ETMEN PROGRAMLAMADA CEPHE TABANLI PROGRAMLAMA BAKIŞI

M. Fatih Hocaoğlu

İstanbul Medeniyet Üniversitesi, mfatih.hocaoglu@medeniyet.edu.tr Kadıköy/İstanbul

Agena Bilişim ve Savunma Teknolojileri, hocaoglu@agenabst.com

ÖZ

Etmen tabanlı simülasyon sistemi, EtSiS, bildirimsel betik simülasyon ve etmen programlama dili sunar. Genişletilmiş durum grafiği diyagramlarının işletimi ile uygulama geliştirilmesine imkan veren betik yapının paradigma arkaplanında çoklu programlama paradigmaları ile tümleştirilmiş durum tabanlı programlama paradigması vardır. Cephe tabanlı programlama paradigması durum tabanlı programlama ile tümleştirilen paradigmalar arasında, saçılmış kodları enazlayan, saçılmış isterlere ve kümelenmiş isterlere çözüm geliştirilmesine ilişkin paradigma arka planını sunar.

Nesne yönelimli tasarım düşey derinlikte bir yazılım tasarımı sunarken, cephe tabanlı programlama, düşey eksende derinleşen nesne yönelimli tasarımlara yatay eksende bir ilişkilendirme sağlar. Bu ilişkilendirme ile saçılmış ve çapraz kesen isterleri (cross-cutting requirements) ve kümelenmiş isterleri sağlayan tasarımlar elde edilir. EtSiS'in durum tabanlı programlama paradigmasının bileşeni olan cephe programlama paradigması, zayıf yazılım bağımlı betik tanımlamalar ile bu isterlere cevap verir. Bu amaçla, saçılmış isterleri sağlayan durum tanımlarının davranışlara ve davranış tanımlarının davranış listelerine dağıtımı, yazılım bağımlılığı oluşturulmaksızın, gerçekleştirilir. Çözüm bunların ötesinde, simülasyon ve etmen programlama dünyasında farklı kavramsal dünya tasvirleri için işletim zamanında şartlara bağlı olarak, modelleme cephesi değiştirerek, farklı modelleme isterlerine veya farklı ister kümelerine çözüm üretir.

Geliştirilen çözüm, cephe programlama paradigmasını, yazılım tasarımından işletim zamanına, dinamik olarak yönetilen bir yapı olarak sunar ve esnek, zayıf-bağımlılığa ve yüksek tutarlılığa sahip tasarım sağlar.

Anahtar Kelimeler: Betik programlama, Bildirimsel dil, Cephe tabanlı programlama, Dinamik cephe, EtSiS

ASPECT ORIENTED PROGRAMMING PERSPECTIVE IN AGENT PROGRAMMING

ABSTRACT

Agent driven Simulation Framework, AdSiF provides a declarative scripting agent programming language. State oriented programming paradigm combined with multi-programming paradigms is at the background of script, which allows programming by extended state charts. Aspect oriented programming paradigm draws a solution background related with scattered codes, scattered requirements and tangled requirements.

While object oriented programming paradigm gives a vertical software design, aspect orientation enhances this vertically deep design by horizontal association. By the association, software design that satisfies cross-cutting requirements and tangled requirements is got. Aspect oriented programming paradigm, which is one of the components of AdSiF's state oriented programming paradigm, provides a solution by a loosely-coupled script description to the problems mentioned. For this purpose, state descriptions to behaviors and behaviors to behavior containers satisfying scattered requirements are distributed without arising any software dependency. Furthermore, in simulation and agent programming world, the solution provides a solution by shifting modeling aspects conditionally for conceptually different modeling requirements and tangled requirements.

The solution carries aspect oriented programming from design time to execution time and provides a dynamically manageable, and flexible, loosely coupled and high coherent design.

