ETSİS: Etmen Tabanlı Simülasyon Sistemi

Mehmet F. HOCAOĞLU

Agena Bilişim ve Savunma Teknolojileri Ltd. Simülasyon Sistemleri
hocaoglu@agenabst.com.tr

ÖZ

Çoklu programlama paradigmasını destekleyen Etmen tabanlı Simülasyon Sistemi tanımladığı durum tabanlı programlama paradigması ve ontolojik arka planı ile simülasyon modelleme ve etmen programlama dili olarak güçlü bir adaydır. Bildiride önerilen dilin tekrar kullanılabilirlik, esneklik, karşılıklı işletilebilirlik ve ortogonalite kriterlerini sağlamadaki başarısı, desteklediği paradigmalar ve genişletilen miras mekanizması, genişletilen davranış yönetimi ve değişken tabanlı senkronizasyon yapısı sunulmuştur. Önerilen bildirimsel betik dilin geliştirilen yeni özellikleri açıklanmış ve genel işletim kavramı bir örnek ile detaylandırılmıştır.

Anahtar Kelimeler: EtSiS, Etmen, Durum tabanlı Programlama, Paradigma, Simülasyon

AdSiF: Agent driven Simulation Framework

Abstract

Agent driven simulation framework (AdSiF) is a powerful candidate for agent programming and simulation modeling environment with multi-programming paradigm it supports, with state oriented programming paradigm it re-interprets and its ontological background. The success of AdSiF in satisfying software criteria such as reusability, extendibility, interoperability, flexibility and orthogonality, and software development paradigms Object Oriented Programming, Logic Programming, State Oriented Programming and Aspect Oriented Programming it supports are examined in the paper. An example is gives to show fundamental properties of the modeling approach and new properties discussed in the paper.

Keywords: AdSiF, Agent, Paradigm, Simulation, State oriented Programming

1. GİRİŞ

AdSiF genel amaçlı, multi-paradigma bildirimsel (declarative) betik (scripting) dildir [1]. Dilin tekrar kullanılabilirlik (reusability) ve karşılıklı işletilebilirlik (interoperability) konusundaki ana vurgusu, multi-paradigma desteği ve modellemecilere sağladığı etmen programlama yaklaşımıdır. AdSiF varlıkbilimsel (ontological (ontolojik)) betimlemesinde nesne yönelimli programlama paradigması (NYP), mantık programlama, etmen-tabanlı programlama paradigması [2, 3] ve cephe tabanlı (aspect oriented) programlama paradigmalarını tümleştirir. Tümleştirmeyi

durum tabanlı programlama olarak isimlendirdiği ve dünya betimlemesini, gramerini oluşturduğu bir paradigmada gerçekleştirir. AdSiF'in içerdiği paradigmalardaki zenginlik, sağladığı tekrar kullanılabilirlik, karşılıklı işletilebilirlik, esneklik özelliklerinin yanısıra dünya tasvirine de yansır. Herbir paradigmanın dünya tasvirinin tümleşimi etmen programlama yaklaşımında kendini gösterir. Yaklaşımda, davranış planlama karakteristiği ile karşılık veren (responsive) bir yapı, otonomi sağlanır ve sahip olduğu insan biçimsel (anthropomorphic) karakteristik ile geliştirilen uygulamaların ve simülasyon modellerinin daha sosyal, esnek ve karşılıklı işletilebilir (interoperable) olmalarını sağlar. Geliştirilen modelin çevresine ait bilgiye sahip olması ve muhakeme yürütmesi, mantık programlama paradigmanın bir getirisidir ve simülasyon modellerine "Dual dünya betimlemesi" kurmasını sağlar.

AdSiF yazılım mühendisliği kavramlarına sağladığı katkılardan farklı olarak, simülasyon teknolojisinde kesikli ve sürekli olay simülasyon teknolojilerinin işletim yaklaşımlarına yeni bakışlar ve işletim zamanlı geliştirme kavramını katmıştır. Sürekli olay simülasyonlarının kesikli uzaya resmedilme yaklaşımları, varlık durumsallığı standardizasyonu, işletim zamanlı davranış semantiği yönetimi ve varlıklar arası ilişkilendirme esasları önemli katkılar olarak ele alınmıştır. Tüm bunlara ek olarak AdSiF'in 1) Genişletilen miras mekanizması, 2) Genişletilen davranış yönetimi, 3) İlişki tanımı ile varlık ontolojisindeki kavramsal genişlemeyi ve 4) Davranış değişkenleri ile davranış senkronizasyonu bildiride, yeni özellikler olarak, ele alınmıştır.

1. AdSiF: MODELLEME VE SİMÜLASYON ÇERÇEVESİ

AdSiF sistem betimlenmesi için bildirimsel bir metot sunar. Temelde bir sistem zaman esasına, girdilere, durumlara, davranışlara ve sistem davranışlarını, diğer bir ifadeyle, sistem dinamik karakteristiğini yöneten bir mekanizmaya sahiptir. AdSiF varlıkbilim bakışıyla bir sistemin dinamik karakteristiği, içsel (internal) ve harici (external) durum geçişlerini, durum ve davranış eylem prosedürleri yönetimini ve ayrıca, durum ve davranışlarla temporal ilişkili davranışların yönetimini içerir.

