T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SİMÜLASYONLARIN ETKİLEŞİMLİ OLARAK ÇALIŞMASI ve HLA

YÜKSEK LİSANS TEZİ

Elektrik-Elektronik Müh. Ahmet ZENGİN

Enstitü Anabilim Dalı : ELEKTRİK-ELEKTRONİK MÜH.

Enstitü Bilim Dalı : ELEKTRONİK

TEMMUZ 1999

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SİMÜLASYONLARIN ETKİLEŞİMLİ OLARAK ÇALIŞMASI ve HLA

YÜKSEK LİSANS TEZİ

Elektrik-Elektronik Müh. Ahmet ZENGİN

Enstitü Anabilim Dalı : ELEKTRİK-ELEKTRONİK MÜH.

Enstitü Bilim Dalı : ELEKTRONİK

Bu tez 10 / 08 / 1999. tarihinde aşağıdaki jüri tarafından Oybirliği/Oyçokluğu ile kabul edilmiştir.

Doç. Dr. Hüseyin EKİZ	Doç. Dr. Ercan ÖZTEMEL	Yrd. Doç. Dr. Ayhan Özdemir
Jüri Başkanı	Jüri Üyesi	Jüri Üyesi

ÖNSÖZ

Simülasyonların gerek eğitim ve geliştirme alanlarında ve gerekse askeri alanlarda çok büyük fayda ve kolaylıklar sağladığı bilinen bir gerçektir. Gerçek hayattaki uygulamaların masraflı oluşu, yapılacak hataların telafisinin olmaması gibi bir takım dezavantajlar yazılım geliştiricilerini simulasyon tasarlama ve geliştirme alanına yöneltmiştir. Bugün hemen hemen her amaca yönelik simülasyonların mevcut olması ve bazen bu mevcut simulasyonların ihtiyaca cevap veremez durumuna düşmesi simülasyon geliştiricilerini başka alanlara yönlendirmiştir. Bu alanlardan biri simülasyonların karşılıklı bir şekilde çalışabilmesi ve simülasyon parçalarının(veya bileşenlerinin) tekrar kullanımıdır. Bu tezde anlatılan High Level Architecture(HLA) konusu bunu amaçlamaktadır.

HLA, Amerika Birleşik Devletleri içerisinde yaygın simülasyon sistemlerinin standart geliştirme ve onaylama işlemi olarak DMSO tarafından geliştirilmiştir. Bu çalışmada, askeri simülasyon sistemleri için son derece büyük öneme haiz olan HLA yapısının genel kavramları üzerinde durulmuştur. RTI bu yapının en önemli unsurlarından biridir. Bu nedenle RTI konusu daha detaylı olarak incelenmiştir.

Bu tezin hazırlanmasında çalışmalarıma ışık tutan Doç. Dr. Hüseyin EKİZ Bey'e, çalışmalarımda yardımlarını esirgemeyen ve geniş imkanlar sunan Doç. Dr. Ercan ÖZTEMEL Bey'e ve doküman ve kaynak temininde büyük yardımları olan Sayın Cüneyt FIRAT Bey'e sonsuz şükranlarımı sunarım.

Yanıkköy, 20 Temmuz 1999 Ahmet ZENGİN

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

HLA	: High Level Architecture
RTI	: Runtime Infrastructure
DMSO	: Defence Modelling and Simulation Office
DoD	: Department of Defence
M&S	: Modeling and Simulation
VV&A	: Verification, Validation, and Accreditation
CTF	: Common Technical Framework
CMMS	: Conceptual Models of the Mission Space
DS	: Data Standardization
DIF	: Data Interchange Format
MSRR	: Modeling and Simulation Resource Repository
CSS	: Common Semantics and Syntax
ADS	: Authoritative Data Sources
DQ	: Data Quality
DS	: Data Security
FOM	: Federation Object Model
SOM	: Simulation Object Model
FED	: Federation Execution Data
FEDEP	: Federation Execution and Development Process
DDM	: Data Distribution Management
DIS	: Distributed Interactive Simulation
MOM	: Management Object Model
OMT	: Object Model Templates
URL	: Uniform Resource Locator
OO	: Object Oriented
POC	: Point of Contact

OOAD	: Object-Oriented Analysis And Design
DARPA	: Defence Advanced Research Projects Agency
AI	: Artificial Intelligence
CORBA	: Common Object Request Broker
API	: Application Programmer's Interface
RID	: RTI Initialization Data
LRC	: Local RTI Component

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Amerikan Savunma Bakanlığı (The Department of Defence - DoD), Savunma Modelleme ve Simülasyon Ofisinin önderliği(DMSO) altında, DoD kapsamındaki modelleme ve simulasyon konularıyla ilgili bir yüksek derece yapı geliştirmektedir. Bu yapının merkezi bileşenlerinden biri RTI'dır(Runtime Infrastructure).

Bu çalışma, HLA ve RTI konusuna bir giriş sağlar. HLA'nın temelinde yatan konular ve HLA'nın üç bileşeni geniş bir şekilde açıklanmıştır. Daha sonra, bu çalışmada, RTI tarafından simulasyon uygulamalarına sağlanmış olan çeşitli hizmet sınıfları örnekler verilerek açıklanmıştır. Ayrıca, DIS ile RTI'ın kısa bir karşılaştırması da şekillerle anlatılmaya çalışılmıştır.

Yüksek derece yapı(HLA), DoD kapsamında modelleme ve simulasyon konuları için bir yapı sağlar. Bu yapının amacı, DoD bünyesinde simulasyonların karşılıklı olarak uyumlu bir şekilde çalışabilmesini ve simulasyon parça veya bileşenlerinin tekrar kullanımını sağlamak ve geliştirmektir. HLA, üç temel kavram üzerine kurulmuştur. Bunlar: Nesne Model Kalıpları, RTI ve HLA uyum kurallarıdır.

RTI ve uyum kuralları, bütün HLA-uyumlu simülasyonlar karşısında değişmemişlerdir. Ancak, birbirini etkileşim halindeki simulasyonların her grubu, veya federasyon, simulasyonlar arasındaki veri ve olayların karşılıklı transferi için temel bir usul veya bir kaide tanımlamalıdırlar. Bu temel kaidenin biçimi ve içeriği, Nesne Model Kalıpları tarafından tanımlanmıştır. Federasyon Nesne Modelinin(FOM) tanımı, HLA uyumlu bir federasyonun oluşma sürecinin bir parçasıdır.

RTI, en basit şekliyle, simülasyon sistemleri için gerekli olan hizmetler sağlayan bir bilgisayar yazılımıdır. RTI'ın işlevleri, HLA Arabirim Tanımlamaları(HLA Interface Specification) kısmında ayrıntılarıyla açıklanmıştır.

Bu tezin üniversite ve simülasyon geliştirme alanlarında yararlı olması ve hizmet vermesi dileğiyle.

SUMMARY

INTEROPERABLE SIMULATIONS AND HLA

Keywords: HLA, RTI, Interoperability, DoD.

The Department of Defense (DoD), under the leadership of the Defense Modeling and Simulation Office (DMSO), is developing a high level architecture for modeling and simulation within the DoD. One of the central components of this architecture is the Runtime Infrastructure (RTI).

This study provides an introduction to the RTI. After a brief introduction to the High Level Architecture (HLA) and the three components that comprise the HLA, this study describes the various classes of services provided by the RTI to simulation applications. A few examples are given comparing the RTI with DIS.

The High Level Architecture provides an architecture for modeling and simulation within the Department of Defense (DoD). The intent of this architecture is to foster the interoperation of simulations and the re-use of simulation components within the DoD. The HLA is defined by three concepts: The Object Model Templates, RTI, HLA Compliance Rules.

The RTI and compliance rules are unchanged across all HLA-compliant simulations. However, each group of interacting simulations, or federation, must define a basis for the exchange of data and events between simulations. The format and content of this basis is defined by the Object Model Templates. The definition of the Federation Object Model (FOM) is one part of the process of creating an HLA compliant federation.

The RTI is a collection of software that provides commonly required services to simulation systems. However, this goal is occasionally moderated when it is clear that an additional service could be used across multiple simulation domains.

The RTI is also intended to provide a measure of portability (across computing platforms, operating systems, and communication systems) and simulation interoperability. The services of the RTI are described by the HLA Interface Specification.

BÖLÜM 1. GİRİŞ

Bilindiği üzere simülasyonlar gerek eğitim ve geliştirme alanlarında ve gerekse askeri alanlarda çok büyük fayda ve kolaylıklar sağlamaktadır. Gerçek hayattaki uygulamaların masraflı oluşu, yapılacak hataların telafisinin olmaması gibi bir takım dezavantajlar yazılım geliştiricilerini simulasyon tasarlama ve geliştirme alanına yöneltmiştir. Bugün hemen hemen her amaca yönelik simülasyonların mevcut olması ve bazen bu mevcut simulasyonların ihtiyaca cevap veremez durumuna düşmesi simülasyon geliştiricilerini başka alanlara yönlendirmiştir. Bu alanlardan biri simülasyonların karşılıklı bir şekilde çalışabilmesi ve simülasyon parçalarının(veya bileşenlerinin) tekrar kullanımıdır. Bu tezde anlatılan High Level Architecture(HLA) konusu bunu amaçlamaktadır.

Sadece bir tek monolitik simülasyonun kullanıcıların bütün isteklerini karşılayamaması ve simülasyonlardan tam bir şekilde yararlanma amacıyla onların bir araya getirilme zorunlulukları gibi bir takım gerekçeler HLA'nın temel dayanak noktalarıdır. Bu gelecekteki teknolojik yeniliklerin ve çalışma biçimi türlerinin birbirine uyumlu olması fikriyle bağdaşmaktadır.

Nesnelerin ve gerçek olayların bir bilgisayar sistemi üzerindeki simulasyonu, eğitim alanda en makul olan ve en etkili bir yoldur. Gerçek dünyadaki nesnelerin modelleri matematiksel bir formda ifade edilir ve bu modeller görsel dünyada ifade edebilmek için mümkün olabildiğince gerçek dünyaya benzetilmeye çalışılan bilgisayar sisteminde kullanılır.[1]

Askeri eğitim ve ölçüm sistemlerinde, bu düşünceden kapsamlı ve etkili bir şekilde faydalanılır. Askeri simülasyonların başlıca üç amacı Prova(Rehearsal), Eğitim(Training) ve Kazanmadır(Acquisition). Görevleri yerine getirme konusunda askerlerin yetenekleri, prova simülasyonları yardımıyla saptanabilir. Diğer taraftan,

farklı askeri ekipmanları ve silah sistemlerinin kullanımı konusundaki deneyimlerin kazanılması veya artırılması, eğitim simülasyonları yardımıyla gerçekleştirilir. Simülasyonlar, yeni veya değişen sistemlerin kullanımlarını sağlamada bireylere yardımcı olur. Bu yüzden, genel bir kavram olarak simülasyon, askeri alanların önemli bir parçasıdır.

HLA, Amerika Birleşik Devletleri içerisinde yaygın simülasyon sistemlerinin standart geliştirme ve onaylama işlemi olarak DMSO tarafından geliştirilmiştir. Bu arada, diğer ülkeler ve tabi ki Türkiye, HLA standartlarına uymak için kendi sistemlerinde değişim yoluna gittiler. Bu, HLA'nın öneminin bir göstergesi ve bir kanıtıdır.

Bu çalışmada, askeri simülasyon sistemleri için son derece büyük öneme haiz olan HLA yapısının genel kavramları üzerinde durulmuştur. RTI bu yapının en önemli unsurlarından biridir[1]. Bu nedenle RTI konusu daha detaylı olarak incelenmiştir.

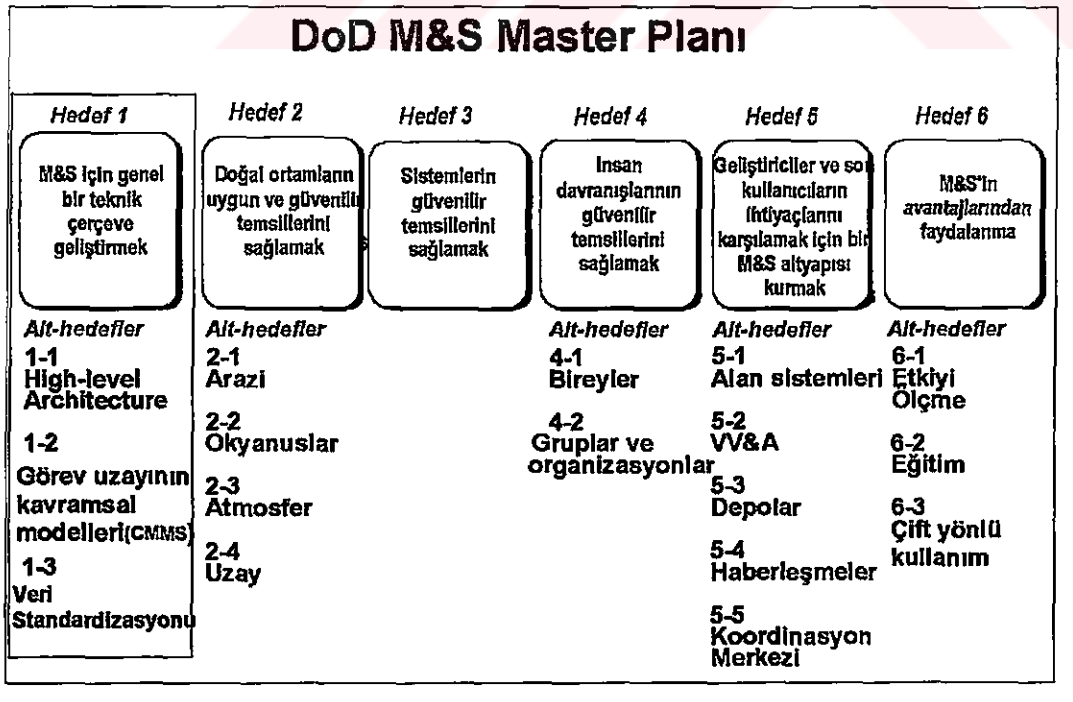
BÖLÜM 2. GENEL HLA TANIMLAMALARI VE RTI

2.1. Giriş

HLA(High Level Architecture), Amerikan Savunma Bakanlığı(Department of Defence-DoD) tarafından benimsenmiş olan DoD Modelleme ve Simulasyon Master Planı'nın(DoD Modeling and Simulation Master Plan-DoD 5000.59-P) 1-1 nolu hedefine uygun olarak, Ekim 1995'te geliştirilmiştir.[2]

2.2. DoD M&S Master Planı

Şekil 2.1'de görüldüğü gibi, DoD Modelleme ve Simülasyon Master Planı, Modelleme ve Simülasyon(M&S) sahalarıyla ilgili 6 adet hedefi kapsar. Bu planın 1. Hedefi, M&S konuları hakkında genel bir teknik çerçeve veya herkese açık teknik bir iskelet geliştirmektir. [3]



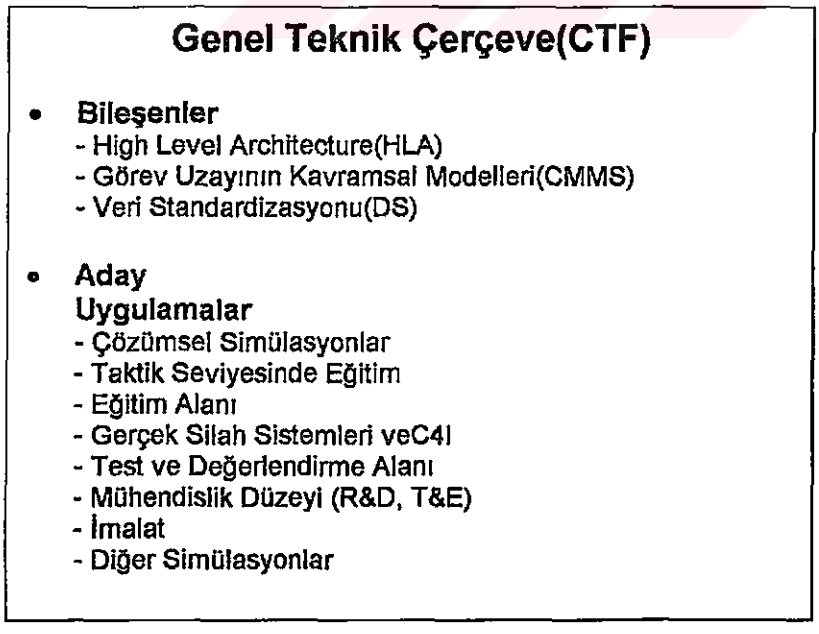
Şekil 2.1. DoD M&S Master Planı

Nasıl bir iskelet sisteminin bütün bileşenleri birbirleriyle bağlantılıysa veya iskelet sistemindeki bir parça diğer bir parçanın fonksiyonunu tamamlama gibi bir özelliğe sahipse, burada geçen iskelet veya çerçeveden kasıtta odur. Yani burada bahsi geçen iskelet yapısının bütün bileşenleri birbirini tamamlayıcı bir özelliğe sahiptir. Şimdi genel teknik çerçevenin amacını ve bileşenlerini kısaca irdeleyelim.

2.3. Genel Teknik Çerçeve(The Common Technical Framework-CTF)

Genel teknik çerçevenin(The common technical framework-CTF) amacı, fiyat olarak en uygun, birbirleriyle uygun bir şekilde çalışabilir(interoperable) ve yeniden kullanılabilir(reuse) simülasyonların ve modellerin gelişmesine yardımcı olmaktır.[3] Genel teknik çerçeve, veya daha açık bir şekilde belirtmek gerekirse HLA, Dod Modelleme ve Simülasyon topluluğunda en yüksek çabanın sarf edildiği konudur.

Genel teknik çerçevenin üç alt hedefi veya bileşeni vardır. Bunlar (1) Yüksek Seviyeli Yapı(High Level Architecture-HLA), (2) Görev Uzayının Kavramsal Modelleri(Conceptual Models of the Mission Space-CMMS), ve (3) Veri Standardizasyonudur(Data Standardization-DS). Şekil 2.2'de genel teknik çerçevenin bileşenlerinin ve aday uygulamaların bir taslağı görülmektedir.



Şekil 2.2. Genel Teknik Çerçeve

2.3.1. Görev Uzayının Kavramsal Modeli(Conceptual Model of Mission Space-CMMS)

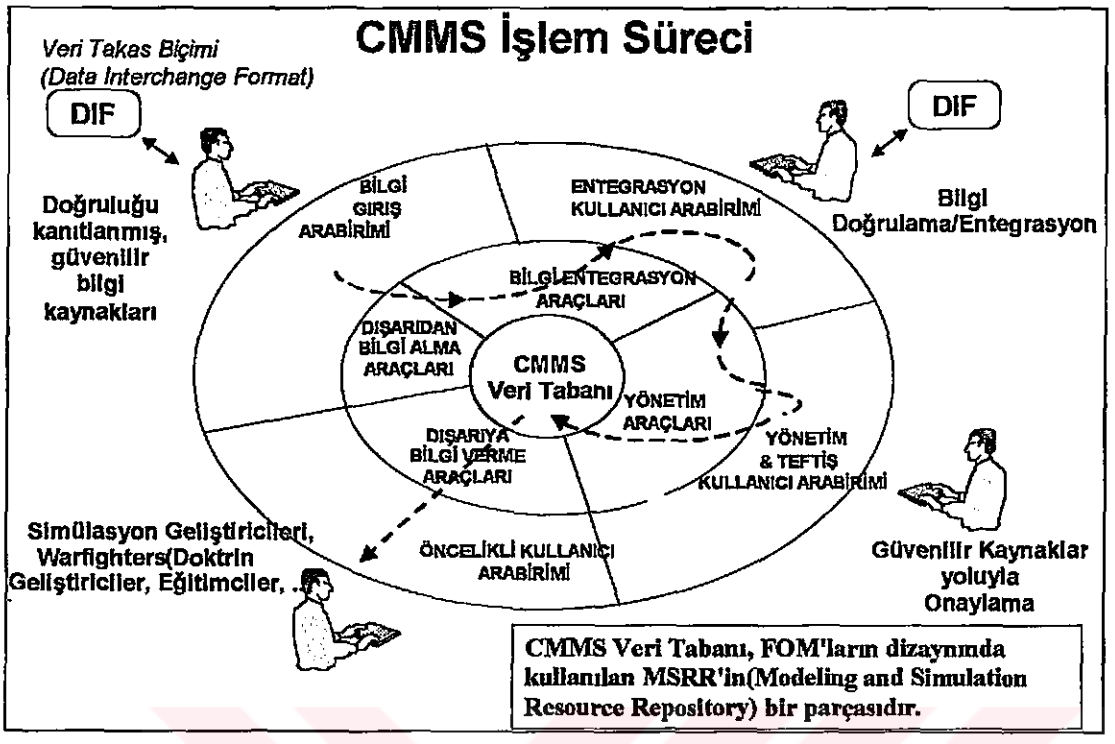
Görev Uzayının Kavramsal Modeli(CMMS), gerçek dünyanın en düşük düzeyde soyutlanması işlemidir. CMMS, belirli bir görev ve varlık/organizasyon tarafından düzenlenen uygun aksiyon ve etkileşimlerin onaylanması yoluyla bilgi elde etmede genel bir çerçeve vazifesi görür. CMMS, özel bir görev alanıyla ilişkisi olan değişik varlıklar arasındaki aksiyonların hiyerarşik bir yapıda tanımlanmasıdır(Şekil 2.3).

Görev Uzayının Kavramsal Modeli (CMMS)

- **İfade biçimini standart bir şekle getirir ve bilgi edinme konusunda ortak bir anlayış tesis eder.**
- **Belirli bir görev veya organizasyon doğrultusunda düzenlenen uygun aksiyon ve etkileşimlerin onaylanmasını organize eder.**
- **Ortak bir gerçek dünya tasvirini simülasyon kullanıcılarına sağlar.**
- **Belirli bir görev alanıyla ilişkili olan değişik varlıklar arasında etkileşim ve aksiyonları düzenler.**

Şekil 2.3. Görev Uzayının Kavramsal Modelinin işlevleri

Görev uzayının kavramsal modeli, tutarlı ve güvenilir M&S temsillerinin inşa edilmesi konusunda simülasyon kullanıcılarına temel bir çözüm yolu sağlar. CMMS'in başlıca hedefi, özellikle DoD simülasyon geliştirme çalışmaları dahilinde, DoD simülasyonları arasındaki güvenilir bilginin paylaştırılması yoluyla, simülasyon parçaları ve bileşenlerinin yeniden kullanımını ve simülasyonların karşılıklı çalışabilmesini kolaylaştırmaktır. CMMS, askeri operasyonlar konusunda gerekli olan temel bilginin bir meta-modelini sağlayacaktır. CMMS sistemi, bu bilgiyi ele geçirip depolayacak, ve simülasyon geliştiricileri ve kullanıcıları için kolayca elde edilebilir hale getirecektir. Şekil 2.4'te CMMS işlem süreci göstermektedir.



Şekil 2.4. CMMS işlem süreci

Görev uzayı yapısı, araçlar ve kaynaklar; DoD simülasyon sistemindeki çevrenin, sistemlerin ve insan davranışlarının güvenilir, birbiriyle uyumlu ve tutarlı tasvirlerinin gelişimine imkan vermesi için gerekli veri ve detaylara bir erişim sağlayacaktır.[3]

2.3.2. Veri Standardizasyonu (Data Standardization-DS)

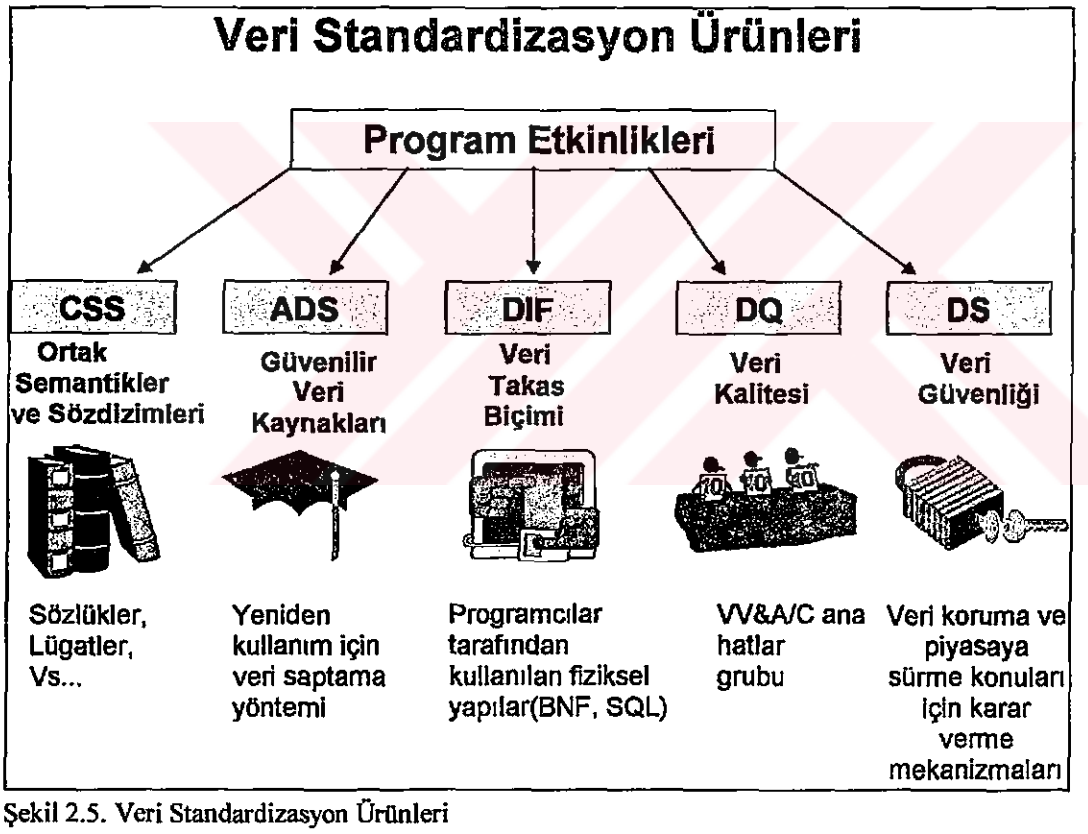
Veri Standardizasyonu programı, veri gereksinimleri, standartlar, kaynaklar, güvenlik ve doğrulama, onaylama ve belgeleme için karar mekanizmaları, prosedürler ve metodolojiler tesis ederek, modeller, simülasyonlar ve C4I sistemleri arasında yeniden kullanımı, uyumlu bir şekilde karşılıklı çalışabilmeyi ve veri paylaşımını kolaylaştırmaya uğraşır.

Veri standardizasyon programının temel ürünleri:

(1) Ortak Semantik ve Sözdizimi(Common Semantics and Syntax-CSS): Veri parçaları için ortak sözlükler, lügatler, taksonomiler(canlıların sınıflandırılması) ve araçlar tanımlar.

(2) Veri Takas Biçimleri(Data Interchange Formats-DIF): Veri takas işlemi için programcılar tarafından kullanılan fiziksel yapılardır(BNF, SQL).

Diğer veri standardizasyon destek ürünleri; (1) Güvenilir Veri Kaynakları(Authoritative Data Sources-ADS), yeniden kullanım konusunda verinin saptanması için temel yöntemlerdir, (2) Veri Kalitesi(Data Quality-DQ), bir VV&A/C ana hatlar grubu vazifesi görür, ve (3) Veri Güvenliği(Data Security-DS), veriyi koruma ve piyasaya sürme işlemi ile ilgili politikalar üretir(Şekil 2.5).



Şekil 2.5. Veri Standardizasyon Ürünleri

2.3.3. Yüksek Seviyeli Yapı(High Level Architecture-HLA)

HLA'nın amacı, kendisi ile C4I sistemleri arasındaki bütün tipte ve modeldeki simülasyonların karşılıklı olarak uyumlu bir şekilde çalışmasını(interoperability) kolaylaştırmak için genel bir yüksek seviye simulasyon yapısını kurmaktır [4]. HLA,

M&S bileşenlerinin yeniden kullanımını kolaylaştırması ve M&S topluluğu içerisinde bir standartlaşma oluşturması doğrultusunda dizayn edilmiştir(Şekil 2.6).

DoD'a bağlı özel birimler ve Savunma Bakanlığının fonksiyonel alanları için geliştirilen simülasyonların, Yüksek Seviyeli Yapıya(High Level Architecture) uyması gerekmektedir.

Şekil 2.6. HLA Tanımı

HLA, DoD modelleme ve simulasyon konusuyla alakalı uygulama alanlarını baştan sona içine alacak şekilde, geniş bir uygulanabilirlik ölçüsüne sahip olarak tasarlanmıştır. Bunlar, başta eğitim, analiz ve mühendislik konularını kapsar.

Bu çalışmalarda kullanılan yapının tanımı, çoğunlukla kabul edildiği şekilde birdir(Şekil 2.6). Bu konu üzerinde yapılan tüm çalışmaların amacı yüksek seviyeli yapının gelişimine katkıda bulunmaktır.

DoD'a bağlı Savunma Modelleme ve Simulasyon Ofisinin(Defence Modeling and Simulation Office-DMSO) önderliği altında geliştirilmiş olan bu yapının merkezi bileşenlerinden biri RTI'dır(Runtime Infrastructure). Ancak, HLA, belirli bir uygulamayla, belirli bir yazılım veya programlama dilinin kullanımıyla sınırlı değildir. Zamanla, teknolojinin gelişmeye müsait olması nedeniyle, yeni ve farklı

uyarlamalar veya uygulama programları, HLA'nın genel yapısı içinde yerini alacaktır. [4]

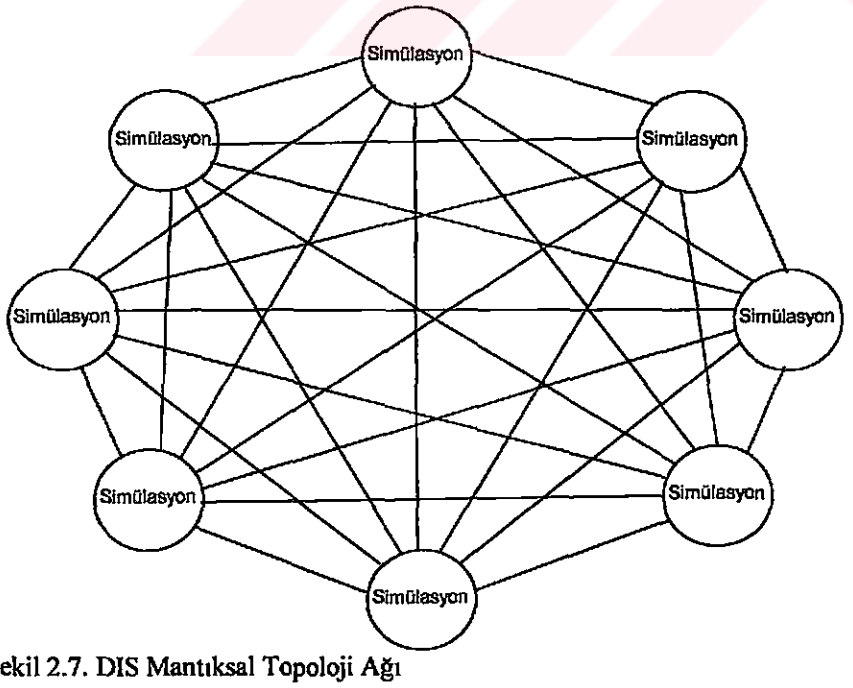
Daha öncede belirtildiği gibi, HLA, DIS gibi daha önceki standartların yerini almıştır. Şimdi DIS standardı ile HLA yapısı arasında bir karşılaştırma yapalım:

2.4. DIS & HLA Karşılaştırması

DIS standardının yapısal özellikleri aşağıda verilmiştir.

1. Bütün simülasyonları merkezi bir bilgisayar kontrol etmez.
2. Özerk simülasyon uygulamaları, bir veya daha fazla simülasyon varlığının durumunu sürdürmeden sorumlu olur.
3. Standart bir protokol, veriyi doğru yere iletmek için kullanılır.
4. Bir varlığın durumundaki değişiklikler, simülasyon uygulamaları tarafından iletilir.
5. Diğer varlıkların ve olayların algılanması, alıcı uygulamalar tarafından saptanır.
6. Hesap algoritmaları, mesajları işleme sokmak için kullanılır.

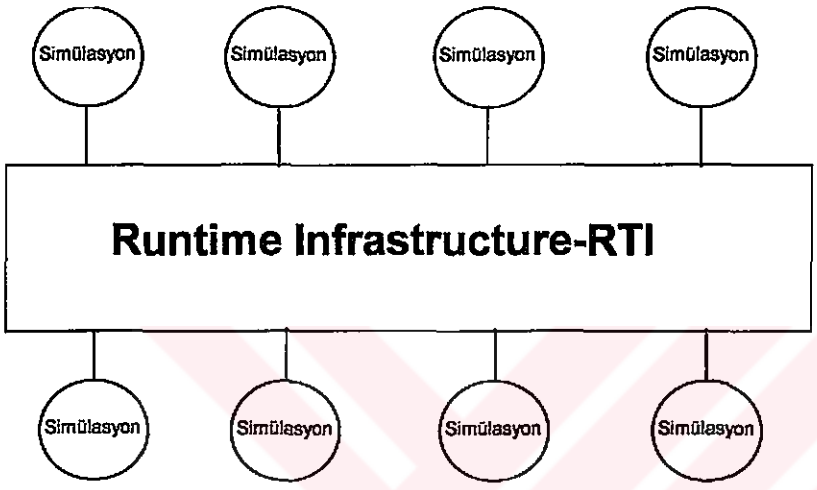
Aşağıdaki grafiksel gösterim, DIS mantıksal topoloji ağını göstermektedir.[5]



Şekil 2.7. DIS Mantıksal Topoloji Ağı

Burada gösterilen diyagramda, birbirlerine mantıksal bir bağlantı(çoğu kez tam bir ağ topolojisi olarak ta adlandırılır) ile bağlı sekiz adet simülasyon görülmektedir. Bu modelde, her simülasyon, arada bir tercümana gerek olmaksızın, diğer bir simülasyon ile doğrudan konuşur.

Eğer bu sekiz adet simülasyon HLA yapısı kullanılarak birleştirilirse, Şekil 2.8'deki yapı oluşur.



Şekil 2.8. HLA Yapısı

Bu yöntemde, simülasyonlar birbirleriyle direk bir bağlantıya sahip değildir. Bu simülasyonlardan her biri, bir diğeri ile doğrudan hiçbir zaman konuşamaz. Arada bulunan RTI vasıtasıyla simülasyonlar diğer simülasyonlarla haberleşebilir.[5]

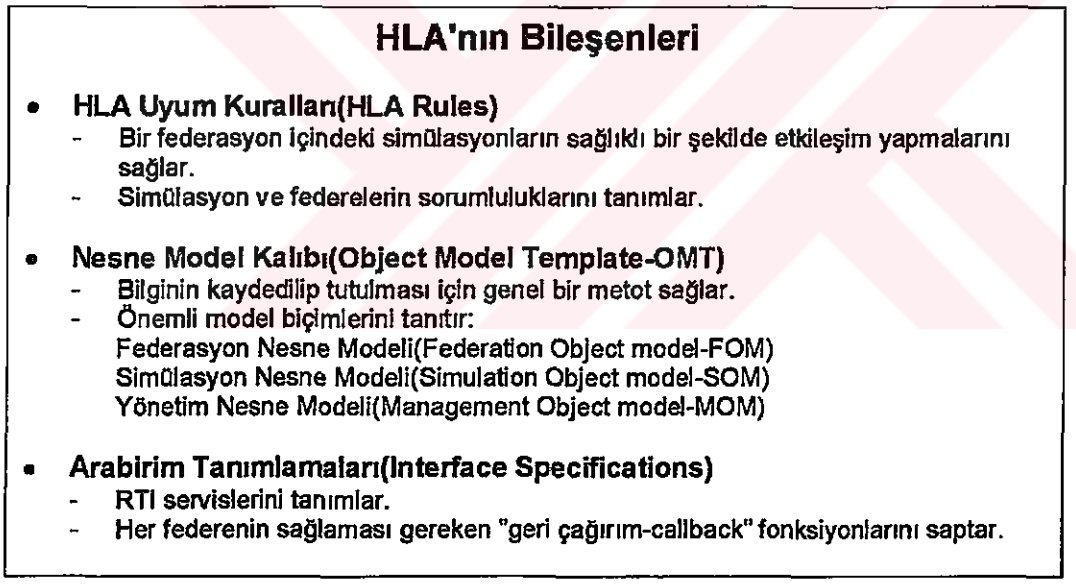
HLA, üç temel kavram tarafından tanımlanmıştır. Bunlar:

1. HLA Uyum Kuralları(HLA Compliance Rules)
2. Nesne Model Kalıpları (Object Model Templates) ve
3. Arabirim Tanımlamalarıdır(Interface Specifications).[6]

Şekil 2.9'da HLA bileşenlerinin kısaca özeti görülmektedir. Şimdi bu HLA bileşenlerini biraz genişletelim.

2.5. HLA Kuralları

HLA tanımının ilk bileşeni, HLA'nın temelinde yatan prensipler veya kurallardır. Birbiriyle etkileşimli simülasyonlar kümesi, veya başka bir değişle federasyon, verinin(data) veya olayların(events) simülasyonlar arasında karşılıklı olarak transferi için bir usul tanımlamak zorundadır. HLA uyum kuralları olarak adlandırılan bu kurallar, federasyonun çalışması veya işlemesi sırasında, federelerin kusursuz ve sağlıklı bir şekilde birbirleriyle etkileşimini(interaction) sağlayabilmek için uyulması gereken kurallardır. Yani bu kurallar, bir federasyon içerisindeki simülasyonların kusursuz bir şekilde etkileşim yapmalarını temin eder. HLA federasyonları içinde, Runtine Infrastructure(RTI) ve federelerin sorumluluklarını tanımlar. On adet kuraldan oluşur. Bu kurallar kendi aralarında federe ve federasyon kuralları olarak ikiye ayrılmıştır.



Şekil 2.9. HLA Bileşenlerinin Özeti

1) Federasyon kuralları: Federasyonlar, veya etkileşimli olarak çalışan simulasyonlar grubu, veya federeler, OMT formatı dahilinde bir FOM'a sahip olmaya gereksinim duyarlar. Çalışma süresi boyunca, herhangi bir anda, bir nesnenin bir örneğinin belirli bir özelliği(attribute) sadece bir federe tarafından sahip olunabildiği için, bütün nesne tasvirleri veya gösterimleri federeler içerisinde(RTI içinde değil)

meydana gelir. Federeler arasındaki bilgi alış verişi, HLA ara birim tanımlamalarını kullanan RTI aracılığıyla olur.

