JADE İle FIPA Uyumlu Çok-Etmenli Bir Açık Artırma Sisteminin Gerçekleştirilmesi

İsmail Baş
Ege Üniversitesi, Bilgisayar Mühendisliği Bölümü, İzmir
issmailbas@gmail.com

Özet: Bu çalışmada JADE etmen çerçevesini kullanan ve FIPA standartlarına uyan çok-etmenli bir açık artırma sistemi tasarlanmış ve gerçekleştirilmiştir. Bildiride etmenlerin konteynır denilen ortamda oluşturulup bu ortam içerisine dahil edilmesi, rollerinin belirlenmesi, görevlerinin belirlenmesi, birbirleri ile olan etkileşimlerin belirlenmesi, görevlerini yerine getirdikten sonra sistemden ayrılmaları ve sistemin nasıl oluşturulduğu ve çalıştığıyla ilgili çalışmalara yer verilmiştir.

Anahtar Sözcükler: Yazılım Etmenleri, Yazılım Etmenlerinin İletişimi, Çok etmenli Sistemler, Yazılım Emenlerinin Rolleri ve Görevleri

1. Giriş

Nesne tabanlı programlamada sınıflardan türetilen objeler yani nesneler sadece yazılım geliştiricisinin o nesneye tanımlamış olduğu sınırlar çerçevesinde amaç ve görevleri gerçekleştirirler. Nesneler kendileri karar veremez ancak ve ancak onlara tanımlanmış olan yol üzerinden amaca ulaşırlar ve bu yolun sınırlarının dışına hiçbir şekilde çıkamazlar. Etmenler ise nesnelerden daha üstün özelliklere sahip olan oluşumlar veya bileşenlerdir. Etmenler de aynı şekilde belli bir amacı gerçekleştirmek için tanımlanmışlardır ancak bu amaca ulaşabilmek için izleyecekleri yolu kullanmakta ve bu amacı gerçekleştirebilmek için insiyatif almakta özgürdürler. Kendilerine belirli bir yol tanımlanmamıştır, sadece amaca ulaşabilmeleri için o amaca giden yolu nasıl inşa edeceklerinin bilgisi ve koşulları verilmiştir. Etmenler bu bilgiler veya koşullara göre karar verir ve gerekirse bazı durumlarda insiyatif alarak amaca ulaşmaya çalışırlar. Bu bakımdan etmenler zeki davranışlar gösterebilen oluşumlar veya bileşenlerdir. Çok etmenli sistemler ise bir etmenin kendi başına ulaşmakta zorlandığı amaçlara etmenlerin bir araya gelerek, belli roller ve görevler üstlenerek, birbirleri ile iş birliği yaparak, koordine olarak, anlaşarak ulaştıkları sistemlerdir. Bu sistemler birden çok etmeni bir araya getirerek sosyal etkileşim içeren amaçları kapsayan ortamların tasarlanıp simüle edilebilmelerini sağlar.

Açık artırmalar herhangi değer bulabilecek bir ürün, mal veya eşyanın satıcı tarafından taban fiyat belirlenerek bu ürünü almak isteyen alıcı topluluğuna bu

taban fiyat üzerinden teklifin yapıldığı, alıcıların ise bu ürünü alabilmek için herhangi bir alıcının kabul ettiği son fiyatı arttırmak zorunda olduğu ve en yüksek fiyatı veren alıcıdan daha yüksek bir fiyat veren alıcı çıkmayana kadar devam eden, sonunda da en yüksek fiyatı veren alıcının satışa sunulan ürüne sahip olduğu satış yöntemidir. Bu çalışmada bu tarz bir satış yöntemi yukarıda ifade edilmiş olan çok etmenli sistemlerle tasarlanıp uygulanmıştır. Uygulanmış olan bu sistem FIPA'nın (Foundation For Intelligent Physical Agents) standartlaryla uyumlu olarak çalışan JADE (Java Agent Development Framework) çerçevesini kullanmaktadır. FIPA, çok etmenli sistemlerde etmenlerin en iyi şekilde birbirleri ile iş birliği yapabilmeleri, anlaşabilmeleri, etkileşimde bulunabilmeleri için evrensel standartlar ortaya koyan, bunu herhangi bir kazanç için yapmayan, çok etmenli yazılım mimarilerinin ortak bir standardını belirleyerek bu yazılım mimarilerinin geliştirilmesini kolaylaştırmak amacıyla bir araya gelmiş bir topluluktur. JADE ise Java üzerinde FIPA'ya uyumlu etmenlerin oluşturulabilmesi ve kullanılabilmesini sağlayan bir yazılım çerçevesidir. Bildirinin ikinci bölümünde sistemin nasıl tasarlandığı, üçüncü bölümünde sistemin çalıştırılması ve elde edilen sonuçlar, dördüncü bölümde ise bu çalışma sonucunda hangi çıkarımların yapıldığı anlatılmıştır.

2. Sistemin Tasarlanması

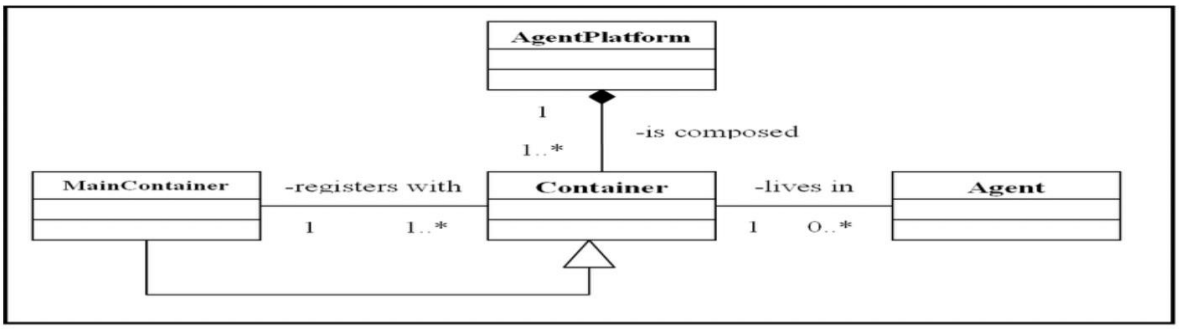
Sistemin tasarımında etmenleri barındıran ve bu etmenlerin aynı ortamda olabilmelerini sağlayarak iletişim kurabilmelerine imkân veren JADE çerçevesinde konteynır diye adlandırılan bir ortam, bu ortam içerisinde açık artırmayı düzenleyen satıcı etmen ve satıcı etmenin satışa sunduğu ürünlere teklif yapan alıcı etmenler yer almaktadır. Bu etmenlerin FIPA ile uyumlu bir servise kaydedilmesi ve iletişimlerini bu servis üzerinden gerçekleştirmeleri sistem içerisinde tanımlanmıştır. Etmenlerin davranış türleri, özellikleri ve rolleri de tanımlanmış olup sistemin tasarımı tamamlanmıştır. Sistemin tasarımının daha detaylı anlatımı bu bölümün alt başlıkları altında anlatılmıştır.