AdSiF'in yazılım mühendisliği perspektifini oluşturan üç temel kavram *Yazılım Çerçevesi* (*Framework), betik dil* ve *Yazılım Etmenleri* olarak ele alınır. Bir yazılım çerçevesi olarak AdSiF bir seri yazılım tasarım kuralı seti getirir ve programlama paradigması ve ontolojik bakışı olan taslak bir tasarımı varsayar. AdSiF sahip olduğu multi-programlama paradigmasını durum tabanlı programlama paradigması bünyesinde tümleştirdiği için tasarım kurallarındaki ana odak bu paradigmadır. Literatürde durum tabanlı programlama kavramı ile modelin durumsallık tanımlamalarının yapılarak, durum tanımlamalarının programlanması veya durum tanımlamalarına uygun kod üretilmesi olarak ele alınmıştır [4]. Oysa burada, durumların programlanması değil, durumlarla modelleme ve programlamayı esas alan bir programlama ve altında yatan ontoloji önerilmektedir. Tasarımı yapılan simülasyon modelleri davranış diyagramları

genişletilmiş bir durum otomatı formatındadır ve doğrudan AdSiF tarafından işletilirler. Bölüm 7 Şekil 1'de küçük ölçekte bir simülasyon örneği için hazırlanmış davranış diyagramları görülmektedir ve bu diyagramların işletimi ile simülasyon yürütülmektedir. Takip eden bölüm, paradigmaların ortak bir ontoloji tanımı çevresinde nasıl tümleştirildiğine ışık tutar.

2. VARLIKBİLİMSEL (ONTOLOJİK) BAKIŞ

Aşağıdaki tanımlamada AdSiF ontolojik tanımlaması, ontolojiyi oluşturan bileşenlerin desteklediği paradigmalar ile ele alınmıştır.

"Varlıklar kendilerini diğer varlıklardan ayırt eden özniteliklere ve işletimleri ile öznitelik değer değişimlerini yöneten atomik fonksiyonlara sahiptirler (NYP). Varlıklar sahip oldukları fonksiyonları ardışık olarak, davranış olarak tanımlanan, belirli bir semantik yapı içerisinde yürütürler ve davranışlar paralel ve/veya seri olarak işletilirler. Modele ait her bir atomik fonksiyon durum tanımlamaları ile sarılırlar (Durum tabanlı Programlama –State Oriented Programming). Varlıklar birbirleri ile olay iletimleri ile etkileşirler ve birbirleri arasında ilişkilendirmeler kurulabilir. Birbirleri ile olan etkileşimlerinde amaç güdümü, otonomi ve reaktif davranışlar sergilerler (Etmen Tabanlı Programlama - AgOP). Kategorik bir bakış ile, varlıklar sahip olduğu davranış kategorileri ile farklı davranış bakışlarını temsil ederler (Cephe Tabanlı Programlama - Aspect Oriented Programming). Varlıklar bulundukları çevre ve çevreyi paylaştıkları varlıklarla ilgili inanışlara (believes) ve olgulara (facts) dual olgu dünya tasviri olarak sahiptirler. Dual dünya tasvirlerinde başaracakları bir seri amaçlara ve muhakeme mekanizmalarına ve karar yapım yordamlarına sahiptirler (Mantık Programlama-Logic Programming)." AdSiF ontolojik bakışı, NYP'nin dayandığı klasik Descartian varlık betimlemesini üç katmanda genişletir.

Nesne Betimlemesi; Klasik NYP'den en önemli ayırt edici farklılık, AdSiF model sınıf yapıları atomik fonksiyonlar arasında herhangi bir semantik ilişkilendirme içermemesidir. Sınıf fonksiyonları modelleme detayına göre belirlenen ve daha alt adımlara ayrılamayan atomik düzeyde eylemler olarak modellenirler. Sınıf fonksiyonları oldukça basit bir parametre setine sahiptirler ve sıklıkla parametresiz kullanılırlar, boş veya bool geri dönüş değerine sahiptirler. Metot parametreleri yerine davranışı tetikleyen olaya ait parametreler veya öznitelikler kullanılır. Bu bakış açısıyla, AdSiF'de programlama yazılım etmeni geliştirme ile benzerlikler taşır [3].

Davranış karakteristiği; Atomik fonksiyonların durumlar ile sarılması durum tabanlı programlamaya doğru ilk adımdır. Varlıklar ilgili durum içerisinde tanımlı süreyi geçirirler ve durum tarafından sarılan eylemleri yürütürler. AdSiF varlık betimlemesinde, herhangi bir zaman anında, varlık en az bir durum içerisinde bulunur. Varlıkların davranış semantiği, ardışık durum işletimlerinin gerçekleştirildiği, durum otomatı formundaki davranışlarla modellenir. Her bir davranış kuruluş amacına uygun,

mantıklı bir sonuç üretir. Üretilen sonuç varlık çevresini ve çevreyi paylaşan diğer varlıkları etkileyen, onlarda bir davranış başlatan veya durumlarını değiştiren bir sonuç olarak görülür.

