T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK SEVİYELİ MİMARİ (HLA) TEMELLİ DAĞITIK İMALAT BENZETİMİ

DOKTORA TEZİ

End. Yük. Müh. Özer UYGUN

Enstitü Anabilim Dalı : ENDÜSTRİ MÜHENDİSLİĞİ

Tez Danışmanı : Doç. Dr. Cemalettin KUBAT

Aralık 2007

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK SEVİYELİ MİMARİ (HLA) TEMELLİ DAĞITIK İMALAT BENZETİMİ

DOKTORA TEZİ

End. Yük. Müh. Özer UYGUN

Enstitü Anabilim Dalı : ENDÜSTRİ MÜHENDİSLİĞİ

Enstitü Bilim Dalı : ENDÜSTRİ MÜHENDİSLİĞİ

Bu tez 28/12/2007 tarihinde aşağıdaki jüri tarafından Oybirliği ile kabul edilmiştir.

Prof. Dr. Harun TAŞKIN	**Doç. Dr. Cemalettin KUBAT**	**Prof. Dr. İbrahim ÖZSERT**
Jüri Başkanı	**Üye**	**Üye**
Prof. Dr. Ercan ÖZTEMEL		**Prof. Dr. Türkay DERELİ**
Üye		**Üye**

TEŞEKKÜR

Herşeyden önce tez danışmanım sayın Doç. Dr. Cemalettin Kubat'a her türlü desteği için sonsuz teşekkürlerimi sunarım. Sayın Prof. Dr. Ercan Öztemel'e de tez konumun belirlenmesinde ve sonrasında, değerli katkılarından dolayı teşekkürlerimi bir borç bilirim.

Tez izleme jürimde bulunan Bölüm Başkanımız Prof. Dr. Harun Taşkın ve Prof. Dr. İbrahim Özsert'e de yardımlarından ve katkılarından dolayı teşekkür ederim.

HLA konusu ile ilgili kolay erişilemeyecek dokümanları ve HLA-RTI yazılımını sağlayan TÜBİTAK MAM'da çalışan Burak Selçuk Soyer'e teşekkür ederim. Uygulamanın geliştirilmesi esnasında bu bilgiler oldukça faydalı olmuştur.

Çalışmamla ilgili olarak çeşitli fikir alışverişlerinde bulunduğum oda arkadaşım Fuat Şimşir ile C++ ve yazılım geliştirme konusunda değerli bilgisinden faydalandığım arkadaşım Hasan Kaçamak'a da değerli katkılarından dolayı teşekkür ederim.

Çalışmalarım esnasında kendilerine yeterince vakit ayıramadığım değerli anneme, eşime ve diğer aile fertlerime de sabır ve hoşgörülerinden dolayı sonsuz teşekkür ederim.

İÇİNDEKİLER

KISALTMALAR LİSTESİ

ABD : Amerika Birleşik Devletleri
AGV : Automated Guided Vehicle (Otomatik Yönlendirmeli Araç)
ALSP : Aggregate Level Simulation Protocol (Tümleşik Seviye Benzetim Protokolü)
AMG : Architecture Management Group (Mimari Yönetim Grubu)
API : Application Program Interface (Uygulama Programı Arayüzü)
BOM : Base Object Model (Temel Nesne Modeli)
COTS : Commercial off-the-shelf (Ticari Paket Program)
DARPA : Defense Advanced Research Projects Agency (ABD Savunma Bakanlığı İleri Araştırma Projeleri Birimi)
DIF : Data Interchange Format (Veri İletim Formatı)
DIS : Distributed Interactive Simulation (Dağıtık Etkileşimli Benzetim)
DLL : Dinamic Link Library (Dinamik Link Kütüphanesi)
DMS : Distributed Manufacturing Simulation (Dağıtık İmalat Benzetimi)
DMSO : Defense Modeling & Simulation Office (ABD Savunma Bakanlığı Modelleme ve Benzetim Birimi)
DoD : Department of Defense (ABD Savunma Bakanlığı)
FED : Federation Execution Data (Federasyon İşletim Verileri)
FEDEP : Federation Execution and Development Process (Federasyon Çalıştırma ve Geliştirme Süreci)
FOM : Federation Object Model (Federasyon Nesne Modeli)
HLA : High Level Architecture (Yüksek Seviyeli Mimari)
IDL : Interface Definition Language (Arayüz Tanımlama Dili)
IEEE : Institute of Electrical and Electronics Engineers (Elektrik ve Elektronik Mühendisleri Enstitüsü)
JIT : Just-in-time (Tam Zamanında)
MOM : Management Object Model (Yönetim Nesne Modeli)

NATO	: North Atlantic Treaty Organisation (Kuzey Atlantik İttifak Organizasyonu)
OMDT	: Object-Model Development Tool (Nesne Modeli Geliştirme Aracı)
OMT	: Object Model Template (Nesne Model Şablonu)
ör.	: Örneğin
PDU	: DIS Protocol Data Units (DIS Protokolü Veri Birimleri)
RID	: RTI Initialization Data (RTI Başlangıç Verileri)
RTI	: Run-Time Infrastructure (Çalışma Anı Altyapısı)
SIMNET	: Simulator Networking (Simülatör Ağı)
SISO	: Simulation Interoperability Standards Organization (Benzetim Karşılılklı İşlerlik Standartları Organizasyonu)
SOM	: Simulation Object Model (Benzetim Nesne Modeli)
vd.	: Ve Diğerleri
vs.	: Vesaire
XML	: Extensible Markup Language (Genişletilebilir Biçimleme Dili)

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar Kelimeler: Yüksek Seviyeli Mimari, HLA, Dağıtık İmalat Benzetimi

Benzetim, bir karar verme aracı olarak her zaman kullanılagelmiştir. Benzetimin en çok kullanıldığı alanların başında imalat gelmektedir. Ancak geleneksel benzetim teknikleri tek başına karmaşık dağıtık imalat problemlerini modellemeye yetmemektedir. Son yıllarda imalat süreci oldukça karmaşık hale geldiğinden ve çoğu zaman dağıtık ortamlarda gerçekleştirildiğinden dağıtık imalat benzetimi modellerine ihtiyaç duyulmaktadır. Farklı benzetim modellerini bir araya getirerek bilgi alışverişini gerçekleştirecek ortak mimariye olan ihtiyaç, başlangıçta askeri alanlarda ortaya çıkmıştır. Benzer ihtiyac, dağıtık imalat benzetiminde de hissedilmektedir. Böyle bir mimari benzetim modellerinin yeniden kullanılabilirlik ve karşılıklı işleyebilirlik özelliklerine sahip olmasını sağlayabilmelidir. Bununla birlikte, bilgi alışverişinde bulunacak olan benzetim modelleri farklı platformlarda, farklı ticari benzetim paketleriyle veya farklı uygulama yazılımları ile modellenmiş olabilmektedir. HLA, bu tür problemleri çözmekte kullanılabilecek dağıtık benzetim mimarisidir. Amerikan Savunma Bakanlığı tarafından askeri benzetim sistemlerinde kullanılmak üzere 1995'lerden itibaren geliştirilen bu mimari, imalatta ve diğer sivil alanlarda da uygulanmaktadır. 2000 yılında IEEE tarafından 1516 kodu ile dağıtık benzetim standardı olarak kabul edilmesiyle sivil alanlarda kullanımı yaygınlaşmıştır.

Bu çalışmada HLA temelli dağıtık imalat benzetimi ele alınmıştır. Dağıtık imalat benzetiminin yararları, HLA'nın imalatta kullanım gerekçesi, HLA temelli dağıtık imalat benzetimi geliştirmek için neler yapılması gerektiği açıklanmıştır. Çeşitli HLA temelli dağıtık imalat tasarım örnekleri verilmiştir. Örnek bir uygulama ile HLA'nın pratik olarak dağıtık imalat benzetiminde nasıl uygulandığı ortaya konulmuştur.

HIGH LEVEL ARCHITECTURE (HLA) BASED DISTRIBUTED MANUFACTURING SIMULATION

SUMMARY

Key Words: High Level Architecture, HLA, Distributed Manufacturing Simulation

Manufacturing is one of the areas where simulation is used more widely as a decision making tool. But traditional simulation techniques are not capable to simulate complex manufacturing systems. Since manufacturing processes became more complex and mostly performed in distributed environments, distributed manufacturing simulation models are needed. So, a common framework is required to integrate and exchange information of different manufacturing simulation models. The framework should have the capability of reusability and interoperability since simulation models could be modeled at different environments with different application programming languages. HLA can meet all of the requirements mentioned above. HLA was developed by American Department of Defense (DoD) since 1995 for military simulations, and after it was accepted as an IEEE standart (No:1516) in year 2000, HLA is being used not only for military simulations but also for civil applications.

In this study HLA based distributed manufacturing simulation is examined. The benefits of distributed manufacturing simulation, reasons why HLA is used in manufacturing, what sould be done in order to develop HLA based distributed manufacturing simulation is explained in the study. It will also be put forward how HLA is implemented practically in distributed mandufacturing simulation through a given scenario.

BÖLÜM 1. GİRİŞ

1.1. Giriş

Benzetim, bir karar verme aracı olarak her zaman kullanılagelmiştir. Benzetimin en çok kullanıldığı alanların başında imalat gelmektedir. Ancak geleneksel benzetim teknikleri tek başına karmaşık dağıtık imalat problemlerini modellemeye yetmemektedir. Son yıllarda imalat süreci oldukça karmaşık hale geldiğinden ve çoğu zaman dağıtık ortamlarda gerçekleştirildiğinden dağıtık imalat benzetimi modellerine ihtiyaç duyulmaktadır. Dağıtık imalat sürecini modelleyebilmek için coğrafi olarak farklı yerlerdeki imalat benzetim modellerinin etkileşmesi gerekebileceği gibi aynı ortamdaki farklı imalat alt sistemlerinin benzetim modellerinin de birlikte çalışabilmesi önemlidir.

Farklı benzetim modellerini bir araya getirerek bilgi alışverişini gerçekleştirecek ortak mimariye olan ihtiyaç, başlangıçta askeri alanlarda ortaya çıkmıştır. Benzer ihtiyaç, dağıtık imalat benzetimi olarak hissedilmektedir. Böyle bir mimari benzetim modellerinin yeniden kullanılabilirlik ve karşılıklı işleyebilirlik özelliklerine sahip olmasını sağlayabilmelidir. Bununla birlikte, bilgi alışverişinde bulunacak olan benzetim modelleri farklı platformlarda, farklı ticari benzetim paketleriyle veya farklı uygulama yazılımları ile modellenmiş olabilmektedir. HLA, bu problemleri çözmekte kullanılabilecek dağıtık benzetim mimarisidir.

Amerikan Savunma Bakanlığı tarafından askeri benzetim sistemlerinde kullanılmak üzere 1995'lerden itibaren geliştirilen bu mimari, imalatta ve diğer sivil alanlarda da uygulanmaktadır. 2000 yılında IEEE tarafından 1516 kodu ile dağıtık benzetim standardı olarak kabul edilmesiyle sivil alanlarda kullanımı yaygınlaşmıştır.

Bu çalışmada HLA temelli dağıtık imalat benzetimi ele alınmıştır. Dağıtık imalat benzetiminin yararları, HLA'nın imalatta kullanım gerekçesi, HLA temelli dağıtık imalat benzetimi geliştirmek için neler yapılması gerektiği açıklanmıştır. Çeşitli HLA temelli dağıtık imalat tasarım örnekleri ele alınmıştır. Örnek bir uygulama ile HLA'nın pratik olarak nasıl uygulandığı ortaya konulmuştur.

RTI (çalışma anı altyapısı), HLA'nın omurgasıdır. HLA'nın özellikleri ve servisleri RTI üzerinden çalışmaktadır. İçerisinde RTI barındıran imalat benzetimi geliştirme ortamı henüz ticari yazılım paketi olarak var olmadığından RTI ile benzetim geliştirme ortamları günümüzde bütünleşik değildir (Taylor vd., 2006; McLean vd., 2005; Mönch vd., 2003). Bu çalışmada, ABD Savunma Bakanlığı Modelleme ve Benzetim Ofisi (DMSO) tarafından geliştirilen RTI1.3NG-V6 HLA çalışma anı altyapısı üzerinden çalışan, Microsoft Visual C++ ortamında örnek bir HLA temelli dağıtık imalat benzetimi geliştirilmiştir.

McLean vd. (2005), HLA temelli ya da HLA'yı destekleyen ticari bir dağıtık imalat benzetim yazılımının henüz geliştirilmediğini belirtmektedirler. Mönch vd. (2003), atölye kontrol sistemlerine ve optimize edici yazılımlara da henüz karşılıklı işlerlik özelliği kazandırılamadığını belirtmişlerdir. Bu sebeple dağıtık imalat benzetimi tasarımcısı ve geliştiricisi, programlama dilleri ile veya benzetim paketlerini kullanarak dağıtık benzetim geliştirmek için oldukça çaba harcamak zorunda kalmaktadır. Bu ise dağıtık benzetimin, özelde ise HLA'nın yaygınlaşmasındaki en büyük engellerdendir. Kullanım zorluğu ve zaman gerektiren dağıtık benzetim geliştirmedeki bu durumun önümüzdeki yıllarda giderilmesi umulmaktadır.

HLA-RTI'ın bilgisayarda kullanılması için, RTI yazılımının kurulma işlemlerinin dışında bilgisayarda çeşitli ayarların yapılması gerekmektedir. Bu çalışmada gerekli hazırlıklar ve düzenlemelerden de bahsedilecektir. Bunun dışında RTI'ın özelliklerinin, dolayısıyla HLA servislerinin, benzetim geliştirme ortamı olarak kullandığımız Visual C++ içerisinde kullanılabilir hale getirilmesi de gerekmektedir. İlerleyen bölümlerde uygulamalı olarak ilgili işlemler ayrıntılı bir şekilde verilecektir.

1.2. Dağıtık İmalat Benzetiminin Gerekliliği

İmalat endüstrisinde pazar küresel hale gelmiştir. Karmaşık ürünler artık tek bir fabrika tarafından üretilmemekte; bileşenler farklı şirketlerce üretilmekte ve başka bir yerde montajı yapılmaktadır. Bu, şirketlerin birbirlerine olan bağımlılığını artırmaktadır ve sistemlerin incelenebilmesi için benzetim modellerinin dağıtık hale getirilmesini gerektirmektedir.

İmalat sistemleri birçok alt sistemden meydana gelmektedir. Her alt sistem tekil olarak tasarlanıp eş zamanlı olarak optimize edilmeye çalışılmaktadır. Her alt sistem belirli bir amaç için kullanıldığından modellenme yöntemleri farklıdır. Günümüzde farklı modellenmiş alt sistemleri bir araya getirmek, benzetim modellerinin tanımlanmasında bir standardın olmamasından ve standart bir benzetim modelleme dili geliştirilmediğinden dolayı oldukça güçtür. Bu sebeple farklı benzetim modelleri kullanıldığında tasarımcılar işbirliği yapmakta zorlanmaktadır. Tüm alt sistemleri senkronize çalıştırmakla sistemin tamamını incelemek kolay değildir. Bu gibi sebeplerden dolayı dağıtık benzetim kullanılmaktadır. Dağıtık benzetim, farklı benzetim modellerini birbirine bağlayarak ve senkronize ederek bir benzetim sisteminin çalıştırılmasıdır. Bununla birlikte dağıtık benzetim ile ilgili pek çok araştırma makalesinin olduğu, ancak bunlarda belirtilen fikirlerin endüstride uygulama alanı bulma sayısının az olduğunu da tespit edilmiştir (Hibino vd., 2002).

Dağıtık imalat benzetimi kullanma sebeplerinden bazıları şunlardır:

- Bilgisayarın işlem yükünü azaltarak benzetimin performansını artırmak,
- Tasarlanmış bir modeli yeniden kullanabilmek, modülerlik sağlamak,
- Benzetim modellerinin etkileşimini, karşılıklı işlemesini sağlamak,
- Farklı benzetim dillerinin özelliklerinden yararlanmak,
- Çeşitli benzetim programlarının farklı özelliklerinden yararlanmak,
- Farklı platformların özelliklerinden faydalanmak,
- Benzetim temelli olmayan programları dağıtık benzetimle bütünleştirerek o programların özelliklerini dağıtık benzetime kazandırmak,
- Dağıtık çalışma imkânı sağlamak ve dağıtık işbirliği oluşturabilmek,

- Sisteme uzaktan erişim imkânı kazandırmak.
- Dağıtık imalat ile ilgili teknolojilerin gelişimine katkıda bulunmak, sanal imalat ortamlarının oluşturulmasına ön ayak oluşturarak teknolojinin itilmesini sağlamak,
- Tekil benzetim ile gerçek imalat sisteminin dahil edemediğimiz başka özelliklerini dağıtık benzetimin karşılıklı işlerliği ile dahil etmek ve bu şekilde gerçek sistemi daha fazla temsil etmek.

1.3. Yüksek Seviyeli Mimari (HLA) Kullanımının Gerekliliği

Dağıtık benzetimlerin birçoğu farklı türdeki benzetimlerin birleşiminden meydana gelmektedir. Bu benzetimlerin yazılımı, çalıştığı ortam, ele aldığı konular farklıdır. Farklı amaçlar için tasarlanmış bu benzetimlerin yeni bir benzetim için yeniden kullanılması gerekebilmektedir. Ancak maalesef, bu benzetimlerin daha büyük bir benzetimin parçası olabilecek hale getirilmesi için birçok değişikliğe uğraması gerekecektir. Bazı durumda değişiklik yapmaktansa sistemi yeniden tasarlamak daha yerinde olmaktadır. Bu sebeple geleneksel sistemler, yeniden kullanılabilirlik ve karşılıklı işleyebilirlik özellikleri bakımdan yetersizdir. Bu özellikler HLA'nın çıkış noktalarıdır (MEOSS, 2006).

Yeniden kullanılabilirlik, benzetim modellerinin farklı benzetim senaryolarında ve uygulamalarında, çok fazla değişiklik gerektirmeksizin tekrar kullanılabilmesidir. Bu özellik sayesinde benzetimler hem daha kısa sürede hem de daha az maliyetle oluşturulmaktadır. Karşılıklı işlerlik, yeniden kullanılabilirlik özelliğiyle yakından ilgilidir. Karşılıklı işleyebilme, benzetimlerin bir araya getirilmesiyle, karşılıklı veri iletimi ve etkileşimler neticesinde birlikte çalışabilmesi demektir. Bu özellik sayesinde farklı platformlarda dağıtık olarak çalışan benzetimler gerçek zamanlı olarak birleştirilebilmektedir (Dahmann vd., 1999b).

HLA'nın amacı, yeniden kullanılabilirliği ve karşılıklı işlerliği desteklemektir. Karmaşık benzetim sistemlerin etkinliğinin artması için en önemli gereksinimler bunlardır. Bunlar dışında ölçeklenebilirlik özelliği de önemlidir. Ölçeklenebilirlik, dağıtık benzetim sistemine yeni tekil benzetim modellerinin dâhil olması veya

sistemden ayrılması ile tüm sistemin çalışmasının devam edebilmesidir (Xiaoxia ve Qiuhai, 2003).

Dağıtık benzetim sistemleri kurmak için çeşitli mimariler geliştirilmiştir. HLA, dağıtık benzetim için geliştirilmiş en sonuncu mimaridir. Başlangıçta askeri amaçlı benzetimler için geliştirilen HLA, zamanla sivil alanlarda ve imalatta da uygulanmaya başlanmıştır. IEEE tarafından 2000 yılında 1516 kodu ile dağıtık benzetim standardı olarak kabul edilmesinin ardından kullanım yaygınlığı artmaya başlamıştır.

İmalat süreçlerinin gün geçtikçe karmaşık hale gelmesi ve coğrafi olarak dağıtık olmakla birlikte önemli işbirlikleri ile gerçekleştiriliyor olması, dağıtık imalat süreçlerinin modellenmesinde ve analizinde HLA'nın kullanımını önemli hale getirmektedir.

Tedarik zincirleri, dağıtık imalat için tipik bir örnektir. Dolayısıyla tedarik zincirleri HLA'nın en önemli uygulama alanı olmaya adaydır. Terzi ve Cavaliery (2004), tedarik zinciri bağlamında benzetimi ele aldıkları araştırmalarında HLA'nın tedarik zincirlerinde kullanılabileceğini belirtilmişlerdir. Bandinelli vd. (2006), tedarik zincirlerinde benzetimin kullanılmasını araştırmışlardır. Dağıtık tedarik zinciri benzetimlerinin, tedarik zinciri planlama ve optimizasyonunda etkin bir araç olabileceğini ifade etmişlerdir. Dağıtık tedarik zinciri benzetimlerini mümkün kılan teknoloji olarak da HLA'yı açıklamışlardır.

1.4. Tezin Amacı

Bu çalışmanın amacı, yeniden kullanılabilen ve birlikte çalışabilen dağıtık imalat benzetim sistemlerinin oluşturulmasını sağlamaktır. Sınırlı kaynaklardan dolayı eldeki imkânların en etkin kullanılmasını sağlamak amacıyla farklı yerlerde bulunan sistemlerin kullanılması ihtiyacı vardır. Tüm imalat fonksiyonlarında olduğu gibi benzetim çalışmalarında da bu gereksinim kaçınılmazdır. Dolayısıyla farklı yerlerde bulunan benzetim sistemlerini birlikte kullanabilmek (interoperability), aynı zamanda bu sistemleri tekrar tekrar kullanmak (reusability) benzetime dayalı

problem çözme etkinliğini dikkate değer şekilde artırmaktadır. Geleneksel benzetim sistemleri bu anlamda yetersizdir.

HLA, sivil alanlarda kullanılıyor olması bakımından yeni bir konu olduğu için HLA temelli dağıtık imalat benzetimi konusunda yapılan çalışma sayısı çok değildir. Bu çalışmaların bir kısmı sistem tasarımı niteliğindedir. Hibino vd. (2002), dağıtık benzetim ile ilgili pek çok araştırma makalesinin olduğunu, ancak bunlarda belirtilen fikirlerin endüstride uygulama alanı bulma sayısının az olduğunu tespit etmiştir. Dolayısıyla bu çalışmada, HLA'nın imalatta kullanılmasına yönelik çeşitli tasarım örnekleri verilmiştir. HLA'nın imalatta kullanılabilirliğini göstermek bakımından da bir senaryo üzerinden uygulamalı bir dağıtık imalat benzetimi ele alınmıştır.

İmalat birçok sistemin karşılıklı işlerliğini ve etkileşimini gerektirmektedir. Klaslik benzetim teknikleri ile bunları gerçekleştirmek mümkün görünmemektedir. Örneğin üretim çizelgeleme ile bakım çizelgeleme birbirlerinden bağımsız düşünülmemelidir. Bu çalışmada verilen tasarım örneklerinden birinde bu iki çizelgelemenin bütünleştirilmesinde HLA'nın kullanılması önerilmektedir. Başka imalat sistemlerinin bütünleştirilmesinde bu örnek ve verilen diğer örnekler fikir verici niteliğinde olacaktır.

Tedarik zincirleri de bütünleştirilmeyi, karşılıklı bilgi alışverişini ve karşılıklı işlerliği gerektirmektedir. Bu çalışmada tedarik zincirlerinin HLA ile karşılıklı işleyişi, uygulamalı olarak geliştilen dağıtık imalat benzetimi senaryosu ile gösterilmeye çalışılmıştır.

1.5. Tezin İçeriği

Bu çalışma altı ana bölüm ve eklerden oluşmaktadır. Birinci bölüm giriş bölümüdür. Burada dağıtık imalat benzetiminin gerekliliği, HLA kullanımının gerekliliği, tezin amacı ve içeriği belirtilmiştir. İkinci bölümde ise benzetimden kısaca bahsedilerek dağıtık benzetim konusu, gelişimi ile birlikte ele alınmış, imalatta dağıtık benzetimin kullanılmasından ve öneminden bahsedilmiştir.

HLA konusu üçüncü bölümde detaylı bir şekilde anlatılmıştır. HLA ile ilgili kavramlar, teknik bileşenleri, HLA'nın servisleri, RTI'ın özellikleri, HLA nesne model şablonu, federasyon geliştirme ve çalıştırma süreci gibi konular ele alınmıştır. Üçüncü bölümün devamında HLA'nın imalatta kullanımıyla ilgili literatür çalışması verilmiştir. HLA'nın kullanıldığı başka uygulama alanları ve HLA ile birlikte kullanılan diğer bilimsel konular ile ilgili literatür çalışması da burada verilmiştir. Daha sonra dağıtık bir imalat benzetimi mimarisi açıklanmıştır.

Dördüncü bölümde HLA temelli dağıtık imalat sistemi tasarımı örnekleri verilmiştir. Beşinci bölümde ise örnek bir uygulama üzerinden HLA temelli dağıtık imalat benzetiminin geliştirilmesi ele alınmıştır. Oluşturulan federasyon, içerisindeki federeler ve bunların karşılıklı etkileşimi ayrıntılı bir şekilde anlatılmıştır.

Altıncı bölümde ise sonuçlar ve öneriler tartışılmıştır. Bu çalışmadan elde edilen çıktılar ve gelecekte yapılması gerekenler ifade edilmiştir. HLA ile birlikte çalışılabilecek diğer konular belirtilmiştir.

Ekler kısmında ise geliştirilen örnek federasyon uygulamasının dosyaları ve kaynak kodları verilmiştir. Böylece HLA ile ilgilenmek isteyenlere daha fazla katkı sağlamak amaçlanmıştır.

BÖLÜM 2. BENZETİM VE İMALAT

2.1. Giriş

Benzetim birçok alanda olduğu gibi imalatta da oldukça fazla uygulama alanı bulmaktadır. Klasik benzetim ile çeşitli imalat problemleri analiz edilmekte ve karar verme aracı olarak kullanılmaktadır. Ancak günümüzün karmaşık ve coğrafi olarak dağıtık imalat ortamlarını klasik benzetim yöntemleri ile modellemek artık yetersiz kalmaktadır. Dolayısıyla dağıtık benzetim tekniklerini imalat için kullanma gereksinimi ortaya çıkmaktadır.

Bu bölümde benzetim ele alınarak imalat için yararlarından bahsedilecektir. Dağıtık benzetim de benzer şekilde incelenecek ve gelişimi ele alınacaktır. İmalatta dağıtık benzetimin kullanımı ve önemi de belirtilecektir. Dağıtık benzetimin birçok faydasının yanı sıra, kullanımının birtakım zorlukları da olabilmektedir. Bu bölümün sonunda dağıtık benzetimin muhtemel zorluklarından da bahsedilecektir.

2.2. Benzetim Nedir

Benzetim gerçek hayattaki süreçlerin ve operasyonlarının zamana göre taklit edilmesidir. Elle veya bilgisayarla benzetim yaparken sistemin yapay bir geçmişi oluşturulur ve oluşturulan yapay geçmiş gerçek sistemin işleyiş özellikleri hakkında yorum yapmak için kullanılır (Law ve Kelton, 1991).

Bir sistemin benzetimi, bu sistemi temsil edebilecek bir model oluşturma işlemidir. Bu model, temsil ettiği sistem üzerinde yapılması çok pahalı olan veya mümkün gözükmeyen işlemlerin yapılmasına imkân verir. Bu işlemlerin etkisi altındaki model incelenir. Bundan gerçek sistemin veya ona ait alt sistemlerin davranışları ile ilgili özellikler ve tepkiler öngörülür (Erkut, 1992).

Bazı durumlarda problemi çözmek ve analiz etmek için basit matematik modeller yeterlidir. Bu modeller diferansiyel denklemler, olasılık teorisi, cebir metotları ve diğer matematiksel tekniklerdir. Bazı modellerin basit analitik yöntemlerle çözülmesi mümkün olmadığından bu durumlarda sistemi taklit eden bilgisayar temelli nümerik benzetim modelleri kullanılır. Benzetimden gerçek sistem gözleniyormuş gibi veriler toplanır ve bu yolla üretilen veriler sistemin performansını tahmin etmede kullanılır (Law ve Kelton, 1991).

Modelleme ve Benzetim bağlamında kullanılan bir takım terimlerin anlamları aşağıda verilmiştir.

Model: Sistemin, varlığın veya bir sürecin fiziksel, matematiksel ya da mantıksal gösterimidir.

Benzetim: Modelin bir zaman boyunca uygulanması yöntemidir.

Modelleme ve Benzetim: Modelin, prototipin ve simülatörlerin istatistiksel olarak veya bir zaman boyunca kullanılması ile yönetimsel ve teknik kararlar için verilerin üretilmesidir. Modelleme ve benzetim terimleri bazen birbirinin yerine kullanılmaktadır.

Simülatör: Benzetim modelini uygulayan araç, bilgisayar programı veya sistemdir.

Oyun: Önceden tanımlanmış kaynaklar ve kısıtlar altında katılımcıların belirli amaçları gerçekleştirmeye çalıştıkları bir benzetim oturumudur. Bu benzetim oturumunda katılımcılar stratejik kararlar verirler ve bilgisayar da bu kararların sonuçlarını belirler. Oyun terimi yapısal benzetim veya yüksek seviye model ile aynı anlamdadır.

Monte Carlo Benzetimi: Rassal istatistiksel örnekleme şemalarının kullanıldığı ve elde edilen sonuçlarla bilinmeyen değerlerin tahminlerinin belirlendiği benzetim türüdür.

Sanal Benzetim: Canlı kullanıcıların, benzetim modeli kurulmuş sistemleri kullandığı ve sisteme dâhil olduğu benzetimdir. Sanal benzetimde canlı kullanıcı genellikle merkezi konumdadır. Yeteneğini geliştirme, bir aracı kullanmayı öğrenme gibi (uçak kullanma, ameliyat gerçekleştirme) amaçlar için, karar verme yeteneğini geliştirmek için (bir faaliyete kaynak atama verimliliği) kullanılabilmektedir.

Canlı Benzetim: Gerçek/canlı insanların gerçek sistemleri kullandığı ve etkileşim halinde olduğu benzetim sistemleridir.

Yapısal Benzetim: Benzetim modeli kurulmuş kişilerin benzetim modeli kurulmuş sistemleri işlettiği benzetimlerdir. Gerçek insanlar bu benzetimlerde girdi sağlar ancak çıktıların belirlenmesine müdahil değillerdir.

2.3. Benzetimin Avantajları

Benzetimin çok çeşitli avantajları olabilmektedir. Bunlardan bazıları aşağıda belirtilmiştir:

- Yeni politikalar, işletme prosedürleri, karar kuralları, bilgi akışı, organizasyonel prosedürler, vb., sistemin devam eden işleyişini aksatmadan incelenebilir.
- Yeni donanım tasarımı, fiziksel yerleşim, ulaştırma sistemleri, kaynakları kullanmaya başlamadan test edilebilir.
- Bir şeyin nasıl ve niye olduğu hakkında hipotezler test edilebilir.
- Değişkenlerin etkileşimi hakkında bilgi elde dilebilir.
- Değişkenlerin sistemin performansına etkisi ve önemleri hakkında bilgi elde edilebilir.
- İşlem aşamasındaki ürünler, enformasyon ve malzeme gibi varlıkların nerelerde fazlaca beklediğini gösteren darboğaz analizleri yapılabilir.
- Benzetim çalışması, bireylerin sistem hakkında ne düşündüklerinden ziyade sistemin gerçekte nasıl çalıştığının anlaşılmasına yardımcı olur.

- Benzetimle değişik senaryolara "Eğer-Ne" analizleri ile cevap bulunabilir. Bu özellikle yeni bir sistem tasarlarken önemlidir.
- Zaman sıkıştırılıp genişletilebilir. İşlem yavaşlatılarak veya hızlandırılarak araştırılan şeyin durumu istenilen şekilde incelenebilir.

2.4. İmalatta Benzetimin Faydaları

İmalatta benzetim, yeni üretim tesislerinin, depoların ve dağıtım merkezlerinin tasarlanmasında başarılı bir şekilde kullanılmaktadır. Benzer şekilde mevcut sistemde önerilen değişikliklerin incelenmesinde de kullanılmaktadır. Benzetimi kullanan mühendisler ve analistler, teçhizata ve fiziksel imkanlara yatırımın, malzeme tutma ve tesis yerleşiminde önerilen değişikliklerin etkisinin incelenmesinde de benzetimin yararlı olduğunu görmüşlerdir. Aynı zamanda işçiliğin ve operasyon kurallarının incelenmesinde, depo yönetimi kontrol yazılımının ve üretim kontrol sisteminin içerisinde olması önerilen kuralların ve algoritmaların test edilmesinde de oldukça yararlı olduğu belirlenmiştir. Yöneticiler de mevcut sistemi denenmemiş değişiklikler ile karmaşaya sürüklememek için benzetimi bir test aracı olarak kullanmayı faydalı bulmaktadırlar (Banks vd., 2001).

İmalat, benzetimin en çok kullanıldığı alanların başında yer almaktadır. Bunun sebeplerinden bazıları şunlardır (Law ve Kelton, 2000):

- Birçok endüstride artan rekabet, kaliteyi ve üretkenliği artırıcı etkisinden dolayı otomasyonun önemini artırmıştır. Otomatik hale getirilmiş sistemler çok karmaşık olduklarından sadece benzetim ile analiz edilebilmektedirler.
- Ekipman ve tesis maliyetleri oldukça yüksektir. Bu sebepten dolayı, yeni tesis kurulmadan önce benzetim ile test ve analizi gerçekleştirilmelidir.
- Bilgisayarlarla çalışma maliyeti, hızlı ve ucuz bilgisayarlar ile oldukça azalmıştır.
- Benzetim yazılımlarındaki iyileşmeler (ör. kullanıcı arayüzlerinin grafiksel hale gelmesi) model geliştirme sürelerini azaltmış, bu ise, imalat analizlerinin zamanında yapılmasına daha fazla imkan sağlamıştır.

- Animasyon imkânları ile benzetimin imalat yöneticileri tarafından anlaşılması ve kullanılması artmıştır.

Benzetimin gerçek faydalarından biri, sistemi kurmadan önce alternatiflerini inceleme fırsatı vermesidir. Benzetim hem endüstride hem de akademik çalışmalarda, karmaşık imalat sistemlerinin analizinde kullanılan standart bir araç haline gelmiştir. Bunun sebebi aslında basittir: İmalat sistemlerini tam olarak tanımlayabilecek bir matematiksel model kurmak neredeyse mümkün değildir. İmalat sistemlerini modellemede kuyruk teorisini kullanmak önerilmiştir. Böylece, kapasite, darboğaz, kapasite kullanımı gibi sorulara cevap aranmaktadır. Fabrika benzetimi için kullanıcı dostu yazılımların artmasıyla birlikte problemlerin araştırılması ve çözüm önerilerinin incelenmesinde benzetimin kullanım yaygınlığı da artacaktır. Bu tür yazılımlar birçok istatistiksel dağılımların kullanılmasını sağladığı gibi kullanıcı kendi dağılımını da tanımlayabilmektedir. Bunlarla bir model kurmak, farklı senaryolara uyarlamak ve eğer-ne soruları için modeli değiştirmek oldukça basittir. Böylece birçok performans ölçümleri de otomatik olarak üretilebilmektedir (Parsaei vd., 1997).

Benzetim metotlarındaki gelişmeler, daha ucuz maliyete sahip daha yüksek performanslı bilgisayarların gelişimi ve özel amaçlı benzetim dillerinin geliştirilmesi benzetimi, yöneylem araştırmasında ve sistem analizleri başta olmak üzere tüm bilimsel disiplinlerde en fazla kullanılan araçlardan biri yapmıştır.

Bir imalat ortamında benzetimin kullanılmasının en önemli faydalarından bir diğeri de, kısmi değişikliklerin imalat sistemine etkilerinin analiz edilmesini sağlamasıdır. Belirli bir iş istasyonunda yapılan değişikliğin tüm sisteme etkisi böylece tahmin edilebilir. Bunun dışında, benzetimin yukarıda listelenen avantajları ışığında imalata sağladığı faydalar aşağıdaki gibi sıralanabilir:

- Artan mamul akışı,
- Proses içi parça envanterinin azalması,
- Makinelerin veya işçilerin verimli kullanımının artırılması,

- Ürünün müşteriye tam zamanında dağıtımının sağlanması,
- Finansal ihtiyaçların veya işletme masraflarının azalması,
- Önerilen sistemin beklendiği gibi çalışıp çalışmadığının test edilmesi,
- Benzetim modelini oluşturmak için toplanan bilgilerin, sistemin daha iyi anlaşılmasını sağlaması.

2.5. Dağıtık Benzetim

Dağıtık benzetim, dağıtık bilgisayarcılık teknolojileri ile geleneksel sıralı benzetim tekniklerini birleştirmektedir (Saad vd., 2003). Dağıtık benzetim, bilgisayar benzetimlerinin çok işlemcili bilgisayarlarda çalıştırılmasıdır. Bu, çok işlemcili bir bilgisayarda gerçekleştirilebileceği gibi bir ağ ile birbirine bağlanmış bilgisayarlar arasında da gerçekleştirilebilmektedir. Dağıtık benzetim işlem süresinde oldukça kazanç sağlamaktadır. Çok büyük modellerde ve sanal ortamlarda hız çok önemli olduğundan dağıtık benzetim zaman tasarrufu sağlamaktadır. Coğrafi olarak farklı yerlerdeki kullanıcıları bir araya getirebilmekte ve farklı veri tabanlarını kullanmak, dağıtık benzetim ile sağlanabilmektedir (Fujimoto, 2000).

Hibino vd., (2002), Fujimoto'nun dağıtık benzetimi, farklı simülatörleri birbirine bağlayarak ve senkronize ederek bir benzetimin çalıştırılması olarak tanımladığını kaydetmektedir. Dağıtık benzetimde modeli kurulacak sistem alt öğelerine ayrılmakta ve birbirleriyle bağlantılı iş istasyonlarında benzetim modeli çalıştırılmaktadır (Saad vd., 2003).

Dağıtık benzetimi tanımlarken, belirli işleri gerçekleştirmek için birbirine bağlanmış bilgisayarlar olarak düşünebiliriz. Burada önemli olan, ortak amaçların gerçekleştirilmesidir. Bu amacı gerçekleştirmek için birtakım işlevlerin yerine getirilmesi gerekmektedir. Dağıtık benzetimde bu işlevler kısmen veya tamamen katılımcı benzetimler tarafından yerine getirilmektedir. Benzetimin işlevlerini dağıtık olarak yerine getirmenin bazı faydaları vardır (Wicox vd., 2000):

Performans yükünün dengelenmesi: Bilgisayar işlem yükünün birçok bilgisayara dağıtılması sistemin tamamının performansı açısından yararlıdır. Bu durumda her katılımcı benzetim, tüm yükün belirli bir kısmını yüklenecektir.

Yapılmış bir modelden faydalanmak: Bir benzetimde gerekli olan bir işlev, daha önceden başka bir bilgisayardaki benzetim modeli ile gerçekleştirilmiş olabilmektedir. Bu durumda, benzetimin bu işlevi diğer bilgisayardan elde edilmek üzere bilgisayarlar birbirine bağlanabilir. Benzetim yazılımının veya donanımının taşınıp diğer bilgisayara kurulması her zaman mümkün olmamaktadır. Diğer bilgisayarda var olan ve gerekli bir işlevi gerçekleştirebilen benzetimin yeniden yazılması da ilave çaba gerektirdiğinden bilgisayarların birbiriyle bağlantılı hale getirilmesi bir çözüm olmaktadır.

Teknolojinin itilmesi: Bu kavram genellikle teknoloji geliştirmede kullanılmaktadır. Burada amaç, bir şeyin mümkün olduğunu ispatlamak ve teknolojinin yeteneklerini ve sınırını test etmektir. Dağıtık benzetim konusundaki çalışmalar da teknolojinin itilmesi ve geliştirilmesine katkı sağlamaktadır.

Dağıtık benzetim, bağımsız işleyen ve birbirleriyle etkileşim halinde olan çoklu benzetim yazılım prosesleridir. Dağıtık benzetimde kullanılan yazılımlar farklı tedarikçiler tarafından üretilse ve modüller fiziksel/coğrafi olarak farklı bilgisayar sistemlerinde çalışsa da etkileşim halindedirler.

Amerikan Savunma Bakanlığı'ndaki bütçe kısıtlamaları, eğitim için yeni seçenekler aranması ihtiyacını doğurmuştur. Bu arayış, bilgisayar teknolojilerinin ve hızının gelişmesinden dolayı benzetim ve modelleme teknolojilerine doğru olmuştur. Benzetim temelli eğitimin geliştirilmesi, askeri savunma ile ilgili eğitim faaliyetlerinin birçok problemine çözüm oluşturmuştur. Bu problemler şunlardır (Wilcox vd., 2000):

1. Maliyet

Yeteri miktardaki gerçek donanımın eğitim için tahsis edilmesi pahalıya mal olmaktadır. Örneğin gerçek silahların ateşlenmesi için yine gerçek bir uçak

kullanmak oldukça pahalıdır. Bunun yerine, gerçek donanım kullanmaktansa uygun simülatörler geliştirmek daha ucuza mal olacaktır.

2. Gerçek hayatta gerçekleştirilmesi zor olan durumlar

Bazı durumların gerçek hayat eğitimlerinde gerçekleştirilmesi imkânsız olmasa bile zordur. Örneğin denizci, karacı gibi birçok askeri kuvvetin temsil edilmesi veya uluslar arası katılımın gerçekleştirilmesi gibi. Bazen de henüz prototip aşamasında geliştiriliyor olup henüz var olmayan bir donanımın/ürünün denenmesi de gerekmektedir; bu da benzetim ile mümkün olabilmektedir.

3. Politik ve sosyal sebepler

Halk arasında, özellikle canlı eğitimin gerçekleştirilebilmesi için bir alanın kullanılması konusunda artan bir hassasiyet söz konusudur.

Yukarıda sayılan sebeplerden dolayı Amerikan Savunma Bakanlığı İleri Dağıtık Benzetim kavramına odaklanmıştır. İleri Dağıtık Benzetimin amaçları ve faydaları şöyle özetlenebilir (Wilcox vd. 2000):

- Yeniden kullanılabilirlik: Benzetimlerin geliştirilmesinde zaman ve maliyet kazancı sağladığı için benzetim modellerinin bileşenlerinin yeniden kullanılabilirliği önemli hale gelmiştir. Yeniden kullanılabilirlik kavramı ile benzetim modelinin tamamının veya bir kısmının, verinin ve/veya yazılımın yeniden kullanılabilmesi kastedilmektedir.

- Sentetik (yapay) ortam: Sentetik (yapay) ortam, ileri dağıtık benzetimin anahtar faktörüdür. İleri dağıtık benzetim içerisindeki her katılımcı benzetimin ortak bir sentetik ortamı görmesi zorunluluktur.

- Daha fazla aslına uygunluk: İleri dağıtık benzetim ile çeşitli detayların daha iyi temsil edilmesi mümkün olmaktadır. İleri dağıtık benzetim içerisindeki katılımcı benzetimler böylece daha gerçekçi bir ortamda olmakta ve benzetim

bileşenlerinin davranışı daha gerçekçi olmaktadır. Analizci açısından da daha detaylı verilerin elde edilmesi elverişli hale gelmektedir.

- Mümkün olan en iyi: İleri dağıtık benzetim, fiziksel yer sınırlarını kaldırmayı amaçlamaktadır. Farklı türdeki benzetimlerin karşılıklı işlemesini sağlayarak da benzetimin sınırlarını genişletmeyi amaçlamaktadır. Bu, ele alınan problemin en iyi benzetim modelinin kullanılması potansiyelini sağlamaktadır. Örneğin, yerel olarak mevcut olan kaynaklar ile sınırlı kalmayıp kaynakların ortak kullanımı mümkündür. Böylece dağıtık olmayan senaryoda sağlanamayan birçok durum sağlanabilecektir; gerçek teçhizatın, sistemlerin ve personelin eğitimin bir parçası haline gelmesi ile daha işlevsel bir temsil gerçekleştirilebilecektir.

- Zaman şemaları: Çeşitli zaman şemalarını desteklemek önemlidir (gerçek zaman ve gerçek zamandan daha hızlı/daha yavaş zaman). Teknik açıdan benzetimlerin birbirleriyle bağlanmasında zaman anlayışı önemlidir, bununla birlikte insan açısından da cevap süresi meselesi ve bileşenlerin davranışı söz konusu olduğunda zaman önemlidir.

- Ölçeklenebilirlik: İleri dağıtık benzetimde çeşitli tekniklerle ölçeklenebilirlik iyileştirilebilmektedir; böylece tek bir dağıtık benzetimdeki katılımcı sayısı artabilmektedir.

- Taşınabilirlik: İleri dağıtık benzetim birçok seviyede taşınabilirliği amaçlamaktadır. Birincisi, uygulama alanı kapsamında teknolojinin ve fikirlerin taşınabilirliği ile ilgilidir. Her ne kadar ileri dağıtık benzetimin çıkışı askeri kaynaklı olsa da ilgi alanının askeri alanlar dışına taşınma çabaları sürekli devam etmektedir. İkincisi ise, yazılımların ve modellerin taşınabilirliği ile ilgilidir. Dağıtık benzetim terminolojisinde bunlara karşılıklı-işlerlik denmektedir.

Dağıtık benzetim kapsamı içerisinde olan farklı benzetim türleriyle ilgili üç kabul görmüş tanım vardır:

1. Canlı: Bu benzetim türünde gerçek insanlar ve donanım katılımcı olarak bulunmaktadır ve bunlara arayüzler mevcuttur.

2. Sanal: Benzetim sistemiyle insanların etkileştiği benzetim çeşididir.

3. Yapısal: Bu benzetim türünde her şey bilgisayar ile modellenmiştir; benzetim modeli olarak tasarlanmış insanlar benzetim ortamında işlemektedir.

Dağıtık bir benzetim yukarıda bahsedilen üç tür benzetim türünün kombinasyonundan da oluşabilmektedir. Dağıtık benzetimlerle bütünleşik olarak benzetim olmayan yazılımlar da kullanılabilmektedir.

Senkronizasyon dağıtık benzetimde önemlidir. Bu problemin üstesinden gelmek için benzetim modelleri birbirleriyle "zaman etiketli" mesajlarla (timestamped messages) haberleşmektedirler. Bu gereksinime yerel etken kısıtı denmektedir (Fujimoto 2001). Bu konuda senkronizasyon veya zaman yönetimi yaklaşımları geliştirilmiştir. Yerel etken kısıtı için önerilen protokoller kabaca ikiye ayrılmaktadır: Katı ve iyimser protokoller. Katı protokoller yerel etken kısıtını uygulamakta ve modellerin, olayları işletirken zaman etiketinde azalmaya doğru bir sıraya izin vermemektedirler. İyimser yaklaşıma göre ise yerel etken kısıtı ihlal edilebilir ancak ihlalin gerçekleştiği nokta tespit edilip geriye gidilerek zaman etiketi sırasına göre yeniden proses işletilebilmektedir.

Bazı dağıtık benzetimler, özellikle dağıtık imalat benzetimleri için katı bir şekilde senkronize edilmiş ortamlar zorunlu olmayabilir. Bu durumda yaklaşık senkronizasyon mekanizmaları kullanılabilir. Yaklaşık senkronizasyon yaklaşımının uygulanması katı senkronize sistemlerden daha kolaydır (Saad vd., 2003).

2.6. Dağıtık Benzetimin Gelişimi

1983 yılına kadar ABD Savunma Bakanlığındaki eğitim simülatörleri genellikle tek bir kişi veya iş için tasarlanmışlardı. Bu simülatörler kişiyi veya takımı askeri araçları kullanma konusunda eğitmek amacıyla kullanılmaktaydı. Simülatörleri

geliştiren ve yapanlar gerçekçi olmak için çok fazla gayret göstermekteydiler. Simülatörlerin her birinin maliyeti 1 milyon $ ile 50 milyon $ arasındaydı. Bu simülatörler, kişileri veya takımları hep birlikte işbirliği ile eğitmeye imkan vermiyordu. Ortak çalışma yapmak için tüm eğitimcileri ve ekipmanı eğitim alanına taşımak ve senaryoyu orada uygulamak gerekmekteydi. Bu eğitimler sırasında bazıları ölümle sonuçlanan kazalarla karşılaşılmakaydı.

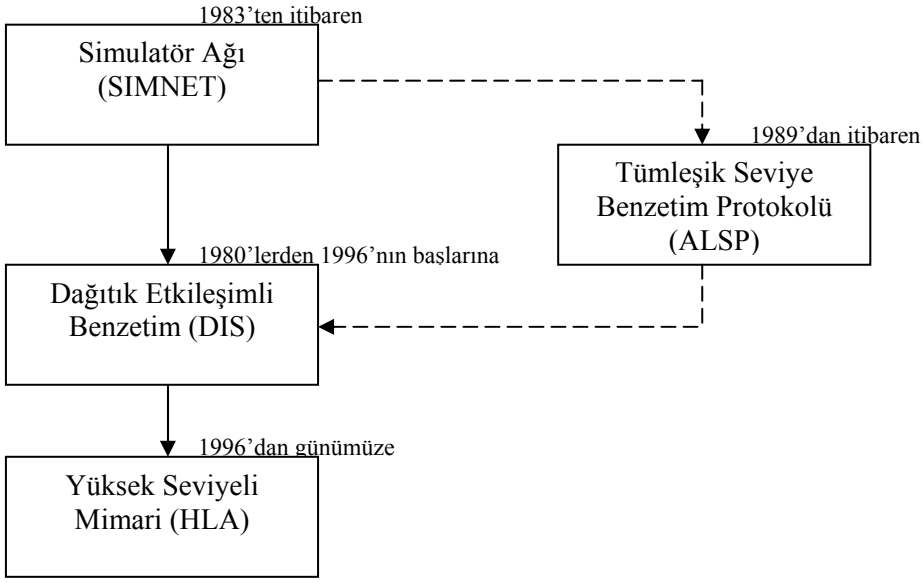
1983 yılında ABD Savunma Bakanlığı İleri Araştırma Projeleri Birimi (DARPA) bu durumu düzeltmek üzere karar aldı. DARPA'nın misyonunun bir parçası, yüksek seviyeli benzetim teknolojilerinin desteklenmesidir. DARPA, uzun menzilli füzeler, lazer güdümlü silahlar, sanal gerçeklik ve internet gibi teknolojilerin de öncülerindendir.

Bu karardan sonra DARPA, işbirliği ile çalışılabilecek benzetimlere imkân veren SIMNET'i (SIMulator NETwork) geliştirdi. SIMNET düşük maliyetliydi, farklı işler için kullanılabiliyordu ve simülatörler bir ağ ile birbirine bağlanabilmekteydi. SIMNET ile ortamın tamamının doğru bir şekilde modellenmesinden ziyade eğitimin tamamlanmasına yetecek kadarıyla benzetim modellenmekteydi. Simülatörlerin her biri yaklaşık 250.000 $ maliyetinde olmaktaydı.

Dağıtık benzetimin gelişimini, benzetim mimarilerinin gelişimi ile açıklamak mümkündür. Benzetimler arasında karşılıklı işlerlik ile ilgili standartlar ve kurallar sağlaması bakımından mimariler önemlidir. Şekil 2.1, benzetim mimarilerinin gelişimini göstermektedir (Wilcox vd. 2000).

Benzetim Ağı (SIMNET) 1983 ile 1990 yılları arasında DARPA tarafından desteklenen bir projeydi. Bu projenin amacı, tank mürettebatının taktik eğitimi için uygun bir benzetim ortamı geliştirmekti. SIMNET'in en önemli katkısı takım eğitiminin önemini vurgulaması ve takım eğitimlerini desteklemede benzetim teknolojilerinin kullanımını sağlamasıdır. Ortaya çıkan tank simülatörleri ağı ile düzenli ve yoğun olarak savaş tatbikatları gerçekleştirilmiştir. Buna ilave olarak SIMNET, yeni taktiklerin, doktrinlerin ve silah sistemlerinin incelenebilmesi için

ortam sağlamıştır. ABD ordusunda halen çalışan 250 SIMNET simülatörü bulunmaktadır (Miler ve Thorpe, 1995).



Şekil 2.1. Dağıtık benzetim mimarilerinin gelişimi.

SIMNET'te bilginin alışverişi için özel olarak ortak bir protokol geliştirilip kullanıldığı için karşılıklı işlerlik meselesi SIMNET'in anahtar öğelerinden biri değildi. Ancak ABD savunma bakanlığının diğer alanlarda kullanmak istediği tam olarak bu değildi. 1990'da DARPA'nın desteklemesi ile MITRE organizasyonu var olan yapısal savaş simülatörlerinin tamamı için karşılıklı işlerliği destekleyecek genel bir protokol tasarlamayı araştırmıştır. Burada amaç, benzetimin içerisinde, kendi özelliklerini muhafaza eden birçok varlığın (ör, donanma veya filo içerisindeki her bir varlığın) aynı anda temsil edilebilmesidir. Bu çalışmanın sonucunda ALSP ortaya çıkmıştır (JTC, 2006). ALSP projesi, ABD ordusu içerisindeki çeşitli kısımların kendi savaş benzetimleri için ayrı ayrı yatırımlar yaptıklarını ortaya çıkarmıştır. SIMNET tecrübesine ilaveten bu proje ile farklı benzetimlerin, bunların yeniden tasarlanmasına gerek duyulmadan birbiriyle karşılıklı işlemesinin yararlı olacağı görülmüştür.

SIMNET ve ALSP projelerinden elde edilen tecrübe, dağıtık etkileşimli benzetimin (DIS: Distributed Interactive Simulation) gelişmesine katkıda bulunmuştur (SISO, 2006). Heterojen (farklı yazılım ile veya farklı ortamlarda tasarlanmış) benzetimlerin DIS protokolü ile sentetik ortamda etkileşmesi mümkün hale gelmiştir. DIS, benzetimler arası iletişime odaklanarak birçok benzetim türünü (yapısal, canlı, vs.) desteklemiştir. Bunun sonucunda DIS protokolü veri birimleri (PDU) geliştirilmiştir. PDU, benzetimler arası iletişimin standardını belirlemektedir (IEEE, 1995a ve 1995b). Benzetim modeli kurulmuş nesnenin tüm durumu her PDU yayımıyla iletilmektedir. Bu durum, DIS benzetimlerinde veri toplamayı kolaylaştırmaktadır. PDU verileri elde edildiğinde, nesnenin o andaki tüm durumu hakkında bilgi elde edilmektedir. Ancak bu durum ölçeklenebilirlik kısıtını sağlayamamaktadır. Çünkü PDU ile birçok veri iletilmekte ancak bu verilerin bir kısmı ya değişmeyen verileri içermekte veya karşı tarafı ilgilendirmeyen verileri barındırmaktadır. Bu ise, gereksiz verilerin iletilmesinden dolayı, iletişim ağında aşırı yük meydana getirmektedir. Bu gibi kısıtlardan dolayı daha gelişmiş dağıtık benzetim mimarilerine ihtiyaç duyulmuştur.

Birçok benzetim türünü desteklemesine rağmen DIS'in kısıtları vardır. PDU iletiminden kaynaklanan ölçeklenebilirlik sorunundan bahsedilmiştir. Bundan başka, DIS ile ortak bir sentetik ortamın temsil edilmesiyle ilgili kısıt da mevcuttur. DIS, tüm katılımcı benzetimlerin ortak veritabanına erişim sağladıklarını ve üzerinde anlaşmaya varılmış bir grafiksel gösterimi kullandıklarını garanti altına almamaktadır. DIS ile uygulanan bu yöntem gerçek zamanlı benzetimlerin uygulanmasında kısıt oluşturmuştur.

1991'de Amerikan Savunma Bakanlığı, Savunma Modelleme ve Benzetim Ofisi'ni (DMSO, 2006), dağıtık benzetim çalışmalarını birleştirmek için kurmuştur. DMSO çalışmalarının köşe taşını HLA teşkil etmiştir ve en öncelikli çaba HLA'ya verilmiştir. HLA'dan önceki dağıtık benzetim çalışmalarından tecrübe edilenler ve öğrenilenler geliştirilerek veri iletişim standardı oluşturulmuştur. Odak noktası, DIS'te uygulanan iletişim içeriğinden iletişim mekanizmasına kaydırılmıştır. DIS gibi HLA da Amerikan Savunma Bakanlığı bünyesinde kullanılmak üzere geliştirilmiştir. Ancak HLA'ya olan ilgi, onun daha geniş benzetim çevrelerinde de

kullanılabilirliğini ortaya koymuştur. Böylece sadece Amerika'da değil, Amerika dışında da ve savunma dışı uygulamalarda da kullanılabilir hale gelmiştir (Wilcox vd., 2000).

1996'dan beri dağıtık benzetim ile ilgili gelişmeleri rapor edenlerin en başında Benzetim Karşılıklı-işleyebilirliği Standartları Organizasyonu (SISO, 2006) gelmektedir. Dağıtık benzetim çalışmalarının yaygınlaşması amacıyla çeşitli atölye çalışmaları ve sempozyumlar düzenlemektedir. Çok çeşitli organizasyon temsilcilerinden birçok üyesi mevcuttur. DMSO ile SISO birlikte çalışarak birçok HLA standardı geliştirmişlerdir (Wilcox vd., 2000). 2000 yılında ise IEEE tarafından modelleme ve benzetim standardı olarak kabul edilmiştir (Xiaoxia ve Qiuhai, 2003).

Şekil 2.1'de dağıtık benzetim mimarilerinin gelişimi hiyerarşik olarak verilmişti. Burada dikkat edilmesi gereken husus, yeni bir dağıtık benzetim mimarisi geliştirildiğinde, daha önceki mimarilere göre modellenmiş sistemlerin hemen ortadan kalkmadığıdır. HLA ile ilgili çalışmalar devam etse de daha önce geliştirilmiş olan SIMNET, ALSP ve DIS tabanlı sistemler halen çalışmaktadır. Daha eski benzetim mimarileri ile geliştirilmiş sistemlerin HLA ile iletişim sağlayabilmeleri için çeşitli arayüz geliştirme çalışmaları da gerçekleştirilmiştir. Bu çalışmalarda bazılarını Cox ve Wood (1996), Cox vd, (1996) ve Wood vd., (1997)'de görmek mümkündür.

2.7. Dağıtık İmalat Benzetimi

Hibino vd. (2002), dağıtık benzetim ile ilgili pek çok araştırma makalesinin olduğunu, ancak bunlarda belirtilen fikirlerin endüstride uygulama alanı bulma sayısının az olduğunu tespit etmiştir. Bununla beraber dağıtık imalat benzetimi ile ilgili olarak belirtilenler aşağıda ele alınacaktır.

McLean ve Riddick (2000a), dağıtık imalat benzetimini birkaç bakış açısıyla ele almışlardır. Bunlardan birincisine göre dağıtık imalat benzetimi, bağımsız olarak işleyen ve birbirleriyle etkileşim halinde olan çoklu yazılım süreçleri olarak düşünülebilir. Bu benzetim yazılım süreçleri, imalat tedarik zinciri büyüklüğündeki

bir sistemden tek bir imalat tezgahı küçüklüğündeki bir sistemi modellemiş olabileceği gibi benzetim yazılımlarını farklı yazılım tedarikçileri sağlamış olabilir. Modüller, coğrafi olarak dağıtık durumdaki farklı bilgisayarlarda çalışıyor da olabilir.

Diğer bakış açısına göre, dağıtık benzetim sistemleriyle birlikte veya etkileşimli olarak, benzetim programı olmayan imalat yazılımları da çalışabilir. Örneğin bir bilgisayar destekli imalat uygulaması bir makine takımı için kontrol programı oluşturduktan sonra bunun doğruluğunu denemek için bir simülatöre gönderebilir.

Başka bir bakış açısına göre dağıtık imalat benzetimi, içinde model inşa araçları, benzetim motorları, görüntü sistemleri ve çıktı analiz yazılımını barındıran işlevsel modüllerden teşkil edilmiş tek bir benzetim sistemidir.

Saad vd. (2003), dağıtık imalata sanal imalat kurumları olarak da ifade etmektedir. Buna göre dağıtık imalat, coğrafi olarak dağıtık ortamlarda işleyen ve birbirleri ile modern iletişim teknolojileri ile bağlı kurumlardır. Bu kurumlar belirli bir ürün veya üretim hattı için işbirliği yaparlar. Bu tür oluşumlar her zaman sürekli birliktelikler değildir ve her kurum bu işbirliklerinin yanı sıra kendi işlerini de yapmaktadırlar. Bu çalışmalarında, dağıtık imalat için bir metodoloji önermişler ve bunu basit bir modelle denemişlerdir. Yazarlar, dağıtık imalat benzetimine yeni bir yaklaşım getirmiş, biçimsel bir metodoloji ve mevcut ticari benzetim yazılımlarıyla birlikte yaygın ve ucuz teknolojilerin kullanılmasıyla uygulama yaklaşımı geliştirmişlerdir. Böylece imalatta dağıtık benzetimin hızlı geliştirilmesi ve uygulamasının daha az karmaşıklığı hedeflenmiştir. Benzetimde modelleme metodolojisi bir benzetimin nasıl modelleneceğine odaklanmıştır. Bu çalışmalarında senkronizasyon ve kullanılan teknoloji ve yazılım konusuna özellikle değinmişlerdir. MSMQ (Microsoft Message Queue), Arena benzetim yazılımı ve bu yazılımla uyumlu olduğu için Visual Basic for Application (VBA) kullanılarak hipotetik bir dağıtık imalat benzetimi gerçekleştirmişlerdir. Bununla birlikte Saad vd. (2003) HLA, CORBA, GRIDS ve MSMQ (Microsoft Message Queue) araçları kullanılarak dağıtık imalat sistemleri ve tedarik zinciri teşebbüslerine literatürde rastlamanın mümkün olduğunu söylemişlerdir.

