Etmen tabanlı Dağıtık Etkileşimli Simülasyon

M. Fatih Hocaoğlu [a, b]

[a] İstanbul Medeniyet Üniversitesi, Mühendislik ve Mimarlık Fakültesi,
mfatih.hocaoglu@medeniyet.edu.tr
[b] Agena Bilişim ve Savunma Teknolojileri, 34912 İstanbul Teknopark /İstanbul,
hocaoglu@agenabst.com

ÖZ

Bu çalışmada, paralel simülasyonda işletim zamanlı senkronizasyon algoritması seçimini zeki olarak yürüten bir dağıtık simülasyon model kütüphanesi geliştirilmiştir. Etmen olarak tasarlanan simülasyon modeli (DS-a) ve senkronizasyon algoritmaları ile, bir ağ üzerinde herhangi bir merkezi koordinasyon bağımlılığı olmadan, işletim sağlanmıştır.

DS-a yönetimini üstlendiği simülasyon modellerinin simülasyon servislerini yürütürken, koşum performansını en iyileyecek senkronizasyon algoritmalarını kural tabanlı olarak seçmektedir. Simülasyon yönetiminin modeller aracılığı sağlanması önemli bir esneklik sağlamıştır. Etmen Tabanlı Simülasyon Sistemi (EtSiS) tarafından desteklenen çoklu programlama paradigmaları ile sağlanan güçlü programlama arkaplanı ve dağıtık simülasyonda dinamik olarak simülasyon senkronizasyonunun bir otonom etmen ile gerçekleştirilmesi, çalışmanın merkezi yenilikleri olarak görülmektedir.

Anahtar Kelimeler: Durum tabanlı Programlama, EtSiS, Etmen, Paralel Simülasyon, Zeki algoritma seçimi

Agent based Distributed Interactive Simulation

Abstract

In this study, a simulation model library that choices synchronization algorithms for parallel simulation in execution time is developed. Using the model that is developed as an agent named DS-a and a series of synchronization algorithms, simulation executions are achieved on a network without any centralized coordination.

While DS-a provides simulation services to simulation models it is in charge of, it also choices simulation synchronization algorithms to maximize execution performance as rule based. Managing simulation execution by simulation models provides a distinguished flexibility. The powerful programming background that are provided by multi programming paradigms supported by Agent driven simulation framework – AdSiF and parallel simulation execution synchronization by an autonomous agent are the central concepts of the study.

Keywords: AdSiF, Agent, Intelligent selection, Parallel simulation, State oriented programming

1. GİRİŞ

Etmen tabanlı paralel simülasyon işletimi ve zeki algoritma seçiminin temel motivasyonu Etmen tabanlı Simülasyon Sistemi EtSiS'in [1]–[3] herhangi bir merkezi koordinasyona ihtiyaç duymadan (non-federated) ayrık olarak işletilen (stand-alone) simülasyon modellerinin paralel olarak işletilebilmeleri ve işletimde zaman senkronizasyonun etmen tabanlı olarak gerçekleştirilmesinin sağlanmasıdır [4]–[6].

Etmen tabanlı olmaktan beklenen amaç; zaman senkronizasyonu ve olay yönetimi için kullanılacak algoritmaların EtSiS davranış tanımlamaları ile yönetilmesi ve kural tabanlı olarak, farklı algoritmalar arasından seçim yapılabilmesini sağlamaktır. Literatürde ele alınan senkronizasyon algoritmaları farklı performans kriterlerini en iyilemektedirler [7]. Algoritma seçiminden amaç simülasyonun işletim performansını dikkate alarak performansını eniyileyecek algoritmanın işletim zamanlı olarak seçilmesinin sağlanmasıdır. Paralel simülasyonda karşılaşılan en önemli sorunlardan biri kilitlenme (deadlock) tespiti ve çözümlenmesidir. İşletim performansında ve algoritma seçimlerinde kilitlenmenin enazlanması önemli bir seçim kriteri olarak ele alınmıştır.

Simülasyonun paralel ve kooridnatörsüz olarak işletimini yürütmesi için bir paralel işletim etmeni geliştirilmiştir. Geliştirilen model ile merkezi koordinatörlü federatif simülasyon işletimlerinden farklı olarak, bir network üzerinde herhangi bir merkezi koordinasyon bağımlılığı olmadan işletim sağlanmaktadır. Modelin kazandırdığı önemli katkılardan biri, simülasyon yönetiminin model katmanından yürütülebilme esnekliğidir.

