MÜHENDİSLİK FAKÜLTESİ

Uluslararası Mühendislik Araştırma ve Geliştirme Dergisi
International Journal of Engineering Research and Development

UMAGD, (2019) 11(1), 342-349.

10.29137/umagd.481584

Cilt/Volume:11 Sayı/Issue:1 Ocak/January 2019

Araştırma Makalesi / Research Article

Sürekli Olay Simülasyonlarında Kesiklileştirme ve Semantik Bilgiye Dayalı Sorgu Yapıları

Discretizing Continuous Event Simulation and Semantic Information Based Query Structures

Mehmet Fatih Hocaoğlu

[1] *İstanbul Medeniyet Üniversitesi, Endüstri Mühendisliği Bölümü, 34700,Göztepe Istanbul, TÜRKİYE*

***Başvuru/Received**: 12/11/2018* ***Kabul/Accepted**: 20/01/2019* ***Son Baskı/Final Edition**: 31/01/2019*

Öz

Bu çalışmada, sürekli olay simülasyonlarının durum vektörüyle kesikli bir uzaya resmedilmesi, etmen tabanlı zeki koşum kayıt mekanizması ve sorgu betik yapısı ele alınmıştır. Zeki kayıt mekanizması, "Doğru verinin, doğru zamanda kaydedilmesi" olarak tanımlanmıştır. Simülasyon modelinin yürüttüğü tüm hesaplamalar ve muhakeme mekanizmasıyla (reasoning engine) yaptığı tüm çıkarımlar, koşum izi etmeni olarak tanımlanan ve durum programlama yaklaşımıyla kontrol edilen, yapılar olarak tanımlanır. Modeller arası semantik ilişkilendirmelerin koşum etmenlerine olan fonksiyonel yansımaları için betik yapılar tasarlanmış ve bu amaçla, kaydedilen veriler için özel bir betik sorgu yapısı geliştirilmiştir. Simülasyon modellerinin ve etmenlerin geliştirildiği ve işletildiği EtSiS (Etmen tabanlı Simülasyon Sistemi)'in ontolojik betimlemesinin bir parçası olan, ilişki kavramının işletim sonrası koşum yörüngesi sorgulamada nasıl kullanıldığı ve kayıt mekanizmasının yönetiminin nasıl yürütüldüğü, bir örnekle, gösterilmiştir.

Anahtar Kelimeler

"EtSiS, Simülasyon çıktı analizi, Simülasyon yörüngesi, Simülasyon yörünge sorgusu, Sorgu betiği, Varlıklar arası ilişki"

Abstract

In this study, a method to map continuous event simulations into discrete event spaces using state vectors, an agent-based intelligent execution trace mechanism and a query script structure are represented. The intelligent trace mechanism is defined as "the correct data being recorded at the right time". All calculations carried out by simulation models and all inferences made by the reasoning engine are defined as structures that are defined as trace agents and controlled by State Oriented Programming approach. Scripting structures have been designed for the functional implications of the inter-model semantic associations to the execution, and for this purpose a special scripting query structure has been developed for the traced data.

How the concept of relationship, which is a part of ontological commitment of AdSiF (Agent driven Simulation Framework), which all simulation models and agents are developed and executed, is used in questioning the execution trajectory and how the management of the recording mechanism is carried out are shown with an example.

Key Words

"AdSiF, Execution Trajectory, Post-data analysis, Query on trajectory, Query script, Relation between entities"

Sorumlu Yazar: mfatih.hocaoglu@medeniyet.edu.tr

1. GİRİŞ

Etmen tabanlı simülasyon sistemi (EtSiS) kesikli ve olay simülasyonları için modelleme ve işletim imkânı sağlarken, aynı zamanda simülasyon modellerinin muhakeme yeteneği olan zeki etmen olarak tasarlanmasına imkân verir (Mehmet Fatih Hocaoglu, 2005, 2011). EtSiS ortaya koyduğu durum tabanlı programlama paradigması, tanımladığı ontolojik betimleme ve betik programlama grameri, EtSiS'i bir programlama ortamı, bir yazılım geliştirme çerçevesi (Framework), olarak tanımlamamızın ana etkenleri arasındadır.

EtSiS model tasarımında ana odak birden fazla paradigmayı tümleştiren durum tabanlı programlama paradigmasıdır. Literatürde durum tabanlı programlama kavramı ile modelin durumsallık tanımlamaları yapılarak, durum tanımlamalarının programlanması veya durum tanımlamalarına uygun kod üretilmesi olarak ele alınmıştır (Nomoto, 2004; Sterkin, 2008). 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 EtSiS çekirdek yorumlayıcısı tarafından işletilirler (Mehmet Fatih Hocaoglu, 2011). Durum tabanlı programlama, sağladığı durum otomat gösterimi ile bir doğal dil karşılığıyla eşleştirilebilen davranışlar ve beraberinde zeki bir koşum kayıt mekanizması sunar.

Bu çalışmada, EtSiS'in ontolojik betimlemesinin bir parçası olan, ilişki kavramına yeni bir işlev kazandırılmıştır. Bir ilişkinin taraflarının sahip olduğu koşum izi etmenleri arasında bir fonksiyon eşlemesi yapılarak varlık koşum izi etmen değerleri arasında geçişkenlik sağlanmış ve koşum sonrası sorgulama etkinliği artırılmıştır.

Literatürde simülasyon işletimi esnasında meydana gelebilecek sistem hatalarına karşı, simülasyonun kaldığı zaman noktasından tekrar başlatılmasını sağlayacak koşum enstantane kayıtları üzerinde çalışmalar görülmektedir (Meneses & Kalé, 2015). Simülasyon koşum loglarının model hata ayıklama amaçlı kullanımı (Kemper & Tepper, 2009) ve performans analizinde etkin parametrenin kural tabanlı olarak tespiti amaçlı farklı kullanımlar mevcuttur (Vasyutynskyy, Gellrich, Kabitzsch, & Wustmann, 2010). Sistem koşum yörüngesinin alternatif sistem tasarımları arasındaki farklılıkların görülmesinde istatistiksel analizler, dinamik zaman eğrilmesi (time warp) yaklaşımları kullanılmaktadır (Johnstone et al., 2015). Tüm bu analizler için yeterli uzunlukta, doğru frekansta ve sistem davranışını doğru temsil eden koşum verisinin alınması analizlerin doğruluğu ve hızlı işlenmesi için önemlidir.

