İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

KOALİSYON KURULUMU İLE GÖREV PAYLAŞIMI İÇİN BİR ÇOKLU ETMEN TABANLI YÜRÜTMA ORTAMI

YÜKSEK LİSANS TEZİ

Merve ÖZBEY

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

OCAK 2014

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

KOALİSYON KURULUMU İLE GÖREV PAYLAŞIMI İÇİN BİR ÇOKLU ETMEN TABANLI YÜRÜTMA ORTAMI

YÜKSEK LİSANS TEZİ

Merve ÖZBEY
504081520

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Nadia ERDOĞAN

OCAK 2014

İTÜ, Fen Bilimleri Enstitüsü'nün **504081520** numaralı Yüksek Lisans Öğrencisi **Merve ÖZBEY**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"KOALİSYON KURULUMU İLE GÖREV PAYLAŞIMI İÇİN BİR ÇOKLU ETMEN TABANLI YÜRÜTME ORTAMI"** başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Prof. Dr. Nadia ERDOĞAN** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Yrd. Doç. Dr. Sanem SARIEL TALAY** İstanbul Teknik Üniversitesi	.....
	Yrd. Doç. Dr. Yunus Emre SELÇUK Yıldız Teknik Üniversitesi	.....

Teslim Tarihi : **16 Aralık 2013**
Savunma Tarihi : **20 Ocak 2014**

Anneme,

ÖNSÖZ

Tez çalışmam süresince ilminden faydalandığım, desteğini ve ilgisini hiç eksik etmeyen ayrıca tecrübelerinden yararlanırken göstermiş olduğu hoşgörü ve sabırdan dolayı değerli hocam sayın Prof. Dr. Nadia Erdoğan'a,

Çalışmamı daha rahat sürdürebilmem için ellerinden geldiğince yardımcı olan, araştırmamın her aşamasında destek olan başta Hakan Sarıbıyık ve Serkan Taş olmak üzere tüm Likya Teknoloji çalışanlarına,

Bu günlere gelmemde en büyük paya sahip olan anneme, ablama ve dostlarıma sonsuz teşekkürlerimi sunarım.

Ocak 2014

Merve Özbey
Bilgisayar Mühendisi

İÇİNDEKİLER

Sayfa

ÖNSÖZ ..... **vii**
İÇİNDEKİLER ..... **ix**
KISALTMALAR ..... **xi**
ÇİZELGE LİSTESİ ..... **xiii**
ŞEKİL LİSTESİ ..... **xv**
ÖZET ..... **xvii**
SUMMARY ..... **xix**
1. GİRİŞ **1**
1.1 Problemin Tanımı 1
1.2 Literatür Araştırması 1
2. ETMEN SİSTEMLERİ **3**
2.1 Etmen Tanımı 3
2.2 Etmenlerin Özellikleri 3
2.3 Çoklu Etmen Sistemleri 4
3. JAVA AGENT DEVELOPMENT FRAMEWORK (JADE) **5**
3.1 JADE Platformunun Genel Yapısı 6
3.2 FIPA 7
3.3 Etmen Platformu 7
3.4 Etmen Yönetimi 8
3.4.1 Etmen başlatma 10
3.4.2 Etmen sonlandırma 10
3.4.3 Etmen davranışları 11
3.5 Etmen Haberleşmesi 12
3.5.1 FIPA iletişim modeli 13
3.5.2 ACLMessage sınıfı 15
3.5.3 Mesaj gönderme ve alma 15
4. ETMEN SİSTEMLERİNDE KOALİSYON KURULUMU **17**
4.1 Etmen Sistemleri ve Koalisyonlar 17
4.2 Koalisyon Kurulum Algoritması 18
4.3 Küme Örtüleme ve Küme Bölüntüleme (Set Covering ve Set Partitioning) 19
4.4 Shehory-Kraus Koalisyon Kurulum Algoritması 20
4.5 Sistemin Mimari Tasarımı 25
4.5.1 Veri tabanı tasarımı 26
4.5.2 Veri tabanı etmeni 29
4.5.3 Denetçi etmen 30
4.5.4 İşçi etmen 32
4.5.5 Koalisyon yöneticisi etmeni 32
4.5.6 Olayların zinciri 35
4.6 Kullanıcı Arabirimi 36
5. BAŞARIM DEĞERLENDİRMESİ **37**
5.1 İşçi Etmen Sayısının Koalisyon Kurulum Süresine Etkisi 37

5.2 Görev Sayısının Koalisyon Kurulum Süresine Etkisi 38
6. SONUÇ VE ÖNERİLER 41
KAYNAKLAR 43
EKLER 45
ÖZGEÇMİŞ 65

KISALTMALAR

JADE	**:** Java Agent Development Framework
FIPA	**:** The Foundation of Intelligent Physical Agents
ACL	**:** Agent Communication Language
DPS	**:** Distributed Problem Solving
TILAB	**:** Telecom Italia Lab
AMS	**:** Agent Management System
DF	**:** Directory Facilitator
ACC	**:** Agent Communication Channel
API	**:** Application Programming Interface
AID	**:** Agent Identifier
XML	**:** Extensible Markup Language
SQL	**:** Structured Query Language
SL	**:** Semantic Language
DAI	**:** Distributed Artificial Intelligence
MAS	**:** Multi-Agent Systems
XSD	**:** XML Schema Definition
Java EE	**:** Java Enterprise Edition
JSF	**:** Java Server Faces

ÇİZELGE LİSTESİ

Sayfa

Çizelge 4.1 : İşçi etmen bilgileri. 27
Çizelge 4.2 : Görev bilgileri. 27
Çizelge 4.3 : Raporlama bilgileri. 28
Çizelge 4.4 : Kullanıcı bilgileri. 28

ŞEKİL LİSTESİ

Sayfa

Şekil 3.1 : FIPA etmen platformu mimarisi. 8
Şekil 3.2 : Etmenin yaşam döngüsü. 9
Şekil 3.3 : JADE haberleşme mimarisi. 13
Şekil 4.1 : Algoritma 1. aşama. 22
Şekil 4.2 : Algoritma 2. aşama. 23
Şekil 4.3 : Algoritma 3. aşama. 24
Şekil 4.4 : Koalisyonlar kurulmadan önceki etmen etkileşimi. 26
Şekil 4.5 : Koalisyonlar kurulduktan sonraki etmen etkileşimi. 26
Şekil 4.6 : Veri tabanı etmeni akış şeması. 30
Şekil 4.7 : Denetçi etmenin akış şeması. 31
Şekil 4.8 : İşçi etmenin akış şeması. 33
Şekil 4.9 : Koalisyon yöneticisi etmeni akış şeması. 34
Şekil 4.10 : Akış diyagramı. 35
Şekil 4.11 : Kullanıcı giriş ekranı. 36
Şekil 4.12 : Görev tanımlama ekranı. 36
Şekil 5.1 : Farklı görev sayılarında işçi etmen sayısının koalisyon kurulum süresine etkisi. 37
Şekil 5.2 : Farklı *k* değerlerinde işçi etmen sayısının koalisyon kurulum süresine etkisi. 38
Şekil 5.3 : Görev sayısının koalisyon kurulum süresine etkisi. 39

KOALİSYON KURULUMU İLE GÖREV PAYLAŞIMI İÇİN BİR ÇOKLU ETMEN TABANLI YÜRÜTME ORTAMI

ÖZET

Çoklu etmen sistemleri, birbirleriyle iletişim kuran ve işbirliğinde bulunan birden fazla akıllı otonom etmenin bir araya gelerek oluşturduğu sistemlerdir. Etmenler kendi yetenekleri ve bilgilerinin yeterli olmadığı problemleri bir araya gelerek oluşturdukları bu sistemlerle çözebilirler. Görev paylaşımı yaklaşımı ile etmenlerin görevleri birlikte yerine getirmesi, etmen koalisyonu olarak adlandırılmaktadır.
Bu çalışmada iş yapma yeteneği olan etmenler tanımlanan görevleri yerine getirmek üzere koalisyonlar kurarlar. Koalisyon kurma aşamasında gerçeklenen bir algoritma ile etkili bir şekilde tüm görevleri ya da tüm etmenleri kapsayacak şekilde koalisyonlar kurulur. Koalisyon kurma problemi taşımacılık şirketi problemi üzerinden uygulanmıştır.
Öncelikle sistemde olması gereken bileşenler belirlenmiştir: veri tabanı, veri tabanı işlerini yerine getirecek veri tabanı etmeni, sistemin genel koordinasyonunu sağlayacak denetçi etmen, sadece kendi koalisyonundan sorumlu olacak koalisyon yöneticisi etmenleri ve en başta belirlenen görevler için koalisyon kurulumunu yapıp bu görevleri yerine getirecek işçi etmenler. Her bir etmen çeşidi kendi uzmanlaştığı işleri yerine getirir, hangi etmenlerle iletişim kurması gerekiyorsa sadece o etmenlerin iletişim bilgilerine sahiptir. Koalisyonlar kurulmadan önce işçi etmenler birbirleriyle ve denetçi etmenle iletişim halindedir; koalisyonlar kurulduktan sonra ise aynı koalisyondaki işçi etmenler kendi içlerinde ve sadece o koalisyonun koalisyon yöneticisi etmeni ile iletişimde bulunurlar, denetçi etmenle ilişkileri kalmaz. Koalisyon kurulumu gerçekleşir gerçekleşmez her bir görev birer koalisyon tarafından yerine getirilir. Görevler tamamlandıktan sonra koalisyon yöneticisi etmenleri görevlerin tamamlandığını denetçi etmene bildirir, denetçi etmen de bu çalışmayla ilgili ayrıntıların veri tabanı etmeni aracılığıyla veri tabanına kaydedilmesini sağlar.
Sistemin gerçeklenmesi sırasında etmenlerin yaratılmasını, yönetilmesini ve birbirleri ile haberleşmesini sağlayan araçlar ve yardımcı kütüphaneler sunan JADE platformu kullanılmıştır. JADE, etmen ve çoklu etmen sistemlerinin standartlarını belirleyen FIPA tarafında oluşturulan standartları destekleyen çoklu etmen geliştirme ortamı ve kütüphanesidir. Etmenler arası haberleşme ve etmenlerin davranışlarını kontrol etmek için esnek ve genişletilebilir bir yapı sunmaktadır. JADE'in bu özellikleri kullanılarak etkili bir şekilde koalisyon kurulumu yapan ve görevlerin bu koalisyonlar aracılığıyla yerine getirildiği bir çalışma ortaya çıkartılmıştır.

Yapılan deneyler sonucunda işçi etmen sayısındaki, görev sayısındaki ve en büyük koalisyon boyutundaki artışın koalisyon kurulum sürelerine etkisi incelenmiştir, her durumda koalisyon kurulum süresi beş saniyeyi geçmemiştir.

Bu çalışma ile etmenlerin koalisyon kurarak görev paylaşımı yapmaları sonucu ele alınan problemin çözümünde hızlı ve etkili bir şekilde görev paylaşımı yapılarak görevlerin yerine getirilmesi sağlanmıştır.

MULTI-AGENT BASED EXECUTION ENVIRONMENT FOR TASK ALLOCATION VIA COALITION FORMATION

SUMMARY

Multi-agent systems are systems of intelligent autonomous agents that communicate and collaborate with each other. Agents cooperate and solve problems on multi-agent systems when the capability and knowledge of each agent alone is not sufficient to solve the problems. Task execution via task allocation with agents' cooperation is defined as **agent coalition**. Coalition formation is an important method for cooperation in a multi-agent environment. It is a process where agents form coalitions and work together to solve a joint problem via cooperating or coordinating their actions within each coalition. Membership in a coalition may increase the agents ability to accomplish tasks, achieve its goals or maximize its utility.

In this research, agents that have task execution capabilities, form coalitions to execute predefined tasks. In the coalition formation phase, coalitions are formed efficiently performing an algorithm that covers all of the tasks or all agents. Coalition formation algorithm has been applied to a transportation problem.

In our system, agents cooperate to maximise the total outcome of the system and they are not concerned with their personal profits. There is no central authority for distribution of tasks among the agents. Autonomous agents form coalitions by themselves. Agents in the execution environment follow the Shehory-Kraus's algorithm to negotiate and decide on coalitions. There exist several variations of the algorithm, such as disjoint coalitions, overlapping coalitions and coalitions for tasks that have precedence orders. Our implementation focuses on disjoint coalitions, therefore an agent can join only one coalition at a time. After a task is completed the members of the coalition can join new coalitions. Furthermore, agents cannot join an already formed coalition due to extra agent addition cost generated. Last assumption is that tasks to be performed have no precedence order and inter dependency.

The framework builds on various types of agents that are defined and implemented to handle different issues. In the design phase, we have determined the actors of the system, specifying their tasks and responsibilities. After associating each actor with an agent type, protocols that define in detail the interaction and coordination between agents were developed. In addition, an efficient and fast messaging infrastructure has been developed for effective agent communication.

In the framework, four different types of agents cooperate to provide the environment where coalitions are formed. Database agent that will perform database operations, controller agent that will control system's overall coordination, coalition manager agents that will be liable only its coalition and employee agents that will perform coalition formation on predefined tasks and execute those tasks.

Before coalition formation, employee agents communicate with each other and the controller agent. After a coalition is formed employee agents in same coalition communicates with each other and the coalition manager agent responsible of that coalition but no longer with controller agent. Next, each task is performed by a coalition directly. After task execution is completed coalition manager agents inform the controller agent that task has been completed. With this information, the controller agent administers recording of that execution's details via the database agent.

For database component, native XML database eXist selected. XML Shema Definition (XSD) of all XML files are prepared and xQuery query language used for database queries. Employee agent definitions, task definitions, user definitions and execution reports are stored in XML files on database. In addition, XSD and xQuery files are stored on database.

Database agent is the only agent in our system that has the capability of executing database operations. The database agent accepts database requests, executes queries and delivers results. Only, the controller agent has direct communication with the database agent. If other agents need to query the database, they transmit their requests through the controller agent to database agent and get results in the same way.

Employee agents are responsible for performing the coalition formation algorithm on predefined tasks and execute those tasks. These agents firstly register on active agent list announcing controller agent. After receiving the active agent list and the task list from controller agent, employee agents start coalition formation. Using coalition formation algorithm, they calculate which coalition they will join. One employee agent can join only one coalition at a time, because of disjoint coalitions assumption. Employee agents inform the controller about coalitions and then wait for coalition managers' starting messages. After getting the message, employee agents join their coalitions, execute their tasks and inform coalition manager agents that they have completed subtasks. If an error occurs during subtask execution, employee agents inform their own coalition manager agents.

Central management usually becomes a bottleneck; therefore the controller agent conveys some of its tasks to coalition manager agents as to decrease its work load. The controller agent monitors agents' status and task control period. When a predefined task control period ends, the controller agent restarts coalition formation mechanism, therefore there is no need to reset the system after new tasks have been defined.

Controller agent monitors coalition formation phases; after coalition formation is completed, the controller agent starts a new coalition manager agent for each coalition, sends them the employee agents' identifications and task's details. Next, controller agent waits for coalition managers' messages. After receiving all coalition managers' messages, prepares an execution report and sends it to database agent to insert into the database.

The controller agent is also responsible for information sharing. The controller agent provides the necessary information to other agents, for example, it sends employee agents and task details of a coalition to manager of that particular coalition.

Furthermore, the controller agent is responsible for error checking; when an error message is received from a coalition manager agent, the controller agent updates the state of relevant task's state to "failed" on database through database agent. The failed task is executed on the next round. With this approach, our system becomes fault tolerant.

For every coalition, one coalition manager exists. Coalition manager agent is responsible of only its coalition and employee agents on that coalition. Coalition manager agent checks whether employee agents have completed subtasks; after a task is completed, the coalition manager informs the controller agent and shuts down itself.

Coalition manager agent is responsible for error checking, similar to the controller agent. If a subtask execution results in error, coalition manager stops all other subtasks in the coalition and informs controller agent about error. In addition to error handling, the coalition manager checks employee agents at predefined periods to see if they are standing. If any employee agent is not standing, coalition manager stops all other subtasks in the coalition and informs controller agent about error. In both error cases, coalition manager shuts down itself as the coalition no longer exists.

To insert new task definitions to system, we need an interface. For this purpose, a web application has been developed using Primefaces library. The application starts with a login screen, and then continues with the task definition screen. A newly defined task waits the next task control period for execution.

The JADE platform is used for creating, managing and supporting the communication of the agents during the design and implementation of the coalition formation environment. The JADE environment and its library provides a rich set of tools for this purpose. JADE is a multi-agent system development environment and library which supports the standards defined by FIPA, an organization that defines the standards for multi-agent systems. JADE provides a flexible and extensible structure for agent communication and controlling behaviours of agents. Using JADE's properties, an agent based system which provides task allocation using coalition formation efficiently has been developed.

We tested our system approximately 3000 runs for different numbers of employee agents, for different numbers of tasks and for different values of maximum coalition sizes and have observed their effects on coalition formation time. Increases in the number of employee agents and coalition size increase coalition formation time fastly, however, the effect of variation in the number of tasks was different. Increase of task count increased coalition formation time up to three tasks. After three tasks, test results gave us approximately the same values. Coalition formation time did not exceed five seconds on any test.

Experimental results show that the framework fully provides the requirements for a negotiation environment for multi-agents to form coalitions and execute tasks. The framework can be used as a test bed for further research as well.

In this study, we provide a solution to the problem of task allocation through coalition formation and implementation of a multi-agent based execution environment for this purpose. We show that task allocation and task execution can be performed quickly and efficiently via an application on a transportation problem.

1. GİRİŞ

1.1 Problemin Tanımı

Bir DPS (Distributed Problem Solving) sistemindeki otonom etmenlerden oluşan gruplarda görev paylaşımı yapılması gerekmektedir [1]. DPS sistemlerinde etmenler kişisel faydalarını gözetmeksizin sistemin genel faydasını arttırmaya yönelik işbirliğinde bulunurlar. Yapılan çalışmada görevler yerine getirilirken sistemin bütününün faydası maksimize edilmeye çalışılmıştır. Öncelik sırası olan görevler kapsam dışı bırakılmıştır. Görevleri etmenlere dağıtan merkezi bir sistem bulunmamaktadır, otonom etmenler tanımlanmıştır. Görevler alt görevlere bölünerek, her bir alt görev etmenlerin koalisyonu ile gerçekleştirilebilir. Bir görevin koalisyon üyesi olan etmen, diğer bir görevin de koalisyon üyesi olamaz çünkü ayrık koalisyon yaklaşımı kullanılmıştır. Yapılan çalışmada taşımacılık şirketi problemi ele alınmıştır.

1.2 Literatür Araştırması

Etmen sistemlerinde koalisyon kurulumu ile ilgili yapılmış çeşitli çalışmalar bulunmaktadır. Bu konuyla ilgili en temel çalışmalardan biri olan O. Shehory ve S. Kraus'un [1] çalışmasında koalisyon kurulumunda görev paylaşımı için çeşitli metotlar önerilmiştir. Bu metotlar ayrık koalisyonlar, kesişen koalisyonlar ve öncelik sıralı görevler için oluşturulan koalisyonlardır. Yapılan tez çalışmasında ayrık koalisyon metodu kullanılmıştır.

İçlerinde O. Shehory'nin bulunduğu başka bir çalışmada [2] koalisyon kurma aşamasında kullanılan, koalisyonun maksimum boyutu sınırını hesaplanıyor. Koalisyonun maksimum boyutu koalisyona katılacak maksimum etmen sayısına karşılık geliyor. Bu çalışmada sınır olabildiğince küçültülüyor, böylece koalisyon kurarkenki iterasyon sayısını azaltmış oluyor. Yapılan tez çalışmasında bu sayı en başta sabit olarak verildiği için bu tür bir hesaplama yapılmamaktadır, fakat

koalisyon kurma sırasındaki hesaplamaların karmaşıklığının artmasını engellemek amacıyla olabildiğince küçük tutulmuştur.

O. Shehory ve S. Kraus'un birlikte bulundukları başka bir çalışmada [3] zaman kısıtı ve eksik bilgi olması durumlarında çalışabilen bir algoritma sunuluyor. Buradaki eksik bilgi: her etmen diğer tüm etmenlerin bir görevin alt görevini yapma maliyetini bilmiyor. Yapılan testlerle verilen çözümün optimal çözümlere yakın sonuçlara ulaştığını gösteriyorlar. Aynı çalışmanın devamı niteliğindeki başka bir çalışmada [4] zaman kısıtı ve eksik bilgiye rağmen yine iyi sonuçlar elde ettiklerini gösteriyorlar. Yapılan tez çalışmasında bu tür kısıtlar bulunmamaktadır.