2) Federe kuralları: Federe kuralları, sadece belirli federelere uygulanır. HLA idaresi altındaki her federe, genel bilgilerini, OMT'yi kullanarak, kendi SOM'ları içinde belgelemelidir. SOM içerisinde bulunan bu bilgileri temel alan federeler, bilgi alıp vermeli, nesne mülkiyet özelliklerini transfer etmeli ve nesne özelliklerini güncelleştirmelidir.

Bu Federasyon ve federe kuralları, ayrıntılı bir şekilde, Bölüm 3 içerisinde açıklanmıştır.

2.6. HLA Nesne Model Kalıpları

Nesne Model Kalıpları (Object Model Templates), her federasyon ve federe için gerekli olan HLA Nesne Modeli ile ilgili bilgileri kaydetmek ve tutmak için önerilmiş olan genel bir metottur. Bir federe tarafından yönetilen tüm nesneler ve etkileşimler, OMT standardına uygun olarak tanımlanmıştır(Şekil 2.10). OMT, HLA Nesne Model bilgisini açıklamak için genel bir metot sağlar.

Nesne Model Kalıbı

- **Nesne Model Kalıbı(OMT)**
  - HLA nesne model dokümantasyonu için genel bir çerçeve sağlar.
  - Simülasyon bileşenlerinin yeniden kullanımının ve simülasyonların karşılıklı çalışabilirliğinin gelişmesini sağlar
- **Gerekli Bilgi**
  - Nesne Sınıf Yapısı Tablosu
  - Nesne Etkileşim Tablosu
  - Özellik/Parametre Tablosu
  - FOM/SOM Sözlüğü
- **İsteğe Bağlı Bilgi (OMT İlaveleri)**
  - Bileşen Yapısı Tablosu
  - Birlikler Tablosu

Şekil 2.10. Nesne Model Kalıbı

Federasyon Nesne Modeli(FOM), Simülasyon Nesne Modeli(SOM) ve Yönetim Nesne Modeli(MOM), OMT yardımıyla tanımlanmıştır. Şekil 2.11, bu modellerin kısa bir özetini göstermektedir.

HLA nesne modeli, 'nesne' terimi kapsamında simulasyonun temel unsurlarının bir açıklamasıdır.

Nesne Modelleri

- **Federasyon Nesne Modeli(Federation Object Model-FOM)**
  - Her federasyon başına bir tane
  - Paylaşıma açık bütün bilgileri tanıtır(mesele, nesneler, etkileşimler gibi.)
  - Federeler arasındaki dağıtım meselelerine çözümler getirir(mesela, verinin düzenli olarak kodlanması)
- **Simülasyon Nesne Modeli(Simulation Object Model-SOM)**
  - Her federe başına bir tane
  - Bir federenin belirgin özelliklerini tanımlar
  - Harici bir şekilde kullanılabilen nesneler ve etkileşimleri sunar veya takdim eder.
  - Federenin dahili işlemleri üzerine odaklanır
- **Yönetim Nesne Modeli(Management Object Model-MOM)**
  - Genel tanımlama
  - Bir federasyonu yönetmek için kullanılan nesne ve etkileşimleri tanımlar

Şekil 2.11. Nesne Model Özeti

HLA, nesne modellerinin içerdiği bilgi konusunda herhangi bir kısıtlama koymaz. HLA, her federe ve federasyonun nesne modelinin standart bir nesne model kalıbını kullanmasını şart koşar. Bu kalıplar, simülasyonların yeniden kullanımını kolaylaştırmak için, üstü kapalı konuların herkes tarafından anlaşılmasına vasıta olması için tasarlanmıştır. Bu eksiksiz kalıplar, serbestçe kullanılabilecek şekilde tasarlanmışlardır ve nesne model kalıbı verisi hakkında mantıklı düşünmeyi sağlamak için bir takım araçlar geliştirilmiştir, tekrar kullanımı(reuse) ve masrafsız bir şekilde bilgi değişimini kolaylaştır.[4]

Nesne modelleri şu şekilde tarif edilir:

-Tasarlanmış bir simulasyon veya bir federasyonda gerçek dünyayı temsil etme işlemi için seçilmiş olan müşterek nesneler grubudur.

-Bu nesnelerin özellikleri(attributes), birlikleri(associations) ve etkileşimleridir.

-Nesneleri gerçek dünyada temsil etme hususunda, uzayla ilgili ve zamana ait çözümü kapsar.
-Nesnelerin temsil edilmesinde faydalanılan önemli modeller ve algoritmalar sağlar.

Not : Buradaki nesne modeli terimi, nesneye yönelik analiz ve dizayn metodolojileri konusundaki bazı metinlerde kullanılmış olan terimden farklıdır.

Federasyon Nesne Modeli (Federation Object Model-FOM), belirli bir federasyon için gerekli bütün müşterek bilgilerin(nesneler, özellikler, birlikler ve etkileşimler) bir açıklamasıdır.

Simulasyon Nesne Modeli (Simulation Object Model - SOM) ise, bir federasyonda harici bir şekilde kullanılabilen özel bir simulasyon için nesneleri, özellikleri ve etkileşimleri tarif eder.

HLA SOM, sonraki federasyonlara aktarılabilen nesneler, özellikler ve etkileşimlerin tipleri açısından simulasyonu (federe) tarif eder. SOM, iç tasarım bilgisinden ayrıdır; daha doğrusu, federasyonun parçası olarak bilgi değişimi için simulasyonun kabiliyetleri hakkında bilgi sağlar. SOM, bir federenin bir federasyon içine dahil olabilmesi için o federenin uygun olup olmadığını değerlendirmeyi de kolaylaştırır.

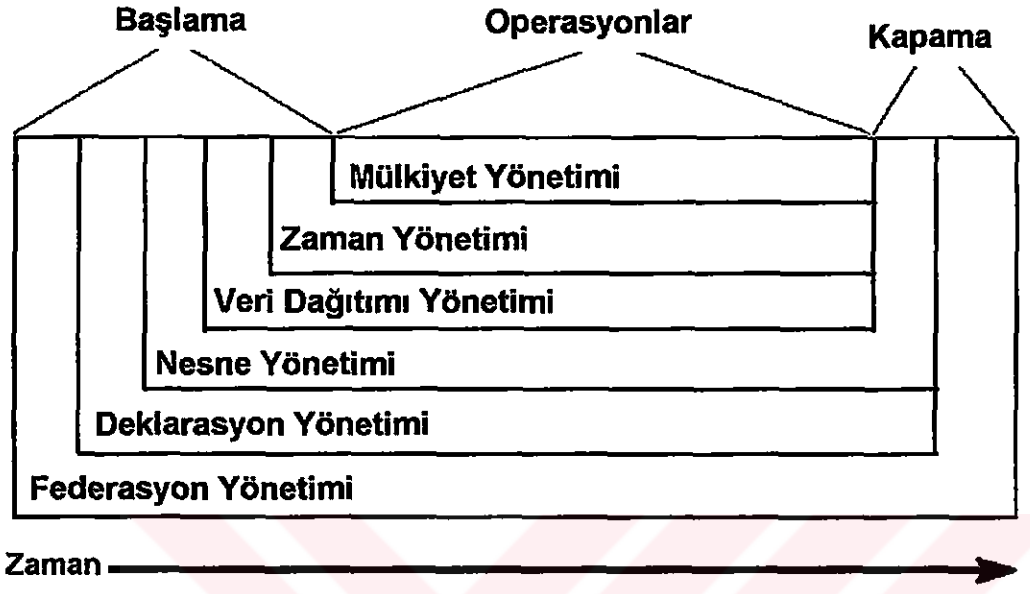
HLA veri ve yapıyı birbirinden ayırır. OMT'ye uygun olarak tanımlanmış nesne ve etkileşimler, HLA-türevli yazılıma göre ayarlanmaksızın değiştirilebilir ve tasarlanabilir.

Sonuç olarak; FOM, SOM ve MOM, HLA Nesne Model Kalıbı(HLA Object Model Template-OMT) olarak adlandırılan standart bir form kullanılarak belgelenmiştir.

2.7. HLA Arabirim Tanımlamaları

DoD, HLA konusuyla alakalı RTI ve federeler arasındaki fonksiyonel arabirimi ayrıntılı bir şekilde tanımlamıştır. Ayrıntılı bilgileri ve şartları belirten bu belge, Arabirim Tanımlamaları olarak adlandırılmıştır. RTI, bu tanımlamaların bir

uyarlamasıdır. Nasıl ki dağıtık(distributed) bir işletim sistemi uygulamalara bir takım hizmetler veya servisler sağlarsa, RTI'da benzer bir şekilde federelere servisler sağlar. HLA Arabirim Tanımlamaları(Interface Specifications), bu hizmetlere erişim yolu olarak tanımlanmıştır.



Şekil 2.12. Bir Federasyonun Yaşamı Üzerindeki HLA RTI Servisleri

HLA arabirim Tanımlamaları içinde tanımlanmış olan RTI hizmetleri, kurma(setup), çalışma ve kapama(shutdown) sırasında federasyon tarafından kullanılır. Şekil 2.12'de, bir federasyon ve çeşitli hizmet gruplarının federasyonun yaşamı üzerinde ne zaman kullanılabileceği görülmektedir.

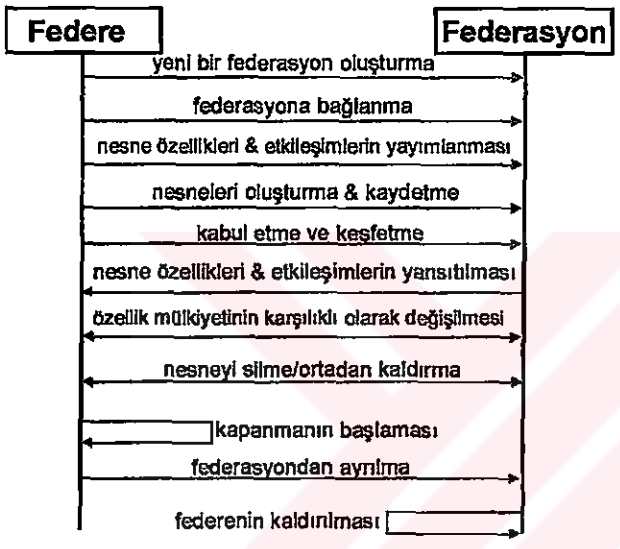
Bölünmüş Yönetim Alanları

Yönetim Alanı	Tanımlanmış Aktiviteleri
Federasyon Yntm	Biruygulamanın kontrolu
Deklarasyon Yntm	Karşılıklı veri değişiminin görüşülmesi
Nesne Yntm	Varlığın yaşam biçimi ve karakterlistiklerinin iletilmesi
Mülkiyet Yntm	Özellik mülkiyetinin paylaşımı
Zaman Yntm	Federasyon saatini etkileme
Veri Dağıtımı Yntm	Bilginin belirli bir yolla gönderilmesi
RTI Destek Hizmetleri	Uygulamadan yararlanma

Şekil 2.13. Bölünmüş yönetim alanları

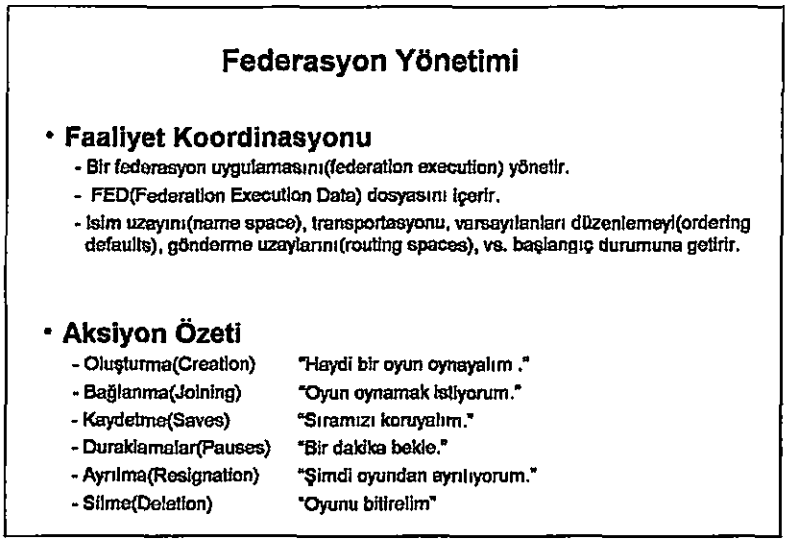
Arabirim Tanımlamaları, federelerin, federasyon veya bir başka federe ile nasıl bir şekilde etkileşim yapacaklarını ifade eden bir belgedir. Şekil 2.14'te bir federe ve bir federasyon arasındaki karşılıklı olarak yapılan etkileşimin(interplay) bir örneği görülmektedir.

Bu doküman altı yönetim alanına bölünmüştür. Aşağıda bu altı yönetim alanı kısaca açıklanmıştır. Daha sonraki bölümlerde RTI ve bu altı yönetim alanı daha ayrıntılı bir şekilde ele alınacaktır. Şekil 2.13'de herbir yönetim alanının fonksiyonları kısaca özetlenmiştir.



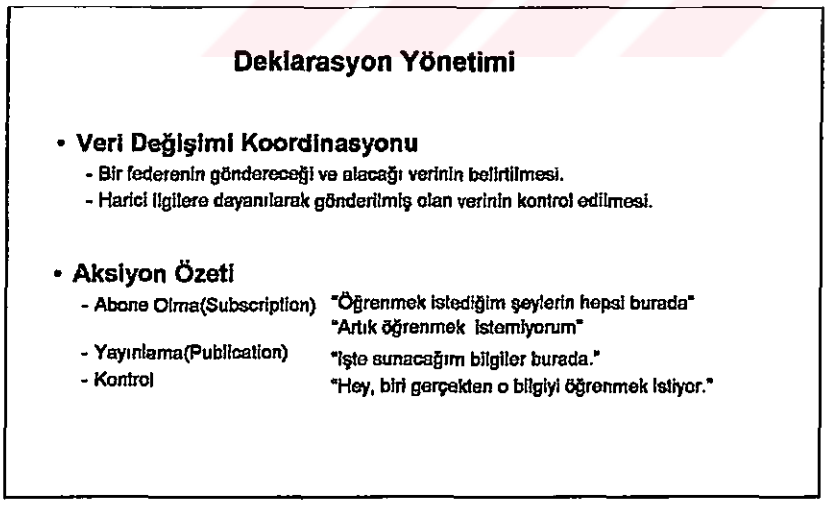
Şekil 2.14. Federe-Federasyon Karşılıklı Etkileşimi

i-Federasyon yönetimi: Federasyon yönetimi, federasyonları oluşturma, federelerin federasyonlara bağlanması, federasyon boyunca senkronizasyon noktalarının gözlemlenmesi, federasyon kapsamında kayıtların ve yeniden kurmaların gerçekleştirilmesi, federelerin federasyonlardan ayrılması ve federasyonların yıkılması gibi bir takım işlemleri kapsar. Şekil 2.15, Federasyon Yönetimi profilini göstermektedir.



Şekil 2.15. Federasyon Yönetimi Özeti

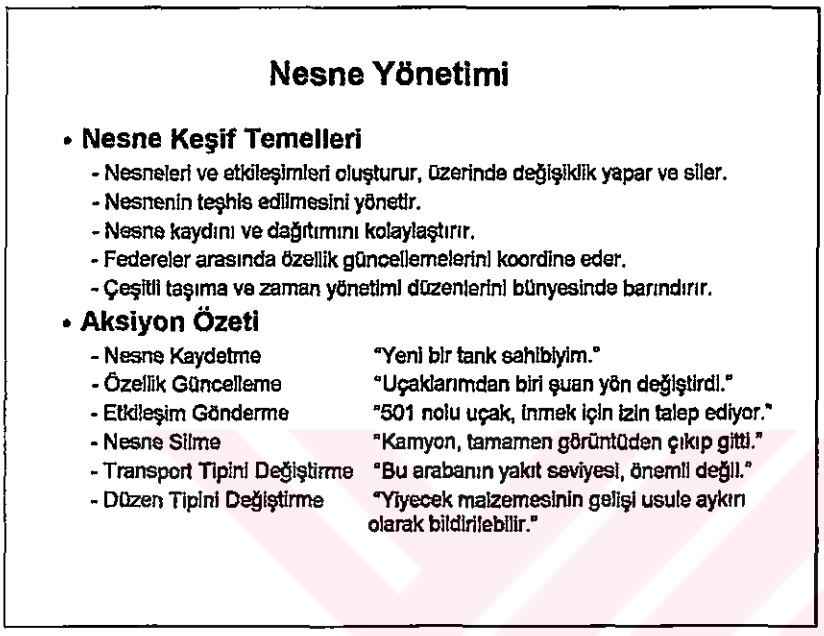
ii-Deklarasyon Yönetimi: Deklarasyon yönetimi, yayımlama(publication), abone olma(subscription) ve bir takım destekleyici kontrol mekanizmalarını kapsar. Nesneleri(veya nesne parçaları) veya etkileşimleri üreten federeler, tam olarak neyi yayınlayabileceklerini deklare etmelidirler. Şekil 2.16, deklarasyon yönetimi yoluyla üstesinden gelinen aksiyonların özetini ve temel koordinasyon görevlerini göstermektedir.



Şekil 2.16. Deklarasyon Yönetimi Özeti

iii- Nesne Yönetimi: Nesne Yönetimi, nesne üreticisi tarafı ile ilgili olarak örnek kaydetme(registration) ve örnek güncellemeleri(updates), nesne tüketicisi tarafı ile

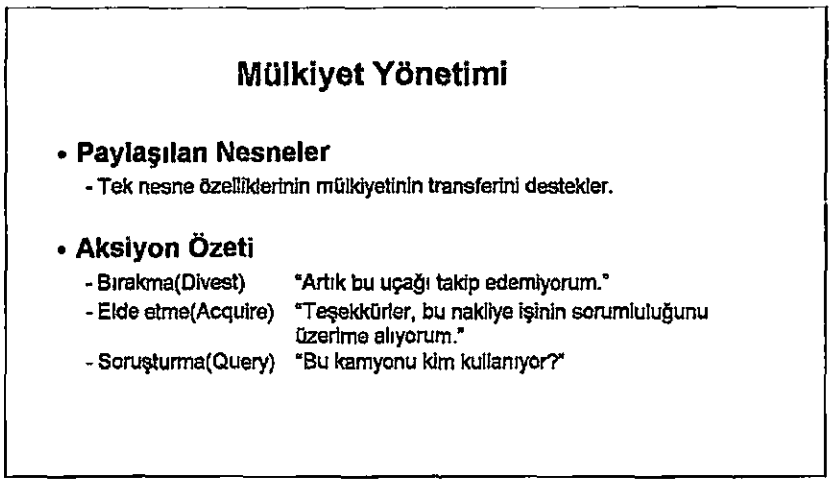
ilgili olarak ta örnek keşfetme(discovery) ve yansıtma(reflection) işlevlerini kapsar. Nesne yönetimi, aynı zamanda, etkileşimleri gönderme ve alma ile ilişkili olan bir takım metotları, tüketicinin talebine dayanan örnek güncellemelerini kontrol etme işlemini ve diğer çeşitli destek fonksiyonları içerir. Şekil 2.17, nesne yönetimi tarafından gerçekleştirilen aksiyonların bir özetini ve nesne keşfinin temellerini göstermektedir.



Şekil 2.17. Nesne Yönetimi Özeti

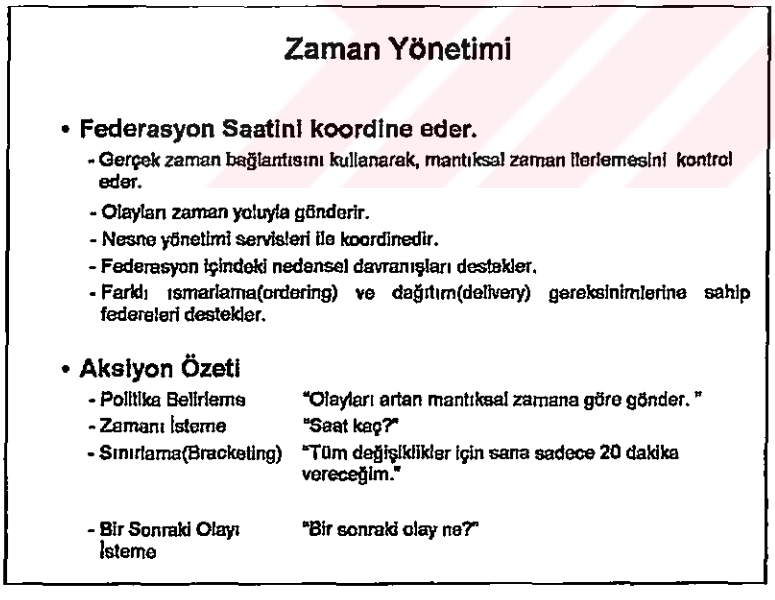
iv- Mülkiyet Yönetimi: RTI, nesne örneklerini güncelleştirme ve silme sorumluluğunu paylaşmayı birkaç sınırlama ile birlikte federelere bırakır. Herhangi bir anda bir nesne örneği, sadece bir tek federe tarafından sahiplenilebilir. Bu gibi durumlarda, nesne örneğine sahip olan federe, nesne ile ilgili bütün özellikleri güncelleme ve o nesne örneğini silme sorumluluklarına da sahip olur. Ancak bir tek nesne örneği ile ilgili güncelleme sorumluluğunu paylaşmak iki veya daha çok federe için mümkün olabilir. Bir nesnenin güncelleme sorumluluğu paylaşıldığı zaman, katılımcı federelerin herbiri, nesne özelliklerinin herkese açık olmayan(exclusive) kümesinin sorumluluğuna sahip olur(Belirli bir nesne örneği için bazı özellikler sahip olunamazlar. Mesela, hiçbir federe güncelleme sorumluluğuna sahip değildir.). Sadece bir federe, belirli bir zamanda özel bir nesnenin özel bir özelliğinin güncelleme sorumluluğuna sahip olabilir. Ayrıca, bir federe, belirli bir zamanda, sadece bir nesne örneğini silme imtiyazına(*privilege to delete*) sahip olur. Burada

bahsi geçen nesne yönetiminin faaliyetleri Şekil 2.18'de özetlenmiştir. Mülkiyet yönetimi ile ilgili ayrıntılı bilgiler Bölüm 5.9 da belirtilmiştir.



Şekil 2.18. Mülkiyet Yönetimi Özeti

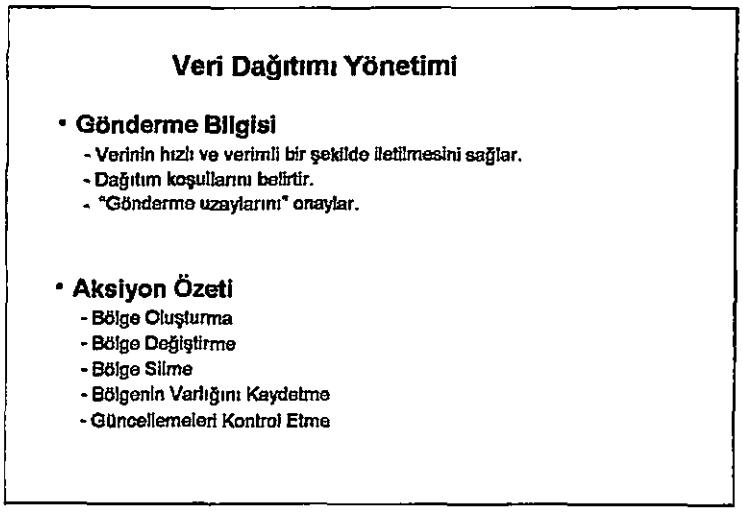
v-Zaman Yönetimi: Zaman yönetiminde, zaman yönetim politikalarını uyarlama ve zamanın sağlıklı bir şekilde ilerleyişinin sağlanması için gerekli olan mekanizmalar bulunmaktadır. Şekil 2.19'da zaman yönetimi özeti görülmektedir.



Şekil 2.19. Zaman Yönetimi Özeti

vi- Veri Dağıtımı Yönetimi: Veri Dağıtımı Yönetimi(Data Distribution Management-DDM), yayımlama ve abone olma ilgilerini izole etmeye destek olmak için esnek ve

kapsamlı bir mekanizma sağlar. DDM, verinin sağlıklı bir şekilde alınıp verilmesini sağlar. Şekil 2.20'de veri dağıtımı yönetiminin aksiyonlarının özeti görülmektedir.



Şekil 2.20. Veri Dağıtımı Yönetimi Özeti

2.8. HLA'nın Faaliyet Alanı

HLA, fonksiyonel alanların(mesela, eğitim, olasılık planlaması, analiz ve kazanma-acquisition gibi) geniş bir aralığına uyarlanabilir.[4]

HLA, aynı zamanda bütün simülasyon tiplerine uygulanabilir olarak geliştirilmiştir, ve HLA'ya uygun olarak yeni simülasyonların geliştirilmesi hedef alınmıştır. HLA'nın mevcut simülasyon programları üzerinde uygulanmasında genel olarak bir engel söz konusu değildir, ancak bilinen bir gerçektir ki, halihazırdaki mevcut sistemlerden bazıları HLA'nın yapısına kolayca uydurulabilirler iken, diğerlerinin sisteme dahil edilebilmeleri çok zor ve aynı zamanda pahalı olabilir.

2.9. HLA'nın Rolü

HLA, hem tekrar kullanımı(reuse) hem de karşılıklı çalışabilirliği(interoperability) kolaylaştırmak maksadıyla, simulasyon sistemi geliştiricileri(developers) ve planlayıcıları(policy makers) olarak adlandırılan iki türlü kullanıcıya hizmet etmesi için tasarlanmıştır. RTI, simülasyon sistemi dizaynına ve uygulama meselelerine hitap etmesi için sistematik ve tutarlı bir temel sağlar.[4]

Geliştiriciler, spesifik sistemlerin tasarımına bir giriş olarak HLA'yı kullanacaklardır. Tasarım aşamasında, sistem geliştiricilerine HLA tanımlamaları sağlanarak, yeni üretilecek olan simülasyonların karşılıklı olarak çalışabilirliği konusu için bir temel sağlanmış olacaktır. Birçok zor konu hala sistem düzeyinde çözülmeye ihtiyaç duymaktadır. En geniş kapsamlı HLA dizaynının bile, sistem seviyesine gelebilmek için pek çok zor şeyi terk etmesi üzerinde durulması gereken önemli bir konudur. Bugün, ölçeklenebilirlik(scalability) veya çok seviyeli güvenlik(multi-level security) benzeri bütün zor problemleri çözen bir sihirli değnek yapısı yoktur. HLA, bireysel sistem uyarlamalarının bu tür konulara da hitap etmesi için bir çerçeve(framework) sağlar.

Planlayıcılar(policy makers), DoD ile ilgili birimlerin veya belirli program hedeflerinin desteklemek maksadıyla simülasyonların nasıl geliştirileceği konusunda kararlar verebilmek için yapı olarak HLA'yı kullanacaklardır. Pek çok durumda, simülasyonların birbirleriyle karşılıklı çalışabilirlik(interoperability) konusu, katılımcı simülasyonların belirli biçimlerde dizayn edilmiş olmalarını gerektirebilir. Ancak, HLA, simülasyonlar üzerinde bu gibi kısıtlamalar koymaz; bu tür kısıtlamalar, planlayıcıların ilgi alanına girer.

2.10. HLA Dizaynının Temeli

HLA'nın geliştirilme sebebinin temel dayanak noktaları aşağıda verilmiştir. Bunlar;

- Tek bir monolitik simulasyon kullanıcıların isteklerini karşılayamaz.
- Simülasyonlardan tam bir şekilde yararlanmak veya faydalı olacak şekilde onları bir araya getirmek, simulasyonların geliştirilmeleriyle mümkün olmaz.
- Gelecekteki teknolojik kabiliyetler ve çalışma biçimi türleri birbirine uyumlu olmalıdırlar.

Sonuç: Simülasyon federasyonlarının kurulabilmesi için birleştirilebilir(composable) bir yönteme ihtiyaç vardır.[4]

HLA tasarımı, kesin varsayımlar üzerine kurulmuştur. İlk olarak, HLA, modelleme ve simulasyon konusunda DoD'un fonksiyonel ihtiyaçlarının tümünü tek bir simülasyonun çözemeyeceği varsayımına dayanmıştır. Kullanıcıların ihtiyaçları, çok çeşitlidir. İhtiyaç duyulan uygulamaların teknik karmaşasını, tek bir simülasyon içerisine sığdırmak bugün itibariyle mümkün değildir, ancak gelecekte mümkün olabilir. Ayrıca, kullanıcıların gereksinimlerinin sürekli değişmesi nedeniyle, ileride simülasyonların hangi amaçla veya ne tür bir birleşim içinde kullanılacaklarını şimdiden tahmin etmekte mümkün değildir. Belki teknolojinin sürekli durum değiştirmesi ile paralel olarak, HLA, teknik buluş ve yeniliklere fazladan kaynak ve zaman ayırmaya bu bağlamda ihtiyaç duyacaktır.

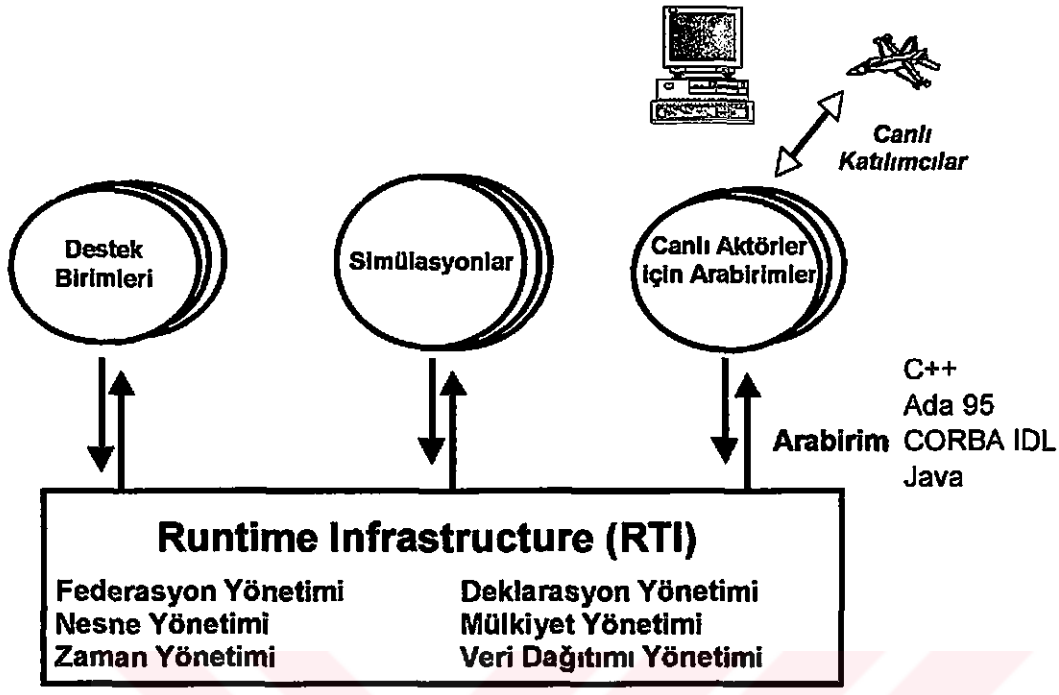
DoD içinde modelleme ve simulasyonun bütün yararlarını önceden söylemek mümkün olmadığı için, değişik biçimlerde yeniden kullanılabilen çeşitli simulasyonlar bakımından meseleyi ele almak önemlidir. Bu demektir ki, simulasyonlar geliştirilirken dikkat edilmesi gereken şey, diğer simulasyonlar ile birlikte kolayca çalışabilecek şekilde dizayn edilmiş olmalarıdır(yeni ve farklı uygulamaların yükünü kaldırabilmek için). HLA, özel simülasyonların dizaynındaki kısıtlamaları mümkün olduğunca en aza indirmeye gayret ederken, beklenmeden ortaya çıkan uygulamalar tarafından ihtiyaç duyulan yeni özellikleri destekleyebilecek genişlikte dizayn edilmiş olmak içinde çalışır.

Bu kabuller, farklı şekillerde HLA tasarımını etkilemişlerdir. Bu yapı, geniş işlevli modüler parçalara ve arabirimlere sahip olmalıdır. Ayrıca, HLA, simulasyonlar arasındaki karşılıklı çalışabilirlik(interoperability) konusu için gereken altyapı sisteminden özel simulasyonlar için ihtiyaç duyulmuş olan işlevselliği ayırmıştır.

2.11. HLA'nın Fonksiyonel Yapısı

Şekil 2.21'de HLA'nın genel fonksiyonel bileşenlerine nasıl bölündüğü görülmektedir. Birinci anahtar bileşen, federelerdir. Bir federe, bir bilgisayar simulasyonu, güçlendirilmiş bir simülatör, bir destekleme birimi (izleyici-viewer veya veri kayıt ünitesi gibi) veya bir canlı oyuncu veya aktöre arabirim, hatta ölçüm

yapılan bir alan dahi olabilir. Nesne temsillerinin tümü, federeler dahilinde olur. HLA, federe içinde neyin nasıl temsil edildiği konusunda bir kısıtlama koymaz.



Şekil 2.21. HLA'nın Fonksiyonel Yapısı

HLA, bütün federelerin, bir simulasyon içindeki nesnelerin başka simülasyonlarda ki nesnelerle etkileşimli olabilmesine izin verme konusunda belirlenmiş yeteneklere sahip olmasını gerektirir. Bunlar, HLA Kuralları konusunda daha detaylı olarak ele alınacaktır. Bir nesne modeli her federe ve bütün federasyon için tanımlanmıştır. Bir simulasyondaki nesneler, federasyon nesne modelinde(Federation Object Model) belirtildiği gibi veri değişimi boyunca diğer simülasyonlarda ki nesneler ile etkileşimli olurlar.

İkinci fonksiyonel bileşen, RTI'dır(Runtime Infrastructure). Daha öncede belirtildiği gibi RTI, federasyonlar için dağıtılmış bir işletim sistemi gibi çalışır. RTI, federe etkileşimleri ve federasyon yönetimi gibi destek fonksiyonları için federenin işletilmesinde simulasyonları destekleyen bir takım hizmetler sağlar. Bu hizmetler, daha sonraki bölümlerde ayrıntılı bir şekilde ele alınacaktır. Federeler arasındaki etkileşimler, RTI yoluyla olur. RTI ve Federeler arasındaki arabirim(interface), HLA'nın bir parçası olarak belirtilmiştir ve bütün simulasyonlar ve federasyonlar için geneldir.

Üçüncü fonksiyonel bileşen, arabirimdir(interface). HLA arabirim tanımlamaları, federeler arasındaki etkileşimleri desteklemek için RTI hizmetlerinin gerekli olmasında ve RTI ile etkileşimli olabilmelerinde, federeler için standart bir yol sağlar. Bu ara birim(interface), bağımsız bir uygulamadır ve özel nesne modelleri ve herhangi bir federasyonun veri alışverişi gereksinimlerinden de bağımsızdır.[4]

Simülasyon sistemlerinin diğer iki kabiliyeti, bu yapı tarafından desteklenmiştir. İlk olarak, HLA, simulasyon verisinin toplanmasını ve simülasyon faaliyetlerinin izlemesini sağlar. HLA içinde, bu araçlar, HLA arabirimini kullanan RTI ile etkileşim halinde olur ve simulasyonlar gibi davranır.

İkinci olarak, HLA, ölçme platformları veya canlı C2 sistemleri gibi, canlı katılımcılar için ara birimler sağlar. Canlı katılımcılar, HLA'nın nazarında simulasyon gibi davranan herhangi bir şey sayesinde simüle edilen dış dünyayla etkileşim halinde olurlar ki bu, simüle edilen dünyada gerçek dünya tasvirlerinin gereksinimlerini karşılar ve simüle edilen dünyadan canlı sisteme geçişi kolaylaştırır.

2.12. HLA Nesne Görünümü

HLA, dünya nesne görüşü etrafında yapılandırılmıştır. Bu bakış açısı, çeşitli nedenlerden dolayı faydalıdır. İlk olarak, simulasyon içerisinde temsil edilen gerçek dünya yaşamını tanımlamak için pratik bir yol sunar. Simulasyonlar arasında ve içindeki gerçek dünya tasvirlerinin genel anlayışını ilerletir. Simülasyon kullanıcıları ile geliştiriciler arasındaki iletişimi kolaylaştır. Yani, yalnız özel simülasyonların geliştiricileri arasında değil, aynı zamanda hem kullanıcılar hem de geliştiriciler arasında iletişim için yararlı bir mekanizma sağlar. Simulasyonlar ile bir federasyon içindeki genel simulasyon ortamlarını bir araya getirmek için çalışır.

Gerçek odur ki, nesneler tarafından tanımlanılmış olan bu yapı, simülasyonların mutlaka nesneye yönelik bir programlama dili(object oriented language) dahilinde geliştirilmesi anlamına gelmez. Yani HLA Nesne bakışının kullanımı, uygulama

yöntemleri hakkında hiçbir şey şart koşmaz (örneğin; nesneye yönelik programlama dillerinin kullanımı gibi).

Nesneler, simülasyonlarla ilgili olan gerçek dünya elemanlarıdırlar. Bu nesneler, özellikleri arasında belirlenmiş olan kimlikleri, durumları, davranışları ve akrabalıkları ile nitelendirilirler.

Nesneler, üç temel özelliğe sahip olurlar. Bunlar:

Kimlik(identity): Diğer nesnelerden bir nesneyi ayırt etmemize yarayan belirleyici bir özelliktir(mesela isim gibi).

Durum(state): Bir anlık zaman içinde bir nesnenin bütün statik ve dinamik özelliklerinin birleşimidir.

Davranış(behaviour): Bir nesnenin durum değişiklikleri karşısında göstermiş olduğu etkiler veya tepkilerdir.

Bir nesnenin bir başka nesneye akrabalığı(relationship) aşağıdaki 3 terim aracılığıyla belirlenmiştir:

Özellikler(attributes): Diğer nesnelere ulaşabilen bir nesnenin durum değişkenleri ve diğer parametreleridir.

Ortaklık(association): İki nesne arasındaki kavramsal ilişki.

Etkileşim(interaction): Bir nesnenin durumunun bir başka nesnenin durumunda yaptığı etkidir.

