Kanı-İstek-Hedef Etmenlerinin Model Güdümlü Geliştirilmesi

B.Afşar[1] , G. Kardaş[2] , N. Y. Topaloğlu[1] , O. Dikenelli[1]

[1]Bilgisayar Mühendisliği Bölümü, Ege Üniversitesi, İzmir

[2]Uluslararası Bilgisayar Enstitüsü, Ege Üniversitesi, İzmir

[1]{bekir.afsar, oguz.dikenelli, yasemin.topaloglu}@ege.edu.tr
[2]geylani.kardas@ege.edu.tr

Özetçe

Kanı-İstek-Hedef(KİH) mimarisi etmenlerin çevrelerine dair bilgiler tutmasına, bu bilgiler doğrultusunda tasarım hedeflerini gerçekleştirebilmek için planlarını oluşturmalarına izin veren bir mimaridir. Bu mimariye ait etmenlerin gerçekleştirim ayrıntılarından daha üst soyutlama seviyesine çıkılarak geliştirilmesi geliştirim sürecini hızlandıracaktır. Yazılım geliştirme odağını koddan modellere çeviren ve farklı soyutlama seviyelerindeki modelleri kullanarak yazılım geliştirmedeki karmaşıklığı azaltmayı hedefleyen Model Güdümlü Geliştirme(MGG) yaklaşımının KİH etmen yazılımlarının hızlı bir şekilde geliştirilmesinde önemli bir yaklaşım olduğu görülmektedir. Bu bildiride KİH etmenlerinin MGG yaklaşımı kullanılarak geliştirilmesi için izlenilen yöntem anlatılmıştır.

Model Driven Development of Belief-Desire-Intention Agents

Abstract

Belief-Desire-Intention (BDI) is an agent architecture that supports the storage of known facts about the environment, inference on these facts and planning according to the design goals. Software development of BDI agents can be easier and more efficient by working on different abstraction levels. We believe that Model Driven Development (MDD), which aims to change the focus of software development from code to models, also provides the development of BDI agents within this context. This paper introduces an MDD approach for developing BDI Agents.

1. Giriş

Etmenler, algılayıcıları ("*sensor*") yardımıyla bulunduğu ortamı algılayan ve etkileyicileri ("*effector*") ile bu ortamı etkileyen yazılım ya da donanım sistemleridir[1]. Yazılım etmenleri("*software agents*") kullanıcısı adına bulunduğu ortamda otonom ve özerk eylemlerde bulunabilen, çevresindeki değişimlere duyarlı ve tepkide bulunan bilgisayar sistemleridir. Etmenlerin içsel yapısı ve işleyişi ile ilgili Etmen Tabanlı Yazılım Mühendisliği kapsamında çeşitli etmen mimarileri önerilmiştir. Bu mimariler, etmenin karar verme süreci ve işleyişine, bulunduğu ortama dair bilgileri nasıl gösterdiğine, kullandığı veri yapıları ve bu yapılar arasındaki bilgi akışını nasıl gerçekleştirdiğine göre farklılık göstermektedir. Etmenlerin geçmişe dair bilgileri tutmadığı ve geleceğe dair planlar yapmadığı ancak çevresinden gelen tepkilere etkide bulunduğu karşıt-eylemli("*reactive*") mimariler veya bulunduğu ortama ve geçmişe dair bilgilerin tutulduğu, geleceğe dair planların yapıldığı Kanı-İstek-Hedef(KİH) mimarileri değinilen etmen mimarilerine örnek olarak

verilebilir. Bu çalışma kapsamında KİH etmen mimarilerinin geliştirimi üzerinde durulacaktır.

Kanı-İstek-Hedef ("*Belief-Desire-Intention*" - KİH)[2] teorisi kanı, istek ve hedef ile olası ortam durumlarını mantıksal olarak göstermeyi hedeflemektedir. Hedefler, kanıların ve isteklerin oluşturduğu bir alt kümedir. Bu mimariye göre, kanı ("*belief"*), etmenin bulunduğu ortama ait inandığı bilgilerdir. İstek ("*desire"*), etmenin başarmayı amaçladığı şeylerdir. Hedef ("*intention"*) ise, etmenin üzerinde çalıştığı amaçtır, bu amaç doğrultusunda gerçekleştirdiği planlardır. KİH yazılım etmenlerinin gerçekleştirimi için çeşitli platformlar bulunmaktadır[JACK [1] , JASON [2] , SPARK [3] , 3APL [4] , vb.]. Bu mimariye ait etmenlerin gerçekleştirimi kanı-istek ve hedeflerin açık bir şekilde gerçekleştirilmesi zorunluluğundan dolayı oldukça karmaşık bir yapıya sahiptir.

Etmenin sahip olduğu bilgilerin mantıksal olarak gösterimi ile bu bilgiler ışığında ve tasarım hedefleri doğrultusunda ortaya çıkacak olan planların gerçekleştirimi hem zaman alıcı olması hem de zorluk derecesinin yüksek olması sebebi ile oldukça güçtür. Bu geliştirim sürecini daha kolay ve hızlı bir şekilde gerçekleştirmek için Model Güdümlü Geliştirme(MGG) iyi bir çözüm olarak karşımıza çıkmaktadır. MGG, yazılım geliştirmede önemli olanın probleme dair çözümün olduğunu vurgulamakta ve öncelikli hedefin çözümü yansıtan modellerin geliştirilmesi olduğunu söylemektedir. Bu doğrultuda gerçekleştirim ayrıntılarından daha üst soyutlama seviyelerine çıkılmakta ve süreç tamamen problem çözümüne dayalı olarak gelişmektedir.