Bu çalışmalardan farklı olarak 2003 yılında yapılan bir çalışmada [5]; dinamik olarak değişen ortamlardaki çalışmalardaki koalisyon değerlerinin bilindiği ve sabit olduğu kabulleri yerine bu değerleri bulanık küme teorisi ile Kernel'in oyun kuramı stabilitesi kavramını birleştirerek stabil koalisyonlar elde etmişlerdir.

Çoklu etmen sistemlerinin organizasyonel yapısı ile ilgili bir çalışmada [6] etmenlerin rol tanımı yapılmıştır. Ayrıca her bir etmen grubuna birer grup yöneticisi ataması yapılmıştır, grup yöneticisi o gruptaki etmenlerden sorumludur. Yapılan tez çalışmasında bu çalışmadaki grup yöneticisi gibi her koalisyona birer koalisyon yöneticisi etmeni ataması yapılmıştır.

Koalisyon kurulum algoritmalarından ziyade çoklu etmen sistemleri ortam tasarımı ve etmen tasarımı ile ilgili B. Alfonso, V. Botti, A. Garrido ve A. Giret 'in birlikte 2012 yılında yaptıkları çalışmada [7] etmenlerin rol tanımları ve etmenlerin görüştüğü ortamın tanımı gibi yürütme ortamı ile ilgili ayrıntılı bir tasarım anlatılmıştır. Bu tasarım gerçeklenerek çoklu etmen sistemiyle su hakları pazarlığı konusunda simülasyon yapılmıştır.

2011 yılında yayınlanan, yine güncel bir uygulama örneği olan çalışmada [8] çoklu etmen sisteminde koalisyon kurulumu ile elektrik piyasası simülasyonu yapılmıştır.

Ocak 2013'te yayınlanan koalisyon kurulumunun detaylı bir şekilde ele alındığı bir çalışmada [9] koalisyon kurulumundaki denge sürecinin tanımı yapılmış ve koalisyonel kısıtların olduğu klasik oyun teorisindeki temel kavramlar anlatılmıştır. Koalisyonel stabilitenin standart kavramlarının bazılarının, koalisyon kurulumundaki denge süreci kavramıyla nasıl kapsanabileceğini gösterilmiştir.

2. ETMEN SİSTEMLERİ

2.1 Etmen Tanımı

Etmenler, tasarım hedefleri doğrultusunda bir ortam içerisinde bağımsız hareket edebilen bilgisayar sistemleridir [10]. Bir etmen sistemi oluşturmak için, ortamda çalışan ve ortamla iletişim halinde olan tek bir etmen yeterli olsa da, genellikle birçok etmen aynı ortamda birlikte çalışır. Birden fazla etmenin; aynı ortamda, birbirleriyle çelişen veya uyum halinde çalıştıkları ve birbirleriyle mesajlaşma yoluyla iletişim kurabildikleri sistemlere ise çoklu etmen sistemleri adı verilir.

[11] Etmen tabanlı sistemler 1990'lardan itibaren bilgisayar dünyasının en canlı ve önemli araştırma alanlarından biri olmuştur. Etmen sistemleri kavramı; bilgisayar ağları, yazılım mühendisliği, nesneye dayalı programlama, yapay zekâ, insan bilgisayar etkileşimi, dağınık ve paralel sistemler, kontrol sistemleri, karar verme ve bilgi çıkarımı gibi birçok bilgi teknolojisinde geniş kullanım alanı bulmaktadır. [11] Etmen sistemlerinin otonom, zeki ve hedef tabanlı yapısı çeşitli alanlardaki yeni nesil yazılımlar için umut verici çözümler sunmaktadır.

2.2 Etmenlerin Özellikleri

Bir yazılımın etmen olarak kabul edilebilmesi için bazı özellikleri sağlıyor olması gerekir. Bu özelliklerle ilgili tüm çevrelerce kabul gören ortak bir tanımda karar kılınamamıştır. En yaygın kabul gören tanım ise Jennings ve Wooldridge tarafından yapılan tanımdır [12], bu tanıma göre bir etmenin sahip olması gereken özellikler şu şekilde sıralanabilir:

- **Otonom (autonomy):** Etmenlerin diğer etmenlerin erişimine açık olmayan nitelikleri vardır. Etmenler, bu niteliklerin değerlerine bakarak başka yapılarla (insan, diğer etmenler) doğrudan etkileşime girmeden karar verebilmelidir.

- **Sosyal yetenek (social ability):** Etmenler diğer etmenlerle (bazen de insanlarla) etmen dili denilen dillerle ya da herhangi bir programlama diliyle iletişime geçebilirler ve hatta amaçlarına yönelik ortaklıklar kurabilirler.
- **Reaktivite (reactivity):** Bir etmen çevresini algılar ve çevresindeki değişikliklere göre yeni davranışlar sergiler.
- **Proaktiflik (pro-activeness):** Proaktif sistemler sadece çevresindeki değişikliklere tepki vererek değil, inisiyatif alarak da çalışabilirler. Bu şekilde, ortamdaki diğer etmenlerin kararlarında değişiklik yapmalarına neden olabilecek değişiklikler yapabilirler.

2.3 Çoklu Etmen Sistemleri

Daha önce de bahsedildiği gibi etmenler sadece çevrelerini algılamak ve onlara tepki vermekte yetinmezler. Etmenler sosyal ilişkiler kuran aktif yazılım bileşenleridir. Etmenler arasındaki ilişki bazen rekabet, bazen ortaklık temelli bazen de her iki temele birden dayanacak şekildedir. Çoklu etmen sistemleri, birbirleriyle iletişim kuran birden fazla akıllı etmenin bir araya gelerek oluşturduğu sistemlerdir. Etmenler kendi yetenekleri ve bilgilerinin yeterli olmadığı problemleri bir araya gelerek oluşturdukları bu sistemlerle çözebilirler. Çoklu etmen sistemlerinin özellikleri [13] şu şekilde sıralanabilir:

- Her bir etmenin problemin tamamını çözmek için eksik bilgisi ve yeteneği vardır.
- Tüm sistemi kontrol edemezler.
- Veri merkezileşmemiştir. Bir tek noktadan tüm veriye erişim mümkün değildir.
- Hesaplamalar asenkron yapılır.

3. JAVA AGENT DEVELOPMENT FRAMEWORK (JADE)

JADE, TILAB tarafından geliştirilen, noktadan noktaya iletişim mimarisini temel alan, FIPA standartlarına uygun olarak tasarlanmış bir dağıtık çoklu etmen geliştirme platformudur [14]. Yürütme anında bir etmen platformu, ağdaki aynı işletim sistemine sahip olmayan farklı makineler üzerine dağıtılabilir ve uzaktan bir grafik arabirimi ile yönetilebilir. İletişim mimarisi, etmenler arasında esnek ve verimli bir şekilde, eş zamanlı ve eş zamanlı olmayan haberleşmeye izin verir. FIPA etmen ve iletişim modelini tamamen destekleyecek şekilde gerçeklenmiştir ve sistemle bütünleştirilmiştir. JADE, FIPA üyesi olan veya olmayan birçok şirket ve akademik grup tarafından kullanılmaktadır. JADE, tamamıyla Java dilinde yazılarak bu dilin esnek yapısını ve hazır kütüphanelerini kullanarak kolaylıkla etmen geliştirme yapısı sunmayı hedeflemiştir. JADE'in kullanıcılarına kullanımı ve özelleştirmeyi kolaylaştırmak için hazır olarak sunduğu özellikler aşağıdaki şekilde sıralanabilir [15]:

- Etmenler, tam bir dağıtık sistem oluştururlar. Her etmen, farklı bir thread olarak çalışır ve farklı makineler üzerinde çalışabilir. Etmenler bulundukları makinenin ve işletim sistemlerinin özelliklerinden bağımsız olarak JADE'in haberleşme altyapısını kullanabilir.
- AMS (Agent Management System – Etmen Yönetim Sistemi), DF (Directory Facilitator – Sarı Sayfalar Servisi), ACC (Agent Communication Channel – Etmen İletişim Kanalı) başta olmak üzere FIPA standartlarına uyumludur. Böylece aynı standardı kullanan farklı etmenler, birbirleriyle kolayca haberleşebilirler. FIPA iletişim şartnamesinin kullanılması için gerekli olan kütüphaneler kullanıma hazır şekilde geliştiricilere sunulmuştur [16].
- Grafik arabirimi ile etmenler kontrol edilebilir.
- Etmenlere ilişkin kod ve verinin farklı platformlardaki katılımcı bilgisayarlar arasında taşınabilmesini sağlar.
- Davranış modelleri ile çoklu, paralel ve eş zamanlı etmen aktivitesini düzenler.

- Etmenlerin, dağıtık etmen platformunda benzersiz bir şekilde isimlendirilmesi JADE tarafından otomatik olarak yapılmaktadır. Her etmen başlatıldığında ona benzersiz bir isim atanır ve arama servisine kaydedilir. Etmenler, sağlanan API'ler ve görsel araçlar sayesinde yerel ortamdan veya uzaktan başlatılabilir.
- Etmen olmayan ve JADE ortamına bağlanmamış dış uygulamalar tarafından da etmenlerin başlatılmasını sağlayan bir arabirim mevcuttur.
- Yaygın, güçlü ve esnek bir programlama dili olan Java ile gerçeklenmiştir ve açık kaynak kodludur.
- Uygulamaya özel içerik dilleri ve ontolojiler tanımlamaya izin verir.
- Java dilinin Serializable özelliğini kullanarak mesajların içeriğine istenilen bir nesne eklenebilmektedir.

3.1 JADE Platformunun Genel Yapısı

JADE platformu aşağıdaki Java paketlerinden oluşur [15].

- *jade.core* paketi sistemin çekirdeğini oluşturur. Agent isimli sınıfı içerir. Yazılacak sistemdeki tüm etmenler bu sınıftan türetilirler. *jade.core.behaviours* isimli alt paketindeki Behaviour sınıfı ise etmenler tarafından yürütülecek görev sınıfları için ana sınıfıdır. Bu pakette çeşitli amaçlara yönelik hazırlanmış davranış-görev sınıfları bulunur.
- *jade.lang.acl* paketi ise FIPA ACL standartlarına uygun olarak etmen haberleşmesini sağlayan sınıfları barındırır.
- *jade.content* paketi uygulamalar tarafından tanımlanan ontolojileri ve içerik dillerini desteklemek için kullanılan sınıflardan ve alt paketlerden oluşur.
- *jade.domain* paketi ve alt paketleri FIPA standartları tarafından AMS ve DF özelliklerini yani yaşam döngüleri ve sarı sayfalar servislerini sağlayan etmen sınıfları tutarlar.
- *jade.gui* paketi oluşturulacak uygulamalar için kullanıcı arabirimi geliştirmekte kullanılan sınıfları barındırır.

- *jade.mtp* paketi sistemlerin JADE platformuna entegre çalışması için gerçeklemeleri gereken Mesaj İletim Protokolüne dair ara yüzleri ve bu ara yüzlerinin gerçeklenmiş bazı örneklerinden oluşur.
- *jade.proto* paketinin içerdiği sınıflar standart etkileşim protokollerini gerçeklerler. Aynı zamanda kullanıcılar tarafından protokol tanımlanması için gereken ana sınıflar da bu pakettedir.
- *jade.wrapper* paketi JADE platformuna üst seviye özellikler eklemelerini ve başka uygulamaların JADE etmenlerini yürütmelerini sağlayan sınıflardan oluşur.

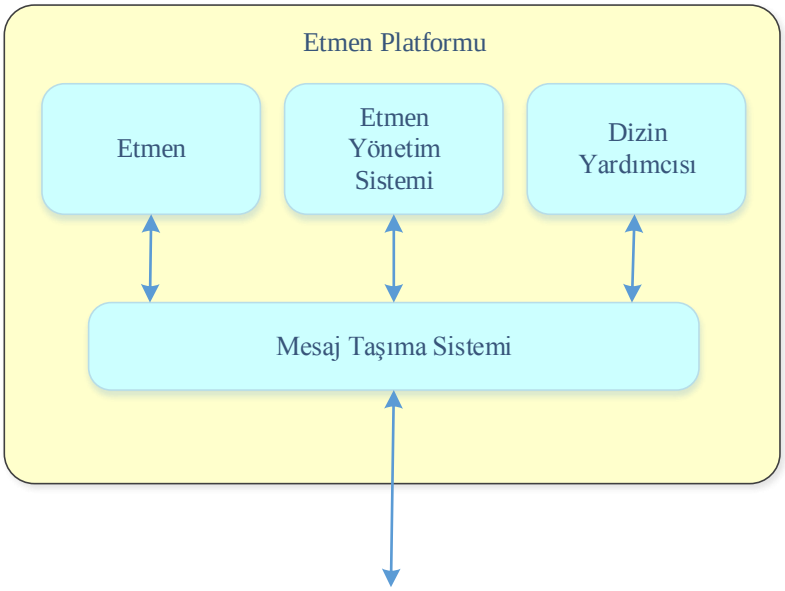
3.2 FIPA

FIPA, etmenler ve etmen tabanlı sistemlerin birlikte çalışabilmesi için gerekli standartları belirleyerek etmen sistemler endüstrisini teşvik etmeyi amaçlayan uluslararası bir organizasyondur [17]. FIPA'nın belirlediği standartlar, soyut bir etmen sistemi mimarisi belirler ve geliştirilecek mimariye en az sayıda kısıt getirmeyi hedefler. FIPA'nın belirlediği standartlarda en çok, ticari ve endüstriyel kullanım amaçlı etmen ortamları hedeflenmiştir [18]. JADE, FIPA tarafından belirlenen soyut etmen mimarisine göre tasarlanmıştır ve bileşenleri bu mimariye göre belirlenmiştir.

3.3 Etmen Platformu

JADE çalışma ortamının her çalışan örneğine Taşıyıcı (Container) adı verilir. Her taşıyıcı birden fazla etmeni aynı anda bulundurabilir. Aktif taşıyıcılar topluluğuna Platform adı verilir. Bir platformda her zaman bir ana taşıyıcı aktif halde bulunur ve diğer taşıyıcılar çalışmaya başladıkları anda platforma kayıt olurlar. Bir platformda açılan ilk taşıyıcı, ana taşıyıcıdır ve daha sonra başlatılan taşıyıcılar normal taşıyıcılardır Normal taşıyıcılar başlatılırken, kayıt olacağı ana taşıyıcının adresinin ona bildirilmesi gereklidir [17].

Etmen platformu, FIPA mimarisinde tanımlanan ve Şekil 3.1'de [15] görülen bileşenlerden oluşur.



Şekil 3.1 : FIPA etmen platformu mimarisi.

AMS (Agent Management System – Etmen Yönetim Sistemi) etmeni tüm etmen platformunun yönetiminden sorumludur. Her sistemde yalnız bir tane AMS bulunur. AMS etmenler için yaşam döngüsü servisleri sunar, etmenlere kimlik atanması (AID) ve durumlarının takip işlemlerini yapar. Platforma bağlanacak her etmen AMS'den geçerli bir kimlik almak zorundadır.

DF (Directory Facilitator – Dizin Yardımcısı) etmenler arası sarı sayfalar servisi sunar.

ACC (Agent Communication Channel – Mesaj İletim Sistemi) platform içinde mesaj iletimi yapısını kuran bileşendir. Bu mesajlaşmaya, uzak platformlarla yapılan mesajlaşma da dâhildir.

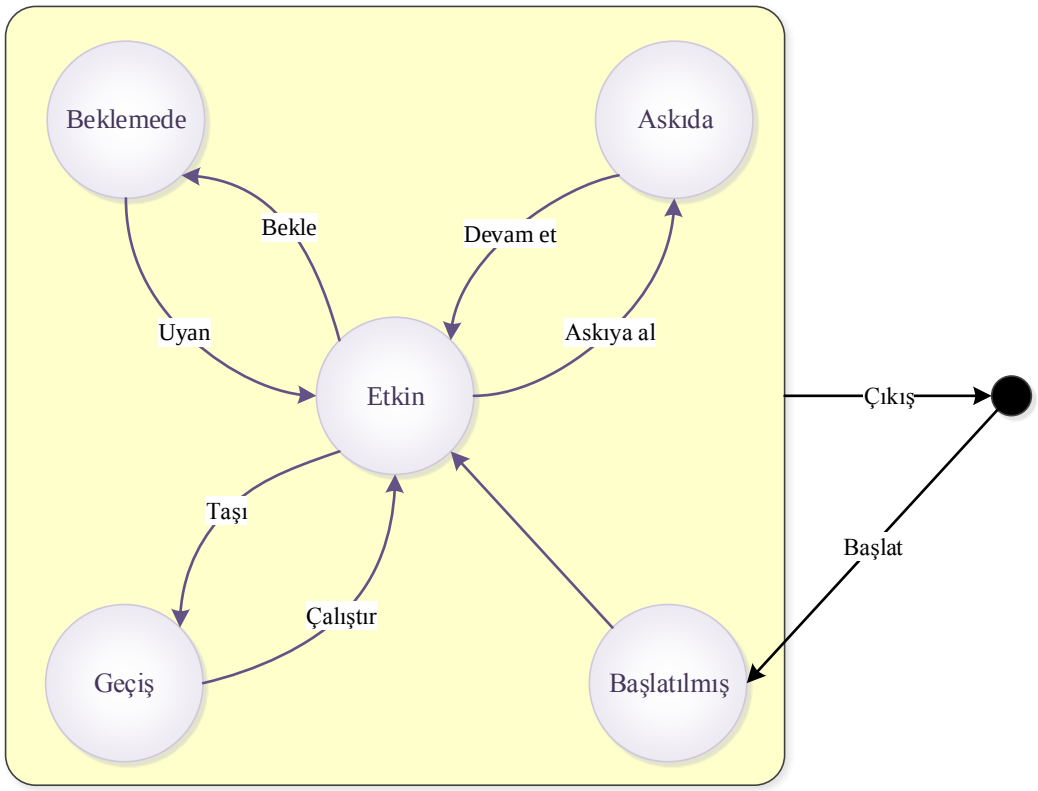
JADE FIPA standartlarıyla ve yukarıda tanıtılan mimariyle tam uyumludur. JADE platformu çalıştırıldığında AMS, DF ve ACC bileşenleri başlatılır ve çalıştırılır. Aynı zamanda bu bileşenlerin her biri farklı bilgisayarlarda çalıştırılabilecek yapıya sahiptirler [15].

3.4 Etmen Yönetimi

JADE platformunda etmenler, *jade.core.Agent* sınıfı kullanılarak başlatılırlar. Bu sınıftan kalıtımla türetilen sınıflar ile özelleştirilmiş etmenler başlatılır. Etmen ilk çalıştırıldığında bu sınıfın *setup()* metodu çağrılır. Bu metot ile etmen, kullanacakları

kaynakları oluşturur, eğer bir servis sunacak ise kendisini DF'ye kayıt ettirir ve diğer başlangıç koşulları ile ilgili işlemlerini yapar. Etmen yok edilirken *takeDown()* metodu çağırılır. Bu metot ile etmen kullandığı kaynakları iade eder ve DF'ye kayıt olduysa bu kaydını siler [15].

FIPA standartlarına göre bir etmen çalışma süresi boyunca çeşitli durumlarda bulunabilir. Durum geçiş diyagramı Şekil 3.2'de [15] verilmiştir.



Şekil 3.2 : Etmenin yaşam döngüsü.

- **Başlatılmış (Initiated):** Etmen oluşturulmuş, fakat AMS'ye kayıt olmamıştır. Henüz bir ismi ve adresi yoktur ve diğer etmenler ile haberleşemez.
- **Etkin (Active):** Etmen oluşturulmuş ve AMS'ye kayıt olmuştur. İsmi ve adresi vardır. Tüm JADE özelliklerine erişebilir durumdadır.
- **Askıda (Suspended):** Etmenin çalıştığı thread ve etmenin davranışları durdurulmuştur.
- **Beklemede (Waiting):** Etmen bloke olmuş durumdadır ve çalışmaya devam etmek için bir koşulun gerçekleşmesini beklemektedir.