Simülasyon yörünge kaydı, çoğunlukla modelcinin analiz aşamasında gerekli olacağını düşündüğü değişkenlere ait verilerin simülasyon koşumu esnasında bir kayıt mekanizması kurarak kaydetmesiyle sağlanır. Ana motivasyon mümkün olduğunca geniş bir veri kümesinin, mümkün olan en yüksek sıklıkta kaydedilmesidir. Geliştirilen çözümlerde sıklıkla kaydedilecek veri ile model arasında organik bir yazılım bağımlılığı kurulur. Bazı uygulamalarda rezerve edilmiş değişkenlere, kaydedilecek verilerin atanması ile daha esnek yapılar kurulmaya çalışıldığını görülür (Kelton, Sadowski, & Sturrock, 2006), (C. Dennis Pegden, Shannon, & Sadowski, 1995). Bu çalışmada, doğru zamanda doğru verinin kaydedilmesiyle gereksiz veri kaydının nasıl önüne geçileceği gösterilmiştir. Ayrıca, ilişkili varlıklar arasında kurulan ilişkilerin yansımaları ile hesaplanabilen verinin kaydedilmesi engellenmiştir. Simülasyon koşumlarında değişen amaçlara göre kaydedilecek verinin seçilebileceği bir mekanizma olan, koşum izi etmeni tanıtılmış ve veri kayıtları üzerinde yapılacak sorgular için geliştirilen betik dil tanıtılmıştır.

Bu çalışma, şu bölümlerden oluşmaktadır. İkinci bölümde, EtSiS'in genel tanımı, davranış, durum, davranış cepheleri (listeleri) kavramları ve ilişki tanımı detayları ele alınmıştır. Üçüncü bölümde, simülasyon işletiminde zeki koşum kaydı tasarımı ve ilişki kavramının koşum kaydı ve sorgulamasında kullanımı anlatılmıştır ve dördüncü bölümde ise verilen örnek senaryo ile kullanımı gösterilmiştir. Sonuç ve tartışmalar bölümde, tekniğin avantajları ele alınmıştır.

2. EtSiS: ETMEN TABANLI SİMÜLASYON SİSTEMİ

EtSiS cephe tabanlı programlama, mantık programlama paradigmalarını, durum tabanlı programlama paradigmasında tümleştiren, simülasyon ve etmen programlama için bir betik dil sunar. DtP temel olarak bir dil gramerine sahip olarak tasarlanmış, hiyerarşik sonlu durum otomatlarının betik programlamasıdır. EtSiS davranış olarak tanımladığı, hiyerarşik durum otomatlarının durum geçişlerini yürüterek işleten ve davranış ve içerdiği durumların eriştikleri fonksiyonları çalıştırarak simülasyon işletimini sağlar. İşletim kapsamında, durumlar için tanımlanmış zaman yönetimleri, davranışların faz yönetimleri (aktive olma, iptal olma, askıya alınma, tamamlanma, ve tekrar aktive edilme), varlıklar arasında iletilen olayların işletimleri, davranışların diğer davranışlar ile olan etkileşimleri (başlatma sonlandırma gibi faz geçişi sağlama), muhakeme algoritmalarına erişim ve muhakeme sonuçları ile davranışların yönetimi ve ilişkili simülasyon işletimine dair tüm adımlar mevcuttur (Hocaoğlu, 2018).
Bir EtSiS modeli aşağıdaki gibi 7 bileşen ile gösterilir;

$M=<X, Y, \{S, F, P, \mu\}, t_a, \partial ext, \partial int >$

X: Girdi kümesi (modeller arası iletilen ve M modeli tarafından alınan olaylar),

Y: Çıktı kümesi (M modelinin yayınladığı olaylar),

{S, F, P, μ}: Model davranışları kümesi (hiyerarşik durum otomatları formunda tasarlanmış, özelleştirilmiş yapılar). S davranış durumlarını ve F ve P ise, sırasıyla, duruma ilişkilendirilen olguları (facts) ve önermeleri (predicates) ifade eder.

t_a: S→T^{inf} zaman ilerleme fonksiyonu davranış durumlarının hayatta kalma süresini gösterir. Durumun tanımlı süreyi tüketerek çıkması durumunda iç geçiş sağlanır. Bir olay ile kesilmesi durumunda ise durum içerisinde kaldığı süre tüketilerek harici geçiş sağlanır. Her geçiş davranış diyagramında bir sonraki duruma geçişi ifade eder. Her durum geçişi bir olay yayınlama noktasıdır.

∂ext: Q x X → Y bir olayın model davranışlarında nasıl bir değişime sebep olduğunu belirleyen harici geçiş fonksiyonlarıdır. Burada Q= {(s,te)|s E S, te E (T n [0, ta(s)]}te durum tarafından harcanan son süredir.

∂int: S→S durumun tanımlı süresini tamamlayarak gerçekleştirdiği iç geçiş fonksiyonlarıdır.

μ: Durumlara ilişkilendirilmiş çıktı değer çağrılarını (fonksiyon, önerme parametresi, olay parametresi, plugin fonksiyon parametresi ve öznitelik) tanımlar ve koşum izi kayıtlarını oluşturur.

3. ZEKİ KOŞUM KAYDI

EtSiS'de iki tip koşum yörüngesi mevcuttur. Birinci tip koşum yörüngesi, simülasyon işletimi esnasında üretilen ve kaydedilmek üzere seçilen verilerin zaman etiketli değerleri olarak tanımlanır ve koşum izi etmeni olarak isimlendirilir. Koşum izi etmeni modele ait fonksiyonların geri dönüş değerleri, model öznitelik değerleri, model önerme çıktı parametre değerleri (mantık programlama), tetikleyici olay parametreleri ve davranış etki sahalı parametre tanımları (davranışa özel tanımlı parametreler) ile belirlenir.

