler diğerlerinin adreslerini takip etmek zorunda değildir. Hareketli etmenlerin ortamda rahatça dolaşmasına imkan tanınır.

2.5.2.1 Yönlendirici özellikleri

- Toplama ve Kuyrukta Tutma: Tüm mesajlar yönlendirici dosya sisteminde tutulur.
- Mesaj Yönlendirme: Etmen çevrim-içi çalıştığı sürece mesajları iletilir.
- Etmen Adlandırma Hizmeti (ANS-Etmen Adlandırma Hizmetlisi): Etmelerin IP adreslerinin takip edilmesini gerektirmeyen bir hizmet sunar. Sadece etmenin adı bilinerek mesaj yollanabilir. Etmen bir etmenin adresini yönlendiriciden isteyebilir ve bu etmenle doğrudan iletişim kurabilir.
- Kayıt: Yönlendiriciyi kullanmak isteyen etmen, yönlendiriciye kayıt olmak zorundadır.

- Güvenlik: Etmenin adı ve şifresi tutularak güvenliği sağlanır.

- Etmen Listeleme: Bir etmen sistemdeki tüm kayıtlı etmenler ve çevrim-içi veya çevrim-dışı olma durumlarını öğrenmek istiyorsa yönlendiriciye 'list-agents' mesajı yollar. Karşılığında 'registered-agent' mesajını alır.

- Bağlantı Kesme: Bir etmen yeniden bağlanana dek mesaj almak istemiyorsa yönlendiriciye 'disconnect-agent' mesajı yollar. Eğer bir etmen yönlendiriciye bu mesajı yollamazsa yönlendirici herhangi bir anda mesaj geldiğinde bu etmene mesajı yollamak ister. Eğer etmen istemsizce 'disconnect-agent' mesajını yollayamadan bağlantısını kestiyse yönlendirici, etmenin bağımsız çalışan bir uygulama olup olmadığını sınar. Eğer öyleyse yönlendirici etmene bağlanamaz ve belli zaman aralıklarında bağlantıyı kontrol eder. Eğer bağlantı denemeleri belli sayıda deneme için başarısızlıkla sonuçlanırsa etmenin otomatik olarak kaydı silinir. Kontrol etme süresi ve maksimum deneme sayısı değiştirilebilir parametrelerdir.

- Mesaj rezervasyonu: Bazı mesajlar belli zamanlarda ve belli konumlarda alınmak üzere saklanabilir. Alıcı, etmenin kendisi veya başka bir etmen olabilir.

- Ölçekleme: Sistemde tek bir adlandırma hizmetlisinin olmasının ölçekleme başarımı açısından etkili olması göz önüne alındığında çoklu yönlendiricili bir sistem gerçekleyerek bu problem ortadan kaldırılabilir.

Varsayımlar:

- TCP/IP tabanlı iletişim söz konusudur.

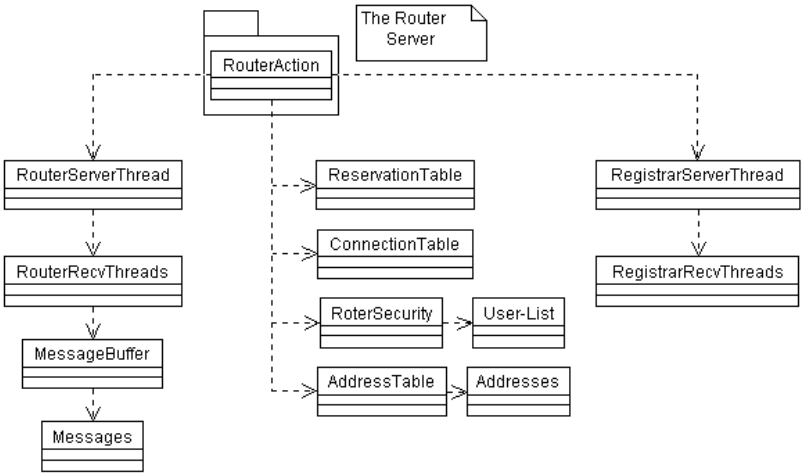
- Her bir kayıtlı etmen için sadece tek bir bağlantı vardır (Bir etmen yönlendiricide birden fazla soket açamaz.)

- Mesajlar çözümlenebilir KQML mesajlarıdır ve alıcı kesinlikle belirtilmelidir.

- Tüm mesajlar yönlendirici dosya sisteminde saklanır. Dosya sisteminden 'delete-message' mesajı ile silinmelidir.

2.5.2.2 Yönlendirici Bileşenleri

Yönlendirici, kendisinden beklenen hizmeti gerçekleyebilmek için çeşitli sınıfların işbirliğine ihtiyaç duyar. Yönlendirici Bileşenleri Şekil 2.8'de görülmektedir.



Şekil 2.8 Yönlendirici (RouterServer) Bileşenleri Diyagramı

“RouterAction” sınıfı tüm veri üyelerini başlangıç koşullarına getirir. Saklanan mesajları yönetmek üzere çalışmaya hazır hale getirir.

“RegistrarAction” sınıfı, bir görevcik olarak çalışır ve etmen kayıtlarını alır. “RegistrarServerThread” sınıfı, yönlendiriciye kaydolmak isteyen etmenin gelen bağlantı isteğini kabul eder ve “RegistrarRecvThread” sınıfını uyandırır. “RegistrarRecvThread” sınıfı kayıt için gerekli olan bilgiyi alır. Daha sonra kayıtlı etmen listesini günceller. Kayıt süreci boyunca “RegistrarRecvThread” güvenli bir bağlantı (“RegistrarSecurity”) sağlar.

Adres Tablosu (AddressTable), tüm kayıtlı etmenlerin adreslerini tutar.

Bağlantı Tablosu (ConnectionTable), o anda çevrim-içi olan alıcı görevciklerin listesini tutar. Alıcı görevcik etmen adı ile aynı olan tekil bir ada sahiptir.

Saklama Tablosu (ReservationTable), saklanan mesaj bilgilerini tutar.

“RouterSecurity”, bağlantılarda güvenliği sağlamak için kullanılır. Bir etmen yönlendiriciye bağlamak istediğinde “RouterSecurity” sınıfı, etmenin bağlantı protokolüne uyup uymadığını kontrol eder. Şifre, kontrol edilir.

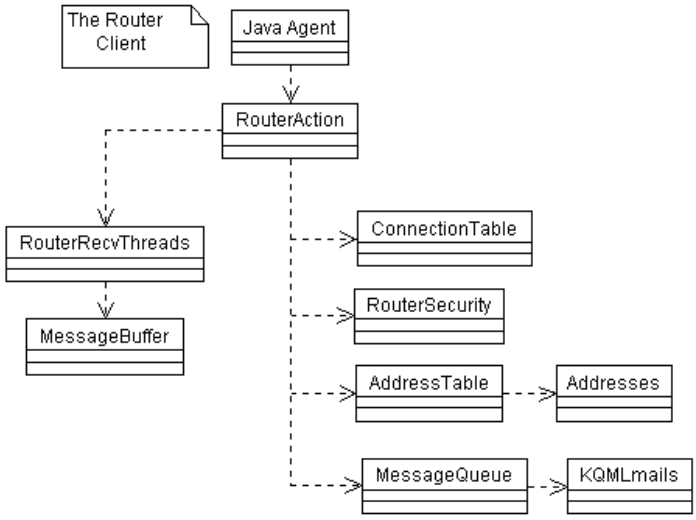
“RouterServerThread” sınıfı kayıtlı etmenlerden gelen bağlantı isteklerini kabul eder ve “RouterRecvThread” sınıfını yaratır.

Çoklu “RouterRecvThread” sınıfları diğer etmenlerle haberleşmek isteyen tüm etmenlere atanır. “RouterRecvThread” sınıfı bağlantıdan gelen mesajları alır ve

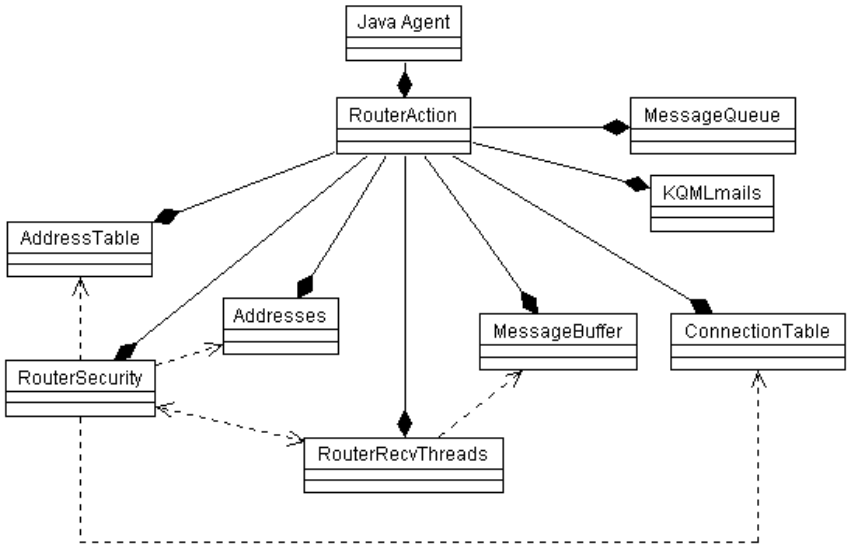
yönlendiriciye mesajları yöneltir. “RouterRecvThread” sınıfı gelen mesajları almak için kendi tampon mesaj kuyruğuna sahiptir.

2.5.2.3 Yönlendirici İstekçisi

Yönlendirici İstekçisi, çoklu etmen sistemi içinde mesajları karşılayarak bunlara hizmet etmek üzere tanımlanan sınıftır. Sistemde tasarlanan etmenlerin bu sınıftan türetilmiş olması gerekmektedir. Yönlendirici İstekçisi bileşenleri Şekil 2.9’da verilmiştir. Yönlendirici nesne ilişkileri de Şekil 2.10’da verilmiştir.



Şekil 2.9 RouterClient Sınıfı Diyagramı



2.10 RouterClient Sınıfı Nesne İlişkileri

“AgentAction” sınıfı gelen mesajlar için Act() metodunu gerçekler. “AgentAction” tüm veri üyelerini başlangıç koşullarına getirir.

Adres Tablosu (AddressTable), diğer etmenlerin adreslerini tutar. Bu tabloda en azından iki adresin (yönlendirici istekçisi ve yönlendiricinin adresleri) tutulması gerekir.

Bağlantı Tablosu (ConnectionTable), bağlantı bilgisini tutar. Yönlendirici istekçisi bağlantı kanalını sadece yönlendirici için kullanmaz. Diğer etmenlerle bağlantı kurmak için de kullanılabilir.

“BrecvThread” sınıfı yönlendiriciye bağlıdır ve yönlendirici ile mesaj alış-verişini gerçekler.

“RouterSecurity” sınıfı bağlama ve bağlantıyı kesme protokol mesajlarının alınıp yollanmasını sağlar.

“MessageBuffer” nesnesi yönlendiriciden alınan mesajları saklamak için kullanılır.

Mesaj kuyruğu vektörü, “MessageBuffer” nesnesinden alınan mesajların saklandığı yerdir.

2.5.2.4 Yönlendirici Durumları

Yönlendirici Hizmetlisi için çekirdek sınıf, “RouterRecvThread” sınıfıdır. Etmenin türüne göre etmenlerden gelen mesajlar farklı durumlar yaratabilir.

- Durum 1. (Alıcı: Yönlendirici) Bu durumda mesaj doğrudan işlenir. Mesaj, ‘delete-message’, ‘request-address’, ‘list-users’, ‘reserve-message’, ‘disconnect’ mesajlarından biri olmalıdır.

- Durum 2. (Alıcı: Çevrim-içi çalışan kayıtlı bir uygulama etmeni) Bu durumda alınan mesaj alıcının “RouterRecvThread” sınıfı mesaj kuyruğuna eklenir.

- Durum 3. (Alıcı: Çevrim-dışı çalışan kayıtlı bir uygulama etmeni ve etmene bağlantı mümkün) Bu durumda alıcı için etmene bağlanarak “RouterRecvThread” sınıfı yaratılır. Alınan mesaj alıcının “RouterRecvThread” sınıfı mesaj kuyruğuna eklenir.

- Durum 4. (Alıcı: Çevrim-dışı çalışan kayıtlı bir uygulama etmeni ve etmene bağlantı mümkün değil) Bu durumda mesaj alıcının gelen dosyasına kaydedilir.

- Durum 5. (Alıcı: Çevrim-içi çalışan kayıtlı bir aplet etmeni) Bu durumda alınan mesaj alıcının “RouterRecvThread” sınıfı mesaj kuyruğuna eklenir.

- Durum 6. (Alıcı: Çevrim-dışı çalışan kayıtlı bir aplet etmeni) Bu durumda mesaj alıcının gelen dosyasına kaydedilir.

- Durum 7. (Alıcı: Kayıtlı olmayan bir etmen) Bu durumda mesajı yollayan etmene hata mesajı yollanır.

2.5.2.5 Mesaj tipleri

Yönlendiriciye istekçilerden gelen mesaj tipleri şu şekilde sınıflandırılabilir:

Bağlantı Süreci: CONNECTION CERTIFICATION, CONNECT, RECONNECT-AGENT, DISCONNECT

Kayıt: REGISTER, IDENTIFY, WHOIAM, UNREGISTER

Etmenleri Listeleme: LIST-AGENTS, REGISTERED AGENT

Adres Sorgulama: REQUEST-ADDRESS, AGENT-ADDRESS

Mesaj İşlemleri: RESERVE-MESSAGE, DELETE-MESSAGE

2.5.2.6 Yüksek Seviyeli Yönlendirici İstekçi Sınıfı Metotları

“createServerThread (String id, int priority)” : Etmen bir uygulama yazılımı şeklinde oluşturulduysa hizmetliyi dinleyen bir birim olarak oluşturulmalıdır.

“setRouterAddress (Address)”: “Address” nesnesi kullanılarak Yönlendirici adresi belirlenir.

“setRegistrarAddress (Address)”: “Address” nesnesi kullanılarak “Registrar” adresi belirlenir.

“setMyAdress (Address)”: “Address” nesnesi kullanılarak etmen adresi belirlenir.

“register (Address registrarAddress, Address myAddress): (Throws Connection Exception)” Yönlendiriciye kayıt için kullanılır. “registrarAddress” ve “myAddress” parametre olarak aktarılır.

“register()”: Yönlendiriciye kayıt için kullanılır. “RegistrarAddress“ ve “myAddress” daha önceden belirlenmiş olmalıdır.

“connect()”: Yönlendiriciye bağlanmak için kullanılır. “AgentAction” için Yönlendirici Adresi ve “myAddress” daha önceden belirlenmiş olmalıdır.

“connect(Address myAddress)”: Yönlendiriciye bağlanmak için kullanılır. “AgentAction” için Yönlendirici Adresi daha önceden belirlenmiş olmalıdır.

“connect (String myName)”: Yönlendiriciye bağlanmak için kullanılır. “AgentAction” için Yönlendirici Adresi ve “myAddress” daha önceden belirlenmiş olmalıdır.

“sendMessage (String receiver, String msg):”: Yönlendirici de dahil olmak üzere diğer etmenlere mesaj yollamak için kullanılır.

“sendMessage (String msg):”: Yönlendiriciye mesaj yollamak için kullanılır.

“sendKQMLMessage(String msg):”: KQML çözümleme denetimi ile Yönlendiriciye mesaj yollamak için kullanılır.

“addToDeleteBuffer(int)”: Okunan mesajı silmek için kullanılır. Sonunda yönlendiriciye ‘delete-message’ mesajı yollanır.

“disconnect()”: Yönlendiriciden bağlantıyı kesmek için kullanılır.

“unregister()”: Yönlendiriciden etmen kaydını silmek için kullanılır.

“EndAction()”: Temizleme ve AgentAction’ ı sonlandırmak için kullanılır.

3. ETMENLERDE ÖĞRENME

3.1 Giriş

Bu bölümde, etmenlere akıllı sıfatını yükleyen öğrenebilme yetisinin bilgi işlemsel açıdan nasıl ve hangi şekillerde gerçeklenebileceği açıklanmıştır. Etmenler, ortamları ile etkileşim kurarak içinde bulundukları öğrenme sürecinde çeşitli yöntemler ile yardım alabilirler. Bu yardım, doğrudan gerekli olan bilginin verilmesi, örnek gösterilmesi veya sadece onay verilmesi şeklinde olabilir. Bu farklılıklar öğrenme sistemlerinde farklı yöntemler oluşturmuştur.

3.2 Etmenlerde Öğrenme

Öğrenme, yeni bilgilerin, düşünsel veya bilişsel yeteneklerin kazanılması ve kazanılan bilgi ve yeteneklerin gelecek sistem etkinliklerinde kullanılması süreçlerinden oluşur. Çoklu etmen sistemlerinde öğrenme süreci tek bir etmen tarafından yürütüldüğünde (diğer etmenlerle iletişim olmadığı durumlar) merkezi olarak gerçeklenmektedir. Merkezi olmayan öğrenme sürecinde etkinlikler farklı etmenler tarafından yürütülmektedir. Çoklu etmen sistemlerinde bir çok merkezi öğrenme sürecindeki birim aynı veya farklı öğrenme amaçlarını gerçeklemek üzere aynı anda aktif olabilir. Aynı işlem, farklı merkezi olmayan süreçlerde bulunan gruplar için de geçerlidir. Bir etmen bir çok merkezi veya dağıtılmış öğrenme sürecinde aynı anda bulunuyor olabilir (Weiss, 2000).

Etmenlerin öğrenebilecekleri konular,

- Parametrelere bağlı kısıtlamalar veya genel olarak tasarım elemanları
- Tasarım parametreleri arasındaki bağımlılıklar
- Tasarım kuralları
- Kurallar, davranışlar ve görevler için ön koşullar
- Diğer etmenlerin kararları için öngörü şeklinde sıralanabilir (Grecu ve Brown, 1996).

3.3 Öğrenme Yöntemleri

Öğrenme, dağıtılmış, ardışıl veya paralel olarak gerçeklenebilir. Dağıtılmış öğrenme sürecinde etkileşim seviyesi, etkileşim süreci, etkileşim sıklığı, etkileşim örüntüsü ve etkileşimin değişkenliği önemli konulardır. Öğrenme sürecinde etmen, gelecek durumlar için geçmiş durumlarını değerlendirir.

- Açıklamaya dayalı öğrenme: Planları ve stratejileri genelleştirerek etmenler arasında erişilebilir kılmak için uygulanır.
- Çıkarım ile öğrenme: Tasarım durumlarını sınıflandırmak, diğer etmenlerin davranışsal modellerini yapılandırmak gibi amaçlar için uygulanır.
- Doğrudan Öğrenme: Genelleştirme yapmadan öğrenilir. Saklanan bilgi yeni durumlar için tekrar kullanılmaz.
- Yeni durumlar karşısında öğrenme: Yeni bir örnekle karşılaşıldığında öğrenme durumu gerçekleşir (tembel öğrenmesi). Tasarım aşamasında öngörülmeyen ve genelleştirilmeye açık olamayan durumlar için uygulanır.
- Benzerlik ve kıyaslama ile öğrenme: Çözümü bilinmeyen bir problemi çözümü bilinen bir probleme benzeterek öğrenme yöntemidir.
- Örneklerden Öğrenme: Olumlu ve olumsuz örneklerden genel bilgiye ulaşılır.
- Sınıfları Oluşturabilmek için Öğrenme (Unsupervised Learning): Bölütleme yöntemleri de bu türden bir öğrenme yöntemidir.
- Pratik yaparak, deneyimle öğrenme: Destekli öğrenme, genetik algoritmalar bu türden öğrenme yöntemleridir. Bilinmeyen bir ortamda, dolaylı bilgiler alınarak öğrenilir (indirect supervision). Ortamdan öğrenme için destek verilir.

3.3.1 Öğrenme için Ortamdan Alınan İşaret

Ortamdan alınan işaret (geri besleme) öğrenmenin türünü belirler. Geri beslemeye göre üç tür öğrenme yöntemi mevcuttur:

- Yönlendiricili Öğrenme (Supervised Learning): Geri besleme, öğrenen birimden istenen davranışı belirler ve öğrenen birimin amacı, davranışını mümkün olduğunca istenen davranışa yaklaştırmaktır.
- Destekli öğrenme (Reinforcement Learning): Geri besleme, öğrenen birimin davranışının ne kadar iyi olduğunu belirtir. Öğrenen birim bu geri beslemeyi artırmaya çalışır.

- Desteksiz öğrenme (Unsupervised Learning): Öğrenen birim, geri besleme almaz. Öğrenen birim, deneme-yanılma yöntemleri ile kendi kendine organizasyon ile en uygun davranışı seçmelidir.

Bu durumların her biri için öğrenme geri beslemesi ortam tarafından veya etmen tarafından (etmen içinde ayrı bir birim tarafından) verilir. Bu durumda geri beslemeyi üreten ortam veya etmenin kendisi yönlendiricili öğrenme için "öğretici" olarak, destekli öğrenme için "eleştirmen" olarak ve desteksiz öğrenme için sadece "pasif gözlemleyici" olarak görev yapar (Weiss, 2000).

3.4 Destekli Öğrenme

Destekli Öğrenme yönteminde sistemde doğrudan bir bilgi verici birim bulunmaz ve gerekli örnekler sisteme verilmez. Etmenin öğrenebilmesi için ortamını algılaması, bir davranış sergilemesi ve bir amacı olması gerekir. Destekli Öğrenme, etkileşimli bir öğrenme yöntemidir. Etkileşimli öğrenmede, istenen davranış veya amaç için doğru olan ve etmenin gerçekleştireceği tüm davranışlardaki durumlar için örnekler yoktur. Etmen, kendi tecrübelerinden öğrenmek zorundadır. Etmen daha fazla ödül alabilmek için geçmişte denediği ve ödül üretmek için etkin olduğunu keşfettiği davranışları tercih eder. Fakat bu türden davranışları keşfedebilmesi için daha önceden seçmediği davranışları denemesi gerekir. Etmen ödül alabilmek için bildiklerini kullanabilmelidir ve gelecekte daha iyi davranışlar sergileyebilmek için araştırmalıdır. Stokastik bir görevde her bir davranış, beklenen ödülün güvenilir bir tahminini elde etmek için çok defa denenmelidir. Etmen, bu öğrenme sürecinde kesin olmayan bir ortamda bulunur (Sutton ve Barto,1999).

Destekli öğrenme yönteminde, etmenin ortam hakkındaki belirsizliklere rağmen kendi amacını gerçeklemek üzere karar-verici etmen ve ortamı arasında etkileşim oluşur. Etmenin davranışlarının ortamın bir sonraki durumunu (bir sonraki satranç pozisyonu, robotun bir sonraki konumu vb.) etkilemesine izin verilir. Böylelikle etmene daha sonraki adımlarda imkan tanınır. Doğru seçimler, davranışların dolaylı ve gecikmeli sonuçlarının dikkate alınmasını ve dolayısıyla önlem alma ve planlama gerektirir.

Tüm örneklerde davranışların etkileri tümüyle tahmin edilemez, böylece etmen ortamı sıkça görüntülemeli ve uygun bir şekilde tepki vermelidir.

Etmen, zaman içinde başarımını artırmak için tecrübesini kullanabilir. Etmenin başlangıçta sahip olduğu bilgi (benzer görevlerdeki önceki deneyimlerinden oluşturulabilir, tasarımda belirlenmiş veya evrimsel olabilir) neyin yararlı olduğu veya neyin kolay öğrenilebilir olduğunu etkiler. Davranışı düzenlemek için ortamla etkileşim gereklidir.

3.4.1 Destekli Öğrenme Sistemi Bileşenleri

Destekli öğrenme sisteminde etmen ve ortam yanında dört bileşen tanımlanmalıdır. Bunlar, yöntem (policy), ödül fonksiyonu (reward function), değer fonksiyonu (value function), ve gerekirse ortamın bir modelidir.

Yöntem (Policy): Öğrenen etmenin belli bir anda davranış biçimini tanımlar. Yöntem, ortamın algılanan durumundan etmenin davranışına karşılık bir eşleştirmedir. Bazı durumlarda yöntem, basit bir fonksiyon veya bir başvuru tablosu iken diğer durumlarda arama süreci gibi geniş anlamlı bir işlem gerektirebilir. Yöntem, tek başına davranışı belirlemesi açısından öğrenen etmenin çekirdeğini oluşturur. Genellikle yöntemler stokastiktir.

Ödül Fonksiyonu (Reward function): Destekli Öğrenme probleminde amacı belirler. Ödül fonksiyonu, tüm algılanan durumlardan (durum-davranış çifti) o durumun istenebilirlik seviyesini belirten tek bir sayıya (ödül) eşleştirme yapar. Destekli öğrenen etmenin asıl amacı uzun vadede toplam ödülünü maksimum hale getirmektir. Ödül fonksiyonu etmen için hangi davranışların iyi hangilerinin kötü olduğunu belirler. Anlıktırlar ve etmenin yüzleştiği problem özelliklerini tanımlar. Etmen, düşük ödül getiren bir davranış seçtiğinde yöntem aynı durumda başka davranışlar seçmek üzere değiştirilebilir. Genellikle ödül fonksiyonu stokastiktir.

Ödül fonksiyonu ara bir durumda neyin iyi olduğunu belirlerken, değer fonksiyonu uzun vadede neyin iyi olduğunu belirler. Bir durumun değeri bu durumdan başlandığında etmenin toplayabileceği tahmini ödül miktarını belirler. Ödüller, ortam durumlarının anlık istenebilirlik ölçüsünü belirlerken, değerler takip eden durumlarda ve bunlara ilişkin ödüller göz önüne alındığında uzun vadeli istenebilirliği belirler. Örneğin bir durum düşük ödüle sahip olurken yüksek bir değere sahip olabilir. Çünkü yüksek ödüller getiren diğer durumların öncüsü olabilir. Ya da tersi doğru olabilir. İnsan biyolojik sistemi göz önüne alındığında haz ve ağrı ödüller olurken, değer o durumda sevinçli olma veya üzüntülü olma durumuna karşılık düşürülebilir.

Ödüller, algısal olup birincil öneme, değerler ödüllerin tahmini olmak üzere ikincil öneme sahiptir. Ödüller olmazsa değerler de olmaz. Değerlerin tahmin edilmesinin tek amacı, daha fazla ödül almaktır. Davranış seçimleri ise değerlere dayanarak oluşturulur. Daha yüksek değerlere sahip durumlara (ödüllere değil) ulaşan davranışlar seçilir. Çünkü bu durumlar uzun vadede daha yüksek ödüller getirir. Değerleri belirlemek ödülleri belirlemekten daha zor bir iştir. Ödüller doğrudan ortam tarafından verilir. Fakat değerler tahmin edilir, ve etmenin tüm yaşam süresi boyunca yeniden tahmin edilir. Bu noktada tüm destekli öğrenme algoritmalarının en önemli bileşeni değerleri etkin bir şekilde tahmin etmek için yöntem geliştirmektir.

Ortamın modeli: Destekli öğrenme sisteminde model, ortamın davranışını taklit eden bir bileşendir. Örneğin belli bir durum ve davranış çifti için model bir sonraki durum ve ödülü verebilmelidir. Modeller, davranışların ve etkilerinin simülasyonunu gerçekleyerek olası durumları değerlendirebilmek üzere planlamada kullanılır.

3.4.2 Destekli Öğrenme Problemi

Öğrenen ve karar veren etmenin kendisinden başka etkileşim kurduğu her şey ortam olarak tanımlanır. Etkileşim sürekli devam eder ve etmen davranış seçer, ortam bu davranışa yanıt verir ve etmen için yeni bir durum sunar. Ortam, etmenin zaman içinde artırmaya çalıştığı ödülleri verir.

Etmen ve ortam, ayrık zaman adımları sırası boyunca $(t = 0,1,2..)$ etkileşim kurarlar. Her bir t zaman adımında etmen ortamının $s_t \in S$ durumunu algılar. S, tüm olası durumları içerir. Buna göre bir $a_t \in A$ davranışını seçer. A, s_t durumunda gerçeklenebilecek davranışların kümesidir. Bir adım sonra davranışının sonucu olarak sayısal bir $r_{t+1} \in R$ ödülünü alır ve kendisini yeni bir s_{t+1} durumunda bulur.

Her bir zaman adımında etmen, durumlardan her bir davranışı seçme olasılığına eşleştirmeyi gerçekler. Bu eşleştirme yöntem (π_t) olarak adlandırılır. $\pi_t(s,a)$, $a_t = a$ ve $s_t = s$ olma olasılığını belirtir. Destekli öğrenmede, etmenin deneyimlerine bağlı olarak yöntemini nasıl değiştireceği belirlenir. Etmenin amacı uzun vadede ödüllerini artırmaktır.

Zaman adımları gerçek zamanın sabit aralıkları olmak zorunda değildir. Davranışta ve karar vermede keyfi adımlar olabilir.

3.4.3 Örnek Bir Destekli Öğrenme Problemi

Bilgisayar oyunlarında, bilgisayarın kullanıcısına karşı akıllı birimler olarak görünebilmesi için makine öğrenmesi yöntemleri kullanılmaktadır. Bu oyunlarda makine, kullanıcı hizmetine sunulmadan önce kendi kendine belirli sayıda (bazı oyunlar için oldukça fazla) oyun oynayarak oyun içindeki tüm durumlar için oyunu kazanabilme olasılığı belirleyerek gerçek oyunda bu değerleri kullanabilmelidir. Makine için kendisiyle oynadığı oyunların her biri birer denemedir. Bu denemeler esnasında en uygun stratejileri öğrenmektedir. Makine öğrenmesi yöntemleri kullanılarak kendi kendine öğrenen bilgisayar sistemleri arasında tavla ve satranç (Deep Blue) yazılımları sayılabilir. Bu oyunlardaki durum sayısı oldukça fazladır. Bu nedenle modelleme için uygun yöntemler bulunmalıdır (Yapay sinir ağları kullanılarak genelleştirme yapılması gibi). Destekli öğrenme probleminin çözülme yöntemini anlamak üzere Tic-Tac-Toe örneği göz önüne alınabilir (Şekil 3.1).

X	O	O
O	X	
		X

Şekil 3.1 Tic-Tac-Toe Örneği

Her bir durumun (oyundan bir sahne) bir satırda yer aldığı bir tablo hazırlanır. Durumlara karşılık olarak yazılmış olan sayılar bu durumdan kazanabilme olasılığının tahmin edilen son değerini verir. Bu tahmini değer, durumun değerini belirtir. Tablonun tümü de öğrenilen değer fonksiyonudur. Bir durumdan başlayarak kazanabilme olasılığı bir diğer duruma göre daha fazla ise karşılık düşen değerler de aynı oranda fazla olacaktır.

Bir satırında X X X içeren tüm durumlar 1 olasılığına sahiptir. (X' leri koyan oyuncu oyunu kazanmıştır.) Bir satırında O O O içeren tüm durumlar 0 olasılığına sahiptir. (Karşı taraf oyunu kazanmıştır.) Tüm diğer durumlar için başlangıçta 0.5 değeri atanır (Kazanma şansı: %50).

Oyun esnasında makine, bir sonraki durumda en yüksek kazanma olasılığına sahip durumu seçer ve ona göre oynar. Tablonun düzenlenmesi aşamasında değeri yüksek olmayan rasgele davranışlar da seçilebilir (Bu durumların tecrübe edilmesi gerekmektedir.). Oyun esnasında durumların değerleri değiştirilir. Daha önceki durumun değeri sonraki durumun değerine yakın olacak şekilde düzeltilir. Bu da önceki durumun mevcut değerinin sonraki durumun değerine yakın olacak şekilde

değiştirilmesi ile gerçeklenir. Bu şekilde durum güncellemeleri sonucunda durumlardan izlenmesi gereken en uygun yöntemler ortaya çıkar.

Destekli öğrenmede, etmenin amacı uzun vadede aldığı ödülleri maksimum hale getirmektir. Sistemde ödüllerin belirlenmesi önemli bir konudur. Örneğin satranç oynayan etmen, sadece kazandığı zaman ödül almalıdır. Karşı tarafın taşını alma ya da tahta kontrolünü kazandığı zaman ödül vermek makinenin asıl görevi gerçeklemek yerine bu alt görevleri yerine getirmesine sebep olur. Kaybetme pahasına da olsa rakibin taşını almaya çalışacaktır. Ödül işareti etmenin neyi başaracağını (nasıl başaracağını değil) belirler (Sutton ve Barto,1999).

3.4.4 Destekli Öğrenme Sisteminin Yapısı

t . adımdan sonra alınan ödüller, $r_{t+1}, r_{t+2}, ..r_T$ şeklinde ise etmen, beklenen ödülü maksimum yapmaya çalışır. Zamana göre hafifletilmiş ödül fonksiyonu:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + .. = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad 0 \le \gamma \le 1, \text{ hafifletilmiş ödül fonksiyonu (3.1)}$$

Geçmiş bilgisini tutmak, sonraki tahminler açısından gereklidir. Tüm geçmiş bilgiyi tutmak yerine tüm uygun bilginin gözüktüğü bir yol seçilmelidir. Tüm uygun olan bilgiyi tutan durum işareti Markov özelliğine sahiptir. Örneğin dama oyunundaki pozisyon (tahta üzerindeki taşların mevcut konumu) Markov durumudur. Çünkü bu konuma gelene dek yapılan tüm sıralı oyunlar hakkında gerekli olan bilgiyi verir. Oyun esnasında oyunun geleceği açısından önemli olan kısım tutulur.

Destekli öğrenme problemi için Markov özelliği tanımlanmaktadır. Durum ve ödül sayılarının sonlu olduğu varsayılmaktadır. Markov özelliği sistemde mevcut durum ve davranış bilindiğinde, bir adım dinamiğinin bir sonraki durum ve beklenen bir sonraki ödülünün tahmin edilebilmesini sağlar. Destekli öğrenmede Markov özelliğinin önemli olmasının sebebi kararlar ve değerlerin sadece mevcut durumun bir fonksiyonu olarak varsayılmasından gelir. Bunların etkin olabilmesi için durumun bilgi verici olması gerekir. Markov özelliği ile ortam dinamiği şu şekilde belirtilebilir:

$$P_r = \left\{ s_{t+1} = s', r_{t+1} = r \middle| s_t, a_t \right\} \qquad (3.2)$$

Destekli öğrenme sistemi Markov özelliğini sağlıyorsa, Markov Karar Süreci (MDP, Markov Decision Process) olarak adlandırılır. Herhangi bir s,a çifti için her bir sonraki s' durumunun olasılığı:

$$P_{ss'}^{a} = P_r\left\{s_{t+1} = s' \middle| s_t = s, a_t = a\right\} \tag{3.3}$$

Herhangi bir s,a çifti ve sonraki s' durumu için sonraki ödülün beklenen değeri:

$$R_{ss'}^{a} = E\left\{r_{t+1} \middle| s_t = s, a_t = a, s_{t+1} = s'\right\} \tag{3.4}$$

Bu iki büyüklük MDP dinamiğinin en önemli yapı taşlarını belirler.

π yöntemi uygulandığında s durumunun değeri, $V^{\pi}(s)$, s durumundan başlayarak π yöntemi izlendiğinde beklenen değerdir.

$$V^{\pi}(s) = E_{\pi}\left\{R_t \middle| s_t = s\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s\right\} \tag{3.5}$$

Terminal durum için beklenen değer her zaman için sıfırdır.

Etmenin s durumunda a davranışını gerçekleyip, π yöntemini izlemesi durumunda beklenen değer:

$$Q^{\pi}(s,a) = E_{\pi}\left\{R_t \middle| s_t = s, a_t = a\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a\right\} \tag{3.6}$$

$Q^{\pi}(s,a)$, π yöntemi için davranış değer fonksiyonudur. V^{π} ve Q^{π}, deneyimlerden tahmin edilebilir. Örneğin etmen π yöntemini izlerse ve tüm durumlar için gerçek ödüllerin ortalamasına sahip olursa ortalama, durumlara rastlama sayısı sonsuza giderse $V^{\pi}(s)$'e yakınsar. Eğer bir durumdaki her bir davranış için ayrı ortalamalar tutulursa bu ortalamalar benzer şekilde $Q^{\pi}(s,a)$'ye yakınsar.

Destekli öğrenmede kullanılan değer fonksiyonlarının temel özelliği belli rekürsif bağıntıları sağlamalarıdır. Herhangi bir π yöntemi ve herhangi bir s durumu için s değeri ve olası sonraki durumların değerleri arasında aşağıdaki tutarlılık koşulunun sağlanması gereklidir:

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} P^{a}_{ss'} \left[R^{a}_{ss'} + \gamma V^{\pi}(s') \right] \quad (3.7)$$

(3.7) eşitliği V^{π} için Bellman eşitliği olarak tanımlanır ve bir durumun değeri ile sonraki durumların değeri arasıdaki ilişkiyi verir.

Bir destekli öğrenme problemini çözmek, uzun vadede çok ödül alacak yöntemi seçmek demektir.

Değer fonksiyonları, yöntemler üzerinde kısmi sıralama tanımlar. Eğer $\forall s \in S$ için $V^{\pi}(s) \geq V^{\pi'}(s)$ ise $\pi \geq \pi'$ bu kurala uyan en az bir yöntem vardır. Bu da optimal yöntem olarak adlandırılır. π^{*} şeklinde ifade edilir.

Optimal durum-değer fonksiyonu:

$$V^{*}(s) = \max_{\pi} V^{\pi}(s) \quad \forall s \in S \text{ için} \quad (3.8)$$

Optimal davranış-değer fonksiyonu:

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a) \quad \forall s \in S \text{ ve } \forall a \in A(s) \text{ için} \quad (3.9)$$

Q* için Bellman optimallik eşitliği şu şekilde yazılabilir:

$$Q^{*} = \sum_{s'} P^{a}_{ss'} \left[R^{a}_{ss'} + \gamma \max_{a'} Q^{*}(s',a') \right] \quad (3.10)$$

Sonlu Markov Karar Zincileri için Bellman optimallik eşitliğinin yöntemden bağımsız olarak tek bir çözümü vardır (Sutton ve Barto,1999).

3.4.5 Destekli Öğrenmede Çözüm Yöntemleri

- Dinamik Programlama: Matematiksel olarak iyi bir öğrenme yöntemi sunar. Ortamın tam ve doğru modelini gerektirir.
- Monte Carlo Yöntemleri (MC): Bir model gerektirmez ve kavramsal olarak basittir, fakat adım-adım artımlı işlemeye elverişli değildir.

- Geçici Fark (Temporal Difference) Yöntemleri: Modele ihtiyaç duymazlar ve tamamen artımlıdırlar. Analizi zordur.

Bu yöntemler, etkinlikleri ve yakınsama hızları açısından farklılık gösterirler.

3.4.6 Geçici Fark Yöntemleri

Geçici Fark Yönteminde (TD, Temporal Difference) öğrenme sürecinde sadece bir sonraki adımda alınan geri besleme değerlendirilir. Değer fonksiyonu güncellemesi ve optimum değer fonksiyon değerleri (3.11) ve (3.12)'de verilmiştir.

$$V(s_t) \leftarrow V(s_t) + \alpha\left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right] \tag{3.11}$$

$$V^{\pi}(s) = E_{\pi}\left\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) \middle| s_t = s\right\} \tag{3.12}$$

En basit TD yöntemi TD(0) olarak bilinir. TD(0), durum düğümü için değer tahminini kendisinden bir sonraki duruma örnek geçişe dayanarak günceller.

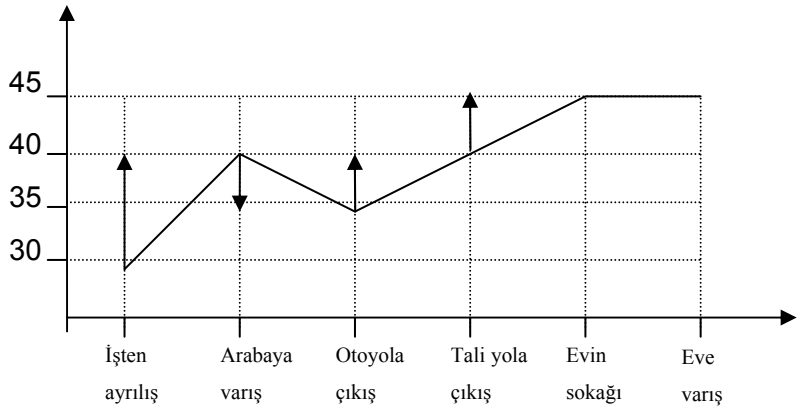
3.4.6.1 Örnek bir TD(0) güncellemesi

Problem, işyerinden eve varış için geçen süreyi tahmin etmek olarak verilmiş olsun. İş yerinden ayrıldıktan sonra zaman bilgisi kaydedilir. Saatin 18.00 olduğu göz önüne alınarak eve varış için 30 dakikalık bir varış süresi tahmini yapılır. Arabaya ulaşıldığında saatin 18.05 olduğu ve yağmur yağdığı gözlemlenir. Yağmurda trafiğin yavaş ilerlemesi göz önüne alınarak tahmin o andan itibaren 35 dakika (veya toplamda 40 dakika) olarak değiştirilir. 15 dakika sonra otoyolda her şey yolunda gitmektedir ve tali yola sapılır. Otoyoldaki hız nedeniyle tahmini süre 35 dakika olarak değiştirilir. Fakat tali yolda (tek bir aracın geçebileceği darlıkta) yavaş bir kamyonetin peşine takılma sonucu evin bulunduğu caddeye ancak 18.40' ta ulaşılır. 3 dakikada eve ulaşılır. Durumlar sırası, zamanlar ve tahminler Tablo 3.1'de verilmiştir.

Bu örnekteki ödüller, her bir adımda harcanan sürelerdir. Tüm durumlar için ödül, bu durumdan eve gitme süresidir (γ=1). Şekil 3.2'de örneğe ilişkin beklenen değerler grafiği verilmektedir (Sutton ve Barto,1999).

Tablo 3.1 TD(0) Öğrenme Örneği

Durum	Geçen Süre(dk)	Tahmini Kalan Süre	Tahmini Toplam Süre
İşten ayrılış	0	30	30
Arabaya varış	5	35	40
Otoyola çıkış	20	15	35
İkinci yol, kamyonetin ardında	30	10	40
Evin sokağına giriş	40	3	43
Eve varış	43	0	43



Şekil 3.2 TD(0) Örneği için Beklenen Değerler Grafiği

3.4.6.2 TD yöntemlerinin avantajları

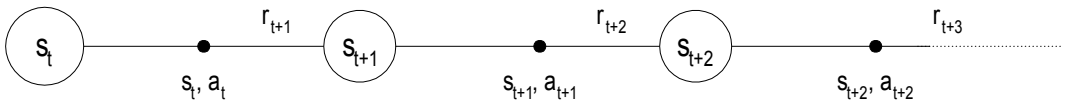
TD yönteminde tahminler, diğer tahminlere göre öğrenilir. Bu yöntem ortamın modeline gereksinim duymaz. Sonraki davranışların ne olduğundan bağımsız olarak geçişlerden öğrenir. TD yönteminin herhangi bir sabit yöntem için V_π' ye yakınsadığı ispatlanabilir. TD yöntemleri MC yöntemlerinden daha hızlı olarak sonuca yakınsar. Sonlu sayıda durumdan oluşan problemler için artımlı öğrenme yöntemleri kullanılarak, yöntem belli bir cevaba yakınsayana dek deneyim sürdürülür.

