Rol Tabanlı Çoklu Etmen Sistemleri İçin İletişim Mimarisi

Alpaslan Burak Eliaçık[1] Erdem Eser Ekinci[2] Oğuz Dikenelli[3]

[1,2,3] Bilgisayar Mühendisliği Bölümü, Ege Üniversitesi, İzmir

[1]e-posta: aburakeliacik@gmail.com [2] e-posta: erdemeserekinci@gmail.com
[3] e-posta: oguz.dikenelli@ege.edu.tr

Özetçe

Rol kavramının, yazılım mühendisliği çalışmaları çerçevesinde yazılımların kullanıcılarıyla ve birbirleriyle olan ilişkilerini modellemek ve programlamak için kullanılması üzerine literatürde birçok araştırma yer almaktadır. Üstelik bu çalışmalar nesneye yönelik programlamadan ilgiye yönelik programlamaya kadar çeşitli paradigmaları da içine almaktadır. Ancak diğer programlama paradigmalarına kıyasla, çoklu etmen sistemlerinin dağıtık ve işbirlikçi doğası rol kavramının yapısıyla uyum sağlamakta ve çoklu etmen sistemlerinin modellenmesinde ve gerçekleştiriminde uygulanabilirliğini daha elverişli kılmaktadır. Bu çalışmada, çoklu etmen sistemlerinde rollerin programlanabilir birimler olarak kullanılması için FIPA tanımını genişleten rol tabanlı somut bir iletişim mimarisi önerilmiştir. Önerilen iletişim mimarisi sıfırdan başlayıp gerçekleştirmek yerine SEAGENT çoklu etmen sistemi geliştirme çerçevesi üstüne artırım yapılarak gerçekleştirilmiştir.

1. Giriş

Antik çağlarda tiyatro ile ortaya çıkan rol kavramı, sosyologlar tarafından genişletilerek insan organizasyonundaki bilgi kümelerinin incelenip, organizasyondaki bireylerin inanç, arzu, davranışlarını ve birbirleriyle olan ilişkilerini tanımlamak/tahmin etmek amacıyla kullanılan genel bir teoriye dönüştürülmüştür[22]. Bu tanıma paralel olarak rol kavramı yazılım mühendisliği çalışmaları çerçevesinde de yazılımların kullanıcılarıyla ve birbirleriyle olan ilişkilerini tanımlamakta ve programlamakta kullanılması üzerine birçok araştırma yapılmıştır[4,14,18,19,20,21]. Bu araştırmalar gereksinim analizinden gerçekleştirime kadar tüm süreçleri ve değişik programlama paradigmalarını ele almaktadır[16,17]. Günümüzde de rol kavramı yaygın kabul görmüş metodolojilerin (RUP [1],[21]) analiz ve tasarım süreçlerinde aktör ('actor') ya da oyuncu ('player') gibi adlarla yer alarak endüstride kullanım alanı bulmuştur. Ayrıca tasarım deseni olarak nesneye yönelik dillerde uygulanarak gerçekleştirim seviyesinde de yer almaktadır[2].

Öte yandan çoklu etmen sistemlerinin dağıtık ve işbirlikçi doğası rol kavramının yapısıyla uyum sağlamakta ve çoklu etmen sistemlerinin modellenmesinde ve gerçekleştiriminde uygulanabilirliğini diğer programlama paradigmalarına kıyasla daha elverişli kılmaktadır. Zaten birçok önemli ÇES metodolojilerinin analiz ve tasarım safhalarında etmen organizasyonlarının modellenmesinde roller kullanılmaktadır [23,24,25,26,27]. Çoklu-etmen organizasyonlarının rollerle modellenmesi bir yana, rollerin programlanabilir yazılım birimlerine dönüştürülmesi bu metodolojilerin gerçekleştirimle doğrudan bağlantı sağlayacağından ÇES tabanlı yazılım geliştirim süreçlerine büyük bir ivme kazandıracaktır. Ancak rollerin gerçekleştirilebilir yazılım birimine dönüştürülmesi için temel özelliklerinin belirlenmesi gerekmektedir. Kristensen[11] nesne tabanlı sistemler için ayrıntılarıyla rollerin temel özelliklerini tanımlamıştır. Kristensen'in ardından, Depke[10] bu özellikleri çoklu etmen sistemlerini modellemek için tekrardan yorumlamıştır. Biz Depke'nin tanımladığı rol özellikleri ile aynı fikirde olup bu özellikleri rolün gerçekleştirilebilir yazılım birimi olarak kullanılması için yeniden tanımladık. Bu özelliklere kısaca değinmek gerekirse: *Görünürlük*, rol ve rolün alt bileşenleri olan kural, hedef, plan, protokol doğrudan programlanabilir olmalıdır. *Bağımlılık*, rol etmen tarafından ilklendirilmesi ve etmen tarafından kapsanmalıdır. *Kimlik*, işbirliği ve koordinasyon sırasında iletişimi gerçekleştirebilmek için tarafların benzersiz bir tanımlayıcı ile iletişimi desteklemelidir. *Devingenlik*, etmenin yaşam döngüsü sırasında yeni rolleri ilklendirebildiği veya ilklendirdiği rolleri sonlandırabildiği bir ortamda ÇES mimarilerinin bu roller arasında gerçekleşen iletişimi sağlamalıdır. *Çokluk*, etmen birden fazla rolü aynı zamanda ilklendirebilmeli ve bu ilklendirdiği roller arasında iletişimi desteklemelidir.

Ancak rollerin yalnızca ÇES'lerin modellenmesinde değil, aynı zamanda programlanmasında da kullanılması için Depke'nin tanımladığı özelliklerden yola çıkarak etmen mimarileri genişletilmelidir. Biz bu desteğin sağlanabilmesi için yani roller programlanabilir elemanlar olarak ÇES'lerde kullanılabilmesi için 4 temel mimari gereksinimi belirledik. Bu gereksinimler şunlardır:

- **Rol Örneği:** Rollerin programlanabilir olması, aynı zamanda ilklendirilmesi ve tanımlandığı etmen ile birlikte organizasyonda tekil bir eleman oluşturması gerekmektedir. Rol örneği olarak adlandırdığımız bu eleman ÇES çerçevesi içerisinde doğrudan desteklenmelidir.
- **Programlanabilir Protokol:** Rollerle birlikte kural, hedef, plan ve protokol gibi alt bileşenleri de açık bir şekilde programlanabilir ve gözlenebilir olmalıdır. Özellikle iletişim bakış açısından bakıldığında rollerin birbiriyle olan ilişkilerini tanımlayan protokollerin, diğer unsurlardan bağımsız programlanabilir olması gerekmektedir.
- **İşletilebilir Protokol:** ÇES iletişim altyapısı rol örnekleri arasında koordinasyonu sağlamak için açık bir şekilde programlanmış olan protokolleri çalıştırılabilir olmalıdır.

[1] Rational Unified Process, http://www-01.ibm.com/software/awdtools/rup/

[2]Dealing with Roles, Martin Fowler, http://www.martinfowler.com/apsupp/roles.pdf

- **Rol Yönetim Servisi:** Organizasyonda tekil birer eleman olarak ilklendirilen rol örnekleri ÇES altyapısı tarafından yönetilmeli etmenlerle ilişkileri kayıt altında tutulmalıdır.

Bu bildiride ÇES çerçevelerinde rollerin desteklenebilmesi için tanımlanan bu 4 temel gereksinimden yola çıkarak SEAGENT çoklu etmen sistemi geliştirme çerçevesinin[1] rollerin desteklenmesi için hazırlanan iletişim altyapısına yer verilmektedir. Bildirinin geri kalan bölümleri şöyle sıralanmıştır: Bölüm 2, rol kavramını iletişim mimarisinde desteklemek için FIPA şartnamesinde belirlenmiş olan iletişim mimarisinin üstüne yapılmış artırımları ayrıntılı olarak anlatılmaktadır. Bölüm 3 tanımlanmış olan artırımlar sonucundan oluşan iletişim mimarisi gerçekleştiriminin ayrıntılarını anlatmaktadır ve bu bölüm 3'ün alt bölümlerinde ise ilk olarak 3.1. bölüm protokol ontolojisini ve protokol ontolojisinin SEAGENT da kullanılan HTN planlama paradigmasına nasıl tümleştirildiğini anlatmaktadır. Takip eden alt bölüm protokollerin iletişimsel HTN planları ile birlikte indirgemesini açıklamaktadır. En son alt bölüm de ise, iletişim esnasındaki farklı rol örnekleri arasında gerçekleşen bir iletişim sürecini anlatmaktadır. Ayrıca bu bölümlerden sonra önerilen iletişim mimarisinin çalışma esaslarını bir Takas uygulaması örneği üzerinden genel bir bakış verdikten sonra en son olarak çalışmayı sonuçlandırmadan, diğer rol tabanlı iletişim mimarilerini ele alınacaktır.