İkinci tip koşum yörüngesi ise, bir simülasyon varlığının işletim esnasında yürüttüğü tüm davranışların, davranışların durum geçişlerinin ve işletilen mesajların zaman etiketli kaydı olarak tanımlanır ve davranış izi olarak isimlendirilir. Davranışlara ait tüm faz geçiş zamanları (başlatılma zamanı, bitirilme zamanı, askıya alınma zamanı, tekrar aktive edilme zamanı) kaydedilir. Davranışın faz geçişlerine ek olarak, bir davranışın işletimi esnasında hangi durumları işlettiği, durumların giriş ve çıkış zamanları kaydedilir. Örnek olarak, bir uçağın uçuşu süresince sahip olduğu irtifa değerleri, hızı, oryantasyon bilgileri, yakıt miktarının her biri birinci tip koşum yörüngesi olarak tanımlanır. Uçuşu esnasında yürüttüğü her bir davranışın faz geçişleri, davranışların yürüttükleri durum geçişleri zaman etiketli olarak ikinci tip koşum yörüngesini oluşturur.

Modele ait her bir davranışın bir doğal dil ifade karşılığı vardır. Davranış izi koşum yörüngesi, bir modelin senaryo boyunca yürüttüğü davranışların, diğer bir ifadeyle, neler yaptığının durum tabanlı bir dil ile ifade edilmesidir. Davranış koşum yörüngelerinden hangi durumda ne kadar süre ile kalındığının tespiti mümkün olduğu için modelin hangi davranışı ne kadar süreyle yaptığı sonucu da çıkarılabilir. Benzer şekilde, hangi davranışların paralel olarak ne kadar süreyle yürütüldüğü, hangi davranışın başlatılmasıyla hangi davranışın sonlandırıldığı aynı şekilde sorgulanabilir.

Daha önce de ifade edildiği gibi, zeki simülasyon yörünge kaydı doğru verinin doğru zamanda kaydedilmesi olarak tanımlanır. Bir modelin modelleme çözünürlüğünü davranışların durum çözünürlüğü belirler. Durumlar ne kadar küçük zaman adımlı ve ne kadar atomik seviyede işlev çözünürlüğe sahip ise model çözünürlüğü o kadar yüksektir. Koşum izi etmeninin içerdiği veri yapısı hangi durum içerisinde üretiliyor ise, koşum izi etmeni ilgili durum içerisinde deklare edilir. Bu verinin yalnızca üretildiği ve/veya değerinin değiştiği zamanlarda kaydedilmesini sağlar. Koşum izi etmeni tanımlaması hangi veri setinin hangi durum içerisinde kaydedileceğinin tanımlamasını içerir. Tanımlama yapılan durumları içeren davranışların ilgili durumu işlettikten sonra, durum çıkış fazında koşum izi kaydı gerçekleşir. Koşum izi kayıt değeri, koşum izi etmen deklarasyonundan alınır ve zaman etiketi olarak ilgili durumun kayıt fazı olan çıkış fazı zamanı olarak belirlenir.

Genel olarak, koşum izi kaydı veya simülasyon yörüngesi kaydında karşılaşılan sorunu aşağıdaki gibi özetleyebiliriz;

- Sürekli olay simülasyonlarında, zaman adımlı işletimlerde her zaman adımında durum vektörünün kaydedilmesi; bu durumda durum değişkenlerinde bir değişim olup olmadığı kontrol edilemeden tüm vektörün kaydedilmesi kayıt hacmini artırır ve işletim hızını düşürür. Kullanıcı tanımlı olarak durum vektörü, sürekli olay simülasyonlarında bir modelin kalitatif değer noktalarını içerir. Şöyleki, bir kovanın su ile dolduruluyor olması sürekli bir olaydır ve durum vektörü kovanın boş olması ve tam dolu olması gibi iki kalitatif değer ile tanımlanabilir. Ara tüm değerler, kovanın bir miktar dolu olması ve taşması, tanımlı kalitatif değerler arasında yeralır (Mehmet Fatih Hocaoglu, n.d.). Benzer şekilde bir uçağın uçuşunda her bir simülasyon ilerleme adımı, sonraki rota bacağı noktasına olan süre, kalan yakıtın harcanma süresi ve çizelgelenen bir çarpışma olayı varsa olay gerçekleşme zamanı ile durum vektörüne eklenir. İlerleme adımı minimum zamanlı adım olarak belirlenir ve yöntemle simülasyon işletimi kalitatif değer noktalarına erişim zamanının hesaplanması ile kesiklileştirilir.

- Kesikli olay simülasyonlarında tanımlı olay, kesikli zaman adımlarında veya sabit zaman aralıklarında durum vektörünün kaydedilmesi;

    - Sabit zaman aralıklarında durum değişimlerinin yakalanamaması durumu ile karşılaşılır.
    - Kesikli zaman kayıt durumunda sürekli olaya göre göreceli olarak daha az kayıt alınabilse de kayıt zaman tabanlı yapılabildiği için bir filtre mekanizması, ancak, probleme özgü geliştirilebilir.
    - Tanımlı şartlara göre kayıt en az kayıt hacmini doğurur fakat kaydedilecek değerlerin filtrelenmesi ve kayıt şart tanımları probleme özgü tanımlanabilir.

Geliştirilen yaklaşımın iki çözüm ayağı vardır. Birinci ayak, koşum izi kayıtlarının tanımlı durum geçişlerinde ve yalnızca ilgili duruma ilişkilendirilmiş verilerin kaydının sağlanmasıdır ve EtSiS bunun için tanımlı bir gramere sahiptir. Dil betimlemesi

içerisinde yürütülen ve model kodları ile derlenmeyi gerektirmeyen bir yapı sunulduğu için herhangi bir çözüm ile yazılım bağımlılığı doğurmaz.

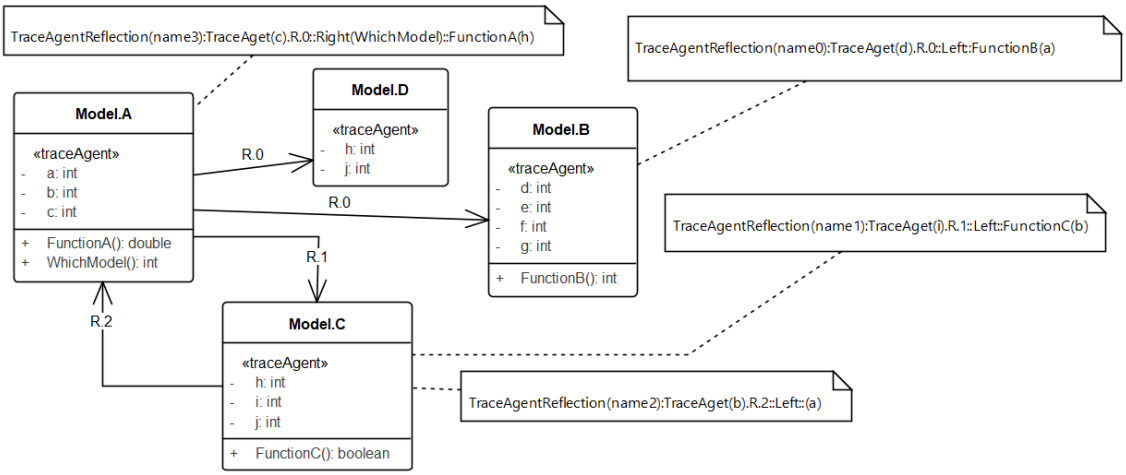
Çözümün ikinci ayağı ise aralarında ilişki tanımlanmış varlıkların koşum yörüngelerinin, ilişki üzerinden tanımlanmış fonksiyonlar ile hesaplanabilir veriler olarak kullanımıdır.