Dağıtık benzetimin endüstriye uygulanabilirliğini artırmak için gerçekleştirilen en önemli çalışmalardan biri IMS-MISSION projesidir. 1990'lardan sonra birçok ticari benzetim yazılımı ortaya çıkmış ancak bu yazılımlar yeni endüstri şartlarını karşılayamaz hale gelmiştir. Yeni endüstri ve imalat sistemi artık küresel olarak dağıtıktır veya artık sanal kurumlar söz konusudur. MISSION projesinin amacı böylesi bir ortamda benzetim uygulamalarını destekleyecek platform geliştirmektir (MISSION, 2001a).

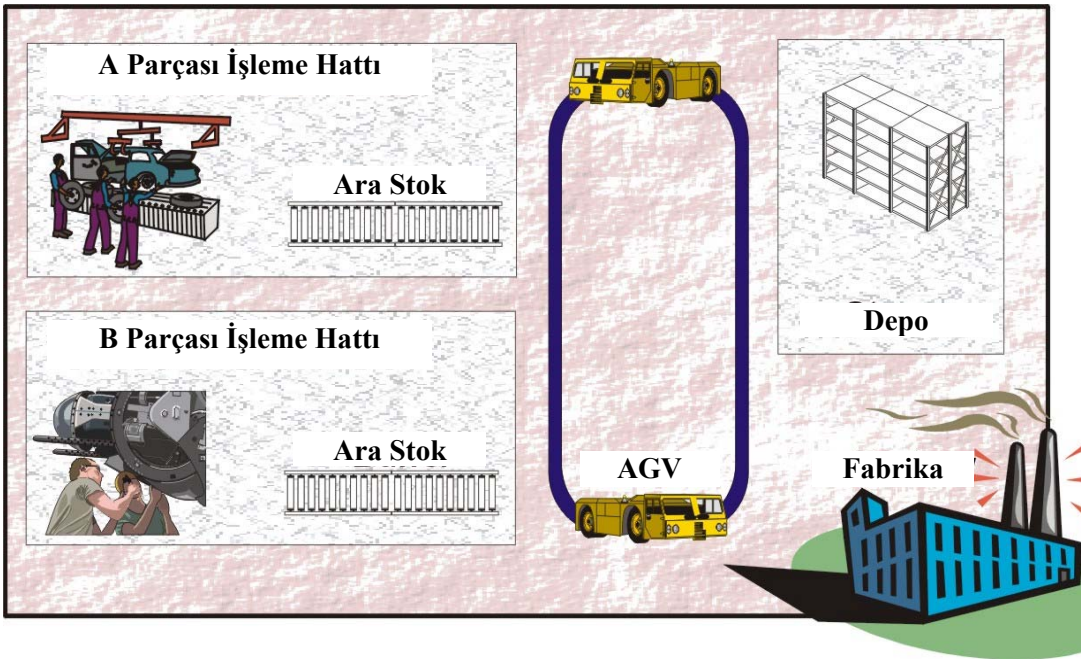
MISSION projesinin genel amacı, İmalat Sistem Mühendisliği sürecini, bu sürecin çeşitli yönlerinde kullanılacak araçlarla bütünleştirerek desteklemektir. Ancak bu süreci hızlandırmak için araçlar, benzetim kullanımıyla ve benzetim araçlarının diğer yazılımlarla bütünleştirilmesiyle ilgili olmalıdır. Böylece imalat sistem mühendisliği, ve imalat süreçlerinin dağıtık benzetimi kapsanmış olacaktır (MISSION, 2001a).

McLean ve Riddick (2000a), dağıtık imalat benzetim sistemlerinin birbirleriyle, diğer imalat yazılım uygulamalarıyla ve imalat veri ambarlarıyla bütünleştirilmesini sağlayan referans bir mimari önermektedirler. Bu mimari uluslar arası Zeki İmalat Sistemleri (IMS-Intelligent Manufacturing Systems)'in MISSION projesinin bir parçası olarak geliştirilmektedir. IMS projeleri hakkında ayrıntılı bilgi için <www.ims.org> internet sayfasına bakılabilir (IMS, 2006).

Dağıtık imalat benzetiminde en büyük problemlerden biri entegrasyon sorunudur. Bu sorunu çözmek için dağıtık imalat benzetimi üç işlevsel bakış açısıyla ele alınmalıdır: dağıtık bilgisayar sistemleri, benzetim sistemleri, imalat sistemleri. Her biri diğerlerinin bakış açısıyla birbirleriyle ilişkilendirilmelidir (McLean ve Riddick, 2000a).

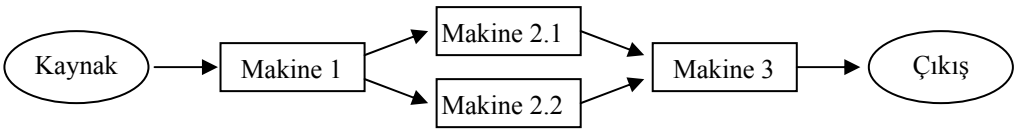
HLA'da programlama zorluğunu aşmak için MISSION projesinde şablon kütüphanesi oluşturulmuştur. Bu, benzetim senaryosuna dahil olan federeler için konfigürasyon dosyalarının oluşturulması sağlayacaktır ve yoğun programlama gerektirmeyecektir.

Şekil 2.2, basit bir dağıtık imalat sistemini göstermektedir. A ve B parçaları işlenmek üzere iki proses hattı vardır ve buradan çıkan parçalar otomatik taşıma araçlarını (AGV) ara stokta beklerler. AGV, A ve B parçalarını ara stoklardan alarak depoya taşır. Buradaki proses hatları, AGV ve depo tamamen ayrı benzetim modelleridir (MISSION, 2001b).



Şekil 2.2. Basit bir dağıtık imalatın temsili gösterimi

Schuman vd. (1998) ise küçük bir imalat sisteminde malzeme akışının benzetimi için bir federasyon oluşturmuştur. Parçalar sanal fabrikaya bir kaynaktan girerler ve sırasıyla üç makinede işlem görürler. Makine 1 ve makine 3'ten birer adet, makine 2'den iki adet mevcuttur. Her makine arasında taşıma sistemi vardır. Anlatılan bu sistem Şekil 2.3'de gösterilmiştir.



Şekil 2.3. Schuman vd. (1998) tarafından benzetim modeli kurulan malzeme akışı sisteminin temsili

Hibino vd. (2002), hipotetik bir imalat ortamında basit bir motor üretiminin benzetimini ele almıştır. Buna göre motorun dört parçası için imalat hattı alt sistemi bir de satın alınan parçalar söz konusudur. Bunlar bir depoda toplanır ve nihai ürün haline getirilmek üzere montaj hattına taşınır. İmalat hattı alt sistemlerinin farklı fabrikalarda olduğu varsayılmıştır ve buralarda kullanılan benzetim yazılımları da farklıdır (SIMPLE++, QUEST, GAROPS). Bu farklı ortamlar arası iletişim HLA ile sağlanmıştır.

2.8. İmalat için Dağıtık Benzetimin Önemi

İmalat endüstrisinde pazar küresel hale gelmiştir. Karmaşık ürünler artık tek bir fabrika tarafından üretilmemekte; bileşenler farklı şirketlerce üretilmekte ve başka bir yerde montajı yapılmaktadır. Bu, şirketlerin birbirlerine olan bağımlılığını artırmaktadır. Bu sebeple tedarikçilerin de fabrika benzetimine dahil edilmesi gereklidir. Bu sebeple, benzetim modellerinin, lojistik problemlerin çözümünde kullanılabilmesi için karşılıklı çalışabilir olması zorunludur (Schumann vd., 1998).

Pazarın küreselleşmesi ve dünya çapında rekabet koşulları imalat işletmelerini dağıtık imalat kurumlarına dönüştürecek ortaklıklar yapmaya zorlamaktadır. Birliktelik gerçekleştirilmeden önce oluşturulması düşünülen kurumun sürdürülebilirliğinin ve şirket fonksiyonlarının bu sanal kurumdan nasıl etkileneceğinin araştırılması gerekmektedir. Dağıtık benzetim bu gibi durumlarda karar verebilmek için önemli bir araçtır (Saad vd., 2003).

Benzetim bir karar verme aracı olarak imalatta kullanılagelmiştir. Benzetimin en çok kullanım bulduğu alan imalattır. Ancak geleneksel sıralı benzetim tek başına karmaşık dağıtık imalat kurumlarının benzetim modellerini kurmaya yetmemektedir (Saad vd., 2003).

Hibino vd. (2002), birçok yazarın (Fukuda, Hibino, Mitsuyuki, Kojima, Fujii, Kidani, Ogita, Kaihara… gibi) yeni imalat sistemleri tasarlamak için simülatörlerin önemli olduğunu söylediğini belirtmiştir. İmalat sistemleri eskisinden çok daha karmaşık ve büyük ölçekli tasarlandıklarından imalat sistemlerinin birçok kişi

tarafından eş zamanlı olarak tasarlanmaları gerekmektedir. Ürünlerin artık hızlı bir şekilde tasarlanıp piyasaya sunulması zorunludur. Tasarım ve değerlendirmede faaliyetlerin çok kısa sürede gerçekleştirilmesine ihtiyaç vardır (Hibino vd., 2002).

Dağıtık imalat tamlamasının vurguladığı gibi dağıtık imalat kurumları (Saad vd. (2003) dağıtık imalat kurumlarını sanal imalat kurumları olarak da adlandırmaktadır) coğrafi olarak dağıtık ortamlarda işlerler ve modern iletişim teknolojileri ile birbirlerine bağlanırlar. Sanal imalat kurumlarında birçok şirket ürün üretmek için işbirliği yaparlar. Böyle bir işbirliği ile farklı şirketler bilgisini, kaynaklarını ve özel imalat tecrübesini bir araya getirerek, bir şirketin tek başına gerçekleştiremeyeceği büyük ölçekte iş fırsatı doğururlar ve rekabet üstünlüğü elde ederler (Saad vd., 2003).

İmalat sistemleri, sistemin gerektirdiği çeşitli spesifikasyonlara bağlı olarak birçok alt sistemden teşekkül eder. Her alt sistem tekil olarak tasarlanıp eş zamanlı olarak optimize edilmeye çalışılır. Bunun için her alt sistem bir simülatörde bir birim olarak tasarlanır. Alt sistemler incelendikten sonra tüm sistemin analizi gereklidir. Her simülatör belirli bir amaç için kullanıldığından simülatörlerin modellenme yöntemleri farklıdır. Günümüzde farklı modellenmiş bu simülatörleri bir araya getirmek, benzetim modellerinin tanımlanmasında bir standardın olmamasından ve standart bir benzetim modelleme dili geliştirilmediğinden dolayı oldukça güçtür. Bu sebeple farklı simülatör modelleri kullanıldığında tasarımcılar işbirliği yapmakta zorlanırlar. Tüm alt sistemleri senkronize çalıştırmakla sistemin tamamını incelemek kolay değildir. Bu gibi sebeplerden dolayı dağıtık benzetim kullanılmaktadır. Fujimoto dağıtık benzetimi, farklı simülatörleri birbirine bağlayarak ve senkronize ederek bir benzetimin çalıştırılması olarak tanımlar (Hibino vd., 2002).

Benzetim, şirket işlerinin karmaşık proses yapısını planlamak için önemli bir araç haline gelmiştir. Benzetim daha çok şirketin birkaç anahtar faaliyetinde uygulama alanı bulsa da artık kurumun hayat döngüsünün her aşamasında eş zamanlı olarak kullanılması önemli hale gelmiştir (Schumann vd., 1998).

Benzetim, belirli bir yazılımın üstün olan bir işlevinden yararlanmak, tekil sistem modellerinin özel bilgilerini korumak ve paralel bilgisayar işlemcileri kullanarak

benzetimin çalışma hızını artırmak için de kullanılabilir (McLean ve Riddick, 2000a).

Tasarım aşamasında benzetim belirli üretim imkânlarının kapasitelerini belirlemede kullanılabilir. Aynı zamanda kurulacak fabrikanın etkinliğini ve çıktısını tahmin etmede de kullanılabilir. Aynı benzetim modelinin, alt seviyelerdeki çeşitli kontrol stratejilerinin incelenmesinde de kullanılması mümkündür (Schumann vd., 1998).

Dağıtık yaklaşım benzetimin işlevselliğini artırır. Dağıtık imalat benzetim sistemleri aşağıdaki amaçlar için kullanılabilir (McLean ve Riddick, 2000a):

- Tedarik zincirini modellemede kullanılabilir. Böylece bir şirkettin bilgisi diğer zincir üyelerine de açık hale gelir.
- Çok seviyeli imalat sistemlerinin farklı derecelerde benzetim modellerini kurmak için kullanılabilir. Böylece alt seviye benzetim sonuçları üst seviye benzetimlere bilgi sağlar.
- Aynı fabrikadaki faklı benzetim gereksinimleri olan birçok sistemin benzetim modelini kurmak için farklı benzetim yazılımları kullanılabilir. Bazı benzetim yazılımları her gereksinimi sağlamayabilir. Bu durumda bu açığı kapatmak için sistemin bazı kısımları farklı benzetim yazılımlarıyla modellenir.
- Daha büyük modellere entegre edilebilen düşük maliyetli, çalışma zamanlı bir dizi benzetim modeli oluşturabilmek için kullanılabilir.
- Diğer bilgisayarların işlemci gücünden, belirli bir işletim sisteminin özelliklerinden ve çeşitli araçlardan (sanal gerçeklik arayüzü gibi) yararlanmak için kullanılabilir.
- Farklı yerlerdeki kullanıcıların benzetim modellerini çalıştırabilmeleri için eşzamanlı erişim olanağı sağlar. Böylece işbirlikçi çalışma ortamı oluşturulur.
- Farklı işlevler için benzetim faaliyetlerini (model kurma, görsellik, çalıştırma, analiz etme) destekleyecek farklı türlerdeki birçok yazılımın kullanımına imkân sağlar.

Dağıtık benzetim sistemleri sayesinde eşzamanlı mühendislik uygulamalarının da dağıtık olarak gerçekleştirilmesi mümkündür (Hibino vd. 2002).

Günümüz bilgisayarlarının yüksek grafik yetenekleri sayesinde benzetim, kullanıcı eğitiminde de kullanılmaktadır (Schumann vd., 1998).

2.9. Dağıtık Benzetimin Zorlukları

Dağıtık benzetim uygulamalarının birçok faydası yanında birtakım zorluklarından da bahsetmek mümkündür. Aşağıda, dağıtık benzetimin bazı muhtemel zorlukları listelenmiştir:

- Dağıtık benzetimin geliştirilmesi ve girdi verilerinin elde edilmesi zaman alıcı, uygulanması, yönetimi ve benzetim sonuçlarının gösterimi karmaşık ve maliyeti yüksektir (Saad vd., 2003; Schuman vd., 1998).
- İmalat benzetiminde kullanılacak uygun yazılımlar her zaman mevcut değildir. Bu sebeple mevcut yazılımlar ile örneğin HLA bütünleştirilmeye çalışılarak dağıtık benzetim gerçekleştirilir (Schuman vd., 1998; Hibino vd. 2002)
- Dağıtık benzetim, uzmanlıkla birlikte programlama dili bilgisi gerektirir (Saad vd., 2003).
- Yazılım sistemlerinin boyutu ve karmaşıklığı arttıkça tüm sistemin yapısının tasarım ve spesifikasyonu, seçilecek algoritma ve veri yapısından daha fazla önem arz eder (McLean ve Riddick, 2000a).
- Dağıtık benzetim iletişim teknolojisinin sağladığı hıza bağlıdır (Saad vd., 2003).
- Dağıtık bir imalat benzetiminin entegrasyon sorunu halledilmelidir. Bu sorunla, yazılım tedarikçileri ve benzetim teknolojisini kullanan endüstriyel kullanıcılar karşı karşıyadırlar. Tarafsız benzetim arayüzleri veri iletimi ve model paylaşımı maliyetlerini azaltacaktır (McLean ve Riddick, 2000a).
- Dağıtık imalat kurumları çok karmaşık ve heterojendir. Klasik imalat kontrol sistemlerinin, dağıtık imalatın karmaşık ve dinamik yapısına uyum sağlama kapasiteleri düşüktür. Bu sebeple dağıtık imalat mimarileri geliştirme teşebbüsleri vardır. (Saad vd., 2003).
- Dağıtık benzetim modellerinin senkronizasyonu gereklidir (Saad vd., 2003).

BÖLÜM 3. YÜKSEK SEVİYELİ MİMARİ (HLA) VE İMALAT

3.1. Giriş

1990'larda Amerikan Savunma Bakanlığının ihtiyacı ile şimdiki küresel dağıtık kurumların modelleme ve benzetimi arasında benzerlikler vardı. Bir ortamda çalışan tekil bir benzetim artık ihtiyacı karşılayamaz hale gelmişti. Çünkü artık benzetim sistemleri birçok farklı sistemden bilgi alması gerekmekteydi. Üstelik gerek duyulan benzetim sistemleri dağıtık ortamda bulunmaktaydı. Birçok benzetim modeli vardı ancak yeni benzetim modelleri geliştirme ihtiyacı oluşmuştu. Mevcut sistemlerin ve bilgilerin dağıtık ortamda kullanılabilme gereksinimi meydana gelmişti. Dahası, çalışma anında benzetim sistemlerinin güncellenebilmesi önemli hale gelmişti. Amerikan Savunma Bakanlığı dağıtık benzetim mimarisi geliştirilmesi için bir proje başlattı. Bu yaklaşımın sonuçlarından biri de HLA olmuştur (MISSION, 2001a).

Paralel/dağıtık işleyen benzetim uygulamalarının gereksinimlerini karşılamak ve etkinliğini artırmak için karmaşık sistemler aşağıdaki temel talepleri karşılamalıdır (Xiaoxia ve Qiuhai, 2003):

Benzetim uygulamaları arası karşılıklı işleyebilirlik: Farklı şekillerde tasarlanmış benzetim uygulamaları arasında bilgi alışverişi ve etkileşim gerçekleştirilebilmelidir.

Uyarlanabilirlik ve yeniden kullanılabilirlik: Farklı yerlerde, farklı zamanlarda, farklı platformlarda tasarlanmış benzetimlerin birbirleriyle bağlanabilmesi için bu benzetimlerin tek tip, mantıksal bir mimaride tasarlanması gerekmektedir.

Ölçeklenebilirlik: Dağıtık benzetim sistemlerinde, sisteme giren ve çıkan benzetim uygulamaları olduğunda bu, diğer benzetim uygulamalarının çalışmasını etkilememelidir.

Bu talepler yeni bir benzetim tekniği çerçevesini gerektirmektedir. Bu sebeple DMSO (Amerikan Savunma Bakanlığı Savunma Modelleme ve Benzetim Birimi) 1995'de HLA'yı (High Level Architecture) önermiştir. Daha sonra IEEE 1516 standardı olmasıyla birlikte, HLA, sadece askeri standart olmaktan çıkmış, sivil alanlarda da kullanıma açık hale getirilmiştir. HLA şu anda NATO'nun ve İsveç gibi bazı ülkelerin de benimsediği bir standarttır (PITCH, 2006a). NASA da 2001 yılından sonraki tüm benzetimlerde HLA kullanmaya karar vermiştir. HLA sadece savaş oyunlarında/benzetimlerinde değil eğitim, analiz, mühendislik ve eğlence alanlarında da kullanılabilmektedir (Xiaoxia ve Qiuhai, 2003).

3.2. HLA'nın Tanımı

HLA (High-Level Architecture: Yüksek Seviyeli Mimari), bilgisayar temelli benzetim sistemlerini birbirine bağlayan bir standarttır. Böylece bu sistemler birlikte çalışıp bilgi alış verişinde bulunabilirler. Büyük, tek parça bir benzetim sistemini sil baştan kurmaktansa HLA, var olan benzetim sistemlerini yeni sistemlerle birleştirebilir (PITCH, 2006a).

Benzetim Karşılıklı-Çalışma Standartları Organizasyonu (Simulation Interoperability Standards Organization: SISO) ise HLA'nın, benzetimlerin karşılıklı işleme ve yeniden kullanılmasını desteklemek üzere, tüm benzetim sınıflarında uygulanabilir ortak bir mimari sağlamak amacıyla geliştirildiğini söylemektedir.

HLA, hiçbir tekil benzetimin, tüm kullanımların ve tüm kullanıcıların isteklerini karşılayamayacağı iddiasına dayanmaktadır. Belirli bir amaç için geliştirilmiş bir benzetim veya bir benzetim dizisi HLA federasyonu kavramı altında (birbiriyle ilişkili bir araya getirilmiş benzetimler) diğer uygulamalara uygulanabilir. HLA'nın amacı, farklı benzetimlerde elverişli olan yeteneklerin tekrar kullanımını destekleyecek bir yapıyı sağlamaktır (Dahmann vd., 1998, Dahmann vd., 1999a).

HLA modelleme ve tasarıma nesne-yönelimli bir metot sunmaktadır. Gerçek sistemin modelini "nesne" kavramı ile geliştirmektedir. Geliştirilen bu model gerçek

dünyanın soyut bir halidir. Sistemin ölçeklenebilirliğini, sistem bileşenlerinin yeniden kullanımını ve karşılıklı işleyebilmesini sağlamaktadır. HLA sadece benzetim verilerinin toplanmasını ve benzetim faaliyetlerinin gösterimini desteklemez; canlı katılımcılar için de arayüz sağlayabilmektedir (Xiaoxia ve Qiuhai, 2003).

HLA, mevcut sistemleri yeni amaçlar için kullanabilme imkânı sağlamaktadır. Farklı programlama dillerini ve işletim sistemlerini bir arada kullanmak da mümkündür (PITCH, 2006a). HLA belirli bir uygulama biçimi, belirli bir yazılım veya programlama dili kullanımı öngörmez. Teknolojinin gelişmesiyle birlikte zaman içerisinde HLA çerçevesi ile farklı uygulamalar mümkün hale gelecektir (Dahmann vd., 1999a).

3.3. HLA'nın Gelişim Süreci

Benzetimlerin yeniden kullanılabilirliği ve karşılıklı işleyebilirliği, Amerikan Savunma Bakanlığı (DoD) tarafından 1990'ların başında ihtiyaç olarak tespit edilmiştir. Bu doğrultuda modelleme ve benzetim için yeni bir vizyon ortaya konmuş ve bunu takiben mastır plan (DMSO, 1995) hazırlanarak bu vizyonu gerçekleştirmek için hangi kritik alanlara yatırım yapılması gerektiği ortaya konmuştur. Birinci amaç, modelleme ve benzetim geliştirmek için ortak teknik bir mimarinin geliştirilmesi olmuştur. Bu amaç, HLA geliştirilerek gerçekleştirilmiştir (Kuhl vd., 2000).

HLA 1.3 versiyonuna kadar Amerikan Savunma Bakanlığı tarafından geliştirilmiştir. Bu gelişim sürecinde akademi ve endüstriden de destek alınmıştır. 1995'te üç endüstri takımı HLA tanımı için kavramlar geliştirmişlerdir. Bu çalışmaların neticeleri başka modelleme ve benzetim projeleri ile birleştirilerek HLA'nın başlangıç tanımına ulaşılmıştır. Bu tanım 31 Mart 1995'te, başlangıç tanımından HLA'yı daha da geliştirmek için kurulmuş bir Mimari Yönetim Grubu'na (Architecture Management Group: AMG) sunulmuştur. Mimari Yönetim Grubu işbirlikçi çalışmalar sonucu çeşitli prototiplerden sonra çok geniş aralıktaki uygulama ihtiyaçlarını karşılayacak mimariyi geliştirmişlerdir. Bunun sonucu Ağustos 1996'da HLA'nın temel tanımı oluşmuştur. Ardından mimari yönetim grubu kullanım

tecrübelerinden faydalanarak altı aylık zaman periyotlarıyla HLA'yı geliştirmeye ve güncellemeye devam etmiştir. 1998'de HLA spesifikasyonlarının 1.3 versiyonu kabul edilmiştir (Dahmann vd., 1998, Dahmann vd., 1999a).

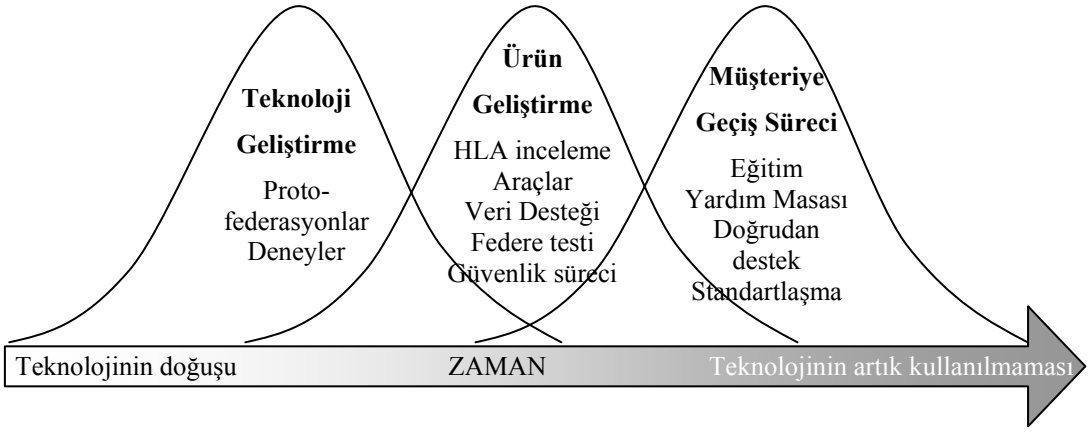
Aralık 1997'de Amerikan Savunma Bakanlığının geliştirdiği HLA, IEEE tarafından 1516 standart numarası ile taslak olarak kabul edilmiştir (Toft, 1999). IEEE bu standardı 2000 yılında kabul etmiştir (PITCH, 2006c).

IEEE 1516 standardı olmasıyla birlikte HLA, sadece askeri standart olmaktan çıkmış, sivil alanlarda da kullanıma açık hale getirilmiştir. Bu tarihten itibaren çeşitli HLA RTI ve ilgili araçlar geliştirilmeye başlanmıştır. RTI'ın ilk ticari ve tam olarak uygulaması olan pRTI 1516, Aralık 2001 tarihinden itibaren Pitch Technologies tarafından kullanıma sunulmuştur. Bununla birlikte Visual OMT 1516 gibi nesne modellemenin görsel olarak yapılabildiği araç da geliştirmişlerdir. Daha sonra çeşitli tedarikçiler başka araçlar geliştirerek kullanıma sunmuşlardır (PITCH, 2006c).

Her teknoloji gibi HLA'nın da geliştirilip kullanıcıya araç olarak sunulmasına kadar bir takım aşamalar söz konusudur. Önce bir teknoloji olarak HLA geliştirilmiştir. Ardından bu teknolojiyi uygulamaya sokacak araçlar geliştirilir. Daha sonra da bu araçlar HLA temelli benzetim uygulamaları yapacak kullanıcılara sunulur. HLA teknoloji dönüşüm eğrisi Şekil 3.1'de gösterilmektedir. HLA'nın geliştirilme süreci bu teknoloji dönüşüm eğrisine benzemektedir. Teknoloji dönüşüm eğrisindeki aşamalar ne kadar eş zamanlı hale getirilebilirse teknolojinin geçiş süreci ve adaptasyonu o nispetle kolaylaşır. HLA için de bu aşamaların oldukça eş zamanlı olarak yürütüldüğü söylenebilir (Kuhl, 2000).

HLA halen ABD Savunma Bakanlığı'nın kullandığı bir standarttır. HLA şu anda NATO'nun ve İsveç gibi bazı ülkelerin de benimsediği bir standarttır (PITCH, 2006a). NASA da 2001 yılından sonraki tüm benzetimlerde HLA kullanmaya karar vermiştir. HLA sadece savaş oyunlarında/benzetimlerinde değil eğitim, analiz, mühendislik ve eğlence alanlarında da kullanılabilmektedir (Xiaoxia ve Qiuhai, 2003). HLA Amerika'da ve sadece savunma/savaş benzetimleri için geliştirilmiş

olmasına rağmen kısa sürede diğer alanlardan ve ülkelerden ilgi görmüştür (Wilcox vd., 2000).



Şekil 3.1. HLA teknoloji dönüşümü

HLA'nın NATO'da da kabul görmesi ve uluslar arası eğitim faaliyetleri ile HLA'ya olan ilgi ABD dışındaki ülkelere genişlemiştir. Bunda, ABD Savunma Bakanlığı'nın (DoD), gelecekteki benzetimleri HLA ile gerçekleştirme ve daha önceki benzetim modellerinin HLA ile uyumlu hale getirilmesi politikaları da etkili olmuştur. Bu politika kararı diğer ülkelerdeki gelişmeleri de etkilemiştir. DoD ile çalışmak isteyen organizasyonlar da bu karar doğrultusunda kendilerini HLA'ya uyarlama gereği hissetmişlerdir. DoD, HLA'ya geçiş sürecine katkıda bulunmak amacıyla çeşitli eğitim programları düzenlemiş, yazılım ve araç destekleri sağlamıştır.

HLA kavramı geliştirilmeye başlandığında RTI da geliştirilmeye başlanmıştır. CORBA kullanılarak IDL API ile geliştirilen ilk RTI yazılımına "X serisi" denmektedir. Daha sonra Aralık 1996'da C++ ile 1.0 versiyonu geliştirilmiştir. Tanıtım sürümü (F.0) de denen bu sürüm, federasyon yönetim kontrol işlevleri ve veri dağıtım yönetimi dışındaki birçok servisleri desteklemekteydi. Mayıs 1997'deki yeni sürümü ise veri dağıtım hizmetleri dışındaki tüm hizmetleri karşılamaktaydı. Veri dağıtım yönetimi Synthetic Theatre of War (STOW) isimli bir kapsamlı bir projenin konusuydu. STOW'da 300'den fazla federe ve 5000 nesne gerçek zamanlı işleyebilmiştir. Böylece veri dağıtım yönetimi de gerçekleştirilmiş oldu ve Mayıs 1998'de tüm servisleri karşılayan HLA 1.3 sürümü ortaya çıkmıştır (Dahmann, 1998).

HLA 1.3 sürümünü birçok kişi DMSO'dan elde etmiştir. DMSO yeni nesil RTI 1.3'ü 6. sürümüne kadar (RTI 1.3 NG V6) geliştirmiştir. DMSO'nun geliştirdiği RTI'ın son sürümü olan RTI 1.3 NG V6 yazılımına 30 Eylül 2002 tarihine kadar destek vermiş, bu tarihten sonra RTI yazılımının ticari kuruluşlardan temin edilmesi gerektiğini duyurmuştur (DMSO, 2002b). Günümüzde DMSO, RTI desteği vermemekte, IEEE 1516 standardını ve ticari bir RTI yazılımı temin edilmesini önermektedir. RTI olarak ticari sürümlerin kullanılmasını teşvik etmekle DMSO, RTI'ın daha da gelişeceğine inanmaktadır.

IEEE 1516 standardına uyumlu ticari RTI'lar da geliştirilmiştir. Araştırma merkezleri ve üniversiteler de RTI geliştirilmesi konusunda çalışmalar yapmaktadırlar. DMSO resmi internet sitesinde (DMSO, 2006) RTI hizmetleri sağlayan kuruluşların listesi verilmektedir. Mevcut RTI yazılımlarının bir kısmı HLA 1.3 standardına uygun geliştirilmiş bir kısmı da IEEE 1516 standardına geçişi sağlamışlardır.

RTI'ın doğru çalıştığının temini için sertifikalı olması gerekmektedir. Bu sertifikasyonu Amerikan Savunma Bakanlığı Modelleme ve Benzetim Birimi (DMSO) gerçekleştirmektedir. Testten geçirilip doğrulanmış RTI yazılımlarının listesini <https://www.dmso.mil/public/public/transition/hla/rti/statusboard> bağlantısından görmek mümkündür. Sertifikalı bir RTI tüm spesifikasyonları uygulayabilme ve standarda uygun davranabilme garantisi altındadır.

3.4. HLA Kullanımının Gerekçesi

Karmaşık benzetimlerin birçoğu farklı türdeki benzetimlerin birleşiminden meydana gelmektedir. Bu benzetimlerin yazılımı, çalıştığı ortam, ele aldığı konular farklıdır. Farklı amaçlar için tasarlanmış bu benzetimlerin yeni bir benzetim için yeniden kullanılması gerekebilir. Ancak maalesef, bu benzetimlerin daha büyük bir benzetimin parçası olabilecek hale getirilmesi için birçok değişikliğe uğraması gerekecektir. Bazı durumda değişiklik yapmaktansa sistemi yeniden tasarlamak daha yerindedir. Bu sebeple geleneksel sistemler, iki önemli özellik bakımından

yetersizdirler. Bunlar yeniden kullanılabilirlik ve karşılıklı işleyebilme/çalışabilme özellikleridir (MEOSS, 2006)

Yeniden kullanılabilirlik, benzetim modellerinin farklı benzetim senaryolarında ve uygulamalarında tekrar kullanılabilmesidir. Bu özellik sayesinde benzetimler hem daha kısa sürede hem de daha az maliyetle oluşturulmaktadır. Karşılıklı işleyebilme/çalışabilme de yeniden kullanılabilirlik özelliğiyle yakından ilgilidir. Karşılıklı işleyebilme, benzetimlerin yeniden kod yazılmasına ihtiyaç kalmaksızın bir araya getirilmesiyle, birlikte çalışabilmesi demektir. Bu özellik sayesinde farklı platformlarda dağıtık olarak çalışan benzetimler gerçek zamanlı olarak birleştirilebilir.

Yeni bir silah sisteminin geliştirildiğini ve bunun daha önce var olan bir uçakta deneneceğini varsayalım. Bu durumda tüm sistemi yeni baştan tasarlamaya gerek yoktur. Yeni silahın benzetim modeli ile eskiden var olan uçağın benzetim modelini birleştirmek daha uygundur. Benzer şekilde imalat yapan bir fabrika yeni bir tedarikçi edinmiş olabilir. Bu durumda fabrika ve tedarikçilerini yeni baştan tasarlamaya gerek olmamalıdır. Fabrikanın benzetim modeline yeni tedarikçinin benzetim modeli eklenebilmelidir. Bununla birlikte fabrika ve tedarikçileri farklı bilgisayarlarda farklı platformlarda modellenmiş olabilir. HLA burada açıklamaya çalıştığımız türdeki ihtiyaçları karşılamak üzere geliştirilmiş bir standarttır. HLA coğrafi olarak farklı yerlerde bulunan farklı platformlarda farklı yazılımlarla modellenmiş benzetim sistemlerinin bilgi alışverişini sağlayarak birlikte çalışmasını temin eden bir standarttır.

HLA'nın geliştirilmesinde etken olan gereksinimleri şu şekilde sıralamak mümkündür (Calpin vd., 2001):

- Hiçbir tekil benzetim tüm kullanıcıların ihtiyaçlarını tek başına karşılayamaz. Kullanıcılar çok çeşitli ilgi alanlarına sahiptir ve gereksinimleri farklılık gösterir.
- Benzetim geliştiricileri, benzetimi yapılacak konu hakkındaki bilgilerine göre farklılık göstermektedir. Hiçbir geliştirici tüm detaylar hakkında uzman değildir.
- Hiç kimse benzetimlerin tüm kullanım şekillerini bilemez, ön göremez.

- Gelecekteki araçları ve teknolojileri kullanabilmek mümkün olabilmelidir.

Bu gereksinimler, benzetim için yeni bir çerçevenin gerekliliğini ortaya koymuştur. Yeni bir çerçevenin geliştirilmesi konusundaki çalışmalar ve analizler şu sonuçları ortaya çıkarmıştır:

- Büyük benzetim problemlerini küçük parçalara ayırmak mümkün olmalıdır. Böylece problemi tanımlamak, hatasız kurmak ve doğrulamak daha kolay olacaktır.
- Ortaya çıkan parçaları daha büyük, doğru ve kullanışlı benzetim olacak şekilde bir araya getirmek mümkün olmalıdır.
- Bu bileşenleri başka bileşenler ile birleştirip tamamen yeni bir benzetim ortaya çıkarmak mümkün olmalıdır.
- Benzetimlerde ortak olan işlevler benzetimin kaynak kodundan arındırılıp servisler olarak ortaya konmalıdır. Bu servisler genel bir altyapı içerisinde her bir benzetim için kullanılabilir olmalıdır.
- Bileşenler ile bu bileşenlerin etkileşimde bulunduğu genel altyapı arasındaki arayüzler, bu altyapının altında yatan teknoloji veya altyapıyı oluşturma yaklaşımı değişikliklerinden bağımsız olmalıdır.

Yukarıda belirtilen yeni benzetim çerçevesinin sahip olması gerektiği özellikler HLA ile karşılanmıştır.

HLA, hiçbir tekil benzetimin, tüm kullanımların ve tüm kullanıcıların isteklerini karşılayamayacağı fikri üzerine bina edilmiştir. Belirli bir amaç için geliştirilmiş bir benzetim veya bir benzetim dizisi HLA federasyonu kavramı altında (birbiriyle ilişkili bir araya getirilmiş benzetimler) diğer uygulamalara uygulanabilir. HLA'nın amacı, farklı benzetimlerde elverişli olan yeteneklerin tekrar kullanımını destekleyecek bir yapıyı sağlamaktır (Dahmann vd., 1999a).

HLA'nın en belirgin özelliklerinden biri karmaşık benzetim problemlerini daha basit parçalara bölmesidir. Diğer bir bakış açısıyla ifade edecek olursak farklı ortamlardaki benzetimleri bir araya getirerek daha büyük çapta bir benzetim meydana

getirebilmesidir. Böylece, bilgisayar işlemcisinin yükü dağıtılarak azaltılmış olacak, simülatörler gibi ağır donanıma sahip sistemler veya gerçek sistemleri taşıma gereksinimi ortadan kalkacaktır (Canazzi, 1999).

HLA belirli bir uygulamanın veya özel bir yazılım ve programlama dilinin kullanımını gerektirmemektedir. Zaman geçtikçe bu çerçeve içerisinde çok farklı uygulama alanları da mümkün hale gelecektir. HLA, yeni ve değişen kullanıcı ihtiyaçlarını karşılayabilecek benzetim yatırımlarına imkan tanıyan ortak bir mimariye sahiptir. Dahası, uygulamayı değil de mimarinin sadece anahtar öğelerini standart hale getirerek, destekleyici yazılım gelişmeleri (örneğin nesne modeli geliştirme araçları), uygulamaların performans gereksinimlerine uygun hale getirilebilmektedir. Bunlar aynı zamanda işlemci ve network teknolojilerinde yeni yatırımlara yol açabilecektir (Dahmann vd., 1999b).

3.5. HLA ile İlgili Tanımlamalar

Birçok teknoloji gibi HLA'nın da kendine özgü terminolojisi vardır. Bu kısımda HLA'nın belli başlı terimleri açıklanacaktır.

1. Federe (Federate): Tekil bir benzetim uygulaması, simülatör veya uygulama programıdır. Federelerin her biri bağımsız olarak çalışabilmekte; HLA ile daha büyük benzetim uygulaması olacak şekilde birleştirilebilmektedir. Federe sadece bir benzetim modeli olmak zorunda değildir. Veri toplama sistemleri, veri analizi ve görüntüleme sistemleri ve hatta canlı katılımcılar da federe olarak adlandırılmaktadır. Federe tek bir varlığı (entity) modellemiş olabileceği gibi (örneğin bir fabrikadaki bir transfer hattı) birçok varlığı da (örneğin bir fabrikadaki tüm tezgâhlar) modellemiş olabilir (Kuhl vd., 2000) .

2. Federasyon (Federation): Birbiriyle etkileşimli birden fazla federeden oluşan daha karmaşık yapıdaki dağıtık benzetim uygulamasıdır. HLA federasyonu altındaki her bir sisteme federe denir.

3. Federasyonun Çalıştırılması (Federation Execution): Federasyonunun çalıştırıldığı her bir oturuma denmektedir.

4. Federasyon Nesne Modeli (Federation Object Model: FOM): Genellikle kısaca FOM olarak kullanılmaktadır. Federasyon içerisinde, federeler arasında paylaşılan verileri tanımlayan ortak bir nesne modeldir. FOM, ileride daha ayrıntılı olarak ele alınacaktır.

5. Benzetim Nesne Modeli (Simulation Object Model: SOM): Genellikle kısaca SOM olarak kullanılmaktadır. Federenin federasyon içerisinde diğer federelerin paylaşımına sunacağı verilerin tanımlandığı nesne modelidir. Federe ile ilgili daha başka arayüz bilgileri de içermektedir. Her bir federe için tanımlanmış tüm SOM'ların içerisindeki nesneler FOM'da da olmalıdır. SOM, ileride daha ayrıntılı olarak ele alınacaktır.

6. Nesne Model Şablonu (Object Model Template: OMT): Genellikle kısaca OMT olarak kullanılır. Federasyon içerisinde kullanılan verilerin ve bazı arayüz bilgilerinin biçimini, tipini ve yapısını tanımlamak için kullanılan standart bir şablondur. Tüm FOM ve SOM'lar OMT ile uyumlu olacak şekilde yazılır. Verileri ve arayüz protokollerini tanımlamak için OMT iki format sunmaktadır: birincisi tablo biçimi, diğeri ise DIF formatıdır (Data Interchange Format). Tablo biçimi, insanların kolayca okuyabileceği bir dizi tablodan oluşmaktadır. OMT DIF ise metin formatındadır ve FED (Federation Execution Data) dosyası için kullanılmaktadır. RTI, FED dosyasını okuyarak FOM hakkında bilgi sahibi olur. Bununla birlikte destek araçları bilgi alışverişi için DIF biçimini kullanmaktadır. Örneğin nesne modeli geliştirme aracı DIF biçimini kullanarak RTI'a bilgi gönderebilmektedir (Reid, 2000).

7. Nesne (Object): HLA nesne yönelimli modelleme kullanmaktadır. HLA nesnesi, benzetimin modellediği varlıklardan (entity) her biridir. Diğer bir ifade ile HLA nesnesi "benzetim içerisinde oynayan 'aktörleri' temsil eder" denebilir (Reid, 2000). Federenin sahip olduğu veriler nesneler içerisinde gösterilmektedir. FOM, tüm nesne sınıflarını tanımlamaktadır. Her federe, yayımlayacağı ya da başka

federeden alacağı verileri nesne olarak kendi SOM'unda, FOM'daki nesne sınıflarıyla uyumlu olacak şekilde tanımlamalıdır. Nesneler özelliklere sahiptir. Bu özellikler bir takım değerler almaktadır. HLA verileri, bu nesne özellikleri içerisinde saklanmaktadır.

8. Etkileşim (Interaction): Bir HLA etkileşimi, federasyon içerisindeki federelerin gönderdiği veya aldığı yayın mesajlarıdır. Yayımcı federe etkileşim gönderirken, alıcı (abone) federe ise etkileşim almaktadır. Eğer yayımlanan bir etkileşimi hiçbir federe almazsa, etkileşimin taşıdığı veri kaybolur. Tüm etkileşim sınıfları FOM içerisinde tanımlanmalıdır. Benzer şekilde etkileşim yayımlayacak veya alacak her federe, kendi SOM'unda bu etkileşimleri tanımlamalıdır. Etkileşimlerin parametreleri vardır. Etkileşimler verilerini parametrelerinde saklarlar.

9. Çalışma-Anı Altyapısı (Run-time Infastructure: RTI): Genellikle kısaca RTI olarak kullanılır. HLA arayüz spesifikasyonunu uygulayan, dağıtık bir işletim sistemi gibi çalışan bir yazılımdır. Federasyonun çalıştırılmasını RTI sağlamaktadır. RTI ileride daha ayrıntılı olarak ele alınacaktır.

3.6. HLA'nın Teknik Bileşenleri

HLA'da federe olarak adlandırılan çeşitli benzetim sistemleri, federasyon olarak adlandırılan daha büyük benzetime dâhil edilir. Bunun için şunların yapılması gerekmektedir (PITCH, 2006a):

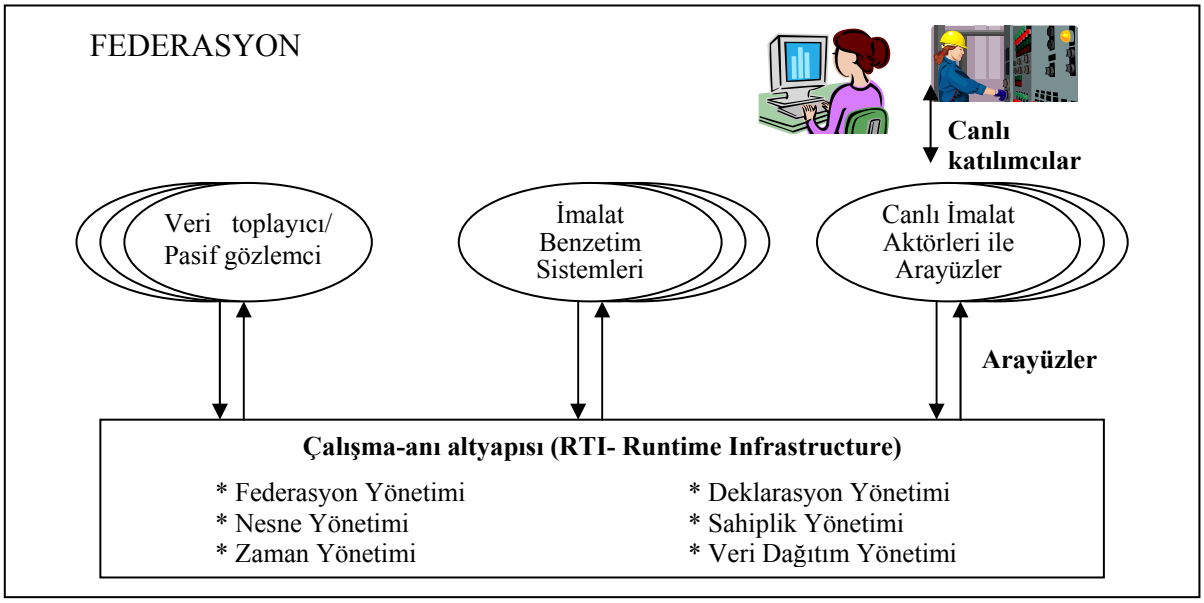
1. İletilecek bilgiler belirlenmelidir. Bu işlemi gerçekleştirmek için Federasyon Nesne Modeli (Federation Object Model: FOM) kullanılır. Bu, federasyonun ortak dilini tanımlayan dosyadır. Bu dosya hangi nesne sınıflarının (örneğin; araba), hangi özelliklerinin (örneğin; marka, hız) ve hangi işlemlerinin (örneğin; gaza basılması) federasyon içerisinde iletileceğinin bilgisini vermektedir. HLA'nın esnek olmasının sebeplerinden bir tanesi de kendi özel federasyonunuz için, ihtiyaçları karşılayacak bir FOM geliştirebilme imkânı sunmasıdır.

2. Federasyon içerisindeki diğer katılımcı sistemlere (federeler) çalışma-anı altyapısı (Run-Time Infrastructure: RTI) yazılımı ile bilgiler iletilmelidir.

Bir HLA federasyonunda şu öğeler bulunmaktadır (Kuhl vd., 2000):

- Destekleyici yazılım olan RTI,
- Federasyondaki federeler arası bilgi değişimi için gerekli, ortak nesne modeli olan FOM
- Federeler

Şekil 3.2'de HLA federasyonunun bileşenleri gösterilmiştir.

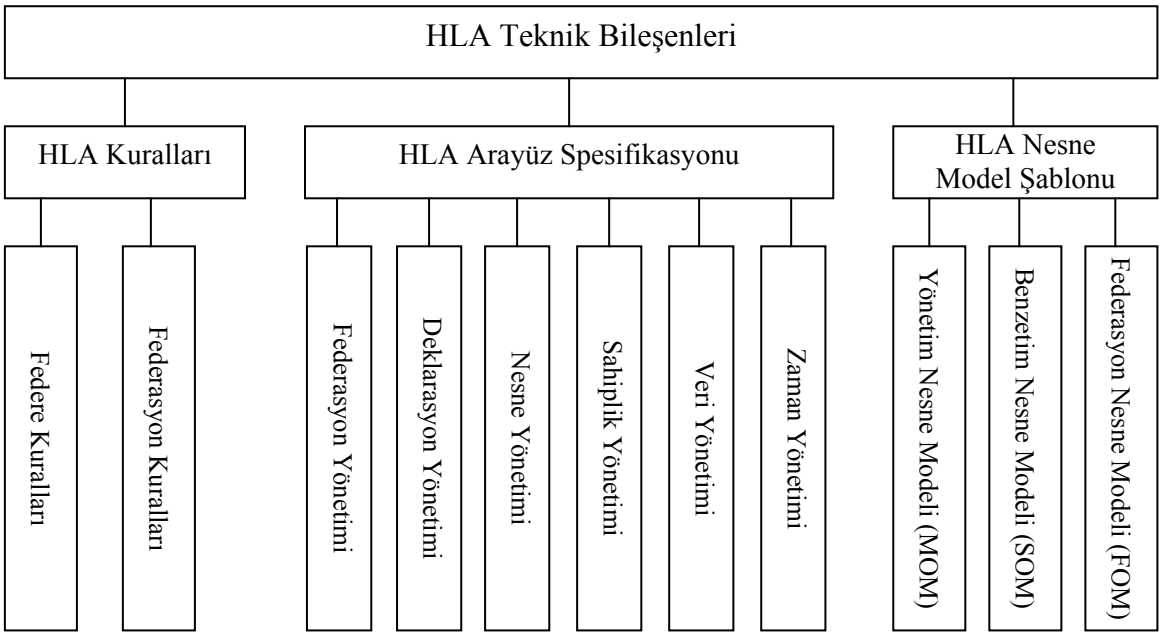


Şekil 3.2. HLA federasyonu ve bileşenleri

Teknik olarak HLA üç bileşenden oluşur (IEEE, 2000):

- HLA kuralları
- HLA federe arayüz spesifikasyonu
- HLA nesne model şablonu (Object Model Template: OMT)

HLA'nın teknik bileşenlerini Şekil 3.3'deki gibi göstermek mümkündür (Xiaoxia ve Qiuhai, 2003). Bu teknik bileşenlerin ayrıntıları aşağıda ele alınacaktır.



Şekil 3.3. HLA'nın teknik bileşenlerinin özet gösterimi

3.6.1. HLA kuralları

HLA, federelerin ve federasyonun uymak zorunda olduğu kurallara sahiptir. Bu kurallar federasyon içerisindeki federelerin doğru bir şekilde iletişim kurmalarını sağlamaktadır. Aynı zamanda federelerin ve federasyonun sorumluluklarını da tanımlamış olmaktadır (Xiaoxia ve Qiuhai, 2003). HLA'in 10 kuralı vardır. Bunlardan beş tanesi federasyona ait kurallar, beş tanesi de federelere ait kurallardır.

Federasyon kuralları bir federasyonu meydana getirebilmek için gerekli zemini temin eder. Bunlar:

- Dokümantasyon gereksinimleri (Kural 1),
- Nesne gösterimi (Kural 2),
- Verilerin karşılıklı değişimi (Kural 3),
- Arayüz gereksinimleri (Kural 4)

– Özellik sahipliği (Kural 5)

ile ilgili kurallardır.

Federe kuralları ise her bir tekil federeyi ilgilendiren kurallardır. Bunlar:

– Dokümantasyon (Kural 6),
– İlgili nesne özelliğinin kontrolü ve transferi (Kural 7, 8 ve 9),
– Zaman yönetimi ile ilgili kurallardır (Kural 10).

HLA karşılıklı işleyebilirlik için yapısal bir temel sağlamaktadır. HLA kurallarının birçoğu bunu temin etmek için oluşturulmuştur. HLA kuralları aşağıdaki gibi açıklanmaktadır (IEEE, 2000).

3.6.1.1. Federasyon kuralları

Kural 1**:** Federasyonlar, HLA Nesne Model Şablonu'na (OMT) göre dokümante edilmiş bir HLA Federasyon Nesne Modeli'ne (FOM) sahip olmalıdır.

FOM, çalışma anında iletimi gerçekleştirilecek veriler ve veri iletimi şartları hakkında, federeler arasındaki mutabakatı belgelendirmelidir. FOM, federe oluşturmanın ana öğelerindendir. HLA hangi verilerin FOM içerisine dahil edileceğinin ön tanımlamasını yapmaz, bu federasyon geliştiricisi ve kullanıcısının sorumluluğundadır. FOM, yeniden kullanılabilirliği desteklemesi bakımından OMT formatına göre dokümante edilmelidir. OMT formatı IEEE Std 1516.2-2000 belgesinde ayrıntılı bir şekilde ele alınmıştır (IEEE, 2001b).

Kural 2**:** Bir federasyonda benzetimle ilgili nesne örneklerinin gösterimleri federeler içerisinde olmalıdır, çalışma anı altyapısında (RTI) olmamalıdır.

HLA altında yatan temel fikirlerden bir tanesi de belirli bir benzetime özgü işlevi genel amaçlı olandan ayırt etmektir. Bu sebeple nesne örneklerinin gösterimi benzetim (daha genel ifadesiyle federe) içerisinde yer alması gerekmektedir. RTI ise federasyon içerisinde nesne örneklerinin iletimi işlevini, tıpkı bir dağıtık işletim

sistemi gibi sağlar. Ancak RTI, federasyon yönetimi nesne modeli ile ilgili nesne özelliklerine sahip olabilir. Nesne gösterimlerinin federelerde yapılmasının sebebi federasyon destek servisleri ile benzetim işlevinin birbirinden ayrılmasıdır.

Federasyon destek servislerinin benzetim işlevlerinden ayrı tutulmasının sebeplerinden biri, federeleri, uygulama amaçları doğrultusunda nesne gösterimlerini yapmalarına odaklanmalarını sağlamaktır. Federasyon destek servisleri ya da RTI servisleri ise federasyonun çalışmasını, zaman koordinasyonunu, veri dağıtımını vb. kapsar. RTI nesne özellikleri ve etkileşimleriyle ilgili verileri RTI servislerine destek olmak için kullanır (örneğin deklerasyon yönetimi). Ancak bu veriler sadece kullanılmış olur, RTI bu verileri değiştirmez.

Kural 3**:** Federasyon çalışıyorken, federeler arasındaki FOM verileri RTI ile iletilmelidir.

HLA altında, federasyona katılan federeler arasındaki iletişim, RTI servisleri marifetiyle iletilen veriler ile gerçekleştirilir. Federeler hangi bilgileri sağlayacaklarını ve hangi bilgileri istediklerini RTI'ya bildirmelidir. RTI da buna göre federeler arası koordinasyonu, senkronizasyonu ve veri iletimini sağlayacaktır.

Doğru verinin doğru zamanlarda sağlanması ve doğru bir şekilde kullanılmasının sorumluluğu federelere aittir. RTI'ın görevi verinin, kullanıcı federelere, deklare edilmiş isteklerine göre iletimini sağlamaktır.

Kural 4**:** Federasyon çalışıyorken, federeler RTI ile HLA arayüz spesifikasyonuna göre etkileşmelidir.

HLA, federeler ile RTI arasındaki arayüzlere destek olan RTI servislerine erişim için standart bir spesifikasyon sağlamaktadır. Bu standart spesifikasyon IEEE Std 1516.1-2000 belgesinde tanımlanmıştır (IEEE, 2001a). Federeler ile RTI arasındaki etkileşimde bu standart arayüzler kullanılmalıdır. Arayüz spesifikasyonu belirli bir federe verisinin iletimi hakkında söz sahibi değildir. Federeler arası veri iletişim gereksinimleri FOM'da tanımlanmalıdır. Arayüzler, federe veri iletim

gereksinimlerinden ayrı tutulmuştur. Bu özellik, arayüz spesifikasyonunun birçok farklı dağıtık modelleme ve benzetim uygulamalarında kullanımını mümkün hale getirmiş olur.

Federeler ile RTI arasındaki ortak standart arayüz, uygulama program arayüzleri (Application Program Interface: API) ile gerçekleştirilir. Böylece HLA bağımsız bir geliştirme ve uygulama sağlamış olur.

Kural 5**:** Federasyon çalışıyorken, bir nesne özelliği herhangi bir anda en fazla bir federe tarafından sahiplenilmelidir.

HLA, aynı nesnenin farklı özelliklerine, farklı federelerin sahip olmasına izin verir. Ancak federasyon içerisinde bilgi tutarlılığının temin edilmesi için bir nesne özelliğinin herhangi bir anda en fazla bir sahibi (özelliğin değerini değiştirme hakkına sahip federe) olmalıdır. Bununla beraber HLA, sahipliği, çalışma esnasında dinamik olarak bir federeden diğer federeye aktarma mekanizmasına da sahiptir. Bu özellik ile esneklik sağlanmış olur ve farklı kullanıcı ihtiyaçları için çeşitli benzetim kombinasyonları denenebilir.

3.6.1.2. Federe kuralları

Kural 6**:** Federeler, HLA nesne model şablonuna (OMT) göre dokümante edilmiş bir benzetim nesne modeline (Simulation Object Model: SOM) sahip olmalıdır.

Federeler ile kastedilen, federasyona dahil olan benzetimler ve diğer uygulamalardır (benzetim yöneticisi, veri toplayıcı programlar, canlı varlık arayüzleri ve pasif gözlemciler, vb.). HLA, her federenin bir benzetim nesne modeline (SOM) sahip olmasını gerekli tutmaktadır. SOM, nesne sınıflarını, sınıf özelliklerini ve federenin federasyon içerisinde yapacakları etkileşim sınıflarını içerir.

HLA, SOM içerisinde hangi verilerin bulunacağının ön tanımlamasını yapmaz. SOM içerisinde bulunacak verilerin tanımlanması sorumluluğu benzetim geliştiricisine

aittir. HLA, oluşturulan SOM'ların HLA nesne model şablonu (OMT) formatında olmasını gerekli kılmaktadır.

HLA'nın en büyük amacının benzetimlerin karşılıklı çalışmasını ve yeniden kullanılabilir olmasını sağlamak olduğu bilinmektedir. HLA bu amacı, federeler içerisindeki bilgi gösterimlerine erişime izin vererek gerçekleştirmektedir.

SOM oluşturmakla federeler kendi yeteneklerinin en bariz temel özelliklerini belgelendirmiş olmaktadırlar. Bu, federelerin ne iş yapacağının kolayca anlaşılmasını sağlayacaktır. HLA, yeniden kullanımı destekleyen doğru ve kullanışlı bilginin geliştirilmesi ve korunmasının, benzetim geliştiricilerinin en önemli önceliklerinden biri olduğu kabulüne dayanmaktadır. Benzetim nesne modelleri (SOM) ile bilginin gösterimi de bunu sağlamaktadır.

Kural 7**:** Federeler kendi benzetim nesne modellerinde (SOM) belirlendiği şekliyle özellikleri güncellemek ve/veya yansıtmak ve etkileşim iletmek ve/veya almak yeteneğine sahip olmalıdır.

HLA, federelerin kendi içsel kullanımları için geliştirilmiş nesne gösterimlerinin ve etkileşimlerinin, diğer federelerin nesneleri ile birlikte kullanılmasına izin vererek dışsal kullanımı destekler. Federasyonun çalışması da bu şekilde mümkün hale gelmektedir. Dışsal etkileşim yetenekleri federelerin benzetim nesne modellerinde (SOM) tanımlanmalıdır. Federeler, içsel olarak hesaplanarak güncelleştirilmiş bir nesne özelliğinin değerini yaymak ve federasyon içerisindeki diğer federelerin etkileşim gösterimlerini de kullanabilmekle yükümlüdür.

Kural 8**:** Federeler kendi benzetim nesne modellerinde (SOM) belirlendiği şekliyle özelliklerin sahipliğini, federasyon çalışıyorken dinamik olarak transfer edebilme ve/veya alabilme yeteneğine sahip olmalıdır.

HLA, farklı federelerin aynı nesnenin farklı özelliklerine sahip olmasına izin vermektedir. Bu özellik sayesinde belirli bir amaç için tasarlanmış bir benzetim ile başka bir amaç için tasarlanmış diğer bir benzetim, yeni bir gereksinimi karşılamak

üzere bir araya getirilebilmektedir. Nesne özelliklerinin sahipliği, federasyon çalışıyor halde iken, bir federeden diğer federeye aktarılabilir. Bir federenin nesne özelliğinin sahipliği ve yansıtılabilirliği ile bu özelliğin çalışma anında dinamik olarak transfer edilebilirliği ilgili federenin benzetim nesne modelinde yazılı olmalıdır.

Kural 9**:** Federeler kendi benzetim nesne modellerinde (SOM) belirlendiği şekliyle güncelleme yapabildiği nesne özelliklerinin şartlarını değiştirebilmelidir.