- **Silinmiş (Deleted):** Etmenin çalışması tamamen sonlanmıştır başka bir deyişle etmenin AMS kaydı silinmiş durumdadır.
- **Geçiş (Transit):** Bir hareketli etmen başka bir konuma taşınırken bu duruma geçer. Sistem bu etmene gönderilen mesajları tampon belleğe alır ve geçiş tamamlandığında mesajları etmene iletir.

3.4.1 Etmen başlatma

JADE platformunda, yeni bir etmen başlatılırken şu adımlar uygulanır [15]: etmen kurucu metodu çalıştırılır, etmene benzersiz bir isim verilir, yeni etmen AMS'ye kayıt edilir, etmenin durumu etkin yapılır ve son olarak etmenin *setup()* metodu çağırılır. FIPA standartlarına göre bir etmenin evrensel benzersiz bir isime ve başka etmenlerin yeni oluşturulan etmenle haberleşmesi için bir adres kümesine ihtiyacı vardır.

Geliştirici, etmeni başlatmak için *setup()* metodunu kullanır. Bu metodun içerisinde şu işlemler yapılabilir:

- Gerekli görüldüğü durumlarda AMS'ye kayıt edilmiş veriler değiştirilir.
- Gerekli görülen durumlarda, etmenin açıklaması ve ortama sunduğu servisler ayarlanır. Ayrıca ihtiyaç duyulursa etmen birden çok DF'ye kayıt ettirilir.
- Görev kuyruğuna *addBehaviour()* metodu kullanılarak yeni davranışlar eklenir.

Etmeni başlatmak için kullanılan *setup()* metodunda en az bir adet görev tanımlanmalıdır. Bu metodun sonlanması ile otomatik olarak hazır davranış kuyruğundan alınan ilk davranış çalıştırılır ve davranışlar sonlandıkça kuyruktan alınan sıradaki davranış çalıştırılır. Bir davranış çalışırken başka bir davranış onun çalışmasını kesemez. Etmene davranış ekleme ve etmenden davranış çıkartma metotları etmenin görev kuyruğunu yönetmek için kullanılır.

3.4.2 Etmen sonlandırma

Etmenin *doDelete()* metodu çağrıldığında etmen sonlandırılır. Etmenin kullandığı kaynakları temizlemesi ve kayıt olduğu servislerden kaydını silmesi işlemlerinin yapılabilmesi için otomatik olarak *takeDown()* metodu çağırılır, *takeDown()* metodunun çağrılması ile etmen yok edilir. Bu metot çağrıldığı sırada etmenin AMS kaydı silinmemiş haldedir, başka etmenlere mesaj gönderebilir ve sonlandırılmakta

olduğunu haber verebilir. Metot sonlandıktan sonra etmen için başlatılmış olan thread sonlandırılır [15].

3.4.3 Etmen davranışları

Bir etmenin görevleri yerine getirmesi, davranış (behaviour) tanımı aracılığıyla olur. Bir davranış *jade.core.Behaviours.Behaviour* sınıfının genişletilmesi ile tanımlanır. Bu davranışın etmen tarafından yürütülebilmesi için etmen sınıfı içerisinde *addBehaviour()* metoduyla etmene eklenmesi gerekir. Davranışlar, etmenin açılışında çalıştırılan *setup()* metodunda veya daha sonra başka bir davranış içerisinden de eklenebilir.

Behaviour sınıfını genişleten her sınıf, soyut bir metot olan *action()* metodunu gerçeklemek zorundadır. *action()* metodunda, davranışın çalışması sırasında yapılacak işler tanımlanır. Bir davranışın *action()* metodu sonlanmadan başka bir davranışın *action()* metodu çalıştırılmaz.

JADE platformu, davranışlarla çalışmayı kolaylaştırmak için farklı özelliklere sahip çeşitli davranış sınıfları bulundurur. Bu sınıflar kullanım amaçlarına göre doğrudan kullanılabilir veya özelleştirilmiş bir çalışma şekli için ilgili soyut metotları gerçeklenerek genişletilebilir.

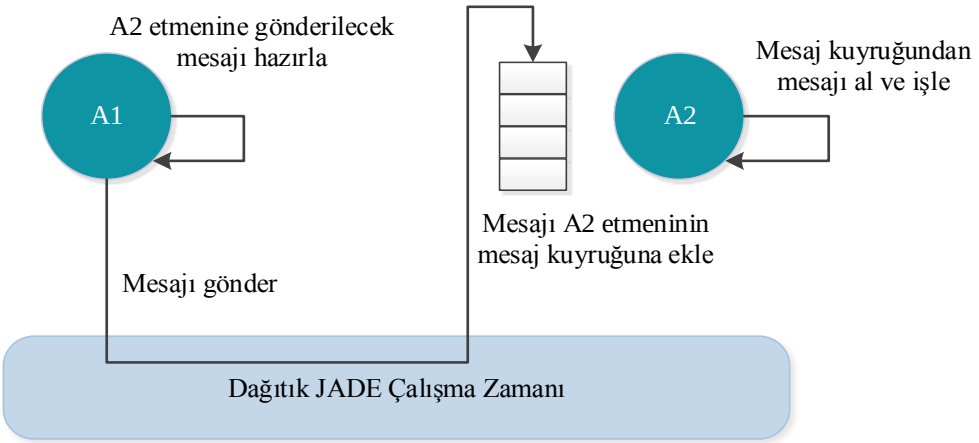
- **Behaviour:** Behaviour sınıfının genişletilmesiyle, genel amaçlı davranış sınıfları oluşturulur. Oluşturulan sınıf, içerisinde bir durum değişkeni tutar ve bu değişkenin değerine göre yapacağı işleme veya davranışın sonlanmasına karar verir [17].
- **OneShotBehaviour:** Bir defa çalışan ve hemen sonlanan davranışlar oluşturulur [17]. Bu davranışlar tüm işlemlerini *action()* metodunun sadece bir defa çağrılacağını göz önünde bulundurarak yapmalıdır.
- **CyclicBehaviour:** Bu tanımdaki davranışlar, hiç bir zaman sonlanmazlar. Davranışın her çağrılmasında *action()* metodundaki aynı işlemler tekrarlanır [17]. Davranış ancak *removeBehaviour()* metodu ile sonlandırılır.
- **TickerBehaviour:** Belirli zaman aralıklarıyla gerçekleştirilmesi gereken görevler için kullanılır [17]. Davranışa parametre olarak verilen zaman aralığı her dolduğunda, *onTick()* metodu çağırılır. Bu metot sonlandığında, davranış zaman aralığı tekrar dolduğunda çalışacak şekilde görev kuyruğuna eklenir.

- **WakerBehaviour:** Belirli bir süre geçtikten sonra veya belirli bir tarih ve saatte gerçekleştirilmesi istenen görevler için kullanılır [17]. Davranışın çalışması istenen tarih ve saat geldiğinde, davranışın *onWake()* metodu çalıştırılır ve bu metodun çalışması bittikten sonra etmen sonlandırılır.
- **SequentialBehaviour:** Birden fazla davranışın sırayla çalıştırılmasını sağlayan davranıştır. Bu sınıfın *addSubBehaviour()* metodu kullanılarak çalıştırılması istenen davranışlar sırayla eklenir. Davranış çalışmaya başladığında tüm alt davranışlarını sırayla çağırır. Bir alt davranış sonlandığında sıradaki alt davranışa geçilir [17].
- **ParallelBehaviour:** Birden fazla davranışın paralel olarak çalıştırılmasını sağlayan davranıştır. Bu sınıfın *addSubBehaviour()* metodu ile çalıştırılması istenen davranışlar eklenir.
- **FSMBehaviour:** Birden fazla davranışın, sonlu durumlu makine modeline göre çalıştırılmasını sağlayan davranıştır. Sonlu durumlu makine modeline uygun çalışmasını sağlamak için bir durum geçiş tablosu tutulur. Bu durum geçiş tablosundaki her düğüm bir alt davranışı temsil eder. Alt davranış sonlandığında çalıştırılacak yeni davranışı belirlemek için durum tablosundaki değerlerle karşılaştırılır. Karşılaştırma sonucunda çalıştırılacak yeni alt davranış belirlenir veya eğer sonlanma durumuna ulaşıldıysa ana davranış sonlandırılır [17].

Yapılan çalışmada OneShotBehaviour, CyclicBehaviour, TickerBehaviour, SequentialBehaviour ve ParallelBehaviour sınıfları kullanılmıştır.

3.5 Etmen Haberleşmesi

Etmen haberleşmesi JADE'in en temel özelliğidir ve FIPA iletişim modeline uygun olarak tasarlanmıştır. İletişimin temelinde eş zamanlı olmayan mesaj taşıma işlemi vardır. Her etmenin, başka etmenlerin gönderdiği ve JADE çalışma zamanı tarafından iletilen mesajların ulaştığı etmen mesaj kuyruğu vardır. JADE haberleşme mimarisi, ana hatlarıyla Şekil 3.3'te [17] verilmiştir. Diğer etmenlerden gelen mesajlar dağıtık JADE çalışma zamanı üzerinden geçerek mesajın gönderildiği etmenin mesaj kuyruğuna gelir. İlk gönderilen mesaj ilk önce işlenmek üzere gelen mesajlar mesaj kuyruğundan alınarak sırayla işlenir.



Şekil 3.3 : JADE haberleşme mimarisi.

3.5.1 FIPA iletişim modeli

Etmen sistemleri için oluşturulmuş FIPA modelinin merkezinde, etmenlerin uygulama tarafından talep edilen işlemleri yapabilmeleri için birbirlerine anlamlı mesajlar aktarabilmesini sağlayan etmen haberleşmesi bulunur [18]. Bu nedenle etmenler arasında mesaj aktarım yöntemi, mesajların temsil edilme yöntemi (XML, ikili nesneler, gibi) ve diğer özellikleri tanımlayan etmen mesajlaşma dili (ACL) oluşturulmuştur. FIPA standartlarına göre bir ACL mesajında bulunan alanlar aşağıda verilmiştir [19]:

- performative: Mesajın iletişimsel eylemini bir başka deyişle mesajın türünü belirtir.
- sender: Mesajı gönderen etmeni belirtir.
- receiver: Mesajın gönderileceği etmeni veya etmenler kümesini belirtir.
- reply-to: Mesaja verilecek cevabın gönderilmesi gereken etmeni belirtir.
- content: Mesajın içeriğini bulundurur. İçerik bilgisi alıcı etmen tarafından yorumlanır.
- language: Mesajın içeriğini bulunduran content alanının biçimsel dilini (SQL, FIPA-SL, Prolog, vb.)belirtir.
- encoding: content alanının karakter kodlamasını belirtir.
- ontology: Mesajdaki content alanındaki sembollere anlam verebilmek için hangi kelimeler kümesinin kullanıldığını belirtir.
- protocol: Eğer mesaj FIPA tarafından tanımlanmış bir protokole göre gönderilirsa, bu protokolü belirten alandır.

- conversation-id: Etmenin, yaptığı farklı mesajlaşmaları birbirinden ayırt etmek için kullandığı alandır.
- reply-with: Alıcı etmenin cevap verirken kullanması gereken ifadeyi belirtir.
- in-reply-to: Mesajın, hangi tipten bir mesaja cevap olarak gönderildiğini belirtir.
- reply-by: Alıcı etmenin aldığı mesaja son cevap verme tarihini belirtir.

FIPA standartlarına göre bir mesajın iletişimsel eylem tipi başka bir deyişle türü aşağıdakilerden birisi olabilir [20]:

- accept-proposal: Daha önce propose ile yapılan bir önerinin kabul edildiğini belirtir.
- agree: Daha önce request ile yapılması istenen bir eylemin yapılmasının kabul edildiğini belirtir.
- cancel: Bir etmenin, diğer etmene daha önce gönderdiği bir işlemin yapılması isteğinin artık geçerli olmadığını belirtmesini sağlar.
- CFP: Bir etmenin, mesajı alan etmenlerden bir öneri beklediğini belirtir.
- confirm: Göndericinin, alıcı etmene bir önermenin doğru olduğunu bildirdiğini belirtir.
- disconfirm: Göndericinin, alıcı etmene bir önermenin yanlış olduğunu bildirdiğini belirtir.
- failure: Yapılması istenen bir eylemin başarısız olduğunu belirtir.
- inform: Göndericinin, alıcı etmene bir önermenin doğru olduğunu bildirdiğini belirtir.
- inform-if: Göndericinin, alıcı etmene bir önermenin doğru veya yanlış olduğunu bildirdiğini belirtir.
- not-understood: Göndericinin, alıcının yaptığı bir eylemi anlamadığını belirtir.
- propagate: Alıcı etmenin, mesajın içeriğine gömülü bir mesajı, belirtilen etmenlere göndermesi gerektiğini belirtir.
- propose: Göndericinin, alıcı etmenin belirli koşullar altında bir eylem yapmasını önerdiğini belirtir.
- proxy: Alıcı etmenin, verilen bir açıklamaya göre gerekli etmenleri seçerek gömülü olan mesajı seçtiği etmenlere göndermesi gerektiğini belirtir.

- query-if: Göndericinin, alıcı etmene bir önermenin doğru olup olmadığını sorduğunu belirtir.
- query-ref: Göndericinin, nesnenin kaynağını belirten bir ifade ile alıcı etmenden bir nesneyi istediğini belirtir.
- refuse: Yapılması istenen bir eylemin reddedildiğini belirtir.
- reject-proposal: Daha önce yapılan bir önerinin kabul edilmediğini belirtir.
- request: Göndericinin, alıcı etmenden bir eylem veya başka bir iletişimsel eylem yapmasını istediğini belirtir.
- request-when: Göndericinin, alıcı etmenden bir koşul gerçekleştiğinde bir eylem yapmasını istediğini belirtir.
- request-whenever: Göndericinin, alıcı etmenden bir koşul her gerçekleştiğinde bir eylemi tekrar yapmasını istediğini belirtir.
- subscribe: Göndericinin, alıcıdan belirli bir değer her değiştiğinde haberdar edilmesini istediğini belirtir.

3.5.2 ACLMessage sınıfı

JADE kütüphanesinde bir ACL mesajı, jade.lang.acl.ACLMessage sınıfı ile tanımlanır. Bu sınıf, FIPA modeline göre gerekli olan tüm alanları bulundurur. Bu alanların her biri için *get()* ve *set()* metotları gerçeklenmiştir. FIPA tarafından tanımlanan tüm iletişimsel eylem tipleri yani mesaj türleri, ACLMessage sınıfı içerisinde birer sabit değişken olarak tanımlanmıştır [15].

3.5.3 Mesaj gönderme ve alma

Mesaj gönderme işlemi, ACLMessage sınıfının *send()* metoduyla, mesaj alma işlemi ise *receive()* ve *blockingReceive()* metotlarıyla yapılır. *blockingReceive()* metodu adından da anlaşılacağı üzere mesaj gelene kadar etmenin ana thread yapısını bloke eder ve diğer davranışlarını yapmasını engeller, bu nedenle kullanımında dikkatli olunmalıdır. *receive()* metodu ise mesaj gelene kadar null değerini döndürür fakat diğer davranışların çalışmasını engellemez.

4. ETMEN SİSTEMLERİNDE KOALİSYON KURULUMU

4.1 Etmen Sistemleri ve Koalisyonlar

Koalisyonun sözlük anlamı "Çeşitli güçlerin bir araya gelmesiyle oluşturulan birlik, ortak yönetim" olarak tanımlanmıştır [21]. Etmen sistemlerinde koalisyon da bu tanıma uygun olarak etmenlerin bir araya gelmesiyle oluşturulan birliktir. Etmen sistemlerinde koalisyon kurulmasının amacı; tek bir etmen tarafından yerine getirilemeyen ya da tek bir etmenin efektif olarak yerine getiremeyeceği görevlerin etmen koalisyonu aracılığıyla etkin bir şekilde yerine getirilmesidir.

Dağıtık sistemlerde etmen koalisyonu kurmak için görev paylaştırma metotlarının anlatıldığı bir çalışmada [1] etmen sistemlerinin çeşitlerinden bahsedilmiştir:

- Distributed Artificial Intelligence (DAI) : Görevleri çalıştırmak için birkaç etmenin ilişkili olduğu sistemdeki problemlerle ilgilenmektedir.
- Multi-Agent Systems (MAS) : İşbirliği aracılığıyla her etmenin kendi kişisel faydasını arttırmaya çalıştığı sistemlerdir.
- Distributed Problem Solving (DPS) : Kişisel faydadan ziyade sistemin genel faydasını arttırmaya yönelik etmenlerin işbirliğinde bulunduğu durumdur.

Bu çalışmada sistemin genel faydası öncelikli olduğu için DPS yaklaşımı kullanılmıştır. O. Shehory, S. Kraus [1] koalisyon kurulum algoritması temel alınmış ve algoritmanın üzerinde gerçekleneceği çoklu etmen tabanlı bir yürütme ortamı tasarlanıp gerçeklenmiştir.

Etmen koalisyonları kurulumu için farklı yaklaşımlar önerilmiştir [1]; ayrık koalisyonlar, kesişen koalisyonlar, öncelik sıralı görevler için oluşturulan koalisyonlar. Yapılan çalışmada ayrık koalisyon kurulumu gerçeklenmiştir. Ayrık koalisyonlarda aynı etmenin birden fazla koalisyonda görev alma durumu bulunmamaktadır, ancak katıldığı görev tamamlandıktan sonra yeni görevler için oluşturulacak koalisyona katılma hakkı bulunmaktadır.

Ayrıca aynı çalışmada [1] kurulumu tamamlanmış olan bir koalisyona sonradan etmen eklemenin koordinasyon ve iletişim maliyetlerinden dolayı faydalı olmayacağı savunulmuştur. Bu durum göz önünde bulundurularak yapılan çalışmada koalisyon kurulduktan sonra o koalisyondaki etmenler sabit tutulmuştur.

4.2 Koalisyon Kurulum Algoritması

Bu bölümde gerçeklenen Shehory-Kraus algoritması ayrıntılı olarak ele alınacaktır. Sistemin çalışacağı ortam için bazı tanımlar ve kabuller yapılmıştır.

Kabuller:

- Etmenler birbirleri ile iletişim kurabilir, tartışabilir, anlaşma yapabilir.
- Bir koalisyona yeni etmen ekleme maliyetinden dolayı koalisyonları genişletme durumu kapsam dışı tutulur.
- Bir koalisyon aynı anda tek bir görev üzerinde çalışabilir (ayrık koalisyonlar).
- Görevler arası bağımlılık olmadığı kabul edilir.

Tanımlar:

Etmen kümesi *n* tane etmenden oluşur, $N = \{A_1, A_2, \dots, A_n\}$. Görev kümesi *m* tane birbirinden bağımsız görevden oluşur, $T = \{t_1, t_2, \dots, t_m\}$. Her etmen ve her görev için birer kabiliyet vektörü tanımlanır. Etmenler için olan kabiliyet vektörlerinde, her bileşen o etmenin bir görevi yerine getirebilmesi için gerekli bir kabiliyeti ile ilgili sayısal değeri içerir. Görevler için tanımlanan kabiliyet vektörlerinde ise her bileşen o görevin yerine getirilebilmesi için gerekli bir kabiliyetin sayısal değerini içerir. Kabiliyet vektörleri, negatif olmayan reel sayılardan oluşur. Bir görevin gerçekleştirilmesi ile kazanılan fayda, hesaplamalardaki karmaşıklığı düşürmek için kaynak miktarının lineer bir fonksiyonu olarak ele alınmıştır.

- Etmenlerin kabiliyet vektörü : $B_i = \langle b_1^i, b_2^i, \dots, b_r^i \rangle$
- Görevlerin kabiliyet vektörü: $B_l = \langle b_1^l, b_2^l, \dots, b_r^l \rangle$
- Bir *C* koalisyonunun kabiliyet vektörü B_c, bu koalisyona katılan üyelerin kabiliyetleri toplamından oluşur.
- Bir *C* koalisyonunun bir *t* görevini yapabilmesi için, görevin kabiliyet vektörü B_t 'nin her bileşeni için $b_i^t \leq b_i^C$ koşulunu sağlamalıdır, $0 \leq i \leq r$.