Keywords: AdSiF, Aspect oriented programming, Declarative language, Dynamic aspect, Script programming

1. GİRİŞ

EtSiS genel amaçlı, bildirimsel (declarative) betik (scripting) dildir ve çoklu programlama paradigmasını durum tabanlı programlama paradigması ile tümleştiren bir yapıya sahiptir [1], [2]. EtSiS varlık bilimsel (ontological) betimlemesinde, nesne yönelimli programlama paradigması (NYP), mantık programlama, etmen-tabanlı programlama paradigması ve cephe tabanlı (aspect oriented) programlama (CtP) paradigmalarını tümleştirir. EtSiS'in kuşattığı paradigmalar; sağladığı tekrar kullanılabilirlik, karşılıklı işletilebilirlik, esneklik özelliklerinin yanısıra dünya tasvirine de yansır. Herbir paradigmanın dünya tasvirinin tümleşimi, etmen karakteristiğine sahip simülasyon modellerinin tasarımı için önemli bir altyapı sağlar. Yaklaşım, davranış planlama karakteristiği ile karşılık veren (responsive), otonom ve insan biçimsel (anthropomorphic) karakteristiği ile geliştirilen uygulama ve simülasyon modellerinin daha sosyal, esnek ve karşılıklı işletilebilir (interoperable) olmalarını sağlar. Modellerin bulundukları çevreye ait bilgiye erişir olması ve bu bilgiye dayalı olarak muhakeme

yürütmesi, mantık programlama paradigmanın bir getirisidir ve simülasyon modellerine "Dual dünya betimlemesi" kurmasını sağlar.

Bu çalışmada, EtSiS'in cephe tabanlı programlama paradigması çözüm yaklaşımı ele alınmış ve çözümün kavramsal model ve modelleme tekniğine olan katkıları örneklendirilerek tartışılmıştır. Takip eden bölümde Cephe Tabanlı Programlama paradigmasına ilişkin bir bilgilendirme yeralacaktır. Bölüm 3'de EtSiS'in cephe tabanlı programlama paradigması çözümü ve Bölüm 4 simülasyon alanında kategorik bir uygulama bakışı örnek uygulama ile getirilmiştir. Sonuç bölümünde EtSiS'in cephe tabanlı programlama desteğinin sağladığı avantajlar ele alınmıştır.

2. CEPHE TABANLI PROGRAMLAMA

Cephe tabanlı programlama[1], [4] farklı tasarım amaçlarını kod seviyesinde kategorize etmeyi ve yazılımlara modüler bir yapı kazandırmayı amaçlar[5]. Cephe tabanlı programlamanın temel entereseleri; dağıtılmış isterler (çapraz kesen) ve kümelenmiş isterler olarak karşımıza çıkar. Dağıtılmış isterler özellikle modülleştirmede önemli bir engel olarak görülmektedir. Yaklaşım simülasyon dünyasında, bir modelden modelleyicilerin farklı kavramsal dünya ve çözünürlük için bekledikleri davranışların kategorize edilmesi olarak yorumlanabilir. Cephe tabanlı programlama çözümü, isterler aşamasından başlanarak, analiz ve mimari tasarımı içeren modelleme düzeyini ve kodlamayı içine alan bir çözüm olarak ele alınır. Literatürde AspectJ-Like ve Hyper J-Like çözümleri yaygın olarak tercih edilir. Aspect J-like yaklaşımın esasını, birleşim noktalarının (pointcuts) belirlenmesi oluşturur. Hyper J-like yaklaşımında, bağımsız olarak geliştirilmiş modellere ait durum grafiklerinin birleştirilmesi sağlanır ve Aspect J-Like'dan farklı olarak bir tekrar geliştirme (refactoring) gerektirir.

Cephe tabanlı programlamanın yazılım metriklerinden özellikle modülerlik, basitlik ve okunabilirlik konusunda önemli bir desteği olduğu görülmüştür [6]. Literatürde CtP ölçütlerinin sayısal olarak ele alındığı oldukça sınırlı sayıda çalışma mevcuttur [7]. Kersten and Murphy NYP uygulamalarında CtP uygulama başarımlarını pratik uygulamalar ile göstermişlerdir ve uygulama kuralları geliştirmişlerdir [8].