HLA, federelerin nesne özelliklerine sahip olmasına (bunların değerini güncelleştirebilmesine) ve bu değerlerin RTI yoluyla diğer federelere iletmesine imkân verir. Farklı federeler, nesne özelliğinin güncellenmesi durumunda farklı şartlar belirleyebilir. Kullanımı geniş benzetimler, farklı federasyonların gereksinimlerini karşılayabilmek için kullanıma açık nesne özelliklerini ilettiklerinde, nesne özelliğinin şartlarını ayarlayabilmelidir. Bir federenin belirli bir nesne özelliğinin şartlarının güncellenebilir olduğu, bu federenin benzetim nesne modelinde yazılı olmalıdır.

Kural 10**:** Federeler, federasyonun diğer üyeleri ile veri alışverişini koordine edebilecek şekilde yerel (lokal) zamanı yönetebilmelidir.

HLA zaman yönetim yapısı, kendi içlerinde farklı zaman yönetim mekanizmaları kullanan farklı federelerin birlikte çalışabilmesini desteklemek amacındadır. HLA bu özelliği, federelerin her zaman yönetim hizmetini karşılayabilmesini sağlayacak kesin gerekliliklere uymasını sağlayarak gerçekleştirir. Bu amaçla, farklı zaman yönetimi kullanan federeler arası etkileşimi destekleyecek bir zaman yönetimi yaklaşımı geliştirilmiştir. Federelerin kendi kullandıkları zaman akış mekanizmasını (adım adım artan, olay artırımlı, bağımsız artan…) RTI'a açık bir şekilde bildirme zorunluluğu bulunmamaktadır ancak, RTI servislerini kullanmak zorundadır.

3.6.2. HLA arayüz spesifikasyonu

HLA'nın iki temel amacından biri yeniden kullanılabilirliktir. Yeniden kullanılabilirlik ile akla en başta verilerin, benzetim modellerinin ve yazılımların yeniden kullanılması gelse de en önemli zaman ve maliyet kazancından biri de yazılım altyapısının yeniden kullanılabilirliğidir. Yeni bir benzetimin geliştirilmesi maliyetinin yaklaşık %75'i altyapısı ile ilgili olduğu tahmin edilmektedir. Diğer %25'lik kısmı ise benzetimin işlevselliği ile ilgilidir. HLA'nın diğer temel amacı ise karşılıklı işleyebilirliktir. Bu, federelerin işlevselliği ile iletişim işlevselliğinin birbirlerinden ayrılması ile gerçekleştirilmiştir. İletişimin işlevselliği HLA arayüz spesifikasyonunun altı servisi ile gerçekleştirilmektedir (Wilcox vd., 2000).

HLA arayüz spesifikasyonu federeler ile çalışma anı altyapısı (RTI) arasındaki arayüzleri tanımlar. Bu arayüzlere RTI hizmetleri demek de mümkündür. HLA arayüz spesifikasyonu IEEE, 2001a belgesinde ayrıntılı olarak açıklanmıştır. RTI, spesifikasyona uyan bir yazılımdır ancak kendisi spesifikasyonun bir parçası değildir (MEOSS, 2006).

Arayüz spesifikasyonu federelerin federasyon ile ve birbirleriyle nasıl etkileşeceklerini belirlemektedir. Federasyonun oluşturulmasını ve yok edilmesini sağlamaktadır. Nesnelerin dağıtılmasının temin edilmesini ve federeler arası yönetilmesini desteklemektedir. Arayüz spesifikasyonu federasyonun zaman yönetimini de gerçekleştirmektedir.

HLA federeler arası etkileşimde standart bir uygulama programı arayüzü (Application Program Interface: API) kullanılmasını gerektirmektedir. Bu, dağıtık bir federasyonda federeler arası bilginin etkin bir şekilde iletilmesini ve federelerin yeniden kullanımını destekleyecektir. RTI, dağıtık bir işletim sisteminin uygulamalara hizmet sunması gibi federelere hizmetler sunmaktadır. Bu arayüzler altı temel RTI hizmet gurubu altında düzenlenmiştir (IEEE, 2001a):

1. Federasyon Yönetimi
2. Deklarasyon Yönetimi

3. Nesne Yönetimi
4. Sahiplik Yönetimi
5. Zaman Yönetimi
6. Veri Dağıtım Yönetimi

Bu RTI hizmetleri daha geniş biçimiyle ele alınacaktır, ancak daha önce RTI'ın ne işe yaradığına bakmakta fayda vardır.

3.6.2.1. RTI'nın işlevleri

HLA'nın kendisi bir çerçeve sunmaktadır, yazılım değildir. Çalışma-anı altyapısı (RTI), HLA arayüz spesifikasyonlarının bir uygulaması olarak karşımıza çıkar ve HLA'nın çekirdeğidir. RTI aynı anda birden fazla federasyonu çalıştırma yeteneğine sahip olabilmektedir. RTI, dağıtık bir işletim sistemi gibi davranarak federasyon içerisindeki katılımcı benzetim sistemlerinin (federelerin) birbirlerine bağlanmasını ve bilgi değişimini sağlamaktadır. Kendilerinde var olan nesnelerle, bunların özellikleriyle ve etkileşimlerin değişimi ile iletişim kurulmaktadır. RTI, federasyonun zamanını senkronize edebilmektedir.

RTI, bir özelliğin güncelleştirilme sorumluluğunun federeler arası transfer edilmesi, veri dağıtımının yönetimi, federasyonun yönetimi gibi daha başka gelişmiş işlevlere de sahiptir.

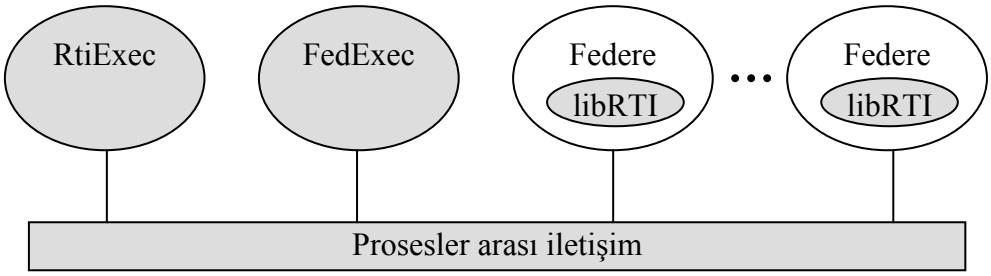
RTI'yı HLA'nın omurgası gibi düşünmek mümkündür. RTI üç bileşene sahiptir:

- RTI çalışma süreci; RtiExec
- Federasyon çalışma süreci: FedExec
- Kütüphane: libRTI

RtiExec global bir süreçtir; federasyonların çalıştırılmasını (FedExec) ve sonlandırılmasını sağlamaktadır. Belirli bir federasyonun federeleri RTI bileşenlerini başlatabilmek için RtiExec vasıtasıyla iletişim kurmaktadırlar. RtiExec aynı zamanda FedExec'lerin benzersiz isimde olmasını temin etmektedir. Birçok federasyonun aynı

RTI içerisinde aynı anda çalışması mümkün olduğundan her bir FedExec'in farklı isimlerde olması önemlidir.

FedExec ise bir federasyona federelerin katılmasını ve ayrılmasını sağlayarak federasyonu yönetmektedir. Federeler arasında veri iletimini de denetlemektedir. HLA arayüz spesifikasyonunun yukarıda maddelenen ve aşağıda açıklanacak olan işlevlerini ise libRTI sağlamaktadır. Federe (benzetim) geliştiricileri RTI özelliklerini, diğer bir deyişle HLA arayüz spesifikasyonlarını libRTI ile modellerine dahil etmektedirler. Federasyonun çalışması esnasında libRTI, RtiExec ve FedExec arasında iletişim gerçekleşmektedir. RTI'ın bileşenleri ve federeler ile libRTI vasıtasıyla etkileşimi Şekil 3.4'te gösterilmiştir.



Şekil 3.4. RTI'ın bileşenleri ve federeler ile etkileşimi

Bir federasyonun çalıştırılmasında aşağıdaki adımlar izlenmektedir:

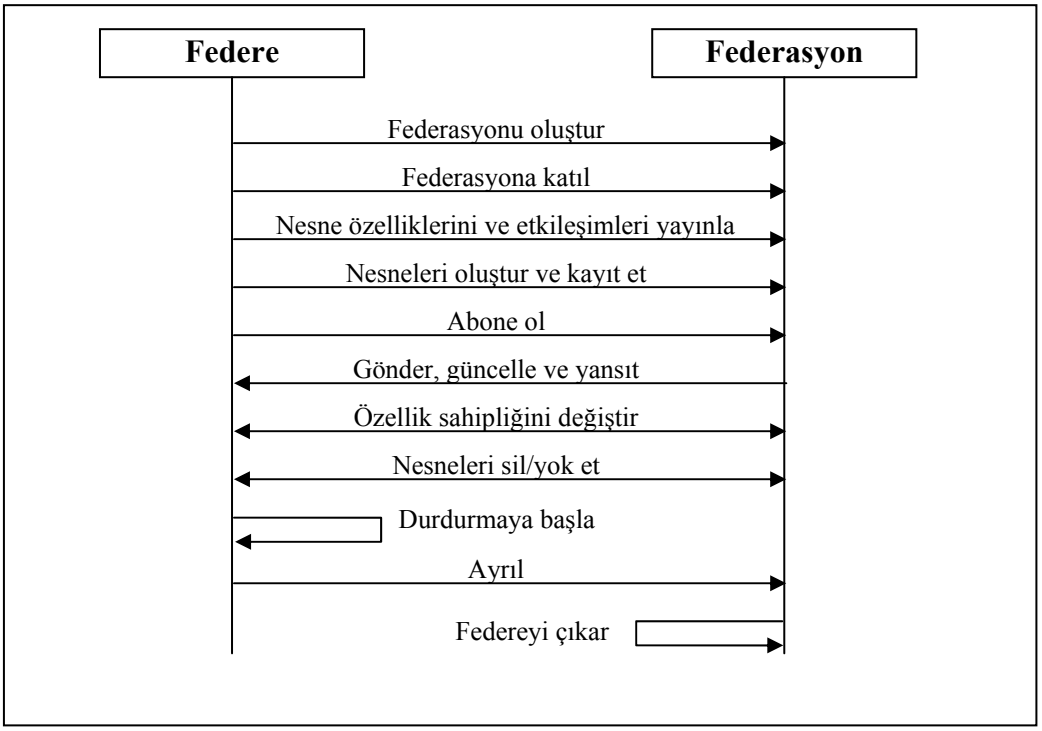
- RTI bir terminalden belirli bir IP adresi ile çalıştırılır. Böylece diğer federeler RTI'ın konumunu tespit edebilecek ve işlevlerinden yararlanabilecektir. Bu işlem RtiExec sürecini başlatmaktadır.

- Bir federe RtiExec'e kayıt olur ve bir federasyon oluşturur. Bu işlem FedExec sürecini başlatarak federasyonun ismini RtiExec'te kaydeder. Bu isim benzersizdir; aynı isimle başka bir federasyon oluşturulamaz. FedExec aynı zamanda RtiExec'e iletişim adresini de kaydetmiş olmaktadır.

– Diğer katılımcı/üye federeler artık yeni oluşturulan federasyona dahil olabileceklerdir. Her federe RtiExec'ten federasyonun (FedExec) adresini alacak ve federasyona katılma yönergesini işletecektir.

– Federasyonun içerisindeki federeler işlemlerini bitirdiklerinde federasyondan ayrılmalıdırlar (resign komutu ile). Tüm federeler ayrıldığında ve federasyonun işlemi bittiğinde federasyon RtiExec'ten kaldırılmalıdır (destroy komutu ile).

Federe ile federasyon arasındaki etkileşim Şekil 3.5'te verilmiştir.



Şekil 3.5. Federe ile federasyon arasındaki etkileşim

RTI'ın hizmetleri altı grup altında toplandığı daha önce belirtilmişti. RTI'ın ne işe yaradığı belirtildikten sonra aşağıda RTI hizmetlerinden bahsedilmektedir.

3.6.2.2. Federasyon yönetimi

Federasyon yönetimi, bir federasyonu oluşturmak, dinamik olarak kontrol etmek, federasyonu modifiye etmek ve federasyonun çalışmasını sonlandırmak için gerekli servisleri barındırmaktadır. İşlevlerini şu şekilde sıralamak mümkündür:

- Federasyonu oluşturmak ve sonlandırmak,
- Federelerin federasyona katılmasını ve federasyondan ayrılmasını sağlamak,
- Federasyonun çalışmasını diğer federeler ile senkronize etmek,
- Federasyon durumunu saklamak

3.6.2.3. Deklarasyon yönetimi

Deklarasyon yönetimi, bilgilerin federeler arası etkin bir biçimde iletilmesiyle ilgilidir. Federeler içerisinde, federasyon çalışıyorken hangi verilerin bu federe tarafından diğer federelere iletileceği (yayınlama) ve hangi verileri diğer federelerden isteyeceği (abone olma) tanımlanmaktadır. Deklarasyon yönetimi servisleri ile şunlar gerçekleştirilebilir:

- Nesne sınıfları yayınlanır (özelliklerini güncelleyeceği nesneler belirtilir)
- Nesne sınıflarına abone olunur/alınır (başka federelerde oluşturulmuş nesne özelliklerini alacağı belirtilir)
- Etkileşim sınıfları yayımlanır (diğer federelere göndereceği etkileşimler belirtilir)
- Etkileşim sınıfları alınır
- Tüm yayınlar ve alımlar durdurulabilir.

3.6.2.4. Nesne yönetimi

Nesne yönetimi, nesnelerin oluşturulması, silinmesi, tanımlanması, güncellenmesi gibi nesne seviyesindeki servisleri sağlamaktadır. Federe tarafından bir nesne oluşturulduğunda nense RTI ile kaydedilir. Diğer federeler ise, eğer bu nesneye abone iseler, deklarasyon servisi ile bu nesnenin varlığını keşfederler. Nesne

özelliklerinin veya etkileşimlerin güncellenmesi nesne yönetimi servisi ile gerçekleştirilir. Nesne yönetimi servisleri şunları yerine getirmektedir:

- Federelerin yayımladıkları nense sınıflarına uygun olarak, nesne örneklerinin oluşturulması ve silinmesi
- Federelerin abone oldukları nense sınıflarına uygun olarak nesne örneklerinin araştırılması
- Nesne özellik değerlerinin güncellenmesi (yayımcılar için) ve yansıtılması (alıcılar için)
- Etkileşim gönderimi ve alımı.

3.6.2.5. Sahiplik yönetimi

Sahiplik yönetimi servisleri ile RTI, HLA nesne özelliklerinin sahipliğini, diğer bir ifadeyle, nesne özellik değerinin güncellenmesi sorumluluğunun hangi federede olduğunu kontrol etmektedir. Bir nesne özelliğinin güncellenmesini sadece, o özelliğin sahipliğini elinde bulunduran nesne gerçekleştirebilmektedir. Sahiplik yönetimi, federasyon çalışıyorken bir nesnenin özelliğinin sahipliği bilgisini (bu özelliğin hangi federasyon tarafından güncellenebileceği) dinamik olarak bir federeden diğer federeye aktarma ile ilgili hizmetler sunmaktadır. Sahiplik yönetimi servisleri deklarasyon yönetimi servisleri ve nesne yönetimi servisleri ile etkileşim halinde çalışmaktadır.

3.6.2.6. Zaman yönetimi

Zaman yönetimi, veri alışverişlerinin senkronizasyonunu sağlamakta ve benzetim zamanının ilerleyişini koordine etmektedir. Federasyon zaman çizgisi üzerinde bir federenin alacağı pozisyonunu RTI, zaman yönetimi servisi ile gerçekleştirmektedir. Zaman ilerleyişi nesne yönetim servisleriyle koordineli olarak gerçekleştirilmektedir.

Bir nesne özelliğini güncelleyen veya etkileşim gönderen federe zaman ayarlayıcı (TimeRegulation) durumundadır. Bilgi güncelleme veya etkileşim gönderme zaman etiketleriyle gerçekleştirilmektedir. Bu zaman etiketi, federasyon çalışmasının zaman

ekseninde bir noktaya denk gelmelidir. Bir nesne özelliğini alan (yansıtan) veya etkileşim alan federe ise zaman kısıtlı (TimeConstrained) durumundadır. Bu federeler, yayımlanan nesne özelliklerinin veya etkileşimlerin zaman etiketine bağımlıdırlar. Diğer bir ifade ile zaman kısıtlı federelerin zaman ilerleyişi diğer federelere bağlıdır. Diğer federeler ile eşzamanlı çalışması gereken federelerde kullanılmaktadır. Zaman kısıtlı federe, RTI izin verene kadar zamanını ilerletememektedir. Zaman yönetimi servisleri, federasyon çalışıyorken, zaman ayarlayıcı ve zaman kısıtlı federeler arası koordinasyonu sağlamaktadır.

Genel olarak bir federe "ayarlayıcı (regulating)" veya "kısıtlı (constrained)" veya "ayarlayıcı ve kısıtlı (regulating and constrained)" ya da "ne ayarlayıcı ne de kısıtlı (neither regulating nor constrained)" zaman servislerinden birini kullanabilmektedir. Bir federasyon, farklı zaman servisini kullanan federeleri içerebilmektedir (DMSO, 2002a).

3.6.2.7. Veri dağıtım yönetimi

Veri dağıtım yönetimi, federasyonun çalışması esnasında verilerin federeler arasında etkin bir şekilde yol olması için gerekli servisleri sağlamaktadır. Federeler veri dağıtım yönetimini, uygun olmayan verilerin iletimini ve alımını azaltmak için kullanabilmektedirler. Deklarasyon yönetimi nesne sınıf özellikleri seviyesinde veri uygunluğu bilgisi sağlarken veri dağıtım yönetimi buna, nesne örneklerinin özellikleri seviyesinde veri gereksinimlerini daha da rafine etme yeteneği katar.

Bu servislerin tümü federasyon hayat döngüsü boyunca kullanılmamaktadır. Bu servislerin kullanım aşamaları Şekil 3.6'da gösterilmiştir. HLA, veri azaltma ve filtreleme teknikleri ile daha önceki dağıtık benzetim mimarilerine göre daha iyi ölçeklenebilirlik sağlamaktadır. Bu iki şekilde sağlanmaktadır. Birincisi, kısmi güncellemeler iletilmektedir, diğer bir ifade ile sadece değişen veriler iletilerek mesajın büyüklüğü azaltılmaktadır. İkincisi ise, Deklarasyon Yönetimi ve Veri Dağıtım Yönetimi servisleri ile önemli ölçüde veri filtreleme gerçekleştirilmekte, böylece mesaj sayısı azaltılmaktadır (Wilcox vd., 2000).

Şekil 3.6. Federasyon hayat döngüsü esnasındaki RTI servislerinin durumu

3.6.2.8. HLA Nesne Model Şablonu (OMT)

Yeniden kullanılabilirlik ve karşılıklı/birlikte çalışabilme, federenin içerisinde kullanılan ve diğer federelere iletilen nesnelerin ve etkileşimlerin ortak bir format ile ayrıntılı bir şekilde tanımlanmasını gerektirmektedir. Bu, federeler ile RTI arasındaki arayüzler ve RTI servislerinin spesifikasyonu için de gereklidir. HLA nesne model şablonu (Object Model Template: OMT) HLA nesne model bilgilerinin yazılması için bir standart sağlamaktadır.

Daha açık bir ifadeyle OMT, FOM ve SOM'un nasıl yazılması gerektiğini açıklamaktadır. Bununla birlikte OMT, nesne özellikleri ve etkileşim parametreleri için temel veri türlerinin ve karmaşık veri türlerinin nasıl oluşturulup tanımlanacağını belirtmektedir (Canazzi, 1999).

HLA nesne modellerinin standart bir yapıda olmasını gerektiren çeşitli sebepler vardır (IEEE, 2001b):

- Paylaşılmış verilerin alışverişini belirleyen, ortak kabul gören bir mekanizma oluşturmak ve federasyonun üyeleri arasında genel koordinasyonu sağlamak,
- Federasyonun üyelerinin potansiyel yeteneklerini açıklayan ortak standart bir mekanizma sağlamak,
- HLA nesne modelleri geliştirmek için ortak uygulama araçları geliştirilmesini kolaylaştırmak.

HLA nesne modelleri, nesne sınıfları, bunların özellikleri ve etkileşimleri ile ilgili bilgilerin belirlendiği bir takım birbiriyle ilişkili bileşenlerden oluşmaktadır. Bu bileşenlerin bilgi içeriğini çeşitli formatlarda göstermek mümkündür. Ancak HLA bu bilgileri tablolar şeklinde gösterilmesini gerektirmektedir. HLA nesne model şablonu aşağıdaki tablolardan oluşmaktadır (IEEE, 2001b):

- Nesne modeli tanımlama tablosu: HLA nesne modeli ile ilgili önemli tanımlayıcı bilgilerin yer aldığı tablodur.
- Nesne sınıf yapısı tablosu: Tüm benzetim veya federasyonun nesne sınıflarının isimlerinin kaydedildiği ve bunların sınıf-altsınıf ilişkilerinin tanımlandığı tablodur.
- Etkileşim sınıf yapısı tablosu: Tüm benzetim veya federasyonun etkileşim sınıflarının isimlerinin kaydedildiği ve bunların sınıf-altsınıf ilişkilerinin tanımlandığı tablodur.
- Özellik tablosu: Bir benzetim veya federasyonun nesne özelliklerinin tanımlandığı tablodur.
- Parametre tablosu: Bir benzetim veya federasyonun etkileşim parametrelerinin tanımlandığı tablodur.
- Boyut tablosu: Nesne örneğinin özelliklerinin ve etkileşimlerin filtrelenmesi için boyutların tanımlandığı tablodur.
- Zaman gösterim tablosu: Zaman değerlerinin gösterimlerinin tanımlandığı tablodur.
- Kullanıcı-tanımlı etiket tablosu: HLA servislerinde kullanılan etiketlerin gösterimlerinin tanımlandığı tablodur.
- Eşzamanlama tablosu: HLA eşzamanlama servislerinde kullanılan gösterimlerin ve veritiplerinin tanımlandığı tablodur.
- Taşıma tipi tablosu: Kullanılan taşıma mekanizmasının açıklandığı tablodur.
- Anahtar tablosu: RTI tarafından kullanılan parametreler için başlangıç ayarlarının tanımlandığı tablodur.
- Veritipi tablosu: Nesne modeli içerisindeki veri gösterimlerini detaylandırmak için kullanılan tablodur.

- Not tablosu: OMT tablosundaki herhangi bir öğenin açıklamasını detaylandırmak için kullanılan tablodur.
- FOM/SOM sözlüğü: HLA nesne modelinde kullanılan tüm nesnelerin, özelliklerin, etkileşimlerin ve parametrelerin açıklandığı tablodur.

Bir federasyon geliştirmek için temelde nesne sınıf yapısı tablosu, özellik tablosu, etkileşim sınıf yapısı tablosu ve parametre tablosu yeterli olmaktadır. İhtiyaç halinde diğer tablolar da tanımlanabilmektedir.

Bir nesne modeli oluşturmak için gerekli kurallar ve terminoloji ayrıntılı bir şekilde IEEE Std 1516.2-2000 belgesinde açıklanmaktadır. Daha ayrıntılı bilgi için bu kaynağa başvurulabilir (IEEE, 2001b).

Federasyon oluşturulurken OMT'ye uygun şekilde üç nesne modeli tanımlamak gerekmektedir (Xiaoxia ve Qiuhai, 2003):

- Federasyon Nesne Modeli (Federation Object Model: FOM)
- Benzetim Nesne Modeli (Simulation Object Model: SOM)
- Yönetim Nesne Modeli (Management Object Model: MOM)

Bunlardan MOM federasyonun yönetimi ve kontrolü için gerekli olan nesneleri ve etkileşimleri sağlamaktadır. Önemli oldukları için FOM ve SOM aşağıda daha ayrıntılı olarak ele alınmaktadır.

3.6.2.9. Federasyon Nesne Modeli (FOM)

FOM, HLA'nın temel bileşenlerinden biridir. Her federasyonun bir FOM'a sahip olması gerekmektedir. FOM, federeler arası RTI yoluyla ortak veri iletimini ve federasyon içerisinde paylaşıma açık / ortak nesneleri, özellikleri ve etkileşimleri tanımlamaktadır. Bir anlamda federelerin benzetim nesne modellerinin kesişimidir. Federasyonun tasarım zamanında ortak nesneler ve etkileşimler belirlenmelidir.

FOM evrensel bir tanımlamadır. RTI ve tüm federeler, nesnelerin ve etkileşimlerin belirli bir şekilde tanımlı olmasına gereksinim duymaktadır. FOM, federasyon içerisinde bilinen tüm nesne ve etkileşim sınıflarını tutmakta ve her nesnenin sahip olduğu tüm özellikleri ve her etkileşimin sahip olduğu tüm parametreleri vermektedir. Bununla birlikte özelliklerin ve parametrelerin biçimlerini ve veri türlerini de kesin bir şekilde tanımlamaktadır.

3.6.2.10. Benzetim Nesne Modeli (SOM)

Her federenin de bir SOM'u olmalıdır. SOM, federenin en belirgin özelliklerini tanımlamakta, dışsal olarak kullanılabilecek nesne ve etkileşimleri vermektedir. Federasyonun bir parçası olarak federenin diğer federeler ile ne şekilde etkileşimde bulunacağı bilgisi SOM'da belirtilmektedir. Federenin diğer federelere ileteceği nesne özelliklerini, diğer federelerden elde edeceği nesne özelliklerini ve başlatacağı etkileşimler ile diğer federelerden edineceği etkileşimleri içerisinde barındırmaktadır. Bununla birlikte SOM, RTI ile ne şekilde etkileşeceğiyle ilgili bir takım bilgiler de içermektedir.

3.6.2.11. HLA nesneleri

Her nesne FOM içerisinde bulunan nesne sınıfının bir örneğidir/durumudur. Nesne sınıflarına nesne model tasarımcısı karar vermektedir. Her nesne sınıfının bir dizi özelliği vardır. Özellik, nesne durumunun belirgin ve fark edilebilir bir parçasıdır. Federasyon açısından, bir nesnenin tüm özelliklerinin değerleri, o nesnenin tam olarak durumunu verecektir. Herhangi bir anda nesnenin bir özelliğini güncelleme sorumluluğu sadece bir federeye ait olmalıdır. Bu federe yeni değerleri diğer federelere, federasyon çalışıyorken, RTI servisleri marifetiyle iletecektir. Nesne özelliğine yeni değer üretilmesine güncelleme, yeni değerin diğer federeler tarafından alınmasına da yansıtma denir. Nesne özelliğini güncelleme yetkisi aynı anda sadece bir federenin elindedir ve bu federe o nesne özelliğinin sahibidir denir. Nesne özelliğinin sahipliğinin başka federeye aktarılması işlemi RTI servisleriyle gerçekleştirilir (IEEE, 2001a).

HLA nesneleri, nesne-yönelimli programlama dillerindeki nesnelere, nesne özellikleri içermeleri ve nesne sınıflarını temel almaları bakımından benzemektedir. Bir nesne, nesne sınıfının tüm özelliğini kalıtsal olarak almaktadır. HLA nesnelerinde federe ve federasyonun bilgi alışverişi için yetenek gereksinimlerine odaklanılmıştır. Örneğin SOM'un amacı diğer federelere sağlayacağı nesne ve etkileşimleri tanımlamaktır. Bu federenin nasıl tasarlandığı ve içsel olarak nasıl işlediğiyle alakalı olarak (geleneksel nesne modelleri için) tasarım dokümanlarına ve ilgili diğer dokümanlara ihtiyaç vardır (IEEE, 2001b).

3.7. Federasyon Geliştirme ve Çalıştırma Süreci (FEDEP)

HLA'nın kullanım alanı yaygınlaştıkça birçok benzetim kullanıcıları HLA'ya geçiş yapmak durumunda kalmaktadır. Birçok muhtemel HLA kullanıcıları için HLA'ya geçiş, iş yapma tarzlarını değiştirmek anlamına gelecektir. Mevcut benzetimlerden HLA'ya geçiş arttıkça yeni kullanıcılar için bir rehber gerekli hale gelmiştir. IEEE HLA'yı standart olark kabul ettikten sonra HLA temelli dağıtık benzetim geliştiricileri için IEEE Std 1516.3 belgesini tavsiye niteliğinde geliştirmiştir (IEEE, 2003). Bu belge HLA FEDEP sürecini açıklamaktadır; HLA federasyonlarının oluşturulmasını ve çalıştırılmasını genelleştirilmiş süreçlele açıklamaktadır. IEEE bu belgeyi hazırlamakla, HLA kullanan organizasyonların kendi yönetim ve sistem tasarımı tarzlarını değiştirmeyi amaçlamamaktadır, sadece genel bir çerçeve sunmaktadır. HLA FEDEP katı kurallar koymaktan kaçınmakta, HLA kullanıcıları için "tüm bedenlere uyan" bir süreç belirlememektedir. FEDEP, HLA federasyon uygulamalarının gereksinimlerine göre değiştirilebilir genel sistem mühendisliği metodolojisi tanımlamasıdır (IEEE, 2003).

HLA'nın başlangıcından itibaren, federasyon geliştirilmesiyle ilgili tecrübeler neticesinde, federasyon geliştirme ve icra süreci (FEDEP) şekillenmiştir. DMSO tarafından tanımlanan FEDEP süreci (DMSO, 1999), IEEE Std 1516.3 belgesinde daha da geliştirilimiştir (IEEE, 2003). FEDEP dağıtık benzetim geliştirmek için genel bir çerçeve tanımlayan sistem mühendisliği süreç modelidir (Dahmann vd., 1999a).

HLA federasyonu geliştirmek ve çalıştırmak için izlenmesi gereken temel yolu yedi aşamalı olarak belirlemek mümkündür. Aşağıda açıklanan yedi aşama Şekil 3.7'de görülmektedir (IEEE, 2003).

1. Adım: Federasyonun Amaçlarını Belirlemek**.** Federasyon kullanıcıları, sponsor ve federasyon geliştirme takımı amaçları belirlemeli ve bu amaçlar üzerinde mutabakat sağlamalıdırlar. Belirlenen bu amaçları gerçekleştirmek üzere nelerin yapılması gerektiğini yazılı hale getirmelidirler.

2. Adım: Federasyonun Kavramsal Analizini Gerçekleştirmek**.** Problem uzayının karakteristiklerine dayanarak gerçek dünya probleminin uygun bir gösterimi geliştirilmelidir.

3. Adım: Federasyonu Tasarlamak. Federasyonun katılımcıları olan federeler tasarlanır ve gerekli işlevler federelere atanır. Federasyonun geliştirilmesi ve uygulanması için bir plan hazırlanır.

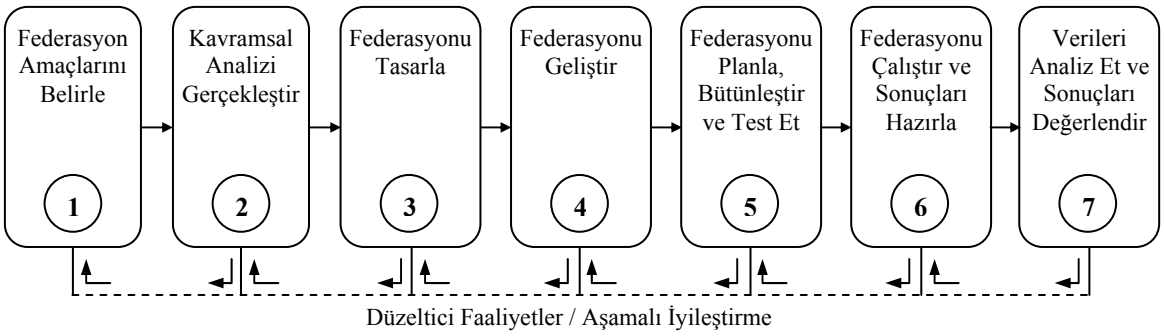
4. Adım: Federasyonu Geliştirmek. Federasyon Nesne Modeli (FOM) geliştirilir, federe anlaşmaları sağlanır, yeni federeler tasarlanır ve/veya var olan federelere değişiklikler uygulanır.

5. Adım: Federasyonu Planlamak, Bütünleştirmek ve Test Etmek. Federasyonu bütünleştirmek için gerekli tüm faaliyetler yerine getirilir ve karşılıklı işleme gereksinimlerinin karşılanıp karşılanmadığı test edilir.

6. Adım: Federasyonu Çalıştırmak ve Sonuçları Hazırlamak**.** Federasyon çalıştırılır, federasyonun çalışmasından elde edilen sonuçlar ön işleme tabi tutulur.

7. Adım: Verileri Analiz Etmek ve Sonuçları Değerlendirmek**.** Federasyonun çalışmasından elde edilen veriler analiz edilerek incelenir, sonuçlar kullanıcılara ve sponsora raporlanır.

Yedi aşamalı bu süreç, geliştirilecek uygulamaya bağlı olarak çeşitli şekillerde hayata geçirilebilir. Dolayısıyla bir HLA federasyonu geliştirmek için gerekli süre ve çaba da önemli ölçüde değişiklik gösterecektir. Örneğin çok büyük ve karmaşık bir uygulamanın kapsamını belirlemek için federasyon geliştirme takımının haftalarca çalışması gerekebilmektedir. Ancak daha az kapsamlı bir uygulama için aynı iş, bir günde veya daha az bir sürede gerçekleştirilebilir.



Şekil 3.7. Federasyon geliştirme ve çalıştırma süreci (FEDEP) genel görünümü

Federasyon uygulaması için gerekli insan gücü de federasyonun kapsamına bağlıdır. Bazı karmaşık uygulamalarda bir görevi üstlenen birkaç kişilik ekipler kurulması gerekirken, basit uygulamalarda bir kişi birkaç görevi üstlenebilir. Bir HLA federasyonunda kişilerin üstlenebilecekleri görevler şunlar olabilir: federasyon kullanıcısı/sponsor, federasyon yöneticisi, teknolojistler, güvenlik analizcisi, doğrulama, geçerleme ve akreditasyon analizcisi, işlevsel alan uzmanları, federasyon tasarımcıları, icra planlayıcıları, federasyon bütünleştiricileri, federasyon operatörleri ve veri analizcisi. Burada sayılan görevler ve roller federasyonun kapsamına göre değişiklik gösterecektir.

Federasyonun geliştirilme süresini, çabasını ve yedi aşamalı bu sürecin değişkenliğini etkileyen diğer bir faktör ise geliştirilmekte olan federasyonda kullanılan federelerin yeniden kullanılabilirliğiyle alakalıdır. Federasyon içerisinde kullanılan federeler şayet daha önceden başka federasyonlarda kullanılan federelerden küçük değişiklikler yapılarak kullanılıyorsa bu durum federasyonun geliştirilme süresini, çabasını ve maliyetini olumlu etkileyecektir (DMSO, 1999; IEEE, 2003).

FEDEP sürecinde verilen yedi aşama, federasyon inşası için genel bir çerçeve sunmaktadır. Kullanıcı kendi uygulama alanının gereksinimlerine ve kısıtlarına göre bu FEDEP sürecini düzenleyip değiştirebilir.

FEDEP süreci tablo olarak Tablo 3.1'de gösterilmiştir. Bu tabloda her sürecin temel faaliyetlerini de görmek mümkündür.

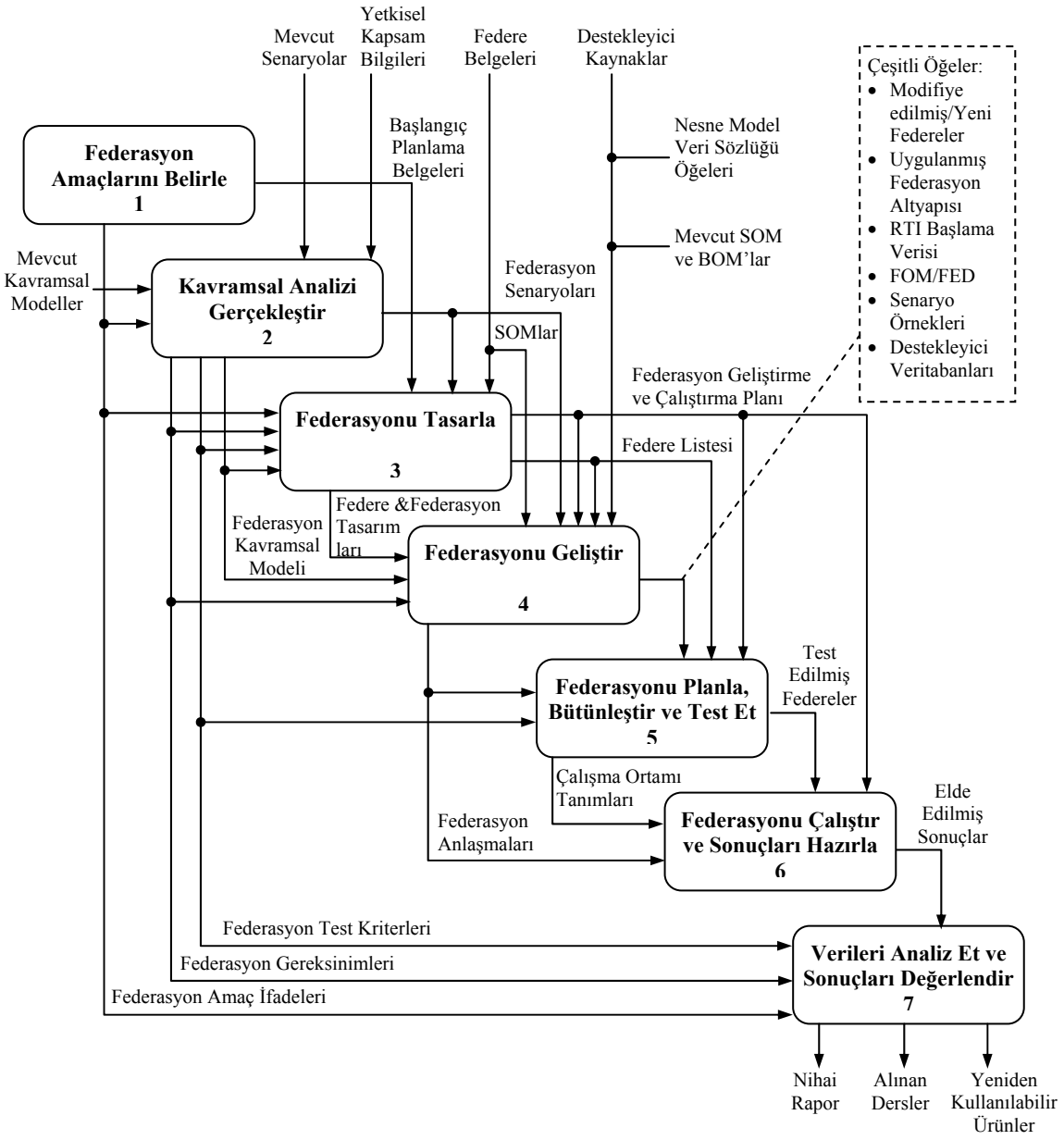
Tablo 3.1. Federasyon geliştirme ve çalıştırma süreci (FEDEP) tabo görünümü

(1) **Federasyon Amaçlarını Belirle**	(2) **Kavramsal Analizi Gerçekleştir**	(3) **Federasyonu Tasarla**	(4) **Federasyonu Geliştir**	(5) **Planla, Bütünleştir ve Test Et**	(6) **Federasyonu Çalıştır ve Sonuçları Hazırla**	(7) **Verileri Analiz Et ve Sonuçları Değerlendir**
Kullanıcı / sponsor ihtiyaçlarını belirle Amaçları geliştir	Senaryo geliştir Federasyon kavramsal modelini geliştir Federasyon gereksinim-lerini geliştir	Federeleri seç Federasyon tasarımını hazırla Plan hazırla	FOM geliştir Federasyon anlaşmalarını oluştur Federe tasarımlarını uygula Federasyon altyapısını uygula	Çalıştırmayı planla Federasyonu bütünleştir Federasyonu test et	Federasyonu çalıştır Federasyon çıktılarını hazırla	Verileri analiz et Sonuçları değerlendir ve geri bildirimde bulun

FEDEP sürecinin daha ayrıntılı bir görüntüsü Şekiş 3.8'de verilmektedir. Bu görüntü, Şekil 3.7'de verilen yedi aşama arasındaki bilgi akışını göstermektedir. FEDEP modelinde belirtilen aşamalar burada detaylandırılmıştır. Bu şekilde verilen faaliyetler oldukça sıralı ve birbirleriyle katı ilişkili gibi görünse de amaç, federasyonun geliştirilmesi ve çalıştırılması için keskin bir yaklaşım önermek değildir (IEEE, 2003).

Tecrübeler, FEDEP faaliyetlerinin eş zamanlı olarak yapıldığını ortaya koymuştur. FEDEP, HLA federasyonları geliştirme ve çalıştırma süreçerine otomasyon uygulama fırsatları için bir çerçeve olarak da kullanılmıştır (DMSO, 1999).

Görüleceği gibi FEDEP süreci bir sistem mühendisliği yaklaşımıdır. Benzer yaklaşım dağıtık imalat benzetim federasyonları için de kullanılmaya elverişlidir. FEDEP sürecinin yedi adımı ayrıntılı bir şekilde IEEE Std 1516.3 belgesinde ele alınarak girdi ve çıktılarıyla açıklanmıştır. Daha ayrıntılı bilgi için FEDEP standart dokümanına bakılabilir (IEEE, 2003).



Şekil 3.8. Federasyon geliştirme ve çalıştırma süreci (FEDEP) ayrıntılı görünümü

3.8. Yüksek Seviyeli Mimarinin (HLA) İmalatta Kullanımı

HLA'in dağıtık imalat alanında kullanılması ile ilgili çalışmalara rastlamak mümkündür. Ancak HLA temelli dağıtık imalat uygulamaları yeterince yaygınlaşamamıştır. Bunun sebepleri arasında, HLA konusunun yeni bir konu olması, başlangıçta askeri amaçlı benzetimlerde kullanılması, imalat gibi sivil alanlara uygulanmasının daha geç olması olabilir. Bunlarla birlikte mevcut imalat benzetim paketlerinin henüz HLA'yı desteklememesi de sayılmalıdır. McLean vd. (2005), HLA temelli ya da HLA'yı destekleyen ticari bir dağıtık imalat benzetim yazılımının henüz geliştirilmediğini belirtmektedirler. Mönch vd. (2003), atölye kontrol sistemlerine ve optimize edici yazılımlara da henüz karşılıklı işlerlik özelliği kazandırılamadığını belirtmişlerdir.

Askeri alanda gelişen HLA ile yapılmış ilk sivil çalışmalardan biri Bluemel vd.'nin (1998) yaptıkları çalışmadır. "Lojistik benzetimleri için HLA ve SLX kullanımı" ismini taşıyan bu çalışma, imalat ve lojistik konularındaki eğilimleri irdeleyerek bunların benzetime etkilerini tartışmaktadır. Bütünleşik planlama ortamı dedikleri bir sistem ile benzetim modellerini SLX paketi ile geliştirmeyi ve aralarındaki etkileşimi HLA ile sağlamayı amaçlamaktadırlar.

Başka bir çalışmada fabrika benzetimine genel bir bakış sunduktan sonra SLX benzetim yazılımı ile HLA'yı bütünleştirmişler ve prototip bir federasyon geliştirmekte uygulamışlardır (Schumann vd., 1998).

Bir diğer çalışma ise Schulze vd.'nin (1999a ve 1999b) taşıma uygulamalarında yaptıkları çalışmadır. Askeri alanda yapılan benzetimler ile sivil alanda yapılanların teknik olarak benzediğini, ancak terminoloji olarak bazı değişikliklerin olduğunu ifade etmişler ve HLA'nın dağıtık benzetimde gelinen son nokta olduğunu belirtmişlerdir. HLA'nın askeri alan dışında sivil alanlarda da uygulanabileceğini vurgulayarak, taşıma alanında bir örnek sunmuşlardır.

McLean vd. (2005), HLA altyapısına dayanan dağıtık bir imalat benzetimi mimarisi önermişlerdir. Yazarlar, HLA temelli ya da HLA'yı destekleyen ticari bir dağıtık

imalat benzetim yazılımının henüz geliştirilmediğini belirtmektedirler. Bu sebeple HLA ile benzetimler arasında, benzetimlerin HLA ile bütünleşmesini ve iletişimi sağlayacak bir arayüze ihtiyaç vardır. Bu arayüz benzetimlerden iletilen bilgiyi HLA'nın anlayacağı formata, HLA'dan iletilen bilgiyi de benzetimin anlayacağı formata dönüştürmekle sorumludur. Bunu "uyarlayıcı arayüz" olarak adlandırmak mümkündür. Uyarlayıcı arayüzü içerisinde kullanım kolaylığı ve bilgilerin nesne yönelimli olarak ifade edilebilmesi nedeniyle XML (Extensible Markup Language) dili kullanılmasını önermektedirler.

Hâlihazırda kullanılmakta olan çeşitli ticari benzetim yazılımları mevcuttur. Bu benzetim paketleri kullanılarak çeşitli benzetimler meydana getirilmektedir. Ancak bunların entegrasyon gereksinimleri baş göstermektedir ve bunlardan hiçbiri henüz HLA uyumlu değildir. Taylor vd. (2006) bu ihtiyacı belirttikten sonra ticari kesikli-olay benzetim yazılımlarının entegrasyonu ve karşılıklı işlemesi için HLA'nın kullanılmasını ve bununla birlikte HLA'ya birkaç standart daha ilave edilmesi gerektiğini belirtmişlerdir. McLoughlin ve Heavey (2001) ise, imalat sistemlerinde bileşen temelli benzetimleri ele aldıkları çalışmada çeşitli benzetim paketlerinin HLA uyumluluğunu ve kullanılabilirliğini araştırmışlardır.

GeorgiaTech araştırma enstitüsünün bilişim, teknoloji ve telekomünikasyon laboratuarı (ITTL-Information Technology and Telecommunications Laboratory) eski benzetimlerin HLA'ya dönüştürülmesi konusunda araştırma projesi yürütmektedir. HLA'yı kullanarak genel bir dağıtık benzetim yaklaşımı tanımlayacak ve uygulayacak bir arayüz geliştirmek bu projenin hedefidir. Bu arayüzü DSIF (The Distributed Simulation Interface Framework) olarak adlandırmaktadırlar. DSIF, HLA arayüz yazılımı üretmek için HLA nesne modeli şablonu formatında bir nesne modeli kullanmaktadır (Hilderbrand, 2006).

Hibino vd. (2002), çeşitli simülatörleri senkronize ederek çok büyük imalat sistemlerinin kolayca incelenebilmesi için dağıtık bir benzetim sistemi geliştirmişlerdir. Bu dağıtık benzetim sistemi için HLA'yı önermişlerdir. Tasarlanan bu dağıtık sistem benzetimi, motor üreten hipotetik bir imalat sistemi örnek olay olarak kullanılarak desteklenmiştir. Bu çalışmada, ticari imalat sistem

simülatörlerinden QUEST, SIMPLE++ ve GAROPS birbirleriyle ilişkilendirilmiştir. HLA'nın kullanımına imkan veren mevcut bir ticari benzetim yazılımı olmadığı için imalat sistemi ile HLA arasında iletişimi sağlayacak bir İmalat Adaptörü geliştirmişler ve ikisi arasındaki iletişimde XML formatlı mesajlar kullanmışlardır. Adaptör, simülatörlerden gelen XML mesajlarını HLA arayüz tanımlama mesajlarına ve HLA/RTI'dan gelen mesajları da XML biçimine dönüştürmektedir. Böylece simülatörler HLA/RTI ile iletişim kurabilmekte ve HLA servislerinden faydalanabilmektedir.

Hibino ve Fukuda (2006), dağıtık benzetimlerin tanımlanması sürecini desteklemek ve yönetmeyi, tanımlama parametrelerini kaydedip yeniden kullanabilmeyi amaçlamaktadırlar. Diğer amaçları ise, tek bir imalat sistemi tasarımcısı ile dağıtık benzetim sistemlerinin uzaktan çalıştırılmasıdır. Bununla ilgili olarak kullanıcı destek sisteminin gereksinimlerini belirlemek istemektedirler. Hipotetik olarak motor üretimi uygulaması ile amaçlarını desteklemektedirler.

Tedarik zincirleri, dağıtık imalat için tipik bir örnektir. Dolayısıyla tedarik zincirleri HLA'nın en önemli uygulama alanı olmaya adaydır. Terzi ve Cavaliery (2004), tedarik zinciri bağlamında benzetimi ele aldıkları araştırmalarında HLA'ya da değinmişlerdir. Benzetimin tedarik zincirlerinde kullanılmasını araştıran bu çalışmada HLA'nın da tedarik zincirlerinde kullanılabileceği belirtilmiştir. Lenderman vd. (2003), yarı iletken endüstrisi tedarik zincirinin dağıtık benzetim yaklaşımı ile modellenmesini önererek, teknik olarak HLA'nın buna uygunluğunu ifade etmişlerdir. Bandinelli vd. (2006), tedarik zincirlerinde benzetimin kullanılmasını araştırmışlardır. Dağıtık tedarik zinciri benzetimlerinin, tedarik zinciri planlama ve optimizasyonunda etkin bir araç olabileceğini ifade etmişlerdir. Dağıtık tedarik zinciri benzetimlerini mümkün kılan teknoloji olarak da HLA'yı açıklamışlardır.

Linn vd. (2002), tedarik zincirinde ulaştırma davranışını incelemek üzere iki makinede çalışan benzetim prototipi geliştirmişlerdir. Bu çalışmalarında, Arena benzetim yazılımı ile HLA çalışma anı altyapısını entegre etmişlerdir. Gan vd. (2000) ise HLA temelli tedarik zinciri performansını incelemişlerdir. Farklı benzetim

programları ve farklı diller ile geliştirilmiş çeşitli benzetimlerin olması durumda HLA'nın kullanılması gerektiğini belirtmişlerdir. Tedarik zincirlerinde karar vermeye yardımcı olacak HLA temelli dağıtık benzetim teknolojilerine dayalı örnekler, Chong vd. (2004) ve Lendermann, vd. (2004)'de görülebilmektedir.

Mertins vd. (2005), de benzer şekilde tedarik zincirlerinin dağıtık modellenmesi ve benzetimini ele almışlardır. Tedarik zincirlerinin davranışlarını analiz etmek için kesikli olay benzetim modellerini önermektedirler. Bu benzetim modellerinin tek bir benzetim gibi çalışmasını temin etmek için ise HLA ile birlikte XML kullanmayı önermektedirler.

Taylor vd. (2002), çalışmalarında dağıtık tedarik zinciri benzetimi için yeni bir altyapı olarak genel bir dağıtık benzetim çalışma-anı altyapısı tanıtmışlardır. Dağıtık bir tedarik zinciri modelini federasyon olarak tasarlayıp otomotiv endüstrisinden örnek olay ile ele almışlardır. Bu çalışmanın temel amacı, tedarik zincirinde daha önceden geliştirilmiş benzetim modellerinin yeniden kullanımını, teknolojik müdahaleleri en aza indirgeyerek, dağıtık benzetim ile sağlamaktır.

Jian vd. (2004), katılımcı ürün tasarımı için kullanılacak HLA temelli yazılım mimarisi önermektedir. Burada Matlab ve Adams gibi ticari benzetimlerin HLA uyarlayıcılar ile sisteme dâhil olabileceğini ifade etmişlerdir.

Taylor vd. (2005) ise Ford motor fabrikasında, motor üretiminin dağıtık ortamda benzetimini ele almışlardır. Benzetim zamanını azaltmak için HLA temelli benzetim tasarlanmış ve Ford için dağıtık benzetim kullanımını önermişlerdir.

İmalatta benzetimin kullanımının artmasıyla birlikte kullanılan benzetim modellerinin çeşitliliği de artmaktadır. Bunun neticesinde bu modellerin bütünleştirilip karşılıklı çalışma ihtiyacı ortaya çıkmaktadır. Price vd. (2004), farklı imalat benzetimlerin birleştirilip birçok benzetimin tek bir benzetim gibi çalıştırılmasına yönelik çok az çalışma olduğunu ifade ederek HLA üzerinden çözüm önermişlerdir.

Rabelo vd., (2003 ve 2005), imalat kurumlarının karar türlerini; kurum kararları, tasarım kararları, mühendislik kararları ve üretim kararları olmak üzere dörde ayırdıktan sonra üretim kararları ile ilgilendiklerini belirtmişlerdir. Üretim kararları için kesikli olay benzetim modelleri kullanılacaktır. Bu kesikli olay benzetim modelleri bütünleştirilmek zorundadır. Bunun için HLA kullanmak önerilmiştir. Böylece lokal değişikliklerin tüm kuruma etkisi analiz edilebilecektir.

Lendermann vd. (2005) bütünleşik imalat ve servis sistemleri konusunda yaptıkları çalışma ile ilgili bilgiler vermektedirler. Bu çalışmalarının amacı, imalat ve servis ağlarında, kritik işletme süreçlerinin tasarım, analiz ve uygulamasının tek bir benzetim/uygulama çerçevesi kullanılması ile nasıl gerçekleştirilebileceğidir. Bu çerçevenin mimarisi, ticari benzetim paketlerinin ve web temelli işletme uygulamalarının birleştirilmesi esasına dayanmaktadır. Bu şekilde ilgili uygulamaların yeniden kullanılabilirliği artırılmak istenmektedir.

Kurum benzetimi kavramının benzetim uygulamacıları arasında kabul görmeye başladığını belirten Datar (2000), kurum benzetimini ele aldığı çalışmasında bunu mümkün kılan teknoloji olarak HLA'yı belirtmektedir. McLean ve Riddick (2000b) ise benzetimi tedarik zinciri çapından atölye çapına kadar çeşitlilik göstereceğini ifade ederek bunların entegrasyonu için belirli bir standardın olmadığını belirtmektedirler. Dağıtık imalat benzetimlerinin birbirleriyle, diğer uygulama araçlarıyla ve veritabanları ile entegrasyonu için HLA tabanlı bir mimari önermektedirler.

Mclean ve Leong (2001a ve 2001b) ise benzer şekilde imalat benzetimlerinin ürünün nasıl üretileceğine etki edeceğini ifade ederek, etkileşim problemleri, maliyet gibi sebeplerle benzetimin herkes tarafından rahatça kullanılamadığını vurgulamaktadırlar. Bu sorunları aşmak için standart arayüzlere ihtiyacı vardır. İlgili çalışmalarında, standartlaştırılması gereken benzetim arayüzlerini açıklamaktadırlar.

Saad vd. (2003), pazarın küreselleşmesi ve dünya çapında rekabetin zorlamasından dolayı kurumların işbirliklerine gitmesi gerektiğini ve bu şekilde dağıtık imalat kurumları oluşturmaları gerektiğini söylemektedirler. Birleşmeden kaynaklanan

etkiyi görmenin en iyi yolunun dağıtık benzetim olduğunu ifade etmektedirler. Burada kullanılması gereken teknoloji olarak ise HLA'yi önermektedirler.

Venkateswaran vd. (2004), hiyerarşik bir üretim planlama çalışması yapmışlardır. Buna göre üst seviyede sistem parametrelerinin bulunduğu bileşen ve alt seviyede ise atölye çapında kesikli olay benzetim modeli vardır. Karar verici sistem parametrelerini seçmekte, bu ise alt seviyede kesikli olay benzetimini etkilemektedir. Sistem parametreleri ile alt seviye kesikli olay benzetimleri arasında HLA kullanımı öngörülmüştür. Aynı çalışma daha geniş bir şekilde Venkateswaran ve Son (2005) tarafından ele alınmıştır.

Strassburger vd. (2003), dijital fabrikanın oluşturulabilmesi için dağıtık imalat benzetimi oluşturulması gerektiğini belirtmişlerdir. Dağıtık imalat benzetimi için kesinlikle HLA önermekte, başka araçların kullanışsız olacağını ifade etmektedirler. Dijital fabrika için birçok farklı (heterojen) benzetim gerekeceğini söylemektedirler. Ticari benzetim paketlerinin, etkileşimi sağlayabilmeleri için HLA'yı desteklemeleri gerektiğini belirtmektedirler.

Hibino vd., (2006), ise gerçek imalat benzetim ortamı ile sanal imalat ortamının birleştirilmesini önermektedirler. Benzetimleri birleştirici araç olarak HLA düşünülmüştür.

3.9. Yükesek Seviyeli Mimarinin (HLA) Diğer Alanlarda Kullanımı

HLA'yı birçok benzetim sistemini birleştirmekte kullanmak mümkündür. Benzetim sistemleri tamamen bilgisayar temelli olabileceği gibi insanları da içerebilir. Gerçek bir insanın benzetimi oluşturulmuş bir aracı, örneğin uçuş simülatörünü kullanması durumundaki benzetimlere sanal benzetim (virtual simulation) denir. Diğer bir tür benzetim ise yapısal benzetimdir (constructive simulation). Bu benzetim türünde gerçek insan yerine benzetim modeli oluşturulmuş insanlar benzetimi oluşturulmuş araçları kullanırlar. Başka bir benzetim çeşidi de canlı benzetimdir. Bu benzetim çeşidinde ise gerçek insanlar gerçek araçları kullanırlar ve diğer benzetim sistemleri ile etkileşim halindedirler. Örneğin askeri bir tatbikatta askerlerin diğer benzetim

sistemlerine radyo frekanslarıyla çalışan teçhizatla bağlanmaları gibi (PITCH, 2006a).

Amerikan savunma bakanlığı (Department of Defence: DoD) ile NASA'nın misyonu temelde farklılık arz etmekle birlikte teknolojik ihtiyaçları genellikle örtüşmektedir. NASA'nın benzetim ihtiyaçları DoD'un benzetim ihtiyaçlarına benzemektedir. NASA da karmaşık işletim sistemleri kullanmakta, maliyeti ve riski azaltmak için simülatörler kullanmaktadır. NASA benzetim ihtiyaçlarını karşılamak için HLA temelli tasarımlar kullanma kararı almıştır (Reid, 2000; Xiaoxia ve Qiuhai, 2003).

HLA temelli bir benzetimde aynı tür alt benzetim sisteminden birden fazla sayıda olabilir. Örneğin birçok Boing 747 veya F-16 benzetimi aynı sistem içerisinde bulunabilir. HLA temelli benzetim çalışıyorken sisteme yeni bileşenler katılabilir veya sistemden bileşenler çıkabilir. Bu bileşenler sadece benzetim sistemleri olmayabilir. İnsanlar için arayüz yazılımları, veri toplama, veri analizi ve veri gösterim işlevlerini gerçekleştiren yazılımlar da HLA temelli benzetim sistemine (federasyon) bir bileşen olarak katılabilir veya sistemden ayrılabilir. (MEOSS, 2006).

HLA başlangıçta askeri amaçlı benzetimler için geliştirildiğinden askeri uygulamalarına çokça rastlamak mümkündür. IEEE standardı olmasından sonra sivil alanlarda ve imalatta da kullanılır hale gelmiştir.

Sivil alanda HLA'nın kullanımına ilk örneklerden biri Klein vd.'nin (1998) yaptıkları trafik benzetimidir. Klasik benzetimlere göre esneklik ve yeniden kullanılabilirlik sağladığını ifade ettikleri çalışmalarında HLA temelli dağıtık bir trafik benzetimi geliştirmişlerdir.

HLA çeşitli amaçlar için geliştirilmiş benzetimleri desteklemektedir. Örneğin, eğitim/tatbikat ve öğretim alanında kullanılabilmektedir. Ürünlerin benzetim ile incelendiği, analiz ve benzetim temelli araçlarda ve mühendislik uygulamalarında ve hatta eğlence alanlarında da HLA kullanılabilmektedir. Bu çok farklı uygulama alanları, HLA'nın geliştirilmesinde ve iyileştirilmesinde çok farklı ihtiyaçların dikkate alındığını göstermektedir (Dahmann vd., 1999a).

Borshchev vd. (2002), hibrit sistemlerin dağıtık benzetimi için HLA ile birlikte bir java uygulaması olan AnyLogic aracını kullanmışlardır. HLA burada, dağıtık hibrit benzetim bileşenleri arasındaki iletişim ve senkronizasyon aracı olarak kullanılmıştır.

Bilgisayar oyunları benzetim ile birlikte kullanılarak eğitim amaçlı bilgisayar oyunları geliştirilmektedir. Bireysel ve takım savaş taktiklerinin öğretilmesi için "Doom" isimli bilgisayar oyunu modifiye edilerek kullanılmıştır. Denizcilerin operasyonel eğitimi için kullanılmak üzere MaK Technologies tarafından HLA-uyumlu bilgisayar oyunu geliştirilmiştir (Coleman, 2001).

Klein (2000), çeşitli bileşenleri bir araya getirmek suretiyle birçok amaca hizmet eden benzetim temelli sistemler için yeni bir bilişim teknolojisi yaklaşımı önermektedir. HLA yazılım mimarisi temelli bu yaklaşımda, sistem tasarımı ve geliştirilmesi, işletilmesi, eğitim ile birlikte risk ve senaryo yönetimi konularında esneklik, yeniden kullanılabilirlik ve karşılıklı işlerlik amaçlanmaktadır.