Mantıksal perspektif; Varlıklar çevresi ve çevredeki varlıklarla ilişkin oldukları bilgiler ile oluşturdukları içsel "dual dünya tasvirine (fact equivalent dual world)" sahiptir. Varlıklar çevrelerinde oluşan olayları ve diğer varlıkları algılarlar, varlıklara ve oluşan olaylara reaksiyon göstermelerinin ötesinde, edindiği çevresel varlıklara ilişkin olguları güncel olarak oluşturdukları içsel modellerinde tutarlar. Olgular yalnızca güncel değerleri değil, olguların tarihsel akışını gösteren önceki değerlerini, modelleyici tercihine bağlı olarak, muhafaza edebilirler. Söz konusu olgular çevre ve içerdiği varlıkların sahip oldukları durumlar, olaylar ve öznitelikleri ile hesaplama sonucu üretilen değerleri üzerinde kurgulanır. Karar yapım algoritmaları, mantık programlama söyleyişiyle, önermeler, olguları muhakeme amaçlı girdi olarak kullanırlar ve çevre, varlıklar ve durumlar için bir karar sonucu üretirler. Muhakeme, doğruluk değeri ve önermelerin sahip oldukları çıktı ("out") parametrelerini üretir. Üretilen kararlar ve parametreler, varlık için bir davranış tetikleyici sonuç olarak durum tabanlı programlama davranış yapıları ile tümleştirilirler.

3. Programlama Paradigmaları

Takip eden bölümlerde AdSiF'in varlık betimlemesini oluşturan paradigmalar ve paradigmaların tümleşik varlık betimlemesine sağladığı katkı ile durum tabanlı programlamadaki tümleştirilmeleri ele alınmıştır.

3.1.Cephe Tabanlı Programlama Bakışı

Bir varlığın sahip olduğu davranışlar modelleyicinin beklentilerine göre kategorize edilirler. Bu bakış, davranış perspektifi ile saçılmış (scattered) isterlerin model davranışlarına dağıtılmasını ve tümleşik bir ister setinin (tangled) sınıflandırılmasını sağlar. Semantik olarak bir birinden ayrık olan davranışların farklı kategoriler içerisine taşınması, modelleyiciye modellerin farklı davranış cephelerine (aspects) göre yönetilmesini sağlar. Ayrıca, saçılmış isterleri karşılayan davranışların veya durumların, sırasıyla, davranış setlerine ve davranışlara dağıtımı düşük-bağımlılık ve yüksek tutarlılık ile saçılmış ister problemini çözer. Bölüm 7'da verilen örnekte davranışlara loglama işlevini gerçekleştiren bir durumun eklenmesi veya davranış kümesine bu amacı sağlayan bir davranışın eklenmesi, model semantiğini bozmadan saçılmış bir isterin karşılanmasında bir çözüm olarak kullanılabilir. Daha genel bir örnek olarak, bir simülatörün iki farklı çalışma modu olduğunu varsayalım. Burada iki farklı simülatör modellemek yerine, modlar arasındaki davranış farklılıkları davranış kategorileri olarak belirlenir. Farklı davranış kategorisinde, yürütülen davranışların hesaplama prosedürleri ortak veya benzer olanları bir tek model sınıfında modellenebilirler. Yaklaşım

simülatörden beklenen fonksiyonların davranış seti olarak ayrıştırılması ile *yığın şeklindeki isterlerin* karşılandığı bir cephe programlama çözümüdür.

AdSiF'in Cephe tabanlı Programlamaya olan bir diğer önemli desteği Geç-Yüklemeli-Atomik-fonksiyon uzantılarıdır (Delayed-Loaded Atomic Function Plug-ins (D_lAFPs). D_lAFPs davranış durumları tarafından işletilen atomik fonksiyonun işletim zamanında yüklenmesini sağlar. Özellik, varlıklara işletim zamanında dahi geliştirme, modeli genişletme imkanı verir ve AdSiF'in genişletilebilirlik derecesi için önemli bir ölçüdür.

3.2.Mantık Programlama Bakışı

Mantık programlama "horn ifadeleri (horn clause)" ifadelerin koleksiyonu olarak tanımlanır [5]. Mantık programlamaya nesne tabanlı programlama bakışı ile, mantık nesneleri "hakkında doğru olarak neleri bildiğimiz ile ilgileniriz" [6]. AdSiF durum tabanlı programlama paradigması ile olan tümleşimindeki ana fikir, mantık nesnelerinin doğruluk değerleri ve muhakeme sonucu elde edilen parametreler ile davranışların yönetilmesidir. Mantık nesneleri muhakeme operasyonlarında çevre ve diğer nesnelere ait olguların ve önermelerin bulunduğu bilgi tabanını kullanır. Mantık programlama ile varlıkların çevre ve çevreyi paylaştıkları varlıklar hakkında sahip oldukları bilgiler (olgular) doğrudan veya dolaylı etkileşimler ile güncellenir. Olgular zaman etiketli olarak güncel bilgi veya zaman çerçeveli bilgi olarak tutulurlar. Elde edilen olgular üzerinde muhakeme yapan önermelerden elde edilen doğruluk değerleri ve parametreler davranışları tetiklemekte kullanılır. Simülasyonda çözüm için geçmiş değerlere matematiksel bağımlılık olan problemlerde [9, 10] ve yapay zeka literatüründe çerçeve (frame) problemi [11] olarak bilinen problemler için çözüm adayıdır.