3.1. İlişki Tabanlı Koşum Sorgulama

Modeller arasında kurulan ilişkiler üç farklı amaç için kullanılır. Bunlardan ilkinde, ilişkiler tarafların mesajlaşmaları için bir haberleşme kanalı oluştururlar. Varlıklar birbirlerine doğrudan kimlik bilgisine sahip olmadan mesaj iletimini ilişki tanımı üzerinden yürütebilirler. Örnek olarak, A modeli ile B modeli arasında R ilişkisi tanımlanmış olsun ve bu ilişkilendirmede A ilişkinin sol, B ise sağ tarafında yer alsın (A→RB). A modeli B modelini "R isimli ilişki ile ilişkilendirildiği ve bu ilişkinin sağında yeralan varlık" olarak dolaylı bir tanımlama yaparak, modelin kimliğini bilmeden olay mesajı iletimi yürütebilir. Aynı, ilişki farklı modeller ile de tanımlanabilir, bu durumda ilişkinin diğer ucunda hangi varlığın olduğundan bağımsız bir haberleşme ağı sağlanır. Özellik model seviyesinde bir soyutlama sağlarken (abstraction), farklı modeller ile olan aynı olay etkileşimi durum tabanlı programlama yaklaşımında çok biçimlilik (polymorphism) olarak ele alınabilir.

İlişkilerin kurulması ve kaldırılması safhalarında tarafların tanımlı fonksiyonları işletmeleri ve tanımlı davranış listelerini aktive etmeleri sağlanır. İkinci tip kullanım olarak nitelendirebileceğimiz kullanım, ilişkinin kurulması ve kaldırılması aşamaları için bir hazırlık eylemi yürütürken, aktive edilen davranış cepheleri ile dinamik cephe yönetimi sağlanır. İlişkinin varlığı ve yokluğu farklı davranış yapılarına, farklı modelleme cephelerine dinamik geçişi sağlar (M Fatih Hocaoglu, 2017).

Üçüncü kullanım konsepti ise, ilişkilendirilen varlıklar arasında bir koşum izi etmeni değer eşlemesi amacıyla kullanımıdır. Şekil *1*'de görüldüğü gibi, ilişkilendirilen iki varlığın öznitelikleri arasında bir eşleme yapılmıştır. Eşlemede, hangi modelin hangi koşum izi etmeninin, ilişkili varlığın hangi koşum izi etmeni değeri ile, hangi fonksiyon kullanılarak belirleneceği gösterilmiştir.



Şekil 1. Varlıklar arası ilişkiler ve koşum izi etmeni ilişkilendirme

Şekil *1*'deki A modeli ile B modeli arasında kurula R0 ilişkisi ile Model.B'ye ait koşum izi etmeni "d"nin değerinin, ilişkinin solunda yeralan modelin koşum izi etmeni "a"nın değerinin FunctionB ile hesaplanarak tespit edileceğini göstermektedir. Kurulan bu ilişkilendirme "Trace Agent Reflection" olarak isimlendirilmiş ve bir isim verilmiştir. Burada ilişkinin solunda yalnızca bir adet model olduğu için koşum izi etmeninin hangi modele göre belirleneceğinin doğrudan belirtilmesine gerek yoktur. Oysa, Model.A üzerinde yapılan deklarasyonda, R.0 ilişkisi ile ilişkilendirilen iki adet model mevcuttur. Bu durumda, hangi modele ait koşum izi etmeni değerinin Model.A'nın "c" isimli koşum izi etmeninin değerinin belirlenmesinde kullanılacağı bir fonksiyon ile belirlenir. Örnekte, fonksiyon "WhichModel" olarak girilmiştir. Tespit edilen model kimliği kullanılarak, örnekte Model.D olarak belirlenen modelin "h" isimli koşum izi etmeni, yine Model.A'ya ait olan FunctionA isimli fonksiyonun girdi parametresi olarak kullanılarak değeri belirlenir. Eğer bir koşum izi etmeni herhangi bir fonksiyondan geçirilmeden, değeri doğrudan bir değer etmene eşlenecek ise ya herhangi bir fonksiyon deklarasyonu yapılmaz ya da bir boş fonksiyona gönderilir ve aşağıdaki gibi tanımlanır.

```
double Model.C::EmptyFunction(double a)
{return a;}
```

Şekil *1*'de Model.C ile Model.A arasında R.2 ilişkisi kullanılarak yapılan deklarasyon değerin doğrudan kullanımına örnek olarak verilmiştir.

Simülasyon koşumun hızının artırılması için zaman zaman birbiri ile ilişkili olan varlıkların pozisyon güncellemesi gibi işlemler, ilişkili varlığın diğer varlığı kullanım aşamasına kadar ertelenir. Örnek olarak, bir füze taşıyan uçağın her hareketinde füzenin

pozisyon ve oryantasyon bilgisinin de güncellenmesi gerekir fakat bu füzenin ateşlenmesine kadar ertelenebilir. İkinci bir durum ise, ilişkili varlıklar arasında eşleştirilmiş koşum izi etmenleri birbirlerine göre hesaplanabileceği için simülasyon esnasında ilişki varolduğu sürece hesaplanabilen koşum izi etmeni için kayıt alınmaz. Füze pozisyon ve oryantasyonu aralarında "*Taşır*" ilişkisi varolduğu sürece uçağın pozisyonuna ve oryantasyonuna göre hesaplanabildiği için, koşum kayıt hacmini azaltmak adına, ilişkinin kurulu olduğu süre boyunca, birbirleriyle ilişkilendirilmiş olan koşum izi etmeni kayıtlarının alınmaması buna örnek olarak verilebilir. İlişkinin kaldırılması, örnek olarak füzenin ateşlenmesiyle, kayıt alınmasına bir başlangıç olacak ve koşum izi etmenleri veri kaynaklarından biri olan model öznitelik değerleri için ilk değerler ilişkili varlık tarafından atanacaktır. F16'nın füzeyi ateşlediği anda pozisyon, oryantasyon ve ilk hız değerlerini atayıp "*Taşır*" ilişkisini kaldırması tipik bir örnek durumdur.