EtSiS'in temel tasarım özelliği olan Etmen tabanlılık ve çoklu programlama desteğinin verdiği avantajlar dağıtık simülasyonda zaman yönetiminin dinamik olarak, bir otonom etmen kararı ile yapılabiliyor olması çalışmanın merkezi bir yeniliği olarak görülmektedir. Geliştirilen sistem simülasyon işletimi EtSiS'in sağlamış olduğu çekirdek yorumlayıcı ile sağlanmakta ve simülasyon modelleme yaklaşımında bir standart yapı sağlamaktadır. Sağlanan betik dil avantajları ile yeni modelleme yaklaşımlarının modellenmesine imkan sağlanmaktadır. Savunma alanında geliştirilen simülasyon çalışmalarının, ağırlıklı olarak HLA tabanlı olarak geliştirildiğini görmekteyiz. Son yıllarda DIS (Distributed Interactive Simulation)'e dönük bazı geri dönüşler dikkat çekmektedir [8], [9].

Takibeden bölümlerde, simülasyonda etmen güdümü, Paralel simülasyonda senkronizasyon algoritmaları ve bir etmen modeli ile yönetimi ve uygulama detayları yeralacaktır.

2. Simülasyonda Etmen Güdümü

Simülasyon işletiminde etmen güdümü simülasyon modellerinin dağıtık koşumu için koordinasyonlarının üstlenilmesi, kural tabanlı olarak paralel işletim senkronizasyon algoritması seçimi için gerekli performans kriterlerinin takibinin yapılması ve davranış yapısı olarak modellenmiş senkronizasyon algoritmalarının tanımlı şartlarda aktive edilmesi olarak belirlenmiştir. Algoritma seçimi simülasyon işletimi esnasında senkronizasyon algoritmalarının, karşılaşılan soruna uygun olarak, performansı en iyileştirecek şekilde değiştirilmesi sağlanır. Algoritma seçimi kural tabanlı bir muhakeme mekanizması ile yürütülür. Muhakeme altyapısı EtSiS'in muhakeme motoru kullanılarak gerçekleştirilir. Simülasyon işletiminde işletilen mesaj sayısı, en düşük frekansa sahip simülasyon bileşenlerinin sayısı ve frekans değeri ve modeller arası mesajlaşma ağının karmaşıklığı, kuralların tanımlanmasında önemli parametreler olarak kullanılmıştır. Kural örnekleri Şekil 1'de görülmektedir.

algorithm(MessageNumber, EntityNumber,'nullMessageAlg'):- MessageNumber < EntityNumber.

algorithm(lowerBound, constant,'rollBackAlg'):-lowerBound < constant.

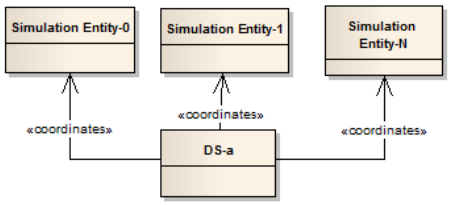
Şekil 1: Algoritma Seçim Kural Setleri

Örnekte işletim esnasında mesaj sayısının varlık sayısını geçmesi durumunda nullMessageAlg algoritmasının seçimi, minimum zaman adımı tanımlı sabit adımın altına indiğinde ise rollBackAlg algoritması seçimi öngörülmüştür. İşletim esnasında seçilen algoritma DS-a modeline ait ilgili davranış listelerini aktive eder. Algoritmalar arasında geçiş esnasında uygulanması gereken ön işlemler, aktive edilen davranış listesinin bir başlangıç adımı olarak yürütülür [1], [10].