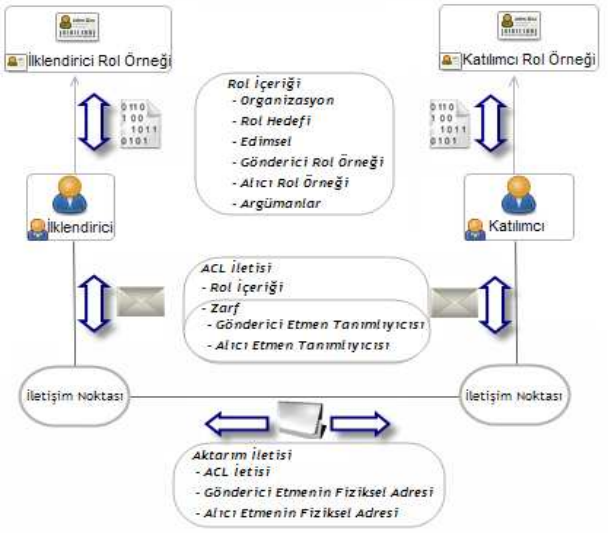
2. Rol Kavramının FIPA Tabanlı İletişim Yaklaşımında Desteklenmesi

Çoklu etmen sistemleri alanında, istenilen ÇES ortamı etmenlerin herhangi bir zamanda sisteme katılıp, ayrılabileceği açık organizasyonlardır. Bu açıklık etmen sistemlerinde heterojenliğe sebep olur. FIPA, bu heterojenlik yüzünden farklı etmen çerçevelerinin birbirleriyle iletişimi sağlamak amacıyla doğmuş bir kuruluştur. FIPA'nın çalışmaları etmen çerçeveleri için soyut bir mimari şablonu sunmaktadır. Bu bağlamda rollerin programlanabilir bir eleman olarak iletişim altyapısına dahil olmasıyla FIPA'nın tanımladığı soyut mimarisi genişletilmesi gerekir. Bu etkilerin rollerin iletişimde kullanılması için tanımlanan *Rol Örneği*, *Programlanabilir Protokol*, *İşletilebilir Protokol* ve *Rol Yönetim Servisi* gereksinimlerini karşılaması gerekir.

Bu bölümde rol tabanlı iletişimi desteklemek için, Rol Örneği, Programlanabilir Protokol ve İşletilebilir Protokol gereksinimlerini karşılayan artırımların etmen iletişim altyapısı ve FIPA etmen iletişim dili tanımı üzerinde yaptığı değişiklikler tanımlanmıştır. Bundan sonra rol örneklerini ve etmenlerin eşleşmelerini yönetmekten sorumlu Rol Yönetim Servisi (RYS) olarak adlandırdığımız yeni bir sistem servisi ortaya çıkarılmıştır. Altyapıda etmen eşleşmelerini ve rol örneklerini yönetebilen RYS'ye benzer servis yaklaşımları [6, 15] da tanımlanmıştır. Bu araştırmada bizim hedefimiz RYS'nin sistemdeki işlevselliğini açıklamak yerine, RYS'nin rol tabanlı iletişim sırasında bizim önerdiğimiz iletişim mimarisi ile olan bağımlılığını tanımlamaktır.

FIPA Etmen İleti Taşıma Servisi[3] etmenleri etkileştirmek için iki katmanlı bir iletişim altyapısı önermektedir; Etmen Katmanı ve İletişim Katmanı. Etmen katmanı iletişimde etmen seviyesindeki bilgilerin iletişim için kullanılması sağlarken, İletişim seviyesi ise en son fiziksel olarak gerekli olan iletilerin adreslenmesini ve bu adreslenmiş iletileri alıcılarına iletilmesini sağlar. Ama rol tabanlı etmen organizasyonlarında, bu iki katmanlı iletişim mimarisi aktif rol örnekleri arasında gerçekleşen rol tabanlı iletişim desteğini sağlamak için yeterli değildir. Bu nedenle, rol desteğini sağlamak için Rol Katmanını Etmen Katmanı üzerine eklenmiştir. Önerdiğimiz iletişim altyapısı Şekil 1'de gösterilmektedir.

İletişim mimarisinin de, iletişim kuran rol örnekleri arasında iletim için bilgi sağlamaktan sorumlu olan Rol Katmanı iletişim altyapısının üst tarafında yer alır. Rol Katmanı hedef, plan ve protokol gibi iletişim için gerekli olan bilgileri etmen iç mimarisinden elde eder. Önerdiğimiz iletişim altyapısında, Etmen Katmanı mantıksal etmen adreslerinin tanımlanmasından sorumludur. Bu etmen tanımları her organizasyon için özel olan RYS'den talep edilir. Taşıma Katmanı iletinin en dış paketine iletişime geçen etmenlerin fiziksel adreslerini eklemekten sorumlu olan önerdiğimiz iletişim mimarisinin en alt katmanıdır. Bu katman etmen fiziksel adreslerini, işlevselliği FIPA Etmen Yönetim Servis şartnamesinde[4] tanımlanan Etmen Yönetim Servisinden sağlar. İleti hazırlama süreci Bölüm 3-3'de detaylı olarak açıklanacaktır.



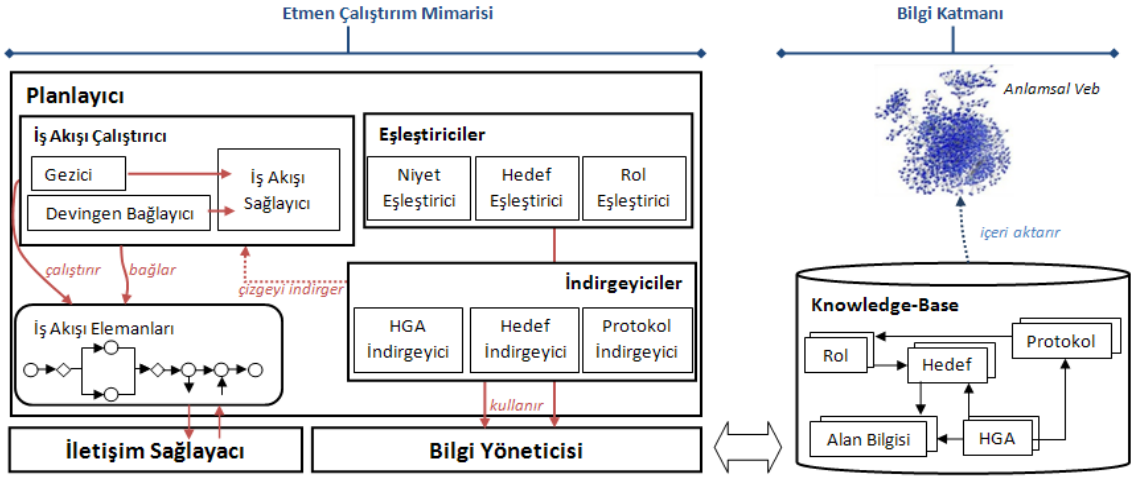
Şekil 1: Üç Katmanlı İletişim Mimarisi

Bu önerilen altyapı internet aracılığıyla etmenler arasında bilgi aktarımını sağlamak için kullanılan bir ileti üretir. Bu ileti iç içe oluşan üç paketten oluşur. Bu mesajın paketleri içten dışa doğru sırasıyla Rol İçeriği, ACL İletisi ve Aktarım İletisinden oluşur. Ayrıca bu paketler FIPA Soyut Mimari Şartnamesinde[5] de İçerik, ACL İletisi ve Aktarım İletisi olarak tanımlanmaktadır. Rol tabanlı iletişimi desteklemek için bu paketlerin dahili bilgilerini yeniden düzenlenmiştir.