Bu tercih koşum izi kayıtları sorgulanmasında bir farklılığı gerektirir. Bu amaçla EtSiS'in koşum izi etmeni ve davranış izi sorgulamaları için bir betik sorgu diline ilişki sorgu kavramı dahil edilmiştir. Davranış izi betik sorgusu, sorgulanacak davranışın faz sorgularını, örneğin, davranışın ne zaman aktive olduğu ne zaman tamamlandığı, ne zaman iptal edildiği vb., ve davranış durum sorgularını içerir. Örneğin, F16'nın *Fsa_Fly* isimli davranışının içerdiği *St_Fly* isimli duruma giriş ve çıkış zamanları sorgulanabilir. Birden fazla sorgu mantıksal ifadeler ve matematiksel operatörler ile birleştirilerek karmaşık sorgu olarak tanımlanabilir. Şekil 2'de bir durum sorgu betiği görülmektedir. Sorgu "*Plane*" tipindeki tüm varlıkların "*Fsa_Move*" davranışının aktive edilmesi ile ilgilidir ve köşeli parantez içerisinde yazılan davranış aktivasyon zamanı, davranıştan çıkış zamanı, davranış aktif kalma süresi sorgu sonucunda alınmıştır. Sorgu Scn1 isimli senaryo koşumu için yürütülmüştür.

SCENARIO<Scn1>→MODELTEMPLATE<Plane>→FSA<Fsa_Move>→PHASE<ACTIVATED>)

←[FSAENTRYTIME, FSAEXITTIME, FSADURATION]

Şekil 2. Davranış İzi Sorgusu

Şekil 3'de görülen sorguda plane tipindeki tüm modellerin Fsa_Move davranışı aktivasyon fazı sorgulanırken "ve" mantıksal operatörü ile F16-1 tekil modelinin (instance) aynı zamanda Fsa_Active isimli davranışının askıya alınıp alınmadığı sorgulanmaktadır. Söz konusu faz geçiş zamanları sorgu sonucu olarak alınmaktadır.

((SCENARIO<Scn1>→MODELTEMPLATE<Plane>→FSA<Fsa_Move>→PHASE<ACTIVATED>) &&
(Scn1>→ENTITY<F16-1>→FSA<Fsa_Active>→PHASE<SUSPENDED>))←[FSAENTRYTIME]

Şekil 3. Davranış izi sorgusu

Koşum izi etmeni üzerinde yürütülecek betik sorgu yürütülecek senaryo koşumu ismini ve sorgulanacak koşum izi etmenini içerir. Sorgulama koşum izi etmeninin değer tipine bağlı olarak matematiksel filtrelemeler (>, <, ≥, ≤, ≠, ==) ve mantıksal ifadelerle (and, or) birleştirilmiş karmaşık ifadeler olarak tanımlanabilir. Şekil 4'deki Yukseklik isimli koşum izi etmeninin 20. zaman noktasından daha sonraki değerleri ve Hız isimli koşum izi etmeninin 340. zaman noktasından önceki değerleri, YakıtTuketimi isimli koşum izi etmeninin ve Hız etmeninin değerlerinin 200'den büyük veya eşit olduğu sorgu değerleri (VALUE) elde edilir (Şekil 4). Karmaşık sorgu yapısında da görülebileceği gibi, her bir sorgu bileşeni farklı senaryolarla ilişkilendirilebildiği için farklı senaryo koşumları arasında kıyaslama yapabilme imkânı sunulur.

((SCENARIO<Scn1>→ENTITY<F16-
1>→TRACEAGENT<Yukseklik>→TIME≥<20>)&&((SCENARIO<Scn1>→ENTITY<F16-
1>→TRACEAGENT<Hiz>→TIME≤<340>)&&((SCENARIO<Scn1>→ENTITY<F16-
1>→TRACEAGENT<YakitTuketimi>→VALUE≥<200>)&& (SCENARIO<Scn1>→ENTITY<F16-1>-
→TRACEAGENT<Hiz>→Value ≥ 200)))) ←[VALUE]

Şekil 4. Karmaşık koşum izi etmeni sorgusu

Koşum sonrası sorgulamada ilişkinin mevcut olduğu durumlarda sorgu betiği sorgulanan koşum izi etmeninin ilişkili varlık üzerinden gerçekleştirir. İlişki kaldırıldıktan sonraki durumlar için ise sorgu doğrudan varlık üzerinden yürütülür. Şekilde görülen Model.A ve Model.B arasında kurulan koşum izi etmen eşlemesine ait sorgu aşağıdaki şekilde tanımlanır. Model.B'ye ait d isimli koşum izi etmeni değeri name0 isimli eşleme (Reflection) belirlenir. Belirlemede Model.A'ya ait "a" isimli koşum izi etmen değeri FunctionB isimli fonksiyona gönderilir ve elde edilen geri dönüş değeri Model.B "d" koşum izi etmeni sorgusu olarak kullanılır. Sorgu geri dönüş değerleri ise değer (VALUE) ve zaman (TIME) olarak belirlenmiştir. Sorgulama Model.A'nın "a" koşum izi etmeni üzerinden yürütülse de koşum izi etmeni sorgusu için alınan değerler FunctionB'de hesaplanarak koşum izi etmeni "d"'nin değeri bulunur ve bu değer koşum izi etmeni "a"'nın ilgili zamandaki değer karşılığıdır.

