HLA Uyumlu Benzetim Sistemlerinin Ardıl-İşlem Çizenekleriyle Sınanması

Rukiye SÜTBAŞ[1] Turgay ÇELİK[2] Kayhan İMRE[3]

[1,2,3]Hacettepe Üniversitesi Bilgisayar Mühendisliği Bölümü, Beytepe, Ankara

[1]e-posta: rukiye@cs.hacettepe.edu.tr [2]e-posta: turgay@cs.hacettepe.edu.tr

[3]e-posta: ki@hacettepe.edu.tr

Özet

Yeniden kullanılabilir ve birlikte çalışabilir dağıtılmış benzetim sistemlerinin geliştirilmesinde Yüksek Düzeyli Mimari (HLA – *High Level Architecture*) yaygın olarak kullanılır. Federasyon geliştirme ve çalıştırma süreci (FEDEP – *Federation Development and Execution Process*) ile HLA uyumlu benzetim sistemleri geliştirilirken izlenmesi gereken adımlar tanımlanmıştır. Bu süreç tanımları karmaşık olduğundan, yardımcı araçların kullanılması gereklidir. Bu ihtiyacı karşılamak üzere, bütünleşik bir HLA araç topluluğu geliştirilmektedir. Ortaya çıkacak araçların oluşturacağı bütünleşik platformun, FEDEP'te tanımlı geliştirme ve sınama süreçlerini iyileştirmesi ve hızlandırması amaçlanmaktadır. Bu bildiri kapsamında, FEDEP sürecinin federasyonun işletilmesi ve sonuçlarının incelenmesi adımını kolaylaştırmak üzere geliştirilen sınama aracı incelenecektir.

Anahtar Kelimeler: Ardıl-İşlem Çizenekleri, *High Level Architecture* (HLA), *Management Object Model* (MOM), Kara Kutu Testi, *Unified Modeling Language* (UML)

Testing HLA Simulations With Sequence Diagrams

Abstract

High Level Architecture (HLA) is widely used for developing reusable and interoperable distributed simulation systems. Federation Development and Execution Process (FEDEP) has been defined for describing a generalized process for building HLA federations. The complexity of this process requires usage of automation tools to simplify development steps. To cover this requirement, an HLA toolkit has been under development. It is expected that this toolkit will improve and accelerate development and testing phases of FEDEP. In this paper, testing tool developed for simplifying last step of FEDEP, execute federation and prepare results, will be outlined.

Keywords: Black Box Testing, High Level Architecture (HLA), Management Object Model (MOM), Sequence Diagram, Unified Modeling Language (UML)

1. Giriş

Yazılım geliştirme maliyetli ve zaman alıcı bir süreçtir. Benzetim sistemlerinin çoğu zaman dağıtılmış ortamlarda çalışması gerekir ve klasik uygulamalara göre daha karmaşık yazılımlardır. Bu sebeple, geliştirilen benzetim sistemlerinin kolayca yeniden kullanılabilmesi, geliştirme maliyetini önemli ölçüde düşürecektir. Aynı zamanda benzetim bileşenlerinin diğer benzetim bileşenleriyle birlikte çalışabilir olması, geliştirme sürecinin etkinliğini artıracaktır. Benzetim bileşenlerinin yeniden kullanılabilmesi ve birlikte çalışabilmesi için, önceden tanımlanmış kurallara uymaları gerekir. Bu ihtiyaçları gidermek üzere, Amerikan Savunma Bakanlığına bağlı olan DMSO (*Defense Modeling and Simulation Office*), HLA belirtimini geliştirmiştir [1]. HLA, temelde savunma sanayiine yönelik olarak geliştirilmesine rağmen, günümüzde sürücü eğitim sistemleri, çok kullanıcılı oyunlar gibi sivil sektördeki uygulamalarda da yaygın olarak kullanılmaktadır.

HLA'da benzetim sistemi federasyon, benzetim bileşenleri de federe olarak adlandırılır. Federeler arası etkileşim, yayımlama/üye olma (*publish/subscribe*) mekanizmasına dayanır. HLA'da yayımlama/üye olma mekanizması, koşum zamanı altyapısı (RTI – *Runtime Infrastructure*) adı verilen bir altyapı aracılığıyla sağlanır.