KİH mimarisine uygun olarak etmen geliştirme platformlarından en çok kullanılanlardan birisi de JADEX [5] [3] etmen platformudur. Bu durumdan hareketle çalışmamızda KİH etmenlerinin MGG yaklaşımı ile geliştirilmesi için JADEX platformu seçilmiştir. Bu çalışmada JADEX KİH etmenlerinin MGG yaklaşımı ile geliştirilmesi için gerçekleştirilen model güdümlü geliştirme aracı ve kod üreticisi anlatılacaktır.

Bildirinin geriye kalan kısmı şu şekilde düzenlenmiştir. Bölüm 2'de JADEX etmen mimarisi anlatılmıştır. Bölüm 3'te JADEX etmen mimarisi için geliştirilen JADEX üst-modelinden("*metamodel*") söz edilmiştir. Bölüm 4'te JADEX KİH etmenlerinin model güdümlü geliştirilmesi anlatılmıştır. Bölüm 5'te model güdümlü KİH geliştirme ile ilgili diğer çalışmalar aktarılmıştır. Bölüm 6'da ise sonuç ve ileriye yönelik çalışmalar yer almaktadır.

2. JADEX Etmen Mimarisi

JADEX, hedef yönelimli etmenlerin, Kanı-İstek-Hedef mimarisine uygun olarak oluşturulmasına izin veren bir yazılım çatısıdır. JADEX mimarisinde etmenler, dışarıdan mesajların geldiği ve dışarıya mesaj gönderen bir kapalı kutu olarak ele alınmaktadır. JADEX, KİH mimarisini, etmenlerin kanı, amaç ve planlarının tanımlanması ile gerçekleştirmektedir. Kanılar herhangi bir Java nesnesi olabilir ve kanı tabanında ("*beliefbase"*) depolanmaktadırlar. Amaçlar etmenin hedeflediği durumu göstermektedir. Etmenin davranışları amaçlarına göre belirlenmektedir.
JADEX, akıl yürütme ("*reasoning*") sürecini iki aşamada gerçekleştirmektedir. Birinci aşama, dışarıdan gelen mesajlar, içsel olaylar ve amaçlar doğrultusunda planların seçilmesi ve çalıştırılmasından oluşmaktadır. İkinci aşama ise amaçların yeniden düzenlenmesi, yeni amaçların belirlenmesi ve hangi amacın gerçekleştirileceğine karar verilmesinden oluşmaktadır.

Etmenin bulunduğu ortam ile ilgili sahip olduğu bilgiler kanı tabanında tutulmaktadır. Kanıların

[1] http://www.agent-software.com.au/products/jack/ , son erişim: Nisan 2010
[2] http://jason.sourceforge.net/JasonWebSite/Jason%20Home.php , son erişim: Nisan 2010
[3] http://www.ai.sri.com/~spark/ , son erişim: Nisan 2010
[4] http://www.cs.uu.nl/3apl/ , son erişim: Nisan 2010
[5] http://jadex.informatik.uni-hamburg.de/, son erişim: Nisan 2010

gösterimi oldukça basittir. Henüz kanılar üzerinde herhangi bir çıkarsama mekanizması tanımlanmamıştır. Kanılar birbirlerinden metinsel tanımlayıcıları ile ayrılırlar. Bilinen gerçekler ("fact") tanımlayıcılar ile eşleştirilerek tanımlanırlar. JADEX'te iki tür kanı yapısı bulunmaktadır. Birincisinde yalnızca bir tane gerçek vardır. İkincisinde ise bir gerçek kümesi bulunmaktadır. Kanılar dinamik olarak değiştirilebilir. Dışarıdan gelen mesajlar sonucu gerçekleşen olaylar veya hedeflerin gerçekleştirilmesi sonucu oluşan ortam değişiklikleri kanıların güncellenmesine ve bu güncellemenin etmenin bilgi tabanına yansıtılmasına neden olur.

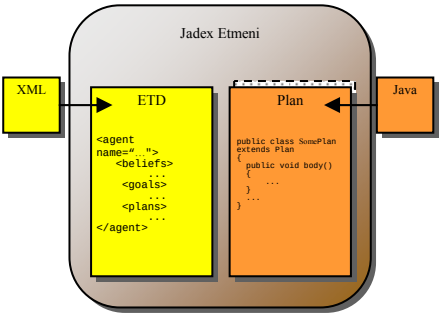
JADEX etmenlerinin gerçekleştirmek istedikleri, amaçlarla ("*goal*") tanımlanır. Geleneksel KİH mimarisinden farklı olarak, JADEX mimarisi amaçların açık bir şekilde tanımlanmasına izin vermektedir. Amaçlar birbirinden bağımsız bir şekilde tanımlanabileceği gibi bir hiyerarşiye uygun olarak da tanımlanabilirler. Amaçlar, JADEX'in çıkarsama sürecinde güncellenebilir, değişebilir veya yenileri eklenebilir.

JADEX etmenlerinin amaçlarını gerçekleştirmek için gerekli olan eylemler, planlarla tanımlanır. Planların başlıkları ve hangi şartlar gerçekleştiğinde çalıştırılacağı etmen tanımlama dosyasında tanımlanır. Belirtilen plan başlığına uygun bir şekilde planın işleyişi Java ile kodlanır. Java'nın sunmuş olduğu tüm nesne yönelimli özellikler burada kullanılabilir. Çıkarsama mekanizması haricindeki bileşenler(kanı-istek-hedef)in tamamı yeniden kullanılabilir yapıdadır ve yetenek("*capability*") adı verilen bir yapı altında tutulmaktadır.