- İşbirliği yapan *C* koalisyonu üyelerinin birleşik faydasına (joint utility) koalisyon değeri denir, *V* ile gösterilir. Bu *V* değeri koalisyona katılan etmenlerin kabiliyet vektörlerinden ve koalisyondaki etmen sayısından doğrudan etkilenir.
- Koalisyon maliyeti *c*, koalisyon değerinin çarpmaya göre tersi olarak hesaplanır.
- Koalisyonel rasyonalite etmenlerin koalisyon değerini artırmalarına, koalisyon maliyetini azaltmalarına öncülük eder [1]. Bu çalışmada etmenlerin grup-rasyonel (group-rational) olduğu kabul edilmiştir. Etmenler kendi kişisel faydalarını gözetmeden faydalı olacakları koalisyonu oluşturur. Görevleri yerine getirdiklerinde fayda sağlarlar ve ortak faydayı artırırlar.
- Koalisyon kurma sürecinde etmen popülasyonu sabittir. Etmenler tüm görevleri ve diğer tüm etmenler hakkında bilgi sahibidirler. Sadece bir koalisyona katılmış olan üyeler belirli bir görevin yerine getirilmesi için gerekli detayları bilirler.
- *C* koalisyonu dışındaki etmenlerin *C* içindeki koalisyonel aktivitelerin detaylarını bilmeleri gerekmez. Her etmen katıldığı koalisyon ile ilgili detayları bilir.

4.3 Küme Örtüleme ve Küme Bölüntüleme (Set Covering ve Set Partitioning)

Görev kümesindeki görevleri, etmen kümesindeki etmenlerden gruplar oluşturup paylaştırarak tüm görevlerin yapılmasını sağlama problemi küme örtüleme ve küme bölüntüleme problemlerine benzemektedir [1]. Burada görevleri etmen gruplarına atamak "Set Covering Problem (SCP)", etmenleri alt gruplara ayırmak "Set Partitioning Problem (SPP)" olarak düşünülebilir.

- $N = \{A_1, A_2, \dots, A_n\}$ etmen kümesinin alt kümeleri $S = \{C_1, C_2, \dots, C_m\}$ öyle ki; $C_j \subseteq N$ ve $S \subseteq 2^N$.
- S' kümesi set-cover ve $S' \subseteq S$ öyle ki; $\bigcup_{C_j \in S'} C_j = N.$

S' kümesindeki koalisyon üyelerinin birleşimi tüm etmenlerin bulunduğu küme olan *N* ile eşittir, bu da S' kümesinin set-cover olduğunu gösterir.

S' kümesinin elemanları aynı zamanda ikili ayrık (pairwise disjoint) kümelerse, diğer bir deyişle $\forall C_i, C_j \in S',\ i \neq j$ ve $C_i \bigcap C_j = \emptyset$ ise S' kümesi aynı zamanda set partitioning kümesidir.

4.4 Shehory-Kraus Koalisyon Kurulum Algoritması

[1]'de verilen algoritma greedy distributed küme örtüleme ve küme bölüntüleme algoritmasıdır. Gerçeklenen yürütme ortamında, iş yapma yeteneği olan etmenler tanımlanan görevleri yerine getirmek üzere koalisyonlar kurarlar. Kullanılan algoritma DPS sistemine uygundur, diğer bir deyişle etmenler kendilerinin değil, sistemin genel faydasını artırmaya odaklı çalışırlar.

Algoritma üç aşamada gerçeklenmektedir:

1. Aşama: İlk aşamada olası tüm koalisyonlar hesaplanır.

2. Aşama: Oluşturulan tüm koalisyonların değerleri hesaplanır. Böylece aralarından en uygun olanı tercih edilebilir.

3. Aşama: Bu aşamada ise, etmenlerin katılacakları koalisyonu seçme iterasyonları yapılır.

Gerçeklenen çoklu etmen altyapısı, algoritmanın her aşamasında görevleri yerine getirecek çok sayıda işçi etmenin ve sistemin söz konusu algoritmayı gerçekleyecek şekilde çalışmasını mümkün kılan, değişik görev ve sorumluluklara sahip etmenlerin bir arada çalışmasına izin veren bir sistemdir. Tasarlanan haberleşme altyapısı ile etmenler birbirleriyle yoğun bir etkileşim içinde çalışırlar; ürettikleri bilgileri ve koalisyona katılım tercihlerini birbirlerine iletirler.

Her aşama akış şemaları ile birlikte ayrıntılı olarak ele alınmıştır. Akış şemalarında kullanılan tanımlar şöyledir:

k : İzin verilen en büyük koalisyonun büyüklüğü

L_i : Etmenin üzerinde çalışacağı koalisyon listesi (Long-term commitment list)

T : Görev kümesi

L_i^{cr} : Her A etmeni için o anda değeri hesaplanan koalisyonun bulunduğu küme

B_c^{pc} : Koalisyonun potansiyel kabiliyet vektörü (Coalitional Potential Capabilities vector)

E_c : C koalisyonu T görev listesindeki görevleri yaptığında alacağı sonuçların kümesi (Expected outcomes)

e_j : C koalisyonu t_j görevini yaptığında beklenen net değer

V_c : Koalisyon değeri

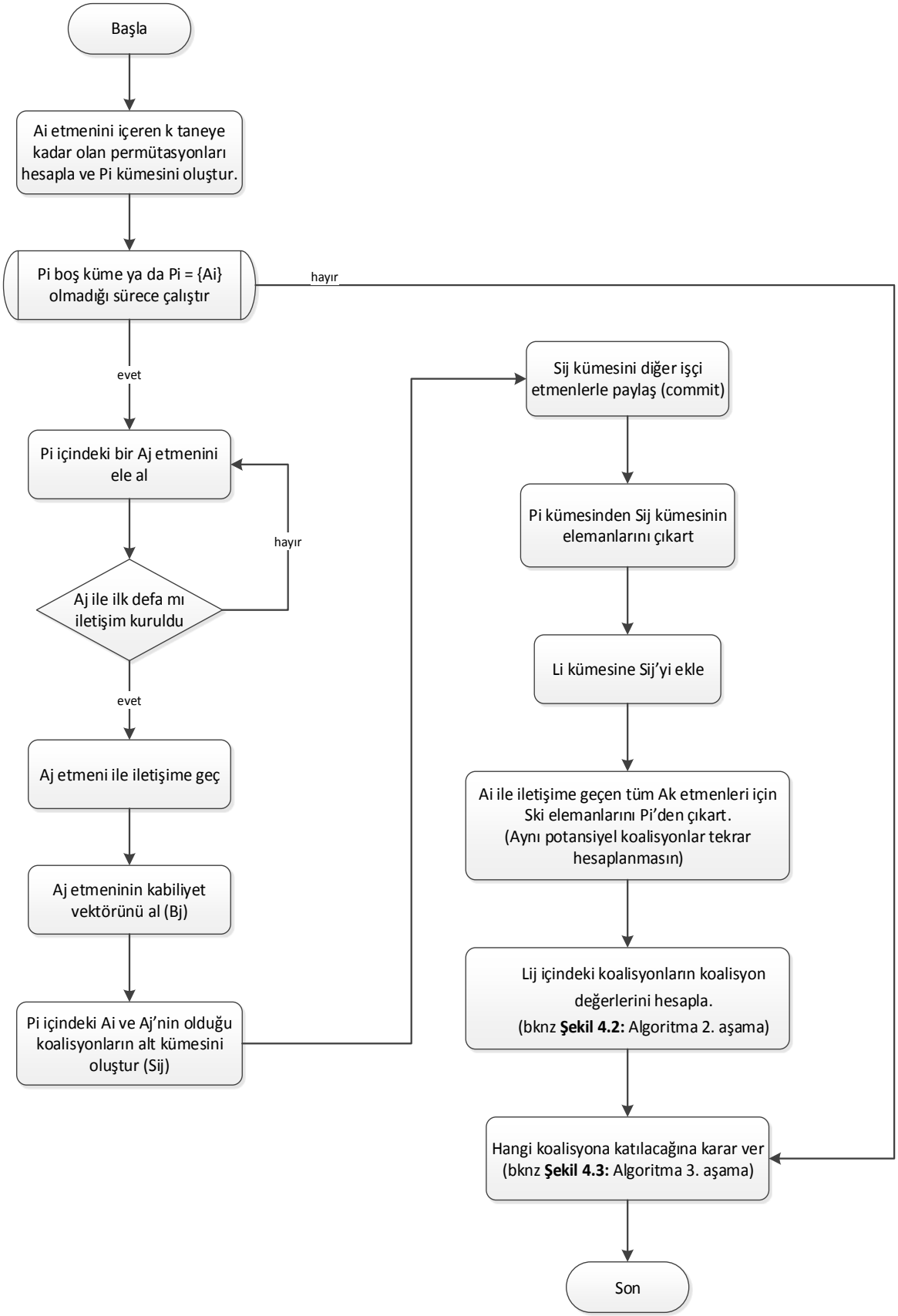
$|C_i|$: Koalisyonun boyutu (koalisyona katılacak işçi etmen sayısı)

$w_i = c_i / |C_i|$: Koalisyonel ağırlık

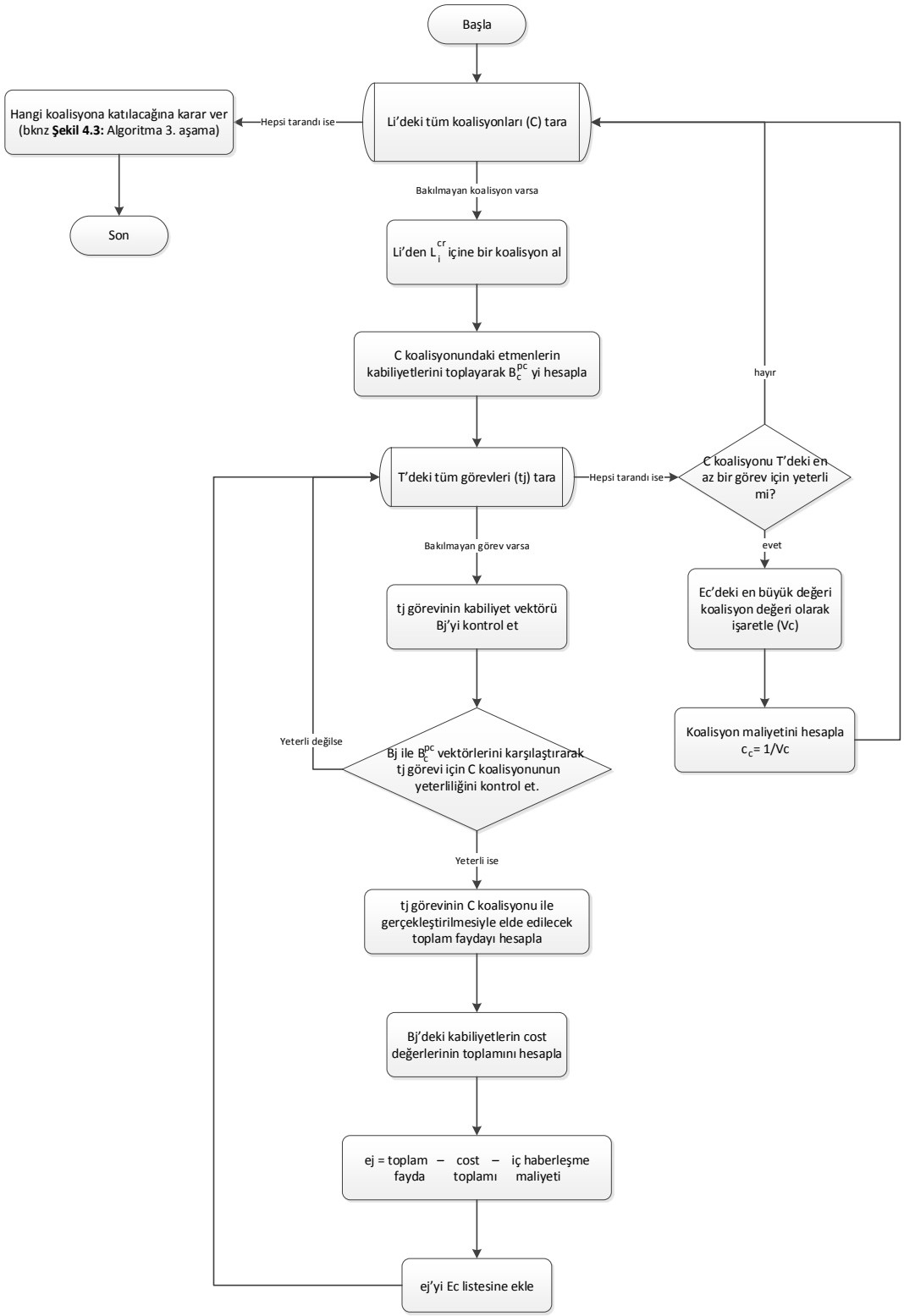
1. Aşama: Koalisyon kurulumuna katılacak her etmen Şekil 4.1'de verilen akışı takip etmektedir. Algoritmanın bu aşamasında, etmen öncelikle diğer etmenlerle birlikte hangi koalisyonları kurabileceğini bulmak için permütasyonları hesaplar. Permütasyonlardaki etmen sayısı en fazla k tane olmalıdır; bu değer en başta belirlenir, sayının büyük olması etmenler arası iletişim maliyetinden dolayı önerilmemektedir [1]. Etmen belirlediği permütasyonlar içinden kendisinin yer almadığı permütasyonları çıkararak bir potansiyel koalisyon listesi elde eder. Böylece her etmen sadece kendisinin dâhil olduğu olası koalisyonlarla ilgilenmiş olur.

Potansiyel koalisyon listesindeki her bir koalisyon için algoritmanın 2. aşamasında kullanılmak üzere diğer işlemler gerçekleştirilir. Koalisyon değerlerinin hesap edilme aşamasına geçmeden, her etmen diğer etmenlerle iletişime geçer ve onlardan edindiği bilgiler ışığında, diğer etmenlerde bulunan potansiyel koalisyonları kendi potansiyel koalisyon listesinden çıkartır.

2. Aşama: 1. Aşamayı tamamlayan her etmen Şekil 4.2'de verilen akışı izler. Bu aşamada potansiyel koalisyon listesindeki her bir koalisyonun potansiyel kabiliyet vektörünün (koalisyondaki etmenlerin kabiliyet vektörleri toplamı) görev listesindeki görevlerin kabiliyet vektörleri için yeterli olup olmadığı iteratif olarak kontrol edilir. Yeterli olan koalisyonlar için görevler yerine getirildiğinde elde edilecek değer, diğer bir deyişle beklenen değer, hesaplanır. Beklenen değer; kabiliyet vektöründeki bileşenlerin lineer bir fonksiyonuyla hesaplanır. Beklenen değerler içinden en büyük değer seçilir ve ona karşılık gelen koalisyon maliyeti hesaplanır. Bu aşama tamamlandığında her bir koalisyonun koalisyon değeri ve maliyeti hesaplanmış olur. Bu değerler algoritmanın 3. aşamasında etmenlerin koalisyon seçimine yön verecektir.

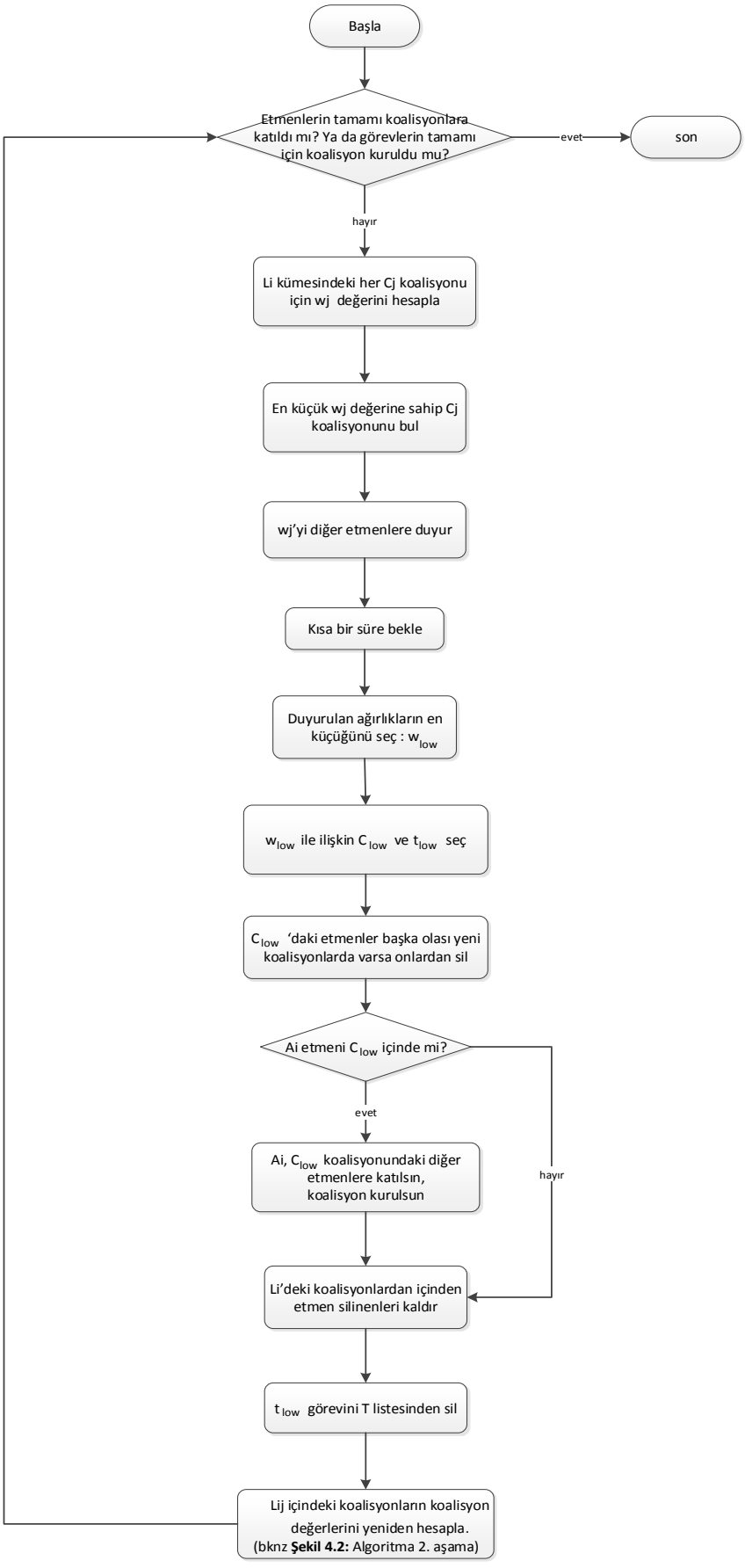


Şekil 4.1 : Algoritma 1. aşama.



Şekil 4.2 : Algoritma 2. aşama.

3. Aşama: Algoritmanın üçüncü aşamasında etmenler Şekil 4.3'te verilen akışı izleyerek koalisyonel ağırlıkları hesaplarlar. Etmenler birbirleriyle iletişimde bulunarak en küçük koalisyon ağırlığına sahip olan koalisyonu belirlerler.



Şekil 4.3 : Algoritma 3. aşama.

Eğer etmen en küçük koalisyon ağırlığına sahip olan koalisyondaki etmen listesinde yer alıyor ise, o koalisyona katılacağını diğer etmenlere bildirir. Oluşturulan koalisyonlar ve o koalisyonlara ait görevler teker teker listelerden kaldırılır. Bu işlemler tüm görevlere bir koalisyon atanana kadar, ya da, her etmen bir koalisyona katılana kadar devam eder. Iterasyon sona erdiğinde, hangi etmenin hangi koalisyona katılacağı belirlenmiş olur ve etmenler görevleri yerine getirmeye hazır olurlar.