3.3.Etmen Karakteri

Etmen-tabanlı programlama klasik hesaplama yaklaşımlarından iki noktada farklılık göstermiştir. İlki, klasik yaklaşımın öngördüğü şekilde yazılım tekil bir çözüm geliştirici olmaktan ziyade, farklı varlıklarla etkileşebildiği bir topluluk içerisinde yer alıyor olmasıdır. İkincisinde ise, hesaplama prosedürleri rutin bir algoritmik işletimin ötesinde, hesaplama prosedürleri bir amaç içerir ve duygusal özellikler olarak tanımlanabilecek özelliklere de sahiptir [7]. Bu etmenlerin bir işlemi yerine getirirken uyguladıkları stratejiler amaç ve amacı uygulamadaki duygusal karakteristiğini gösterir [8]. AdSiF amaçları ve ilgili stratejileri mantık programlama ve davranış yapılarını daha üst düzey programlama ile tümleştirerek gerçekleştirir. Varlığa ait her bir davranışın anlamlı bir sonuç ürettiği dikkate alınırsa, davranışlar varlığın eylemlerini ifade eden cümle yapıları olarak tanımlanabilir. Mantık programlama etmen için amaç modellemesi ve çelişki çözümlemesi işlevlerini gerçekleştirir [9] ve amacın işletimini, varlıkların davranışlarını kelime olarak kullanan, daha üst düzey davranış cümlelerini yöneterek yürütür. Bir etmenin amacının olması, etmen yazılımlarını yalnızca bir insan benzeri yazılım olmaktan öte, bir işi olan insan benzeri olmalarını sağlar. Amaç güdümü

ve strateji uygulama, etmenleri klasik yazılımlardan daha fazla insan benzeri davranışa yaklaştırır ve otonomi, reaktif olma özellikleri bu özelliği güçlendirir. Klasik yaklaşım ile geliştirilen bir yazılım, tanımlı prosedürleri takip ederek işletimi sonlandırır, bir etmen bir çevre içerisinde bulunduğu için çevresi var olduğu sürece, yürütecek bir amacı bulunmasa veya başarılmış olsa bile, kendisi için tanımlı olan olaylara ve durum değişimlerine reaksiyon gösterir. Çevresinde oluşan olaylara veya durum değişimlerine gösterilen reaksiyon davranış olarak modellenir ve davranış işletimi sonucunda durumsal farklılık (dolaylı etkileşim) ve/veya olay yayınlaması (doğrudan etkileşim) gerçeklenir.

4. AdSiF Varlık İlişkileri

AdSiF'in ontolojisinin bir parçası olarak varlıklar aralarında tasarım zamanlı ilişki ve işletim zamanlı ilişki olmak üzere iki tip ilişkilendirme mevcuttur. Tasarım zamanlı ilişkilendirme, modeller arası miras (inheritance), kümeleme (aggregation) ve birleştirme (composition) ilişkileri olarak tanımlanır. Kümeleme ve birleştirme ilişkisinde, alt modellere bileşen olarak sahip olan ana model, alt varlıkların zaman ve olay yönetimini üstlenir.

İşletim zamanlı ilişkilendirme işletim zamanında etkindir. Modellerin tasarımlarında doğrudan bir ilişkilendirme söz konusu değildir. İlişki tanımının sağında ve solunda yer alan varlıkların her biri için ilişkinin aktif olması ve pasif olması durumunda yürütecekleri davranışlar tanımlanır. Örnek olarak; Bölüm 7'da verilen depo örneğinde yer alan ilişkilendirme dikkate alınırsa, Depo modelinin iki adet musluk modeli ile "Doldurur" ve "Boşaltır" ilişkileri ile ilişkilendirildiği görülmektedir ("<relation.left="Musluk".right="Depo.name="Doldurur"/>"). *Musluk* ilişkilendirmenin solunda, *Depo* ise sağında yer almaktadır. İlişki aktif olduğunda *Musluk* modeli ilişkinin sağında yer alan *Depo* modeline su aktarma olayı gönderdiği ve *Depo* modelinin *Musluk* modeline açma, kapama olayı gönderdiği örnekte görülmektedir.

5. DURUM TABANLI PROGRAMLAMA PARADİGMASI ELEMANLARI

Paradigma AdSiF'in içerdiği tüm paradigmaları bünyesinde kaynaştıran özgün bir paradigma olması sebebiyle, alt bölümlerde paradigmanın temel elemanları kısaca anlatılmıştır.

5.1.Davranışlar ve Davranış Setleri

En yalın anlamı ile bir davranışın işletilmesi, ardışık olarak sıralanmış durumların işletimidir. Durum işletimi duruma atanan sürenin harcanması ve duruma atanan atomik metotların yürütülmesidir. Durum ve davranışların her ikisi de, işletiminin doğal akışı içerisinde, ilgili tüm fazlarda, temporal ilişkili davranış yönetimleri gerçekleştirilir ve akışın tamamlanması durumunda kendilerine ilişkilendirilmiş olayları ilgili adreslere

gönderirler. Bir davranışın aktif olma (Active), iptal edilme (Canceled), askıya alınma (suspended), tekrar işletimi (askıya alınmış bir davranış için) (resumed) ve sonlandırılması (finished) olarak tanımlanan durumsal tanımlayıcıları vardır.