HLA'yı zeki etmenler ile birlikte kullanmak da mümkündür. Das ve Reyes (2002), çoklu zeki etmenlerin tasarlanmasında HLA ve genetik algoritmayı birlikte kullanmayı önermektedirler. Maamar (2003), zeki etmenleri ve HLA'yi kullanarak eğitim senaryoları için benzetim ortamları tasarlayacak bir proje üzerinde çalışmaktadır. HLA burada, eğitim oturumlarına katılan katılımcılar arasında iletişim aracı olarak düşünülmüştür. Lees vd. (2007), HLA_AGENT isimli bir araç tanıtmaktadırlar. Bu araç, SIM_AGENT isimli etmen aracı ile HLA dağıtık benzetim mimarisini birleştirmektedir.

Göktürk ve Polat (2003), dağıtık benzetim standardı olarak HLA'nın çoklu etmen benzetimlerine iyi bir aday olduğunu ifade etmektedirler. Çalışmalarında HLA ve etmen dili olan KQML kullanılarak etmen iletişimi yaklaşımı sunmaktadırlar.

Shen vd. (2002), sanal ortamda elektronik ticaret sistemine farklı bir yaklaşım sunmaktadırlar. Bu amaçla HLA temelli sanal bir ortam oluşturmayı önermektedirler.

Çalışmaları, kullanıcı arayüz tasarımı, HLA üzerinden katılımcı elektronik ticaret ve çoklu etmen mimarisine odaklanmaktadır.

Zaman yönetimi dağıtık benzetimde önemli konulardan bir tanesidir. Huang vd. (2005), benzetimlerin belirli bir zaman senkronizasyonu mekanizması kullanılarak tasarlandığı durumlarda bazı HLA servislerinin düşük seviyeli ve kullanımı zor olduğunu belirtmişlerdir. Bu zorluğu aşmak için akıllı zaman yönetimi adını verdikleri bir etmen geliştirmişlerdir. Bu etmen, kullanıcı ile HLA arasında bir arayüz oluşturarak zaman senkronizasyonu meselesini çözecektir.

Lu ve Hsu (2007), kablolu veya kablosuz ağlar üzerinden işbirliği yapabilen hibrit benzetim ortamı önermektedirler. Bu çalışmada IEEE 1516 HLA standardı ve IBM Aglets mobil etmen sistemi birlikte önerilmiştir. Bu şekilde, HLA temelli benzetim uygulamaları sadece masaüstü bilgisayarlar ve kablolu ağ bağlantıları ile sınırlandırılmamış olacaktır. HLA benzetimlerine bağlanmak için her türlü mobil araçlar kullanmak mümkün olacaktır. Deneysel çalışma sonuçları, mobil etmen temelli veri dağıtımının, büyük çaplı HLA temelli benzetimlerin uyarlanabilirliğini ve uygulanabilirliğini artıracağını göstermektedir.

Son yıllarda bilgisayar oyunlarında ve sanal ortamlarda zeki etmenlerin kullanımı da yaygınlaşmaktadır. Lees vd. (2006), bilgisayar oyunlarında kullanılmak üzere HLA uyumlu zeki etmenler geliştirmişlerdir.

Benzetimin farklı uygulamalardaki yararları, benzetimin bir problem çözme aracı olarak kullanılması, bilgisayar teknolojilerinin gelişmesi, benzetimin dağıtık ortamlarda yapılma ihtiyacı gibi sebeplerle artık web temelli ve web üzerinden çalışan benzetim uygulamaları konusunda araştırmalar artmaktadır. Bunun hareket noktası, modelin kavramsal tasarımının, inşasının, çalıştırılmasının ve analizinin artık dağıtık olması, iş birliği ve etkileşimi gerektirmesidir. Tüm bunlar için standartlaştırma gerekmektedir. Bu konuda HLA'den faydalanmak mümkündür (Page ve Opper, 2000).

Dağıtık benzetimlerde veri dağıtım yönetim aracı olarak HLA'nın kullanılması ve veri dağıtımının etkinliğinin artırılmasına yönelik çalışmalara rastlamak da mümkündür. Bu konuda Morse ve Zyda (2002), Petty (2002), Petty ve Morse (2004), Boukerche ve Roy (2002), Boukerche vd. (2006) çalışmalarından yararlanılabilinir.

Boukerche ve Dzermajko (2004), ise büyük çaplı benzetimlerde bilgisayar işlemcisinin yükünü azaltmak için veri dağıtım yönteminin iyi tespit edilmesi gerektiğini belirtmektedirler. HLA ile birlikte çeşitli veri dağıtım tekniklerinin performansını incelemişlerdir.

Morse vd. (2006), çoklu benzetim sistemlerinin HLA ile birbirine bağlanması gerektiğinin önemini ele almışlardır. Yazarlar, veri modelinin iletilmesinde HLA'nın nasıl kullanıldığını vurgulamışlar ve HLA'nın çerçeve ve kurallarını, federe arayüz spesifikasyonunu ve nesne model şablonunu açıklamışlardır. Bununla birlikte benzetimlerin web üzerinden çalışmasını ve etkileşmesini de kısaca ele almışlardır.

Cengiz ve Oğuztüzün (2002) ise, programlama dilleri arasında karşılıklı etkileşim gerçekleşebilmesi için Microsoft bileşen nesne modeli (COM) temelli bir arayüz önermişlerdir. Bu arayüz, kullanıcı tarafından geliştirilen HLA kodları ile RTI arasında aracılık yapacaktır. Böylece COM nesnelerini kullanabilen Borland C++ ve Microsoft Visual Basic gibi programlama dilleri tarafından geliştirilen federelerin etkileşimi amaçlanmaktadır.

Chen vd. (2003), alternatif senaryoların eş zamanlı karşılaştırması için benzetim klonlaması geliştirmişlerdir. Çalışmalarında HLA temelli dağıtık benzetimlerin klonlanması ile ilgili meseleleri ele almışlardır. Alternatif çözümleri çeşitli açılardan değerlendirmişlerdir.

Diğer bir çalışmada Cai vd. (2005) dağıtık ortamda çalışan HLA temelli benzetimlerin bilgisayar yüklerini ele almışlar ve bazı bilgisayarların daha fazla, bazılarının ise daha az işlem yüküne sahip olabileceği belirtmişlerdir. Bu probleme çözüm olarak ise, benzetim bileşenlerinin, iş yükü çok olan bilgisayarlardan iş yükü

daha az olanlara iletilmesini önermişlerdir. Bu amaçla, HLA federelerinin iletilmesi konusunda bir protokol ve çerçeve önermektedirler.

Farklı bir çalışma ise HLA temelli petri netler konusunda yapılan çalışmadır. Bayarou vd. (2002), dağıtık stokastik petri netlerin davranışlarını incelemek üzere HLA temelli stokastik petri net benzetimini açıklamaktadırlar.

Diğer farklı bir çalışma ise Li vd. (2006) tarafından yapılan, HLA/RTI üzerinden mültimedya iletişimi konusundaki çalışmadır. Bu çalışmada, dağıtık bir benzetim ortamında HLA/RTI üzerinden çoklu ortam (multimedya) iletişimi için bir çözüm önermişlerdir. Bu çözüm, yüksek hacimde ses, video, resim ve metin verilerinin iletimini desteklemektedir. Bu çalışmada çoklu ortam FOM/SOM tasarlanmış ve çoklu ortam iletişimi için prototip bir sistem oluşturulmuştur. Yapılan deneyler, gerçek zamanlı iletişim için ihtiyaçların karşılandığını ve yüksek derecede iletişim etkinliğinin sağlandığını göstermiştir.

Bir başka çalışmada Liyu vd. (2006), orman yangınları ile mücadele amaçlı HLA temelli bir benzetim önermektedirler.

Trcka (Radosevic) vd. (2006) ise dağıtık yapı performansı benzetimi çalışmaktadırlar. Programları tek tek kullanıp sonuç almak yerine birbirlerine bağlayarak dağıtık benzetim ortamı oluşturmanın daha yararlı olduğunu ifade etmektedirler. Dağıtık benzetim için ise bir alternatif olarak HLA'yı önermektedirler.

HLA tıbbi benzetim uygulamalarında da kullanılmaktadır. Petty ve Windyga (1999), medikal benzetim sistemlerinde HLA kullanımına dair çalışma yapmışlardır.

HLA'nın ekonomik ve finansal benzetimlerde kullanımına dair çalışmaya da rastlamak mümkündür. Calpin vd. (2001), HLA'nın ekonomik benzetimlerde kullanılabileceğini ortaya koymuştur.

3.10. HLA Temelli Dağıtık İmalat Benzetimi Mimarisi

Bu kısımda dağıtık imalat sistemleri için bir mimari tanıtılacak ve gerekli arayüzler açıklanacaktır. Burada verilen mimariyi McLean vd. (2005) önermişlerdir. Gerekli görüldüğü yerlerde şekilsel değişiklikler yapılmıştır. Sunulan mimarinin amacı, imalat benzetimlerini ve diğer imalat yazılımlarını dağıtık ortamda birbirleriyle iletişimini ve etkileşimini sağlamaktır.

Bu mimari HLA altyapısına dayanmaktadır. HLA temelli ya da HLA'yı destekleyen ticari bir dağıtık imalat benzetim yazılımı henüz geliştirilmemiştir. Bu sebeple HLA ile benzetimler arasında, benzetimlerin HLA ile bütünleşmesini ve iletişimi sağlayacak bir arayüze ihtiyaç vardır. Bu arayüz, benzetimlerden iletilen bilgiyi HLA'nın anlayacağı formata, HLA'dan iletilen bilgiyi de benzetimin anlayacağı formata dönüştürmekle sorumludur. Bunu "uyarlayıcı arayüz" olarak adlandırmak mümkündür. Uyarlayıcı arayüzü içerisinde kullanım kolaylığı ve bilgilerin nesne yönelimli olarak ifade edilebilmesi nedeniyle XML (Extensible Markup Language) dili kullanılması mümkündür.

Standart mimari ve arayüzler geliştirmek, benzetim modellerinin inşasıyla ilgili ve benzetim ile diğer yazılım uygulamaları arasında veri iletişimiyle ilgili maliyetleri azaltacaktır. Bunun sonucunda benzetim teknolojilerinin kullanımı yaygınlaşacaktır. Hâlihazırda küçük atölyeler, model geliştirme ve veri iletimi ile ilgili bir takım zorluklar sebebiyle benzetim teknolojilerini tam anlamıyla kullanamamaktadırlar. Küçük atölyelerin kendi ihtiyaçlarını karşılamak için benzetim geliştirebilecek veya çeşitli uygulama yazılımlarından verileri aktarabilecek nitelikte elemanları da genellikle mevcut değildir. Benzer sebepler benzetim teknolojilerinin kullanımını azaltmaktadır.

İmalat benzetimleri ve diğer imalat yazılımları arasında verilerin kolay iletimi için enformasyon modelleri de geliştirilmelidir. Enformasyon modellemenin amacı, enformasyonun imalat ortamında iletilebilmesi için standart, bilgisayar tarafından yorumlanabilir bir enformasyon gösterimi geliştirmektir. Enformasyon modeli şu ihtiyaçları karşılayabilmelidir:

- Tüm imalat hayat döngüsünün veri gereksinimlerini desteklemelidir,
- Benzetimler ve diğer imalat yazılımları arasındaki veri iletimini temin etmelidir,
- Atölye simülatörlerinin inşasını desteklemelidir,
- Atölye imalat yazılımlarının test edilmesini ve incelenmesini desteklemelidir.

HLA'da bilgi gösterimi için standart tabloların kullanılması ortak bilgi gösterimi sağlamakta ve benzetim modellerinin birbirine bilgi iletimini kolaylaştırmaktadır.

3.10.1. Dağıtık imalat benzetimi mimarisi

Dağıtık imalat benzetimi, bağımsız olarak işleyen ve birbirleriyle etkileşim halinde olan çoklu yazılım proseslerinden oluşmuş imalat benzetimidir. Bununla birlikte bu benzetim yazılım prosesleri imalat tedarik zinciri büyüklüğündeki bir sistemi modelleyebileceği gibi tekil bir imalat makinesini de modelleyebilmektedir. Benzetimler farklı yazılım tedarikçilerinin sunduğu benzetim yazılımları ile geliştirilmiş olabilmektedir. Modüller coğrafi olarak dağıtılmış farklı bilgisayar sistemlerinde çalışabilmektedir. Benzetimin dağıtık olmasının sebebi, farklı benzetim yazılımlarının üstünlüklerinden faydalanmak olabileceği gibi benzetimin çalışma hızını artırmak da olabilmektedir.

Dağıtık imalat benzetimi ile benzetim olmayan imalat yazılımlarının da çalıştığı ve benzetim yazılımları ile etkileştiği ortamlar da kastedilmektedir. Mühendislik yazılımları benzetim sistemleri ile etkileşebilir ve simülatöre incelenmek üzere veri gönderebilmektedir. Örneğin bilgisayar destekli bir imalat yazılımı bir makine için kontrol programı üretebilir ve bu programın doğruluğu test edilmek üzere bir simülatöre gönderilebilmelidir.

Başka bir açıdan dağıtık imalat benzetimi, günümüzde tek bir benzetim sistemi içerisinde bulunan fonksiyonel modüllerin dağıtık olarak bir araya getirilmesidir. Böyle bir sistem model inşa araçları, benzetim motorları, görüntüleme sistemleri ve çıktı analiz yazılımlarından meydana gelmelidir.

3.10.2. Dağıtık imalat benzetim sistemleri geliştirme nedenleri

Dağıtık yaklaşım benzetimin işlevselliğini artırmaktadır. Dağıtık imalat benzetim sistemleri aşağıdaki amaçlar için kullanılabilmektedir (McLean ve Riddick, 2000a; McLean vd., 2005):

- Tedarik zincirini modellemede kullanılabilir. Böylece bir şirkettin bilgisi diğer zincir üyelerine de açık hale getirilebilir.
- Çok seviyeli imalat sistemlerinin farklı derecelerde benzetim modellerini kurmak için kullanılabilir. Böylece alt seviye benzetim sonuçları üst seviye benzetimlere bilgi sağlayabilir.
- Aynı fabrikadaki faklı benzetim gereksinimleri olan birçok sistemin benzetimini oluşturmak için farklı benzetim yazılımları kullanılabilir. Bazı benzetim yazılımları her gereksinimi sağlamayabilir. Bu durumda bu açığı kapatmak için sistemin bazı kısımları farklı benzetim yazılımlarıyla modellenir.
- Daha büyük modellere entegre edilebilen düşük maliyetli, çalışma zamanlı bir dizi benzetim modeli oluşturabilmek için kullanılabilir.
- Diğer bilgisayarların işlemci gücünden, belirli bir işletim sisteminin özelliklerinden ve çeşitli araçlardan (sanal gerçeklik arayüzü gibi) yararlanmak için kullanılabilir.
- Farklı yerlerdeki kullanıcıların benzetim modellerini çalıştırabilmeleri için eşzamanlı erişim olanağı sağlar. Böylece işbirlikçi çalışma ortamı oluşturulur.
- Farklı işlevler için benzetim faaliyetlerini (model kurma, görsellik, çalıştırma, analiz etme) destekleyecek farklı türlerdeki birçok yazılım lisansının kullanımına imkan sağlar.

3.10.3. Yazılım mimarisi

Yazılım sistemlerinin boyutu ve karmaşıklığı arttıkça, tüm sistemin yapısının tasarımı ve spesifikasyonu, seçilecek algoritma ve veri yapılarından daha önemli hale

gelmektedir (Shaw ve Garlan,1996). Benzetim teknolojileri sağlayan şirketlerin ve endüstriyel kullanıcıların karşılaştıkları entegrasyon problemini çözmek için bir dağıtık imalat benzetimi mimarisine ihtiyaç vardır. Genel benzetim arayüzleri veri aktarım ve model paylaşım maliyetlerini azaltacak ve böylece benzetim teknolojilerinin kullanımını artıracaktır. Dağıtık imalat benzetimleri için genel bir mimarinin tanımlanması, bu benzetim sistemlerinin bütünleştirilmesi için gerekli enformasyon modellerinin, arayüzlerin ve protokollerin belirlenmesinde ilk adımdır.

Dağıtık imalat benzetimi üç fonksiyonel kısma ayrılabilir. Bunlar:

- Dağıtık bilgi işleme sistemleri,
- Benzetim sistemleri,
- İmalat sistemleridir.

Dağıtık imalat benzetimi, bu bakış açılarından gelen ortak bilgiye ihtiyaç duymaktadır. Bu da dağıtık imalat benzetiminin aslında disiplinler arası bir konu olduğunu göstermektedir.

3.10.4. Dağıtık bilgi işleme sistemleri bakış açısı

Bu bakış açısı benzetimi, bir bilgisayar ağı üzerinde eş zamanlı çalışan ve birbiriyle haberleşen bir dizi bilgisayar ve yazılım prosesi olarak görür. Bu bakış açısında yazılım proseslerinin benzetim veya benzetimle ilgili olması önemli değildir ve benzetim veya imalat verilerinin içeriği ile ilgilenilmemektedir.

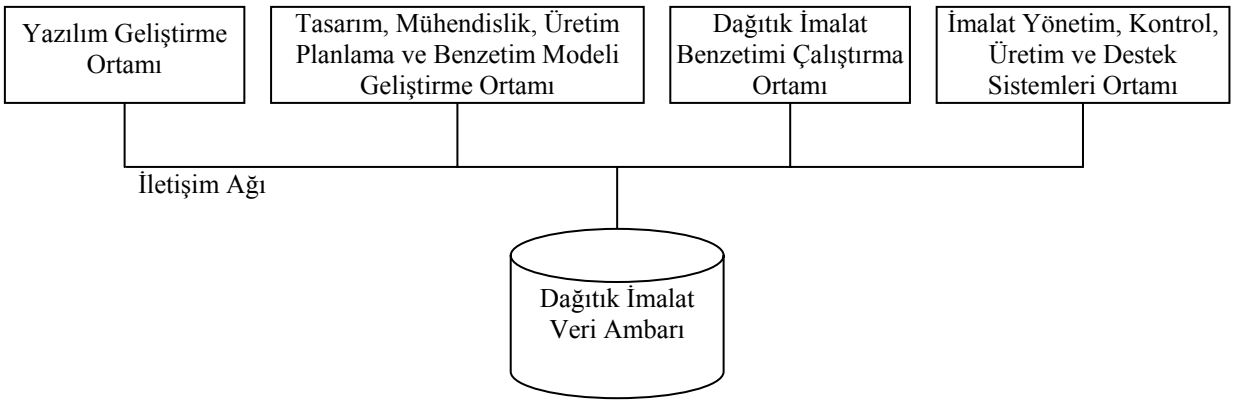
Bu bakış açısı, dağıtık sistemler olarak benzetimlerin geliştirileceği ve çalıştırılacağı ortamları oluşturacak altyapıyı sağlamaktadır. Bu bakış açısının öğeleri şunlardır: donanım platformları, işletim sistemleri, dağıtık bilgisayar prosesleri, entegrasyon altyapıları, proses başlatma ve senkronizasyonu, yazılım geliştirme ortamları, iletişim sistemleri, enformasyon modelleri ve veri sözlüğü, iş akışı yönetim sistemleri, veritabanı yönetim sistemleri ve veritabanları, ürün verisi yönetim sistemleri, bilgisayar dosya sistemleri ve dosyalar, sistem kurma ve bakım araçları, bilgisayar

güvenlik ve veri koruma servisleri, lisans doğrulama sistemleri, internet erişim mekanizmaları.

Dağıtık imalat benzetimi beş bilişim sistemi gurubuna ayrılabilir (McLean vd., 2005):

1. Yazılım geliştirme sistemleri,
2. Tasarım, mühendislik, üretim planlama ve benzetim modeli geliştirme sistemleri,
3. Dağıtık imalat benzetimi çalıştırma sistemleri,
4. İmalat yönetim, kontrol, üretim ve destek sistemleri,
5. Dağıtık imalat veri ambarı sistemleri.

Şekil 3.9, bu sistem öğelerinin mantıksal ilişkisini vermektedir. Burada veri ambarı diğer öğeler tarafından ortak kullanılmaktadır.

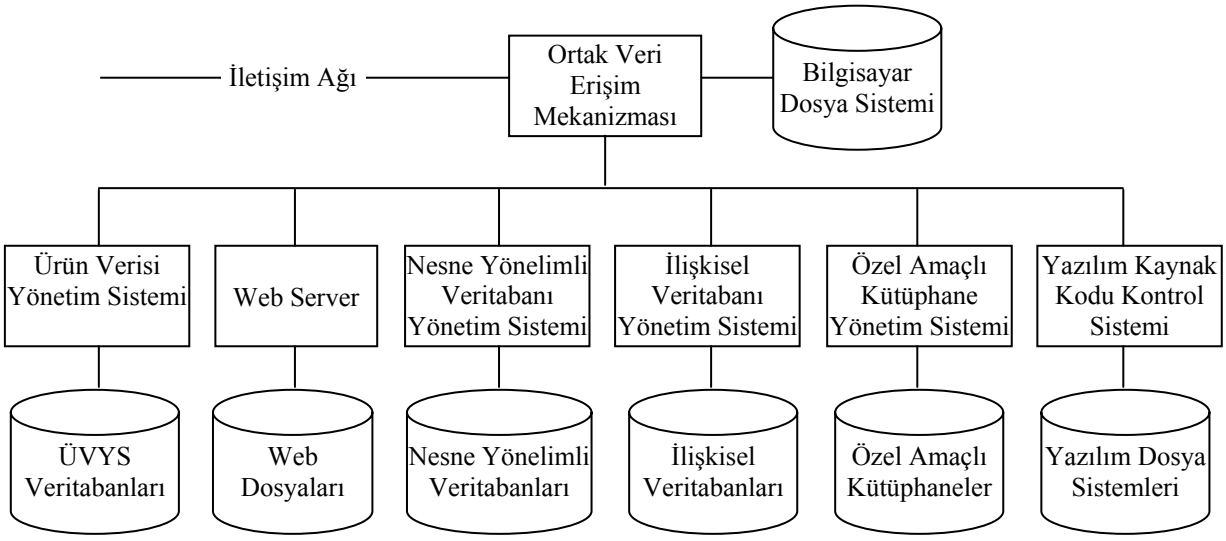


Şekil 3.9. Dağıtık imalat benzetimi mimarisi temel öğelerinin ilişkisi

Yazılım geliştirme ortamı benzetim motorları, görsellik sistemleri, entegrasyon altyapıları ve diğer yazılım uygulamaları geliştirmek için kullanılmaktadır. Tasarım, mühendislik, üretim planlama ve benzetim modeli geliştirme ortamı, benzetim ve imalat tarafından kullanılan modellerin ve verilerin oluşturulması için sistemler

içermektedir. Dağıtık imalat benzetimi çalışma ortamı benzetimleri yönetmek ve bütünleştirmek için benzetim motorları çalıştırma modelleri, görsellik sistemleri ve altyapı sistemlerini içermektedir. İmalat yönetim, kontrol, üretim ve destek sistemleri ortamı imalat operasyonlarını gerçekleştirmek için kullanılan "gerçek" sistemlerden oluşmuştur.

Yukarıda bahsedilen ortamları birbirleriyle ve dağıtık imalat veri ambarı ile bağlayan bir iletişim ağı olmalıdır. Veri ambarı çeşitli enformasyon sistemi ortamları tarafından kullanılan çeşitli veri depolarından ve yönetim sistemlerinden meydana gelmiştir. Tasarım, mühendislik, üretim planlama, gerçek imalat sistemleri, benzetim modeli geliştirme ve dağıtık imalat benzetimlerini çalıştırma ile ilgili verileri depolayan dosya sistemlerini, web sayfalarını, veritabanlarını ve kütüphaneleri birbirine bağlamalıdır. Veri ambarı tek bir bilgisayar üzerinde olabileceği gibi coğrafi olarak dağıtık ağ üzerinde de olabilmektedir. Dağıtık ortamdaki tüm yazılımların ve uygulamaların veri ambarına erişimini basitleştirmek için ortak bir veri erişim arayüzü kullanılmalıdır. Dağıtık veri ambarının öğeleri Şekil 3.10'daki gibi ilişkilendirilebilir.



Şekil 3.10. Dağıtık imalat veri ambarının bileşenleri

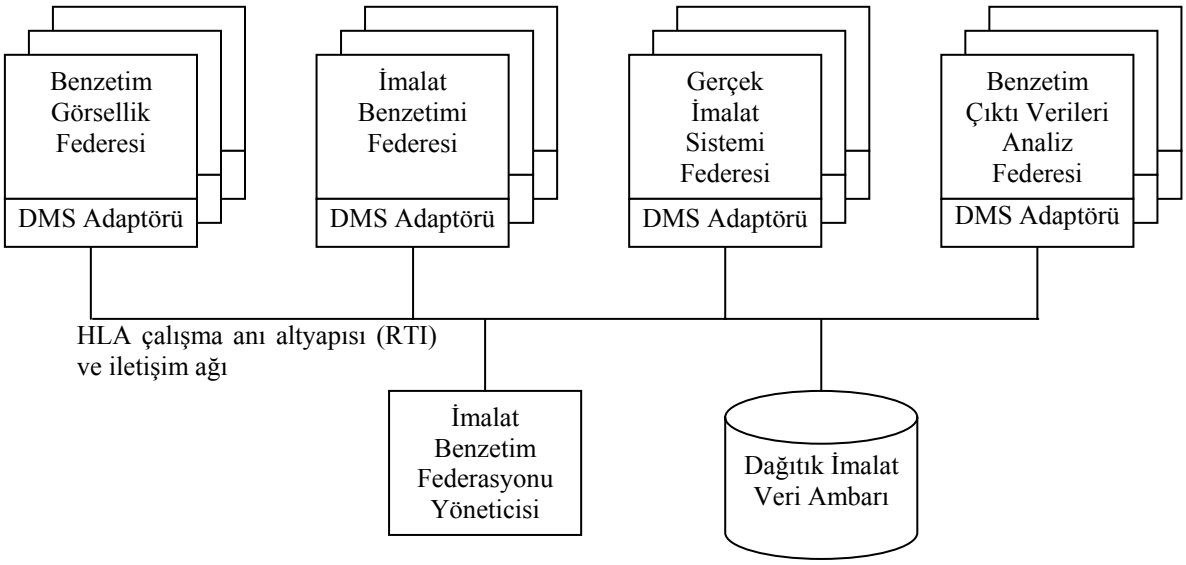
Şekil 3.10'da verilen bileşenlerin tümü her uygulamada olmak zorunda değildir, bununla birlikte uygulamanın türüne göre ilave bileşenler de eklenebilmelidir.

3.10.5. Benzetim sistemleri bakış açısı

Bu mimari bakış açısı, benzetimlerin inşa edilmesi, başlatılması, çalıştırılması, gözlenmesi, etkileşimi ve analizi ile ilgilidir. Bu bakış açısının temel öğeleri şunları içermektedir: benzetim koordinasyonu ve yönetimi, görsellik sistemleri, imalat veri hazırlama ve model geliştirme araçları, benzetim modelleri, kesikli olay ve proses benzetim motorları, bileşen modülü ve şablon kütüphanesi, matematiksel ve analitik modeller, girdi dağılımları, zamanlama ve olay takvimleri ve çıktı analiz araçları.

Şekil 3.11, dağıtık imalat benzetim ortamının çeşitli öğeleri arasındaki ilişkiyi göstermektedir. Bu ortamı bütünleştirici altyapı olarak HLA çalışma anı altyapısı (RTI) kullanılmıştır. HLA temelli dağıtık benzetim sistemine federasyon denmektedir. HLA RTI ile birbirine bağlanmış her bir benzetim, görsellik sistemi, gerçek üretim sistemi ve çıktı analiz sistemi ise federe olarak adlandırılmaktadır. Federasyon içerisinde paylaşılacak verilerin tanımlanması için ortak bir federasyon nesne modeli (FOM) oluşturulmalıdır. Her bir federe ise kullanacağı verileri tanımladığı, FOM'un öğelerini içeren bir benzetim nesne modeline (SOM) sahiptir.

Dağıtık imalat benzetimi (DMS: Distributed Manufacturing Simulation) federelerinin her birine DMS adaptörü eşlik etmektedir. DMS adaptörü, federelerin kullandığı tüm FOM nesnelerinin iletilmesi, alınması ve güncellemesi işlevlerini yerine getirecektir. DMS adaptörünün amacı, bazı yönetsel işlerin yapılmasını sağlayarak dağıtık imalat benzetimlerinin geliştirilmesini kolaylaştırmaktır. DMS adaptörü basitleştirilmiş zaman yönetim arayüzü, yerel nesne örneklerinin otomatik depolanması, başka federelerdeki ilgili nesne örneklerinin yönetilmesi, ilgili eylemlerin yönetimi ve izlenmesi gibi hizmetleri sağlamaktadır.



Şekil 3.11. Dağıtık imalat benzetim ortamı öğelerinin HLA RTI ile bütünleştirilmesi

Dağıtık imalat benzetimin düzgün bir şekilde işlemesini temin etmek için çeşitli işlevlerin yerine getirilmesi gerekmektedir. Dağıtık imalat benzetim federasyonu yöneticisi bu işlevleri gerçekleştirecek bileşendir. Federasyona katılan federelere başlangıç bilgilerini vermek, federasyon zaman yönetimine yardımcı olmak, kullanıcılara arayüz sağlayarak kullanıcıların federasyonu izlemesi, ona müdahale etmesini ve federasyon servislerini kullanmasını temin etmek gibi işlevleri yerine getirebilmelidir.

3.10.6. İmalat sistemleri bakış açısı

İmalat sistemleri bakış açısı, imalat organizasyonları ve sistemlerinin davranışlarının ve verilerinin modellenmesi ile ilgilenmektedir. Bu bakış açısının temel öğeleri şunlar olabilir:

1. İmalat organizasyonel şablonları ve yapıları ile birlikte işletme prosesleri ve organizasyonel modeller,
2. Tedarik zinciri sistemleri – rafineriler, fabrikalar, depolar, dağıtıcılar, taşıma sistemleri, toptancılar, perakendeciler, müşteriler,

3. İmalat tesisi departmanları ve alt sistemleri – tasarım, mühendislik, satın alma, finans, atölyeler, iş hücreleri, üretim hatları, iş istasyonları, depo alanları, nakliye,
4. Üretim kaynakları ve destek ekipmanları – makine araçları, muayene araçları, malzeme tutma sistemleri, ara depolar, robotlar, işçiler,
5. Araçlar ve malzemeler – kesici aletler, el aletleri, takım aparatlar, sarf malzemeleri, bileşenler, proses içi stok,
6. İmalat bilişim sistemleri – tasarım, mühendislik, üretim planlama ve çizelgeleme, araç yönetimi, atölye veri toplama sistemleri,
7. İmalat dokümanları ve verileri – iş akışları, iş emirleri, işler, ürün verileri, parça tasarımları, proses planları, üretim takvimi, çizelgeler, tesis yerleşimi ve diğer referans veriler.

Farklı imalatçılar farklı tedarik zinciri organizasyonları oluşturabilmektedir. Dağıtık imalat sistemi mimarisi, farklı sistem konfigürasyonlarına elverişli olmalı ve bütünleşmeyi sağlayabilmelidir. Mimari, belirli bir imalat organizasyonuna özel olmamalıdır. Ancak, dağıtık imalat benzetimi için bir FOM geliştirmeyi ve tanımlamayı gerektirmektedir.

FOM içerisindeki nesneler daha detaylı bilgi içeren dosya sisteminde kaydedilmiş dokümanlara referans olabilirler. Böyle bir doküman örneğin parça tasarım dosyası veya proses planı dosyası olabilir. XML yeni dosya türleri tanımlamak için kullanılabilmektedir. XML ile veri değerlerine ek olarak anlamsal bilgilere sahip veriler tanımlanabilmektedir. Bu yaklaşımın yararları şunlardır:

1. Desteklenen doküman tipleri seti kolayca genişletilebilir,
2. Her bir doküman biçimi kolayca değiştirilebilir,
3. Ticari benzetim yazılım araçları dokümanları oluşturabilir, öğelerine ayırabilir, yorumlayabilir ve görüntüleyebilir,
4. Başka kaynakların XML dokümanları desteklenebilir,

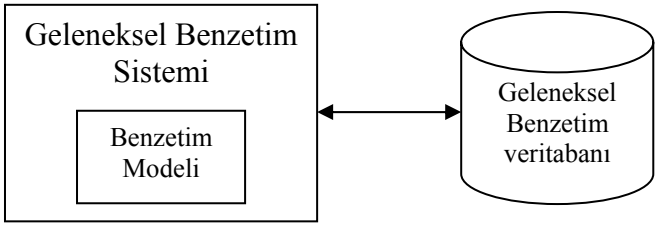
5. Dosya yapılarının çeşitli örnekleri oluşturularak dosyanın sahip olduğu anlamsal bilgiler defalarca kopyalanabilir,
6. XML destekli programlar veriyi zeki bir yaklaşımla görüntüleyebilir,
7. Dosyaların anlamsal geçerliliği test edilebilir.

Dağıtık imalat benzetim verisi olarak kaydedilebilecek bir çok standart dosya türleri mevcuttur. Bunlara CAD dosyaları veya resim dosyaları örnek gösterilebilir. Ancak standart olmayan dosya türleri de olabilmektedir. Çizelgeler, ürün ağaçları ve proses planları bunlar arasında sayılabilir. Kısa dönemli kullanımlar için standart olmayan dosyaların kabuledilebilir gösterimleri uygun görülebilir. Ancak bunlar üzerinde değişiklik yapmak ve geliştirmek diğer veri öğelerini etkileyebilecektir. Yeni dosya türlerini tanımlayabilecek ve genişletebilecek bir mekanizma gereklidir. XML bu gereksinimi sağlayabilecek bir mekanizmadır.

3.10.7. Benzetimlerin HLA-RTI kullanılarak bütünleştirilmesi

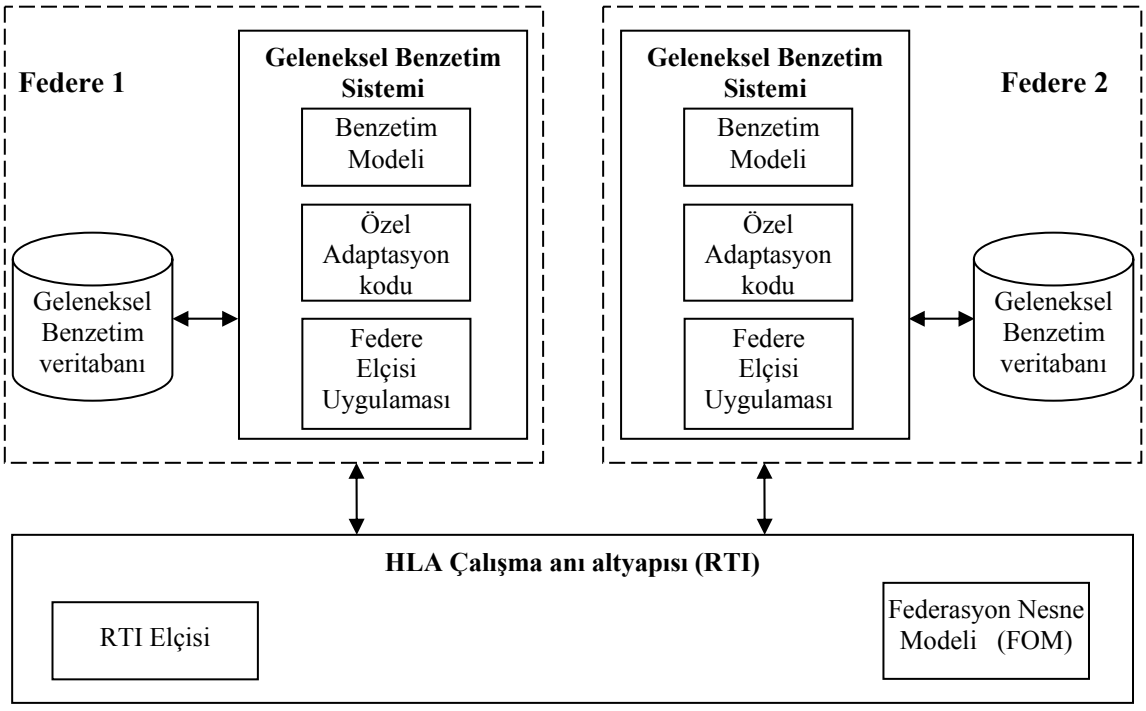
Bu kısımda geleneksel bir benzetim yazılımının HLA ile uyarlayıcı arayüzü kullanılarak birleştirilmesi ve dağıtık benzetim haline getirilmesinden bahsedilecektir. Geleneksel benzetim ile HLA teknolojilerini desteklemeyen imalat yönelimli veya genel amaçlı kesikli olay benzetim araçları kastedilmektedir. Bu sistemler genellikle hazır ticari benzetim paketleri (COTS: Commercial off-the-shelf) ile oluşturulmaktadır. Bununla birlikte genel amaçlı veya özel amaçlı bilgisayar yazılımları ile de tümden tasarlanabilmektedir.

Şekil 3.12'de basit bir dağıtık olmayan geleneksel benzetim uygulaması temsil edilmektedir. Bu benzetimler, modeli çalıştıran bir benzetim icra sistemine sahiptirler. Benzetim modeli, benzetimi yapılacak mantıksal sistemin davranışsal bir gösterimidir. Benzetim yazılımı çalıştırılan modelin görselliğini sağlamakla birlikte ve çalışma boyunca oluşan olayların istatistiksel raporlamasını da desteklemektedir. Benzetim için gerekli girdi verileri ve benzetim esnasında oluşan veriler bir veritabanında saklanmaktadır.



Şekil 3.12. Geleneksel bir benzetim sisteminin basitleştirilmiş görüntüsü

Şekil 3.13 ise geleneksel bir benzetimin HLA ile bütünleştirilmesini ve dağıtık hale getirilerek federasyonun oluşturulmasını göstermektedir. FOM, federeler arasında iletişimi yapılacak verilerin tanımlamasını vermektedir. Geliştirilen her dağıtık benzetimin FOM'u genellikle farklı olmaktadır. RTI Elçi (Ambassador) uygulaması, federelerin RTI'a bilgi göndermesinde gerekli olan arayüzü sağlamaktadır. RTI Elçisi federasyonun oluşturulmasının, nesnelerin sınıf tanımlamalarının, nesneler ve etkileşimler ile bilgi iletiminin ve zamanı ilerletmenin yönetilmesini sağlamaktadır.



Şekil 3.13. Geleneksel benzetimin HLA RTI ile bütünleştirilmesiyle federasyonun oluşturulması

RTI Elçisinin RTI'a bilgi iletimini sağlamasına benzer şekilde Federe Elçisi (Federate Ambassador) de RTI'dan bilgi alma mekanizmasını sağlamaktadır. Federe Elçisinin içerisinde, federasyonda meydana gelen değişimlere karşılık RTI'ın federeye nasıl bilgi göndereceğini tanımlayan yöntemler bulunmaktadır. Federe Elçisi arayüzü RTI yazılımı tarafından sağlanmamaktadır. HLA kuralları, Federe Elçisinin geleneksel benzetimler tarafından sağlanmasını gerektirmektedir. Bununla birlikte Federe Elçisi federasyon içerisinde kullanılan FOM'da tanımlanan bilgiler ile de uyumlu olmalıdır.

3.10.7.1. HLA/RTI bütünleştirmesi ile ilgili zorluklar

Geleneksel benzetim sistemleri HLA ile uyumlu tasarlanmadıkları için HLA ile çalışabilir hale getirilmeleri için yazılımlar yazılması gerekmektedir. Normalde bu kodlar karmaşıktır. Bununla birlikte bazı kodlar yeniden kullanılabilse de tasarlanan her dağıtık benzetim için birçok ilave kod yazılmalı ve değişiklikler yapılmalıdır. Aşağıda, HLA kullanımı için gerekli uyum kodlarının karmaşıklığı ve yeniden kullanılabilirliği ile ilgili bazı meseleler ele alınacaktır.

RTI arayüzü karmaşıklığı: RTI Ambassador arayüzü ve Federate Ambassador arayüzü içerisinde birçok yazılım diliyle yazılması gereken birçok metot/prosedür vardır. Kullanılan RTI'a, tasarlanan federasyona ve FOM içerisindeki veri tanımlamalarına bağlı olarak bir metodu çalıştırmak farklı çıktıların ve sonuçların oluşmasına neden olabilecektir. RTI arayüzlerinin zenginliği benzetimlerin bütünleştirilmesinde esneklik sağlamakla birlikte bunların öğrenilmesi oldukça uzun zaman almaktadır.

RTI'ın örtük/dolaylı metot çalıştırma mimarisi: RTI mimarisi "implicit invocation architecture" (örtük/dolaylı metot çalıştırma mimarisi) denilen yapıya dayanmaktadır. Bu yaklaşımda bir federe, RTI Ambassador arayüzünün bir metodunu çağırarak federasyonun çalışmasını etkileyerek durumunu değiştirebilmektedir. Federasyonun durumundaki değişiklik ile ilgili bilgiler, federeler tarafından çalıştırılan Federate Ambassador'a iletilir. Bu, etkin ve esnek bir

yöntem olmakla beraber geleneksel benzetim sistemlerinin bu yapıya uyarlanması zordur. Çünkü geleneksel benzetim sistemleri entegrasyon/bütünleşme için genellikle sadece prosedürel mekanizmaları desteklemektedir.

Geleneksel benzetimler tarafından uygun olmayan bütünleştirme mekanizmaları sağlaması: RTI'ın arayüzlerini kullanmak için RTI'ın desteklediği dillerden biri ile uyarlama kodları yazılmalıdır. Bu diller şu anda C, C++, Java ve CORBA IDL'dir. Bazı benzetim sistemleri bu dillerle yazılmış fonksiyonları çalıştırabilme mekanizmasına sahiptir, ancak bir çoğu da sahip değildir. Bu durum uyarlama kodlarının yazımını karmaşıklaştırmaktadır.

Ortak zaman yönetimi: Dağıtık benzetimde, daha öncesinde federelerin kendilerinin yaptığı zaman yönetimi kontrollünü terk edip ortak zaman yönetimine geçmeleri gerekmektedir. RTI birçok zaman yönetimi mekanizmasını desteklemektedir. Bunlardan uygun birinin seçilmesi ve geleneksel benzetimlerin bu zaman ilerletme yönetimine uyarlanması gerekmektedir. Uyarlama kodlarının yazılması önemli derecede öngörü ve tecrübe gerektirmektedir.

FOM nesne örneklerinin depolanması ve bakımı: Birçok geleneksel benzetim, parça ve makine gibi varlıkların temsil edilmesini sağlamakta ve bunlarla ilgili bilgiler benzetimin çalıştırılması esnasında saklanmaktadır. Bu varlıkların tanımlamaları farklı geleneksel benzetimlerde farklı bir şekilde yapılmış olabilmektedir. Bu varlıklarla ilgili bilgilerin benzetimler arasında iletimi için, bu varlıkların genel bir temsilinin nesne ve bu nesnenin özellikleri olarak FOM'da tanımlanması gerekmektedir. Ancak HLA/RTI nesne sınıf örneklerinin depolanması için bir mekanizmaya sahip değildir. HLA/RTI sadece belirli bir nesne örneğinin sahibi, nesne örneğinin sınıfı, nesne örneğinin özellikleri ile ilgili bilgileri saklamaktadır. Bu sebeple nesne örneklerinin saklanmasını geleneksel benzetimler sağlamalıdır. FOM nesnelerinin saklanması ve nesnenin geleneksel benzetim içerisindeki temsili ile FOM'daki temsili arasındaki durum değişikliklerinin koordinasyonu için uyarlama kodları yazılmalıdır.

3.10.7.2. Uyarlama arayüzü ile bütünleştirme

Bir önceki kısımda geleneksel imalat benzetimlerinin sadece HLA ile bütünleştirilmesi ile ilgili birtakım konulara değinilmiştir. Anlaşılacağı üzere dağıtık imalat benzetimi geliştirilirken Federate Ambassador ve uyarlama kodları da geliştirilmelidir. Bu oldukça zahmetli bir iştir ve bütünleştirilecek her geleneksel benzetim için benzer işlemler yapılmalıdır.

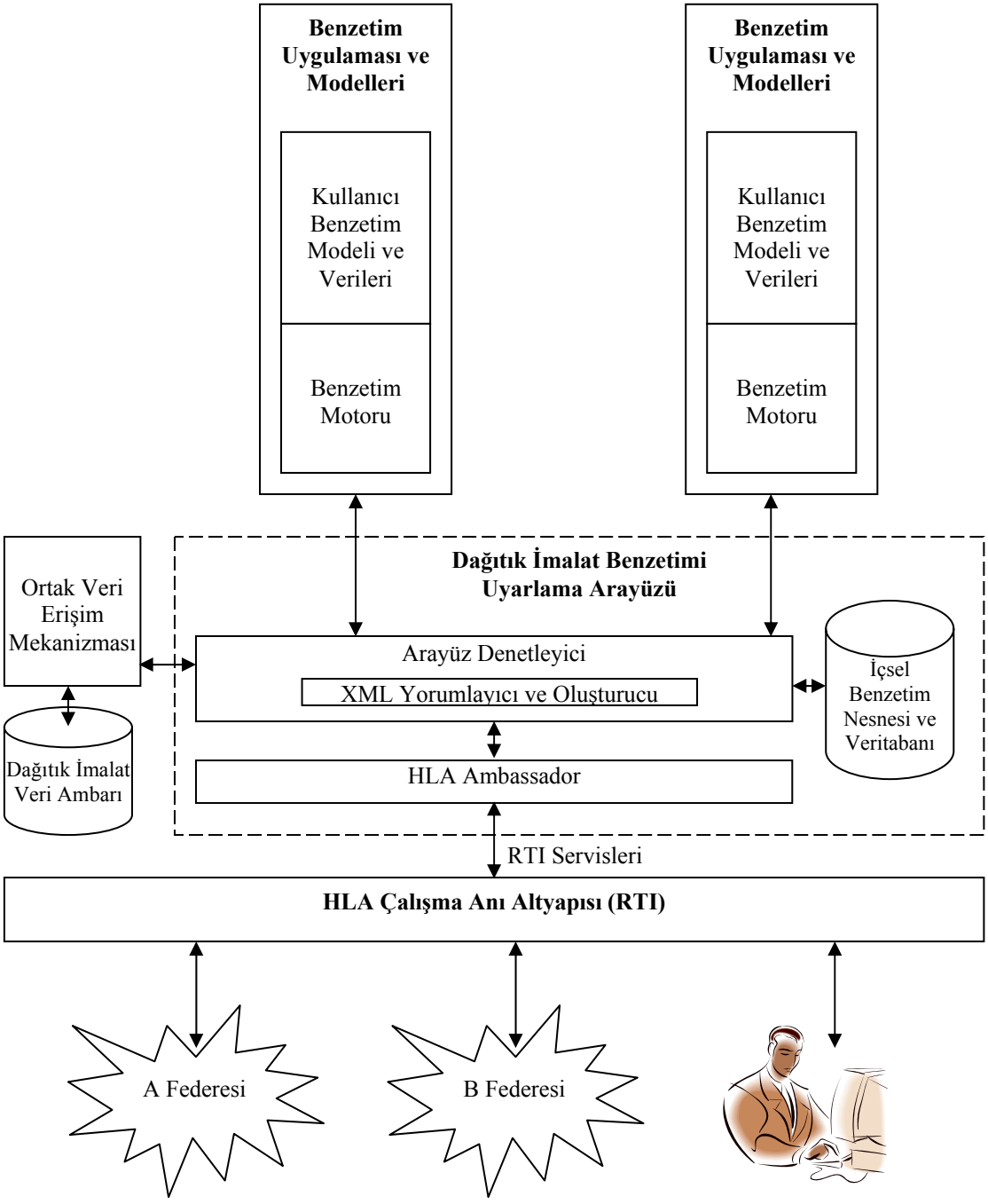
Şekil 3.14, geleneksel bir benzetimin uyarlama arayüzü ile bütünleştirilmesini göstermektedir. Geleneksel benzetimlerin doğrudan HLA/RTI ile bütünleştirilmesi yerine, uyarlama arayüzü ile etkileşmesinin sağlanması, benzetimlerin bir araya getirilmesini ve karşılıklı çalışmak üzere bütünleştirilmesini kolaylaştıracak ve hızlandıracaktır.

Uyarlama arayüzünün amacı, geleneksel benzetimlerin dağıtık benzetimlere entegrasyonunu kolaylaştıracak bir yöntem sunmaktır. Geleneksel benzetimlerin uyarlama arayüzü ile bütünleştirilmesi için de uyarlama kodları yazılmalıdır. Ancak geleneksel sümülasyonların bütünleşmesini sağlayacak arayüz basitleştirildiğinde, bütünleşme çabası, doğrudan HLA/RTI ile bütünleştirmeye göre oldukça azaltılmış olacaktır.

3.10.7.3. Uyarlama Arayüzünün Amaçları

Aşağıda uyarlama arayüzü kullanılarak dağıtık benzetimlerin bütünleştirilmesinin amaçlarından bahsedilecektir. Bu amaçlar gerçekleştirilebilirse dağıtık benzetimlerin Şekil 3.13'te ifade edildiği gibi bütünleştirilmesinden daha basit bir bütünleştirme gerçekleştirmek mümkün olacaktır.

Arayüz karmaşıklığını azaltmak: RTI Ambassador ve Federate Ambassador'da ayrı ayrı var olan yazılım metotları/prosedürleri yerine uyarlama arayüzünde her ikisini de kapsayacak ancak daha az miktarda kod ve yazılım prosedürü olacaktır.



Şekil 3.14. Benzetimlerin uyarlama arayüzü ile bütünleştirilmesi

Geleneksel benzetimlerden Federate Ambassador uygulamasını kaldırmak: Geleneksel benzetimlerde Federate ambassador geliştirme zorunluluğu olmayacaktır. Bu işlevi uyarlama arayüzü yerine getirecektir. Federate Ambassador'u uyarlama arayüzü uygulayacak ve RTI'dan bilgi bu şekilde uyarlanmış olarak, federenin anlayacağı tarzda alınacaktır.

Prosedürel geleneksel benzetimler ile bütünleşmeyi kolaylaştıran arayüz tanımlamak: Uyarlama arayüzünün metotlarını çalıştırmanın sonuçlarının birçoğu hemen geleneksel benzetime iletilecektir. Federeye asenkron olarak gönderilmesi gereken bilgilerin, uyarlama arayüzünde o federeyle ilişkili bir mesaj listesinde saklanması sağlanacaktır. Bu liste, federasyondaki diğer federelerin faaliyetleri sonucu oluşan bilgileri de saklayacaktır. Uyarlama arayüzü bu tür bilgilerin saklanmasını sağlayacak ve bu bilgilere geleneksel benzetimlerin erişmesi için metotlar içerecektir.

XML kodları içeren genel bir FOM nesnesi tanımlanması ile değişikliklerin enformasyon modeline etkisini azaltmak: Her dağıtık benzetim için farklı FOM'lar tanımlama gereksiniminden kurtulmak için federeler arasında iletilecek nesneler ve özellikleri FOM'da tanımlanmayacaktır. FOM'da genel bir nesne tanımlanacak, bu nesnenin örnekleri federeler arasında iletilecektir. Bu genel FOM nesnesi "string" tipinde bir özelliğe sahip olacak ve bu özellik XML parçaları içerecektir.

Federeler arasında iletilecek nesnelerin depolanmasını sağlamak: Federeler (benzetimler) arasında iletilecek nesnenin tanımı her benzetim içerisinde farklı şekilde yapılmış olabilmektedir. Benzetimler sadece kendi nesnelerin saklanmasını sağladıklarından ve RTI'ın nesneleri saklama mekanizması olmadığından dolayı iletilecek nesnelerin saklanması sağlanmalıdır. Bu görevi DMS arayüzü yapacaktır. Her uyarlama arayüzü geleneksel benzetimlerin federasyon içerisinde diğer federeler ile paylaşacakları nesneleri oluşturmasına, değiştirmesine ve silmesine imkân verecek metotlar içermektedir.

Zaman koordinasyonunu basitleştirmek: RTI, birçok esnek ve zengin zaman senkronizasyon yöntemi sunmakla beraber bunlar oldukça karmaşıktır. DMS uyarlama arayüzü "zaman adımlı" senkronizasyon yaklaşımını sağlamaktadır. Geleneksel benzetimlerin belirli bir benzetim zamanına gitmesini ve bu zamanın uygun olup olmadığının kontrol edilmesini DMS uyarlama arayüzü yapmaktadır. Uyarlama arayüzü, ilerlenecek zamanın uygunluğunu kontrol ettikten sonra geleneksel benzetim bulunduğu zamandan ilerlemek istediği zamana kadar benzetimi gerçekleştirecektir. Daha sonra, benzetimi çalıştırdığı esnada federasyonda hangi

işlemlerin gerçekleştirildiğinin bilgisini de DMS uyarlama arayüzünden isteyecektir (McLean ve Riddick, 2000a; McLean vd., 2005).

BÖLÜM 4. HLA TEMELLİ İMALAT SİSTEMLERİ TASARIM ÖRNEKLERİ

4.1. Giriş

Bu bölümde dağıtık imalat sistemlerinin HLA temelli tasarım örnekleri verilmiştir. Burada verilecek örnekler, doktora çalışması sırasında gerçekleştirilen çalışmalardır. Aşağıda verilen üç farklı HLA temelli dağıtık imalat sistemi tasarımı ile özellikle HLA'da nesne tanımlaması ile ilgili bilgiler detaylandırılmış olacaktır.

4.2. HLA Temelli Dağıtık İmalat Benzetimine Örnek Bir Senaryo

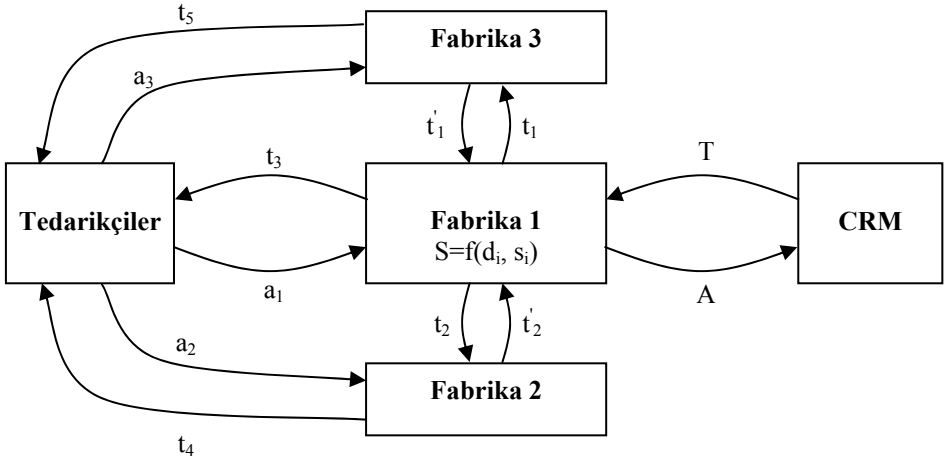
İmalat, benzetimin kullanıldığı en yaygın alanlardan birisidir. Fakat geleneksel benzetim sistemleri ile günümüz karmaşık imalat ortamlarını modellemek, bu modelleri birbirleriyle karşılıklı işler hale getirmedikçe mümkün değildir. Bunun için de dağıtık benzetim teknikleri ve mimarileri kullanılmalıdır. İmalat faaliyetleri artık genellikle dağıtık ortamlarda yapıldığından, dağıtık imalat benzetim modellerine ihtiyaç vardır. Dağıtık benzetim modelleri ile coğrafi olarak birbirinden farklı yerlerdeki imalat modelleri karşılıklı işler hale getirilebildiği gibi aynı ortamdaki alt modeller de birbiriyle işler hale getirilebilmektedir. Bu sebeple, modelleri birbirine bağlayan ve bilgi iletişimini sağlayan dağıtık benzetim mimarileri gerekmektedir. Dağıtık benzetim mimarisi yeniden kullanılabilirlik ve karşılıklı işleyebilirlik özelliklerine sahip olmalı ve böylece farklı ortamlarda ve farklı programlama dilleri ile modellenmiş modeller birbirine bağlanabilmelidir.

HLA, yukarıda bahsedilen tüm ihtiyaçları karşılayabilecek bir mimari sunmaktadır. Burada dağıtık bir imalat senaryosu verilerek HLA'nın dağıtık imalat benzetiminde nasıl kullanılabileceği ele alınacaktır. Nesne sınıf yapısı tablosu, etkileşim sınıf

yapısı tablosu, özellik tablosu, parametre tablosu ve veri yapısı tablolarına bazı örnekler verilecektir. Bu örneklerle SOM ve FOM'u tanımlamada kullanılan Nesne Model Şablonu da anlatılmış olacaktır (Uygun vd, 2006a).

4.2.1. Senaryo

Basit bir dağıtık imalat sistemi Şekil 4.1'deki gibi düşünülebilir. Fabrika 1, Fabrika 2 ve Fabrika 3 ile işbirliği halinde nihai ürünü üreten ana fabrikadır. Müşteri İlişkileri Yönetimi (CRM) Fabrika 1 ile çeşitli açılardan ilişki içerisindedir, ancak burada sadece konuyla ilgisi olması bakımından talep (T) ele alınmaktadır. CRM'den gelen talep Fabrika 1'in girdisidir. Burada benzetim yazılımı tarafından kapasite planlamada, ana üretim çizelgede, ürün ağacında, imalat kaynakları planlamasında ve benzeri yerlerde kullanılabilmektedir. Fabrika 1 bu işlemler sonucunda Fabrika 2 ve 3'e olduğu gibi tedarikçilerine t_i talebini iletmektedir. Fabrika 2 ve 3, t_1 ve t_2 bilgilerini işleyerek kendi gereksinimlerini belirlemede kullanabilmektedir. Fabrika 2 ve 3, Fabrika 1 gibi tedarikçilerle etkileşim halinde olabilmektedir. Tedarikçi t_3 bilgisini Fabrika 1'den, t_4 bilgisini Fabrika 2'den ve t_5 bilgisini de Fabrika 3'ten almakta ve fabrikalara a_1, a_2 ve a_3 bilgilerini iletmektedir. Fabrika 1 diğer fabrikalardan ve tedarikçilerden bilgileri alarak CRM'e arz (A) bilgisini göndermektedir.



Şekil 4.1. Basit bir dağıtık imalat sisteminin gösterimi

CRM, fabrikalar ve tedarikçiler farklı benzetim paketleri ile modellenmiş olabilirler. Bu sebeple benzetim modeli ile ilgili dağılımlar gibi ayrıntılı bilgiler (gelişler arası zaman, işlem süresi, bozulmalar, vb.) benzetim paketinin menüleri kullanılarak ayarlanabilmektedir. Burada önemli olan ayrı ayrı modellenmiş benzetim modellerinin birbirleriyle etkileşim edebilir hale getirilmesidir.

Buradaki karşılıklı işleyebilme problemi şudur: Sistemler arasındaki dönüştürücü mekanizması ne olursa olsun (temsilci, adaptör, etmen, vs.) bir benzetim modeli (federe) bilgisini RTI'a belirli bir format ile gönderebilmeli ve diğer benzetim modelleri bu bilgiyi belirli bir formatta alabilmelidir. Gönderilen ve alınan bilgilerin gösterimi konusunda ortak bir karar olmalıdır. Bu amaçla HLA-OMT, bilgilerin gösterimi konusunda çeşitli imkânlar sunmaktadır (Taylor, 2003).

4.2.2. Nesne modeli tanımlaması

HLA, federeler arasında bilgilerin belirli bir format kullanılarak iletilmesini gerektirmektedir. Bir federe içerisinde kullanılan ve diğer federeler ile paylaşılabilen bilgiler benzetim nesne modelinde (SOM) tanımlanmaktadır. Tüm federe içerisinde kullanılan bilgiler aynı zamanda federasyon nesne modelinde (FOM) de tanımlanmalıdır. Hem SOM'u hem de FOM'u tanımlamak için HLA nesne model şablonu (OMT) kullanılmaktadır. SOM ve FOM'u çeşitli yönlerden tanımlamak için OMT 14 tablo sunmaktadır, ancak federeler arasında bilgilerin iletimi için temelde 5 tablo kullanılmaktadır. Bunlar:

1. Nesne sınıf yapısı tablosu: HLA nesne sınıflarını kaydetmek ve bunların sınıf-altsınıf ilişkilerini tanımlamak için kullanılmaktadır.
2. Etkileşim sınıf yapısı tablosu: HLA etkileşim sınıflarını kaydetmek ve bunların sınıf-altsınıf ilişkilerini tanımlamak için kullanılmaktadır.
3. Nitelik tablosu: HLA nesne sınıf niteliklerinin (bu nitelikler federeler arasında iletilebilmektedir) özelliklerini tanımlamak için kullanılmaktadır.
4. Parametre tablosu: HLA etkileşim sınıf parametrelerinin özelliklerini tanımlamak için kullanılmaktadır.

5. Veri yapısı tablosu: Nesne modeli içerisinde verilerin gösterimini detaylandırmak için kullanılmaktadır.

Federeler arasında iletilecek bilgilerin gösterimini iki şekilde gerçekleştirebilmekteyiz: sınıf özelliklerinin değerlerini yayımlayarak veya etkileşim yayımlayarak (Taylor, 2003).