4.5 Sistemin Mimari Tasarımı

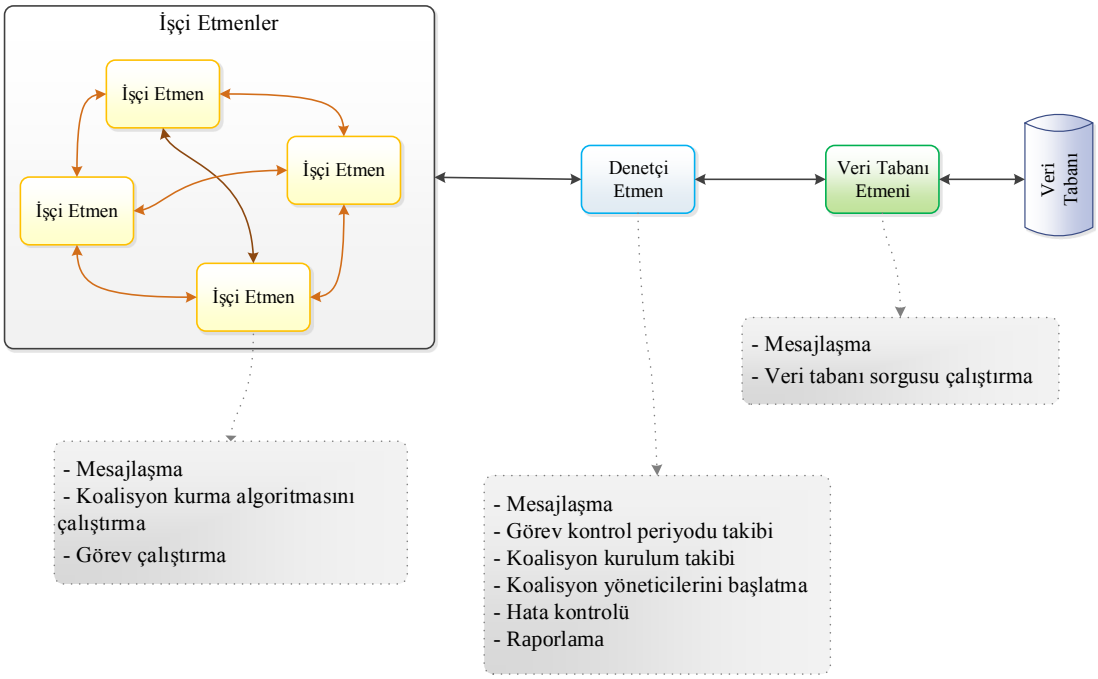
Gerçekleştirilen koalisyon kurulumu ve yürütme ortamının tasarımında yer alan etmenlere çeşitli roller atanmıştır ve her biri uzmanlaştığı işi yapmaktadır. Sistemde görev yapan ana etmenler şunlardır:

- Denetçi etmen
- Veri tabanı etmeni
- Koalisyon yöneticisi etmen(ler)i
- İşçi etmen(ler)

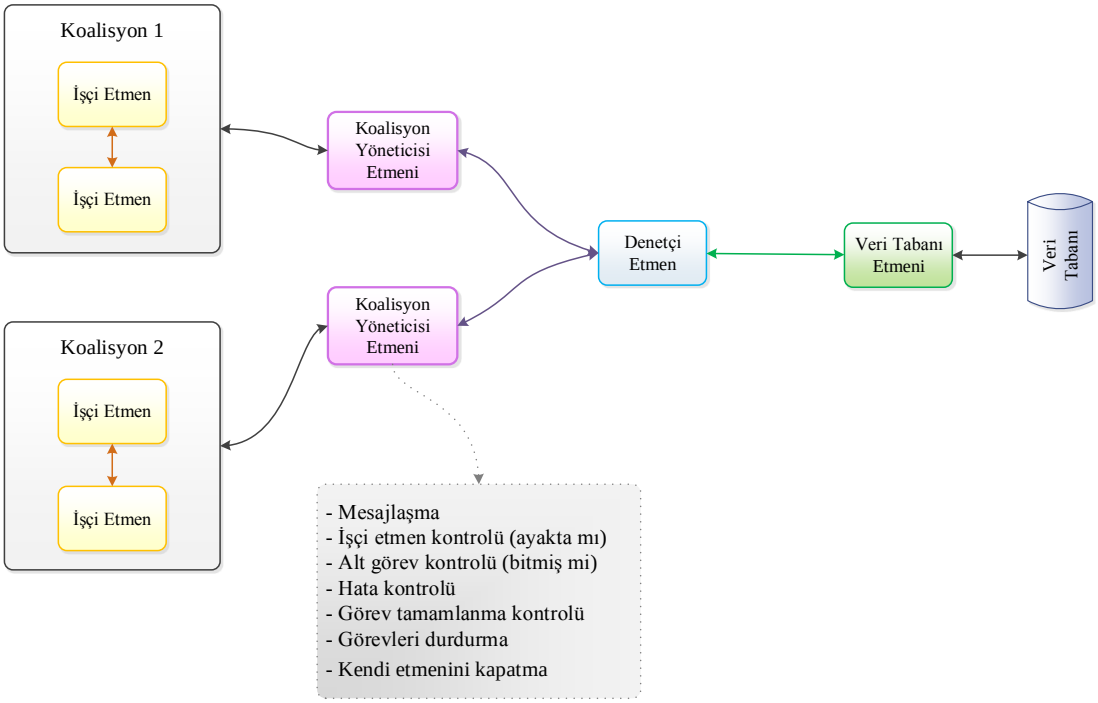
Aşağıdaki bölümlerde etmen türleri ve uzmanlaştıkları görevler ayrıntılı olarak anlatılmıştır.

Yapılan çalışmanın mimari tasarımı, etmenlerin birbirleriyle etkileşimi dikkate alındığında, koalisyonlar kurulmadan önce ve koalisyonlar kurulduktan sonra farklılık gösterir. Koalisyonlar kurulmadan önce (Şekil 4.4) işçi etmenler sadece birbirleriyle ve denetçi etmenle iletişim halindedirler. Denetçi etmen de, işçi etmenlerle ve veri tabanı etmeni ile iletişim halindedir. Veri tabanı ile iletişimde olan tek etmen veri tabanı etmenidir.

Koalisyonlar kurulduktan sonra (Şekil 4.5) koalisyon yöneticisi etmenleri devreye girer, denetçi etmen ile işçi etmenlerin iletişimi kesilir. İşçi etmenler sadece bulundukları koalisyondaki işçi etmenlerle ve bulundukları koalisyonun koalisyon yöneticisi etmeniyle iletişimde bulunurlar. Koalisyon yöneticisi etmenleri denetçi etmenle iletişim halindedir, bu sayede denetci etmen işçi etmenlerle doğrudan iletişimde olmasa bile koalisyon yöneticisi etmenleri aracılığıyle işçi etmenlerle ilgili gerekli verileri alır. Bu yapı her koalisyonun kendi içerisinde yönetimini kolaylaştırır. Veri tabanı etmeni yine veri tabanıyla ve denetçi etmenle iletişim halindedir.



Şekil 4.4 : Koalisyonlar kurulmadan önceki etmen etkileşimi.



Şekil 4.5 : Koalisyonlar kurulduktan sonraki etmen etkileşimi.

4.5.1 Veri tabanı tasarımı

Yapılan çalışmadaki bütün veriler doğal XML [22] veri tabanı olan eXist [23] veri tabanında tutulmaktadır. XML tanımları için XSD [24] tanımları yapılmıştır. Veri tabanı sorguları için de XQuery [25] sorgu dili kullanılmıştır.

İşçi etmen tanımları, görev tanımları, kullanıcı tanımları ve çalışma raporlarının veri tabanında tutulmasına karar verilmiştir. Bu tanımlar veri tabanında xmls dizini altında ilgili XML dosyasında tutulmaktadır, her bir tanımın ayrıntısı aşağıda verilmiştir:

İşçi etmenler (employeeAgents.xml):

İşçi etmenlerle ilgili tanımların tutulduğu veri dosyasıdır, Çizelge 4.1'de detaylandırılmıştır, Ek A1'de örnek dosya içeriği verilmiştir.

Çizelge 4.1 : İşçi etmen bilgileri.

Alan	Açıklama	Örnek
ID	İşçi etmenlerin tekil id numarası	1
localName	Etmenin adı	Agent1
platformName	Etmenin bulunduğu platform adı	employeePlatform
address	Etmenin bulunduğu adres bilgisi	http://merve-laptop:5600/acc
agentState	Etmenin durumu	ACTIVE
capabilityVector	Etmenin kabiliyet vektörü, kabiliyet vektöründe her bir kabiliyetin türü, değeri ve birimi bulunmaktadır.	capabilityType: AMOUNT capabilityValue: 4000 capabilityValue: kg

Görevler (tasks.xml):

Görev tanımlarının tutulduğu veri dosyasıdır, Çizelge 4.2'de detaylandırılmıştır, Ek A2'de örnek dosya içeriği verilmiştir.

Çizelge 4.2 : Görev bilgileri.

Alan	Açıklama	Örnek
ID	Görevlerin tekil id numarası	1
taskName	Görevin adı	Task1
taskState	Görevin durumu	WAITING
capabilityVector	Görevin kabiliyet vektörü, kabiliyet vektöründe her bir kabiliyetin türü, değeri ve birimi bulunmaktadır.	capabilityType: AMOUNT capabilityValue: 6000 capabilityValue: kg

Raporlama (report.xml):

Her çalışma sonrasında o çalışmaya ait ayrıntılı raporlamanın tutulduğu veri dosyasıdır, Çizelge 4.3'te detaylandırılmıştır, Ek A3'te örnek dosya içeriği verilmiştir.

Çizelge 4.3 : Raporlama bilgileri.

Alan	Açıklama	Örnek
ID	Raporun tekil id numarası	1
taskControlPeriodTime	Periyot zamanı	2014-01-03T10:25:15.034 +02:00
maxCoalitionSize	Maks. koalisyon büyüklüğü	2
coalitionFormationDuration	Koalisyon kurulum süresi (ms)	134
coalitionDetails	Koalisyon detayı	-
coalitionManager	Koalisyon detayındaki koalisyon yöneticisi bilgileri	coalitionManagerAgent1, managerPlatform, http://merve-laptop:6000/acc, logs/coalitionManagerAgent1.log
taskDetails	Koalisyon detayındaki görev bilgisi	task1, FINISHED, start: 2014-01-03T10:25:42.537+02:00, stop: 2014-01-03T10:26:06.451+02:00
employeeAgentList	Koalisyon detayındaki işçi etmen listesi	-
agentInf	İşçi etmen listesindeki etmenlerin detay bilgisi	A1, successfull: true, start: 2014-01-03T10:25:42.537+02:00, stop: 2014-01-03T10:26:06.353+02:00

Kullanıcılar (users.xml):

Çizelge 4.4'te verildiği üzere kullanıcı tanımlarının tutulduğu veri dosyasıdır, kullanıcı detaylarını barındırır. Ek A4'te örnek dosya içeriği verilmiştir.

Çizelge 4.4 : Kullanıcı bilgileri.

Alan	Açıklama	Örnek
ID	Kullanıcının tekil id numarası	1
userName	Kullanıcı adı	admin
password	Kullanıcı şifresi	******

Yukarıda verilen xml dosyalarının her biri için xsd (XML Schema Definition) şemaları oluşturulmuştur. Xbean kütüphanesi kullanılarak tüm xsd tanımlarının Java nesnesi karşılıkları oluşturulmuştur [26]. Böylece veri tabanından xml yapısında alınan veriler sisteme Java nesneleri olarak geçirilip rahat bir şekilde kullanım sağlanmıştır.

Ortak tanım şeması (commonDefinitions.xsd): Birden fazla şema dosyasında kullanılan ortak tanımlar bu dosyada birleştirilerek tek bir yerden kullanım sağlanmıştır, Ek B1'de dosya içeriği verilmiştir.

Kullanıcı tanım şeması (users.xsd): users.xml dosyasının şema tanım dosyasıdır, Ek B2'de dosya içeriği verilmiştir.

İşçi etmen tanım şeması (employeeAgents.xsd): employeeAgents.xml dosyasının şema tanım dosyasıdır, Ek B3'te dosya içeriği verilmiştir.

Görev tanım şeması (tasks.xsd): tasks.xml dosyasının şema tanım dosyasıdır, Ek B4'te dosya içeriği verilmiştir.

Raporlama tanım şeması (report.xsd): report.xml dosyasının şema tanım dosyasıdır, Ek B5'te dosya içeriği verilmiştir.

Veri tabanı sorguları için modül dosyaları oluşturulmuştur, bu modüller xquery fonksiyonlarından oluşmaktadır. Modüller genel olarak ilişkili oldukları xml dosyalarıyla ilgili sorguları bulundurmaktadır.

agentsModule.xquery: employeeAgents.xml dosyası ile ilgili sorguların bulunduğu modül dosyasıdır. İşçi etmenler listesinde arama yapma, tüm etmen listesini getirme, yeni etmen kaydı ve etmen güncelleme gibi fonksiyonları bulundurur. Ek C1'de dosya içeriği verilmiştir.

tasksModule.xquery: tasks.xml dosyası ile ilgili sorguların bulunduğu modül dosyasıdır. Görev listesinde arama yapma, tüm listeyi getirme, görev tanımı güncelleme gibi fonksiyonları bulundurur. Ek C2'de dosya içeriği verilmiştir.

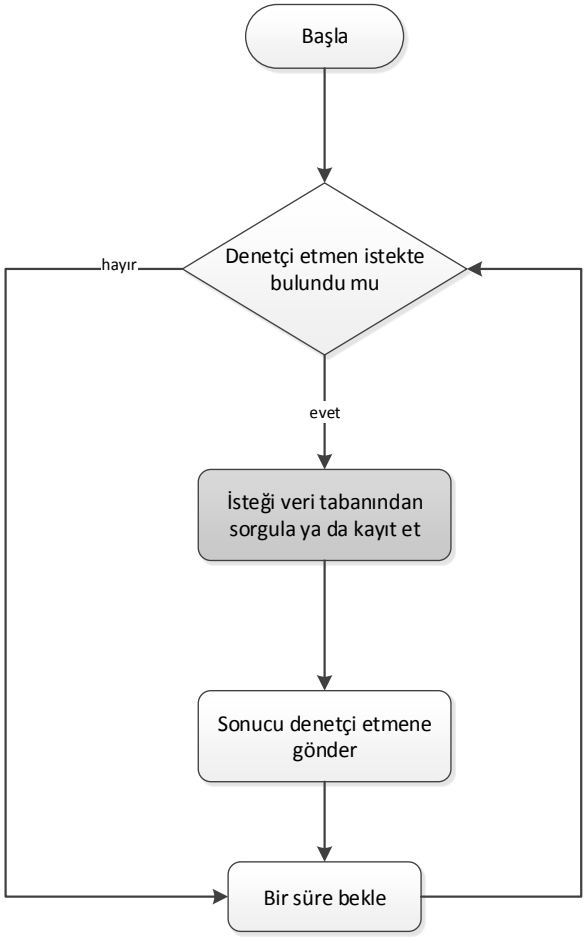
reportModule.xquery: report.xml dosyası ile ilgili sorguların bulunduğu modül dosyasıdır. Çalışma raporunu kaydetme fonksiyonlarını bulundurur. Ek C3'te dosya içeriği verilmiştir.

userModule.xquery: users.xml dosyası ile ilgili sorguların bulunduğu modül dosyasıdır. Kullanıcı girişi için gerekli bir fonksiyon bulundurur. Ek C4'te dosya içeriği verilmiştir.

4.5.2 Veri tabanı etmeni

Veri tabanı etmeni sistemde veri tabanı üzerinde işlem yapma yeteneğine sahip olan tek etmendir, akış şeması Şekil 4.6'da verilmiştir.

Bu etmenin görevi veri tabanı ile ilgili istekleri kabul etmek ve işlemleri gerçekleyerek sonuçları isteklilere iletmektir.

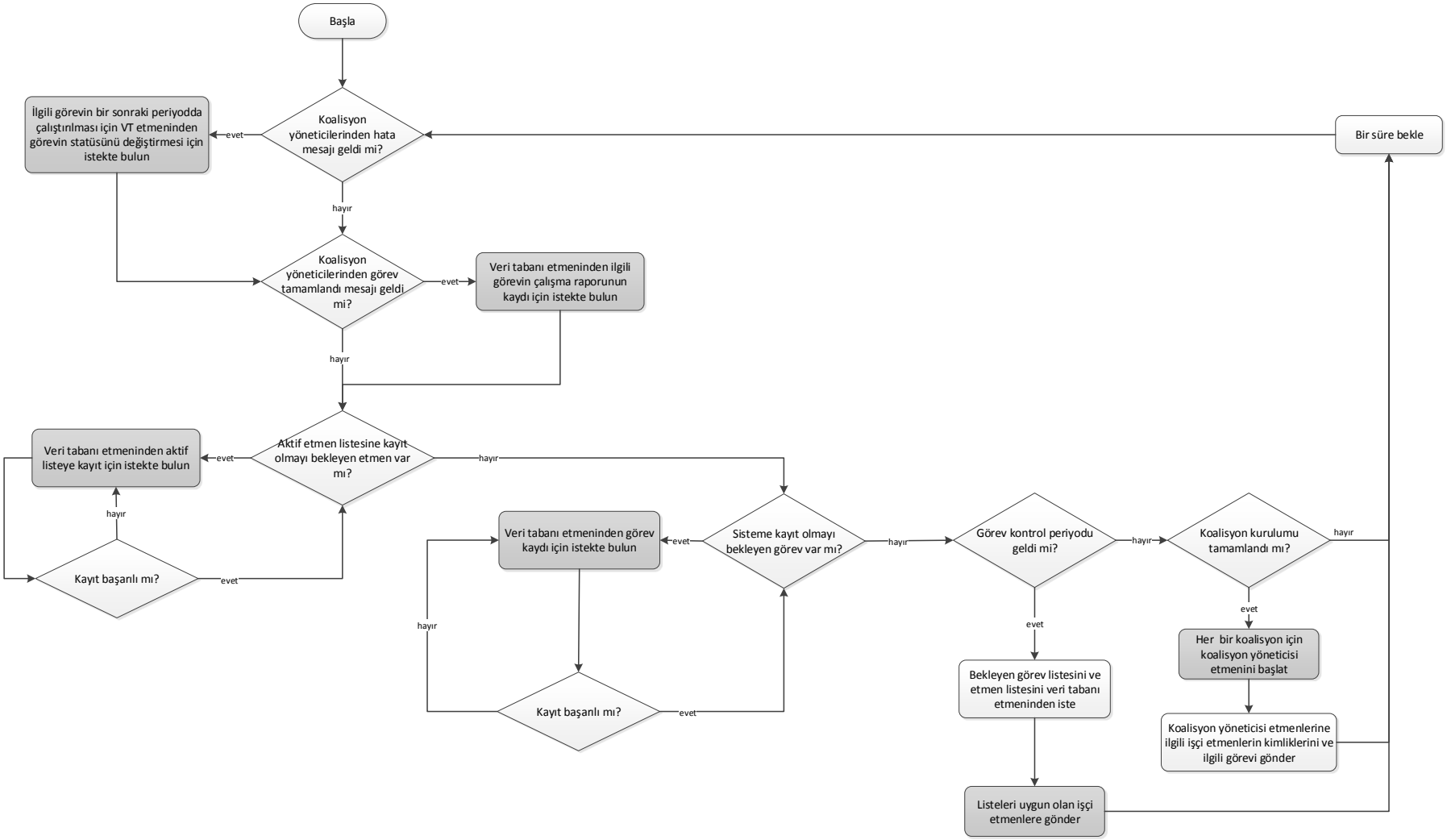


Şekil 4.6 : Veri tabanı etmeni akış şeması.

Veri tabanı sorgusuna ihtiyacı olan etmen, bu etmenle iletişime geçerek isteğini bildirir ve daha sonra sonucu alır. Örneğin denetçi etmen sistem ilk açıldığında görev listesini ve aktif işçi etmen listesini veri tabanı etmeninden ister, veri tabanı etmeni de listeleri sorgulayarak denetçi etmene sonuçları gönderir. Veri tabanı etmeniyle iletişimi olan tek etmen denetçi etmen olduğu için diğer etmenler veri tabanı sorgusuna ihtiyaç duyduklarında bu istediği denetçi etmen üzerinden iletirler.

4.5.3 Denetçi etmen

Denetçi etmen sistemin genel koordinasyonundan sorumludur. Etmenlerin durumlarını ve koalisyon kurulumu aşamalarını izler, görev kontrol periyodunu takip eder. Belirtilen görev kontrol periyodu süresi doldukça sistemdeki koalisyon kurulum işleyişini yeniden başlatır böylece sisteme yeni görevler eklendikçe sistemi kapatıp açmaya gerek kalmaz. Denetçi etmenin akış şeması ayrıntılı olarak Şekil 4.7'de verilmiştir.



Şekil 4.7 : Denetçi etmenin akış şeması.