TD yöntemleri,

- Sarsa Öğrenmesi
- Q Öğrenmesi olarak iki alt gruba ayrılabilir.

3.4.7 Sarsa Öğrenmesi

Sarsa ismi $(S)\, s_t, (A)\, a_t, (R)\, r_{t+1}, (S)\, s_{t+1}, (A)\, a_{t+1}$ bileşenlerinden gelmektedir. Yönteme bağımlıdır. İlk adım durum-değer fonksiyonu yerine davranış-değer fonksiyonunu öğrenmektir. Özel olarak yönteme dayanan çözüm için tüm s durumları ve a davranışları ve o anki davranış yöntemi π için $Q^{\pi}(s,a)$ tahmin edilmelidir. Bu V^{π}'yi öğrenmek için kullanılan TD yöntemiyle yapılabilir.



Şekil 3.3 Durum Geçişleri

Buradaki geçişler durum-davranış çiftleri arasında yapılmaktadır. Her bir durum-davranış çiftinin değeri öğrenilir. Bu süreç içinde bir Markov zinciri oluşturulur (Şekil 3.3). Güncelleme (3.13) formülüne göre gerçeklenir.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right] \tag{3.13}$$

Bu güncelleme, terminal olmayan bir s_t durumundan gerçeklenen tüm geçişlerde yürütülür.

$$Q(s_{t+1}, a_{t+1}) = 0 \qquad (\ s_{t+1} \text{ terminal durum ise }) \tag{3.14}$$

3.4.8 Q Öğrenmesi

Bir adımlı Q öğrenmesi, yöntemden bağımsız yürütülen bir öğrenme türüdür (Watkins,1989). Güncelleme (3.15) formülüne göre gerçeklenir.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right] \tag{3.15}$$

Bu durumda öğrenilen davranış-değer fonksiyonu Q, (takip edilen yöntemden bağımsız olarak) doğrudan optimal davranış-değer fonksiyonuna yakınsar. Bu da algoritmanın basitleşmesini sağlar.

3.4.8.1 Q Öğrenmesi Algoritması

Q Öğrenmesi algoritması Q öğrenmesi yöntemini kullanan bir algoritmadır. Algoritma, deterministik olmayan ödüller ve davranışlar için de güncelleme işini optimal değere yakınsayacak şekilde gerçekler. Bu yönteme ilişkin algoritma Q Öğrenmesi Algoritması olarak bilinir ve Şekil 3.4'te görülmektedir.

Q Öğrenmesi Algoritması

Her bir s,a için $Q(s_t, a_t)$ *tablo kaydını sıfırla,*

O anki durumu al: s_t

Belli bir yakınsama koşulu sağlanana dek

{

Bir davranışı a_t *seç ve bu davranışı gerçekle*

O anki ödülü al

Yeni s' durumunu gözlemle

$Q(s_t, a_t)$ *tablo girişini aşağıdaki kurala göre güncelle*

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right]$$

$s_t \leftarrow s_{t+1}$

}

Şekil 3.4 Q Öğrenmesi Algoritması

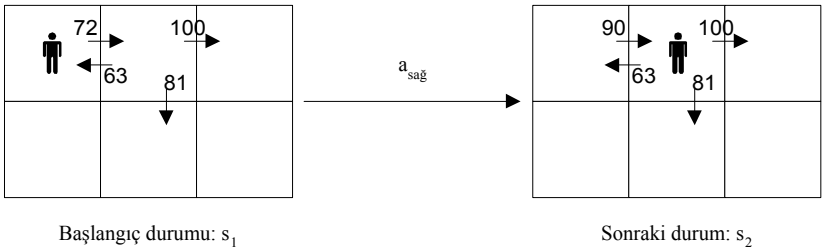
3.4.8.2 Bir Q-Öğrenmesi Örneği

Determinist ödül ve davranış olması durumunda güncelleme için geçerli formül:

$$Q(s,a) \leftarrow r + \gamma \max_{a'} Q(s',a') \tag{3.16}$$

Bu durumda Şekil 3.5'teki ortamda yaşayan etmenin amacı, varış noktasına (V) ulaşmak olsun. $Q(s,a)$ değerleri s durumunda a davranışı gerçeklendiğinde varışa ulaşma değerleri olacaktır. Etmen, öğrenme sürecinden sonra her bir durumdan varışa ulaşmak için gerekli olan en kısa yolu bulabilmektedir. Bir durumun Q değerinin güncellemesi, etmen sonraki duruma geçip ödülünü aldıktan sonra yapılır. Bu örnekte 100 puanlık ödül, sadece V durumuna geçişte alınmaktadır ve diğer durumlara geçişte ödül 0 olmaktadır (Şekil 3.5).

γ=0.9 için,



Şekil 3.5 Q Öğrenmesi Örneği (Mitchell, 1997)

$$Q(s,a) \leftarrow r + \gamma \max_{a'} Q(s',a')$$

$$\leftarrow 0 + 0.9 \max_{a'}(63,81,100)$$

$$\leftarrow 90$$

3.4.9 TD Yöntemlerinin Karşılaştırılması

Yapılan çalışmalar sonucu, çevrim-içi çalışmada Q-öğrenmesi yönteminde, Sarsa yöntemine göre sistemin daha çok ceza puanı (negatif geribesleme) aldığı gözlenmiştir. Fakat bunun yanında Q-öğrenmesinin optimal yöntem değerlerini öğrenme başarımı daha yüksektir. Sarsa yöntemi, davranış seçimini de dikkate alır (yönteme bağımlı olarak).

4. BÖLÜT ANALİZİ

4.1 Giriş

Bu bölümde, tez kapsamında etmenler tarafından yürütülen bölüt analizi süreçleri için gerekli olan temel bilgiler verilmiştir. Bölüt analizi bileşenleri, yöntemler, ve henüz çözülememiş olan bazı problemlerden bahsedilmiştir. Veriler üzerindeki bölütlerin analizinin bilgi işlemsel olarak nasıl gerçeklendiğine ilişkin algoritmalar tanıtılmıştır.

4.2 Bölüt Analizi

Veriler, düzenli olan belli altyapılardan oluşabilirler. Bölüt Analizi, veri içindeki düzenli yapıların bulunması işlemidir. Veri içindeki yapı bulunmakla, veri, benzer örüntüler, özellikler ve diğer karakteristiklere göre sınıflandırılmış olmaktadır. Bölüt analizi, veri ambarlığı, bilgi toplama, doküman sınıflandırma, örüntü işleme, işaret kodlama ve makine öğrenmesi gibi çok çeşitli alanlarda kullanılır. Bu uygulamalardaki en büyük zorluk, elde istatistiksel bir ön bilginin bulunmayışıdır. Karar veren mekanizma, veri üzerinde mümkün olduğunca varsayım yapmalıdır.

Bölüt analizinin sosyal boyutları incelendiğinde, Aristoteles' in ortaya sunduğu sınıflandırma modeli ilk çıkış noktası olarak alınabilir. İnsan beyninde de bölütleme sürecinin (görsel algılayıştaki boyut problemi göz ardı edilirse) başarıyla gerçeklediği gözlenmektedir. Bu konu ilk olarak tıp alanında matematiksel bir boyuta geçirilmiştir. Bölüt analizi yöntemleri hasta teşhisinde kullanılmıştır. Hastalar, taşıdıkları çeşitli semptomlara göre sınıflandırılmış. Bu sınıflandırma bilgileri teşhis için kullanılmıştır. Teşhis veya tanıma için kullanılan bir çok alanda, gözlem yoluyla verideki yapılar bulunmaya çalışılır.

Bölütleme (desteksiz sınıflama) işlemini sınıflama işleminden ayıran önemli özellikler vardır. Sınıflama işleminde, daha önceden belirlenmiş etiketli örüntüler bulunmaktadır. Bu işlemde amaç, yeni gelen ve etiketlenmemiş bir veriyi uygun sınıfa yerleştirmektir (veriyi etiketlemek). Eldeki etiketlenmiş örüntüler (eğitim

örüntüleri) yeni örüntüyü etiketlemek için kullanılan sınıf açıklamalarına sahiptir. Bölütleme işleminde ise amaç, eldeki bir grup etiketlenmemiş örüntüden anlamlı bölütler oluşturmaktır. Etiketler, aynı zamanda verilerin yapısı üzerinde belirlenen bölütler anlamına gelir ve tamamıyla gelen verilerin ürünüdür.

4.2.1 Bölüt Analizi Bileşenleri

Bölüt analizi yapan programlar, veriyi daha önceden tanımlanmış kalıplara yerleştirerek inceler veya veri üzerindeki doğal grupları inceler. Bölüt analizi çok boyutlu bir uzayda nokta veya ölçüm vektörü olarak temsil edilen bir grup verinin benzerlik ölçüsüne göre bölütlere dağıtılması işlemidir. Belli bir bölüt içindeki veriler birbirlerine, farklı bölütlerdeki verilerden daha çok benzerler.

Bölütleme sonucu bölütlere dağılan veriler şu özellikleri taşırlar:

1. Bir bölüt içindeki veriler, farklı bölüttekilere oranla birbirlerine daha çok benzerler.

2. Bir bölüt, daha az sayıda nokta ile diğer bölütlerden ayrılır ve bağıl olarak daha yoğun noktalardan oluşur (Jain ve diğ., 1999).

Bölütlemede önemli olan iki konu, çiftler arasındaki benzerliğin nasıl ölçüleceği ve bölütler oluşturulduktan sonra iyileştirmenin nasıl gerçekleneceğidir.

Bölüt analizinde hangi ölçütün ve özelliklerin kullanılacağı önemlidir. Bu özellikler nümerik veya mantıksal olabilir. Özelliklerin sayısı ve tipi, sınıflama ölçütü tipi gerektiğinde veri işlendikçe değiştirilmelidir (Ross, 1995). Veriler her bir boyutun bir özelliğe karşılık geldiği çok boyutlu vektörler olarak ifade edilirler. Bu özellikler nicelik veya nitelik belirtebilir. Örneğin kullanılan özellikler ağırlık ve renk ise “20, siyah” ifadesi 20 birim ağırlığında olan siyah bir nesnenin gösterilimine karşılık düşer. Özellikler iki çeşide ayrılabilir (Jain ve diğ,1999):

1. Nicel özellikler

   - Sürekli değerler (örn. ağırlık)

   - Ayrık değerler (örn. bilgisayar sayısı)

   - Aralık değerleri (örn. bir olayın süresi)

2. Nitel özellikler

   - Nominal veya sırasız (örn. renk)
   - Ordinal (örn. askeri rütbe, sıcaklığın niteliksel olarak değerlendirmesi: “serin” veya “sıcak”, ses yoğunluğu: “sessiz” veya “sesli”)

Bölüt Analizi Ana Bileşenleri:

- “x”, bölütleme algoritması tarafından kullanılan tek bir veri örneğidir. Genellikle m ölçüm içeren bir vektördür. “$x = (x_1 x_2 ..x_m)$”
- “x_j”, x örneğinin tekil bir skaler bileşeni, bir özellik değerini ifade eder.
- m, örneğin veya örnek uzayının boyutunu belirler.
- “$X = \{x_1, x_2 ..x_n\}$”, bölütlenecek veri topluluğudur. i. örnek “$x_i = (x_{i,1} ..x_{i,m})$” olarak ifade edilir. Bir çok durumda bölütlenecek veri topluluğu nxm veri matrisi olarak değerlendirilir.
- “c”, bölüt sayısını belirtir, $2 \le c < n$.
- Kesin bölütleme yöntemleri her x_k veri örneğini bir A_i etiketli bölüte atar. Belli bir veri topluluğu (X) için tüm bölütlerin topluluğu $A = \{A_1, A_2 ..A_c\}$ şeklinde belirtilir.
- Uzaklık ölçüsü, verilerin benzerliğini sayısallaştırmak için özellik uzayında kullanılan bir metriktir.
- Bulanık bölütleme süreçleri her bir x_k veri örneğini, her bir j bölütüne belli bir oransal üyelik derecesi (μ_{kj}) ile atar.

Bölütleme, n veriden oluşan bir X veri topluluğunun c alt bölüt sayısını belirleme ve X' i c $(2 \le c < n)$ bölüte ayırma süreçlerini içerir. $c = 1$, veride bölütler olduğu hipotezinin çürütülmesidir. $c = n$, her bir verinin tek başına bir bölüt oluşturduğu durumdur (Ross, 1995).

Her zaman için özelliklerden, en açıklayıcı ve ayırt etme için en faydalı olanlarını yalıtmak ve analiz aşamasında bu özellikleri kullanmak gerekir. Özellik seçimi işlemi, mevcut özelliklerden sonuçta kullanılacak olan en uygun alt kümeyi belirlemektir. Özellik türetme ise orijinal özelliklerden yeni özellikler hesaplama yöntemidir. Her iki durumda da sınıflandırma başarımını ve bilgi işlemsel etkinliği artırma amaçlanmaktadır. Sınıflandırma yapılırken özellik seçme işlemi gerçeklenebilir. Fakat bölütleme işleminde, çeşitli özellik alt kümeleri için veriler ayrı ayrı bölütlenir ve belli bir geçerlilik ölçütü kullanılarak bu sonuçlar değerlendirilir. Özellik azaltmanın bir başka yararı da verinin anlaşılabilir boyutlarda tanımlanmasını sağlamaktır.

4.2.2 Benzerlik Ölçüleri

Benzerlik, bir bölütün tanımlamasında kullanıldığından dolayı bir çok bölütleme süreci için aynı özellik uzayında bulunan iki veri arasındaki benzerliği ölçmek gereklidir. Özelliklerin çeşitliliğinden dolayı, uzaklık ölçüsü dikkatli seçilmelidir. Uzaklık ölçüsünü kullanarak özellik uzayında veriler arasındaki farklılıkları hesaplamak en çok kullanılan bir yöntemdir (Jain ve diğ,1999).

Sürekli değişkenler için en popüler metrik Euclid uzaklığıdır ve Minkowski metriğinin özel bir halidir ($p = 2$). Minkowski metriği (4.1) denkleminde tanımlanmıştır.

$$d_p = (x_i, x_j) = \left(\sum_{k=1}^{m} (x_{i,k} - x_{j,k})^p\right)^{1/p} = \left\| x_i - x_j \right\|_p \qquad (4.1)$$

Euclid uzaklığı, yoğun ve yalıtılmış veriler için iyi bir ölçüttür. Bu yöntemin bir sakıncası geniş ölçekli özelliklerin birbirlerine göre baskın konuma gelmesidir. Bunu engellemek için sürekli özellikler normalize edilir (belli bir genel aralığa veya sapmaya göre) veya özellikler için ağırlıklar belirlenir. Sürekli olmayan bazı özellikler için uzaklık belirlemesi yapmak zor olabilir. Bu tez çalışmasında kullanılan veriler bu türde özellikler içermediği için bu problem üzerinde çalışma yapılmamıştır.

4.3 Bölütleme Yöntemleri

Çok boyutlu veri toplulukları içinde mevcut bulunan çeşitli yapıların bulunması için geçerli olan genel olarak kabul görmüş bir algoritma yoktur. Farklı bölütleme

algoritmaları aynı veri topluluğu üzerinde farklı bölütler oluşturabilir. Bu farklılık, bölüt şekli, benzerlik ölçüsü ve bölütleme ölçütü seçimlerinden kaynaklanmaktadır.

İnsanlar iki boyutlu olarak görsel açıdan otomatik bölütleme sürecini başarıyla uygulamaktadır. Fakat bir çok problem, daha yüksek boyutlarda çalışmayı gerektirir. Buna ek olarak veri her zaman ideal şekillerde oluşmaz. Bu da bir çok algoritmanın oluşma sebebidir. Yeni çıkan bir algoritma diğerine göre belli bir örüntü dağılımı için daha iyi sonuçlar üretmektedir.

Bölütleme Algoritmalarındaki temel farklılıklar:

- Birleştirmeli (Agglomerative) / Bölmeli (Divisive): Algoritmik yapı ve işlemden gelen bir faklılıktır. Birleştirmeli yaklaşım her bir verinin tek bir bölüt içinde olması ile başlar ve belli bir durma ölçütü sağlanana dek bu bölütleri birleştirir. Bölmeli yöntemde tüm veriler tek bir bölüt içindedir ve belli durma ölçütüne kadar bölüt sayısı artırılarak veriler bölünür.

- Ardışıl (Monothetic) / Eş Zamanlı (Polythetic): Özelliklerin bölütleme sürecinde ardışıl veya eş zamanlı olarak kullanılması ile ilgilidir. Bir çok algoritma, veriler arasındaki uzaklığı belirlerken tüm özellikleri eş zamanlı olarak hesaba katar. Basit bir ardışıl algoritma bölüt analizi için özellikleri ardışıl olarak değerlendirir.

- Kesin / Bulanık: Kesin bölütleme algoritması, işlem sırasında ve çıkışta her bir veriyi farklı bir bölüte atar. Bulanık bölütleme algoritması, her bir bölüt için verilere bir üyelik derecesi atar. Bulanık olan bir bölütleme sonucu, kesin hale dönüştürülebilir.

- Determinist / Stokastik: Bu özellik, kısmi yaklaşımlarda karesel hata fonksiyonunu optimize etmek için kullanılır. Bu optimizasyon mevcut yöntemler kullanılarak veya tüm olası etiketlerin durum uzayı rasgele taranarak gerçeklenir.

- Artımlı (Incremental) / Artımlı Olmayan: Bu özellik, bölütlenecek veri topluluğu çok büyük olduğunda ve yürütme zamanı ve bellek gereksinimi, algoritmanın yapısını etkilediğinde ön plana çıkar (Jain ve diğ. ,1999).

Bölütleme algoritmaları hiyerarşik ve iteratif bölmeli (partitional) olarak ikiye ayrılır. Bölütleme algoritmasında aranması gereken şartlar, hız, güvenilirlik ve tutarlılıktır.

4.3.1 Hiyerarşik Bölütleme Algoritmaları

Hiyerarşik algoritmalar, bölüt içi dağılımı minimize eden ve bölütler arası dağılımı maksimize eden bölütlemeler oluşturmaya çalışır. Optimum çözümün bulunduğunu garantilemek için n adet m boyutlu verinin c adet bölüte ayrıldığı tüm olası bölütlemeler değerlendirilir. Bu işlem, bilgi işlemsel açıdan oldukça yüklü bir iştir. Hiyerarşik algoritmalar çıktı olarak dendogramlar üretirler. Dendogram, veri üzerinde farklı bölütler üretmek üzere herhangi bir seviyede kesilebilir.

Hiyerarşik Bölütleme Algoritmalarının en yaygın olarak kullanılanları, Single-link algoritması ve Complete-link algoritmasıdır. Bu algoritmalar, birbirlerinden bölütler arası çiftli benzerlikleri karakterize etmeleri yolları ile farklılaşmışlardır. Single-link yönteminde iki bölüt arasındaki uzaklık, bu iki bölüte ait çiftler (ilk veri ilk bölütten ve ikinci veri diğerinden olmak üzere) arasındaki minimum uzaklıktır. Complete-link algoritmasında ise iki bölüt arasındaki uzaklık, bu iki bölüte ait çiftler arasındaki maksimum uzaklıktır. Her iki algoritmada da bölütler, minimum uzaklık ölçütü göz önüne alınarak bitiştirilir. Single-link algoritması Complete-link algoritmasından daha esnektir. Örneğin, Single-link algoritması Complete-link algoritmasından farklı olarak ortak merkezli bölütleri çıkarabilir. Fakat bir çok uygulamada Complete-link algoritması Single-link algoritmasından daha yararlı hiyerarşiler oluşturur.

4.3.1.1 Single-link Algoritması

Birleştirmeli Single-link algoritması, hiyerarşik bir yolla veriler arasındaki uzaklığa bağlı olarak her bir adımda iki bölütü birleştirir. Birleştirme işlemi, minimum uzaklıklar arası minimum uzaklık ölçütüne göre yapılmaktadır. Single-link algoritmasının yürütme adımları Şekil 4.1'de görülmektedir. Sonuçta oluşan dendogram istenilen noktada kesilebilir. Single-link algoritması, dağınık veya uzatılmış bölütler üretir ve zincirleme etkisine uğrar.

---

1. *Her bir veriyi kendi bölütüne yerleştir. Tüm farklı sırasız veri çiftleri için veriler arası uzaklık listesini oluştur. Bu listeyi küçükten büyüğe doğru sırala.*
2. *Sıralı liste üzerinde tarama yap. En yakın bölütleri minimum uzaklık ölçütüne göre bul. Bu bölütleri bir bölüt haline getir. Yakınlık matrisini güncelle.*
3. *Eğer tüm veriler bir bölüt içinde ise dur. Aksi takdirde 2. Adıma geri dön.*

Şekil 4.1 Single-link Bölütleme Algoritması

4.3.1.2 Complete-link Algoritması

Birleştirmeli Complete-link algoritması, hiyerarşik bir yolla veriler arasındaki uzaklığa bağlı olarak her bir adımda iki bölütü birleştirir. Birleştirme işlemi, maksimum uzaklıklar arası minimum uzaklık ölçütüne göre yapılmaktadır. Complete-link algoritmasının yürütme adımları Şekil 4.2'de görülmektedir. Sonuçta oluşan dendogram istenilen noktada kesilebilir. Complete-link algoritması, sıkı bağlı veya yoğun bölütler üretir.

---

1. *Her bir veriyi kendi bölütüne yerleştir. Tüm farklı sırasız veri çiftleri için veriler arası uzaklık listesini oluştur. Bu listeyi küçükten büyüğe doğru sırala.*
2. *Sıralı liste üzerinde tarama yap. En yakın bölütleri maksimum uzaklık ölçütüne göre bul. Bu bölütleri bir bölüt haline getir. Yakınlık matrisini güncelle.*
3. *Eğer tüm veriler bir bölüt içinde ise dur. Aksi takdirde 2. Adıma geri dön.*

Şekil 4.2 Complete-link Bölütleme Algoritması

4.3.2 İteratif Bölmeli Bölütleme Algoritmaları

İteratif Bölmeli (Partitional) algoritmalar, veri üzerinde tek bir bölütleme elde eder. İstenen bölüt sayısı önceden belirlenmektedir. Bu tür algoritmalar, büyük veri bölütleri ile çalışırken avantajlıdır. Çoğunlukla bölütleri yerel (belli bir alt bölüt üzerinde tanımlanan) veya global (tüm veriler üzerinde tanımlanan) olarak tanımlanan belli bir ölçüt fonksiyonunu optimize ederek oluşturur. Optimum ölçüt değeri için olası tüm etiketleme topluluğu üzerinde gerçeklenen arama, bilgi işlemsel olarak oldukça zaman alır. Pratikte, algoritma farklı başlama durumları için çalışır ve tüm çalışmalar sonucu en iyi konfigürasyon, çıkış bölütlemesi olarak alınır.

Bu algoritmaların çoğu Karesel Hata Algoritmaları olarak bilinir. Veriler ve merkezler arası uzaklıkları denetleyerek bölütleri oluşturur. Yalıtılmış ve yoğun bölütler için doğru sonuçlar üretir. Bu algoritmalar içinde en önemli olanları:

1. *k*-means (Hard *c*-means olarak da anılmaktadır) Algoritması
2. Fuzzy (Bulanık) *c*-means Algoritması
3. Graf Teorisine Dayanan Algoritmalar
4. En Yakın Komşu (Nearest Neighborhood) Bölütleme Algoritması

Karesel Hata: Belli bir X veri topluluğu için karesel hata şu şekilde hesaplanır:

$$e^2(X,L) = \sum_{j=1}^{c} \sum_{i=1}^{n_j} \left\| x_i^{(j)} - v_j \right\|^2 \quad (4.2)$$

Buradaki $x_i^{(j)}$ j.bölütteki i. veriyi, v_j, j. bölütün merkezini, n toplam bölüt sayısını ve n_j bir bölütteki veri sayısını belirtir.

4.3.2.1 *k*-means Bölütleme Algoritması

k-means bölütleme algoritması, karesel hata ölçütünü kullanan yaygın olarak kullanılan bir algoritmadır. Kolay gerçeklenmesi ve n adet veri için O(n) zaman karmaşıklığından dolayı popülerdir. Belli bir rasgele bölütleme ile başlar ve belli bir yakınsama ölçütü (örn. bir bölütten diğerine yeniden atama olmaması, veya karesel hatanın belli bir değere yakınsaması) sağlanana dek bölütlere atama işlemi devam eder. Bu algoritmanın en temel problemi ilk bölütlemenin seçimine duyarlı olması ve ilk bölütleme uygun seçilmediyse yerel bir minimuma doğru yakınsayabilmesidir. *k*' yı seçme, başlangıç bölütlemesi, bölütleme güncelleme, bölüt sayısını düzeltme ve durma ölçütü konularında genel kurallar olmadığından dolayı problemler yaşanır.

Şekil 4.3'te *k*-means algoritması temel olarak verilmiştir *(Jain ve Mao, 2000)*. Temel algoritma ilk dört adımdan oluşmaktadır. 5. adım algoritmaya iyileştirme olarak eklenmiştir.

---

1. *k bölüt merkezini k adet rasgele seçilen veriye ya da veri topluluğunu içeren uzayda k adet rasgele tanımlanan noktaya uygun şekilde seç.*
2. *Her bir veriyi en yakın bölüt merkezine ata.*
3. *Mevcut bölüt üyelik değerlerini kullanarak yeniden bölüt merkezlerini hesapla.*
4. *Eğer bir yakınsama koşulu karşılanmamışsa 2. Adıma geri dön. Tipik bir yakınsama ölçütü: karesel hatanın minimum olması veya verilerin yeni bölütlere minimal düzeyde atanması veya hiç atanmaması. (Ölçüt fonksiyonunun optimum değeri)*
5. *Bölüt sayısını mevcut bölütleri birleştirerek, parçalayarak, küçük veya sınır dışındaki bölütleri silerek düzelt.*

Şekil 4.3 *k*-means Bölütleme Algoritması

4.3.2.2 *k*-means (HCM) Algoritması Bileşenleri

k-means algoritması Hard c-means (HCM) algoritması olarak da anılmaktadır. Bu algoritmada, bir veri örneği sadece tek bir veri bölütüne dahil edilir. $\{A_i, i = 1,2..c\}$, X' in c-bölütleme topluluğudur (Ross, 1995).

$$\bigcup_{i=1}^{c} A_i = X \tag{4.3}$$

$$A_i \cap A_j = \phi \quad \forall\ i \neq j \text{ için} \quad \text{ve} \quad \phi \subset A_i \subset X \qquad \forall\, i \text{ için} \tag{4.4}$$

Tüm bölütlerin toplamı veri örnekleri evrensel kümesini oluşturur. (4.4)'e göre bir veri örneği birden fazla bölüte dahil olmadığı için bölütlerin kesişmez. Bir bölüt boş olamaz veya tüm veri örneklerini içeremez. $c = 2$ için (4.3) ve (4.4)'e göre,

$$A_2 = \overline{A}_1 \qquad A_2 \cup \overline{A}_1 = X \qquad A_1 \cap \overline{A}_1 = \phi \tag{4.5}$$

Bu durumda aşağıdaki denklemler tanımlanabilir.

$$\bigcup_{i=1}^{c} \chi_{A_i}(x_k) = 1 \qquad \forall k \text{ için} \tag{4.6}$$

$$\chi_{A_i}(x_k) \wedge \chi_{A_j}(x_k) = 0 \qquad \forall k \text{ için} \tag{4.7}$$

$$0 < \sum_{k=1}^{n} \chi_{A_i}(x_k) < n \qquad \forall i \quad \text{için} \tag{4.8}$$

Karakteristik fonksiyon $\chi_{A_i}(x_k)$ şu şekilde tanımlanır:

$$\chi_{A_i}(x_k) = \begin{cases} 1 & x_k \in A_i \\ 0 & x_k \notin A_i \end{cases} \tag{4.9}$$

U $(c \times n)$ matrisi $x_{ij}\ (i = 1,2..c;\ j = 1,2..n)$ elemanlarından oluşmaktadır.

X için bir kesin bölütleme uzayı (4.10)'da tanımlanmıştır.

$$M_c = \left\{ U \mid \chi_{ik} \in \{0,1\}, \sum_{i=1}^{c} \chi_{ik} = 1; 0 < \sum_{k=1}^{n} \chi_{ik} < n \right\} \quad (4.10)$$

Herhangi bir $U \in M_c$ bir kesin c-bölütlemedir. Her hangi bir M_c c-bölütlemenin büyüklüğü şu şekilde tanımlanır: (n veri noktası için tekil c-bölmelerin sayısını verir.)

$$\eta_{Mc} = \left(\frac{1}{c!}\right)\left[\sum_{i=1}^{c}\binom{c}{i}(-1)^{c-i}.i^{n}\right] \quad (4.11)$$

n adet veri örneği için olası c-bölütlemeler arasından M_c bölütleme uzayı için en uygun c-bölütlemenin nasıl seçileceği sorusunun cevabını bölütleme ölçütü vermektedir. Hard c-means algoritması için sunulan bir hedef fonksiyonu (objective function), uzaklığı karakterize etmek için Euclid metriğini kullanan sınıf içi karesel hata toplamından hesaplanır. Bu fonksiyon $J(U, v)$ olarak belirtilir. U partition matrisidir. Ve v bölüt merkezi vektörüdür.

Hedef fonksiyonu şu şekilde verilir:

$$J(U, v) = \sum_{k=1}^{n}\sum_{i=1}^{c} \chi_{ik} (d_{ik})^2 \quad (4.12)$$

d_{ik} (m boyutlu özellik uzayında, R_m) k.veri örneği x_k ve i. bölüt merkezi v_i arasındaki Euclid uzaklığı ölçüsüdür.

$$d_{ik} = d(x_k - v_i) = \|x_k - v_i\| = \left[\sum_{j=1}^{m}(x_{kj} - v_{ij})^2\right]^{1/2} \quad (4.13)$$

Her bir veri örneğinin R_m uzayında konumunun tanımlanması için m koordinat gerektiğinden, bölüt merkezleri de aynı uzay içinde m koordinatla tanımlanır. Dolayısıyla i. bölüt merkezi m uzunluğunda bir vektördür:

$$v_i = \{v_{i1} v_{i2} .. v_{im}\} \quad (4.14)$$

v_i' nin j. Koordinatı (4.15)'te tanımlanmıştır.

$$v_{ij} = \frac{\sum_{k=1}^{n} \chi_{ik} \cdot x_{kj}}{\sum_{k=1}^{n} \chi_{ik}} \tag{4.15}$$

J fonksiyonu için minimum değeri üreten bölütleme optimum bölütleme U^* olarak alınır.

$$J^*(U^*, v^*) = \min_{U \in M_c} J(U, v) \tag{4.16}$$

U^*'ı bulmak ($M_c \to \infty$) pratik açıdan oldukça zor bir problemdir. Bu problemi gidermek için iteratif optimizasyon arama algoritması kullanılır. Şekil 4.4'te Hard c-Means (HCM) algoritması (*k*-means algoritmasının daha teorik açıklaması) görülmektedir.

---

1. *c'yi ($2 \le c < n$) sabitle ve başlangıç U matrisini oluştur.*

$$\underset{\sim}{U}^{(0)} \in M_c$$

$r = 0,1,2..$ *adımlarını gerçekle.*

2. *c merkez vektörlerini hesapla.*

$$\{ \underset{\sim}{U}^{(r)} \ ile \ v_i^{(r)} \}$$

3. $U^{(r)}$*'yi güncelle. Güncellenmiş karakteristik fonksiyonları ($\forall\ i,\ k$ için) hesapla.*

$$x_{ik}^{(r+1)} = \begin{cases} 1 & d_{ik}^{(r)} = \min\{d_{jk}^{(r)} \ \forall\ j \in c \ için\} \\ 0 & aksi \ takdirde \end{cases}$$

4. *Eğer* $\| \underset{\sim}{U}^{(r+1)} - \underset{\sim}{U}^{(r)} \| \le \varepsilon_L$ *ise dur, aksi takdirde* $r = r + 1$ *ve 2. Adıma geri dön.*

Şekil 4.4 Hard c-means Algoritması

Temel *k*-means algoritmasının başarımını artırmak için bir çok girişimde bulunulmuştur. Bunlar:

1. Bir bulanık ölçüt fonksiyonu kullanmak: Fuzzy c-means (veya Fuzzy *k*-means)

2. Genetik algoritmaları kullanmak ve sonuç bölmelerini optimize etmek için arama yöntemleri kullanmak.

3. Yapay Sinir Ağı (Artificial Neural Network) kullanmak.

Bir çok *k*-means algoritması sürümü mevcuttur. Bazıları başlangıç bölütlemesini iyi seçmeye çalışır.

Bir başka sürüm, sonuç bölütlerinin varyansları önceden belirlenen bir eşik değerini aştığında ilgili bölütleri parçalayan veya merkezleri arasıdaki uzaklık bu eşik değerinden küçük olduğunda bölütleri birleştiren bir algoritma sunar. Şekil 4.3'te verilmiş olan *k*-means algoritması bu sürüme göre yazılmıştır. Bu sürüm kullanılarak uygun eşik değerlerinin belirlenmesi ile herhangi bir bölütlemeden başlayıp optimal bölütlemeye erişmek mümkün olmaktadır.

4.3.2.3 Graf Teorisine Dayanan Algoritmalar

MST (Minimal Spanning Tree) algoritmasına benzer hiyerarşik yaklaşımlar graf teorisine dayanır.

4.3.2.4 En Yakın Komşu Bölütleme Yöntemi

En Yakın Komşu Bölütleme yöntemi (yakınlık, bölütlerin oluşturulmasında anahtar rol oynadığından) en yakın komşu uzaklıkları bitiştirme yoluyla veri üzerinde tarama yapar. Etiketli olmayan verileri sınıflamak için kullanılır.

4.3.2.5 Bulanık Bölütleme Yöntemi

Bulanık bölütleme yönteminde her bir veri, belli bir bölüte belli bir üyelik değeri ile dahil olur. Bulanık bölütlemede her bir bölüt bulanık olarak tanımlanmaktadır. Üyelik değerine belli bir eşik değeri uygulayarak bulanık bölütlemeden kesin bölütlemeye geçilir. En popüler bulanık bölütleme algoritması Fuzzy c-Means (FCM)' dir. Bezdek tarafından 1981' de FCM algoritmasının genelleştirilmiş hali sunulmuştur. Şekil 4.5'te Temel Bulanık Bölütleme Algoritması tanıtılmaktadır.

1. *n nesneyi k bölüte (n x k üyelik matrisi U ' yu seçerek) ata. ($\mu_{ij} \in [0,1]$)*

2. *U ' yu kullanarak bulanık ölçüt fonksiyonunu bul. Örn. Belli bir bölütleme için ağırlıklı karesel hata ölçütü. Bir olası bulanık ölçüt fonksiyonu:*

$$E^2(x,U) = \sum_{i=1}^{N}\sum_{k=1}^{K} u_{ij} \|x_i - c_k\|^2$$

Buradaki $c_k = \sum_{i=1}^{n} u_{ik} x_i$, k. bulanık bölüt merkezidir. Bu ölçüt fonksiyonunu azaltmak için verileri yeniden ata ve U ' yu yeniden hesapla.

3. *2.adımı U önemli ölçüde değişmeyinceye kadar tekrarla.*

Şekil 4.5 Temel Bulanık Bölütleme Algoritması

4.3.2.6 Fuzzy c-Means Algoritması

Fuzzy c-Means (FCM) bulanık bölütleme yöntemini kullanır. $\{\tilde{A}_i, i = 1,2,..c\}$ bulanık bölütleri X veri noktaları evrensel kümesinde bir bulanık c bölütleme olarak tanımlanmaktadır (Ross, 1995). FCM algoritması sonucunda tek bir nokta birden fazla bölütte kısmi üyeliğe sahip olabilir. i . bölütdeki k . verinin üyelik değeri:

$$\mu_{ik} = \mu_{\tilde{A}_i}(x_k) \in [0,1] \qquad (4.17)$$

Bir verinin üyelik değerleri toplamının "1" olması gerekmektedir.

$$\sum_{i=1}^{c} \mu_{ik} = 1 \quad \forall k = 1,2,..n \text{için} \qquad (4.18)$$

Kesin bölütlemede olduğu gibi boş bölüt ve tüm verileri içeren bir bölüt olamaz. Bu özellik (4.19) ile ifade edilir.

$$0 < \sum_{k=1}^{n} \mu_{ik} < n \qquad (4.19)$$

$$\mu_{ik} \wedge \mu_{jk} \neq 0 \tag{4.20}$$

$$\bigcup_{i=1}^{c} \mu_{\tilde{A}_i}(x_k) = 1 \quad \forall k \tag{4.21}$$

$c = 2$ için,

$$\tilde{A}_i \cap \tilde{A}_j \neq \phi \qquad \phi \subset \tilde{A}_i \subset X \tag{4.22}$$

Böylelikle bir bulanık kısmi matrisler ailesi yazılabilir. M_{fc}, n veri örneği üzerinde c bölüt için şu şekilde tanımlanabilir:

$$M_{fc} = \left\{ \underset{\sim}{U} \mid \mu_{ik} \in [0,1]; \sum_{i=1}^{c} \mu_{ik} = 1; 0 < \sum_{k=1}^{n} \mu_{ik} < n \right\} \quad (i = 1,2..c; k = 1,2..n) \tag{4.23}$$

$\tilde{U} \in M_{fc}$ bir bulanık c-bölütlemedir. Birbirini örten bölütlerden oluşur ve bölüt üyeliklerini ifade etmek için sonsuz sayıda üyelik değeri söz konusudur ($\eta M_{fc} = \infty$).

Fuzzy c-Means Algoritması, tüm bu hesaplamalara ağlı olarak oluşturulabilir. n veriden oluşan topluluğu c' li bölmelere ayırmak için Fuzzy c-Means bölmeleme matrisini $(\tilde{U})$ belirleyen bir yöntem olarak bulanık c-bölütleme için J_m hedef fonksiyonu şu şekilde tanımlanır.

$$J_m(\underset{\sim}{U}, v) = \sum_{k=1}^{n} \sum_{i=1}^{c} (\mu_{ik})^{m'} (d_{ik})^2 \tag{4.24}$$

burada μ_{ik}, i. bölütteki k. verinin üyelik derecesidir. m', ağırlık parametresidir ($m' \in [1, \infty)$). Bu parametre, bölütlemedeki bulanıklık miktarını belirtir. Kesin bölütlemede olduğu gibi J_m fonksiyonu yüksek değerlere sahip olabilir. En küçük olanı en iyi bölütleme olacaktır. Sonsuz sayıda bulanık bölütleme oluşturulabileceğinden dolayı en olası, optimum çözüm, karmaşık olmayan bir arama ile bulunmalıdır.

Her bir bölüt merkezinin koordinatları kesin bölütlemede olduğu gibi hesaplanabilir (4.25).

$$v_{ij} = \frac{\sum_{k=1}^{n} \mu_{ik}^{m'} . x_{kj}}{\sum_{k=1}^{n} \mu_{ik}^{m'}} \tag{4.25}$$

Kesin bölütlemede olduğu gibi optimum bulanık c-bölütleme, bölütlemelerin en küçüğü olacaktır.

$$J_m^*(\underset{\sim}{U}^*, v^*) = \min_{M_{fc}} J(\underset{\sim}{U}, v) \tag{4.26}$$

Çoğu optimizasyon probleminde olduğu gibi (4.26)'nın çözümünün global optimum olduğu garanti edilemez. Önceden tanımlanan doğruluk seviyesine göre en iyi çözüme ulaşılmaya çalışılır. Tüm bu hesaplamalara göre FCM Algoritması Şekil 4.6'da tanıtılmaktadır.

---

1. *c' yi* $(2 \le c < n)$ *sabitle ve* m' *için bir değer seç. Başlangıç* $\underset{\sim}{U}^{(0)}$ *bölütleme matrisini oluştur. Bu algoritmadaki her bir adım* r $(r = 0,1,2..)$ *ile etiketlenecek.*

2. $\{v_i(r)\}$ c *merkezlerini hesapla.*

3. r *. adım için* $\underset{\sim}{U}^{(r)}$ *partition matrisini güncelle.*

$$\mu_{ik}^{(r+1)} = \left[\sum_{j=1}^{c} \left(\frac{d_{ik}^{(r)}}{d_{jk}^{(r)}} \right)^{2/(m'-1)} \right]^{-1} \qquad (I_k = 0 \text{ için})$$

veya $\mu_{ik}^{(r+1)} = 0$, $i \in \tilde{I}_k$ *olan tüm* i *bölütleri için*

buradaki $I_k = \{i \mid 2 \le c \le n; d_{ik}^{(r)} = 0\}$

ve $\tilde{I}_k = \{1,2,..c\} - I_k$

$$\sum_{i \in I_k} \mu_{ik}^{(r+1)} = 1$$

4. $\| \underset{\sim}{U}^{(r+1)} - \underset{\sim}{U}^{(r)} \| \le \varepsilon$ *ise dur, aksi takdirde* $r = r + 1$ *ve 2. adıma dön.*

Şekil 4.6 FCM Iteratif Optimizasyon Algoritması

4. adımda çözümün yeteri kadar iyi olup olmadığını belirlemek üzere iki ardışık adım arasında hesaplanan farklılık, belirlenen ölçüte göre değerlendirilir.

4.3.3 Bölütleme Yöntemlerin Karşılaştırılması

k-means yöntemi, yürütme zamanı açısından en etkin olan algoritmalardan biridir. *k*-means algoritması ve diğer sürümleri büyük veri topluluklarına uygulanabilir. Ayrıca *k*-means algoritmasının yerel olarak optimum sonuca ulaştığı görülmüştür (Jain, 1999). *k*-means' te başlangıç dağılımının uygun şekilde belirlenmesi ile etkin sonuçlara ulaşmak mümkündür. Dolayısıyla *k*-means' in iyileştirilmiş bir sürümü olan FCM, iyi sonuçlar üretmektedir. Bölüt tabanlı doküman sınıflama uygulamasında hiyerarşik algoritmaların kısmi algoritmalara göre daha iyi sonuçlar verdiği gözlenmiştir.