3. Paralel İşletim Modeli

Paralel simülasyon işletim modeli (DS-a: Distributed Simulation- agent) ilişkilendirildiği modellerin simülasyon koordinasyonunu üstlenir. Koordinasyon zaman yönetimleri ve olay etkileşimlerinin yönetilmesini içerir. DS-a modeli davranış içeriği, simülasyon zaman yönetiminin, olay etkileşimlerinin ve algoritma seçimlerinin yapılmasına yöneliktir. Modelin davranışları standart olarak EtSiS yorumlayıcısı tarafından yorumlanır ve bu yorumlanan davranışlar seçilmiş olan paralel işletim senkronizasyon algoritmasına uygun olarak simülasyonun koordinasyonunu yürütür. Bu anlamda, EtSiS çekirdek yorumlayıcısının işlettiği davranış yapısı simülasyon işletim semantiğinden bağımsızdır ve işletim bir üst katmanda simülasyon işletim semantiğini yürütür ve en üst katmanda ise uygulama modellerinin semantik yapıları işletilir. Şekil 2'de DS-a modelinin simülasyon modelleri ile ilişkilendirilmesi görülmektedir. Aralarında "coordinates" ilişkisi kurulan modellerin yönetimi DS-a modeli tarafından üstlenilir. Kurulan ilişki ile EtSiS'in ontoloji tanımından gelen [11], ilişki kuralları işletim semantiği yürütülür ve modellerin görev devri otomatik olarak yürütülür. Görev devri bir tür model cephesinin (aspect) aktivasyonu olarak değerlendirilebilir. Zira, modeller ilişki kurulduğunda üstlendiği görevi davranış listelerini değiştirerek üstlenmektedir [1], [12].

Paralel simülasyon işletiminde temel yaklaşım zaman ilerlemesinin her modelin; 1) Olay kuyruğunda yeralan olay zamanlarını dikkate alarak gerçekleştirmesi, ve 2) İç/dış-durum (internal and external state transition) geçişlerine göre ilerleme zamanlarını belirlemesi, temel kuralları esas alınarak belirlenir.



Şekil 2: DS-a Model İlişkilendirme

Olaylar modellerin diğer modellerin bir davranışını tetiklemek (bir iş yapmasını istemek; etmen programlamada temel eylem kategorileri (speech action types) [13]) ve bilgi aktarmak (parametreler) amacıyla bir eylem gerçekleştirildiğinde oluşturulan ve eylemleri oluşturan durum tanımlamalarının sonlanması ile gönderilen, etkileşim nesneleridir. Çözümün dayandığı olay iletimi iki yöntemle sağlanır; Çekme tabanlı olay dağıtımı (Pull) ve İtme tabanlı olay dağıtımı (push) olarak belirlenmiştir.

Çekme tabanlı Olay Dağıtımı

Algoritma şu şekilde tasarlanmıştır; olay gönderen model ileteceği olayları bir listede tutar, istemci model eğer listesinde gönderici modele ait herhangi bir olaya sahip değilse istekte bulunur ve listede kendisine gönderilecek olayları çeker. Listeden çektiği

olayları zaman etiketli olarak, işletim sırasına koyar. Bu durum istemci modeli sonraki simülasyon adımlarında, aldığı olaylardan daha önce bir olay almayacağını garanti eder.

İtme tabanlı Olay Dağıtımı

Olay gönderici modelin, olayları doğrudan adres modellere iletmesi ile gerçekleşir. İstemci modelin bir olay isteğinde bulunması sözkonusu değildir. Burada kısaca ele alınan olay etkileşim yöntemleri farklı senkronizasyon algoritmalarında kullanılmaktadır.

Paralel işletimde aşağıdaki temel noktalar üzerinde durulur; 1) Gerçekleştirilen koşumların seri işletim sonuçları ile aynı olması, 2) Kilitlenmenin (deadlock) çözümlemesi ve 3) Şebeke gecikmelerinin yönetimi. Çözüm yaklaşımı Muhafazakâr (Conservative) ve İyimser (optimistic) çözüm yaklaşımları etrafında oluşturulmuştur.

Muhafazakar yaklaşımlar simülasyonun güvenli zamana kadar işletilmesi esasına dayanır ve güvenli zaman simülasyon ortamının paylaşıldığı diğer modellerden geçmişe dönük bir olay alınmayacağının garanti edildiği işletimlerdir.

İyimser yaklaşımlar, yüksek işletim hızı getirirken, beraberinde zaman zaman geri dönüşleri gerektiren durumlarla da karşılaşılır. Muhafazakar yaklaşımlar, performans olarak daha az etkin olsalar da, herhangi bir nedensellik (causality) problemi oluşturmazlar. İleri bakış (Lookahead) zamanın büyük olduğu sistemlerde muhafazakar yaklaşımlar iyi sonuç verirken, ileri bakış zamanın küçük kaldığı sistemlerde, genel olarak kötü ve hatta zaman zaman seri işletilen bir simülasyondan daha kötü performans sonuçları üretir. Özellikle, öncelikleri yüksek bileşenler olan simülasyon uygulamalarında ve sıfıra yakın, küçük frekanslı çözümler gerektiren bileşenleri olan simülasyon uygulamalarında [14], paralel işletime katılan makineye orantılı performans oldukça kötü etkilenir. Muhafazakar yaklaşımların tüm simülasyon modellerini herhangi bir nedensellik kısıt (causality constraint) problemi oluşturmamak için sıklıkla modelleri bloke ettiği için paralellikten ciddi ödünler verilir. Boş mesaj iletimi (Null message passing) yaklaşımında sıklıkla çok yoğun mesajlaşmanın getirdiği performans kayıpları ile karşılaşırız.