2.13. Simülasyonların Karşılıklı Çalışabilirliğinin(Interoperability) Tanımı

M&S Interoperability Tanımı(DoDD 5000.59): Bir simülasyon, diğer bir simülasyona hizmetler veya faydalar sağlar, ve başka başka simülasyonlardan hizmetler ve yardımlar alır, ve bu hizmetlerden kendi çıkarları doğrultusunda faydalanır, böylece, bu karşılıklı veri alış verişi, simülasyonların birlikte çalışabilmelerini olanaklı kılar.[4]

2.13.1. HLA ve Simülasyonların Karşılıklı Çalışabilirliği

Burada sunulan simülasyonların karşılıklı çalışabilirliği(interoperability) konusu, daha öncede belirtildiği gibi HLA'nın bütün amacını yansıtır. Bu amaç, farklı simülasyonların ortak bir amaç doğrultusunda veriyi etkin olarak paylaşabilmesini sağlamaktır.

Tanımdan da anlaşıldığı gibi, simülasyonların karşılıklı çalışması sırasında iki durum söz konusudur. Bunlar; *etkin veri paylaşımı(effective data sharing)* ve *tutarlı veri yorumudur(consistent data interpretation)*. Simülasyonların karşılıklı çalışabilmesi, sadece karşılıklı olarak veri değişimini değil, aynı zamanda, bu verinin tutarlı olarak karşıdaki simülasyonların anlayacağı bir şekilde yorumlanmasını da gerektirir.

HLA, federelerin, RTI ile aralarındaki arabirim yoluyla diğer federeler ile karşılıklı veri değişimi yapabilecek bir şekilde dizayn edilmiş olmalarına ihtiyaç duyar. Bu, federelerin bir federasyon ile çalışmalarını düzenlemelerine, veri değişimi yapabilmelerine ve federasyonlara katılabilmelerine olanak sağlar.

HLA Federasyon nesne modelleri, karşılıklı olarak değişilen verinin tutarlı bir şekilde yorumlanmasını kolaylaştırır. Federasyon nesne modeli, nesnelerin genel özellikleri, nesnelerin kurduğu ortaklıklar, etkileşimler, kararlılık seviyeleri ve nesne tasvirlerinde kullanılan önemli modeller ve algoritmalar konularında bilgiler sağlar.

2.13.2. Simülasyonların karşılıklı çalışabilirliğini gerçekleştirme

Karşılıklı çalışabilirlik(interoperability) konusu, bu konu üzerinde çalışmalar yapan çalışma gruplarının kendi gereksinimlerine göre bir önem derecesine sahiptir. Evrensel(universal) bir karşılıklı çalışabilirlik, ortak bir nesne modeli gerektirmelidir. Evrensel karşılıklı çalışabilirlik(orijinal amaca veya teknik uygulamaya bakmayarak, bir simülasyonun diğer bir simülasyonla karşılıklı çalışabilme yeteneği), bugünün teknolojisiyle mümkün değildir[4]. Gerçekçi bir şekilde belirtmek gerekirse, bu konu üzerinde çalışmakta olan grupların ihtiyaçları yoluyla belirlenen karşılıklı çalışabilirlik düzeyi sayesinde, simulasyonların karşılıklı çalışabilirliğine kademe kademe ulaşılacaktır.

Yeniden kullanım(reuse) ve taşınabilirlik(portability), simülasyonların karşılıklı çalışabilirlik konusundan farklıdır. Simülasyonların karşılıklı çalışabilirliği, çalışma sırasında, farklı federeler arasındaki mantıksal bilgi değişimi ile ilgilenirken, yeniden kullanım(reuse), yeni bir simülasyonun geliştirilmesi sırasında, bileşenlerin(örneğin; fikirler, bütün simülasyonlar, kod satırları, gibi.) adaptasyonu işlemiyle ilgilenir.

Taşınabilirlik(portability), mevcut bir simülasyonun bir bilgisayar platformundan veya işletim sisteminden bir başkasına adaptasyonunun adıdır.

2.13.3. Simülasyonların karşılıklı çalışabilirliğinin(interoperaility) aşamaları

HLA kuralları, arabirim tanımlamaları ve nesne model kalıpları, karşılıklı çalışabilirlik için minimum temel araçları sağlar. Bunlar çalışma anındaki veri transferi için gerekli mekanizmaların kurulmasına yardımcı olurlar ve farklı amaçlar için uygun simülasyonları saptamada bir takım yöntemler sağlarlar.

Ekstra karşılıklı çalışabilirlik, simülasyonların kendileri içindeki iç tasvirlerinde daha fazla tutarlılık(consistency) gerektirir. Örneğin;

- Genel olmayan nesneler ve özelliklere ilişkin anlaşma

- HLA tarafindan şart koşulmasının dışında, modeller ve algoritmalar hakkındaki bilginin yayınlanması
- Modeller/algoritmalar ve parametre değerlerinin paylaşılması
- Uygulanabilir simulasyon bileşenlerinin paylaşımı

Herhangi bir özel uygulama için gerekli olan bu tutarlılığın boyutu, o uygulamanın karakteristiklerine bağlıdır.

HLA kendi içinde karşılıklı çalışabilirlik konusunu garanti etmek için yetersizken, simülasyon ve federasyon geliştiricilerinin kendi hedeflerine ulaşmada ihtiyaç duydukları derecede karşılıklı çalışabilirliği elde etme konusunda teknik bir çerçeve sağlar.

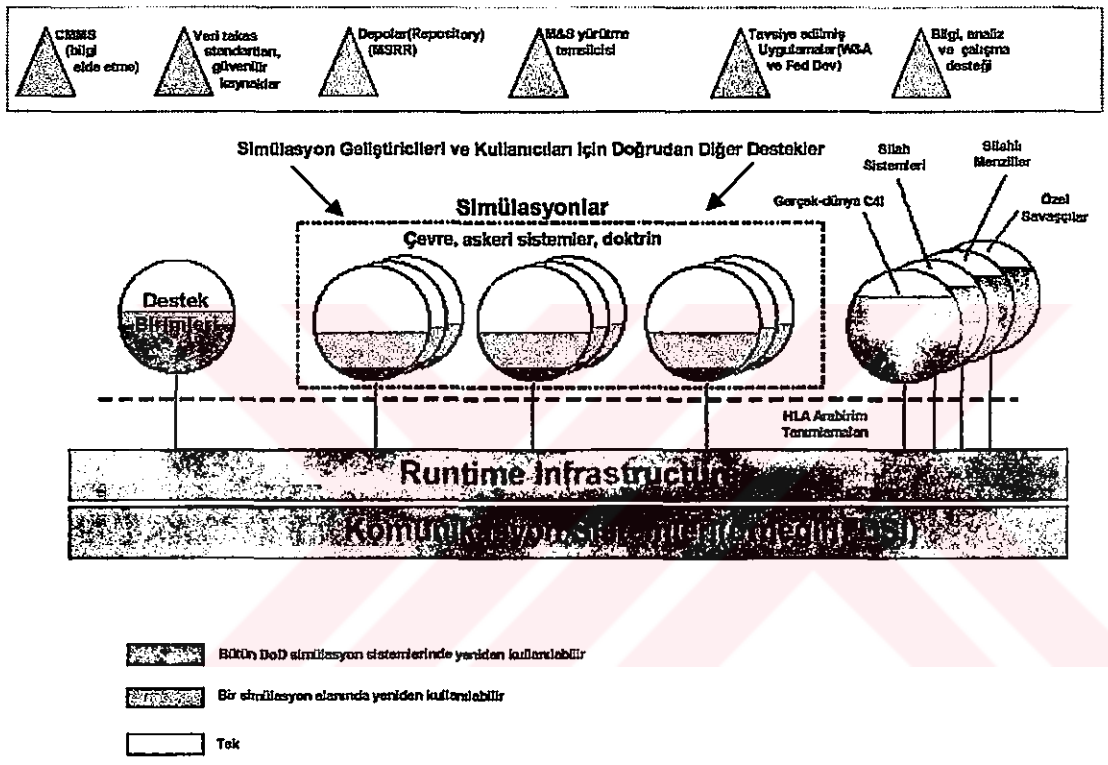
2.14. Yeniden Kullanım(Reuse) için Elverişli Durumlar

HLA, simülasyonların yeniden kullanımı için genel bir yapı sağlar. Daha öncede belirtildiği gibi, HLA, DoD içindeki bütün kullanımların ve kullanıcıların gereksinimlerini bir tek simülasyon veya modelin karşılayamayacağı dayanak noktası üzerine kurulmuştur. Bir amaç doğrultusunda geliştirilmiş simülasyonlar kümesi veya özel bir simülasyon, HLA'nın federasyon kavramı(bir araya getirilebilir etkileşimli simülasyonlar kümesi) altında bir başka uygulamayı kullanabilir. HLA'nın amacı, farklı simülasyonlar içerisindeki mevcut yeteneklerin yeniden kullanımını destekleyecek olan bir yapı kurmak ve en nihayetinde yeni bir amaç için yapay bir ortam oluşturmada gereksinim duyulan zamanı ve maliyeti en aza indirmektir.

Yeniden kullanım konusunun önemli bir alanı, HLA federasyonlarını destekleyen yazılımdır. Burada bahsi geçen yazılım, farklı simülasyon uygulamalarının geniş bir alanını destekleyen bir yazılım sınıfıdır. Bu yeniden kullanılabilir yazılımın geliştiricilere sağlaması yoluyla, maliyetlerin, geliştirme zamanlarının ve gelecekteki simülasyon gelişimleri için teknik risklerin uygun bir şekilde kullanılması amaçlanmıştır.

Yeniden kullanılabilir destek yazılımına en iyi örnek RTI'dır. RTI federasyonun belkemiğidir. RTI, DoD içerisindeki M&S uygulamalarının geniş bir aralığında uygulanabilen ve yeniden kullanılabilen bir yazılım olarak tasarlanmıştır.

RTI, daha önce de belirtildiği gibi, simülasyonlar için temel hizmetler grubu ve canlı oyuncular için arabirimler sağlar. Genel olarak federeler olarak adlandırılan uygulama yönetim parçaları, RTI aracılığıyla karşılıklı çalışabilir. Çünkü, HLA arabirim tanımlamaları bütün uygulamalar için standarttır.



Şekil 2.22. Simülasyonların yeniden kullanımının(reuse) yapısı

Yeniden kullanım alanları Şekil 2.22'de görülmektedir. Yeniden kullanılabilen parçalar veya bileşenler, şekil üzerinde, gri renk tonuyla gösterilmiştir. MSRR, şekil üzerinde üçgenler kullanılarak gösterilmiştir.

Yeniden kullanım konusunun bir başka önemli alanı da, simülasyon ve federasyon geliştirme çalışmaları için bir takım kaynaklar sağlamasıdır. Bunlar; CMMS(simülasyon geliştiricileri için genel bir dünya görünümü sağlayarak, gerçek dünyanın en düşük düzeyde soyutlanması), genel veri standartları ve genel çevre

tasvirleridir. Bu kaynaklara ve diğer kaynaklara, Modelleme ve Simülasyon Kaynak Deposu(Modelling and Simulation Resource Repository-MSRR) aracılığıyla ulaşılabilir.

2.15. Federasyon Geliştirme ve Uygulama İşlemi(FEDEP-The Federation Development and Execution Process)

Federasyon Geliştirme ve Uygulama İşlemi(FEDEP- The Federation Development and Execution Process), ilk HLA proto-federasyonlardan kazanılan deneyimlere dayanılarak geliştirilmiştir ve HLA federasyon uyarlamalarına ek olarak kullanıcı deneyimleri sayesinde de gelişimini sürdürmüştür. FEDEP' in bir belge şeklinde düzenlenmesinin altında yatan neden, diğer HLA federasyon uygulamalarının bir deneyim bilgi tabanı olarak kullanabileceği bir biçimde federasyon geliştirme uygulamalarını bir çatı altında toplamaktır. FEDEP'in amacı, bir federasyonun gelişmesindeki anahtar faaliyetleri tanımlama şeklinde rehberlik yapmaktır ancak bir HLA federasyonunu geliştirme işlemini tek yönlü tarif etmek değildir. Bu bağlamda FEDEP, kısıtlamalar koymaktan daha ziyade tanımlayıcı bir rehber olarak görülebilir. [7]

FEDEP'in otomasyon olanaklarını incelemek amacıyla, FEDEP'i beş aşamaya ayırmak mümkündür: Tasarım, Gelişim, Test, İşletim ve Analiz. Tasarım, şu FEDEP faaliyetlerinden oluşur:

- Federasyon Finansörlüğü(Federation Sponsorship)
- Kavramsal Analiz(Conceptual Analysis)
- Senaryo Geliştirme(Scenario Development)
- Federasyon Tasarımı(Federation Design).

Gelişim aşaması:

- FOM Geliştirme
- Federasyon Geliştirme

Test aşaması:

- Federasyonu Test Etme

İşletim aşaması:

- Federasyonun İşletilmesi

ve Analiz aşaması ise:

- Sonuçlar
- Geribesleme' den oluşur.

Bu faaliyetler ile ilgili adımlar, yapı içerisindeki bileşenleri tanımlamak için kullanılacaktır.

Aşağıda özetlenmiş olan beş adımlı aşama, Şekil 2.23'de gösterilmektedir:

1. *Adım:* Federasyonun finansörü ve federasyon geliştirme ekibi, bir takım amaçlar üzerinde mutabakata varmalıdırlar ve o amaçları gerçekleştirmek için neyin başarılması gerektiğini belgelemelidirler.
2. *Adım:* Gerçek dünya alanının bir temsili(varlıklar ve görevler) geliştirilir ve bir takım gerekli nesneler ve etkileşimler açısından tanımlanılır.
3. *Adım:* Federasyonun katılımcıları belirlenir(daha önceden tanımlanmamışsa) ve karşılıklı bilgi değişim şartlarını ve sorumluluklarını açıkça belgelemek için bir FOM geliştirilir.
4. *Adım:* Bütün gerekli federasyon uygulama faaliyetleri gerçekleştirilir ve test etme işlemi, karşılıklı çalışabilme şartlarını sağlamak için gerçekleştirilir.
5. *Adım:* Federasyon işletilir, çıktılar analiz edilir ve federasyon finansörüne(sponsor) geribesleme sağlanır.

Şekil 2.23. Beş Adımlı FEDEP İşlemi

BÖLÜM 3. HLA KURALLARI

3.1. Giriş

High Level Architecture(HLA), bir takım kurallar, arabirim tanımlamaları ve nesne model kalıbı(Object Model Template-OMT) tarafından tanımlanmıştır. HLA'nın genel prensiplerine ek olarak bu bölümde, HLA federasyonları ve federelerine tatbik edilen bir kurallar kümesi tarif edilmiştir. Eğer bir HLA uyumundan bahsedilecekse zaten bir takım kurallar göz önünde tutulmuş olmalıdır. Şu an on ana kural mevcuttur. Ancak çalışmalara paralel olarak bu sayı arttırılabilir ve bu kurallar dokumanı geliştirilebilir. [8]

DoD genel teknik çerçevesinin HLA kapsamındaki bütün hedefi simulasyonların karşılıklı çalışabilirliğini(interoperability) ve simulasyon bileşenlerinin yeniden kullanımını(reuse) sağlamaktır. HLA, simulasyonların karşılıklı çalışabilirliği(interoperability) için yapısal bir temel sağlar; burada tanımlanan kuralların çoğu bu konuyla alakalıdır. Ancak, kabul edilmelidir ki HLA, simulasyonların karşılıklı çalışabilirliği(interoperability) için gereklidir, yeterli değildir.

3.2. HLA Kurallarının Özeti

HLA Kuralları federasyonlar ve federeler için olmak üzere ikiye ayrılır.

Federasyonlar için kurallar:

1. Federasyonlar, HLA Nesne Model Kalıbına(HLA Object Model Template-OMT) uygun olarak belgelenmiş bir HLA Federasyon Nesne Modeline(HLA Federation Object Model-FOM) sahip olacaktır.

2. Bir federasyonda, FOM içerisindeki nesnelerin bütün temsil veya gösterimleri, Runtime Infrastructure(RTI) içerisinde değil, federeler dahilinde olacaktır.

3. Federasyon işletimi veya uygulaması sırasında, federeler arasındaki FOM verilerinin karşılıklı değişimleri, tamamıyla RTI yoluyla gerçekleşecektir.

4. Federasyon işletimi veya uygulaması sırasında, federeler, HLA arabirim tanımlamalarına uygun olarak Runtime Infrastructure(RTI) ile etkileşimli olacaklardır.

5. Federasyon işletimi sırasında, nesne örneğinin bir özelliği, herhangi bir zamanda sadece bir federe tarafından sahip olunacaktır.

Federeler için kurallar:

6. Federeler, HLA Nesne Model Kalıbına(Object Model Templates-OMT) uygun olarak belgelenmiş olan bir HLA Simülasyon Nesne Modeline(Simulation Object Model-SOM) sahip olacaklardır.

7. Federeler, SOM'da belirtildiği gibi, kendi SOM'ları içerisindeki nesnelerin herhangi bir özelliğini güncelleştirebilecek(update) ve/veya yansıtabilecek(reflect) ve harici bir şekilde SOM nesne etkileşimlerini gönderebilecek ve/veya alabilecektir.

8. Federeler, SOM'da belirtildiği gibi, federasyon işletimi sırasında dinamik olarak özelliklerin mülkiyetini transfer edebilecek ve/veya teslim alabilecektir(accept).

9. Federeler, SOM'da belirtildiği gibi, nesne özelliklerinin güncelleştirilmelerini sağlayan şartlar altında(örneğin eşikler-thresholds) değişebilecektir.

10. Federeler, federasyonun diğer üyeleri ile veri alışverişini düzenlemeleri konusunda kendilerine izin veren yerel zamanı(local time) yönetebilecektir.

3.2.1. Federasyon kuralları

Bu kısımda, HLA federasyonlarına uygulanan beş kural açıklanmıştır. Her kural, kapsamı dahilinde tanımlanmıştır.

3.2.1.1. Federasyonlar, HLA Nesne Model Kalıbına(HLA Object Model Template-OMT) uygun olarak belgelenmiş bir HLA Federasyon Nesne Modeline(HLA Federation Object Model-FOM) sahip olacaktır. (Kural 1)

FOM, çalışma anında verinin karşılıklı değişimi konusunda federeler arasındaki anlaşmayı belgeler(mesela; değişimler belli bir değeri aştığı zaman güncellemelerin gönderilmesi gibi.). Aslında FOM, bir federasyonun tanımlanmasındaki başlıca unsurlardan biridir. HLA, hangi verinin FOM dahilinde olduğunu belirtmez(bu, federasyon kullanıcılarının ve geliştiricilerinin sorumluluğu altındadır). HLA, bir federasyonun yeni kullanıcıların kendi istekleri doğrultusunda yeniden kullanımını sağlamak için belli bir format(HLA Nesne Model Kalıbı-OMT) dahilinde belgelenmiş olan FOM'lara gereksinim duyar.

Karşılıklı bilgi değişimi konusundaki mutabakatın biçimi, HLA'nın önemli unsurlarından biridir. HLA, tamamen bağımsız bir alandır ve geniş kullanım alanlarında federasyonları desteklemek için kullanılabilir. FOM, HLA'nın belirli bir uygulamasındaki karşılıklı veri alışverişini tanımlamanın bir yoludur. Genel bir biçimde belgelenmiş olan sonuçların gereksinimleri ve bu anlaşma veya mutabakatların gelişimlerin biçimlendirilmesi vasıtasıyla, HLA bir federasyonun yeniden kullanımında yardımcı olan bir araç ve bir federasyonun anahtar elemanlarını anlama konusunda bir yol, bir yöntem sağlar. Ek olarak, FOM, federasyon için RTI'ın(Runtime Infrastructure) çalıştırılması ve başlatılması sırasında kullanılan bazı veriler için bir temel sağlar.

3.2.1.2. Bir federasyonda, FOM içerisindeki nesnelerin bütün temsil veya gösterimleri, Runtime Infrastructure(RTI) içerisinde değil, federeler dahilinde olacaktır. (Kural 2)

HLA'nın ardında yatan temel fikirlerden biri, her amaca uygun bir destek altyapısından simulasyonun özel işlevlerini ayırmaktır. HLA içerisinde, simule edilen nesnelerin temsili(mesela; özelliklerin mülkiyeti, değerleri güncelleme sorumluluğuna sahip olma olarak tanımlanmış olan "mülkiyet(ownership)" alanındadır) simulasyonlar(veya daha genel olarak federeler) içerisinde yapılır; RTI federasyon karşısında nesnelerin etkileşimini desteklemek için ihtiyaç duyulan yaygın bir işletim sistemi benzeri işlevsellik sağlar. Bütün nesne özellikleri federeler tarafından sahip olunmuştur, RTI tarafından değil.

Federasyon destek servislerinden simulasyon işlevselliğinin bu ayrılığı, birkaç sebepten dolayı sona ermiştir. İlk olarak, RTI servisleri, DoD(veya diğer) kullanıcılarının en geniş ihtiyaçları karşısında federasyonları desteklemek için ihtiyaç duyulan temel bir yeniden kullanılabilme yeteneklerin seti olması için tasarlanmıştır. Bunlar, aslında zaman düzenleme, veri dağıtımı vs. gibi federasyon işlemlerini destekleyen düzenleme ve yönetim hizmetleridir. Bu, kullanıcı veya uygulama alanının ihtiyaçlarını karşılaşmak için nesneleri temsil etme temel hedeflerine odaklanmaları konusunda federeleri serbest bırakma ek avantajını sağlar. Bu yaklaşım, bu temel yaygın hizmetler dahilinde zaman ve kaynak harcaması nedeniyle simulasyon geliştiricilerinin yükünü hafifletir. RTI, RTI servislerini desteklemek için nesne özellikleri ve etkileşimleri hakkındaki veriyi kullanabilir, ancak bunlar sadece RTI tarafından kullanılırlar, değiştirilemezler.

3.2.1.3. Federasyon işletimi veya uygulaması sırasında, federeler arasındaki FOM verilerinin karşılıklı değişimleri, tamamıyla RTI yoluyla gerçekleşecektir. (Kural 3)

HLA, FOM'a uygun olarak, nesne özellik değerlerinin ve etkileşimlerin koordineli bir şekilde değişimini desteklemek için, Runtime Infrastructure(RTI) içindeki hizmetlere ulaşma amacıyla bir takım arabirimler tanımlar. HLA çatısı altında,

federasyonlara katılan nesneler arasında olan etkileşimler, RTI hizmetleri kanalıyla veri değişimi yapılarak gerçekleştirilir. FOM'a dayanarak federeler, RTI'a, federe içindeki nesnelerin durum değiştirmesi ile ilgili olan özellik ve etkileşim verisinin yanında, sağlayacakları ve gereksinim duyacakları verinin içeriğinin ne olduğu konusunda da bilgi sağlayacaklardır. Ondan sonra RTI, koordinasyonu, senkronizasyonu sağlayacak ve tutarlı ve uygun bir federasyon uygulamasına imkan vermesi için, federeler arasında veri alışverişi sağlayacaktır.

Doğru veriyi doğru zamanda temin ederek ve federeler tarafından uygun bir şekilde kullanılacak olan bu veriyi, federelerin mesuliyetine bırakarak; RTI, federelerin, deklare etmiş oldukları gereksinimlerine uygun olarak kullanması için veri dağıtımını sağlar.

Bir federasyon uygulamasının yaşamı üzerindeki ve bir federasyon içindeki tüm katılımcıların koordinasyon ihtiyaçlarını uygun bir biçimde temin etmek için, mutlaka RTI servisleri kullanılmalıdır. Eğer bir federasyon, RTI servis takımından başka, paylaşılan nesnelerin veya nesneler arasındaki etkileşimlerin durum değişikliklerini temsil eden veriyi değiştirirse, bu dağıtılan uygulamanın tutarlılığı ihlal edilmiş olur. Federasyonlara genel RTI hizmetlerinin sağlanmasının gerekçesi, genellikle simulasyonlar arasındaki veri alışverişinde tutarlılığı sağlamada ve yeni federasyonların teşkili ve gelişiminin maliyetini azaltmada gerekli olan temel bir işlevsellik sağlamaktır.

3.2.1.4. Federasyon işletimi veya uygulaması sırasında, federeler, HLA arabirim tanımlamalarına uygun olarak Runtime Infrastructure(RTI) ile etkileşimli olacaklardır. (Kural 4)

HLA, RTI ve federeler arasındaki arabirimleri destekleme maksadıyla RTI hizmetlerine ulaşmak için standart bir tanımlamalar sağlar(bakınız HLA Arabirim Tanımlamaları). Federeler, RTI ile etkileşim halinde olabilmek için bu standart arabirimi kullanırlar. Arabirim tanımlamaları, simulasyonların RTI sistemi ile nasıl bir şekilde etkileşim halinde olabileceklerini tanımlar. Ancak, RTI ve arabirimin, farklı özelliklere sahip veri alışverişine gereksinim duyan geniş uygulama alanlarında

kullanılacağından dolayı, arabirim üzerinden değişilen özel nesne verisi hakkında bir şey belirtmez. FOM içerisinde, sadece federeler arasındaki veri alışverişinin şartları ve gereksinimleri tanımlanmıştır.

Federeler ile RTI arasında standart hale getirilmiş genel bir arabirimi gerektirerek, ortak bir API ile birlikte, HLA bağımsız gelişim ve uygulamalara izin verir. Simulasyonlar, RTI uygulaması olmadan bağımsız olarak çalışabilirler ve RTI için arabirimler geliştirebilirler ve RTI gelişimleri, simulasyon gelişiminin belirgin bir karşılığı olmadan devam edebilir. Nesne veri alışverişi gereksinimlerinden bağımsız olan arabirimler, FOM mekanizması boyunca uygun hale getirilmiş olan özel uygulamalara ihtiyaç duyulması sayesinde, DoD uygulamalarının geniş bir spektrumu(tayfi) karşısında genel bir arabirim tanımlamalarının yeniden kullanımına izin verir.

3.2.1.5. Federasyon işletimi sırasında, nesne örneğinin bir özelliği, herhangi bir zamanda sadece bir federe tarafından sahip olunacaktır. (Kural 5)

HLA, farklı federelerin, aynı nesnenin farklı özelliklerine sahip olmasına izin verir(örnek olarak, bir uçak simulasyonu, uçmakta olan bir sensörün konumuna sahip olabilirken, sensörün diğer özellikleri, bir sensör sistemi modeli tarafından sahip olunabilir.). Federasyon içinde veri uyumluluğunu sağlamak için, HLA, herhangi bir zamanda, bir nesne özelliğine sadece bir federenin sahip olmasına izin verir. HLA, uygulama sırasında bir federeden başka bir federeye dinamik bir şekilde mülkiyet(ownership) transferi için de bir mekanizma sağlar.

Uygulama sırasında mülkiyeti elde tutmak için araçlar sağlayarak ve özellik düzeyindeki nesne mülkiyetini tanımlayarak HLA, kullanıcı ihtiyaçlarını karşılamak maksadıyla simulasyonların çeşitli kombinasyonlarından faydalanma için esnek bir araçseti(toolset) sağlar.

3.3.2. Federe kuralları

Bu bölüm federelere uygulanan beş kuralı tarif eder. Her kural kendi kapsamı dahilinde temel bir açıklama olarak tanımlanmıştır.

3.3.2.1. Federeler, HLA Nesne Model Kalıbına(Object Model Templates-OMT) uygun olarak belgelenmiş olan bir HLA Simülasyon Nesne Modeline(Simulation Object Model-SOM) sahip olacaklardır. (Kural 6)

Federeler, simulasyonlar(bir modeli gerçek zamandan bağımsız bir şekilde uygulama metodu) veya federasyona katılan başka uygulamalar olabilir(simulasyon yöneticileri, veri kayıt üniteleri, canlı varlık arabirimleri ve pasif izleyiciler). HLA, her federenin HLA simulasyon nesne modeline(SOM) sahip olmasını şart koşar. HLA SOM, nesneleri, özellikleri ve federasyon içerisinde aleni olarak meydana getirilebilen federe etkileşimlerini içerir.

HLA, SOM içerisinde bulunan verinin kapsamı hakkında bir bilgi belirtmez; SOM içindeki verinin mesuliyeti simülasyon geliştiricilerine aittir. HLA, SOM'ların sadece tavsiye edilen bir format(HLA Nesne Model Kalıbı) dahilinde yazılmış olmalarını şart koşar.

Bilindiği gibi HLA'nın hedeflerinin başında, simulasyonların yeniden kullanımını(reuse) ve karşılıklı çalışabilirliği(interoperability) sağlamak gelmektedir. HLA, simülasyonlar içindeki temsillere erişime izin vererek ve simulasyonlar düzeyinde yeniden kullanımı temin ederek bu hedefleri sağlamaya doğru yönelir.

Federeler içinde mevcut nesne temsilleri hakkındaki bilgiye uygun bir şekilde erişiminin olmayışı, yeniden kullanımı(reuse) engelleyici bir etki yapar. SOM, yeni federasyonlardaki potansiyel federe uygulamalarının kolay bir şekilde tanımlanmasını sağlamak maksadıyla, bu federelerin göze çarpan yeteneklerini belirtmelerini şart koşarak bu engelleyici etkiyi giderir. Anlaşılan odur ki, potansiyel bir kullanıcı tarafından gereksinim duyulan ayrıntılı bilgi, SOM kapsamının dışındaki kaynaklara doğru yönelecektir. Ancak, varsayılmıştır ki, yeniden kullanım

üzerine kurulmuş simulasyonların nitelendirilmeleri konusuna kolay bir giriş sağlama yoluyla, potansiyel kullanıcılar, simulasyonların kendi uygulamalarına uyarlanabilmeleri düşüncesi için daha çok çaba harcayıp harcamayacakları konusunda daha etkili karar verebilirler(Modelleme ve Simulasyon Kaynak Deposunda(Modelling and Simulation Resource Repository-MSRR) bulunan simulasyon tarifleri aracılığıyla, özel simulasyon dokümantasyonu yoluyla veya simulasyon geliştiricisi ile doğrudan etkileşimli olarak).

3.3.2.2. Federeler, SOM'da belirtildiği gibi,kendi SOM'u içerisindeki nesnelerin herhangi bir özelliğini güncelleştirebilecek(update) ve/veya yansıtabilecek(reflect) ve harici bir şekilde SOM nesne etkileşimlerini gönderebilecek ve/veya alabilecektir. (Kural 7)

HLA, diğer federeler içinde temsil edilen nesnelerin federasyon uygulamalarının bir parçası olarak kullanılmaları için geliştirilen etkileşimleri ve nesne temsillerini oluşturma konusunda federelere izin verir. Dış ortam ile etkileşim konusundaki yetenekler, federenin SOM'u içerisinde belirtilecektir. Bu federe yetenekleri, federe içersinde dahili olarak tasarlanmış olan nesnelerin özellik değerlerini ihraç etme zorunluluğunu kapsar ve harici olarak temsil edilen nesnelere sahip bu etkileşimleri kullanma kabiliyetine sahiptirler(mesela; başlama-initiate, sezme-sensing ve tepki gösterme-reacting gibi.). Dahili nesneleri/özellikleri/etkileşimleri, harici bir şekilde takdim edebilme kabiliyetine sahip federeleri tasarlama yoluyla, simulasyonun yeniden kullanım mekanizmaları, başlangıçtan itibaren konumlarını koruyacaklardır.

3.3.2.3. Federeler, SOM'da belirtildiği gibi, federasyon işletimi sırasında dinamik olarak özelliklerin mülkiyetini transfer edebilecek ve/veya teslim alabilecektir(accept). (Kural 8)

HLA, farklı federelerin aynı nesnenin farklı özelliklerini edinmelerine izin verir(örnek olarak, bir uçak simulasyonu, uçmakta olan bir sensörün konumuna sahip olabilirken, sensörün diğer özellikleri, bir sensör sistemi modeli tarafından sahip olunabilir.). Bu yetenek, yeni bir gereksinimi karşılaşmak maksadıyla, herhangi bir maksat için tasarlanmış bir simulasyon ile başka bir simulasyon arasında bağlantı

kurulmasını mümkün kılar. HLA'ya uygun olarak tasarlanan simulasyonlar, nesne özelliklerinin mülkiyetini transfer edebilme kabiliyetine sahip bir şekilde dizayn edilerek, gelecekte mümkün olan geniş bir alanı kapsayan federasyonlar içerisinde bir federe olmak için temel yapısal araçlar temin ederler. Bir federenin, hem sahip olunabilen, hem de yansıtılabilen ve uygulama esnasında dinamik bir şekilde transfer edilebilen nesne veya özellikleri, o federenin SOM'u içerisinde belirtilir.

3.3.2.4. Federeler, kendi SOM'ları içerisinde de belirtildiği gibi, nesne özelliklerinin güncelleştirilmelerini sağlayan şartlar (örneğin eşikler-thresholds) değişebilecektir. (Kural 9)

HLA, federelerin, simulasyon içinde temsil edilen nesnelerin özelliklerine sahip olmalarına(yani;güncel değerler meydana getirebilmek için) izin verir ve daha sonra, RTI yoluyla bu değerleri diğer federeler için elde edilebilir hale getirir. Farklı federasyonlar, özelliklerin farklı şartlar altında güncelleştirileceğini belirtebilir(mesela; irtifa değeri üzerindeki değişiklik miktarı:1000 feet gibi bir eşik değerini aştığı zaman güncelle gibi). Geniş bir alanda kullanılabilen simulasyonlar, farklı federasyonların gereksinimlerini karşılamak maksadıyla, kendi özelliklerini ihraç etme şartlarını ayarlama yeteneğine sahip olacaklardır. Belirli bir federenin kendine özgü nesne özelliklerini güncellemesinin koşulları, o federenin kendi SOM'u içinde belgelenmiş olacaktır.

3.3.2.5. Federeler, bir bakıma federasyonun diğer üyeleri ile veri alışverişini düzenlemelerine izin verecek olan, yerel zamanı(local time) yönetebilecektir. (Kural 10)

HLA'nın zaman yönetim yapısı, farklı dahili zaman yönetim mekanizmalarından yararlanan federeler arasında karşılıklı çalışabilirliği sağlamak için tasarlanmıştır(bakınız Zaman Yönetimi konusu). HLA, her hizmeti gerçekleştirmek için kaçınılmaz olan gereksinimlere bağlı kalan federelere bu yeteneği sağlar. Bu hedefleri başarmak için, tek bir zaman yönetimini birleştirme yaklaşımı, tamamen farklı federeler arasındaki zaman yönetiminde karşılıklı çalışabilirlik(interoperability) sağlamak maksadıyla geliştirilmiş olacaktır. Farklı

simülasyon kategorileri, bu birleşik yapı içersindeki özel durumlar olarak değerlendirilir ve genellikle, RTI'ın bütün kapasitesinin sadece bir kısmını kullanırlar. Federeler, RTI'ın özel zaman akış mekanizmasına kendi içlerinde belirgin bir şekilde ihtiyaç duymaz, ancak diğer federeler ile birlikte karşılıklı veri alışverişinde koordinasyonu sağlayan RTI hizmetlerini(Zaman Yönetim hizmetleri) kullanacaktır. Bu kural bir federenin bir bakıma kendi yerel zamanını(local time) yönetmesi gerektiğini ifade eder. Bu, bir federasyonun gereksinimlerini karşılamak için RTI zaman yönetim servislerinden faydalanan bir federenin diğer federasyon üyeleri ile birlikte koordinasyonuna izin verir.

BÖLÜM 4. HLA NESNE MODEL KALIBI

4.1. Giriş

Bu bölüm, HLA Nesne Model Kalıbı(Object Model Template-OMT) tanımlamalarını ve bileşenlerini kısaca açıklamaktadır. HLA Nesne Model Kalıbı, nesneleri(objects), özellikleri(attributes), etkileşimleri(interactions) ve parametreleri içeren, HLA nesne modelleri bilgisini kaydetmek için sözdizimi(syntax) ve belirli bir format tavsiye eder. Ancak, nesne modelleri içinde bulunacak olan özel veriyi(örnek olarak; taşıtlar, eşya türleri vs.) tanımlamaz.[9]

Bilindiği gibi, bir yüksek seviyeli yapı olan HLA, M&S için ortak bir yapı sağlaması için geliştirilmiş olan birleşik bir yapıdır. Bu yapının amacı, simulasyonların etkileşimli bir şekilde karşılıklı çalışabilirliğini kolaylaştırıp, simülasyonların ve simülasyon bileşenlerinin yeniden kullanımının gelişmesine yardımcı olmaktır. Bu genel hedeflerin desteklenmesinde, HLA, özel federelerin ve federasyonların, federasyonun hedeflerini gerçeklemek için çalışma anında karşılıklı değişilen verinin kimliğini saptamaya yarayan bir nesne modeli tarafından tanımlanmış olmasını şart koşar. [9]

HLA OMT, simülasyon veya federasyon nesneleri sınıflarının, bu nesnelerin özellik sınıflarının ve etkileşim sınıfları hakkındaki bilgileri belgelemek için bir kalıp sağlar. Bu genel kalıp, farklı simulasyon ve federasyonların anlaşmasını ve birbirine benzetilmesini kolaylaştırır. Bu doküman, hem gerekli olan bilginin içerik türünü belirtir hem de, HLA nesne modellerinin içeriğini uyarlama konusunda bir format belirtir.

4.2. Nesne Model Kalıp Mantığı

HLA nesne modellerini tanımlamak için, standart hale getirilmiş bu yapısal çerçeve veya kalıp, aşağıdaki sebeplerden dolayı HLA'nın başlıca bileşenlerinden biri olmuştur:

- Bir federasyonun üyeleri arasında genel bir koordinasyon ve verinin karşılıklı değişiminin belirlenmesine yardımcı olan bir mekanizma sağlar.
- Potansiyel federasyon üyelerinin yeteneklerini tarif etmek için standart bir genel mekanizma sağlar.
- HLA nesne modellerinin gelişmesine yardımcı olacak bir takım araçların tasarımını ve uygulamasını kolaylaştır.

HLA nesne modelleri, bir HLA Simülasyon Nesne Modeli(Simulation Object Model SOM) oluşturularak, özel bir federasyon üyesini(federeyi) tanımlamak için kullanılabilir; veya bir Federasyon Nesne Modeli(Federation Object Model-FOM) oluşturularak, etkileşimli federeler kümesini(federasyon) tanımlamak için kullanılabilir. Diğer bir deyişle; HLA Nesne Model Kalıbının(HLA Object Model Template-OMT) başlıca hedefi, simulasyon bileşenlerinin yeniden kullanımını ve simulasyonların karşılıklı çalışabilirliğini (interoperability) kolaylaştırmaktır. Bu konuda anlatılan HLA nesne modelleri, başka türlü belirtilmedikçe, hem SOM hem de FOM'u kapsar.