2.1 Etmenlerin Bir Arada Bulunabileceği Bir Konteynır Ortamının Oluşturulması ve Etmenlerin Bu Ortama Dahil Edilmesi

Etmenlerin bir konteynır ile yani bir ortam ile ilişkilendirilip bu ortam içerisinde çalıştırılmaya başlanması **Main.java** dosyası içerisinde gerçekleştirilmiştir. Sistemin çalışmasının ilk tetiklendiği yer bu java dosyası içerisindeki **Main** sınıfı içerisinde yer alan **public static void main(String[] args) {}** fonksiyonudur. Bu fonksiyon içerisinde **jade.core.Agent** sınıfından kalıtlanmış olan etmen nesneleri oluşturularak konteynır içerisine dahil edilmişler ve start metotları çalıştırılarak sistem içerisinde çalışmaya başlamaları sağlanmıştır.

Buradaki işlemler daha detaylı anlatılacak olursa öncelikle **jade.core.Runtime** sınıfına ait bir JADE çalışma ortamı oluşturulur. **jade.core.Profile** sınıfına ait bir profil oluşturularak bu profil nesnesi üzerinden konteynırın adı, konteynırın hangi sunucuda çalışacağı (localhost vb.), bu konteynırın JADE'in sağlamış olduğu arayüzü çalıştırıp çalıştırmayacağı gibi ayar parametreleri profilin **setParameter** metodu ile oluşturulur. Runtime nesnesinin **createMainContainer** metodu çağrılarak bu metoda oluşturulan profil parametre olarak verilir böylece konteynır bu profilin tanımlandığı özellikler ile oluşur. Bu konteynırın **createNewAgent** metodu çağrılarak hangi etmen oluşturulacak ise programda o etmenin sınıfının tanımlandığı yolu ve hangi ad ile oluşturulacağı parametre olarak verilerek etmen konteynır içerisinde oluşturulur. Etmen oluşturulduktan sonra etmenin **start** metodu çalıştırılır ve start metodu çalıştırılan her etmen sistemde çalışmaya başlar. Sistemde çalışmaya başlayan etmenler, sınıflarının içerisinde tanımlanmış olan davranışlara göre sistemde belli bir amacı gerçekleştirmeye çalışırlar. Bu çalışmada da test için üç adet açık artırmaya katılan alıcı tipinde etmen, bir adet de açık artırmayı düzenleyen satıcı etmen oluşturulmuş ve aynı konteynır içerisine dahil edilerek çalıştırılmışlardır. Alıcı etmenler **auction.BidderAgent** sınıf dosyası içerisinde tanımlanmışlar, satıcı etmenler ise **auction.AuctionerAgent** sınıf dosyası içerisinde tanımlanmışlardır.

Etmenler çalıştıkları zaman ilk olarak **setup** metodunu çalıştırırlar. **setup** metodu içerisinde FIPA servisine kayıtlanır ve gerekli davranışları kuyruk yapılarına alarak bu davranışları FIPA standartlarına uygun bir şekilde gerçekleştirmeye başlarlar. Etmenler, ortamda işleri bittiğinde ise **doDelete** metodunu çağırarak kendilerini imha ederler ve konteynır içerisinden kaldırılırlar. **doDelete** metodunun çalışabilmesi için **takeDown** metodu etmen içerisinde tanımlanmalıdır. Bu metotta etmenler sistemden çıkmadan önce gerçekleştirmesi gereken işlemleri gerçekleştirirler ve bu işlemlerden sonra sistemden ayrılırlar. **setup ve takedown** metotları etmenlerin içinde tanımlı olması gereken en önemli iki metottur.



Şekil 1. JADE'in temel elemanlarının UML diagramı ile gösterimi (Regionally Distributed Architecture for Dynamic e-Learning Environment Bildirisinden Alınmıştır)

2.2 Satıcı Etmen Sınıfının Tanımlanması ve Davranışlarının Belirlenmesi

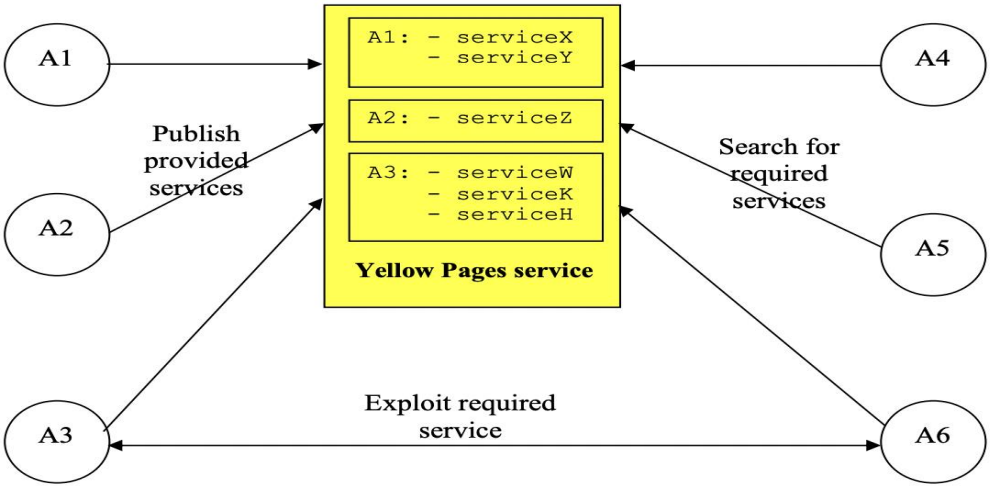
Satıcı etmenin niteliklerinde (attribute) java**.util.*** koleksiyonu içerisinde yer alan rastgele sayılar üretmeye yarayan bir **Random** tipinde nesne, etmenler ile iletişim kurabilmek için etmenlerin adresini ve bilgilerini içeren **jade.core.AID** sınıfına ait bir etmen tanım nesnesi, hangi ürünlerin hangi taban fiyattan satışa sunulacağını belirten **String** tipinde diziler içeren bir **ArrayList** tipinde liste (bu liste iki boyutlu iç içe tanımlanmış bir listedir), **String** tipinde **açık** artırması devam eden mevcut ürün, kaç adet teklif yapıldığının sayısının tutulduğu **int** tipinde bir sayı, satıcının adının tutulduğu bir **String** değişken ve davranışların çalışmasını kontrol edebilmek için de **Boolean** tipinde bir lock değişkeni tanımlanmıştır.