[3] Fipa Agent Message Transport Service Specification, http://www.fipa.org/specs/fipa00067/SC00067F.pdf

[4] Fipa Agent Management Service, http://www.fipa.org/specs/fipa00023/SC00023K.pdf

[5] Fipa Abstract Architecture Specification, http://www.fipa.org/specs/fipa00001/SC00001L.pdf



Şekil 2: Protokol Eklentisi İle Birlikte Etmen İç Mimarisi

Rol İçerik Paketi Aktarım iletisinin en iç seviyesini oluşturur ve bu seviye rol katmanı tarafından hazırlanır. Rol İçeriği, hangi rol hedefinin çalıştırılmak istenmesi niyetini niteleyen edimseli, gönderici ve alıcı rol örneklerini, alıcının çalıştırılmak istenilen rol hedefini, rollerin ait olduğu organizasyonu ve rol hedefine ait argümanları içerir. İleti formatımızda ACL iletisi edimsel'i içermez. Edimsel'in paket seviyesini ACL iletisinden rol içeriğine değiştirilmiştir, çünkü protokolü görünebilirliğini sağlamak için sistemde ilklendirilip çalıştırılması gerekir. Protokol plan ile ilişkili olduğu için plan ile aynı seviyede değerlendirilmelidir. Bu nedenle Edimsel protokolün bir özelliği olduğundan rol seviyesinde yorumlanmıştır. Edimsel özelliğinin paket değişiminin dışında, başka bir değişiklik ACL iletisi üzerinde gerçekleştirilmemiştir. İletişim mimarisinde ileti biçiminde en dış paket olan Aktarım İletisini aktif etmenlerin birbirleri ile iletişim kurmaları için kullanılmaktadır. Önerdiğimiz altyapıda FIPA'da kullanılan Aktarım İletisi üzerinde herhangi bir değişiklik yapılmadan aynı anlamda kullanılmıştır. İletişim altyapısıyla hazırlanan Aktarım iletisine ait bir örnek işaretleme dili gösterimi Şekil 3'de belirtilmektedir.

```
<Transport Message>
  <Sender Agent Physical Address> udp://155.223.24.108 </Sender Agent Physical Address>
  <Receiver Agent Physical Address> udp://155.223.24.119 </Receiver Agent Physical Address>
  <ACL Message>
    <Envelope>
      <Sender Agent Identifier> Agent1@seagent.ege.edu.tr </Sender Agent Identifier>
      <Receiver Agent Identifier> Agent2@seagent.ege.edu.tr </Receiver Agent Identifier>
    </Envelope>
    <Role Content>
      <Organization> Barter </Organization>
      <Role Goal> RecordStock </Role Goal>
      <Performative> request </Performative>
      <Sender Role Instance> Customer </Sender Role Instance>
      <Receiver Role Instance> Barter </Receiver Role Instance>
      <Arguments> ..... </Arguments>
    </Role Content>
  </ACL Message>
</Transport Message>
```

Şekil 3: Aktarım İletisi Örneği

Bu bölümün ardından, FIPA iletişim mimarisini genişleterek geliştirdiğimiz iletişim altyapısını bir çoklu etmen sistemine nasıl tümleştirdiğimiz detaylıca anlatılacaktır.

3. İletişim Mimarisinin Gerçekleştirilmesi

Rol kavramını programlanabilir eleman olarak iletişim seviyesinde desteklemesi için gerekli olan *Rol Örneği*, *Programlanabilir Protokol*, *İşletilebilir Protokol* ve *Rol Yönetim Servisi* artırımlarının SEAGENT çoklu etmen sistemine uygulandığında altyapıda mimarisel değişiklikler gerçekleşmiştir.

SEAGENT çoklu etmen sisteminin[1] temel özellikleri anlamsal veb desteği ve iş akışı üzerinde HGA tabanlı plan çalıştırmasıdır. SEAGENT planlayıcısı[2] anlamsal tanımlı (OWL [9] ontolojisi ile) bileşenleri kullanarak bu özellikler desteklenir. SEAGENT çalışma-zamanı yapıları OWL biçimlendirme dili kullanarak modellenmektedir ve bu modelin örnekleri etmenin depolama bileşeni olan Bilgi tabanında depolanmaktadır. Hedef rollerin ana arzularının bilgilerini sağlar. Alan bilgisi organizasyon ile ilgili ontolojileri tutar. HGA modeli Sycara ve arkadaşlarının[13] tanıttığı HGA yapısına benzemektedir. HGA gösteriminde önceden tanımlanmış bir hedefi gerçekleştirmek için oluşan planlar iki tip görevden oluşur: Davranış olarak adlandırılan bileşik görevler ve Eylem olarak tanımlanan basit görevler. Eylemler doğrudan çalıştırılabilir elemanlardır.

Etmen bir diğer etmenlerin istekleri veya etmenin dahili niyeti tarafından üretilen amaçları gerçekleştirmeye çalıştığında, Eşleyici bileşeni amaca uygun hedefi bulabilmek için rol örneğinin bilgi tabanındaki hedef ontolojilerini sorgular. Hedefe karar verildikten bir süre sonra, Hedef İndirgeyicisi seçilen hedef ontolojisini çalışma zamanında iş akışı elemanlarından oluşan etmen eylemlerini temsil etmek için kullanılan iş akışına indirger. İş akışı çalışması eşleşen hedefi gerçekleştirmek için gerekli olan uygun planı bulmak amacıyla Eşleyiciyi tekrardan tetikler. HGA İndirgeyicisi eşleşen planı iş akışı formuna indirger ve bu iş akışı formu İş Akışı Çalıştırıcısının Devingen Bağlayıcı bileşeni tarafından çizgeye bağlanır. SEAGENT iş akışı mimarisi somut çizgeyi çalışma zamanında yükleyen sonradan bağlama yeteneğini[8] destekler. Bu yetenek sayesinde bizim iş akışımız devingen olarak genişleyebilir hale gelir.

İndirgeme sürecinden sonra, çizgenin çalıştırılma safhası başlar. Planlayıcı her çizge için yeni bir Gezici yaratır. Daha sonra rol hedeflerini gerçekleştirmek için Gezici tarafından bu üretilmiş olan çizge çalıştırılır.

Yukarıda bahsedilen SEAGENT mimarisine, rol kavramının programlanabilir bir eleman olarak eklenebilmesi için gerekli olan arıtırımlar SEAGENT platformuna uygulandığında ilk olarak rol örneğinin altyapıda ilklendirilmesi gereklidir. Rolün

sistemde ilklendirilmesi için öncelikle rol kavramı sistemde tanımlanmalıdır. Rol ve rolü oluşturan alt bileşenler SEAGENT platformunda ontolojiler yardımıyla programlanır. Bu bağlamda etmen bir rolü ilklendirmeye başladığı zaman, role özelleşmiş ontolojilerin yardımıyla tanımlanmış bilgi kümesi ile rol örneği yaratılır. Etmenin yaşam döngüsünde rolün iletişim dahil bütün sorumlulukları rol örneği tarafından gerçekleştirilir. Rol kavramının alt bileşenlerinden biri olan protokolün iletişim sırasında görünebilirliğini sağlamak için önceki bölümde söz edildiği gibi protokol kavramı planlayıcıya bir çalışma-zamanı girdisi olarak eklenmelidir ve altyapı bu kavramı yorumlayıp çalıştırabilmelidir. Bu nedenle planlayıcı altyapısına protokolünün de çalışma-zamanı girdisi olması için yeni iş akışı bazı artırımlar yapılmıştır. Artırımları yapılmış SEAGENT planlayıcı mimarisi ana bileşenlerinin iletişim bakış açısından Şekil 2'de temsil edilmiştir. Bu artırımlar, SEAGENT mimarisinde çalışma-zamanı girdilerinin sistemde kullanıldığı üç aşama olan sistemde nasıl saklandığını, indirgeme sürecinde nasıl yorumlandığını ve çalışma zamanında nasıl çalıştırıldığını protokol kavramı için sağlamaktadır. En son artırım olan rollerin yönetilmesi özelliği ise SEAGENT platformunda RMS olarak adlandırılan bir merkezi sistem servisinin eklenmesiyle gerçekleştirilmiştir. Böylelikle rollerin organizasyon seviyesinde etmenler ile olan ilişkileri yönetilmeye başlanmıştır.

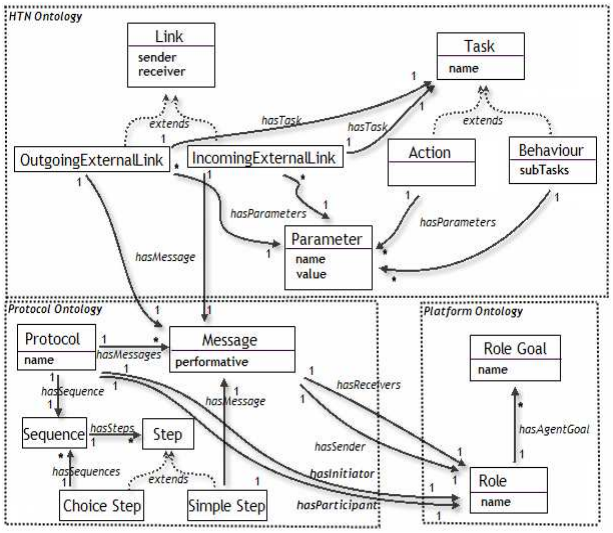
Alt bölümlerde bu artırımların detayları anlatılacaktır. Protokolün SEAGENT altyapısına uyum sağlaması için ontolojik olarak saklanması ve bu tanımlanmış olan ontolojinin diğer ontolojiler olan ilişkileri Bölüm 3-1'de, tanımlanmış protokol ontolojisinin iş akış diyagramını nasıl indirgeneceğinin detaylıca anlatımı Bölüm 3-2'de, açılan protokol desteği sağlanmış iş akışının çalışma mantığı ve rollerin bu süreçte RMS ile olan ilişkileri ise Bölüm 3-3'de açıklanacaktır.