HLA uyumlu büyük ve dağıtılmış benzetim sistemlerinin geliştirilme süreci karmaşıktır. HLA'nın dinamik veri dağıtım mantığı bu süreci daha da karmaşıklaştırır [6]. Bu nedenle süreç adımlarını tanımlayan bir modele ihtiyaç duyulur ve DMSO, bu ihtiyacı karşılamak için, Federasyon Geliştirme ve Çalıştırma Sürecini (FEDEP) tanımlamıştır. FEDEP, benzetim sisteminin amaçlarının ve yeteneklerinin belirlenmesi, kavramsal modelinin oluşturulması, tasarlanması, geliştirilmesi, alt birimlerinin birleştirilmesi ve sınanması, federasyonun işletilmesi ve sonuçlarının incelenmesi adımlarından oluşur [3].

Bu bildiride, FEDEP sürecinin "federasyonun işletilmesi ve sonuçlarının incelenmesi" adımı ele alınacaktır. HLA uyumlu sistemler, klasik uygulamalarda olduğu gibi kesme noktaları (*breakpoint*) koyarak işletimi durdurup niteliklerin değerlerini incelemeye izin vermez, çünkü bileşenler dağıtılmış ortamda koşut olarak çalışırlar. Yayımlama/üye olma mekanizmasına dayanan sistemlerin sınanmasında yaygın olarak kullanılan bir yöntem, çalışma sonrası gözden geçirmedir (*After Action Review - AAR)*. HLA belirtimi de yayımlama/üye olma mekanizmasını kullandığından, HLA uyumlu sistemler AAR yöntemi kullanılarak sınanabilir. Bu yöntemde sistem çalışırken niteliklerin değer değişimi, bileşenlerin etkileşimleri ile ilgili bilgiler toplanır. HLA sistemlerinden çalışma zamanında veri toplamak için, yönetim nesne modeli (*Management Object Model - MOM*) geliştirilmiştir [4]. Çalışma sonrasında bu veriler çeşitli yöntemlerle çözümlenerek beklenmeyen davranışlar tespit edilir ve sistemin performansı hakkında bilgi elde edilir.

İzleyen kesimlerde, öncelikle, yönetim nesne modelinden bahsedilecek, daha sonra AAR yönteminin HLA sistemlerine uygulanmasıyla ilgili önceden yapılmış çalışmalar incelenecek, son kesimde geliştirilen araçtan ve sağladığı avantajlardan bahsedilecektir.

2. Yönetim Nesne Modeli (*Management Object Model–MOM*)

HLA yeni tanımlandığı dönemlerde, federelerin RTI (Koşum Zamanı Altyapısı) hakkında yönetim verilerine erişmesi ve bazı durumlarda federasyonu denetlemesi gerekliliği ortaya çıkmıştır. RTI, federasyon ve federeler hakkında hangi bilgilerin gerekli olabileceği konusunda yapılan çalışmalar

sonrasında, MOM ortaya çıkmıştır. Bir federe benzetim verilerine üye olabildiği gibi, yönetim verilerine de üye olabilir. Ayrıca bu verileri başka bir federe değil, doğrudan RTI sağlar. MOM sayesinde bir federe yönetim verilerini edinmesi yanında, birtakım etkileşimlerle federasyonu denetleyebilir. Bu etkileşimler de doğrudan RTI tarafından algılanıp tepki verilir.

MOM bileşenleri, nesne ve etkileşim sınıflarıdır, diğer nesne ve etkileşim sınıfları gibi nesne modelinin elemanlarıdır ve etkin olmaları için nesne modeli dosyalarına (FOM/SOM, FED) eklenmelidirler [2]. Bu sınıflar federasyona katılan federelerin özellikleri hakkında bilgi sahibi olmak için kullanılır. Örneğin bir federe MOM nesne sınıflarından biri olan *Manager.Federate* sınıfına üye olursa, federasyona katılan her federe için bu sınıfın bir nesnesi, RTI tarafından üye olan federeye bildirilecektir. MOM bileşenlerinin çoğu, federelerin yerine yerel RTI bileşenleri (LRC – *Local RTI Component*) tarafından yayımlanır/üye olunur.