Bir davranış üç farklı şekilde aktive edilir. Bunlar;

- ***Olay ile aktivasyon;*** Tanımında "*S"* giriş durumuna ve "*e"* tetikleyici olayına sahip bir davranış, yürütülmekte olan herhangi bir davranışın "S" durumu içerisinde olduğu anda aldığı "*e"* olayı ile aktive edilir.

- ***Güdüm şartı ile aktivasyon;*** Güdüm şartı, bir davranışın olay almaksızın, tanımlı aktivasyon şartının sağlanması ile aktive edilmesidir. Şart tanımları ile aktif bir davranışın iptal edilmesi, askıya alınması, sonlandırılması ve askıya alınmış bir davranışın tekrar aktive edilmesi gerçekleştirilebilir.

- ***Temporal ilişkili aktivasyon;*** Bu durumda davranış, temporal ilişkili olduğu diğer bir davranış veya durum tarafından aktive edilir. Aktivasyon için kısıt ve gecikme süresi tanımlanabilir. İlişkilendirme davranış ve durumun fazları ile aktive edilecek davranış arasında yapılır (5.3 Sınıflar, Öznitelikler, Temporal İlişkiler).

Davranış setleri, Cephe Tabanlı Programlama bölümünde değinildiği gibi, modelin farklı cephelerden bakışını oluşturan davranışları kategorik olarak düzenleyen ve ayrık setler (kümeler) içerisinde toplayan bir yapıdır. Davranış setleri aktif olma ve pasif olma şartlarına sahiptir. Aktif olma şartının işletim zamanında sağlanması durumunda içerdiği davranışlar model tarafından kullanılabilir.

5.2.Durumlar

Durum modelin içerisinde belirli bir zaman için bulunduğu tanımlı bir safhadır. Modeller, tasarım çözünürlüğüne bağlı olarak, bir durumunda iken durumun farklı fazlarında tanımlanmış atomik fonksiyonlarını işletirler. Durumların operasyonel tanımlayıcıları; **a) Giriş şartı;** Giriş şartı bir duruma girişin gerçekleşip gerçekleşmeyeceğine karar veren şart tanımıdır, **b) Durum eylemleri;** Modelin ilgili durumun tanımlı fazlarında yürüteceği fonksiyonların tanımıdır. Bir durum sahip olduğu üç fazı için model atomik fonksiyonlarına ait referanslara sahiptir. Birincisi, Durum-eylem-fonksiyonu olarak tanımlanır ve duruma giriş fazında işletilir. İkincisi çıkış-faz-fonksiyonu olarak tanımlanır ve durumdan içsel geçiş fazında işletilir ve sonuncusu harici-geçiş-fonksiyonu olarak tanımlanır ve durumun harici geçişi esnasında işletilir. İçsel geçiş varlığın içerisinde bulunduğu duruma atanan zamanı harcadıktan sonra davranışın takip eden durumuna geçişi olarak tanımlanır. Dışsal geçiş modelin mevcut durumdan çıkışına durum zamanının tükenmesi değil, modelin aldığı bir olay sebebiyle gerçekleşir. **c) Olaylar;** Bir duruma/davranışa diğer modellere veya kendisine gönderilmek üzere olay ilişkilendirilebilir. Bu, durumun yerine getirdiği eylemden sonra diğer modellerden yapmasını beklediği eylemleri tetikleyen bir mesajlaşma, bir

bilgilendirme olabileceği gibi, bir eylem isteği de olabilir [6]. **d) Zaman hesaplayıcıları;** Süre hesaplayıcıları tanımlandığı durum için durumda geçirilecek süre hesabını yapar. **e) Koşum izi etmenleri;** Koşum sırasında kayıt altına alınacak koşum yörüngesi değişkenlerinin tanımlanması sağlanır. Tanımlaması yapılan bir koşum izi etmeni işletileceği durum tanımında belirtilir. Koşum izi etmenlerinin model kodları içerisine gömülü olmaması, etmenlerin model kodları ile olan yüksek-tutarlıklı-düşük-bağımlılığa sahip bir tasarıma sahip olduğunu gösterir.

5.3. Sınıflar, Öznitelikler, Temporal İlişkiler ve Miras Mekanizması

AdSiF'e ait iki tip soyut model sınıfı mevcuttur. Bunlar; AdSiFModelEntity sınıfından türeyen model ve AdSiFModelBuilder sınıfından türeyen model builder sınıflarıdır. Bir simülasyon modeli zorunlu olarak bu iki kök sınıftan türeyen sınıfa sahip olmalıdır. AdSiFModelEntity sınıfı simülasyon işletim zamanında kullanılan modele ait temel sınıfı ve AdSiFModelBuilder sınıfından türetilen sınıf ise simülasyon senaryosu tasarımında kullanılan model sınıfıdır.

AdSiF iki tip öznitelik kullanım konseptini barındırır. Bunlardan birincisi, NYP teknolojisinde yer alan ve sınıf içerisinde tanımlı öznitelik kullanımıdır. İkinci kullanım konsepti ise, öznitelikler nesne olarak tanımlanır. Tanımlamada öznitelik tanımları bir şablon tanımlama ile gerçekleştirilir. Şablon dinamik bir karakteristiğe sahip olup tasarımcı tarafından belirlenen farklı öznitelik veri alanları ile veri yapısı genişletilebilir.