Koalisyon kurulumunu takip eder, koalisyon kurulumu tamamlandıktan sonra denetçi etmen her koalisyon için birer koalisyon yöneticisi etmeni başlatır ve bu koalisyon yöneticilerine kendi koalisyonlarındaki işçi etmenlerin ve görevin ayrıntılarını iletir, ardından işçi etmenlerle kendi arasındaki iletişimini keser. Bundan sonra koalisyon yöneticilerinden gelecek mesajları bekler. Tüm koalisyon yöneticilerinden ilgili oldukları koalisyonlarla ilgili mesajları aldığında o çalışma detayından bir çalışma raporu hazırlar ve bu raporu veri tabanına kaydetmek üzere veri tabanı etmenine iletir.

Denetçi etmen bunların dışında hata kontrolü yapar, eğer bir koalisyon yöneticisinden hata mesajı gelirse ilgili görevin durumunu başarısız yaparak veri tabanı etmeni aracılığıyla veri tabanında günceller, böylece o görev bir sonraki periyodda çalıştırılır. Bu da sistemin hataya dayanıklı olarak çalışmasını sağlar.

Ayrıca, denetçi etmen diğer etmenlerin ihtiyaç duyduğu verileri sağlar; örneğin koalisyon yöneticisi etmenlerine koalisyondaki işçi etmenlerin kimliklerini gönderir.

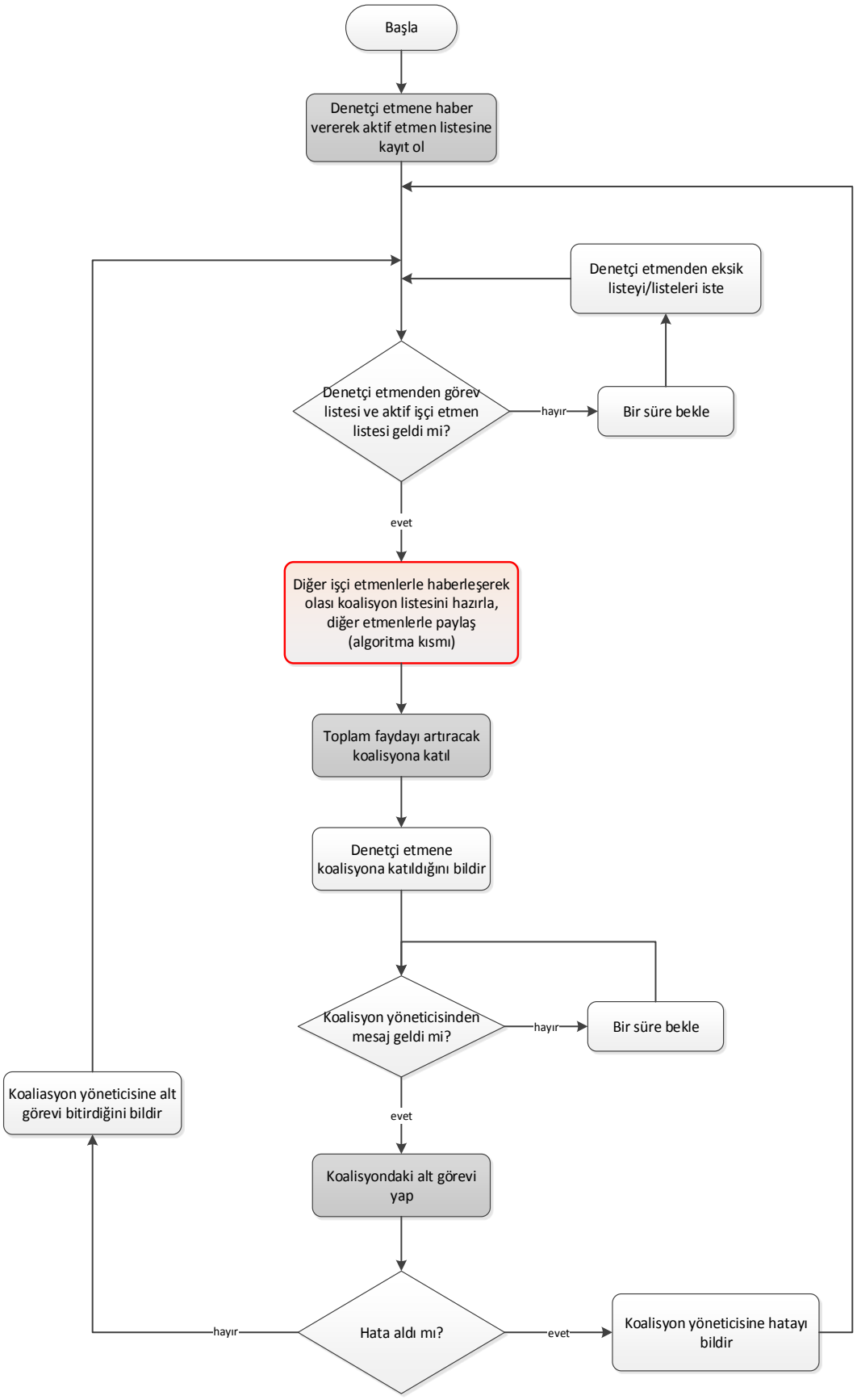
4.5.4 İşçi etmen

Yerine getirilmesi istenen görevleri paylaşıp yerine getirecek olan etmen türüdür. İşçi etmenler açıldıklarında aktif olduklarını denetçi etmene bildirerek aktif etmen listesine kayıt olurlar. Denetçi etmenden görev listesi ve aktif işçi etmen listesi gelir gelmez koalisyon kurulumuna başlarlar.

İşçi etmenler bölüm 4.4'te verilen algoritmayı çalıştırarak hangi koalisyona katılacaklarını hesaplarlar ve denetçi etmene bildirirler. Katılacakları koalisyonun koalisyon yöneticisinden mesaj beklerler, mesaj geldiği anda koalisyona katılıp kendilerine düşen alt görevi yerine getirirler ve koalisyon yöneticisi etmenine alt görevi tamamladıklarını bildirirler. Görev çalıştırma esnasında hata olursa koalisyon yöneticisi etmenine hatayı bildirirler. İşçi etmenin akış şeması Şekil 4.8'de verilmiştir.

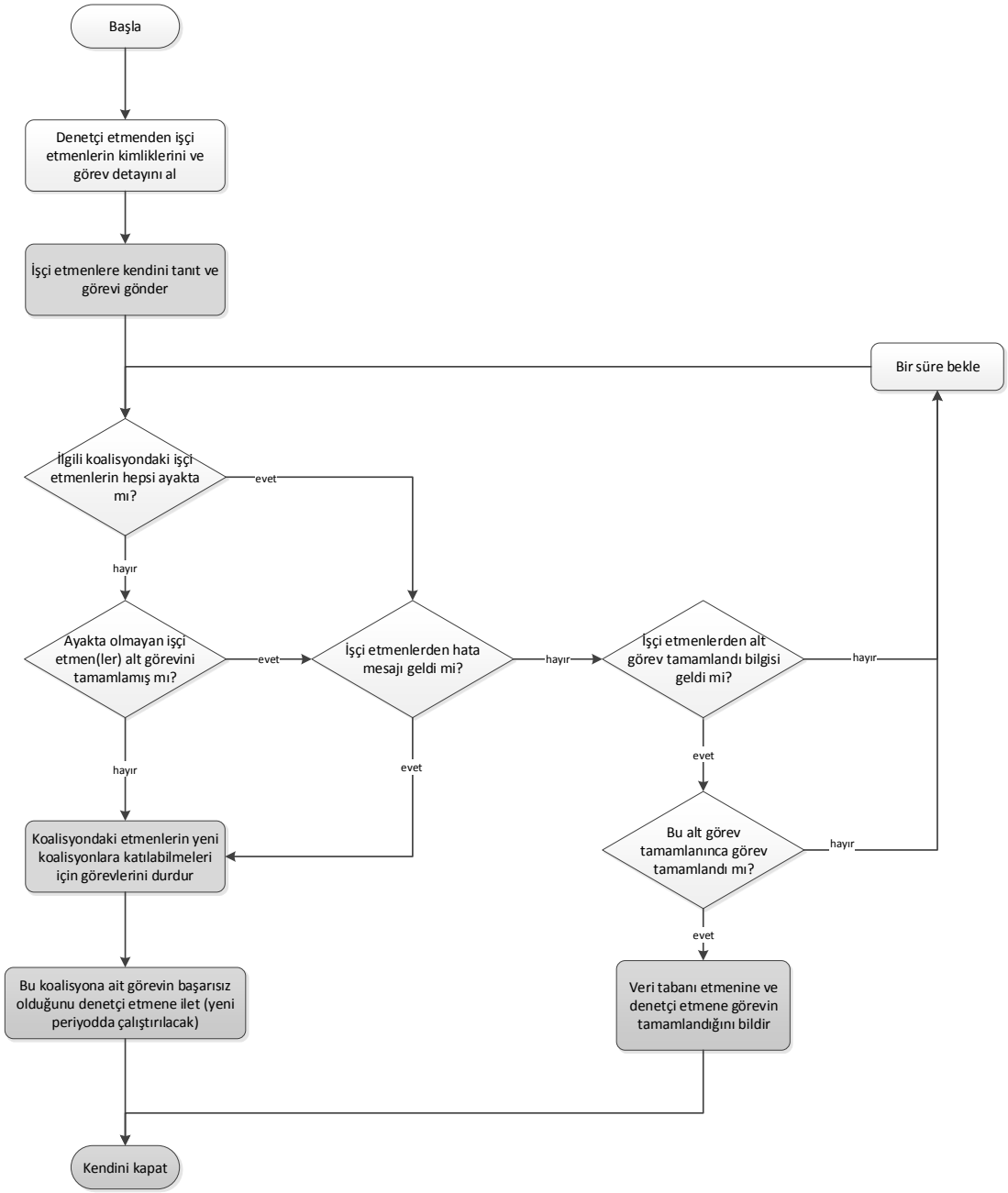
4.5.5 Koalisyon yöneticisi etmeni

J. Ferber, O. Gutknecht ve F. Michel'in çalışmasında [6] etmen gruplarına grup yöneticisi ataması yapılmış ve o gruptan sorumlu olmuştur. Yapılan çalışmada benzer şekilde her bir koalisyon için birer koalisyon yöneticisi ataması yapılmıştır. Bu atamayı koalisyonlar belirlendikten sonra denetçi etmen yapar.



Şekil 4.8 : İşçi etmen akış şeması.

Denetçi etmen, koalisyondaki işçi etmenlerin kimlikleri ile çalıştırılacak görevi koalisyon yöneticisi etmenine bildirir ve işçi etmenlerle koalisyon yöneticisi arasından çekilir. Koalisyon yöneticisi etmeni öncelikle kendi koalisyonundaki işçi etmenlere kendini tanıtır, ardından işçi etmenlerin alt görevleri yapıp yapmadıklarını takip eder, görev tamamlandığında denetçi etmene bildirir ve kendi etmenini sonlandırır. Akış şeması Şekil 4.9'da ayrıntılı olarak verilmiştir.



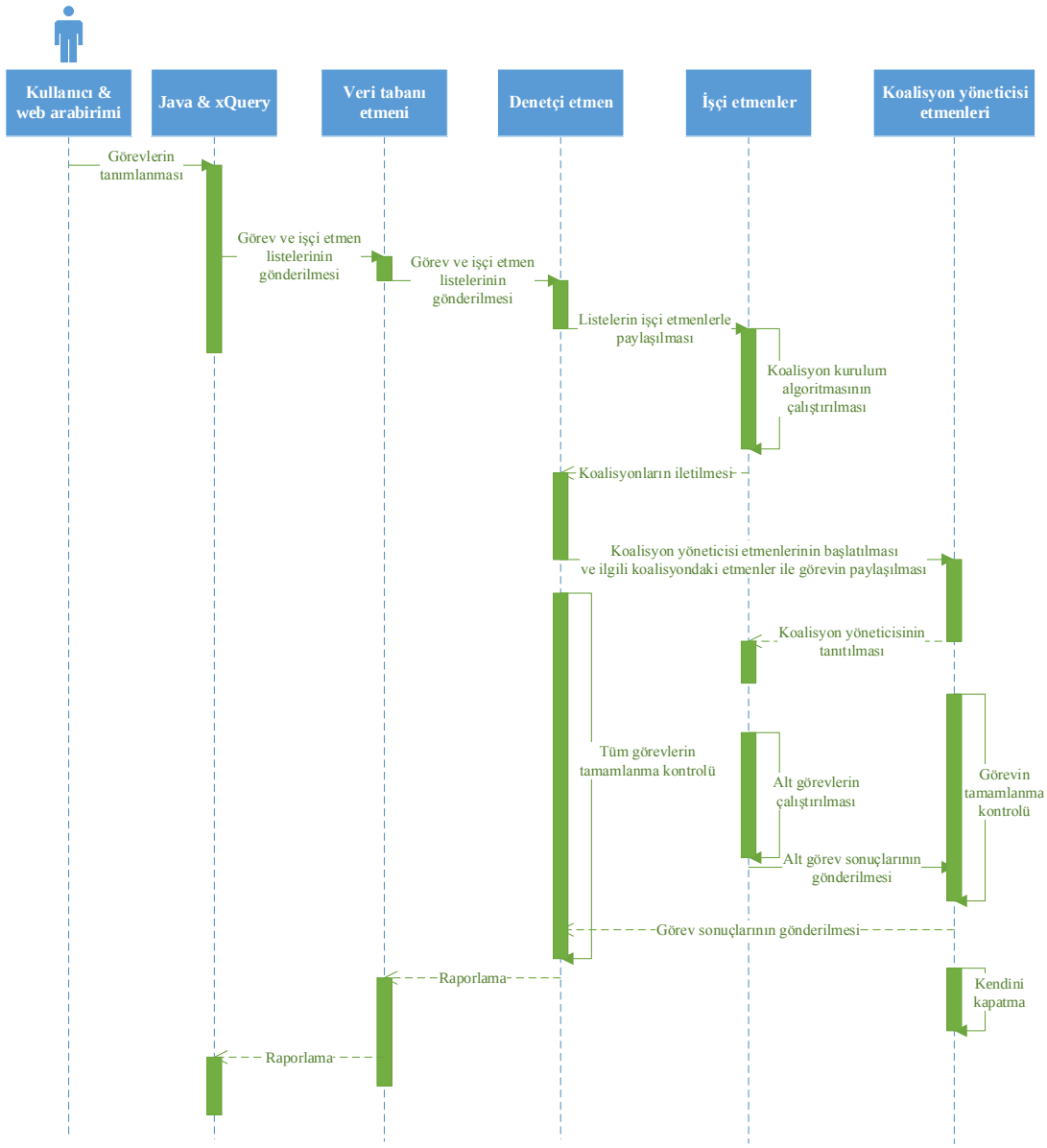
Şekil 4.9 : Koalisyon yöneticisi etmeni akış şeması.

Koalisyon yöneticisi etmeni hata kontrolü yaparak hata alma durumunda görevin başarısız olduğunu denetçi etmene bildirir ve alt görevleri durdurur. Ayrıca

koalisyondaki işçi etmenlerin ayakta olup olmadıklarını belirli aralıklarla kontrol eder. Koalisyondaki bir işçi etmen ayakta değilse yine hatayı denetçi etmene bildirir ve alt görevleri durdurur. Her iki durumda da işlemlerini tamamladıktan sonra kendi etmenini sonlandırır.

4.5.6 Olayların zinciri

Yapılan çalışmanın ana bileşenleri arasındaki mantıksal etkileşimler Şekil 4.10'daki Akış (Sequence) Diyagramı'nda ayrıntılı olarak gösterilmiştir. Diyagramdaki "Kullanıcı & web arabirimi" aktörü sisteme görev kaydetmek için eklenmiştir (bknz. bölüm 4.7 Kullanıcı Arabirimi).



Şekil 4.10 : Akış diyagramı.

4.6 Kullanıcı Arabirimi

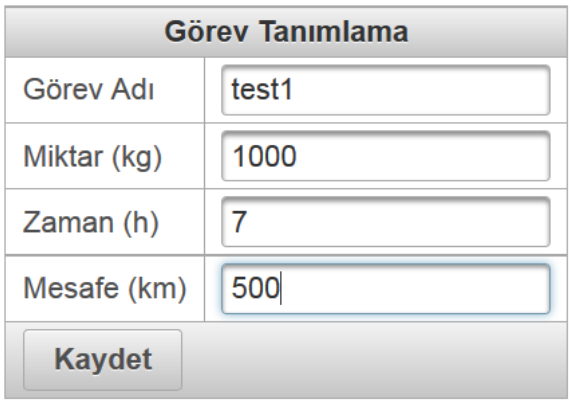
Görev tanımı yapabilmek için kullanıcı arabirimi ihtiyacı doğmuştur. Kullanıcı arabirimi Java EE [27] platformu üzerinde bir JSF [28] bileşen paketi olan Primefaces 4.0 [29] kullanılarak geliştirilmiştir. Bu web uygulaması bir uygulama sunucusu olan Tomcat 7.0 [30] üzerine deploy edilerek çalıştırılmaktadır.

Web uygulamasına "http://merve-laptop:8080/WebInterface/" adresi kullanılarak herhangi bir makine üzerinden erişilebilir. Uygulama açıldığında öncelikle kullanıcı giriş ekranı gelmektedir (Şekil 4.11).



Şekil 4.11 : Kullanıcı giriş ekranı.

Kullanıcı girişi başarılı bir şekilde yapıldığında görev tanımlama ekranı gelir. Taşıma görevi için ilgili alanlar doldurulur, tanımlanan görevler veri tabanındaki tasks.xml'e kaydedilir. Yeni tanımlanan görevler sistemdeki bir sonraki görev kontrol periyodunda çalıştırılmak üzere bekletilir.



Şekil 4.12 : Görev tanımlama ekranı.

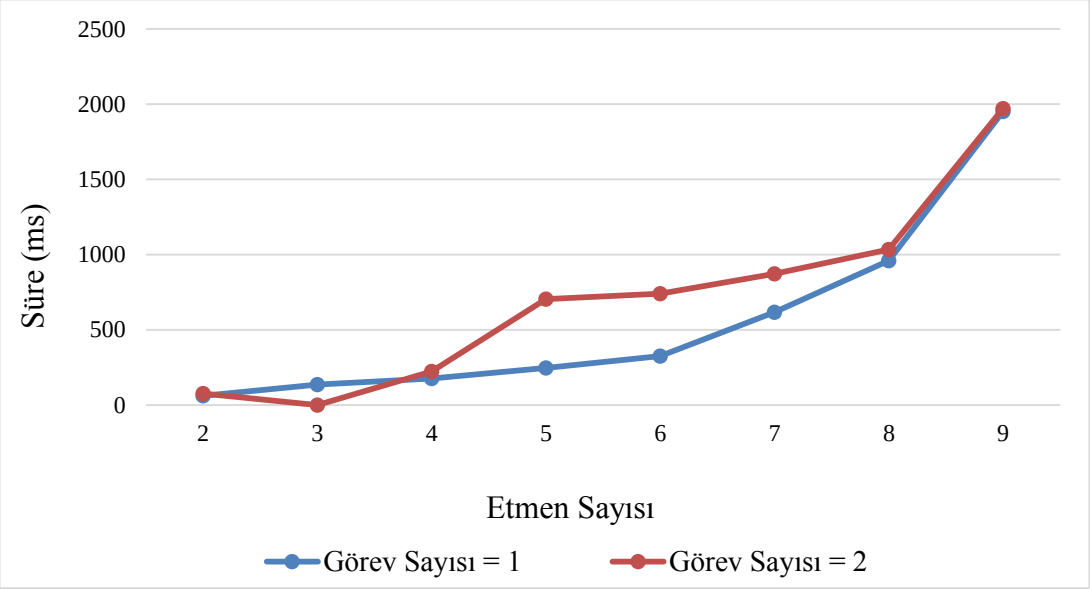
5. BAŞARIM DEĞERLENDİRMESİ

Koalisyon kurulumu ile görev paylaşımı için hazırlanan yürütme ortamında çeşitli testler yapılmıştır. Testler 1.73 GHz Intel Core i7 işlemcili, 8 GB belleğe sahip, 64-bit Windows 7 Professional işletim sistemi kurulu kişisel bilgisayarda yapılmıştır. JADE 4.2.0 [14] sürümü kullanılarak geliştirilen etmenler üzerinde testler gerçekleştirilmiştir.