setup metodu etmenin çalıştırdığı ilk metottur. Burada etmenin adı **jade.core.Agent** içerisinde tanımlı olan **getAID** metodu ile **'getAID().getName()**' şeklinde etmenin ismine erişilir ve etmenin isim değişkeni oluşturulur. Satıcının satışa sunacağı ürünlerin taban fiyatları **Random** nesnesi ile 0 ile 10000 birim fiyat aralığında olacak şekilde her bir ürün için oluşturulur ve ürünler adları ve taban fiyatları ile birlikte satıcının kataloğuna ikili listeler şeklinde eklenir. Böylece **ArrayList({"item_1", 200}, {"item_2", 4000} …….)** şeklinde satıcının ürün kataloğu oluşturulmuş olur. Bu işlemlerden sonra **CyclicBehaviour** davranışı eklenerek satıcı etmenin açık artırmada her ürün satışından sonra bu davranışının çalıştırılması sağlanır. **CyclicBehaviour** döngü gibi devamlı çalışan davranışlar olduğu için ve bu çalışmada satıcı etmenin bu davranışı çalıştırması ilk sefer ve ürün satışlarından sonra gerçekleştirmesi gerektiğinden dolayı **lock** bayrak değişkeni ilk çalışmada **false** olarak tanımlanmış, ürün açık artırmaya girdiğinde **true** olarak değiştirilmiş ve ürün satıldığında yeniden **false** yapılarak buradaki davranışın yeninden çalıştırılması sağlanmıştır. Bayrak değişkeni **false** iken **CyclicBehaviour** çalışır ve **jade.domain.FIPAAgentManagement.DFAgentDescription** tipinde bir etmen tanımlayıcı nesnesi oluşturulur. **jade.domain.FIPAAgentManagement.DFServiceDescription** tipinde de bir servis tanımlayıcı nesnesi oluşturulur. Bu servisin tipi "auction-bidding" olarak tanımlanmıştır. Oluşturulan etmen tanımlayıcı nesnesine **addServices** metodu ile oluşturulan servis parametre olarak bu metoda verilir ve etmenlerin birbirlerine erişebilmesini sağlayacak olan FIPA yapısı oluşturulmuş olur. Satıcı etmen **jade.domain.DFService** sınıfının sağlamış olduğu statik **search** metodu ile "auction-bidding" tipli bir servise kayıtlı olan alıcı etmenleri bularak bu etmenlerin **AID** tipinde kimliklerine erişir ve eğer kataloğunda ürün var ise bu etmenlere teklif yapmaları için istekte bulunur. Satıcı etmen istekte bulunmak için

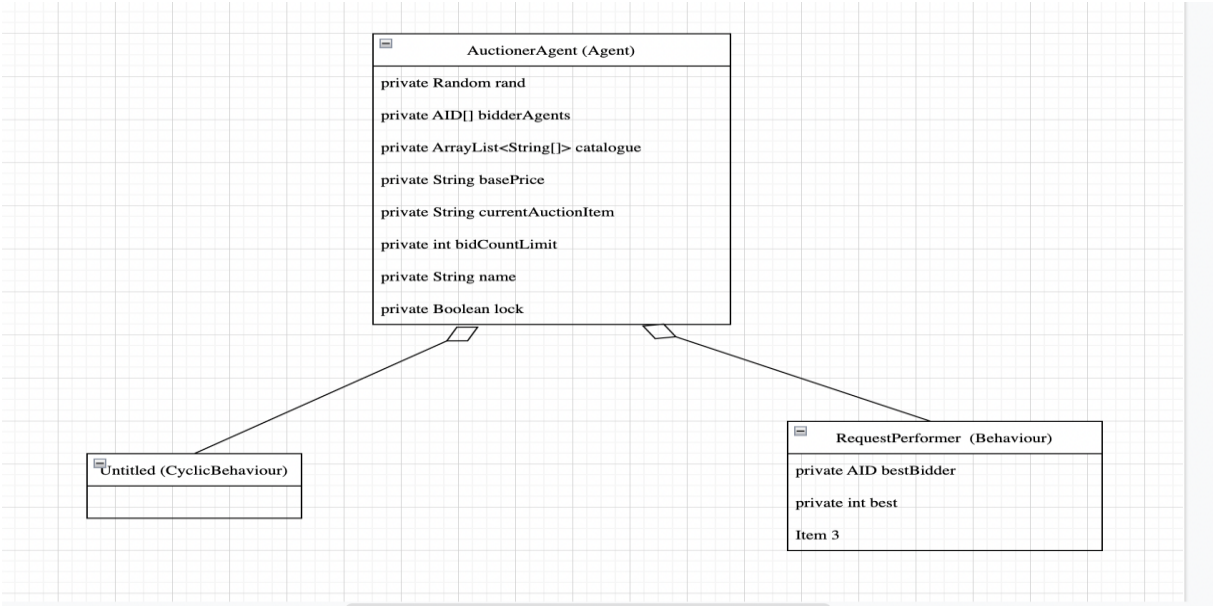
kataloğundan sıradaki ürünü ve ürünün taban fiyat bilgisini çeker. 5 ile 15 arasında da rastgele bir sayı oluşturarak açık artırmanın bitmesi için kaç sefer teklif yapılması gerektiğini önceden belirler. Bu işlemlerden sonra ürün için etmenlerden teklif yapmalarını istemek ve açık artırma işlemini gerçekleştirmek için **switch** koşulu ile çalışan normal bir **Behaviour** davranışını uygular.