Walker ve arkadaşları çalışmalarında CtP'de kullanılabilirlik (usability) metriği için bir başlangıç bakışı sağlamışlardır [9]. Soares ve arkadaşları çalışmalarında AspectJ uygulamalarının özellikle web tabanlı uygulamalardaki başarısını göstermişlerdir [10]. Çalışmalar arasında belki de en dikkat çekici olan, uygulama alanımız olan EtSiS'in etmen tabanlı olma ve simülasyon ve etmen programlama dili sunması yönü ile Garcia ve arkadaşları tarafından yürütülen şablon-tabanlı yaklaşım ile cephe tabanlı programlama arasında bir sayısal değerlendirmeyi çoklu etmen programlama sistemleri için içeren çalışmadır. Çalışmada etmen spesifik çapraz kesen isterler için gelişmiş bir modülerlik sağladığı sonucuna varmışlardır [11]. NYP tasarım şablonları tabanlı yazılım geliştirme ile CtP'yi birleştiren pek çok çalışma arasında Vaira ve Caplinskas

[12] yazılım paradigmasından bağımsız tasarım şablonları oluşturma üzerine yoğunlaşmıştır ve iddiası farklı paradigmalara uygulanabilen tasarım şablonlarının mevcut olduğu yönündedir. Tsang ve arkadaşları çalışmalarında CtP'nin enterese ayrıştırma başarımlarını değerlendirmişlerdir. Çalışmada CK (Chidamber and Kemerer) [13] metriklerini uygulamış ve NYP ve CtP gerçek zamanlı sistemleri kıyaslamışlardır. Çalışma sonucunda CtP'de gelişmiş bir modülerlik ve bağımlılıkda azalma gözlemlemişlerdir [14]. Bağımlılığın (cohesion) CtP'de sağlıklı ölçülmesi konusunda Kumar ve arkadaşları bir çalışma yürütmüşler ve jenerik bir bağımlılık tanımlama çerçevesi sunmuşlardır [15].

CtP'da tasarlanan tasarım cepheleri arasında ayrıştırma ve soyutlanmış tasarımlar ortaya konulması konusunda, özellikle, tasarım cepheleri arasındaki semantik karışıklıklarının çözümlenmesine dönük çalışmalar özellikle son yıllarda yürütülmeye başlanmıştır [16]. Çalışmada tasarım cephelerinin koordinasyonunun ve bağımsızlıklarının tekrar kullanılabilirlik, modülerlik ve genişletilebilirlik metrikleri ile paralel olarak geliştirilebileceği gözlemlenmiştir. Modülerliğin literatürde önemli bir yer işgal ettiği gözlemlenebilir ve güçlendirilmesi yönünde pek çok çalışma yürütülmüştür [17]. Hata toleransını gibi fonksiyonel olmayan isterleri içeren güvenlik ve görev kritik sistem uygulamalarında, özellikle, kümelenmiş ve modüllere saçılmış fonksiyonel olmayan isterlerin sağlanmasında, CtP tasarım şablonları sağlamıştır. Bu şablonlar özellikle hata tespiti, hata yönetimi, geri dönüştürme (recovery) mekanizması ve güvenli sistemlerde sızma kontrolleri için geliştirilmişlerdir [18].

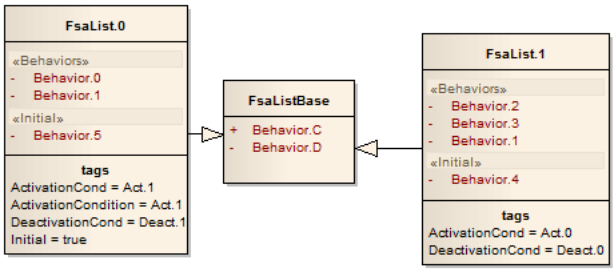
EtSiS'de CtP kullanımındaki temel yaklaşım, çözümün model tümleşik hesaplama ortamı (TMO) olarak görülmesidir [19],[20]. Genel bir ifade ile, model tümleşik hesaplama ortamında ana enterese modeldir ve bir yazılım sistemi farklı soyutlama düzeylerinde bir modeller koleksiyonu olarak görülür ve her biri sistemin farklı bir perspektifini temsil eder. Bu bakışla mühendislik görevlerin her biri bir model tasviri olarak ele alınır [21].