Mimarinin çalışma mekanizmasına göre bir etmen mesaj kuyruğuna gelen bir mesaj doğrultusunda uygun yetenek dosyalarından uygun olayı seçmektedir. Olay tabanlı bir yaklaşım söz konusudur. Etmenin iç mimarisinde uygun planın seçilmesi gerçekleşen olaylara dayanmaktadır. Bu olaylardan birincisi dışarıdan gelen mesajlardır. İkincisi etmenin amaçları doğrultusunda gerçekleşen olaylardır. Üçüncüsü ise etmenin bilgi tabanında tuttuğu kanılara bağlı olarak gerçekleşen olaylardır. Bu olaylar doğrultusunda plan kütüphanesinden uygun plan seçilir ve seçilen plan çalıştırılır.

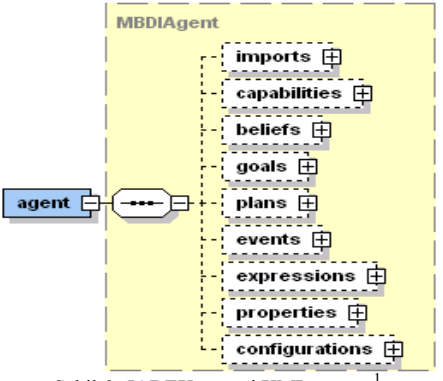
3. JADEX üst-modeli

JADEX yazılım çatısı, etmenlerin yönetilmesi için altyapı sunan bir etmen geliştirim platformudur. Bir JADEX etmeni, planların oluşturulması için JAVA ve etmen tanımlama dosyası ("*Agent Definition File*" - ETD) için XML kullanılarak geliştirilmektedir (Şekil 1).



Şekil-1: JADEX Etmeni

ETD'ler etmenlerin başlangıç kanı, istek ve hedef bilgilerini içermektedir ve KİH mimarisini temsil eden ve JADEX platformuna ait üst-modele uygun bir şekilde geliştirilmelidir. ETD'de kullanılabilecek önerilen üst-modelin temel varlıkları şekil-2'de görülmektedir.



Şekil-2: JADEX etmeni XML şeması[1]

[1] http://jadex.sourceforge.net/jadex-2.0.xsd, son erişim: Nisan 2010

ETD dışından kullanılacak sınıflar veya paketleri kullanabilmek için "*imports*" varlığı kullanılır. JADEX "*import*" etiketi sözdizimi Java'daki "*import*" ile aynıdır.

JADEX, etmenlerin kanılarının, amaçlarının ve planlarının yeninden kullanılabilmesi için yetenek ("*capability*") dosyalarının oluşturulmasını sağlar. Bu yapı etmen tanımlama dosyası ile aynı üst-modele uyumludur. Bu şekilde oluşturulan yeniden kullanılabilir yapıları etmen tanımlama dosyasında kullanabilmek için "*capabilities*" varlığı kullanılır.

Öte yandan JADEX etmeni bulunduğu ortam ile alakalı bilgileri kanılar("*beliefs*" varlığı) ile tanımlar. Etmenin inandığı gerçekler tek bir gerçek olabileceği gibi birden fazla da olabilirler. Java nesneleri ile temsil edilen planlardan JADEX'in sunduğu kanı arayüzü ile kanılara erişilebilir ve kanılara ait gerçekler güncellenebilir. Böylece etmenin sahip olduğu bilgiler dinamik bir şekilde yönetilebilmektedir.

JADEX etmenleri amaçlarını *"goals"* varlığı ile tanımlar ve dört farklı amaç yapısı bulunmaktadır. Bunlardan birincisi olan *"perform goal"*ün herhangi bir başarım veya başarısızlık koşulu yoktur. "*Perform goal*"ler, "*achieve goal*"ün gerçekleşmesi için yapılması gereken ek eylemleri tanımlar. "*Achieve goal*", hedeflenilen durumu tanımlamakta kullanılır ve etmenin asıl başarmak istediği hedeftir. Başarım koşulu sağlanıncaya kadar etmen çalışmasına devam eder. "*Query goal*", etmenin içinde bulunduğu ortam ile alakalı bilgilerin sorgulanması için kullanılan amaç yapısıdır. "*Maintain goal*", etmenin bulunduğu durumu koruması için tanımlanılan amaçtır. Tanımlanılan her türlü amacın bir yaratılma koşulu ve sonlanma koşulu bulunmaktadır. Yaratılma koşulu gerçekleştiğinde amaç gerçekleştirilmek üzere aktif hale getirilir ve ilişkili planlar işletilir. Çeşitli amaçlar gerçekleştirildikçe etmenin içinde bulunduğu ortama ilişkin bilgiler değişebilir. Bu doğrultuda etmenin yeni amaçları ortaya çıkabilir.

Planlar, etmenin bulunduğu ortamda gerçekleştireceği iş akışıdır. JADEX planları iki kısımdan oluşur. Birincisi etmen tanımlama dosyasında tanımlanan plan başlığıdır. İkinci kısmı ise Java nesneleri ile temsil edilen davranış kodlarıdır. Java ile kodlanan kısım planın gövdesini oluşturur ve etmenin gerçekleştirmeyi hedeflediği amaç doğrultusunda yapılması gereken eylemleri içermektedir.