SCENARIO<S0>→ENTITY<Model.B>→TRACEAGENT<d::Reflection(name0)>

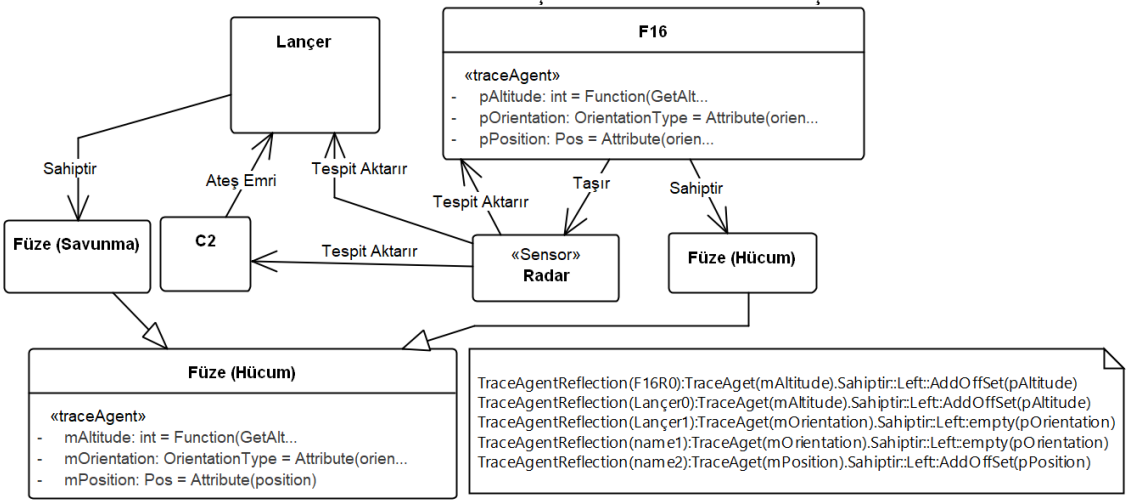
→Value≥200)))←[VALUE, TIME])

Şekil 5. İlişkili varlıklar koşum izi etmeni sorgusu

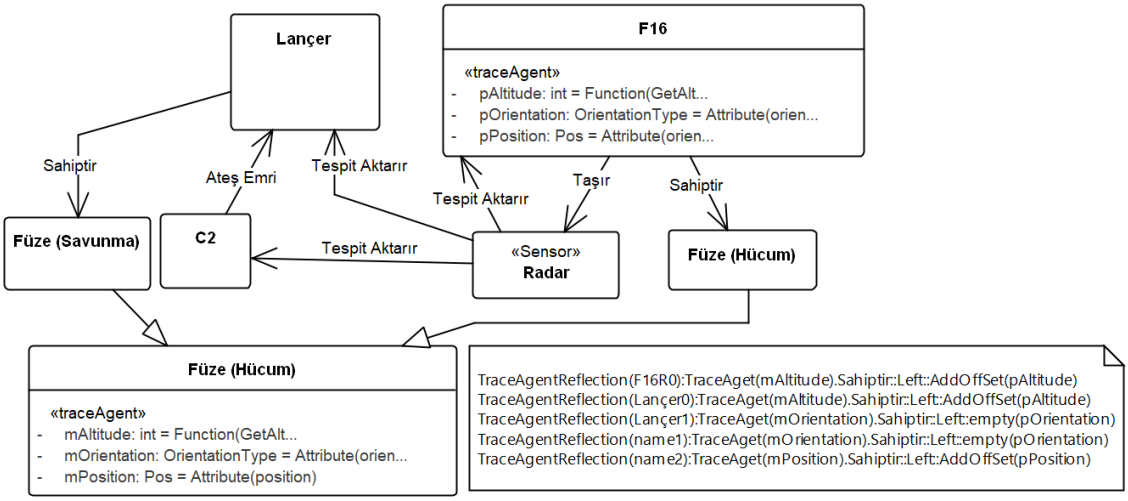
Tanımlama yapılarından kolaylıkla görülebildiği gibi, varlıklar arasında kurulan fonksiyonel bir ilişkilendirme ile yapılan bir koşum kaydından, ilişkili varlık için ilgili zamana ait bir yörünge değeri hesabı yapılmıştır.

4. ÖRNEK SENARYO

Senaryoda kırmızı ve mavi taraflara ait iki F16 uçağı üçer adet havadan yere hücum füzeleri taşımaktadır. Yerde ise bir füze savunma sistemi konuşlanmıştır. Sistem bir komuta kontrol modeli, bir radar, bir lançer ve 10 adet yerden havaya savunma füzelerden oluşmaktadır. Hücum füzeleriyle F6'lar arasında ve lançer ile savunma füzeleri arasında "Sahiptir" ilişkisi kurulmuştur. Varlıklar arasında kurulan ilişkiler ve koşum izi etmen ilişkilendirmeleri



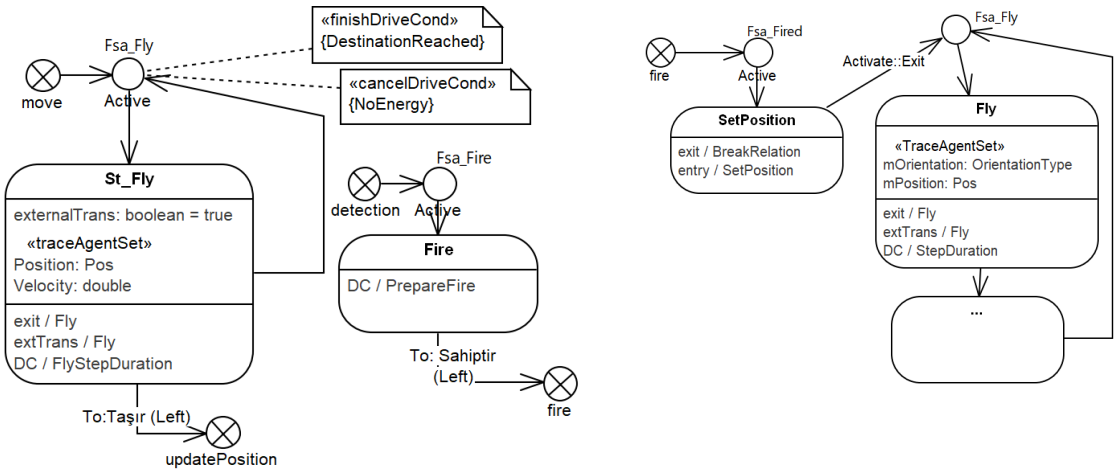
Şekil 6'de gösterilmiştir. Şekilde koşum izi etmen tanımlamaları ve etmenler arası ilişkilendirmeler görülmektedir. Koşum izi etmenlerinin içerdiği veri deklerasyonda belirtilmiştir (Attribute(position), Function(GetAltitude)). Şekilden de görülebileceği gibi, hücum ve savunma füzeleri bir temel füze modelinden türetilmişlerdir ve koşum izi etmenleri (trace agents) miras yoluyla alınır.