4.3. Federasyon Nesne Modelleri(Federation Object Models-FOM's)

Bir HLA federasyonunun gelişmesi sırasında, bütün federasyon üyelerinin, federasyona katılan federeler arasında gereksinim duyulan tüm etkileşimlerin özelliklerine veya tabiatına uygun olarak ortak bir karşılıklı anlayış elde etmesi hassas bir konudur. HLA FOM'un başlıca hedefi, federeler arasında karşılıklı veri değişiminin standart hale getirilmiş genel bir formatta tanımlanmasını sağlamaktır. Bu verinin muhteviyatı, bir federasyonla ilgili bütün nesne ve etkileşim sınıflarının bir listesini(bu sınıfların tanımlanan özelliklerinin ve parametrelerinin ayrıntılı bir açıklamasıyla birlikte) kapsar. HLA FOM'un özel bileşenleri, federeler arasında

karşılıklı çalışabilirliği tesis etmek için gerekli(ancak yeterli değil) olan, "bilgi model kontratını(information model contract)" saptar.

4.4. Simülasyon Nesne Modelleri(Simulation Object Models-SOM's)

Bir federasyonun oluşumundaki hassas adımlardan biri, sponsorun bütün hedeflerini en iyi bir şekilde karşılamak için özel simulasyon sistemlerinin bileşiminin saptanması işlemidir. HLA SOM, özel bir simulasyonun, HLA federasyonlarına sağlanabildiği kendine özgü yeteneklerin bir tanımlamasıdır. SOM'ların ifade edildiği standart biçim, simulasyon sistemlerinin bir federasyona dahil olabilmesi için o simulasyon sisteminin uygunluğunun belirlemesini kolaylaştırır.

Burada tanımlamış olan HLA OMT formatları, genellikle başka FOM veya SOM'lara da uygulanabilir. Bu yüzden, SOM'lar nesneler, nesnelerin özellikleri ve etkileşimler bakımından nitelendirilmiştir. FOM'lar ve SOM'lar için OMT formatlarının ortak kullanımındaki başlıca fayda, HLA topluluğu içinde nesne modellerini tarif etmek için ortak bir referans çerçeve sağlamasıdır. Bazı durumlarda, bu genelleme(commonality), bir FOM içerisine "örnek parçalar(piece parts)" olarak dahil olan SOM bileşenlerine izin verebilir.

4.5. Nesneye yönelik(Object-Oriented - OO) nesne modelleri ilişkisi

HLA OMT, HLA nesne modelleri için standart hale gelmiş bir yapıdayken, FOM'lar ve SOM'lar, nesneye yönelik analiz ve dizayn(object-oriented analysis and design-OOAD) metodolojisi içindeki nesne modellerinin genel tanımlamalarına tam anlamıyla uygun değildir. OOAD literatürü içerisindeki bir nesne modeli, sistemi tam olarak anlamak maksadıyla geliştirilen bir sistemin soyutlanması olarak tanımlanmıştır. Bu anlayışı tesis etmek için, nesneye yönelik(object-oriented-OO) metodolojilerin çoğu, nesneler ve nesnelerin kendi transformasyonel(algoritmik) tanımları arasındaki bütün statik ve dinamik ilişkilerin tam tanımını içererek, "gerçek-dünya(real-world)" sisteminin görünümlerinin ayrı ayrı tanımlanmasını tavsiye eder.

HLA ve OOAD prensipleri ve kavramları arasındaki farklılıklar, özel nesne seviyesinde ortaya çıkar. OOAD literatüründe, nesneler, veriyi ve operasyonları bütünleştiren bir yazılım olarak tanımlanmıştır. HLA içinde nesneler, federeler içinde bulunan HLA nesne özellik değerlerini etkileyen davranışlar ve çalışmalar aracılığıyla, uygulama sırasında federeler arasında karşılıklı olarak değişilen özelliklerin(attributes) saptanması yoluyla tam bir şekilde tanımlanmış olur. HLA nesneleri, normalde gerçek dünya varlıklarını(mesela; tank, uçan bir cisim,...) tasvir etmek için tasarlanmış olurken, OOAD nesneleri, hem elle tutulur hem de kavramsal olabilir. farklı meseleler ve ilgiler yoluyla yönlendirilen sınıf-altsınıf ilişkilerini tanımlamak için hiyerarşik yapılar kullanıldığı halde, m*iras(inheritance)* kavramı, hem OOAD ve hem de HLA nesnelerine temel teşkil eder. HLA nesneleri etkileşimler veya özellik güncellemeleri aracılığıyla etkileşirken, OOAD nesneleri, bir nesnenin, bir başka nesne tarafından sağlanan bir çalışmayı veya servisi talep etmesi sırasında iletilen mesaj aracılığıyla etkileşirler. Ayrıca, OOAD nesneleri, nesnenin sınıf tanımına bağlı olan işlemler ile, güncelleme sorumlulukları arasında ilişkili kurarken, HLA nesne özelliklerinin güncellenmesinin mesuliyeti bir federasyon içindeki farklı federeler arasında dağıtılmıştır.

Paylaşılan terminoloji içinde ifade edilen terminolojideki değişikliklere ek olarak, diğer farklılıklar, OMT tablo tanımlarını baştanbaşa tanıtan yeni terimleri beraberinde getirir.

4.6. HLA OMT Bileşenleri

HLA nesne modelleri, nesne sınıfları, nesne sınıflarının özellikleri ve nesne sınıf etkileşimleri hakkındaki bilgileri içeren birbirleriyle ilişkili bir grup bileşenden oluşur. Bu HLA OMT bileşenlerinin içerdiği bilgileri bir çok farklı yolla göstermek mümkünken, HLA, bu bileşenlerin tablo formunda olmasını gerektirir. HLA nesne modelinin özünü oluşturan kalıp, tablo formatını kullanır ve aşağıdaki bileşenlerden oluşur:

- ***Nesne model kimlik tablosu****(object model identification table)*: Önemli tanıma bilgisi ile HLA nesne modelini birleştirmek için kullanılır.

- ***Nesne sınıf yapı tablosu****(object class structure table)*: Bütün simülasyon/federasyon nesne sınıflarının isimlerini kaydetmek için ve sınıf-altsınıf ilişkilerini tanımlamak için kullanılır.
- ***Etkileşim sınıf yapı tablosu****(interaction class structure table)*: Bütün simülasyon ve federasyon etkileşim sınıflarının isimlerini kaydetmek için ve sınıf-altsınıf ilişkilerini tanımlamak için kullanılır.
- ***Özellik tablosu****(attibute table)*: Bir simulasyon/federasyon içindeki nesne özelliklerinin belirgin yanlarını belirtmek için kullanılır.
- ***Parametre tablosu****(parameter table)*: Bir simulasyon/federasyon içindeki etkileşim parametrelerinin belirgin yanlarını belirtmek için kullanılır.
- ***Gönderme uzayı tablosu****(routing space table)*: bir federasyon içerisindeki nesne özellikleri ve etkileşimler hakkındaki gönderme uzaylarını belirtmek için kullanılır.
- ***FOM/SOM sözlüğü****(FOM/SOM lexicon)*: Tablolarda kullanılan bütün terimleri tanımlamak için kullanılır.

Hem özel simülasyonlar(federeler) hem de federasyonlar, bir HLA nesne modeli hazırlarken, OMT çekirdek bileşenlerinin yedisini de kullanacaksa da, bazı durumlarda, tabloların bir kısmının içi boş olabilir. Bununla birlikte, HLA nesne modellerinin tümü, en az bir tane nesne sınıfı veya bir adet etkileşim sınıfını içermelidir.

Federasyonlar genellikle kendi federelerinin bazı nesneleri arasındaki etkileşimleri destekleyeceklerken, bazı federeler(gizli bir olay izleyicisi gibi) etkileşimler ile meşgul olmayabilir, bu yüzden etkileşim sınıf yapı tablosu, bazı HLA nesne modelleri için boş olabilir. Eğer belirli bir nesne modeli içinde etkileşimler desteklenmemişse, parametre tablosunun içi boş olabilir. Umulur ki, federeler, genellikle federasyon karşısındaki çıkarları ile ilgili olan özellikler ile nesnelere sahip olacaktır; böyle durumlarda, bu nesneler ve özellikler, mutlaka belgelenmiş olacaktır. Bununla birlikte, nesne sınıf yapı tablosu ve özellik tablosu boş iken, bir federe veya bütün bir federasyon, sadece etkileşimler yoluyla karşılıklı bilgi değişimi yapabilir. Gönderme uzayı tablosu, SOM'lar için daima boş olabilir, ve eğer bir

federasyon içinde hiçbir veri dağıtımı yönetimi(DDM) servisi kullanılmazsa, bir FOM için de boş olabilir.

HLA OMT bileşenlerinin sonuncusu olan FOM/SOM sözlüğü, burada anlatılan HLA nesne modeli içinde kullanılan terimlerin anlamlarını açık ve kesin bir şekilde açıklamak için gereklidir. Bir HLA nesne modeli içinde daima en az bir tane terim bulunduğu için, bu sözlük içinde en az bir adet terim tanımlı olacaktır.

OMT tablolarından herhangi birine herhangi bir giriş, hazır tablo yapısının dışındaki ilave tanımlayıcı bilgi içeren notlar eklemek şeklinde olabilir. Bu "notların" en belirgin özelliği, verinin etkin kullanımını kolaylaştırmada gerekli olan özel tablo girişleri ile açıklayıcı bilgiyi birleştirme konusunda kullanıcılara izin vermesidir.

4.7. HLA OMT Örnekleri

4.7.1. Nesne model kimlik tablosu örneği

Tablo 4.1, basit bir nesne model kimlik tablosu örneğini göstermektedir. Bu örnekte, *Amaç* girişi, birçok pratik uygulama için bu tabloda bulunulan girişlerden daha kısadır.

4.7.2. Nesne sınıf yapı tablosu örneği

Tablo 4.2, basit bir sistemi temsil etmek için nesne sınıf yapı tablosunun(Object Class Structure Table) nasıl kullanıldığını anlatan bir örneği göstermektedir. Burada, temsil edilen tipik mahalli bir restoran sistemidir. Bu restoranın işletilmesinin simülasyonu örneği, bir federenin daha geniş ölçüdeki bir federasyonda(bir restoranlar zincirinin işletilmesini koordine edilmesinin ve birleştirilmesinin temsili gibi) bulunması ihtimali nedeniyle seçilmiş olabilir. Bu örneğin amacı, bu sistem için tam bir SOM' u belirtmek değil, sistem ile ilgili bilgileri tutmak için OMT tablolarının nasıl kullanıldığını göstermektir.

Nesne Model Kimlik Tablosu	
Kategori	**Açıklama**
İsim	Restoran SOM
Versiyon	1.0 Alpha
Tarih	1 Jan 1998
Amaç	Bir restoran federesi için bir nesne modeli örneğini sağlamak.
Uygulama alanı	Restoran işleri
Sponsor	Sodexho
POC(irtibat-point of contact)	Mr. Joseph Smith
POC organizasyonu	Joe'nun Yeri
POC telefonu	(977) 555-1234
POC e-mail	smithj@fedfoods.com

Tablo 4.1. Nesne model kimlik tablosu örneği

Bu örnekte, en üst düzeydeki beş nesne sınıfından meydana gelen tam bir nesne sınıf hiyerarşisinin bir altkümesi(subset) gösterilmiştir. Bu özel simülasyon için, sınıf ayrışması(decomposition), ilk üç sınıf için gerekli değildir. Dördüncü sınıf için, ilk ayrışma, beş yaprak sınıf ile sonuçlanmıştır. Beşinci sınıf için birkaç ayrışma düzeyi restoran menüsünün kısmi bir temsili ile gösterilmiştir. Bu hiyerarşideki daha derin seviyeler özellikler (attributes) gibi modellenmiştir (mesela; *Midye_Yahnisi(Clam_Chowder)*, *Manhattan* veya *New_England*'ın numaralandırılan değerlerini temsil etmek için bir özellik tipi ile, yaprak düğümü olabilmiştir.). Ancak, bu örnekteki modelleyici(modeler), özel sınıflar olarak çok özel yiyecek tiplerin temsilini tercih etmiştir.

4.7.3. Etkileşim sınıfı yapı tablosu örneği

Tablo 4.3, bir önceki bölümde bahsi geçen restoran örneği ile ilgili bir takım etkileşimleri göstermektedir. Burada, restorandaki işler, müşteriler ile restoranın personeli arasında cereyan eden bir takım etkileşimlere göre tanımlanmıştır. En üst düzeyde, restoran federesi, genel müşteri-personel işlemlerini başlatabilir. Bu,

örneğin bir restoran zincirinin yönetim çalışmalarını simüle eden federelerin, özel restoranlarda genel faaliyetleri takip etmelerini sağlar. Bu yüksek-derece etkileşim sınıfı daha sonra, restoranda gerçekleşen temel tipteki müşteri-personel etkileşimlerinin içerisine ayrıştırılır.

Nesne Sınıf Yapı Tablosu				
Müşteri(PS)				
Hesap(PS)				
Sipariş(PS)				
Personel(PS)	Karşılayıcı(PS)			
	Garson(PS)			
	Veznedar(PS)			
	Bulaşıkçı(PS)			
	Aşçı(PS)			
Yemek(S)	Ana_Servis(PS)			
	İçecek(S)	Su(PS)		
		Kahve(PS)		
		Gazoz(S)	Kola(PS)	
			Portakal(PS)	
			Kök_Bira(PS)	
	İştah_Açıcılar(S)	Çorba(S)	Midye_Yahnisi(S)	Manhattan(PS)
				New_England(PS)
			Sığıreti_Arpa(PS)	
	Baş_Yemek(S)	Sığır_Eti(PS)		
		Tavuk(PS)		
		Deniz_Ürünleri(S)	Balık(PS)	
			Karides(PS)	
			Istakoz(PS)	
		Makarna(PS)		
	Meze(S)	Mısır(PS)		
		Karalahana(PS)		
		Fırınlanmış_Patates(PS)		
	Tatlı(S)	Kek(PS)		
		Dondurma(S)	Çikolatalı(PS)	
			Vanilyalı(PS)	

Tablo 4.2. Nesne Sınıf Yapı Tablosu- SOM Örneği

İkinci düzeydeki bütün sınıflar, doğrudan federe tarafından başlatılabilirken, buna ilave olarak, *sipariş_alma(order_taken)*, *Yemek_Verme(food_served)* ve

ücret_ödeme(pay_bill), federasyonun ihtiyaçlarına bağlı olarak, restoran federesi tarafından da doğrudan başlatılabilen daha ayrıntılı sınıflar içerisine ayrıştırılır. Ayrıca, bu federe aynı zamanda, diğer restoranların kendi müşterilerine hizmet edebilme oranını takip etmek ve zincir içinde bulunan diğer restoranlardaki müşteri varış faaliyetini takip etmek için, *Musterinin_Oturması(Customer_Seated)* ve *müşterinin_ayrılması(Customer_Leaves)* sınıfı etkileşimlerini sezme(S ile gösterilir) yeteneğini gösterir.

Etkileşim Sınıf Yapı Tablosu		
Musteri_ Personel_ Islemleri(I)	Musterinin_Oturması(IS)	
	Siparis_Alma(I)	Cocuk_Menusunden_ Siparis_Alma(I)
		Yetiskin_Menusunden_ Siparis_Alma(I)
	Yemek_Verme(I)	Icecek_Verme(I)
		Cerez_Verme(I)
		Bas_Yemek_Verme(I)
		Tatli_Verme(I)
	Musterinin_Odemesi(I)	Kredi_Karti_ile_Odeme(I)
		Nakit_Odeme(I)
	Musterinin_Ayrilmasi(IS)	

Tablo 4.3. Etkileşim Sınıf Yapı Tablosu- SOM Örneği

Not: Bu bölümde fazlaca ayrıntıya girmemek maksadıyla OMT bileşenlerinin ayrıntılı açıklamasına yer verilmemiştir. Ancak OMT bileşenlerinin tablo yapısını anlamak çok önemlidir. OMT bileşenlerinin tam anlamıyla anlaşılması için bu tabloların ve örneklerin yapısının iyice anlaşılması gerekir. Bu nedenle, HLA Nesne Model Kalıbı(HLA Object Model Template) dokümanının tam olarak göz önünde tutulması şarttır. Bu doküman(omt1-3.pdf), http://hla.dmso.mil adresinden veya HLA RTI Primer eğitim CD'sinden elde edilebilir.

BÖLÜM 5. HLA ARABİRİM TANIMLAMALARI(INTERFACE SPECIFICATION)

5.1. Giriş

HLA, M&S için yaygın bir yapı sağlaması maksadıyla geliştirilen entegre bir yapıdır. HLA, federeler arasındaki etkileşimlerin standart bir API kullanılarak yapılmasını şart koşar. HLA Arabirim Tanımlamaları, bir federasyon uygulaması içinde hızlı ve verimli bir şekilde karşılıklı veri değişimini ve özel federelerin yeniden kullanımını sağlamak için federeler tarafından kullanılan standart servisleri ve arabirimleri tanımlar. Bu bölümde, HLA Arabirim Tanımlamalarının temel unsurlarının ayrıntılı bir açıklaması bulunmaktadır.[10]

Daha öncede belirtildiği gibi, nasıl ki bir işletim sistemi uygulamalar için bir takım servisler sağlıyorsa, aynı şekilde RTI'da federelere servisler sağlar. Bu arabirimler, altı adet temel RTI servis grubu içinde tertiplenmiştir:

a) Federasyon Yönetimi(federation management)
b) Deklarasyon Yönetimi(declaration management)
c) Nesne Yönetimi(object management)
d) Mülkiyet Yönetimi(ownership management)
e) Zaman Yönetimi(time management)
f) Veri Dağıtımı Yönetimi(data distribution management)

Bu altı servis grubu, RTI ve federeler arasındaki arabirimi ve HLA federeleri tarafından kullanılması için RTI tarafından sağlanan bilgisayar yazılımını tanımlar. Bu servislerin başlangıç seti, birden fazla federasyon karşısında en fazla gereksinim duyulan fonksiyonları sağlaması için dikkatle seçilmiştir. Sonuç olarak, federe

uygulamaları, bu bölümde tanımlanan servislerin çoğuna gereksinim duyacaklardır. RTI, federeden "RTI başlatmalı(RTI initiated)" olarak adlandırılan bir takım servisler talep eder(Her yönetim alanıyla ilgili servisler açıklanırken "†" sembolü ile gösterilmiştir.).

5.2. HLA Federasyon Nesne Model Çerçevesi

Nesne model çerçevesinin kısa ve öz tanımı, federeler ile RTI arasındaki arabirimin ve RTI hizmetlerinin tanımlanması için gereklidir. Bir federasyon nesne modelini(federation object model-FOM) tanımlamak için faydalanılan kurallar ve terminoloji, HLA Nesne Model Kalıbı(HLA OMT) konusu içerisinde tanımlanmıştır. Simulasyon Nesne Modeli(simulation object model-SOM), federenin yeniden kullanımı sağlamak ve federenin diğer faaliyetlerine destek olmak için bir federenin önemli özelliklerini tanımlar. Aslında, SOM, RTI ve onun servisleriyle ilgili değildir. Diğer taraftan FOM, federeler-arası konular ile uğraşır ve RTI'ın kullanımı ile ilgilidir. FOM'lar aşağıda belirtilen öğeler tarafından tarif edilir:

- Planlanmış bir federasyonda gerçek dünyayı tasvir etmek için seçilmiş olan nesne sınıfları kümesi,
- Gerçek dünya nesneleri arasındaki karşılıklı etkileşimi tasvir etmek için seçilmiş olan etkileşim sınıfları kümesi,
- Bu sınıfların özellikleri ve parametreleri ve,
- Gerçek dünyayı temsil eden bu sınıfların bütün özelliklerini içeren detaylar.

Her nesne, FOM içinde bulunan herhangi bir nesne sınıfının bir örneğidir[10]. Nesne sınıfları, arzu edilen örgütsel düzeni sağlamayı kolaylaştırmak için nesne model tasarımcısı tarafından seçilirler. Her nesne sınıfı, kendisiyle ilişkili bir takım özelliklere sahiptir. Bir *özellik(attribute)*, nesne durumunun belirgin ve tanımlanabilir bir parçasıdır. Bu konu içinde geçen, "özellik işaretçisi(attribute designator)" denilince özellik anlaşılır ve "özellik değeri(attribute value)" denilince de, onun içeriği anlaşılır. Meseleyi federasyon açısından ele alırsak, bir nesne örneğinin bütün özellik değerleri kümesi, o örneğin durumunu tam bir şekilde tanımlayacaktır. Federeler, ilave durum bilgisi ile federeler arasında açıklanmayan

bir nesne örneği arasında ilişki kurabilir, ancak bu HLA federasyon nesne modelinin alanının dışındadır.

Federeler, iletişimin başlıca araçlarından biri olarak nesne örneklerinin durumunu kullanır. Herhangi bir zamanda, sadece bir federe, belirli bir nesne örneğinin özelliğini simüle etmekten sorumludur. O federe, RTI servisleri aracılığıyla, federasyon uygulaması içindeki diğer federelere, o örnek özellik için yeni değerler sağlayacaktır. Federenin yeni örnek özellik değerleri sağlaması, o örnek özellik değerini *güncelleme(updating)* olarak adlandırılır. Federelerin bu değerleri alması da, o örnek özelliği *yansıtma(reflecting)* olarak ifade edilir.

Bir örnek özellik değerini güncelleme hakkı, bir federasyonun çalışması esnasında belirli bir zamanda sadece bir federe tarafından sahip olunabilir. Bir örnek özellik değerini güncelleme hakkına sahip olan federe, o örnek özelliğe sahiptir denir. RTI, nesne örneği özelliklerinin mülkiyetlerinin(ownership) federeler arasında karşılıklı değişimine izin veren bir takım servisler sağlar. Bir nesne örneğini kaydeden(register) federe, o nesne örneğinin privilegeToDeleteObject(nesne silme imtiyazı) örnek özelliğine otomatik bir şekilde sahip olur(bütün federeler, açık bir şekilde yayımladıkları bütün nesne sınıfları için privilegeToDeleteObject'i otomatik olarak yayınlayacaktır). RTI, federelere, diğer özellikler için olduğu gibi, "privilegeToDeleteObject" özelliğini de transfer etmeyi sağlayan servisler sağlayacaktır.

Her nesne örneği, bir işaretçiye(designator) sahiptir. Bir nesne örneği işaretçisinin değeri, her federasyon uygulaması için tektir. Nesne örneği işaretçileri, dinamik olarak RTI tarafından üretilir.

FOM çerçevesi, aynı zamanda, herbir nesne modeli için etkileşim sınıflarını hesaba katar. Olması muhtemel etkileşim türleri ve onların parametreleri, FOM içerisinde belirtilmiştir. Etkileşim, bir nesneden başka bir nesneye doğru isteğe bağlı olarak yöneltilebilen açık bir aksiyondur.[10]

Bir *federasyon*, belirli bir FOM'un, belirli bir federeler kümesinin ve RTI hizmetlerinin bir kombinasyonudur. Bir federasyon, genellikle bir federasyon nesne modeli ve bu nesne modeliyle kendi semantikleri arasında ilişki kurabilen bir federeler kümesi kullanılarak belirli biç amaç doğrultusunda dizayn edilir.

5.3. Genel Terminoloji ve Kabuller

Bu tez içinde geçen çeşitli varlık isimleri veya kavramlar(sınıflar, özellikler, parametreler, bölgeler, federeler, nesne örnekleri vs.), klasik anlamlarından daha farklı anlamlara sahip olabilir.

Aşağıdaki veri grupları, RTI'ın çalışmasını ve federasyon uygulamasını gerçekleştirmek için gereklidir:

- Federasyon uygulama verisi(federation execution data-FED) - FOM'dan türetilen(sınıf, özellik, parametre isimleri, vs.) ve çalışma anında RTI tarafından kullanılan bilgi. Bir federasyon uygulamasını oluşturma, bir FED'e bir federasyon uygulaması isminin basit bir şekilde yapıştırılmasıyla olur.

- RTI başlangıç verisi(RTI Initialization Data-RID) - RTI'ın çalıştırılması için ihtiyaç duyulan özel bilgi. RID, muhtemelen, RTI çalışmaya başladığı zaman sağlanacaktır.

5.4. Federasyon Yönetimi(Federation Management)

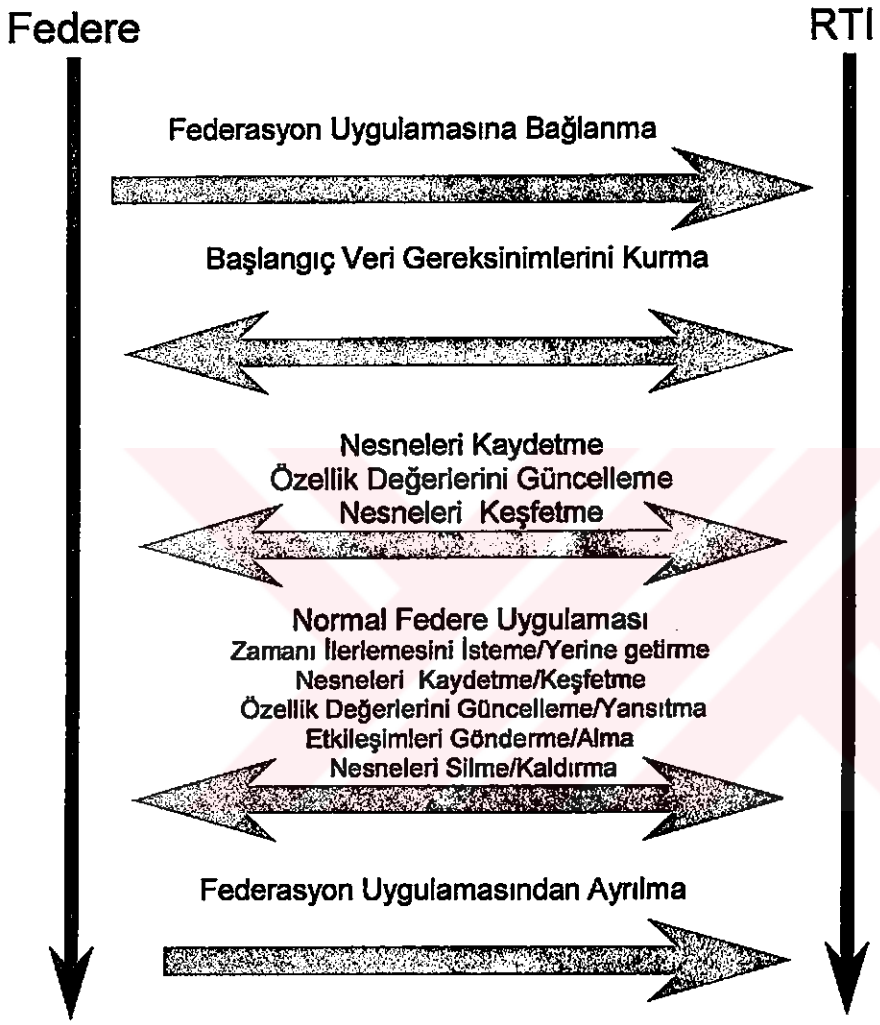
Federasyon Yönetimi, bir federasyon uygulamasının oluşturulması, dinamik kontrolü, üzerinde değişiklik yapma ve silme gibi bir takım aktiviteleri kapsar.

Şekil 5.1. Federasyon Uygulamasının Temel Durumları

Bir federe federasyon uygulanmasına bağlanamadan önce, federasyon uygulaması mutlaka mevcut olmalıdır. Şekil 5.1, temel federasyon yönetimi hizmetleri kullanılarak, bir federasyon uygulamasının tüm durumlarını göstermektedir.

Bir federasyon uygulaması meydana geldiği zaman, federeler federasyon kullanıcısı için anlamlı olan herhangi bir düzen dahilinde bu federasyona bağlanabilir ve federasyondan ayrılabilirler. Şekil 5.2'de, bir federenin federasyon uygulaması içerisine katılması sırasında, federe ile RTI arasındaki temel ilişkinin genelleştirilmiş

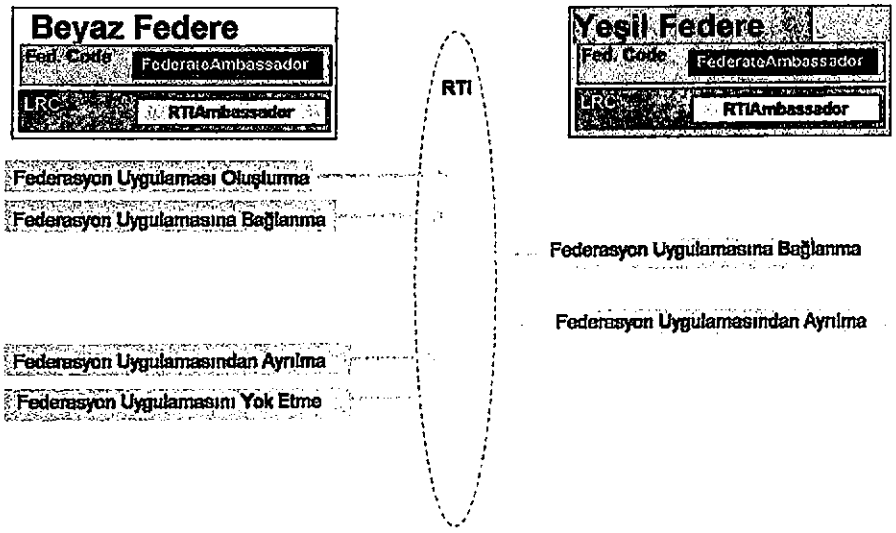
bir yapısı görülmektedir. Şekil 5.2'deki kalın oklar, RTI servis veya hizmet gruplarının talep edilmesini temsil eder. HLA kavramı, birden fazla federenin, ne bir tek federasyon uygulamasına katılmasından ötürü tek bir yazılım sistemine engeldir nede birden fazla federenin birden çok (bağımsız) federasyon uygulamasına katılması sebebiyle belirli bir sisteme engel olacaktır.



Şekil 5.2. Federe-RTI ilişkisinin genelleştirilmiş bir görünümü

5.4.1. Federasyon Yönetimi Servisleri

Şekil 5.3, federasyon yönetiminin çalışma süreci ile ilgili başlıca işlevleri göstermektedir.



Şekil 5.3. Federasyon Yönetimi Yaşam Döngüsü

5.4.1.1. Federasyon Uygulaması Oluşturma(Create Federation Execution)

Federasyon Uygulaması Oluşturma servisi, yeni bir federasyon uygulaması oluşturacak ve onu desteklenen federasyon uygulamaları grubuna katacaktır. Bu servis tarafından oluşturulan her bir federasyon uygulaması, diğer federasyon uygulamalarının tümünden bağımsız olacak ve RTI içerisindeki federasyon uygulamaları ile arasında karşılıklı hiç bir iletişim olmayacaktır. FED işaretçi argümanı, federasyon uygulamasını oluşturmak için gerekli olan FED'i belirtecektir.

5.4.1.2. Federasyon Uygulamasını Yok Etme(Destroy Federation Execution)

Federasyon Uygulamasını Yok Etme servisi, bir federasyon uygulamasını, RTI tarafından desteklenen federasyon uygulamaları grubundan kaldıracaktır. Federasyon faaliyeti tamamıyla duracağı için bu servisi talep etmeden önce bütün federeler, federasyondan ayrılmış olmalıdır.

5.4.1.3. Federasyon Uygulamasına Bağlanma(Join Federation Execution)

Federasyon Uygulamasına Bağlanma servisi, bir federeyi bir federasyon uygulamasına bağlayacaktır. Bu servisin talep edilmesi, talep eden federenin belli bir federasyona katılma niyetini gösterecektir.

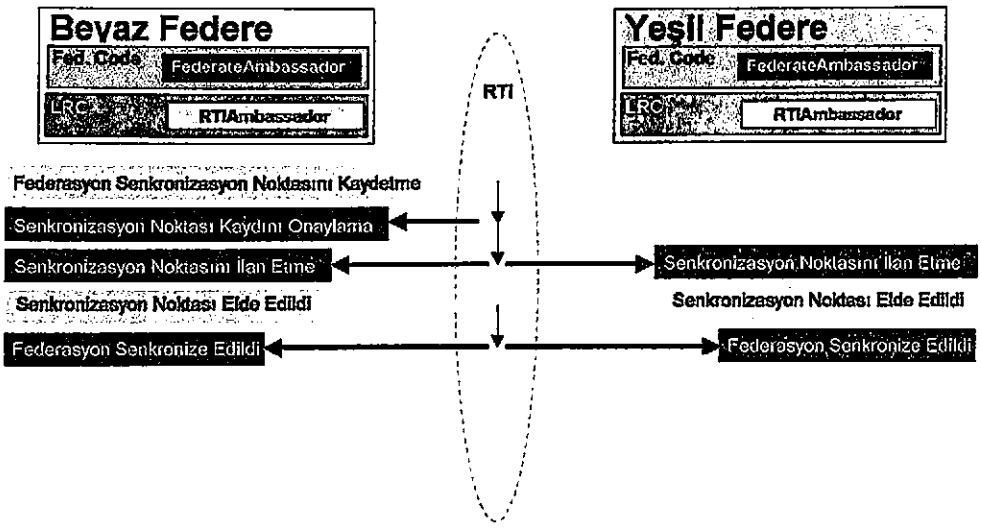
5.4.1.4. Federasyon Uygulamasından Ayrılma(Resign Federation Execution)

Federasyon Uygulamasından Ayrılma servisi, federasyon bağlantısının kesilmesinin istendiğini gösterecektir. Ayrılmadan önce, federeler tarafından tutulan örnek özelliklerin mülkiyeti bırakılmalıdır. Federe, bu örnek özelliklerinin mülkiyetini diğer federelere transfer edebilir, daha sonraki bir zamanda mülkiyet tekrar elde etmek için onları serbest bırakabilir, veya onların bir parçası olan nesne örneğini silebilir(federenin bu nesne örneklerini silme imtiyazına sahip olduğu varsayılarak). Federeye kolaylık olarak, *Federasyon Uygulamasından Ayrılma* servisi, aşağıdaki aksiyonların birini veya daha fazlasını gerçekleştirmek için RTI'a yöneltilen bir aksiyon argümanını barındıracaktır. Bu aksiyonlar şunlar olabilir:

- Gelecekte tekrar mülkiyetini elde etmek için sahip olunan örnek özellikleri serbest bırakma. Bu, örnek özellikleri bir sahipsiz(unowned) duruma(bu onların değerlerinin güncelleştirilmeyeceği anlamına gelmektedir) getirecektir, ki bu, bu örnek özelliklerini, bir başka federe tarafından mülkiyeti elde edilebilir(eligible) yapacaktır.
- İmtiyaza sahip olan federenin nesne örneklerinin tümünü silmesi(*Nesne Örneği Silme* Servisinin talep edilmesi gerekmektedir).

5.4.1.5. Federasyon Senkronizasyon Noktasını Kaydetme(Register Federation Synchronization Point)

Federasyon yönetimi, bir federasyona katılan federeler arasındaki aktiviteleri senkronize etme konusunda bir takım servisler içerir. Bilindiği gibi RTI, bilginin, federeler arasında sağlıklı bir şekilde alınıp verilmesi için uygun mekanizmalar sağlar. Karşılıklı değişilen bilgi ve zaman arasında ilişki kurma suretiyle federe faaliyetlerini koordine etmek mümkündür. Federasyon yönetimi senkronizasyon servisleri, federelerin senkronizasyon noktalarını açıklamalarına izin verir. Şekil 5.4'te federasyon yönetimi senkronizasyon servisleri görülmektedir.



Şekil 5.4. Federasyon Yönetimi Senkronizasyonu

Federasyon Senkronizasyon Noktasını Kaydetme servisi, senkronizasyon noktası etiketinin kaydını başlatmak için kullanılacaktır. Senkronizasyon noktası etiketi başarılı bir şekilde kaydedildiği zaman, RTI, *Senkronizasyon Noktasını İlan Etme†* servisi ile etiketin varlığından birkaç federeyi veya bütün federeleri haberdar edecektir(Şekil 5.4).

5.4.1.6. Senkronizasyon Noktası Kaydını Onaylama †(Confirm Syncronization Point Registration †)

Senkronizasyon Noktası Kaydını Onaylama† servisi, daha önceden talep edilmiş olan bir federasyon senkronizasyon noktasını kaydetme isteğinin durumunu federeye gösterecektir. RTI, *Federasyon Senkronizasyon Noktasını Kaydetme* servisine karşılık olarak bu servise başvuracaktır. Bu servisle birlikte sağlanan bir pozitif başarı göstergesi argümanı, etiketin başarılı biçimde kaydedildiği federeye bildirecektir. Negatif başarı göstergesi ise, etiketin zaten kullanımda olduğunu veya bu etiketin kaydının başarısız olduğunu federeye bildirecektir. Negatif başarı göstergesiyle sonuçlanan herhangi bir kayıt denemesi, federasyon uygulaması üzerinde olumsuz veya bozucu bir etkiye sahip olmayacaktır.

5.4.1.7. Senkronizasyon Noktasını İlan Etme † (Announce Synchronization Point †)

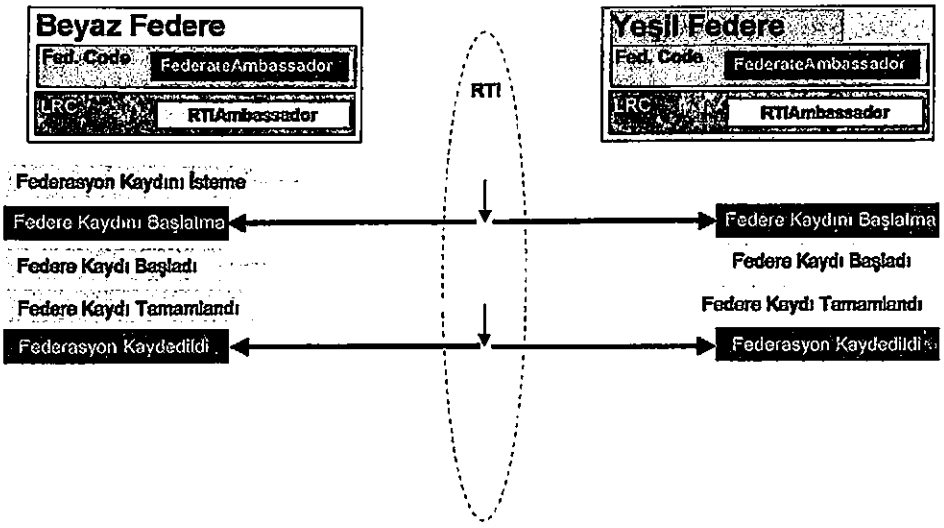
Senkronizasyon Noktasını İlan Etme† servisi, yeni bir senkronizasyon noktası etiketinin varlığından bir federeyi haberdar edecektir. Bir senkronizasyon noktası etiketi, *Federasyon Senkronizasyon Noktasını Kaydetme* servisi yoluyla kaydedildiği zaman, RTI, etiketin varlığından onları haberdar etmek maksadıyla, belirli bir federe grubu üzerinde, veya uygulama içerisindeki bütün federeler üzerinde, *Senkronizasyon Noktasını İlan Etme†* servisine başvuracaktır. *Senkronizasyon Noktasını İlan Etme†* servisi vasıtasıyla belirli bir senkronizasyon noktasının varlığından haberdar olan federeler, o nokta için bir senkronizasyon seti oluşturacaklardır. Bir senkronizasyon noktasının ilan edilmesinden sonra, ancak federasyonun o nokta üzerinde senkronize olmasından önce, federasyon uygulamasından ayrılan federeler, senkronizasyon setinden kaldırılmış olmalıdırlar.