Etmenlerin en önemli özelliklerinden biri zamanla gelişen olaylar karşısında o olaylara cevap verebilmesidir. JADEX iki tür etmen olayı tanımlamaktadır. Birincisi etmenin kendi içsel olayları için tanımlanan ve "*internal event*" adı verilen olaylardır. İkincisi ise etmenin bir veya birden fazla etmen ile gerçekleştirdiği haberleşmeler yani mesaj olayları ("*message events"*)dır. Olaylar etmene ait çeşitli hedefleri tetikleyebilir. Sağlanan haberleşme mekanizması FIPA [1] standartlarına uygun bir şekilde tanımlanmıştır.

JADEX etmenlerinin başlangıç ve bitiş durumları *"configurations"* yapısında tanımlanmaktadır. Başlangıç-bitiş planları ve amaçları gibi etmenin doğar doğmaz sahip olması gereken özellikler tanımlanır ve etmen yaratıldığı anda bu özellikleri oluşturulur.

4. JADEX Etmenlerinin Model Güdümlü Geliştirilmesi

Çoklu etmen sistemlerinin model güdümlü geliştirilmesi kapsamında yaptığımız bir önceki çalışmamızda modellerin platformdan bağımsız seviyede görsel olarak geliştirilmesi ve farklı platformların ÇES geliştirimi için kullanılması üzerinde durulmuştur[4]. Bu çalışmada ise KİH mimarisine uygun etmenlerin model güdümlü bir şekilde geliştirilmesi hedeflenmiştir. KİH mimarisini temel alan ve KİH modeline uygun etmenlerin gerçekleştirimi için kullanılan JADEX platformu bilinen en yaygın platformdur. JADEX etmen çerçevesinde yazılım etmeni geliştirirken gerçekleştirilmesi gereken XML tabanlı ETD çok karmaşık bir

[1] http://www.fipa.org, son erişim: Nisan 2010

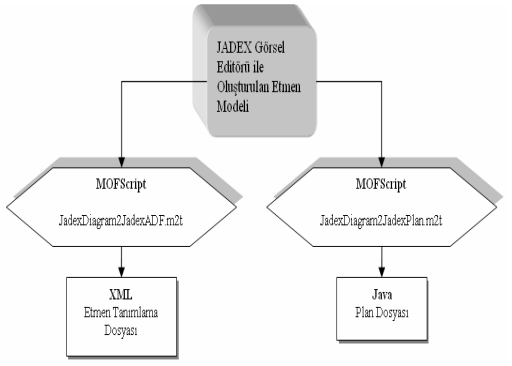
yapıya sahiptir. Yazılım geliştiricileri çözülmesi gereken problemim çözümünden daha ziyade ETD'leri kodlamada zaman harcamaktadırlar. JADEX etmenlerinin gerçekleştiriminde statik etmen bilgilerinin tutulduğu ETD'lerin kodlanmasındaki karmaşıklığın ve zorluğun giderilmesi için daha üst soyutlama seviyesine çıkmak ve model güdümlü yaklaşımı kullanmak gerekmektedir. Bu kapsamda kullanıcıların JADEX etmenlerine ait KİH yapısını görsel olarak modelleyebilecekleri ve bu modelden JADEX ETD ile plan kodlarını üretebilecekleri JADEX platformuna özgü bir yazılım geliştirme ortamı gerçekleştirilmiştir.

JADEX etmenlerinin model güdümlü geliştirimine olanak sağlayacak ortamımız yine model güdümlü teknolojiler kullanılarak gerçekleştirilmiştir. Ortamın geliştiriminde MGG için gerekli olan araçlar Eclipse[1] yazılım geliştirme ortamı kullanılarak gerçekleştirilmiştir. Eclipse ortamında görsel geliştirim araçları, GMF[2](*"Graphical Modeling Framework"*) kullanılarak oluşturulmaktadır. GMF farklı iş alanları ("domain") için grafiksel modelleme editörlerinin hazırlanmasını sağlayan bir çerçevedir. Editörlerin geliştirilmesi sırasıyla; çalışılacak alanı temsil eden modellerin oluşturulması, diyagram tanımlarına ait modellerin oluşturulması, çalışılacak alana ait model elemanları ile diyagram modelindeki elemanlar arasındaki eşlemelerin gerçekleştirilmesi ve ilgili modelleme editörünün kod üretimi adımlarından oluşmaktadır. Bu doğrultuda öncelikle JADEX etmenlerine ait modelleri görsel olarak geliştireceğimiz editörün GMF kullanarak geliştirilmesi için EMF[3] (*"Eclipse Modeling Framework"*) ile JADEX üst-modelini temsil eden Ecore modeli oluşturulmuştur. İkinci adımda görsel elemanlara ait model geliştirilmiştir. Daha sonra oluşturulan Ecore modeli ile görsel öğelere ait model elemanları arasında eşleştirmeler gerçekleştirilmiştir. Son adım olarak görsel geliştirim aracının kodu GMF yardımıyla üretilmiştir.