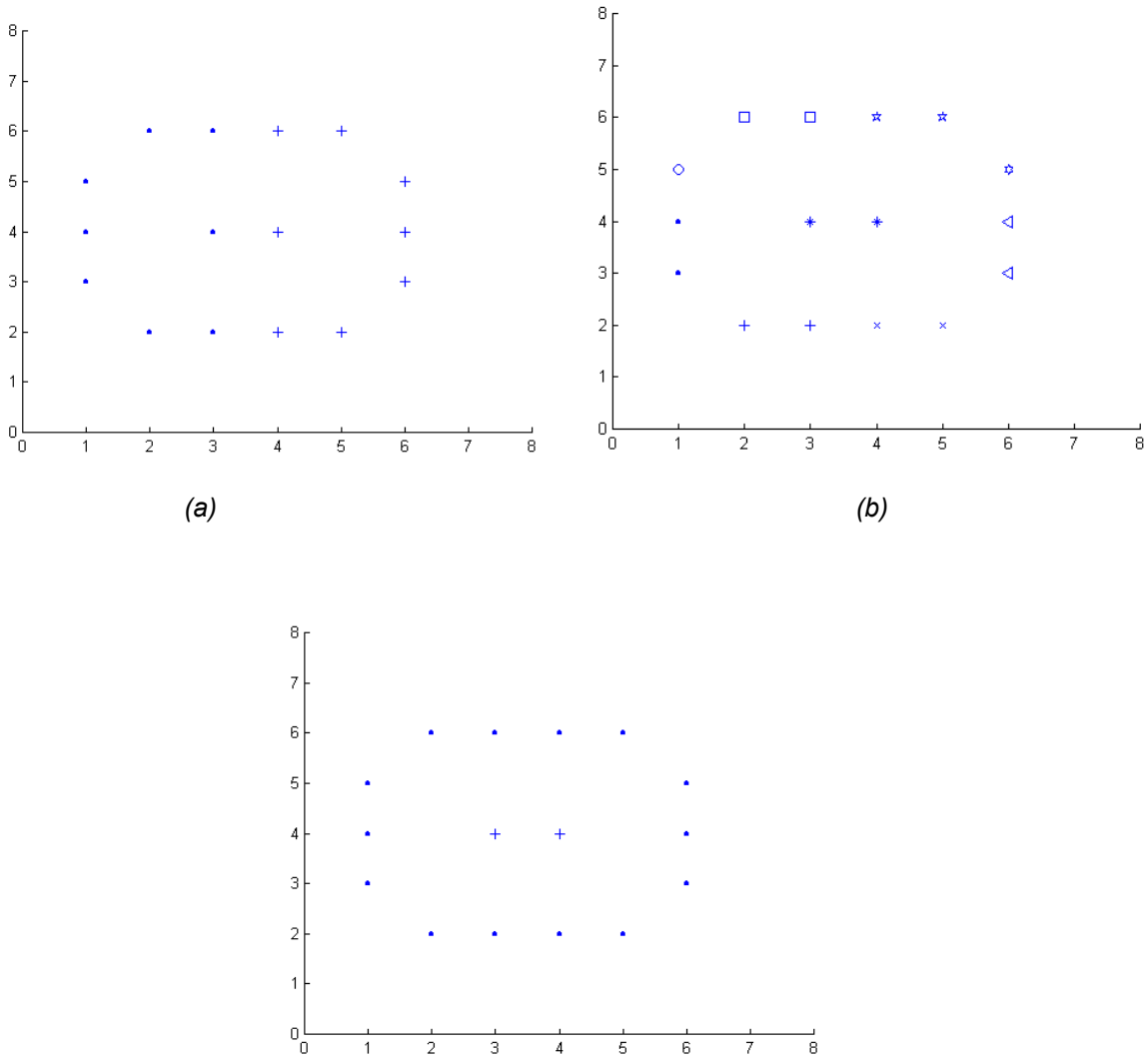
Bölütleme algoritmaları için:

1. Tüm bölütleme algoritmaları verilen bir veri topluluğu üzerinde bölütler (olsa da olmasa da) bulacaktır. Dolayısıyla verinin bölütleme eğiliminin olup olmadığı bölütleme algoritması uygulanmadan önce test edilmelidir.

2. En iyi bölütleme algoritması yoktur. Dolayısıyla belli bir veri grubu için bir çok bölütleme algoritması uygulanmalıdır.

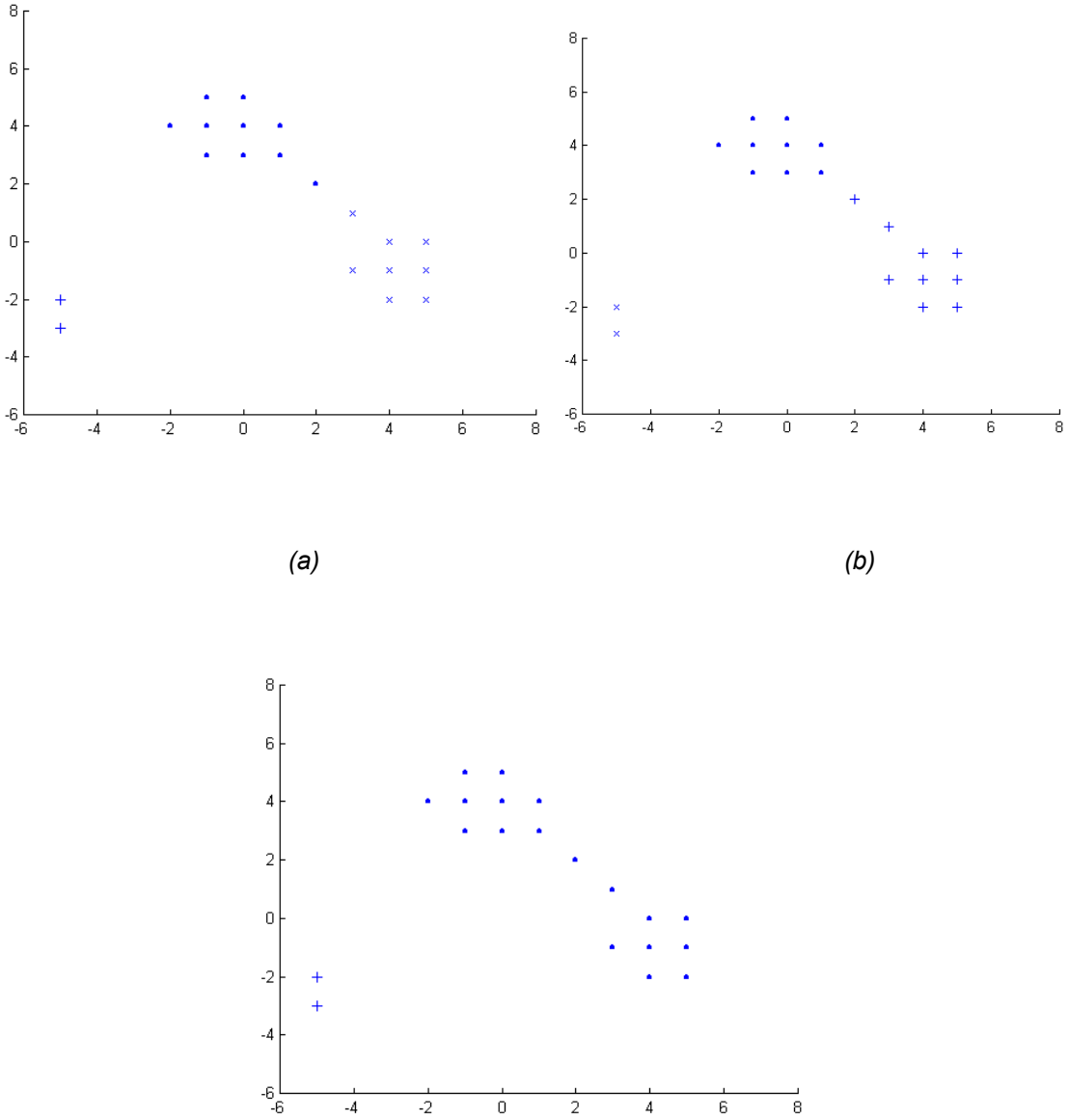
Hiyerarşik algoritmalar, kısmi algoritmalardan daha esnektir. Örneğin Single-link algoritması yoğun olmayan, iyi bir şekilde ayrılmış ve aynı merkeze sahip bölütler üzerinde iyi çalışır. *k*-means algoritması yoğun veriler üzerinde iyi çalışır. Bunun yanında kısmi algoritmaların zaman karmaşıklıkları (ve bellek gereksinimleri) daha düşüktür. Her iki grubun da iyi özelliklerini alan hibrit algoritmalar kullanmak yararlı olabilir.

Algoritmalar, farklı veri yapıları için farklı bölütler oluşturmaktadır. Bir veri topluluğu için FCM, Complete-link ve Single-link algoritmalarının yürütülmesi sonucu oluşan bölütler Şekil 4.7 (a), Şekil 4.7 (b) ve Şekil 4.7 (c)'de görülmektedir. Ortak merkezli ve görsel olarak iki bölütlü olarak algılanabilecek olan veri topluluğunun bu üç algoritma için ürettikleri farklı sonuçları uygulamaya göre farklı değerlendirilebilir.



Şekil 4.7 Bir Veri Topluluğu İçin FCM, Complete-link ve Single-link Algoritmaları Yürütme Sonuçları

Bir başka veri topluluğu için oluşturulan sonuç bölütleri Şekil 4.8 (a), Şekil 4.8 (b) ve Şekil 4.8 (c)'de verilmiştir. FCM ve Complete-link algoritmaları zincir yapısındaki veri grubu için iki bölüt üretmekte iken Single-link algoritması zincirleme etkisine uğrayarak zincir yapısındaki veri grubundaki verileri tek bir bölüte dahil etmiş yani veri üzerinde bir bölütleme tanımlamamıştır. Bu da Single-link algoritmasının veri üzerinde gürültü olması durumunda yanlış sonuçlar üretebileceğini göstermektedir. Sonuçta oluşan durumlar uygulama açısından değerlendirilmelidir.



(a) *(b)*

(c)

Şekil 4.8 Bir Veri Topluluğu İçin FCM, Complete-link ve Single-link Algoritmaları Yürütme Sonuçları

Tez kapsamında, Fuzzy c-Means, Complete-link ve Single-link algoritmaları kullanılarak, Java dilinde, bu algoritmalara ilişkin programlar yazılmıştır. Geliştirilen yazılım sistemi için kullanılan veri boyutu farklı olmasına rağmen algoritmaların çalışmaları düşük boyutlu veriler üzerinde test edilmiş ve program çıktıları anlamlı dosyalara kaydedilip kısa bir MatLab programı yazılarak grafik haline dönüştürülmüştür. Şekillerdeki farklı semboller farklı bölütleri ifade eder.

4.4 Bölüt Geçerliliği Testi

Bir çok durumda verideki bölüt sayısı (c) bellidir. Fakat bölüt sayısı önceden belli değilse c' nin diğer değerleri de analiz edilmelidir. Bu durumda en uygun c sayısının seçilmesi gerekmektedir. Bu problem, bölüt geçerliliği olarak bilinir. Etiketlenmemiş veri için bölüt geçerliliği için kesin bir ölçü yoktur. Bölütleme katsayısı ölçütü diğer yöntemlere kıyasla doğru sonuçlar üreten bir yöntemdir.

4.4.1 Bölütleme Katsayısı (Partition Coefficient)

Bölütleme katsayısı, bölüt geçerliliği için iki bulanık bölütün ayrılma derecesini belirler (Zadeh,1965).

U, n verinin 2-li bölütlemelerini göstermekte iken u_1 ve u_2, bulanık bölütleri arasındaki ayrılma derecesi skaler olarak (4.27) ile ifade edilebilir.

$$\rho(U;2) = 1 - \left[\bigcup_{k=1}^{n} (u_{1k} \wedge u_{2k}) \right] \tag{4.27}$$

Bu sayı, sonlu ve eşit büyüklükteki bölüt çiftleri için iyi bir tanımdır. Bu eşitlik, 2' den c' ye kadar ölçülüp minimumu alınırsa, genel bir ifadeye ulaşılabilir.

$$Z(U;c) = 1 - \left[\bigcup_{k=1}^{n} \left(\bigcap_{i=1}^{c} u_{ik} \right) \right] \forall U \in M_{fc} \tag{4.28}$$

$Z(U;c)$, her kesin bölmeleme için 1' dir. Ayrılma derecesi M_c üzerinde tek başına bir geçerlilik fonksiyonu olamaz.

Ayrılma derecesi yerine bölütleme katsayısı değeri hesaplanabilir. U, n veri topluluğunda c-bölütleme olmak üzere, U için bölütleme katsayısı (4.29)'daki gibi ifade edilebilir (Bezdek,1981).

$$F(U;c) = \sum_{k=1}^{n} \sum_{i=1}^{c} (u_{ik})^2 / n \tag{4.29}$$

$$(1/c) \leq F(U;c) \leq 1 \tag{4.30}$$

$$F(U;c) = 1 \Leftrightarrow U \in M_c \text{ (Kesin Bölütleme)} \tag{4.31}$$

$$F(U;c) = 1/c \Leftrightarrow U = [1/c] \tag{4.32}$$

Teorem: $U \in M_{fc}$, n veri noktasının c' li bölütlemesi olsun. Öyle ise, $1 \le c \le n$ için, F' in bölüt geçerliliği testi için kullanılabilmesi için, herhangi bir bulanık bölütleme algoritması kullanılarak $c = 2,3..n-1$ için X' in bir ya da birden fazla optimal bölütlemesi bulunur. Optimallik testi için tüm adaylar arasından en uygun c sayısı seçilmektedir.

$$\max_c \left\{ \max_{\Omega_c} \{F(U;c)\} \right\} \tag{4.33}$$

Eğer (U^*, c^*) bu koşulu sağlarsa U^*, adaylar arasındaki X' in en uygun bölütlemesidir. Eğer $F(U^*, c^*)$ 1' e yakın değilse, X' in tanımlanabilir bir bölüt altyapısının olduğu varsayılamaz. Sonuçta algoritma bir bölütleme üretememiştir (Bezdek,1981).

4.5 Büyük Veri Topluluklarının Bölütlenmesi

Bir çok uygulamada oldukça büyük verileri bölütlemek gerekebilir. Doküman sınıflandırma ve bilgi filtreleme sistemlerinde 100' den fazla boyutlu milyonlarca verinin bölütlenmesi gerekmektedir. Yaklaşımların çoğu bu sayıda veriyi işleyebilecek kapasiteye sahip değildir. Bu tür bölütleme için *k*-means kullanılabilir (Jain ve diğ.,1999).

Tüm veri topluluğunun ana bellekte tutulduğu durumlar için iyileştirme çalışmaları yapılmış ve bunlara uygun algoritmalar gerçeklenmiştir. Fakat bazı uygulamaların doğası gereği tüm veri topluluğu ana bellekte tutulamaz. Bu problemi çözmek için üç yöntem mevcuttur:

1. Veri topluluğu bir ikincil bellekte tutulur ve bu topluluğun alt kümeleri bağımsız olarak bölütlenir. Daha sonra tüm veri topluluğunu bölütlemek için bir birleştirme adımı yürütülür. Bu yaklaşım böl ve yönet yöntemi olarak adlandırılır. Veri homojen alt kümelere ayrılmalıdır.

2. Artımlı (Incremental) bölütleme algoritması: Tüm veri matrisi ikincil bir bellekte tutulur. Veri parçaları teker teker bölütleme için ana belleğe taşınır. Alandan kazanmak için ana bellekte sadece bölüt temsilcileri bulunur. Örnek bir artımlı algoritma için Fazli (1993) makalesi incelenebilir. Bu algoritmaların sıraya bağımlılığı çözümlenmemiş bir problem olarak kalmıştır.

3. Bölütleme algoritmasının paralel çalışan makinelerde gerçeklenerek etkinliği artırılabilmektedir.

4.6 Bölütlerin Temsil Edilmesi

Bir çok uygulamada (karar vermede önemli olan) sonuç bölütleri net bir şekilde temsil edilmelidir. Bu konu önemli olmasına rağmen araştırmacılar tarafından fazla üzerinde durulmamıştır. Bölütlerin ifade edilmesinde yöntemler,

1. Bölüt merkezi veya bölüt içindeki anlamlı ayrık noktalarla temsil etmek,

2. Sınıflama ağacında düğümlerle temsil etmek,

3. Bağlayıcı mantıksal ifadelerle temsil etmek şeklinde sıralanabilir (Jain ve diğ., 1999).

Bunlardan en popüler olanı bölüt merkezi ile temsil etmektir. Bölütler yoğun olduğunda iyi sonuç verir. Fakat bölütler uzatılmış ve yoğun değil ise uygun bir şekilde temsil mümkün olamaz. Bir bölütü temsil etmek üzere kullanılan nokta sayısı şeklin karmaşıklığı arttıkça artmalıdır. Veriler, karikatürize yüz çizimleri olarak da gösterilebilir (Everitt, 1993). Buradaki verilerin farklı boyutlarda sahip olduğu özellikler yüz üzerindeki farklı organlarla temsil edilmiştir. Veriler farklılaştıkça çizimler de farklılaşmaktadır.

5. YÜZ İFADELERİNİN ANALİZİ

5.1 Giriş

Sosyal iletişimde en önemli araçlardan biri olan yüz ifadelerinin (duygu durumsal göstergeler) analizi sosyal bilimlerin en önemli çalışma alanlarından birini oluşturmaktadır.

İnsan yüzü analizi, bilgi işlemsel olarak, bilgisayar mühendisliği ve özellikle görüntü işleme ve bilgisayar grafiği alanlarında da ilgi çekici bir konudur ve bu konuda çok sayıda çalışma yapılmaktadır.

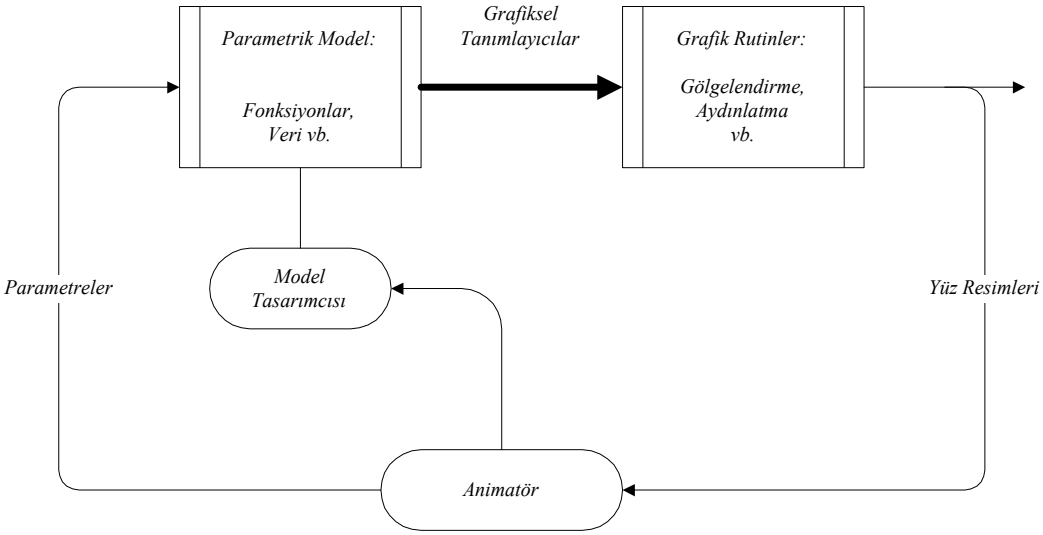
Yüz ifadeleri, hem insan psikolojisi hem de ifade yeteneği ile ilgili olduğundan bu konuda analiz yapmak için disiplinler arası bir çalışma yapmak gerekmektedir. Bu tezin kapsamında yapılan çalışma, bilgisayar mühendisliği açısından yüz ifadelerinin analizi için uygun ve etkin bir araç oluşturmaktır. Bundan önceki bölümlerde tanıtılan etmen sistemleri, etmenlerde öğrenme ve kümeleme konuları bu tez kapsamında incelenmiş ve bu yapılar uygun şekilde kullanılarak yüz ifadeleri analizi yapan bir sistem tasarlanmıştır.

Bu bölümde, yüz ifadelerinin bilgi işlemsel olarak oluşturulmasında yürütülen ilk çalışmalardan ve son dönemlerde yapılan çalışmalardan örnekler verilmektedir. Tez çalışması kapsamında kullanılan yüz animasyon programının işleyişi anlatılmaktadır.

5.2 Yüz İfadelerinin Parametrik Analizi

Bilgisayar ile yüz animasyonu yapılırken ilk zamanlarda izlenen yöntem, sıralı olarak yüz resimlerinin ekranda oluşturulmasıdır. İki boyutlu ve üç boyutlu olarak yüz animasyonu üretiminde iyi sonuçlar veren bu yöntemin en büyük dezavantajı, oluşturulmak istenen yüz animasyonunda her bir pencerenin önceden belirlenmesi gerekliliğidir. Yüz animasyonunun süresi uzadığında özellikle üç boyutlu olarak yüz animasyonu gerçeklemenin bilgi işlemsel gereksinimi de fazla olmaktadır. Bir diğer

yöntem olarak yüz ifadelerinin parametrik olarak analizi çalışmaları ilk olarak Parke tarafından ortaya atılmıştır (Parke, 1982). Bu yöntemle, yüzün parametrik modelinde parametre kümesindeki değerlerin uygun şekilde belirlenmesi ile üç boyutlu olarak yüz animasyonu yapılmaktadır. Böylelikle animasyon, istenen sırada istenen parametrelerin değiştirilmesi ile yapılabilmektedir. Dolayısıyla sadece parametreler üzerinde çalışmak yeterlidir. Bu yöntemin en önemli uygulama alanı, yüz animasyonları üretmek olsa da bir başka yan ürünü de görüntü aktarımı esnasında veri sıkıştırması uygulamasıdır. Görüntü aktarımı esnasında hem alıcı hem de verici tarafa yüzün parametrik modeli verildiğinde, sadece parametrelerin aktarımı ile verici taraftan yollanan görüntü alıcı tarafta aynen üretilebilmektedir. Bu da aktarım için harcanan veri yolu genişliği açısından oldukça büyük bir tasarruf sağlamaktadır.



Şekil 5.1 Yüz Animasyonu İçin Parametreleştirilmiş Resim Sentezi Modeli (Parke,1982)

Şekil 5.1'de parametrik yüz animasyonu yaratılması adımları görülmektedir. Parametrik modeli üretmek üzere kullanılan algoritmalar, fonksiyonlar ve verinin önemi büyüktür. Bu model, parametrik değerleri giriş olarak almaktadır ve çıktı olarak vektör ve poligon tanımlayıcıları gibi grafik ilkelleri üretmektedir. Tanımlayıcılar grafik rutinleri tarafından resimleri oluşturmak üzere kullanılmaktadır. Etkileşimli bir ortamda parametreler animatör ve model tasarımcısı tarafından değiştirilerek yüz animasyonu üretilmektedir. Model üretim süreci üretilen resimlerin kalitesini belirler. Bunlar, çizgi kesitlerinden üretilen vektör resimler olabileceği gibi karmaşık aydınlatma modellerinin kullanıldığı karesel veya kübik yüzeyler olabilir. En çok kullanılan yöntemde poligonsal yüzeyler üzerinde çalışılır.

İstenilen parametreleştirilmiş modelleri elde edebilmek için uygun parametrelerin üretilmesi ve bu parametrelere göre görüntü sentezi modellerinin oluşturulması gerekir.

En genel parametre kümesi, hem yüzün yapısal modelinin ve şeklinin oluşturulması için hem de ifade üretmek üzere etkin olanlarıdır. En uygun parametre kümesinin oluşturulması üzerine yapılan en kapsamlı çalışma Ekman' ın FACS (Facial Action Coding System) modelidir. FACS sisteminde yaklaşık olarak 50 bağımsız yüz hareketi tanımlanmıştır. Bunlar ayrık olarak veya kombinasyonları ile yüz ifadelerini oluşturmak üzere görev alırlar.

Parke (1982) makalesinde basit bir ifadede hangi parametrenin içerildiği problemini açık bir problem olarak tanımlamaktadır. İfadede en önemli bölümün gözler ve dudaklar olmasından dolayı bu bölümler en çok incelenen bölümler olmaktadır. Yaklaşık olarak 15 parametre veya daha fazlası ile bir yüz modeli elde etmek olasıdır.

Parke, görüntülerin gerçeklikten ne kadar uzak olursa kabul edilebilirlik oranının o ölçüde fazla olduğunu ve gerçeğe yakın görüntülerin daha çok eleştirilmeye açık olduğunu belirtmektedir. Dolayısıyla görüntüleri yapay ve sentetik yapmak kabul edilebilirliğini artırmaktadır.

5.3 FACS ile Yüz İfadeleri Analizi

FACS (Facial Action Coding System), Ekman ve Fiersen tarafından 1978 yılında geliştirilen ve insan yüz ifadelerini hareket birimleri (AU, Action Unit) ile açıklamaya yarayan bir sistemdir (Tian ve Cohn, 2001). 44 adet AU'da (12'si yüzün alt bölümü ve 18'i yüzün üst bölümünde olmak üzere 30'u anatomik olarak yüz kasları ile ilişkilendirilmiş) bir ya da birden fazla kas ve uygun aktivasyon seviyeleri belirlenmiş durumdadır. İfadede AU' lar tek başına bulunabilir veya bunların bir kombinasyonu şeklinde yer alırlar. AU' lar ifadedeki etkilerine göre katkılı (additive) veya katkısız (non-additive) olarak yer alırlar. Bu atomik birimlerin sayısının az olmasına rağmen 7000 farklı AU kombinasyonu gözlemlenebilir.

5.4 Örnek Bir Parametrik Yüz Animasyon Programı

Terzopoulos ve Waters tarafından FACS' ın daha az sayıda parametreli bir tipi olarak sentetik karakterlerin gerçekçi animasyonlarını üretmek üzere bir model sunulmuştur. Modele göre hazırlanan programda yüz modeli parametrik bir şekilde oluşturulmuştur (Terzopoulos ve Waters, 1993). İnsan yüzü anatomisinin, fiziğinin ve özellikle önemli yüz kaslarının davranışı ve düzenlemelerinin yüz görüntü analizi açısından önemli bir temel oluşturmasından yola çıkarak az sayıda parametre ile yüz modelini oluşturmuşlardır. Sundukları gerçekçi yüz modelinde yüzün hiyerarşik yapısı oluşturulmuştur. Hiyerarşik yapıda görüntü, geometri, fizik, kaslar, denetim ve ifade katmanlarını oluşturmuşlardır.

İfade: Parametreler ile denetlenebilen seviyede yüzün ifadelerine ilişkin bilgiler oluşturulabilmektedir. Programın orijinalinde belirlenmiş olan altı yüz ifadesine ilişkin parametre kümesinden seçim yapılabilir. Psikoloji ile bağlantılı olarak yüz ifadesi belirlenir.

Denetim: Kas denetim süreci, ifade komutlarını yüz modelinde ilgili kasları aktive etmek üzere denetimi gerçekler.

Kaslar: Gerçek yüzde olduğu gibi kaslar, modelin hareket mekanizmasının bileşenleridir. Kaslar birbirlerini etkileyecek şekilde birlikte hareket ederler.

Fizik: Yüz modeli insan yüzüne yakın bir gerçeklikte görünür. Deri modelinde lineer olmayan elastik yaylarla birbirine bağlı noktalar bulunmaktadır.

Geometri: Yüz modelinin geometrik temsili sabit olmayan (boyutları gerçek yüzdeki kıvrımlarla belirlenmiştir) çokgen yüzeyli elemanların oluşturduğu ağdan meydana gelir.

Görüntü: Görüntü üretme algoritmaları ile aydınlatılmış yüz ifadesi oluşturulur.

5.4.1 Grafiksel Yüz Animasyon Programı

Tez kapsamında kullanılan grafiksel yüz animasyon programı, Keith Waters tarafından OpenGL kütüphanesi kullanılarak C dilinde yazılmıştır (SimpleFace). Bu programda standart yüz ifadeleri (mutluluk, şaşırma, üzüntü, korku, iğrenme) için belirli parametreler önceden belirlenmiştir. Bu parametreler yüz kasları ile

ilişkilendirilmiş olduğundan parametrelerin değişimi yüz ifadesinin değişimini sağlar. Sadece parametreler üzerinde değişiklik yapılarak yüz ifadesi değiştirilmiş olur.

5.5 Grafiksel Yüz Animasyonu Uygulamaları

Yüz animasyonu uygulamaları, çoklu ortam, video, animasyon ve simülasyon filmlerinin üretimi, yapay karakterlerde yüz ifadelerinin belirlenmesi, yüz ifadelerinin analizi, yüz ifadelerinin yeniden oluşturulması ve video görüntüsünden yüz ifadelerinin grafiksel olarak oluşturulması gibi konularda oldukça geniş bir alana yayılmıştır. Özellikle yapay karakterlerin oluşturulmasında yüz ifadelerinin oluşturulması süreci en zahmetli bölüm olmaktadır. Animasyon yazılım ve uygulamaları hazırlayan araştırmacılar için yüz ifadelerinin belirlenmesi, üzerinde dikkatlice çalışılması gereken uzun bir süreçtir (SIGGRAPH 97). Maes' in bir çalışmasında poker oynayan etmenler bulunmakta ve her biri farklı görüntülerde ekranda görünmektedir (Koda ve Maes, 1996). Poker oyununda oyuncuların ellerindeki kartların niteliğini gizleyebilmesi açısından yüz ifadelerinin önemi oldukça fazladır. Bu oyun için üretilen yapay karakterler, karikatürize bir bayan, yapay bir köpek ve basit çizgilerle çizilen bir yüze sahip başka bir karakterden oluşmaktadır. Bu karakterler ekrana video görüntüsü yansıyan oyuncu ile karşı karşıyadır. Yapay karakterler belirli durumlar için önceden tanımlanmış görüntüsel ifadeler oluşturabilmektedir. Bu çalışmada kullanıcılar üzerinde yapılan ankete göre akıllı görünüm, beğenilebilirlik gibi ölçütler için gerçek insan figürünün tüm koşullarda en iyi sonucu verdiği, bunun yanında yapay köpek karakterinin karikatür bayan karakterine oranla beğenilirliğinin ve akıllı görünümünün daha fazla olduğu sonucu çıkmıştır. Bu da gerçeğe yakın bir yüz ifadesine daha eleştirel yaklaşılması sonucunu oluşturmuştur.

Yüz ifadesi üretmek üzere yapılan çalışmalar yazılımsal ve donanımsal olarak yürütülmektedir. Özellikle MIT Media ve AI laboratuarlarında yüz ifadelerine sahip robotlar üzerinde çalışmalar yapılmaktadır (Kismet). Viseme olarak adlandırılan ve farklı dudak hareketlerinin görüntülerinin uygun şekilde birleştirilmesi ile konuşan yüz animasyonları elde edilmektedir (Ezzat ve Poggio, 1998).

Bu çalışmalar, henüz statik olarak yapılmakta ve ifadeyi doğrudan anlayarak buna ilişkin yüz ifadesini üreten, haber veya masal okuyan karakterler için yapay zeka, psikoloji, felsefe ve dilbilimsel disiplinlerde uzman araştırmacılar ortak çalışma yürütmektedir.

6. JAVA VE SÜREÇLER ARASI İLETİŞİM

6.1 Giriş

Tez kapsamında gerçeklenen yazılım, Java Programlama dili kullanılarak gerçeklenmiştir. Bu bölümde, Java programlama dilinin genel özelliklerine ve çok görevcikli çalışma, süreçler arası iletişim için soket kullanımı türünden sunduğu hizmetlere kısaca değinilmiştir.

6.2 Java Programlama Dili ve Özellikleri

Sun Microsystems tarafından geliştirilen Java dili nesneye yönelik bir programlama dilidir. Bu programlama dili, hem tasarımcı tarafından oluşturulan sınıfların (class) tasarlanması hem de mevcut sınıf kütüphanelerinden faydalanılmasına olanak tanır.

6.2.1 Taşınabilirlik ve Platformdan Bağımsız Olma

Java programlama dilinde yazılan sınıflar Java Sanal Makinesi (JVM, Java Virtual Machine) olan tüm ortamlarda yorumlanabilir. Bu özellik de Java' da yazılmış uygulama veya apletlerin tüm ortamlarda yorumlanması ve yürütülebilmesini sağlar. Dolayısıyla Java' da yazılan uygulamalar taşınabilir özelliğe sahiptir (İşletim sistemlerinden kaynaklanan farklıklar sonucu ortamlar arası farklı yorumlamaların oluştuğu göz önüne alınmalıdır). Java uygulama ve apletlerinin taşınabilmesi için JVM tarafından yorumlanabilen ve donanım mimarisi ve işletim sistemi arayüzünden bağımsız olan yüksek seviyeli sekizli kodlara (bytecode) dönüştürülmesi gerekmektedir.

6.2.2 Nesneye Yönelik Tasarım

Javanın tamamıyla nesneye dayalı bir dil olmasından dolayı oluşturulan tüm programlar da birer sınıf olmaktadır ve sınırlamalara bağlı olarak metotları diğer sınıflar tarafından yürütülebilmektedir. Java programlarının bu modüler yapısı hem

mevcut kütüphane sınıflarının hem de tasarımcı tarafından oluşturulan sınıfların etkin şekilde entegrasyonunu sağlar.

6.2.3 Çok Görevcikli Çalışma

Yazılım sistemlerinde çoklu ve paralel çalışmaya imkan tanıyan çok görevcikli çalışma, Java programlama dili tarafından desteklenmektedir. Bu destek, paralel çalışmada karşılaşılabilecek önemli problemlerin üstesinden gelebilecek şekilde verilmektedir. Sistemi kaynak paylaşım ihlallerinden korumak üzere görevciklerin senkronize bir şekilde ve zamanda paylaşımlı olarak çalışabilmesine imkan verilir. Bir anda tek bir görevciğin belli bir metodu yürütmesi ve diğer görevciklerin bu metodun yürütülmesi aşamasında gerekiyorsa beklemesi sağlanır. Java kütüphanesi *Thread* sınıfı ile görevcik çalışmasını başlatma, yürütme, bekletme, askıya alma, çalışmasına devam etme, durdurma ve görevciğin durumunu denetleme hizmetlerini verir.

6.2.4 Diğer Özellikler

RMI Uzaktan Yordam Çağırma (Remote Method Invocation) hizmeti, dağıtılmış Java uygulamalarının geliştirilerek uzaktaki Java nesnelerinin farklı konaklardaki JVM' ler tarafından çağrılabilmesini sağlar. İstekçi ve hizmetli yazılımlar arasında nesne taşınması, RMI ile hareketli kodlar üretilerek mümkündür.

Veritabanı bağlantılılığı (JDBC, Java Database Connectivity) özelliği uygulamaların, herhangi bir veritabanına bağlanması, bu veritabanında SQL (Standart Query Language) dilini kullanarak sorgulama yapması ve güncellemesine imkan tanır.

Java, dağıtılmış veya ağ uygulamalarında güvenlik mekanizması kullanılmasına imkan tanır. Programın JVM tarafından yorumlanmasından önce sekizli kodun tanımlanan güvenlik kurallarına uyup uymadığı kontrol edilir. Onaylama yöntemleri açık-anahtar şifrelemesine dayanır. Tasarımcı tarafından ek güvenlik mekanizmaları eklenebilir.

6.3 Java' da Süreçler Arası İletişim

Java, süreçler arasında soket-tabanlı iletişim modeline imkan tanır. Uygulamalar, soketlere dosya erişimleri gibi erişerek yazma ve soketlerden okuma yapabilir. Soketler kullanılarak üst seviyede haberleşme ortamı sağlanmış olur. Java, tasarımcının stream ve datagram soketlerini kullanmasına imkan tanır. Stream soketi ile TCP protokolü kullanılarak bağlantılı iletişim söz konusudur. Datagram soketi ile UDP protokolü kullanılarak bağlantısız iletişim sağlanır. Dolayısıyla Java programlama dili etkin hizmetli-istekçi programlarının yazılması için imkan tanır. Hem istekçi hem de hizmetli için ayrı soket sınıfları tanımlanmıştır ("ServerSocket" ve "Socket" sınıfları). Soket sınıfları ve metotları "java.net" pakedi içinde tanımlanmıştır (Java/Net).

6.3.1 İletişim İçin Tanımlanmış Sınıflar

"Stream" Soket tabanlı iletişimde hizmetli ve istekçinin yürütmesi gereken adımlar kendi sınıflarında tanımlanmıştır. Her iki birim de iletişim kuracakları bilgisayar adresi ve port numarasını belirlemek zorundadır. Bu tanımlamalar yapıldıktan sonra soketler üzerinden etkin veri aktarımı sağlanır. Hizmetli her an istekçiden gelen isteklere yanıt verebilmek üzere soketleri dinler ve istek geldikçe hizmetini sunar. Hizmetli birden fazla isteğe öncelikli ya da önceliksiz olarak hizmet vermek üzere çok görevcikli bir yapı içinde soket iletişimini gerçekleyebilir.

6.3.1.1 Hizmetli Soketi

"ServerSocket" sınıfı etkin hizmetli süreçlerinin oluşturulmasında kullanılan sınıftır. Farklı parametrelerle hizmetli soketinin oluşturulması için farklı kurucu metot tanımlamaları mevcuttur. Tanımlanmış "ServerSocket", kurucuları ile oluşturulduktan sonra ilgili porta istek gelmesini bekler. Portlara dosyaya yazma veya dosyadan okuma yöntemleri ile erişebilmek için "DataInputStream" veya "DataOutputStream" nesnelerinin yaratılması gerekir. Çoklu istekçi hizmeti için portlara ilişkin çok görevcikli bir yapı oluşturmak etkin bir çözüm olmaktadır.

6.3.1.2 İstekçi Soketi

"Soket" sınıfı istekçi süreçlerin oluşturulmasında kullanılan sınıftır. Hizmetli soketinde olduğu gibi farklı kurucular tanımlanmıştır. Soketin bağlantı kuracağı adres ve port numarası bilinmelidir. Tanımlanmış "Socket", kurucuları ile

oluşturulduktan sonra her hangi bir anda istek yaratabilir. Portlara dosyaya yazma veya dosyadan okuma yöntemleri ile erişebilmek için “DataInputStream” veya “DataOutputStream” nesnelerinin yaratılması gerekir.

Hizmetli ve istekçi soketlerinin oluşturulması ve ilgili sınıf metotları ile soketlere yazma ve soketlerden okuma yapılması, hizmetli ve istekçi süreçlerin iletişimi için yeterli değildir. Uygulama geliştirilirken süreçlerin etkin iletişimi için ve aykırı durumların oluşmasını engellemek üzere uygun protokolün tasarlanması gerekmektedir.

7. YÜZ İFADELERİNİ ANALİZ EDEN BİR SİSTEMİN TASARIMI

7.1 Giriş

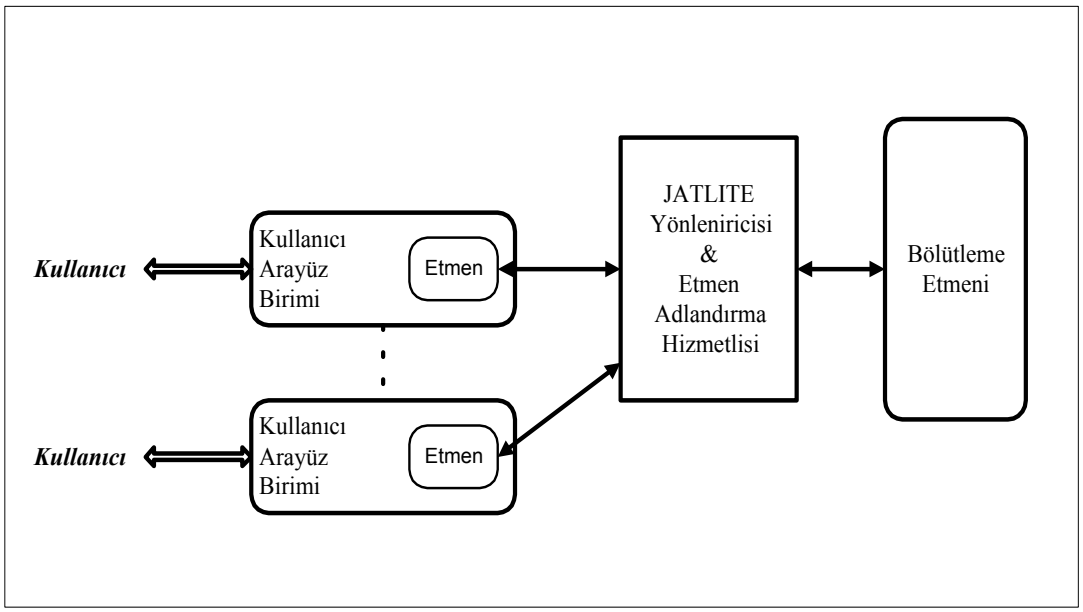
Yürütülen tez çalışmasında, yüz ifadelerini analizi eden bir sistem gerçeklenmiştir. Yüz ifadelerinin analizi için kullanıcı kesitleri değerlendirilmektedir. Sistem herhangi bir uzman girişine gereksinmeyecek şekilde tasarlanmıştır. Kullanıcıların oluşturmuş oldukları ifadelere göre, veriler üzerindeki yapılar incelenmektedir. Yüz ifadelerinin analizi hem çeşitli sözel ifadelere karşılık olarak yüz ifade bölütlerinin belirlenmesi hem de hangi yüz parametresinin hangi ifadede ne kadar etkin rol oynadığı belirlenerek yapılmaktadır. Bu analiz için önceki bölümlerde anlatılan yapılar sistematik şekilde bir araya getirilmiş ve uygulamanın kurulan çatısı içinde etkin şekilde rol almıştır.

Tez konusu çoklu etmen mimarisi kullanılarak tasarlanmıştır. Hazırlanmış olan tüm programlar Java dilinde yazılmıştır. Etmen oluşturma aracı olarak seçilen JATLite, Java dilinde hazırlanmış bir araç olup, tasarımcının uygulamasını oluşturan etmenlerin entegre edilebileceği bir ortam sunar. Gerçeklenen sistem içinde, verilen parametrelere göre yüz ifadelerini oluşturan hazır bir Grafiksel Yüz Animasyon Programı kullanılmıştır.

Sitemdeki etmenler, kullanıcıları ile tasarlanmış olan kullanıcı arayüzü aracılığı ile ve diğer kullanıcılar ile JATLite' ın sunduğu KQML katmanı üzerinden KQML mesajları ile haberleşir. Her kullanıcı, kendisine özgü yüz ifadelerini kendisine atanan arayüzü kullanarak oluşturur. Daha sonra kullanıcıya ait bir kesit belirlemek üzere diğer kullanıcıların çizmiş olduğu yüz ifadelerine kullanıcının puan vermesi de sağlanarak etmen inançları oluşturulur. Sistemdeki tüm etmenlerin verilerini toplayarak analiz eden birim olan Bölütleme Etmeni, verideki bölütleri belirleyerek toplumun genel kesitini oluşturur. Böylelikle sistem, yerel etmen inançları ile toplum kesit bilgisinin karşılaştırılmasına imkan tanır. Bölütleme Etmeni, bölütleme sürecini etkin olarak gerçekleyebilmek için yüz ifadelerinde etkili olan parametrelerin ağırlıklarını öğrenir. Bu öğrenme sürecinde sistemdeki tüm kullanıcılar etkin rol oynar.