5.4.1.8. Senkronizasyon Noktası Elde Edildi(Synchronization Point Achieved)

Senkronizasyon Noktası Elde Edildi servisi, federenin belirtilen senkronizasyon noktasına ulaştığından RTI'ı haberdar edecektir. Belirli bir noktanın senkronizasyon seti içindeki bütün federeler, bu servise başvurduğunda, RTI, yeni bağlanan federeler üzerinde *Senkronizasyon Noktasını İlan Etme†* servisine başvurmayacaktır.

5.4.1.9. Federasyon Senkronize Edildi †(Federation Syncronized †)

Federasyon Senkronize Edildi† servisi, belirli bir senkronizasyon noktası için *Senkronizasyon Noktası Elde Edildi* servisine başvuran ve o noktanın senkronizasyon seti içerisinde bulunan bütün federeleri federeye bildirecektir. Yani bu servis, belirli bir nokta üzerinde senkronize olmuş olan federeleri göstererek, o noktanın senkronizasyon seti içerisindeki bütün federeler üzerinde başvurulacaktır. Bir noktanın senkronizasyon seti, senkronize edildiğinde(set içerisindeki bütün federeler üzerinde *Federasyon Senkronize Edildi†* servisine başvurulduğu zaman), o nokta, artık daha fazla kaydedilmeyecektir ve o noktanın senkronizasyon seti mevcut olmayacaktır.



Şekil 5.5. Federasyon Kaydı Etkileşim Diyagramı

5.4.1.10. Federasyon Kaydını İsteme(Request Federation Save)

Federasyon Kaydını İsteme servisi, bir federasyon kaydının meydana gelmesi gerektiğini RTI'a belirtecektir. RTI, bir federe üzerinde *Federe Kaydını Başlatma†* servisine başvurarak, durum kaydı yapmasını o federeye bildirecektir(Şekil 5.5). Herhangi bir zamanda sadece bir adet kayıt isteği, değerlendirmeye alınacaktır. Yeni bir kayıt isteği, herhangi bir kayıt isteğinin yerini alacaktır. Bununla birlikte, bir kayıt isteği, bir kayıt işlemi devam ederken yapılamaz.

5.4.1.11. Federe Kaydını Başlatma †(Initiate Federate Save †)

Federe Kaydını Başlatma† servisi, durumunu kaydetmesi için federeye talimatta bulunacaktır. Federe, *Federe Kaydını Başlatma†* servis talebinin ardından en kısa sürede kayıta başlamalıdır. *Federasyon Kayıt İsteği* servisi yoluyla kayıt yapma istendiği zaman, bu servis yoluyla RTI'a ulaştırılan etiket, federeye de sağlanmış olacaktır. Federe, *Federasyon Uygulamasına Bağlanma* servisini talep ettiğinde sağlamış olduğu, federasyon uygulamasının ismini, kendi federe işaretçisini, kendi federe tipini ve bu etiketi, kaydedilmiş olan durum bilgisini ayırt etmek veya tanımak için kullanacaktır. Federe, *Federe Kaydını Başlatma†* servisi başvurusunu aldıktan sonra, derhal federasyona yeni bilgi sağlamayı kesecektir. Federe, sadece

Federasyon Kaydedildi† servis başvurusunu aldıktan sonra, federasyona yeni bilgi sağlamaya yeniden başlayabilir.

5.4.1.12. Federe Kaydı Başladı(Federate Save Begun)

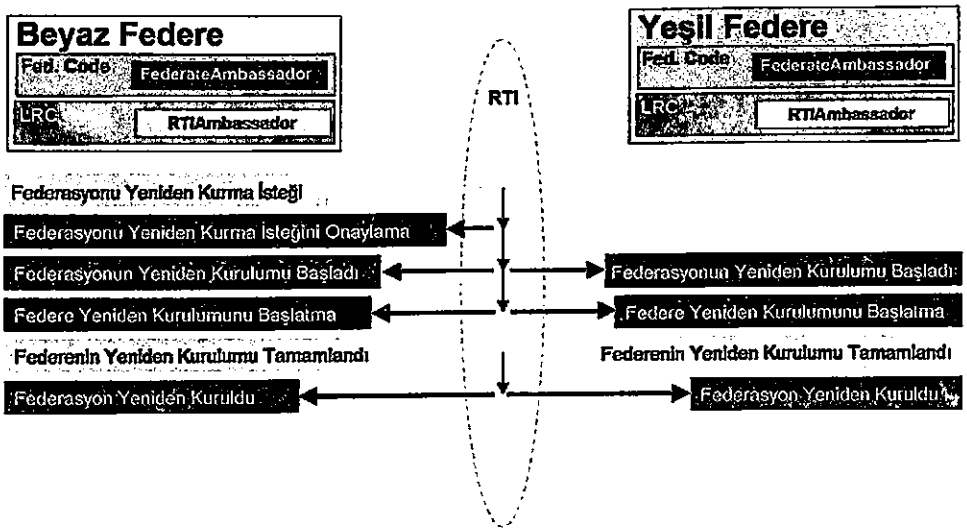
Federe Kaydı Başladı servisi, federenin kendi durumunu kaydetmeye başladığını RTI'a bildirecektir.

5.4.1.13. Federe Kaydı Tamamlandı(Federate Save Complete)

Federe Kaydı Tamamlandı servisi, federenin kendi kayıt işlemini tamamladığını RTI'a bildirecektir. Bu servis le birlikte RTI'a sağlanmış olan kayıt-başarı göstergesi argümanı, federenin kaydı başarıp başaramadığını RTI'a bildirecektir.

5.4.1.14. Federasyon Kaydedildi †(Federation Saved †)

Federasyon Kaydedildi† servisi, federasyonun kaydedilme işleminin tamamlandığını federeye bildirecektir ve kaydın başarılı bir şekilde mi yoksa başarısız bir şekilde mi tamamlandığını federeye bir argüman(federasyon kayıt-başarı göstergesi) vasıtasıyla gösterecektir. Kayıt-başarı göstergesi argümanı, eğer başarıyı gösteriyorsa, bu, *Federe Kaydını Başlatma†* servis başvurusuna maruz kalan bütün federelerin, başarıyı gösteren bir kayıt-başarı göstergeli *Federe Kaydı Tamamlandı* servisine başvurduğunu göstermektedir. Eğer kayıt-başarı göstergesi argümanı, başarısızlığı gösteriyorsa, bu, *Federe Kaydını Başlatma†* servisine maruz kalmış olan bir veya daha fazla federenin, başarısızlığı gösteren bir kayıt-başarı göstergeli *Federe Kaydı Tamamlandı* servisine başvurduğunu göstermektedir veya RTI bu federelerin birinde veya daha fazlasında bir eksiklik bulmuş demektir. *Federe Kaydını Başlatma†* servisi başvurusunu alan bütün federeler, *Federasyon Kaydedildi†* servisi başvurusunu da RTI'dan alacaklardır. Eğer RTI'dan *Federe Kaydını Başlatma†* servis başvurunu alan bir federe, daha *Federasyon Kaydedildi†* servisini almadan önce federasyon uygulamasından ayrılırsa, bu federasyon kaydında bir hataya neden olacaktır ve o federe, bir başarısızlığı gösteren bir kayıt-başarı göstergeli *Federasyon Kaydedildi†* servisini alacaktır.



Şekil 5.6. Federasyonu Yeniden Kurma İşlemi

5.4.1.15. Federasyonu Yeniden Kurma İsteği(Request Federation Restore)

Federasyonu Yeniden Kurma İsteği servisi, federasyon uygulamasının yeniden kurulması(restore) işlemini başlamak için RTI'a yöneltilecektir. Federasyonun yeniden kurulum işlemi, muhtemelen, *Federasyonu Yeniden Kurma İsteği* servisi başvurusunun onaylamasından sonra hemen başlayacaktır. Federasyonun yeniden kurulum isteğinin onaylanması, *Federasyonu Yeniden Kurma İsteğini Onaylama†* servisi ile gösterilecektir(Şekil 5.6).

5.4.1.16. Federasyonu Yeniden Kurma İsteğini Onaylama †(Confirm Federation Restoration Request †)

Federasyonu Yeniden Kurma İsteğini Onaylama† servisi, daha önceden talep edilmiş olan federasyon yeniden kurulum işleminin statüsünü federeye belirtecektir. Bir pozitif istek başarı göstergesi(request success indicator), RTI yeniden kurma durum bilgisini federeye bildirir. Bir federeden daha fazla federe, aynı anda federasyonu yeniden kurmaya kalkarsa, sadece bir federe pozitif, diğerleri negatif yanıt alacaktır. Negatif istek başarı göstergesiyle sonuçlanan bir federasyon yeniden kurulum girişimi, federasyon uygulaması üzerinde herhangi bir olumsuz etkiye sahip olmayacaktır.

5.4.1.17. Federasyonun Yeniden Kurulumu Başladı † (Federation Restore Begun †)

Federasyon Yeniden Kurulumu Başladı† servisi, federasyonun yeniden kurulması işleminin yakında gerçekleşeceğini federeye bildirir. Federe, *Federasyonun Yeniden Kurulumu Başladı†* servis başvurusunu aldıktan sonra derhal federasyona yeni bilgi sağlamayı durduracaktır. Federe, sadece *Federasyon Yeniden Kurulduf* servis başvurusunu aldıktan sonra federasyona yeni bilgi sağlayabilir.

5.4.1.18. Federe Yeniden Kurulumunu Başlatma †(Initiate Federate Restore †)

Federe Yeniden Kurulumunu Başlatma† servisi, daha önceden kaydedilmiş olan durumunu bildirmesi için federeye talimat verir. Federe, o anki federasyon uygulamasının ismini, bir federasyon kayıt etiketini ve federe işaretçisini içeren uygun yeniden kurma durum bilgisini seçecektir. Bu servisi talep etmenin sonucu olarak, *Federasyon Uygulamasına Bağlanma* servisi tarafından sağlanmış olan federe işaretçisinin değeri değişebilir.

5.4.1.19. Federenin Yeniden Kurulumu Tamamlandı(Federate Restore Complete)

Federenin Yeniden Kurulumu Tamamlandı servisi, federenin kendi yeniden kurma işleminin sonuçlandığını RTI'a bildirecektir.

5.4.1.20. Federasyon Yeniden Kuruldu †(Federation Restored †)

Federasyon Yeniden Kurulduf servisi, federasyon yeniden kurulum işleminin tamamlandığını federeye bildirecek ve başarılı mı yoksa başarısız bir şekilde mi tamamlandığını belirtecektir. Eğer, yeniden kurulum-başarı göstergesi(restore-success indicator) argümanı başarıyı gösteriyorsa, bu şu demektir: *Federasyon Yeniden Kurulumu Başladı†* servis başvurusuna maruz kalan bütün federeler, başarıyı gösteren bir yeniden kurma-başarı göstergeli *Federenin Yeniden Kurulumu Tamamlandı* servisine başvurmuştur. Eğer, yeniden kurma-başarı göstergesi

argümanı başarısızlığı gösteriyorsa, bu, bir veya daha fazla federe, başarısızlığı gösteren bir yeniden kurma-başarı göstergesi ile birlikte *Federenin Yeniden Kurulumu Tamamlandı* servisine başvurmuştur demektir veya, RTI bu federelerin biri veya birkaçı üzerinde bir hata yakalamış demektir. *Federasyonun Yeniden Kurulumu Başladı†* servisi cevabını alan bütün federeler, *Federasyon Yeniden Kuruldu†* servisi cevabını da alacaktır. Eğer, *Federasyonun Yeniden Kurulumu Başladı†* servisi cevabını alan bir federe, *Federasyon Yeniden Kuruldu†* servisi cevabını almazdan önce federasyon uygulamasından ayrılırsa, bu ayrılış, federasyonun yeniden kurulum işleminde bir başarısızlık olarak algılanacaktır, ve *Federasyon Yeniden Kuruldu†* servisi, başarısızlık içeren bir yeniden kurma-başarı göstergesi ile talep edilmiş olacaktır.

5.5. Deklarasyon Yönetimi(Declaration Management)

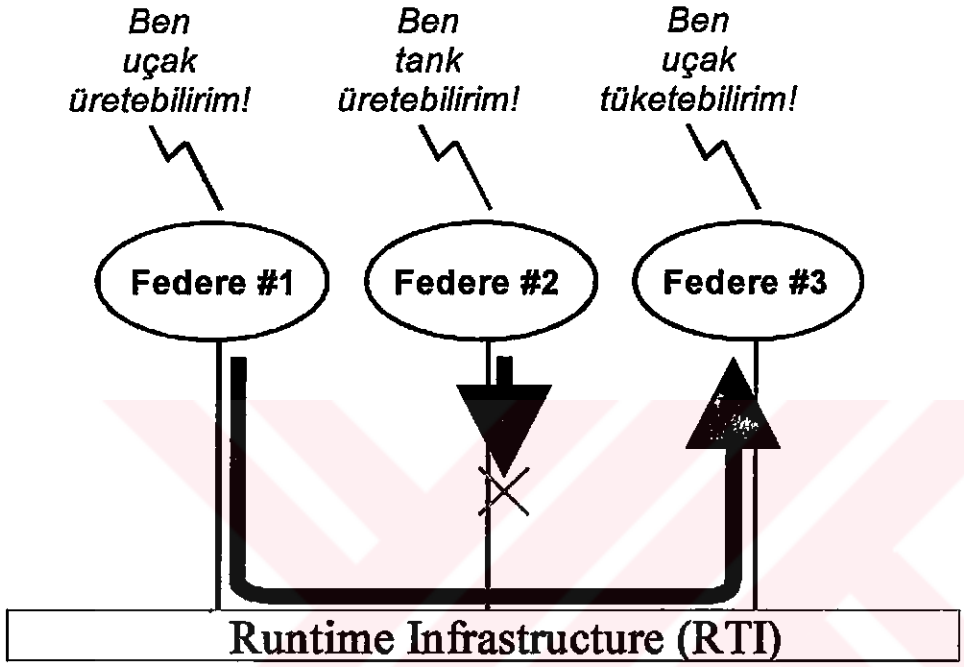
Federeler, bilgi üretme konusundaki niyetlerini ilan etmek, açığa vurmak veya deklare etmek için deklarasyon yönetimi servislerini kullanacaklardır. Bir federe, nesne örneklerini kaydetmeden, örnek özellik değerlerini güncellemeden ve etkileşimleri göndermeden önce, uygun deklarasyon yönetimi servislerine başvuracaklardır. Federeler, bilgi alma konusundaki istek ve niyetlerini ilan etmek için deklarasyon yönetimi servislerini veya veri dağıtımı yönetimi servislerini kullanacaklardır. Bir federe, nesne örneklerini keşfetmeden, örnek özellik değerlerini yansıtmadan ve etkileşimleri almadan önce, uygun deklarasyon yönetimi veya veri dağıtımı yönetimi servislerine başvuracaktır. Federasyon Uygulama Verisi(Federation Execution Data-FED) içinde tanımlı olan nesne ve etkileşim sınıf hiyerarşileri;

- Nesne örnekleri kaydedilebilen nesne sınıflarını,
- Nesne örnekleri keşfedilebilen nesne sınıflarını,
- Güncelleştirilmiş ve yansıtılmış olarak elde edilebilen örnek özelliklerini,
- Gönderilebilen etkileşimleri,
- Alınan etkileşimlerdeki etkileşim sınıflarını, ve
- Gönderilmek ve alınmak suretiyle elde edilebilen parametreleri belirler.

Deklarasyon yönetimi servislerinin yaptığı etkiler, federasyon zamanından bağımsız olacaktır.

Deklarasyon yönetimi, yayımlamayı(publication), abone olma(subscription) ve bir takım yardımcı kontrol fonksiyonlarını kapsar. Nesneleri(veya nesne parçalarını) veya etkileşimleri üreten federeler, neyi yayınlayabileceklerini(veya üretebileceklerini) eksiksiz bir şekilde belirtmelidirler. Bunun yanında nesneleri(veya nesne parçalarını) veya etkileşimleri tüketen(veya kullanan) federeler de, kendi tüketim ilgilerini veya yayımlanan bu nesnelere ve etkileşimlere abone olma(subscription) ilgilerini deklare etmelidirler.

RTI, katılımcı federelerin neleri üretebildiklerini ve tüketim taleplerinin ne olduğunu takip eder ve tüketici talebine dayanan üretimleri kontrol altında tutmak için kontrol sinyallerini kullanır. Şekil 5.7'de görüldüğü gibi, RTI, kontrol sinyallerini kullanarak, neyi iletmeleri gerektiğini üreticilere bildirmektedir. Buradaki hedef, komünikasyon ağları üzerindeki trafiği kontrol etmek ve bu trafiği komünikasyon ağlarını en az meşgul edecek şekilde ayarlamaktır.



Şekil 5.7. Kontrol Sinyal Şeması

5.5.1. Nesne Terminolojisine Bakış

Bu konuyla alakalı temel HLA terminolojisini anlamak bu konunun tam bir şekilde anlaşılması için çok önemlidir. Burada bu terminolojiden bazıları kısaca açıklanmıştır(nesne sınıfları, etkileşim sınıfları vs.).

Nesne sınıfları(object classes), özelliklerden oluşur. Nesne sınıfları, sürekli(*persist*) şeylerin tiplerini(*types*) tanımlar. Örneğin, "tank", bir nesne sınıfı olabilir. "Tank" tipi nesneler, belirli özelliklere sahip olur(mesela, boyut, ağırlık ve menzil gibi). Hakiki, gerçek tanklar, tank nesne sınıfının birer örnekleridirler(*instances*). Tek başına "Nesne" terimi, bazen özel bir nesne sınıfının bir örneğini tanımlamak için kullanılır.

Etkileşim sınıfları(interaction classes) ise, parametrelerden oluşur. Etkileşim sınıfları, olayların *tiplerini(types)* tarif eder. Etkileşim örnekleri, özel olaylardır. Açıkça söylemek gerekirse, nesnelerin özelliklerden oluşması ve etkileşimlerin de parametrelerden oluşması nedeniyle nesneler ve etkileşimler bu bağlamda birbirine benzer. HLA, doğasında olan bu simetrinin farkındadır ve yeri geldiği zaman bunu bir sonuca ulaşmak için kullanır. Nesneler ile etkileşimler arasındaki en temel fark, sürekliliktir(*persistence*). Nesneler süreklidir, ancak etkileşimler sürekli değildir.

Bir füze, bir nesne sınıfı tarafından mı, yoksa bir etkileşim sınıfı tarafından mı tanımlanabilir? Bunun cevabı, füze örneklerinin sürekliliğine ve simülasyonun türüne göre değişir. Füze fırlatma rampaları ve hedefleri üzerine odaklanmış bir simülasyon, füzeleri(veya füze rampalarını) olaylar gibi algılayabilir. Fırlatma rampası bir füzeyi ateşler ve bu bazı hasarlara neden olur. Füze havadayken, simülasyona göre önemsiz olabilir. Buradaki füze, muhtemelen fırlatma rampası ile hedef arasında olan bir etkileşim olarak modellenmiş olabilir. Bir başka simulasyon, füzelerin uçuş anındaki(in-flight) karakteristikleri üzerine odaklanabilir. Dolayısıyla burada füzeyi fırlatma veya hedefi vurma ikinci derece bir öneme sahiptir. Burada füze, süreklidir ve bir nesne olarak modellenmiş olması gerekir.

5.5.2. Nesne Hiyerarşileri

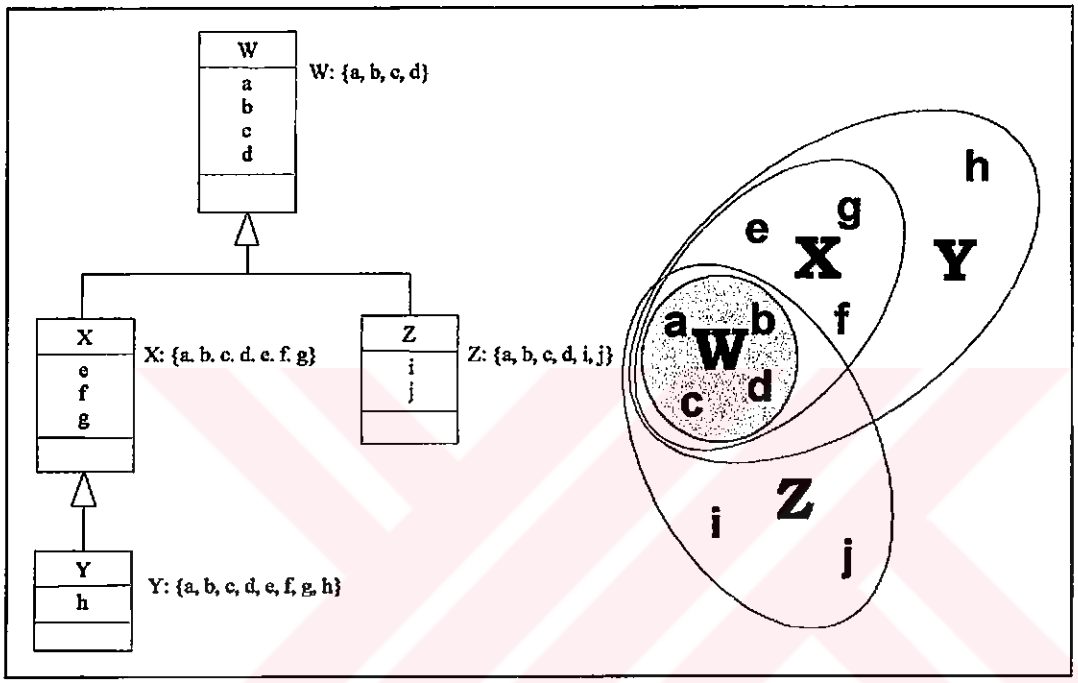
Şekil 5.8'de, bir sınıf hiyerarşisi yapısı ve ona eşlik eden bir Venn diyagramı görülmektedir. Nesne sınıfları ve etkileşim sınıfları hiyerarşik bir yapıda tasarlanabilir. Mesela, varsayalım ki, W tipindeki nesneler, "a", "b", "c" ve "d" özelliklerinden oluşmuş olsun-kısaltılmış olarak "{a,b,c,d}". W nesne sınıfını genişleterek yeni nesne sınıfları tanımlamak mümkündür. W nesne sınıfı, X ve Z nesne sınıflarını üretmek maksadıyla genişletilmiştir. Bunun yanında X nesne sınıfı da, Y nesne sınıfını üretmek için, bir kademe daha genişletilmiştir.

Nesneye-yönelik(object-oriented-OO) programlama meraklıları, bu gibi hiyerarşik temsilleri daha iyi anlayacaklardır. Çeşitli topluluklar, nesne hiyerarşilerini tarif

etmek ve tanımlamak için farklı tabirler kullanırlar. Aşağıda bunlara birkaç örnek verilmiştir:

X, W'den genişletilmiştir.
W, bir temel türdür.
X, W'den türetilmiştir.
Y, W'nin soyundan gelmiştir.
W, Z'nin ebeveynidir.

Y, X'den miras alır.
W, Z'nin atasıdır.
X, W'nin bir çocuğudur.
Y ve Z, yaprak nesnelerdir.



Şekil 5.8. Sınıf Hiyerarşisi ve Venn Diyagramı

Temel görüş odur ki; bir nesne sınıfı, yeni bir nesne sınıfı üretmek için genişletildiği zaman, oluşan bu yeni nesne sınıfı, genişletilmiş olan sınıfın bütün özelliklerini(büyük bir ihtimalle daha fazlasını) kapsar. Şekil 5.8'deki nesne hiyerarşisi ve Venn diyagramı, W, X, Y ve Z nesne sınıfları arasındaki ilişkileri göstermektedir. W nesne sınıfı, {a, b, c, d}olan dört özelliğe sahiptir, X sınıfı buna, {e, f, g}özelliklerini ekler, ve böylelikle X sınıfı örnekleri, {a, b, c, d, e, f, g} özelliklerine sahip olur.

5.5.3. Nesneleri Yayımlama(Publishing) ve Abone Olma(Subscribing)

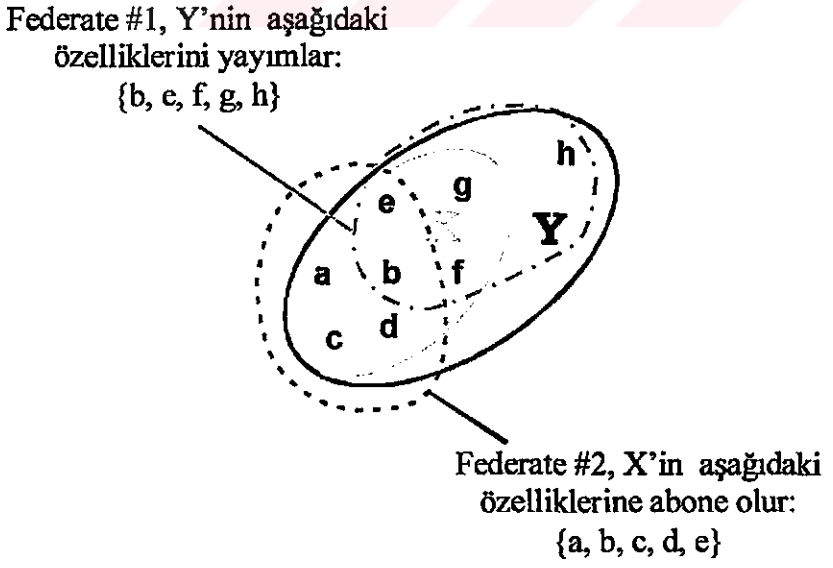
Her federe, üretmeyi planladığı nesne sınıflarını ve etkileşim sınıflarını *yayımlamalıdır(publish)*. Buradaki yayımlama kelimesinden maksat nesne

sınıflarının veya etkileşim sınıflarının herkese ilan edilmesi veya açıklanmasıdır. Bir federenin, belirli bir sınıfın mevcut özelliklerinin sadece bir altkümesini yayınlaması mümkündür.

5.5.3.1. Nesne Yayımlama(Object Publication)

Y nesne sınıfı, {a, b, c, d, e, f, g, h} özelliklerini kapsar. Bir federe, Y ile ilgili özelliklerin tümünü belirtmeksizin, Y nesne sınıfının örneklerini oluşturabilir. Mesela, Y herhangi bir uçak türü olsun. Belirli bir federe, uçak örnekleri hakkındaki bilginin bir bölümünü bilebilir(mesela, pozisyon bilgisi gibi) ancak, eksik kısımları tamamlama konusunda diğer federelere güvenir(mesela, uçak hakkındaki bilgi gibi). Bu durumda, federe, Y ile ilgili olarak hangi özellikleri yayımlayabileceğini belirtmelidir. Şekil 5.9'da, Federe #1, Y nesne sınıfının {b, e, f, g, h} özelliklerini yayımlayabileceğini göstermektedir.

Her federe, her bir sınıf için üretebileceği(tanıtacağı veya güncelleyebileceği) özellikleri tam bir şekilde belirtmek zorundadır. Birden fazla federe, Y örneklerini yayımlayabilir. Federe #3, Y nesne sınıf ile ilgili özelliklerin tümünü yayımlayabilir. Federe #7, Y'nin {a, c, f} özelliklerini yayımlayabilir.



Şekil 5.9. Nesne Yayımlama

5.5.3.2. Etkileşim Yayımlama(Interaction Publication)

Her federe, *Etkileşim Sınıfını Yayımlama* servisini kullanarak, nesne sınıfları ile birlikte, üretmeyi planladığı etkileşim sınıflarını tam olarak beyan etmelidir. Etkileşimler, ya bir bütün olarak ortaya konurlar yada hiç yapılmazlar. Eğer federe, bir etkileşimi yayımlama niyetinde olduğunu ifade ederse, o federenin, etkileşimle ilgili olan bütün parametreleri belirtebilme yeteneğinde olması gerekir.

5.5.3.3. Nesneye Abone Olma(Object Subscription)

Federeler, *Nesne Sınıf Özelliklerine Abone Olma(Subscribe Object Class Attributes)* servisi yoluyla, belirli nesne sınıflarına olan ilgilerini belirtirler. Nesne aboneliği, nesne yayımlama konusundan temelde farklılık gösterir. Bir federe, bir nesne sınıfına abone olduğu zaman, o sınıfın bütün nesne örnekleri hakkındaki bilgiye olan ilgisini ifade ediyor demektir. Mesela, bir federe X nesne sınıfına abone olarak, federasyondaki diğer federeler tarafından üretilen bütün X sınıfı örneklerini de keşfedecektir(discover). Bunu biraz daha açmak gerekirse, bir federe X'e abone olarak, sanki X sınıfının örnekleriymiş gibi (diğer federeler tarafından üretilen) Y sınıfının da bütün örneklerini keşfedecektir. Bu durum, *tip terfisine(type promotion)* bir örnektir.

Ne zaman bir federe, belirli bir nesne sınıfına olan abone olma ilgisini ifade ederse, RTI, federenin o nesne sınıfının torun sınıflarının örneklerine de ilgi duyduğunu varsayar. W sınıfına abone olan bir federe; X, Y ve Z sınıflarının örneklerini W sınıfının örnekleri olarak görür. Bu bazı bakımlardan faydalı olabilir. Diyelim ki W sınıfı, tüm uçakları temsil etsin. Y sınıfı, ticari uçakları temsil ederken, X sınıfı da, askeri uçakları temsil edebilir. Bir federe, bütün uçaklar hakkında bilgi edinmek isteyebilir ancak, askeri ve ticari gösterimleri içeren detayları önemsemeyecektir.

Eğer bir federe, bir nesne örneğinin nesne sınıfına abone olursa veya o örnek, abone olunan bir nesne sınıfına terfi ettirilirse(hiyerarşi içinde yukarıya doğru), federe, bu iki durumda da yeni bir nesne örneğinden haberdar edilir. Bir nesne, terfi ettirildiği(promote) zaman, orjinal sınıfa özgü olan özellikleri bırakır. Y nesne

sınıfının bir örneği, {a, b, c, d, e, f, g, h} özelliklerine sahiptir. Bir federe X nesne sınıfına abone olarak, Y örneğini bir X olarak keşfedebilir. "h" özelliği X sınıfının örnekleri içerisinde temsil edilmediği için, o bilgi kaybedilmiş olur.

Bir federe, bir sınıf hiyerarşisi içinde birden fazla sınıfa abone olabilir. Eğer bir federe, W ve X sınıflarına abone olduysa, aşağıdakiler doğru olur:

- W nesne sınıfının örnekleri, terfi olmaksızın görülürler.
- X nesne sınıfının örnekleri, terfi olmaksızın görülürler.
- Y nesne sınıfının örnekleri, X nesne sınıfının örnekleri olarak görülürler.
- Z Nesne sınıfını örnekleri, W nesne sınıfının örnekleri olarak görülürler.

Bir federe bir nesneyi keşfedince, örnek nesne sınıfını öğrenir. Eğer bir federe X nesne sınıfının bir nesne örneğini keşfederse, nesnenin tipini her zaman X olarak zannedecektir. Eğer bir federe, Y sınıfına değil de X sınıfına abone olursa, Y örneklerini, X örnekleri olarak keşfedecektir. Eğer federe daha sonradan Y sınıfına abone olursa, önceden X örnekleri olarak keşfedilmiş olan nesne örnekleri, X örnekleri olarak görülmeye devam edecektir(terfi yoluyla). Sonradan keşfedilen Y nesne sınıfı örnekleri, Y nesne sınıfının örnekleri olarak keşfedilmiş olacaktır.

5.5.3.4. Etkileşime Abone Olma(Interaction Subscription)

Her federe, nesne sınıflarında olduğu gibi, almayı istediği etkileşim sınıflarına abone olur. Bir etkileşim sınıfına ait olan özel parametrelere abone olmak mümkün değildir. Daha öncede belirtildiği gibi, etkileşimler, ya bir bütün olarak ortaya konurlar yada hiç yapılmazlar. Nesne sınıflarında olduğu gibi, eğer bir federe, (a) bir örneğinin etkileşim sınıfına abone olursa veya (b) o örnek, abone olunan bir etkileşim sınıfına terfi ettirilirse(hiyerarşi içinde yukarıya doğru), federe bu iki durumda da yeni bir etkileşim örneğinden haberdar edilir. Bir etkileşim örneği terfi ettirildiği zaman, sadece abone olunan sınıfın parametreleri, alıcı federeye sunulmuş olur.

5.5.3.5. Kontrol Sinyalleri

Yukarıdaki Şekil 5.9'da, Federe #1, Y örneklerini üretebilme yeteneğinde olduğunu göstermektedir, ancak, sadece {b, e, f, g, h} özelliklerini sağlayabilmektedir. Aynı şekil üzerinde, Federe #2, X nesne sınıfının {a, b, c, d, e} özelliklerine abone olur. Federe #1 tarafından üretilen Y özellikleri, Federe #2 tarafından X örnekleri olarak keşfedilmiş olur.

Federe #2, Federe #1 tarafından üretilmiş olan Y özelliklerinin sadece birkaçına ilgi göstermektedir. Önceden de ele alındığı gibi, Federe #2, "h" özelliğine ulaşamaz(bu özellik, X sınıfının bir parçası olmadığı için). Üstelik, Federe #2'nin, {f, g}özelliklerine hiç ilgisi yoktur. Federe #1'in üretebildiği Y:{b, e, f, g, h} bilgisinden, sadece Y:{b, e} bilgisine gereksinim duyulur(Federe #2'nin federasyon içerisinde diğer federe olduğu varsayılarak).

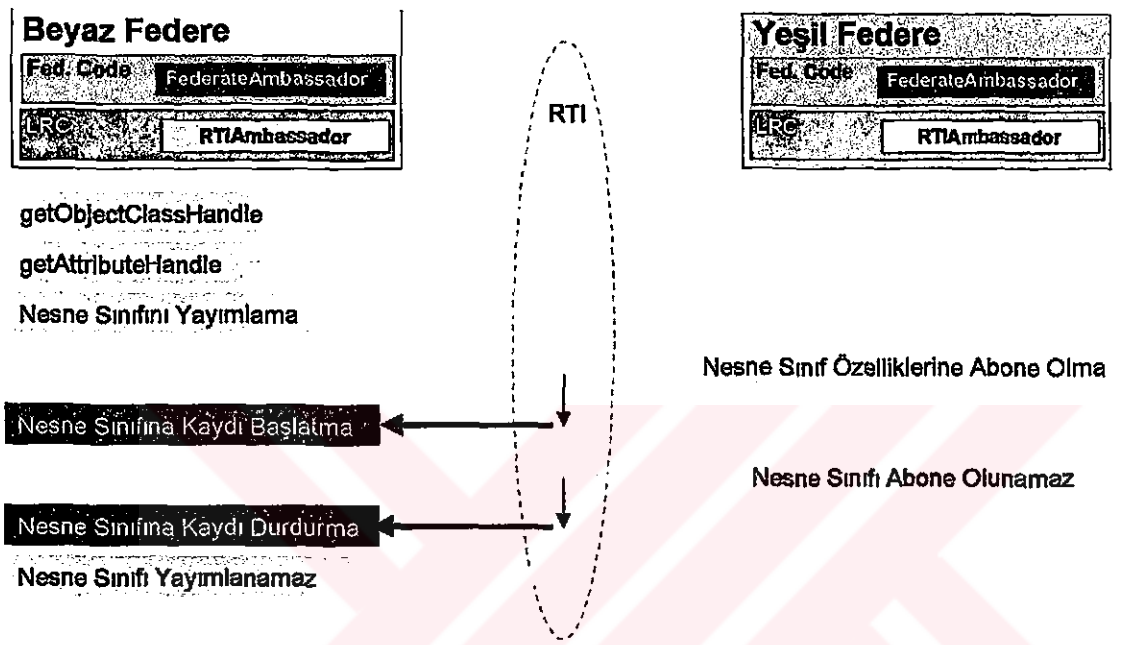
RTI, Federe #1'in ürettiği bilgiyi göstermek için kontrol sinyalleri yayımlar. Yerel RTI Bileşeni(Local RTI Component-LRC), bir tüketicinin var olduğunu göstermedikçe, federe nesne güncellemelerini üretmekten sakınmalıdır. Eğer Federe #1 sahnede en önceliğe sahipse(mesela hiç tüketicinin olmaması durumu), RTI, Y örnek bilgisini kaydetmeyi başlatmak için sinyal üretmeyecektir.

Federe #2 ortaya çıkınca, LRC, Y nesne sınıfının her bir örneğini kaydetmesi ve Y:{b, e} güncellemelerini sağlamaya başlaması gerektiğini Federe #1'e işaret edecektir. Eğer Federe #2 bu esnada ayrılırsa, Federe #1, Y nesne sınıfının örneklerinin kaydını ve Y:{b, e} güncellemelerinin sağlanmasını durduracaktır.

Her LRC, kendi federesini tüketici talebine dayanan nesne özelliklerinin ve etkileşimlerin üretimini başlatma ve durdurma konusunda bilgilendirir. Her federenin Simülasyon Nesne Modeli(Simulation Object Model-SOM), federenin LRC tarafından sağlanan kontrol sinyallerini kullanma yada kullanmama ölçüsünü tanımlayacaktır.

5.5.4. Deklarasyon Yönetimi Servisleri

Şekil 5.10'da görülen etkileşim diyagramı, nesne yayımlama ve abone olma metotları kullanılarak, istenilen bilgiyi oluşturma için gerekli olan işlemleri göstermektedir.