HLA sistemlerinde verilerin toplanması, bu işle görevlendirilmiş özel federeler tarafından gerçekleştirilir. Bu izleyici federeler MOM bileşenlerini kullanarak, federasyondaki diğer federelerin RTI ile arasındaki mesaj alışverişini kaydedebilir. Kaydedilen bu veriler, çalışma sonrası gözden geçirme araçlarına veri kaynağı olarak kullanılabilir.

3. İlgili Çalışmalar

Çalışma zamanında MOM bileşenlerini kullanarak veri toplayıp işleyen birtakım araçlar geliştirilmiştir. Bu kesimde FMT (*Federation Management Tool*), FVT (*Federation Verification Tool*), *MAK Data Logger* ve *hlaResults* araçları incelenecektir.

FMT, DMSO ile MAK firmasının ortak çalışmasıyla geliştirilmiştir [5]. FMT, federasyon çalışırken MOM bileşenlerini kullanarak çalışma hakkında bilgi toplar ve topladığı bilgileri kullanıcı arayüzleri aracılığıyla federasyon yöneticisine sunar. FMT'nin veriyi işlemesi çalışma anında olur, çalışma sonrasında veriyi çözümleme özelliği yoktur. FMT genişletilmeye olanak sağlayan bir eklenti arayüzüne sahiptir [7].

FVT, DMSO desteğiyle *Georgia Tech Research Institute* tarafından geliştirilmiştir [8]. FVT, federelerin, niteliklerin günlenmesi/yansıtılması (*update/reflect*), etkileşimlerin gönderilmesi/alınması ve HLA servislerinin çağrılması sorumluluklarını yerine getirdiklerini doğrular [9]. FVT, belli bir servis çağrısı dizisi sorgulamaya olanak sağlamamaktadır. Ayrıca kullanıcıya sunulan sonuç verileri, zaman bilgisi, ilgili servis çağrılarının hangi sırada oluştuğu bilgileri açısından yetersizdir ve sonuç raporu kullanıcıya federasyondaki beklenmeyen durumları kolayca tespit edecek bilgiyi sağlayamamaktadır.

MAK Data Logger, *MAK Technologies* tarafından geliştirilmiştir. HLA ve DIS (*Distributed Interactive Simulation*) uyumlu benzetim sistemlerinin çalışma zamanı verilerinin toplanması ve çalışma sonrası tekrar izlenmesine olanak sağlar [10]. Toplanan verilerin çözümlenmesi için, veri tabanına kaydedilen ham verileri işleyecek ek bir uygulamaya ihtiyaç duyulur.

hlaResults, *Virtc* firması tarafından geliştirilmiştir. *MAK Data Logger*'da olduğu gibi, HLA ve DIS (*Distributed Interactive Simulation*) uyumlu benzetim sistemlerinin çalışma zamanı verilerinin toplanması ve çalışma sonrası tekrar izlenmesine, ayrıca toplanan verilerin çözümlenmesine olanak sağlar [11]. Kullanıcı nesne sınıfı nitelik değerlerinin ve etkileşimlerin izlenmesi için test tanımları oluşturabilirken, ilgilendiği servis çağrıları düzeyinde test tanımları yapamaz.

4. MOM ve Ardıl-İşlem Çizenekleriyle Federasyonun Sınanması

Geliştirilen sınama aracı, kullanıcının kara kutu testi (*black box testing*) yapmasına olanak sağlayacaktır. İşlevsel sınama olarak da bilinen kara kutu testinde, kullanıcı sistemin nasıl çalıştığı konusunda fikir sahibi değildir. Sadece geçerli girdileri ve beklenen çıktıların ne olması gerektiğini bilir ve buna göre sınama tanımlarını oluşturur. Kullanıcının sınama tanımlarına göre, sistem sınanır ve sonuçlar kullanıcıya sunulur.

Bu kesimde, sınama aracı geliştirilirken izlenen yol incelenecektir. Federasyonun sınanması için ilk adım, federasyonun çalışması sırasında çalışma verilerinin toplanmasıdır. Bu amaçla geliştirilen MOM, önceki kesimlerde ele alınmıştır. Sınama aracı geliştirilirken, genel yaklaşıma uyarak veri toplama sürecinde MOM kullanılmış ve toplanan veriler, çözümleme safhasında kullanılacak biçimde kaydedilmiştir.