Behaviour tipindeki davranışlar **done** metodunu uygular. Her switch koşulundan sonra bu **done** metodu çalıştırılır ve **done** metodunda belirli koşul sağlanmış ise bayrak yani **lock** değişkeni false yapılarak yukarıda açıklanan **CyclicBehaviour** yeniden çalışır. Teklif isteklerinin yapılacağı **Behaviour** sınıfının özniteliklerinde (attribute) en iyi teklifi yapan alıcı etmenin tutulacağı **AID** tipinde bir nesne, mevcut durumda yapılan en iyi teklifin fiyatını tutan **int** tipinde değişken, mevcut durumda kaç adet teklif yapıldığını belirten **int** tipinde değişken, açık artırmanın 1 turunda (1 tur her etmenin teklif yapması demek) yapılan mevcut teklif sayısını tutan **int** tipinde değişken, alıcı etmenlerden gelen cevapları mesaj kuyruğuna alabilmek için **jade.lang.acl.MessageTemplate** tipinde bir kuyruk nesnesi ve **switch** koşullarını kontrol edebilmek için **int** tipinde bir adım (step) değişkeni yer almaktadır. Step değişkeni ilk olarak 0'dır ve etmen 0'ıncı duruma (ilk durum) girerek **jade.lang.acl.ACLMessage** tipinde **ACLMessage.CFP** parametresi ile tanımlanmış bir ACL mesajı oluşturur. Bu parametre karşı taraftan teklif yapılması için mesajın teklif çağrısı niteliğinde olmasını sağlar. Bütün alıcı etmenler bu mesajın alıcılarına eklenir, mesajın içeriği açık artırmaya sunulan ürünün taban fiyatı olarak belirlenir, "auction" adı altında mesajın iletişim kimliği belirlenir ve mesaja özgü benzersiz özgün bir kimlik numarası oluşturularak mesaj tüm alıcılara gönderilir ve tüm alıcılardan bir teklif yapılması istenir. Step değişkeni 1 olarak ayarlanır. Mesaj gönderildikten sonra bir mesaj kuyruğu oluşturularak alıcıların yaptıkları tekliflerin bu mesaj kuyruğunda tutulması amaçlanır. Step değişkeni 1 ise gelen teklifler kuyruktan alınır ve en iyi fiyatı veren alıcı etmenin AID kimlik nesnesi ve teklif ettiği fiyat davranışın öznitelik değişkenlerinde güncellenir. Eğer 1 tur tamamlanmış ve hala alıcı etmenlerden teklif bekleniyor ise step değişkeni 0 yapılır ve yeniden tüm alıcı etmenlerden teklif yapılması istenir. Alıcı etmenler önceden belirlenen teklif limitine ulaşacak seviyede teklif yapana kadar bu işlem tekrarlanır. Teklif limitine ulaşıldığından en yüksek fiyatı veren alıcı ve teklif ettiği ücret davranışın özniteliklerinde güncellenir ve step değişkeni 2 yapılır. Step değişkeni 2 ise yine bir ACL mesajı oluşturulur ancak mesaj bu sefer **ACLMessage.ACCEPT_PROPOSAL** parametresi ile oluşturulur. Bu parametre mesajın teklifin kabul edildiğini belirten bir mesaj türü olduğunu belirtir ve açık artırmadaki mevcut ürünün adı mesajın içeriğine eklenerek alıcı etmene mesaj gönderilir. Bunun sonucunda satıcı etmen alıcı etmeni kimliği ile birlikte açıklayarak ürünü hangi fiyattan satın aldığını ilan eder. Eğer hiç teklif yapılmamış ise ürünün satılmadığını ilan eder. Bu 2. Adımda satıcı etmen kataloğundaki ürünü rafa kaldırır yani ürün yukarıda tanımlanmış ArrayList yapısından çıkartılır. Son olarak step değişkeni 3 olarak güncellenir.

done metodu çalıştığında step değişkeni 3 ise **lock** bayrak değişkeni yeniden false olarak güncellenerek CyclicBehaviour davranışının yeniden tetiklenmesi sağlanır. Bu işlemler satıcı etmenin elinde ürün kalmayana kadar devam eder. Satıcı etmenin elinde ürün kalmadığında CyclicBehaviour içerisinde bir **ACLMessage.INFORM** parametresi ile bir bilgi mesajı oluşturulur ve alıcı etmenlere herhangi bir ürünün kalmadığı ve açık artırmanın kapatıldığı bildirilir. Bu mesajdan sonra **doDelete** metodu çalıştırılarak etmen sonlanır ve sistemden çıkar. Bu bilgi mesajını alan alıcı etmenler de işlemlerini sonlandırır ve onlar da **doDelete** metodunu çağırarak sistemden ayrılırlar.



Şekil 2. Sarı Sayfalar servisi (https://jade.tilab.com/doc/tutorials/JADEProgramming-Tutorial-for-beginners.pdf)