3. EtSiS CEPHE TABANLI PROGRAMLAMA ÇÖZÜMÜ

EtSiS modelleme yaklaşımında bir model tüm modelleme bakışlarını kuşatırken, farklı modelleme bakışlarının her biri bir davranış kategorisi olarak ele alınır. Semantik olarak birbirinden 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 [2]. Bu bakış, davranış perspektifi ile saçılmış (scattered) isterlerin model davranışlarına dağıtılmasını ve tümleşik bir ister setinin (tangled) sınıflandırılmasını sağlar. Yaklaşımın TMO'dan farklılaşan yönü, modelleme bakışlarının farklı modellere değil, aynı modelin farklı davranış kategorilerine

ayrıştırılmasında yatmaktadır. Aspect J benzeri bakışla, birleşim noktalarını senkronizasyon durumları işletiminde, içsel durum geçişlerinde, davranış tetiklemelerinde ve olay gönderimlerinde gerçekleştirir.

Davranış listeleri belirli bir modelleme bakışını sağlayan davranışlar kümesi olarak tanımlanabilir. Davranış listeleri temsil ettikleri bakışa ilişkin bir grup isteri karşılamaya yönelik olarak tasarlanırlar. Bir model en az bir aktif davranış kategorisine sahiptir ve aktif kategori sayısı bir ile sınırlı değildir. Bir davranış kategorisi simülasyon veya etmen çevresi değişimine, kendi durum değişimine bağlı olarak farklı kategorileri aktifleştirerek/pasifleştirerek bakışlar arasında, tanımlı şartlar sağlandıkça geçiş mümkün kılınır. Davranış listeleri ile yürütülen cephe ayrımları bağımlılık ve çelişkiyi yönetmek için önemli bir esneklik sunar. Birbirinden ayrık olması zorunlu olan (zaman zaman çelişki doğuran) davranışların ayrık davranış listelerinde tutulmaları ve aktivasyon şartlarının birbirlerini olumsuzlayan şekilde tasarlanabiliyor olması ile aralarında bağımlılık bulunan davranışların aktif kök davranış listesinde tutulabiliyor olması önemli bir esneklik sağlar.

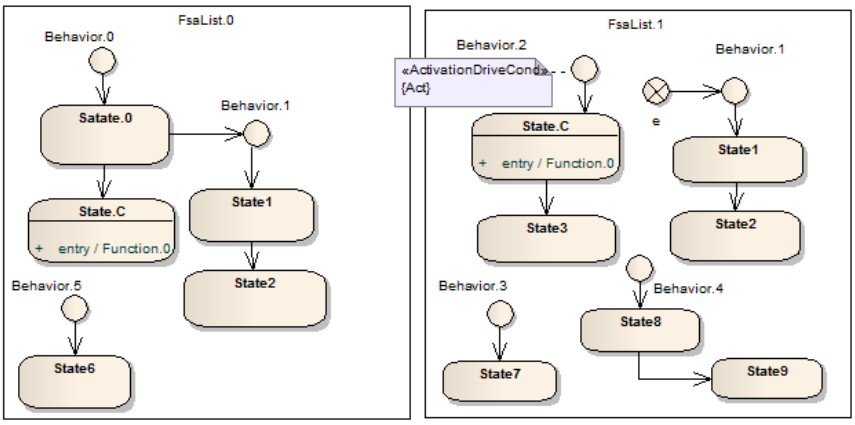


Şekil 1. Davranış Liste Yapıları

Şekil 1'de görülen FsaList.0 ve FsaList.1 isimli listelerin türetildiği FsaListBase*public* olarak tanımlı *Behavior.C* ve private tanımlı *Behavior.D* isimli davranışlara sahiptir. *Public* davranış türetilmiş davranışlar tarafından miras yoluyla alınır ve her iki davranış listesinde yürütülecek fonksiyonları işleten davranış olarak ele alınır. Davranış listelerinde ayrıştırılmış olan davranışlar (*FsaList.0*'da *Behavior.0, Behavior.5* ve *FsaList.1*'in sahip olduğu *Behavior.2, Behavior.3* ve *Behavior.4*) iki farklı cepheye ait davranış işletimlerini gerçekleştirir. *Act.1* ve *Deact.1* olarak isimlendirilen mantıksal ifade tanımları, sırasıyla, işletim zamanında şartların sağlanması durumunda ait oldukları davranış listesi *FsaList.0*'ı aktive ve pasive ederler. Aynı şekilde FsaList.1 için tanımlanan şartlar geçerlidir. FsaList.0 ilgili modelin başlangıç davranış listesi olarak tanımlanmıştır. Sırasıyla, davranış listeleri başlangıç davranışları ise *Behavior.5* ve *Behavior.4* olarak belirlenmiştir.