Federasyon için veya federelerin her biri için HLA nesne modeli tanımlarken tüm OMT bileşenleri tamamlanmalıdır. Bununla birlikte bazı tablolar boş olabilmektedir. Örneğin tüm federeler arasında etkileşim mümkün olmakla beraber, bazı federeler etkileşimde bulunmayabilmektedirler. Bu durumda bu federenin SOM'unda etkileşim sınıf yapısı tablosu sadece HLA tarafından gereken bir etkileşim içerecek ve parametre tablosu ise boş kalacaktır (IEEE, 2001b).

Senaryomuzda CRM'den Fabrika 1'e bir talep geldiğinde Fabrika 1 bunu bir siparişe dönüştürür. Diğer bir deyişle talep CRM'in bir çıktısıdır ve Fabrika 1 bunu girdi olarak alarak sipariş bilgisine dönüştürmektedir. Talebin sadece bir parça/ürün için yapıldığı varsayılmaktadır. Ürünün, talebin ve iş emrinin gösterimi aşağıdaki gibidir (HLA-CSPIF, 2003'ten uyarlanmıştır).

Urun = {ÜrünID, ÜrünAdı, diğer nitelikler}
Talep = {TalepID, ÜrünID, ÜrünAdı, TalepTarihi, TerminTarihi, TeslimalmaTarihi, Miktar, vb.}
Sipariş = {SiparişID, ÜrünID, ÜrünAdı, SiparişTarihi, TerminTarihi, BitişTarihi, vb.}

"HLAobjectRoot" nesne sınıfı, bir FOM veya SOM içerisindeki tüm nesnelerin üst sınıfıdır. Nesne sınıfları, federasyon içerisindeki katılımcı benzetimlere, HLA nesne örneklerinin bilgilerine abone olmayı sağlamaktadır. Nesne sınıfları aynı zamanda benzetim nesnelerinin karakteristiklerini (niteliklerini) tanımlamak için temel oluşturur. Bu nitelikler, nesnelerin örnekleri için değil nesne sınıfları için tanımlandıklarından dolayı, farklı nesne örnekleri aynı nitelikler için farklı değerler alabilmektedir. Bununla birlikte, temel HLA servisleri (IEEE, 2001a'da açıklandığı

gibi) bir federasyon icrası esnasında katılımcı federelerin nesne sınıflarına abone olmalarına imkân verdiği için, RTI, tüm nesne sınıfların ve bunların niteliklerin bilgisine ihtiyaç duymaktadır. Bu şekilde RTI, federasyon icrasına katılan federeler arasında HLA nesne bilgilerini sınıf yapısıyla yayabilmektedir (IEEE, 2001b). Tablo 4.1, CRM federesi için muhtemel bir nesne sınıf yapısını göstermektedir. CRM federasyonu içerisinde daha başka nesne sınıfları tanımlamak mümkündür, ancak biz burada Talep nesnesi ile ilgili olduğumuzdan tabloda Talep nesnesi verilmiştir.

Nesne sınıf yapısı tablosundaki her nesne sınıfının yanında parantez içerisinde, bu nesne sınıfın yayımlama veya abonelik durumunu belirten bilgi olmalıdır. Bu bilgiler ve anlamları şunlardır:

- P (Publish-Yayımlama): Federe, bu nesne sınıfının en azından bir niteliğini yayımlama yeteneğine sahiptir.
- S (Subscribe-Abone): Federe, bu nesne sınıfının en azından bir niteliğinin aboneliğinin olması yeteneğine sahiptir.
- PS (Publish/Subscribe-Yayımlama/Abone): Federe, bu nesne sınıfının en azından bir niteliğini yayma ve en azından bir niteliğinin aboneliğinin olması yeteneğine sahiptir.
- N(Neither-Hiçbiri): Federe, bu nesne sınıfının hiçbir niteliğinin yayımlanması ve abone olması yeteneklerine sahip değildir.

Tablo 4.1. CRM federesi için örnek bir nesne sınıf yapısı tablosu

HLAobjectRoot (N)	Talep (P)	OzelTalep (PS)	…
		KesinTalep (PS)	…
		TahminiTalep (PS)	…
	…	…	…

Tablo 4.2, nesne sınıf özellikleri tablosunun bir örneğini göstermektedir. HLA nesne örneklerinin nitelikleri RTI yoluyla güncellenmekte ve federasyon içerisindeki federelere yine RTI yoluyla iletilmektedir. Tablo 4.2, gerekli sınıf nitelikleri ile

genişletilebilmektedir. Tablodaki "Object" sütununda, nesne sınıf yapısı tablosundaki nesne isimlerinden biri olmalıdır. "Attribute" sütunu, nesne sınıfının niteliklerini listelemektedir. "Datatype" sütunu, niteliğin veri yapısını belirtmektedir. "Update type" sütunu, nesne niteliklerinin bir örneğinin güncellenme politikasını belirlemekte kullanılmaktadır. Bu politikalar şunlar olabilmektedir:

- Static: Niteliğin değeri statiktir; federe bu değeri başlangıçta ve gerekli olursa güncellemektedir.
- Periodic: Federe bu niteliği düzenli zaman aralıkları ile güncellemektedir.
- Conditional: Federe bu niteliği belirli şartlar oluştuğunda güncellemektedir.
- NA: Federe bu niteliğe değer sağlamamaktadır.

"Update condition" sütunu sınıf niteliğinin güncellenme politikasını açıklamaktadır. "D/A" sütunu ise niteliğin sahiplik bilgisini açıklamaktadır:

- D (Divest-Devredebilir): Federe, nesne niteliğini yayımlama yeteneğine sahiptir ve bu niteliğin örneğinin sahipliğini başka bir federeye devredebilmektedir.
- A (Acquire-Edinebilir): Federe, nesne niteliğini yayımlama yeteneğine sahiptir ve bu niteliğin örneğinin sahipliğini başka bir federeden edinebilmektedir.
- N (NoTransfer-Transfer edilemez): Federe, bu niteliğin örneğinin sahipliğini ne devredebilme ne de elde edebilme özelliğine sahiptir.
- DA (Divest/Acquire-Devredebilir/Edinebilir): Federe, bu niteliğin örneğinin sahipliğini hem devredebilme hem de elde edebilme özelliğine sahiptir.

"P/S" sütunu federenin veya federasyonun, nesne niteliğini yayımlama ve abone olma yeteneğini belirtmektedir. "Available dimensions" sütunu ise, eğer federe veya federasyon veri dağıtım yönetimi servisini kullanıyorsa, sınıf niteliğinin çeşitli boyutlarda ilişkilerini kaydetmektedir. "Transportation" sütunu ise nitelik için kullanılan taşıma türü için kullanılmaktadır. HLA tarafından iki taşıma türü kullanılmaktadır. Bunlar "HLAreliable" ve "HLAbestEffort" olup IEEE, 2001a'da ayrıntılı olarak açıklanmaktadır. Son sütun olan "Order" sütununda ise bu niteliğin örneklerinde kullanılacak bilginin geliş sırası tanımlanmaktadır. Bu sütunda kullanılabilecek değerler şunlardır:

- Receive: Bu niteliğin örnekleri, alıcı bir federeye belirsiz bir sırada iletilmektedir.
- TimeStamp: Bu niteliğin örnekleri, alıcı bir federeye belirli bir sırada iletilmektedir. Bu sıra, nesne örneğinin nitelikleri güncellendiğinde atanan bir zaman etiketi ile belirlenmektedir.

Tablo 4.2. CRM federesi için örnek bir nitelik tablosu

Object (Nesne)	**Attribute** (Özellik)	**Datatype** (Veritipi)	**Update Type** (Güncelleme Tipi)	**Update Condition** (Güncelleme Durumu)	**D/A**	**P/S**	**Available Dimensions** (Elverişli Boyutlar)	**Transpor-tation** (Taşıma)	**Order** (Sıra)
HLA object Root	HLAprivilege ToDelete Object	NA	NA	NA	N	N	NA	HLAreliable	Time Stamp
Talep	TalepID	HLAASCII string	Conditional	Gerektiğinde	N	PS	NA	HLAreliable	Time Stamp
	MusteriID	HLAASCII string	Conditional	Gerektiğinde	N	PS	NA	HLAreliable	Time Stamp
	UrunID	HLAASCII string	Conditional	Gerektiğinde	N	PS	NA	HLAreliable	Time Stamp
	UrunAdi	HLAASCII string	Conditional	Gerektiğinde	DA	PS	NA	HLAreliable	Time Stamp
	Miktar	HLAinteger 32BE	Conditional	Gerektiğinde	DA	PS	NA	HLAreliable	Time Stamp
	TalepTarihi	HLAfloat32 BE	Conditional	Gerektiğinde	DA	PS	NA	HLAreliable	Time Stamp
	TerminTarihi	HLAfloat32 BE	Conditional	Gerektiğinde	DA	PS	NA	HLAreliable	Time Stamp
	TeslimAlma Tarihi	HLAfloat32 BE	Conditional	Gerektiğinde	DA	PS	NA	HLAreliable	Time Stamp
…	…	…	…	…	…	…	…	…	…

Tablo 4.3'de Fabrika federeleri için (Fabrika 1, 2 ve 3 benzer nesne sınıf yapısı tablosuna sahip olabilirler) muhtemel bir nesne sınıf yapısı tablosu gösterilmektedir. Bu tablonun, federasyon içerisinde kullanılacak diğer gerekli bilgiler ile genişletilmesi mümkündür. Fabrika federelerinin nitelik tablosu ise Tablo 4.4'de verilmiştir.

Tablo 4.3. Fabrika federeleri için örnek bir nesne sınıf yapısı tablosu

HLAobjectRoot (N)	Urun (S)	Satınalınan (PS)	…	…
		Uretilen (PS)	…	…
		…	…	…
	Siparis (S)	SipariseYapma (PS)	…	…
		StokaYapma (PS)	…	…
		…	…	…
	Calisan (S)	Yonetici (PS)	YoneticiTipi1 (PS)	…
			YoneticiTipi2 (PS)	…
			…	…
		Isci (PS)	IsciTipi1 (PS)	…
			IsciTipi2 (PS)	…
			…	…
	…	…	…	…

Tablo 4.4. Fabrika federeleri için örnek bir nesne özellikleri tablosu

Object (Nesne)	**Attribute** (Özellik)	**Datatype** (Veritipi)	**Update Type** (Güncelleme Tipi)	**Update Condition** (Güncelleme Durumu)	**D/A**	**P/S**	**Available Dimensions** (Elverişli Boyutlar)	**Transport-tation** (Taşıma)	**Order** (Sıra)
Siparis	SiparisID	HLAASCII string	Conditional	Gerektiğinde	N	PS	NA	HLAreliable	Time Stamp
	MusteriID	HLAASCII string	Conditional	Gerektiğinde	N	PS	NA	HLAreliable	Time Stamp
	UrunID	HLAASCII string	Conditional	Gerektiğinde	DA	PS	NA	HLAreliable	Time Stamp
	Miktar	HLAinteger 32BE	Conditional	Gerektiğinde	DA	PS	NA	HLAreliable	Time Stamp
	SiparisTarihi	HLAfloat32 BE	Conditional	Gerektiğinde	DA	PS	NA	HLAreliable	Time Stamp
	TerminTarihi	HLAfloat32 BE	Conditional	Gerektiğinde	DA	PS	NA	HLAreliable	Time Stamp
	BitişTarihi	HLAfloat32 BE	Conditional	Gerektiğinde	DA	PS	NA	HLAreliable	Time Stamp
Calisan	ID	HLAASCII string	Conditional	Gerektiğinde	N	PS	NA	HLAreliable	Time Stamp
	Adi	HLAASCII string	Conditional	Gerektiğinde	DA	PS	NA	HLAreliable	Time Stamp
	Adres	Address Type	Conditional	İşçi istediğinde	DA	PS	NA	HLAreliable	Time Stamp
	Telefon	HLAASCII string	Conditional	İşçi istediğinde	DA	PS	NA	HLAreliable	Time Stamp
Urun	ID	HLAASCII string	Conditional	Gerektiğinde	N	PS	NA	HLAreliable	Time Stamp
	Tanim	HLAASCII string	Conditional	Gerektiğinde	N	PS	NA	HLAreliable	Time Stamp
	TedarikSuresi	HLAASCII string	Conditional	Gerektiğinde	N	PS	NA	HLAreliable	Time Stamp
	EmniyetStoku	HLAASCII string	Conditional	Gerektiğinde	N	PS	NA	HLAreliable	Time Stamp
…	…	…	…	…	…	…	…	…	…

Bir federe veya federasyonun tamamı bilgileri etkileşim yoluyla da iletebilmektedir. Etkileşim, bir federasyon icrası sırasında bir federenin yaptığı ve diğer federeleri etkileyebilen eylemler olarak tanımlanmaktadır (IEEE, 2001b). Benzetimler arası karşılıklı işlerliğin temel belirleyicilerinden bir tanesi etkileşimlerdir. Etkileşim sınıf yapısı tablosu sınıf-altsınıf ilişkisini belirlemekte ve nesne sınıf yapısı tablosuna benzemektedir. Tablo 4.5, CRM federasyonu için örnek bir etkileşim sınıf yapısı tablosunu göstermektedir. "HLAinteractionRoot" etkileşim sınıfı, FOM veya SOM içerisindeki diğer tüm etkileşim sınıflarının üst sınıfıdır. Etkileşim isimlerini takip eden P, S, PS ve N kısaltmaları, nesne sınıf yapısı tablosundaki gibi, federenin etkileşim sınıfının yayımlama veya abone olma yeteneğini belirtmek için kullanılmaktadır.

Tabloda verilen "TalepEylemleri" etkileşimi, yönetim operasyonları ve eylemleri izlemek için oluşturulmuştur. Bu etkileşim, CRM-Fabrika arasındaki birtakım ilişkileri gösteren alt sınıflara sahiptir. "TalepGonderildi", taleplerin Fabrika 1'e gönderilmesi içindir. "TalepKarsilandi" etkileşimi, gönderilen talebi Fabrika 1 tarafından karşılandığında kullanılmaktadır. Ödeme gerçekleştirildiğinde "FaturaOdendi" etkileşimi kullanılır ve talep "TalepKapandi" etkileşimi ile kapatılmaktadır.

Tablo 4.5. CRM federesi için örnek bir etkileşim sınıf yapısı tablosu

HLAInteractionRoot (N)	TalepEylemleri (P)	TalepGonderildi (PS)	…
		TalepKarsilandi (PS)	…
		FaturaOdendi (PS)	…
		TalepKapandi (PS)	…
	…	…	…

Benzer bir şekilde, fabrika federeleri için etkileşim sınıf yapısı tablosu örneği Tablo 4.6'da verilmektedir.

Tablo 4.6. Fabrika federeleri için örnek bir etkileşim sınıf yapısı tablosu

HLAInteractionRoot (N)	SiparisEylemleri (P)	SiparisAlindi (PS)	…
		SiparisAcildi (PS)	…
		SiparisBasladi (PS)	…
		SiparisBitti (PS)	
	…	…	…

Etkileşim parametreleri, etkileşim sınıflarına uygun ve faydalı bilgiler ilişkilendirilmek için kullanılmaktadır. Nesne sınıflarından farklı olarak etkileşim parametrelerinin kendisine abone olunamamaktadır. Nesne sınıflarının niteliklerine abone olmak mümkündür ancak etkileşim parametrelerine tek başına abone olunamamakta, etkileşimi sınıfına abone olunabilmektedir. Bu sebeple taşıma, geliş sırası gibi bilgiler parametre seviyesinde değil, etkileşim sınıfı seviyesinde tanımlanmaktadır. Bir etkileşim sınıfı herhangi bir parametreye sahip olmak zorunda değildir ancak, taşıma ve geliş sırası gibi bilgilerin tanımlanması için parametre tablosuna tüm etkileşimlerin yazılması gerekmektedir. Bu bilgiler parametre tablosunda tanımlanmakla beraber etkileşim sınıfı seviyesinde, etkileşim sınıfının bilgisi olarak tanımlanmaktadır. Etkileşim herhangi bir parametresi bulunmuyorsa, parametre sütununa "NA" girilmelidir.

Tablo 4.7. CRM federesi için örnek bir parametre tablosu

Interaction (Etkileşim)	**Parameter** (Parametreler)	**Datatype** (Veritipi)	**Available dimentions** (Elverişli boyutlar	**Transportation** (Taşıma)	**Order** (Sıra)
TalepGonderildi	UrunID	HLAASCIIString	DemandID	HLAreliable	TimeStamp
	Miktar	HLAinteger32BE			
	Tzamani	TlpZamani			
TalepKarsilandi	AlinisZamani	HLAfloat32BE	DemandID	HLAreliable	TimeStamp
	Dogruluk	CorrectDemand			
…	…	…	…	…	…

CRM federesinin parametre tablosu Tablo 4.7'de verilmiştir. Bu tablodaki "Datatype" sütunu parametrenin veri tipini tanımlamaktadır. Şayet etkileşimin herhangi bir parametresi bulunmuyorsa buraya "NA" girilmelidir. Tablodaki

"Available dimensions" sütunu ise, eğer federe veya federasyon bu etkileşim için veri dağıtım yönetimi (DDM) servislerini kullanıyorsa, etkileşim sınıfının ilişkilerini kaydetmektedir. Bu etkileşim için DDM servisleri kullanılmıyorsa bu sütuna "NA" girilmelidir. Etkileşimin taşıma türü ise "Transportation" sütununda belirtilmektedir. HLA tarafından "HLAreliable" ve "HLAbestEffort" olmak üzere iki taşıma tipi kullanılabilmekte ve bunlar IEEE, 2001a'da ayrıntılı olarak açıklanmıştır. Son sütun olan "Order" sütunu ise bu etkileşim için bilgilerin geliş sırasını belirtmektedir. Bu sütunda şu değerler kullanılabilmektedir:

- Receive: Bu etkileşim, alıcı bir federeye belirsiz bir sırada iletilmektedir.
- TimeStamp: Bu etkileşim, alıcı bir federeye belirli bir sırada iletilmektedir. Bu sıra, etkileşim gerçekleştiğinde atanan bir zaman etiketi ile belirlenmektedir.

Birçok OMT tablosunda veri tipi bilgisinin girilmesi gerekmektedir. Bu tablolarda kullanılan veri tipleri şu tablolardan birinde tanımlamaktadır: simple datatype table (basit veri tipi tablosu), enumarated datatype table (numaralı veri tipi tablosu), fixed record datatype table (sabit kayıt veri tipi tablosu), array datatype table (dizi veri tipi tablosu), variant record datatype table (çeşitli kayıt veri tipi tablosu). Bunlardan sabit kayıt veri tipi tablosuna örnek, Tablo 4.8'de verilmiştir.

Tablo 4.8. Fixed record datatype table (sabit kayıt veri tipi tablosu) için bir örnek

Record name (Kayıt ismi)	**Field** (Alan)			**Encoding** (şifreleme)	**Semantics** (Anlamı)
	Name (İsim)	**Type** (Tip)	**Semantics** (Anlama)		
DogruTalep	MiktarDogru	HLAboolean	Correct quantity	HLAfixedRecord	Talep doğru bir şekilde karşılandı mı?
	UrunDogru	HLAboolean	Correct item		
	KaliteTamam	HLAboolean	Quality status		
TlpZamani	TalepZamani	HLAfloat32BE	Demand time	HLAfixedRecord	Talep zamanı detayları
	TerminZamani	HLAfloat32BE	Due time		

Bu çalışmada HLA temelli dağıtık bir imalat benzetimi, bir senaryo verilerek incelenmeye çalışılmıştır. Bilgilerin RTI yoluyla standart bir şekilde iletilebilmesi için HLA bilgilerin belirli bir standart içerisinde tanımlanmasını gerekli kılmaktadır. Bu çalışmada, federasyon içerisinde iletilecek bilgilerin nasıl tanımlandığını

örneklendirmek için nesne sınıf yapısı tablosu, etkileşim sınıf yapısı tablosu, nitelik tablosu, parametre tablosu ve veri tipi tablosuna örnekler verilmiştir.

4.3. HLA Temelli Bakım ve Üretim Çizelgeleme Sistemi

Üretim ve bakım çizelgeleme imalatın en önemli konuları arasındadır. Çizelgeleme ile ilgili literatürün çoğunda makinelerin her zaman elverişli olduğu varsayılmaktadır. Hâlbuki gerçekte, arızalar ve önleyici bakım çizelgelemesi endüstride rastlanan durumlardır (Graves ve Lee, 1999). Bakım çizelgeleme ve üretim çizelgeleme birbirlerine bağlıdır, birinin çizelgelenmesi diğerinin çizelgelemesini etkilemektedir. Bu sebeple her iki çizelgelemenin ilişkili ve karşılıklı işler halde tasarlanması verimli çizelgelerin elde edilmesini sağlayacaktır. Bir bakımın ne kadar önemli olup olmadığını üretim çizelgeleme sistemi bilemez. Eğer bakım acil derecesinde önemli ise üretim çizelgelemede değişiklikler yapılmalıdır. Dolayısıyla her iki sistemin haberleşir olması gerekmektedir. Burada, sistemler arasında etkileşimi sağlamak ve bilgi iletiminde bulunmak için HLA temelli bütünleşik bakım ve üretim çizelgeleme önerilecektir (Uygun vd., 2006b).

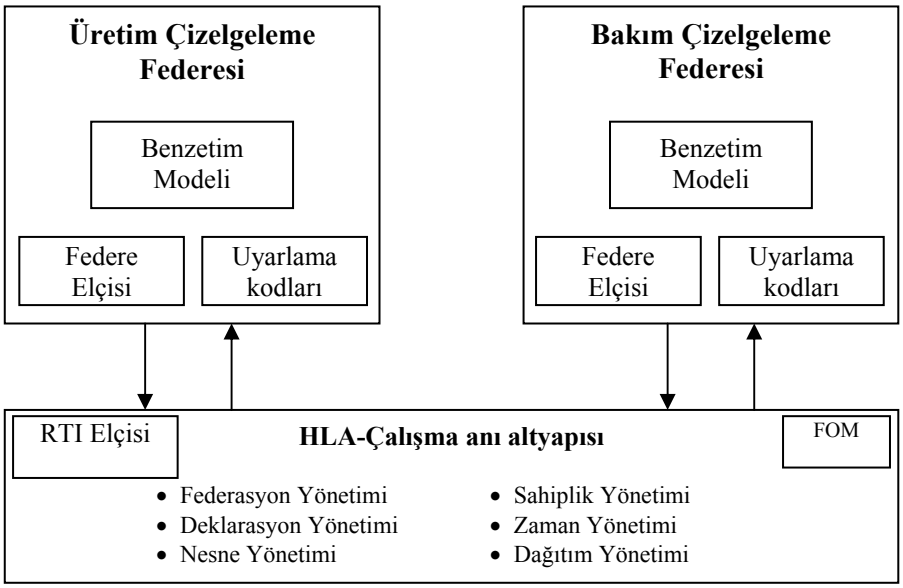
Önleyici bakım, makine ve teçhizatı belirli şartlar altında tutmak için gerekli planlanmış bakımlardır. Bu politika, makinenin beklenen en erken arızalanma zamanına dayanmaktadır. Periyodik muayene, durum izleme/kontrolü, kritik parça değişimi, kalibrasyon, yağlama ve yakıt temini, periyodik bakıma örnekler arasındadır.

Üretim çizelgeleri, uygun bir önleyici bakım politikası seçilmediği için genellikle ekipman arızaları sebebiyle sekteye uğramaktadır. Ancak yine de, üretimi hızlandırma adına öngörülen koruyucu bakımlar geciktirilmektedir. Her ne kadar imalat verimliliği üretim çizelgeleme ve koruyucu bakım planlarının eş zamanlı optimizasyonu ile iyileştirilecek olsa da gerçek imalat ortamlarında bunlar birbirinden bağımsız olarak planlamakta ve uygulanmaktadır (Sortrakul vd., 2004).

Üretim çizelgeleme ve koruyucu bakım planlamayı eş zamanlı düşünen çalışma sayısı çok değildir. Cassady ve Kutanoğlu (2003) üretim çizelgeleme ve koruyucu

bakım planlamayı eş zamanlı belirleyen bütünleşik bir model önermişlerdir. Sloan ve Shantikumar (2000) tek makineli durum için birleşik bir metot ve uygulama önermektedir. Allaoui ve Artiba (2004) bakım kısıtları altında, akış sürelerine ve teslim zamanlarına dayalı çeşitli amaçları optimize edecek hibrit bir akış çizelgeleme problemini ele almışlardır. Graves ve Lee (1999), çeşitli zaman aralıklarında makine bakımının yapılması gerektiği ve bu sebeple bakım esnasında makinenin elverişli olmadığı duruma göre tek makine çizelgeleme problemini çalışmışlardır.

Üretim çizelgeleme ve bakım planlama problemlerini birbirinden bağımsız çözmek, aralarında olması gereken bilgi alışverişini ihmal etmek anlamına gelmektedir. Burada iki sistem arasında karşılıklı işlerliği sağlamak için HLA temelli üretim ve bakım çizelgeleme ele alınacaktır. Üretim çizelgesi ve bakım çizelgesi birer federe olarak ele alınacak ve HLA-RTI ile etkileşim halinde olacaklardır. Şekil 4.2, üretim ve bakım çizelgeleme federasyonunu göstermektedir.

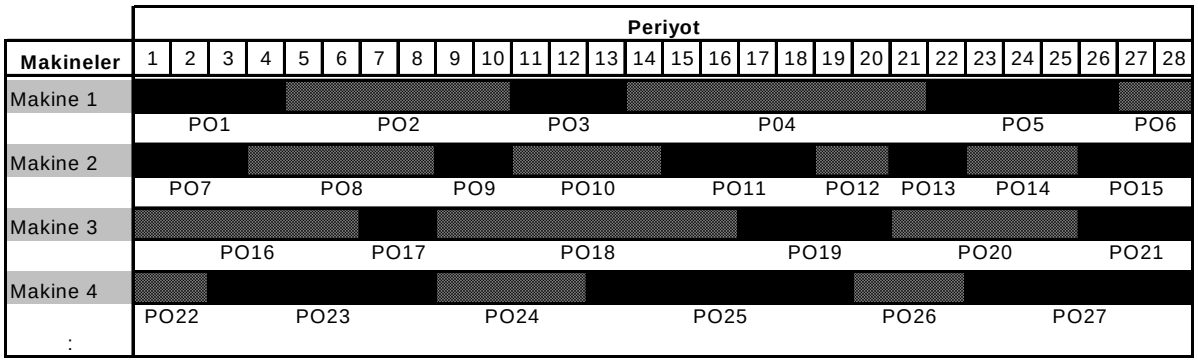


Şekil 4.2. Bütünleşik üretim ve bakım çizelgeleme federasyonu

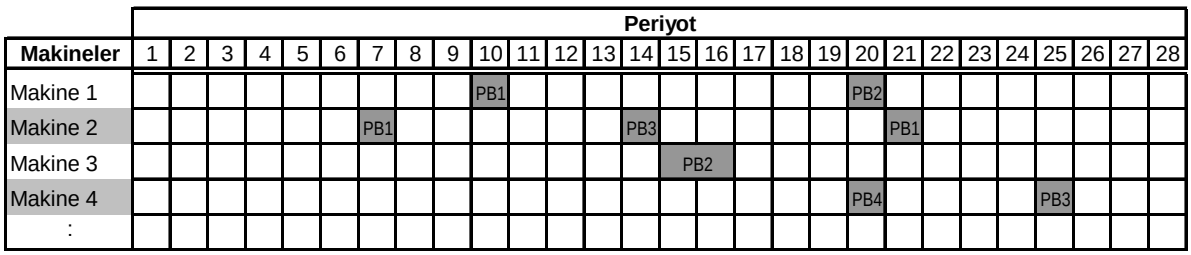
RTI elçisi (ambassador), federelerin RTI'a bilgi gönderme arayüzünü oluşturmaktadır. Bu arayüz federasyonun oluşturulmasını, nesne sınıf yapılarını, nesne ve etkileşimler ile bilgi iletimini ve federasyon zamanının ilerletilmesini

yönetmektedir. Federe elçisi ise RTI'dan gelen bilgileri almaktadır ve her federede var olmak zorundadır. Federe elçisi, federasyonda meydana gelen değişiklikleri RTI yoluyla federeye bildirmektedir. Bildirimler, herhangi bir federenin isteği üzerine geribildirim şeklinde de gerçekleşmektedir. FOM, federasyon içerisinde iletilecek tüm bilgilerin tanımlamalarını içermektedir. Günümüzde hiçbir yazılım HLA'yı doğrudan desteklemediğinden sistemlerin HLA ile haberleşmesini temin etmek için birtakım uyarlama kodları yazılmalıdır. Bu, RTI'dan gelen bilgilerin sistemin anlayacağı hale ve sistemden RTI'a gönderilecek bilgilerin ise RTI'ın anlayacağı hale çevrilmesini sağlayarak arayüz oluşturmaktadır.

Tipik bir üretim planı Şekil 4.3'deki gibidir. Burada makinelere yüklenmiş farklı iş emirleri (PO1, PO2, vs.) görülmektedir. Verilen bu üretim çizelgesinde makinelerin bakımları görülmemektedir. Şekil 4.4 ise aynı makinelerin aynı periyotlardaki önleyici bakım (planlı bakım) çizelgesini vermektedir.



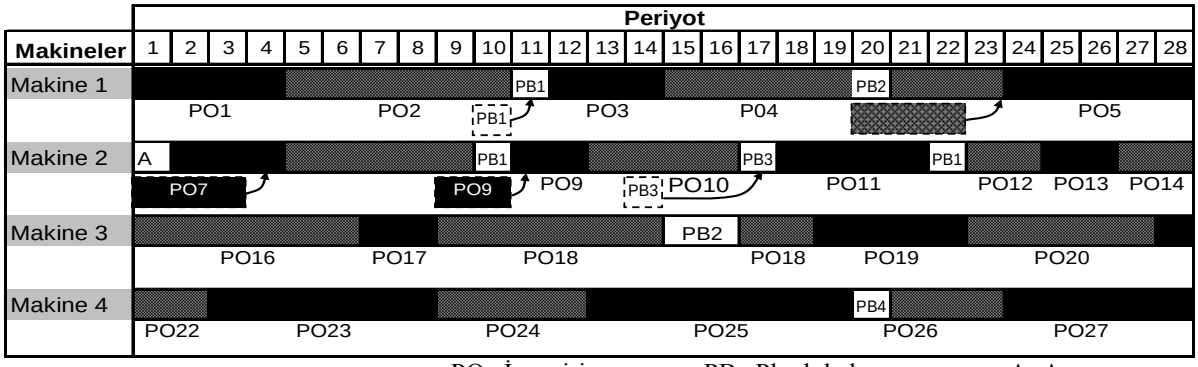
Şekil 4.3. Üretim çizelgesinin basit bir görünümü



Şekil 4.4. Bakım planlama çizelgesinin basit bir görünümü

Üretim çizelgesi ve bakım çizelgesi ayrı ayrı gösterilmiştir. Gerçekçi bir çizelgelemede ise bunların bütünleşik olması gerekmektedir. Her iki çizelgelemenin bütünleştirilmiş hali ise Şekil 4.5'de verilmiştir.

Üretim çizelgelemeci, müşteriye doğru bir teslim tarihi vermek için çizelgeleme sırasında bakım çizelgesini ve muhtemel arızaları göz önünde bulundurması gerekmektedir. Çizelgelemeci, müşteriden önemli veya acil bir sipariş aldığında önceden planlanmış koruyucu bakımın tarihini değiştirerek onun yerine üretim yapmayı tercih edebilmektedir. Bu durumda bakım çizelgeleme değiştirilmeli ve güncellenmelidir.



PO_i: İş emiri numarası, PB_i: Planlı bakım numarası, A: Arıza

Şekil 4.5. Bütünleşik üretim ve bakım çizelgesi

Örneğin birinci makinenin 10. periyotta gerçekleştirilmesi gereken planlanmış bakımı, aynı periyotta devam eden bir üretim (PO2) söz konusu olduğu için 11. periyota ertelenmiştir. Bununla birlikte, bazen önceden planlanmış koruyucu bakım veya daha önce ertelenmiş ve artık daha fazla ertelenemeyecek koruyucu bakım, öneminden ve önceliğinden dolayı ertelenmez, bunun yerine o periyoda denk gelen üretim ertelenmektedir. Örneğin üçüncü makinenin 15 ve 16. periyotlarında gerçekleştirilmesi gereken planlı bakım önceliğinden dolayı ertelenmemiş, aynı periyotlarda devam etmesi gereken PO18 üretimi durdurularak ertelenmiştir ve planlanmış bakımdan sonra tamamlanmıştır (Şekil 4.5'e bakınız).

Üretim çizelgelemeci, koruyucu bakımın önemini ve önceliğini bilemeyebilmektedir. Bu sebeple bakım çizelgelemeden bu ve benzeri bilgileri edinmek zorundadır. Dolayısıyla üretim çizelgeleme sistemi ile bakım çizelgeleme sistemi arasında bilgi alışverişi ve etkileşim gerçekleştirilmelidir. Bu iki sistemi karşılıklı işler hale getirmek için HLA kullanılabilir. Bakım politikasının belirlenmesi, çizelgeleme kurallarının buna göre işletilmesi ve makinelere iş yüklenmesi, her iki sistemin birlikte ele alınmasını gerektirmektedir. İki çizelge arasında uzlaşma, HLA ile sağlanabilir.

Aşağıda, iki sistemin uzlaşması için transfer etmesi gereken bilgilerin tanımlaması verilecektir.

4.3.1. Nesne modeli tanımlaması

Bu bölümde verilen ilk örnekte nesne modeli tanımlaması ile ilgili açıklamalar verilmişti (Bölüm 4.2.2). Bu sebeple burada bilgi tekrarı olmaması için ayrıntıya girilmeden üretim çizelgesinde ve bakım çizelgesinde tanımlanması gereken bilgilerin bir kısmı verilecektir.

Tablo 4.9 bakım çizelgeleme federesinde tanımlanması gereken nesnelerin bir kısmını vermektedir. Benzer şekilde üretim çizelgeleme federesinde tanımlanması gereken bazı bilgiler Tablo 4.10'da verilmiştir.

Tablo 4.9. Bakım çizelgeleme federesi için nesne sınıf yapısı tablosu

HLA object Root (N)	Bakım (PS)	OnleyiciBakim (PS)	HaftalikOnleyiciBakim (PS)	Yaglama (PS)
				Ayar (PS)
				Temizlik (PS)
			AylikOnleyiciBakim (PS)	
			AltiAylikOnleyiciBakim (PS)	
			SayacliKoruyucuBakim (PS)	
		Ariza (PS)	AcilAriza (PS)	
			ErtelenebilirAriza (PS)	
		Kontrol (PS)		
	Personel (S)	Mekanikci (PS)		
		Elektrikci (PS)		

Tablo 4.10. Üretim çizelgeleme federesi için nesne sınıf yapısı tablosu

HLA object Root (N)	isEmri (PS)	Acil_isEmri (PS)	
		Kesin_isEmri (PS)	
		Planlanmis_isEmri (PS)	
	Makine (PS)	Tip1 (PS)	
		Tip2 (PS)	
	Calisan (PS)	Kaynakci (PS)	
		Montajci (PS)	

Nesnelerin özellikleri olmaktadır. Bilgiler nesne özelliklerinin değeri olarak saklanmaktadır. Tablo 4.11 bakım çizelgeleme federesi için tanımlanmış nesnelerin bazı özelliklerini vermektir.

Tablo 4.11. Bakım çizelgeleme federesi için özellik tablosu örneği

Object (Nesne)	**Attribute** (Özellik)	**Datatype** (Veritipi)	**Update Type** (Güncelleme Tipi)	**Update Condition** (Güncelleme Durumu)	**D/A**	**P/S**	**Available Dimensions** (Elverişli Boyutlar)	**Transpor-tation** (Taşıma)	**Order** (Sıra)
HLA object Root	HLAprivilege ToDelete Object	NA	NA	NA	N	N	NA	HLAreliable	Time Stamp
Bakım	BakımEmriID	HLAASCII string	Conditional	Gerekirse	N	PS	NA	HLAreliable	Time Stamp
	MakineID	HLAASCII string	Conditional	Gerekirse	N	PS	NA	HLAreliable	Time Stamp
	BasZaman	HLAfloat32 BE	Conditional	Gerekirse	DA	PS	NA	HLAreliable	Time Stamp
	Suresi	HLAfloat32 BE	Conditional	Gerekirse	DA	PS	NA	HLAreliable	Time Stamp
	BitZaman	HLAfloat32 BE	Conditional	Gerekirse	DA	PS	NA	HLAreliable	Time Stamp
	Gerekli İsYetenegi	HLAASCII string	Conditional	Gerekirse	DA	PS	NA	HLAreliable	Time Stamp
Makine	MakineID	HLAASCII string	Conditional	Gerekirse	N	PS	NA	HLAreliable	Time Stamp
	MakineTanim	HLAASCII string	Conditional	Gerekirse	DA	PS	NA	HLAreliable	Time Stamp
	SonBakim Tarihi	HLAfloat32 BE	Static	Gerekirse	DA	PS	NA	HLAreliable	Time Stamp
	Durumu	MakDurum	Conditional	Gerekirse	DA	PS	NA	HLAreliable	Time Stamp

Federeler arası iletişim sadece nesne özelliklerinin güncellenmesi ile değil, etkileşimler ile de gerçekleştirilmektedir. Tablo 4.12 bakım çizelgeleme federesi için örnek bir etkileşim sınıf yapısı tablosunu göstermektedir. Benzer şekilde Tablo 4.13 üretim çizelgeleme federesi için etkileşim sınıf tablosu örneğini vermektedir.

Tablo 4.12. Bakım çizelgeleme federesi için örnek bir etkileşim sınıf yapısı tablosu

HLA Interaction Root (N)	Bakim Etkilesimi(PS)	BakimEmriTalebi (PS)	Arizadan (PS)	
			PlanliBakimdan (PS)	
		BakimEmriAcildi (PS)		
		BakimBasladi(PS)	ArizaBakimBasladi (PS)	
			PlanliBakimBasladi (PS)	
		BakimBitti (PS)		

Tablo 4.13. Üretim çizelgeleme federesi için örnek bir etkileşim sınıf yapısı tablosu

HLA Interaction Root (N)	isEmri Etkilesimi (PS)	BakimEmriTalebi (PS)	ArizaSebebiyle (PS)
			PlanliBakimdan (PS)
		isEmriAcildi (PS)	
		isEmriBasladi (PS)	
		isEmriKapandi (PS)	

Nesnelerin özellikleri olduğu gibi etkileşimlerin de parametreleri vardır. Nesnelerin özellikleri güncellenerek iletişim sağlanırken, nesne özelliklerinden farklı olarak parametreler kendi başlarına yayımlanıp abone olunmaz, etkileşimin kendisi yayımlanıp abone olunur. Bu sebeple nesnelerin parametreleri olmak zorunda değildir.

HLA'nın tanımlanmış çeşitli veri tipi vardır. Bunun dışında kullanıcı kendi veri tipini de tanımlayabilmektedir. Bunun için HLA'da veri tipi tanımlama tabloları mevcuttur. Bunlardan örnek olarak numaralı veri tipi tablosu Tablo 4.14'de verilmiştir.

Tablo 4.14. Numaralı veri tipi tablosu örneği

Name (İsim)	**Representation** (Gösterim)	**Enumerator** (Sayı karşılığı)	**Values** (Değer)	**Semantics** (Anlamı)
isYetenekSeviyesi	HLAinteger32BE	Yeni	1	İş gücü yetenek seviyeleri
		Orta	2	
		Sertifikali	3	
		Usta	4	
MakDurumu	HLAinteger32BE	Calisir	1	Makine durumları
		PlanliBakim	2	
		Ariza	3	
		Diger	4	
PB_oncelik	HLAinteger32BE	DusukOncelik	1	Planlı Bakım önceliği
		YuksekOncelik	2	

4.4. Tedarik Zincirinin HLA ile Bütünleştirilmesi

İmalat dünyasında tedarik zincirinin önemi büyüktür. Etkili bir tedarik zincirinde bilginin paylaşılması, doğru zincir üyesine doğru bilginin doğru zamanda ulaşması gerekmektedir. Bununla birlikte bilginin doğru biçimde ulaştırılması gerekliliği de artık kaçınılmazdır. Coğrafi olarak dağıtık organizasyonlar bilişim ve iletişim teknolojilerini kullanarak tedarik zincirleri oluşturmakta ve ortak mimariler kullanarak bilgilerini birbirlerine iletmektedirler. Bu durum her bir zincir üyesinin pazarda rekabet gücünü artırmakta ve müşteriye daha iyi ürün ve hizmet ile sonuçlanmaktadır (Kubat ve Uygun, 2007).

Tedarik zincirleri dağıtık benzetim için tipik bir örnek teşkil etmektedir. Bilgisayar destekli bilişim ve iletişim teknolojilerinin artan önemi ile birlikte tedarik zincirlerinde dağıtık benzetim ve iletişim mimarileri önem kazanmaktadır. Tedarik zincirleri doğal olarak karmaşık yapıdadırlar, bu sebeple geleneksel benzetim teknikleri ile modellenmesi pek mümkün değildir. Bu amaç için HLA gibi dağıtık benzetim mimarilerine ihtiyaç duyulmaktadır.

HLA dağıtık benzetim sistemlerinin modellenmesine veya dağıtık ortamdaki modellerin birbirleriyle etkileşmesine yardımcı olduğu için tedarik zincirinde de HLA'nın kullanılması oldukça faydalı olacaktır. Tedarik zincirinde, nihai ürünün

üretilmesi için birçok alt üretim işlemleri farklı coğrafi bölgelerde gerçekleşmekte ve bunlar arasında bilgi, malzeme, maliyet gibi unsurlar hareket etmektedir. Tedarik zincirinde nihai ürünü üretmenin toplam üretim zamanı ve maliyetinin azaltılmasında etkin bilgi akışı hayatidir. HLA bilginin doğru modellenmesi ve etkin bir şekilde iletimi için kullanılabilecek bir mimaridir. Burada tedarik zincirinde HLA'nın kullanılması ile açıklamalara yer verilmiştir (Kubat ve Uygun, 2007).

4.4.1. Tedarik zinciri yönetimi

Literatüre bakıldığında tedarik zincirinin tanımı genellikle birbirlerine benzemekte, aynı amaç farklı şekillerde ifade edilmektedir. Buradan tedarik zincirinin ne olduğu ile ilgili bir karmaşanın olmadığı ve tanımı ile ilgili evrensel bir kabulün olduğu anlaşılmaktadır. Genel bir ifadeyle tedarik zinciri yönetimi malzemelerin temini, bunların yarı mamul ve mamullere dönüştürülmesi ve nihai mamullerin müşterilere ulaştırılması ile ilgili imkânların, tesislerin ve dağıtım birimlerinin oluşturduğu bir ağdır. Simchi-Levi vd. (2000) tedarik zinciri yönetimini, mamullerin üretimini ve dağıtımını doğru miktarlarda, doğru yerlerde ve doğru zamanlarda gerçekleştirerek tüm sistemin maliyetini minimize etmeye çalışırken gereksinimleri de karşılayabilmek amacıyla tedarikçileri, imalatçıları, ambar ve depoları etkin bir şekilde bütünleştirmek için uygulamaya konulan bir dizi yaklaşım olarak tanımlamaktadır.

Tedarik zinciri yönetimi süreçleri ile maliyetler, stok seviyeleri ve ürünlerin pazara çıkma süreleri azaltılmaya çalışılırken müşteri ilişkileri ve tatmini artırılmaya çalışılmaktadır. Bu amaçları gerçekleştirebilmek için tedarikçilerle üreticiler arasındaki malzeme ve bilgi akışını koordine ve kontrol etmek için yapısal yaklaşımlar gerekmektedir. Markland vd. (1995), tedarik zincirinin dört karakteristiğinin olduğunu söylemektedir:

- Tedarik zincirleri bağımsız fonksiyonlar bütünü değildir, aksine bütünleşik tek bir sistemdir.
- Tedarik zincirleri stratejik karar vermeye dayanmaktadır.

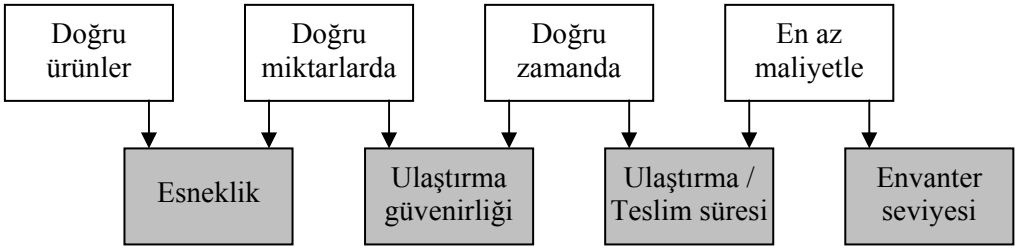
- Tedarik zincirleri envanterleri yeniden organize eder ve zincirin parçaları arasında denge oluşturur.
- Tedarik zinciri boyunca sistem entegrasyonu mevcuttur.

Tedarik zinciri yönetiminin ilk çıkışı JIT ve Toyota Üretim Sistemi ile ilgisi vardır. Bu anlayışa göre tedarikçilerin Toyota fabrikası ile ilişkileri düzenlenmek, az ve doğru miktarda parçayı doğru zamanda temin etmek amaçlanmaktaydı. Daha sonra tedarik zinciri yönetimi önem kazanmıştır ve farklı açılardan geliştirilmiştir.

Geleneksel olarak tedarik zinciri bir kurumu ve bu kurumun birçok tesisini, tedarikçilerini ve dağıtım merkezlerini içermekteydi. Ancak daha sonraları tedarik zinciri anlayışı gelişerek kurum sınırları ötesine taşmış, dikey yapılanmadan ziyade birçok şirketi içine alan sanal organizasyonlara dönüşmüştür.

4.4.2. Tedarik zinciri yönetiminin amaçları

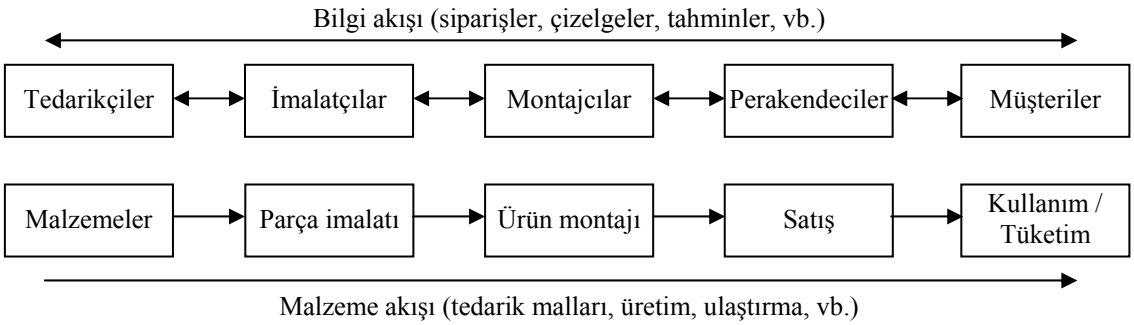
Tedarik zinciri yönetiminin genel amacı doğru miktardaki doğru ürünün doğru yere, doğru zamanda minimum maliyet ile ulaştırılmasıdır. Bu amaç, esneklik ve ulaştırma güvenirliği ile birlikte, ulaştırma ve tedarik etme sürelerinin ve envanter seviyelerinin azaltılmasını gerektirmekte ve belki de zorunlu kılmaktadır. Belirtilen bu amaçlar Şeklil 4.6'da tedarik zincirinin ilgilendiği alanlar olarak özetlenmektedir. Şekilde alt sırada ortadaki iki kutuda bulunan, ulaştırma güvenirliği ve ulaştırma süresi müşteri hizmetleri ile ilgili yönlerdir, ancak ilk kutuda olan esnekliğe ve son kutuda olan envantere oldukça bağlıdır (Cutting-Decelle vd., 2006).



Şekil 4.6. Tedarik zincirinin amaçları ve hiyerarşisi

Tedarik zinciri kavramı, bilgi ve malzeme akışlarını, tesis operasyonlarını ve lojistiği yönetmeyi içermektedir. Bunu yaparken bir dizi ortak prensipler, stratejiler, politikalar ve performans ölçümleri uygulanmaktadır. Müşteri taleplerindeki değişimleri karşılayabilmek için tedarik zinciri esnek ve çevik olmalı, minimum maliyetlerle kaynak kullanımını gerçekleştirebilmelidir. Bu bakımdan tedarik zincirindeki bağımsız her üyenin senkronizasyonu sağlanabilmelidir. Bu ise, tedarik zinciri üyeleri arasındaki koordinasyon ve bilgi alışverişine bağlıdır.

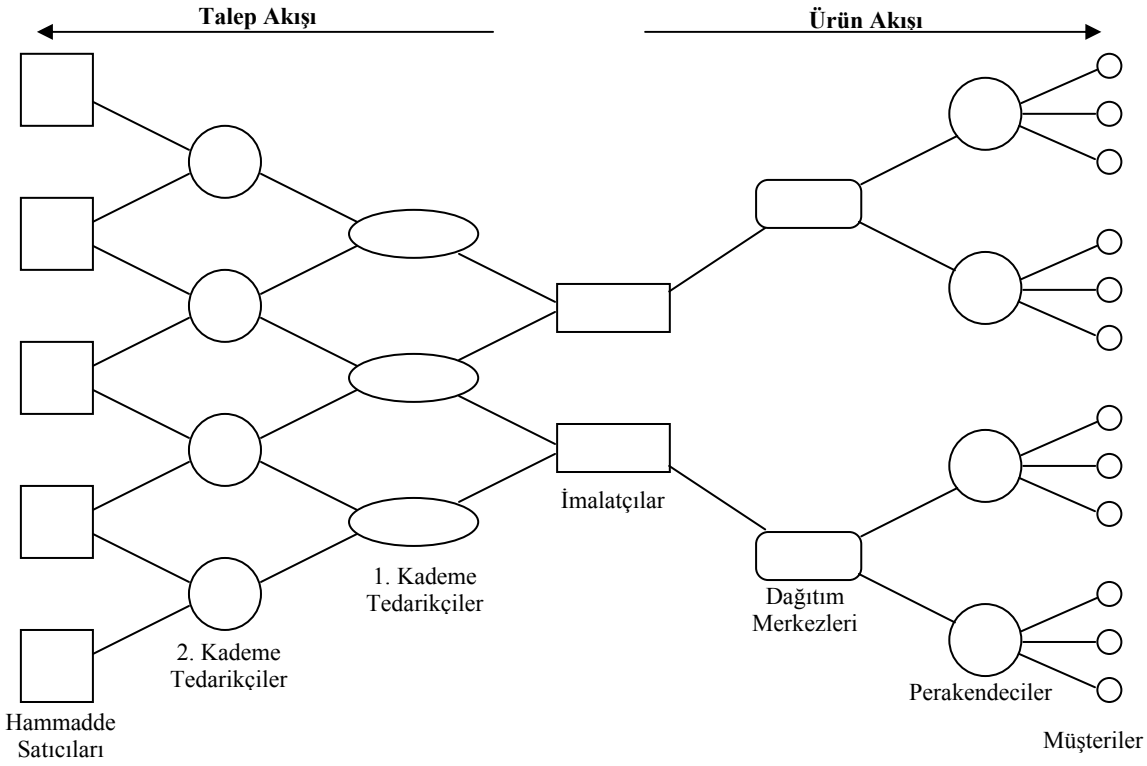
Tüm malzemelerin ve ürünlerin akışını kontrol etmek için tedarik zincirinde bilgi akışı hayati öneme sahiptir. Tedarik zincirinin genel bir yapısı Şekil 4.7'de verilmiştir. Bilgi akışı aslında çift yönlü olmakla birlikte daha çok müşterilerden tedarikçilere doğru iletilmekteyken malzeme akışı ise tedarikçilerden tüketicilere doğru olmaktadır (Vrijhoef ve Kosleka, 1999).



Şekil 4.7. İmalatta tedarik zincirinin genel yapısı

4.4.3. Tedarik zinciri ağının yapısı

Bir tedarik zinciri, sistemler, alt-sistemler, operasyonlar, faaliyetler ve bunların birbiri ile ilişkisinden oluşan karmaşık bir ağa sahip olabilmektedir. Bu ağın içerisinde tedarikçiler, taşıyıcılar, imalat tesisleri, dağıtım merkezleri, perakendeciler ve tüketiciler bulunmaktadır. Tedarik zincirini bir ağ olarak ifadesi Şekil 4.8'de görülmektedir (Chandra ve Kumar, 2001).



Şekil 4.8. Tedarik zinciri ağı

Böylesi bir sistemin tasarlanması, modellenmesi ve uygulanması oldukça güçtür. Bunu gerçekleştirebilmek için sistem elemanlarının birbiriyle çok iyi ilişkilendirilmesi, bilgi akışının net olarak sağlanması gerekmektedir. Şekil 3'te görülen talep akışı ve ürün akışı yönlerinin her ikisi için de bilgi akışı çok önemlidir. Aksi takdirde sistemin bir ağ oluşturması, bütünleştirilmesi ve senkronizasyonu mümkün olmayacaktır. Bu sebeple tedarik zincirinin modellenmesinde ve sistem elemanları arasında etkin bilgi etkileşimi bakımından HLA önerilmiştir.

Tedarik zincirinde her alt bölge üst bölgenin tedarikçisi durumundadır. Her bölge, malzeme temini ve üretim olmak üzere iki tür temel fonksiyonu yerine getirmektedir. Alt bölgeden malzeme temin edilmekte, bunlar depolanmakta ve üretimde kullanılmaktadır. Üretim işlemi, fabrikasyon (imalat) veya montaj işlemidir. Girdi malzemeleri çıktı malzemesi olacak şekilde imal edilmekte veya montajı yapılmaktadır. Tedarik zincirinin her bölgesi iki tür stoka sahiptir. Bunlar girdi stokları ve çıktı stoklarıdır.

4.4.4. Tedarik zincirinin HLA ile bütünleştirilmesi ihtiyacı

Tedarik zinciri oluşturan her bir halka kendi çapında bağımsız işlemekle beraber sonuçları diğer halkanın işleyişini de etkilemektedir. Dolayısıyla halkalar arasında dinamik bir bilgi akışı ve etkileşimi gereklidir. Bir sistemin çıktılarının başka bir sistemin girdileri olması ve buna göre çalışması, bundan etkilenerek faklı sonuçlar üretilmesi, neticede bu yeni sonuçların da başka sistemi etkilemesi söz konusudur. Bu etkileşim zincirleme olarak zincirin tüm halkalarında devam etmektedir.

HLA'nın karşılıklı işleyebilirlik ve yeniden kullanılabilirlik özellikleri sayesinde bu etkileşimin gerçekleştirilmesi mümkündür. HLA'nın geliştirilmesine sebep olan karşılıklı işleyebilirlik özelliği, bir sistemin sonucunun diğer sistemin girdisi olmasını, onun çalışmasını ve üreteceği çıktıları etkilemesini ifade etmektedir.

Tedarik zincirinin bir ucunda bulunan tedarikçiler ile diğer ucunda bulunan müşteriler arasındaki her kademede, sistemler birbirini etkilemektedir. Nihai üretim ve montajın yapıldığı ana üretim ile tedarikçilerinin HLA ile iletişim gerçekleştirmesi tedarikçilerin ve ana üreticinin işleyişini değiştirecektir.

HLA ile tedarik zincirinin bütünleştirilmesi benzetim imkanlarını artıracak ve benzetimin avantajları kullanılabilecektir. Tedarik zinciri içerisinde bulunan herhangi bir katılımcı federenin durumundaki değişikliklerin diğer federeleri nasıl etkileyebileceğini önceden kestirmek ve buna göre önlem almak mümkün olabilecektir.

Tedarik zincirinin HLA ile bütünleştirilmesinin diğer faydaları şu şekilde belirtilebilir:

- Tedarik zincirindeki tüm sistemlerin birbiriyle haberleşmesi temin edilmiş olur,
- Benzetim yazılımlarını ve benzetim dışı yazılımları bünyesinde barındırabilir,
- Farklı yerlerde farklı yazılımların kullanımına imkan verir,
- Yazılımların bağımsız olarak çalışması ve birbiriyle haberleşmesine imkan tanır,

- Çeşitli seviyelerde benzetimlerin gerçekleştirilmesine imkan verir,
- Farklı benzetim gereksinimleri ile birçok benzetimin yapılmasına imkan sağlar.

Tedarik zinciri üyeleri arasında veri iletimi gerçekleştirmek suretiyle her üyenin işleyişinin diğer üyeler ile uyumlu hale getirilmesi amaçlanmaktadır. İletilen veriler ürün, planlama, sipariş, stok gibi verileri içermektedir. Bu veriler, bir kurumun işleyişini düzenlediği gibi diğer kurumlar ile müzakere / etkileşim sağlanmak amacına da hizmet edecektir. Tüm bunların neticesinde, müşteriye daha kaliteli ürün ve hizmet sağlanmış olacaktır.

Birçok araştırmacı benzetim tabanlı gerçek zamanlı çizelgeleme sistemleri üzerinde durmakta bu yolla sistemin durumunu gözlemlemeyi ve karar verme işlemini gerçek zamanda yapabilmek istemektedirler. Bunu gerçekleştirmek sistemin şunlara sahip olması gereklidir (Cutting-Decelle vd., 2006):

- Mevcut veritabanlarına bağlanıp bilgi alabilmek için arayüzler,
- Çok kısa bir sürede, neredeyse gerçek zamanda benzetimi çalıştırabilecek yazılım ve donanım,
- Yeni görevler atayabilmek, sistemin performansıyla ve durumuyla ilgili geribesleme alabilmek için kontrol sistemiyle arayüzler.

Benzetimler, sistemi uygulamadan önce işleyiş performansını incelemeye imkan vermektedirler. Karar vermek için güçlü "eğer-ne" analizleri yapmamızı sağlamaktadırlar. Çoğu benzetim aracı, planlamacı kişilere destek olmak bakımından karar verme arayüzlerine sahiptirler. Benzetimler karar vericilere bilgi sağlayarak doğru karar vermelerine destek olmaktadırlar. Ancak karar verici, tedarik zincirinde daha iyi performans elde edebilmek için benzetimden elde ettiği bilgiyi yorumlama ve düzenleme yeteneğine sahip olmalıdır.

Cutting-Decelle vd. (2006) tedarik zinciri için çeşitli benzetim araçlarının geliştirildiğini haber vermektedir. ProModel gibi benzetim paketleri de belirli seviyelerde tedarik zincirinin modellenmesinde kullanılabilmektedir. Bu benzetimin tedarik zincirlerinde kullanılabildiğini ve önemini göstermektedir. Tedarik

zincirlerinde benzetimin kullanıldığı en önemli alan, çeşitli açılardan tedarik zincirinde "eğer-ne" analizleri yapmaktır. Eğer tedarik zinciri bir tek kurumu ilgilendiriyorsa bu durumda tedarik zincirinin ayrıntılı modellenmesi sorun teşkil etmemektedir. Ancak modelleme kurum sınırlarının ötesine gittiğinde, birçok kurum bilgilerini paylaşmak istememektedir. Bu ise, benzetimin tedarik zincirinde kullanımını engellemektedir (Gan vd., 2000).

Dağıtık benzetimin kullanımı her katılımcı benzetimin özel bilgilerini gizlemesine imkan vermektedir. Birçok imalat sisteminin çeşitli derecelerde benzetimi elverişli gelmektedir. Daha önce modellenmiş bir benzetim küçük değişikliklerle yeniden kullanılabilmekte, farklı bilgisayarların güçlerinden ve farklı benzetim yazılımlarının özelliklerinden faydalanma imkanı dağıtık benzetim ile ortaya çıkmaktadır.