Geliştirilen editör ile JADEX platformuna özgü etmen modelleri görsel olarak hızlı ve etkin bir şekilde oluşturulabilmektedir. Editör kullanıcılarına JADEX'in üst-modeline ait varlıkları sürükle bırak yapısı ile sağ taraftaki palette sunmaktadır(Şekil-8). Geliştiriciler istedikleri üst-model varlığını JADEX'in KİH mimarisine uygun bir şekilde seçip kullanabilmektedirler. Bu varlıklara ait özellikleri editörün alt tarafında bulunan özellik ("*properties*") penceresinden girebilmektedirler. Kullanılan varlıklar arası ilişkiler yine sağ tarafta bulunan palet aracılığı ile kurulabilmektedir. Ayrıca editör JADEX KİH mimarisine uygun olmayan ilişkilerin kurulmasını engellemektedir. Örneğin bir Kanı varlığı sadece Etmen düğümü ile ilişkilendirilebilir. Başka bir KİH varlığı(Plan, Amaç, vb..) ile ilişkilendirilemez. Böylece kullanıcılar geliştirmek istedikleri etmen modelini hatasız ve hızlı bir şekilde oluşturabilmektedirler.

Görsel geliştirim aracı ile JADEX platformuna ait oluşturulan model, bu çalışma kapsamında geliştirilen koda dönüşüm aracı ile JADEX etmenlerinin gerçekleştirimi için gerekli olan kaynak kodlara dönüştürülmektedir. Bir JADEX etmeni oluşturabilmek için etmenin statik bilgilerinin tutulduğu XML tabanlı ETD ile etmenin planlarının gerçekleştirildiği işletilebilir Java dosyaları gerekmektedir. Görsel editör yardımı ile gerçekleştirilen etmen modelleri MOFScript[4] kullanılarak Java kodlarına ve XML dosyalarına dönüştürülmektedir. Modelden metin elde edilmesi için iki ayrı MOFScript kodu gerçekleştirilmiştir. Birinci dönüşüm kodu görsel editörle oluşturulan modeli etmen tanımlama dosyasına dönüştürmektedir. İkinci dönüşüm kodu ise oluşturulan görsel modeli Java kodlarından oluşan plan dosyasına dönüştürmektedir.

[1] http://www.eclipse.org/ , son erişim: Nisan 2010
[2] http://www.eclipse.org/gmf/ , son erişim: Nisan 2010
[3] http://www.eclipse.org/modeling/emf/ , son erişim: Nisan 2010
[4] http://www.eclipse.org/gmt/mofscript/ , son erişim: Nisan 2010



Şekil-3: Modelden koda dönüşüm süreci

Şekil 3'de gösterilen süreç, gerekli olan dönüşüm kodlarının yazılması ile gerçekleştirilmiştir. Gerçekleştirimi yapılan kod dönüşümünde görsel editörle oluşturulan JADEX etmen modeli girdi olarak alınmakta ve JADEX ETD çıktı olarak üretilmektedir. Şekil-4'de modelde bulunan etmen düğümünün ETD'deki etmen yapısına nasıl dönüştürüldüğünü gösteren bir kod parçası yer almaktadır. Oluşturulan dosyanın ismi JADEX'in ETD dosyaları için kullandığı isim konvansiyonuna uygun olarak isimlendirilmiştir. ETD dosyaları bir XML dosyası olduğu için dosya uzantısı '.xml' olarak belirlenmiştir.

```
texttransformation JadexDiagram2JadexADF(in jadex:"http://jadex/5.0") {
  main () {
    jadex.objectsOfType(jadex.Agent)->forEach(agent) {
     agent.generateAgentFile()
     }
}
    jadex.Agent::generateAgentFile(){
      file (self.name+".agent"+".xml")
      <%<agent xmlns="http://jadex.sourceforge.net/jadex"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://jadex.sourceforge.net/jadex
                        http://jadex.sourceforge.net/jadex-2.0.xsd"%>
                        '
                        name= "' self.name'"'
                        '
                        package= "' self.package'"'
                        '
                        description= "' self.description'"'
                        '
                        propertyfile="' self.propertyfile'"'
                        '
                        abstract="' self._abstract

      '>'
```

Şekil-4: Modelden ETD koduna dönüşümü sağlayan MOFScript kodundan bir parça

Gerçekleştirimi yapılan diğer kod dönüşümünde ise yine görsel editörle oluşturulan JADEX etmen modeli girdi olarak alınmakta ve çıktı olarak JADEX plan dosyası üretilmektedir. Şekil 5'de yer alan dönüşüm kod parçasında da görüldüğü gibi plan dosyaları Java dosyalarıdır. Bu yüzden dosya uzantısı '.java' olarak belirlenmiştir.

```
texttransformation JadexDiagram2JadexPlan (in jadex:"http://jadex/5.0") {
  main () {
    jadex.objectsOfType(jadex.Body)->forEach(body) {
     body.generatePlanFile()
     }
}
  jadex.Body::generatePlanFile(){
     file (self.class +".java")
       jadex.objectsOfType(jadex.Agent)->forEach(agent){
          agent.writePackage()
       }
```

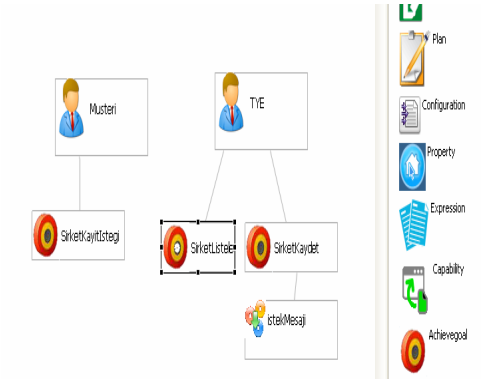
Şekil-5: Modelden Plan dosyasına dönüşümü sağlayan MOFScript kodundan bir parça