Şekil 6. Varlıklar arası ilişkiler ve koşum izi etmeni ilişkilendirme

Uçak modeli "*move*" olayı aldığında Active durumu içerisinde ise *Fsa_Fly* isimli davranışı aktive eder. Davranış döngüsel bir davranıştır ve durum çıkışında kendisine verilen zaman ilerlemesine veya kendi istediği *FlyStepDuration* zaman adımı kadar *St_Fly* durumunu işleterek uçar. Durum çıkışında tanımlı koşum izi etmenlerinin kaydını gerçekleştirir. Sensörler uçaklarla ilişkilidir (Taşır ilişkisi ile) ve uçakların her hareketinde pozisyonları, *updatePosition* olayıyla, güncellenmektedir (Şekil 7).

Senaryoda uçaklar birbirlerini, sahip oldukları sensörlerle, hedef tespit ettikleri anda füzelerini ateşlemektedirler. *Detection* olayını alan *F16* modeli *Fsa_Fire* davranışını aktive eder ve Fire durumunda *PrepareFire* fonksiyonu tarafından hesaplanan süre kadar bekledikten sonra, "*Sahiptir*" ilişkisi ile ilişkilendirilmiş olduğu varlığa (bu durumda füze varlığı) *fire* olayını gönderir. F16'ların konumları değişmiş olmasına rağmen, etkin bir koşum hızı için, "*Sahiptir*" ilişkisi ile sahip olduğu füzelerin pozisyon ve oryantasyon bilgilerini güncellemez. Ateş emri alan füze (fire olayı ile) olay parametresi ile iletilen pozisyon ve oryantasyon parametrelerini alır ve *SetPosition* durumunda kendi pozisyonu olarak atar ve uçak ile arasında kurulu ilişkisini kaldırır (Şekil 7). Durum çıkışında füzenin uçuş davranışı olan *Fsa_Fly* aktive edilir (Davranış özet olarak verilmiştir). *Fsa_Fly* davranışı *Fly* durumu içerisinde deklare edilmiş olan *position* ve orientation isimli koşum izi etmenlerinin ürettikleri değerler durum çıkışında kaydedilir. Koşum sonrası analizde, füze *pozition*, *orientation* ve *mAltitude* koşum etmeni üzerinde yapılan sorgulamalarda, "*Sahiptir*" ilişkisi kaldırılıncaya kadarki kısım için tanımlı ilişkilendirme dikkate alınarak (reflection) yürütülür.



Şekil 7. Uçak Davranışları ve Füze davranışları

Sorgu betiği *mAltitude* koşum izi etmeni için Şekil 8'de görülmektedir. Sorguda füzeye ait *mAltitude* isimli koşum izi etmeni F16 modeline ait *pAltitude* isimli koşum izi etmeni üzerinden sorgulanmakta ve *pAltitude* koşum izi etmeninin alınıp *AddOffset* isimli fonksiyona girdi parametresi olarak gönderilmesi ve fonksiyon geri dönüş değerinin füze yükseklik değeri olarak alınması ile sağlanmıştır. Alınan yükseklik değeri 200 değerinden büyük veya eşitse değer ve değerin tespit edildiği zaman sorgu sonucu olarak döndürülmektedir.

(SCENARIO<Scn1>→ENTITY<Füze-1> →TRACEAGENT< mAltitude::Reflection(F16R0)>

→Value >= 200))))←[VALUE, TIME])

Şekil 8. İlişki durumunda füze yüksekli sorgusu

Lançer ile füze arasında kurulan Lançer0 isimli koşum izi ilişkilendirme tanımı sonucunda, hücum füzelerinin pozisyonu, ateşleninceye kadar ilişkili oldukları lançer pozisyonu üzerinden hesaplanacaktır. Radar tarafından komutana iletilen tespit doğrultusunda (C2 ve Radar arasındaki "Tespit Aktarır" ilişkisi üzerinden) komutan (C2 varlığı) "Ateş Emri" ilişkisi ile ilişkilendirildiği lançere angajman emri verir. Radar ile lançer arasındaki "Tespit Aktarır" ilişkisi ile lançer hedefi ateşleme gerçekleşinceye kadar takip eder. Ateşlenme sonrası ilişki kaldırılacağı için pozisyon bilgisi kendi koşum izi etmeni tarafından pozisyon özniteliği kullanılarak kaydedilecektir. Benzer şekilde, Lançer1 isimli ilişkilendirmeyle, lançer namlu oryantasyonu ile savunma füzesi oryantasyonu eşlenmiştir. Senaryo varlıkları arasındaki tüm iletişim ilişki tanımları üzerinden gerçekleşir. Bu bir varlığın mesajını ilişkili varlığa gönderirken ilişkinin diğer ucunda hangi varlığın olduğunu bilmesinin gerekli olmadığını gösterir. Varlıklar arasındaki bu doğrudan olmayan ilişki, simülasyondaki varlıkların birbirlerinden tamamen bağımsız olmalarını sağlar.

5. SONUÇ VE TARTIŞMALAR

Sürekli olay simülasyonlarında sıklıkla kullanılan zaman dilimleme yönteminde (Cellier, F. E., Kofman, 2006; Murray-Smith, 1995), belirlenen her bir zaman adımında modele ait bir durum vektörü oluşturulur ve zaman adımlarında yörünge kaydı gerçekleştirilir. Bu yaklaşımda, durum vektörünü uzun süre değiştirmeyen varlıklar için aynı verinin gereksiz olarak çok defa kaydedilmesini, zaman adımından daha kısa sürede durum vektörü değişimi yapan varlıklar için ise adımlar arasında kalan değişimlerin kaydedilmemesi sonuçlarını doğurur. Durum vektörleri, modele ait kalitatif değer noktalarını da sıralı olarak içerir. Simülasyon kalitatif değer noktaları üzerinden ilerletilerek kesiklileştirme sağlanır. EtSiS'de durum vektörü, zaman dilimleme yönteminde olduğu gibi her zaman adımında değil, değer değişimlerinde ve kesiksizleştirilmiş durum çıkışlarında gerçekleştirilir. Bu sabit zaman adımlı işletime göre daha hızlı ve alınan kayıtların daha anlamlı olmasını sağlar. Çözüm hesaplama duyarlılığını da yükseltir. Çünkü çok küçük zaman adımlarında çift duyarlıklı (double precision) hatası yükselirken, büyük zaman adımlı işletimlerde Fourier yuvarlatma hataları yükselir. Sürekli olay simülasyonları için EtSiS'in sağladığı durum tabanlı kesiklileştirme, sürekli ve kesikli olay simülasyonlarının her ikisi için, çözümü ortak bir çözüm haline getirir.