Sınama sürecinde ikinci aşama, sınama tanımlarının oluşturulmasıdır. Dağıtılmış sistemler söz konusu olunca, kullanılan sınama tanımı oluşturma yöntemlerinden biri ardıl-işlem çizenekleridir. DMSO'nun tanımladığı, HLA uyumluluk sınama belirtiminde, sınama tanımları, ileti ardıl-işlem çizgeleriyle (*Message Sequence Chart - MSC*) ifade edilir [13]. MSC'lerle sistem davranışı tanımlanabilir, dağıtılmış sistemdeki bileşenlerin birbirleriyle ve ara katman yazılımıyla olan etkileşimleri modellenebilir [14]. UML ardıl-işlem çizenekleri, işlevsel olarak MSC'lere benzerler. Geliştirilen sınama aracında, sınama tanımlarının oluşturulması için ardıl-işlem çizenekleri kullanılmıştır.

Sınama sürecindeki son adım, toplanan verilerin sınama tanımlarına göre çözümlenmesidir. Sınama aracı, bu süreçte, tüm başarılı eşleşmeleri, başlayan fakat sonlanamayan eşleşmeleri tespit ederek, etkileşim ayrıntılarını da içerecek şekilde çizelgeler ve grafikler aracılığıyla kullanıcıya sunar. Böylece kullanıcı beklenmeyen durumların tam olarak hangi noktalarda oluştuğunu tespit edebilir.

Geliştirilen sınama aracının önceki kesimde incelenen araçlara göre avantajları şu şekilde sıralanabilir:

- Kullanıcı aramak istediği servis çağrıları dizileriyle, sınama tanımları oluşturabilir.
- Sınama tanımları ardıl-işlem çizenekleriyle, görsel bir ortamda, ilgili servis çağrıları belirtilerek kolayca oluşturulabilir. Kullanıcı ilgili servis çağrısını, doğrudan yazmak yerine her bileşen için önceden özel olarak hazırlanmış sinyal listesinden seçtiği için hata yapma olasılığı azalır.
- Çözümleme sonuçları, kullanıcının beklenmeyen durumları kolayca tespit etmesine olanak sağlayacak şekilde çizelgeler ve grafiklerle sunulur.
- Geliştirilen sınama aracı, sonraki kesimlerde ele alınacak, HLA araç topluluğunun bir parçasıdır. Bu sayede kullanıcı aynı görsel aracı kullanarak modelleme ve sınama süreçlerini gerçekleştirebilir.