İyimser yaklaşım ise muhafazakâr yaklaşımın aksine herhangi bir güvenli zaman sınırlaması olmaksızın ilerlenilen ve geçmişe dönük bir olay alındığında ilgili zamana geri dönülen bir işletim olarak ele alınır. İyimser yaklaşımlar tam bir paralellik sağlarlar. Fakat nedensellik kısıtının sağlanmadığı durumlarda geri dönüş (rollback) ciddi bir performans kaybına sebep olur. Ayrıca, geri dönüş noktalarının kayıt altına alınması, hem performans kaybına sebep olur hem de ciddi bir programlama yükü getirir. EtSiS geridönüş noktaları için bir varsayılan çözüm sunar. Modele ait öznitelikler ve durum vektörü (davranış yapıları) istenilen zaman anı için kaydedilir ve istenildiğinde geri yükleme otomatik olarak sağlanır.

EtSiS'de simülasyon zamanı işletimi kesikli olay işletim esaslı olarak ele alınmaktadır. Yaklaşım sürekli olay işletimlerini kesikli bir uzaya resmederek yürütür ve bu koşum hızını artırırken bir çözünürlük kaybının önüne geçer. Sürekli zaman ve kesikli olay simülasyonlarının her ikisinde de ilerlenilecek zaman noktassı varlık zaman isteklerine göre belirlendiği için, koşum frekansı dinamik olarak belirlenir. Sürekli olay simülasyon uygulamalarında, zaman adımının modelin kendi dinamiğine, modellemeyi gerçekleyen denklem takımlarının çözüm adımı genişliğine göre belirlenmesi ve her modelin farklı bir zaman ilerleme adımı belirlemesi esasına dayalı olarak sürekli olay simülasyon işletimi sağlanır. Dinamik frekans burada iki noktada sağlanır. Bunlardan birincisi, simülasyona katılan modellerin birbirlerinden farklı zaman adımları yürütmesi ile oluşan dinamik frekans, diğeri ise, modelin simülasyon işletimi süresince gerçekleştireceği zaman adımlarının koşum boyunca dinamik değişiminin sağlanmasıdır. Burada ana nokta EtSiS'in koordinator merkezli mevcut çözümünde varolan dinamik frekanslı sürekli olay simülasyonu çözümünün, merkezi bir federatif yapı olmaksızın gerçekleştirme ve her simülasyon modelinin bir biri ile senkron fakat bağımsız koşumunun sağlanmasıdır.

3.1. Muhafazakâr Algoritmalar (Conservative Algorithms)

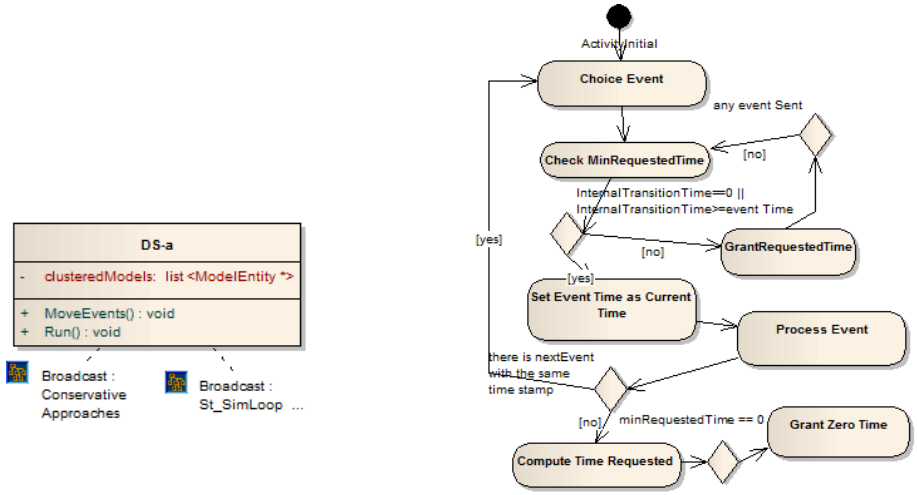
Muhafazakar çözüm algoritmaları alt bölümlerde ele alınmış olup, bunlar bir yönetici model tarafından yönetilmeyen ve yönetilen yaklaşımlar olarak iki ana kısımda düşünülebilir. Boş mesaj iletimi, senkron koşumlar ve sinyal protokolü muhafazakar algoritmalar kapsamında ele alınmışlardır [7].