Modellerin farklı davranış perspektifleri içerisinde ortak olan isterler, davranışlara dağıtılmış olan ve isteri sağlayan fonksiyonu saran durumlar tarafından veya davranış listelerine dağıtılmış isterleri sağlayan davranışları içerirler (Şekil 2). Şekilden de görülebileceği gibi çapraz kesen bir fonksiyon olan *Func.0* fonksiyonu *State.C* ile farklı

davranış listelerinde farklı davranışlara dağıtılarak, saçılmış isteri sağlayan işlev modüller arasında dağıtılmıştır. Benzer şekilde *Behavior.1*'in sağladığı ister farklı davranış listelerine dağıtılarak işlevi gerek farklı modellere gerek aynı modelin farklı cephe tasarımlarına taşınmıştır. Dağıtılmış olan durum ve davranışların birleşim noktası uygulamaları dağıtıldıkları model veya davranış listesinde farklılık gösterebilir.



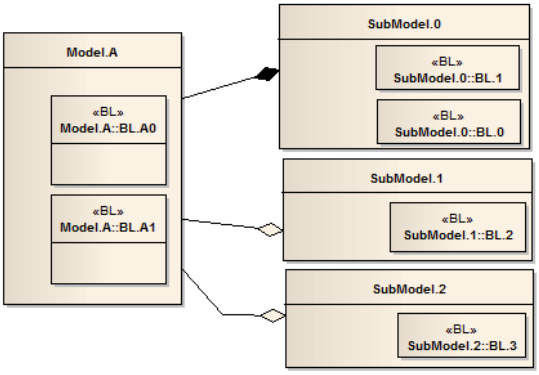
Şekil 2. Davranış Listeleri ile Cephe Yönetimi

Örneğimizde, *FsaList.0*'da *Behavior.1* ve *State.C* içsel durum geçişi ile işletilirken, *FsaList.1*'de *Behavior.1* olay tetiklemesi ile ve *State.C* aktivasyon şartlı olarak işletilmektedir. Bu tasarım ile düşey eksende derinleşen yazılım tasarımı, yatay eksende genişletilmiş ve ilişkilendirilmiştir.

Davranış listeleri ve davranışlara dağıtılmış durumlar ile uygulanan çözüme ek olarak, farklı isterler setini sağlayan farklı modeller tümleştirilerek isterler sağlanır. Tümleştirme işlemi kompozisyon (composition) veya kümeleme (aggregation) olmak üzere iki tipte gerçeklenir. Model birden fazla alt modeli içerecek şekilde modellenir. Bileşen modeller tüm davranış yapılarını muhafaza ederler. Üst model olarak tüm alt modelleri içeren model, alt modellerin zaman ve olay yönetimlerini üstlenir ve bu amaçla kendi içerisinde bir simülasyon alt kümesi oluşturur. Şekil 3'de görülen tasarımda *Model.A* ile *SubModel.0* ile composition ve *Submodel.1* ve *SubModel.2* ile aggregation ilişkisi tanımlanmıştır. *Model.A* içerdiği modellerin zaman yönetimini üstlenir ve olay dağıtımı için bir arayüz görevi üstlenir. Dağıtımı gerçekleştirilen her model olay işletimini kendisi gerçekleştirir. Her bileşen modelin farklı bir cepheyi davranış listeleri ile yönettiği düşünülürse, MIC'de olduğu gibi, cephelerin modellere dağıtımı, daha üst bir kategori olan modeller ile çözülür.

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.

Yaklaşım oldukça esnek, düşük yazılım bağımlılığına ve yüksek tutarlılığa sahip estetik bir çözüm sağlar. İşletim zamanlı olarak aktif davranış kategorilerinin, tanımlı şartlara ve kullanıcı/model etkileşimlerine bağlı olarak, değiştirilebiliyor olması ve ilgili fonksiyon uzantılarının işletim zamanında yüklenebilmesi CtP'yı tasarım zamanından işletim zamanına taşıyan önemli bir açılım olarak görülmektedir.