7.2 Sistem Mimarisi

Sistemde kullanıcı ile iletişim halinde bulunan Kullanıcı Arayüz Birimleri, tüm verileri analiz eden Bölütleme Etmeni ve etmenler arası işlemleri gerçekleyen JATLite Yönlendiricisi ve bileşenleri bulunur. Şekil 7.1'de tüm sistem bileşenleri görülmektedir. Sistemde gerçeklenmek istenen amaç, belirli ifadeler için kullanıcılar tarafından çizilen çeşitli yüz ifadelerine ilişkin bilgileri analiz ederek toplum için genel olan ifadeleri ve tek tek etmenlerde oluşturulan ifade bölütlerini (etmen inançları) belirlemek ve bu sonuçlarının karşılaştırılmasını sağlamaktır. Bu analiz için sistem bileşenlerinin çoklu bir kullanıcı ortamında çalıştırılması gerekmektedir.



Şekil 7.1 Çoklu Etmen Sistemi Bileşenleri

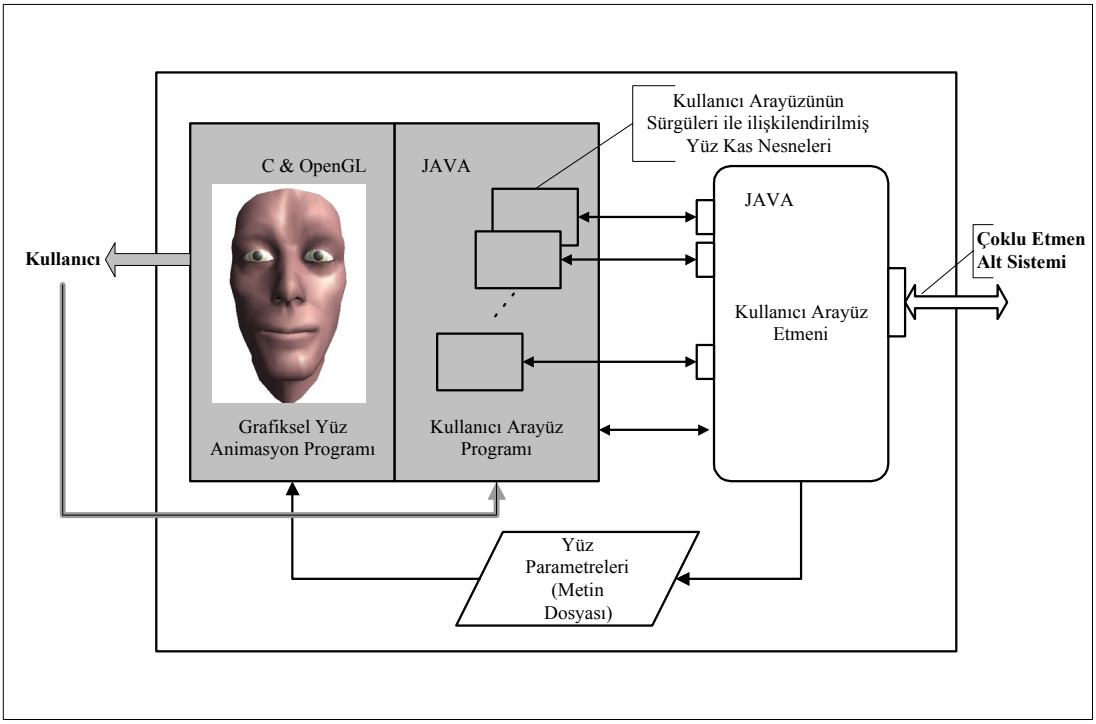
7.2.1 Kullanıcı Arayüz Birimi

Kullanıcı Arayüz Birimi, kullanıcı ile haberleşen Kullanıcı Arayüz Programı, Grafiksel Yüz Animasyon Programı ve diğer etmenlerle haberleşerek sistemde kullanıcıyı temsil eden Kullanıcı Arayüz Etmenini içerir. Şekil 7.2'de Kullanıcı Arayüz Birimi görülmektedir.

Kullanıcı Arayüz biriminin tasarlanan bölümü Java dilinde yazılmıştır. Grafiksel Yüz Animasyon Programı ise C dilinde ve OpenGL kütüphanesi kullanılarak hazırlanmış bir programdır. İki programın birbirleri ile etkileşimli çalışması okuma/yazma yapabildikleri bir metin dosyası üzerinden olmaktadır. Dosyaya yazma işlemini gerçekleyen kısım, kullanıcının istekleri doğrultusunda yüz kasları için gerekli olan parametre değerlerini belirleyen Kullanıcı Arayüz Birimidir. Bu birim, yüz ifadesine

ilişkin olarak kullanıcıdan istek geldikçe yüz parametrelerini içeren dosyayı günceller. Kullanıcıya, yaptığı değişikliklerin her bir adımda yüz ifadesi olarak gösterilebilesi için Grafiksel Yüz Animasyon programının belirli zaman adımlarında değişiklik olup olmadığını denetlemesi gerekmektedir.

Etmenlerin birbirleri ile veri alış-verişi yapmaları, JATLite aracının KQML katmanı kullanılarak gerçeklenir. Kullanıcı Arayüz Etmenleri sistemdeki tüm etmenlerle haberleşebilir.



Şekil 7.2 Kullanıcı Arayüz Birimi

7.2.1.1 Grafiksel Yüz Animasyon Programı

Grafiksel Yüz Animasyon Programı, çeşitli parametrelerle kontrol edilebilen yüz ifadelerini çizebilmek için kullanılabilecek grafiksel bir programdır. OpenGL kütüphanesi kullanılarak C dilinde K. Waters tarafından hazırlanmıştır (SimpleFace). Program, yüz kaslarına ilişkin bilgileri "expression-macros.dat" dosyasından okuyabilme özelliğine sahiptir. Bu dosyayı okuyarak tüm yüz kaslarını ilgili parametre değerlerine göre hareket ettirir. Bu programın, gerçeklenen sistemde etkin olarak kullanımını sağlamak üzere belirli aralıklarla ilgili dosyayı okuyarak güncelleme yapması sağlanmıştır. Programın yüz dokusunu kullanan sürümü AUX kütüphanesi ile hazırlanmıştır. Tez kapsamında bu program, belirli adımlarda okuma işlemini gerçeklemek üzere GLUT kütüphanesinde çalışır hale getirilmiştir. Program,

GLUT kütüphanesinde tanımlı “TimerFunction(int value)” fonksiyonunu kullanarak belirli aralıklarda parametrelerin bulunduğu dosyada güncelleme olup olmadığını test eder. Güncellemenin yapıldığını belli bir dosyanın yaratılmış (semaphor.txt) olması ile anlar. Dolayısıyla istenilen ifadeleri göstermek üzere Grafiksel Yüz Animasyon Programını kullanacak olan bir programın güncellemeyi gerçekleyip "semaphor.txt" dosyasını oluşturması gerekmektedir. Grafiksel Yüz Animasyon Programı istenilen parametrelerle yüz ifadesini oluşturduktan sonra bu dosyayı yeni güncellemeleri algılamak üzere siler.

Veri dosyasında kullanıcıdan alınması gerekli olan parametre bilgileri yer alır. Tüm parametrelerin sıfır olduğu durum Şekil 7.3’te verilmiştir. Bu parametreler sırasıyla, dudak, dudak büzme, kaş (iç), kaş (orta), kaş (dış), dudak ve burun (birlikte), dudak ve burun (iç), kaş bitiştirme/ayrıştırma, göz ve burun (birlikte) kaslarını (sol ve sağ) kontrol eder. Bu parametreleri değiştirmek ve güncellemenin yapıldığını bildirmek suretiyle, Grafiksel Yüz Animasyon Programının istenilen parametrelere ilişkin yüz ifadelerini çizmesi sağlanır. Sadece tek bir parametrenin değişmesi istendiğinde diğer parametrelerin sabit kalması gerekmektedir. Bunu Grafiksel Yüz Animasyon Programını kullanan uygulama programı sağlamalıdır. Grafiksel Yüz Animasyon Programı kaynak kodları açık olduğundan yüzde istenilen değişiklikler gerçeklenebilir. Örneğin ağız açma ve göz bebeğini hareket ettirme işlemleri ücretsiz elde edilebilen bu sürüme eklenmemiştir. Grafiksel Arayüz programında doku kaplaması ile farklı kullanıcı grupları için farklı yüzler elde edilebilir.

```
Left_Zygomatic_Major 0.00  Right_Zygomatic_Major 0.00
Left_Angular_Depressor 0.00  Right_Angular_Depressor 0.00
Left_Frontalis_Inner 0.00  Right_Frontalis_Inner 0.00
Left_Frontalis_Major 0.00  Right_Frontalis_Major 0.00
Left_Frontalis_Outer 0.00  Right_Frontalis_Outer 0.00
Left_Labi_Nasi 0.00  Right_Labi_Nasi 0.00
Left_Inner_Labi_Nasi 0.00  Right_Inner_Labi_Nasi 0.00
Left_Lateral_Corigator 0.00  Right_Lateral_Corigator 0.00
Left_Secondary_Frontalis 0.00  Right_Secondary_Frontalis 0.00
```

Şekil 7.3 “Expression-macros.dat” dosyası içeriği

7.2.1.2 Kullanıcı Arayüz Programı

Kullanıcı Arayüz Programı, kullanıcı ile etkileşim halinde olan sistem bileşenidir. Kullanıcı, arayüzdeki sürgüleri kullanarak çizimini gerçekleyebilir. Sürgüler yüzdeki her bir kas ile ilişkilendirilmiş durumdadır. Kullanıcı, diğer kullanıcıların çizimleri için puan girişini de arayüz programının penceresinden yapar. Kullanıcı Arayüz Programı, parametrelerde yapılması istenilen değişikliği Kullanıcı Arayüz Etmenine bildirir.

Arayüzdeki sürgülerin herbiri için bir reaktif nesne tanımı yapılmıştır. Reaktif nesneler kullanıcıdan aldıkları komutlara göre ortam üzerinde değişiklik yapabilen akıllı olmayan basit nesnelerdir. Görevleri kendileri ile ilişkili parametre üzerinde değişiklik yapmaktır. Bu reaktif nesneler ortamları olan yüz üzerinde kendilerine düşen görevi yerine getirmekte ve ilgili oldukları yüz parametresini istenen değere getirmektedirler. Kullanıcı, yaptığı her değişikliğin sonucunu Grafiksel Yüz Animasyon Programı penceresinde görebilir ve böylelikle çizmek istediği yüz ifadesini oluşturur. Her bir yüz parametresinin ayrı bir nesne olarak modellenmesinin sebebi ileri bir uygulamada bir yüz ifadesini oluşturan parametrelerin bilgisayarlar arasında dağıtımını yapmaktır. Ayrıca tüm bu reaktif nesneler farklı kullanıcılar tarafından da kontrol edilebilir. Reaktif nesneler ortamda değişiklik yapmak üzere kendi isteklerini Kullanıcı Arayüz Biriminin önemli bir parçası olan ve Çoklu Etmen Sistemi içinde akıllı bir etmen olarak görev yapan Kullanıcı Arayüz Etmenine yollar. Kullanıcı Arayüz Etmeni, kendisine atanmış tüm reaktif nesneler için bir güncelleme hizmetlisi olarak görev yapar.

Kullanıcı Arayüz Programı etkinleştiğinde kullanıcıya ait olan rasgele erişilebilir kullanıcı kesit dosyası (“UserProfile.dat”) açılır. Kullanıcının her bir soru için oluşturduğu parametre bilgileri burada saklanır. Bu dosyaya, kullanıcı kesit bilgilerinin diğer etmenler ve Bölütleme Etmenine yollanması için kayıt tabanlı rasgele erişim mümkündür. Kullanıcı Kesitinin standart şekilde kaydedilmesi için oluşturulan kesit, rasgele erişilebilir dosya kayıtları yapısı “UserProfile” sınıfı içinde tutulur. Bu sınıfa ilişkin ayrıntılar VIII. Bölümde bulunabilir. Bu dosyada sadece tek bir kayıt içindeki ilgili veriye ilişkin olarak, kaydı seçmek üzere dosya işaretçisi, dosya içindeki uygun konuma (kayıt boyutu veri tipleri uzunluğu toplamıdır.) getirildikten sonra get/set işlemleri ardından read/write ile dosya işlemleri gerçeklenir.

7.2.1.3 Reaktif Nesneler

Reaktif nesneler, kendileri için tanımlanmış olan sınıf (AgentFrame) yapısına uygun olarak ilgili port bilgisi ile yaratılırlar. Her bir parametre için bir adet olmak üzere 18 reaktif nesne yaratılır. Her bir reaktif nesne kullanıcı arayüz programındaki ilgili sürgüden sorumludur.

Reaktif nesneler kendileri için ayrılan port üzerinden Kullanıcı Arayüz Etmeni ile soket haberleşmesi yapmaktadır. Her bir reaktif nesne gerekirse farklı bir makinede olacak şekilde Kullanıcı Arayüz Etmeni ile haberleşebilir. Reaktif nesneler ile Kullanıcı Arayüz Etmeni farklı makinede olduğunda yerel adres yerine ilgili makinenin IP adresi bilinmelidir.

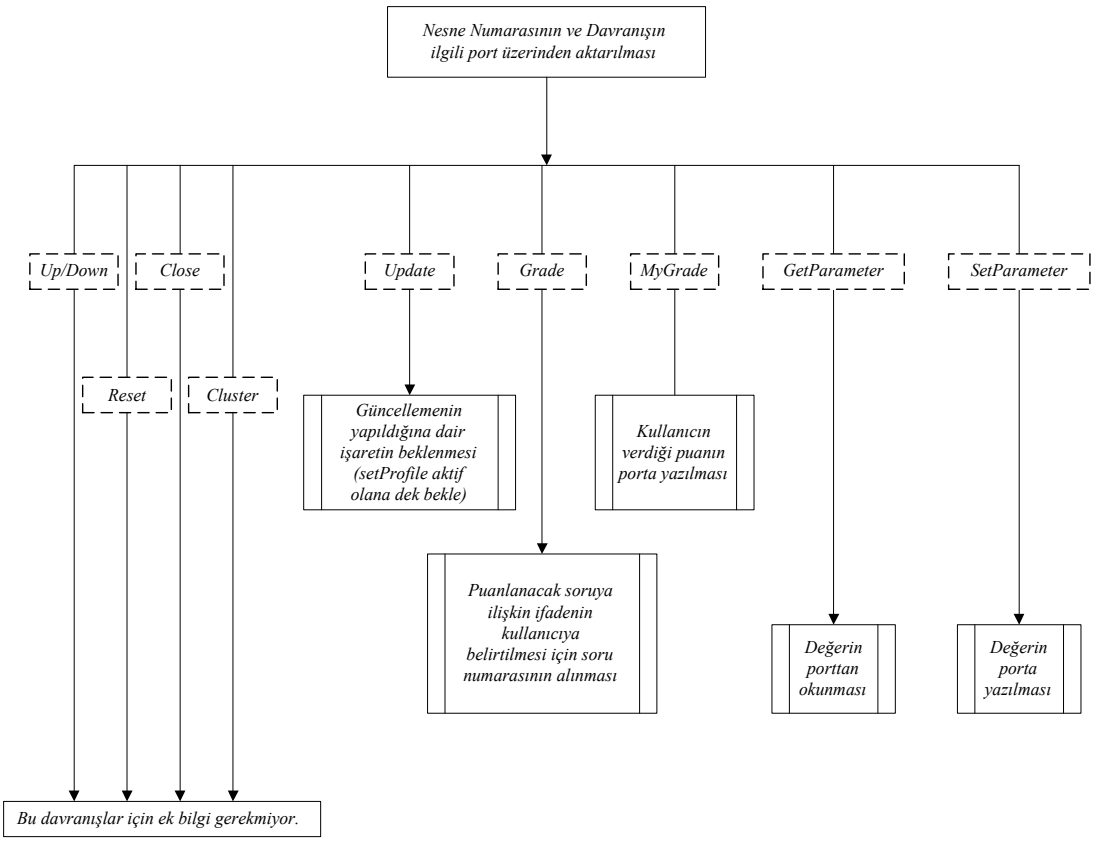
Reaktif nesneler yürüttükleri davranışa göre Kullanıcı Arayüz Etmeni ile belli bir protokole göre haberleşmektedir. Kullanıcı Arayüz Etmenine yollanan ilk bilgi gerçeklenen davranıştır. Davranışlar ve açıklamaları Tablo 7.1'de bulunabilir.

Tablo 7.1 Reaktif Etmenler İçin Tanımlanmış Davranışlar

Davranış	**Açıklaması**
Up / Down	İlgili parametrenin üzerinde yapılmak istenen davranışı tanımlar.
Reset	Parametre bilgisini sıfırlar.
Update	Kullanıcı çiziminin kaydedilmesini sağlar.
Grade	Başka kullanıcıların yüz ifadelerini puanlama isteğini belirtir.
MyGrade	Diğer kullanıcıların çizimlerine verilen puanı yollar.
SetParameter	İlgili parametreyi istenen değere göre günceller .
GetParameter	İlgili parametreye ilişkin değerin yollanması için istekte bulunur.
Cluster	Kesitin güncellenmesi amacıyla bölütleme isteğinde bulunur.
Close	Kullanıcı Arayüz Etmeni ile soket bağlantısının kesilmesini sağlar.

Kullanıcı Arayüz Programında Up, Down, Reset, Close, Cluster davranışları bilgiyi davranış içinde içerdiklerinden, Kullanıcı Arayüz Etmeni ek bilgi alış-verişi gerektirmez. "Update" davranışı Kullanıcı Arayüz Etmeninin ilgili soruya ilişkin çizilen yüz ifadesinin parametre bilgilerini kullanıcı kesit dosyasına (UserProfile.dat) kaydetmesini sağlar. Reaktif nesneler işlemlerine devam edebilmek için bu güncellemenin sonlanmasını bekler. Güncellemeden sonra kullanıcıya yüz şekli çizdirilmek istenen yeni ifade (kullanıcıya sorulan sorular) ekranda belirir. Yeni ifadeye ilişkin yüz şeklinin çizilebilmesinde kullanıcıya kolaylık sağlamak üzere yüz

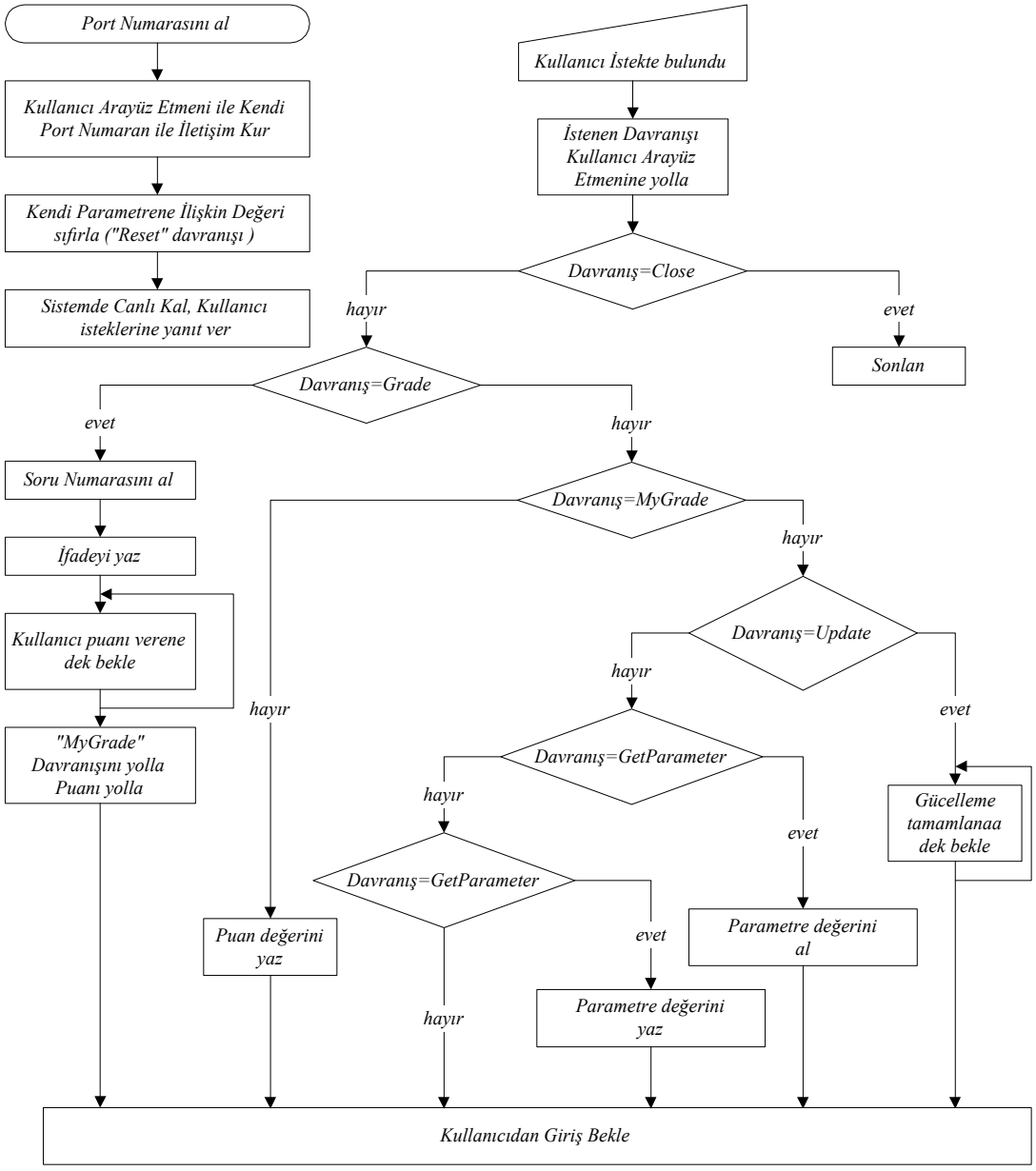
şeklinin nötr hale gelmesi için tüm reaktif nesneler Kullanıcı Arayüz Etmenine "Reset" davranışını yollarlar. Reaktif nesneler belirli adımlarda Kullanıcı Arayüz Etmeninden kendi parametrelerine ilişkin değerleri "GetParameter" davranışı ile öğrenirler ve "SetParameter" ile parametreyi istenen değere getirirler. Şekil 7.4'de Reaktif nesnelere ilişkin davranışlar için belirlenen protokol gösterilmektedir.



Şekil 7.4 Reaktif Nesnelerin Tüm Davranışları İçin Kullanıcı Arayüz Etmeni İle Önceden Belirledikleri İletişim Protokolü

Reaktif nesneler, yeni ifadenin oluşturulması aşamasında kullanıcıdan gelen her isteği "Up /Down" davranışı ile iletirler. Kullanıcı istediği anda diğer kullanıcıların çizmiş oldukları yüz ifadelerini değerlendirebilecek durumdadır. Bu isteği karşılamak üzere nesnelerden herhangi biri "Grade" davranışını yollar. Bu isteği sonucunda Kullanıcı Arayüz Etmeni, Grafiksel Yüz Animasyon Programının diğer kullanıcının çizimini göstermesi için gerekli olan güncellemeyi yapar ve ilgili soru numarasını reaktif nesneye geri döndürür. Bu soru numarasını alan etmen ilgili ifadeyi (soruyu) kullanıcısına göstererek soru ile ilişkili yüz resminin anlaşılmasını sağlar. Nesne, kullanıcısının bu ifadeye ilişkin çizim için vermiş olduğu puanı "MyGrade" davranışı ile Kullanıcı Arayüz Etmenine yollar. Böylelikle puanlama için kendisine düşen görev tamamlanmış olur. Eğer diğer kullanıcılardan herhangi bir bilgi gelmemişse

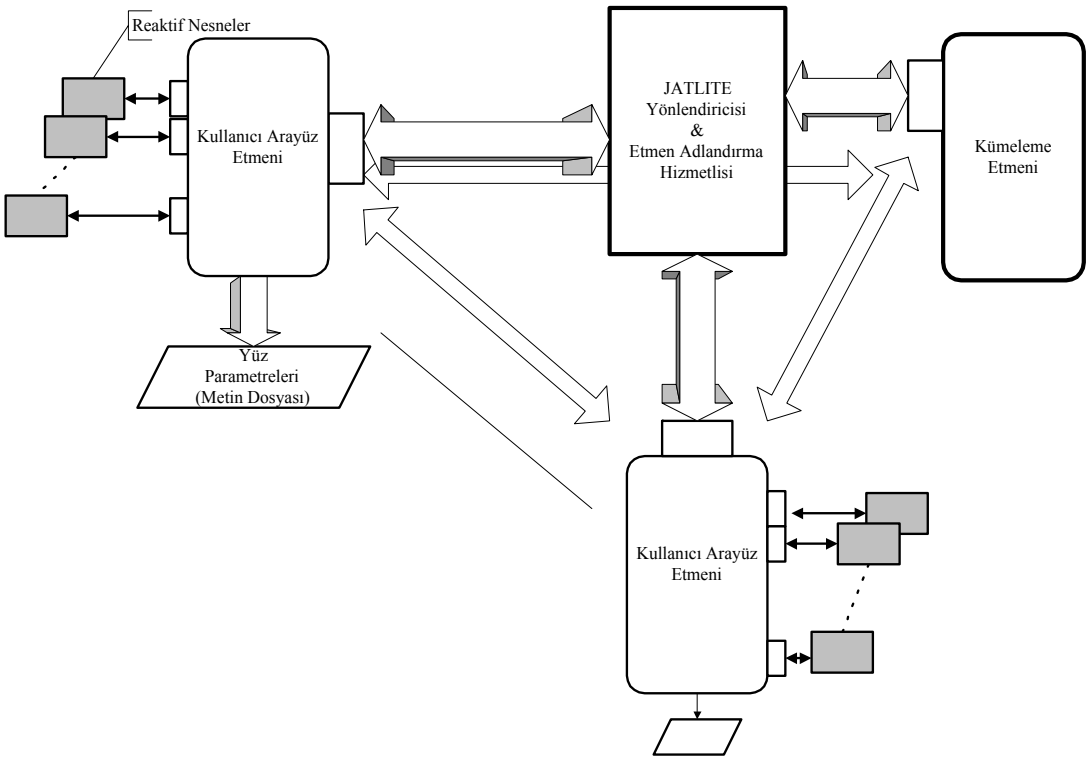
kullanıcıya mesaj verilerek diğer işlemlere devam etmesi sağlanır. Kullanıcı, kendisine gelen yüz ifadelerini puanlandırdıktan sonra kendi kullanıcı kesitini oluşturmak üzere bölütleme sürecini başlatabilir. Bu işlem için Kullanıcı Arayüz Etmenine "Cluster" davranışı yollanır. Reaktif nesneler program sonlandırılacağı zaman Kullanıcı Arayüz Etmeni ile bağlantının düzgün şekilde sonlanması için "Close" davranışını yollar. Şekil 7.5'de reaktif nesnelerin yaşam çevrimi görülmektedir.



Şekil 7.5 Reaktif Nesne Yaşam Çevrimi

7.2.1.4 Kullanıcı Arayüz Etmeni

Kullanıcı Arayüz Etmeni, Kullanıcı Arayüz Biriminin en önemli bileşenidir. Bu etmen, çoklu etmen sistemindeki akıllı etmenlerden biridir. Kullanıcı kesitini oluşturma, diğer etmenlerle haberleşme, kendi reaktif nesnelerine hizmet etme ve bölütleme yeteneklerine sahiptir.



Şekil 7.6 Kullanıcı Arayüz Etmenleri, Bölütleme Etmeni ve JATLite Yönlendiricisi Arasındaki İletişim Altyapısı

Şekil 7.6'da Kullanıcı Arayüz Etmeni ve sistem birimleri arasındaki iletişim alt yapısı görülmektedir. Kullanıcı Arayüz Etmeni, kullanıcı kesitini daha önceden hazırlanan sorulara ilişkin olarak kullanıcısının çizmiş olduğu yüz ifadelerinin parametreleri ve kullanıcısının diğer kullanıcıların çizimlerine verdiği puan bilgilerini kullanarak oluşturur. Kendi kullanıcısının çizdiği yüz ifadelerinin parametrelerini saklayabilmek için sürgülere iliştirilmiş reaktif nesnelerle iletişim kurarak ve Grafiksel Yüz Animasyon Programında gerekli güncellemeler yaparak bir dizi işlem yapar. Reaktif nesneler için bir güncelleme hizmetlisi olarak çalışır. Bunun için her bir reaktif nesne için ayrı bir port açarak, bu portlar üzerinden bağlantılı soket haberleşmesi ile iletişim kurar. Kullanılan port adresleri makine adresinden bağımsız olarak [4001-4018] olarak belirlenmiştir. Bu portlar için hizmet veren Kullanıcı Arayüz Etmeni portlar için çalışan görevcikleri (thread) için çok görevcikli çalışmada karşılaşılan

problemleri kotarabilecek yapıda olmalıdır. Kısa aralıklarla birden fazla porttan gelen güncelleme isteğine karşın, güncellenen tek dosya olduğundan dolayı paylaşım ihlallerinin oluşmasını engellemek üzere ve tutarlı bir sistem oluşturabilmek için kritik bölgelerde kısıtlama oluşturmak üzere semafor yapılarının kullanılması gerekmiştir.

Kullanıcı Arayüz Etmeni, özkaynağını (güncellediği dosya) görevcikler arasında etkin şekilde paylaştırarak ilgili reaktif nesnelerden gelen davranış isteklerine yanıt verir ve gerekli hizmeti yürütür.

Kullanıcı kesiti oluşturulduktan sonra bu bilgiyi sistemin tüm verilerini analiz eden Bölütleme Etmenine (çoklu etmen sistemi altyapısını kullanarak) KQML mesajları ile yollar. Bundan sonra kendi yüz ifadesi bilgilerini diğer kullanıcı arayüz etmenleri ile paylaşabilecek duruma gelmiştir. Herhangi bir anda reaktif nesnelerden gelen "Grade" isteğine karşılık olarak kendisine diğer etmenlerden gelen mesaj bilgilerini kullanıcısına göstermelidir. Sistemde dolaşan bilgiler, sorulara ilişkin olarak belirlenmiş yüz parametre bilgileridir. Bu parametre bilgilerini alarak kullanıcıya yüz ifadeleri olarak gösterebilmek için "expression-macros.dat" ve "semaphor.txt" güncellemelerini yapar. Kullanıcı, ilgili yüz ifadesi için bu çizime ilişkin olarak bir puan verir. Bu şekilde puanlamalar ile kullanıcı kesitinin oluşturulması için gerekli olan bilgiler de toplanmış olmaktadır.

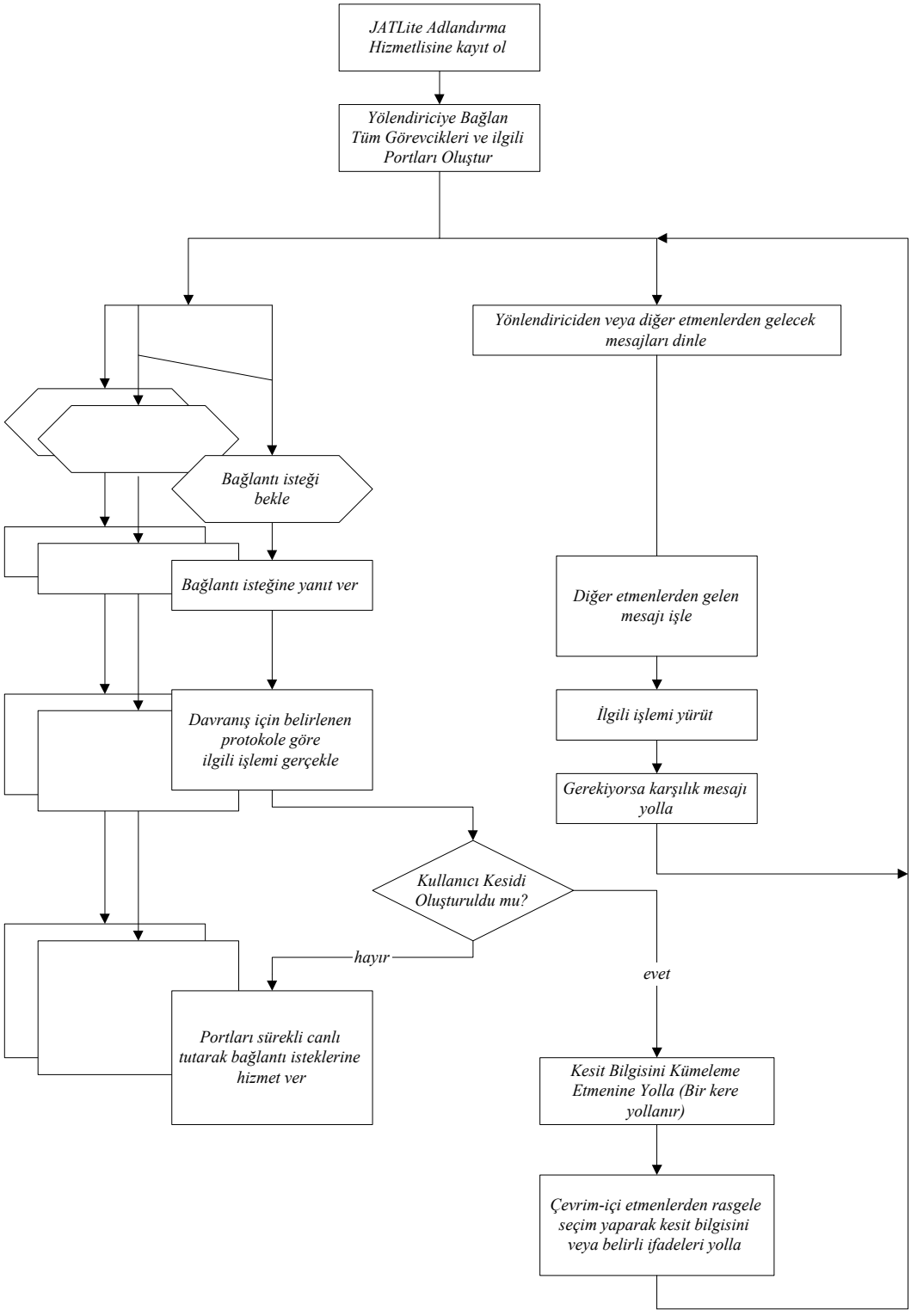
7.2.1.5 Kullanıcı Arayüz Etmeninin Çoklu Etmen Sistemi İçindeki Rolü

Her bir kullanıcı etmeni çoklu etmen sistemine dahil olabilmek ve diğer etmenlerle etkileşim kurabilmek için JATLite Yönlendiricisi ve Adlandırma Hizmetlisi' ne kaydolmalıdır. Kullanıcıdan istenen kullanıcı adı ve şifre ile sisteme kayıt gerçeklenir. Kayıt işleminin yapılmasından sonra etmen istediği zaman sisteme dahil olabilir veya sistemden ayrılabilir. Etmenin kaydını yaptırabilmesi için bilgisayar adresi, yönlendirici ve diğer etmenlerle iletişim kuracağı portu belirlemesi gerekmektedir. Şekil 7.7'de etmen kaydına ilişkin bilgiler yer almaktadır.

```
MyAddress
AgentThesis,bag2.cs.itu.edu.tr,5558,MessageRouter,(agent-info  :password AgTh)
RouterAddress
Router, manolya.cs.itu.edu.tr,4444,MessageRouter,(MessageRouter)
RouterRegistrarAddress
RouterRegistrar, manolya.cs.itu.edu.tr,4445,MessageRouter,(MessageRouterRegistrar)
RegisterRequest
yes
MaxIdleTime
1000
```

Şekil 7.7 Etmen Kaydı İçin Gerekli Olan Bilgiler

Kullanıcı, kayıt yapıldıktan sonra öğrenme sürecinde bir birey olarak yer alır. Kullanıcı Arayüz Etmenine Bölütleme Etmeninden birkaç puanlama mesajı gelir. Bu puanlama mesajları parametre ağırlıklarının öğrenilmesi için Bölütleme Etmeninin ürettiği ifadelere ilişkin parametreleri içerir. Kullanıcıdan bu ifadelere ilişkin olarak ifadeyi sağlıyor veya sağlamıyor şeklinde geri besleme alınır. Öğrenme süreci sonlandığında kullanıcının kendi kesitini oluşturmak üzere Kullanıcı Arayüz Programını kullanması sağlanır. Kullanıcı Arayüz Programı ile sürekli iletişim kurarak kullanıcı kesitinin oluşturulması için gerekli güncellemeleri yapar. Reaktif nesnelerle tanımlanmış olan portlar üzerinden haberleşir. Bağlantı isteğinde bulunan reaktif nesne yürütmek istediği davranışı belirtir. Bu esnada diğer Kullanıcı Arayüz Etmenlerinden gelen yüz ifadelerine ilişkin parametre bilgilerini "UserProfile" sınıf tanımlamalarına sahip "AgentGrades.dat" dosyasında tutar. Bu sınıfta önceki kesit bilgilerine ek olarak puan için ayrılmış integer tipinden bir veri alanı (grade) bulunmaktadır. Bu alan, kullanıcının (istediği zaman) ifade için verdiği puana göre doldurulmaktadır. Kullanıcı Arayüz Etmeni, Kullanıcı Arayüz Programından "Grade" davranışı geldiğinde sırasıyla kendisine gelen yüz ifadelerinin soru numaralarını Arabirime yollar ve Grafiksel Yüz Animasyon programında gelen ifadenin şeklinin oluşması için gerekli dosya güncellemesini yapar ve güncelleme yaptığını belirten işareti koyar. Böylelikle kullanıcının hem ifadeyi hem de ifadeye karşılık çizilen şekli aynı anda görmesi sağlanarak bu çizime ilişkin bir puan vermesi için gerekli ortam hazırlanmış olur. Bu süreç içinde kullanıcıdan gelecek olan puanı bekler. Kullanıcı puanı geldikten sonra ilgili dosyaya erişerek sadece ifadeye ilişkin kaydın "grade" alanını günceller. Böylelikle veriler bölütleme süreci için hazırlanmış olur. "Up/Down" davranışları için "expression-macros.dat" dosyasından ilgili parametrenin önceki değerinin alınıp davranışa göre yeni değerinin dosyaya yazılması gerekmektedir. Bu arada diğer parametrelere ilişkin bilgilerin değişmemesi gerekmektedir. Dosyaya güncelleme gerçeklendiğinden dolayı kullanıcının yürüttüğü isteği Grafiksel Arayüz Programında Yüz İfadesinde değişiklik olarak döndürmüş olur. Böylelikle kullanıcı, yaptığı değişiklikleri yüz çiziminde gözlemlemiş olur. Etmenlerden gelen "Update" isteği sonucunda tüm parametrelerin son değerlerini UserProfile.dat dosyasına kaydeder. "GetParameter" isteğine karşılık olarak etmen parametresinin son değerini yollar. "SetParameter" davranışı sonucunda porta yazılan değeri ilgili parametrenin yeni değeri olarak kaydeder. "Cluster" isteğine karşılık olarak bölütleme işlemini gerçekler. Portlar için "Close" isteği geldiğinde kendisini de sonlandırmadan önce portları düzgün şekilde kapatır ve çoklu etmen sisteminden bağlantısını keser. Şekil 7.8'de Kullanıcı Arayüz Etmeni görevcikleri ve çoklu etmen sistemi arayüzü görülmektedir.



Şekil 7.8 Kullanıcı Arayüz Etmeni Görevcikleri ve Çoklu Etmen Sistemi Arayüzü

Sistemde işlenen KQML mesajları ve işlenme durumlarına ilişkin bilgiler Tablo 7.2'de bulunmaktadır.

Tablo 7.2 KQML Mesajları

Mesaj Komutu	Karşılık Mesajı	Yollanan Birim	Yollayan Birim
List-agents	Registered-agent	Yönlendirici	Tüm Etmenler
Registered-agent	Yok	Tüm Etmenler	Yönlendirici
Parameter-in	Yok	Tüm Etmenler	Tüm Etmenler
Parameter-all	Yok	Tüm Etmenler	Tüm Etmenler
Parameter-out	Yok	Tüm Etmenler	Tüm Etmenler
Learning-process	Reply-learning	K. Arayüz Etmenleri	Bölütleme Etmeni
Ask-weights	Reply-weights	Tüm Etmenler	Tüm Etmenler
Reply	Yok	Tüm Etmenler	Tüm Etmenler
Error	Hatalı mesaj	Tüm Etmenler	Tüm Etmenler
Connect	Hatalı ise mesaj	Yönlendirici	Tüm Etmenler
Register	Hatalı ise mesaj	Yönlendirici	Tüm Etmenler

JATLite Yönlendiricisine yönelik mesajlara ilişkin açıklamalar II. Bölümde bulunabilir. “Parameter-in”, “Parameter-all“ komutları sırasıyla tek ve tüm yüz ifade parametrelerini yollamak için, “Parameter-out” komutu parametre isteğinde bulunmak için, “Learning-process” komutu öğrenme sürecinde kullanıcıdan puan almak ve “Ask-weights” komutu parametre ağırlıklarını öğrenmek için kullanılır. KQML mesajlarına ilişkin yürütülen işlemlere ilişkin ayrıntılı bilgi VIII. bölümde verilmektedir.

Sistemde etmenler arası gönderilen örnek bir KQML mesajı şu şekildedir:

```
(tell
        :sender AgentThesis
        :receiver ClusterAgent
        :language default
        :ontology FacialAnimationWorld
        :content (parameter-in 1 2.6 2.2 0.1 0.1 0.3 1.3 0.1 0.1 -0.2 2.2 0.0 0.0 0.0 0.0 0.0
                0.0 0.2 0.0))
```

Bu mesajda 1. soruya ilişkin parametre bilgileri AgentThesis etmeninden ClusterAgent etmenine yollanmaktadır. Mesajların çözümlemesi ve gerekli işlemlerin yapılması etmenlerin görevidir. Etmen, “parameter-in” komutunu çözümleyip yollanan parametre bilgilerinin hangi soruya ilişkin olduğunu belirleyip bu bilgiyi gerekli bölüme aktarır. Böylelikle bir yüz ifadesi, sadece 18 parametre ile etmenlerin bildikleri bir ontoloji tanımına göre aktarılabilmektedir.

7.2.2 Kullanıcı Kesit Analizi ve Bölütleme

Kullanıcı kesitinin oluşturulmasında etkin rol oynayan bileşenler; kullanıcı kayıtları, diğer kullanıcıların çizimleri ve bu çizimlere verilen puanlar ile etmenin etkilenme katsayısıdır.