Şekil 5.10. Nesne Yayımlama ve Abone Olma

5.5.4.1. Nesne Sınıfını Yayımlama(Publish Object Class)

Nesne Sınıfını Yayımlama servisi yoluyla federe tarafından nakledilen bilgi, birden fazla yolla kullanılacaktır. Bu servis ilk olarak, federenin daha sonradan nesne örneklerini kaydedebileceği bir nesne sınıfını gösterecektir. İkinci olarak, nesne sınıfının sınıf özelliklerini belirtecektir ki, federe o sınıfın nesne örneklerinin uygun örnek özelliklerini edinebilme yeteneğindedir. Sadece örnek özelliği edinen federe, federasyona o örnek özellik için yeni değerler sağlayacaktır. Federe bir örnek özelliğin sahibi olmak suretiyle onun değerini güncelleme konusunda ehliyetli konuma gelir. Federe örnek özelliği güncelleme işlemini iki yolla yapabilir:

1. Yayımlanmış bir sınıfın nesne örneğinin kaydedilmesi yoluyla. Federe kaydedilen nesne örneği sınıfındaki uygun sınıf özellikleri yayımladığı için, bir nesne örneği kaydedildikten sonra kaydeden federe, o nesne örneğinin bütün örnek özelliklerinin sahibi olacaktır.

2. Nesne örneklerinin örnek özellikleri elde etmek için mülkiyet yönetimi servislerinin kullanılması yoluyla. Federe bilinen nesne örneği sınıfındaki uygun sınıf özelliklerini yayımladığı için, federe sadece nesne örneklerinin bu örnek özelliklerini elde edebilir.

Bu servisin her kullanımı, aynı nesne sınıfı için daha önceki servis taleplerinde RTI'a belirtilen bütün bilgiyi yenileyecektir. Aynı nesne sınıfı için daha önceki servis taleplerinde bulunmasının yanısıra aynı zamanda bu servis talebinde de bulunan bir sınıf özelliği, belirtilen nesne sınıfının yayımlı bir özelliği olarak devam edecektir. Yine aynı nesne sınıfı için bu servis talebinde bulunan ancak önceki servis talebinde gözükmeyen bir sınıf özelliği, belirtilen sınıfın yayımlı bir özelliği olarak meydana gelecektir. Son olarak aynı nesne sınıfı için bu servis talebinde bulunmayan ancak önceki servis talebinde bulunan bir sınıf özelliği, belirtilen sınıfın yayımlı bir özelliği olarak kalacaktır.

5.5.4.2. Nesne Sınıfı Yayımlanamaz(Unpublish Object Class)

Nesne Sınıfı Yayımlanamaz servisi, federenin belirtilen nesne sınıfının nesne örneklerini daha fazla kaydedemeyeceğinden RTI'ı haberdar edecektir. Federe, sahip olduğu nesne örneklerinin tüm örnek özelliklerinin mülkiyetini kaybedecektir. Bu demektir ki, federe, belirtilen nesne sınıfı olarak bilinen sınıfın nesne örneklerinin hiçbir örnek özellik değerlerini daha fazla güncellemeyecektir.

5.5.4.3. Etkileşim Sınıfını Yayımlama(Publish Interaction Class)

Etkileşim Sınıfını Yayımlama servisi, federenin federasyon uygulamasına göndereceği etkileşim sınıflarını RTI'a bildirecektir.

5.5.4.4. Etkileşim Sınıfı Yayımlanamaz(Unpublish Interaction Class)

Etkileşim Sınıfı Yayımlanamaz servisi, federenin belirtilen sınıfın etkileşimlerini daha fazla gönderemeyeceğini RTI'a bildirecektir.

5.5.4.5. Nesne Sınıf Özelliklerine Abone Olma(Subscribe Object Class Attributes)

Nesne Sınıf Özelliklerine Abone Olma servisi, nesne örneklerinin keşfinden federeyi haberdar edecek olan RTI'a bir nesne sınıfını belirtecektir. Bir nesne sınıfına abone olarak federe, aynı zamanda, bir takım sınıf özellikleri sağlayabilir. Bu servise başvurmanın bir sonucu olarak keşfedilen bütün nesne örnekleri için, sadece belirtilen sınıf özelliklerine uyan örnek özelliklerinin değerleri, RTI sayesinde federeye sağlanmış olacaktır(*Özellik Değerlerini Yansıtma*ı servisi yoluyla). Sağlanmış olan bir takım sınıf özellikleri, belirtilen nesne sınıfının elde edilebilir özelliklerinin bir alt kümesi olacaktır.

Abone olmanın bir sonucu olarak bir federe bir nesneyi sadece bir sınıfın varlığı olarak keşfedecektir.

Bu servisin her kullanımı, aynı nesne sınıfı için herhangi bir daha önceki *Nesne Sınıf Özelliklerine Abone olma* servis başvurusunda RTI'a belirtilen bütün bilginin yerini yenisiyle değiştirecektir.

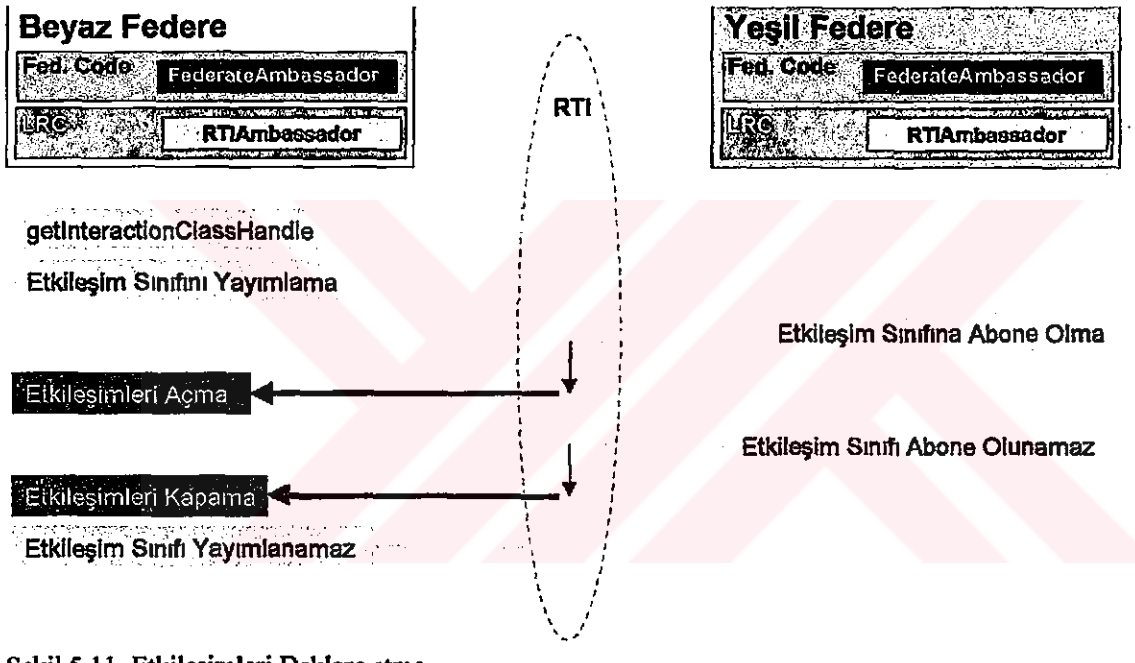
5.5.4.6. Nesne Sınıfı Abone Olunamaz(Unsubscribe Object Class)

Nesne Sınıfı Abone Olunamaz servisi, belirtilen nesne sınıfında nesne örnek keşfinin durdurulmasını isteyen federeden RTI'ı haberdar edecektir. Belirtilen nesne sınıfı olarak bilinen sınıfın bilinen nesne örneklerinin kapsam dahilindeki örnek özelliklerin tümü, kapsam dışına çıkacaktır.

5.5.4.7. Etkileşim Sınıfına Abone Olma(Subscribe Interaction Class)

RTI'ın federe nezdinde *Etkileşim Almaf* servisine başvurması yoluyla gönderilen etkileşimlerden federeyi haberdar etmesi gereken etkileşim sınıfını belirtir.

Bir etkileşim, bir federe tarafından alındığında, alınan etkileşim sınıfı, eğer abone olunmuşsa, etkileşimin gönderilen sınıfı olacaktır. Aksi taktirde, alınan sınıf, etkileşimin alındığı zamanda abone olunan gönderilmiş sınıfın en yakın süper-sınıfı olur. Parametreler sadece onun süper-sınıflarından ve etkileşimin alınan sınıfından alınılacaktır.



Şekil 5.11. Etkileşimleri Deklare etme

Eğer bir federe, bir etkileşim sınıfı miras ağacında birden çok yere abone olursa, gönderilen ilgili her etkileşim, abone olan federede en çok bir etkileşimin alınması ile sonuçlanacaktır.

5.5.4.8. Etkileşim Sınıfı Abone Olunamaz(Unsubscribe Interaction Class)

Etkileşim Sınıfı Abone Olunamaz servisi, federenin, belirtilen etkileşim sınıfının etkileşimlerinin daha fazla gönderilmeyeceğini RTI'a bildirecektir.

5.5.4.9. Nesne Sınıfına Kaydı Başlatma †(Start Registration For Object Class †)

Nesne Sınıfına Kaydı Başlatma† servisi, federenin yayımladığı nesne sınıfındaki sınıf özelliklerinin en az birine federasyon uygulaması içindeki bir başka federe tarafından aktif bir şekilde abone olunduğu için, belirtilen nesne sınıfının yeni nesne örneklerinin kaydını bildiren federeyi haberdar edecektir. Federe, belirtilen sınıfın nesne örneklerinin kaydı ile başlamalıdır.

5.5.4.10. Nesne Sınıfına Kaydı Durdurma † (Stop Registration For Object Class †)

Nesne Sınıfına Kaydı Durdurma† servisi, federenin yayımladığı nesne sınıfındaki sınıf özelliklerinin hiçbirine federasyon uygulaması içindeki bir başka federe tarafından aktif bir şekilde abone olunmadığı için, belirtilen nesne sınıfının yeni nesne örneklerinin kaydını bildirmemesini federeye haber verecektir. Federe, belirtilen sınıfın yeni nesne örneklerinin kaydını durdurmalıdır.

5.5.4.11. Etkileşimleri Açma †(Turn Interaction On †)

Etkileşimleri Açma† servisi, belirtilen sınıfa veya onun bir süper-sınıfına federasyon uygulamasında ez az bir tane federe tarafından aktif bir şekilde abone olunduğu için ilgili etkileşim sınıfını federeye bildirecektir.

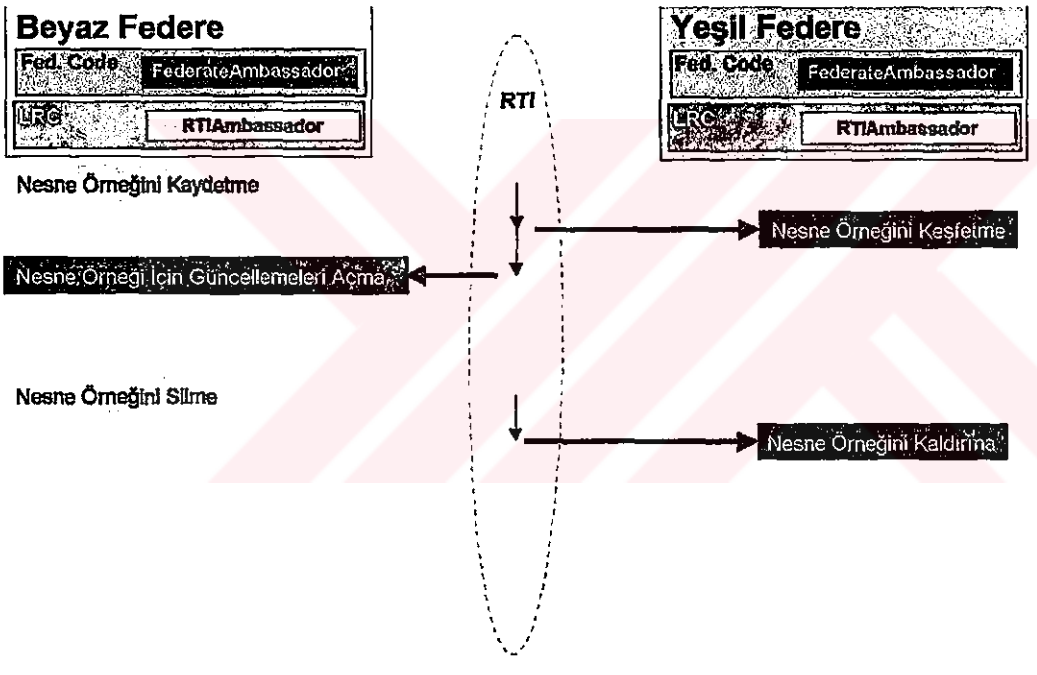
5.5.4.12. Etkileşimleri Kapama †(Turn Interaction Off †)

Etkileşimleri Kapama† servisi, belirtilen sınıfa veya onun bir süper-sınıfına federasyon uygulamasında ez az bir tane federe tarafından aktif bir şekilde abone olunmadığı için etkileşim sınıfının konuyla ilgili olmadığını federeye bildirecektir.

5.6. Nesne Yönetimi(Object Management)

RTI servislerinin bu gurubu, nesne örneklerini kaydetme(registration), üzerinde değişiklik yapma(modification) ve silme(deletion); etkileşimleri gönderme ve alma konuları ile ilişkilidir. Nesne yönetimi, nesne üretimi kısmında örnek *kaydetme(registration)* ve örnek *güncellemeleri(updates)* işlemlerini, nesne tüketim kısmında ise örnek *keşfi(discovery)* ve *yansıtma(reflection)* işlemlerini kapsar. Nesne yönetimi aynı zamanda, etkileşimleri gönderme ve alma, tüketici talebine dayanan örnek güncellemelerini kontrol etme gibi bir takım metotları da içerir.

5.6.1. Nesne Örneklerini Kaydetme, Keşfetme Ve Silme



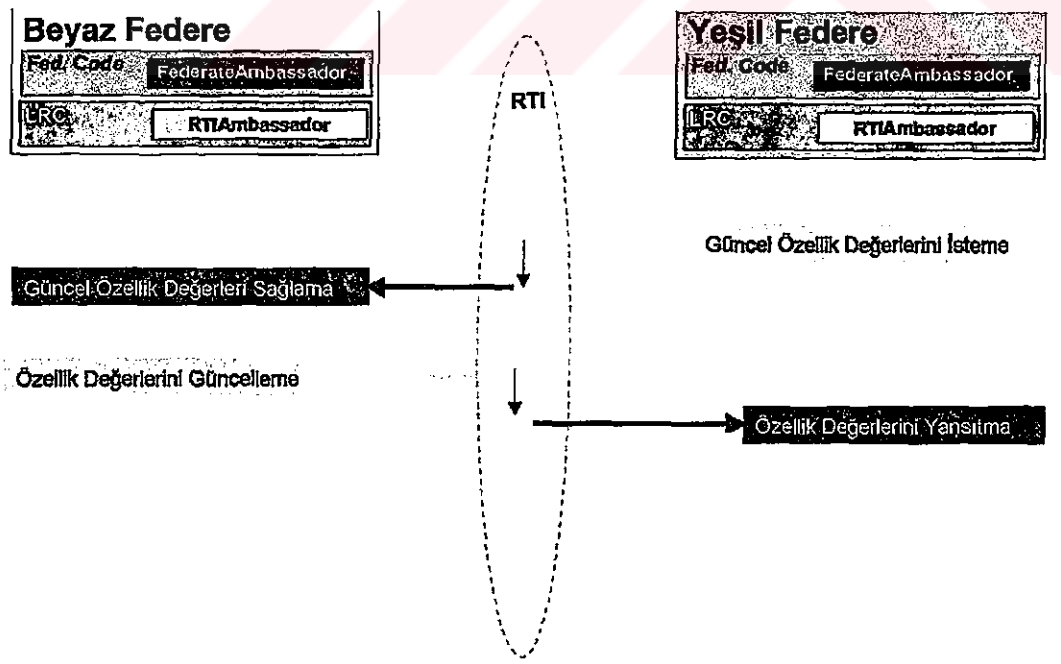
Şekil 5.12. Nesne örneklerini kaydetme, keşfetme ve silme

Şekil 5.12'de, nesne örneklerini kaydetmek ve keşfetmek için gerekli olan etkileşimler gösterilmektedir. *Nesne Örneğini Kaydetme* servisi, yeni bir nesne örneğinin vücut bulduğunu RTI'a bildirir. Kaydetme işlemi, bir nesne örneğini federasyona tanıtır. Ancak ilgili örnekle alakalı özellik değerlerini sağlamaz. Bu ikinci bir adım gerektirir. Her nesne, bir federe tarafından tamamen silinip ortadan kaldırılabilir. Bir nesneyi oluşturan(kaydeden) federe, başlangıçta o nesneyi silme imtiyazına sahip olur. Şekil 5.12'de kayıtlı bir nesneyi kaldırmak için *Nesne*

Örneğini Silme servisine başvurulmuştur. RTI tarafından ilgili federe için başvurulan *Nesne Örneğini Kaldırma†* servisi ise, federeye daha önceden kaydetmiş olduğu nesnenin artık var olmadığını haber verir.

Bir federe, bir nesne örneğini kaydetmek için *Nesne Örneğini Kaydetme* servisini kullandığı zaman yada bir nesne örneğini keşfetmek için *Nesne Örneğini Keşfetme†* çağrısını aldığı zaman, o nesne örneği federe için bilinen(known) hale gelir ve nesne örneği, o federede bilinen bir sınıfa sahip olur. Eğer bir federe, bir nesne örneğini kaydetmenin sonucu olarak onun hakkında bilgiye sahip olursa, o nesne örneğinin bilinen sınıfı onun kayıtlı sınıfı olur. Eğer bir federe, bir nesne örneğini keşfetmenin sonucu olarak bir nesne örneği hakkında bilgiye sahip olursa, nesne örneğinin bilinen sınıfı onun keşfedilen sınıfı olur.

Bir federe, *Nesne Örneğini Keşfetme†* servisi aracılığıyla bir nesne örneğini keşfeder. *Nesne Örneğini Keşfetme†* servisi talep edildiği zaman, bu servis talebi için bir argüman olan sınıf, nesne örneğinin *keşfedilen sınıfı* olarak adlandırılacaktır. Keşif anında, keşfedilen sınıf, aday keşif sınıfı ile aynı olacaktır. Keşiften sonra, keşfedilen sınıf değişemez. Ancak aday keşif sınıfı değişebilir. Nesne örneği bilinen(known) olarak kaldığı sürece, onun aday keşif sınıfı ilgiden mahrum olur.



Şekil 5.13. Nesne Özelliklerini Güncelleme ve Yansıtma

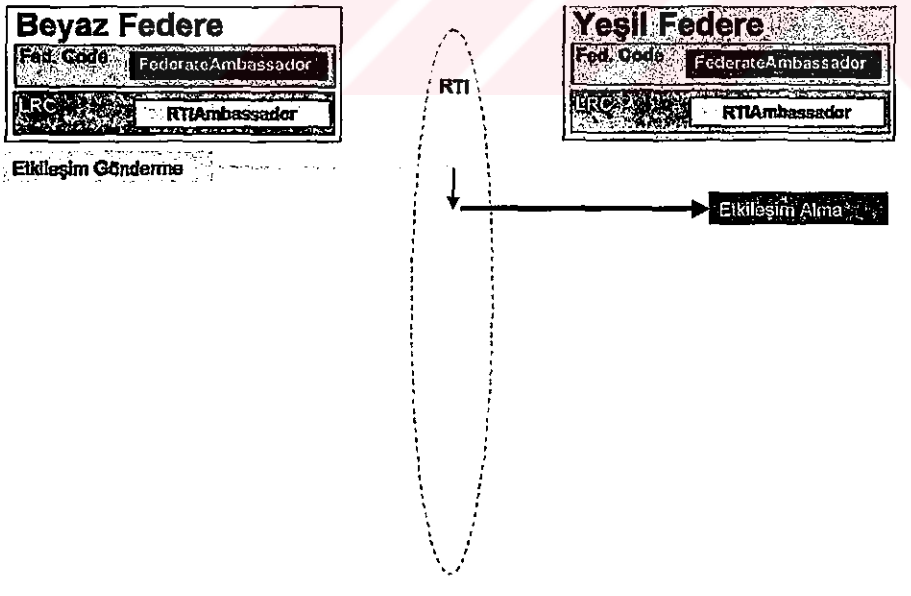
Nesne Örneğini Keşfetme† servisi talep edildiği zaman, burada, ister veri dağıtımı yönetimi kullanılmış olsun ister kullanılmamış olsun, doğrudan doğruya keşfeden federenin kapsamına girecek olan yeni keşfedilen nesne örneğinin bir parçası olarak bir örnek özellik olacaktır.

5.6.2. Nesne Özelliklerini Güncelleme ve Yansıtma

Şekil 5.13'de, nesne örneklerini güncellemek ve yansıtmak için gerekli olan etkileşimler görülmektedir. Keşfetme, kaydetmenin karşılığıdır. Yansıtma ise özellik güncellemelerine karşılık düşer.

5.6.3. Etkileşimlerin Karşılıklı Değişimi

Daha öncede belirtildiği gibi, nesneler kalıcı, ancak etkileşimler kalıcı değildir. Tertiplenen herbir etkileşim, gönderilir ve daha sonra unutulur. Etkileşim alıcıları, etkileşimi alır, şifresini çözer(decode) ve kullanır. Şekil 5.14'te etkileşimlerin üretilmesi ve tüketilmesi işlemi görülmektedir.



Şekil 5.14. Etkileşimlerin Karşılıklı Değişimi

Bir federe, *Etkileşim Almaƚ* servisi aracılığıyla herhangi bir etkileşimi alır. *Etkileşim Almaƚ* servisi talep edildiği zaman, bu servis talebinde bir argüman olan sınıf, bu servisi talep etmenin bir sonucu olarak alınan etkileşimin *alınan sınıfı(received class)* olarak adlandırılacaktır. Alınan sınıf alma anında, alınan sınıf adayı ile aynı olacaktır.

5.6.4. Nesne Yönetimi servisleri

5.6.4.1. Nesne Örneğini Kaydetme(Register Object Instance)

RTI, bir tek nesne örneği işaretçisi oluşturacak ve sağlanmış olan nesne sınıfının bir örneği ile onu bağlayacaktır. Uygun sınıf özellikleri, kaydeden federe tarafından yayımlanan nesne örneğinin bütün örnek özellikleri, kaydeden federe tarafından edinilmiş olur.

5.6.4.2. Nesne Örneğini Keşfetme †(Discover Object Instance †)

Nesne Örneğini Keşfetmeƚ servisi bir nesne örneğini keşfetmesi, bulması veya farkına varması için federeyi haberdar edecektir. Nesne örneği, başka bir federe tarafından kaydedildiği zaman, nesne örneği keşfedilmiş olacaktır.

5.6.4.3. Özellik Değerlerini Güncelleme(Update Attribute Values)

Özellik Değerlerini Güncelleme servisi, federe tarafından sahip olunan örnek özelliklerinin o anki değerlerini federasyona sağlayacaktır. Federe, FED içerisinde belirtildiği gibi örnek özellik değerlerindeki herhangi bir değişikliği bildirmelidir. *Özellik Değerlerini Yansıtmaƚ* servisi ile birlikte kullanılan bu servis, RTI aracılığıyla sağlanan karşılıklı veri alışverişi mekanizmasının temelini oluşturacaktır.

5.6.4.4. Özellik Değerlerini Yansıtma †(Reflect Attribute Values †)

Özellik Değerlerini Yansıtmaƚ servisi, yeni değerler ile belirlenmiş olan örnek özellikleri federeye sağlayacaktır. Benzer gönderme ve mesaj-düzenleme tiplerine

sahip olan örnek özellikleri için bir *Özellik Değerlerini Güncelleme* servisi talebi kapsamındaki bütün örnek özellik/değer çiftleri, bir adet uygun *Özellik Değerlerini Yansıtma†* servis talebi kapsamında olacaktır.

5.6.4.5. Etkileşim Gönderme(Send Interaction)

Etkileşim Gönderme servisi, bir etkileşimi federasyon içerisine gönderecektir. FED içerisinde de tanımlandığı gibi, etkileşim parametreleri, belirtilen sınıf ve bütün süper-sınıflar içinde olabilirler

5.6.4.6. Etkileşim Alma †(Receive Interaction †)

Etkileşim Alma† servisi, başka bir federe tarafından gönderilmiş olan bir etkileşimi federeye iletecektir.

5.6.4.7. Nesne Örneğini Silme(Delete Object Instance)

Nesne Örneğini Silme servisi, federe tarafından sahip olunan bir nesne örneğinin, federasyon uygulamasından kaldırılması isteğini federasyona bildirecektir. Nesne örneği, federasyon uygulamasından kaldırıldığı zaman, nesne örneğinin özelliklerini edinmiş olan bütün federeler, o özelliklere daha fazla sahip olamayacaktır. RTI, nesne örneğinin silindiğini federelere bildirmek için *Nesne Örneğini Kaldırma†* servisini kullanacaktır. Bu servisi talep eden federe, belirtilen nesne örneğinin silme imtiyazı(privilegeToDeleteObject) özelliğine sahip olmalıdır.

5.6.4.8. Nesne Örneğini Kaldırma †(Remove Object Instance †)

Nesne Örneğini Kaldırma† servisi, bir nesne örneğinin federasyon uygulamasından silindiğini federe bildirecektir.

5.6.4.9. Nesne Örneğini Sınırlı Silme(Local Delete Object Instance)

Nesne Örneğini Sınırlı Silme servisi, belirtilen nesne örneğini, sanki RTI talep eden federeye o nesne örneğini keşfetmesi için hiç haber vermemiş gibi işleme tabi tutmasını RTI'a bildirecektir. Nesne örneği, federasyon uygulamasından kaldırılmayacaktır. Federe, belirtilen nesne örneği için silme imtiyazı(privelegeToDeleteObject) örnek özelliğini sahiplenmeye ihtiyaç duymaz.

5.6.4.10. Özellik Taşıma Tipini Değiştirme(Chance Attribute Transportation Type)

Bir nesne örneğinin herbir özelliği için taşıma(transportation) tipi, başlangıçta FED içerisindeki nesne sınıf tanımındaki gibi olacaktır. Bir federe, uygulama sırasında taşıma tipini değiştirmeye karar verebilir. Eğer talep eden federe, onun taşıma tipi değiştikten sonra bir örnek özelliğinin mülkiyetini kaybederse ve o örnek özelliğinin mülkiyetini daha sonra tekrar elde ederse taşıma tipi, FED içerisinde tanımlanıldığı gibi olacaktır.

5.6.4.11. Etkileşim Taşıma Tipini Değiştirme(Change Interaction Transportation Type)

Herbir etkileşim için taşıma tipi, başlangıçta FED içerisindeki etkileşim sınıfı tanımlamasındaki gibi olacaktır. Bir federe, uygulama esnasında taşıma tipini değiştirmeye karar verebilir.

5.6.4.12. Özellikler Kapsama Alanında †(Attributes In Scope †)

Özellikler Kapsama Alanında† servisi, nesne örneğinin belirtilen özelliklerinin federe için kapsam dahilinde olduğunu federeye bildirecektir. Bu servis talebinden sonra, RTI, nesne örneğinin özellikler kümesindeki herhangi bir özellik için *Özellik Değerlerini Güncelleme†* servisini yayımlayabilir.

5.6.4.13. Özellikler Kapsam Dışında †(Attributes Out Of Scope †)

Özellikler Kapsam Dışında† servisi, nesne örneğinin belirtilen özelliklerinin federe için kapsam dışında olduğunu federeye bildirecektir. RTI, nesne örneğinin özellikler kümesindeki herhangi bir özellik için sonradan *Özellik Değerlerini Yansıtma†* servis taleplerini yayımlamamayı garanti edecektir.

5.6.4.14. Güncel Özellik Değerini İsteme(Request Attribute Value Update)

Güncel Özellik Değerini İsteme servisi, belirtilen özellik değerlerinin güncelleştirilmesini uyarmak için kullanılacaktır. Bu servis kullanıldığı zaman, RTI, *Güncel Özellik Değerini Sağlama†* servisini kullanarak, belirtilen özelliklerin güncel değerlerini sahiplerinden isteyecektir. Bir nesne sınıfı belirtildiği zaman, RTI o sınıfın bütün nesne örnekleri için belirtilen örnek özelliklerinin değerlerini isteyecektir.

5.6.4.15. Güncel Özellik Değerini Sağlama †(Provide Attribute Value Update †)

Güncel Özellik Değerini Sağlama† servisi, belirli bir nesne örneğinin federe tarafından edinilmiş olan özelliklerinin güncel değerlerini federeden talep edecektir. Federe, talep edilen örnek özellik değerlerini federasyona sağlamak için, *Özellik Değerini Güncellemeyi Sağlama†* servisine bir *Özellik Değerlerini Güncelleme* servis talebi ile karşılık verecektir.

5.6.4.16. Nesne Örneğini Güncellemeyi Açma †(Turn Updates On For Object Instance †)

Nesne Örneğini Güncellemeyi Açma† servisi, belirtilen nesne örneğinin belirtilen özellik değerlerinin federasyon uygulamasında bir yerde istenildiğini federeye belirtecektir.

5.6.4.17. Nesne Örneğini Güncellemeyi Kapama †(Turn Updates Off For Object Instance †)

Nesne Örneğini Güncellemeyi Kapama† servisi, belirtilen nesne örneğinin belirtilen özellik değerlerinin federasyon uygulamasında herhangi bir yerde istenilmediğini federeye belirtecektir.

5.7. Mülkiyet Yönetimi(Ownership Management)

Mülkiyet yönetimi servisleri, örnek özelliklerinin mülkiyetini federeler arasında transfer etmek için RTI ve federeler tarafından kullanılacaktır. Federelerin kendi aralarında örnek özelliklerinin mülkiyetini transfer etme yetenekleri, bir federasyon içinde belirli bir nesne örneğinin müşterek bir modelinin çıkarılmasını gerektirecektir. Bir örnek özelliğe sahip olan federe, o örnek özelliğe yeni değer sağlamak için sadece *Özellik Değerlerini Güncelleme* servisine başvuracaktır.

Bir örnek özelliğin herhangi bir zamanda birden fazla federe tarafından sahip olunması mümkün değildir, ancak örnek özellik tüm federeler tarafından sahiplenilmemiş(unowned) durumda olabilir. Federenin perspektifinden bakılacak olursa, herhangi bir örnek özellik, ya sahiplenilmiş(owned) ya da sahiplenilmemiş(unowned) durumdadır.

Bir nesne örneğinin kaydedilmesi üzerine, o nesne örneğini kaydeden federe, nesne örneğinin kaydedilmiş sınıfındaki uygun sınıf özelliklerini yayımlaması için o nesne örneğinin bütün örnek özelliklerine sahip olacaktır. O nesne örneğinin diğer bütün örnek özellikleri, tüm federeler tarafından sahiplenilmemiş(unowned) durumda olacaktır. Bir nesne örneğinin keşfedilmesi üzerine, keşfeden federe, o nesne örneğinin hiçbir örnek özelliğine sahip olamayacaktır. Eğer federe, bir örnek özelliğe sahip değilse, o örnek özellik için *Özellik Mülkiyetini Elde Etmeyi Bildirme†* servis talebini alana kadar, o örnek özelliğine sahip olamayacaktır.

Bir örnek özelliğinin mülkiyeti, ya onu elde etmeyi isteyen bir federe yoluyla yada kendisinin olan o örnek özelliği elden çıkarmayı isteyen federe yoluyla, bir federeden bir başka federeye transfer edilecektir. Mülkiyet, rızası olmadan bir federeden alınıp bir başka federeye verilmez.[10]

5.7.1. Mülkiyet ve Yayımlama(Ownership and publication)

Bir örnek özelliğin mülkiyeti, o örnek özelliğe karşılık olan sınıf özelliğinin, örnek özelliğin bilinen(known) sınıfında yayımlanıp yayımlanmadığı ile yakından

alakalıdır. Bir federe, bir nesne örneğinin uygun örnek özelliğini edinmek amacıyla o nesne örneğinin bilinen(known) sınıfında sınıf özelliğini yayımlayacaktır. Bu demektir ki;

- Bir nesne örneğinin ilgili örnek özelliğinin sahibi olmadan önce federe, o nesne örneğinin bilinen sınıfında bir sınıf özelliğini yayımlayacaktır.

- Eğer bir örnek özelliğe sahip olan federe, örnek özelliğin bilinen sınıfında uygun sınıf özelliğini yayınlamayı durdurursa, o örnek özellik, derhal sahiplenilmemiş(unowned) duruma gelecektir.

5.7.2. Mülkiyet Transferi

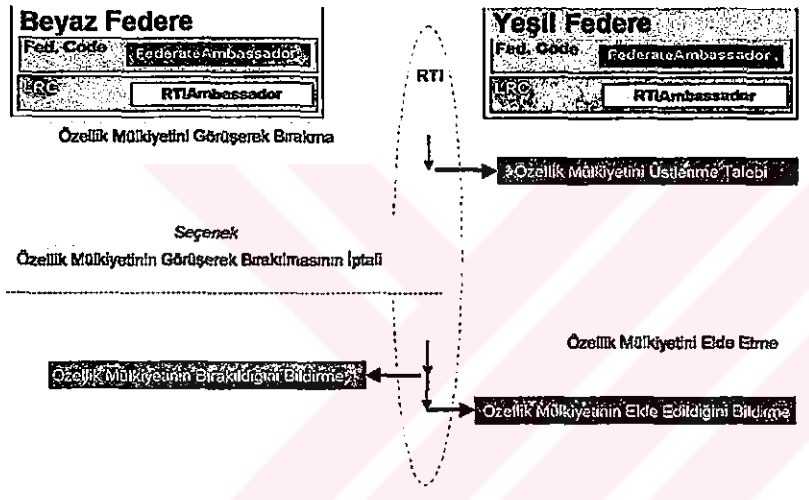
Başarılı bir şekilde elden çıkarılmış(divested) olan bir örnek özellik, o örnek özelliği elden çıkaran(divesting) federe için sahiplenilmemiş olacaktır. Eğer bir örnek özellik sahiplenilmemiş olursa, örnek özelliğinin tanınan sınıfında onun ilgili sınıf özelliği yayımlanmış ya da yayımlanmamış olabilir. Eğer bir sınıf özelliği yayımlanırsa, federenin uygun örnek özelliği elde etmesi mümkün olacaktır ve o örnek özelliğin mülkiyeti *Özellik Mülkiyetini Üstlenme Talebi†* servisi yoluyla RTI tarafından sağlanabilir. Federenin, kendi örnek özelliğini elinden çıkarmayı yada bırakmayı deneyebilmesi için beş yol ve eğer bir örnek özelliğe sahip değilse bir tane elde etmeyi deneyebilmesi için de iki yol vardır.

5.7.2.1. Bırakma(Divestiture)

Bir federe, kendi sahip olduğu bir örnek özelliği, sahiplenilmemiş yapabilmek için beş farklı yolu kullanır. Bunlar:

a) Federe, *Özellik Mülkiyetini Şartsız Bırakma* servisine başvurabilir, bu durumda örnek özellik derhal o federe tarafından(gerçekte bütün federeler tarafından) sahiplenilmemiş olur.

b) Federe, *Özellik Mülkiyetini Görüşerek Bırakma* servisine başvurabilir ki bu, eğer RTI o örnek özelliğe sahip olmaya istekli olan başka bir federenin yerini saptayabilirse, federenin kendi örnek özelliğini bırakmayı istediğini RTI'a bildirecektir. Eğer bazı federeler, bir örnek özelliği elde etmek için çalışmaktaysalar, bu federelerin o örnek özelliğe sahip olmayı istediklerinin bir göstergesidir. Eğer RTI, örnek özelliği elde etmeyi isteyen başka bir federenin yerini saptayabilirse, RTI mülkiyeti bırakmak isteyen federe nezdinde *Özellik Mülkiyetinin Bırakıldığını Bildirme†* servisine başvurarak, o örnek özelliğe artık daha fazla sahip olamayacağını bırakmak isteyen federeye bildirecektir(Şekil 5.15).



Şekil 5.15. Mülkiyeti elden çıkarma(bırakma) etkileşim diyagramı

c) Federe, *Özellik Mülkiyetini Serbest Bırakma Cevabı* servisine başvurabilir (federe, belirtilen örnek özellik için *Özellik Mülkiyetini Serbest Bırakmayı İsteme†* servis talebini almasına cevap olarak bu servise başvurur). Bu nedenle, eğer *Özellik Mülkiyetini Serbest Bırakma Cevabı* servisi, serbest bırakılan örnek özellikler arasındaki bir örnek özellik ile geri dönerse, o örnek özellik, sahiplenilmemiş olacaktır.

d) Federe, örnek özelliğinin tanınan sınıfında örnek özelliğin ilgili sınıf özelliğini yayımlamayı durdurabilir, ki bu, örnek özelliğin derhal o federe tarafından (gerçekte bütün federeler tarafından) sahiplenilmemiş olmasıyla sonuçlanacaktır.

e) Federe, federasyon uygulamasından ayrılabilir. Bir federe, Özellikleri Serbest Bırakma(Release Attributes) seçeneği ile başarılı bir şekilde federasyon uygulamasından ayrıldığı zaman, federe tarafından sahip olunan örnek özelliklerinin tümü, o federe tarafından(gerçekte bütün federeler tarafından) sahiplenilmemiş olacaktır.

Bir federenin kendi örnek özelliğini bırakabilmesini sağlayan bu beş yoldan sadece *Özellik Mülkiyetini Görüşerek Bırakma* servisi iptal edilebilir. Bir *Özellik Mülkiyetini Görüşerek Bırakma* servis talebi, ya örnek sahiplenilmemiş duruma gelene kadar, yada *Özellik Mülkiyetinin Görüşerek Bırakılmasını İptal Etme* servis talebi yoluyla bırakan federenin bırakma isteğini iptal etmesine kadar askıda kalır. Bırakma işleminin iptal edilmesi, başarılı bir şekilde gerçekleşecektir.