Şekil 3. Satıcı etmen ve davranışları arasındaki ilişkinin UML diyagramı

2.3 Alıcı Etmen Sınıfının Tanımlanması ve Davranışlarının Belirlenmesi

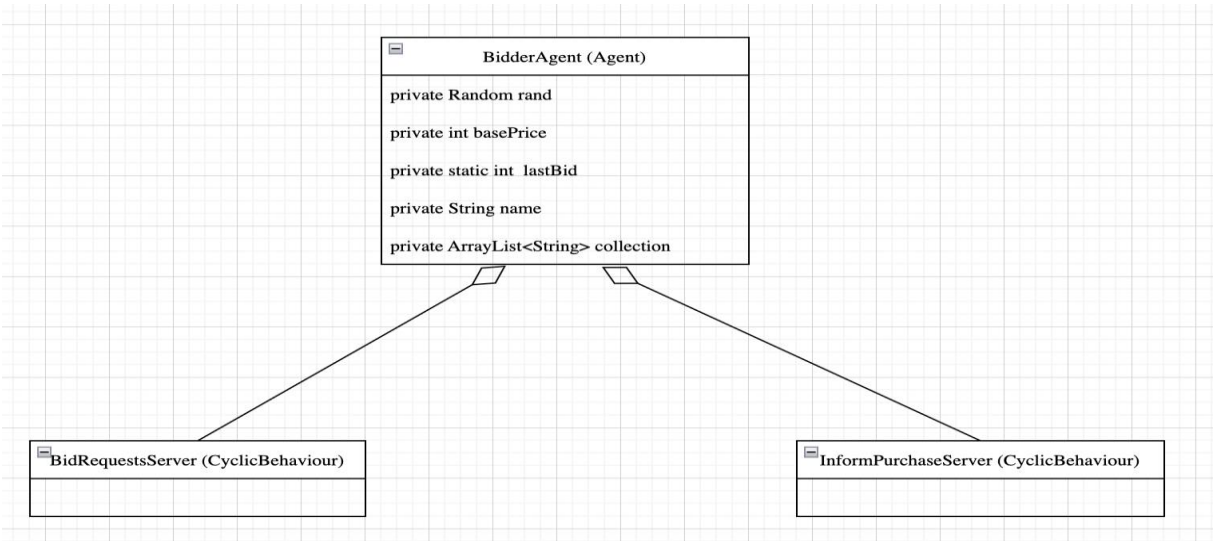
Alıcı etmen sınıfının özniteliklerinde **java.util.Random** tipinde bir değişken, satıcı etmenin teklif etmiş olduğu taban fiyatı tutan **int** tipinde bir değişken, yapılan en yüksek teklifin tutulduğu **static int** tipinde bir değişken, kendi adlarını tutan **String** tipinde bir değişken ve satın almış oldukları ürünlerin koleksiyonunu tutan **ArrayList<String>** tipinde bir değişken yer almaktadır. Son yapılan teklifin statik bir değişken olmasının nedeni bu değişkenin her alıcı etmen için ortak olması gerektiğindendir. Çünkü her alıcı etmenin en yüksek yapılan teklifin değerini bilmesi ve her teklif yaptığında bu değeri güncelleyebilmesi gerekmektedir. Bu da ancak bu değişkenin tüm alıcı etmenlerde ortak olması ile sağlanabilir. Bunu sağlayan anahtar kelime de **static** kelimesidir.

Alıcı etmenlerin setup metodu çalıştığında adları **getAID().getName()** ile oluşturulur. Satın alacakları eşyaların tutulacağı koleksiyon boş olarak oluşturulur. FIPA'nın servis yapısı bu etmenler içerisinde aynı satıcı etmende olduğu gibi tanımlanır. Tek farkı satıcı etmen bu servisi bu servise kayıtlı olan etmenleri sorgulamak için kullanır ancak bu servise kayıt olmaz. Alıcı etmenler ise kendilerini bu servise kayıtlayarak satıcı etmenin bu servis üzerinden kendilerini sorgulayabilmelerini sağlamış olurlar. Bu işlemlerden sonra alıcı etmenlerden teklif yapılması istenildiğinde teklif yapmalarını sağlayacak olan CyclicBehaviour tipindeki bir davranış ve eğer alıcı etmen bir ürün satın almışsa bu ürünü aldığını onayladığı ve koleksiyonunu güncellediği CyclicBehaviour tipindeki diğer bir davranış alıcı etmenin davranışlarına eklenerek bu davranışlar karşı taraftan gelen istek ve bildirimlere karşı bir döngü halinde dinleme durumuna geçerler.

Karşı taraftan teklif taleplerini dinleyen davranış aynı satıcı etmendeki gibi bir mesaj şablonu oluşturur. Bu mesaj şablonunun parametresini **ACLMessage.CFP** olarak belirler. Böylece sadece teklif çağrısı tipinde olan mesajlar bu mesaj şablonuna düşer. Mesajlar **myAgent.receive** fonksiyonuna mesaj şablonunun parametresi verilerek alınmaya çalışılır. Eğer mesaj gelmemiş ise döngü başa döner fakat bir teklif çağrısı gelmişse bu mesaj şablonuna düşer ve mesaj null olarak dönmez. Böylece alıcı etmen teklif yapabilir durumdadır. Gelen mesaj üzerinden çalıştırılan **createReply()** fonksiyonu ile bu mesaja ACLMessage tipinde karşı bir cevap nesnesi oluşturulur. ACLMessage cevap nesnesi üzerinden de **setPerformative** fonksiyonu çağrılır ve bu fonksiyonun parametresi **ACLMessage.PROPOSE** olarak belirlenir. Bu, mesajın teklif tipinde bir mesaj olduğunu belirtir. Alıcı etmen bu işlemlerden sonra bir teklif belirlemek ister. Bu teklifin oluşturulması için basit bir matematik uygulanmıştır. Bu matematik uygulanırken rastgele sayılar oluşturabilmek için **Random** tipinde olan öznitelik değişkeninden faydalanılmıştır. Öncelikle 0 ile 1 arasında bir sayı üretilir. Eğer 1

gelirse teklif yapılır 0 gelir ise teklif yapılmaz. Ancak yapılmayan teklifler de satıcı etmen tarafında teklif yapılmış kabul edilir ama bu teklifin değeri 0'dır. Eğer 1 geldi ise teklifin yapılmasına karar verilmiştir ve bunun için basit bir matematik işlemi daha uygulanır. Öncelikler 0 ile 100 arasında bir sayı üretilir. Bu sayı taban teklifin ne oranda artırılacağının belirlenmesini kontrol eder. Eğer sayı 1 ile 5 arasında ise taban teklif yüzde 25 ile yüzde 100 arasında bir oranla artırılır. Yani taban fiyat yüzde 5 ihtimalle yüzde 25 ile yüzde 100 arasında bir oranla artırılır. Aynı şekilde buna benzer diğer koşulda taban fiyat yüzde 10 ihtimalle yüzde 5 ile yüzde 25 arasında bir oranla artırılır. Geriye kalan yüzde 85 ihtimalle de taban fiyat yüzde 1 ile yüzde 5 arasında bir oranla artırılır. Teklif artırıldıktan sonra son yapılan en yüksek teklif yeniden güncellenir ve bu teklif String tipinde bir değere dönüştürülerek ACLMessage tipinde olan reply nesnesinin **setContent** fonksiyonuna parametre olarak verilerek mesajın içeriği bu teklif ile oluşturulur ve satıcı etmene bu mesaj iletilir. Teklifin String tipine dönüştürülmesinin nedeni **setContent** fonksiyonunun parametreyi String olarak kabul etmesindendir. Etmen teklifi yaptıktan sonra tekrar dinleme durumuna geçer. Eğer gelen mesajın türü **ACLMessage.Inform** yani bilgi amaçlı bir mesaj ise mesajın içeriğine bakar. Eğer satıcı etmen tarafından gelen mesajın içeriği "Auction has closed" ise bu açık artırmanın sonlandığı anlamına gelir ve etmen **doDelete()** metodunu çağırarak işlemlerini sonlandırır ve sistemden ayrılır.