3.1.1. Boş Mesaj İletimi

Boş mesaj iletimi (Null message passing) algoritması üç farklı yaklaşımla ele alınmıştır. Bunlardan birincisi mesajlaşma trafiği daha yoğun olan tüm modellere null mesaj gönderimi (Broadcast), ikincisi kuyruğunda olay eksiği olan modelin ilgili modelden boş mesaj isteği yapması (Demand based) ve üçüncü yaklaşım modeller arasında ilişkilendirmeye dayalı işletim (Causality) olarak ele alınmıştır. Muhafazakar simülasyon işletimi temel algoritması ve boş mesaj kilitlenme çözümünün oturtulduğu esaslar burada ele alınmıştır. Çözümün varsayımları ve ilerleme mantığı aşağıda sunulmuştur. Çözüme ait varsayımlar şu şekilde özetlenir; 1) Simülasyon modelleri olayları zaman etikeli olarak küçük zamandan büyük zamanlı olana doğru işletirler, 2) Simülasyon modelleri ve aralarındaki ilişkiler statiktir, 3) Her olay zaman etiketli olarak gönderilir (bu kısıt EtSiS ilişki yönetmi ile esnetilebilir fakat çalışmanın dışında tutulmuştur) ve 4) Şebeke gecikmesi yoktur, 5) En düşük zamanlı olay simülasyonun ilerleyeceği en düşük zamanı gösterir ve daha küçük zamanlı bir olayın gelmeyeceği varsayılır (lower bound on the time stamp (LBTS)).

3.1.1.1. Algoritma

Simülasyon modelleri zaman ilerlemelerini iki şekilde belirlerler; 1) İçsel geçiş (internal transition) ve 2) Olay zamanı. İçsel zaman geçişi durum geçişi ile durumlara tanımlanmış zaman parametresi ile belirlenir. İşletim adımları Şekil 3'de görülmektedir. Algoritma bir davranış formunda geliştirilmiş olup EtSiS yorumlayıcısı tarafından DS-a modeli davranışı olarak yorumlanarak işletilir. Şeklin ilk bölümünde DS-a modelinin koşum sırasında değişim yaptığı senkronizasyon algoritmaları davranış listeleri görülmektedir.



Şekil 3 Muhafazakâr Paralel Simülasyon İşletim Algoritması

Durum tanımlamaları ve yürütülen eylemler;

"Choice Event": Olay listesinden en küçük zaman etiketine sahip olan olay seçilir.

"Process Event": Seçilen olayın tetiklediği davranışlar çizelgelenir (EtSiS yorumlayıcı),

"Compute Time Request": Davranış işletimleri yürütülür ve zaman istekleri belirlenir. Zaman isteği olarak minimum zaman isteği aktif davranışlar üzerinden belirlenerek iletilir (EtSiS)

"Grant Time": Simülasyon modeline içsel geçiş zamanı verilir.

Karar Düğümleri:

EventList: *No event in the Queue*: Kuyrukta herhangi bir modele ait (kendisi dışında) olayın kuyrukta bulunmaması durumunda deadlock oluşur ve bu durumda iç geçiş zamanı da dâhil herhangi bir zaman ilerlemesi verilmez.

There is more event in the Queue: Zaman ilerlemesi gerçekleştirilmek üzere sonraki karar adımına ilerlenilir.

Time: *Time Request* < *Next Event Time*: İçsel geçiş zamanının kuyrukta bekleyen en küçük zaman etiketli olay zamanından daha küçük olması durumunda içsel geçiş zamanı kadar ilerleme sağlanır.

Time Request >= *Next Event Time*: Kuyruktaki olay işletilir ve olay zamanı ilerleme zamanı olarak verilir.