4.1. Durum Çalışması:

Geliştirilen model güdümlü görsel araç ile kod üreticisinin kullanımını göstermek için etmen tabanlı e-takas sistemi kullanılmıştır. Takas sistemi geleneksel ticarete alternatif olarak düşünülen, hizmetlerin ve ürünlerin para kullanılmadan değiştirilmesi prensibine dayanan ticari bir sistemdir. Etmen tabanlı takas sistemi ise etmenlerin hizmetleri ve ürünleri sahiplerinin istek ve özelliklerine göre aralarında değiştirdikleri sistemdir. Takas sistemine dahil olan tüm etmenler üstlendikleri rollere göre isimlendirilmektedirler. *Müşteri* etmeni sisteme yeni takas teklifleri ekleme ve gelen takas tekliflerini değerlendirme rolüne sahip etmendir. *Takas Yöneticisi Etmeni(TYE)* ise sisteme gelen takas tekliflerini toplama, uygun takas tekliflerini eşleme ve ilgili müşteri etmenlerini pazarlık için bilgilendirmekle yükümlü etmendir. Pazarlık süreci sonlandığında müşteri etmeni takas yöneticisi etmenine anlaşma mesajı göndermektedir. Daha sonra takas yöneticisi etmeni, *Kargo* etmenini uyarmakta ve ürünlerin müşteri etmenleri arasında değişimini gerçekleştirmesi için görevlendirmektedir.

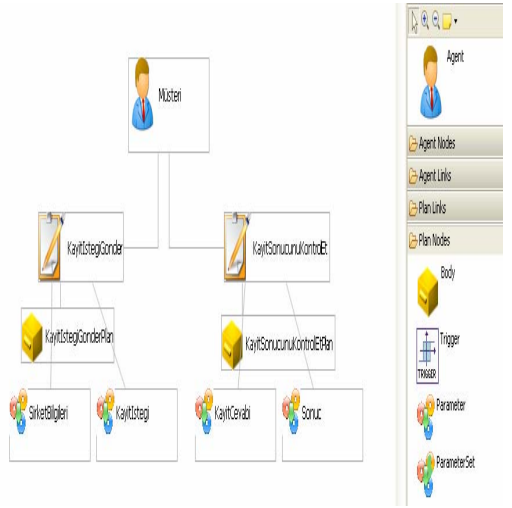
Bu çalışmada, müşteri etmeninin takas yöneticisi etmen aracılığı ile takas sistemine kayıt olma senaryosu gerçekleştirilmiştir. Müşteri etmeni diğer takas işlemlerini gerçekleştirebilmek için öncelikle sisteme kayıt olması gerekmektedir. Müşteri etmenleri temsil ettikleri müşteri özellikleri ve aradıkları ürün bilgileri ile sisteme kayıt olurlar. Belirtilen özellikler ve istekler doğrultusunda takas işlemleri gerçekleştirilmektedir. Kayıt olan müşteri etmeni takas işlemlerini gerçekleştirmek ve diğer müşteri etmenleri ile etkileşime geçmek için hazır hale gelir. Sözü edilen etmen tabanlı e-takas sisteminin tam tasarımı ve gerçekleştirimi [5]'de ayrıntılı bir şekilde anlatılmıştır.

Kayıt senaryosunun model güdümlü gerçekleştirimi için öncelikle sistemin organizasyon yapısı modellenmiştir. Sistemde olması gereken etmenler ve o etmenlere ait hedefler belirlenmiştir. JADEX'te rol kavramı bulunmadığından takas sisteminde bulunan roller etmenler tarafından temsil edilmektedir. Şekil 6'da görüldüğü üzere kayıt senaryosu için *Müşteri* ve *Takas Yönetici Etmeni* olmak üzere iki etmen bulunmaktadır. Onlara ait hedefler sağ tarafta bulunan palet yardımı ile modellenebilmektedir.



Şekil-6: Takas Sistemi Organizasyon modeli

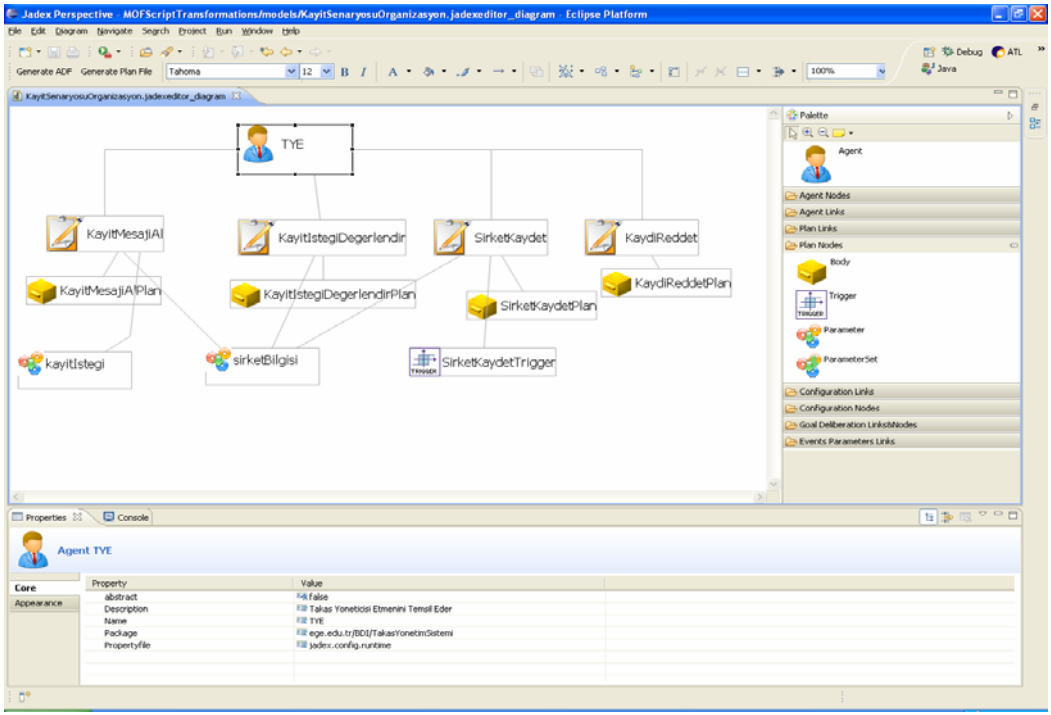
Müşteri etmeninin hedefi *SirketKayitIstegi* ile kendisini sisteme kayıt ettirmektir. TYE etmeni *SirketListele* ve *SirketKaydet* hedefleri ile kayıt isteğinde bulunan şirketleri kaydeder ve kayıt olan müşterileri listeler. Bir şirketin kaydedilebilmesi için *istekMesaji*'nin gelmesi gerekmektedir.