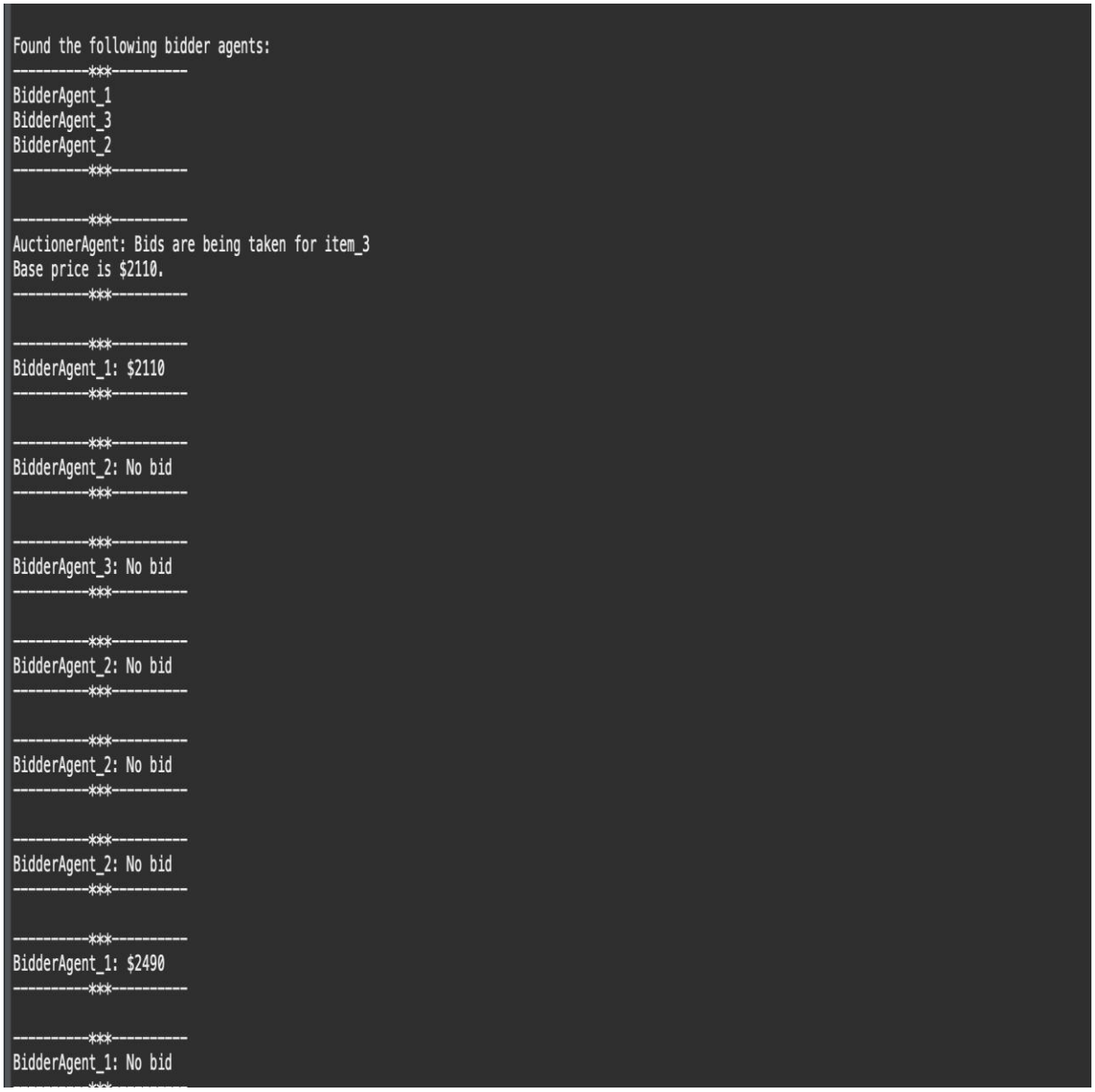
Eğer alıcı etmen bir ürünü satın almışsa satıcı etmen tarafından gönderilen kendisinin ürünü satın aldığının bilgisini içeren mesajı diğer davranış içerisinde döngüsel bir halde aynı şekilde dinler. Eğer gelen mesajın eylemi teklifin kabulü ise bu mesajı alır. Mesajın içerisinde satın aldığı ürünün ismi bulunmaktadır. Bu ürünün adını kendi koleksiyonuna ekler ve yapılan son teklifi yeniden 0'a eşitleyerek kendisinin satışta olan mevcut ürünün yeni sahibi olduğunu ilan eder. Sistem böylece yeni ürünün açık artırmaya sunulmasına hazır hale gelir.