Temporal ilişkiler davranışlardan davranışlara ve durumlardan davranışlara ilişkilendirmeyi sağlar. İlişkilendirme durum ve davranışların durumsal belirleyicileri üzerinden yapılır. Durumdan davranışa yapılan bir tanımlama durumun giriş ve çıkış fazları için, davranıştan yapılan bir ilişkilendirme ise davranış durumsal tanımlayıcıları ile yapılır ve ilişkilendirilen davranışın, aktive edilmesi, askıya alınması, sonlandırılması ve tekrar işletimi sağlanır. Temporal ilişkilendirme bir uygulama şartı ve uygulama gecikme süresi içerir.

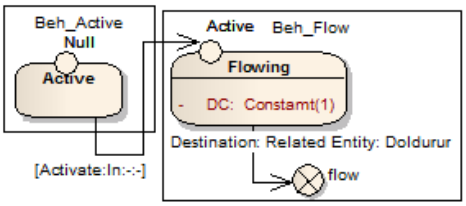
Miras ilişkisi modeller arasında tanımlanabileceği gibi [1], durumlar ve davranışlar arasında da tanımlanabilir. Durumlar arasında yapılan miras ilişkisinde temel durum için tanımlanan; atomik fonksiyonlar, ilişkilendirilen olaylar, giriş şartı ve temporal ilişkiler türetilen durum tarafından alınır. Davranışlar arasında tanımlanan miras ilişkisinde davranış durum tanımlamaları dışındaki tanımlamalar türetilen davranış tarafından alınır. Model davranış setleri (kümeleri) arasında yapılan miras ilişkisinde türetilen davranış seti temel setten içerdiği davranışları alır. Modeller arasında kurulan miras ilişkisinde türetilmiş varlık, temel varlıktan öznitelikleri, koşum izi etmenlerini, öznitelik aboneliklerini (subscription set), durum tanımlamalarını, mantıksal ifade tanımlarını ve davranışları alır.

6. DEĞİŞKEN İLE DAVRANIŞ SENKRONİZASYONU

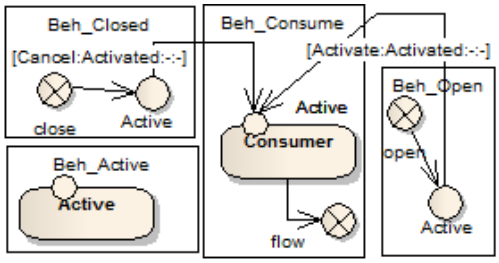
Modeller davranış ve model etki sahalı (scope) değişkenlere sahiptirler. Birden fazla davranış tarafından kullanılan bir değişkenin, davranışın herhangi bir noktasında kullanımında, ilgili davranış, değişkene değer atanıncaya kadar askıya alınır. Örnek olarak, bir X değişkeni davranışa ait bir Durumun giriş şartı tanımlamasında X>5 şeklinde bir mantıksal ifade ile kullanılmış olsun. Eğer X değişkenine bir değer ataması yapılmamış ise X değişkeni ile hesaplama yapacak davranış, X değişkenine, bir başka davranış tarafından değer atanıncaya kadar askıda bekler.

7. SİMÜLASYON ÖRNEĞİ: SU DEPOSU SİMÜLASYONU

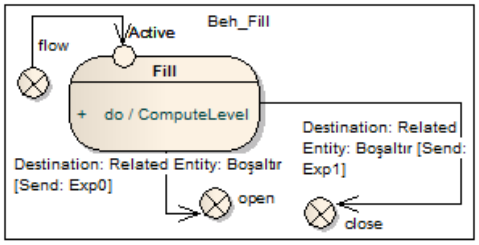
Örneğimizde, su akışı sağlayan bir musluk ile doldurulan ve maksimum seviyesine yaklaşıldığında tahliye vanasını açan bir sistem simüle edilmektedir. Simülasyonda depo ve iki adet musluk mevcuttur. Şekil 1'de modellere ait davranışlar ve modeller arası ilişkilendirmeler görülmektedir. Musluk modeli için iki farklı davranış seti tanımlanmıştır (doldurma davranış seti ve boşaltma davranış seti). Doldurma ve boşaltma muslukları için bunlardan uygun olanlar aktif olarak seçilirler. Bu önceki bölümlerde ifade edilen davranış listelerinin kullanım konseptini göstermektedir. Depo modelinin sahip olduğu ve ifadelerden örneği açıklamaya yetecek öznitelikler şu şekildedir.



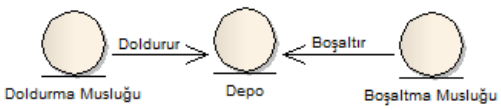
(a) Musluk Modeli Dolum Davranış Diyagramı



(c) Musluk Modeli Boşaltım Davranış Diyagramı



(b) Depo Modeli Davranış Diyagramı



(d) Modeller Arası İlişkiler

Şekil 1 : Depo Simülasyonu

waterLevel; Tamsayı tipinde bir özniteliktir ve depodaki su seviyesini gösterir.

maxLevel; Tamsayı tipinde bir özniteliktir ve en yüksek su seviyesini temsil eder.