Şekil-7: Müşteri Etmenine ait plan modeli

Daha sonra belirlenen etmenlerin hedeflerini gerçekleştirebilmeleri için yapması gereken eylemleri temsil eden planlar modellenmiştir(Şekil-7, Şekil-8). Müşteri etmeni kendisini sisteme kaydettirmek için *KayitIstegiGonder* planını işlettirir. Kayıt isteği TYE'ye ait *SirketKaydet* planını tetikler. TYE *KayitMesajiAl* ve *KayitIstegiDegerlendir* planlarını işleterek müşteri etmenin sisteme kaydeder veya istemediği bir durum söz konusu ise kayıt isteğini reddeder ve sonucu müşteri etmenine bildirir.

Bundan sonraki adımda müşteri etmeninin sahip olduğu bilgiler editör yardımı ile kanı olarak eklenir. Modelleme süreci tanımlandıktan sonra yukarıda anlatılan MOFScript ile geliştirilmiş modelden metine kod dönüştürücüsü ile etmen tanımlama dosyaları ve etmenlere ait plan kodları elde edilir. Şekil-9'da TYE için üretilen ETD yer almaktadır.



Şekil-8: TYE ait plan modeli

```
<agent xmlns="http://jadex.sourceforge.net/jadex"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://jadex.sourceforge.net/jadex
                           http://jadex.sourceforge.net/jadex-2.0.xsd"
                   name= "TYE"
                   package= "ege.edu.tr/BDI/TakasYonetimSistemi"
                   description= "Takas Yoneticisi Etmenini Temsil Eder"
                   propertyfile="jadex.config.runtime"
                   abstract="false">
    <goals>
      <achievegoal name="SirketKaydet" recur="false" description="" exclude="when_tried" exported="false" posttoall="false" randomselection="false" recalculate="true" recurdelay="0"
        <parameter name="istekMesaji" class="" description="" direction="in" dynamic="false" exported="false" optional="false" transient="false" updaterate="0">
        <creationcondition>
          <!-- Write Creation Condition -->
        </creationcondition>
        <contextcondition>
        <dropcondition>
        <recurcondition>
        <targetcondition>
        <failurecondition>
      </achievegoal>
      <performgoal name="SirketListele" recur="false" description="" exclude="when_tried" exported="false" posttoall="false" randomselection="false" recalculate="true" recurdelay="0"
    </goals>

    <plans>

      <plan name= "KayitMesajiAl" description="" exported="false" priority="0">

      <plan name= "KayitIstegiDegerlendir" description="" exported="false" priority="0">

      <plan name= "SirketKaydet" description="" exported="false" priority="0">

        <parameter name="sirketBilgisi" class="" description="" direction="receive" dynamic="false" exported="false" optional="false" transient="false" updaterate="0">

        <parameter name="kayitIstegi" class="" description="" direction="receive" dynamic="false" exported="false" optional="false" transient="false" updaterate="0">

        <body name="" description="" class="SirketKaydetPlan" inline="false" type="" language=""/>
        <trigger name="SirketKaydetTrigger" description="">
          <goal ref="SirketKayitIstegi"/>
        </trigger>
      </plan>
```

Şekil-9: Oluşturulan TYE ETD

5. İlgili Çalışmalar

Etmen tabanlı yazılım geliştiricileri etmen yazılımlarını daha etkin bir şekilde geliştirebilmek için model güdümlü geliştirimi uygun bir yöntem olarak görmüşlerdir. Çoklu etmen sistemlerinin model güdümlü geliştirilmesi kapsamında çeşitli çalışmalar bulunmaktadır. Bunlardan [6] ve [7] çoklu etmen sistemlerinin model güdümlü geliştirilmesi ile alakalı platforma özgü yapılan çalışmalardır. [8]'de tanıtılan uygulama çerçevesi etmen yazılımlarını elde etmek amacıyla tasarım düzeyini, geliştirim ayrıntılarından daha üst seviye olan alan düzeyine yükselterek geliştirmeyi sağlayan bir model güdümlü mimari öngörmektedir. [9]'da ise etmen tabanlı tasarım ve çoklu etmen sistemleri arasındaki etkileşimi tamamlamak için model güdümlü bir mimari önerilmektedir. Çoklu etmen sistemlerinin model güdümlü geliştirilmesi için yapılan başka bir çalışma ise Hahn ve ark.[10] tarafından gerçekleştirilmiştir. Bu çalışmada platformdan bağımsız bir etmen üst-modeli önerilmekte ve JADE ve JACK çoklu etmen sistemlerine özgü modeller üretilmektedir.