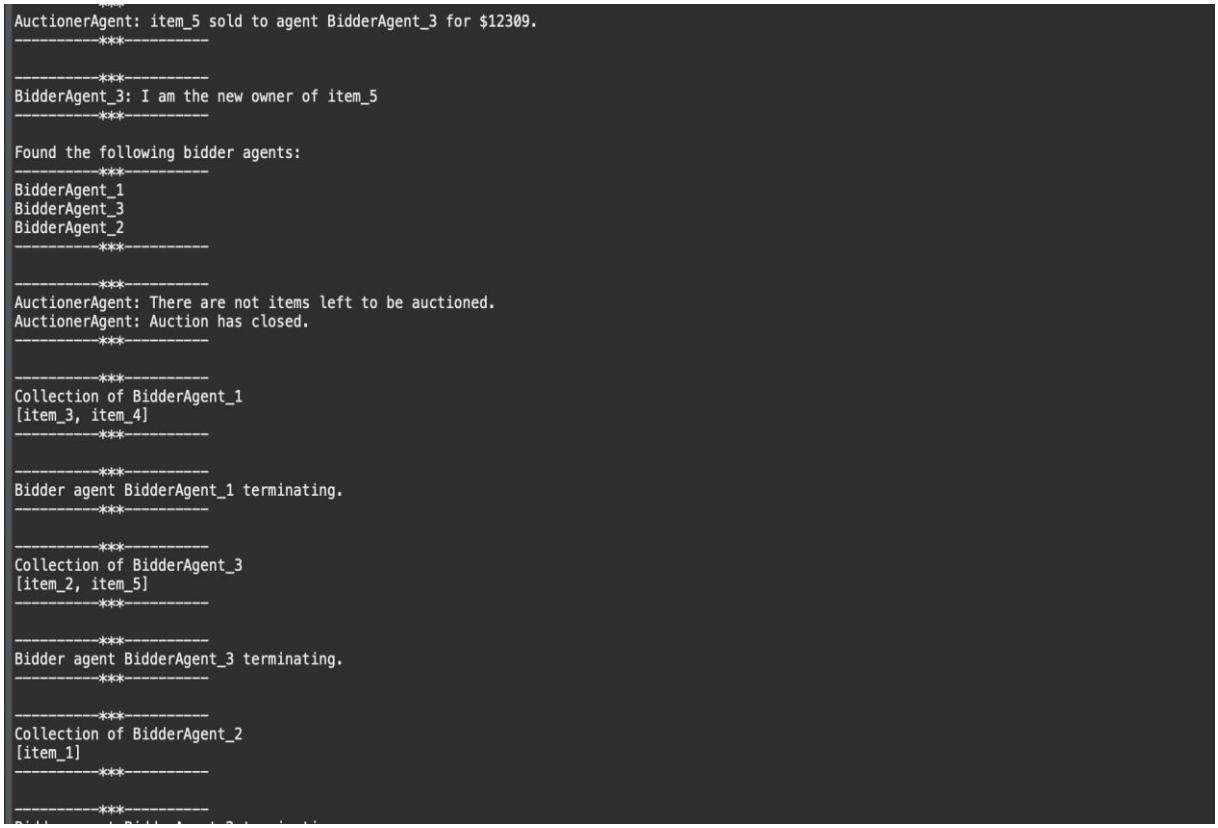
Şekil 4. Alıcı etmen ve davranışları arasındaki ilişkinin UML diyagramı

3. Sistemin Çalıştırılması ve Testi

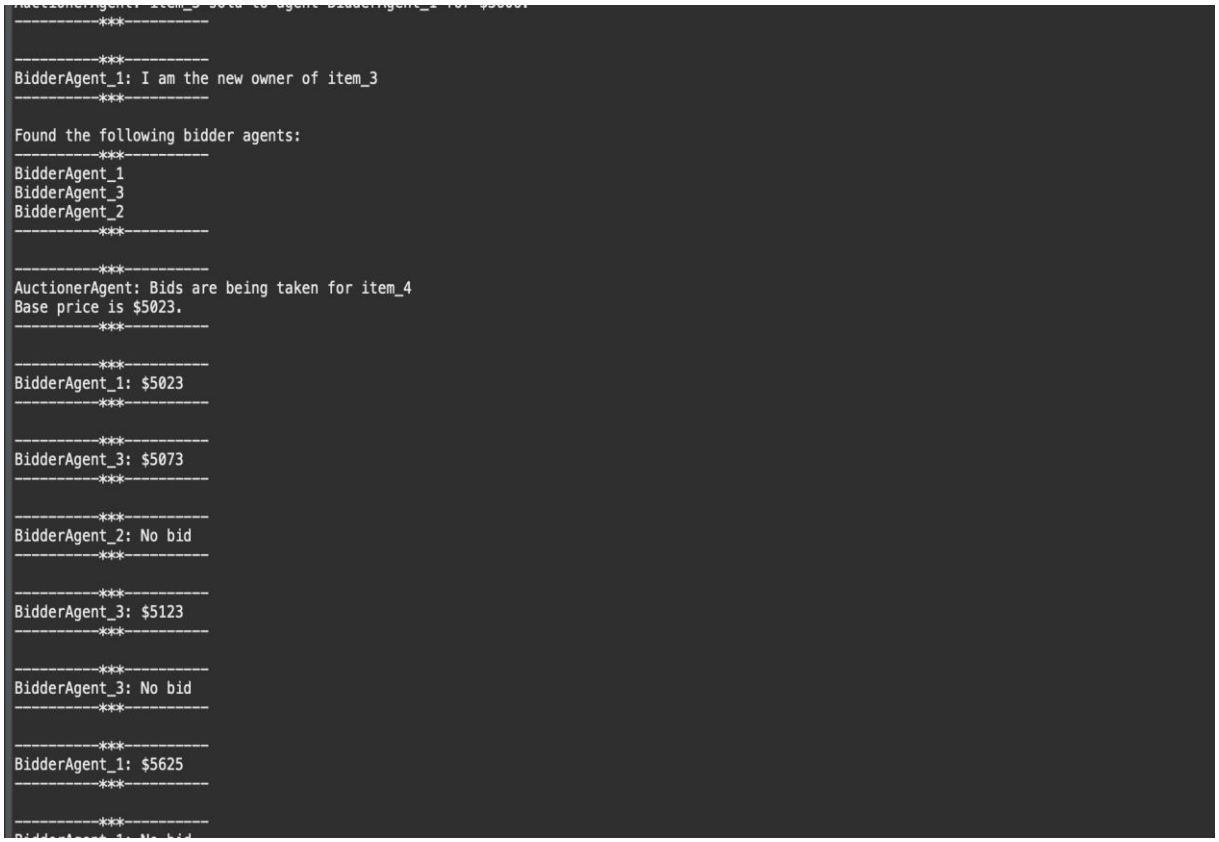
Sistem main.Java fonksiyonunun çalıştırılması ile test edilmiştir. Sistem aynen planlanan doğrultuda çalışmış olup bazı çalışmalarında satıcı etmen ürünü sattıktan sonra tıkanabildiği de olmuştur. Bunun nedeninin etmenlerin asenkron çalıştığı için birbirlerinin işlerini zamansız bir şekilde tetikleme ihtimali ile ilgili olduğu düşünülmektedir. Eğer böyle ise etmenleri belli zamanlar içerisinde uyutarak bu çakışmaların önlenebilmesi mümkün olabilir. Ancak problemin kesin kaynağının bu olduğuna henüz karar verilmemiştir. Yine de yüksek ihtimalle ilerleyen zamanlarda bu problemin üstesinden gelinebileceği düşünülmektedir.