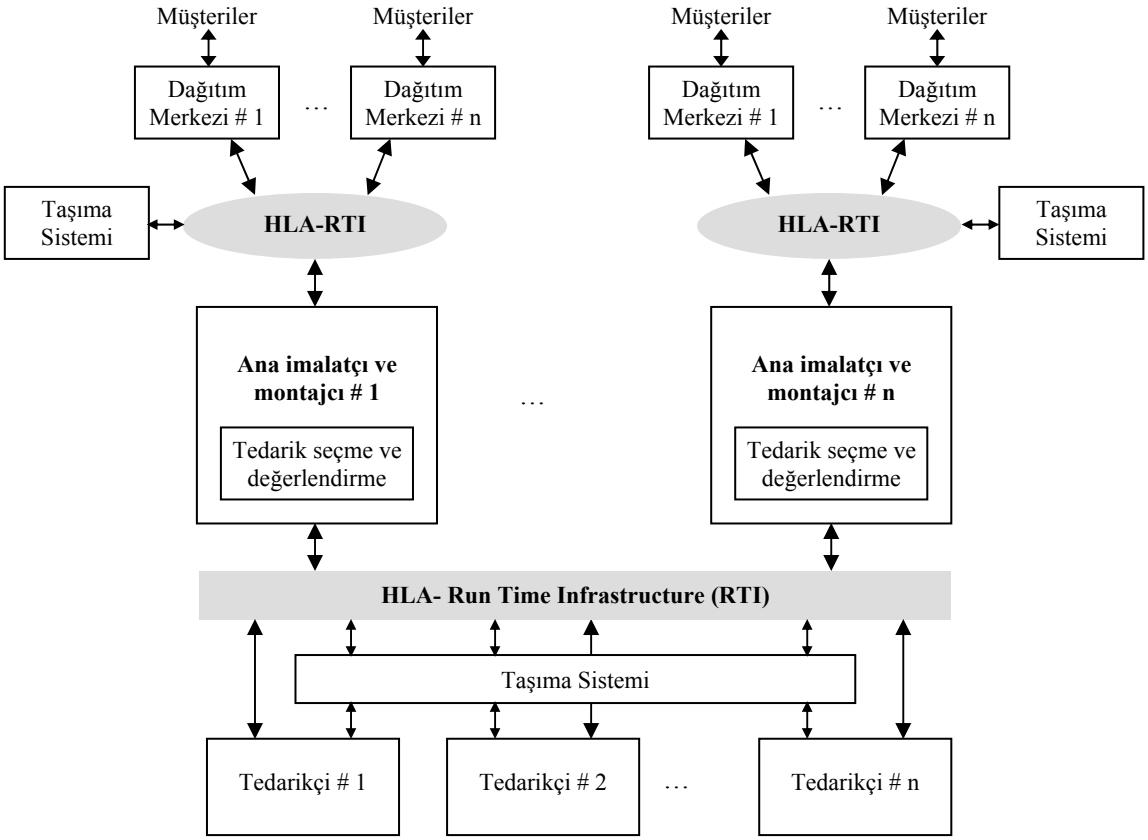
HLA tüm benzetim üyelerinin bağımsız olarak çalışmalarına imkan verirken gerçek zamanlı olarak birbirleriyle etkileşmelerini ve karşılıklı işlemelerini sağlamakta ve böylece tedarik zincirinin amacının gerçekleşmesine destek olmaktadır. Tedarik zincirine katılan üyelerin gönderdikleri bilgileri sadece ilgili diğer üyeler almakta ve sadece gerekli bilgiler gerektiği zamanda gönderilmektedir. Bilgiler iletilirken tümüyle değil, sadece değişen ve güncellenen kısmıyla gönderilmekte, sistemin yükü gereksiz yere artırılmamaktadır.

Dağıtık benzetim ile sistem katılımcı modellere (şirketlere, tedarikçilere vb.) ayrılabilmekte ve her bir model kendi başına işleyebilmektedir. Her katılımcı model, diğer modellerin detaylarıyla ilgilenmeden çalışmasını sürdürebilmekte, gerektiği zamanda gerektiği bilgiyi diğer modellerden edinebilmektedir. HLA ile sistemin üyesi olan şirketlerin farklı işletim sistemleri, platformlar ve programlama dilleri kullanması veri iletişimini engellememektedir.

4.4.5. HLA temelli tedarik zinciri modeli

HLA terminolojisine göre tedarik zincirinin tümüne federasyon, içerisindeki her bir modele (tedarikçi, imalatçı, dağıtım merkezi, vb.) federe denmektedir. Her bir sistem kendi amacını gerçekleştirirken karşılıklı işleyerek tüm tedarik zincirinin amacını da

yerine getirmeye çalışmaktadır. Örnek bir HLA temelli tedarik zinciri modeli Şekil 4.9'da verilmiştir.



Şekil 4.9. HLA temelli tedarik zinciri modeli

Modele göre dağıtım merkezleri müşterilerden talepleri almakta ve diğer çeşitli tahminler ile toplam talebi belirleyerek ana imalatçı ve montajcısına bildirmektedir. Her ana imalatçının kendi dağıtım merkezine ve dağıtım sistemine sahip olduğu varsayılmıştır. İmalatçılar dağıtım merkezlerinden elde ettikleri talep bilgilerini işleyerek gerekli imalat ve satınalma bilgilerini oluşturmakta, çizelgelemeler ve planlamalar yapmaktadır. Bu işlem genellikle malzeme ihtiyaçları planlaması veya kurum kaynakları planlaması yazılımları kullanılarak yapılmaktadır. Dolayısıyla bu gibi yazılımlardan elde edilecek bilgiler uyarlama arayüzleri ile HLA formatına dönüştürülmelidir.

İmalat fonksiyonlarının her birinin HLA temelli modellenmesi ve tedarik zincirin içerisinde alt-federasyon (tedarik zinciri federasyonunun bir federesi) olarak çalışması da mümkündür. Böylece her bir imalat federesinin tedarik zinciri federasyonu ile bütünleştirilmesi kolaylaşacaktır. Bunun için imalat alt fonksiyonları ayrı ayrı federeler olarak tasarlanmalı ve aralarında bilgi iletimini RTI ile gerçekleştirmelidir.

İmal edilen nihai ürünler taşıma sistemi ile dağıtım merkezlerine taşınmaktadır. Taşıma sisteminin de HLA-RTI altyapısını kullanması ve ana imalatçı ile bu yolla iletişim kurması mümkündür. Doğru zamanda doğru taşıma aracının temin edilmesi, doğru ürünlerin yüklenmesi ve doğru dağıtım merkezine iletilmesi bu şekilde desteklenmiş olacaktır.

Diğer taraftan sistemin tedarikçiler katmanı da mevcuttur. Bu katmandaki tedarikçileri ana imalatçıya bağlayan HLA-RTI ile dağıtım merkezlerini ana imalatçıya bağlayan HLA-RTI ayrı düşünülebilmektedir. Çünkü dağıtım merkezleri ile tedarikçiler arasında doğrudan bağlantı, bilgi alışverişi ve etkileşim söz konusu değildir.

Ana imalatçı toplam talepten ürettiği satınalma bilgilerine göre tedarikçilerle etkileşime geçecek, doğru hammadde ve yarı mamulün, doğru zamanda ve doğru miktarda, doğru tedarikçiden temin edilmesine çalışacaktır. Aynı hammaddeyi sağlayan veya benzer kalitede yarı mamul üreten birçok tedarikçi olabileceğinden, ana imalatçının belirli değerlendirme kriterleri altında tedarikçi seçme ve değerlendirme sisteminin olması gerekmektedir. Buradan çıkacak sonuca göre bilgi doğru tedarikçiye ulaştırılacaktır. Gerekli bilgi iletişimi taşıma sistemi ile de gerçekleştirilecek ve malzemelerin taşınması sağlanacaktır.

Aynı tedarikçinin birden fazla ana imalatçıya tedarik hizmeti vermesi olağandır. Bu sebeple ana imalatçılar çeşitli tedarikçilerle ve tedarikçiler çeşitli ana imalatçılarla bilgi alışverişi yapmakta ve etkileşim halindedir. HLA nesne model şablonunu (OMT) ve RTI servislerini kullanarak bilginin yayımlanması ve abone olunması ile bilginin farklı imalatçılar ve tedarikçiler arasında iletimi sağlanabilmektedir.

Tedarik zincirindeki her bir federenin dinamik etkileşimi için iletişim kanallarında HL-RTI kullanılmalıdır. İletilecek bilgilerin gösterimi ise HLA-OMT ile gerçekleştirilecektir. Dağıtık imalat benzetiminde bilgi iletişimi ve gösterimi için Uygun vd. (2006a) çalışmasından faydalanmak mümkündür.

BÖLÜM 5. HLA TEMELLİ DAĞITIK İMALAT BENZETİMİ: ÖRNEK UYGULAMA

5.1. Giriş

Bu bölümde örnek bir senaryo üzerinden HLA temelli bir dağıtık imalat benzetimi geliştirilmiştir. Örnek uygulama, DMSO'nun RTI1.3NG-V6 versiyonu çalışma anı alt yapısı üzerinden geliştirilecektir. Dağıtık imalat uygulamasını geliştirmeden önce, bilgisayara RTI'ın kurulması gerekmektedir. Daha sonra senaryoda gerekli nesnelerin HLA-OMT yapısında tanımlanması gerçekleştirilmiştir. Ardından senaryonun benzetim algoritmasının kodları yazılmıştır.

Aşağıda, HLA temelli dağıtık imalat benzetimi geliştirilmesinde gerçekleştirilen işlemler anlatılmıştır.

5.2. RTI'ın Kurulması

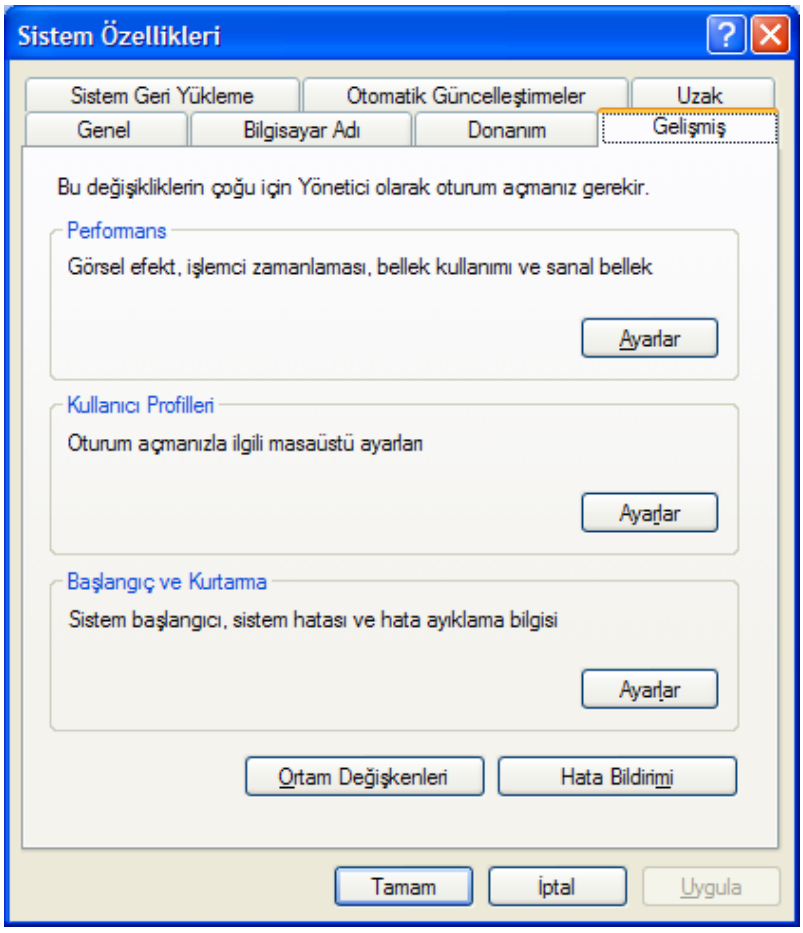
RTI1.3NG-V6 versiyonunun Windows XP SP2 işletim sistemine yüklenmesi için MS Visual C++ 6.0 SP5 derleyicinin veya daha üst sürümünün bulunması gerekmektedir.

İşletim sistemi Windows XP olan bir bilgisayara RTI-1.3 NG V6 kurmak için RTI yazılımın kurulum programı çalıştırılır. Varsayılan olarak program, "Program Files/DMSO" altına yüklenecektir. İstenirse kurulum esnasında programın kurulacağı klasör değiştirilebilmektedir. Ardından, "Ortam Değişkenleri" ayarları gerekmektedir. En sonunda ise RTI'ın düzgün çalışıp çalışmadığı test edilmelidir. Bunun için *helloWorld* programı kullanılmaktadır.

5.2.1. Ortam değişkenleri ayarları

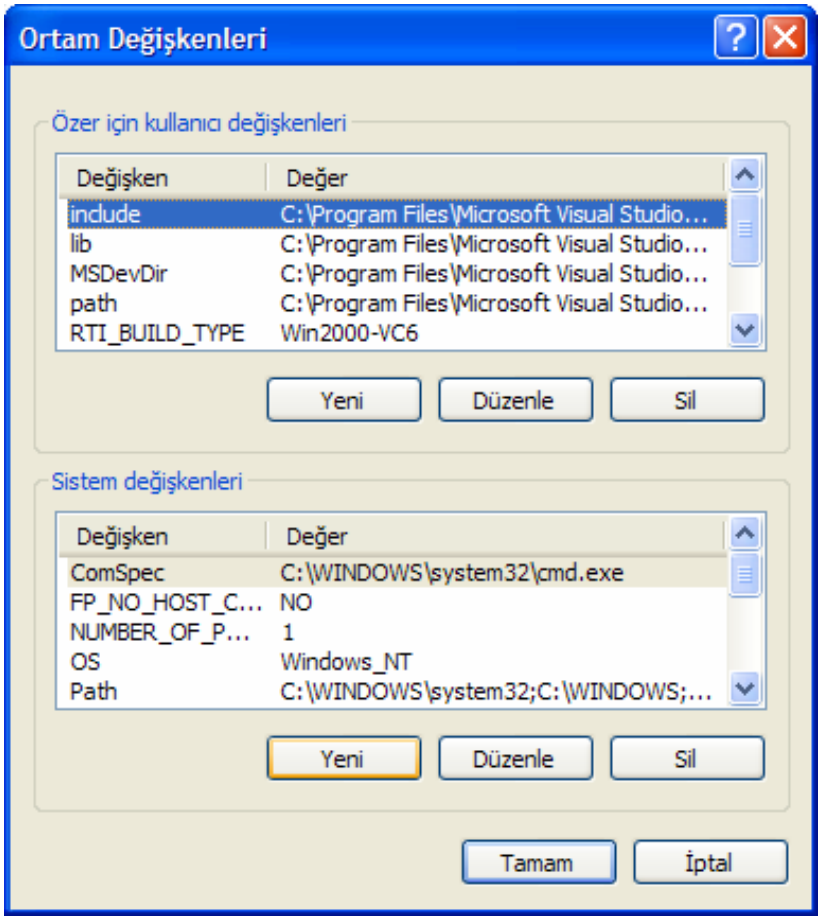
Programı kurduktan sonra çeşitli ortam değişkenleri ayarları gerekmektedir. Ortam değişkenleri ayarlarına erişmek için şu yöntem izlenebilir:

Masaüstünde "Bilgisayarım" simgesi sağ tıklanıp "Özellikler" seçilir. Açılan pencerede "Gelişmiş" sekmesine geçilir (Şekil 5.1).



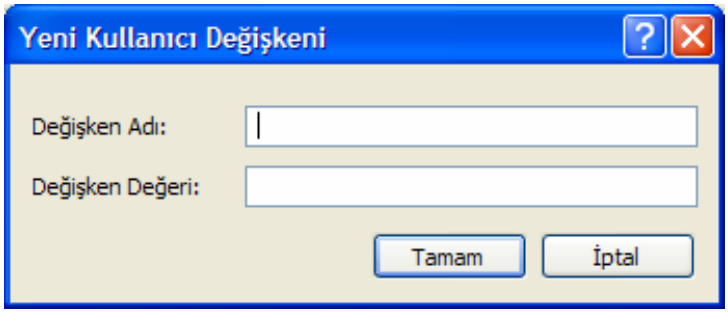
Şekil 5.1. Ortam değişkenlerine erişim

Gelişmiş sekmesindeki "Ortam Değişkenleri" tıklanır ve Şekil 5.2'de görünen pencere açılır.



Şekil 5.2. Ortam değişkenlerinin tanımlanması

Yeni bir ortam değişkeni eklemek için kullanıcı değişkenleri kısmındaki (Şekil 5.2'de görülen pencerede "Özer için kullanıcı değişkenleri" kısmında) "Yeni" düğmesine basılır ve Şekil 5.3'de verilen pencere görüntülenir.



Şekil 5.3. Yeni ortam değişkeni tanımlama

Çıkan pencerede Değişken Adı ve Değişken Değeri kısımları uygun bir şekilde doldurularak "Tamam" düğmesine basılır.

RTI-1.3 NG V6 için gerekli ortam değişkenleri **Tablo 5.1**'de verilmektedir.

Tablo 5.1. RTI-1.3 NG V6 için gerekli ortam değişkenleri

Değişken Adı	**Değişken Değeri**
RTI_HOME	RTI kurulum klasörü seçilir (Ör. C:\Program Files\DMSO\RTI1.3NG-V6)
RTI_BUILD_TYPE	Yazılımın hangi işletim sistemi için geliştirildiği seçilir (Ör. Win2000-VC6)
PATH	Windows dll dosyaları için gereklidir ve RTI NG 1.3 yazılımını kullanan her federe için ayarlanmalıdır. %RTI_HOME%/%RTI_BUILD_TYPE%/bin şeklinde değişken değeri atanır. (Ör. C:\Program Files\DMSO\RTI1.3NG-V6\Win2000-VC6\bin)

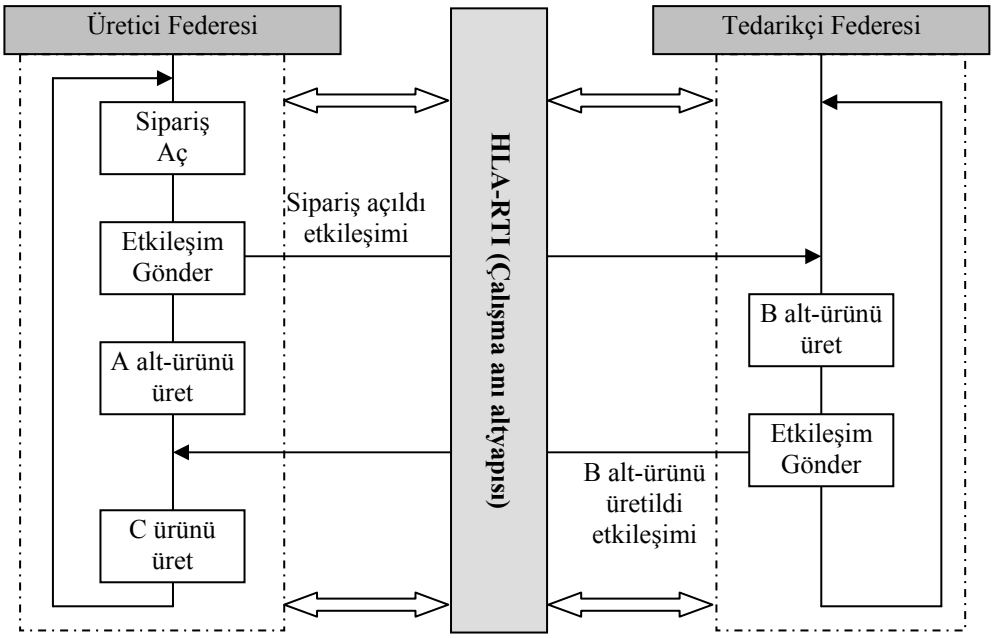
Ortam değişkenlerinde PATH değişkeni muhtemelen daha önceden var olacaktır. Bu durumda PATH değişkeni seçilip "Düzenle" düğmesi tıklanır ve açılan pencerenin "Değişken Değeri" kısmında en sona gidilerek noktalı virgül (;) konur. Daha sonra RTI için yukarıda verilen PATH değişkeni için değer buraya ilave edilir.

Yapılan bu işlemlerden sonra RTI kullanılabilir halde olacaktır. RTI'ın düzgün kurulduğunun ve ortam ayarlarının doğru yapıldığının test edilmesi için, RTI1.3NG-V6 yazılımıyla birlikte kurulan "helloWorld" uygulaması çalıştırılmalıdır. Bunun için Toft (1999)'dan faydalanılabilir.

5.3. Senaryo

Dağıtık imalata uygun bir örnek, uygulamalı olarak ele alınmış ve her aşaması aşağıda açıklanmıştır. Uygulaması yapılan federasyon modeli ana hatları ile Şekil 5.4'de verilmiştir.

Buna göre ana imalatçı C ürününü üretmek için A ve B alt-ürünlerine ihtiyaç duymaktadır. A ürününü kendisi üretirken B ürününü ise tedarik etmektedir. Ana imalatçı birinci federe (üretici federesi), tedarikçi ise ikinci federe (tedarikci federesi) olarak modellenmiştir. Ana imalatçı C ürününü üretmek için önce sipariş açmaktadır. Sipariş açıldığında, bununla ilgili bilgi tedarikçiye de etkileşim gönderilerek haber verilmektedir. Siparişin miktarı ve sipariş zamanı da tedarikçiye gönderilmektedir.



Şekil 5.4. Örnek federasyon uygulamasının modeli

Ana imalatçı A'yı üretmeye başladığında etkileşimi alan tedarikçi de B'yi üretmeye başlamaktadır. A'nın üretim zamanı ile B'nin üretim zamanları birbirinden farklı olmaktadır. Hangisinin daha önce biteceği benzetim esnasında, miktara ve üretim süresine bağlı olarak belirlenmektedir. C'nin üretilmesi için her ikisinin de bitmesi beklenmektedir.

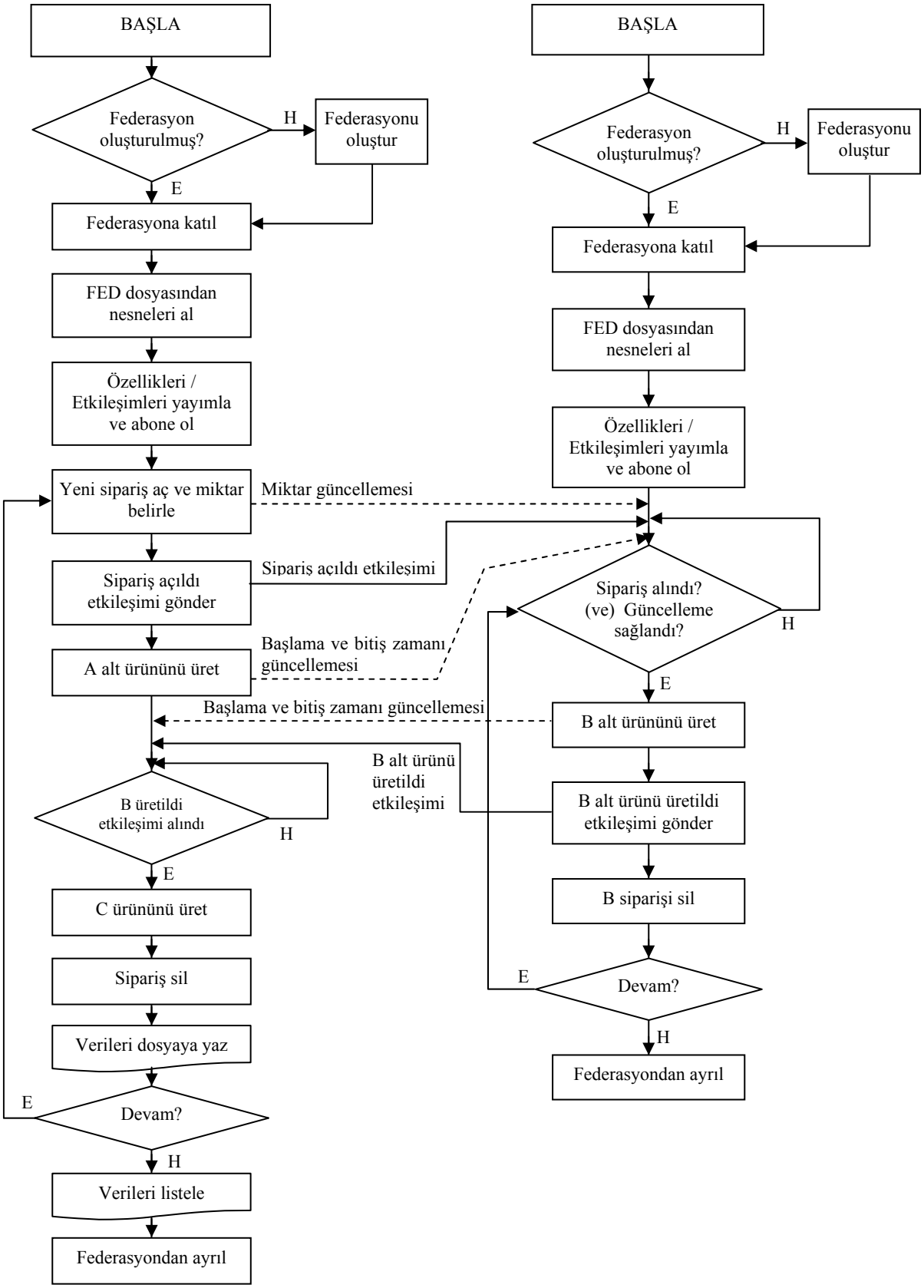
B alt-ürününün bittiğini ana imalatçı, tedarikçinin göndereceği etkileşim ile haber almaktadır. B alt ürününün bitiş zamanı da ana imalatçıya gönderilmektedir. Ana imalatçı bu bilgiden sonra C ürününü üretebilmektedir. C ürününün başlama zamanı, A ve B alt ürünlerinin bitiş zamanlarının karşılaştırılması ile belirlenmektedir. C

ürününün bitiş zamanı ise, miktar ve üretim süresine bağlı olarak hesaplanmakta ve bu, sonraki siparişin açılma zamanının temeli olmaktadır. Burada benzetimin zaman artırımı olay temellidir. Sipariş açmak, A alt-ürününün bitişi, B alt ürününün bitişi ve C ürününün bitişi benzetimin olaylarındandır. Bu olaylar benzetimin zamanını belirleyen temel faaliyetlerdir.

Açıklanan modeli uygulayabilmek için iki adet benzetim oluşturulmalı ve bunlar bir federasyon içerisinde bir araya getirilmelidir. Federasyon çalıştırılırken, RTI'a kayıt olma gibi işlemlerden dolayı federasyonun ve federelerin özgün birer isimleri olmalıdır. Burada federasyon ismi olarak "Uretim", federe isimleri olarak da "Uretici" ve "Tedarikci" seçilmiştir. Federasyon "Uretim" ismi ile RTI'a, federeler ise "Uretici" ve "Tedarikci" isimleri ile federasyonun çalışma oturumuna kaydolmaktadır.

Federasyonun çalıştırılması için bilgisayarda RTI'ın kurulmuş olması gerekmektedir ve bu bir önceki kısımda anlatılmıştır. Federasyonun tasarlanması için ise öncelikli olarak yapılması gereken, nesne modellerinin (FOM ve SOM) tanımlanmasıdır. Her bir benzetimin benzetim nesne modeli (SOM), federasyonun ise federasyon nesne modeli (FOM) oluşturulmalıdır. Ardından C++ ile federe programının geliştirilmesi için gerekli düzenlemeler ve kod yazımı gerçekleştirilmiştir.

Üretici federesi ile tedarikçi federesinden oluşan üretim federasyonunun akış şeması ve işleyişi Şekil 5.5'de verilmektedir. Bu şekilde federeler arası karşılıklı işlerlik de gösterilmektedir. Karşılıklı işlerlik nesne özellik değerinin güncellenmesi veya etkişim gönderimi ile gerçekleşmektedir. Etkileşim gönderimi sürekli çizgi, nesne özelliklerinin değerinin güncellenmesi ile gönderilen bilgiler ise kesikli çizgi ile ifade edilmiştir.



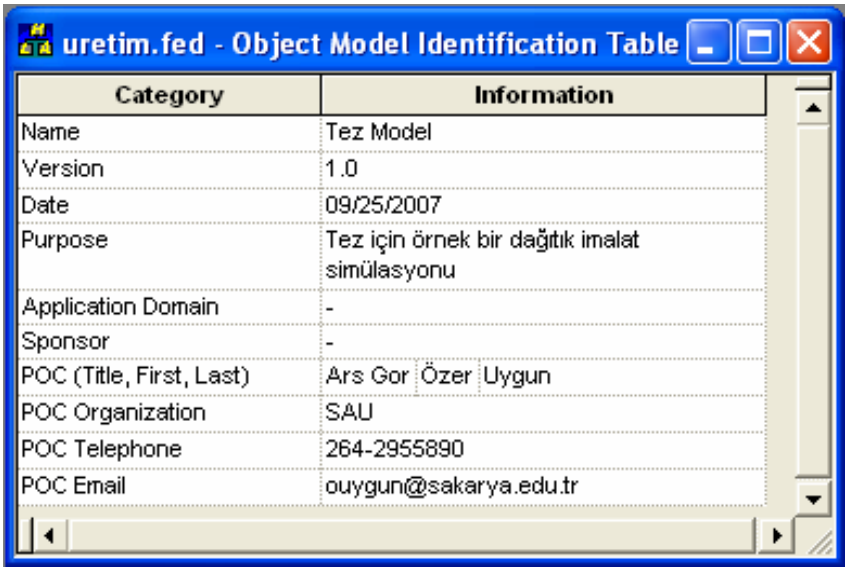
Şekil 5.5. Üretim federasyonu içerisindeki federelerin karşılıklı etkileşimi

Uygulaması yapılan federasyon senaryosu ve federeler arası etkileşim ifade edildikten sonra bu işlemleri gerçekleştirmekte kullanılan nesnelerin tasarlanması işlemine geçilmesi mümkündür.

5.4. Nesne Modelinin Oluşturulması

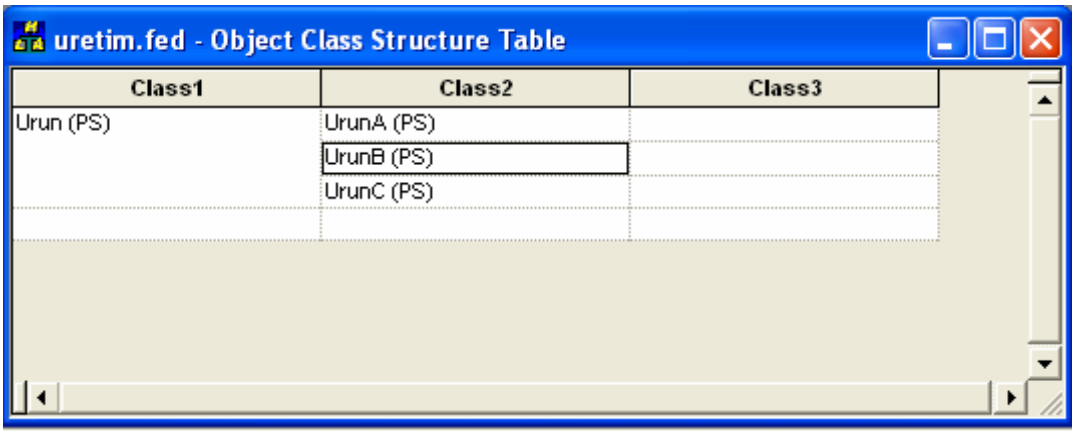
HLA temelli benzetimlerde kullanılması gereken bilgiler HLA'nın gerektirdiği şekilde tanımlanmalıdır. Bu amaçla HLA'nın nesne model şablonu (OMT) yapısına uyulmalıdır. Bu yapıya uygun olarak benzetimde kullanılacak bilgilerin nesne temelli tanımlanması için çeşitli ticari araçlar geliştirilmiştir. Bu uygulama içerisinde gerekli bilgileri tanımlamak için DMSO'nun geliştirdiği, RTI 1.3'ü destekleyen OMDT (Object Model Development Tool) yazılımı kullanılmıştır. Bu yazılım ile nesne sınıfları, bu sınıfların özellikleri, parametre sınıfı gibi tanımlamalar gerçekleştirilmiştir. Bahsedilen yazılımın kurulumu ve kullanım ayrıntılarına girilmeden uygulamaya özgü tanımlamalar açıklanmıştır. Programın kurulumu ve kullanım ayrıntıları için kullanım kılavuzuna bakılabilir (AEgis, 1998).

Yeni bir nesne modeli geliştirilirken öncelikle geliştirilen modelin ismi, hangi amaçla geliştirildiği, kim tarafından geliştirildiği, geliştiricinin iletişim bilgileri gibi tanımların girilmesi yararlı olacaktır. Bu bilgileri girmek, programın işlemi bakımından zorunlu değildir. Ancak bu modeli inceleyen veya uygulayan kullanıcılar bu bilgilerden gerektiği zaman faydalanacak, ihtiyaç halinde geliştirici ile iletişime geçebilecektir. İlgili pencereye Nesne Modeli Tanımlama Tablosu (Object Model Identification Table) denmektedir ve bu pencereye "View" menüsünden ulaşılmaktadır. Geliştirilecek uygulama ile ilgili bu bilgiler Şekil 5.6'da görülmektedir.



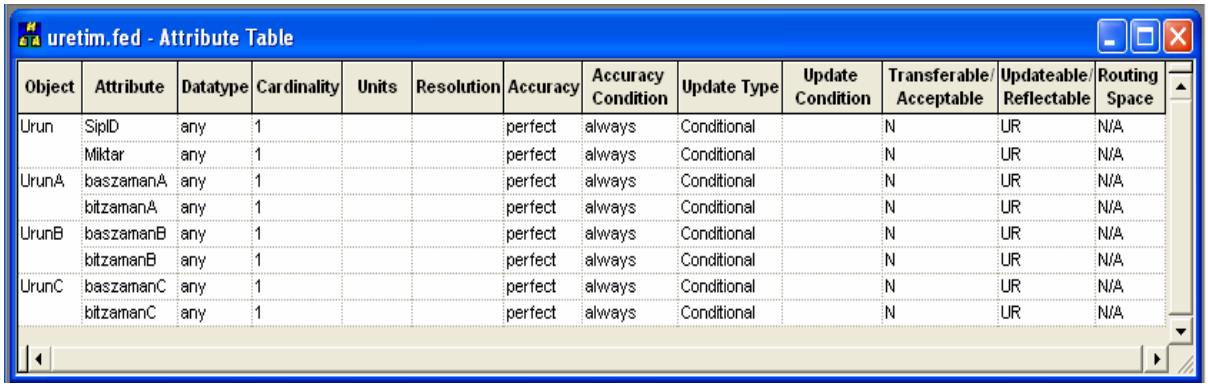
Şekil 5.6. Nesne modeli tanımlama penceresi

Ardından federasyonda kullanılacak nesne sınıfları, "View" menüsünden "Class table" seçilerek tanımlanmalıdır. Örnek uygulamamızda ürün ve alt-ürünler Şekil 5.7'de görüldüğü gibi tanımlanmıştır. Nesneler, hiyerarşik olarak alt nesnelere sahip olabilmektedir. Tanımlamamıza göre "Urun" üst sınıf, "UrunA", "UrunB" ve "UrunC", bunun alt sınıflarıdır. Nesne isimlerinden sonra görülen (PS) ifadesi, nesnelerin hem yayımlanabilir (publish) hem de abone olunabilir (subscribe) olduğunu göstermektedir. Uygulamasını yaptığımız örnekte bilgi alımı ve güncellemesi her iki yönlü olduğu için tanımlamanın böyle olması uygundur.



Şekil 5.7. Nesne sınıf yapısı tanımlama penceresi

Nesne sınıf yapısı tanımlama penceresinde açılan tabloda federasyonda kullanılacak nesneler tanımlandıktan sonra bunların özellikleri de tanımlanmalıdır. Bunun için "View" menüsünden "Attribute table" seçilmelidir. Örnek uygulamamızda ürünün sipariş numarası ve miktarı, "Urun" nesnesinin özellikleri olarak tanımlanmıştır. Alt-ürünlerin ise her birinin başlama ve bitiş zamanları özellik olarak tanımlanmıştır. Örneğin "baszamanA" ve "bitzamanA", "UrunA" alt nesnesinin özellikleridir ve A alt ürününün başlama ve bitiş zamanlarını ifade etmektedir. İlgili pencere Şeklil 5.8'de verilmiştir.



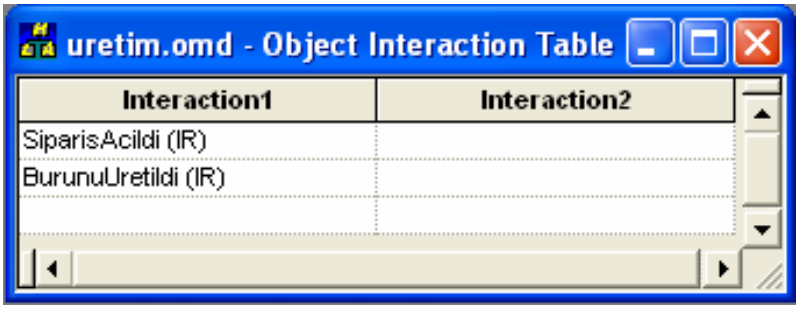
Object	Attribute	Datatype	Cardinality	Units	Resolution	Accuracy	Accuracy Condition	Update Type	Update Condition	Transferable/ Acceptable	Updateable/ Reflectable	Routing Space
Urun	SiplD	any	1			perfect	always	Conditional		N	UR	N/A
	Miktar	any	1			perfect	always	Conditional		N	UR	N/A
UrunA	baszamanA	any	1			perfect	always	Conditional		N	UR	N/A
	bitzamanA	any	1			perfect	always	Conditional		N	UR	N/A
UrunB	baszamanB	any	1			perfect	always	Conditional		N	UR	N/A
	bitzamanB	any	1			perfect	always	Conditional		N	UR	N/A
UrunC	baszamanC	any	1			perfect	always	Conditional		N	UR	N/A
	bitzamanC	any	1			perfect	always	Conditional		N	UR	N/A

Şekil 5.8. Özellik tanımlama penceresi

Pencerede görülen "Datatype" sütunu veri tipini tanımlamak içindir. Açılacak listeden uygun bir HLA veri tipi seçilebilmektedir. Bu uygulamada, veri tipi programlama esnasında tanımlanacağından veri tipleri "any" olarak seçilmiştir. "Cardinality" sütunu, özelliğin boyutunu tanımlamak içindir. Bir dizi veya liste şeklinde verisi olmayan özellikler için 1 seçilmelidir. "Units" sütunu, verinin birimini gösteren açıklama sütunudur (adet, dakika, YTL/saat, gibi). Girilmesi zorunlu değildir. İsteğe bağlı girilen diğer bir sütun ise "Resolution"dır. Özelliğin iki değeri arasında oluşabilecek en küçük farkı ifade etmektedir. Örneğin kesikli değer alan özellikler için kullanılabilir. "Accuracy" sütunu ise benzetim esnasında verinin, verilen değerden maksimum sapmasını yakalamak içindir. Bu kullanılmayacak ise "perfect" seçilir. "Update Type" sütunu, özelliğin verisinin güncellenme şeklini ifade etmek içindir. Yanındaki sütun ise açıklama girmek içindir. "Transferable/Acceptable" sütunu, özelliğin sahipliğinin (kullanım haklarının) bir

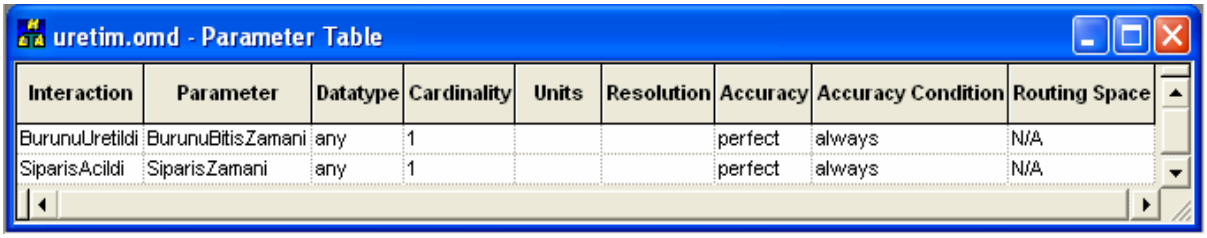
federeden diğerine aktarılabilir olup olmadığı ile tanımlamanın yapıldığı kısımdır. "Updateable/Reflectable" sütunu ise ilgili federenin, özelliğin verisini güncelleme hakkına sahip olup olmadığı, başka federenin ürettiği değeri alıp alamayacağı ile ilgili tanımlamanın yapılmasını sağlamaktadır. Kullanılacak veriler hep güncellenebilir hem de yansıtılabilir olduğundan (U/R) seçilmiştir. "Routing Space" sütunu ise etkileşimlerin belirli filtrelerden geçerek gönderilmesi ile ilgilidir. "Send Interaction with Region" komutu kullanıldığında devreye girmektedir.

Federasyonda iki etkileşim kullanılmaktadır. Bunlardan birincisini ana imalatçı tedarikçiye, diğerini ise tedarikçi ana imalatçıya göndermektedir. Nesne Etkileşimlerin tanımlandığı pencere Şekil 5.9'da verilmiştir.



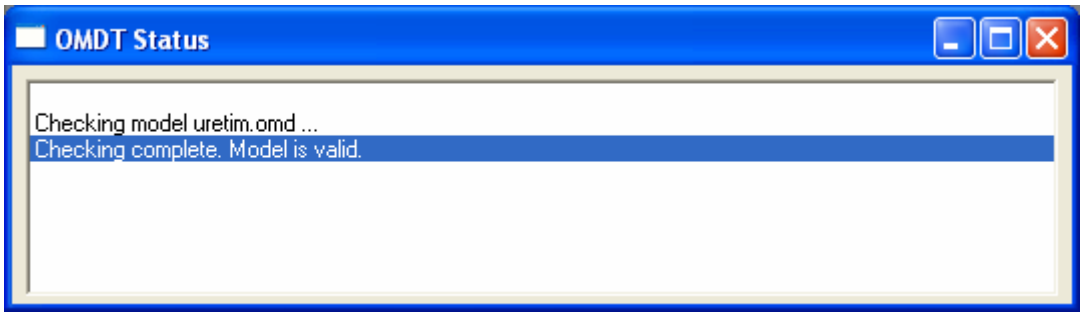
Şekil 5.9. Etkileşim tanımlama penceresi

Nesnelerin özellikleri olduğu gibi, etkileşimlerin de parametreleri olabilmektedir. Örnek federasyonda veriler nesne özelliklerinde tutulmakta, bunların güncellenmesi ve yansıtılması ile iletilmektedir. Dolayısıyla etkileşimler kullanılmıştır ancak parametreler kullanılmamıştır. Etkileşim almak, olayları tetiklemektedir. Tercihen parametreler kullanılacak olsaydı Şekil 5.10'da gösterildiği gibi tanımlanması gerekmekteydi. Buradaki sütunların anlamları, özellik penceresi açıklanırken ifade edilmişti.



Şekil 5.10. Parametre tanımlama penceresi

Gerekli tüm tanımlamalar gerçekleştirildikten sonra nesne modeli kaydedilmelidir. Kayıt işlemi gerçekleştirilirken program, modelin doğruluğunu test etmektedir. Hata ile karşılaşıldığında uyarmaktadır. Eğer modelde hata ile karşılaşılmazsa Şekil 5.11'de görüldüğü gibi modelin geçerli olduğu bildirilmektedir.



Şekil 5.11. Nesne modelinin test sonuç penceresi

Nesne modeli tanımlaması bittikten sonra kaydedilen model .omd uzantısı ile kaydedilmektedir. Federasyonda kullanılacak olan nesne yapısı ise FED dosyası olarak kaydedilmelidir. Tanımlanan nesne modelinin (SOM veya FOM) FED dosyasına dönüştürülmesi için "File" menüsünden "Save as" seçilmeli ve kayıt türü .fed dosyası olacak şekilde değiştirilmelidir.

FED dosyası oluşturulduktan sonra federasyonun çalıştırılmasını sağlayan benzetim yazılımının C++ kodları oluşturulmasına geçilmesi mümkündür. Aşağıda, örnek uygulamanın nasıl geliştirildiği anlatılmıştır.

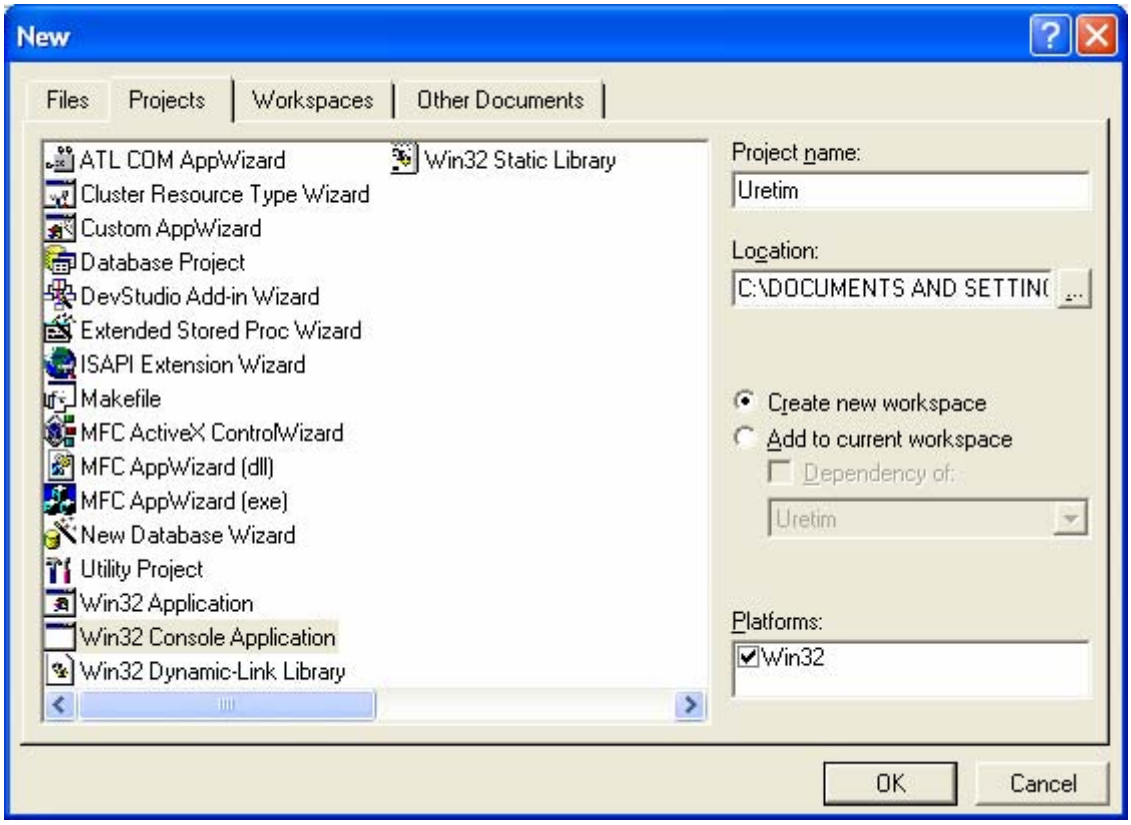
5.5. Federelerin Oluşturulması

Verilen senaryonun Uretici ve Tedarikci federeleri Visual C++ 6.0 ile geliştirilmiştir. Benzetimi (federeyi) geliştirmek için öncelikle Visual C++'da bir projenin başlatılması gerekmektedir. Proje başlatılırken özellikle HLA ile ilgili ve diğer gerekli kütüphanelerin belirtilmesi ve çeşitli düzenlemelerin yapılması, aşağıda anlatılmaktadır.

5.5.1. Visual C++ ile yeni bir federe oluşturmak

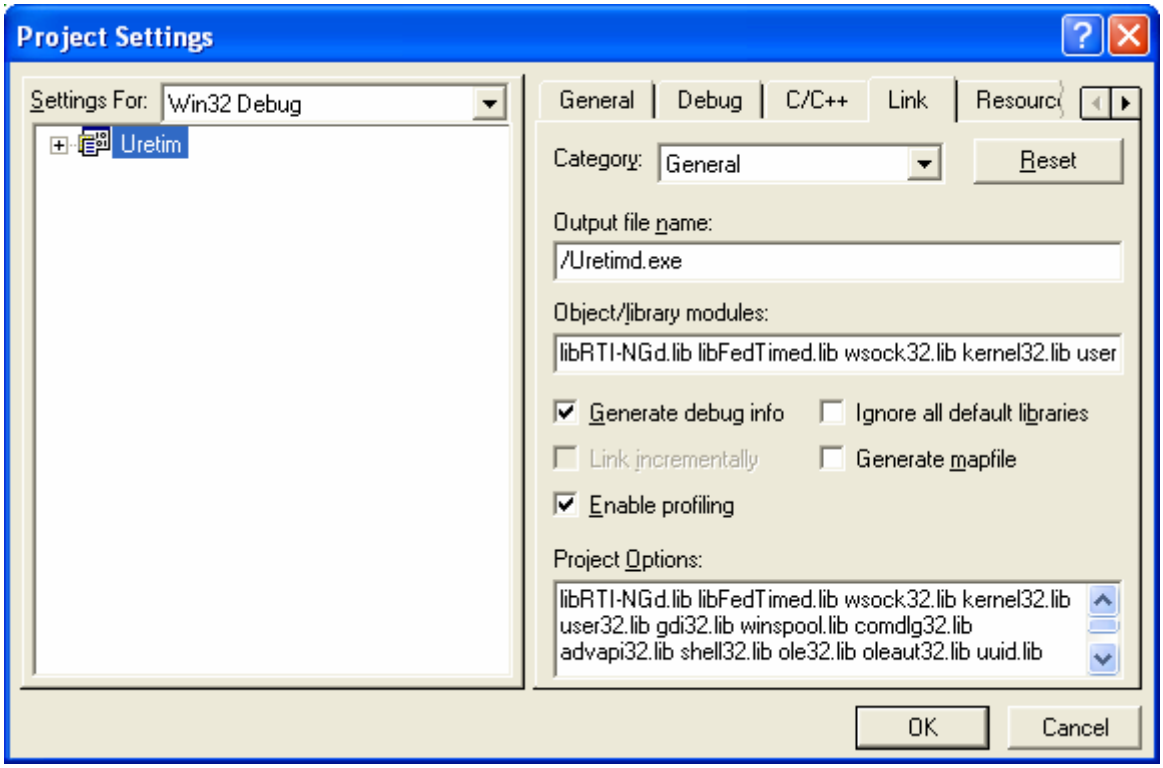
Yeni bir federe oluşturmak için MS Visual Studio C++ çalıştırılarak "File" menüsünden "New" seçilmelidir. Açılan pencerede "Projects" sekmesine geçilir ve "Win32 Console Application" seçilerek "Project name:" kısmına oluşturulacak federeye uygun bir isim yazılmalıdır. Bu uygulamada Uretici federesi için "Uretim" ismi, Tedarikçi federesi için "Tedarikci" ismi kullanılmıştır. Bunun altındaki "Location" kısmında da federenin oluşturulacağı adres seçilmektedir. Bu işlemlerle ilgili pencere Şekil 5.12'de verilmiştir.

Bu işlemin ardından federenin çalışması için hazırlanması gereken dosyalar projeye dâhil edilebilir. Bunun için yine "File" menüsünden "New" seçilir ve oluşturulmak istenen dosyanın türü seçilerek projeye dâhil edilir. Örneğin üretici federesi için uretim.cpp, rti_arayuz.cpp, urun.h gibi kaynak dosyaları ve başlık dosyaları bu şekilde oluşturulacaktır. Var olan bir dosyayı bu projeye eklemek için ise "Project" menüsünden "Add to Project" ve "Files" seçilerek istenen dosya bu projeye eklenebilmektedir. Örneğin "FederateAmbassador.h" dosyaları birbirine çok benzer oldukları için bu yöntem ile projeye eklenerek daha sonra üzerinde gerekli değişiklikler yapılabilmektedir.



Şekil 5.12. Bir federenin C++ projesi olarak başlatılması

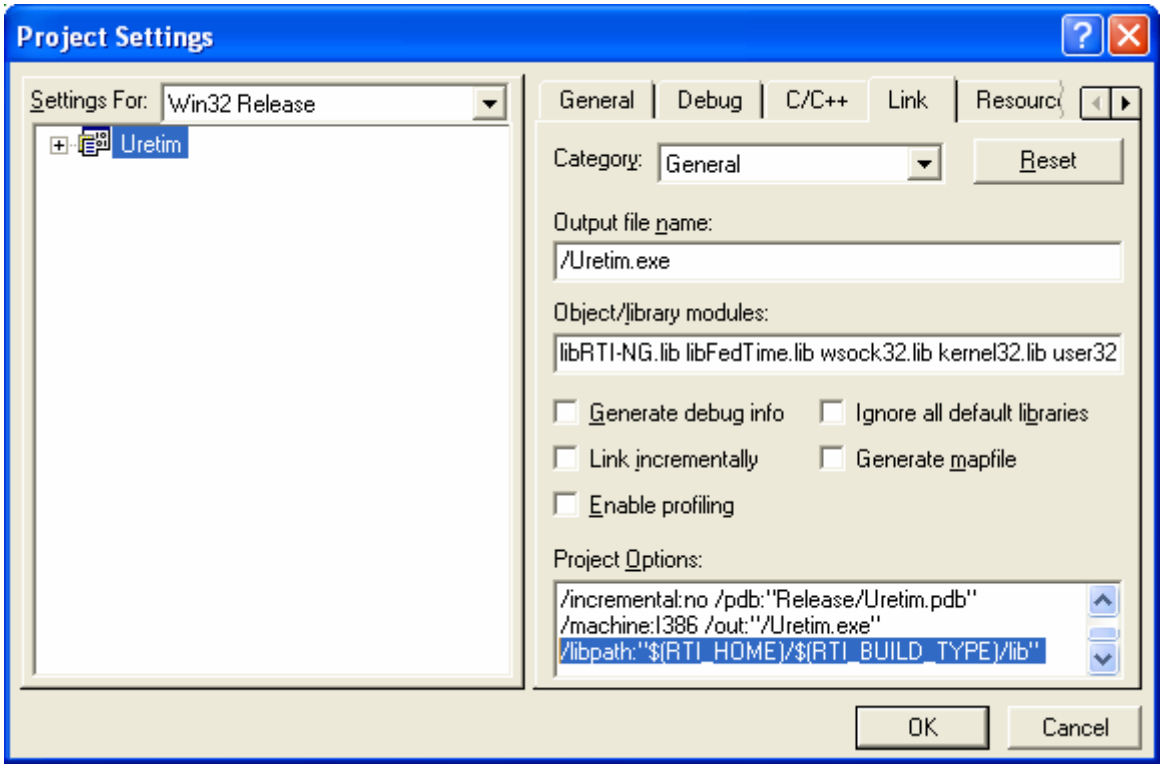
Daha sonra “Projects” menüsünden “Settings” seçilir ve açılan pencerede “Link” sekmesine geçilir. Sol taraftaki “Settings For:” kısmında “Win32 Debug seçilir ve sağ taraftaki “Output file name:” kısmına .exe dosyasının adresi ve ismi tanımlanır. Örneğin üretici federesi için “Uretimd.exe” ismi yazılmıştır. “Object/library modules” kısmına ise aşağıda gösterildiği gibi “libRTI-NGd.lib” ve “libFedTimed.lib” kütüphaneleri eklenmelidir. Eklenen bu kütüphanelerin sonlarında ve .exe dosyasında, debug versiyonunu temsilen ilave bir “d” harfinin bulunduğuna dikkat ediniz (Şekil 5.13). Bu iki kütüphane eklendikten sonra “Project Options:” kısmına otomatik olarak gelmektedir. Ancak Şekil 5.14’de görüldüğü gibi kütüphanelerin adresi (/libpath:"$(RTI_HOME)/$(RTI_BUILD_TYPE)/lib") olarak ayrıca belirtilmelidir.



Şekil 5.13. Win32 Debug için RTI kütüphanelerinin tanımlanması

Ardından sol tarafta "Win32 Release" seçilir. Yukarıdakine benzer şekilde .exe dosyasının ve kütüphanelerin "d"siz olanları ile birlikte, kütüphanelerin adresi tanımlanmalıdır. Bununla ilgili işlemler de Şekil 5.14'de görülmektedir.

Bunun dışında ağdan gelen verinin byte sırasının bilgisayarın anlayacağı hale çevrilmesini ve bilgisayardan ağa gönderilen verinin byte sırasının dönüştürülmesi bakımından, ntohl (network to host long) ve htonl (host to network long) gibi komutların çalışmasını sağlamak için "wsock32.lib" kütüphanesinin eklenmesi gerekmektedir. Bunun için açık olan pencerenin sol tarafındaki "Settings For" ayarını "All Configurations"a getirmek ve sağ tarafta "Object/library modules" kısmındaki kütüphanelere "wsock32.lib" kütüphanesi de ilave etmek gerekmektedir.

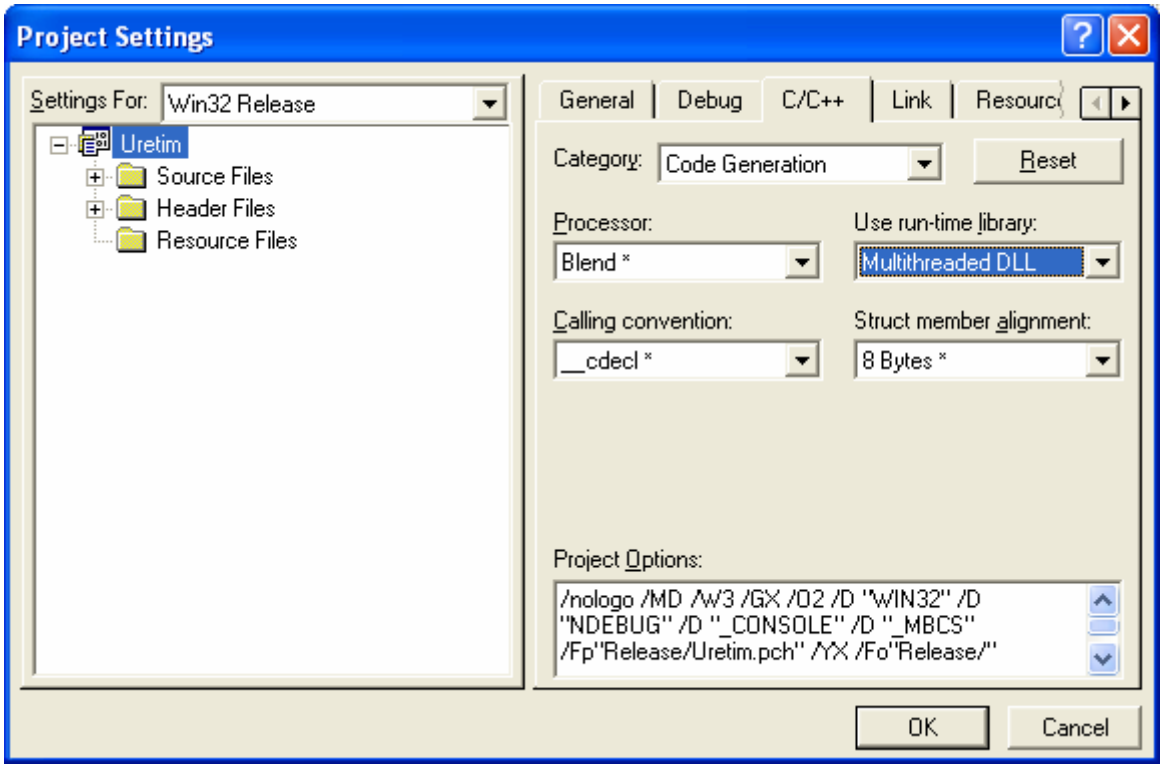


Şekil 5.14. Win32 Release için RTI kütüphanelerinin tanımlanması

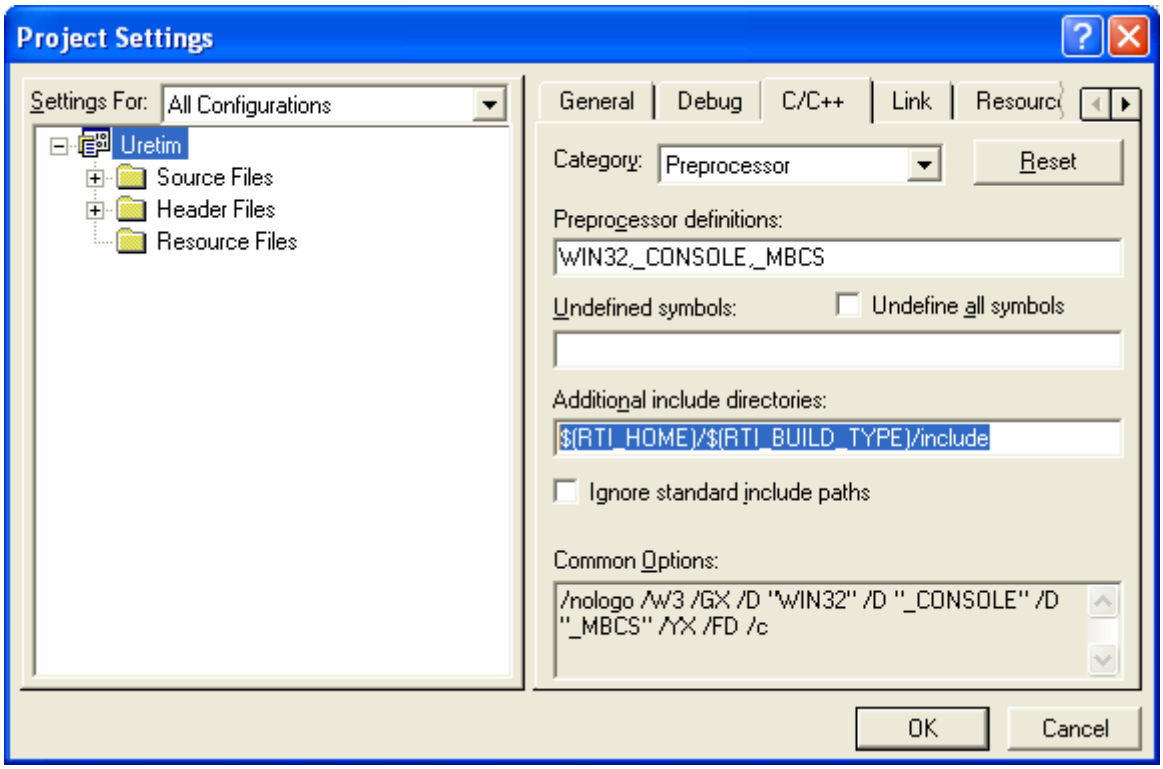
Aynı pencerede “C/C++” sekmesine geçilir. Sol tarafta “Win32 Release” seçili halde iken sağ tarafta “Category:” kısmında “Code Generation” seçilir. “Use run-time library” kısmında ise “Multithreaded DLL” seçilmelidir.