3.1. Protokol Ontolojisi ve Bağımlılıkları

Rol örnekleri arasında etkileşimi yöneten resmi kurallar dizisine protokol denir. Protokol standartları sayesinde, etkileşime katılan rol örnekleri hangi koşullar altında nasıl ileti gönderilip alınacağı üzerinde anlaşırlar, yani kısaca protokol rol örnekleri arasındaki etkileşimin eş zamanlılığını sağlar.

Protokollerin sistemde programlanabilmesi için sistemde bir şekilde temsil edilmelidir. SEAGENT mimarisinde protokoller ontolojiler ile tanımlanan bir şema yardımıyla tanımlanmıştır. Tanımlanan protokol şeması diğer ontolojiler ile olan bağımlılıkları Şekil 4'de temsil edilmiştir. Bizim önerdiğimiz iletişim altyapısındaki bütün iletişim protokolleri bu protokol şemasının bir örneğidir.

Bu protokol şemasında, protokol kavramlarını iki gruba ayrılmıştır; Denetim Kavramları ve Temel Kavramları. Denetim yapıları dizge, adım, seçimsel adım ve basit adım 'dan oluşmaktadır. Dizge, protokolün bir safhası ve diğer denetim yapılarının üst kavramı olan adımın sıralı bir listesidir. Seçimsel adım ve basit adım gibi denetim yapıları adım kavramından türetilmiştir. Seçimsel adım dallanmış protokol adımlarını tanımlar ve içerisinde her dal için bir dizge tutar. Basit adım rol örnekleri arasında gerçekleşen mesajlaşma işlemini tanımlar.



Şekil 4: Protokol Ontolojisi ve Bağımlılıkları

Temel kavramlar, protokol ve ileti kavramlarından oluşur. Ontolojideki protokol kavramı etkileşim hakkında genel bilgi sağlar. Ayrıca protokol kavramı etkileşim protokolündeki ilklendirici, katılımcıları ve protokol dizgesini tanımlar. İleti rol örnekleri arasında aktarılan protokol iletilerini tanımlar. İleti gönderici ve alıcı tarafın yanı sıra alıcı tarafın göndericinin isteğini nasıl gerçekleştireceğini belirten gönderici niyetini de kapsar.

Önerilen protokol ontolojisini SEAGENT'daki diğer yönetim ontolojileri ile birlikte yorumlanması için, protokol ontolojilerin diğer ontolojiler ile arasında olan ilişkileri tanımlanmalıdır. Protokol ontolojisinde, rol protokolün bir katılımcısını veya protokol iletisinin bir tarafını niteler, bu nedenle rol kavramı protokolü tanımlamak için kullanılmalıdır. Rol kavramı SEAGENT 'da organizasyon hakkında bilgi veren Platform ontolojisinde tanımlanmaktadır. Bu nedenle protokolü tanımlamak için gerekli olan rol kavramı platform ontolojisinden elde edilir, ama rol kavramı protokol ontolojisinde örneklendirilirken sadece protokolün ilklendirici veya katılımcı olarak adlandırılan tarafları tanımlamak için kullanılır. Çalışma zamanında ise bu tanımlamalar gerçek rol örnekleri ile eşlenerek çalıştırılır. Bir diğer yönetim ontolojisi olan HGA ontolojisi[6] ise etkileşimi sağlamak için gönderilen ve alınan argümanlardan sorumlu davranışlara sahiptir, bu davranışların protokol ontolojisi ile tümleştirmek için HGA ontolojisinde dışsal bağlantılar tanımlanmıştır. Her dışsal bağlantı bir protokol iletisine referans içerir. Bu referans sayesinde HGA'nin etkileşimsel davranışlarının etkileşim sırasında nasıl davranacağı tanımlanır.

Sonuç olarak, planlayıcı altyapımızda birlikte yorumlamak ve çalıştırmak için HGA ontolojisi ve Platform ontolojisi ile tümleşik bir protokol ontolojisi tanımlanmıştır. Bu protokol gerçekleştirimi sayesinde, sadece iletişimi modellemek için kullanılan protokollerin yerine protokoller gerçekleştirimde somut bir eleman olarak görünür kılınmıştır.

[6] http://seagent.ege.edu.tr/etmen/ontologies/HTNOntology-2.1.4.owl

3.2. HGA Davranışları ve Protokol Akışları ile İş Akışının Oluşturulması

SEAGENT çerçevesinin rol-tabanlı bir iletişim gerçekleştirebilmesi için protokol çalıştırılabilir bir girdi olarak kullanılmalıdır. SEAGENT planlayıcı mimarisi ontolojilerden üretilmiş iş akışı diyagramlarını çalıştırır. Bu bağlamda protokol kavramını çalıştırılabilir bir girdi olarak SEAGENT'da kullanılması için bir önceki bölümde tanımlanan protokol ontolojisinin örneklerini iş akışı elemanlarına çevrilmesi gerekir. Protokol kavramlarını niteleyen iş akışı elemanların eklenmesiyle, iş akışı diyagramı HGA düğümleri, protokol düğümleri, düzenleyici düğümleri, HGA akışları ve protokol akışlarından oluşur. İş akışı diyagramının ontolojilerden nasıl oluştuğunu daha açıkça anlatabilmek için, indirgeme algoritması detaylıca açıklanmalıdır. Rol hedefini yaratmak için kullandığımız indirgeme algoritmasının bir kısmı Şekil 5'de tanımlanmıştır.

```
function İNDİRGE (davranış) returns bir İş Akışı Çizgesi
    iaç ← ÇİZGEYİ-OLUŞTUR(davranış)
    iaç ← HEPSİNİ-EKLE(ALT-GÖREVLERİ-OLUŞTUR(davranış), iaç)
    iaç ← HEPSİNİ-EKLE (İLETİŞİM-DÜĞÜMLERİNİ-OLUŞTUR(davranış), iaç)
    iaç ← HTN-AKIŞLARINI-OLUŞTUR(iaç, davranış)
    iaç ← İLETİŞİM-AKIŞLARINI-OLUŞTUR(iaç, davranış)
    iaç ← İLETİŞİM-DÜĞÜMLERİNİ-BAĞLA(iaç, davranış)
    iaç ← DÜZENLEYİCİ-DÜĞÜMLERİ-YERLEŞTİR(iaç)
    return iaç
```

Şekil 5: İndirgeme Algoritması

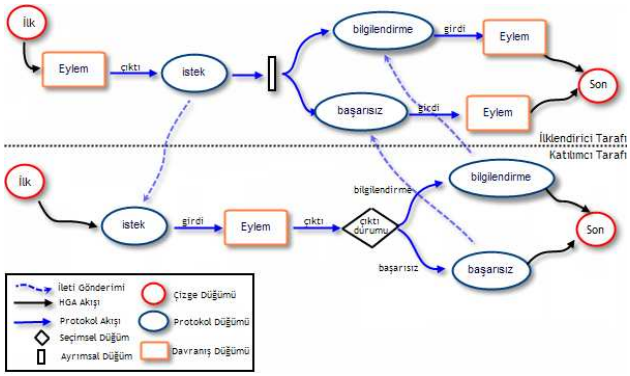
İlk önce, HGA indirgeyicisi ilk düğüm ve son düğüme sahip yeni bir iş akışı yaratarak indirgeme sürecini başlatır. İlk düğüm iş akışı diyagramının çalışmaya başlamasından sorumlu iş akışı diyagramının ilk düğümüdür, son düğüm ise iş akışının ne zaman sonlanacağın belirlemekten sorumlu iş akışı diyagramının son düğümüdür. Hemen akabinde, HGA indirgeyicisi her HGA eylemi için bir HGA düğümü oluşturur ya da her HGA davranışı için bir davranış modeli ve yaratılan davranış modeline ait basit HGA düğümlerini oluşturur. Daha sonra, HGA indirgeyicisi iletişim etkinliklerini belirtmek için dışsal bağlantıları yorumlayarak protokol düğümlerini yaratır. Bu adım indirgeme algoritmamızda HGA davranışları ile protokol akışını tümleştirdiği için iletişim bakış açısından önerdiğimiz algoritmamızda önemli bir yer tutmaktadır. Bu nedenle indirgeme algoritmasının daha iyi anlaşılması için bu adım örnek üzerinden anlatılmalıdır. Bir outgoingExternalLink örneği Şekil 6'da gösterilmektedir.

```
<rdf:Description rdf:nodeID="A2">
  <j.0:hasMessage rdf:resource="Protocol.owl#indv_message_10"/>
  <j.0:hasArgument rdf:resource="HTNOntology.owl#indv_argument_1"/>
  <j.0:hasParameter rdf:nodeID="A1"/>
  <j.0:hasTask rdf:resource="BHSquare.owl#ACSendMessage"/>
  <rdf:type rdf:resource="HTNOntology-2.1.4.owl#OutgoingExternalLink"/>
</rdf:Description>
```

Şekil 6 : Bir Dış Bağlantı Örneği

İlk önce HGA indirgeyicisi A2 dışsal bağlantı örneğini yorumlar. Sonra A2 'nin tipine bakılarak bir gönderici protokol düğümü yaratır. Eğer bağlantının tipi incomingExternalLink ise indirgeyici alıcı protokol düğümü yaratır. Bir sonra protokol indirgeyicisi gönderici protokol düğümünün sorumluluğunu atamak için indv_message_10' un edimselini protokol ontolojisinden elde eder. Bu sayede çalışma sırasında, gönderici protokol düğümü rol içeriğini bu edimsele göre hazırlar.