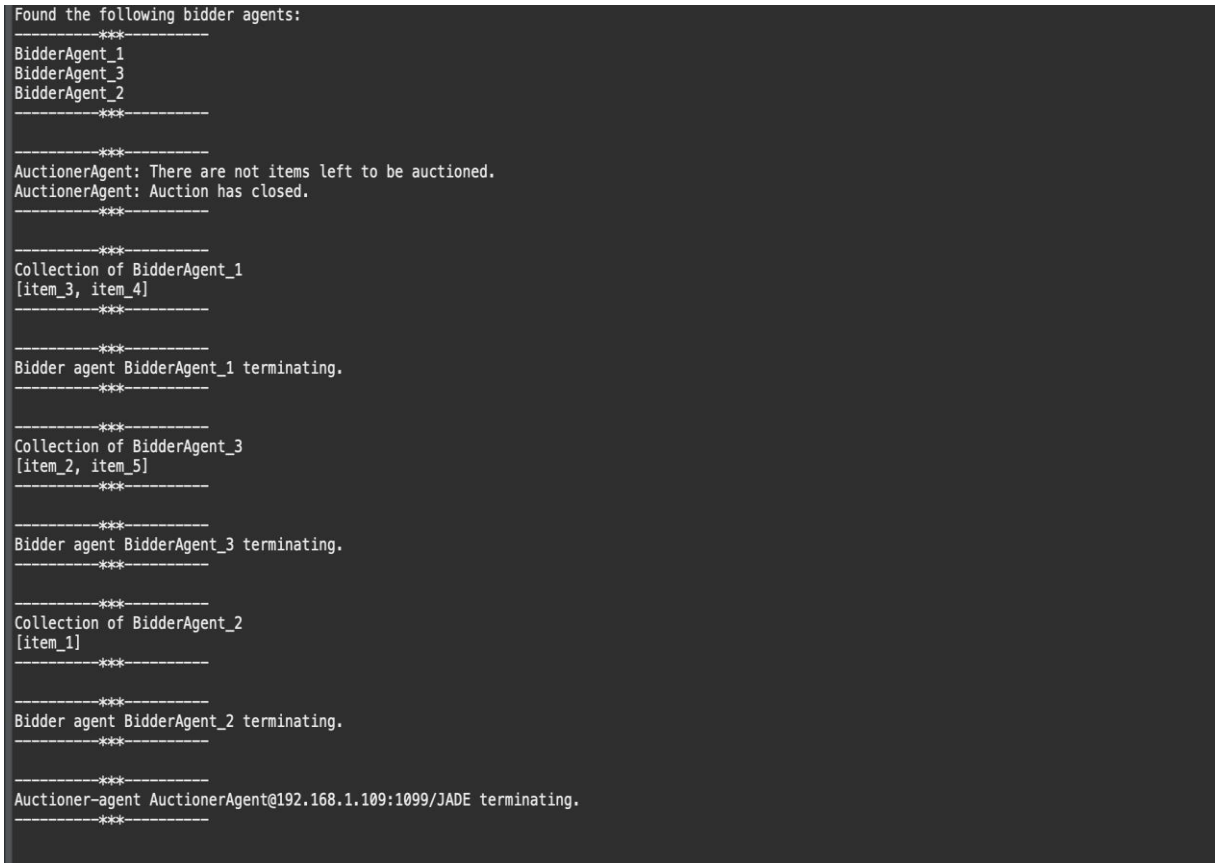


Şekil 5. Test sonuçları



Şekil 6. Test sonuçları





Şekil 7. Test sonuçları

4. Sonuç

Jade çatısı altında bu çok etmenli açık artırma sistemini gerçekleştirmek beklenenden biraz daha kolay olmuştur. Ancak etmenlerin çalışması sırasında bazı sorunlar ile de karşılaşılmıştır. Sniffer çalıştırıldığına da hatalar alınmıştır. Bu yüzden sniffer özelliğinin çalıştırılması da test edilememiştir. Senkronizasyon konusunda daha farklı yaklaşımların veya daha farklı davranış biçimlerinin kullanılması belki de sistemin daha verimli şekilde çalışmasını sağlayabilir. Oluşturulmuş bu sistem üzerinde farklı metot ve yöntemler kullanılarak bu sistemin daha da iyileştirilebileceği ve daha profesyonel amaçlara hizmet edebileceği de düşünülmektedir. Örnek olarak bu sistem eğer profesyonel bir amaç için kullanılacak olursa açık artırma alanında çok daha verimli sonuçlar verebilecek matematiksel bir yapı alıcı etmenler içerisine gömülebilir.

5. Kaynaklar

[1] Bellifemine, F., Poggi, A. and Rimassa, G., "Developing Multi-agent Systems with a FIPA-compliant Agent Framework", Software Practice and Experience, 31: 103-128 (2001).

[2]Alaybegoglu, Aysegul & Kardas, Geylani & Erdur, Cenk & Dikenelli, Oguz. (2007). Design and Implementation of a FIPA Compliant Hotel Reservation Multi-Agent System using SABPO Methodology

[3] Jade.tilab.com. 2022. [online] Available at: <https://jade.tilab.com/doc/tutorials/JADEProgramming-Tutorial-for-beginners.pdf> [Accessed 21 June 2022].

[4] FIPA Standards, http://www.fipa.org/.

[5]Erdur, R. C., "Çok-Etmenli Sistemler", Etmen Tabanlı Yazılım Geliştirme dersi notları, Ege Üniversitesi Bilgisayar Mühendisliği Bölümü, 26 sayfa (2001).

[6]Alzahrani, Saleh & Ayesh, Aladdin. (2008). Regionally Distributed Architecture for Dynamic e-Learning Environment (RDADeLE). 579 - 584. 10.1109/HSI.2008.4581505.