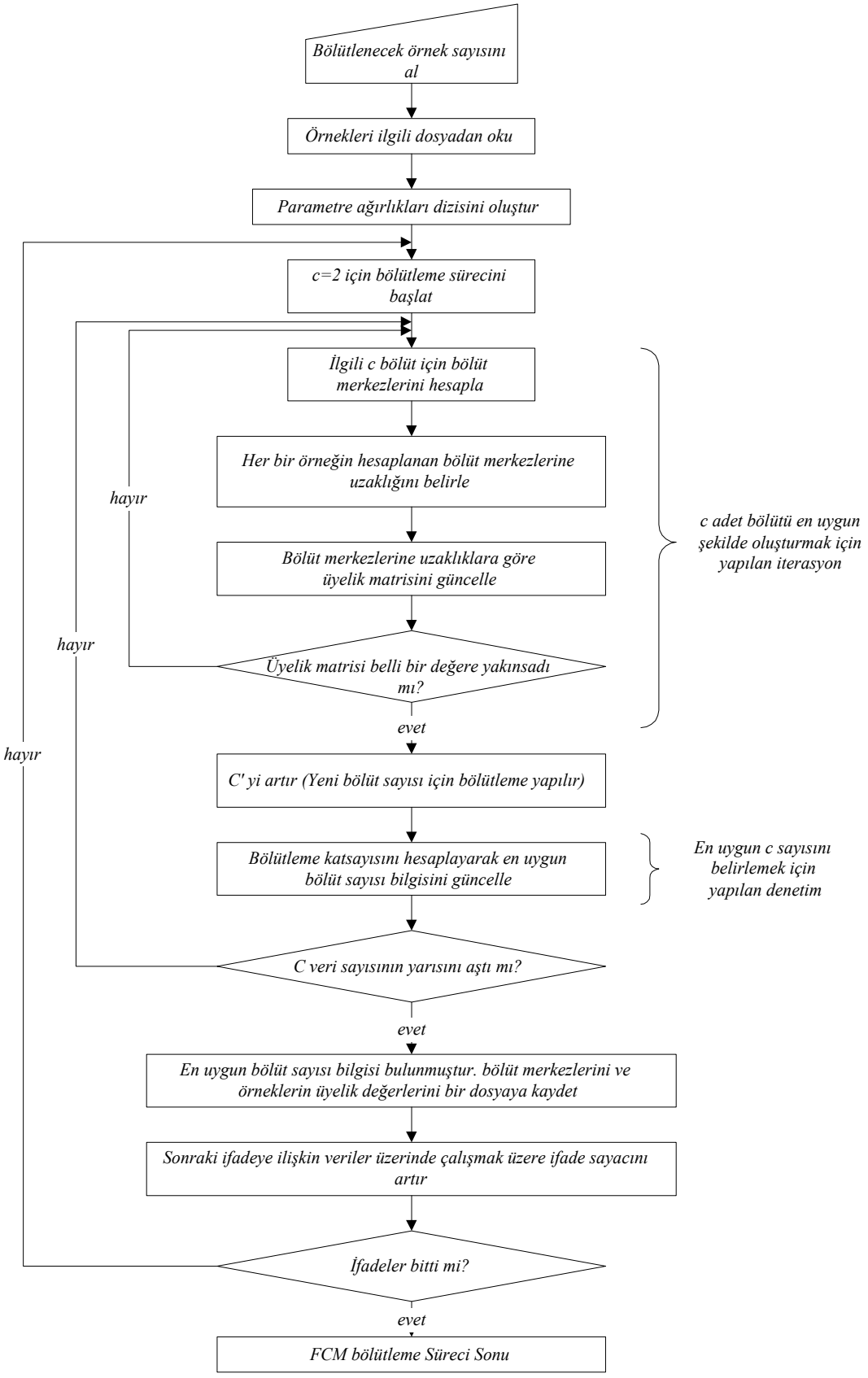
Etkilenme katsayısı (i_c), sistem açılışında her bir kullanıcı etmenine atanan bir değerdir. Etkilenme katsayısı 0 ile 1 arasında değişen ondalıklı bir sayıdır. Bu katsayı etmen inançlarının diğer etmen inançlarından ne kadar etkileneceğini belirler.

Dolayısıyla bir etmen için i_c yüksek olduğunda, bölütlemede diğer etmenlerin oluşturdukları yüz ifadeleri daha etkin rol oynamaktadır. Doğadaki canlıların birbirlerinden etkilenme durumu modellenmeye çalışılmıştır. Böylelikle etmenlerin inançlarının toplumun genel kesiti ile karşılaştırıldığında oluşan farklılıklar ve benzerlikler değerlendirilebilir.

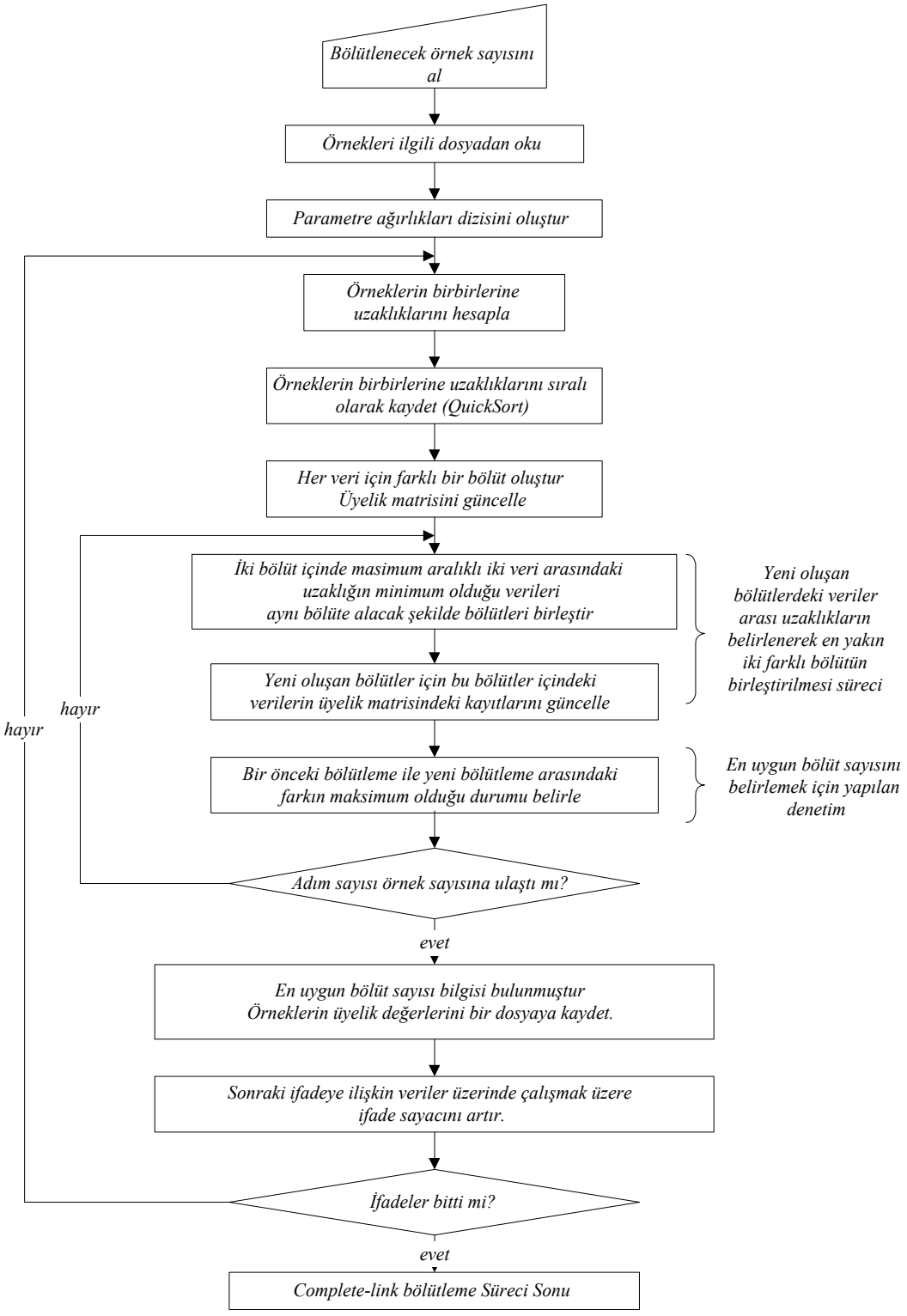
Bölütleme için Kullanıcı Arayüz Etmeni, Bölütleme Etmeninden parametre ağırlıkları bilgisini “ask-one” performative’ i ve “ask-weights” komutu ile alır. Bu ağırlık bilgileri bölütleme sürecinde her bir parametrenin ağırlığı olarak kullanılır.

Bölütleme için üç algoritma gerçeklenmiştir. Her üç algoritma da veriler üzerinde denenebilmektedir. Kullanıcı kesitini oluşturmak üzere bölütleme sürecinde FCM (Şekil 7.9), Complete-link (Şekil 7.10), Single-link (Şekil 7.11) algoritmalarının her biri kullanılabilir.

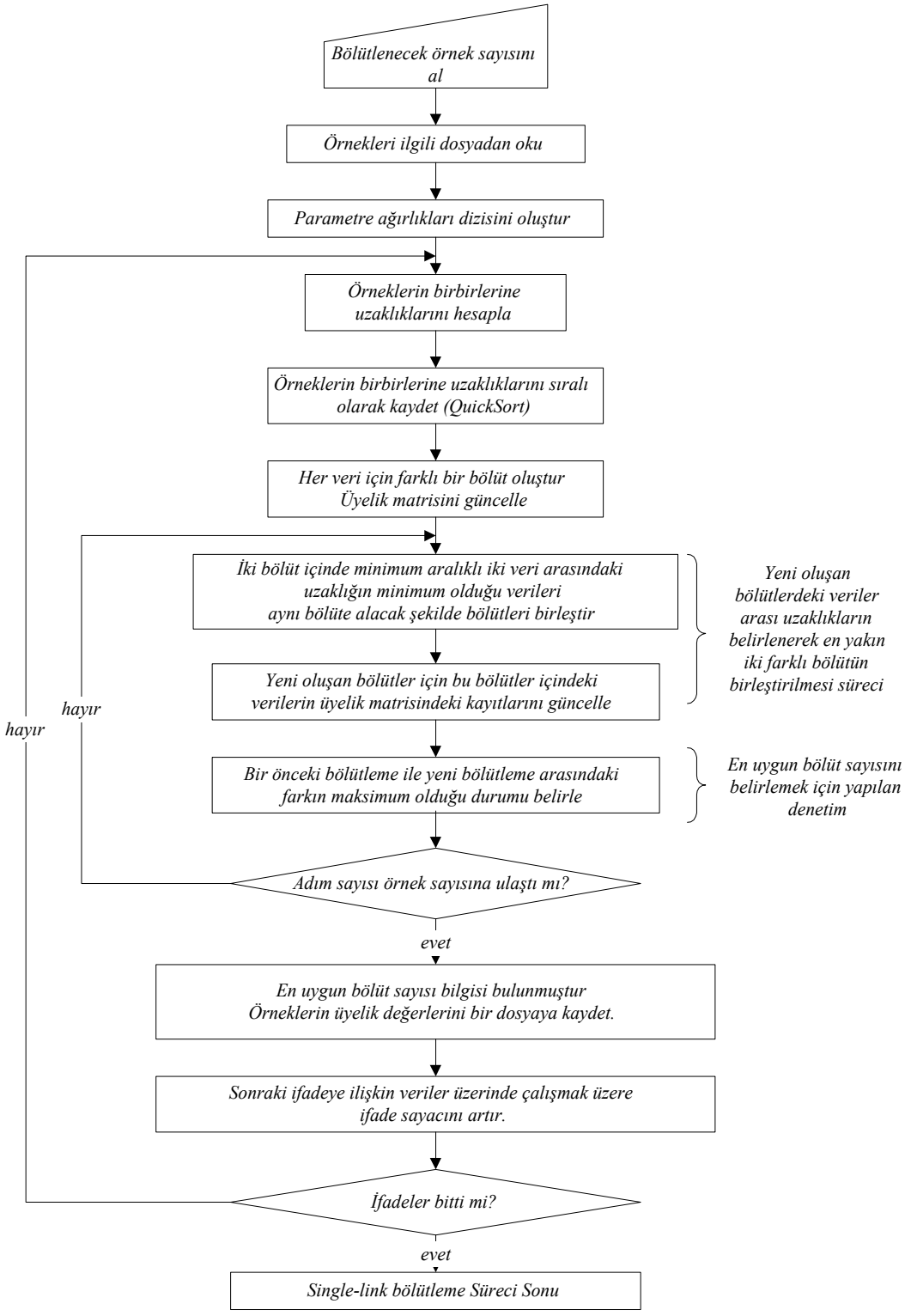
FCM algoritması için bu sürecin sonuçları, bölüt merkezleri ve bölüt içi belli noktaların gösterilimi ile simülasyon şeklinde oluşturulur ve Kullanıcı Arayüz Programı “Bölütleme Sonuçları” (ayrıntıları VIII. Bölümde anlatılmaktadır) ekranında görüntülenebilir. Single-link ve Complete-link algoritmaları sonucunda sadece verilerin hangi bölütlerde yer aldığı bilgisi oluşur. Merkez hesabı ayrıca yapılabilir. Sürecin sonunda ifadelere ilişkin örneklerin hangi bölütlerde yer aldığı bilgisi ilgili algoritma için ilgili dosyalarda tutulur.



Şekil 7.9 FCM Bölütleme Algoritması



Şekil 7.10 Complete-link Bölütleme Algoritması

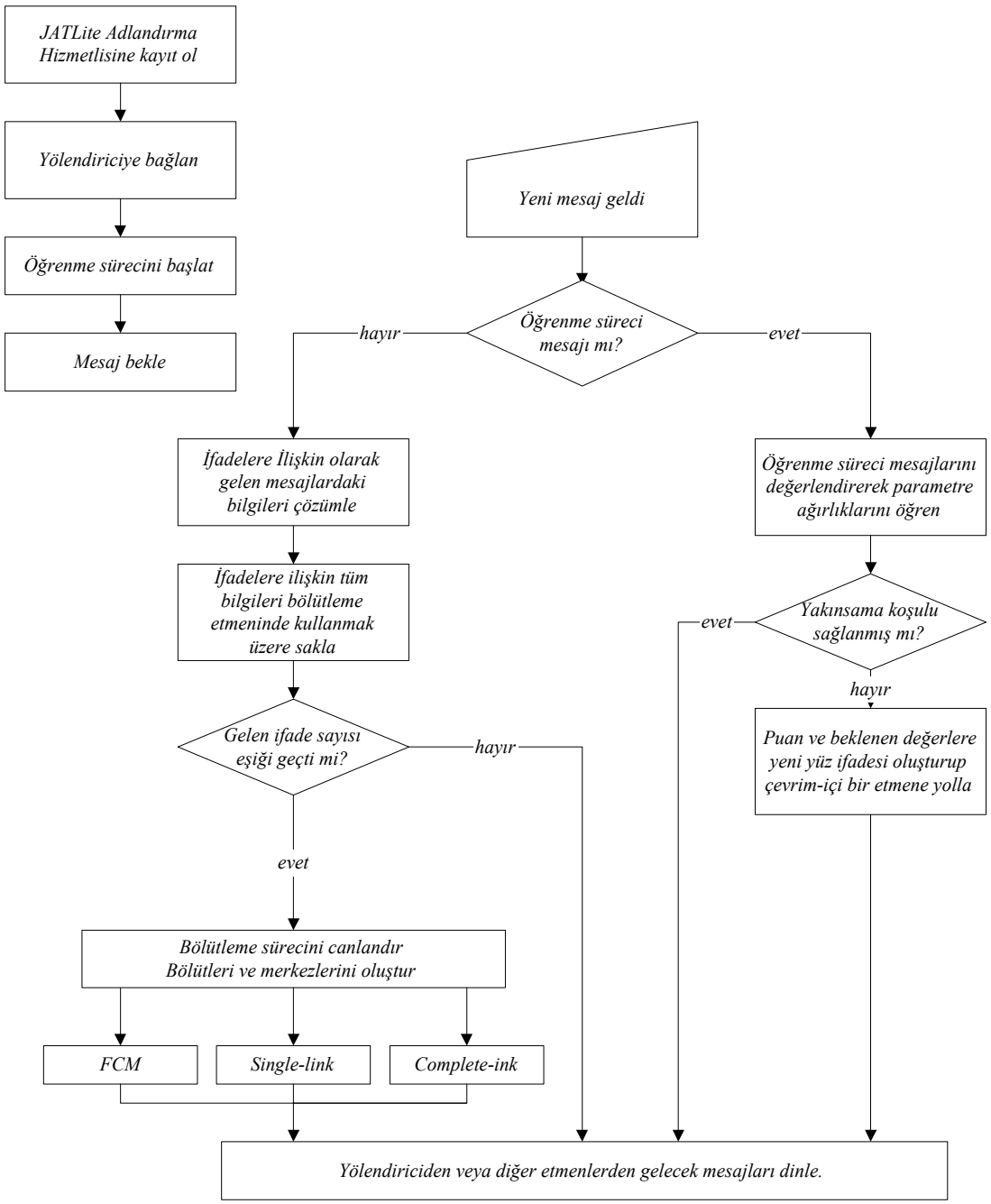


Şekil 7.11 Single-link Bölütleme Algoritması

IV. Bölümde de belirtildiği gibi Complete-link ve Single-link algoritmaları bölütler arası uzaklık ölçümünde farklılaşmaktadır. Sonuçları karşılaştırmak için bu algoritmaların veri topluluğu üzerinde ayrı ayrı denenmesi analiz açısından uygun olmaktadır.

7.2.3 Bölütleme Etmeni

Bölütleme Etmeni de diğer etmenler gibi çoklu etmen sisteminde yer alan akıllı bir etmendir (Şekil 7.12). Bu etmen hem öğrenme hem de bölütleme yapma yeteneğine sahiptir. Toplum kesiti analizi bu etmen tarafından yapılır. Bölütleme etmeninde de bölütleme süreçleri için FCM, Complete-link ve Single-link algoritmalarından biri seçilebilir. Öğrenme sürecinde Q öğrenmesi yöntemi uygulanır.

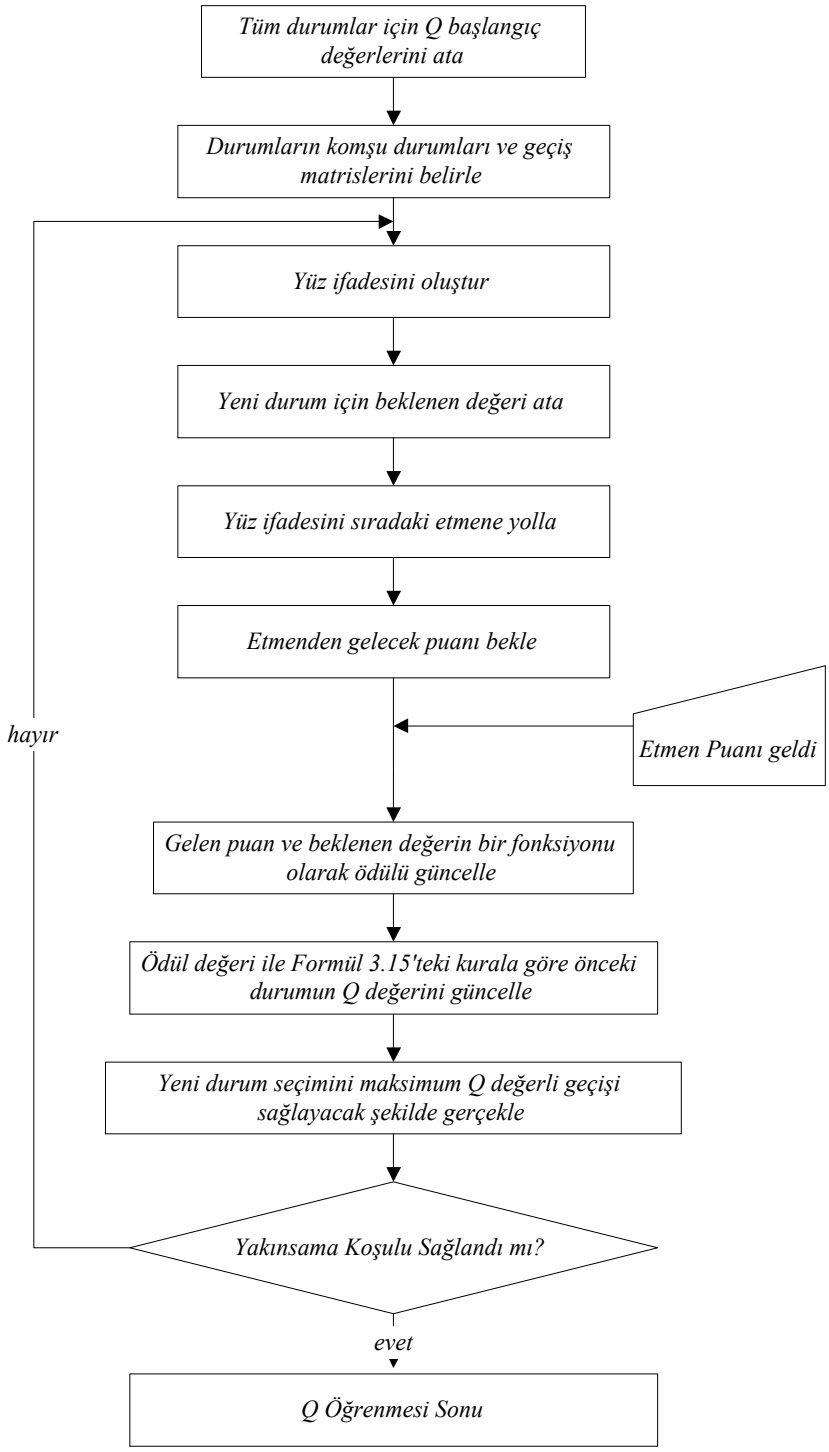


Şekil 7.12 Bölütleme Etmeni Yaşam Çevrimi

Bölütlemede sadece yüz ifadesinde etkili olan parametrelerin işlemine girmesi gerekir. Bu parametrelerin herbiri bölütleme sürecinde birer boyuttur. Bölütleme sürecinin daha etkin çalışabilmesi ve daha doğru sonuçlar üretebilmesi için her bir ifade için parametre ağırlıklarının öğrenilmesi gerekmektedir. Örneğin "gülme" ifadesi için dudak parametresinin ağırlığı büyük olmasına karşın kaş iç parametresinin ağırlığı fazla değildir. Çünkü gülen bir yüzde kaşın değişimi ifadeyi küçük ölçüde değiştirmektedir. Bundan dolayı her bir parametrenin her bir ifade için ağırlığının belirlenmesi gerekmektedir. Bu da hangi parametrenin yüzün hangi ifadesinde etkin olduğunu belirleme işlemidir. Özellikle yapay karakterler için istenilen ifadelerin oluşturulmasında önemli bir katkı sağlar. Bu işlem, sistem kapsamında öğrenme süreci içinde yürütülmektedir. Bu işlemi Bölütleme Etmeni gerçekler. Bölütleme öncesinde parametre ağırlıklarının belirlenmiş olması gerektiğinden dolayı bu süreç sistemin çalışmasında ilk adım olmaktadır.

7.2.4 Öğrenme Süreci

Öğrenme süreci, bölütleme etmeninin parametre ağırlıklarını öğrenmek üzere çeşitli yüz ifadelerini kullanıcı arayüz etmenlerine yollaması ve bu arayüz etmenleri yardımıyla, kullanıcılardan ilgili yüz ifadelerine karşılık puan alınması sürecidir. Şekil 7.13'te öğrenme süreci yürütme adımları verilmiştir. Bu süreçte Bölütleme Etmeni, sistemdeki tüm kullanıcılardan kendi oluşturduğu yüz ifadelerine puan vermelerini sağlayarak en uygun ifadeye ulaşmak için hangi parametreleri değiştirmesi gerektiğini, yani parametre ağırlıklarını öğrenir. Tüm ifadeler için ayrı ayrı parametre ağırlıkları öğrenilmektedir. Kullanıcı Arayüz Etmenleri bölütleme esnasında kullanmak üzere bu parametrelere ihtiyaç duyabilir. Bu durumda Bölütleme Etmenine parametre ağırlıklarını isteme mesajını (ask-weights komutuyla) gönderir. Bölütleme Etmeni, bu istek geldiğinde ilgili etmene ifadeler için kullanılacak olan parametre ağırlıklarını yollar. Eğer Bölütleme Etmeninden bir yanıt gelmezse etmen parametre ağırlıklarını sistemde o anda bağlı olan başka bir etmenden ister.



Şekil 7.13 Öğrenme Süreci Yürütme Adımları

7.2.4.1 Öğrenme Süreci Adımları

Bölütleme Etmeni, rasgele olarak bir yüz ifadesi oluşturur ve sistemdeki bir kullanıcı arayüz etmenine bu yüz ifadesi bilgilerini yollar. Buna karşılık olarak kullanıcı, bu yüz ifadesi için 'olumlu/olumsuz' bir puan verir. Bu puan, Bölütleme Etmeninin parametre ağırlıklarını öğrenme sürecinde bir geri besleme olarak değerlendirilir. Bu geri beslemeleri kullanarak yüz ifadeleri geçişlerinde hangi parametrelerin değiştiğini ve bu parametrelerin ifadeleri ne şekilde etkilediğini göz önüne alır. Bölütleme Etmeni, öğrenme sürecinde sistemdeki tüm kullanıcılar ile etkileşim halindedir. Öğrenme süreci ile sisteme dışarıdan bir yükleme yapılmadan kullanıcı başına belli sayıda puanlama görevi vererek parametrelerin etkinlik durumlarını belirlemek amaçlanmaktadır. Bu süreç, bir öğrenme süreci olduğundan oldukça zaman alıcı ve çok deneme-yanılma adımları içeren bir süreçtir. Bu süreç içindeki en önemli zorluk modellenmek istenen sistemin determinist olmamasıdır. Her kullanıcı, her yüz ifadesine aynı şekilde puan vermeyebilir. Bu durumda tasarlanan sistemin non-determinist olan bu ortamı modelleyebilecek kapasiteye sahip olması gerekmektedir. Bu çalışmada öğrenme süreci için durum sayısını küçük tutmak için bir takım varsayımlar yapılmıştır.

Her bir ifade için 18 parametreden hangilerinin etkili olduğu belirlenmektedir. Bunun için her bir ifade için hangi parametrelerin etkili olduğunu gösteren diziler elde edilmektedir. Determinist bir simülasyon ortamında Şekil 7.14'teki parametre dizisinin belli değerler arasında olması durumunda "+" puan üreten aksi durumda "-" puan üreten bir simülasyon kodu oluşturulmuştur. Bu modelde öğrenen sistem 1 ve 0' lardan oluşan bir dizi elde eder. Bu dizi, hangi parametrelerin ifadede rol aldığını tümüyle belirtir. Örneğin [110001111] dizisi, bir ifadede dudak, dudak büzme, dudak ve burun, dudak ve burun iç, kaş bitiştirme (çatma) ve göz ve burun (birlikte) kaslarına ilişkin parametrelerin etkili olduğunu diğer parametrelerin ifadede değişiklik yaratmadığını gösterir.

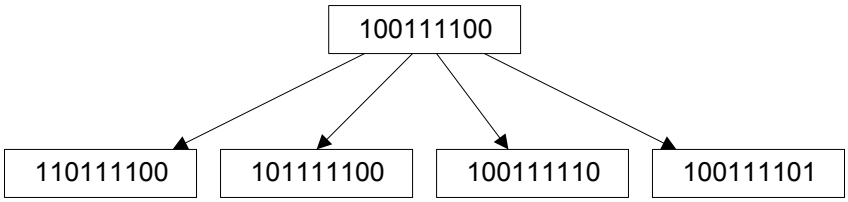
Öğrenme süreci sonunda, Bölütleme Etmeni ifadelere karşılık olarak bu türden dizileri oluşturabilmektedir.

[Zygomatic_Major, Angular_Depressor, Frontalis_Inner, Frontalis_Major, Labi_Nasi, Inner_Labi_Nasi, Lateral_Corigator, Lateral_Corigator, Secondary_Frontalis]

Şekil 7.14 Parametrelerin İfadedeki Etkilerini Gösteren Dizi (Sol, sağ ayrımı yapılmadan, sadece kaslar üzerinde çalışılmıştır.)

7.2.4.2 Öğrenme Modeli

Bölütleme Etmeni etkileşim kurduğu Kullanıcı Arayüz Etmenlerinden sürekli olarak geri besleme alarak tahmini etki dizisini bulmaya çalışır. Bu işlem için Q Öğrenmesi algoritmasını kullanır. Etmenin iletişim kurduğu ortam Kullanıcı Arayüz Etmenlerinin bulunduğu ortam olmaktadır. Bu ortamda etmenin yürüttüğü davranış, bir önceki adımda oluşturmuş olduğu yüz ifadesini değiştirerek başka bir Kullanıcı Arayüz Etmenine yollamaktır. Etmenin içinde bulunduğu durumlar parametre etki dizisinin farklı kombinasyonlarıdır. Parametre etki dizisinin 9 alanı olduğundan ve her bir alan ikili olarak ifade edilebildiğinden dolayı bir durumun kendisine komşu olabilecek 9 farklı durum vardır. Öğrenme sürecinde durumların komşuları (Hamming uzaklığı 1 olan komşular) olarak durum kodunda sadece 0 olan bitlerinin tümlendiği (diğer bitlerin değişmediği) durumlar alınmıştır. Şekil 7.15'te bir durumun komşuları görülmektedir.



Şekil 7.15 Öğrenme Sürecinde Bir Durumun Komşuları

Böylelikle her bir tarama (000000000→..→111111111) için aşağı doğru bir akış söz konusudur. Etmen bir durumdan başlar (başlangıç Durumu: 000000000). Daha sonra Q Öğrenmesi algoritmasına göre bir sonraki adımda en yüksek Q değerine sahip olan durum, davranış çiftini seçer. Buna göre yeni yüz ifadesini üretir.

Kullanıcı Arayüz Etmeni ile etkileşim halinde olan kullanıcı, kendisine gelen yüz ifadesinin gerçekten ifadeyi verip vermediğini belirlemek üzere +/- puan verir. Bu puan Bölütleme Etmeni için bir önceki ifadeye verilen puan ile beraber değerlendirilen bir büyüklüktür. Her bir adımda Bölütleme Etmeni sadece tek bir parametrenin değişip değişmemesi ile ilgilenir.

7.2.4.3 Öğrenme Sürecinde Ödülün hesaplanması

Bölütleme Etmeni, her bir durum için Q değerini sonraki adımda hesaplar. Bunun için ödül fonksiyonunun belirlemesi gerekmektedir. Durum güncellemesinde geçişte alınan ödülün de etkisi vardır.

Tablo 7.3'te ödülün (reward) tahmin edilen değişim ile verilen puanların değişiminin bir fonksiyonu olduğu görülmektedir. Bir adımda hesaplanan ödül, bir önceki adımdaki durumun Q değerinin güncellenmesi için kullanılmaktadır.

Tablo 7.3 Puan Geçişi (önceki puan→şimdiki puan) İle Parametrenin Etkisinin İkili Koduna Karşılık Alınan Puanlar

Puan Geçişi	**0**	**1**
+→-	0	100
-→+	0	100
+→+	100	0
-→-	100	0

Tahmin Edilen Değişim:

0: Değişim olmamalı

1: Değişim olmalı

Her bir adımda Bölütleme Etmeni yeni bir yüz ifadesi oluşturur. Bu oluşturulan yeni ifade yeni durumun bir ifadesidir. Bir sonraki adımda seçilecek olan durum en yüksek Q değerine sahip olan komşu durumdur. Bunun için tüm komşuların Q değerlerinin karşılaştırılması gerekmektedir. Algoritma öncelikle tüm durumları ziyaret eder. Bu işlemin gerekliliği algoritmanın gerçeklenmesi aşamasında oluşmuştur. Başlangıçta algoritmanın en yüksek Q değerli durumları sürekli ziyaret etmesi durumunda yerel maksimumlar oluşmakta ve yüksek ödüllü geçişlere ulaşılamamaktadır. Bundan dolayı ilk önce tüm durumlar ziyaret edilir. Böylelikle tüm yüksek ödüllü olası durumlar taranmış olur. Tüm durumlara ziyaretler tamamlandıktan sonra algoritma sonraki durum olarak en yüksek Q değerli durumları seçme yöntemini kullanır. Bu durumda algoritma en yüksek değerli durumları seçer. Bu seçim sonucu bu durumların Q değerlerinin sürekli artar. Algoritma, Q değerleri sürekli artan durumları belirler ve buna göre ifade için parametre ağırlıklarını oluşturur.

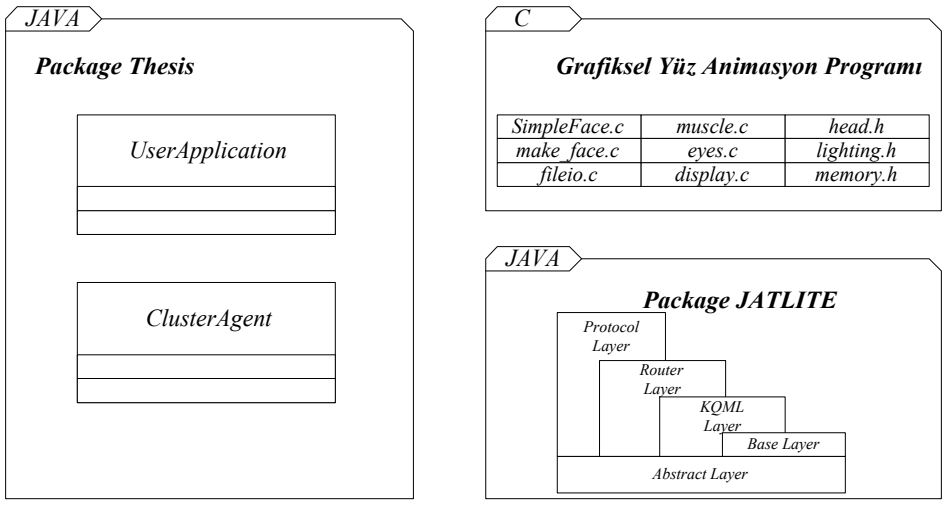
8. SİSTEM YAZILIM MİMARİSİ VE SINIF HİYERARŞİSİ

8.1 Giriş

Bu bölümde tasarlanan sisteme ilişkin gerçeklenen yazılım sınıfları ve aralarındaki bağlantılar verilmektedir. Yazılım, Java programlama dilinde gerçeklenmiştir ve ek olarak hazır Java API sınıfları kullanılmıştır. Kullanılan sınıflar ve tasarlanan sınıflar ile bu sınıfların metotları arasındaki bağlantılar bu bölümde tanıtılmıştır.

8.2 Yazılım Mimarisi

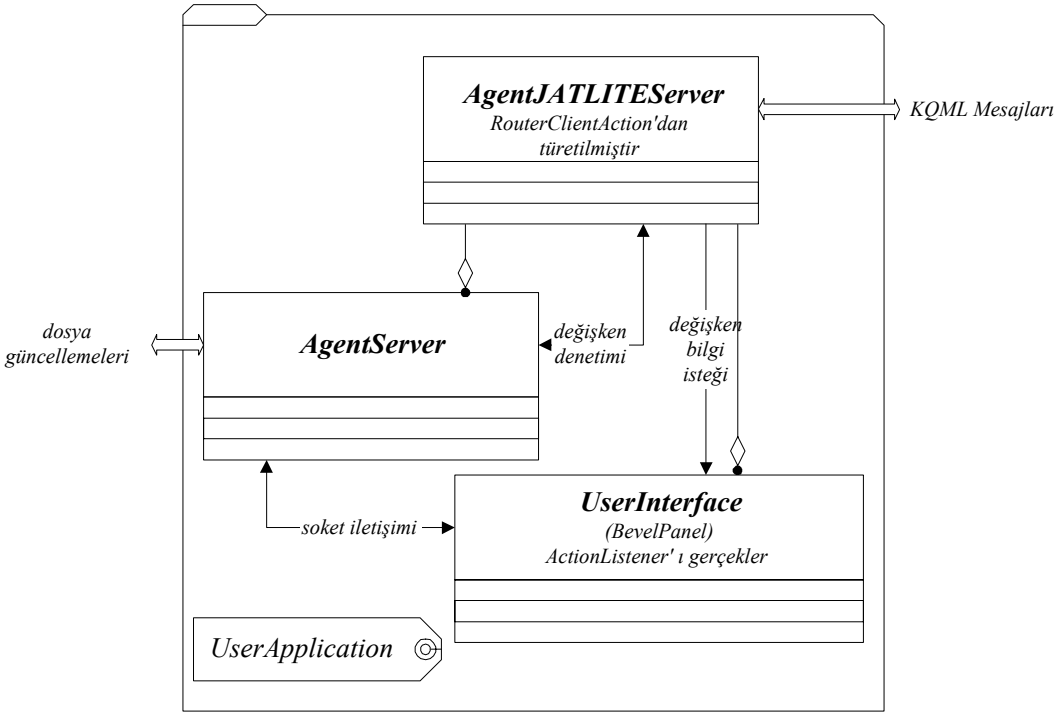
Yazılım sistemi, Java programlama dilinde gerçeklenmiştir. Tez çalışması için oluşturulan "Thesis" pakedi içinde UserApplication ve ClusterAgent sınıfları bulunmaktadır (Şekil 8.1). Gerçeklenen "Thesis" paketi (uygulama paketi) görevcikleri Grafiksel Yüz Animasyon Programı ile dosya üzerinden ve JATLite görevcikleri ile KQML mesajları ile haberleşmektedir.



Şekil 8.1 Yazılım Mimarisi Paketleri

8.3 Kullanıcı Uygulama Sınıfı

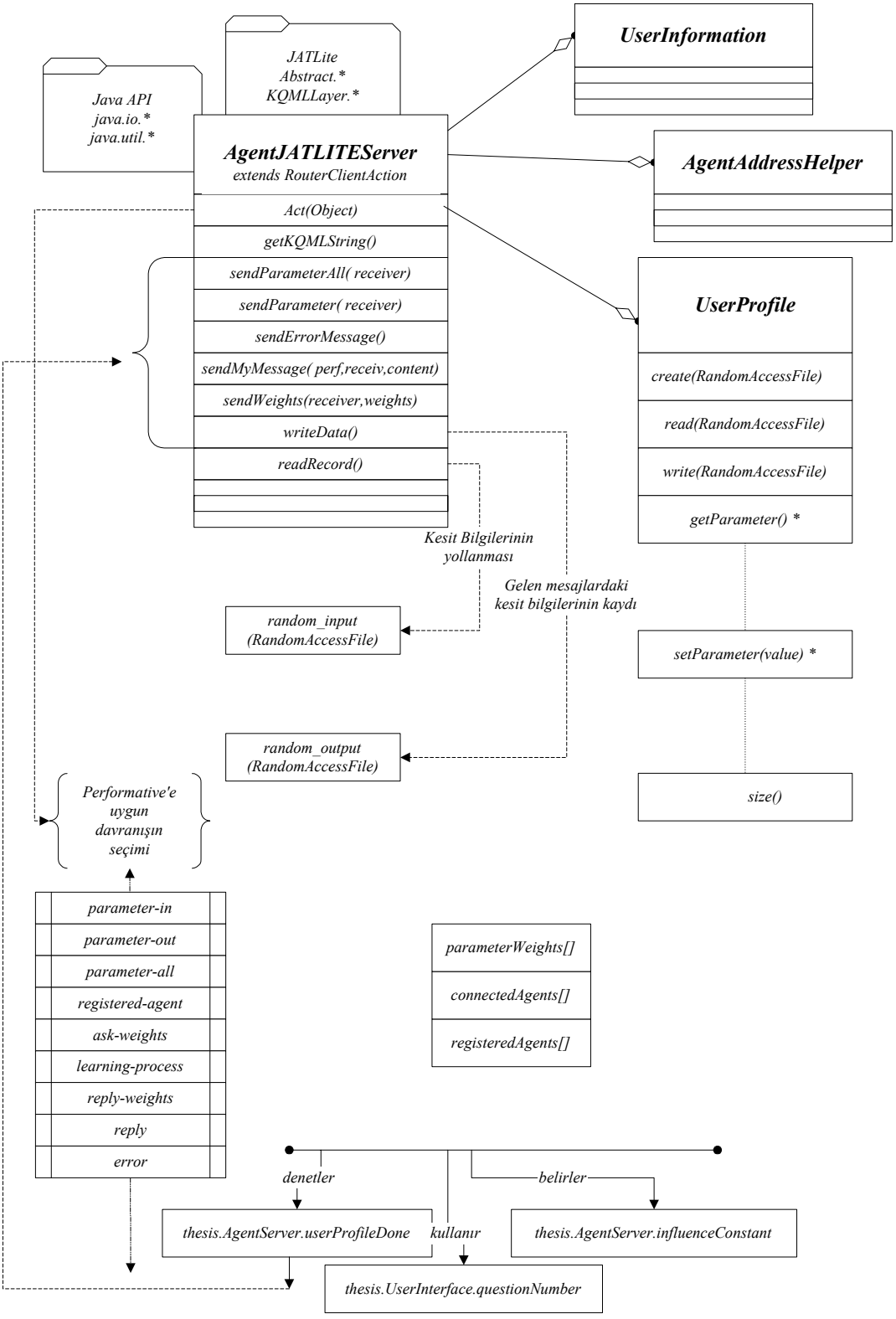
UserApplication sınıfı (Kullanıcı Uygulama sınıfı) kullanıcı ile haberleşen temel bir sınıf olup içinde Kullanıcı Arayüz Birimine ait olan tüm sınıflar bulunmaktadır. UserApplication sınıfı alt sınıfları ile birlikte kullanıcı ile gerçeklenen tüm etkileşimleri gerçekler ve çoklu etmen sistemi ile etkileşimi sağlar. Şekil 8.2'de UserApplication sınıfı ve alt bileşenleri görülmektedir.



Şekil 8.2 UserApplication Sınıfı ve Alt Bileşenleri

8.3.1 AgentJATLITEServer Sınıfı

AgentJATLITEServer sınıfı kullanıcı uygulamasının çoklu etmen sistemi içine dahil olmasını ve JATLite' ın sunduğu hizmetlerden faydalanarak diğer etmenlerle KQML mesajları ile haberleşebilmesini sağlar. Bu sınıfın bir JATLite etmeni olabilmesi için RouterClientAction sınıfından türemiş olması gerekir. Gelen mesajlar ile Act(object o) metodu canlanır. Gelen KQML mesajları bu metot içinde "performative" e göre işlenir. Bu sınıfın değişkenleri, uyandırdığı diğer sınıflar, diğer sınıflardan kullandığı değişkenler ve kendi metotlarına ilişkin şema Şekil 8.3'te verilmiştir.



Şekil 8.3 AgentJATLITEServer Sınıf Hiyerarşisi

AgentJATLITEServer sınıfının metotlarına ilişkin bilgiler Tablo 8.1'de verilmiştir. Bu sınıfın etkin çalışabilmesi için etmenin yerel makine adresi, iletişim kuracağı port numarası ve yönlendirici adresinin alınması gerekir. Bu işlemi gerçekleyen sınıf

AgentAddressHelper sınıfıdır. AgentAddressHelper sınıfı içinde etmene ilişkin dosya (AgentAddressFile) bilgileri okunarak sisteme bağlanmak için gerekli olan bilgiler alınmış olur. AgentJATLITEServer açılışında UserInformation sınıfını uyandırır ve kullanıcıdan etmen adı ve şifresi bilgilerini alır. Bu ad ve şifre ile sisteme bağlantıyı sağladıktan sonra AgentServer ve UserInterface sınıflarını uyandırır. Etmene ilişkin etkilenme katsayısı influenceConstant bu sınıf içinde rasgele olarak etmene atanır. Sistemdeki diğer etmenlerden gelen mesaj tipleri Tablo 8.2'de verilmiştir.