Simülasyon modelinde tanımlı olaylar şu şekildedir.

open; Musluğun açılma davranışını tetikler.

close; Musluğun kapanma davranışını tetikler.

flow; Depoya musluktan su akışını belirtir. Debi isimli, akış miktarını temsil eden integer tipinde bir parametreye sahiptir. Negatif değer boşaltımı ifade eder.

Depo modelinin sahip olduğu mantıksal ifadeler aşağıdaki şekilde tanımlanmıştır.

Exp0; waterLevel>=maxLevel*0,9 şeklinde formüle edilir ve su seviyesinin maksimum su seviyesinin %90'ından büyük olması durumunu ifade eder.

Exp1; wateLevel<=maxLevel*.1 şeklinde formüle edilir ve su seviyesinin maksimum su seviyesinin %10'unun altına inmesi durumunu ifade eder.

Musluklardan birincisi (M1) d_0 m^3/dk debisi ile depoyu su ile doldurmaktadır. Şekil 1 (a)'da görülen "Beh_Active" isimli davranış başlangıç (initial) davranış olarak belirlenmiştir ve simülasyonun t0 zaman anında active edilir. Davranış temporal ilişkili olan "Beh_Flow" isimli davranışı, "Flowing" durumu giriş fazında sıfır zaman sonra active eder. "Beh_Flow" davranışı döngüsel bir davranış olarak tanımlanmıştır. Davranış "Flowing" durumunda, duruma ait zaman hesaplayıcısının belirttiği bir zaman birimi harcar ve davranış her durum çıkışından sonra yinelenir. "Flowing" durumu çıkışında "flow" olayı hedefi "Doldurur" ilişkisi ile ilişkilendirilmiş model olan "Depo" modeline gönderir. Depo modeli başlangıç davranışı "Beh_Active" olarak belirlenmiştir. "flow" olayı Depo modelinde "Beh_Fill" davranışını aktive eder. Durum tarafından "ComputeLevel" metodu yürütülür. Metot "flow" olayı ile alınan debi parametresindeki değer ile akış süresi kullanılarak dolan su miktarı hesaplar ve bulunan değer depo modeline ait "waterLevel" özniteliğine atanır. "Fill" durumu çıkışında "Exp0" mantıksal ifadesinin doğru değerine sahip olması durumunda, "open", "Exp1" mantıksal ifadesinin doğru değerine sahip olması durumunda ise "close" isimli olay Depo modelinin "Boşaltır" ilişkisi ile ilişkilendirildiği modele gönderilir (boşaltma musluğu). Boşaltma musluğu başlangıç davranışı "Beh_Active" olarak tanımlıdır. Model "Active" durumundadır. "open" olayının alınması ile tetiklenen "Beh_Open" davranışı temporal ilişkili olduğu "Beh_Consume" davranışı davranışın aktivasyon fazında aktive eder. "Beh_Consume" davranışı "Consumer" durumunda *DC:Constant(1)* ile hesapladığı zaman adımı kadar süre harcar ve süre sonunda durum çıkışında "Boşaltır" ilişkisi ile bağlı olduğu modele "flow" olayını gönderir. Depo modeli tarafından alınan olay ile tekrar "Beh_Fill" davranışı tetiklenir. Davranış "Fill" durumu ile "flow" olayından aldığı debi parametresi ile *ComputeLevel* metodunu çalıştırır ve hesaplanan değeri "waterLevel" özniteliğine atar (negatif debi değeri alınmıştır). Durum çıkışında olay gönderimleri bir önceki durum gibi ele alınır. "close" olayının gönderildiği durumda, olayı alan boşaltma musluğu "Beh_Closed" davranışını işletir. Davranış temporal ilişkili "Beh_Consume" davranışını iptal eder ve musluk kapatılır.

Simülasyon, muslukların debi oranlarına bağlı olarak simülasyon bir kararlılık noktasına ulaşır. d0=d1 olması durumunda sistem maksimum su seviyesinde kararlılığa ulaşır. d0>d1 olması durumunda depo taşar ve d0<d1 olması durumunda ise deponun dolup minimum seviyeye kadar boşaltılıp tekrar maksimum seviyeye yükseldiği bir simülasyon çevrimi oluşur. Bir karar problemi olarak, doldurma musluğunda meydana gelecek birim zamandaki n katlık bir debi artışının, alarm durumu olduğunu ve tahliye vanası açıklığının maksimum seviyeye ulaştırılması ile karşılık verileceğini düşünelim. Bölüm 3.3'de ifade edildiği şekilde, simülasyona AdSiF'in simülasyon teknolojilerine katmış olduğu, mantık programlama yeniliği kullanılarak, *debi(miktar, time)* şeklinde bir olgu tanımlaması yapalım. Bir karar algoritması olarak aşağıdaki gibi tanımlanır.