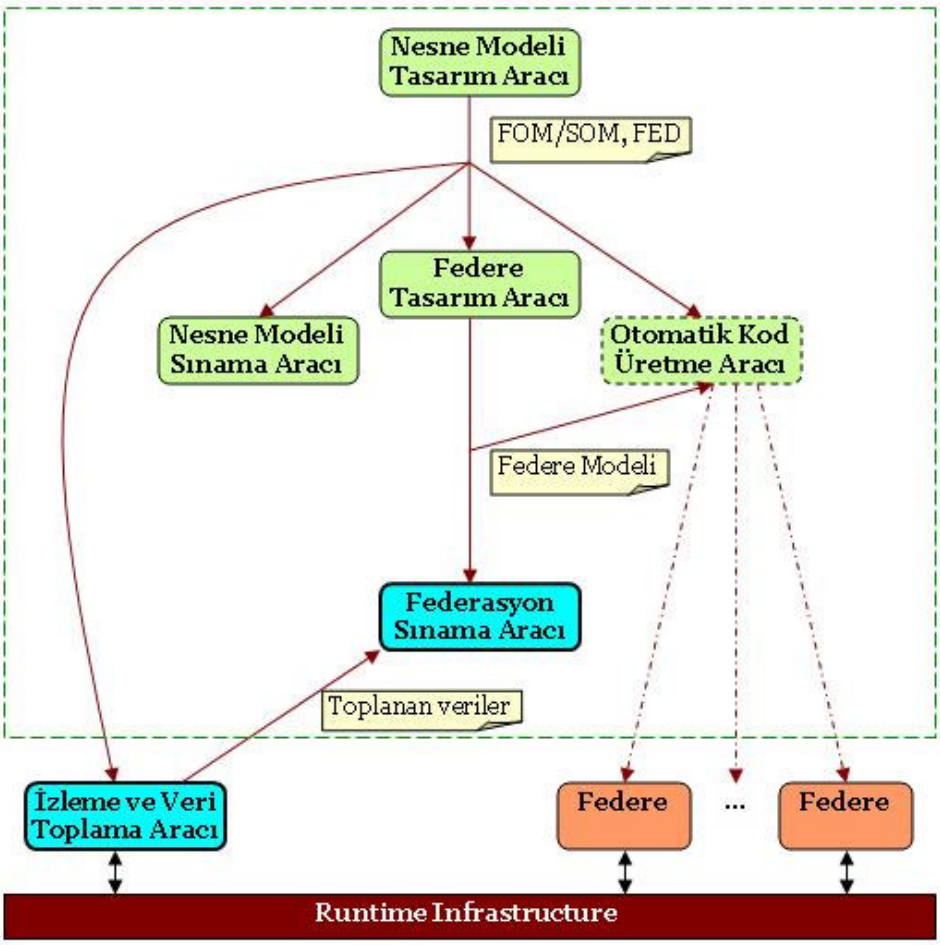
5. Sınama Aracı

Tez kapsamında geliştirilen sınama aracı federasyon çalışma verilerini toplayan bileşen, sınama tanımlarının oluşturulduğu bileşen ve sınama sonuçlarını gösteren bileşen olmak üzere, mantıksal olarak üç bileşenden oluşur.

5.1. Verilerin Toplanması

Bu bileşenin görevi federelerle, haberleşmeyi sağlayan altyapı (RTI) arasındaki servis çağrılarını parametreleri ve dönüş değerleriyle birlikte ve veri çözümleme aşamasında kullanılacak biçimde kaydetmektir. Servis çağrılarını yakalamak için bu amaçla geliştirilen araçlar kullanılabilir. Fakat yakalanan bilginin yorumlanıp özel bir biçimde kaydedilmesi gerektiğinden, seçilen aracın ayrıca eklenti arayüzü de sağlaması gereklidir. FMT, federasyona bir federe olarak katılıp, servis çağrılarını yakalayabilir ve sağladığı eklenti arayüzüne bu servis çağrılarını iletebilir.

Veri toplama bileşeni FMT'ye eklenti olarak geliştirilmiştir. FMT'nin yakaladığı servis çağrılarını yorumlayarak, veri çözümleme aşamasında kullanılacak biçimde ve servis çağrısının etkilediği federenin özel kütüğüne kaydeder. Şekil 1'de gösterilen araç topluluğunun bir parçası olan "izleme ve veri toplama aracı", FMT ve veri toplama bileşenine karşılık gelir.