Kesikli olay simülasyonlarında koşum kaydı tanımlı olayların oluşumu, kullanıcı tanımlı şartların sağlanması ve sabit veya değişken zaman aralıklı koşum kaydı tanımı yapılabilse de, bu ancak uygulama bağımlı bir çözüm olarak, yüksek yazılım bağımlılığı ile gerçekleştirilebilmektedir. Mevcut çözüm belirtimsel bir koşum kayıt yapısı ve betik bir sorgu yapısı sunduğu için herhangi bir yazılım bağımlılığı sunmaz ve tüm tanımlamalar için bir gramer sunar. Koşum izi etmeni tanımlamaları model betik yapısı içerisinde yeraldığı ve model kaynak kodu ile derlenmediği için farklı simülasyon koşum analizleri için, model iç yapısına müdahale edilmeden değiştirilmeleri mümkündür. Bu esneklik, zayıf yazılım bağımlılığı (loosely coupled), ortogonality gibi pek çok yazılım mühendisliği kriterini destekler.

Geliştirilen çözüm ile varlıklar arasında mevcut olan ilişkilerin, koşum yörüngesine olan semantik yansıması tanımlanmıştır. Bunun için ilişkinin varlıklar arasında hangi koşum kaydının nasıl bir fonksiyon ile ilişkili varlığa yansıtıldığı tanımlanmıştır. Koşum sonrası analizler için bu yansımanın sorgu betikleri tanımlanmıştır. Yapılan tanımlamalar deklarasyon düzeyinde olduğu için hem geliştirilmeleri, hem de farklılaşan amaçlar için yeni eklenti ve değişiklikler için esnek bir yapı sunar.

Geliştirilen kayıt ve sorgu mekanizmasıyla daha düşük hacimli veri kaydı imkânı sağlanmıştır. Bu kayıt performansını artırırken, gereksiz sıklıkta veri kaydının önüne geçer ve seçilen zaman adımından daha küçük zaman adımlı eylemlerden doğan değer değişimlerinin kaydedilmeme sorununu da ortadan kaldırır. Zira, zaman adımı ne kadar küçük olsa da, her durum geçişi tanımlı koşum izi etmeleri için kayıt anlamı taşır. Sunulan sorgu betik yapısıyla, genel bir veritabanı kuralı olan, hesaplanabilen verinin kaydedilmemesi ilkesi, sağlanır. Bir modele ait koşum kaydı, ilişkili olduğu modelin ilişki fonksiyonu ile hesaplanarak oluşturulur ve bu sorgu performansını artırır.

Önerilen çözüm ile, geleneksel olarak uygulanan, her simülasyon senaryo varlığının koşum kayıtlarını gerçekleştirmesi yerine, varlıklar arasında tanımlı ilişkilere atanmış fonksiyonlar kullanılarak, sorgulamaların kurulan semantik ilişkiler üzerinden yapılması sağlanmıştır. Analiz, sorgu ve kayıt avantajlarına ek olarak, ilişki varlıklar arasında birbirlerinden bağımsız, birbirlerinin iç yapılarına dayanmayan bir iletişim kanalı oluşturur.

REFERANSLAR

C. Dennis Pegden, Shannon, R. E., & Sadowski, R. P. (1995). Introduction to Simulation Using Siman. McGraw-Hill.

Cellier, F. E., Kofman, E. (2006). Continuous System Simulation (Springer.). Springer.

Hocaoğlu, M. F. (n.d.). Qualitative Reasoning for Quantitative Simulation. https://doi.org/10.1155/2018/7842402

Hocaoğlu, M. F. (2005). AdSiF : Agent Driven Simulation Framework. In Joseph S. Gauthier (Ed.), Hunstville Simulation Conference -HSC2005. Huntsville, Alabama.

Hocaoğlu, M. F. (2011). EtSiS: Etmen tabanlı Simülasyon Sistemi. In 4. Ulusal Savunma Uygulamaları Modelleme ve Simülasyon Konferansı, USMOS'2011.

Hocaoğlu, M. F. (2017). Aspect Oriented Programming Perspective in Software Agents and Simulation. International Journal of Advancements in Technology. https://doi.org/10.4172/0976-4860.1000186

Hocaoğlu, M. F. (2018). AdSiF: Agent driven simulation framework paradigm and ontological view. Science of Computer Programming, 167, 70–90. https://doi.org/10.1016/j.scico.2018.07.004

Johnstone, M., Le, V. T., Zhang, J., Gunn, B., Nahavandi, S., & Creighton, D. (2015). A dynamic time warped clustering technique for discrete event simulation-based system analysis. EXPERT SYSTEMS WITH APPLICATIONS, 42(21), 8078–8085. https://doi.org/10.1016/j.eswa.2015.06.040

Kelton, D. W., Sadowski, R., & Sturrock, D. T. (2006). Simulation with Arena. McGraw-hill Publisher.

Kemper, P., & Tepper, C. (2009). Automated trace analysis of discrete-event system models. IEEE Transactions on Software Engineering, 35(2), 195–208.

Meneses, E., & Kalé, L. V. (2015). CAMEL : Collective-aware Message Logging, (March), 2516–2538. https://doi.org/10.1007/s11227-015-1402-3

Murray-Smith, D. J. (1995). Continuous System Simulation. University og Glasgow, Glasgow, UK: Chapman & Hall.

Nomoto, H. (2004). State Oriented Programming. In Proceedings of the Eighth IEEE International Symposium on High Assurance Systems Engineering (HASE'04), 2004.

Sterkin, A. (2008). State-Oriented Programming. In 6th MPOOL Workshop, Cyprus.

Vasyutynskyy, V., Gellrich, A., Kabitzsch, K., & Wustmann, D. (2010). Analysis of internal logistic systems based on event logs. In IEEE conference on emerging technologies and factory automation (ETFA).