3.1.2. Kilit Çözümü

Boş mesajın tüm modellere yayınlandığı yayınlama (broadcast) yaklaşımı, kilide giren simülasyon modelinin ilgili modelden olay isteğinde bulunmasını gösteren istek tabanlı (demand based) çözüm ve aralarında olay etkileşimi tanımlanmış modeller arasında boş mesaj geçişinin kontrol edildiği bağımlılık tanımlı (causality definition) yaklaşımı ele alınmıştır.

Broadcast Yaklaşım; Yaklaşım model her içsel durum geçişinde (internal state transition) bir boş mesaj (null message) yayınlar. Bu mesaj tüm modeller tarafından alınır. Null mesaj gönderen modelin bir sonraki en yakın muhtemel olay gönderim zamanını içerecektir. Bu şekilde mesajı alan modellerin, gönderen modelden doğacak bir deadlock yaşamamaları garanti altına alınmış olur.

Demand based Yaklaşım; Yaklaşımın esasını kilit oluşumunun çözülmesi oluşturur. Eğer herhangi bir model olay kuyruğunda senaryoda bulunan bir modele ait olay bulunmaz ise deadlock oluşur. Deadlock recovery için model olay kuyruğunda kendisine ait olay bulunmayan modele, en yakın zaman isteğine ilişkin sorgu yürütür. Sorgu sonucu bir *null message* olarak döner. Yeni durumda olay kuyruğunda eksik olay olmayacağı için deadlock çözülür ve simülasyon ilerler.

Causality Definition Yaklaşımı; Bu yaklaşımda modeller arasında mesaj ilişkisi belirlenir ve aralarında herhangi bir mesaj iletimi olmayan modellerin güvenli zaman ilerlemesi için sonraki olay etkileşimi sonsuz zaman olarak belirlenir.

3.2. Merkezi Koordinasyon Tabanlı Çözümler

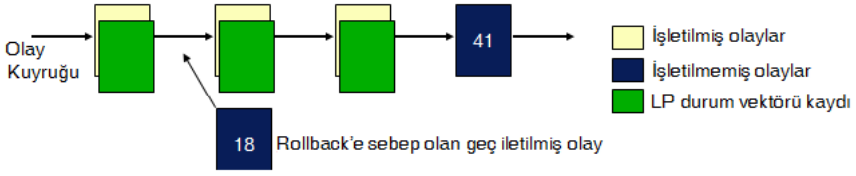
EtSiS simülasyon işletiminin modeller tarafından yönetimine imkan tanır. Bu amaçla farklı senkronizasyon algoritmalarının ve olay yönetim stratejilerinin kontrolünü yöneten bir model ile dağıtık simülasyon işletimi gerçekleştirilir.

Merkezi koordinasyon tabanlı çözüm iki amaç için kullanılır; 1) Kilitlenme tespiti ve çözümü ve 2) Modellerin eşzamanlı işletiminin sağlanması (gerçek zamanlı uygulamalar için). Burada ele alınan model simülasyona yönetici rolü ile katılan ve yapısal olarak standart bir model olan "*Simülasyon Yönetim Modelidir*".

Signal Protocol; Signal protokol çözümü merkezi bir yönetim modeli tarafından işletilen bir algoritmadır. Yaklaşım global kilitlenme tespitinin gerçeklenmesi ve çözümünü hedefler. Merkezi Yönetim modeli kökünde kendisinin yeraldığı dallarında

ve yapraklarında simülasyon modellerinin bulunduğu bir ağaç yapısı oluşturur. Kilitlenme oluşumu aşağıdaki algoritma ile tespit edilir. Boş mesaj ağaç boyunca yayınlanır ve kilitlenme çözülür.

İyimser Yaklaşım: Geri Sarım Mekanizması (Rollback Mechanism); Rollback mekanizması tabanlı çözüm iyimser çözüm yaklaşımı olarak ele alınmıştır. EtSiS'in sağladığı enstantane kayıt mekanizması çözüm için önemli bir altyapı sağlamaktadır. İşletim şu şekilde sağlanmaktadır. Simülasyon modelleri diğer modellerden herhangi bir sonraki ilerleme zamanı sorgulamadan işletimlerine, ilerleme zamanlarının güvenli olduğu varsayımıyla, ilerlemektedirler. Geçmiş zamana dönük bir olay aldıkları durumda olay zamanına geri dönerler. Klasik bir durum olarak senaryo işletiminde simülasyon modelleri birbirlerinden farklı zamanlarda olacaklardır. Geri sarım algoritması işletimi Şekil 4'de şematik olarak gösterilmiştir. Simülasyon modeli olay etkileşimlerinde modelin yürütmekte olduğu davranışları, öznitelikleri ve kullanıcı tanımlı veriyapılarını bir durum vektörü listesi içerine zaman etiketli ve olay ilişkili olarak kaydeder. Kayıt işlemi Şekil 5 (a)'da görülen davranış yapısı ile yönetilir. Olay ile tetiklenen davranışlar *BehaviorRollback* isimli davranışı *aktivasyon* fazında tetikleyen *BehaviorBase* isimli davranıştan türetilirler.