Tablo 8.1 AgentJATLITEServer Sınıfı Metotları

Metot	***Yürüttüğü İşlem***
Act (object o)	Bu metot, RouterClientAction içinde tanımlanmıştır. Uygulama programına göre mesajlar çözümlenmektedir. Mesajın tipine göre ilgili işlem yürütülmektedir.
GetKQMLString()	Yollanmak istenen mesajı KQML mesajı şeklinde paketler.
SendParameterAll(receiver)	Alıcı olarak tanımlanan adrese ilgili parametre dizisi bilgisini yollar. Tüm sorulara ilişkin yüz ifadeleri bu metotla yollanır.
SendParameter(receiver)	Tek bir yüz ifadesine ilişkin parametre dizisini yollar.
SendMyMessage(perf,receiver, content,Message)	"perf" ile belirlenen performative ile içeriği verilen mesaj alıcısına yollanır. Bu metot içinde getKQMLstring() metotu çağrılır.
SendErrorMessage()	Mesajı yollayan etmene mesaj ile ilgili hata olduğunu belirtmek üzere mesaj yollar.
WriteData()	"tell" performative'i ve "parameter-in" ve "parameter-all" komutları ile gelen mesajlardaki parametre bilgilerini "AgentGrades.dat" rasgele erişilebilir dosyasına kaydeder. Bu dosyadaki bilgiler daha sonra puanlama için kullanıcıya gösterilir.
readRecord()	Sistemdeki diğer etmenlere mesaj yollamak için "userprofile.dat" rasgele erişilebilir dosyasından ilgili soruya ilişkin parametre bilgilerini okur.

Tablo 8.2 AgentJATLITE Server Sınıfına Gelen Mesaj Tipleri

Mesaj Tipi	***Açıklaması ve Mesaja Karşılık Yürütülen Görev***
Parameter-in	Bir etmenin bir yüz ifadesine ilişkin olarak parametrelerini yolladığı mesaj tipidir. Gelen mesaj bilgisi çözülerek mesajın içeriği alınır ve writeData() metodu uyandırılır.
Parameter-out	Bu mesaj, mesajı yollayan etmenin parametre isteğini belirtir. Etmenin buna karşılık olarak kesit bilgilerini yollayıp yollamaması kendi elindedir.
Parameter-all	Bir etmenin tüm yüz ifadelerine ilişkin olarak parametrelerini yolladığı mesaj tipidir. Gelen mesaj bilgisi çözülerek her bir yüz ifadesi için mesajın içeriği alınır ve writeData() metodu uyandırılır.
Learning-process	ClusterAgent' tan gelen mesajdır. Etmenin öğrenme sürecine dahil olup yollanan yüz ifadesini puanlaması istenir. Bunun için AgentServer sınıfı ile etkileşimli çalışma yürütülür. Kullanıcının vermiş olduğu puan ClusterAgent' a yollanır.
Registered-agent	Yönlendiriciden gelen mesaj tipidir. Bu mesaj ile sistemdeki tüm etmenler, adres bilgileri ve o anda sisteme bağlı olup olmadıkları bilgileri alınır. registeredAgents ve connectedAgents dizileri bu şekilde oluşturulur. Bu mesaj, yönlendiriciye "list-agent" mesajının yollanmasından sonra gelmektedir.
Ask-weights	Bu mesaj ClusterAgent ile iletişim kuramayan bir etmenin yüz ifadelerine ilişkin parametre ağırlıklarını öğrenmek istediği mesaj tipidir. SendWeights(..) metotu çağrılır. parameterWeights dizisi karşı mesaj olarak yollanır.
Reply-weights	ClusterAgent veya herhangi bir etmenden gelen parametre ağırlık bilgilerini belirten mesajdır.
Reply	Daha önce yollanan mesaja karşılık yollanan mesajdır. Mesaj istenen bilgiye göre işlenir.
Error	Etmenin yollamış olduğu bir mesaja ilişkin hata mesajıdır. Mesaj gerekiyorsa tekrarlanır.

AgentJATLITEServer sınıfı gelen mesajlara ilişkin görevlerini yürütürken kullanıcının kendi kesit bilgisini tamamlamasını denetler. Bu denetlemeyi AgentServer sınıfı içindeki userProfileDone bayrağını gözlemleyerek gerçekler. Eğer kesit bilgisi oluşturulduysa kesite ilişkin tüm parametre bilgileri "sendParameterAll" metodu ile ("tell" performative'i ve "parameter-in" komutu ile) Bölütleme Etmenine yollanır. Daha sonra diğer etmenlerle etkileşim kurmak üzere registeredAgents ve connectedAgents listeleri kullanılarak parametre bilgileri sistemdeki diğer etmenlere de yollanır.

8.3.2 UserProfile Sınıfı

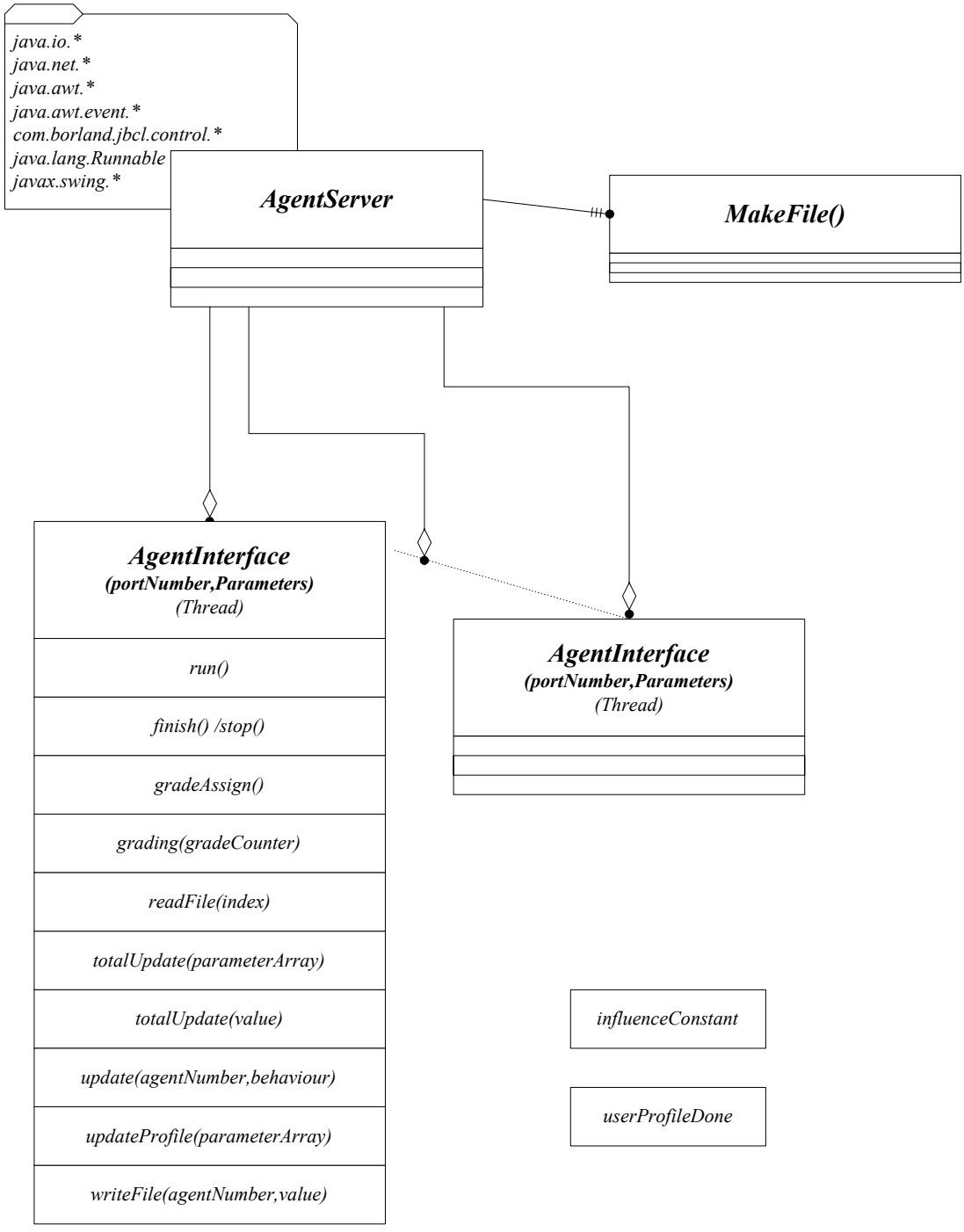
UserProfile Sınıfı, bütün sınıflar tarafından kullanılan bir sınıftır. Bu sınıf, rasgele erişilebilir olarak tanımlanan kayıt dosyalarının kayıt tiplerini belirler. Tüm yüz kas parametreleri, etmen adı, etmen cinsiyeti, soru numarası, puan alanları üzerindeki işlemleri tanımlar. Tablo 8.3'te UserProfile sınıfı metotları ve yürüttükleri işlemler tanıtılmaktadır.

Tablo 8.3 UserProfile Sınıfı Metotları

Metot	***Yürüttüğü İşlem***
create(RandomAccessFile)	MakeFile() sınıfını çağırarak kayıt yapısı belirlenen dosyayı yaratır.
read(RandomAccessFile)	Dosyadan okuma yapar. Tüm parametreler ve etmen bilgileri, soru numarası tek bir kayıt için okunur. get..() metodu ile ilgili alan bilgisine ulaşılır.
write(RandomAccessFile)	Dosyaya yazma yapar. Tüm parametreler ve etmen bilgileri, soru numarası tek bir kayıt için yazılır. Bu metottan önce set..(value) metotları çağırılmış olmalıdır.
get..()	Okumadan sonra ilgili alan bilgisini döndürür. Tüm alanlar için bu işlevi gerçekleyen bir metot mevcuttur.
set..()	Yazmadan önce ilgili alan bilgisinin güncellenmesini sağlar. Tüm alanlar için bu işlevi gerçekleyen bir metot mevcuttur.
Size()	Kayıt alanları bütününün boyutunu döndürür. Dosyada kayıt boyu kadar ilerlemek için kullanılır.

8.3.3 AgentServer Sınıfı

Bu sınıf UserInterface Sınıfının Hizmetlisi olarak çalışmaktadır. Şekil 8.4'te AgentServer sınıf hiyerarşisi görülmektedir.

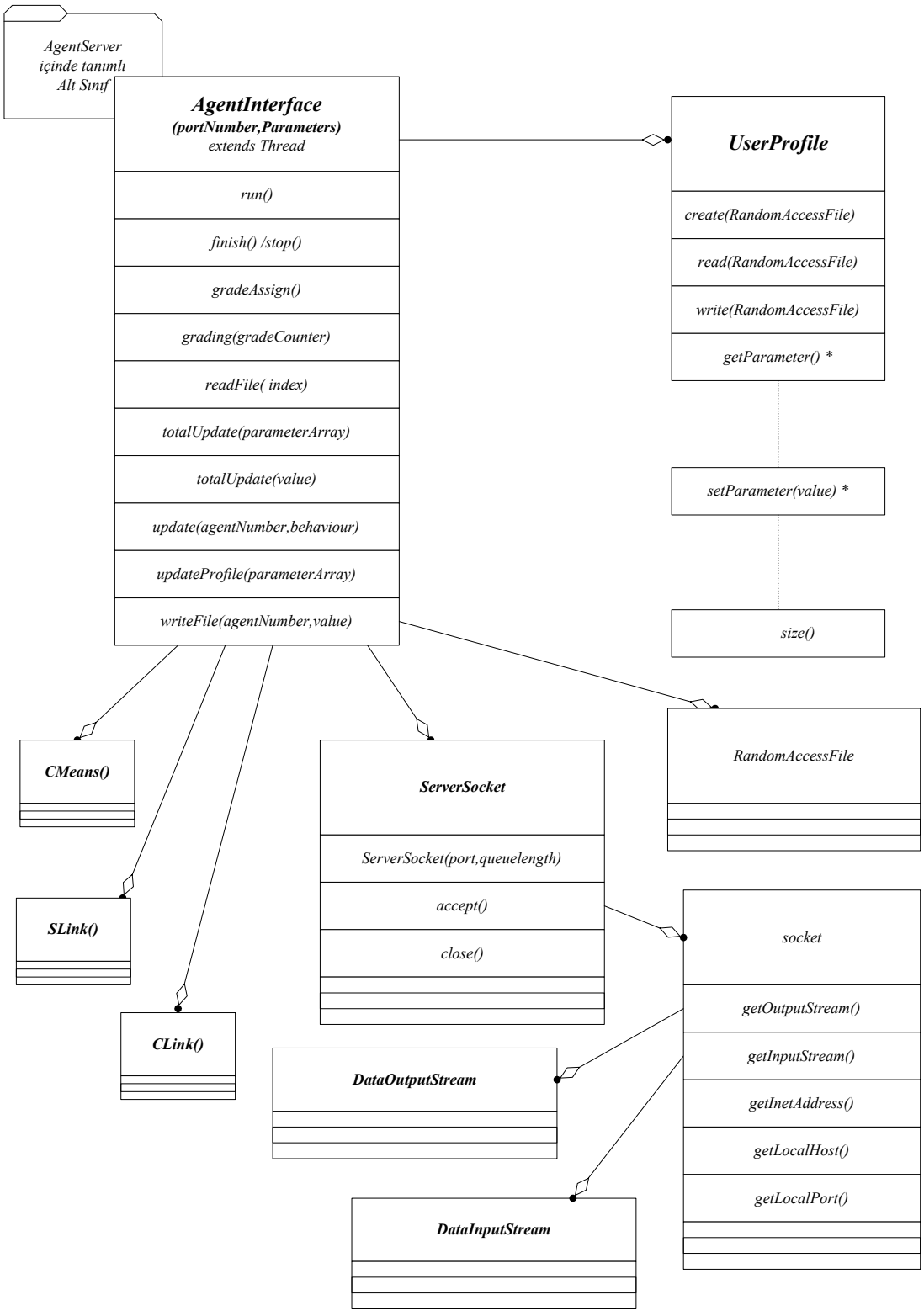


Şekil 8.4 AgentServer Sınıf Hiyerarşisi

Bu sınıfta UserInterface sınıfında yaratılan reaktif nesnelerin bağlantılı iletişim kuracakları hizmetli soketlerinin yaratılması gerekmektedir. Bu görevi AgentInterface sınıfı tek bir soket için gerçeklemektedir. Bu sınıfın tüm yüz parametreleri için aynı türde zamanda paylaşımlı olarak hizmet verebilmesi için her parametre ve ilgili port için hizmetli soketi yaratılır. Bu port değeri, AgentInterface sınıfının yapıcısında parametre olarak alınır. AgentInterface sınıfı çok görevcikli çalışmaya uygun olarak Thread sınıfından türetilmiş olmalıdır. Şekil 8.5'te AgentInterface sınıfı hiyerarşisi görülmektedir. Her bir görevcik için ayrı portlar üzerinden haberleşme sağlanmıştır.

Her bir port için ayrı bir görevcik bulunmaktadır. Görevcik run() metodu içinde soketini açtıktan sonra bağlantı bekler. Bağlantı geldiğinde bağlantı isteğinde bulunan reaktif nesnenin adını ve gerçeklemek istediği davranış bilgilerini alır. Bu davranışa göre yürütülmesi gereken işlemi belirler. Davranışların sonuçları "expression-macros.dat" dosyasını değiştireceğinden dolayı bu dosyanın güncellenmek istenen satır bilgisinin alınması gerekmektedir. Bu dosya rasgele erişilebilir bir dosya olmadığından ve DataOutputStream sınıfı ile dosya güncellemesi yapıldığından tek bir satırın güncellenmesi için satırdan önceki bilgilerin ve satırdan sonraki bilgilerin tutulması gerekir. Bu sınıf ile dosyaya erişildiğinde dosyada araya yazma yapılamamaktadır. Grafiksel Yüz Animasyon Programının okuduğu bu dosya programla birlikte paketlenmiş olduğundan ve Java ile gerçeklenmiş olan yazılım paketinin bu dosyaya yazması ve dosya formatının iki programın da anlayabileceği bir şekilde olması gerektiğinden dolayı, ek veri alanları kullanılarak, ilgili satır dışındaki bilgiler saklanmış ve dosya güncellemesi aşamasında yeniden dosyaya yazılmıştır. "expression-macros.dat" dosyasında her bir satırda iki etmenin bilgisi bulunmaktadır. Bunlar belli bir kasın sol ve sağ bölümleri ile ilişkilidir. Bu yüzden satır seçimi yapıldıktan sonra satırdan sorumlu diğer etmenin bilgisi de saklanmalıdır.

Reaktif nesnelerin yürüttüğü davranışlar için AgentInterface sınıfında yürütülen işlemler Tablo 8.4'te görülmektedir. Tablo 8.5'te de reaktif nesnelerden gelen davranış isteğine karşılık yürütülen işlemler tanıtılmaktadır.



Şekil 8.5 AgentInterface Sınıf Hiyerarşisi

Tablo 8.4 AgentInterface Sınıfı Metotları

Metot	***Yürüttüğü İşlem***
Run()	Bu metot, Thread sınıfının temel metodudur. Thread' in aktif iken yürütmesi istenen işlemler gerçeklenmektedir. Bağlantı isteği algılandıktan sonra etmen ismi ve davranışı alınır. Davranışa uygun işlem yürütülür.
Finish()	Thread sınıfının stop() metodunu içerir. Tüm soketler kapatılarak iletişim sonlandırılır.
GradeAssign()	Kullanıcının ilgili yüz ifadesi için vermiş olduğu puanı "AgentGrades.dat" dosyasındaki ilgili kayda işler.
Grading(gradeCounter)	Kullanıcının belli bir yüz ifadesi için (diğer kullanıcıların çizmiş olduğu) puan vermesini sağlamak üzere "AgentGrades.dat" dosyasından ilgili gradeCounter ile erişilecek kaydı okur ve TotalUpdate(parameterArray) metodunu çağırarak parametre dizisini yollar.
ReadFile()	"expression-macros.dat" dosyasından etmeni ilgilendiren satırın ilgili bölümü okur, önceki ve sonraki satırları geçici bir alanda saklar.
TotalUpdate(parameterArray)	Parametre olarak gelen parameterArray, Grafiksel Yüz Animasyon programında yüz ifadesi olarak görülmek istenen parametre değerleri dizisidir. Bu değerler "expression-macros.dat" dosyasına yazılır. Güncelleme yapıldığına ilişkin olarak da "semaphor.txt" dosyası oluşturulur.
TotalUpdate(value)	Tüm parametrelerin aynı değer ile güncellenmesini sağlar. Özellikle "reset" davranışı için kullanılır. Bu değerler "expression-macros.dat" dosyasına yazılır. Güncelleme yapıldığına ilişkin olarak da "semaphor.txt" dosyası oluşturulur.
Update(agentNumberBehavior)	İlk önce güncellenmek istenen parametre değeri readFile() metodu ile okunur. Davranışa göre güncelleme yapılarak bu değerin yazılması için writeFile(..) metodu çağrılır.
WriteFile(agentNumber,value)	İlgili etmenin sorumlu olduğu satır, daha önceden belirlenmiş güncelleme değeri (value) ile güncellenir. Satır için tutulan önceki ve sonraki satır bilgileri de dosyaya yazılır.
UpdateProfile(parameterArray)	"userprofile.dat" dosyasına kullanıcının ilgili ifadeye (UserInterface.questionCounter) ilişkin çizmiş olduğu yüz parametreleri kaydedilir.

Tablo 8.5 AgentServer Sınıfına Gelen Davranış İsteğine Karşılık Yürütülen İşlemler

Davranış	***Yürütülen Görev***
Update	Yüz ifadesi için oluşturulan en son parametre bilgileri alınarak UpdateProfile() metodu çağrılır.
Cluster	CMeans(), SLink(),CLink() sınıfları uyandırılarak gelen veriler üzerinde bölütleme yapılması sağlanır.
Grade	İlk önce Grading(gradeCounter) metodu ile kullanıcıya yüz ifadesinin gösterilmesini sağlar. Daha sonra puan bilgisi gelene dek askıya alınır. Puan portInput' a ("MyGrade" davranışı ile) geldiğinde GradeAssign() metodunu çağırır.
GetParameter	ReadFile() ile ilgili etmenin parametre değeri portOutput üzerinden yollanır.
SetParameter	İlgili etmenin parametresi WriteFile() ile gelen değere çekilir.
Close	Finish() metodu çağrılır.
Reset	TotalUpdate("0.00") metodu çağrılır.
Up	Update(agentNumber,value) metodu çağrılır.
Down	Update(agentNumber,value) metodu çağrılır.

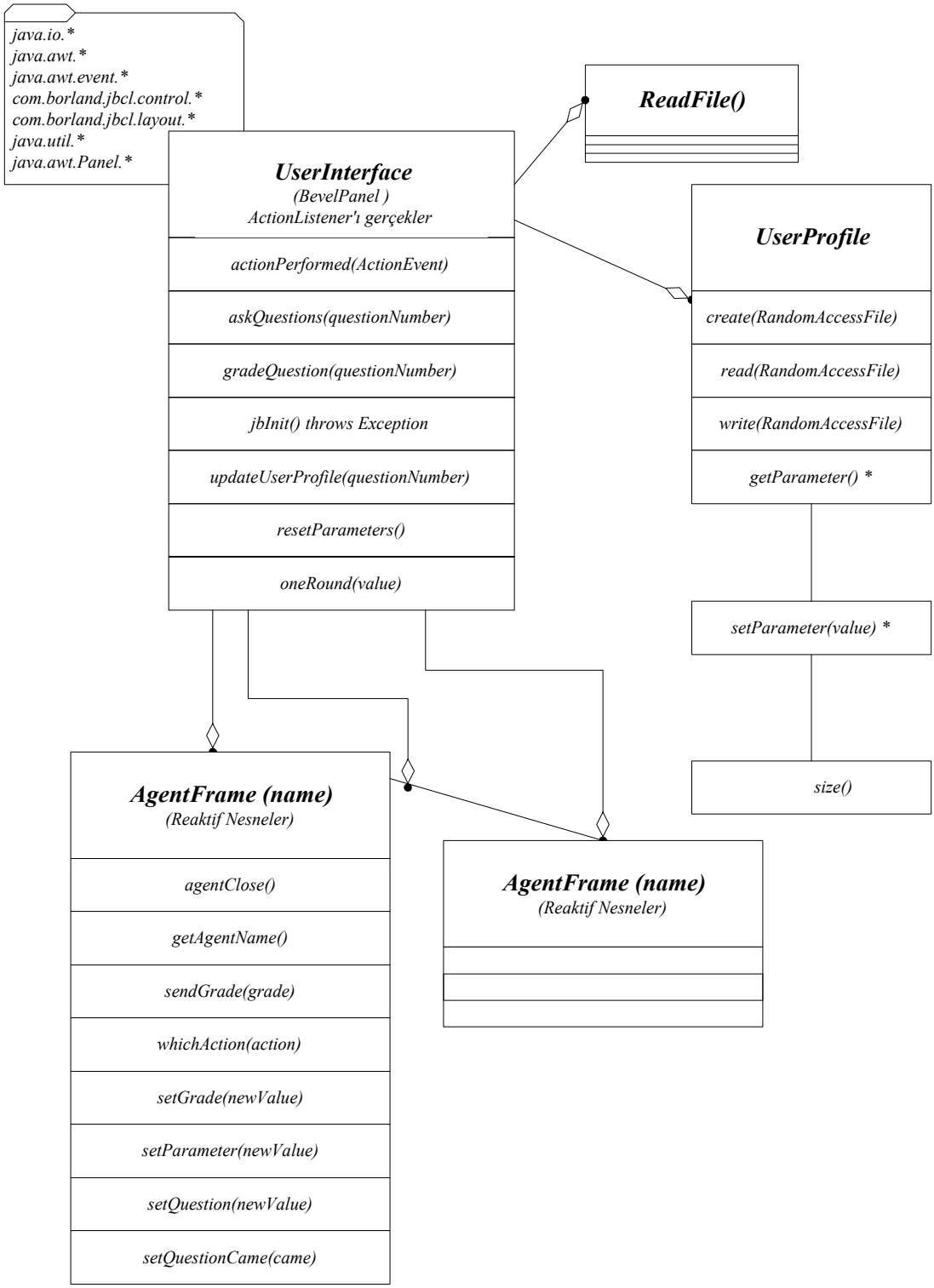
Kullanıcı Uygulama Sınıfı içinde de yer alan CMeans, SLink ve CLink sınıfları aynı şekilde ClusterAgent sınıfında da yer aldığından dolayı bu sınıflara ClusterAgent Sınıfı başlığı altında değinilecektir. İşlev olarak aynı olmalarına rağmen sadece üzerinde çalıştıkları verileri aldıkları dosyalar farklı olmaktadır. Kullanıcı Uygulama Sınıfı içinde, diğer kullanıcılardan gelen veriler "AgentGrades.dat" dosyasında tutulmaktadır.

8.3.4 UserInterface Sınıfı

UserInterface sınıfı reaktif nesnelerin yaratıldığı sınıftır. Kullanıcı ile etkileşim için Java API paketlerinden AWT(Abstract Windowing Toolkit) uygulama için GUI desteği verir. Bu sınıf uyandırıldığında ilk olarak kullanıcı arayüzünü sağlayan bileşenler oluşturulur ve her bir yüz kasından sorumlu reaktif nesneler yaratılır. Bu reaktif nesneler AgentFrame sınıfındandır. AgentFrame Sınıfından türeyen reaktif nesneler kullanıcı isteklerine göre parametre değişim isteklerine yanıt verirler. Tablo 8.6'da AgentFrame sınıfı metotları verilmiştir.

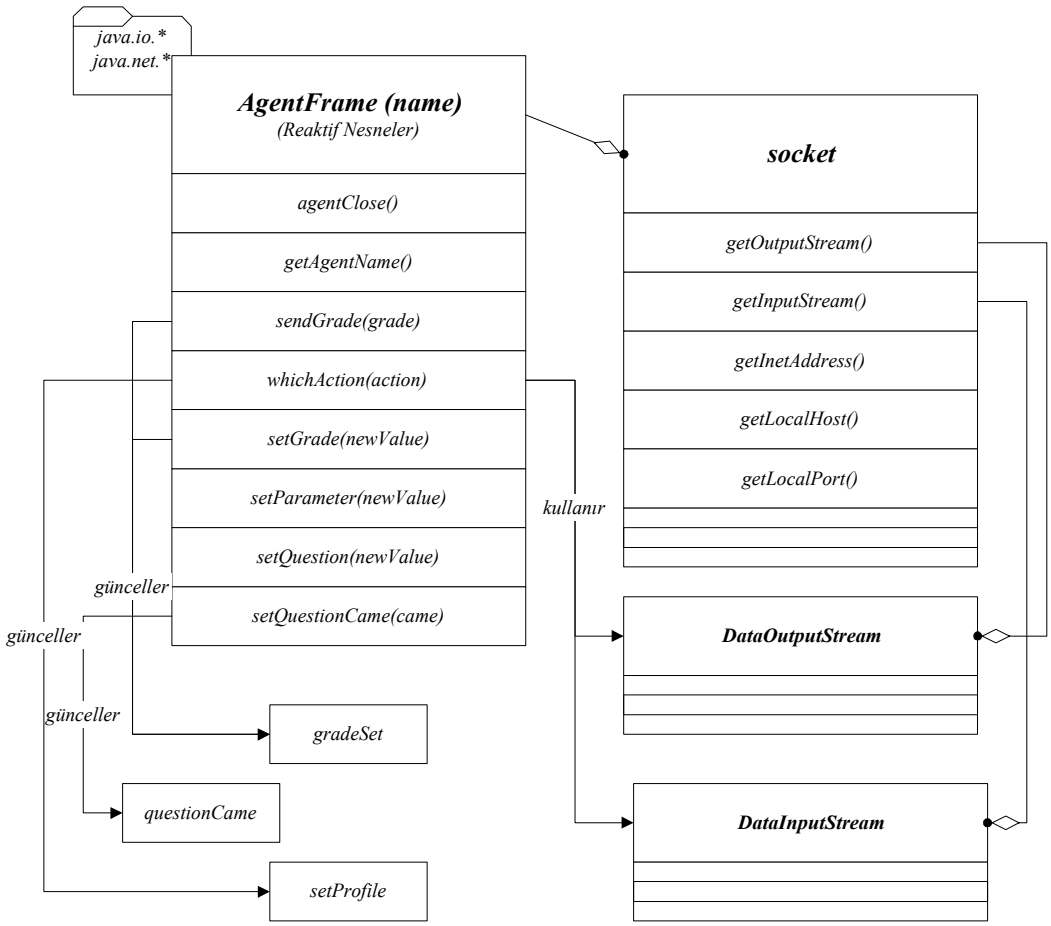
Tablo 8.6 AgentFrame Sınıfı Metotları

Metot	***Yürüttüğü İşlem***
AgentClose()	Reaktif nesnenin iletişim kurduğu istekçi soketini kapatır. Nesnenin varlığını sonlandırır.
GetAgentName()	Nesnenin ismini döndürür.
SendGrade(grade)	Kullanıcının vermiş olduğu puanı (UserInterface sınıfından parametre olarak yollanır) AgentServer' a soket üzerinden yollar.
WhichAction(action)	Nesneye parametre olarak verilen davranışı belirlenen protokole göre AgentServer' a yollar. Davranışa göre ek bilgilerin yollanması gerekiyorsa bu bilgileri de yollar. (Örn. AgentServer' dan soru numarasının alınması, puanın AgentServer' a yollanması gibi)
SetGrade(newValue)	Puanı günceller.
SetParameter(newValue)	"GetParameter" davranışı yürütüldüğünde AgentServer' dan alınan parametre değerini günceller.
SetQuestion(newValue)	AgentServer' dan alınan (puan verilecek olan) yüz ifadesinin sorusuna ilişkin değişkeni günceller.
SetQuestionCame(came)	"Grade" davranışı yürütüldüğünde AgentServer' dan soru numarasının gelip gelmediğine ilişkin bayrağın değerini günceller.



Şekil 8.6 UserInterface Sınıf Hiyerarşisi

Şekil 8.6'da UserIterface sınıf hiyerarşisi görülmektedir. Şekil 8.7'de AgentFrame sınıf hiyerarşisi görülmektedir. Tablo 8.7'de "UserInterace" sınıfı metotları görülmektedir.



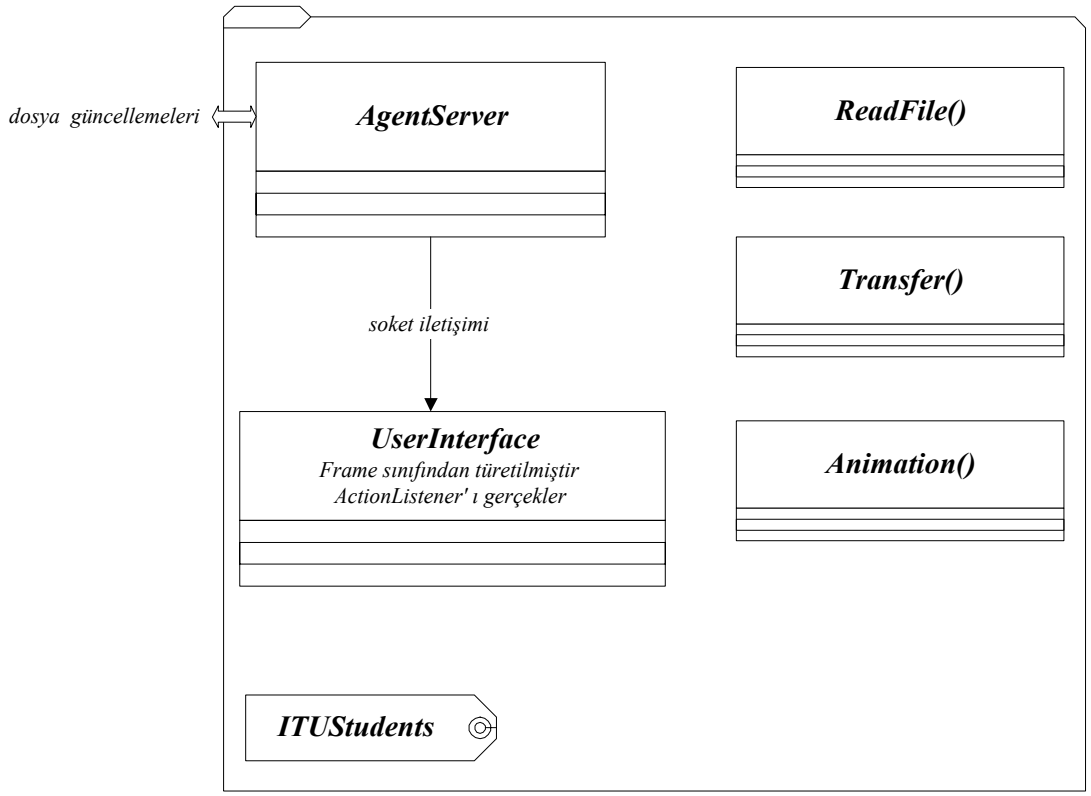
Şekil 8.7 AgentFrame Sınıf Hiyerarşisi

Tablo 8.7 UserInterface Sınıfı Metotları

Metot	***Yürüttüğü İşlem***
actionPerformed(ActionEvent)	Bu metot içinde kullanıcının GUI bileşenlerinden birini kullanması sonucunda ilgili reaktif nesne görevlendirilir.
askQuestions(questionNumber)	Kaydetme işlemi gerçeklendikçe soru sayacı güncellenir. Bu metot ile ilgili sayaç değeri için ilgili sorunun göstergede (display) gösterilmesi sağlanır.
gradeQuestion(questionNumber)	Reaktif nesnelerin yürüttüğü "Grade" davranışı sonucunda AgentServer' dan alınan soru numarasına göre ilgili soru, göstergede gösterilir.
updateUserProfile(questionNumber)	İlgili soruya ilişkin parametre değerlerinin "userprofile.dat" dosyasına kaydedilmesini sağlar. Her bir etmenin "getParameter" davranışı ile kendi parametre değerini öğrenmesi gerekmektedir.
resetParameters()	Tüm reaktif nesnelerin kendi parametrelerini sıfırlamak için "reset" davranışında bulunmasını sağlar.

8.4 Veri Toplama Amaçlı Olarak Hazırlanan Uygulama Programı

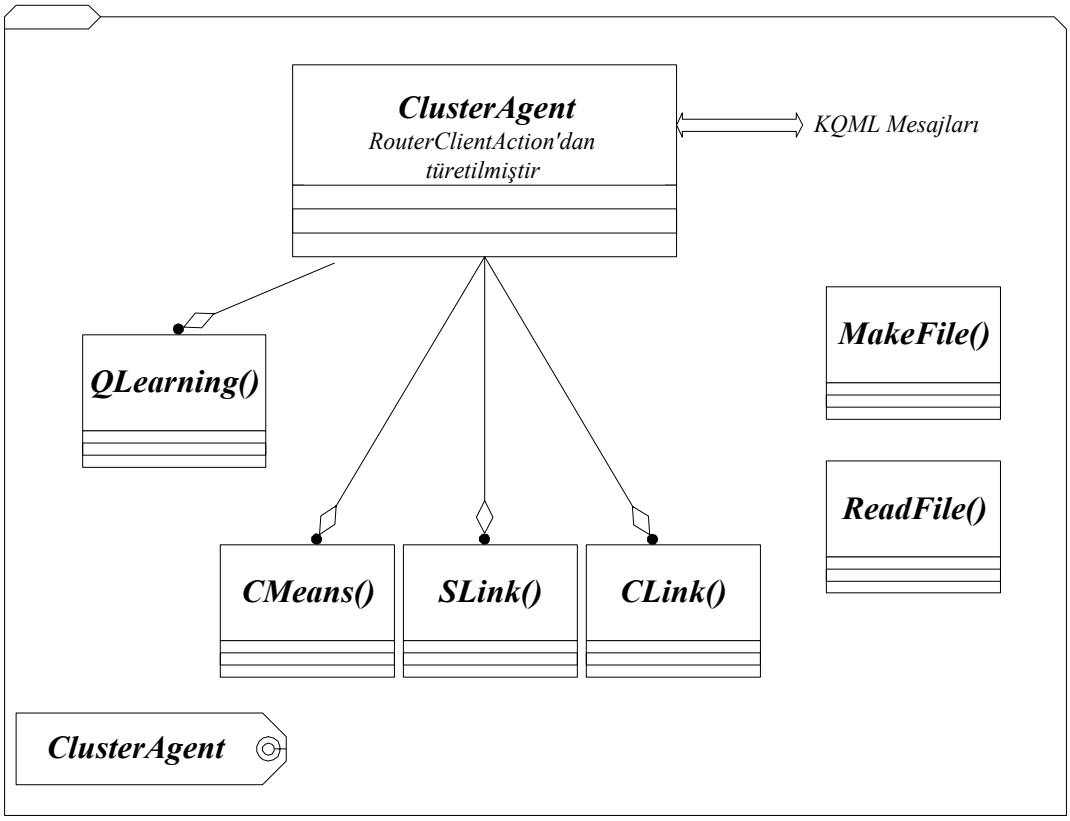
Uygulama Programı ITUStudents Sınıfı tarafından gerçeklenmiştir. Yazılımın şimdiye kadar anlatılan bölümüne ilişkin bileşenlerin bir kısmı kullanılarak veri toplama amaçlı olarak hizmet vermek üzere ITUStudents sınıfı oluşturulmuştur (Şekil 8.8). Bu program bir çoklu etmen sistemine dahil olmamaktadır. Dolayısıyla sadece AgentServer, UserInterface sınıfları ve ek olarak kayıtlar esnasında denetimi gerçekleyebilmek üzere ReadFile Sınıfı bu sınıfa dahil edilmiştir. Animation sınıfı da kullanıcıya örnek yüz ifadeleri oluşturmak üzere eklenmiştir. Animation sınıfı, önceden belirlenmiş parametre dizilerini Grafiksel Yüz Animasyon Programında gösterir. Her bir kullanıcı için farklı isim atamak üzere bir dosyada sayaç bilgisi tutulmuştur. Bazı öğrencilerin giriş esnasında programı açıp kapatma ve sayaç değerinin artmasına sebep olmasıyla boş kayıtlar yaratılmaktadır. Önceden kontrolü yapılmayan bu boş kayıtlarla ilgili problem "Transfer" sınıfı ile çözülmüştür. Bunun için "userprofile.dat" dosyasından boş kayıtlar silinir. Böylelikle bölütleme sürecinin doğru veriler üzerinde çalışması sağlanmıştır.



Şekil 8.8 ITUStudents Sınıfı

8.5 Bölütleme Etmeni Sınıfı

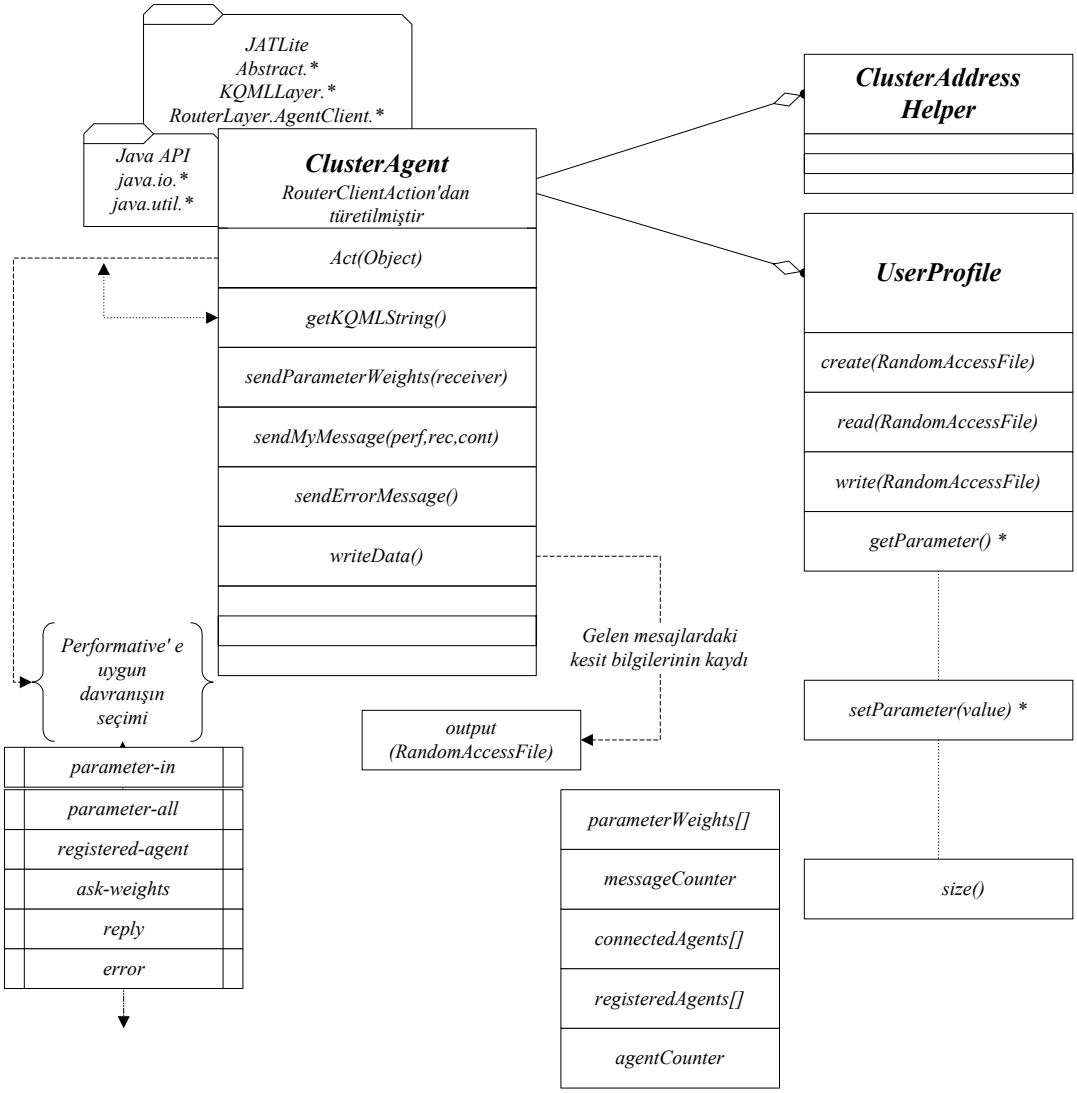
Bölütleme Etmeni çoklu etmen sistemindeki en akıllı birimdir. Bu birim, tüm kullanıcılardan gelen verileri toplayarak, bu veriler üzerinde bölütleme süreçlerini gerçekleyerek topluma ilişkin kesit bilgisini oluşturur. Bölütleme süreci öncesinde bu süreçte kullanacağı parametre ağırlıklarını tek tek kullanıcılar ve etmenleri ile etkileşim kurarak öğrenir. Bu etmen ClusterAgent sınıfı ile temsil edilir.