KİH etmenlerinin model güdümlü geliştirilmesi için Tropos[1] metodolojisini kullanan bir çalışma [11]'de yer almaktadır. Bu çalışmada model güdümlü gerçekleştirilen tasarım modelleri JADE ve JADEX kodlarına dönüştürülmektedir. JADEX için geliştirilmiş model güdümlü platforma özgü bir araç bulunmamaktadır. Yaptığımız çalışma ile JADEX'te KİH mimarisine uygun yazılım etmenleri çok daha hızlı ve doğru bir şekilde geliştirilebilmektedir.

[1] http://www.troposproject.org/ , son erişim: Nisan 2010

6. Sonuç ve İleriye Yönelik Çalışmalar

Bu bildiride KİH mimarisine dayalı yazılım etmenlerinin JADEX platformuna özgü geliştirilen görsel geliştirim aracı ile nasıl modellendiği ve kod üretiminin nasıl gerçekleştirildiği ayrıntılı bir şekilde anlatılmıştır. Bu sayede kullanıcılar JADEX platformunda yazılım etmeni geliştirmek için gerekli olan ETD'leri elle yazmaktan kurtulacak ve görsel modelleme aracı ile hızlı bir şekilde geliştirdikleri platforma özgü modelden otomatik kod dönüşümü ile ETD'leri elde etmiş olacaklardır. Modelden koda dönüşüm ile oluşturulan ETD JADEX'te geçerli ETD olarak kullanılması için sadece hedefler için tanımlanması gereken mantık tabanlı başarım ve başarısızlık koşullarının eklenmesi gerekmektedir. Geliştiriciler için gerekli olan ETD'nin yaklaşık %95'i elde edilmektedir. Üretilen plan kodları ise şablon seviyesindedir ve gerçekleştirim ayrıntılarının eklenmesi gerekmektedir.

Bildiride ayrıntılı bir şekilde anlatılan JADEX platformuna özgü modelleme aracının daha etkin bir araca dönüştürülmesi için çalışmalar devam etmektedir. Görsel olarak daha zengin bir hale getirilmesi düşünülen aracın yeni sürümlerinin çıkarılması ileriki çalışmalarımız arasında yer almaktadır.

7. Teşekkür

Bu çalışma Türkiye Bilimsel ve Teknik Araştırma Kurumu (TÜBİTAK) Elektrik, Elektronik ve Enformatik Araştırma Grubu (EEEAG) tarafından 108E141 no'lu proje kapsamında desteklenmektedir.

8. Kaynakça

[1] **Russell, S.**, **Norvig, P.**, 2003. Artificial Intelligence: A Modern Approach, http://aima.cs.berkeley.edu/ , Ders Kitabı ISBN: 0137903952.

[2] **Rao, A**. ve **Georgeff, M.**, 1995. BDI Agents: From Theory to Practice, First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, USA, ss. 312-319,.

[3] **Pokahr, A.**, **Braubach, L.**, **Walczak, A.** ve **Lamersdorf, W.**, 2007. Jadex - Engineering Goal-Oriented Agents, (Kitap Bölümü) ss.254-258, Developing Multi-Agent Systems with JADE, Wiley Publishing, West Sussex.

[4] **Kardaş, G.**, **Ekinci, E. E.**, **Afşar, B.**, **Topaloğlu, Yasemin.**, **Dikenelli, Oğuz.**, 2009. Ontoloji Tabanlı Çok-etmenli Sistemlerin Model Güdümlü Geliştirilmesi, IV. Ulusal Yazılım Mühendisliği Sempozyumu, İstanbul.

[5] **Cakirlar, I., Ekinci, E. E., Dikenelli, O.**, 2008. Exception Handling in Multi-Agent Systems. In: The 9th Annual International Workshop "Engineering Societies in the Agents World" (ESAW 08).

[6] **Perini, A., Susi, A.**, 2006, Automating Model Transformations in Agent-Oriented Modeling. In: Müller, J.P., Zambonelli, F. (eds.) AOSE 2005. LNCS, vol. 3950, ss. 167-178, Springer, Heidelberg.

[7] **Pavon, J., Gomez, J.,** 2006, Fuentes, R.: Model Driven Development of Multi-Agent Systems. In: Rensink, A., Warmer, J. (eds.) ECMDA-FA 2006. LNCS, vol. 4066, ss.284-298, Springer, Heidelberg.

[8] **Gracanin, D., Singh, H. L., Bohner, S. A. ve Hinchey, M. G.,** 2005, Model-Driven Architecture for Agent-Based Systems, *Lecture Notes in Artificial Intelligence*, 3228:249-261.

[9] **Amor, M., Fuentes, L. ve Vallecillo, A**., 2005, Bridging the Gap Between Agent-Oriented Design and Implementation Using MDA, *Lecture Notes in Computer Science*, 3382:93-108.

[10] **Hahn, C., Madrigal-Mora, C. ve Fischer, K.**, 2009 A platform-independent metamodel for multiagent systems, *Autonomous Agents and Multi-agent Systems*, 18(2):239-266.

[11] **Penserini, L., Perini, A., Susi, A., Morandini, M., Mylopoulos, J.**, 2007, A Design Framework for Generating BDI Agents, *AAMAS'07*, Honolulu, Hawai'i, USA.