Benzer şekilde sol tarafta “Win32 Debug” seçilerek “Use run-time library” kısmında “Debug Multithreaded DLL” seçilmelidir. Bununla ilgili işlemler Şekil 5.15’de gösterilmektedir.

Yine aynı sekmede sol tarafta “All Configurations” seçildikten sonra “Category” kısmında “Preprocessor” seçilir. Şekil 5.16’da gösterildiği gibi “Additional include directories:” kısmına, RTI ile ilgili eklenti dosyalarının bulunduğu klasörün tanımlaması yapılmalıdır. Bunun için belirtilen kısma “$(RTI_HOME)/$(RTI_BUILD_TYPE)/include” ifadesi eklenmelidir.



Şekil 5.15. Code Generation tanımlamaları



Şekil 5.16. RTI eklenti dosyalarının tanımlanması

Ele alınan örnek uygulamada iki federe mevcuttur. Birinci federe üretici, ikinci federe tedarikçi federesidir. Dolayısıyla yukarıda anlatılanlar her iki federe için yapılmalıdır.

5.5.2. Üretici federesi

Üretici federesi, ana üretimin gerçekleştirildiği yerdir. Burada sipariş açılmakta, A alt-ürünü ve C nihai ürünü üretilmektedir. HLA uyumlu tasarlanan üretici federesinde var olan dosyalar ve bunların görevi veya hangi amaçla geliştirildiği **Tablo 5.2**'de verilmiştir.

Tablo 5.2. Üretici federesindeki dosyalar ve bunların amaçları

Dosya İsmi	**Görevi / Amacı**
uretim.cpp	Federenin ana dosyasıdır. Benzetimin algoritması bu dosyada bulunmaktadır. rti_arayuz.cpp dosyasında geliştirilen fonksiyonlar burada hazır olarak belirli bir mantık sırasıyla kullanılmaktadır. Bu dosya, EK A'da verilmiştir.
rti_arayuz.cpp	Üretim federesinin RTI ile etkileştiği fonksiyonların geliştirildiği C++ dosyasıdır. FED dosyasından sınıf yapılarının alınması, federasyonun oluşturulması ve federasyona katılınması gibi fonksiyonlar bu C++ dosyasının içerisindedir. Bunun dışında federe içerisinde gerçekleşen işlemler burada fonksiyon olarak tanımlanmış ve tümü uretim.cpp dosyasında hazır fonksiyon olarak kullanılmıştır. Örneğin siparişin açılması, A ve C alt-ürününün üretilmesi gibi fonksiyonlar buradadır. Bu dosyanın tamamı EK B'de verilmektedir.

Tablo 5.2. Üretici federesindeki dosyalar ve bunların amaçları (Devam)

Dosya İsmi	**Görevi / Amacı**
rti_arayuz.h	rti_arayuc.cpp dosyasında geliştirilecek fonksiyonların parametreleri ile birlikte sisteme tanıtıldığı başlık (header) dosyasıdır. Özellikle rti_arayuz.cpp dosyasına ve gerekli diğer dosyalara başlık dosyası olarak eklenmektedir. EK C'de rti_arayuz.h dosyası verilmiştir.
UrFederateAmbassador.cpp	Uretim federesinin elçi (ambassador) dosyasıdır. Her federede bir elçi dosyası bulunmak zorundadır. RTI'ın kurulumu ile örnek (boş) bir elçi dosyası gelmekte, dosyanın içerisi federenin gereksinimlerine göre geliştirilmektedir. Diğer federelerin RTI'a gönderdiği çağrıların etkisi ilgili federenin elçi dosyasına gelmektedir. Örneğin tedarikçi federesi updateAttributeValues ile bir bilgi güncellemesi yaptığında bunun etkisi üretim federesinin elçi dosyasının reflectAttributeValues fonksiyonuna gelmektedir. Uretim federesinin elçi dosyası EK D'de verilmektedir.
UrFederateAmbassador.h	Elçi dosyasındaki fonksiyonların ve parametrelerinin tanımlı olduğu başlık dosyasıdır. RTI'ın kurulumu ile birlikte gelmektedir. Elçi dosyasının başına, başlık dosyası olarak eklenmektedir.
Urun.h	Federeler arası etkileşimler sonucu bilgiler elçi dosyalarına gelmektedir. Bunların başında bilgi güncellemesi sonucu reflectAttributeValues ve etkileşim gönderimi sonucu receiveInteraction fonksiyonlarının tetiklenmesi gelmektedir ki bunlar UrFederateAmbassador.cpp dosyasının içindeki fonksiyonlardır. Ancak buraya gelen verinin geliştirilen diğer dosyalarda da kullanılması gerekmektedir. Bunun bir çözümü olarak ilgili verileri

Tablo 5.2. Üretici federesindeki dosyalar ve bunların amaçları (Devam)

Dosya İsmi	**Görevi / Amacı**
Urun.h (Devam)	bir sınıf yapısıyla veya global değişkenlerle saklamak ve diğer dosyalardaki fonksiyonların kullanımına sunmaktır. Bu amaçla Urun.h dosyası geliştirilmiş ve diğer dosyalara başlık dosyası olarak eklenmiştir. Örneğin tedarikçi federesi tarafından gönderilen B alt ürününün başlama ve bitiş zamanları elçi dosyasından alınarak rti_arayuz.cpp dosyasında, C ürününün başlama zamanını belirlemek üzere kullanılmaktadır. Bunun için gerekli değişkenler ve sınıf yapısı Urun.h dosyasında tanımlanmıştır ve EK E'de görülmektedir.
Urun.cpp	Urun.h dosyasında tanımlanan sınıf yapısı ve değişkenlerle ilgili fonksiyonların geliştirildiği C++ dosyasıdır. Urun.h, bu dosyaya başlık dosyası olarak eklenmektedir. Urun.cpp dosyası EK F'de verilmiştir.
sonuc.dat	Federasyonun çalışması esnasında oluşan bilgilerin kaydedildiği ve federasyonun sonlandırılmasından önce verilerin okunup ekrana yazdırıldığı çıktı dosyasıdır.
uretim.fed	Uretim federesinin FED dosyasıdır. Federasyonda kullanılacak tüm bilgiler burada tanımlandığından aynı zamanda FOM dosyasıdır. Nesneler, etkileşimler, nesne özellikleri gibi nesne modeli tanımlamalarının HLA-OMT yapısıyla tanımlandığı dosyadır.
RTI.rid	RTI'ın kurulumu ile birlikte gelmektedir ve her federeye dâhil edilmek zorundadır. RTI'ın çalışma şartları (örneğin federelerin farklı bilgisayarlarda çalışması) ile ilgili ayarlamaların yapıldığı dosyadır. RID dosyası daha ayrıntılı olarak Bölüm 5.6.1'te anlatılmıştır.

5.5.2.1. Üretici federesinin işleyişi

Yukarıda anlatılan dosyalar geliştirildiğinde üretici federesi hazır hale gelmiş olacaktır. Üretici federesinin ana algoritması uretim.cpp dosyasında aşağıdaki gibi tanımlanmıştır:

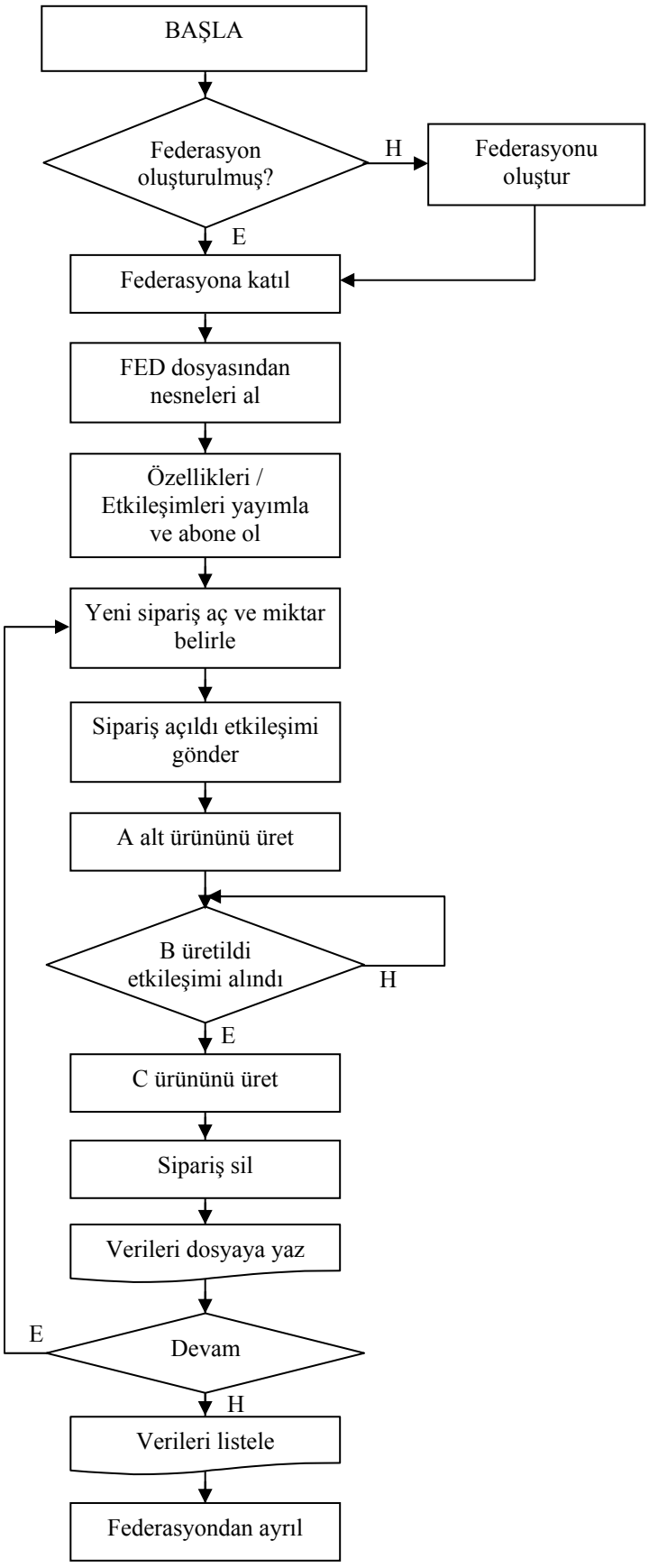
```
void main(int argc,char *argv[])
{
	int i, sipID, AID, CID;
	Urun *obj;
	srand(time(NULL));
	FederasyonaKatil();
	OzellikTutamaciniAl();
	OzellikleriYayimlaAboneol();
	EtkilesimleriYayimlaAboneol();

	for (i=0; i<10; i++)
	{
		cout<<"_____"<<endl;
		obj -> SetCUret(0);
		cout<<"Devam etmek icin bir tusa basiniz."<<endl;getch();
		sipID=YeniSiparis();
		MiktarGuncelle(sipID);
		SipAcildiEtkilesimiGonder();
		AID=SiparisA();
		AUrunuUret(AID);

		while (!Urun::CUret())
		{
			TickRTI();
		}

		CID = SiparisC();
```

```
            CUrunuUret (CID);
            SiparisSil(AID);
            SiparisSil(CID);
            SiparisSil(sipID);
            DosyayaYaz();
            cout<<"_____"<<endl;
        }
        DosyadanListele();
        ofstream dosyayayaz ("sonuc.dat", ios::app);
        dosyayayaz <<endl;
        FederasyondanAyril();
}
```

Ana algoritması yukarıda verilen üretici federesinin program akış çizelgesi Şekil 5.17'de görülmektedir.

Üretici federesi çalışmaya başladığında önce URETIM federasyonunun oluşturulup oluşturulmadığı kontrol edilmektedir. Şayet federasyon oluşturulmamışsa önce federasyon oluşturulur ve sonra federasyona katılım gerçekleşir. Bu işlem için rti_arayuz.cpp dosyasında tanımlanmış olan FederasyonaKatil() fonksiyonu kullanılmaktadır (EK B).

Daha sonra, RTI fonksiyonları kullanılarak, FED dosyası içerisinde tanımlı bulunan HLA-OMT yapısındaki nesnelerin C++ içerisinde kullanılacak değişkenlere atanması işlemi gelmektedir. Bu işlem için rti_arayuz.cpp dosyasında OzellikTutamaciniAl() fonksiyonu yazılmıştır (EK B). Ardından, federasyona nesnelerin hangi özellik değerlerini veya etkileşimleri yayımlayabileceğimizi ve hangilerine abone olabileceğimizi (diğer federelerin gönderdiği verileri alabileceğimizi) bildirmemiz gerekmektedir. Özelliklerle ilgili olarak OzellikleriYayimlaAboneol() fonksiyonu, etkileşimlerle ilgili olarak EtkilesimleriYayimlaAboneol() fonksiyonu tanımlanmıştır (EK B).



Şekil 5.17. Üretim federesinin işlem akış şeması

Buraya kadar yapılan işlemler bir bakıma benzetime hazırlık gibidir. Bundan sonra benzetimin asıl algoritması gelmektedir. Benzetimin ne kadar devam ettirileceği ile ilgili çeşitli politikalar geliştirmek mümkündür. Burada örnek olarak sistem on adım çalıştırılmıştır. Döngünün başında, C ürününü üretmeye geçişi kontrol etmek için obj -> SetCUret(0) ile mantıksal değişkene sıfır değeri atanmıştır. Bu değer, tedarikçi federesinden B alt ürününün üretildiği bilgisi (etkileşimi) alındığında bir (1) değerini almakta ve C ürününün üretimine geçilebilmektedir. Bu sebeple döngünün başına her gelindiğinde bu değişken sıfıra eşitlenmelidir.

C ürününü üretme ile ilgili mantıksal değişken, UrFederateAmbassador.cpp dosyası içerisinde bulunan receiveInteraction() fonksiyonu altında bir (1) değerini almaktadır. Bu fonksiyon, tedarikci federesi tarafından gönderilen B alt-ürününün üretildiği ile ilgili etkileşimin alınması sonucu tetiklenmekte ve içerisindeki kodlar çalıştırılmaktadır. Bununla ilgili olarak aşağıdaki kodlar yazılmıştır:

```
if (theInteraction = = BurunuUretildiID)
{
        cout << "B Urunu uretildi bilgisi alindi."<<endl;
        Urun *obj;
        obj -> SetCUret(1);
}
```

Daha sonra yeni bir siparişin açılma işlemine gelinmektedir. Yeni bir sipariş açmak, FED dosyasında “Urun” olarak tanımladığımız nesnenin yeni bir örneğini (instance) oluşturmak anlamındadır. Her yeni nesne örneğinin farklı bir numarası olacaktır, bu ise sipariş numarası olarak kullanılmıştır. Bu işlem sipID=YeniSiparis() satırında gerçekleştirilmektedir. YeniSiparis() fonksiyonu (EK B), sipariş numarası olarak kullanılacak tamsayı (integer) bir değer döndürmektedir. Tanımladığımız Urun nesnesinin yeni bir örneği, RTI’ın aşağıdaki komutu ile oluşturulmaktadır:

```
id = rtiAmb.registerObjectInstance(urunClassID);
```

Miktar değeri Urun nesnesinin bir özelliği olduğundan, yeni bir Urun nesnesi örneğinin oluşturulmasının ardından miktar değeri tespit edilip güncellenmelidir. Bu amaçla MiktarGuncelle(sipID) fonksiyonu kullanılmaktadır. Fonksiyonun içeriğine Ek B'den bakmak mümkündür. Fonksiyonun parametresi olan sipID, Urun nesne örneğinin numarası, dolayısıyla açtığımız yeni siparişin numarasıdır. Miktar değerinin hangi siparişe ait olduğu bu şekilde ayırt edilmektedir. Miktar değeri her seferinde rassal olarak belirlenmektedir. Örnek olarak 100 ile 150 arasında bir değer oluşturulmaktadır. Oluşturulan miktar değeri Urun nesnesinin miktar özelliğine değer olarak atanmalı ve ardından bu bilgiye ihtiyaç duyan federasyonun diğer federelerine (burada tedarikçi federesine) bildirilmesi gerekmektedir. Bu işlemler ilgili fonksiyonun içerisinde tanımlanmıştır. Burada kullanılan rtiAmb.updateAttributeValues RTI fonksiyonu neticesinde, bu veriye abone olan tedarikçi federesinin elçi dosyası (TedFederateAmbassador.cpp) içerisindeki reflectAttributeValues fonksiyonu tetiklenecektir. Böylece miktar bilgisinin tedarikçi federesine iletilmesi sağlanmaktadır. Bu işlem HLA'nın özelliğidir ve RTI tarafından otomatik olarak gerçekleştirilmektedir. Tabi bunun için tedarikçi federesinin elçi dosyası içerisindeki reflectAttributeValues fonksiyonu uygun şekilde yazılmalıdır.

Siparişin miktarı belirlendikten sonra tedarikçi federesine siparişin açıldığı bilgisi gönderilmektedir. Bu amaçla, içeriği EK B'de verilmiş olan SipAcildiEtkilesimiGonder() fonksiyonu yazılmıştır. Etkileşim göndermek için rtiAmb.sendInteraction RTI fonksiyonu kullanılmaktadır. Bu komut, tedarikçi federesindeki elçi dosyasının receiveInteraction fonksiyonunu tetikleyecektir. Bu fonksiyonun içerisine yazılacak uygun komutlarla tedarikçi federesi sipariş açıldığı bilgisini alacaktır.

Sipariş açıldı bilgisi tedarikçi federesine gönderildikten sonra üretici federesi A alt ürününü, tedarikçi federesi ise B alt ürününü üretmeye başlayacaktır. A alt ürününün üretilmesi işlemlerini AID=SiparisA() ve AUrunuUret(AID) fonksiyonları gerçekleştirecektir (EK B). Bunlardan AID=SiparisA() fonksiyonu FED dosyasındaki Urun nesnesinin alt nesnesi olan UrunA nesnesinin yeni bir örneğini oluşturmaktadır. AUrunuUret(AID) fonksiyonu ile A alt ürününün üretilmesi temsil edilmektedir. Bu fonksiyonun içerisinde A alt ürününün başlama zamanı olarak

federasyon zamanı alınmakta, bitiş zamanı ise miktar değerinin üretim süresi ile çarpılması sonucunun, A'nın başlama zamanına eklenmesi olarak hesaplanmaktadır. A alt ürününün başlama ve bitiş zamanları hesaplandıktan sonra bunlar UrunA alt nesnesinin özelliklerine atanmakta ve güncellenme sağlanmaktadır.

A alt ürününün başlama ve bitiş zamanlarının belirlenmesinden sonra C ürününün üretilmesine geçilecektir. Ancak daha öncesinde tedarikçi federesinden B alt ürününün üretildiği bilgisinin etkileşim olarak alınması beklenmektedir. Bu amaçla programa aşağıdaki döngü eklenmiştir:

```
while (!Urun::CUret())
{
        TickRTI();
}
```

Hatırlanacak olursa programın ana döngüsünün başına obj -> SetCUret(0) satırı eklenmişti. Bu, Urun.h dosyasında (EK E) oluşturulan Urun sınıfının (class) "curet" değişkenine sıfır değeri atamaktadır. Urun::CUret() ise, yine Urun.h dosyasındaki tanımlamaya göre "curet" değerini okumaktadır. Bu sebeple program, yukarıda verilen döngüye girecektir. İlk bakışta yukarıdaki döngü sonsuz gibi görünmektedir. Ancak döngünün içerisindeki TickRTI() fonksiyonu (EK B) federenin güncellenmesini sağlamaktadır. Bu fonksiyonun içerisinde bulunan rtiAmb.tick() komutu federenin güncellenmesine fırsat vermektedir. Bu esnada tedarikçi federesinden gelen B alt ürününün başlama ve bitiş zamanlarının güncellenmesi bilgisi ve B alt ürünün üretildiği etkileşimi, üretici federesindeki elçi dosyasına gelerek ilgili fonksiyonları tetikleyecektir. Tedarikçi federesi B alt ürününü ürettiği bilgisini etkileşim olarak gönderdiğinde UrFederateAmbassador.cpp dosyasındaki (EK D) receiveInteraction fonksiyonu tetiklenecek ve bu fonksiyonun içerisinde "curet" değişkenine bir (1) değeri atanarak yukarıdaki sonsuz döngüden çıkılacaktır. Dolayısıyla döngüden çıkma şartı, B ürünü üretildi etkileşiminin alınmasıdır. Sonsuz döngüden çıkmayı sağlayan, receiveInteraction fonksiyonu içerisindeki satırlar aşağıdaki gibidir:

```
if (theInteraction = = BurunuUretildiID)
{
        cout << "B Urunu uretildi bilgisi alindi..."<<endl;
        Urun *obj;
        obj -> SetCUret(1);
}
```

Burada önce, alınan etkileşimin B ürünü üretilmesi etkileşimi olup olmadığı karşılaştırılmakta, eğer öyle ise "curet" değişkenine obj -> SetCUret(1) satırı ile bir (1) değeri atanmaktadır. while (!Urun::CUret()) şartı artık sağlanmayacağından döngüden çıkılacak ve altındaki satırlarda bulunan C ürününün üretilmesi işlemine geçilecektir.

C ürününün üretilmesini CID = SiparisC() ve CUrunuUret(CID) fonksiyonları temsil etmektedir (EK B). Bu fonksiyonlardaki mantık, A alt ürünü üretme ile ilgili söylediklerimizle aynıdır. Farklı olan, C ürününün başlama ve bitiş zamanlarının belirlenmesi ile ilgili olan kısımlardır. Öncelikle federasyonun zamanı, A ve B alt ürünlerinin bitiş zamanlarına bağlı olarak yeniden belirlenir. Federasyonun yeni zamanı, A ve B alt ürünlerinin bitiş zamanlarından büyük olana ötelenir. Bununla ilgili satırlar aşağıdadır:

```
if (Urun::ABitZaman()>Urun::BBitZaman())
{
        fedZamani = Urun::ABitZaman();
}
else if (Urun::ABitZaman()<=Urun::BBitZaman())
{
        fedZamani = Urun::BBitZaman();
}
```

Ardından C ürününün başlama zamanı federasyon zamanına eşitlenir, daha sonra da miktara ve üretim süresine bağlı olarak C ürününün bitiş zamanı hesap edilir. En

sonunda, yeni siparişin zamanının kaldığı yerden devam etmesi için federasyon zamanına, C ürününün bitiş zamanı atanmaktadır.

Böylece bir siparişin üretimi tamamlanmış olmaktadır. Programın ana döngüsünün sonunda sipariş silinmektedir. Siparişi, SiparisSil(int ID) fonksiyonun içerisindeki rtiAmb.deleteObjectInstance RTI komutu gerçekleştirmektedir (EK B). Siparişi silmekle aslında nesne örnekleri RTI'dan silinmiş olmaktadır. Daha sonra ise elde edilen veriler, DosyayaYaz() fonksiyonu (EK B) ile dosyaya yazılmaktadır. Yazılan veriler şunlardır: Sipariş numarası, miktar, A alt ürününün başlama ve bitiş zamanları, B alt ürününün başlama ve bitiş zamanları ve C ürününün başlama ve bitiş zamanları.

Buraya kadar anlatılanlar ile programın bir döngüsü tamamlanmış olmaktadır. Örnek olarak programın on kere döndürülmesi tanımlandığından yukarıdaki işlemler 10 kez gerçekleşecektir. Daha sonra, programın çalışması esnasında dosyaya yazılan veriler ekrana listelenecektir. Bu işlemi DosyadanListele() fonksiyonu gerçekleştirecektir (EK B). Programın çalışması boyunca elde edilen veriler, programın çıktıları olarak sonuc.dat dosyasına yazılmaktadır. Bu aynı zamanda federasyonun da çıktısıdır.

Elde edilen veriler ekrana yazdırıldıktan sonra federasyondan çıkılmakta ve federasyon yok edilmektedir. Bu işlemi, rti_arayuz.cpp dosyasında tanımlanan FederasyondanAyril() fonksiyonu gerçekleştirmektedir (EK B). Böylece federasyondan en son çıkan, federasyonu da yok edecektir.

5.5.3. Tedarikçi federesi

Tedarikçi federesi, üretici federesinden edindiği bilgilere göre B alt ürününün üretildiği yerdir. B ürünü üretildikten sonra gerekli bilgiler üretici federesine gönderilmektedir. HLA uyumlu tedarikçi federesi oluşturmak için kullanılan ve geliştirilen dosyalar ve bunların amaçları **Tablo 5.3**'de verilmiştir. Üretici federesini oluşturmak için geliştirilen programların kaynak kodları, bu çalışmanın sonunda ek olarak sunulmuştur. Tedarikçi federesiyle ilgili geliştirilen kaynak kodları ise, ekler kısmını fazla büyütmemek için verilmemiştir. Tedarikçi federesinin işleyişi

anlatılırken gerekli görüldükçe ilgili kaynak kodları açıklanmıştır. Verilmeyen kısımları ise üretici federesindeki kodlardan esinlenerek tahmin etmek mümkündür. Örneğin, üretici federesinde A alt ürününün üretilmesinin temsil edildiği kodlar ile tedarikçi federesindeki B alt ürününün üretilmesini temsil eden kodlar benzerdir.

Tablo 5.3. Tedarikçi federesindeki dosyalar ve bunların amaçları

Dosya İsmi	**Görevi / Amacı**
tedarik.cpp	Federenin ana dosyasıdır. Benzetimin algoritması bu dosyada bulunmaktadır. rti_arayuz.cpp dosyasında geliştirilen fonksiyonlar burada hazır olarak belirli bir mantık sırasıyla kullanılmaktadır.
rti_arayuz.cpp	Tedarikçi federesinin RTI ile etkileştiği fonksiyonların geliştirildiği C++ dosyasıdır. FED dosyasından sınıf yapılarının alınması, federasyonun oluşturulması ve federasyona katılınması gibi fonksiyonlar bu C++ dosyasının içerisindedir. Bunun dışında federe içerisinde gerçekleşen işlemler burada fonksiyon olarak tanımlanmış ve tümü tedarik.cpp dosyasında hazır fonksiyon olarak kullanılmıştır. Örneğin B alt ürününün üretilmesi, B alt ürünü üretildi bilgisinin gönderilmesi gibi fonksiyonlar buradadır. Aynı isimle üretici federesi için geliştirilen bu dosyanın benzeri için EK B kullanılabilir.
rti_arayuz.h	rti_arayuc.cpp dosyasında geliştirilecek fonksiyonların parametreleri ile birlikte sisteme tanıtıldığı başlık (header) dosyasıdır. Özellikle rti_arayuz.cpp dosyasına ve gerekli diğer dosyalara başlık dosyası olarak eklenmektedir. Üretici federesi için olan bir benzeri EK C'de verilmektedir..

Tablo 5.3. Tedarikçi federesindeki dosyalar ve bunların amaçları (Devam)

Dosya İsmi	**Görevi / Amacı**
TedFederateAmbassador.cpp	Tedarikçi federesinin elçi (ambassador) dosyasıdır. Üretici federesinin RTI'a gönderdiği çağrıların etkisi tedarikçi federesinin bu elçi dosyasına gelmektedir. Örneğin üretici federesi updateAttributeValues ile bir bilgi güncellemesi yaptığında bunun etkisi tedarikçi federesinin elçi dosyasının reflectAttributeValues fonksiyonuna gelmektedir. Benzeri bir dosyası EK D'de verilmektedir.
UrFederateAmbassador.h	Elçi dosyasındaki fonksiyonların ve parametrelerinin tanımlı olduğu başlık dosyasıdır. Elçi dosyasının başına, başlık dosyası olarak eklenmektedir.
Urun.h	Tedarikçi federesinde kullanılacak nesnelerin değişkenlerinin tanımlandığı dosyadır. Amaç, elde edilen verileri, tedarikçi federesinin her dosyasında kullanma imkanına kavuşmaktır. Daha ayrıntılı bilgi için üretici federesinde verdiğimiz açıklamalara bakılabilir. Üretici federesi için hazırlanan ile neredeyse aynıdır (EK E).
Urun.cpp	Urun.h dosyasında tanımlanan sınıf yapısı ve değişkenlerle ilgili fonksiyonların geliştirildiği C++ dosyasıdır. Urun.h, bu dosyaya başlık dosyası olarak eklenmektedir. Benzeri bir dosya EK F'de verilmiştir.
uretim.fed	Tedarikçi federesinin FED dosyası, üretim federesi için geliştirilen ile aynıdır. Çünkü kullanılan nesneler aynıdır. Burada, benzetimin nesne modeli (SOM) görevini yapmaktadır.
RTI.rid	Her federenin RID dosyası olması gerektiğinden tedarikçi federesine de eklenmiştir. RTI'ın çalışma şartları ile ilgili ayarlamaların yapıldığı dosyadır ve daha ayrıntılı olarak Bölüm 5.6.1'te anlatılmıştır.

5.5.3.1. Tedarikçi federesinin işleyişi

Tedarikçi federesinin görevi, üretici federesine B alt ürününü temin etmektir. Bu işlemi gerçekleştirebilmek için üretici federesinden bir takım bilgileri elde etmesi gerekmektedir. B alt ürününü üretmesi neticesinde oluşan bilgileri ise üretici federesine göndermektedir. Tedarikçi federesinin programının ana algoritması aşağıda verilmektedir:

```
void main(int argc,char *argv[])
{
        int i, Bid;
        Urun *obj;
        FederasyonaKatil();
        OzellikTutamaciniAl();
        OzellikleriYayimlaAboneol();
        EtkilesimleriYayimlaAboneol();
        for (i=0; i<10; i++)
        {
        cout<<"_____"<<endl;
                obj->SetSipAlindi(0);
                obj->SetGuncellendi(0);
                while ((!Urun::SipAlindi()) && (!Urun::Guncellendi()))
                {
                        TickRTI();
                }
                Bid=SiparisB();
                BUrunuUret(Bid);
                BuretildiEtkilesimiGonder ();
                cout<<"_____"<<endl;
                SiparisSil(Bid);
        }
   FederasyondanAyril();
}
```

Ana programı yukarıda verilen tedarikçi federesinin işlem akış şeması Şekil 5.18'de verilmektedir.

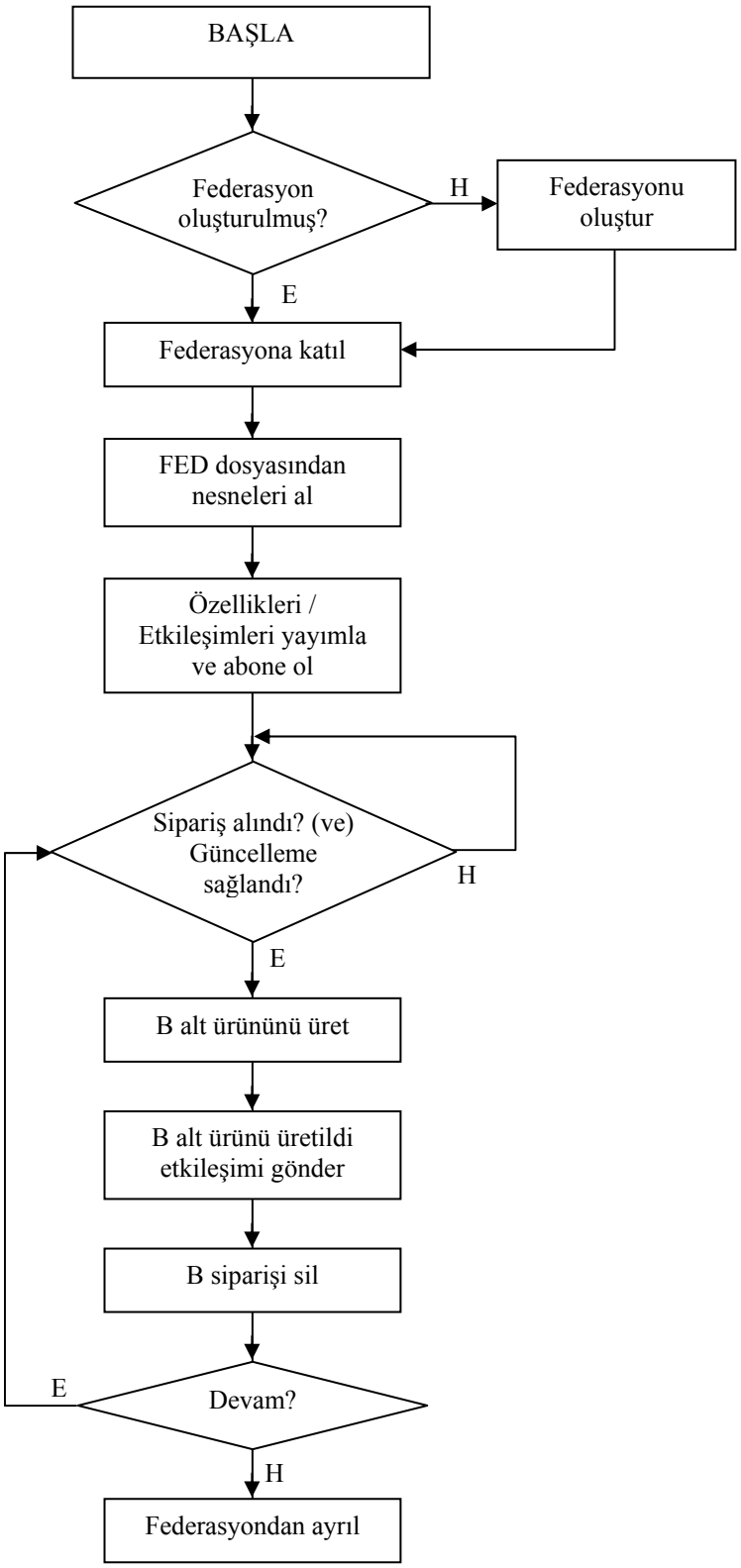
Tedarikçi federesi çalışmaya başlamadan önce üretim federesi çalıştırılmış ise, URETIM federasyonu oluşturulmuş olacaktır. Ancak her ihtimale karşı tedarikçi federesi öncelikli olarak federasyonun oluşturulup oluşturulmadığına bakmaktadır. Eğer oluşturulmamış ise önce federasyon oluşturulur ve ardından federasyona katılma işlemi gerçekleştirilmektedir. Bu işlemi gerçekleştirmek üzere rti_arayuz.cpp dosyasında FederasyonaKatil() fonksiyonu tanımlanmış, ana programda ise bu fonksiyon kullanılmıştır.

FED dosyası içerisinde var olan HLA-OMT yapısındaki nesneleri C++ içerisinde kullanılabilir hale getirilmesi gerekmektedir. Bu amaçla OzellikTutamaciniAl() fonksiyonu geliştirilmiştir. Yine rti_arayuz.cpp içerisinde tanımlanan bu fonksiyon, ana programda hazır fonksiyon olarak kullanılmaktadır.

Tedarikçi federesinin ilgi duyduğu (abone olma) veya üzerinde işlem yapıp güncelleyeceği (yayımlama) nesne özelliklerinin ve etkileşimlerin federasyona haber verilmesi gerekmektedir. Hangi nesne özelliklerinin yayımlanıp hangilerine abone olunacağı OzellikleriYayimlaAboneol() fonksiyonu ile, hangi etkileşimlerin yayımlanıp hangilerine abone olunacağı ise EtkilesimleriYayimlaAboneol() fonksiyonu ile gerçekleştirilmektedir. Bunlar da rti_arayuz.cpp dosyasında tanımlanmışlardır.

Yukarıda yapılan işlemler ile tedarikçi federesi bir bakıma asıl algoritmaya hazır hale getirilmiştir. Tedarikçi federesinin çalışma şartlarını üretici federesine uygun hale getirmek için federe on iterasyon çalıştırılmıştır. Ana döngünün başına, üretici federesinden sipariş açıldığına dair bilginin ve üretici federesinde gerçekleştirilen güncellemelerin alınıp alınmadığını kontrol etmek için ilgili mantıksal değişkenlere aşağıdaki satırlar ile sıfır değeri atanmaktadır.

```
obj->SetSipAlindi(0);
obj->SetGuncellendi(0);
```



Şekil 5.18. Tedarikçi federesinin işlem akış şeması

Yukarıdaki fonksiyonlar, Urun.h dosyasında tanımlanmıştır ve sırasıyla “sip_alindi” ve “guncellendi” değişkenlerine değer atamaktadır. Daha sonra ana programda aşağıdaki döngü ile işlemlere devam edilip edilmeyeceği kontrol edilmektedir:

```
while ((!Urun::SipAlindi()) && (!Urun::Guncellendi()))
{
        TickRTI();
}
```

Bu ifadeler, sipariş alınmadığı ve güncellenme yapılmadığı müddetçe TickRTI() fonksiyonunu çalıştırılmaktadır. TickRTI() fonksiyonunun içerisinde bulunan rtiAmb.tick() komutu, federenin güncellenmesini temin etmektedir. Yani RTI’dan gelen (üretici federesi tarafından gönderilen) işlemler RTI elçisinin (ambassador) tick() fonksiyonu sayesinde işletilmekte ve sonuçlar tedarikçi federesine gelmiş olmaktadır. İşte bu esnada üretici federesinin güncellediği miktar, A alt ürününün başlama ve bitiş zamanları ile siparişin gönderilme etkileşimi tedarikçi federesinde karşılığını bulmaktatır. Yukarıdaki sonsuz döngü gibi görülen döngüden de bu şekilde çıkmak ve işlemlere devam etmek mümkün hale gelmektedir. Döngüden çıkma şartı, sipariş açıldı etkileşiminin alınması (sip_alindi = 1) ve güncellemelerin yapılmış olmasıdır (guncellendi = 1). İlgili değişkenlerin bir (1) değerini alması, tedarikçi federesinin elçi dosyası olan TedFederateAmbassador.cpp dosyasında gerçekleşmektedir. Sipariş alındığında receiveInteraction fonksiyonu tetiklenmekte, güncellemeler alındığında ise reflectAttributesValues fonksiyonu tetiklenmektedir. Bu fonksiyonlarının içerisine yazılan kodlar sayesinde ilgili değişkenlere bir (1) değeri atanmakta ve sonsuz döngüden çıkılmaktadır. Elçi dosyasının reflectAttributesValues fonksiyonunda bulunan aşağıdaki ifadelerden obj->SetGuncellendi(1) satırı ile “guncellendi” değişkenine bir (1) değeri atanmaktadır:

```
if (attrHandle = = baszamanAID)
{
        int basZaman,gbasZaman;
        theAttributes.getValue(i, (char *)&gbasZaman, valueLength);
```

```
        basZaman = ntohl(gbasZaman);
        obj->SetABasZaman(basZaman);
        cout<<"A Bas zaman alindi...: "<<Urun::ABasZaman()<<endl;
        obj->SetGuncellendi(1);
}
```

Benzer şekilde, elçi dosyasının receiveInteraction fonksiyonunda bulunan aşağıdaki ifadelerden obj->SetSipAlindi(1) satırı ile “sip_alindi” değişkenine bir (1) değeri atanmaktadır:

```
if (theInteraction = = siparisAcildiID)
{
cout <<"Siparis acildi bilgisi alindi..." <<endl;
obj->SetSipAlindi(1);
}
```

Güncellemeler sağlandıktan ve siparişin açılma etkileşimi alındıktan sonra program devam etmektedir. Sırada, B alt ürününün üretilmesi işlemleri vardır. Bu işlemler, Bid=SiparisB() ve BUrunuUret(Bid) fonksiyonları ile gerçekleştirilmektedir.

B alt ürününün başlama zamanı, siparişin açılma, dolayısıyla A alt ürününün başlama zamanı ile aynı olmaktadır. Bitiş zamanı ise miktar değeri ile üretim süresinin çarpılması sonucu elde edilmektedir. Bu bilgiler hesaplandıktan sonra değerlerin RTI kurallarına göre nesne özelliklerine atanması gerekmektedir. Ardından da üretici federesine bilgilerin gitmesini sağlamak için RTI’ın updateAttributeValues fonksiyonu kullanılmalıdır. İlgili işlemler BUrunuUret() fonksiyonu içerisinde aşağıdaki gibi tanımlanmıştır:

```
gbasZaman = htonl(basZaman);
gbitZaman = htonl(bitZaman);
RTI::AttributeHandleValuePairSet *isimDegerSeti;
isimDegerSeti = RTI::AttributeSetFactory::create(2);
isimDegerSeti->add(baszamanBID , (char*)&gbasZaman, sizeof(gbasZaman));
```

```
isimDegerSeti->add(bitzamanBID , (char*)&gbitZaman, sizeof(gbitZaman));
try
{
        rtiAmb.updateAttributeValues(RTI::ObjectHandle(id),          *isimDegerSeti,
NULL);
        cout<<"B basZaman: "<<basZaman<<" B bitZaman: "<<bitZaman<<endl;
}
catch (RTI::Exception&e)
{
        cerr<<"Baslangic ve Bitis Zamanlarini Guncellemede Hata: "<<&e<<endl;
}
isimDegerSeti->empty();
delete isimDegerSeti;
```

Yukarıdaki ifadeler ile B alt ürününün üretilmesi ve bilgilerin güncellenmesi temsil edilmiştir. Daha sonra, B alt ürününün üretilmesinin bittiğini, etkileşim ile üretici federesine iletme işlemi gelmektedir. Bu işlemi BuretildiEtkilesimiGonder() fonksiyonu yerine getirmektedir. Hatırlanacağı gibi üretici federesi C ürününü üretmeye geçmeden önce tedarikçi federesinden bu etkileşimi alması gerekmekteydi.

Buraya kadar ifade edilenler ile tedarikçi federesinin ana döngüsünün bir adımı gerçekleşmiş olmaktadır. Program yeniden ikinci adım için döngünün başına giderek üretici federesinden gelecek bilgileri bekleyecektir. Üretici federesi bu arada C ürünü üretmeyi bitirecek, federasyon zamanını C ürününün bitiş zamanına öteleyecek ve başa dönerek yeni sipariş açacaktır. Bilgiler karşılıklı iletilerek etkileşim sağlanacak ve federasyon bir bütün olarak işletilmiş olacaktır.

5.6. Federasyonun Çalıştırılması

Yukarıda anlatılan ışığında ve çalışmanın sonunda ek olarak verilen kodların geliştirilmesi neticesinde oluşturulan federasyonun çalıştırılması sağlanmıştır. Federasyon içerisindeki üretici ve tedarikçi federeleri tek bir makinede çalıştırılabileceği gibi dağıtık olarak iki bilgisayarda da çalıştırılabilmektedir.

Federasyonun çalışma şartları ile ilgili gerekli parametrelerin ayarlandığı dosya RID dosyasıdır. FED dosyası federasyon için gerekli olduğu gibi RID dosyası da federasyonun çalışması için gereklidir ve her iki dosya federenin .exe dosyası ile aynı klasörde olmalıdır.

5.6.1. RID dosyası

Federasyonun çalışması için gerekli diğer bir dosya ise RID (RTI.rid) dosyasıdır. Bu dosya RTI'ın kurulumunda gelmektedir. Örnek bir RID dosya [..\DMSO\RTI1.3NG-V6\Win2000-VC6\doc] klasöründen veya RTI ile birlikte kurulan "HelloWorld" uygulamasından edinmek mümkündür.

Federasyonun RTI'a bağlanması RID dosyasının içerisindeki tanımlamalar ile gerçekleşmektedir. RTI sorunsuz başlatıldığında ilgili pencerede RTI'ın çalıştığı makinenin IP adresi ve RTI için ayrılan port numarası "endpoint" parametresi ile yazdırılmaktadır (örneğin, endpoint = 192.168.1.2:2276). Burada görülen IP numarası ve port adresi, RID dosyasın RtiExecutiveEndpoint parametresinde belirtilmiş olması gerekmektedir. Ya da RID dosyasındaki RtiExecutiveEndpoint parametresinin bulunduğu satırın başına iki adet noktalı virgül (;) konularak bu parametre kapatılır. Bu durumda federe çalıştırıldığında RTI'ı algılamakta, IP adresi ve port numarası belirtmeye ihtiyaç kalmamaktadır. Ancak bu yöntem, federasyonun birden fazla makinede (federelerin ayrı makinelerde) çalışması durumunda federelerin RTI'ın nerede çalıştığını görememesine sebep olabilmektedir. Aynı makinede iki veya daha fazla federenin çalışmasında bu yöntem çalışmaktadır.

Diğer bir yöntem ise RID dosyasındaki "RtiExecutiveEndpoint" parametresi ile belirtilmiş olan IP adresi ve port numarasının (IP adreslerinin yerine bilgisayarın ismi/hostname de kullanılabilmektedir), rtiexec'in çalıştırılması esnasında kullanılmasıdır. Örneğin RTI.rid dosyasında "RtiExecutiveEndpoint 10.9.17.54:1569" olarak belirtilmiş ise rtiexec çalıştırılırken komut satırına aşağıdaki gibi yazmak gerekmektedir:

C:\Program Files\DMSO\RTI1.3NG-V6\Win2000-VC6\bin>rtiexec -endpoint 10.9.17.54:1569

Böylece hangi yöntem kullanılırsa kullanılsın RTI ile federasyonun bağlantısı düzenlenmiş olmaktadır. Aksi halde federasyonun çalıştırılması esnasında RTI'a bağlanamama hatası ile karşılaşılacaktır. Bu durumda önce üretici veya tedarikçi federesi çalıştırılır ardından diğer federe çalıştırılır ve federasyonun işlemesi sağlanır.

Bu açıklamalar, federasyonun bir makinede çalışması ile ilgilidir. Federasyonu birkaç makinede çalıştırmak için aşağıdaki düzenlemeler yapılmalıdır.

5.6.1.1. Federasyonun birden fazla bilgisayarda çalıştırılması

RTI-NG 1.3 versiyonu IP (Internet Protocol) temelli ağ protokollerini destekletmedir. Federasyonun birden fazla makinede çalıştırılması için yine RID dosyasında bazı parametrelerin düzenlenmesi ve rtiexec dosyasının çalıştırılmasının yöntemi değiştirilmelidir. Bilgisayar1 ve Bilgisayar2'de çalışan üretici ve tedarikçi federelerinin birbiriyle konuşabilmesi için iki yöntemden bahsedilecektir.

Birinci yönteme göre Bilgisayar1'de rtiexec "multicastDiscoveryEndpoint" parametresi ile birlikte çalıştırılmalıdır. Örneğin komut satırına aşağıdaki gibi yazılmalıdır:

C:\Program Files\DMSO\RTI1.3NG-V6\Win2000-VC6\bin>rtiexec –multicast DiscoveryEndpoint 224.9.9.2:22605

Burada parametre devamında görülen numaralar, RID dosyasındaki ile aynı olmalıdır (RtiExecutiveMulticastDiscoveryEndpoint 224.9.9.2:22605). Bu yönteme göre RID dosyasındaki "RtiExecutiveEndpoint" parametresi kapalı olmalıdır. Bunun için satır başına iki adet noktalı virgül işareti konulmalıdır. Daha sonra Bilgisayar1'de bulunan üretici veya tedarikçi federesi çalıştırılır.

Bilgisayar2'de rtiexec'in çalıştırılmasına ihtiyaç yoktur. Ancak bu bilgisayarda bulunan federenin de RID dosyasındaki "RtiExecutiveMulticastDiscoveryEndpoint" parametresinin Bilgisayar1'deki ile aynı olması ve "RtiExecutiveEndpoint" parametresinin kapalı olması gerekmektedir. Federe çalıştırıldığında Bilgisayar1'deki rtiexec'e bağlanacak ve iki federe birbiriyle etkileşim halinde olacaktır.

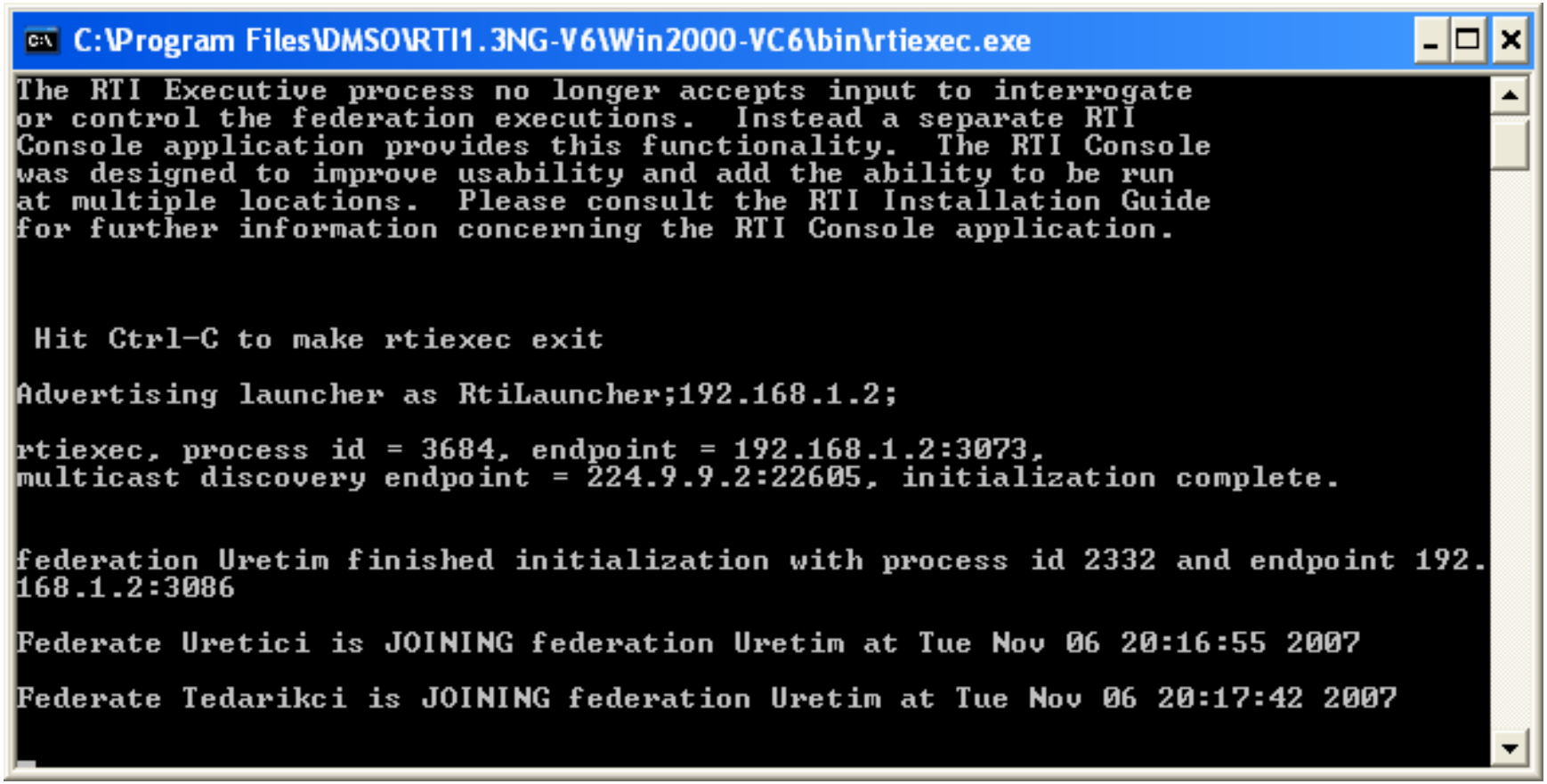
İkinci yönteme göre ise rtiexec "endpoint" parametresi ile birlikte çalıştırılır. Buna göre Bilgisayar1'de komut satırına aşağıdaki gibi bir ifade yazılmalıdır:

C:\Program Files\DMSO\RTI1.3NG-V6\Win2000-VC6\bin>rtiexec -endpoint 10.9.17.54:1569

İlgili federenin RID dosyasındaki "RtiExecutiveEndpoint" parametresinin açık olması ve komut satırında belirtilen ile aynı olması gerekmektedir. Ardından federe çalıştırılmalıdır.

Bilgisayar2'de yine rtiexec'in çalıştırılmasına gerek yoktur. Ancak federenin RID dosyasında "RtiExecutiveEndpoint" parametresinin açık olması ve Bilgisayar1'de belirtilen ile aynı olması gerekmektedir. Bu durumda da federeler birbirini görmekte ve etkileşim sağlanmaktadır.

Yukarıda verilen yöntemlerden herhangi biri ile RTI çalıştırılıp üretim fererasyonunu çalıştırmak mümkündür. Şekil 5.19'da RTI'ın başlatılmış hali, Üretim federasyonunun oluşturulduğu, Üretici ve Tedarikçi federelerinin federasyona katıldığı bilgileri görülmektedir.



Şekil 5.19. RTI'da üretim federasyonun oluşturulması ve federelerin katılması

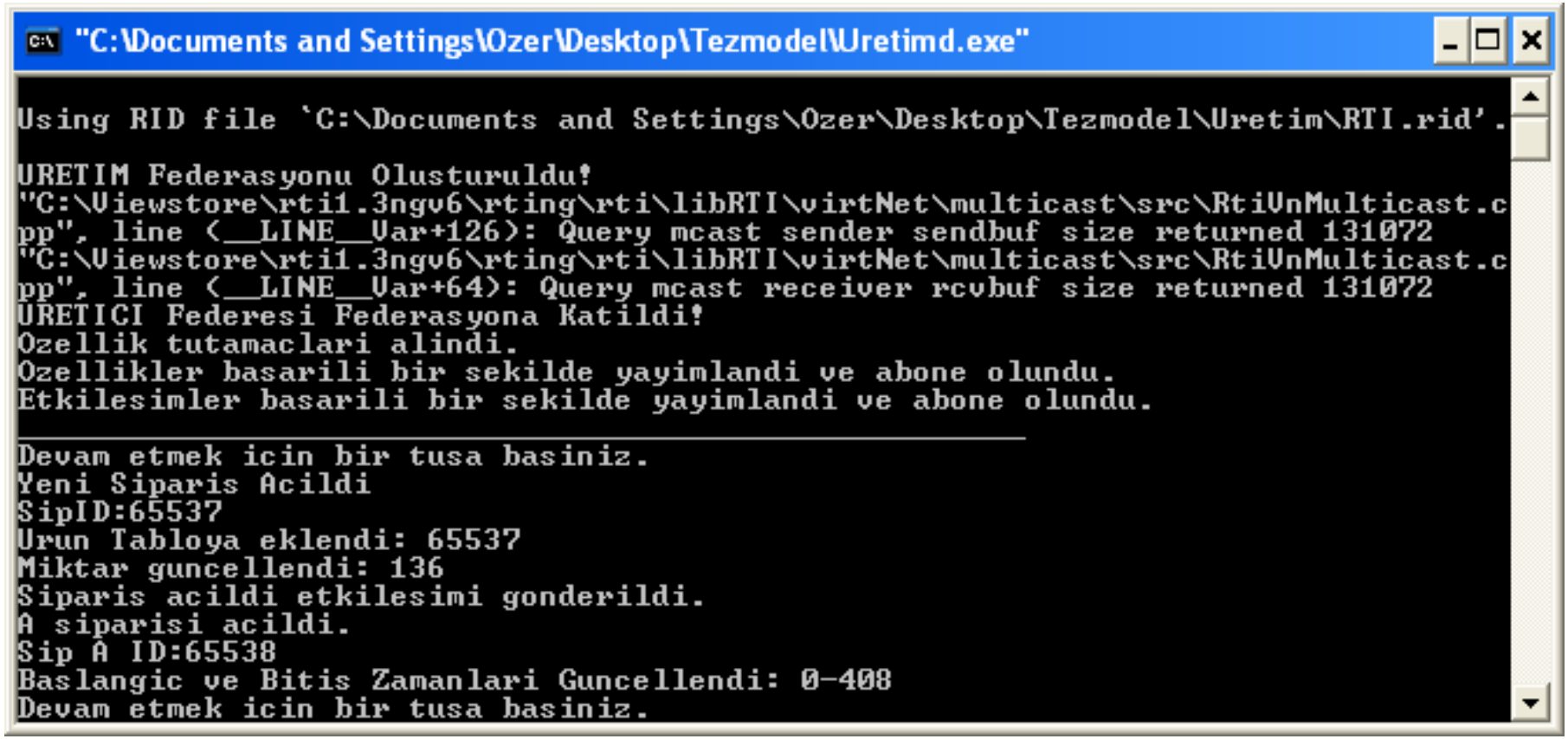
Üretici federesinin program akışını kontrol etmek maksadıyla, programın bazı noktalarda durdurulması ve ekran çıktılarının incelenmesi sağlanmıştır. Şekil 5.20'de, üretici federesinin A alt ürününü ürettiği kısma kadar olan program çıktıları görülmektedir. Buraya kadar üretici federesi sipariş açmış, miktar 136 olarak belirlenmiş ve tedarikçi federesine sipariş açıldığına dair etkileşim gönderilmiştir. A alt ürününün başlama ve bitiş zamanları da ekranda görülmektedir. Bu bilgileri de tedarikçi federesi edinmektedir.

Şekil 5.21'de tedarikçi federesinin çalışması ile ilgili ekran çıktısı görülmektedir. RTI'dan gelen bilgiler de kontrol amaçlı ekrana yazdırılmıştır. Şekilden görüleceği üzere, üretici federesinde açılan siparişin numarası, elçi dosyasının "discoverObjectInstance" fonksiyonu ile fark edilmiştir. Miktar bilgisi de 136 olarak tedarikçi federesinde yakalanmıştır. Siparişin açıldığı ile ilgili bilgi de alınmıştır. A alt ürününün başlama ve bitiş zamanları da, üretici federesinde oluşturulduğu değerlerle tedarikçi federesine bilgi olarak gelmiştir.

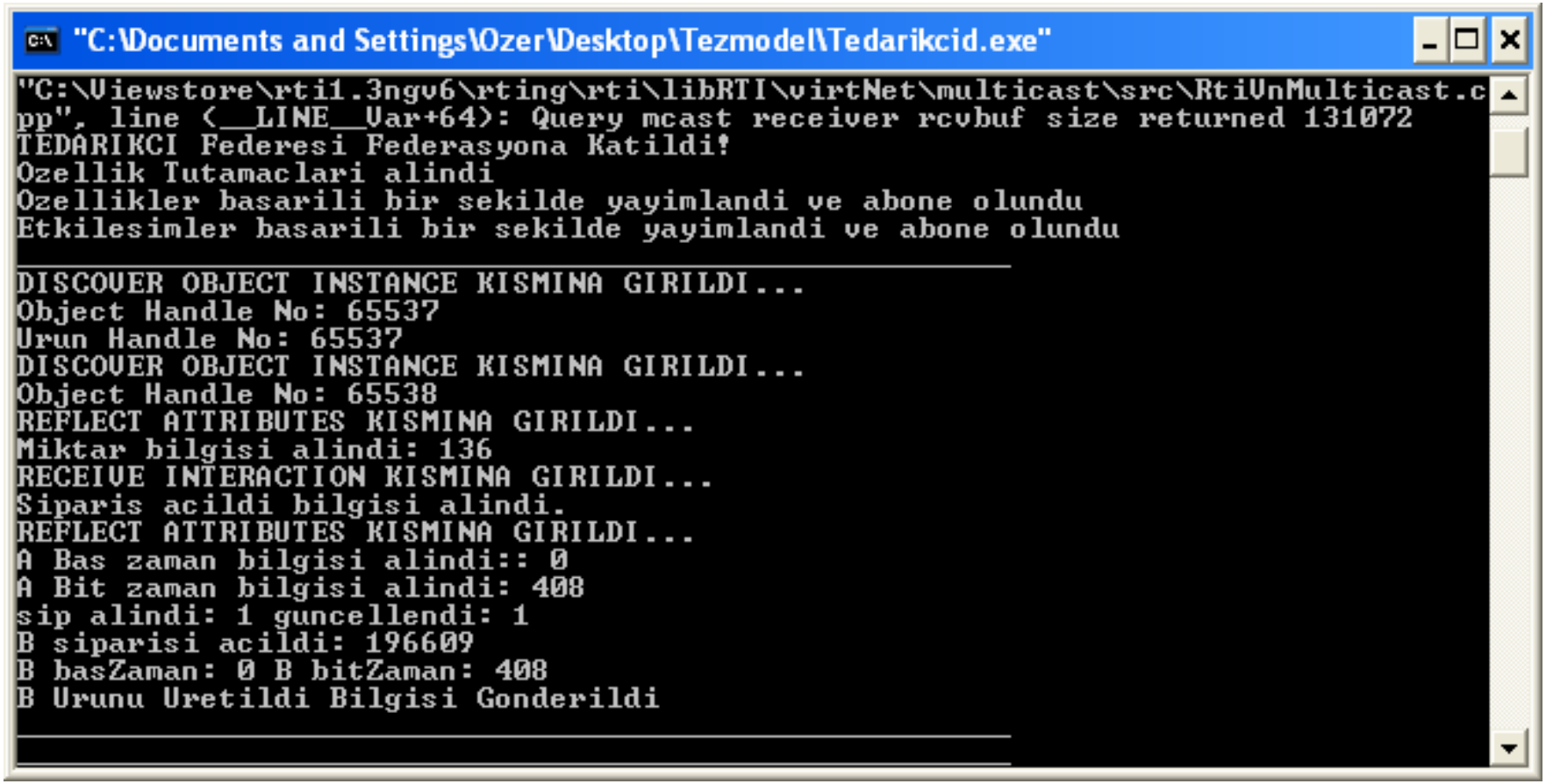
Bu bilgiler ışığında B alt ürünü üretilmiş, bunun da başlama ve bitiş zamanları tayin edilmiştir. Üretici federesine B alt ürününün üretildiği bilgisinin (etkileşiminin) gönderildiği de ekranda görülmektedir.