Şekil 8.9 ClusterAgent Sınıfı

8.5.1 ClusterAgent Sınıfı

ClusterAgent sınıfı QLearning, CMeans, SLink ve CLink sınıflarını yürütme esnasında gereken adımlarda uyandıran sınıftır. Şekil 8.9'da bu sınıfın sınıf hiyerarşisi görülmektedir. Gelen mesajları ve mesaj yollayan etmenlerin sayısını tutar. Gelen mesajlar "ClusterAll.dat" dosyasına kaydedilir. Yürütme esnasında bölütleme işlemi, kullanıcının istediği anlarda her üç algoritma için de gerçeklenebilir. Şekil 8.10'da ClusterAgent metotları ve değişkenleri görülmektedir.



Şekil 8.10 ClusterAgent Sınıf Hiyerarşisi

Tablo 8.8'de ClusterAgent sınıfının metotlarına ilişkin bilgiler verilmektedir. ClusterAgent sınıfı sisteme kayıt işlemini tamamladıktan sonra QLearning sınıfını uyandırır. Bu sınıf, Q Öğrenmesi yöntemi ile sistemdeki tüm kullanıcılar ile etkileşim kurarak yüz ifadelerinin her biri için parametre ağırlıklarını belirleyen sınıftır. ClusterAgent çoklu etmen sistemi içinde mesajlaşmayı sağladığından bu iki sınıfın ortak çalışması gerekmektedir. Qlearning sınıfı yollanacak yeni yüz ifadesini belirleyerek Q değerlerini güncellerken ClusterAgent sınıfı belirlenen yeni yüz ifadesinin yollanmasını ve buna karşılık puanın alınmasını sağlar. ClusterAgent sınıfı, QLearning sınıfından öğreme sürecinin sonlandığına ilişkin bir işaret alana dek bu ortak çalışmayı sürdürür. Bu esnada sistemde sadece parametre ağırlıklarının öğrenilmesine ilişkin mesajlar dolaşmaktadır. Eğer her hangi bir şekilde bir etmenden kesit bilgisine ilişkin mesaj gelirse, ClusterAgent bu mesaja ilişkin bilgileri de kaydedebilir. Öğrenme süreci sonlandıktan sonra parameterWeights

dizisi son halini almış olur. ClusterAgent, mesaj sayısı belli bir değere ulaştıktan sonra, bölütleme sürecini her üç sınıfı da uyandırarak gerçekler. Bölütleme sonuçları çeşitli dosyalarda saklanır. Bölütleme sınıfları ClusterAgent ve sistemdeki diğer Kullanıcı Arayüz Etmenlerinin kendilerine gelen veriler üzerinde bölüt analizi yapmalarını sağlar. Bölütlenecek olan veriler ilgili dosyalardan alınır. ClusterAgent bölütleme sınıfları verileri "ClusterAll.dat" dosyasından, AgentServer bölütleme sınıfları ise verileri "AgentGrades.dat" dosyasından alır. Bu dosyalardaki veriler Userprofile sınıfından türetilmiş olup rasgele erişilebilir kayıtlardan oluşmaktadır. Bölütleme sonuçları her bir bölütleme sınıfı için farklı dosyalara kaydedilir. Tablo 8.9'da ClusterAgent sınıfına gelen mesaj tipleri görülmektedir.

Tablo 8.8 ClusterAgent Sınıfı Metotları

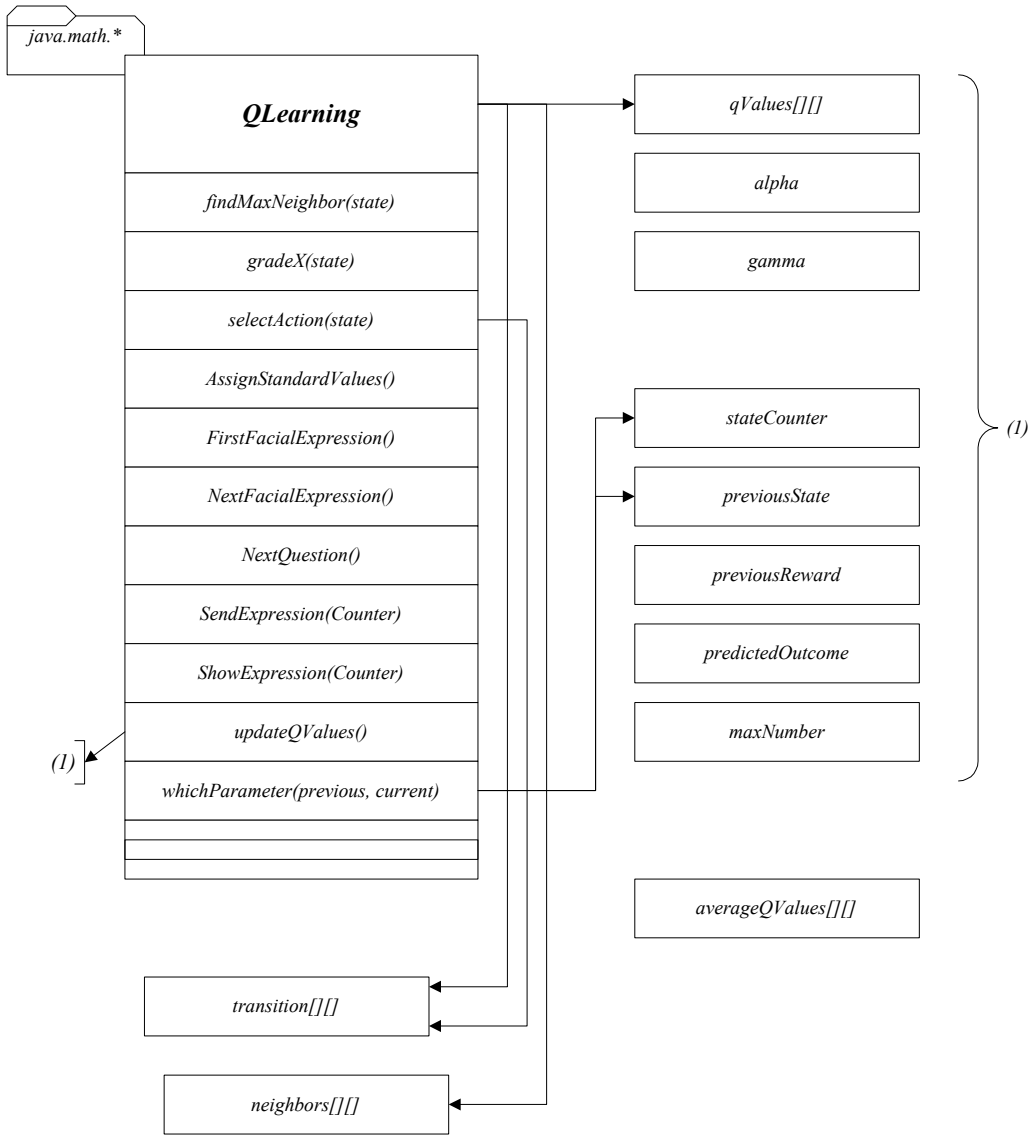
Metot	***Yürüttüğü İşlem***
Act (object o)	Bu metot RouterClientAction içinde tanımlanmıştır. Mesajları işler ve mesajın komutuna göre gerekli işlemleri yürütür.
GetKQMLString()	Yollanmak istenen mesajı KQML mesajı şeklinde paketler.
SendParameterWeights(receiver)	Yüz ifadeleri için parametre ağırlıklarını (öğrenme süreci sonunda belirlenen) yollar.
SendMyMessage(perf,receiver,cont,message)	GetKQMLString metodunu çağırarak mesajı paketler. Performative ve içeriğe göre mesajı alıcı birime yollar. Gerekli mesajı ekrana yazdırır.
SendErrorMessage()	Mesajı yollayan etmene hata mesajı yollar.
WriteData()	"parameter-in" ve "parameter-all" mesajları ile gelen yüz ifadeleri parametre bilgilerini "ClusterAll.dat" rasgele erişilebilir dosyasına kaydeder. Bu dosyadaki bilgiler daha sonra bölütleme sürecinde kullanılır.

Tablo 8.9 ClusterAgent Sınıfına Gelen Mesaj Tipleri

Mesaj Tipi	***Açıklaması ve Yürütülen Görev***
Parameter-in	Bir etmenin bir yüz ifadesine ilişkin olarak parametrelerini yolladığı mesaj tipidir. Gelen mesaj bilgisi çözülerek mesajın içeriği alınır ve writeData() metodu uyandırılır.
Parameter-all	Bir etmenin tüm yüz ifadelerine ilişkin olarak parametrelerini yolladığı mesaj tipidir. Gelen mesaj bilgisi çözülerek her bir yüz ifadesi için mesajın içeriği alınır ve writeData() metodu uyandırılır.
Registered-agent	Yönlendiriciden gelen mesaj tipidir. Bu mesaj ile sistemdeki tüm etmenler, adres bilgileri ve o anda sisteme bağlı olup olmadıkları bilgileri alınır. registeredAgents ve connectedAgents dizileri bu şekilde oluşturulur. Bu mesaj, yönlendiriciye "list-agent" mesajının yollanmasından sonra gelmektedir.
Ask-weights	Bu mesaj, bir etmenin yüz ifadelerine ilişkin parametre ağırlıklarını öğrenmek için yolladığı mesaj tipidir. SendParameterWeights(..) metotu çağrılarak parameterWeights dizisi karşı mesaj (Reply-weights) olarak yollanır.
Reply	Etmenin mesajına karşılık yollanan mesajdır. Mesaj istenen bilgiye göre işlenir. Öğrenme sürecindeki puan bilgisi de bu mesajla gelir.
Error	Etmenin yollamış olduğu bir mesaja ilişkin hata mesajıdır. Mesaj gerekiyorsa tekrarlanır.

8.5.2 QLearning Sınıfı

ClusterAgent' in öğrenme süreci için kullandığı sınıftır. Şekil 8.11'de QLearning sınıfı ve kullandığı değişkenler görülmektedir. Tablo 8.10'da QLearning Sınıfı metotları açıklanmıştır.



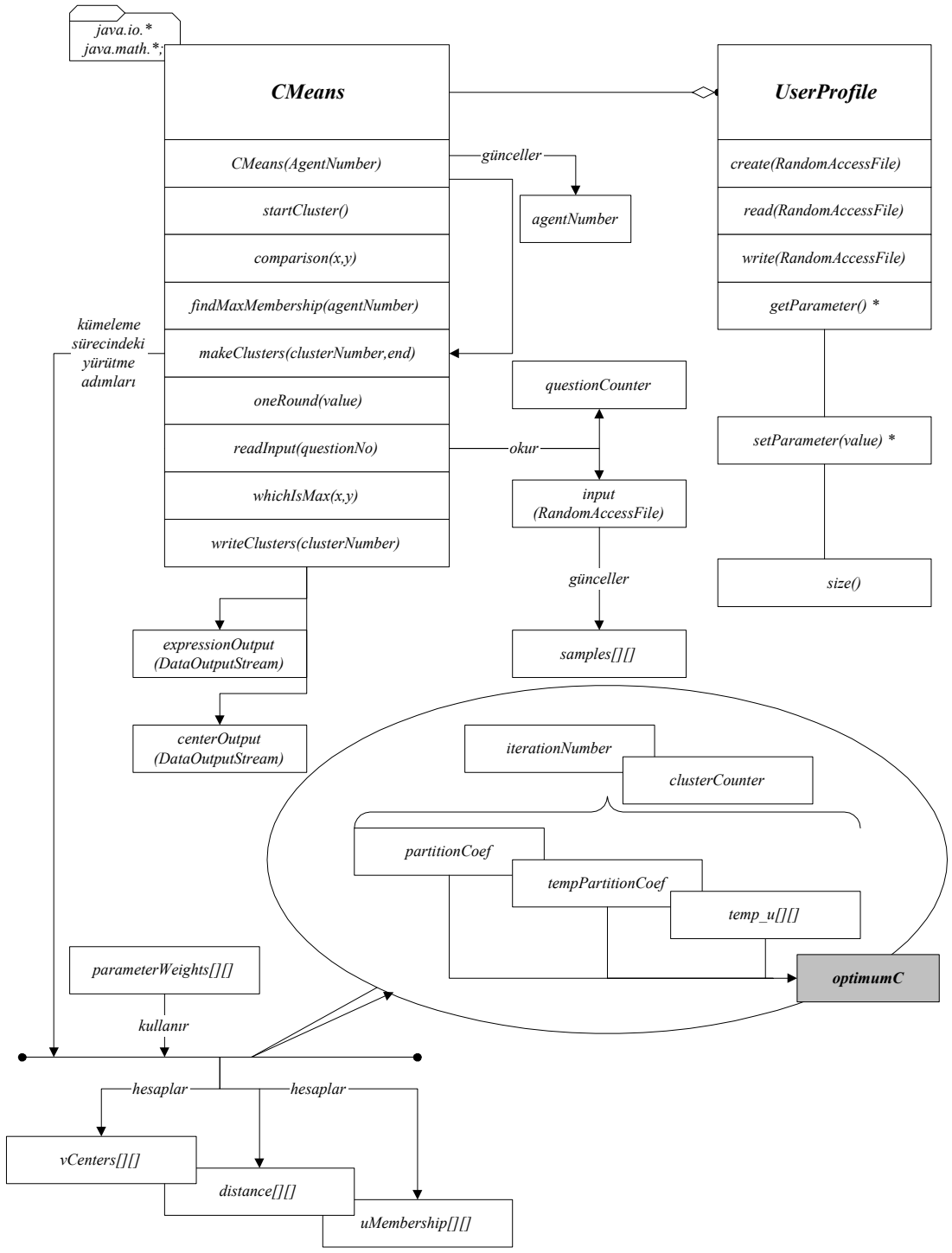
Şekil 8.11 QLearning Sınıf Hiyerarşisi

Tablo 8.10 QLearning Sınıfı Metotları

Metot	***Yürüttüğü İşlem***
findMaxNeighbor(state)	Bu metot, bir durumun komşuları arasında en yüksek Q değerine sahip olan durumu döndürür.
gradeX(state)	Simülasyon programında QLearning sınıfı tarafından her adımda üretilen yüz ifadesi için puan üretir.
AssignStandardValues()	Öğrenme sürecini yürütmeyip önceden tanımlanan değerleri yüklemek için kullanılır (Sistemin diğer bölümlerinin çalışmasını gözlemlemek amacıyla eklenmiştir.).
FirstFacialExpression()	İlk ifadede puan hesaplaması yapılmamaktadır. Bu metot ilk ifade ile ilgili duruma ilişkin mesajın yollanmasına sebep olur.
NextFacialExpression()	Puan hesaplamaları ve mevcut Q güncellemelerinin yapılarak durum değişiminin gerçeklemesi ve yeni duruma ilişkin yüz ifadesinin yollanması için gerekli işlemleri yürütür.
selectAction(state)	Öğrenme sürecinde bir sonraki adımda hangi davranışın seçilmesi gerektiğini belirler. Bu da yeni duruma karşılık düşer.
updateQValues()	Mevcut durum geçişi için alınan ödül (önceki ifadeye verilen puan ve mevcut ifadeye verilen puan ile tahmin edilen değişimin bir fonksiyonu olarak Tablo 7.3'e göre belirlenir.) önceki Q değeri, sonraki durumun maksimum Q değerine sahip geçişinin bir fonksiyonu olarak Formül 3.15' e göre belirlenir.
whichParameter(previous,current)	Tahmin edilen değişimin güncellenmesi için durumlar arası parametre değerlerini denetler ve yeni değeri döndürür.
SendExpression(counter)	İlgili yüz ifadesinin bir etmene yollanmasını sağlar.
Show Expression(counter)	Bir etmene yollanan yüz ifadesinin Grafiksel Yüz Animasyon Programında görülebilmesini sağlar (Bu özellik isteğe bağlı olarak kullanılabilir, sistemin çalışmasını etkilemez.).

8.5.3 CMeans Sınıfı

CMeans Sınıfı bölütleme sürecinde FCM algoritmasını kullanarak bölütleme yapan sınıftır. Bu sınıfa ilişkin değişkenler ve metotlar Şekil 8.12'de verilmiştir.



Şekil 8.12 CMeans Sınıf Hiyerarşisi

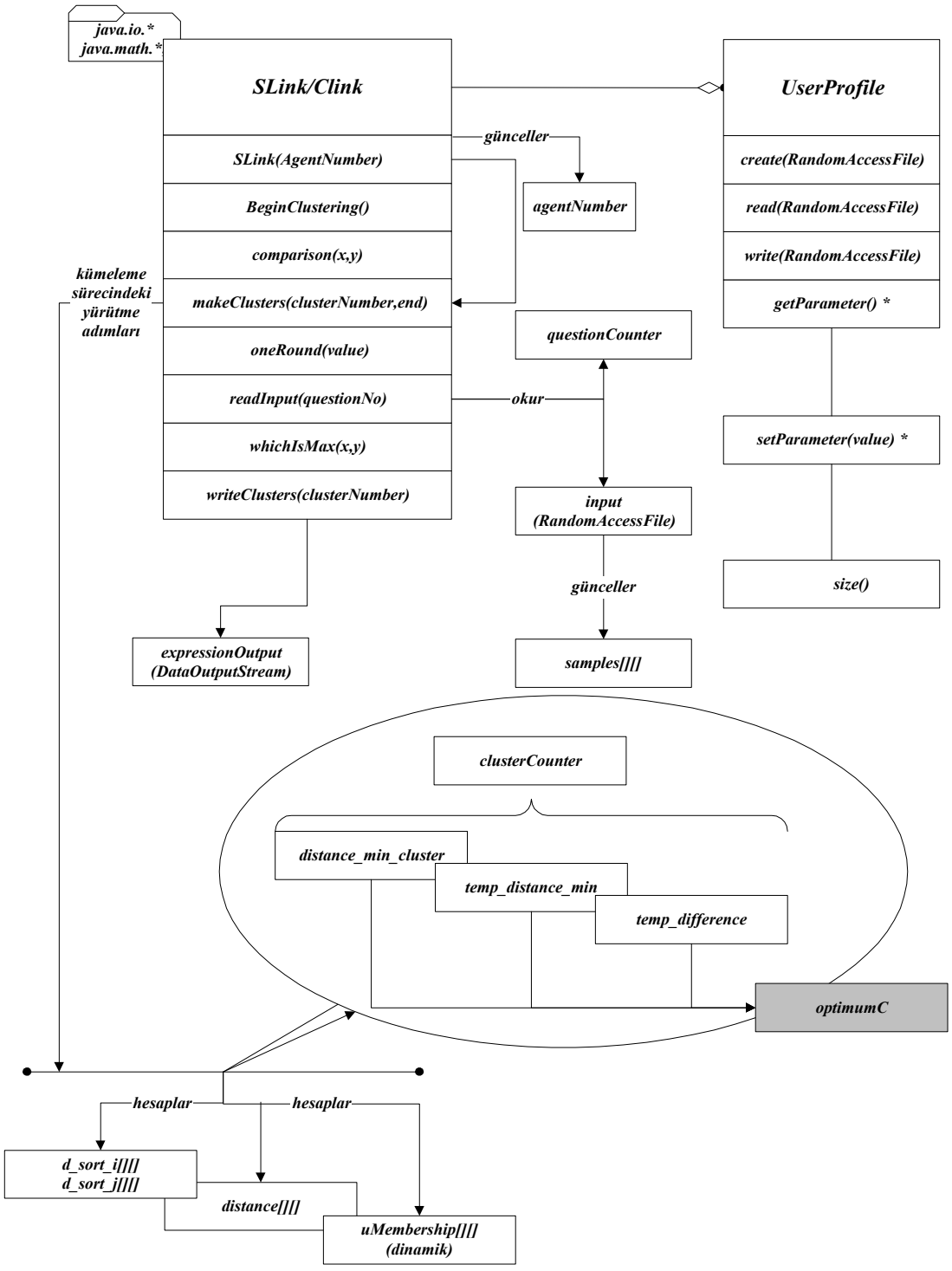
Bölütleme sonuçları hem bölüt merkezleri hem de verilerin hangi bölütlere dahil olduğu bilgileri ile oluşur. Bu sonuçlar Exp_i ve center_i (i, ifade numarasını belirtir.) dosyalarına kaydedilir. Cmeans Sınıfı metotları ve yürütülen işlemler Tablo 8.11'de verilmiştir.

Tablo 8.11 CMeans Sınıfı Metotları

Metot	***Yürütülen İşlem***
CMeans(AgentNumber)	Cmeans sınıfının kurucu metodu olup kedisini uyandıran sınıftan aldığı parametre ile AgentNumber değişkenini günceller. Bu değişken bölütlenecek veri sayısını belirler.
startCluster()	Bölütleme işlemini başlatır. Her bir ifade ve bölüt sayısı için MakeCluster() metodunu çağırır. Sonunda optimum bölüt sayısı bulunur.
findMaxMembership(agentNumber)	Bir veri örneğinin en yüksek üyelikle hangi bölüte dahil olduğunu belirler.
makeClusters(clusterNumber,end)	Parametre olarak girilen bölüt sayısı için bölütleri oluşturur.
readInput(questionNo)	Her bir soru için bölütlenecek veri örneklerini ilgili dosyadan okur.
writeClusters(clusterNumber)	Sonuç bölütlerini hem ekrana yazdırır hem de sonuçları tutan dosyalara yazdırır.

8.5.4 SLink ve CLink Sınıfları

SLink Sınıfı bölütleme sürecinde Single-link algoritmasını kullanarak bölütleme yapan sınıftır. CLink Sınıfı bölütleme sürecinde Complete-link algoritmasını kullanarak bölütleme yapan sınıftır. Bölütleme sonuçları dendogramlar şeklinde oluşmaktadır. Dendogram uygun şekilde kesilip optimum bölüt sayısı belirlendiğinden verilerin hangi bölütlere dahil olduğu bilgileri çıkışa yansıtılır. Bu sonuçlar Exp_i ve (i, ifade numarasını belirtir.) dosyalarına kaydedilir (Slink ve CLink sonuçları farklı olmaktadır.). Şekil 8.13'te SLink ve CLink sınıf metotları ve değişkenleri görülmektedir.



Şekil 8.13 SLink (Clink) Sınıf Hiyerarşisi

IV. bölümde Single-link ve Complete-link algoritmalarının farklılıklarından bahsedilmiştir. Bu farklılık, kullanılan veri yapılarında veya metotlarda bir değişiklik yaratmaz. Tablo 8.12'de SLink ve CLink sınıfları metotları tanıtılmaktadır.

Tablo 8.12 SLink ve CLink Sınıfları Metotları

Metot	*Yürütülen İşlem*
SLink/CLink(AgentNumber)	SLink/CLink sınıfının kurucu metodu olup kedisini uyandıran sınıftan aldığı parametre ile AgentNumber değişkenini günceller. Bu değişken bölütlenecek veri sayısını belirler.
startCluster()	Bölütleme işlemini başlatır. Her bir ifade ve bölüt sayısı için MakeCluster() metodunu çağırır. Sonunda optimum bölüt sayısı bulunur.
makeClusters(clusterNumber,end)	Parametre olarak girilen bölüt sayısı için bölütleri oluşturur. Algoritma gereği tüm bölüt sayıları denenmektedir.
readInput(questionNo)	Her bir soru için bölütlenecek veri örneklerini ilgili dosyadan okur.
writeClusters(clusterNumber)	Sonuç bölütlerini hem ekrana yazdırır hem de sonuçları tutan dosyalara yazdırır.

8.6 Dosyalar

Sistemde denetlenen ve yaratılan dosyalara ilişkin bilgiler Tablo 8.13'te verilmektedir.

Tablo 8.13 Sistemdeki Dosyalar

Dosya Adı	Açıklaması
Name.dat	ITUStudents Sınıfı için kullanıcı sayacını tutan dosya
Expression-macros.dat	Grafiksel Animayon Paketi için parametreleri tutan dosya
UserProfile.dat	Kullanıcı kesitlerini tutan dosya
AgentGrades.dat	Diğer kullanıcı çizimlerine ait bilgilerin tutulduğu dosya
ClusterAll.dat	Bölütleme Etmenine gelen kesit bilgileri tutan dosya
Center_i.dat (i.ifade)	Bölütleme sonucunda oluşan bölüt merkezlerini tutan dosya
Exp_i.dat	i. ifade için bölütlere dağılım bilgisini tutan dosya
AgentAddressFile	Etmen adres bilgilerini tutan dosya
ClusterAddressFile	Bölütleme Etmeni adres bilgilerini tutan dosya
<etmenAdı>.incoming	JATLite paketinin <etmenAdı>' na ilişkin mesaj dosyası
Semaphor.txt	Yüz ifadesinde güncelleme yapıldığını ifade eden dosya

9. UYGULAMA

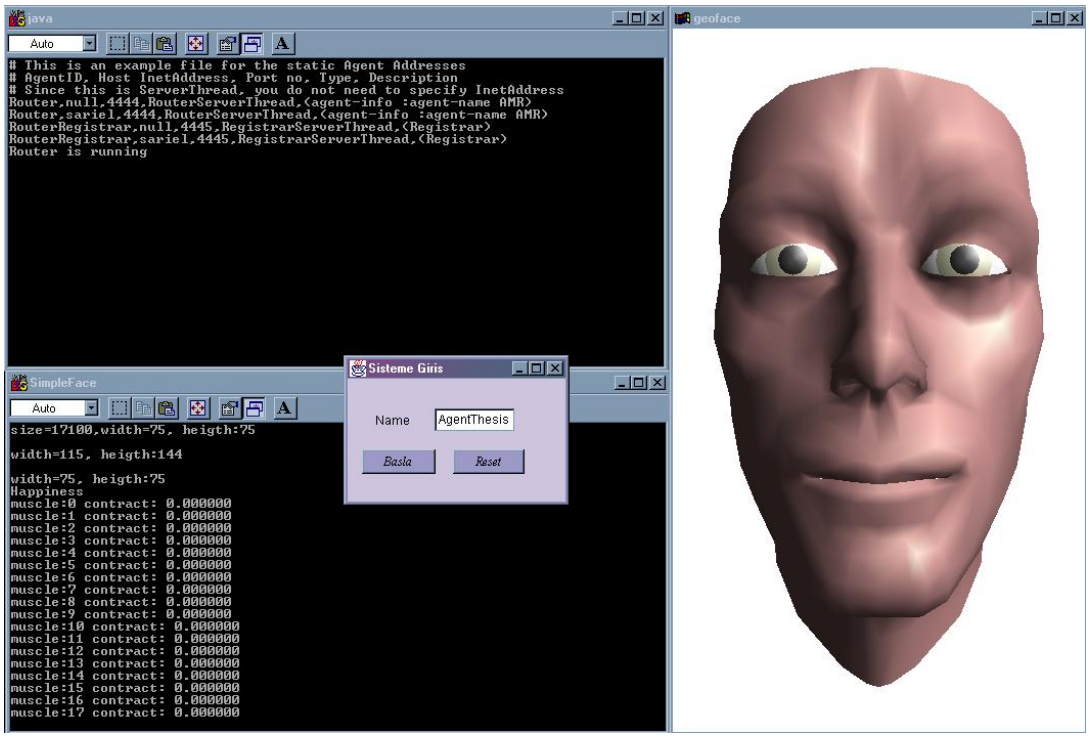
9.1 Giriş

Bu bölümde, gerçeklenen yazılım sisteminin yapılan bir uygulaması tanıtılmakta ve sonuçlar incelenmektedir. Bu sonuçlar, uygulamanın az sayıda bölüm öğrencisi üzerinde denenmesi sonucu toplanan veriler üzerinde bölütleme algoritmalarının gerçeklenmesi ile elde edilmiştir. Öğrenme süreçleri, simülasyon ortamında denenmiş ve çalışması gözlenmiştir.

9.2 Yazılım Uygulaması

Gerçeklenen yazılım için, kullanıcı çizimlerinin yapılmasına, etmen içi gerçekleşen olayların, çoklu etmen sistemi dahilinde gerçekleşen olayların izlenmesine imkan tanıyan, kullanıcı kesit bilgileri ile bölütleme sonuçlarının görülebildiği bir yazılım arayüzü tasarlanmıştır. Tasarlanan sistemde Kullanıcı Arayüz Birimi ve Bölütleme Etmeni farklı programlardır. Bölütleme ve öğrenme süreçleri menülere ilişkin bilgiler Bölütleme Etmeni ekranından tanıtılacaktır. Tanıtılan diğer ekranlar Kullanıcı Arayüz Birimine aittir. “Etmen İçi“ menüsü, gerçeklenen reaktif nesneler ve Kullanıcı Arayüz Etmeni (Reaktif Nesne Hizmetlisi) arasındaki iletişimi görüntüler. “Çizim Ekranı” menüsü, kullanıcının yüz ifadeleri için çizimlerini oluşturduğu menüdür. “MAS Monitörü” menüsü, etmenin çoklu etmen sistemi içinde aldığı mesajları ve mesajların işlenme durumlarını görüntüler. “Kesit Ekranı” menüsü, kullanıcı kesitine ilişkin parametrelerin görüntülendiği ekrandır. “Bölütleme Sonuçları” menüsü, bölütleme süreci sonuçlarının görüntülendiği ekrandır. “Yardim” menüsü uygulamanın kullanımına ilişkin bilgi veren menüdür.

Sistemin ilk açılışında JATLite Yönlendiricisi hizmet vermeye başlar. Kullanıcı Arayüz Programı ile birlikte Grafiksel Yüz Animasyon Programı da etkinleşir. Kullanıcı Arayüz Programında ilk önce etmen adı bilgisi alınmaktadır. Açılış durumunda Yönlendiricinin de aynı makinede çalıştırılması durumunda görülen ekran görüntüsü Şekil 9.1’de verilmiştir.

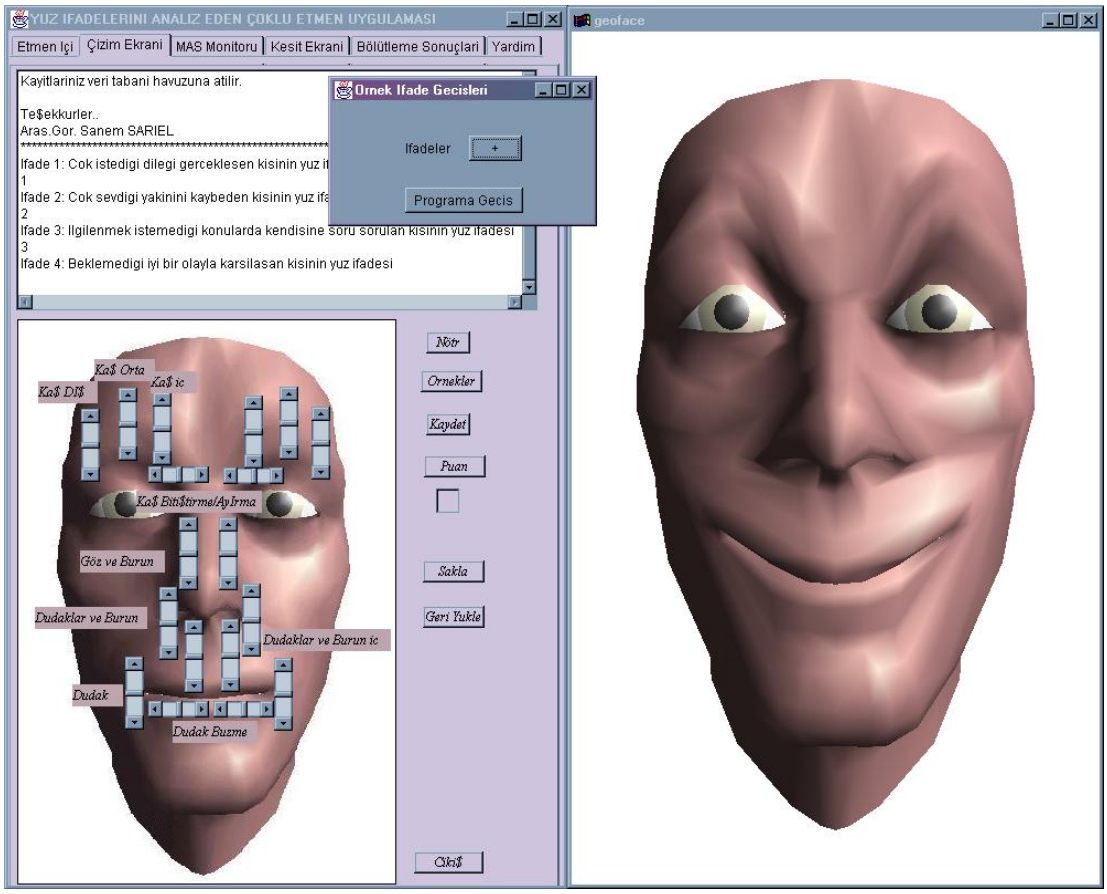


Şekil 9.1 Sistem Açılışında Ekran Görüntüsü

Kullanıcı, ifadeler için çizimlerini “Çizim Ekranı” menüsündeki sürgüleri kullanarak gerçekler. İfadelere ilişkin sorular Tablo 9.1’de verilmiştir. İfadeyi nötr hale getirmek ve daha önceden kaydedilmiş bazı örnek çizimleri görmek mümkündür. Sorulara ilişkin çizimlerin kaydedilmesi, bir sonraki soruya geçilmesine sebep olur. Kullanıcı Arayüz Etmenine gelen (eğer mesaj geldiyse) diğer kullanıcıların çizimleri için “Puan” butonu kullanılarak çizim görüldükten sonra puan verilebilir. “Çizim Ekranı” menüsü ekran görüntüsü Şekil 9.2’de verilmiştir.

Tablo 9.1 İfadeler için Belirlenen Sorulara İlişkin Tablo

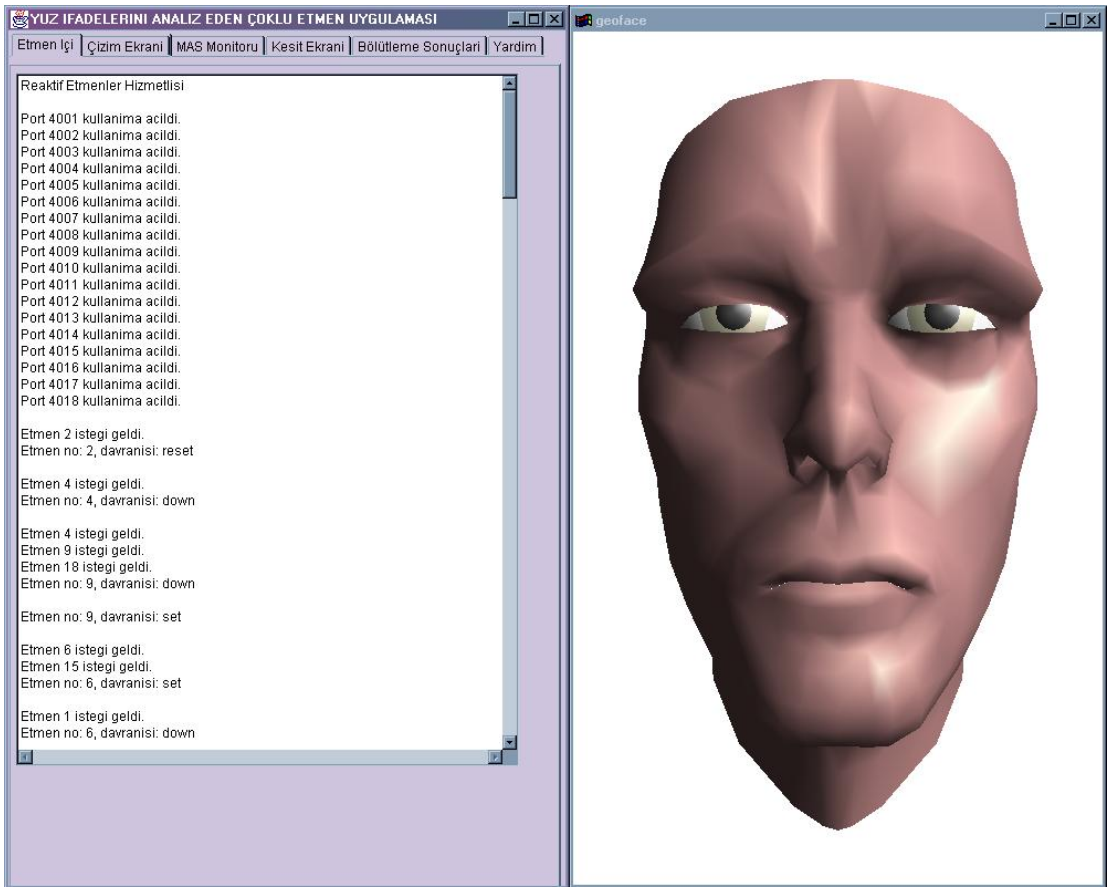
İfade No	**İfade İçin Belirlenen Soru**
İfade 1	Çok istediği dileği gerçekleşen kişinin yüz ifadesi
İfade 2	Çok sevdiği yakınını kaybeden kişinin yüz ifadesi
İfade 3	İlgilenmek istemediği konularda kendisine soru sorulan kişinin yüz ifadesi
İfade 4	Beklemediği iyi bir olayla karşılaşan kişinin yüz ifadesi
İfade 5	Beklemediği kötü bir olayla karşılaşan kişinin yüz ifadesi
İfade 6	Geleceği hakkında hayal kuran kişinin yüz ifadesi
İfade 7	Tüm emekleri boşa giden kişinin yüz ifadesi
İfade 8	Sevdiği kişiye bakmakta olan kişinin yüz ifadesi
İfade 9	Değer verdiği önemli bir insanın yanında büyük hata yapan kişinin yüz ifadesi
İfade 10	Çok sinirlenen bir kişinin yüz ifadesi



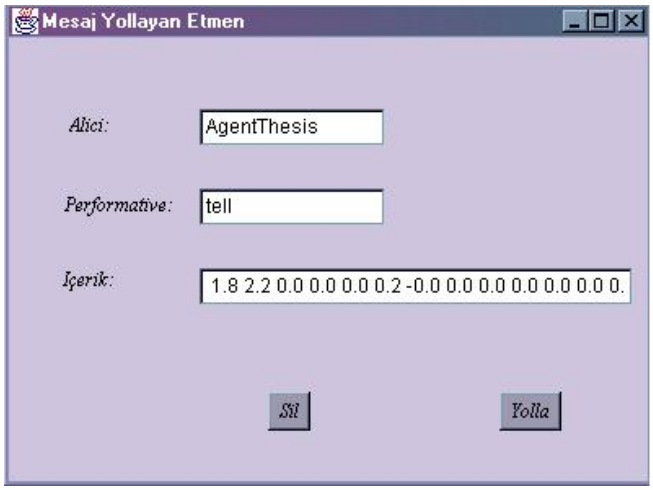
Şekil 9.2 “Çizim Ekranı” Menüsü

“Etmen İçi” menüsü, gerçeklenen reaktif nesneler ve Kullanıcı Arayüz Etmeni (Reaktif Nesne Hizmetlisi) arasındaki iletişimi görüntüler. Bu menü görüntüsü Şekil 9.3’te verilmektedir. Nesneler sırasıyla kendilerine ilişkin yüz parametresini güncellemişlerdir.

“MAS Monitörü” menüsü, etmenin çoklu etmen sistemi içinde aldığı mesajları görüntüler. Sistem içindeki Yönlendirici, Bölütleme etmeni veya diğer Kullanıcı Arayüz Etmenlerinden mesaj gelebilir. Gelen tüm mesajlar (tipleri, içerikleri ve işlenme durumları) bu ekranda görüntülenmektedir. Şekil 9.4’te sadece mesaj yollamak üzere gerçeklenen bir etmenden mesaj yollamak için girilen bilgiler görülmektedir. Ontoloji ve dil tanımı sistem içinde değişmediğinden program içinde sabitlenmiştir. Şekil 9.5’te “MAS Monitörü” Ekranı görülebilir. Farklı Test etmenlerinden (Kullanıcı arayüz etmenleri olarak gerçeklenmiştir.) gelen mesajlar görülmektedir. Ayrıca bu ekranda sistem tarafından etmene rasgele olarak atanan etmen etkilenme katsayısı yansımaktadır.