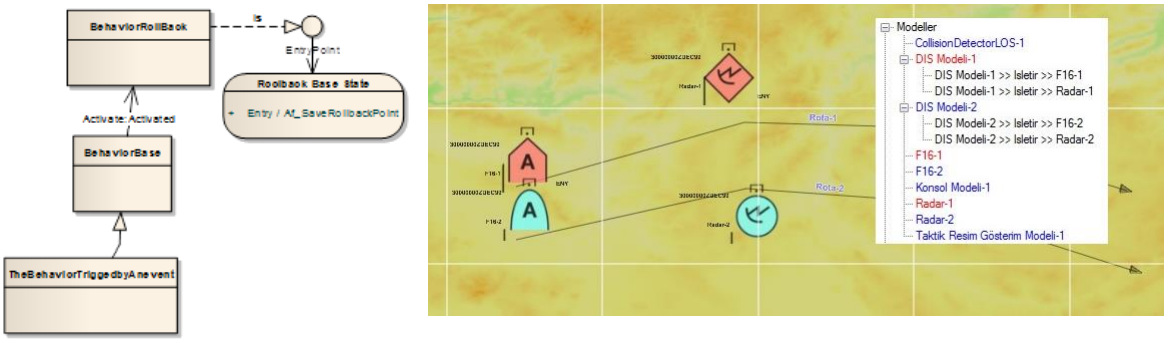


Şekil 4 Geri Sarım İşletimi

Türetilmenin bir sonucu olarak kök davranışa ait olan temporal ilişkili davranış ilişkisi yürütülür. Yürütülen davranış olay ilişkili ve zaman etiketli olarak durum vektörü kaydını oluşturarak listeye ekler. Şebekede geciken veya daha geri zamanda yeralan bir modelden olay gelmesi durumunda (18 zaman etiketli olay) simülasyon modeli durum vektörü listesinden en yakın zamanlı durumu geri yükler. Geri yükleme işleminde en yakın zamanlı durum vektörü simülasyon modeli durumu olarak yüklenir. Durum vektörü listesinden sonraki değerler çıkarılır. Alınan gecikmeli olay ile durum vektörü olay zaman etiketinin aynı olması durumunda bir önceki zaman durum vektörü yüklenir ve gecikmeli olay işletilecek zaman listesine alınır. Algoritma şebeke gecikmelerinden doğan geç mesaj iletimlerini de mevcut akış içerisinde çözer.

4. Senaryo Hazırlama

Senaryo hazırlama aşamasında dağıtık koşumu yapılacak olan modeller senaryoya dahil edilen DS-a modelleri ile ilişkilendirilir ve her bir DS-a modeli şebeke üzerindeki bir makineye atanır. İşletim başlatıldığında DS-a modelleri dağıtıldıkları makineler üzerinde işletime başlatılır ve her bir DS-a modeli kendisine ilişkilendirilmiş olan simülasyon modellerinin yönetimini üstlenir. Görev üstlenme DS-a ile modeller arasında kurulan ilişkinin davranış cephe aktivasyonu olarak gerçekleştirilir. DS-a modeli ilişkili modellerin zaman ve olay yönetimlerini üstlenen davranış listelerini aktive ederken, ilişkilendirilen modeller ise bu işlevlerin yürütülmesine ilişki sorgu ve isteklere cevap verecek davranış listelerini aktive ederler.



(a) (b)

Şekil 5 Geri Sarım Davranış Diyagramı ve Senaryo Tasarımı

5. Örnek Uygulama

Kavramın uygulanışının basit bir gösterimi için savunma alanından bir örnek seçilmiştir. Örneğimizde iki radar ve iki uçak modeli mevcuttur ve radarlar karşı kuvvete ait uçakları tespit etmektedirler. Şekil 5 (b)'de ilgili senaryo yerleşimi görülmektedir. Bir radar ve bir uçak DIS Modeli olarak isimlendirilen ve dağıtık koşum işletimini gerçekleştiren model ile ilişkilendirilmiş ve aynı işlem karşı kuvvete ait varlıklar için de yürütülmüştür. Koşum esnasında her bir DIS modeli "İşletir" ilişkisi ile ilişkilendirildiği modelin zaman ve olay yönetimleri ile senkronizasyon işlemlerini üstlenir.