*openOutValve(X,N,T):-debi(M0,T0),debi(M1,T1), T1-T0>T, M1/M0>N, X is N*1.5.*

İfade şu şekilde yorumlanır; Verilen bir T zaman aralığında, debi N katından daha fazla artmış ise çıkış vanası X kat açılır ve X bu oranın 1.5 katı olarak belirlenir. Önerme bir vana açma/kapatma davranışının aktivasyon şartı olarak tanımlanır ve önermenin doğru değer döndürmesi durumunda davranış aktive edilir. Önermenin geri dönüş değeri X, davranış tarafından parametre olarak kullanılır. Vana açış davranış durum tanımlamaları sıfır zamanlı olarak tanımlanır. Sıfır zamanlı durumların işletilebiliyor olması AdSiF'in simülasyon işletimlerine kattığı diğer bir yeniliktir ve kaynağı önerdiği durum tabanlı programlama paradigmasıdır.

8. SONUÇ

AdSiF simülasyon ve etmen programlama uygulamaları için güçlü bir geliştirme ortamı sunmaktadır. Aşağıda belirtilen yazılım mühendisliği ve simülasyon alanında kullanım kolaylıkları sunmaktadır. Bunlar; 1) Model davranış semantiğinin dışsal veri olarak giriliyor olması, model karmaşıklığını azaltmaktadır. Model davranışlarında yapılacak değişiklikler kodun yeniden derlenmesini gerektirmemektedir. Alternatif yaklaşımlar, sıklıkla model davranış semantiğini modelin derlenen kodunun bir parçası olarak muhafaza etmektedir, 2) Özniteliklerin dışsal veri olarak tanımlanma imkanının, DlAFPs özelliği ile birleştirildiğinde modelin geliştirme sürecinde işletim zamanına kadar uzanan bir esneklik kazanılmaktadır. Alternatif yaklaşımlardan önemli bir fark sağlayan özellik, modelin kaynak kodunu değiştirmeden ve hatta yeniden derlemeden, yeni özellikler katılarak genişletilebilme avantajını sağlar, 3) Simülasyon işletim algoritmalarının davranışlar ile modellenmiş olması, sistemin ileriye dönük genişletmelere açık olma sonucunu doğurmaktadır. Bu özelliği ile yazılım mühendisliği ortagonalite kriterini sağlar, 4) Soyut davranış yorumlayıcı mekanizma sistemi simülasyon dışı uygulama alanlarına da taşımaktadır. Alternatif sistemler bir hesaplama ortamı sağlamaktan çok bir simülasyon geliştirme ve işletim ortamıdır, 5) Simülasyon modellerinin mantık programlama yaklaşımı ile simülasyon modellerinin çevrelerine ilişkin dual dünya betimlemesine sahip olmasını ve muhakeme yapabilmesini sağlar.

Yapay zeka ile tümleştiren ve etmen tabanlı programlama yaklaşımı ile birleştiren özellik, alternatif sistemlerle kıyaslandığında AdSiF tarafından yeni bir özellik olarak sunulmaktadır ve mühendislikte çerçeve (frame) probleminin çözümü için teknik altyapı sağlar, 6) Durum programlama ile sıfır zamanlı simülasyon ilerlemesi, olay listeleri tabanlı simülasyon yaklaşımından farklı olarak, mümkündür, 7) Model kodundan bağımsız tanımlı koşum kayıt etmenleri ile analiz koşum kayıtları alınmasında esneklik sağlar.

AdSiF kesikli ve sürekli olay simülasyonlarını destekler ve etmen programlama imkanı sunarak etmen güdümlü simülasyon ortamı sunar. Davranışlar ve mantık programlamada kullanılan önermeler, doğal dile yakın bir okunma özelliği ile model geliştirme sürecini karmaşıklıktan korur ve izlenebilirliğini artırır. Büyük hacimli simülasyon projelerinde izlenebilirlik takibini model davranışları üzerinden yapılmasını sağlar.

9. KAYNAKÇA

[1] Hocaoğlu, M. F., "AdSiF: Agent Driven Simulation Framework", The Huntsville Simulation Conference 2005, 26-27 October 2005.

[2] Shoham, Y. Agent-Oriented Programming (Technical Report STAN-CS-90-1335). Stanford University: Computer Science Department, 1990.

[3] Shoham, Y. Agent-Oriented Programming. Artificial Intelligence. pp. 51-92. http://www.cs.unb.ca/~ulieru/Teaching/CS6705/Shoham.pdf. Retrieved 2009-06-01., 1993.

[4] Nomoto, H., "State Oriented Programming", Proceedings of the Eighth IEEE International Symposium on High Assurance Sistems Engineering (HASE'04), 2004

[5] Bhattacharya, A., Konar, A., Mandal, A. K., "Parallel and Distributed Logic Programming", Springer, studies in computational intelligence, vol.24, 2006.

[6] McCabe F. G., "Logic and Objects", Prentice Hall, 1992.

[7] Travers, M. D., "Programming with Agents:New metaphors for thinking about computation" Massachusetts Institute of Technology, Doctoral dissertation, 1996.

[8] Wooldridge, M., "An introduction to MultiAgent Sistems", John Wiley & Sons, Ltd, 2002.

[9] Murray-Smith, D. J., "Continuous Sistem Simulation" Chapman & Hall, 1995.

[10] Zeigler, B. P, Praehofer, H., Kim, T. G., "Theory of Modeling and Simulation", Second Edition, Academic Press, Florida, (2000).

[11] J. McCarthy and P. J. Hayes (1969). Some philosophical problems from the standpoint of artificial intelligence. Machine Intelligence, 4:463-502.