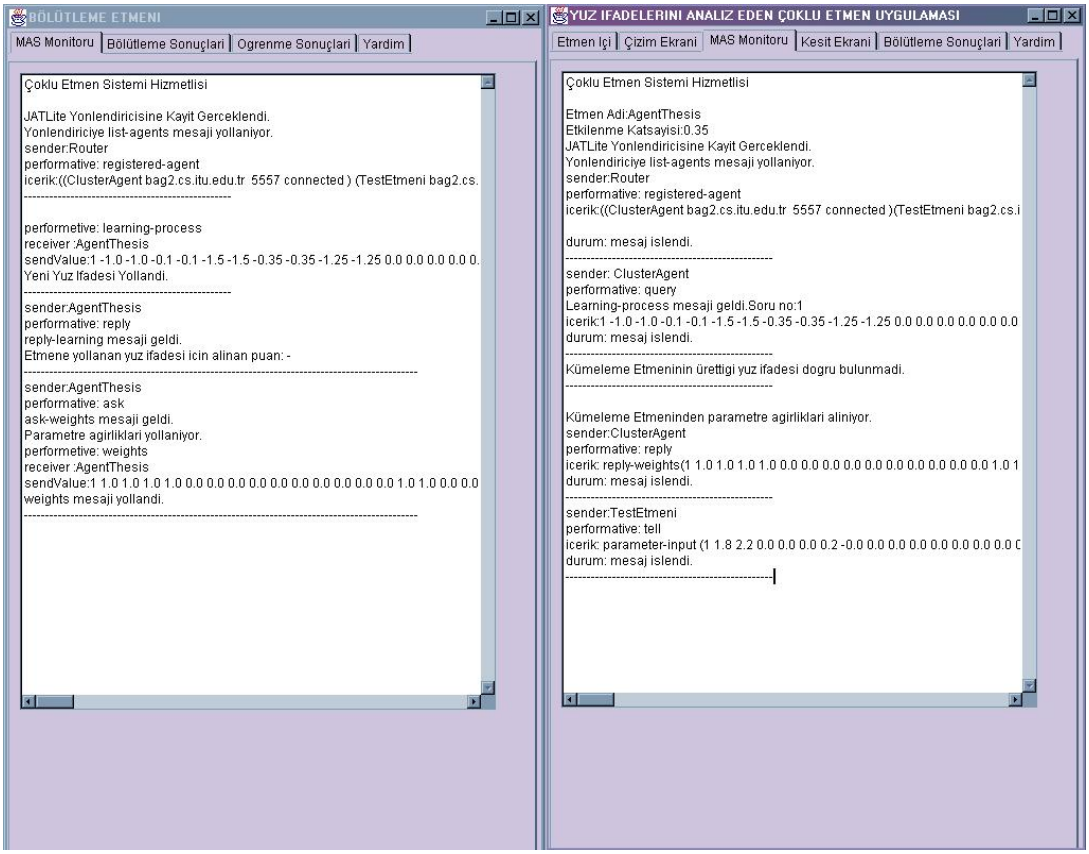


Şekil 9.3 “Etmen İçi” Menüsü



Şekil 9.4 Sadece Mesaj Yollamak İçin Gerçeklenen Etmen

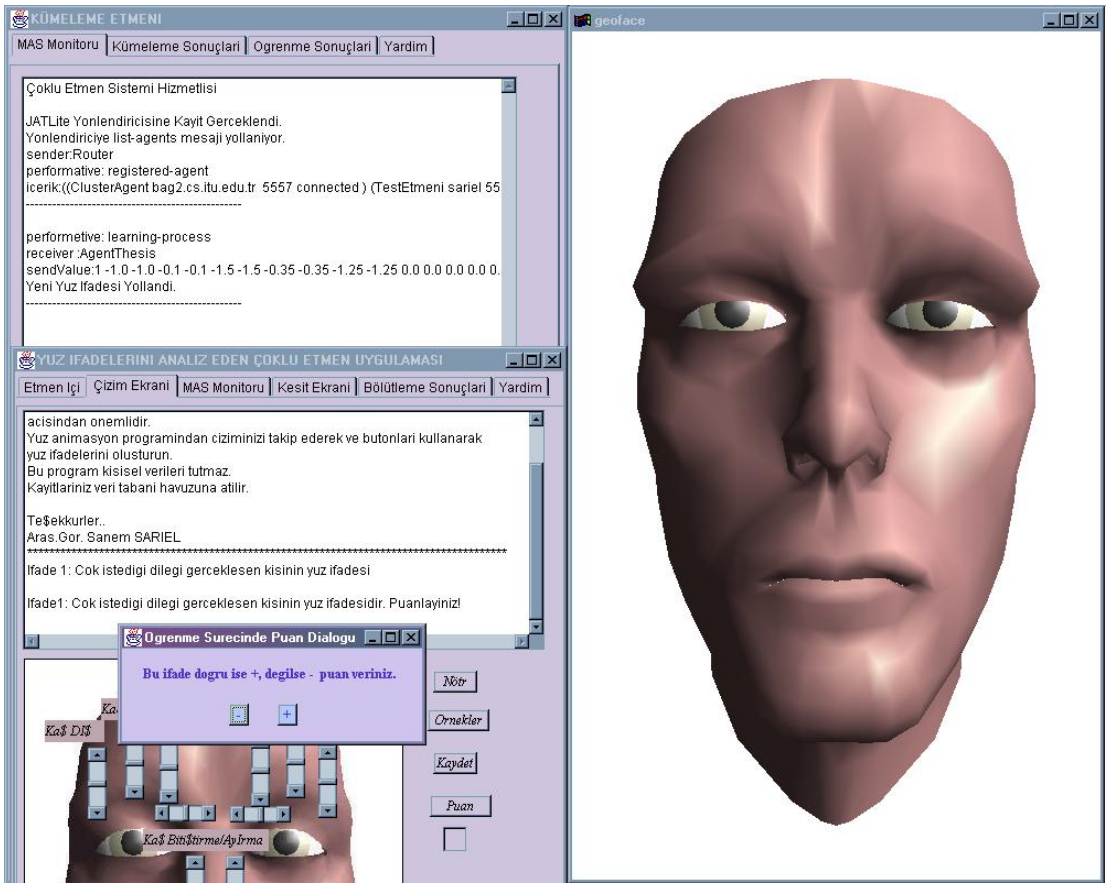
Parametre etkinlik dizilerini öğrenme sürecinde Kullanıcı Arayüz Etmeni ile Bölütleme etmeni arasındaki etkileşim sürecini gösteren ekran görüntüleri Şekil 9.6’da görülmektedir. Bölütleme etmeni ürettiği yüz ifadesini sisteme bağlı bir Kullanıcı Arayüz Etmenine yollar. Kullanıcı Arayüz Etmeninde gelen yüz ifadesi kullanıcıya gösterilerek kullanıcıdan puan vermesi beklenmektedir.



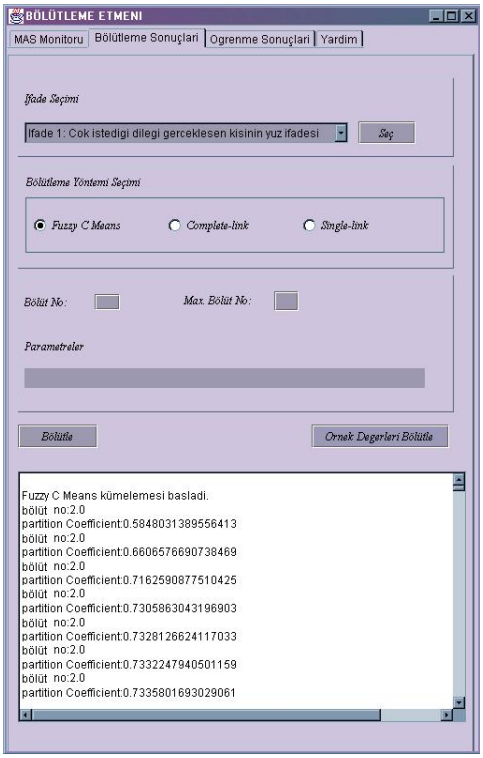
Şekil 9.5 “MAS Monitörü” Menüsü

“Kesit Bilgisi” menüsü, kullanıcı kesitine ilişkin parametrelerin görüntülendiği ekrandır. Bu ekranda kullanıcının yaptığı çizimler ve bunlara ilişkin parametreler ile diğer kullanıcıların çizimleri görülebilir. Parametreler görüntülenirken çizilmiş yüz ifadesini de kullanıcıya göstermek üzere gerekli dosya güncellemeleri yapılır. Böylelikle kullanıcı, çizimleri Grafiksel Yüz Animasyon Programından takip edebilir.

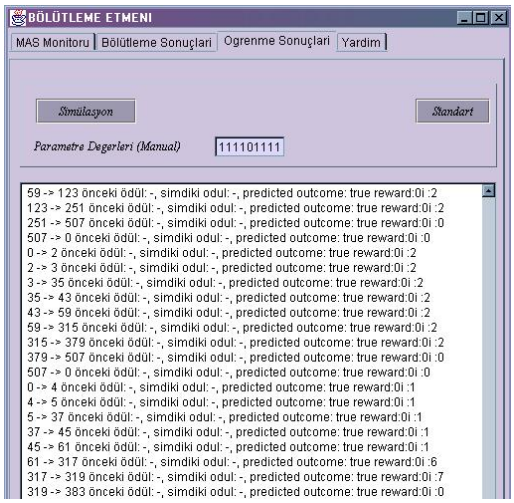
“Bölütleme Sonuçları” puanlama ve bölütleme süreçleri sonucu oluşan bölütlerin görüntülenebildiği ekrandır. Bu ekranda istenilen bölütleme algoritması seçimi yapılarak bölütleme sonuçları Grafiksel Yüz Animasyon Programında yüz ifade geçişleri (animasyon) şeklinde gözlemlenebilir. Şekil 9.7’de Bölütleme Etmeninin “Bölütleme Sonuçları” ekranı tanıtılmaktadır (Tüm etmenlerde aynı şekilde gerçeklenmektedir.). Bölütleme Etmeni öğrenme sürecini gerçeklediğinden dolayı bu sürece ilişkin bir menü bulunmaktadır. Şekil 9.8’deki ekran görüntüsünde öğrenme süreci simulasyonu esnasında ara adımlar görülebilir.



Şekil 9.6 Öğrenme Sürecinde Bölütleme Etmeni ile Kullanıcı Arayüz Etmeni Etkileşimi



Şekil 9.7 “Bölütleme Sonuçları” Menüsü (Bölütleme Etmeni)

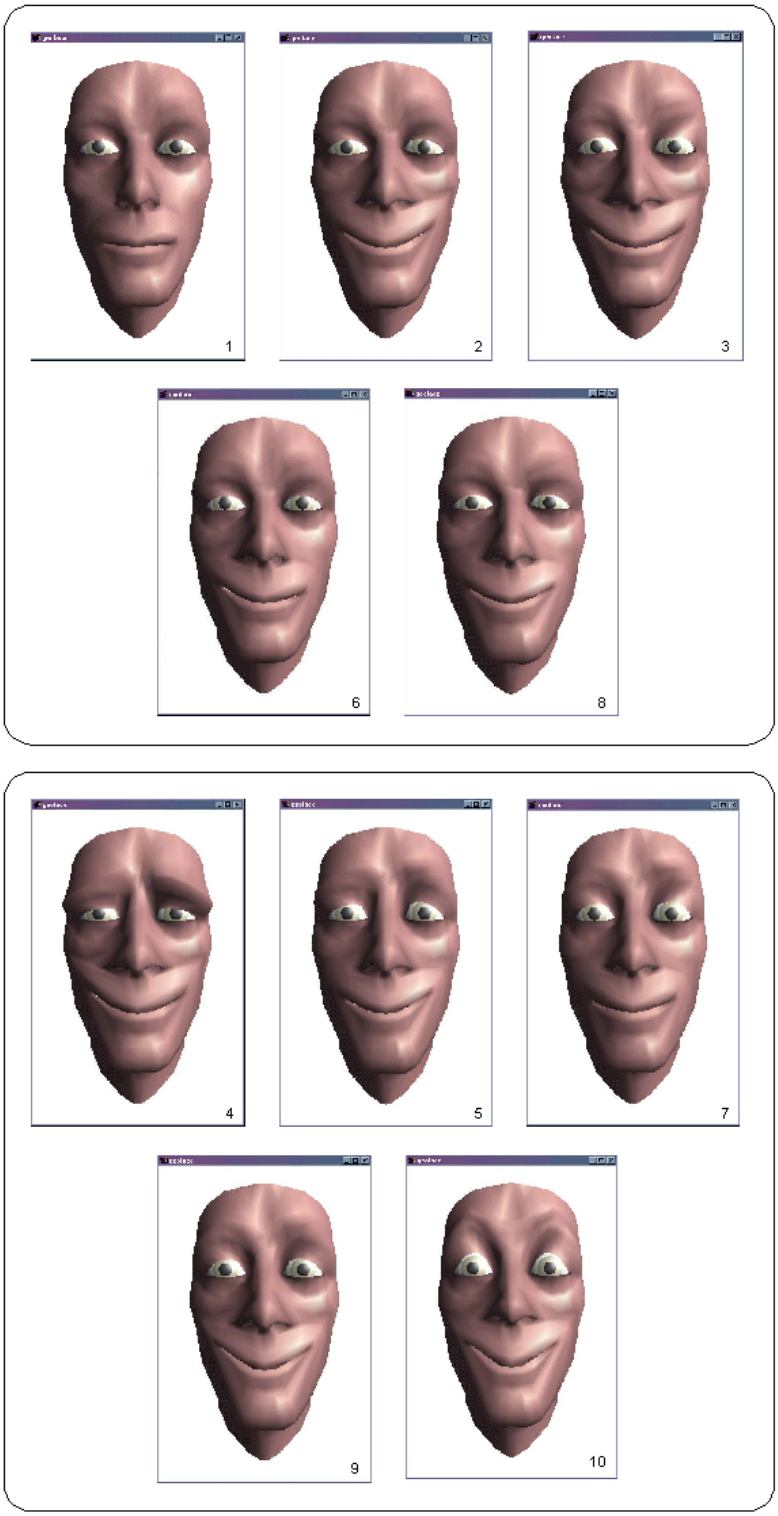


Şekil 9.8 “Öğrenme Sonuçları” Menüsü (Bölütleme Etmeni)

Öğrenme süreci sonunda algoritma en yüksek değerli durumları bulmaktadır. Bu durumların karşılıkları da simülasyon ortamında öğrenilen parametre etkinlik dizileri olmaktadır.

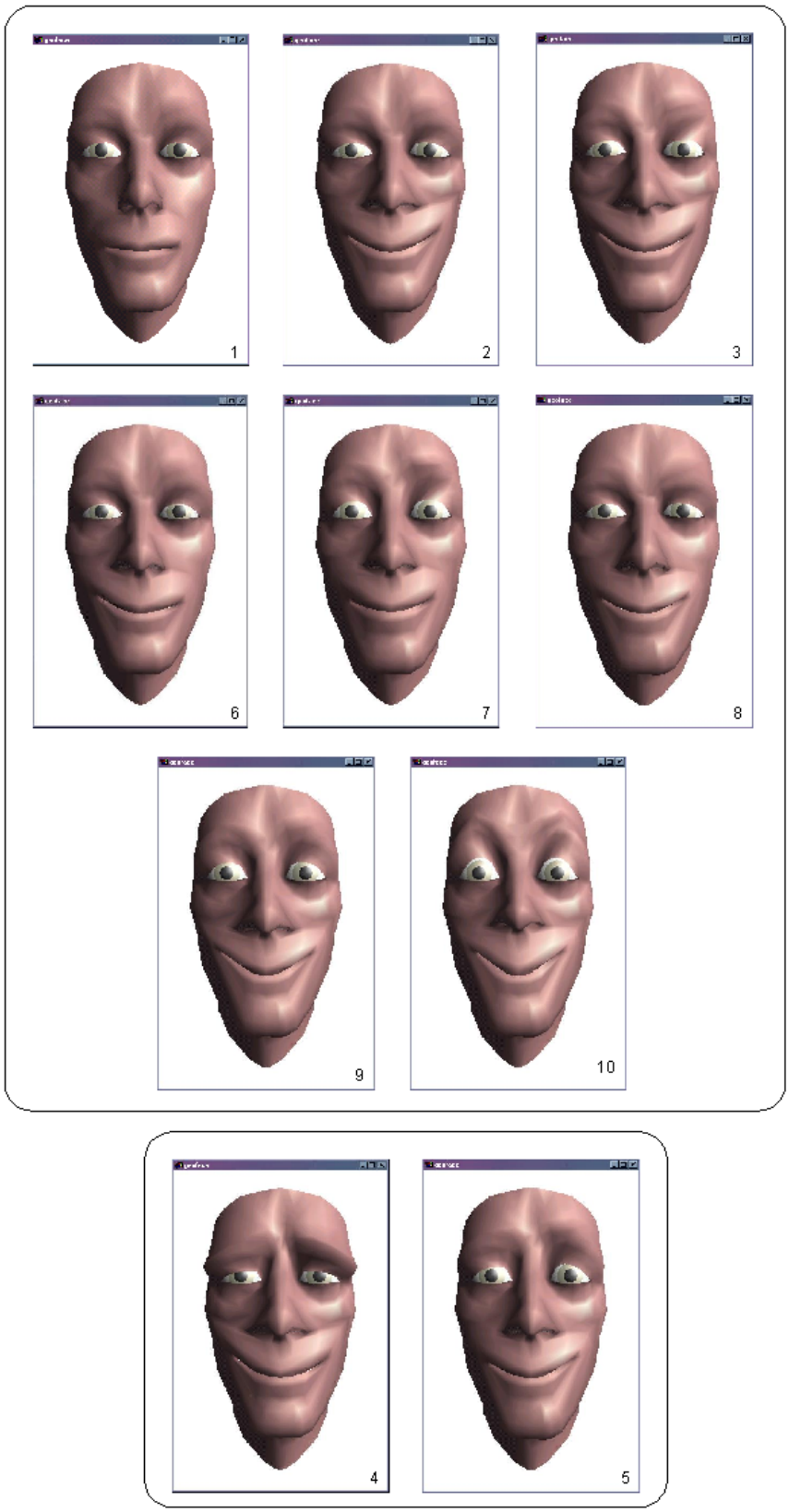
9.3 Uygulama Sonuçları

Bölümümüzün 3. ve 4. Sınıf lisans öğrencileri (20-22 yaşlar arası) için denenen uygulama programı, çizimlerin yapılabilmesi için 10 adet sorunun sorulduğu “Çizim Ekranı” menüsünü içermektedir. Sistemin ürettiği sonuçları göstermek için tezde sadece tek ifade ile ilgili sonuçlar yer almıştır. Doğal olarak bölütlemede sağlıklı sonuçlar almak için daha fazla örnek verinin olması gereklidir. Ancak tezde yüz şekillerini anlamlı sayıda gösterebilmek için bu yol seçilmiştir. Bu veriler üzerinde her üç bölütleme algoritması da uygulanmıştır. Şekil 9.9 FCM Algoritması sonuçlarını, Şekil 9.10 Complete-link algoritması sonuçlarını ve Şekil 9.11 Single-link algoritması sonuçlarını göstermektedir (Complete-link ve Single-link algoritmaları veriler arası uzaklıkları değerlendirdiğinden bu algoritmalarda ilk önce parametre normalizasyonu yapılmamıştır.). Parametre ağırlıklarının sistem sonuçlarına etkisini gözlemlemek üzere parametre ağırlıkları da kullanılarak bölütleme süreci gerçeklenmiştir. Parametre ağırlıklarının öğrenilmesi süreci, bir simülasyon programı ile test edilmiştir. Bu yüzden de sisteme uygulanan parametre ağırlıklarının öğrenilmesi adımları izlemiştir. Bölütleme süreci sonuçlarını alabilmek için parametre ağırlıkları sisteme giriş olarak verildiğinde gözlenen sonuçlar Şekil 9.12, 9.13 ve 9.14’te verilmiştir. Bu şekiller sırasıyla FCM, Complete-link, Single-link algoritmalarına (Tüm algoritmalarda normalizasyon yapılmıştır.) ilişkin sonuçları gösterir.



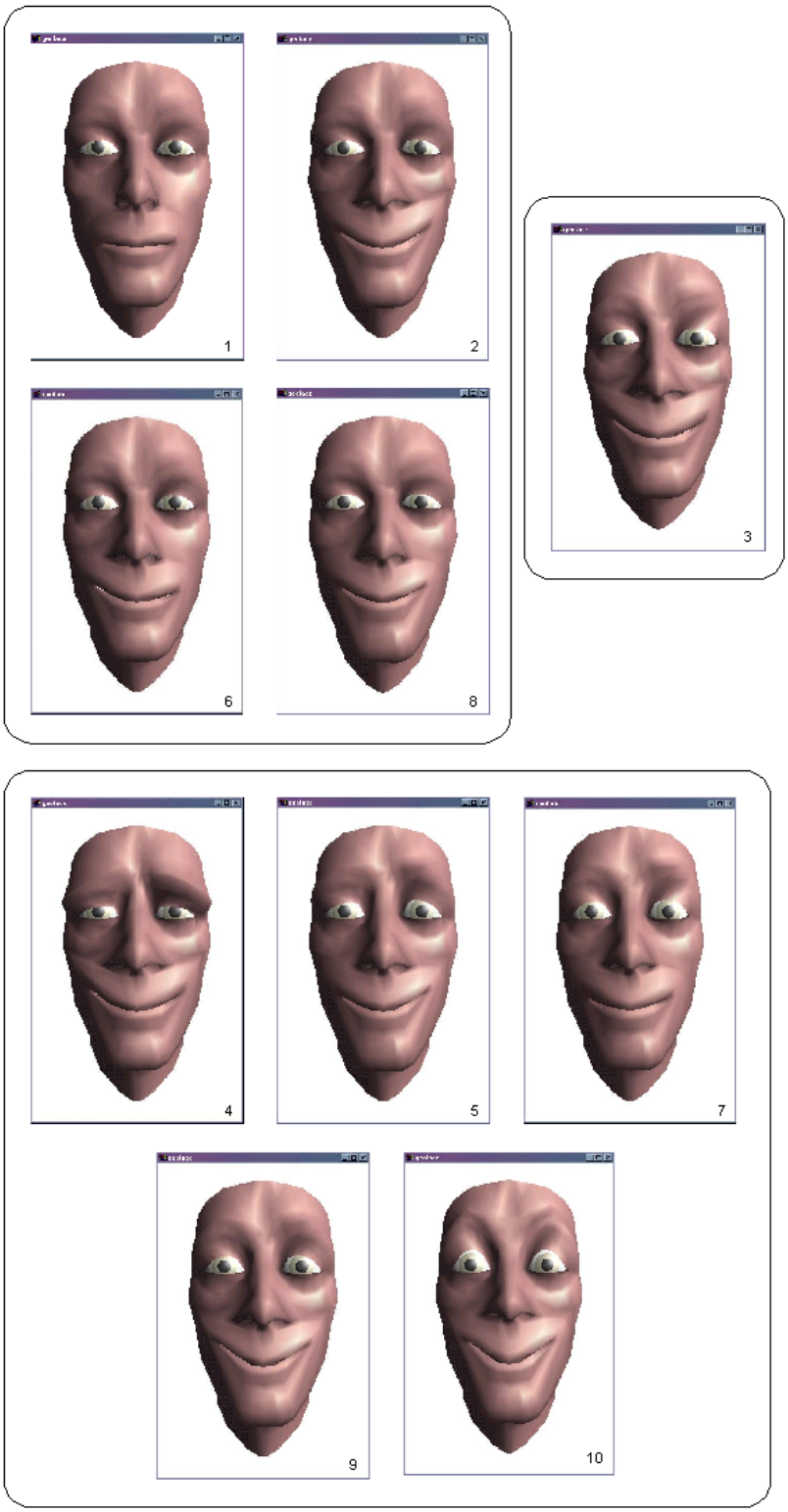
Şekil 9.9 FCM Sonuçları (Parametre Ağırlıkları Etkin Değil)

FCM algoritmasının yürütülmesi sonucunda 1,2,3,6,8 örnekleri bir bölüte dahil olmuş ve diğer örnekler de ikinci bölütü oluşturmuştur.



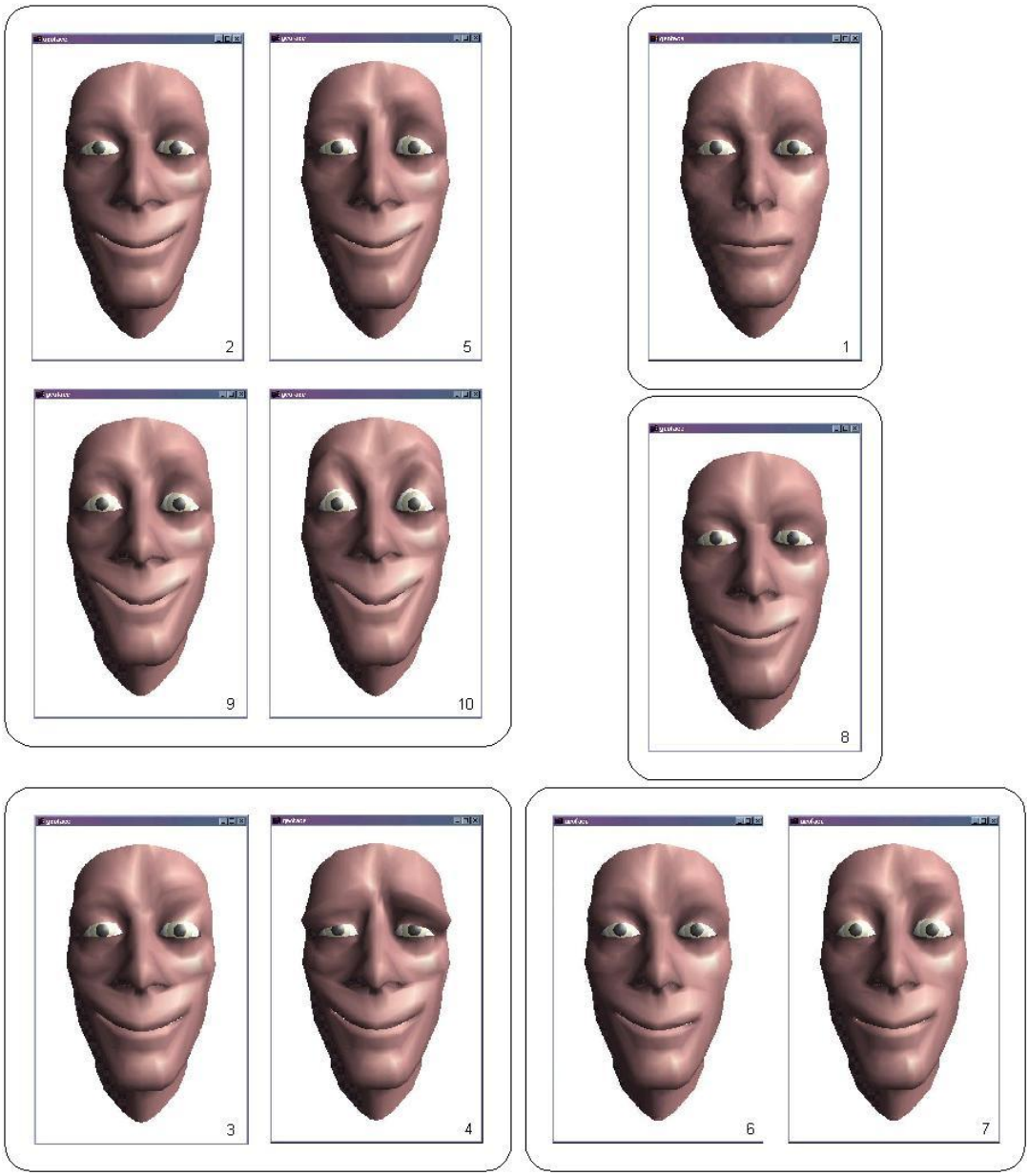
Şekil 9.10 Complete-link Sonuçları (Parametre Ağırlıkları Etkin Değil)

Complete-link algoritmasının yürütülmesi sonucunda 4,5 örnekleri bir bölüte dahil olmuş ve diğer örnekler de ikinci bölütü oluşturmuştur.



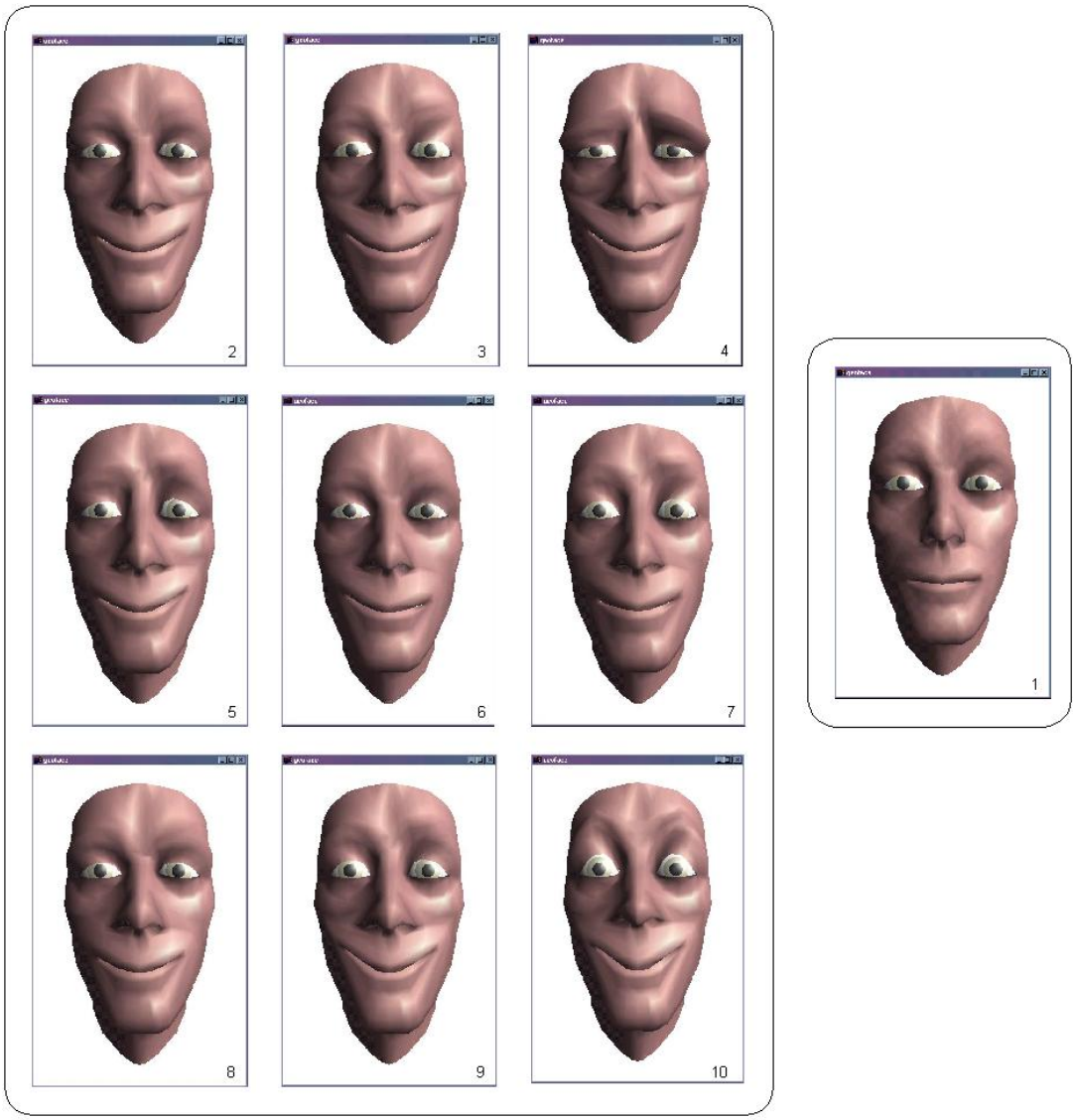
Şekil 9.11 Single-link Sonuçları (Parametre Ağırlıkları Etkin Değil)

Single-link algoritmasının yürütülmesi sonucunda 1,2,6,8 örnekleri ilk bölüte dahil olmuş, 3 örneği üçüncü bölütü oluşturmuş, 4,5,7,9,10 örnekleri de üçüncü bölütü oluşturmuştur.



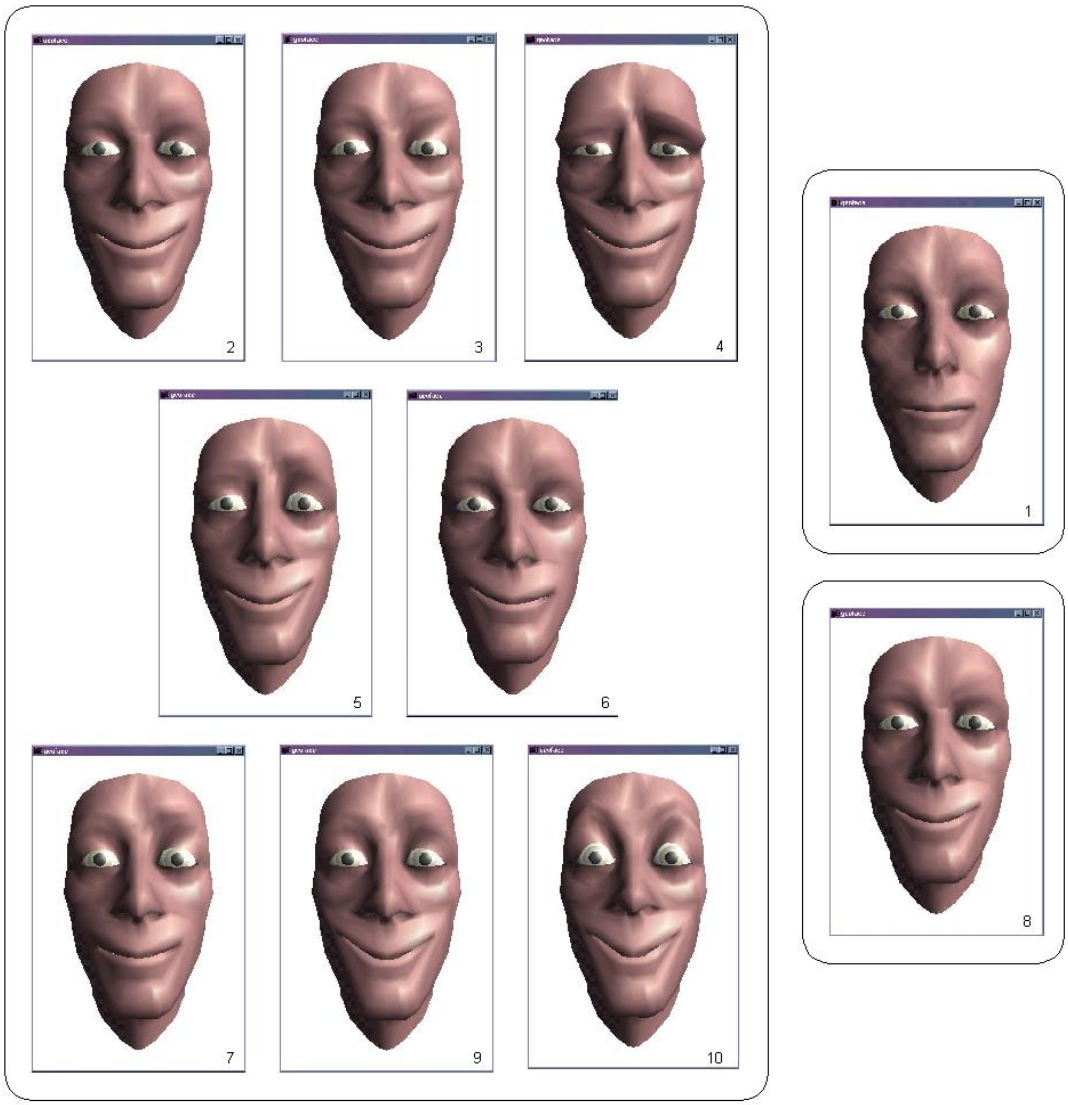
Şekil 9.12 FCM Sonuçları (Parametre Ağırlıkları Etkin)

FCM Algoritması, parametre ağırlıkları etkinleştiğinde daha çok bölüt oluşturmaktadır. 8 örneği ilk bölüte, 1 örneği ikinci bölüte, 3 ve 4 örnekleri üçüncü bölüte, 6 ve 7 örnekleri dördüncü bölüte ve 2,5,9 ve 10 örnekleri beşinci bölüte dahil olmuştur. Bu sonuçlar, parametre ağırlıklarında dudak ve kaş bitirme kasları etkin olarak alındığında elde edilmiştir. Complete-link ve Single-link algoritmaları sonuçları da aynı parametre grubu için geçerlidir.



Şekil 9.13 Complete-link Sonuçları (Parametre Ağırlıkları Etkin)

Complete-link algoritması, parametre ağırlıkları etkin iken sadece 1 örneğini ayrı bir bölüte almış, diğer örnekler için tek bir bölüt oluşturmuştur. Bu denemede parametre normalizasyonu da yapılmıştır. Bölüt sayılarının değişimde parametre normalizasyonunun etkisi olmuştur.



Şekil 9.14 Single-link Sonuçları (Parametre Ağırlıkları Etkin)

Single-link algoritması, parametre ağırlıkları etkin iken 1 ve 8 örneklerini ayrı birer bölüt olarak almış, diğer örnekler için de tek bir bölüt oluşturmuştur. Bu denemede parametre normalizasyonu da yapılmıştır. Bölüt sayılarının değişiminde parametre normalizasyonunun etkisi olmuştur.

Parametre ağırlıklarının eklenmesi sonucunda etkili kaslar üzerindeki denetim daha farklı olmaktadır. Dolayısıyla sağlıklı sonuçlar alabilmek için parametre ağırlıklarının belirlenmesi gerekmektedir. Hangi algoritmanın sonucunun daha iyi olduğuna karar verebilmek için sosyal bilimlerde uzman olan araştırmacılardan görüş alınmalıdır.

10. SONUÇLAR VE TARTIŞMA

Bu tez çalışması kapsamında gerçeklenen yazılım sistemi, bir çoklu etmen sistemi uygulaması olarak tasarlanmış, makine öğrenmesi ve bölütleme yöntemlerine ilişkin birimler oluşturulup uygun bir şekilde entegre edilmiştir.

Toplum kesiti ve tek tek etmenlerin belirlenen etkilenme katsayılarına göre oluşturdukları inançlarının analizi ve bunlar arasındaki farklılıkların belirlenmesi çalışmanın sosyal boyutlarını oluşturmaktadır. Bu analiz, sistemin geniş bir kullanıcı kitlesi üzerinde denenmesini ve disiplinler arası çalışma gerektirdiğinden dolayı, bu tez çalışması kapsamında yürütülmemiştir. Sistemin çalışması gösterilmek üzere, belli bileşenler küçük bir kullanıcı grubu üzerinde denenmiş ve beklenen sonuçlar alınmıştır. Kullanıcılar üzerinde denenmeyen bileşenlerin testleri simülasyon ortamında yapılmış ve sonuçların doğruluğu gözlenmiştir.

Uygulamada uzman kişilerin ve ön bilgilerin olmayabileceği düşünülerek tüm parametreler ve parametre ağırlıklarının sistem tarafından öğrenilmesi de sağlanmaktadır. Böylece çalışma esnasında tamamıyla kendi kendine öğrenen akıllı bir sistem gerçeklenmiştir. Program, modüler olduğundan dolayı belirli süreçlerin işlem dışı bırakılması ile özel bir uygulama amacı ile kolaylıkla bir uygulama yazılımı ürünü elde edilebilir.

Sistemin özellikle küçük çocuklar üzerinde denenmesi amaçlanmıştır. Ancak grafiksel yüz animasyon programı arayüzünün çocukların ilgisini çekmesi için ayrıca bir çalışma yapılması gerekmektedir. Bu konuda yapılabilecek bir iyileştirme ile uygulama daha da ilgi çekici hale gelecektir.

Reaktif nesnelerin ayrı birer birim olarak gerçeklenmesinin nedeni, yapay karakterlerde yüz ifadesi oluşturma türünden ileri çalışmalarda bu özellikten faydalanabilmektir. Bir yüzün ifadesinin oluşturulmasında ayrı birimler şeklinde davranmaları sağlanarak ve gerekiyorsa farklı makinelerde yaşamalarına izin verilerek çalışma başarımında etkili sonuçlar alınabilir. Bölütleme sonuçlarının ayrı dosyalarda oluşturulmasının nedeni ise sonuçların dışarıdan takibinin kolay olmasını sağlamaktır.

Sistemin öğrenme sürecinin yapay sinir ağları ile yapılması ve Q öğrenmesi ile başarım karşılaştırılması, ileri ve ilgi çekici bir konu olarak ortaya çıkmıştır.

Büyük verileri bölütlemede açık olan problemler üzerinde durularak dinamik ve artımlı yöntemler geliştirilmesi de ileri bir konu olarak düşünülebilir.

Bu sistemin gerçek yaşamda uygulamaları; haber veya hikaye okuyan yapay karakterlerin yüz ifadelerinin oluşturulması, sistemin çeşitli kullanıcı grupları üzerinde denenerek gruplar veya kültürler arası farklılıkların ortaya çıkarılması, çeşitli gruplar için kesit bilgileri oluşturarak gruplar arası görüntü iletiminde uygun gruba uygun yüz ifadesinin iletilmesi, kişiye özel hizmet arabirimi tasarımı gibi uygulamalarda kullanılması ve psikolojik olarak hasta çocuklar üzerinde denenerek çok aykırı veriler oluşturan çocukların belirlenmesi olarak verilebilir.

KAYNAKLAR

Aglets, http://www.trl.ibm.com/aglets/

Bezdek, J.C., 1981. Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, New York.

Collinot A. ve diğ., 1996. Agent Oriented Design of a Soccer Robot Team, *Proceedings of the 2nd Int. Conf. on Multi-Agent Systems (ICMAS´96) AAAI Press,* Menlo Park, CA, USA, s. 41 - 47.

David, E., Kraus, S., 1999. Agents for Information Broadcasting, *Intelligent Agents VI Agent Theories, Architectures, and Languages 6th International Workshop*, ATAL'99 Orlando, Florida, USA, July 15-17.

Deepika, C.,1997. JAFMAS: A Java-Based Agent Framework for Multiagent Systems Development and Implementation, *PhD Thesis,* ECECS Department, University of Cincinnati, OH.

Drogoul, A. ve diğ., 1993. MANTA: A New Experimental Results on the Emergence of (Artificial) Ant Societies, *In Simulating Societies Symposium*, Siena, C. Castelfranchi.

Everitt B. S., 1993. Cluster analysis, London : E. Arnold ; New York : Halsted Press.

Ezzat, T. ve Poggio, T., 1998. Mike Talk: A Talking Facial Display Based on Morphing Visemes, *Proceedings of Computer Animation Conference,* Philadelphia, Pennsylvania, June.

Fazlı C., 1993. Incremental Clustering for Dynamic Information Processing, *ACM Transactions on Information Systems*, **11**, 2, April.

Finin, T. ve Fritzon, R., KQML- A Language and Protocol for Knowledge and Information Exchange, *Proceedings of 19th Intl. DAI WorkShop*, s. 127-136.

Jain A. ve Mao J., 2000. Statistical Pattern Recognition: a Review, *IEEE Transactions on Pattern Analysis and Machine intelligence*, **22**, 1, January.

Jain, A. K. ve diğ., 1999. Data Clustering: A Review, *ACM Computing Surveys*, **31**, 3, September.

JATLite, http://java.stanford.edu

Java/Net, http://java.sun.com/j2se/1.4/docs/api/java/net/package/package-summary.html

Kismet, http://www.ai.mit.edu/projects/humanoid-robotics-group/kismet/kismet.html

Koda T. ve Maes, P., 1996. Agents with Faces: The Effects of Personification of Agents, *Proceedings of HCI'96,* London, UK, August 20-23, s. 98-103.

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

Maes, P. ve diğ., 1995. The ALIVE System: Full-Body Interaction with Autonomous Agents, *Proceedings of the Computer Animation '95 Conference*, Geneva, Switzerland, pp. 11-18, April.

Mitchell, T. M., 1997. Machine Learning, McGraw-Hill Companies, Inc.

Parke, F. I., 1982. Parametrized Models for Facial Animation, *IEEE Computer Graphics and Applications,* **2**, 9, 61-68.

RoboCup 2002, http://www.robocup2002.org

Ross, T., 1995. Fuzzy Logic With Engineering Applications, McGraw-Hill Companies, Inc.

SIGGRAPH 97, Panel on Facial Animation http://www.cs.toronto.edu/ ~siggraph97-panel/

SimpleFace, http://www.crl.research.digital.com/publications/books/waters/ Appendix1/appendix1.html

Sutton R. S. ve Barto, A. G., 1999. Reinforcement Learning, Richard S. Sutton and Andrew G. Barto.

Terzopoulos, D. ve Waters, K., 1993, Analysis and Synthesis of Facial Image Sequences Using Physical and Anatomical Models, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **15**, 6, 569-579.

Terzopoulos, D. ve diğ., 1994. Artificial Fishes with Autonomous Locomotion, Perception, Behavior, and Learning, In a Physical World, *Artificial Life*, **1**,4, 327-351.

Tian, Y. ve Cohn, J., 2001. Recognizing Action Units for Facial Expression Analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **23**, 2, 97-114.

Voyager, http://www.recursionsw.com/products/voyager/

Watkins, C.J.C.H., 1989. Learning from delayed rewards, *PhD Thesis*, University of Cambridge, England.

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

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

Zadeh, L. A., 1965. Fuzzy sets. *Information and Control*, **8**, 3, 338-353.

ÖZGEÇMİŞ

Sanem SARIEL 1977 yılında Bandırma' da doğmuştur. 1994 yılında İstanbul Teknik Üniversitesi Kontrol ve Bilgisayar Mühendisliği bölümünü kazanmıştır. Üniversiteye girdiği ilk yıl İngilizce hazırlık eğitimi görmüştür. 1999 yılında Kontrol ve Bilgisayar Mühendisliği bölümünden mezun olmuştur. 1999 yılından beri İTÜ Bilgisayar Mühendisliği bölümünde araştırma görevlisi olarak görev yapmaktadır.