Şekil 3. Model Seviyesinde Cephe Ayrıştırma

4. SİMÜLASYONLAR İÇİN UYGULAMA KATEGORİLERİ

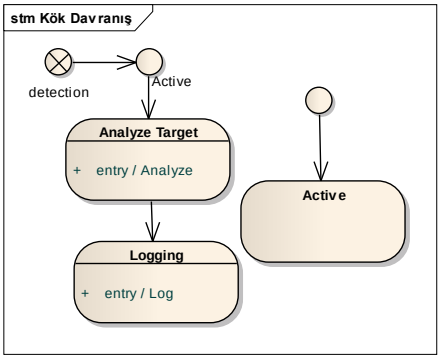
Bu bölümde, uygulama alanı daha genel bir bakış ile ele alınmış, analiz ve eğitim amaçlı olmak üzere iki temel kategorideki simülasyon uygulamaları için bir uygulama şablonu çizilmiştir. Analiz amaçlı simülasyon uygulamalarında modellemeciler, nadiren, gerçek zamanlı ve kullanıcı etkileşimli işletime ihtiyaç duyarlar. Ana beklentiler, simülasyon koşum hızı ve kayıtlarıdır. Mümkün olan en yüksek hıza erişim ve istenilen detay seviyesinde, istenilen şartlar altında ve zamanlarda koşum kayıtlarının alınabiliyor olması analiz simülasyonlarında önemli bir ister setidir.

Eğitim amaçlı simülasyon uygulamalarında, sıklıkla, gerçek zamanlılık, grafik arayüz ve kullanıcı etkileşimi daha önemli isterler olarak karşımıza çıkmaktadır. Temel olarak yapılan mühendislik hesaplamalar aynı olmasına rağmen, farklılaşan isterler, her iki kategoride ortak bir mühendislik çözüm üzerine oturan, farklı fonksiyonları gerektirir. Çözüm kategorik olarak, eğitim amaçlı simülasyonlarda arayüz kontrolleri ve işletim hız kontrolünü farklı davranış listelerinde toplarken, analiz amaçlı simülasyonlarda yüksek çözünürlüklü koşum kayıt davranışları, mümkün olan en yüksek hız işletim kontrolüne ilişkin davranışlar aktif davranış listesi olarak belirlenir. Her iki durum için ortak olan ve mühendislik hesaplama algoritmalarını içeren davranışlar ortak davranış listesi olarak aktif tutulur.

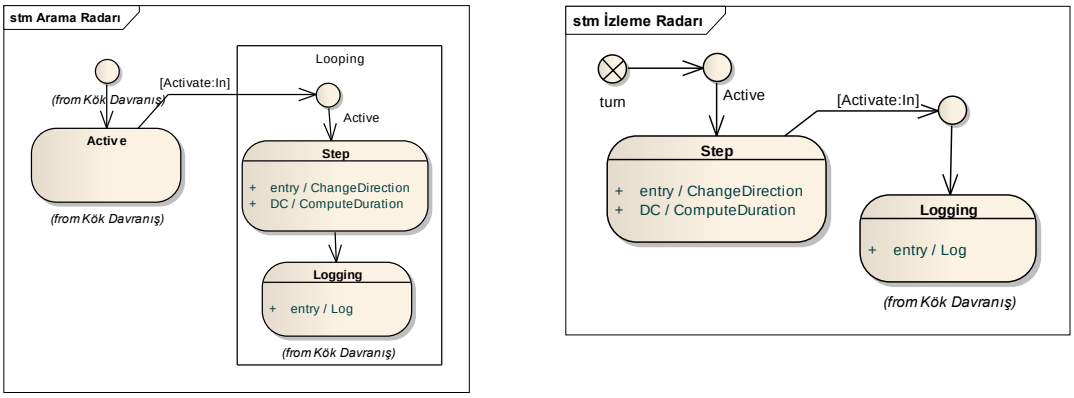
4.1. Örnek Uygulama

Uygulama radar modelinde arama ve izleme davranışlarının modellemesine ilişkindir. İzleme ve arama radarı olarak iki farklı radar modeli geliştirmek yerine, izleme ve arama modu olmak üzere iki farklı modda çalışabilen bir radar modeli geliştirilmiştir.