6. Sonuç

Geliştirilen çözüm ile modelleme katmanından simülasyon işletim algoritmalarına müdahale edilerek işletimin farklı senkronizasyon algoritmaları ile yönetilebileceği bir tasarım oluşturulmuştur. Çözümün sağlanmasında bir simülasyon ve etmen programlama ortamı olan EtSiS tarafından sağlanan dil ve çoklu programlama paradigmasının önemli bir desteği olmuştur.

Geliştirilen kural tabanı ve algoritma seçimleri model davranış betiklerinin yorumlanarak işletilebilir olması sebebiyle işletim zamanlı değişimleri mümkün

kılınmıştır. Çözüm modeli ile bir dağıtık simülasyon yöneticisi geliştirmenin ötesinde, yeni koşum senkronizasyon algoritmalarının denenmesini sağlayacak bir programlama ortamı kazandırılmıştır ve simülasyon yönetimi uygulama katmanı ile ilişkilendirilmiştir. EtSiS çekirdek yorumlayıcısı yalnızca bir dil yorumlayıcısı olarak kullanılmıştır. Çalışmanın sonraki aşamasında farklı algoritma seçimleri için performans metrikleri oluşturulacaktır.

7. KAYNAKÇA

[1] Mehmet F. HOCAOGLU, "Agent based Simulation: Lecturer Notes," AUZEF, Istanbul University, Istanbul, 2014.

[2] M. F. Hocaoglu, "AdSiF : Agent Driven Simulation Framework," *Hunstv. Simul. Conf. - HSC2005*, 2005.

[3] M. F. Hocaoğlu, "AdSiF : Agent driven Simulation Framework," in *USMOS 2011- National Defense Application and Modeling & Simulation Conference -(in Turkish)*, 2011.

[4] M. F. Hocaoğlu, "Dağıtık Etkileşimli Simülasyonda (DIS) Etmen Tabanlı Zaman Senkronizasyon Yönetimi - Sistem Gereksinim Özellikleri Dokümanı." Tübitak Teydeb-AgenaBST Prj Kod: 7120778, İstanbul, 2012.

[5] M. F. Hocaoğlu, "Dağıtık Etkileşimli Simülasyonda (DIS) Etmen Tabanlı Zaman Senkronizasyon Yönetimi - Yazılım Gereksinim Özellikleri Dokümanı." Tübitak Teydeb-AgenaBST Prj Kod: 7120778, İstanbul.

[6] M. F. Hocaoğlu, "Dağıtık Etkileşimli Simülasyonda (DIS) Etmen Tabanlı Zaman Senkronizasyon Yönetimi - Çözüm Raporu." Tübitak Teydeb-AgenaBST Prj Kod: 7120778, İstanbul, 2012.

[7] R. M. Fujimoto, *Parallel and Distributed Simulation Systems*. Wiley Series in Parallel & Distributed Computing, 2000.

[8] T. L. Clarke, *Distributed Interactive Simulation Systems for Simulation and Training in the Aerospace Environment*. The ACM Guide to Computing Literature, 1995.

[9] A. Tolk, *Engineering Principles of Combat Modeling and Distributed Simulation*. Wiley, 2012.

[10] M. F. Hocaoğlu, "AdSiF: Developer Guide." Agena Information & Defense System Ltd. www.agenabst.com, Istanbul, 2013.

[11] M. F. Hocaoğlu, "Conceptual Model and Perdurantist Modeling with Reasoning," *Simul. Notes Eur.*, vol. 24, no. 2, pp. 95–104, 2014.

[12] M. F. Hocaoğlu, "Aspect Oriented Programming Perspective in Agent Programming," in *USMOS 2013- National Defense Application and Modeling & Simulation Conference -(in Turkish)*, 2013.

[13] Wooldridge, M., *An introduction to MultiAgent Systems*. John Wiley & Sons, Ltd, 2002.

[14] D. J. Murray-Smith, *Continuous System Simulation*. Chapman & Hall, 1995.