Testlerde işçi etmen sayısı, görev sayısı ve izin verilen en büyük koalisyon boyutu olan *k* sabiti değiştirilerek sonuçlar incelenmiştir. Deneyler yaklaşık 3000 defa tekrarlanmış ve elde edilen sürelerin ortalamaları sunulmuştur. Süre hesaplamalarında işçi etmenlerin başlatılıp akışlarda gösterildiği üzere kayıt olması ve gerekli bilgileri almaları gibi kısımlar ele alınmamıştır, süreler koalisyon kurulumu aşamalarının ne kadar sürdüğünü göstermektedir.

5.1 İşçi Etmen Sayısının Koalisyon Kurulum Süresine Etkisi

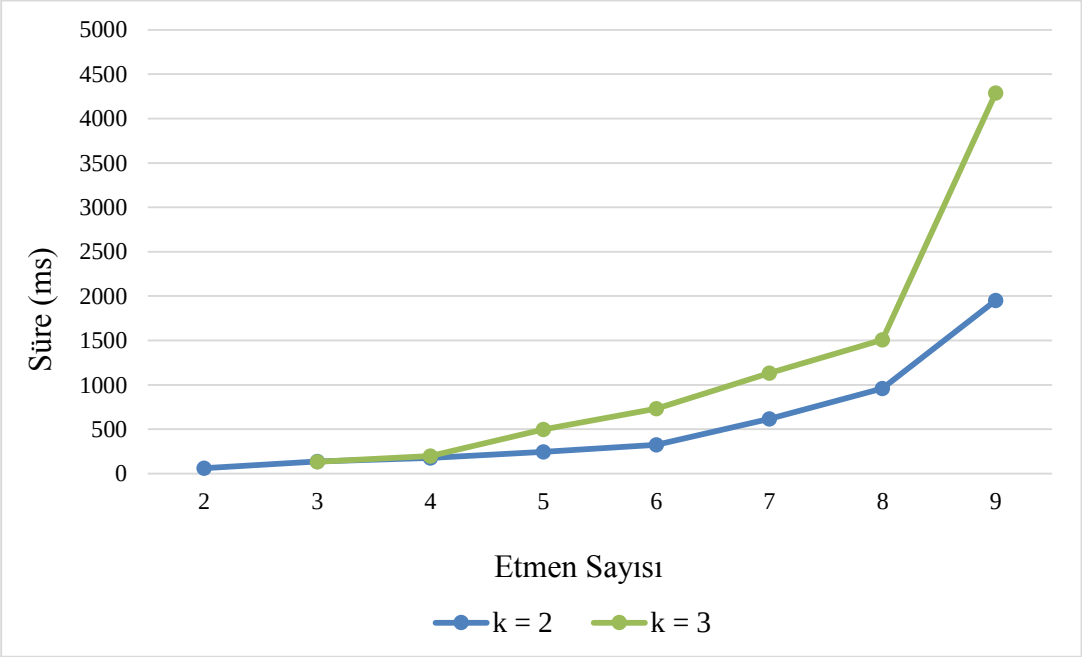
Şekil 5.1'de işçi etmen sayısındaki artışın koalisyon kurulum süresindeki etkisi gösterilmektedir.



Şekil 5.1 : Farklı görev sayılarında işçi etmen sayısının koalisyon kurulum süresine etkisi.

Bu testlerde *k* sabiti 2 olarak belirlenmiştir, 1 ve 2 tane görevin bulunduğu iki farklı sistemde işçi etmen sayısı artırılarak testler gerçekleştirilmiştir. Etmen sayısındaki artış sürede hızlı bir yükselişle sonuçlanmıştır, buna rağmen 9 işçi etmenin bulunduğu testlerde bile 2 saniyeden kısa sürede koalisyon kurulumu tamamlanmıştır. 2 tane görevin bulunduğu sistemde koalisyon kurulum sürelerinin 1 görevli sisteme göre biraz daha büyük olduğu gözlenmiştir.

Şekil 5.2'de yine işçi etmen sayısındaki artışın koalisyon kurulum süresindeki etkisi gösterilmektedir. Bu kısımdaki testlerde hem *k* sabiti değiştirilerek hem de etmen sayısı artırılarak 1 görevli sistemde testler gerçekleştirilmiştir. *k* sabitinin 3 olduğu testlerde 2 olduğu testlere göre daha büyük süreler elde edilmiştir. Bu kısımdaki testlerde 5 saniyeden kısa sürede koalisyon kurulumu tamamlanmıştır.



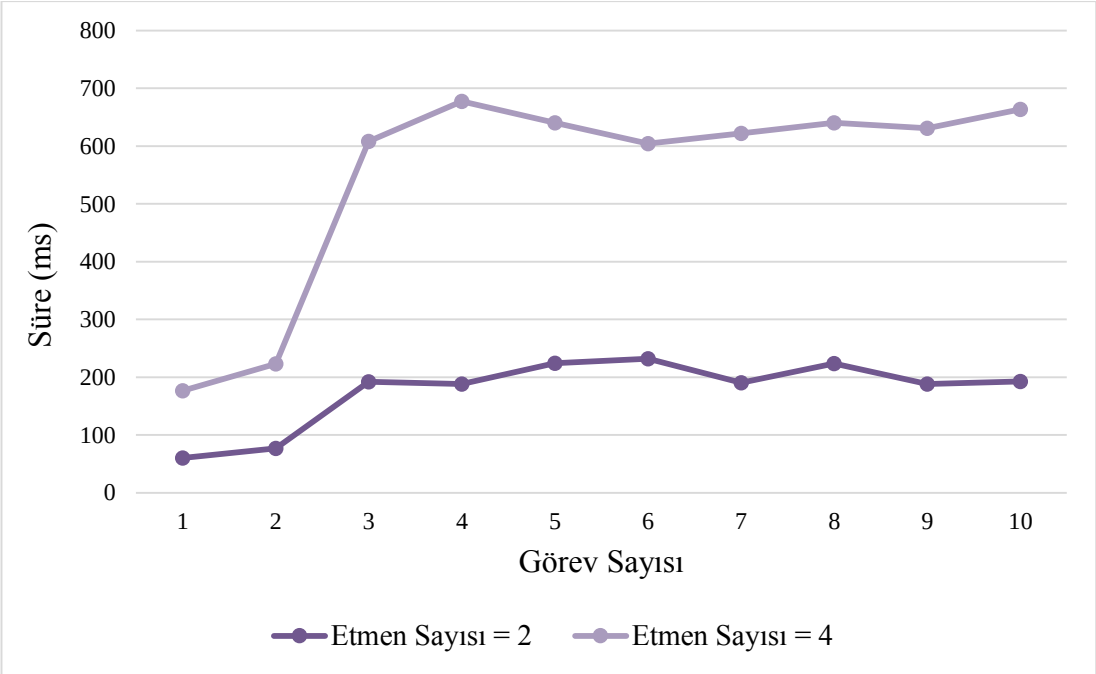
Şekil 5.2 : Farklı *k* değerlerinde işçi etmen sayısının koalisyon kurulum süresine etkisi.

İşçi etmen sayısındaki artışın süreyi bu şekilde artırması, iteratif olarak yapılan hesaplamaların ve işçi etmenler arasındaki mesajlaşmaların artmasından kaynaklanmaktadır.

5.2 Görev Sayısının Koalisyon Kurulum Süresine Etkisi

Şekil 5.3'te ise görev sayısındaki artışın koalisyon kurulum süresindeki etkisi gösterilmektedir. 2 ve 4 tane işçi etmenin bulunduğu sistemlerde görev sayısı

artırılarak testler gerçekleştirilmiştir, *k* sabiti 2 olarak belirlenmiştir. Testlerdeki koalisyon kurulum sürelerine bakıldığında görev sayısı 3'e çıkarıldıktan sonra hem 2 etmenli hem de 4 etmenli testlerde belirgin bir değişiklik gözlenmemiştir; 2 etmenli testlere ortalama 200 milisaniyede, 4 etmenli testlerde ortalama 650 milisaniyede koalisyonlar kurulmuştur.



Şekil 5.3 : Görev sayısının koalisyon kurulum süresine etkisi.

İşçi etmen sayısı, görev sayısı ve *k* sabiti değiştirilerek yapılan testler sonucunda başarılı bir şekilde koalisyon kurulumu gerçekleşmiştir. Her bir testin çalıştırma raporları veri tabanında "xmls/reports" dizini altında ayrı xml dosyalarında tutulmaktadır.

6. SONUÇ VE ÖNERİLER

Bu çalışmada iş yapma yeteneği olan otonom etmenlerin tanımlanan görevleri yerine getirmek üzere koalisyon kurmaları sağlanmıştır. Koalisyon kurma aşamasında gerçeklenen bir algoritma ile etkili bir şekilde tüm görevleri ya da tüm etmenleri kapsayacak şekilde koalisyon kurulumu gerçekleşmiştir. Yapılan çalışmada tüm bu işlemlerin yapılabilmesi için bir yürütme ortamı hazırlanmıştır ve testler bu ortamda gerçekleştirilmiştir.

Bu çalışma ile etmenlerin koalisyon kurarak görev paylaşımı yapmaları sonucu ele alınan problemin çözümünde hızlı ve etkili bir şekilde görev paylaşımı yapılarak görevlerin yerine getirilmesi sağlandığı görülmüştür. Testlerde tüm etmenlerin aynı makinede çalıştırılmasına rağmen hızlı sonuç verdiği görülmüştür. Yine de performansı biraz daha artırmaya yönelik, etmenlerin dağıtık sistemde çalıştırılması sağlanabilir.

Bu çalışmanın devamı olarak kesişen koalisyonların kurulması sağlanabilir. Kesişen koalisyonlarda bir etmenin aynı anda birden fazla koalisyona katılması söz konusudur. Ayrıca görevlerin öncelik sırasına göre çalıştırılması sağlanabilir.

KAYNAKLAR

[1] **Shehory, O. ve Kraus, S.** (1998). Methods for task allocation via agent coalition formation, Artificial Intelligence 101 (1–2) 165–200.

[2] **Sandholm, T., Larson, K., Andersson, M., Shehory, O. ve Tohm, F.** (1998). Worst-case-optimal anytime coalition structure generation. In: Proceedings of AAAI 1998, pp. 46–53.

[3] **Kraus, S., Shehory, O. ve Taase, G.** (2003). Coalition formation with uncertain heterogeneous information, Proceedings of the second international joint conference on Autonomous agents and multiagent systems, AAMAS ’03, July 14-18, Melbourne, Australia.

[4] **Kraus, S., Shehory, O. ve Taase, G.** (2004). The Advantages of Compromising in Coalition Formation with Incomplete Information, Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, p.588-595, July 19-23, New York.

[5] **Blankenburg, B., Klusch, M. ve Shehory, O.** (2003). Fuzzy kernel-stable coalitions between rational agents, Proceedings of the second international joint conference on Autonomous agents and multiagent systems, AAMAS ’03, July 14-18, Melbourne, Australia.

[6] **Ferber, J., Gutknecht, O. ve Michel, F.** (2003). From Agents to Organizations: an Organizational View of Multi-Agent Systems. In: Giorgini, P., Müller, J.P., Odell, J.J. (eds.) AOSE 2003. LNCS, vol. 2935, pp. 214–230.

[7] **Alfonso, B., Botti, V., Garrido, A., ve Giret, A.** (2012). A MAS-based Infrastructure for Negotiation and its Application to a Water-Right Market. ITMAS 2012: Infrastructures and Tools for Multiagent Systems, Valencia, Spain.

[8] **Pinto, T., Morais, H., Oliveira, P., Vale, Z., Praça, I. ve Ramos, C.** (2011). A new approach for multi-agent coalition formation and management in the scope of electricity markets, Energy, Volume 36, Issue 8, August 2011, Pages 5004–5015.

[9] **Ray, D. ve Vohra, R.** (2013). Coalition Formation, Preliminary draft, prepared as a chapter for Handbook of Game Theory, Volume 4, ed. by Peyton Young and Shmuel Zamir, North-Holland The MIT Press, Cambridge, Massachusetts.

[10] **Weiss, G. ve Wooldridge, M.** (1999). Multiagent Systems A Modern Approach to Distributed Artificial Intelligence, The MIT Press, Cambridge, Massachusetts.

[11] **Luck, M., McBurney, P. ve Preist, C.** (2003). Agent Technology: Enabling Next Generation Computing (A Roadmap for Agent Based Computing). AgentLink. ISBN 0854 327886.

[12] **Wooldridge, M. ve Jennings, N.R.** (1995). Intelligent Agents: Theory and Practice. The Knowledge Engineering Review.Vol.10, no. 2, pp. 115-152.

[13] **Wooldridge, M.** (1997). Agent-Based Software Engineering. IEE Proceedings on Software Engineering 26-37.

[14] **JADE Ana Sayfa** <http://jade.tilab.com>, alındığı tarih: 08.12.2012.

[15] **Bellifemine, F., Caire, G., Trucco, T. ve Rimassa, G.** (2010). JADE Programmer's Guide, Telecom Italia Lab.

[16] **Bellifemine, F., Caire, G., Poggi, A. ve Rimassa, G.** (2003). JADE A White Paper, Telecom Italia EXP magazine, Vol. 3, No.3, 6-19.

[17] **Caire, G.** (2009). JADE Programming for Beginners, Telecom Italia Lab.

[18] **SC0001L.** (2002). FIPA Abstract Architecture Specification, Geneva, Switzerland.

[19] **SC00061G.** (2002). FIPA ACL Message Structure Specification, Geneva, Switzerland.

[20] **SC00037J.** (2002). FIPA Communicative Act Library Specification Geneva, Switzerland.

[21] **Güncel Türkçe Sözlük** <http://www.tdk.gov.tr/index.php?option=com_gts&view=gts>, alındığı tarih: 07.12.2013.

[22] **Extensible Markup Language** < http://www.w3.org/standards/xml/core> , alındığı tarih: 09.12.2013.

[23] **eXist-db Ana Sayfa** <http://exist-db.org/exist/apps/homepage/index.html> , alındığı tarih: 06.01.2013.

[24] **XML Schema** <http://www.w3.org/standards/xml/schema>, alındığı tarih: 09.12.2013.

[25] **XML Query** < http://www.w3.org/standards/xml/query>, alındığı tarih: 09.12.2013.

[26] **Martin, B.E.** (2000). Distributed Xbean Applications, Distributed Objects and Applications. Proceedings. DOA '00. International Symposium on, p. 185 – 192.

[27] **Java EE** <https://java.net/projects/javaee-spec/pages/Home>, alındığı tarih: 25.12.2013.

[28] **JSF** <https://javaserverfaces-spec-public.java.net/>, alındığı tarih: 25.12.2013.

[29] **Primefaces Ana Sayfa** <http://www.primefaces.org/index.html>, alındığı tarih: 25.12.2013.

[30] **Tomcat Ana Sayfa** <http://tomcat.apache.org/>, alındığı tarih: 25.12.2013.

EKLER

EK A: Tanımlanmış Veri Dosyaları
- **EK A1:** users.xml
- **EK A2:** employeeAgents.xml
- **EK A3:** tasks.xml
- **EK A4:** report.xml

EK B: XSD Dosyaları
- **EK B1:** users.xsd
- **EK B2:** commonDefinitions.xsd
- **EK B3:** employeeAgents.xsd
- **EK B4:** tasks.xsd
- **EK B5:** report.xsd

EK C: Modül Dosyaları
- **EK C1:** agentsModule.xquery
- **EK C2:** taskModule.xquery
- **EK C3:** reportModule.xquery
- **EK C4:** userModule.xquery

EK A

EK A1: employeeAgents.xml

```
<eagnt:employeeAgentList
xmlns:com="http://www.agentcoalition.com/XML_common_types"
xmlns:eagnt="http://www.agentcoalition.com/XML_employeeagent_types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.agentcoalition.com/XML_employeeagent_types
../xsds/employeeAgents.xsd">
  <eagnt:employeeAgent ID="1">
    <com:localName>A1</com:localName>
    <com:platformName>employeePlatform1</com:platformName>
    <com:address>http://merve-laptop:5600/acc</com:address>
    <com:agentState>ACTIVE</com:agentState>
    <com:capabilityVector>
      <com:capability>
        <com:capabilityType>AMOUNT</com:capabilityType>
        <com:capabilityValue>4000</com:capabilityValue>
        <com:capabilityUnit>kg</com:capabilityUnit>
      </com:capability>
      <com:capability>
        <com:capabilityType>TIME</com:capabilityType>
        <com:capabilityValue>8</com:capabilityValue>
        <com:capabilityUnit>h</com:capabilityUnit>
      </com:capability>
      <com:capability>
        <com:capabilityType>DETAIL</com:capabilityType>
        <com:capabilityValue>1000</com:capabilityValue>
        <com:capabilityUnit>km</com:capabilityUnit>
      </com:capability>
      <com:capability>
        <com:capabilityType>COST</com:capabilityType>
        <com:capabilityValue>3500</com:capabilityValue>
        <com:capabilityUnit>TL</com:capabilityUnit>
      </com:capability>
    </com:capabilityVector>
  </eagnt:employeeAgent>
  <eagnt:employeeAgent ID="2">
    <com:localName>A2</com:localName>
    <com:platformName>employeePlatform2</com:platformName>
    <com:address>http://merve-laptop:5700/acc</com:address>
    <com:agentState>ACTIVE</com:agentState>
    <com:capabilityVector>
      <com:capability>
        <com:capabilityType>AMOUNT</com:capabilityType>
        <com:capabilityValue>3000</com:capabilityValue>
        <com:capabilityUnit>kg</com:capabilityUnit>
      </com:capability>
      <com:capability>
```

```
            <com:capabilityType>TIME</com:capabilityType>
            <com:capabilityValue>6</com:capabilityValue>
            <com:capabilityUnit>h</com:capabilityUnit>
         </com:capability>
         <com:capability>
            <com:capabilityType>DETAIL</com:capabilityType>
            <com:capabilityValue>1000</com:capabilityValue>
            <com:capabilityUnit>km</com:capabilityUnit>
         </com:capability>
         <com:capability>
            <com:capabilityType>COST</com:capabilityType>
            <com:capabilityValue>4000</com:capabilityValue>
            <com:capabilityUnit>TL</com:capabilityUnit>
         </com:capability>
      </com:capabilityVector>
      <eagnt:available>true</eagnt:available>
   </eagnt:employeeAgent>
   <eagnt:employeeAgent ID="3">
      <com:localName>A3</com:localName>
      <com:platformName>employeePlatform3</com:platformName>
      <com:address>http://merve-laptop:5800/acc</com:address>
      <com:agentState>ACTIVE</com:agentState>
      <com:capabilityVector>
         <com:capability>
            <com:capabilityType>AMOUNT</com:capabilityType>
            <com:capabilityValue>2000</com:capabilityValue>
            <com:capabilityUnit>kg</com:capabilityUnit>
         </com:capability>
         <com:capability>
            <com:capabilityType>TIME</com:capabilityType>
            <com:capabilityValue>8</com:capabilityValue>
            <com:capabilityUnit>h</com:capabilityUnit>
         </com:capability>
         <com:capability>
            <com:capabilityType>DETAIL</com:capabilityType>
            <com:capabilityValue>1000</com:capabilityValue>
            <com:capabilityUnit>km</com:capabilityUnit>
         </com:capability>
         <com:capability>
            <com:capabilityType>COST</com:capabilityType>
            <com:capabilityValue>2000</com:capabilityValue>
            <com:capabilityUnit>TL</com:capabilityUnit>
         </com:capability>
      </com:capabilityVector>
   </eagnt:employeeAgent>
   <eagnt:employeeAgent ID="4">
      <com:localName>A4</com:localName>
      <com:platformName>employeePlatform4</com:platformName>
      <com:address>http://merve-laptop:5900/acc</com:address>
      <com:agentState>ACTIVE</com:agentState>
```

```
      <com:capabilityVector>
        <com:capability>
          <com:capabilityType>AMOUNT</com:capabilityType>
          <com:capabilityValue>3000</com:capabilityValue>
          <com:capabilityUnit>kg</com:capabilityUnit>
        </com:capability>
        <com:capability>
          <com:capabilityType>TIME</com:capabilityType>
          <com:capabilityValue>7</com:capabilityValue>
          <com:capabilityUnit>h</com:capabilityUnit>
        </com:capability>
        <com:capability>
          <com:capabilityType>DETAIL</com:capabilityType>
          <com:capabilityValue>1000</com:capabilityValue>
          <com:capabilityUnit>km</com:capabilityUnit>
        </com:capability>
        <com:capability>
          <com:capabilityType>COST</com:capabilityType>
          <com:capabilityValue>4500</com:capabilityValue>
          <com:capabilityUnit>TL</com:capabilityUnit>
        </com:capability>
      </com:capabilityVector>
    </eagnt:employeeAgent>
</eagnt:employeeAgentList>
```