Yukarıda anlatıldığı şekilde yaratılan iş akışı düğümlerinden sonra, HGA indirgeyicisi HGA akışlarını oluşturup yaratılmış HGA düğümlerine bağlamak için bu akışları kullanır. Sonra HGA indirgeyicisi her dışsal bağlantı için bir protokol akışı oluşturur ve bu oluşturulmuş akışları protokol düğümleri ile HGA düğümlerini bağlamakta kullanır. Daha sonra protokol indirgeyicisi protokol ontolojisini yorumlar ve bağlı olmayan protokol düğümleri arasında bağlantıyı sağlamak için protokol akışlarını oluşturur. En sonunda indirgeme süreci, iş akış diyagramının çalışmasını denetlemek için iş akışı diyagramında gerekli görülen yerlere düzenleyici düğümlerini yerleştirir. Bu indirgeme sürecinin çıktısının bir örneği Şekil 7'de gösterilmiştir.



Şekil 7: İndirgenmiş İş Akışı

Sonuç olarak, indirgeme sürecinden sonra, çalıştırılabilir iş akışı diyagramı HGA ve protokol ontolojisi dahilinde elde edilir. Çalıştırılabilir iş akış diyagramı geliştiricinin isteklerini karşılayan iş düğümleri, işbirlikçi davranışlar arasındaki iletişimi destekleyen protokol düğümleri ve verinin aktarımı ve iş akışının denetimi için gerekli olan akışlardan oluşur.

3.3. Protokol Desteği Sağlanan İş Akışının Çalıştırılması

İndirgeme aşamasından sonra, SEAGENT planlayıcısı sistem hedefini gerçekleştirmek için indirgenen iş akışını çalıştırır. Çalışma safhası süresinde, SEAGENT'da etmenler rol örneği üzerinden haberleştikleri için FIPA şartnamesindeki tanımı genişleten rol tabanlı aktarım iletisini kullanır. Rol tabanlı iletişim iletisi iç içe üç katmandan oluşur. Bu oluşan ileti etkileşim sırasında sistemde rol örneğini görünebilir yapmak için önerdiğimiz üç katmanlı iletişim mimarisi ile özdevinimli olarak hazırlanır ve yorumlanır.

SEAGENT'da ortak bir sistem hedefi en az iki işbirlikçi etmen arasında gerçekleşir. Ortak hedefi gerçekleştirirken işbirlikçi etmenler plan çıktılarını üç katmanlı iletişim mimarisi yardımıyla birbirlerine iletirler. Bu süreçte etmenler iletişim sırasında; ilklendirici rolün tanımlayıcısını, katılımcı rollerin tanımlayıcılarını, etkileşimin edimselini, rollerin ait olduğu organizasyonu ve rol hedefini gibi bazı bilgilere ihtiyaç duyar ve bu bilgileri bir yerden toplaması gerekir. Bu bilgiler SEAGENT'da SEAGENT İletişim Oturumu(SİO) olarak adlandırılan bir oturum tarafından elde edilir. Bu bağlamda SEAEGENT'da iletişim sırasında rol seviyesinde, rol içeriğinin hazırlanması ve yorumlanması için SİO'nun tuttuğu sohbet bilgileri kullanılır. Buna benzer oturum yaklaşımları[3] literatürde de gözlenmektedir.

Oturum süreci herkesin bir ana hedef üzerinde anlaşmasıyla başlar. Bu nedenle yukarıda bahsedilen bir oturumun açılması için iletişime geçecek etmenler ve rolü programlanabilir bir eleman olarak sisteme eklenmesi için rol kavramının diğer sistem elemanları ile olan ilişkilerini yöneten RYS arasında bazı uzlaşma süreçleri gerçekleşmelidir. İletişimin başlaması için gerekli olan süreçlerin başında ilk önce, sohbetin ilklendirici tarafı olan rol örneği katılımcı rol örneklerinin tanımlayıcılarını bazı kriterlere göre RYS'den elde eder. Daha sonra, ilklendirici rol örneği ve katılımcı örnekleri ilklendiricinin istediği organizasyon hedefi üzerinde anlaşır. Bu süreçlerden sonra iletişim rol örnekleri arasında başlar.

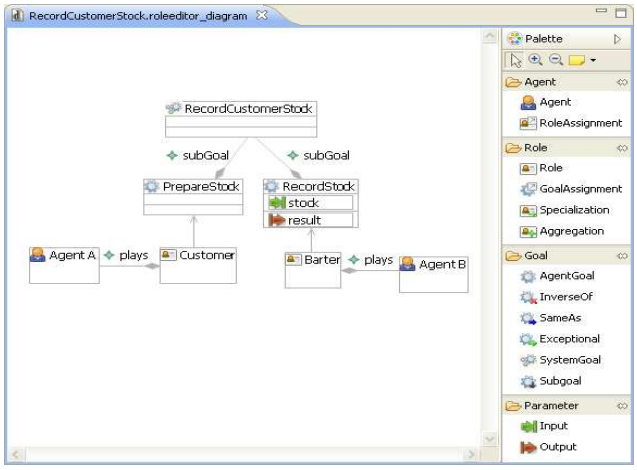
İletişim sırasında etmenler birbirleri arasında argüman aktarımı gerçekleştirirler. Argüman aktarımı sırasında, ilk önce dışarıya parametre göndermek isteyen iş akış diyagramı düğümü bir çıktı üretir ve bu üretilen çıktıyı bir protokol akışı ile gönderici protokol düğümüne aktarır. Daha sonra, gönderici protokol düğümü iletişim altyapısının rol katmanında gelen parametre ve SİO'dan elde ettiği sohbet bilgileri ile rol içeriğini hazırlar. Rol içeriğini hazırlama sürecinden sonra, hazırlanan rol içeriği İletişim Sağlayıcısına yönlendirilir. İletişim Sağlayıcısında, etmen katmanı ACL iletisini RYS'den elde ettiği etmen tanımlayıcıları ile hazırlar ve bu hazırlanan ACL iletisini aktarım katmanına gönderir. Daha sonra, aktarım katmanı aktarım iletisini AMS'den elde ettiği katılımcı iletişim noktasının ip adresleri ile hazırlar. Akabinde etmenin iletişim sağlayıcısının arayüzü olan iletişim noktası aktarım iletisini katılımcı iletişim noktasına gönderir. Daha sonra, SEAGENT iletişim mimarisi internet aracılığıyla özelleşmiş portlara sahip iki iletişim noktası arasında aktarım iletisini gönderir.

Aktarım iletisi katılımcının iletişim noktasına vardığı zaman, aktarım iletisinin ACL paketi bir üst seviyeye doğrudan iletilir. Aktarılan ileti paketi etmen seviyesi ile işlenir ve rol içeriği rol içeriğinde belirtilen rol örneğine aktarılır. Sonra, rol seviyesinde alıcı protokol düğümü gelen rol içeriğini yorumlar ve gelen argümanları ön koşul olarak bekleyen iş akışı düğümüne protokol akışı ile aktarır. Daha sonra katılımcı taraf iş akışını çalıştırmaya devam eder.