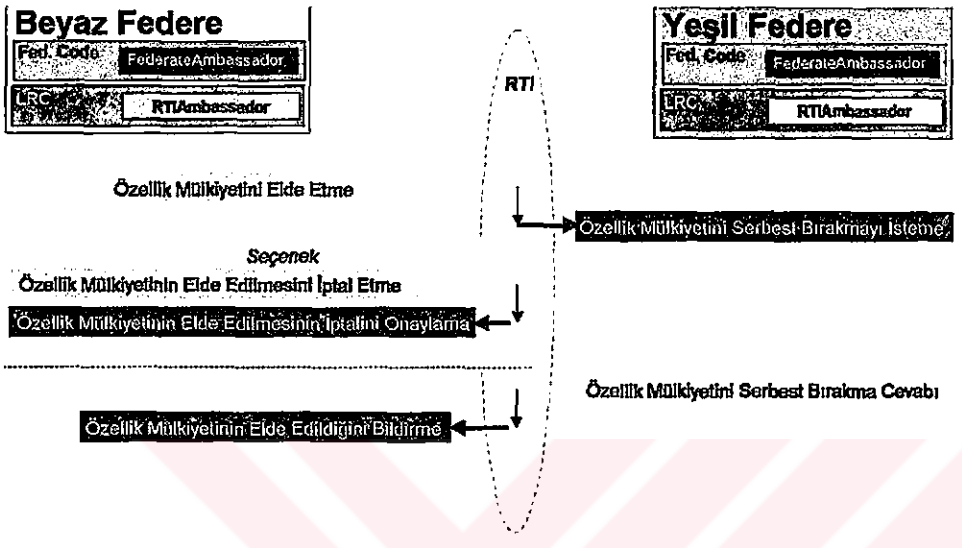
Bir federenin kendi örnek özelliğini bırakabilmesini sağlayan bu beş yoldan üçü, nesne örneğinin bütün federeler tarafından sahiplenilmemiş duruma gelmesiyle sonuçlanacaktır(*Özellik Mülkiyetini Şartsız Bırakma* servis talebi, örnek özelliğinin tanınan sınıfında örnek özelliğin ilgili sınıf özelliğinin yayımlanmasının durdurulması isteği ve *Federasyon Uygulamasından Ayrılma* servis talebi). *Özellik Mülkiyetini Şartsız Bırakma* ya da *Özellik Mülkiyetini Serbest Bırakma Cevabı* servisi kullanıldığı zaman, RTI, örnek özelliğin mülkiyetini federe kaybettikten sonra derhal örnek özelliği teminat altına alacak ve onun mülkiyetini bir başka federeye, verecektir.

5.7.2.2. Elde Etme(Acquisition)

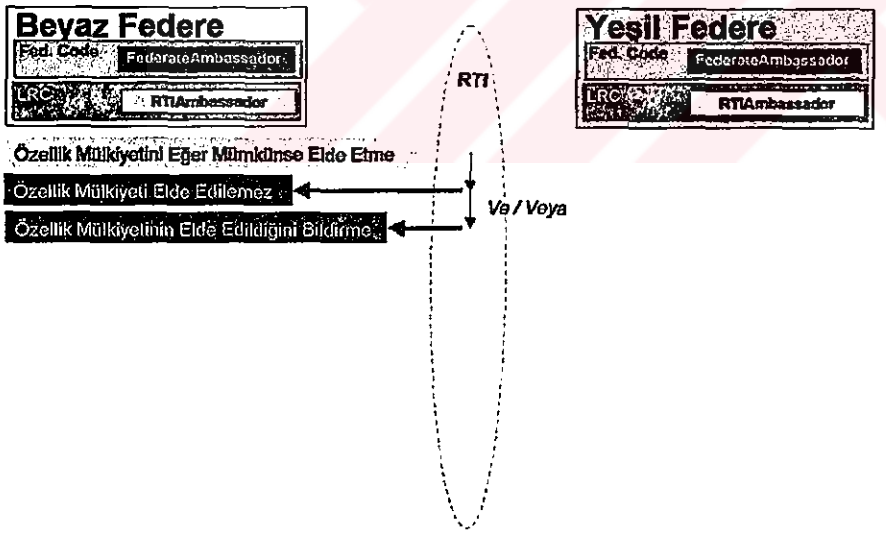
Bir federenin, bir sınıfa ait olan bir nesnenin uygun bir örnek özelliğini elde etmek için belirli bir sınıfta o sınıf özelliğini yayımlamasının iki yolu vardır. Bunlar:

a) Federe, *Özellik Mülkiyetini Elde Etme* servisine başvurabilir. Federe bu servise başvurduktan sonra, RTI, belirtilen örnek özelliğe sahip olan federe nezdinde *Özellik Mülkiyetini Serbest Bırakmayı İsteme†* servisine başvuracaktır(Şekil 5.16).

b) Federe, *Özellik Mülkiyetini Eğer Mümkünse Elde Etme* servisine başvurabilir ki bu, eğer belirtilen örnek özellik bütün federeler tarafından daha önceden sahiplenilmemiş durumdaysa veya örnek özellik sahibi tarafından bırakılma sürecindeyse federenin bu örnek özelliği elde etmeyi istediğini RTI'a bildirecektir(Şekil 5.17.).



Şekil 5.16. Mülkiyet Elde Etme(izinsiz-intrusive) Etkileşim Diyagramı



Şekil 5.17. Mülkiyet Elde Etme(özellik sahipsizken) Etkileşim Diyagramı

İlk elde etme metodu, sanki örnek özellik izinsiz(intrusive) elde ediliyormuş gibi bir izlenim verebilir, çünkü RTI, örnek özelliğe sahip olan federeyi onu elde etmeyi isteyen bir başka federenin varlığından haberdar edecektir ve mülkiyeti isteyen

federe tarafından örnek özelliğin elde edilmesine olanak tanıması için mülkiyete sahip olan federeden onu serbest bırakmasını isteyecektir. Elde etme metotlarından ikincisi de, ilkinin tam tersi olarak, elde etme işlemi gönül rızasıyla oluyormuş gibi bir izlenim vermektedir, çünkü RTI, örnek özelliği elde etmeyi istediğinden mülkiyet sahibi federeyi haberdar etmeyecektir.

Bir federe, *Özellik Mülkiyetini Elde Etme* servisine başvurduğu zaman, örnek özellik elde edilene kadar(*Özellik Mülkiyetinin Elde Edildiğini Bildirme†* servisi yoluyla) yada federe, elde etme isteğini başarılı bir şekilde iptal edene kadar, bu istek askıda kalacaktır. Bir federe *Özellik Mülkiyetini Elde Etmeyi İptal Etme* servisini talep ederek, elde etme isteğini iptal etmeyi deneyebilir. Ancak, *Özellik Mülkiyetini Elde Etmeyi İptal Etme* servisinin başarılı olacağı garanti edilemez. Eğer başarılı olursa, RTI, *Özellik Mülkiyetinin Elde Edilmesinin İptalini Onaylama†* servisine başvurarak iptal etmek isteyen federeye isteğinin başarı ile yerine getirildiğini bildirecektir. Eğer başarısız olursa, RTI, örnek özelliğin mülkiyetini federeye vermek suretiyle, *Özellik Mülkiyetinin Elde Edildiğini Bildirme†* servisine başvurarak, iptal eden federeye bu başarısızlığı bildirecektir.

5.7.3. Nesne Silme İmtiyazı(Privilege to Delete Object)

Bütün nesne sınıfları, *nesne silme imtiyazı(privilegeToDeleteObject)* diye adlandırılan bir özelliğe sahip olacaklardır(by default). Bütün mevcut diğer özellikler için olduğu gibi, *privilegeToDeleteObject* örnek özelliğinin mülkiyeti, federeler arasında transfer edilebilir. *privilegeToDeleteObject* örnek özelliği hakkındaki mülkiyet yönetimi servislerinin kullanım şekli, diğer örnek özellikler için nasılsa öyle olacaktır. Ancak, bir federenin *privilegeToDeleteObject* örnek özelliğine sahip olmayı istemesi, diğerlerinden farklı olur. Tipik bir örnek özelliğinin mülkiyeti, o örnek özellik için yeni değerler sağlama ayrıcalığını federeye verecektir. Bir nesne örneğinin *privilegeToDeleteObject* örnek özelliğinin mülkiyeti, o nesne örneğini federasyon uygulamasından silme hakkını ekstra olarak federeye verecektir. *privilegeToDeleteObject* sınıf özelliği, bütün nesne sınıfları için açık bir şekilde yayımlanır.

5.7.4. Mülkiyet Yönetimi Servisleri

5.7.4.1. Özellik Mülkiyetini Şartsız Bırakma(Unconditional Attribute Ownership Divestiture)

Özellik Mülkiyetini Şartsız Bırakma servisi, federenin belirtilen nesnenin örnek özelliklerine artık daha fazla sahip olamayacağını RTI'a bildirecektir. Bu servis, kabul edici(accepting) bir federenin var olup olmamasına bakmaksızın, örnek özelliğin(veya özelliklerin) bir sahiplenilmemiş duruma girmesine neden olarak(muhtemelen geçici), federeyi mülkiyetten mahrum bırakarak meseleyi halledecektir. Bu servis talebini yapma, bu servisle birlikte belirtilen örnek özelliklerinin tümü için *Özellik Mülkiyetinin Bırakıldığını Bildirme†* servisini gerektiren talep olarak görülecektir.

5.7.4.2. Özellik Mülkiyetini Görüşerek Bırakma(Negotiated Attribute Ownership Divestiture)

Özellik Mülkiyetini Görüşerek Bırakma servisi, federenin belirtilen nesne örneğinin belirtilen örnek özelliklerine artık daha fazla sahip olamayacağını RTI'a bildirecektir. Mülkiyet, ancak federe(veya federeler) kabul ederse transfer edilecektir. Bu servisi talep eden federe, *Özellik Mülkiyetinin Bırakıldığını Bildirme†* servisi yoluyla durdurma iznini alana kadar, belirtilen örnek özelliklerin güncelleme sorumluluğuna sahip olacaktır. Farklı federeler, farklı örnek özelliklerin sahibi olmayı isteyebildiği için, federe, bu servisi her talep ettiğinde bir veya daha fazla *Özellik Mülkiyetinin Bırakıldığını Bildirme†* servis talebini alabilir.

5.7.4.3. Özellik Mülkiyetini Üstlenme Talebi †(Request Attribute Ownership Assumption †)

Özellik Mülkiyeti Üstlenme Talebi† servisi, belirtilen örnek özelliklerin mülkiyetinin transferinin mümkün olduğunu federeye bildirecektir. Bir federenin *Özellik Mülkiyetini Şartsız Bırakma* servisine başvurmasının sonucu olarak, sağlanmış olan

örnek özelliklerin, sahiplenilmemiş olması durumunda, bırakan federe, mülkiyeti üstlenmeyi istemeyecektir.

5.7.4.4. Özellik Mülkiyetinin Bırakıldığını Bildirme †(Attribute Ownership Divestiture Notification †)

Özellik Mülkiyetinin Bırakıldığını Bildirme† servisi, belirtilen örnek özellikler kümesine artık daha fazla sahip olamayacağını federeye bildirecektir. Bunun üzerine, federe, belirtilen örnek özelliklerin değerlerini güncellemeyi durduracaktır. Federe, farklı federelerin farklı örnek özelliklerin sahibi olmayı istemelerinin sonucu olarak, tek bir *Özellik Mülkiyetini Görüşerek Bırakma* servisi talebi için birden fazla bildiri(notification) alabilir.

5.7.4.5. Özellik Mülkiyetinin Elde Edildiğini Bildirme †(Attribute Ownership Acquisition Notification †)

Özellik Mülkiyetinin Elde Edildiğini Bildirme† servisi, federeye belirtilen örnek özelliklere sahip olduğunu bildirecektir. Federe bu durumda, örnek özellik değerlerini güncellemeye başlayabilir.

5.7.4.6. Özellik Mülkiyetini Elde Etme(Attribute Ownership Acquisition)

Özellik Mülkiyetini Elde Etme servisi, belirtilen nesne örneğinin belirtilen örnek özelliklerinin mülkiyetini talep edecektir. Eğer belirtilen örnek özellik, başka bir federe tarafından sahip olunmuşsa, RTI, o örnek özellik için sahip olan federe nezdinde *Özellik Mülkiyetini Serbest Bırakmayı İsteme†* servisine başvuracaktır.

5.7.4.7. Özellik Mülkiyetini Eğer Mümkünse Elde Etme(Attribute Ownership Acquisition If Available)

Özellik Mülkiyetini Eğer Mümkünse Elde Etme servisi, eğer nesne örneği bütün federeler tarafından sahip olunmamışsa yada sahibi tarafından bırakılma sürecindeyse, belirtilen nesne örneğinin belirtilen örnek özelliğinin mülkiyetini talep

edecektir. Eğer belirtilmiş olan nesne örneği başka bir federe tarafından sahip olunmuşsa, RTI, o nesne örneği için, sahip olan federe nezdinde, *Özellik Mülkiyetini Serbest Bırakmayı İsteme†* servisine başvurmayacaktır. Federe, belirtilmiş olan nesne örneklerinin herbiri için, ya *Özellik Mülkiyetinin Elde Edildiğini Bildirme†* yada *Özellik Mülkiyeti Elde Edilemez†* servis talebini alacaktır.

5.7.4.8. Özellik Mülkiyeti Elde Edilemez †(Attribute Ownership Unavailable †)

Özellik Mülkiyeti Elde Edilemez† servisi, belirtilen örnek özelliğin mülkiyetinin elde edilmesinin mümkün olmadığını federeye bildirecektir.

5.7.4.9. Özellik Mülkiyetini Serbest Bırakmayı İsteme †(Request Attribute Ownership Release †)

Özellik Mülkiyetini Serbest Bırakmayı İsteme† servisi, belirtilen nesne örneğinin belirtilen örnek özelliklerinin mülkiyetini serbest bırakmasını federeden isteyecektir.

5.7.4.10. Özellik Mülkiyetini Serbest Bırakma Cevabı(Attribute Ownership Release Response)

Özellik Mülkiyetini Serbest Bırakma Cevabı servisi, federenin belirtilen nesne örneğinin belirtilen örnek özelliklerinin mülkiyetini serbest bırakmaya razı olduğunu RTI'a bildirecektir. Federe, RTI'ın *Özellik Mülkiyetini Serbest Bırakmayı İsteme†* talebine bir cevap teşkil etmesi için bu servisi kullanacaktır.

5.7.4.11. Özellik Mülkiyetinin Görüşerek Bırakılmasının İptali(Cancel Negotiated Attribute Ownership Divestiture)

Özellik Mülkiyetinin Görüşerek Bırakılmasının İptali servisi, federenin belirtilen örnek özelliklerinin mülkiyetinin bırakılmasını artık istemediğini RTI'a bildirmesini sağlayacaktır.

5.7.4.12. Özellik Mülkiyetinin Elde Edilmesini İptal Etme(Cancel Attribute Ownership Acquisition)

Özellik Mülkiyetinin Elde Edilmesini İptal Etme servisi, federenin belirtilen örnek özelliklerin mülkiyetinin elde edilmesini istemediğini RTI'a bildirecektir. Bu servis, RTI'ın iki cevabından birini her zaman alacaktır. Birinci cevap şekli olan, *Özellik Mülkiyetini Elde Edilmesinin İptalini Onaylama†*, belirtilen örnek özelliklerin mülkiyetini elde etme talebinin başarılı bir şekilde iptal edildiğini belirtecektir. İkinci cevap şekli olan, *Özellik Mülkiyetinin Elde Edildiğini Bildirme†*, belirtilen örnek özelliklerin mülkiyetini elde etme talebinin vaktinde iptal edilemediğini ve belirtilen örnek özelliklerin mülkiyetini elde ettiğini federeye belirtecektir. İptal etme işleminin, bazı örnek özellikler için başarılı, bazıları içinse başarısız olabildiği için, federe tek bir *Özellik Mülkiyetinin Görüşerek Bırakılmasının İptali* talebine yanıt olarak bu her iki cevap formunu alabilir. Bu servis, sadece *Özellik Mülkiyetini Elde Etme* servisi yoluyla yapılan örnek özelliklerinin mülkiyetini elde etmek talebini iptal etmek için kullanılacaktır. *Özellik Mülkiyetini Eğer Mümkünse Elde Etme* servisi yoluyla yapılan talepler, açık bir şekilde iptal edilemeyecektir.

5.7.4.13. Özellik Mülkiyetini Elde Etmenin İptalini Onaylama †(Confirm Attribute Ownership Acquisition Cancellation †)

Özellik Mülkiyetini Elde Etmenin İptalini Onaylama† servisi, federeye belirtilen örnek özelliklerin mülkiyetini elde etmeye artık aday olmadığını bildirecektir.

5.7.4.14. Özellik Mülkiyetini Soruşturma(Query Attribute Ownership)

Özellik Mülkiyetini Soruşturma servisi, belirtilen örnek özelliğin sahibinin kim olduğunu saptamak için kullanılacaktır. RTI, *Özellik Mülkiyeti Hakkında Bilgi Verme†* servis talebi aracılığıyla özellik mülkiyetinin sahibi hakkında gereken enformasyonu federeye sağlayacaktır.

5.7.4.15. Özellik Mülkiyeti Hakkında Bilgi Verme †(Inform Attribute Ownership †)

Özellik Mülkiyeti Hakkında Bilgi Verme† servisi, belirtilen örnek özellik hakkında mülkiyet bilgisi sağlamak için kullanılacaktır. Bu servis, bir federe tarafından yapılmış olan *Özellik Mülkiyetini Soruşturma* servisi talebine yanıt olarak RTI tarafından talep edilecektir. Bu servis, federeye, (eğer örnek özellik başka bir federe tarafından sahip olunmuşsa) özellik mülkiyetinin sahibini belirtecektir veya o örnek özelliği elde etmenin müsait olduğunu belirtecektir.

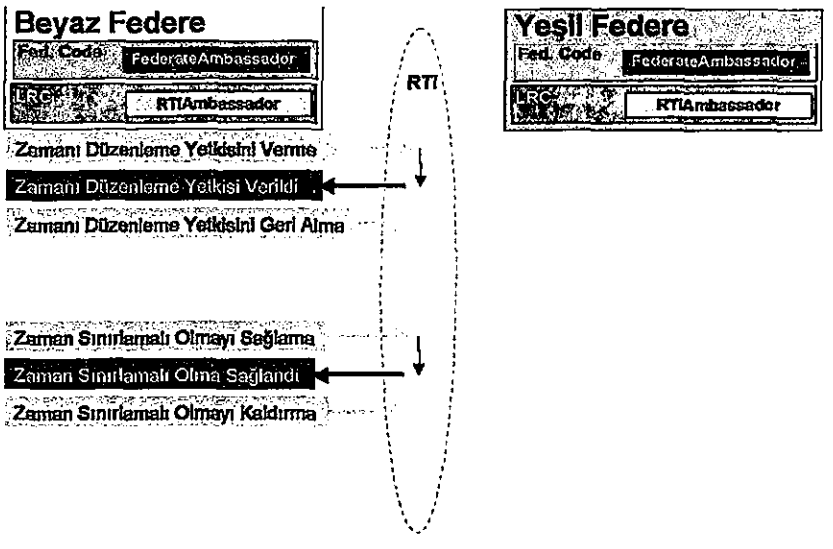
5.7.4.16. Federe Tarafından Sahip Olunan Özellik Olma(Is Attribute Owned By Federate)

Eğer belirtilen nesne örneğinin belirtilen örnek özelliği, talep eden federe tarafından sahip olunuyorsa, *Federe Tarafından Sahip Olunan Özellik Olma* servisi bunu belirlemek için kullanılacaktır. Servis, belirtilen örnek özelliğinin mülkiyet durumunu gösteren bir Boolean değeri sağlayacaktır.

5.8. Zaman Yönetimi(Time Management)

Sistem içerisinde modellenmiş olan zaman, federasyon zaman ekseni boyunca noktalar olarak federasyon içerisinde temsil edilecektir. Zaman yönetimi, herbir federenin federasyon zaman ekseni boyunca ilerleyişini kontrol etme mekanizmaları ile yakından ilgilidir. Her federe, çalışma esnasında eksen boyunca ilerleyebilir. Federenin zamanda ilerlemesi, diğer federelerin ilerlemesi tarafından sınırlandırılmış veya sınırlandırılmamış olabilir.

"Zaman düzenleyici-time regulating" konumunda olan bir federe, federasyon zaman ekseni üzerindeki noktalar ile kendi faaliyetlerinden bazıları(örnek özellik değerlerini güncelleme, etkileşimleri gönderme, vb. gibi) arasında ilişki kurabilir. Bu, federasyon zaman ekseni üzerindeki noktalara tekabül eden faaliyetlere zaman işaretleri atanarak mümkün olur. "Zaman sınırlamalı-time constrained" federe, bir federasyon zaman-işareti(time-stamp) sırası ile bu faaliyetlerin(örnek özellik değerlerini yansıtmak ve etkileşimleri almak gibi) bildirilerini almaya alaka gösterir. Zaman yönetimi servislerinin kullanımı, bir uygulama içinde zaman-düzenleyici(time-regulating) ve zaman-sınırlamalı(time-constrained) federeler arasında bu tip bir koordinasyon sağlar. Burada adı geçen koordinasyon, bu bölümde tanımlanan federe faaliyetleri üzerinde bir takım kısıtlamalar yoluyla sağlanacaktır. Şekil 5.18'de bir federenin nasıl zaman düzenleyici veya zaman sınırlamalı olduğu ve daha sonra bu yetkileri nasıl bıraktığı görülmektedir.



Şekil 5.18. "Düzenleyici" ve "Sınırlamalı" durumlarını kontrol etme.

Ne zaman düzenleyici olan nede zamanı-sınırlandırılan federelerin aktiviteleri, RTI tarafından diğer federeler ile birlikte koordine edilmez, ve bu federeler, zaman yönetimi servislerini kullanmaya ihtiyaç duymaz.

5.8.1. Mesajlar

HLA servislerinin zaman ile koordine edilmesinin yolu *mesajlar* kavramı iledir.

Bir federe tarafından RTI'a gönderilen mesajlar, genellikle bir ya da birden fazla federenin benzer bir mesajı almasıyla sonuçlanır. Örneğin, bir *Özellik Değerlerini Güncelleme* servis talebini temsil ederek gönderilmiş olan bir mesaj, normal yayımlama(publication)/abone olma(subscription) özelliklerine bağlı olarak, uygun federeler üzerinde *Özellik Değerlerini Yansıtma†* servisine bir başvuruyu temsil ederek alınmış olan mesajlarla sonuçlanacaktır. Mesajlar, aynı zamanda, *olayları(events)* da kapsar.

Gönderilen veya alınan her mesaj, ya zaman işareti sıralı(Time Stamped Order-TSO) mesaj, yada alma sıralı(Receive Order-RO) mesaj olacaktır.

5.8.2. Mantıksal zaman(logical time)

Bir federe bir federasyona bağlandığı zaman o federe için bir *mantıksal zaman(logical time)* tayin edilir. İlkin federenin mantıksal zamanı, federasyon zaman ekseni üzerinde başlangıç zamanı olan sıfır anı olarak belirlenir. Bir federasyon içindeki zaman sadece ileriye doğru gidebilir; böylece bir federe kendisinin o anki mantıksal zamanına eşit veya daha büyük olan bir zamana kadar ileri gitmeyi talep edebilir. Bir federenin kendi mantıksal zamanı içinde usulüne göre ilerlemesi için, açık bir şekilde ilerlemeyi talep etmesi gerekir. Ve RTI onaylayana kadar ilerleme gerçekleşmez. Genelde, bir uygulama esnasında herhangi bir anda farklı federeler farklı farklı mantıksal zamanda olabilirler.

5.8.3. Zaman Yönetimi servisleri

5.8.3.1. Zamanı Düzenleme Yetkisini Verme(Enable Time Regulation)

Zamanı Düzenleme Yetkisini Verme servisi, bu servisi talep eden federe için zamanı düzenlemeyi olanaklı kılacaktır ve federe yetkilendirilmek suretiyle TSO mesajlarını gönderebilecektir. Federe, bu gibi servis taleplerinde belirtilen değerler olan argümanlar gibi belirlenen mantıksal zamanı ve lookahead değerini talep edecektir. RTI, federenin mantıksal zamanını istenilen değerde gerçekleştirmeyebilir, çünkü, bir başka federenin geçerli mantıksal zamanından daha küçük olan zaman işaretli bir mesajı gönderme konusunda federeyi yetkilendirebilir. RTI, *Zamanı Düzenleme Yetkisi Verildi†* servisi ile belirlenmiş olan mantıksal zamanı federeye gösterecektir. RTI tarafından belirlenerek federeye belirtilen bu mantıksal zaman, federe tarafından talep edilen değere eşit veya daha büyük olacaktır.

RTI'ın *Zamanı Düzenleme Yetkisi Verildi†* servisini karşılık olarak talep etmesi üzerine, talep eden federe, mantıksal zamanı ve kendi lookahead değerinin toplamına eşit veya daha büyük zaman-işaretlerine sahip olan TSO mesajlarını göndermeye başlayabilir. Zamanı düzenlemeye bir kez yetki verildiği zaman, sıfır lookahead değerine sahip federeler ilave kısıtlamalara maruz kalmaz.

5.8.3.2. Zamanı Düzenleme Yetkisi Verildi †(Time Regulation Enabled †)

Zamanı Düzenleme Yetkisi Verildi† servisinin talebi, daha önceden yapılmış olan zamanı düzenlemeye yetkili olma isteğinin kabul edildiğini federeye belirtecektir. Bu servisin argüman değeri, federenin mantıksal zamanının belirtilen değere kurulduğunu gösterecektir.

5.8.3.3. Zamanı Düzenleme Yetkisini Geri Alma(Disable Time Regulation)

Zamanı Düzenleme Yetkisini Geri Alma servisinin talebi, federenin zamanı düzenleme yetkisinden mahrum olmak istediğini belirtecektir. Daha sonra federe

tarafından gönderilen mesajlar, otomatik bir şekilde RO mesajları olarak gönderileceklerdir.

5.8.3.4. Zaman Sınırlamalı Olmayı Sağlama(Enable Time Constrained)

Zaman Sınırlamalı Olmayı Sağlama servisi, servisi talep eden federenin zamanı sınırlamalı federe olmak istediğini gösterecektir. RTI *Zaman Sınırlamalı Olma Sağlandı†* servisini çağırarak, federenin zaman-sınırlamalı olduğunu belirtecektir.

5.8.3.5. Zaman Sınırlamalı Olma Sağlandı †(Time Constrained Enabled †)

Zaman Sınırlamalı Olma Sağlandı† servisinin çağrılması, daha önceden zamanı kısıtlanmış olmayı isteyen federenin isteğinin gerçekleştiğini belirtecektir. Bu servisin sağlamış olduğu argümanın değeri, federenin geçerli mantıksal zamanını gösterecektir.

Bir federe daha önceki durumundan değişerek zamanı sınırlandırılmış olduğu zaman, federenin mantıksal zamanına eşit veya daha büyük zaman işaretlerine sahip olan ve RTI'ın dahili kuyruklarında(queue) tutulan TSO mesajları, zaman-işareti(time-stamp) sırasına göre teslim edilecektir. Federeye zamanı sınırlandırılmış olmadan önce teslim edilen ve muhtemelen federenin geçerli mantıksal zamanına eşit veya daha büyük olan zaman işaretlerine sahip mesajları içeren TSO mesajları, RO mesajları olarak teslim edileceklerdir.

Eğer federe zaman düzenleyiciyse, argüman federenin mantıksal zamanına eşit olacaktır. Eğer federe zaman düzenleyici değilse, argüman federenin mantıksal zamanına eşit veya daha büyük olacaktır.

5.8.3.6. Zaman Sınırlamalı Olmayı Kaldırma(Disable Time Constrained)

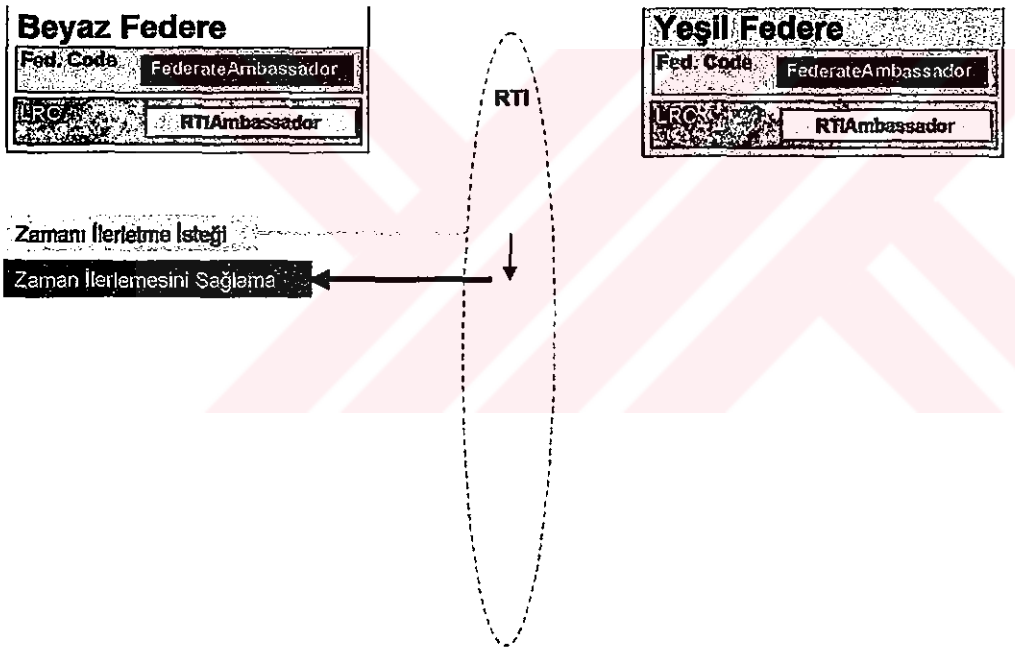
Zaman Sınırlamalı Olmayı Kaldırma servis talebi, artık federenin daha fazla zamanı sınırlandırılmış yani zaman sınırlamalı federe olmak istemediğini gösterecektir. Sonradan gelen tüm TSO mesajları federeye RO mesajları olarak teslim edilecektir.

5.8.3.7. Zamanı İlerletme İsteği(Time Advance Request)

Zamanı İlerletme İsteği servisi, federenin mantıksal zamanını ilerletmesi ve yayınlanmış sıfır veya daha fazla mesajın federeye ulaşması için talep edilmiş olabilir.

Bu servisin talep edilmesi, aşağıdaki mesajlar kümesinin federeye teslim edilmesine sebep olur.

- Federenin RO mesajları olarak alacağı RTI içinde sıralanmış olan tüm mesajlar.
- Federenin belirtmiş olduğu zamana eşit veya daha düşük zaman işaretlerine sahip TSO mesajları olarak alacağı bütün mesajlar.



Şekil 5.19. Zaman-adımlı bir Federenin mantıksal zamanının ilerlemesi

Zamanı İlerletme İsteği talep edilmesinden sonra, mesajlar, RTI'ın *Etkileşim Almaƒ*, *Özellik Değerlerini Yansıtmaƒ* ve *Nesne Örneğini Kaldırmaƒ* servislerine başvurması yoluyla federeye iletilecektir.

Federe, belirtmiş olduğu bir zaman ile birlikte bu servisi talep ederek, federenin lookahead değeri sıfır olsa bile, ilerde herhangi bir zamanda, belirtilen zamana eşit

veya daha düşük bir zaman işaretine sahip TSO mesajını üretmeyeceğini garanti eder. Ayrıca, federe, belirtmiş olduğu zaman ve kendi lookahead değerinin toplamından daha küçük zaman işaretlerine sahip herhangi bir TSO mesajını gelecekte üretemez.

Zaman İlerlemesini Sağlamaƭ servisi, bu isteği tamamlayacak ve belirtilen zamana kadar kendi mantıksal zamanının ilerletildiğini federeye bildirecektir ve verilen zaman eşit veya daha küçük zaman işaretlerine sahip ilave TSO mesajlar, gelecekte federeye teslim edilmeyecektir(Şekil 5.19).

5.8.3.8. Zamanı İlerletme İsteği Mevcut(Time Advance Request Available)

Zamanı İlerletme İsteği Mevcut servisi, federenin mantıksal zamanını ilerletmesi için talep edilecektir. Aşağıdakiler hariç bir T zamanı kadar *Zamanı İlerletme İsteği* servisiyle birbirine benzer:

- RTI, T zamanı için bir *Zaman İlerlemesini Sağlamaƭ* yayımladığı zaman, T'ye eşit zaman işaretlerine sahip bütün mesajların dağıtımını garanti etmeyecektir, ve,
- Federenin T zamanı için *Zaman İlerlemesini Sağlamaƭ* servisini almasından sonra, eğer federenin lookahead değeri sıfırsa, o T'ye eşit zaman işaretli ek mesajları gönderebilir.

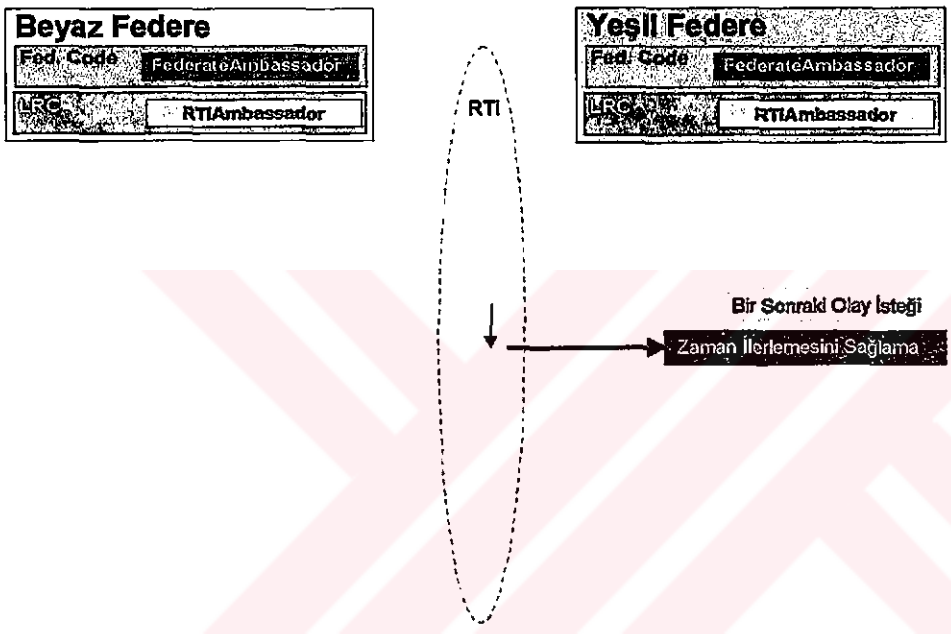
Verilen zamana eşit zaman işaretlerine sahip ilave mesajlar, ileriki bir zamanda gelebilir.

5.8.3.9. Bir Sonraki Olay İsteği(Next Event Request)

Bir Sonraki Olay İsteği servisi, mesajın talep içinde belirtilen mantıksal zamandan daha büyük olmayan bir zaman işaretine sahip olması koşuluyla, federenin mantıksal zamanının, federeye teslim edilen bir sonraki TSO mesajının zaman işaretine kadar ileri götürülmesini talep edecektir.

Bu servisin talep edilmesi, aşağıdaki mesajlar kümesinin federeye teslim edilmesine sebep olur:

- Federenin RO mesajları olarak alacağı RTI içinde sıralanmış olan tüm mesajlar.
- Belirtilen zamana eşit veya daha düşük zaman işaretlerine sahip bir TSO mesajı olarak federe tarafından herhangi bir zamanda alınacak olan en küçük zaman işaretli mesaj ve federenin TSO mesajları olarak alacağı eşit zaman işareti içeren bütün diğer mesajlar.



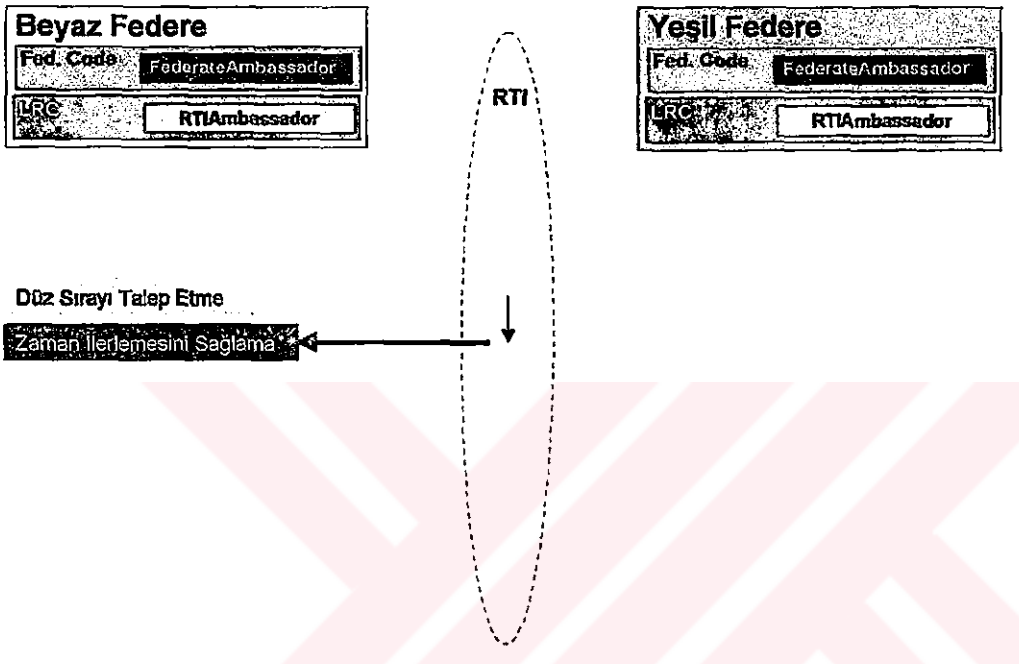
Şekil 5.20. Olay tabanlı bir federenin mantıksal zamanını ilerletmesi

Zaman İlerlemesini Sağlama servisi, bu isteği tamamlayacak ve federenin teslim edilen TSO mesajlarının zaman işaretine kadar kendi mantıksal zamanını ilerlettiğini belirtecektir(Şekil 5.20).

5.8.3.10. Bir Sonraki Olay İsteği Mevcut(Next Event Request Available)

Bir Sonraki Olay İsteği Mevcut servisi, federenin mantıksal zamanını, mesajın, istek içerisinde belirtilen mantıksal zamanından daha büyük olmayan zaman işaretine sahip olması koşuluyla, federeye ulaştırılacak olan bir sonraki TSO mesajın zaman işaretine kadar ilerletmek için talep edilecektir. Aşağıdakiler hariç *Bir Sonraki Olay İsteği* servisiyle aynıdır:

- RTI, bir T zamanı için *Zaman İlerlemesini Sağlama†* servisini piyasaya sürdüğü zaman, T'ye eşit zaman işaretlerine sahip bütün mesajların teslimini garanti etmeyecektir, ve
- Federenin, T zamanı için *Zaman İlerlemesini Sağlama†* servisini almasından sonra, eğer federenin lookahead değeri sıfırsa, T'ye eşit zaman işaretine sahip ilave mesajları gönderebilir.



Şekil 5.21. Optimistic bir federenin mantıksal zamanını ilerletmesi

Bu servisin talep edilmesi, aşağıdaki mesajlar kümesinin federeye teslim edilmesine sebep olur:

- Federenin RO mesajları olarak alacağı RTI içinde sıralanmış olan tüm mesajlar.
- Belirtilen zamana eşit veya daha düşük zaman işaretlerine sahip bir TSO mesajı olarak federe tarafından herhangi bir zamanda alınacak olan en küçük zaman işaretli mesaj ve federenin TSO mesajları olarak alacağı ve eşit zaman işaretine sahip olan RTI içinde sıralanmış bütün diğer mesajlar.