Temel fonksiyonlar olan, tespit hesapları, verilen bir açıda ve yönde dönüş hareketi her iki çalışma modu için de geçerlidir. Farklılık, fonksiyonları yöneten davranışların tasarımlarında görülecektir. İzleme modunda radar dönüş davranışı belirli bir hedefi takip eder nitelikte tasarlanmış olarak izleme davranış listesinde yeralırken, arama radarı modunda, sürekli formda bir dönüş hareketi tasarımı arama davranış listesinde yeralır.



Şekil 4. Kök Model Davranışı



Şekil 5. Gözetleme ve İzleme Davranışları

Şekil 4'de her iki radar davranışı için kök davranış hüviyetindeki aktif olma ve analiz yürütme davranışları yeralır. Farklılığı oluşturan dönüş hareketleri ise Şekil 5'de görülmektedir. Şekilde görülen iki davranış listesi loglama operasyonunu davranışlarında durum tanımı olarak ve alternatif bir tasarım ile, davranış formunda göstermektedir. Şekil 4'de alınan bir tespit olayına karşılık hedef bilgilerinin analizi gerçekleştirilmekte ve analiz sonrasında loglama işlemi yürütülmektedir. Şekil 5'de görülen Arama Radarı davranışında ComputeDuration zaman hesaplayıcısı ile belirlenen zaman adımlarında açısal dönüş yapmakta ve bu hareketi kısıt olarak tanımlı açı aralıkları için yürütmektedir. İzleme radarı davranışında ise, turn olayı ile belirtilen açıya hesaplanan süre kullanılarak dönüş yapılmakta, sürekli formda tekrarlanan bir davranış yürütülmemektedir. Loglama işlemi ise bir durum olarak değil izleme davranışının tetiklediği bir davranış olarak yürütülmektedir.

5. SONUÇ

EtSiS'in cephe tabanlı programlama desteği varolan Model Güdümlü Mimari ve Model Güdümlü Geliştirme (Model Driven Architecture (MDA)/Model Driven Development (MDD)) kavramı ile birleştirildiğinde oldukça güçlü ve esnek bir modelleme ortamı sağlamaktadır. Tasarladığı dünyayı nesne yönelimli programlamanın kurguladığı nesneleri, davranış, durum tasvirleri ve muhakeme mekanizması ile daha ileri bir aşamaya taşır.

Cephe tabanlı programlama yaklaşımı, saçılmış ve kümelenmiş isterleri karşılamada, EtSiS'in davranış ve davranış liste tasarımlarının bir tasarım şablonu olarak kullanımı ile esnek bir çözüm sağlar. Yaklaşım ile, aralarında semantik bir bağ kurulmayan, atomik fonksiyonların kod saçılmaları önlenerek, bakımı ve geliştirilmesi oldukça etkin simülasyon ve etmen modellerinin geliştirilmesi sağlanır. EtSiS betik programlama dili ile saçılmış isterlerin (çapraz kesen) sağlanması gerek isteri sağlayan fonksiyonu saran durum tanımlarının davranışlara dağıtılması gerekse de bağısız davranış formunda davranış listelerinde kullanımı, cephe tabanlı programlama yaklaşımı olarak oldukça esnek ve estetik bir çözüm olarak görülebilir. Davranış listelerinin işletim zamanlı aktivasyonu ve de-aktivasyonu ile işletim zamanlı yazılım uzantısı ile fonksiyonel genişletmeler cephe programlama çözümünü tasarımdan işletim zamanına taşıyan oldukça önemli bir esneklik sağlar.

6. KAYNAKÇA

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

[2] Hocaoğlu, M. F., "EtSiS: Etmen tabanlı Simülasyon Sistemi", *4. Ulusal Savunma Uygulamaları Modelleme ve Simülasyon Konferansı*, USMOS'2011, Ankara, Türkiye.

[3] Kiczales, G.,Lamping, J., Mehdhekar, A., Maeda,C.,Lopes, C. V.,Loingtier, J., Irwin, J., (1997), "Aspect-Oriented Programming", Proceedings of the *European Conference on Object-Oriented Programming (ECOOP)*, Springer-Verlag LNCS 1241.

[4] Lieberher, K.,Orleans, D.,Ovlinger, J., (2001), "Aspect-orientedprogrammingwithadaptivemethods", Communications of the ACM 44(10) 39–41.