4. Durum Çalışması Ve Görsel Destek

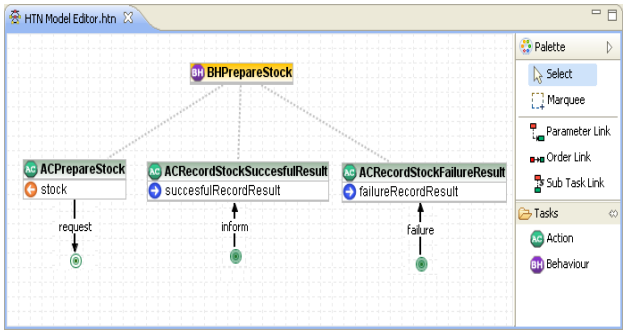
Bu bölümde durum çalışması olarak bir elektronik takas senaryosu SEAGENT Çoklu-Etmen Geliştirme Çerçevesi ile gerçekleştirilmiştir. Bu durum çalışması Müşteri rolü ve Takascı rolü arasında gerçekleşmektedir. Senaryodaki sistem hedefini gerçekleştirebilmek için Müşteri rolü stok hazırlanmasından Takascı rolü ise müşteri stoklarının Takas Uygulamasına kayıt edilmesinden sorumludur. Stok kayıt süreci için ilk olarak müşteri rolünü oynayan ilklendirici etmen, takascı rolünü oynayan etmene stok örneklerini içeren iletiyi gönderir. Daha sonra, katılımcı etmen gelen stok isteğini değerlendirir ve iletişimi ilklendiren etmene stok örneğinin uygun şekilde takas uygulamasına kayıt edilip edilmediği bilgisini iletir. Bu senaryo katılımcıdan gelen iletinin ilklendirici etmen tarafından yorumlanıp uygun olan davranışı çalıştırması ile son bulur.

Yukarıda söz edilen gibi bir sistem golünü SEAGENT'da geliştirmek için, çalışma zamanı girdileri ontoloji kullanarak tanımlanmalıdır. SEAGENT'da çoklu etmen senaryosunu tasarımını kolaylaştırmak için, SEAGENT görsel editör desteği ile sağlar. Görsel editör geliştiricilere kolayca çoklu-etmen sistemi geliştirmek ve SEAGENT çalışma zamanı girdilerini özdevinimli olarak çıkarmak için yardım eder.



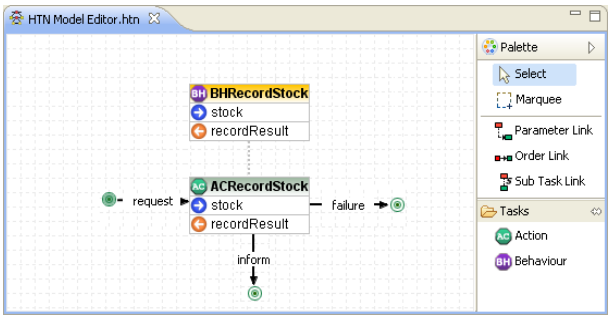
Şekil 8: Hedef Modeli Durum Çalışması

Görsel editör desteğimizi daha açıkça belirtmek için, yukarıda anlatılan durum çalışmasını SEAGENT platformunda nasıl gerçekleştirildiğini anlatalım. SEAGENT'da çoklu-etmen senaryosu gerçekleştirmek için öncelikle sistem hedefi Platform Editörü tarafından geliştirilir. Platform Editör sistem hedeflerini geliştirilmesi için SEAGENT geliştirme ortamına ait çoklu etmen sistemlerinin organizasyon seviyesinden bakılmasını sağlayan görsel geliştirim aracıdır. Bu görsel araç ile geliştirilen stok kaydetme sistem hedefi Şekil 8'de gösterilmiştir. Görüldüğü üzere müşteri stok kaydı Müşteri rolüne atanan PrepareStock ve Takascı rolüne atanan RecordStock rol hedeflerinden oluşan RecordCustomerStock sistem hedefi ile tanımlanmıştır. Sistem golü tasarlandıktan sonra platform editörü platform ontolojisini özdevinimli olarak üretir.



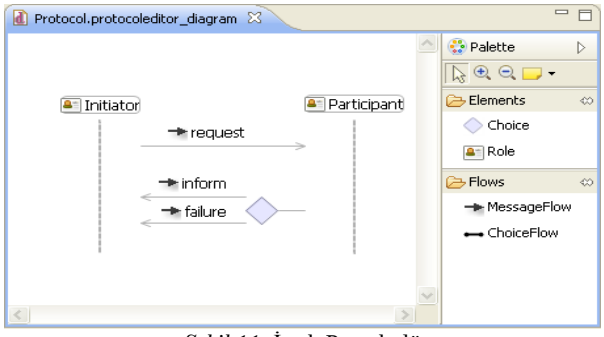
Şekil 9: Stok Hazırlama Davranışı

Geliştirme safhası sırasında platform ontolojisi üretiminden sonra, rol hedeflerinin planları HGA Editör ile tasarlanır. HGA Editör plan ontolojisini ve her eylem için çalıştırılabilir JAVA sınıflarını oluşturur. Rol hedeflerinin planları Şekil 9 ve Şekil 10'da gösterilmiştir.



Şekil 10: Stok Kayıtlanma Davranışı

Eğer üretilen plan etkileşim içeriyorsa, bu planlara etkileşim sırasında senkronize olmalarını sağlamak için protokol atanmalıdır. Bu iletişimsel planlar arasında protokol tanımını sağlamak için, sistemdeki kullanılan protokolleri üreten Protokol Editörü yardımıyla bir protokol atanır. Uygun protokol davranışlara atandıktan sonra, hangi eylemlerin protokol iletisi gönderme ve alma sorumluluğuna sahip olduğunu belirlemek için protokol akışları davranışların uygun olan çıktılarına HGA editör vasıtası ile atanır. Müşteri stok kayıt hedefini geliştirmek için kullanılan İstek Protokolü Şekil 11'de gösterilmiştir.



Şekil 11: İstek Protokolü

Görüldüğü üzere protokol iki katılımcı ve bu katılımcılar arasında bir ileti dizisi içermektedir. İstek Protokol ontolojisi özdevinimli olarak Protokol Editörü tarafından üretilir. Protokol ontolojisi bütün alt bileşenleri ile birlikte bir protokol örneği (Şekil 12'de gösterilmektedir) ve her katılımcı için bir rol örneği içerir.

```
<rdf:Description rdf:about="Protocol.owl#indv_protocol_5757">
  <j.1:hasSequence rdf:nodeID="A0"/>
  <j.1:hasParticipantRole rdf:resource="Protocol.owl#indv_role_7162"/>
  <j.1:hasInitialRole rdf:resource="Protocol.owl#indv_role_6162"/>
  <rdf:type rdf:resource="Protocol.owl#Protocol"/>
</rdf:Description>
```

Şekil 12: Protokol Örneği

Ayrıca ileti aktarımı etkinliklerini tanımlamak için, protokol örneği iletişim adımlarını içeren dizge adında bir listeye sahiptir. Dizge örneği her ileti gönderimi için Şekil 13'de örneği gösterilen bir basit adım örneği,

```
<rdf:Description rdf:nodeID="A2">
  <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
  <rdf:first rdf:resource="Protocol.owl#indv_message_5780"/>
</rdf:Description>
```

Şekil 13: Basit Adım Örneği

veya her dallanmış ileti için Şekil 14'de örneği gösterilen bir seçimsel adım örneği içerir.

```
<rdf:Description rdf:about="Protocol.owl#indv_choice_3102">
  <j.1:hasSequence rdf:nodeID="A2"/>
  <j.1:hasSequence rdf:nodeID="A3"/>
  <rdf:type rdf:resource="Agency.owl#Choice"/>
</rdf:Description>
```

Şekil 14: Seçimsel Adım Örneği

Editörler yardımıyla sistem girdileri üretilen müşteri stok kayıt senaryosunda, ilk önce Müşteri rol örneği stoklarını kayıt edebilmek için *ACPrepareStock* eylemi ile stokları hazırlar. Sonra *ACPrepareStock, ACRecordStock* ön koşulunu önerdiğimiz üç katmanlı iletişim mimarisi yardımıyla atamak için stok çıktısı üretir ve Takascı rol örneğine bu çıktıyı iletir. Akabinde, gelen stok *ACRecordStock*'s ön koşuluna atanır. *ACRecordStock* eylemi gelen stok örneğini değerlendirir, uygun gördüğü takdirde stok örneğini sisteme kayıt eder ve Müşteri rol örneğine kayıt işleminin başarılı yapıldığına dair bir bilgilendirme iletisi gönderir. Aksi takdirde *ACRecordStock* eylemi başarısız sonuç üretilir ve Takascı rol örneği başarısız edimseline sahip bir iletiyi Müşteri rol örneğine gönderir. En sonunda Müşteri rol örneği gelen iletiyi alır ve etkileşimin nasıl sonlanacağını belirlemek için gelen cevap iletisini yorumlar. Eğer gelen cevap iletisi bilgilendirme edimseli içeriyorsa, Müşteri rolü kendi bilgi-tabanını *ACRecordStockSuccesfulResult* eylemi ile günceller. Bunun yanı sıra eğer cevap iletisinin edimseli başarısız ise Müşteri rol örneği durumuna bağlı olarak *ACRecordStockFailureResult* eylemi ile başarısızlığın nedenlerini anlar ve sonra *RecordCustomerStock* hedefini gerçekleştirmek için tekrar dener veya bu durumu önemsemez.