5.8.3.11. Düz Sırayı Talep Etme(Flush Queue Request)

Düz Sırayı Talep Etme servisi, federenin TSO mesajları olarak alacağı, RTI içinde sıralanmış olan bütün mesajların derhal teslim edilmesini talep edecektir. RTI,

olabildiğince çabuk bütün bu mesajları teslim edecektir. Eğer federe, belirtilen zamandan daha küçük zaman işaretlerine sahip ek TSO mesajlarını alamazsa, federenin mantıksal zamanı, belirtilen zamana kadar ileri gidecektir. Aksi halde, RTI, federenin mantıksal zamanını mümkün olduğu kadar ileri götürecektir(Şekil 5.21).

Bu servisin talep edilmesi, aşağıdaki mesajlar kümesinin federeye teslim edilmesine sebep olur:

- Federenin RO mesajları olarak alacağı RTI içinde sıralanmış olan tüm mesajlar.
- Federenin TSO mesajları olarak alacağı RTI içinde sıralanmış olan tüm mesajlar.

5.8.3.12. Zaman İlerlemesini Sağlama †(Time Advance Grant †)

Zaman İlerlemesini Sağlama† servis başvurusu, daha önceden yapılmış olan federenin mantıksal zamanını ileri götürme isteğinin kabul edildiğini gösterecektir. Bu servisin argümanı, federe için mantıksal zamanın bu değere kadar ileri götürüldüğünü belirtecektir.

Eğer onaylama, *Bir Sonraki Olay İsteği* veya *Zamanı İlerletme İsteği* servis başvurularına cevap olarak ortaya çıkarsa, RTI, daha sonra herhangi bir zaman içinde, bu değere eşit veya daha az olan zaman işaretlerine sahip fazladan ilave TSO mesajlarının teslim edilmeyeceğini garanti edecektir.

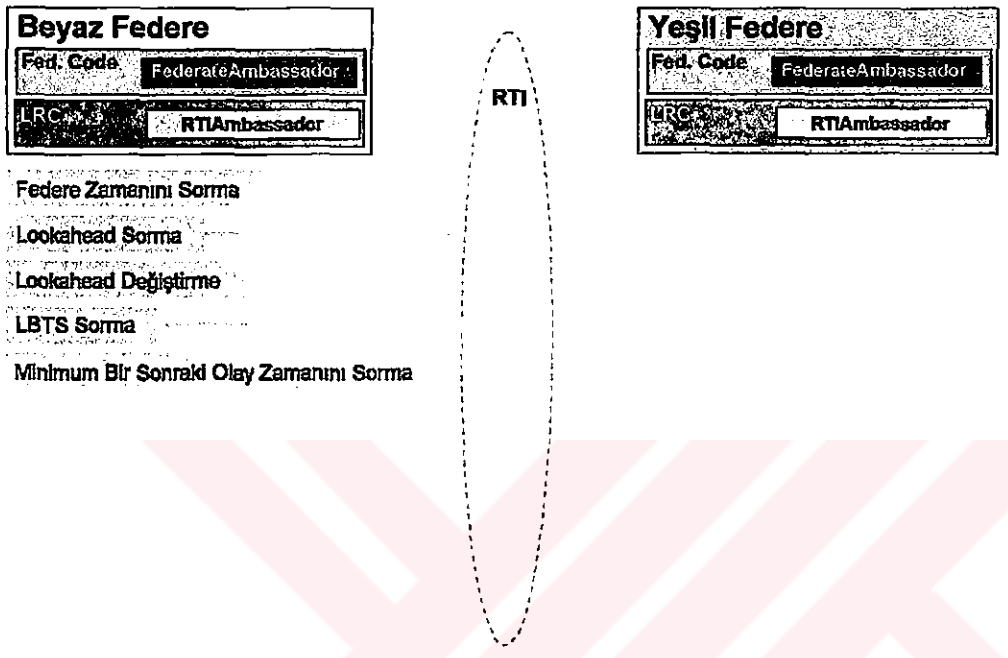
Eğer onaylama, *Zamanı İlerletme İsteği Mevcut*, *Bir Sonraki Olayı İsteme Mevcut* veya *Düz Sırayı Talep Etme* servis başvurularına cevap olarak yayınlanırsa, RTI, daha sonra herhangi bir zaman içinde, onaylanan değerden daha az zaman işaretlerine sahip fazladan ilave TSO mesajlarının teslim edilmeyeceğini garanti edecektir.

5.8.3.13. Asenkron Dağıtımı Sağlama(Enable Asynchronous Delivery)

Asenkron Dağıtımı Sağlama servis talebi, RTI'a daha önceden alınmış olan RO mesajlarını teslim etmesi talimatını verecektir.

5.8.3.14. Asenkron Dağıtımı Kaldırma(Disable Asynchronous Delivery)

Asenkron Dağıtımı Kaldırma servis talebi, RTI'a, bu servisi talep eden federe zamanı-sınırlı olduğu zaman, daha önceden alınmış olan RO mesajlarını teslim etmesi talimatını verecektir.



Şekil 5.22. Zamanla Bağlantılı Soruşturmalar

5.8.3.15. LBTS Sorma(Query LBTS)

LBTS Sorma servisi, bu servisi talep eden federenin o anki LBTS değerini isteyecektir.

5.8.3.16. Federe Zamanını Sorma(Query Federate Time)

Federe Zamanını Sorma servisi, talep eden federenin o anki mantıksal zamanı değerini isteyecektir.

5.8.3.17. Minimum Bir Sonraki Olay Zamanını Sorma(Query Minimum Next Event Time)

Minimum Bir Sonraki Olay Zamanını Sorma servisi, minimum LBTS'yi ve eğer varsa, talep eden federeye teslim edilmek için RTI tarafından elde tutulan bir sonraki gönderilen TSO mesajının zaman işaretini talep edecektir. Bu servisi talep eden federe için mevcut karşılık verilen zaman ile birlikte hiçbir mesaj/olay yoktur.

5.8.3.18. Lookahead Değiştirme(Modify Lookahead)

Lookahead Değiştirme servisi, federenin lookahead'ının gerçek değeri üzerinde değişiklik isteyecektir. Belirtilen lookahead değeri, sıfıra eşit veya daha büyük olacaktır. Eğer talep edilen değer, federenin gerçek lookahead değerine eşit veya daha büyükse, değişiklik, derhal işleme konacaktır ve talep edilen lookahead değeri gerçek değer olacaktır. Eğer, talep edilen değer, federenin gerçek lookahead değerinden daha küçükse, değişiklik, federenin mantıksal zamanını ilerletmesi gibi kademeli olarak işleyecek ve gerçek lookahead, ilkin değiştirilmemiş olacaktır. Talep edilen lookahead değere ulaşılana kadar belirli bir biçimde, federenin gerçek lookahead değeri, mantıksal zamanın her T ilerlemesi için T birim azalacaktır.

5.8.3.19. Lookahead Sorma(Query Lookahead)

Lookahead Sorma servisi, federenin o anki gerçek lookahead değerini RTI'a soracaktır. Eğer federe, gerçek lookahead değerini azaltmaya çalışıyorsa, o anki gerçek lookahead değeri, *Lookahead Değiştirme* servisi içinde verilen talep edilen lookahead değerinden geçici olarak farklı olabilir.

5.8.3.20. Geri Çekme(Retract)

Geri Çekme servisi, federe tarafından daha önceden gönderilmiş olan bir mesaj veya olayı geri çektiğinden federasyon uygulamasını haberdar etmek için federe tarafından kullanılacaktır. *Özellik Değerlerini Güncelleme*, *Etkileşim Göndereme* ve *Nesne Örneğini Silme* servisleri, geri çekilen olayı belirtmekte kullanılan bir olay geri

çekme işaretçisini geri döndüreceklerdir. Bir olayı geri çekme, orijinal olayı alan bütün federelere *Geri Çekme İsteği†* talebine sebep olacaktır.

Bir *Nesne Örneğini Silme* mesajını geri çekme işlemi, ilgili nesne örneğinin yeniden oluşturulmasıyla(reconstitution) sonuçlanacaktır. Bu, *Nesne Örneğini Silme* servisinin talep edilmesi anında, zarar görmüş nesne örneğinin özelliklerinin mülkiyetinin onları edinen federe tarafından yeniden ele geçirilmesine sebep olacaktır.

Sadece TSO içinde gönderilen mesajlar geri çekilebilir. Bir federe, geçmiş zamanda mesajları geri çekemez. Eğer kendi zamanı, federenin o anki mantıksal zamanından daha erkense, bir mesaj, federenin geçmişinde olacaktır.

5.8.3.21. Geri Çekme İsteği †(Request Retraction †)

Eğer RTI, bir federeye zaten teslim edilmekte olan bir olay için legal bir *Geri Çekme* servis başvurusu alırsa, *Geri Çekme İsteği†* servisi, o federe üzerinde talep edilecektir. Eğer adı geçen olay, federeye teslim edilmediyse, bu servis, o federe, üzerinde talep edilmeyecek; olay RTI'ın olay sırasından(kuyruğundan) kaldırılacak ve federeye hiçbir zaman teslim edilmeyecektir.

5.8.3.22. Özelliğin Düzen Türünü Değiştirme(Change Attribute Order Type)

Bir nesne örneğinin herbir özelliği için tercih edilmiş olan düzen türü, FED içerisindeki nesne sınıf açıklamasından kurulmuş olacaktır. Bir federe, uygulama sırasında tercih edilmiş olan düzen türü üzerinde değişikliği seçebilir. *Özelliğin Düzen Türünü Değiştirme* servisini talep etme, belirtilen örnek özellikleri için bütün gelecek *Özellik Değerlerini Güncelleme* servis talebi için düzen türünü değiştirecektir. Bir örnek özelliğin mülkiyeti değiştiği zaman, tercih edilen düzen türü, FED içinde tanımlanan eski haline dönecektir.

5.8.3.23. Etkileşimin Düzen Türünü Değiştirme(Change Interaction Order Type)

Herbir etkileşim için tercih edilmiş olan düzen türü, FED içerisindeki etkileşim sınıf açıklamasından kurulmuş olacaktır. Bir federe, uygulama sırasında tercih edilmiş olan düzen türü üzerinde değişikliği seçebilir. *Etkileşimin Düzen Türünü Değiştirme* servisini talep etme, sadece talep eden federe için belirtilen örnek özellikleri için bütün gelecek *Etkileşim Gönderme* ve *Etkileşimi Bölge İle Birlikte Gönderme* servis talepleri için düzen türünü değiştirecektir.

5.9. Veri Dağıtımı Yönetimi(Data Distribution Management)

Veri Dağıtımı Yönetimi(Data Distribution Management-DDM) servisleri, konu dışı(irrelevant) verinin hem alınmasını hem de iletimini önlemek için federeler tarafından kullanılabilir. Veri üreticileri, kullanıcı-tanımlı(user-defined) uzaylar açısından kendi verilerinin mahiyetlerini veya niteliklerini ilan etmek için DDM servislerini kullanabilirler. Veri tüketicileri de, kendi veri gereksinimlerini aynı uzaylar açısından belirtmek için DDM servislerini kullanabilirler. RTI, bu nitelikler ile gereksinimler arasındaki eşleştirmelere dayanarak veriyi üreticilerden alır tüketicilere dağıtır.

DDM servisleri, aşağıdaki kavramlar ve terimler üzerine kurulmuştur:

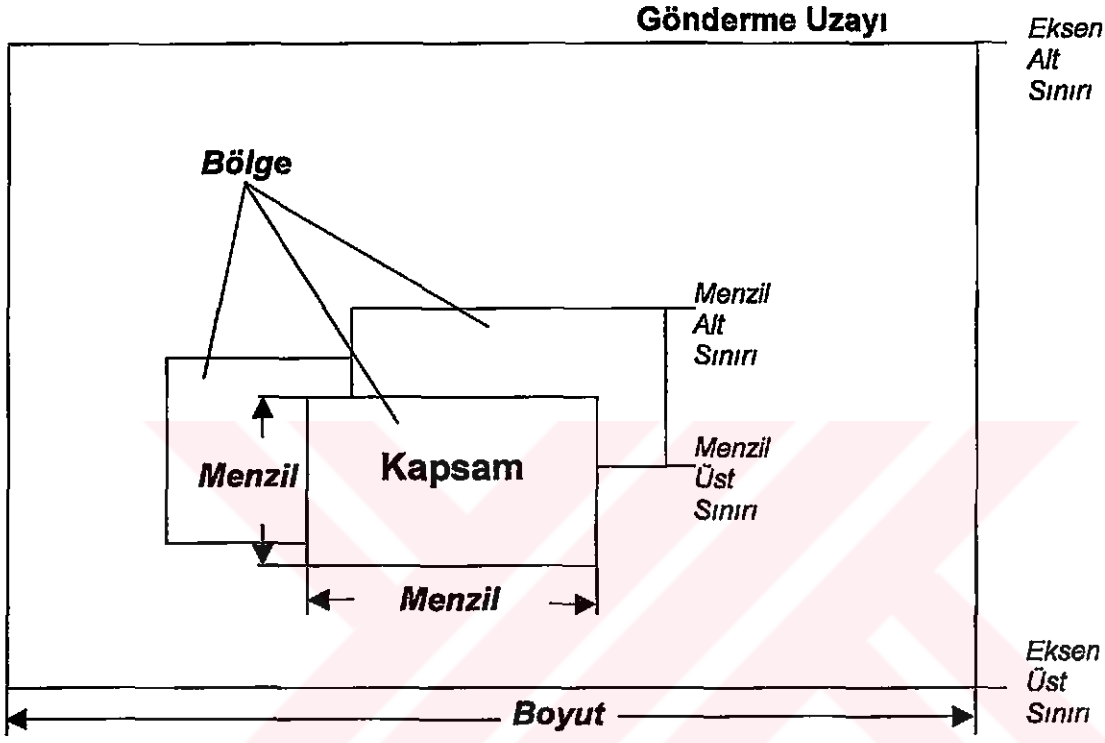
- *Boyut(dimension)*, FED içinde ifade edilen bir koordinat ekseni parçasıdır. RTI, bir değerler çifti ile belirlenen tek bir koordinat ekseni parçası sağlayacaktır. Bu, bütün boyutlar için FED içinde tanımlı bir temel sağlar. Bu değerler çiftinin ilk bileşeni *eksen alt sınırı(axis lower bound)*, ikinci bileşeni *eksen üst sınırı(axis upper bound)* olarak adlandırılacaktır. Bütün boyutlar, aynı koordinat-ekseni parçasını temel alacaktır ve böylece aynı şekilde alt ve üst sınıra sahip olacaklardır.

- *Gönderme Uzayı(routing space)*, çok boyutlu koordinat sistemini oluşturacak olan bir boyutlar zinciri olarak adlandırılır. RTI, açık bir şekilde tanımlanan *varsayılan gönderme uzayı(default routing space)* sağlayacaktır.

- *Menzil(range)*, değerler çifti yoluyla tanımlanan bir boyut üzerinde değişmez bir aralıktır. Bu değerler çiftinin ilk bileşeni, menzilin alt sınırı olarak, ikinci bileşeni de, menzilin üst sınırı olarak adlandırılacaktır.

- *Kapsam(extent)*, gönderme uzayının deklarasyonu içerisinde bulunan boyutlarla aynı biçimde tertip edilen, gönderme uzayındaki herbir boyut için bir tane olan bir menziller düzenidir.

- *Bölge(region)*, aynı gönderme uzayı ile sınırlanmış bir kapsamlar kümesidir. Bölge, gönderme uzayında bir alt-uzayı(sub-space) tanımlayacaktır.
- RTI, her gönderme uzayı için bir *varsayılan bölge(default region)* sağlar. Varsayılan bölge, gönderme uzayının tümünü kaplayacaktır. Belirli bir gönderme uzayı ile ilişkisi olmayan olaylar ve talepler, otomatik olarak varsayılan gönderme boşluğu ile ilişkilendirilir.



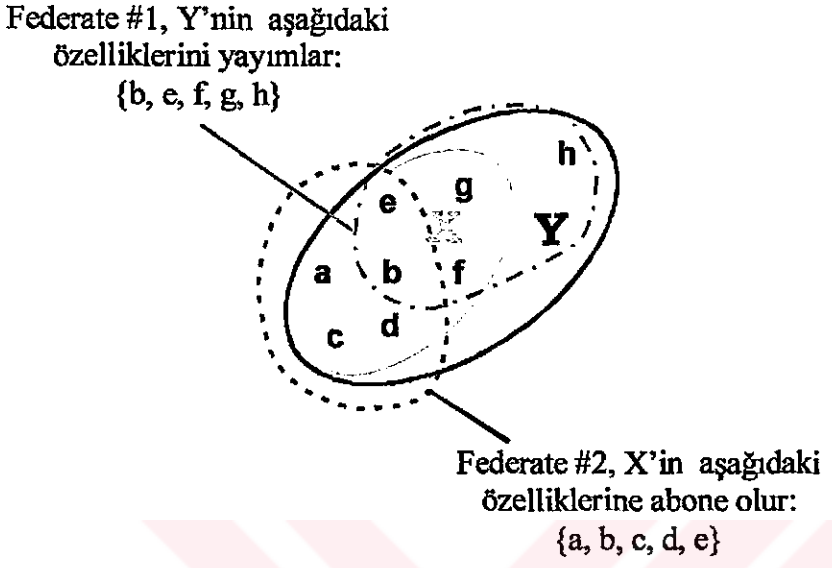
Şekil 5.23. İki boyutlu gönderme uzayı

Yukarıdakileri özetlemek gerekirse, veri dağıtımı yönetimi içerisinde bir federasyon "gönderme uzayı" tanımlıdır. *Gönderme uzayı*, bir "boyutlar" kolleksiyonudur. *Boyutlar*, "bölgeler"i tanımlamak için kullanılır. Her *bölge*, bir takım "kapsamlar" açısından tanımlanır. *Kapsam*, bir gönderme uzayının boyutları arasında tanımlı sınırları belirlenmiş bir menzildir(çok boyutlu bir gönderme uzayı içinde hacmi temsil eder)(Şekil 5.23).

5.9.1. Gönderme Uzayı Örneği

Bütün bu anlatılanlar, biraz karışık görünse de, burada anlatılacak olan örnek burada anlatılanların belki bir nebze olsun anlaşılmasına yardımcı olacaktır. *Deklarasyon*

Yönetimi konusu(Bölüm 5.5) içerisinde aşağıdaki örnek sunulmuştur. Şekil 5.9(burada tekrar Şekil 5.24), Federe #1'in yayımlama ilgisi ve Federe #2'nin buna abone olma ilgisini göstermektedir. Y nesne sınıfı, X nesne sınıfının soyundan gelmektedir(Bakınız Şekil 5.8, burada tekrar gösterilmedi).



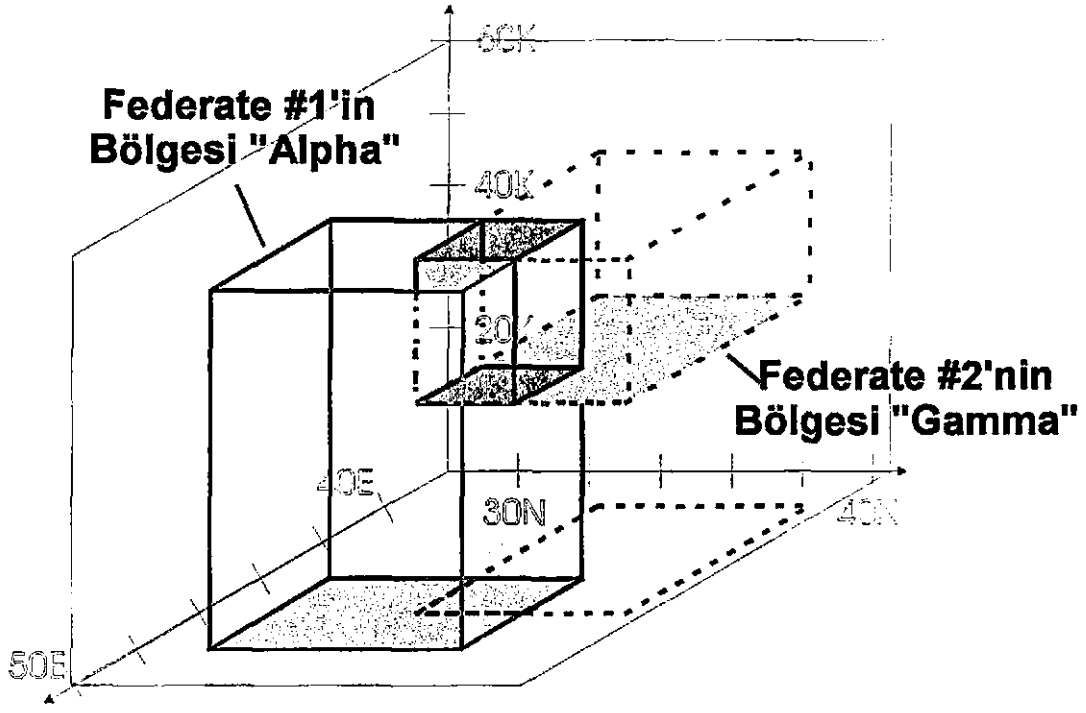
Şekil 5.24. Yayımlama ve abone olma kesişimleri.

Federe #1, Y:{b, e, f, g, h} yi yayımlayabildiğini göstermektedir. Federe #2, X: {a, b, c, d, e}ye abone olmak istediğini belirtmektedir. RTI, Y nesne sınıfının örneklerinin tanıtımını yapacaktır, öyle ki Federe #2, bu örnekleri X örnekleri olarak görür. Y sınıfından üretilen bilgi için bir tüketicinin bulunmasından sonra, RTI, Y örneklerini kaydetmesi gerektiğini Federe #1'e bildirir.

Üç boyut, "boylam", "enlem" ve "irtifa" ile tanımlanmış bir gönderme uzayını ele alalım. Şekil 5.25, böyle bir gönderme uzayını göstermektedir. Bu bölümde anlatılan örnek için, bu gönderme boşluğu, stenografi(shortland) notasyonu R{boylam, enlem, irtifa} ile gösterilmektedir.

Federeler, gönderme uzayı içindeki bölgeler şeklinde kendi abone olma deklarasyonları ve veri güncellemelerini iyi bir şekilde ayarlayabilir. Örneğin, Federe #1, Y nesne sınıfının {b, e, f, g, h} özelliklerini aşağıdaki bölge ile ilişkilendirebilir:

R_{Alpha} {boylam: 44°E - 48°E, enlem: 30°N - 37°N, irtifa: 0 - 50,000 ft}



Şekil 5.25. 3-Boyutlu gönderme uzayı örneği

Benzer bir şekilde, Federe #2, aşağıdaki bölge ile X içindeki {a, b, c, d, e} ye abone olabilir:

R_{Gamma} {boylam: 40°E - 46°E, enlem: 34°N - 40°N, irtifa: 30,000 ft - 50,000 ft}

R_{Alpha} ile R_{Gamma} bölgelerinin birbirinin üzerine geldikleri kısım, oldukça küçüktür. Gerçekte, bu iki bölgenin kesişimi;

$R_{Alpha \cap Gamma}$ {boylam: 44°E-46°E, enlem:34°N-37°N, irtifa:30,000 ft-50,000 ft}'dir.

Bununla birlikte, bu bölgeler kesiştiği için, RTI verinin Federe #1'den Federe #2'ye nakledilmesini sağlayacaktır.

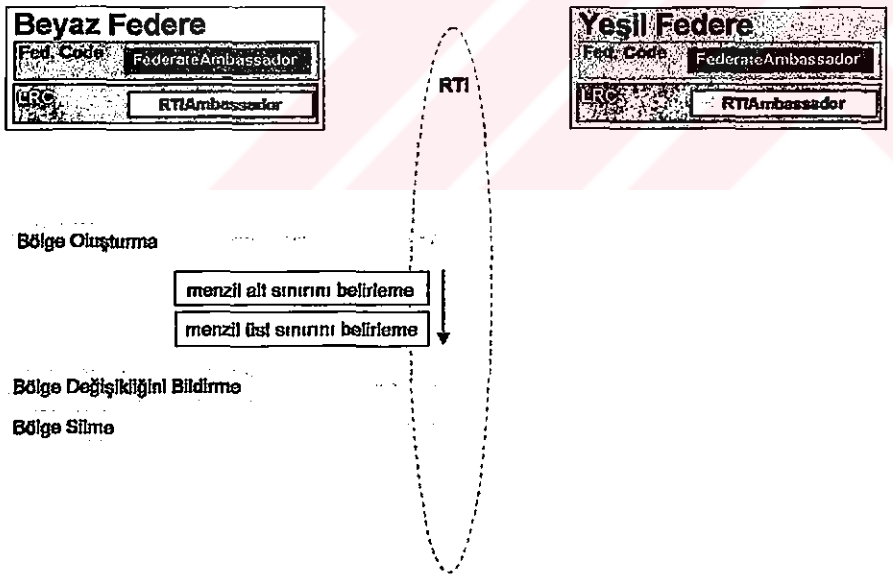
Bu örnekte, R{enlem, boylam, irtifa} gönderme uzayı kullanılmıştır. Örnekteki gönderme uzayı üç boyuta sahiptir. Gönderme uzayını kullanmaya karar veren bir federasyondaki bütün federeler, hem her boyut boyunca en kötü durum üst ve alt sınırları üzerinde hemde gönderme uzayını oluşturan boyutlar üzerinde anlaşma

sağlamalıdırlar. FED dosyası, federasyon içindeki federeler için kullanılabilir boyutları ve gönderme boşluklarını belirtir.

Örnek problemdeki federeler, 40°E ila 50°E arası boylam, 30°N ila 40°N arası enlem ve 0 ila 50,000 feet arası irtifa değerleri ile sınırlı bir gönderme uzayı üzerinde mutabakata varmışlardır. Bunun haricinde federeler, gönderme uzayı içindeki herbir boyutun en alt ve en üst sınırını bilmek zorundadırlar.

RTI, bölgeler ile özellikler arasında sezgisel(intuitive) bağlantılar kurmaz. Örneğin, bir uçak nesne sınıfı, "boylam", "enlem" ve "irtifa" özelliklerini içerebilir. Gönderme uzayı, "boylam", "enlem" ve "irtifa" boyutlarını içerebilir. RTI, "boylam" sınıf özelliği ile "boylam" gönderme uzayı boyutu arasında otomatikman bir ilişki kurmaz. Bölgeler ile olaylar(nesne güncellemeleri, etkileşimler, vs.) arasında ilişki kurmak tamamen üretici federenin seçimine kalmıştır.

5.9.2. Veri Dağıtımı Yönetimi Servisleri



Şekil 5.26. Bölge üzerinde yapılan işlemler.

5.9.2.1. Bölge Oluşturma(Create Region)

Bölge Oluşturma servisi, belirtilen gönderme uzayı boyutlarına ve belirtilen kapsam miktarına sahip olan bir bölge oluşturacaktır. Kapsam kümesi, gönderme uzayı

içindeki bölgenin şeklini çizecektir. Bölge, güncelleme ya da abone olma için kullanılabilir. Şekil 5.26'da bir bölgeyi oluşturma, değiştirme ve silme için gerekli etkileşimler görülmektedir.

5.9.2.2. Bölge Değiştirme(Modify Region)

Bölge Değiştirme servisi, bölgenin kapsam kümesi üzerindeki değişikliklerden RTI'ı haberdar edecektir. Burada bir argüman olarak sağlanan kapsam kümesi, bölgeyi tanımlayan daha önceki kapsam kümesinin tamamen yerini alacaktır.

5.9.2.3. Bölge Silme(Delete Region)

Bölge Silme servisi, belirtilen bölgeyi silecektir. Abone olma ve güncelleme için kullanılmakta olan bir bölge silinemeyecektir.

5.9.2.4. Nesne Örneğini Bölge ile Kaydetme(Register Object Instance With Region)

Nesne Örneğini Bölge İle Kaydetme servisi, bir tek nesne örneği göstergesi oluşturacak ve onu sağlanan nesne sınıfının bir nesne örneği ile birleştirecektir. Uygun sınıf özellikleri kaydeden federe tarafından halen yayınlanmakta olduğu için nesne örneğinin bütün örnek özellikleri, kaydeden federe tarafından sahiplenilmiş olarak saptanacaktır.

Bu servis, bir nesne örneği oluşturmak için kullanılacak ve o nesne örneğinin örnek özellikleri ile güncelleme bölgelerine eş zamanlı bir şekilde ilişkilendirecektir. Eğer bir federe bir güncelleme bölgesiyle ilişkili bir örnek özelliğinin mülkiyetini kaybederse ve daha sonra o örnek özelliğin mülkiyetini tekrar elde ederse, o güncelleme bölgesi artık örnek özellikle ilişkili olmaz.

5.9.2.5. Güncellemeler İçin Bölge İlişkisi Kurma(Associate Region For Updates)

Güncellemeler İçin Bölge İlişkisi Kurma servisi, belirtilen bir nesne örneğinin örnek özellikleri ile güncellemeler için kullanılan bir bölge arasında ilişki kuracaktır.

Bir bölge ile bir örnek özellik arasında ilişki kurmak, *Özellik Değerlerini Güncelleme* servisi talep edildiği an, federenin örnek özelliklerin mülkiyetini ilişki kurulan bölgenin kapsamı içine düşmesini sağlaması anlamına gelecektir.

5.9.2.6. Güncellemeler İçin Bölge İlişkisini Kaldırma(Unassociate Region For Updates)

Güncellemeler İçin Bölge İlişkisini Kaldırma servisi, bölge ve o bölgeyle ilişkili olan bütün örnek özellikler arasındaki ilişkiyi kaldıracaktır.

Bu servis yoluyla ilişkisi kaldırılan örnek özellikler, varsayılan bölge ile ilişkilendirilecektir.

5.9.2.7. Nesne Sınıf Özelliklerine Bölge İle Abone Olma(Subscribe Object Class Attributes With Region)

Bir nesne örneğinin örnek özelliklerinden en az biri kapsam dahilinde olduğu zaman *Nesne Sınıf Özelliklerini Bölge İle Kabul Etme* servisi bir nesne sınıfını belirtecektir, bundan dolayı RTI, nesne örneklerinin keşfinden federeyi haberdar etmeye başlayacaktır. Bu servis ve daha sonraki bununla ilişkili RTI işlemleri, *Nesne Sınıf Özelliklerine Abone Olma* servisi ve yine sonradan gelen ilgili RTI işlemlerine benzer şekilde davranış gösterecektir. Bu servis, ilgili abone olma ve güncelleme bölgelerinin üst üste gelmesi, daha sonraki RTI işlemlerini etkilediği için ek bir işlevsellik sağlayacaktır.

5.9.2.8. Nesne Sınıf Özelliklerine Bölge ile Abone Olmama(Unsubscribe Object Class With Region)

Nesne Sınıf Özelliklerini Bölge ile Abone Olmama servisi, belirtilen bölge içindeki belirtilen nesne sınıfı için nesne örnek keşiflerinden federeyi haberdar etmeyi durdurmasını RTI'ya bildirecektir. Abone olmama, belirtilen bölgeyi kullanan bütün abonelikleri kapatacaktır.

5.9.2.9. Etkileşim Sınıfına Bölge İle Abone Olma(Subscribe Interaction Class With Region)

Etkileşim Sınıfına Bölge İle Abone Olma servisi, bölge dikkate alınarak federeye teslim edilmiş olması gereken etkileşim sınıflarını belirtecektir. Bu servis ve bu servisten sonra gelen RTI işlemleri, *Etkileşim Sınıfını Kabul Etme* servisindekine benzer şekilde olacaktır.

5.9.2.10. Etkileşim Sınıfına Bölge İle Abone Olmama(Unsubscribe Interaction Class With Region)

Etkileşim Sınıfına Bölge İle Abone Olmama servisi, belirtilen bölge içine gönderilen belirtilen etkileşim sınıflarının federeye artık daha fazla bildirilmemesi gerektiğinden RTI'yı haberdar edecektir.

5.9.2.11. Bölge ile Etkileşim Gönderme(Send Interaction With Region)

Bölge ile Etkileşim Gönderme servisi, bir etkileşimi federasyon içine gönderecektir. FED içerisinde tanımlanmış olduğu gibi, etkileşim parametreleri, belirtilen sınıf ve bütün süper-sınıflar içinde olabilir. Bölge, etkileşimin potansiyel alıcılarının kapsamını sınırlandırmak için kullanılacaktır.

5.9.2.12. Özellik Değerini Bölge ile Güncelleme İsteği(Request Attribute Value Update With Region)

Özellik Değerini Bölge ile Güncelleme İsteği servisi, belirtilen özellik değerlerinin güncellenmesini teşvik etmek için kullanılacaktır. RTI, *Özellik Değerinin Güncellenmesini Sağlama†* servisini kullanarak, sahiplerinden belirtilen sınıfın bütün nesne örnekleri için belirli örnek özelliklerin değerlerini isteyecektir.

BÖLÜM 6. SONUÇLAR

HLA'nın modelleme ve simülasyon alanlarında kullanıcılara sağladığı fayda ve kolaylıklar oldukça büyüktür. Fiyat olarak en uygun, birbiriyle uygun bir şekilde çalışabilir ve yeniden kullanılabilir simülasyonların ve modellerin gelişmesinde HLA'nın rolü çok büyüktür. HLA bütün uygulamaları içine alacak şekilde dizayn edilmiştir[15].

Sadece bir tek simülasyonun kullanıcıların bütün isteklerini karşılayamaması ve simülasyonlardan tam bir şekilde yararlanma amacıyla onların bir araya getirilme zorunlulukları gibi bir takım gerekçeler HLA'nın temel dayanak noktalarıdır. Bu aynı zamanda gelecekteki teknolojik yeniliklerin ve çalışma biçimi türlerinin birbirine uyumlu olması fikriyle bağdaşmaktadır. HLA bütün simülasyon tiplerine uygulanabilir olarak tasarlanmıştır. Ancak tabiki asıl hedef HLA'ya uygun simülasyonların geliştirilmesidir. HLA'nın mevcut simülasyonlar üzerinde uygulanabilmesinde bir engel söz konusu değildir, ancak, bilinen bir gerçektir ki, halihazırdaki sistemlerden bazıları HLA'nın yapısına uydurulabilir iken, diğerlerinin sisteme dahil edilebilmeleri çok pahalıya mal olabilir. [14]

HLA tasarımı kesin varsayımlar üzerine kurulmuştur. HLA, özel simülasyonların dizaynındaki kısıtlamaları mümkün olduğunca en aza indirmeye gayret ederken, beklenmeden ortaya çıkan uygulamalar tarafından ihtiyaç duyulan yeni özellikleri destekleyecek şekilde dizayn edilmiştir. HLA'nın üç temel fonksiyonel bileşeni vardır. Bunlardan ilki federelerdir. İkincisi RTI dır. RTI federasyon için dağıtık bir işletim sistemi gibi çalışır. Federeler arasındaki işlemler, RTI yoluyla olur. Üçüncü fonksiyonel bileşen ise arabirimdir. Bu arabirim bağımsız bir uygulamadır ve özel nesne modelleri ve federasyonun veri alış veriş gereksinimlerinden bağımsızdır.

HLA, federelerin arabirimler yoluyla diğer federeler ile veri değişimi yapabilecek şekilde geliştirilmiş olmalarına ihtiyaç duyar. Bu, federelerin bir federasyon ile çalışmalarını düzenlemelerine, veri değişimi yapabilmelerine ve federasyonlara katılabilmelerine olanak sağlar.

Simülasyonların karşılıklı çalışabilirliği konusu, bu konu üzerinde çalışmalar yapan çalışma gruplarının kendi ihtiyaçlarına bağlı olan bir önem derecesine sahiptir. Genel olarak simülasyonların karşılıklı çalışabilirliği ortak bir ortak bir nesne modeli gerektirmelidir.[12]

KAYNAKLAR

[1] Handbook of Simulation, Edited by Jerry BAKS, John Wiley & Sons Inc. A Wiley-Interscience Publication, 1998

[2] Under Secretary of Defense for Acquisition and Technology: "Department of Defense Modeling and Simulation Master Plan, DoD 5000.59-P," October 1995.

[3] High Level Architecture Run-Time Infrastructure Programmer's Guide, RTI 1.3 Version 6. Department of Defense, 12 March 1999

[4] Annotated Briefing on the DoD High Level Architecture for Simulation, Defence Modeling&Simulation Office, April 1997

[5] URL,
http://www.cs.mcgill.ca/~zahra/hla.html

[6] Clavin, James O., Weatherly, Richard. An Introduction to the High Level Architecture, MIT lincoln Laboratory,1996
URL:http://dss.II.mit.edu.tr/dss.web/96.14.103.RTI.Introduction.html

[7] Department of Defense, "Federation Development and Execution Process (FEDEP) Model Version 1.2", May 26, 1998

[8] U.S. Department of Defense: "High Level Architecture Rules, Version 1.3," 5 February 1998.

[9] U.S. Department of Defense: "High Level Architecture Object Model Template Specification, Version 1.3," 5 February 1998.

[10] U.S. Department of Defense: "High Level Architecture Federate Interface Specification, Version 1.3," 20 April 1998.

[11] Dahmann S. Judith, The High Level Architecture and Beyond: Technology Challenges, US Defense Modeling and simulation Office, 1999

[12] Harry M. Wolfson, et. al., "RTI 1.3 Architecture and Implementation," 98S-SIW-202, 1998 Spring Simulation Interoperability Workshop, March 9-13, 1998.

[13] Richard M. Weatherly, et. al., "Steps in the Formalization of the HLA Interface Specification," 98S-SIW-170, 1998 Spring Simulation Interoperability Workshop, March 9-13, 1998.

[14] URL,
http://www.sisostds.org/

[15] FIRAT, C.,"Intelligent Agents for HLA Development and Execution Process", 2nd Midterm Ph.D. Progress Report, Sakarya University, 1999

ÖZGEÇMİŞ

Ahmet Zengin 1975 yılında Sapanca'da doğdu. İlk ve orta öğrenimini Sapanca'da, lise tahsilini Sapanca Lisesinde tamamladı. 1993 yılında girdiği Sakarya Üniversitesi Mühendislik Fakültesi Elektrik ve Elektronik Bölümünden 1997 yılında mezun oldu. 1998 yılında SAÜ Teknik Eğitim Fakültesi Elektronik ve Bilgisayar Eğitimi Bölümü Bilgisayar Bilimleri Ana Bilim Dalına Araştırma Görevlisi olarak atandı. Halen bu görevi sürdürmektedir.