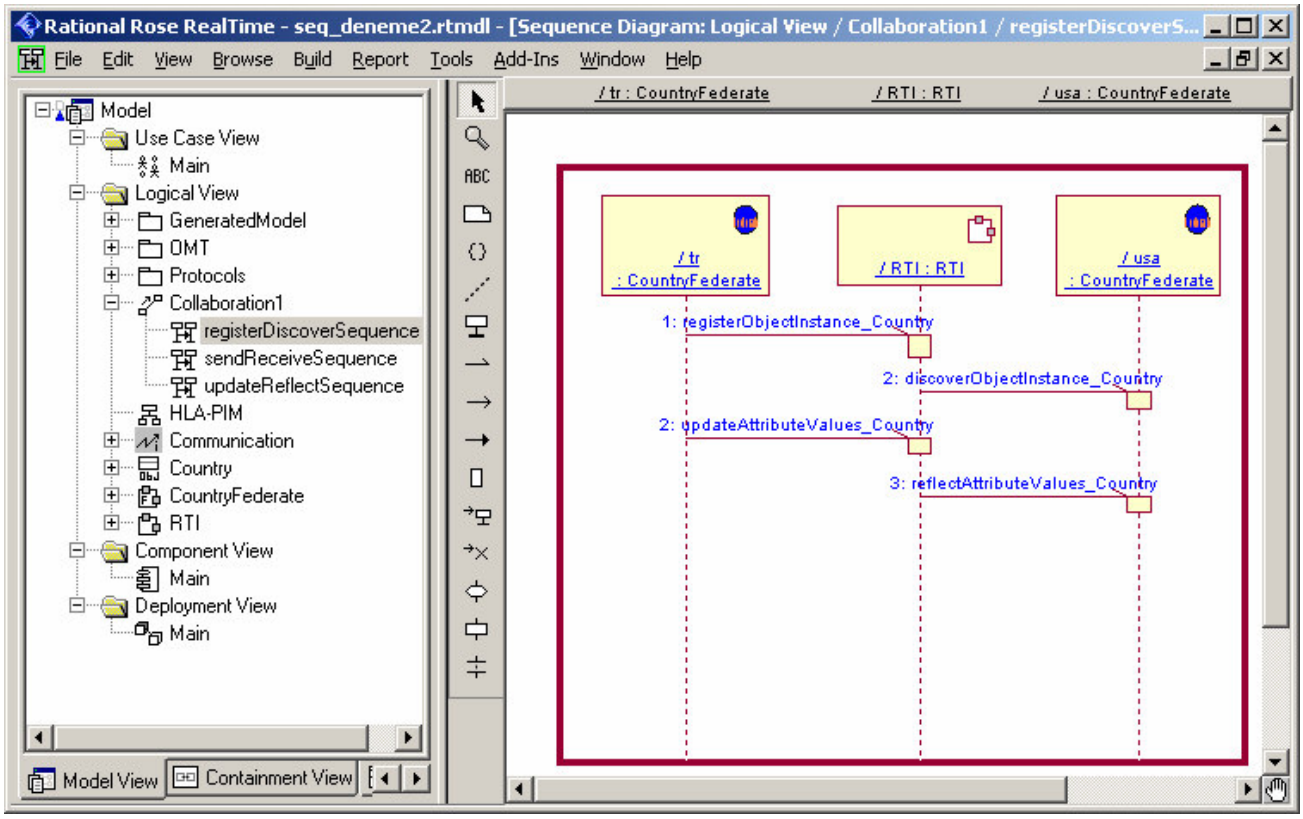


Şekil 1. HLA Araç Setinin Genel Görünümü

5.2. Sınama Tanımlarının Oluşturulması

Sınama tanımları ardıl-işlem çizenekleri kullanılarak oluşturulur. Ardıl-işlem çizeneği, federeler ve RTI arasındaki etkileşimi modelleyeceğinden, nesne olarak federeler ve RTI kullanılacaktır. Federeler

ve RTI arasındaki etkileşim servis çağrılarıyla gerçekleştirildiğinden, çizenekteki mesajlar servis çağrıları olacaktır. Bu özellikleri destekleyen bir sınama aracı, UML ile modellemeye olanak sağlamalı, ayrıca federeler ve servis çağrıları hakkında tanımlamalar yapmayı sağlayan bir modelleme aracıyla bütünleşik olmalıdır. Bu gereksinimler gözönüne alınarak sınama aracı [12]'de detaylı olarak anlatılan bütünleşik bir araç topluluğunun parçası olarak tasarlanmış ve geliştirilmiştir. Araçların genel görünümü Şekil 1'de gösterilmiştir. Araçlar *Rational Rose RealTime* [15] aracına eklenti olarak geliştirilmektedir. Geliştirme sürecinde *Rose RealTime Extensibility Interface* (RRTEI) [16] ve .NET kullanılmaktadır. Nesne modeli tasarım aracı, federe tasarım aracı, federasyon sınama aracının gerçekleştirimi tamamlanmıştır. Şekil 2'deki ardıl-işlem çizeneği, örnek bir sınama tanımı olarak verilmiştir. Örnekteki sınama tanımında, beklenen durum, *Country* nesnesinin bir federe (*tr*) tarafından oluşturulup, değerlerinin günlenmesi; diğer federenin (*usa*) de *Country* nesnesini keşfedip, günleme sonucunda oluşacak değer değişiminden haberdar olmasıdır.