5. Diğer Rol Tabanlı Yaklaşımlar

Çoklu etmen platformları için birçok rol tabanlı mimari önerilmiş ve gerçekleştirilmiştir, her biri iletişim mimarisi için farklı bakış açılarına sahiptir. İlk bölümde rol tabanlı iletişim mimarisinde olması gereken temel özellikleri karşılamak için gerekli olan artırımlardan bahsedilmiştir. Literatürde, bazı önerilen mimariler bu artırımları kendi iletişim mimarilerinde gerçekleştirmişlerdir. Bu mimariler BRAIN[5,6,7], RADE[15] ROLEEP[14], ROPE[4] ve RICA[12]'dır. Bu mimariler arasında anlaşılabilir bir karşılaştırma yapabilmek için, bahsettiğimiz artırımlara odaklanmalıyız. Belirtilen rol tabanlı yaklaşımların birinci bölümde bahsedilen artırımlara göre karşılaştırması Şekil 15'de gösterilmektedir.

	Rol Örneği	Programlanabilir Protokol	İşletilebilir Protokol	Rol Yönetim Servisi
ROLEX	–	–	–	*
RICA	+	–	–	–
ROPE	+	–	–	–
RADE	+	–	–	*
ROLEEP	–	–	–	*
SEAGENT	+	+	+	+

– Özelliği Değil
+ Özelliği
* Kısmen Özelliği

Şekil 15: Rol Tabanlı Yaklaşımların Karşılaştırılması

BRAIN platformu rolü kabiliyet ve beklenilen davranışlar kümesi olarak tanımlayan üç katmanlı bir yaklaşım önermiştir. BRAIN çerçevesinin içinde rol kavramı rol tanımlayıcısı olarak adlandırılan XML gösterimi ile tanımlanmaktadır, bu nedenler rol tanımı devingenliği destekler ve uygulama geliştiriminin değişik safhalarında rol kavramının kullanımına izin verir, ama çalışma zamanında rol örneği gibi etkileşim

katılımcılarını niteleyen benzersiz bir kavrama sahip değildir. Ayrıca BRAIN iletişim mimarisi sadece eylem-olay mekanizmasına dayanır. Bu nedenle çalışma zamanında protokol çalıştırılabilir ve açık şekilde programlanabilir değildir. Buna ek olarak, BRAIN çerçevesi, ROLEX[6] adında devingen olarak rollerin etmenlere atanma desteğini sağlayan bir ortam geliştirilmiştir. BRAIN çerçevesini taban alarak geliştirilen ROLEX ortamı açıklık ve birlikte çalışabilirlik ile alakalı servisleri destekler.

Bir diğer etmen uygulaması geliştirmek için önerilen rol tabanlı etmen geliştirme çerçevesi RADE'dir. Rol kavramı RADE çerçevesi içerisinde hedef ve etkileşim protokolleri gibi alan ilgili kavramlar ile birlikte tanımlanır. Rol örneği somut rol kavramından ortaya çıkarılır ve sistemin çalışma zamanında gerçek eleman olarak kullanılır. Organizasyon bakış açısından, RADE çerçevesi rol örneklerini yönetmek için bir servise sahiptir, bu servis bütün rol örneklerini kayıtlarını saklamaktan, sisteme kayıtlı olan rol örneklerini sorgulamaktan, rol örnekleri yaratıp silmekten sorumludur, bu özellikleri organizasyon seviyesinde sağlamasına rağmen RADE çerçevesi dağıtık ortamların için esnek bir ortam sağlamaz çünkü rol örnekleri merkezi bir sistem servisi tarafından yüklenir ve işletilir.

RICA platformu Etmen İletişim Dilleri ve Organizasyon Modeli bakış açısıyla ilgili konuların tümleşimiyle FIPA standartlarına sosyal kavram desteği sağlayan bir yaklaşımdır. RICA'ya göre, organizasyon modeli rollerin, etmenlerin, etkileşimlerin ve protokol tanımı ile ilgili olan ACL şartnamesinin tanımını içerir. RICA'da etmenin oynadığı rol kavramının örneği platformda somut bir kavram olarak ele alınır. RICA'da BRAIN gibi rol örnekleri etmen platformunda devingen olarak katılamaz. Ayrıca RICA teorisi organizasyon seviyesinde rol kavramlarını yönetecek bir servise sahip değildir.

Diğer bir rol yaklaşımı ROLEEP ortak hedefleri gerçekleştirebilmek için işbirlikçi etmenlerin birlikte çalışmasına imkan sağlayan bir ortam sağlar. ROLEEP'de etkileşimler sırasında etmenler diğerler etmenler ile rol kavramı üzerinden işbirliğine girerler. ROLEEP de etkileşim sırasında, gerçekleşen bütün olayları ve ileti gönderimlerini içerisinde tanımlayan roller nesne olarak adlandırılan programlanabilir kavramların somut metotlarına bağlanır. Bu olaya arayüz bağlama denir. Arayüz bağlama aynı ortamda bulunan diğer rollerden ileti almak için etkileşimin soyut noktasını tanımlar. ROLEEP nesne ve rollerin birleşmesinden oluşan benzersiz bir kavrama sahiptir. Bu kavram ROLEEP'de bizim rol örneği tanımımızla aynı şekilde kullanılır. Ama bu kavram etmen platformunda etkileşimi nitelemek için kullanılmamıştır. Bununla birlikte ROLEEP'de etkileşim protokollerini niteleyen bir tanım bulunmamaktadır. Organizasyon bakış açısından ise ROLEEP rol kavramlarını aramak için kullanılan rol bulucu bir servise sahiptir.

ROPE dağıtık etmen sistemi geliştirmek için rol tabanlı yaklaşım kullanan geliştirim ortamıdır. ROPE mimarisinde rol kavramı etmenler ve işbirliği süreci arasında iyi tanımlanmış bir arayüz sağlar. Böylece çoklu etmen sistemindeki etmen organizasyonu, işbirliği süreci yapısından ayrılır. Ayrıca ROPE geliştirim ortamında etmenler birlikte çalışabilmek için rol kavramlarını ilklendirirler. Etmen rolleri tasarım zamanında belirlenir ve çalışma zamanında etmenler tanımlanan rollerini değiştirebilir. Etkileşim rol örnekleri arasında tanımlanır. ROPE ortamında tanımlanan etkileşimler içerisinde protokol kavramını içermez. Ayrıca ROPE ortamı rol ve etmen ilişkilerini yönetmek için bir rol servisine sahip değildir. Çünkü biraz öncede bahsettiğimiz gibi rol etmen arasındaki ilişki durağandır ve yeni rol örnekleri çalışma zamanında sisteme eklenemez.

Sonuç olarak, rol tabanlı mimari yaklaşımlarını değerlendirmek gerekirse, BRAIN çerçevesi üzerine geliştirilmiş ROLEX ortamı rol tabanlı iletişim için ana unsur olan rollerin ilklendirilip sistemde görünebilirliğini sağlayan bir kavrama sahip değildir. Bununla birlikte iletişimde eş zamanlılığı sağlayan ve rolün bir alt bileşeni olan protokolün sistemde görünebilirliği sağlamaz. Ayrıca ROLEX'in sahip olduğu rol yönetimi sadece devingen olarak etmenlerin davranışlarını değiştirebilmektedir, yani ROLEX ortamı tam anlamıyla bir rol yönetimine sahip değildir. Bu sebeple ROLEX iletişim bakış açısından rol tabanlı bir iletişim gerçekleştirmemektedir. RADE çerçevesi rol kavramını ilklendirip sistemde rol kavramını kısmen görünürlüğünü sağlamaktadır ama iletişimde kullanılan protokolleri davranış kavramlarının içerisinden çıkartıp ayrı programlanabilir ve çalıştırılabilir bir kavram olarak tanımlamamıştır. Bu nedenle devingen olarak sisteme rolün alt bileşenlerinden biri olan protokol sisteme devingen olarak eklenmemektedir. Organizasyon bakış açısından RADE, ROLEX ortamına göre rollerin yönetilebilirliğini sağlar fakat rol yönetiminin gerçekleştirilmesi rol tabanlı iletişimin gereksinimlerini tam anlamıyla sağlamaz. Bu sebeple RADE çerçevesi rol tabanlı iletişimi desteklemez. RICA ve ROPE mimarileri rolün çalışma zamanında ilklendirerek rolün sistemde kısmen görünebilmesini sağlar. İletişim için gerekli olan protokoller bu iki mimaride de çalıştırılabilir ve programlanabilir olmadığı için protokollerin görünebilir değildir. Sadece ROPE yaklaşımında tanımlanmış bir koordinasyon süreci vardır. Bu koordinasyon sürecide iletişim sırasında kullanılan protokollerin çok sınırlı bir gösterimini içerir. Bununla birlikte bu iki mimaride rol yönetimi sistemi olmadığı için çalışma zamanında yeni rol kavramları sisteme eklenemez sadece tasarım sırasında belirlenmiş olan rolleri oynayabilirler. Bu sebeplerden dolayı iki mimarinin iletişim altyapısı rol kavramı dayanmamaktadır. ROLEEP mimarisinde ise roller ilklendirilmez. Roller sadece nesnelerin sohbet sırasında oynaması gerektiği arayüzler olarak tanımlanmıştır. ROLEEP mimarisinde protokoller göz ardı edilmiştir. Rol yönetimi içinse sadece rol arayıcı bir servise sahiptir. ROLEEP 'de bu nedenlerden dolayı rol tabanlı iletişimi desteklememektedir.