EK A2: tasks.xml

```
<tsk:taskList xmlns:com="http://www.agentcoalition.com/XML_common_types"
xmlns:tsk="http://www.agentcoalition.com/XML_task_types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.agentcoalition.com/XML_task_types
../xsds/tasks.xsd">
  <tsk:task ID="1">
    <com:taskName>task1</com:taskName>
    <com:taskState>WAITING</com:taskState>
    <com:capabilityVector>
      <com:capability>
        <com:capabilityType>AMOUNT</com:capabilityType>
        <com:capabilityValue>6000</com:capabilityValue>
        <com:capabilityUnit>kg</com:capabilityUnit>
      </com:capability>
      <com:capability>
        <com:capabilityType>TIME</com:capabilityType>
        <com:capabilityValue>12</com:capabilityValue>
        <com:capabilityUnit>h</com:capabilityUnit>
      </com:capability>
      <com:capability>
        <com:capabilityType>DETAIL</com:capabilityType>
        <com:capabilityValue>1000</com:capabilityValue>
        <com:capabilityUnit>km</com:capabilityUnit>
      </com:capability>
    </com:capabilityVector>
  </tsk:task>
  <tsk:task ID="2">
    <com:taskName>task2</com:taskName>
    <com:taskState>WAITING</com:taskState>
    <com:capabilityVector>
      <com:capability>
        <com:capabilityType>AMOUNT</com:capabilityType>
        <com:capabilityValue>5000</com:capabilityValue>
        <com:capabilityUnit>kg</com:capabilityUnit>
      </com:capability>
      <com:capability>
        <com:capabilityType>TIME</com:capabilityType>
        <com:capabilityValue>8</com:capabilityValue>
        <com:capabilityUnit>h</com:capabilityUnit>
      </com:capability>
      <com:capability>
        <com:capabilityType>DETAIL</com:capabilityType>
        <com:capabilityValue>1000</com:capabilityValue>
        <com:capabilityUnit>km</com:capabilityUnit>
      </com:capability>
    </com:capabilityVector>
  </tsk:task>
</tsk:taskList>
```

EK A3: report.xml

```
<rep:executionReportList
xmlns:com="http://www.agentcoalition.com/XML_common_types"
xmlns:rep="http://www.agentcoalition.com/XML_report_types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.agentcoalition.com/XML_report_types
../xsds/report.xsd ">
   <rep:report ID="3">
      <rep:taskControlPeriodTime>2014-01-
03T16:20:29.342+02:00</rep:taskControlPeriodTime>
      <rep:maxCoalitionSize>2</rep:maxCoalitionSize>
      <rep:coalitionFormationDuration>1047</rep:coalitionFormationDuration>
      <rep:coalitionDetails>
         <rep:coalitionManager>
            <com:localName>coalitionManagerAgent1</com:localName>
            <com:platformName>managerPlatform</com:platformName>
            <com:address>http://merve-laptop:6000/acc</com:address>
            <rep:logName>logs/coalitionManagerAgent1_12.log</rep:logName>
         </rep:coalitionManager>
         <rep:taskDetails ID="1">
            <com:taskName>task1</com:taskName>
            <com:taskState>FINISHED</com:taskState>
            <rep:executionTime>
               <rep:startTime>2014-01-03T16:20:56.504+02:00</rep:startTime>
               <rep:stopTime>2014-01-03T16:21:57.114+02:00</rep:stopTime>
            </rep:executionTime>
         </rep:taskDetails>
         <rep:employeeAgentList>
            <rep:agentInf>
               <com:localName>A2</com:localName>
               <rep:successful>true</rep:successful>
               <rep:executionTime>
                  <rep:startTime>2014-01-03T16:20:56.504+02:00</rep:startTime>
                  <rep:stopTime>2014-01-03T16:21:20.055+02:00</rep:stopTime>
               </rep:executionTime>
            </rep:agentInf>
            <rep:agentInf>
               <com:localName>A6</com:localName>
               <rep:successful>true</rep:successful>
               <rep:executionTime>
                  <rep:startTime>2014-01-03T16:20:56.504+02:00</rep:startTime>
                  <rep:stopTime>2014-01-03T16:21:57.114+02:00</rep:stopTime>
               </rep:executionTime>
            </rep:agentInf>
         </rep:employeeAgentList>
      </rep:coalitionDetails>
      <rep:coalitionDetails>
         <rep:coalitionManager>
            <com:localName>coalitionManagerAgent2</com:localName>
```

<com:platformName>managerPlatform</com:platformName>
<com:address>http://merve-laptop:6100/acc</com:address>
<rep:logName>logs/coalitionManagerAgent2_13.log</rep:logName>
</rep:coalitionManager>
<rep:taskDetails ID="2">
<com:taskName>task2</com:taskName>
<com:taskState>FINISHED</com:taskState>
<rep:executionTime>
<rep:startTime>2014-01-03T16:20:56.506+02:00</rep:startTime>
<rep:stopTime>2014-01-03T16:22:12.054+02:00</rep:stopTime>
</rep:executionTime>
</rep:taskDetails>
<rep:employeeAgentList>
<rep:agentInf>
<com:localName>A7</com:localName>
<rep:successful>true</rep:successful>
<rep:executionTime>
<rep:startTime>2014-01-03T16:20:31.506+02:00</rep:startTime>
<rep:stopTime>2014-01-03T16:22:12.054+02:00</rep:stopTime>
</rep:executionTime>
</rep:agentInf>
<rep:agentInf>
<com:localName>A3</com:localName>
<rep:successful>true</rep:successful>
<rep:executionTime>
<rep:startTime>2014-01-03T16:20:31.150+02:00</rep:startTime>
<rep:stopTime>2014-01-03T16:21:53.114+02:00</rep:stopTime>
</rep:executionTime>
</rep:agentInf>
</rep:employeeAgentList>
</rep:coalitionDetails>
</rep:report>

</rep:executionReportList>

EK A1: users.xml

```
<usr:userList xmlns:com="http://www.agentcoalition.com/XML_common_types"
xmlns:usr="http://www.agentcoalition.com/XML_user_types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.agentcoalition.com/XML_user_types
xmldb:exist:///db/ACS/xsds/users.xsd">
        <usr:user ID="1">
                <usr:userName>admin</usr:userName>
                <usr:password>admin</usr:password>
        </usr:user>
        <usr:user ID="2">
                <usr:userName>merve</usr:userName>
                <usr:password>merve</usr:password>
        </usr:user>
</usr:userList>
```

EK B

EK B1: commonDefinitions.xsd

```
<xs:schema xmlns="http://www.agentcoalition.com/XML_common_types"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.agentcoalition.com/XML_common_types"
elementFormDefault="qualified">
   <xs:element name="capabilityVector">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="capability" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="capability">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="capabilityType"/>
            <xs:element ref="capabilityValue"/>
            <xs:element ref="capabilityUnit"/>
            <xs:element ref="evaluationFunction" minOccurs="0"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="capabilityType">
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="AMOUNT"/>
            <xs:enumeration value="TIME"/>
            <xs:enumeration value="COST"/>
            <xs:enumeration value="DETAIL"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:element>
   <xs:element name="capabilityValue" type="xs:string"/>
   <xs:element name="capabilityUnit">
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="kg"/>
            <xs:enumeration value="m2"/>
            <xs:enumeration value="day"/>
            <xs:enumeration value="h"/>
            <xs:enumeration value="min"/>
            <xs:enumeration value="sec"/>
            <xs:enumeration value="TL"/>
            <xs:enumeration value="m"/>
            <xs:enumeration value="km"/>
         </xs:restriction>
      </xs:simpleType>
```

```
    </xs:element>
    <xs:element name="evaluationFunction" type="xs:string"/>
    <xs:element name="localName" type="xs:string"/>
    <xs:element name="platformName" type="xs:string"/>
    <xs:element name="address" type="xs:string"/>
    <xs:element name="agentState">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="INITIATED"/>
          <xs:enumeration value="ACTIVE"/>
          <xs:enumeration value="IDLE"/>
          <xs:enumeration value="SUSPENDED"/>
          <xs:enumeration value="WAITING"/>
          <xs:enumeration value="DELETED"/>
          <xs:enumeration value="TRANSIT"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="taskState">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="WAITING"/>
          <xs:enumeration value="COMPUTING"/>
          <xs:enumeration value="RUNNING"/>
          <xs:enumeration value="FINISHED"/>
          <xs:enumeration value="FAILED"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="taskName" type="xs:string"/>
</xs:schema>
```

EK B2: users.xsd

```
<xs:schema xmlns="http://www.agentcoalition.com/XML_user_types"
xmlns:com="http://www.agentcoalition.com/XML_common_types"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.agentcoalition.com/XML_user_types"
elementFormDefault="qualified">
   <xs:import namespace="http://www.agentcoalition.com/XML_common_types"
schemaLocation="commonDefinitions.xsd"/>
   <xs:element name="userList">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="user" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="user">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="userName"/>
            <xs:element ref="password"/>
         </xs:sequence>
         <xs:attribute name="ID" use="required">
            <xs:simpleType>
               <xs:restriction base="xs:int"/>
            </xs:simpleType>
         </xs:attribute>
      </xs:complexType>
   </xs:element>
   <xs:element name="userName" type="xs:string"/>
   <xs:element name="password" type="xs:string"/>
</xs:schema>
```

EK B3: employeeAgents.xsd

```
<xs:schema xmlns="http://www.agentcoalition.com/XML_employeeagent_types"
xmlns:com="http://www.agentcoalition.com/XML_common_types"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.agentcoalition.com/XML_employeeagent_types"
elementFormDefault="qualified">
   <xs:import namespace="http://www.agentcoalition.com/XML_common_types"
schemaLocation="commonDefinitions.xsd"/>
   <xs:element name="employeeAgentList">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="employeeAgent" minOccurs="0"
maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="employeeAgent">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="com:localName"/>
            <xs:element ref="com:platformName"/>
            <xs:element ref="com:address"/>
            <xs:element ref="com:agentState"/>
            <xs:element ref="com:capabilityVector"/>
            <xs:element ref="available"/>
         </xs:sequence>
         <xs:attribute name="ID" use="required">
            <xs:simpleType>
               <xs:restriction base="xs:int"/>
            </xs:simpleType>
         </xs:attribute>
      </xs:complexType>
   </xs:element>
</xs:schema›
```

EK B4: tasks.xsd

```
<xs:schema xmlns="http://www.agentcoalition.com/XML_task_types"
xmlns:com="http://www.agentcoalition.com/XML_common_types"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.agentcoalition.com/XML_task_types"
elementFormDefault="qualified">
   <xs:import namespace="http://www.agentcoalition.com/XML_common_types"
schemaLocation="commonDefinitions.xsd"/>
   <xs:element name="taskList">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="task" minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="task">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="com:taskName"/>
            <xs:element ref="com:taskState"/>
            <xs:element ref="com:capabilityVector"/>
            <xs:element ref="userId"/>
         </xs:sequence>
         <xs:attribute name="ID" use="required">
            <xs:simpleType>
               <xs:restriction base="xs:int"/>
            </xs:simpleType>
         </xs:attribute>
      </xs:complexType>
   </xs:element>
   <xs:element name="userId" type="xs:int"/>
</xs:schema>
```

EK B5: report.xsd

```
<xs:schema xmlns="http://www.agentcoalition.com/XML_report_types"
xmlns:com="http://www.agentcoalition.com/XML_common_types"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.agentcoalition.com/XML_report_types"
elementFormDefault="qualified">
   <xs:import namespace="http://www.agentcoalition.com/XML_common_types"
   schemaLocation="commonDefinitions.xsd"/>
   <xs:element name="executionReportList">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="report" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="report">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="taskControlPeriodTime"/>
            <xs:element ref="maxCoalitionSize"/>
            <xs:element ref="coalitionFormationDuration"/>
            <xs:element ref="coalitionDetails" maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attribute name="ID" use="required">
            <xs:simpleType>
               <xs:restriction base="xs:int"/>
            </xs:simpleType>
         </xs:attribute>
      </xs:complexType>
   </xs:element>
   <xs:element name="taskControlPeriodTime" type="xs:dateTime"/>
   <xs:element name="coalitionFormationDuration" type="xs:string"/>
   <xs:element name="maxCoalitionSize" type="xs:int"/>
   <xs:element name="coalitionDetails">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="coalitionManager"/>
            <xs:element ref="taskDetails"/>
            <xs:element ref="employeeAgentList"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="coalitionManager">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="com:localName"/>
            <xs:element ref="com:platformName"/>
            <xs:element ref="com:address"/>
            <xs:element ref="logName"/>
```

```
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="logName" type="xs:string"/>
    <xs:element name="taskDetails">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="com:taskName"/>
          <xs:element ref="com:taskState"/>
          <xs:element ref="executionTime"/>
        </xs:sequence>
        <xs:attribute name="ID" use="required">
          <xs:simpleType>
            <xs:restriction base="xs:int"/>
          </xs:simpleType>
        </xs:attribute>
      </xs:complexType>
    </xs:element>
    <xs:element name="executionTime">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="startTime"/>
          <xs:element ref="stopTime"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="startTime" type="xs:dateTime"/>
    <xs:element name="stopTime" type="xs:dateTime"/>
    <xs:element name="employeeAgentList">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="agentInf" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="agentInf">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="com:localName"/>
          <xs:element ref="successful" minOccurs="0"/>
          <xs:element ref="executionTime"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="successful" type="xs:boolean"/>
</xs:schema>
```

EK C

EK C1: agentsModule.xquery

```
xquery version "1.0";
module namespace agm = "http://www.agentcoalition.com/agm";
declare namespace com = "http://www.agentcoalition.com/XML_common_types";
declare namespace eagnt =
"http://www.agentcoalition.com/XML_employeeagent_types";

declare function agm:activateEmployeeAgent($agent as
element(eagnt:employeeAgent))
{
   let $agentDocumentUrl := "//db/ACS/xmls/employeeAgents.xml"

   for $agentDef in
doc($agentDocumentUrl)/eagnt:employeeAgentList/eagnt:employeeAgent
   where ((fn:lower-case($agentDef/com:localName)=fn:lower-
case($agent/com:localName))
         and (fn:lower-case($agentDef/com:platformName)=fn:lower-
case($agent/com:platformName))
         and (fn:lower-case($agentDef/com:address)=fn:lower-
case($agent/com:address)))
   return update replace $agentDef/com:agentState with
<com:agentState>ACTIVE</com:agentState>
};

declare function agm:getEmployeeAgentList() as element(eagnt:employeeAgent)*
{
   for $agentDef in
doc("//db/ACS/xmls/employeeAgents.xml")/eagnt:employeeAgentList/eagnt:employ
eeAgent
   return $agentDef
};

declare function agm:getActiveEmployeeAgentList() as
element(eagnt:employeeAgent)*
{
   for $agentDef in
doc("//db/ACS/xmls/employeeAgents.xml")/eagnt:employeeAgentList/eagnt:employ
eeAgent
   where ($agentDef/com:agentState="ACTIVE")
   return $agentDef
};

declare function agm:updateEmployeeAgentList($agent as
element(eagnt:employeeAgent))
{
   let $agentCount := count(agm:searchEmployeeAgent($agent))
```

```
   let $result := if ($agentCount = 0)
      then agm:insertEmployeeAgent($agent)
      else agm:activateEmployeeAgent($agent)

   return $result
};

declare function agm:insertEmployeeAgent($agent as
element(eagnt:employeeAgent))
{
   let $agentDocumentUrl := "//db/ACS/xmls/employeeAgents.xml"
   return update insert $agent into
doc($agentDocumentUrl)/eagnt:employeeAgentList
};

declare function agm:searchEmployeeAgent($agent as
element(eagnt:employeeAgent)) as element(eagnt:employeeAgent)*
{
   let $agentDocumentUrl := "//db/ACS/xmls/employeeAgents.xml"

   for $agentDef in
doc($agentDocumentUrl)/eagnt:employeeAgentList/eagnt:employeeAgent
   return if ((fn:lower-case($agentDef/com:localName)=fn:lower-
case($agent/com:localName))
         and (fn:lower-case($agentDef/com:platformName)=fn:lower-
case($agent/com:platformName))
         and (fn:lower-case($agentDef/com:address)=fn:lower-
case($agent/com:address)))
      then $agentDef
      else ( )
};
```

EK C2: taskModule.xquery

```
xquery version "1.0";
module namespace tm = "http://www.agentcoalition.com/tm";
declare namespace com = "http://www.agentcoalition.com/XML_common_types";
declare namespace tsk = "http://www.agentcoalition.com/XML_task_types";

declare function tm:searchTaskById($id as xs:integer) as element(tsk:task)?
 {
   for $taskDef in doc("//db/ACS/xmls/tasks.xml")/tsk:taskList/tsk:task
   where $taskDef/@ID = $id
   return $taskDef
};

declare function tm:updateTask($taskDefinition as element(tsk:task))
{
   for $taskDef in doc("//db/ACS/xmls/tasks.xml")/tsk:taskList/tsk:task
   where $taskDef/@ID = $taskDefinition/@ID
   return update replace $taskDef with $taskDefinition
};

declare function tm:getTaskList() as element(tsk:task)*
{
   for $taskDef in doc("//db/ACS/xmls/tasks.xml")/tsk:taskList/tsk:task
   return $taskDef
};
```

EK C3: reportModule.xquery

```
xquery version "1.0";
module namespace rprt = "http://www.agentcoalition.com/rprt";
declare namespace com = "http://www.agentcoalition.com/XML_common_types";
declare namespace rep = "http://www.agentcoalition.com/XML_report_types";

declare function rprt:getNextId() as xs:integer
{
   let $reportDocumentUrl := "//db/ACS/xmls/report.xml"
   let $doc := doc($reportDocumentUrl)

   let $idList := for $idValue in $doc/rep:executionReportList/rep:report/@ID
                  return $idValue

   let $nextId := max($idList)

   return $nextId + 1
};

declare function rprt:insertReport($report as element(rep:report))
{
   let $reportDocumentUrl := "//db/ACS/xmls/report.xml"
   return update insert $report into
doc($reportDocumentUrl)/rep:executionReportList
};
```

EK C4: userModule.xquery

```
xquery version "1.0";
module namespace user = "http://www.agentcoalition.com/user";
declare namespace com = "http://www.agentcoalition.com/XML_common_types";
declare namespace usr = "http://www.agentcoalition.com/XML_user_types";

declare function user:login($userName as xs:string, $password as xs:string) as
element(usr:user)?
 {
   for $userDef in doc("//db/ACS/xmls/users.xml")/usr:userList/usr:user
   return if (($userDef/usr:userName = $userName) and ($userDef/usr:password =
$password))
   then $userDef
   else ( )
};
```

ÖZGEÇMİŞ



Ad Soyad: Merve ÖZBEY

Doğum Yeri ve Tarihi: ÜSKÜDAR / 09.08.1986

E-Posta: merve.ozbey@live.com

Lisans: Çankaya Üniversitesi / Bilgisayar Mühendisliği

Yüksek Lisans: İstanbul Teknik Üniversitesi / Bilgisayar Mühendisliği

Mesleki Deneyim ve Ödüller:

Kasım 2010 – Aralık 2013 Likya Bilgi Teknolojileri ve İletişim Hiz. Ltd. Şti.

Yayın ve Patent Listesi:

- Özbey, M., Sarıbıyık, H. ve Öztaş, E. (2011). İş Yükü Yönetimi Yazılımları İçin Bir Değerlendirme Metodolojisi, Ulusal Yazılım Mühendisliği Sempozyumu (UYMS), ODTÜ, Eylül 2011.

TEZDEN TÜRETİLEN YAYINLAR/SUNUMLAR

- Özbey, M. ve Erdoğan, N. (2014). Multi-Agent Based Execution Environment for Task Allocation via Coalition Formation. In: 8th International KES Conference on Agents and Multi-agent Systems – Technologies and Applications, June 18-20, 2014 Chania, Greece. (Yayınlanma aşamasında)