Şekil 2. Sınama Tanımı Örneği

5.3. Toplanan Verilerin Çözümlenmesi

Sınama aracının veri toplama ve sınama tanımlarının oluşturulması kesimleri sonlandığında, toplanan veriler sınama tanımlarına göre çözümlenir. Çözümleme aşamasında ardıl-işlem çizeneğindeki sinyal kümesi, toplanan servis çağrıları kümesi içinde aranır. Elde edilen sonuçlar çizelgeler ve grafiklerle kullanıcıya sunulur. Sonuç çizelgeleri,

- gerçekleşen eşleşmeleri, başlama ve bitiş zamanlarıyla birlikte gösteren çizelge
- her bir başarılı eşleşme için, eşleşmelerin ayrıntılarını, yani hangi servis çağrılarından oluştuğunu ve servislerin ayrıntılarını, gösteren çizelgeler
- tamamlanamayan eşleşmeleri ve ayrıntılarını gösteren çizelgelerdir.

Sonuç grafikleri

- eşleşmelerin gerçek zamana göre grafiği
- federelerin gerçek zaman-mantıksal zaman grafiği
- her bir eşleşme için, servislerin gerçek zamana göre grafiğidir.

6. Sonuç

HLA uyumlu benzetim sistemlerinin sınanması zor ve karmaşık bir süreçtir. Bu süreçte çalışma zamanında federasyondan bilgi toplanmalı, toplanan veriler kullanıcının tanımladığı kurallara göre çözümlenmeli ve kullanıcıya anlamlı bir biçimde sunulmalıdır. Bu ihtiyacı karşılamak üzere bütünleşik bir HLA araç setinin bir parçası olarak sınama aracı geliştirilmiştir. Geliştirilen araç FMT aracına yazılan eklenti ile çalışma zamanı verilerini özel bir biçimde toplar. *Rational Rose RealTime* eklentisi olarak geliştirilen sınama bileşeni, ardıl-işlem çizeneğiyle oluşturulan sınama tanımlarını kullanarak toplanan verileri çözümler. Sonuçlar yine sınama aracı tarafından kullanıcının beklenmeyen durumları kolayca tespit etmesine olanak verecek şekilde, çizelgeler ve grafiklerle sunulur.

Geliştirilen sınama aracının da içinde bulunduğu araç setinin, HLA uyumlu benzetim sistemlerinin geliştirilme ve sınanma sürecini hızlandırması ve etkinleştirmesi beklenmektedir.

Kaynakça

[1] Defense Modeling and Simulation Office (DMSO), “High Level Architecture Interface Specification, Version 1.3”, (1998)

[2] Defense Modeling and Simulation Office (DMSO), “HLA Object Model Template, Version 1.3”, (1998)

[3] Defense Modeling and Simulation Office (DMSO), “HLA Federation Development and Execution Process (FEDEP) Model v1.5” , (1999)

[4] F.Kuhl, R.Weatherly, J.Dahmann, “Creating Computer Simulation Sytems-*An Introduction To The High Level Architecture* ”, (1999), Prentice Hall: s.192-199

[5] FMT, http://fmt.mak.com/

[6] Deborah Fullford, Darren Wetzel, “A Federation Management Tool: Using the Management Object Model (MOM) to Manage, Monitor, and Control an HLA Federation”, 99S-SIW

[7] Rolf H. Nelson, “A Next-Generation Federation Management Tool: Using the Management Object Model (MOM) and FOM-specific Data to Monitor an HLA Federation”, 2000F-SIW

[8] FVT, http://dss.gtri.gatech.edu/guide_fvt/index.html

[9] Margaret L. Loper, David Rosenbaum, “The Federation Verification Tool”, 98F -SIW

[10] MAK Data Logger, http://www.mak.com/s1ss5p0.php

[11] hlaResults, http://www.virtc.com/Products/prdFulltext.jsp?ID=c_Rslt

[12] Turgay Çelik, Rukiye Sütbaş, Kayhan İmre, “HLA için Modelleme, Otomatik Kod Üretme, İzleme ve Sınama Araçları”, (2005), USMOS’05 I. Ulusal Savunma Uygulamaları Modelleme ve Simulasyon Konferansı

[13] Margaret L. Loper, “Test Procedures For High Level Architecture Interface Specification”, (1998)

[14] Ekkart Rudolpha, Jens Grabowskib, Peter Graubmann, “Tutorial on Message Sequence Charts MSC”, (1996)

[15] Rational Rose Real Time, http://www-306.ibm.com/software/rational/

[16] Rational Rose Real Time, “Extensibility Interface Reference”, www.isy.vcu.edu/isy/rational/documentation/rational_rosert/rosert_extensibility.pdf