Bu platformların aksine, bir önceki bölümde anlatılan gerçekleştirimler sayesinde SEAGENT ÇES çerçevesinde protokollerin ontolojiler ile tanımlanması ve etmen planlarıyla birlikte iş akışlarına çevrilmesi protokollerin programlanması ve işletilmesi desteği sağlanmıştır. Ayrıca FIPA soyut mimarisinde yapılan FIPA ACL artırımlarıyla etmenlerde ilklendirilen rollerin birbirleri ile iletişimine olanak sağlanmıştır. Son olarakta FIPA soyut mimarisine ek olarak tanımlanan RYS'nin gerçekleştirimiyle roller sistemde tam olarak desteklenmiştir.

6. Sonuç

Birçok etmen platformu etmen etkileşimlerinde esneklik kazandırmak için rolleri kullanmıştır. Ancak rollerin karakteristik özellikleri belirlenip bu çalışmalar tekrar değerlendirildiğinde tam olarak rol desteği verdiklerinden söz edilemez. Bu çalışmada rol kavramının gerçekleştirilebilir yazılım elemanı olması için gerekli temel özelliklerini Krinstensen[11] ve Depke[10] çalışmalarından yola çıkarak tanımlanmıştır. Bu temel özellikleri yorumlayarak var olan ÇES'lerde rollerin gerçeklenebilir yazılım elemanı olabilmesi için gereksinimleri belirlenmiştir. Bu kapsamda FIPA soyut iletişim mimarisini rol kavramı desteklemek üzere genişletilmiştir. Bu mimariye uygun olacak şekilde de SEAGENT ÇES geliştirim çerçevesinin rolleri desteklemesi için bu gereksinimleri gerçekleştirmesi bu bildirinin önceki bölümlerinde ayrıntılarıyla anlatılmıştır.

7. Kaynakça

[1] Oguz Dikenelli. Seagent mas platform development environment. In AAMAS (Demos), pages 1671–1672, 2008.

[2] Erdem Eser Ekinci, Ali Murat Tiryaki, Onder Gurcan, and Oguz Dikenelli. A planner infrastructure for semantic web enabled agents. In OTM Workshops, volume 4805 of Lecture Notes in Computer Science, pages 95–104, Vilamoura, Algarve, Portugal, 2007. Springer.

[3] Matteo Baldoni, Guido Boella, and Leendert van der Torre. Modelling the interaction between objects: Roles as affordances. In Proceedings of First International Conference on Knowledge Science, Engineering and Management, Guilin, China (5th–8th August 2006), volume 4092 of LNCS, pages 42–54, 2006.

[4] Michael Becht, Thorsten Gurzki, JÃOErgen Klarmann, and MatthiasMuscholl. Rope: Role oriented programming environment for multiagent systems. volume 0, page 325, Los Alamitos, CA, USA, 1999. IEEE Computer Society.

[5] Giacomo Cabri, Luca Ferrari, and Letizia Leonardi. A case study in rolebased agent interactions. In WETICE '03: Proceedings of the Twelfth International Workshop on Enabling Technologies, page 42, Washington, DC, USA, 2003. IEEE Computer Society.

[6] Giacomo Cabri, Luca Ferrari, and Letizia Leonardi. The rolex environment for multi-agent cooperation. In 8th International Workshop on Cooperative Information Agents (CIA. Springer-Verlag, 2004.

[7] Giacomo Cabri, Letizia Leonardi, and Franco Zambonelli. Implementing role-based interactions for internet agents. In SAINT '03: Proceedings of the 2003 Symposium on Applications and the Internet, page 380, Washington, DC, USA, 2003. IEEE Computer Society.

[8] Fabio Casati, Stefano Ceri, Barbara Pernici, and Giuseppe Pozzi. Workflow evolution. In ER '96: Proceedings of the 15th International Conference on Conceptual Modeling, pages 438–455, London, UK, 1996. Springer-Verlag.

[9] Mike Dean and Guus Schreiber. OWL web ontology language reference. W3C recommendation, W3C, February 2004.

[10] Ralph Depke, Reiko Heckel, and Jochen M. Kuster. Improving the agentoriented modeling process by roles. In AGENTS '01: Proceedings of the fifth international conference on Autonomous agents, pages 640–647, New York, NY, USA, 2001. ACM.

[11] Bent Bruun Kristensen. Object-oriented modeling with roles. In Proceedings of the 2nd International Conference on Object-Oriented Information Systems, pages 57–71. Springer-Verlag, 1995.

[12] Juan Manuel Serrano and Sascha Ossowski. On the impact of agent communication languages on the implementation of agent systems. In Lecture Notes in Artificial Intelligence, ISSN, pages 0302–9743. Springer, 2004.

[13] Katia Sycara, M. Williamson, and K. Decker. Unified information and control flow in hierarchical task networks. In Working Notes of the AAAI-96 workshop 'Theories of Action, Planning, and Control', August 1996.

[14] Naoyasu Ubayashi and Tetsuo Tamai. Separation of concerns in mobile agent applications. In REFLECTION '01: Proceedings of the Third International Conference on Metalevel Architectures and Separation of Crosscutting Concerns, pages 89–109, London, UK, 2001. Springer-Verlag.

[15] Xiaoqin Zhang, Haiping XU, and Bhavesh ShresthaDr. An integrated role-based approach for modeling, designing and implementing multiagent systems. volume 13, pages 45–60, 2007.

[16] Kendall, Elizabeth A., Aspect-oriented programming for role models, 1999, pp. 294-295

[17] Kendall, Elizabeth A., Agent software engineering with role modelling, Springer-Verlag New York, Inc., 2001, pp. 163-169

[18] Georg Gottlob, Michael Schrefl, Brigitte Röck: Extending Object-Oriented Systems with Roles. ACM Trans. Inf. Syst. 14(3): 268-296 (1996)

[19] Baldoni, Matteo and Boella, Guido and Torre, Leendert Van Der, Modelling the interaction between objects: Roles as affordances, Springer, 2006, pp. 42-54

[20] Steimann, Friedrich and Wissensverarbeitung, Rechnergestützte, On the representation of roles in object-oriented and conceptual modelling, Data and Knowledge Engineering, 2000, pp. 83-106

[21] Wegmann, Alain and Genilloud, Guy, The Role of Roles in Use Case Diagrams,Springer-Verlag,2000,pp. 210-224

[22] B. J. Biddle, Recent Developments In Role Theory, Annual Review of Sociology, 1986, pp. 67-92

[23] James J. Odell and H. Van and H. Van Dyke Parunak and Mitchell Fleischer, The Role of Roles in Designing Effective Agent Organizations, Springer, 2003, pp. 27—38

[24] Zambonelli, Franco and Jennings, Nicholas R. and Wooldridge, Michael, Developing multiagent systems: The Gaia methodology, ACM Trans. Softw. Eng. Methodol., 2003, pp. 317-370

[25] Lin Padgham and Michael Winikoff, Prometheus: A Methodology for Developing Intelligent Agents, 2002

[26] Omicini, Andrea, SODA: Societies and infrastructures in the analysis and design of agent-based systems, Springer-Verlag, 2000, pp. 185-193

[27] Bresciani, Paolo and Giorgini, Paolo and Giunchiglia, Fausto and Mylopoulos, John and Perini, Anna, TROPOS: An Agent-Oriented Software Development Methodology, Journal of Autonomous Agents and Multi-Agent Systems, 2004