[5] Mendonça,N. C.,Silva,C. F.,Maia,I. G.,Rodrigues, M. A. F. (2008), "A LooselyCoupledAspect Language For SOA Applications", *International Journal of Software Engineering and Knowledge Engineering*, Vol. 18, No. 2 243-262.

[6] Madeyski, L., Szała, L., (2007), "Impact of aspect-oriented programming on software development efficiency and design quality: an empirical study*", IET Softw*., 1, (5), pp. 180–187.

[7] Garcia, A.F., Sant'Anna, C., Figueiredo, E., Kulesza, U., de Lucena, C.J.P., and von Staa, A. (2006), 'Modularizing design patterns with aspects: a quantitative study' in Rashid, A., and Aksit, M. (Eds.): 'T. Aspect-oriented software Development I', vol. 3880 of Lecture Notes in Computer Science, pp. 36–74.

[8] Kersten, M., and Murphy, G.C., (1999), 'Atlas: a case study in building a web-based learning environment using aspect-oriented programming'. Proc. ACM SIGPLAN Conf. Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 1999), New York, USA, November, pp. 340–352

[9] Walker, R.J., Baniassad, E.L.A., and Murphy, G.C., (1999), 'An initial assessment of aspect-oriented programming'. Proc. Int. Conf. Software Engineering (ICSE 1999), Los Alamitos, USA, May, pp. 120–130.

[10] Soares, S., Laureano, E., and Borba, P., (2002), 'Implementing distribution and persistence aspects with AspectJ'. Proc. ACM SIGPLAN *Conf. Object-Oriented Programming, Systems, Languages*, and Applications (OOPSLA 2002), New York, USA, November, pp. 174–190

[11] Garcia, A.F., Sant'Anna, C., Chavez, C., da Silva, V.T., de Lucena, C.J.P., and von Staa, A., (2003), 'Separation of concerns in multi-agent systems: an empirical study'. *Proc. Int. Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS 2003)*, vol. 2940 of Lecture Notes in Computer Science, pp. 49–72.

[12] Vaira, Ž., Aplinskas, A.C., (2011), "Software Engineering Paradigm Independent Design Problems, GoF 23 Design Patterns, and Aspect Design", INFORMATICA, Vol. 22,No.2,289–317.

[13] Chidamber, S.R., and Kemerer, C.F., (1994), 'A metrics suite for object oriented design', IEEE Trans. Softw. Eng., 20, (6), pp. 476–493.

[14] Tsang, S.L., Clarke, S., and Baniassad, E.L.A., (2004)'An evaluation of aspect-oriented programming for Java-based real-time systems development'. Proc. Int. Symp. Object-Oriented Real-Time Distributed Computing (ISORC 2004), Vienna, Austria, May, pp. 291–300.

[15] Kumar, A., Kumar, R., Grover, P.S., (2011), "Unified Cohesion Measures For Aspect-Oriented Systems", International Journal of Software Engineering and Knowledge Engineering Vol. 21, No.1, 143–163.

[16] Gordillo, S., Zambrano, A., (2010), "A Fine Grained Aspect Coordination Mechanism", International Journal of Software Engineering and Knowledge Engineering Vol. 20, No. 7 1025–1042.

[17] Sirbi, K., Kulkarni, P.J., (2010), "Enhancing Modularity in Aspect-Oriented Software Systems-An Emperical Study", (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 06, 2009-2014.

[18] Hameed, K., Williams, R., Smith, J., (2010), "Separation of Fault Tolerance and Non-Functional Concerns: Aspect Oriented Patterns and Evaluation", J. Software Engineering & Applications, 2010, 3: 303-311 doi:10.4236/jsea.2010.34036 Published Online April (http://www.SciRP.org/journal/jsea).

[19] Meslati, D., (2009), "On ASPECTJ and Composition Filters: A Mapping of Concepts", INFORMATICA, Vol. 20, No. 4, 555–578.

[20] Heidenreich, F., et al (2009). "On Language-Independent Model Modularisation", In: Rashid, A.,Ossher, H. (Eds.), Transactions on AOSD VI: Special Issue on Aspectsand Model-DrivenEngineering, LNCS, Vol. 5560. Springer-Verlag, pp. 39–82.

[21] Kleppe, A., et al. (2003), "MDA Explained: The Model Driven Architecture PracticeandPromise", Addison-Wesley.