Üretici federesini çalıştırmaya devam ettiğimizde, B alt ürününün üretilmesi ile ilgili bilgilerin üretici federesine geldiği görülecektir. Bununla ilgili ekran görüntüsü Şekil 5.22'de görülmektedir. Şekilden de görüleceği gibi B alt ürünün başlama ve bitiş zamanları, B alt ürününün üretilme etkileşimi üretici federesine gelmiştir. Daha sonra A ve B alt ürünlerinin bitiş zamanlarının büyük olanına göre C ürününün üretimi de gerçekleştirilmiştir. Bununla ilgili bilgiler de ekrana yazdırılmıştır. En sonunda ise sipariş silinmiştir.

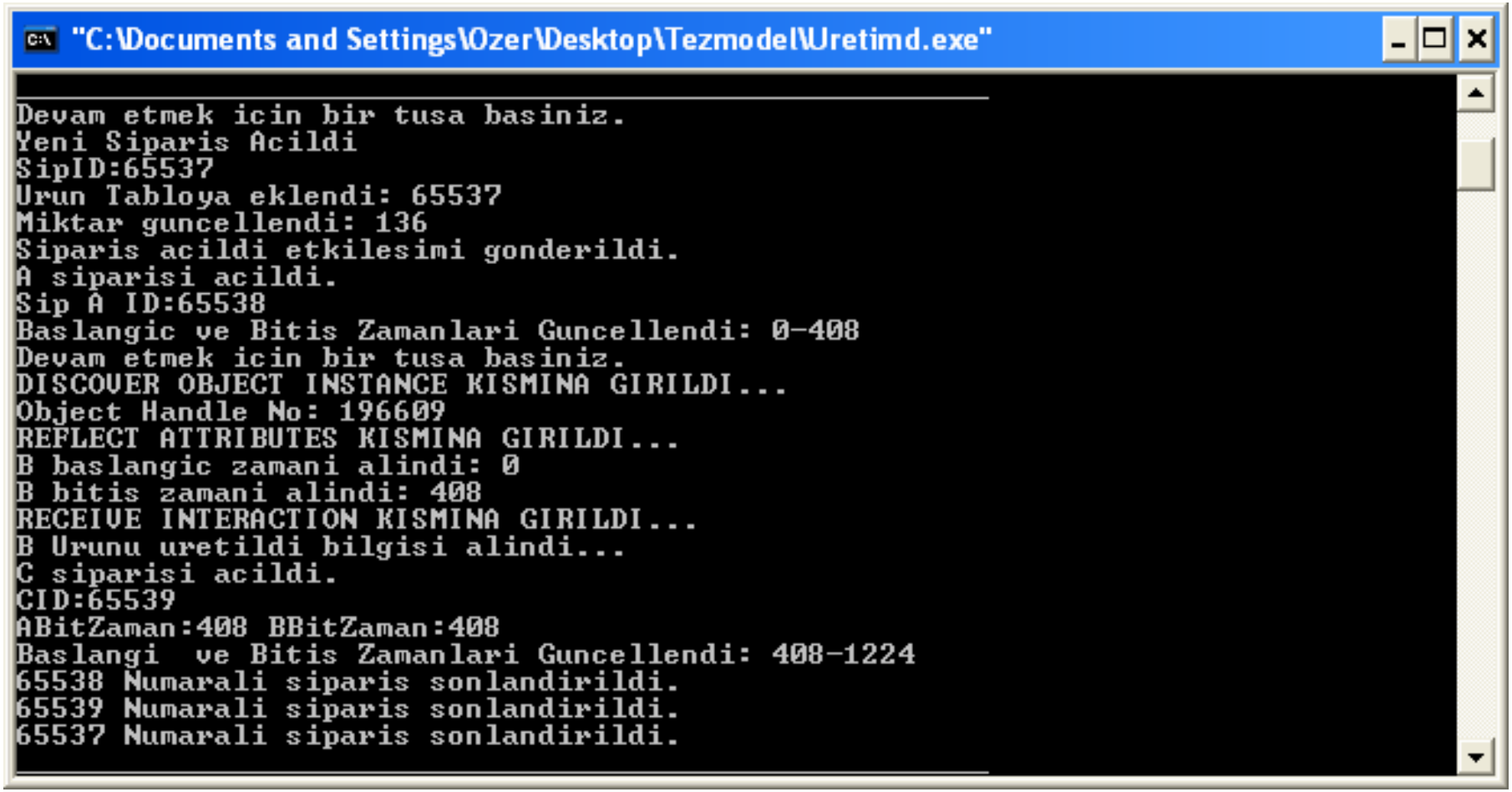
Buraya kadar söylenenler federasyonun bir adım çalıştırılmasıdır. Aynı işlemler yapılan programa göre on adım benzer şekilde çalıştırılacaktır. İki farklı bilgisayarda çalışabilen üretici ve tedarikçi fedelerinin ekran görüntülerinden de anlaşıldığı gibi benzetimler arası ekileşim sağlanmakta ve iki benzetim birlikte çalışabilmektedir.



Şekil 5.20. Üretici federesinin çalışmasının ara aşamada ekran görüntüsü



Şekil 5.21. Tedarikçi federesinin çalışmasının ekran görüntüsü



Şekil 5.22. Üretici federesinin bir adım çalışmasının ekran görüntüsü

Üretim federesinin yapısı ve işleyişi, tedarikçi federesinin yapısı ve işleyişi ile federasyonun çalıştırılması anlatılmıştır. Bu anlatılanlar ışığında federasyon örnek olarak on adım işletilmiştir. Bunun neticesinde elde edilen bilgiler sonuc.dat dosyasına kaydedilmekte ve program sonunda ekrana yazdırılmaktadır. **Tablo 5.4**, Üretim federasyonunun çalıştırılması sonucu elde edilen bilgileri göstermektedir.

Tablo 5.4. Üretim federasyonunun çalıştırılması sonucu elde edilen veri örnekleri

Sipariş No	**Miktar**	**A alt ürünü**		**B alt ürünü**		**C ürünü**	
		Başlama	**Bitiş**	**Başlama**	**Bitiş**	**Başlama**	**Bitiş**
65537	136	0	408	0	408	408	1224
65540	148	1224	1520	1224	1668	1668	2556
65543	134	2556	2958	2556	2824	2958	3762
65546	132	3762	4158	3762	4026	4158	4950
65549	133	4950	5349	4950	5349	5349	6147
65552	128	6147	6531	6147	6403	6531	7299
65555	126	7299	7551	7299	7551	7551	8307
65558	147	8307	8601	8307	8601	8601	9483
65561	148	9483	9927	9483	9779	9927	10815
65564	125	10815	11190	10815	11065	11190	11940

BÖLÜM 6. SONUÇLAR VE ÖNERİLER

6.1. Çalışmanın Özeti

Bu çalışmada HLA temelli dağıtık imalat benzetiminin gerekliliği, çeşitli dağıtık imalat sistemi örnekleri, HLA'nın dağıtık imalat benzetiminde nasıl kullanılacağı gibi konular ele alınmıştır. Örnek bir uygulama ile HLA temelli dağıtık imalat benzetimi geliştirilerek, her aşaması ayrıntılı bir şekilde açıklanmıştır.

Bu çalışmada, kısaca benzetim ve dağıtık benzetim konuları ele alınarak imalatta dağıtık benzetimi kullanımının önemi açıklanmıştır. Dağıtık benzetim mimarisi olan HLA oldukça ayrıntılı bir şekilde anlatılmıştır. İmalatta HLA'nın kullanılmasına yönelik örnek tasarımlar ve uygulamalı bir örnek sunulmuştur.

HLA, benzetimler arası karşılıklı etkileşimin sağlanması için verilerin tutulduğu nesnelerin standart bir şekilde tanımlanmasını gerektirmektedir. Nesne model şablonu denilen bu standart yapıda nesnelerin nasıl tanımlandığı da uygulamalı olarak anlatılmıştır. Benzetimde kullanılacak nesneler HLA'nın istediği yapıda tanımlandıktan sonra bunların uygulama geliştirme dili içerisinde kullanımı gerekmektedir. Verilen uygulamalı örnekte, nesnelerin kullanımı, verilerin yayımlanması ve verilere abone olunması, etkileşim gönderilmesi ve alınması konuları açıklanmıştır.

Bununla birlikte, HLA temelli dağıtık benzetim geliştirmek üzere, HLA'nın omurgası olan RTI'ın (çalışma anı altyapısı) bilgisayarlara kurulumu ve bilgisayarlarda birtakım düzenlemelerin yapılması gerekmektedir. Federasyon geliştirmeye yönelik bu hazırlık süreçleri de anlatılmıştır. Birden fazla bilgisayarda federasyonun çalıştırılması ve farklı bilgisayarlardaki federelerin federasyona bağlanma işlemleri de açıklanmıştır.

Çalışmanın sonunda ise, HLA temelli dağıtık imalat benzetimi geliştirmek üzere geliştirilen kaynak kodları ve kullanılan dosyalar ek olarak verilmiştir.

6.2. Çalışmadan Elde Edilen Sonuçlar

Çalışmanın katkıları ve elde edilen sonuçlar şu şekilde özetlenebilir:

- HLA'nın dağıtık imalat benzetimlerinde kullanılabileceği ve dağıtık imalat benzetiminin gerekliliği ortaya konmuştur.
- Birkaç örnek tasarım ile HLA'nın imalatın hangi alanlarında kullanılabileceğine yönelik fikirler verilmiştir. Verilen tasarım örneklerinden biri, üretim çizelgeleme ile bakım çizelgelemenin karşılıklı etkileşim neticesinde zamanlarının belirlenmesine yöneliktir. Üretim ve bakım çizelgeleme sistemleri, birbirini etkileyen iki sisteme güzel bir örnektir. İmalatta buna benzer birçok işlev söz konusudur. Dolayısıyla imalatta HLA kullanımının yararlı olacağı belirtilmiştir.
- Örnek bir senaryo üzerinden HLA temelli dağıtık bir imalat tasarlanıp geliştirilmiştir. Bu şekilde işlevsel olarak HLA'nın imalata yönelik nasıl kullanılabileceği pratik olarak gösterilmiştir. İmalat federasyonunun geliştirilmesi ile ilgili tüm bilgiler açıklanmıştır.
- Çalışan iki imalat benzetim modelinin karşılıklı işlemesi sağlanmıştır. Karşılıklı işleme HLA nesne özelliklerinin yayımlama ve abone olma yoluyla güncellenmesi veya etkileşimlerin gönderilip alınması şeklinde gerçekleşmektedir. Bu işlevler uygulamalı olarak gösterilmiştir.
- HLA nesne model şablonu (OMT) yapısında, imalat verilerinin iletiminde kullanılacak nesnelerin nasıl tanımlanacağı örneklendirilmiştir. HLA-OMT yapısıyla tanımlanan nesnelerin federasyona FED dosyası olarak dâhil edilmesi gerekmektedir. Tanımlanan nesnelerin FED dosyasından çekilerek programlama dili içerisinde ne şekilde kullanılacağı da gösterilmiştir.
- HLA'nın kullanılabilmesi için gerekli bir takım hazırlık işlemleri söz konusudur. Bu çalışmada, HLA'nın özelliklerini kullanabilmek için gerekli RTI yazılımının kurulumu, bilgisayarlarda yapılması gereken ayarlardan da bahsedilmiştir.

Federasyonun işletilmesi için gerekli, RTI ile birlikte gelen dosyalar ve işlevleri de açıklanmıştır.

- HLA, RTI ile dağıtık bir işletim sistemi gibi davranmakta ve benzetim modellerinin karşılıklı işlemesi için servisler sağlamaktadır. Ancak RTI'ın işlevlerini kullanabilmek için uygulama geliştirme ortamı (programlama dili) gereklidir. Bu çalışmada, RTI'ın servislerinin Visual C++'da nasıl kullanılabileceği gösterilmiştir.
- HLA federasyonunu çalıştırabilmek için hazırlanan dosyalar, geliştirilen kaynak kodları, çalışmanın sonunda ek olarak verilmiştir. Bu şekilde HLA'ya ilgi duyanlara ve HLA federasyonu geliştirmek isteyenlere yardımcı olmak amaçlanmıştır.
- Bu çalışma gerçekleştirilirken HLA'nın kullanım zorluğu da gözlenmiştir. Günümüzde HLA ile dağıtık imalat benzetim modelleri tasarlamak uzmanlık gerektirmektedir. Tasarımcıların ve uygulama geliştiricilerin iyi bir bilgisayar yazılım altyapısı olması gerekmektedir. İmalatta dağıtık benzetimin yaygınlaşamamasının en önemli sebeplerinden bir budur. Günümüzde klasik benzetimi bile kullanmaya vakit ve uzman bulamayan imalat şirketleri dağıtık benzetimi de rahatlıkla kullanamayacaktır. İmalat benzetimi yapmaya elverişli ticari paketler tekil benzetim modellerini geliştirmeye yardımcı olmaktadır, ancak bunlar henüz karşılıklı işleyen dağıtık imalat benzetim modelleri geliştirmede kullanılamamaktadır. Dağıtık imalat benzetiminin kullanılırlığını artırmak için ProModel, Arena gibi ticari benzetim paketlerinin HLA'yı desteklemesi ve dağıtık imalat benzetimine imkân vermesi beklenmektedir.

6.3. Gelecekte Yapılabilecek Çalışmalar

- Dağıtık imalat benzetiminde kullanılabilecek HLA temelli hazır bir benzetim yazılım paketi henüz maalesef yoktur (Taylor vd., 2006; McLean vd., 2005). Bu sebeple benzetim yazılım paketi ile HLA-RTI arasında bilgilerin dönüştürülmesi için bazı arayüzler kullanılmaktadır. Bu arayüzlere adaptör, çevirici gibi adlar verilmektedir. Kullanılan bu arayüzler RTI'dan gelen bilgiyi benzetim paketinin anlayacağı bir yapıya ve benzetim paketinden gelen bilgiyi ise HLA formatına dönüştürmektedir. Benzetim yazılımı ile HLA-RTI arasında kullanılacak arayüz

zeki etmen veya zeki bir arayüz olarak da tasarlanması mümkündür. Bu şekilde RTI ile benzetim yazılımının iletişiminin etkinliği artırılabilir.

- Zeki etmen kullanımı sadece benzetim modeli ile HLA-RTI arasında arayüz olarak düşünülmemelidir. RTI zaten dağıtık bir işletim sistemi gibi davranarak karşılıklı etkileşimi desteklemektedir. Zeki etmenlerin, RTI üzerinden etkileşim kuracak şekilde tasarlanması konusu geliştirilebilir. Burada HLA, zeki etmen tasarımını kolaylaştırıcı bir teknoloji olarak düşünülmelidir. Literatürde bu konuya yönelik bazı çalışmalara rastlamak mümkündür ve bunlar 3. bölümde verilmiştir. Ancak bu konu geliştirilmeye açık bir çalışma alanıdır.

- Benzetim modelleri dışındaki imalat yazılımlarının HLA ile bütünleştirilmesi konusunda kolaylaştırıcı çalışmalara da ihtiyaç görünmektedir. HLA federasyonlarına benzetim olmayan yazılımlar da teorik olarak dâhil edilebilmektedir. Ancak pratikte bu henüz çok kolay değildir.

- HLA, kendi standart yapısıyla tanımlanmış nesneler ile çalışmayı gerektirmektedir. Verilerin saklanması, güncellenmesi, iletilmesi nesne özelliklerinin değerleri ile gerçekleştirilmektedir. İmalata yönelik standart HLA-OMT yapısında nesneler oluşturulması düşünülmelidir. İmalat nesnelerini oluşturmayı kolaylaştırıcı araçların geliştirilmesi, dağıtık imalat benzetiminin kullanımını yaygınlaştırıcı etkiye sahip olacaktır. İmalat ile ilgili nesnelerin tutulduğu veri ambarları da oluşturulabilmelidir. Benzetim için gerekli imalat nesnesi buradan seçilebilmeli ve gerekirse modifiye edilebilmelidir.

- Gerçek imalat ortamlarında kullanılmakta olan birçok veri ambarları ve veri tabanları mevcuttur. HLA'nın veri tabanları ile ilişkilendirilmesi ve bütünleştirilmesi çalışmalarına da ihtiyaç hissedilmektedir. Veri tabanlarında saklanan verilerin HLA federasyonuna aktarılması veya HLA federasyonunda oluşturulan ve güncellenen verilerin veritabanına kaydedilmesi mümkün olmalıdır. Veri tabanlarından alınan verilerin HLA'nın istediği standarda ve HLA nesnelerindeki verilerin veritabanının gerektirdiği yapıya dönüştürülmesi için arayüzler geliştirilmelidir. Ya da bu konuda başka çözümler bulunmalıdır.

- Dağıtık imalat benzetimi geliştirmeyi kolaylaştırmak bakımından, dağıtık benzetimi destekleyen ticari paketler geliştirilmelidir. Dağıtık bilgi işleme ve dağıtık benzetim uzmanlık bilgisi gerektirmekte ve genellikle bilgisayar ve yazılım mühendislerinin ilgi alanına girmektedir. İmalat süreçlerini analiz edecek kişilerde ise bu uzmanlık bilgisinin olması genellikle mümkün görülmemektedir. Bu sebeple HLA'yı standart olarak içeren dağıtık benzetime elverişli hazır paketler geliştirilmelidir.

- Tedarik zincirleri dağıtık imalat modelleri olarak ele alınabilmektedir. Tedarik zincirlerinin modellenmesi ve etkilerinin analiz edilmesinde HLA kullanılmaya başlanmıştır. HLA temelli tedarik zinciri modelleri de çalışılmaya elverişli alanlardandır. Lokal değişkenliklerin zincirin tamamına etkisinin analiz edilmesi ve benzeri konular için, tedarik zinciri öğelerinin dağıtık olarak karşılıklı işleyen HLA temelli benzetim şeklinde tasarlanması çalışılmalıdır.

- Tedarik zinciri modelleri, sanal fabrika ortamları veya dijital fabrikalar için bir aşama olarak düşünülebilir. Sanal imalat, dağıtık sanal ortamlar gibi konularda fikirler üretilmektedir. Sanal imalat ortamlarında etkileşim imkanı sağlayan teknoloji olarak HLA'nın kullanılması konusunda çalışmalar yapılabilir.

- Sistem karmaşıklaştıkça etkileşimi sağlayan veri boyutu da artmaktadır. Bunun sonucunda çeşitli veri dağıtım yöntemleri ve zaman yönetimi algoritmaları konusunda çalışmalar yapılmaktadır. Veri dağıtımının ve zaman yönetiminin etkinliğini artırmaya yönelik çalışmalara da ihtiyaç vardır.

- HLA'nın benzetim modellerinin senkronizasyonuna katkısı büyüktür. İçerisinde çeşitli zaman yönetim politikaları barındırmaktadır. Bununla birlikte Wongwirat ve Ohara (2006), dağıtık sanal ortamlarda birbirleriyle çatışan ve çatışmayan faaliyetlerin uzlaşmacı senkronizasyonunu ele almışlardır. Yaptıkları çalışmada zaman yönetimi tekniği olarak HLA'yı da incelemişlerdir. Çalışmalarının sonucunda HLA'nın zaman yönetimini yumuşak senkronizasyon olarak gördüklerini ve sadece çatışmayan faaliyetlerde kullanılabileceğini ifade

etmişlerdir. Çatışan faaliyetlerde zamanı yönetmek için ise kendileri farklı bir yöntem önermişlerdir. Önerdikleri bu uzlaşmacı senkronizasyon mekanizmasının gelecekte HLA'da da kullanılabileceğini ummaktadırlar. Dolayısıyla HLA'nın sanal dağıtık ortamlarda ve özellikle sanal imalat ortamında kullanılmak istenmesi durumunda bu durum göz önünde bulundurulmalı ve zaman yönetimi konusu geliştirilmelidir.

– Karmaşık sistemlerde farklı bilgisayarlarda çalışan alt sistemlerin bilgi işlem yükleri değişkenlik göstermektedir. İşlem yükü fazla olan bilgisayarlardan işlem yükü daha az olan bilgisayarlara benzetim modelinin kolonlanıp gönderilmesi ve diğer bilgisayarda çalıştırılmasına yönelik çalışmalar da başlamıştır. Bu konu da bilgisayar mühendisleri açısından geliştirilmeye açık bir alandır.

KAYNAKLAR

AEgis, (1998), "High Level Architecture (HLA) Object Model Development Tool (OMDT)", version 1.3, Developer: AEgis Research, Sponsor: DMSO

ALLAOUI, H., ARTIBA, A., (2004), "Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints", Computers & Industrial Engineering, 47 431–450

ALSP, (2006), http://ms.ie.org/alsp/ (23.06.2006 tarihinde erişilmiştir)

BANDINELLI, R., RAPACCINI, M., TUCCI, M., VISINTIN, F., (2006), "Using simulation for supply chain analysis: reviewing and proposing distributed simulation frameworks", Production Planning & Control, Vol. 17, No. 2, March, 167–175.

BANKS, J., CARSON II, J. S., NELSON, B. L., NICOL, D. M., (2001), "Discerete-Event System Simulation", 3rd ed., Prentice-Hall

BAYAROU, K. M., KONAN, K. D. D., SZCZERBICKA, H., (2002), "Exploring Impact of Time Management Services on HLA-Based Petri Nets Simulation Engine", Simulation Practice and Theory 9 143–166

BORSHCHEV, A., KARPOV, Y., KHARITONOV, V., (2002), "Distributed simulation of hybrid systems with AnyLogic and HLA", Future Generation Computer Systems 18 829–839

BOUKERCHE, A., DZERMAJKO, C., LU, K., (2006), "Alternative approaches to multicast group management in large-scale distributed interactive simulation systems", Future Generation Computer Systems 22 755–763

BOUKERCHE, A., VE DZERMAJKO, C., (2004), "Performance evaluation of Data Distribution Management strategies", Concurrency and Computation: Practice And Experience, 16:1545–1573

BOUKERCHE, A., VE ROY, A., (2002), "Dynamic Grid-Based Approach to Data Distribution Management", Journal of Parallel and Distributed Computing 62, 366–392

CAI, W., YUAN, Z., LOW, M.Y.H., TURNER, S.J., (2005), "Federate migration in HLA-based simulation", Future Generation Computer Systems 21 87–95.

CALPIN, J. A., SALISBURY, M. R., VITKEVIHC, JR., WOODWARD, D. R., (2001), "Extending High Level Arhitecture Paradigm to Economic Simulation", Computational Economics, Jun, 17, p141-154

CANAZZI, D., (1999), "yaRTI, a ADA 95 HLA Run-Time Infrastructure", Ada-Europe'99 June 7-11, Santander, Spain

CASSADY, C.R., KUTANOĞLU, E., (2003), "Minimizing job tardiness using integrated preventive maintenance planning and production scheduling", IEE Transactions, 35, 503-513

CENGIZ, Z.S., VE OĞUZTÜZÜN, H., (2002), "A COM Component Set for HLA Interface Specification", Fall Simulation Interoperability Workshop, Orlando, Florida, September

CHANDRA, C., KUMAR, S., (2001), "Enterprise architectural framework for supply chain integration", Industrial Management and Data Systems, 101(6).

CHEN, D., TURNER S. J., GAN, B. P., CAI, W., WEI, J., JULKA, N., (2003), "Alternative Solutions for Distributed Simulation Cloning", Simulation, Vol. 79, No. 5-6, 299-315

CHEN, D., TURNER S. J., CAI, W., GAN, B. P.,LOW, M. Y. H., (2005), "Algorithms for HLA-Based Distributed Simulation Cloning", ACM Transactions on Modeling and Computer Simulation (TOMACS), Volume 15 , Issue 4 (October), 316 – 345.

CHONG, C.S, LENDERMANN, P., GAN, B.P., DUARTE, M.B., FOWLER J.W., CALLARMAN T.E., (2004), "Analysis of a Semiconductor Supply Chain in a Distributed Simulation Testbed", Winter Simulation Conference, 5-8 December, Washington, D.C

COLEMAN, D. S., (2001), "PC Gaming and Simulation Supports Training", United States Naval Enstitute, Proceedings, Annapolis: Feb 2001, Vol. 127, Iss. 2, pg 73, 3 pgs.

COX, A., WOOD, D.D., (1996), "Design and implementation of the BDS-D/HLA gateway", in: Proceedings of the 18th Interservice/Industry Training Systems and Education Conference, Orlando, FL, December

COX, A., WOOD, D.D., PETTY, M.D., JUGE, K.A., (1996), "Integrating DIS and SIMNET into HLA with a gateway", in: Proceedings of the 15th DIS Workshop on Standards for the Interoperability of Defense Simulations, Orlando, FL, September, pp. 517-525

CUTTING-DECELLE A. F., DAS B. P., YOUNG R. I., CASE K., RAHIMIFARD S., ANUMBA C. J., BOUCHLAGHEM N. M., (2006), "Building supply chain communication systems: a review of methods and techniques", Data Science Journal, Volume 5, 05 June, 29-51

DAHMANN, J.S., CALVIN, J. O., WEATHERLY, R. M., (1999a), "A reusable architecture for simulations", Association for Computing Machinery. Communications of the ACM; September; 42, 9; Academic Research Library pg. 79

DAHMANN, J.S., SALISBURY, M., TURRELL, C., BARRY, P., BLEMBERG, P., (1999b), "HLA and Beyond: Interoperability Challenges", Fall Simulation Interoperability Workshop, Orlando, Florida, September

DAHMANN, J.S., FUJIMOTO, R. M., WEATHERLY, R. M., (1998), "The DoD High Level Arhitecture: An Update", Proceedings of the Winter Simulation Conference, 13-16 December, Washington DC, Eds.: D.J. Medeiros, E. F. Watson, J. S. Carson and M. S. Manivannan, p797-804

DAS, S.K., VE REYES, A.A., (2002), "An Approach to Integrating HLA Federations and Genetic Algorithms to Support Automatic Design Evaluation for Multi-Agent Systems", Simulation Practice and Theory 9, 167–192.

DATAR, M.M., (2000), "Enterprise Simulation: Framework for a Strategic Application", Winter Simulation Conference, 10-13 December, Orlando, FL.

DMSO, (1995), "Department of Defence Modeling and Simulation Master Plan", Defence Modeling and Simulation Office, U.S. Department of Defence, U.S. Governmet Printing Office.

DMSO, (1999), "High Level Architecture Federation Development and Execution Process (FEDEP) Model", Version1.5, December 8, 1999

DMSO, (2002a), "RTI 1.3-Next Generation Programmer's Guide", Version 6, Department of Defence Modeling and Simulation Office

DMSO, (2002b), "DMSO RTI Commercialization Announcement", DMSO Software Distribution Center, www.dmso.mil/public/library/projects/hla/rti/announcement.php (06.06.2006 tarihinde erişilmiştir)

DMSO, (2006), Amerikan Savunma Bakanlığı Savunma Modelleme ve Benzetim Birimi (Defence Modeling and Simulation Office) resmi internet sitesi: www.dmso.mil (06.06.2006 tarihinde erişilmiştir.)

ERKUT, H., (1992), "Yönetimde Benzetim Yaklaşımı", İrfan Yayınevi.

FUJIMOTO, R. M., (2000), "Parallel and Distributed Simulation Systems", Lecture Notes, http://www.cc.gatech.edu/classes/AY2000/cs4230_spring/ (22.06.2006'da erişilmiştir.)

FUJIMOTO, R. M., (2001), "Parallel and Distributed Simulation Systems", Winter Simulation Conference, 9-12 December, Arlington, VA, ed. B.A. Peters, J.S. Smith, D.J. Medeiros, and M.W. Rohrer,

GAN, B. P., LIU, L., JAIN, S., TURNER, S. J., CAI, W., HSU, W. J., (2000), "Distributed supply chain simulation across enterprise boundaries", In Proceedings of the Winter Simulation Conference, 10-13 December, Orlando, FL, ed. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 1245-1251

GÖKTÜRK, E. VE POLAT, F., (2003), "Implementing Agent Communication for a Multi-Agent Simulation Infrastructure on HLA", In: Proceedings of the 18th International Symposium on Computer and Information Sciences (ISCIS'03). Volume 2869 of Lecture Notes in Computer Science (LNCS). Springer (2003) 619-626

GRAVES, G.H., LEE C.Y., (1999), "Scheduling maintenance and semiresumable jobs on a single machine", Naval Research Logistics 46 845–863.

HIBINO, H., FUKUDA, Y., YURA, Y., MITSUYUKI, K., KANEDA, K., (2002), "Manufacturing Adapter of Distributed Simulation Systems Using HLA", Winter Simulation Conference, 8-11 December, San Diego, CA

HIBINO, H. VE FUKUDA, Y., (2006), "A User Support System for Manufacturing System Design Using Distributed Simulation", Production Planning & Control, Vol. 17, No. 2, March 2006, 128–142.

HIBINO H. INUKAI, T., FUKUDA, Y., (2006), "Efficient Manufacturing System Implementation Based on Combination Between Real and Virtual Factory", International Journal of Production Research, Vol. 44, Nos. 18–19, 15 September–1 October 2006, 3897–3915.

Hilderbrand, T. N., (2006), "Migrating Legacy Simulations to the High Level Architecture (HLA)", GeorgiaTech Research Institute, Information Technology and Telecommunications Laboratory, http://gtri.gatech.edu/ittl/csit/proj_legacysim.html (03.05.2006 tarihinde erişilmiştir.)

HLA-CSPIF, (2003), "HLA-CSPIF Discussion Document Entity Transfer Specification", Version 1.1.1, 25th August, http://www.cspi-pdg.org/content/ets1_1_1.doc. (20 Şubat 2006 tarihinde erişilmiştir)

HUANG, J.Y., TUNG, M.C., WANG, K.M., LEE, M.C., (2005), "Smart Time Management - the unified time synchronization interface for the distributed simulation", Computer Standards & Interfaces 27, 149–161.

IEEE, (1995a), "Distributed Interactive Simulation – Application Protocols", Std. 1278.1, 21 September

IEEE, (1995b), "Standard for Distributed Interactive Simulation - Communication Services and Profiles", Std. 1278.2, 21 September

IEEE, (2000), "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-Framework and Rules", Std 1516-2000, The Institute of

Electrical and Electronics Engineers, Inc., Approved 21 September 2000, Published 11 December 2000.

IEEE, (2001a), "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-Federate Interface Specification", Std 1516.1-2000, The Institute of Electrical and Electronics Engineers, Inc., Approved 21 September 2000, Published 9 March 2001.

IEEE, (2001b), "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-Object Model Template (OMT) Specification", Std 1516.2-2000, The Institute of Electrical and Electronics Engineers, Inc., Approved 21 September 2000, Published 9 March 2001.

IEEE, (2003), "IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP)", Std 1516.3-2003, The Institute of Electrical and Electronics Engineers, Inc. Computer Sociaty, Approved 20 March 2003, Published 23 Appril 2003.

IMS, (2006), www.ims.org, Intelligent Manufacturing System programı resmi internet sitesi (01 Mart 2006 tarihinde erişilmiştir.)

JIAN, J., ZHANG, H., GUO, B., WNAG, K., CHEN, D., (2004), "HLA-Based Collaborative Simulation Platform for Complex Product Design", The 8th International Conference on Computer Supported Cooperative Work in Design.

JTC: Joint Training Confederation, (2006), "What is ALSP?", http://ms.ie.org/alsp/ (03.05.2006 tarihinde erişilmiştir.)

KLEIN, U., (2000), "Simulation-Based Distributed Systems: Serving Multiple Purposes Through Composition of Components", Safety Science 35 (2000) 29-39

KLEIN, U., SCHULZE, T., STRAßBURGER, S., (1998), "Traffic Simulation Based on the High Level Architecture", Winter Simulation Conference, 13-16 December, Washington DC.

KUBAT C., UYGUN Ö., (2007), "HLA based supply chain management for Sakarya automotive suppliers", 2007 I*PROMS Virtual Conference.

KUHL, F., WEATHERLY, R., DAHMANN, J., (2000), "Creating Computer Simulation Systems: An Introduction to the High Level Architecture", Prentice Hall PTR

LAW, A. M., KELTON, W.D., (1991), "Simulation Modelling and Analysis", 2nd ed., McGraw-Hill

LAW, A. M., KELTON, W. D., (2000), "Simulation Modelling and Analysis", 3rd ed., McGraw-Hill.

LEES, M., LOGAN, B., THEODOROPOULOS, G.K., (2006), "Agents, Games and HLA", Simulation Modelling Practice and Theory, Vol. 14, Issue 6, August, p752-767.

LEES, M., LOGAN, B., THEODOROPOULOS, G.K., (2007), "Distributed Simulation of Agent-Based Systems with HLA", ACM Transactions on Modeling & Computer Simulation; July, Vol. 17 Issue 3, p1-25.

LENDERMANN, P., JULKA, N., GAN, B., P., CHEN, D., MCGINNIS, L., F., MCGINNIS, J., P., (2003), "Distributed Supply Chain Simulation as a Decision Support Tool for the Semiconductor Industry", Simulation, Vol. 79, Issue 3, March, p126-138.

LENDERMANN, P., GAN, B.P., LOH, Y.L., TAN, H.K., LIEU, S.K., MCGINNIS, L.F., FOWLER, J.W., (2004), "Analysis of a Borderless Fab Scenario in a Distributed Simulation Testbed", Winter Simulation Conference, 5-8 December, Washington DC

LENDERMANN, P., LOW, M.Y.H., GAN, B.P., JULKA, N., CHAN, L.P., LEE, L.H., TAYLOR, S.J.E., TURNER, S.J., CAI, W., WANG, X., HUNG, T., MCGINNIS, L.F., BUCKLEY, S., (2005), "An Integrated and Adaptive Decisn-Support Framework For High-Tech Manufacturing And Service Networks", Proceedings of the Winter Simulation Conference, 4-7 December, Orlando, FL

LI, N., PENG, X.Y., ZHANG, M.H., WANG, M., GONG, G.H., (2006), "Multimedia Communication over HLA/RTI", Simulation Modelling Practice and Theory 14 161–176

LINN, R.J., CHEN, C.S., LOZAN, J.A., (2002), "Development of Distributed Simulation Model for the Transporter Entity in a Supply Chain Proses", Proceedings of the Winter Simulation Conference, 8-11 December, San Diego, CA

LIYU, T., CHONGCHENG, C., HONGYU, H., KAIHUI, L., (2006), "Research on HLA-based Forest Fire Fighting Simulation System", Poster presented at the 9th AGILE Conference on Geographic Information Science, Visegrád, Hungary.

LU, T., HSU, C., (2007), "Mobile Agents For Information Retrieval in Hybrid Simulation Environment", Computer Applications 30 244–264

MAAMAR, Z., (2003), "Design of a Simulation Environment Based on Software Agents, and The High Level Architecture", Information and Software Technology 45 137–148

MaK, (2006), MaK Technologies, www.mak.com (06.06.2006 tarihinde erişilmiştir)

MARKLAND, R., VICKERY, S. K., DAVIS, R. A., (1995), "Operations management", West Publishing, New York.

MCLEAN, C., LEONG, S., (2001a), "The Expanding Role of Simulation in Future Manufacturing", Winter Simulation Conference, 9-12 December, Arlington, VA

MCLEAN, C., LEONG, S., (2001b), "The Role of Simulation in Strategic Manufacturing", In Proceedings of the International Working Conference on Strategic Manufacturing,. Working Group on Integrated Production.

MCLEAN, C., RIDDICK, F., (2000a), "The IMS Mission Architecture for Distributed Manufacturing Simulation", Winter Simulation Conference, 10-13 December, Orlando, FL

MCLEAN, C., RIDDICK, F., (2000b), "Integration of Manufacturing Simulations Using HLA", AST 2000 Conference, Washington, DC, April

MCLEAN, C., RIDDICK, F., LEE, Y. T., (2005), "An Architecture and Interfaces for Distributed Manufacturing Simulation", Simulation, Vol. 81, Issue 1, January, p.15-32.

MCLOUGHLIN, M., HEAVEY, C., (2001), "Component-Based Simulation in Manufacturing Systems", Computers & Industrial Engineering International Conferences.

MEOSS, (2006), "Part1: Intoduction to the Hihg Level Architecture", McLeod Enstitute Of Simulation Sciences http://www.ecst.csuchico.edu (06.06.2006 tarihinde erişilmiştir)

MERTINS, K., RABE, M., JAKEL, F.W., (2005), "Distributed Modelling and Simulation of Supply Chains", International Journal of Computer Integrated Manufacturing, Vol. 18, No. 5, July–August, 342 – 349.

MILER, D. C., THORPE, J. A., (1995), "SIMNET: The Advent of Simulator Networking", Proceedings of the IEEE, 83(8), 1114-1123.

MISSION, (1998), "Intelligent Manufacturing System (IMS) Project Proposal: Modelling and Simulation Environments for Design, Planning and Operation of Globally Distributed Enterprises (MISSION)", Version 3.3.

MISSION, (2001a), "Deliverable D24 Final Report from MISSION Project", November 12.

MISSION, (2001b), "Deliverable D13-Document 5: Exchange Objects and Federate Configuration File Structure", February 28.

MITRE, (2006), The MITRE Corporation resmi internet sitesi, http://www.mitre.org/ (23.06.2006 tarihinde erişilmiştir)

MORSE, K. L., LIGHTNER, M., LITTLE, R., LUTZ, B., SCRUDDER, R,. (2006) "Enabling Simulation Interoperability", Computer, January, Vol. 39 Issue 1, p115-117

MORSE, K.L., ZYDA, M., (2002), "Multicast Grouping for Data Distribution Management", Simulation Practice and Theory 9, 121–141

MÖNCH, L., ROSE, O., STURM, R., (2003), "A Simulation Framework for the Performance Assessment of Shop-Floor Control Systems", Simulation, Vol. 79, Issue 3, March.

PAGE, E.H., OPPER, J.M., (2000), "Investigating the Application of Web-Based Simulation Principles Within the Architecture for a Next-Generation Computer Generated Forces Model", Future Generation Computer Systems 17 159-169

PARSAEI, H. R., KOLI, S., HANLEY, T. R., (1997), "Manufacturing Decision Support Systems", Chapman & Hall,

PETTY M.D., WINDYGA P.S., (1999), "A High-Level Architecture-Based Medical Simulation System", Simulation, San Diego: November, Vol. 73, Iss. 5; p281-286

PETTY M.D., (2002), "Comparing High Level Architecture Data Distribution Management Specifications 1.3 and 1516", Simulation Practice and Theory 9, 95–119

PETTY M.D., MORSE, K.L., (2004), "The Computational Complexity of the High Level Architecture Data Distribution Management Matching and Connecting Processes", Simulation Modelling Practice and Theory 12, 217–237

PITCH, (2006a), "High Level Architecture for Beginners", Pitch Technologies, http://www.pitch.se/hla/hlaforbeginners.asp (15.05.2006 tarihinde erişilmiştir.)

PITCH, (2006b), "Differences between HLA 1.3 and HLA 1516", Pitch Technologies, http://www.pitch.se/hla/abouthla1516.asp (15.05.2006 tarihinde erişilmiştir.)

PITCH, (2006c), "pRTI 1516 User's Guide", Pitch Technologies, http://www.pitch.se/prti1516/files/pRTI1516_Users_Guide.pdf, (15.05.2006 tarihinde erişilmiştir.)

PRICE, D.J., WALSH , S.P., NAHAVANDI, S., (2004), "Unifying Manufacturing Simulation Models Using HLA", 2nd IEEE International Conference on Industrial Informatics.

RABELO, L., HELAL, M., JONES, A., MIN, J., SON, Y.J., DESHMUKH, A., (2003), "A Hybrid Approach To Manufacturing Enterprise Simulation", Proceedings of the Winter Simulation Conference, 7-10 December, New Orleans, LA

RABELO, L., HELAL, M., JONES, A., MIN, J., SON, Y.J., DESHMUKH, A., (2005), "Enterprise Simulation: A Hybrid System Approach", International Journal of Computer Integrated Manufacturing, Vol. 18, No. 6, September 2005, 498 – 508

REID, M. R., (2000), "An Evalution of the High Level Architecture (HLA) as a Framework for NASA Modeling and Simulation", Presented at the 25th NASA Software Engineering Workshop, Goddard Space Flight Center, Greenbelt, Maryland, November 30, 2000

SAAD, S.M., PERERA, T., WICKRAMARACHCHI, R., (2003), "Simulation of Distributed Manufacturing Enterprises: A New Approach", Winter Simulation Conference, 7-10 December, New Orleans, LA

SCHULZE, T., STRAßBURGER, S., KLEIN, U., (1999a), "Migration of HLA into Civil Domains: Solutions and Prototypes for Transportation Applications", Simulation, Vol. 73, No. 5, p296-303, November.

SCHULZE, T., STRAßBURGER, S., KLEIN, U., (1999b), "On-Line Data Processing in a Civil Transportation Federation", 1999 Fall Simulation Interoperability Workshop, September

SCHUMANN, M., BLUEMEL, E., SCHULZE T., STRAßBURGER, S., RITTER, K.,C., (1998), "Using HLA for Factory Simulation", 1998 Fall Simulation Interoperability Workshop. September 1998. Orlando, Florida, USA.

SHAW, M., GARLAN, D., (1996), "Software Architecture: Perspectives on An Emerging Dicipline", Prentice Hall.

SHEN, X., RADAKRISHNAN, T., GEORGANAS, N.D., (2002), "v COM: Electronic commerce in a collaborative virtual world", Electronic Commerce Research and Applications 1 (2002) 281–300

SIMCHI-LEVI D, KAMINSKY P, SIMCHI-LEVI E. (2000), "Designing and managing the supply chain: concepts, strategies, and case studies", Irwin McGraw-Hill.

SISO, "What is HLA", Simulation Interoperability Standards Organization, http://www.sisostds.org (15.05.2006 tarihinde erişilmiştir.)

SISO, Simulation Interoperability Standards Organization, www.sisostds.org, (23.06.2006 tarihinde erişilmiştir)

SLOAN, T.W., SHANTHIKUMAR, J.G., (2000), "Combined product and maintenance scheduling for a multiple-product, single-machine production system", Production and Operations Management; 9(4):379–99.

SORTRAKUL, N., NACHTMANN, H.L., CASSADY, C.R., (2005), "Genetic algorithms for integrated preventive maintenance planning and production scheduling for a single machine", Computers in Industry 56 161–168

STRASSBURGER, S., SCHMIDGALL, G., HAASIS, S., (2003), "Distributed Manufacturing Simulation as an Enabling Technology for the Digital Factory", Journal of Advanced Manufacturing Systems, Vol. 2, No. 1, 111-126

UYGUN O., KUBAT C., OZTEMEL E., (2006a), "Scenario based distributed manufacturing simulation using hla technologies", In Proceedings of 5th International Symposium on Intelligent Manufacturing Systems, May 29-31, 908-920

UYGUN O., SIMSIR F., GUNDOGAR E., KUBAT C., (2006b), "HLA Based Distributed Simulation Model for Integrated Maintenance and Production Scheduling System in Textile Industry", In Proceedings of 5th International Symposium on Intelligent Manufacturing Systems, May 29-31, 921-932

TAYLOR, S.J.E., (2003), "HLA-CSPIF: The High Level Architecture COTS Simulation Package Interoperability Forum", In Proceeding Fall Simulation Interoperability Workshop, Orlando, FL,.

TAYLOR, S.J.E., BOHLI, L., WANG, X., TURNER, S.J., (2005), "Investigating Distributed Simulation at The Ford Motor Company", Ninth IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'05)

TAYLOR, S.J.E., SURDA, R., JANAHAN, T., TAN, G., LADBROOK, J., (2002) "GRIDS-SCF: An Infrastructure for distributed Supply Chain Simulation", Simulation-Transactions Of The Society For Modeling And Simulation International 78 (5): 312-320 May 2002

TAYLOR, S.J.E., WANG, X., TURNER, S.J., LOW, M.Y.H., (2006), "Integrating Heterogeneous Distributed COTS Discrete-Event Simulation Packages: An Emerging Standards-Based Approach", IEEE Transactions on Systems, Man, and Cybernetıcs-Part A: Aystems and Humans, Vol. 36, No. 1, January.

TERZI, S., CAVALIERY, S., (2004), "Simulation in The Supply Chain Context: A Survey", Computers in Industry 53 (2004) 3–16

TOFT, P., (1999), "How to Become an HLA Guru in a Short(er) Time", COT/6-6-V1.0, Center for Object Technology, DMI, 1998-99

TRCKA (RADOSEVIC), M., HENSEN, J.L.M., WIJSMAN, A.J.Th.M., (2006), "Distributed Building Performance Simulation-A Novel Approach to Overcome Legacy Code Limitations", HVAC&R Research, Vol. 12, No.3

VENKATESWARAN, J., SON, Y.J., JONES, A., (2004), "Hierarchical Production Planning Using A Hybrid System Dynamic-Discrete Event Simulation Architecture", Winter Simulation Conference, 5-8 December, Washington, D.C.

VENKATESWARAN, J. VE SON, Y.J., (2005), "Hybrid System Dynamic-Discrete Event Simulation-Based Arcbitecture for Hierarchical Production Planning", International Journal of Pruduclion Research, Vol. 43. No. 20, 15 October 2005, 4397-4429.

VRIJHOEF R. VE KOSLEKA L., (1999), "Roles of supply chain management in construction", IGLC-7.

XIAOXIA, S., QIUHAI, Z., (2003), "The Introduction On High Level Architecture (HLA) and Run-Time Infrastructure (RTI), SICE Annual Conference in Fukui, August 4-6, 2003, Fukui University, Japan

WILCOX, P. A., BURGER, A. G., HOARE, P., (2000), "Advanced Distributed Simulation: A Review of Developments and Their Implication for Data Collection and Analysis", Simulation Practice and Theory, 8, 201-232

WONGWIRAT, O., OHARA, S., (2006), "Performance Evaluation of Compromised Synchronization Control Mechanism for Distributed Virtual Environment (DVE)", Virtual Reality (2006) 9: 1–16.

WOOD, D.D., COX, A., PETTY, M.D., (1997), "An HLA gateway for DIS application", in: Proceedings of the 19th Interservice/Industry Training Systems and Education Conference

EKLER

EK A: uretim.cpp Dosyası

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include "Urun.h"
#include <time.h>
#include "rti_arayuz.h"
#include "UrFederateAmbassador.h"
#include <RTI.hh>
#include <iostream.h>
#include <fstream.h>
using std::endl;

void main(int argc,char *argv[])
{
        int i, sipID, AID, CID;
        Urun *obj;

        srand(time(NULL));

        FederasyonaKatil();
        OzellikTutamaciniAl();
        OzellikleriYayimlaAboneol();
        EtkilesimleriYayimlaAboneol();

   for (i=0; i<10; i++)
   {
        cout<<"_____"<<endl;
                obj -> SetCUret(0);
                cout<<"Devam etmek icin bir tusa basiniz."<<endl;getch();

                sipID=YeniSiparis();
                MiktarGuncelle(sipID);

                SipAcildiEtkilesimiGonder();

                AID=SiparisA();
```

```
                AUrunuUret(AID);

                cout<<"Devam etmek icin bir tusa basiniz."<<endl;getch();

                while (!Urun::CUret())
                {
                        TickRTI();
                }

                CID = SiparisC();
                CUrunuUret (CID);

                SiparisSil(AID);
                SiparisSil(CID);
                SiparisSil(sipID);

                DosyayaYaz();

        cout<<"_____________________________________________________"<<endl;
    }
        DosyadanListele();
        ofstream dosyayayaz ("sonuc.dat", ios::app);
        dosyayayaz <<endl;

    FederasyondanAyril();
}
```

EK B: rti_arayuz.cpp Dosyası

```
#include <stdio.h>
#include <conio.h>
#include <sys/types.h>
#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>
#include <stdlib.h>
#include <math.h>
#include <winsock2.h>
#include <process.h>
#include <windows.h>
#define getpid _getpid
#include <RTI.hh>
#include <fedtime.hh>
#include "rti_arayuz.h"
#include "UrFederateAmbassador.h"
#include "Urun.h"
using std::endl;

extern UrunTable objects;


RTI::RTIambassador    rtiAmb;      // RTI Ambassadoru
FedAmb                fedAmb;      // Federe Ambassadoru

int fedZamani; // Federesyonun mevcut zamanı

RTI::ObjectClassHandle urunClassID    = 0;
RTI::ObjectClassHandle urunAClassID   = 0;
RTI::ObjectClassHandle urunBClassID   = 0;
RTI::ObjectClassHandle urunCClassID   = 0;
RTI::AttributeHandle   siparisID      = 0;
RTI::AttributeHandle   miktarID       = 0;
RTI::AttributeHandle   baszamanAID    = 0;
RTI::AttributeHandle   bitzamanAID    = 0;
RTI::AttributeHandle   baszamanBID    = 0;
RTI::AttributeHandle   bitzamanBID    = 0;
RTI::AttributeHandle   baszamanCID    = 0;
RTI::AttributeHandle   bitzamanCID    = 0;

RTI::InteractionClassHandle  siparisAcildiID     = 0;
RTI::InteractionClassHandle  BurunuUretildiID    = 0;
RTI::ParameterHandle         siparisZamaniID     = 0;
RTI::ParameterHandle         BurunuBitisZamaniID = 0;

void SleepSeconds(int howlong)
{
```

```
#ifndef _WIN32
   sleep(howlong);
#else
   Sleep(howlong*1000);
#endif
}

////////////////////////////////////////////////////////////////////////////

void OzellikTutamaciniAl(void)
{
   try
   {
            urunClassID    = rtiAmb.getObjectClassHandle("Urun");
            urunAClassID   = rtiAmb.getObjectClassHandle("Urun.UrunA");
            urunBClassID   = rtiAmb.getObjectClassHandle("Urun.UrunB");
            urunCClassID   = rtiAmb.getObjectClassHandle("Urun.UrunC");

            siparisID = rtiAmb.getAttributeHandle("SipID",urunClassID);
            miktarID = rtiAmb.getAttributeHandle("Miktar",urunClassID);
            baszamanAID  = rtiAmb.getAttributeHandle("baszamanA",urunAClassID);
            bitzamanAID   = rtiAmb.getAttributeHandle("bitzamanA",urunAClassID);
            baszamanBID  = rtiAmb.getAttributeHandle("baszamanB",urunBClassID);
            bitzamanBID   = rtiAmb.getAttributeHandle("bitzamanB",urunBClassID);
            baszamanCID  = rtiAmb.getAttributeHandle("baszamanC",urunCClassID);
            bitzamanCID  = rtiAmb.getAttributeHandle("bitzamanC",urunCClassID);
            cout<<"Ozellik tutamaclari alindi."<<endl;
   }
   catch ( RTI::Exception& e )
   {
     cerr << "Ozelliklerin alinmasinda hata ile karsilasildi: " << &e << endl;
   }
}

////////////////////////////////////////////////////////////////////////////

void OzellikleriYayimlaAboneol(void)
{
   RTI::AttributeHandleSet *ozellikler;
         RTI::AttributeHandleSet *ozelliklerA;
         RTI::AttributeHandleSet *ozelliklerB;
         RTI::AttributeHandleSet *ozelliklerC;

   ozellikler= RTI::AttributeHandleSetFactory::create(2);
   ozellikler->add( siparisID );
   ozellikler->add( miktarID );

   ozelliklerA= RTI::AttributeHandleSetFactory::create(2);
   ozelliklerA->add( baszamanAID );
```

```
   ozelliklerA->add( bitzamanAID );

   ozelliklerB= RTI::AttributeHandleSetFactory::create(2);
   ozelliklerB->add( baszamanBID );
   ozelliklerB->add( bitzamanBID );

   ozelliklerC= RTI::AttributeHandleSetFactory::create(2);
   ozelliklerC->add( baszamanCID );
   ozelliklerC->add( bitzamanCID );

   try
         {
               rtiAmb.subscribeObjectClassAttributes(urunClassID, *ozellikler);
               rtiAmb.subscribeObjectClassAttributes(urunAClassID, *ozelliklerA);
               rtiAmb.subscribeObjectClassAttributes(urunBClassID, *ozelliklerB);
               rtiAmb.subscribeObjectClassAttributes(urunCClassID, *ozelliklerC);

               rtiAmb.publishObjectClass(urunClassID,*ozellikler);
               rtiAmb.publishObjectClass(urunAClassID,*ozelliklerA);
               rtiAmb.publishObjectClass(urunBClassID,*ozelliklerB);
               rtiAmb.publishObjectClass(urunCClassID,*ozelliklerC);
               cout<<"Ozellikler basarili bir sekilde yayimlandi ve abone
olundu."<<endl;
         }
   catch ( RTI::Exception& e )
         {
               cerr<<"Ozellikleri yayimlama ve abone olmada hata ile
karsilasildi"<<&e<<endl;
         }

         ozellikler->empty();
         ozelliklerA->empty();
         ozelliklerB->empty();
         ozelliklerC->empty();

         delete ozellikler;
         delete ozelliklerA;
         delete ozelliklerB;
         delete ozelliklerC;
}

//////////////////////////////////////////////////////////////////////////////

void EtkilesimleriYayimlaAboneol(void)
{
   try
   {
      siparisAcildiID  = rtiAmb.getInteractionClassHandle("SiparisAcildi");
```

```
      BurunuUretildiID = rtiAmb.getInteractionClassHandle("BurunuUretildi");

      rtiAmb.publishInteractionClass(siparisAcildiID);
      rtiAmb.publishInteractionClass(BurunuUretildiID);

      rtiAmb.subscribeInteractionClass(siparisAcildiID);
      rtiAmb.subscribeInteractionClass(BurunuUretildiID);
      cout<<"Etkilesimler basarili bir sekilde yayimlandi ve abone olundu."<<endl;

   }
   catch ( RTI::Exception& e )
   {
       cerr << "Etkilesimleri yayimlama ve abone olmada hata ile karsilasildi:" << &e
<< endl;
   }
}

//////////////////////////////////////////////////////////////////////////////

void FederasyonaKatil(void)
{
int ready=0;
   try
   {
         rtiAmb.createFederationExecution("Uretim", "uretim.fed");
         printf ("URETIM Federasyonu Olusturuldu!\n");
   }
   catch (RTI::FederationExecutionAlreadyExists)
   {
      cerr << "Federasyon daha once olusturulmus." << endl;
   }
   catch (RTI::Exception& x)
   {
      cerr << "Federasyon olusturulamadi: " << &x << endl;
   }

         while (!ready)
   {
      ready = 1;

      try
      {
         rtiAmb.joinFederationExecution("Uretici","Uretim", &fedAmb);
         printf("URETICI Federesi Federasyona Katildi!\n");
      }
      catch (RTI::FederateAlreadyExecutionMember& e)
      {
         cerr << "Error: "<< " Bu federe zaten Uretim federasyonun bir üyesidir." <<
&e << endl;
```

```
        }
        catch (RTI::FederationExecutionDoesNotExist& e)
        {
           cerr << "Federasyonun oluşturulmasını bekliyor: " << &e << endl;
           ready = 0;
           SleepSeconds(1);
        }
        catch (RTI::CouldNotOpenFED& e)
        {
           cerr << "FED dosyası açılamadı." << &e << endl;
        }
        catch (RTI::ErrorReadingFED& e)
        {
           cerr << "FED dosyasını okuyamadı." << &e << endl;
        }
        catch ( RTI::Exception& e )
        {
           cerr << "Federasyona katılamadı." << &e << endl;
        }
    }
}

////////////////////////////////////////////////////////////////////////////

void FederasyondanAyril(void)
{
   try
   {
      rtiAmb.resignFederationExecution(
RTI::DELETE_OBJECTS_AND_RELEASE_ATTRIBUTES );
         printf("Federasyondan Cikildi!\n");
   }
   catch ( RTI::Exception& e )
   {
      cerr << "Federasyondan Ayrılırken hata ile karşılaşıldı:" << &e << endl;
   }

   try
   {
      rtiAmb.destroyFederationExecution("Uretim");
      printf("Federasyon Yok Edildi!\n");
   }
   catch (RTI::FederatesCurrentlyJoined)
   {
   }
   catch (RTI::Exception& ex)
   {
      cerr << "Hata:" << &ex << endl;
```

```
    }
}

////////////////////////////////////////////////////////////////////////////

int YeniSiparis(void)
{
    RTI::ObjectHandle id;
        Urun *obj;

    try
    {
        id = rtiAmb.registerObjectInstance(urunClassID);
                cout<<"Yeni Siparis Acildi"<<endl;
                cout<<"SipID:"<<id<<endl;
                obj ->SetSipNo((int) id);

                if (!objects.FindObject((int)id))
                {
                        objects.AddObject((int)id,CLASS_URUN);
                        cout<<"Urun Tabloya eklendi: "<<id<<endl;
                }
    }
    catch (RTI::Exception& x)
    {
        cerr << "Yeni siparis olusturulmasında hata ile karsilasildi: " << &x << endl;
        return 0;
    }

    return (int)id;
}

////////////////////////////////////////////////////////////////////////////

void MiktarGuncelle(int id)
{
    Urun *obj;
        long mik, gmik;

        mik = rand()%50+100;

        gmik = htonl(mik);

        RTI::AttributeHandleValuePairSet *isimDegerSeti;

        isimDegerSeti = RTI::AttributeSetFactory::create(1);
        isimDegerSeti->add(miktarID, (char*)&gmik, sizeof(gmik));
        try
        {
```

```
                rtiAmb.updateAttributeValues(RTI::ObjectHandle(id),
*isimDegerSeti, NULL);
                obj ->SetMiktar(mik);
                cout<<"Miktar guncellendi: "<<Urun::Miktar()<<endl;
        }
        catch (RTI::Exception&e)
        {
                cerr<<"Miktar guncellemede hata: "<<&e<<endl;
        }
        isimDegerSeti->empty();
        delete isimDegerSeti;
}

////////////////////////////////////////////////////////////////////////////

void SipAcildiEtkilesimiGonder(void)
{
   RTI::ParameterHandleValuePairSet* Parametreler = NULL;
   Parametreler = RTI::ParameterSetFactory::create(0);

   try
   {
          rtiAmb.sendInteraction(siparisAcildiID, *Parametreler, "" );
          cout<<"Siparis Acildi Etkilesimi Gonderildi"<<endl;
   }
   catch ( RTI::Exception& e )
   {
      cerr << "Siparis Acildi Etkilesimini Gondermede Hata:" << &e << endl;
   }
   delete Parametreler;
}

////////////////////////////////////////////////////////////////////////////

int SiparisA(void)
{
   RTI::ObjectHandle id;

   try
   {
       id = rtiAmb.registerObjectInstance(urunAClassID);
       printf("A Siparisi Acildi\n");
       cout<<"Sip A ID:"<<id<<endl;
   }
   catch (RTI::Exception& x)
   {
      cerr << "A siparisi olusturulmasında hata ile karsilasildi: " << &x << endl;
      return 0;
   }
```

```
   return (int)id;
}

////////////////////////////////////////////////////////////////////////////

void AUrunuUret(int id)
{
        long basZaman, bitZaman, gbasZaman, gbitZaman;
        Urun *obj;
        int t1;
        t1 = rand()%2 + 2;
        basZaman = (int) fedZamani;
        bitZaman=basZaman + Urun::Miktar() * t1;

        gbasZaman = htonl(basZaman);
        gbitZaman = htonl(bitZaman);

        RTI::AttributeHandleValuePairSet *isimDegerSeti;

        isimDegerSeti = RTI::AttributeSetFactory::create(2);
        isimDegerSeti->add(baszamanAID,(char*)&gbasZaman,sizeof(gbasZaman));
        isimDegerSeti->add(bitzamanAID , (char*)&gbitZaman, sizeof(gbitZaman));
        try
        {
           rtiAmb.updateAttributeValues(RTI::ObjectHandle(id),      *isimDegerSeti,
NULL);

           obj->SetABasZaman(basZaman);
           obj->SetABitZaman(bitZaman);

           cout<<"Baslangic ve Bitis Zamanlari Guncellendi: "
<<Urun::ABasZaman()<<"-"<<Urun::ABitZaman()<<endl;
        }
        catch (RTI::Exception&e)
        {
          cerr<<"Başlangıç ve Bitiş Zamanlarını Güncellemede Hata: "<<&e<<endl;
        }
        isimDegerSeti->empty();
        delete isimDegerSeti;
}

////////////////////////////////////////////////////////////////////////////

void SiparisSil(int ID)
{
   try
   {
        rtiAmb.deleteObjectInstance(RTI::ObjectHandle(ID),"");
        cout<<ID<<" Numarali siparis sonlandirildi."<<endl;
```

```
   }
   catch (RTI::Exception& x)
   {
      cerr << "Siparisin sonlandırılmasında hata ile karsilasildi: " << &x << endl;
      return;
   }
}

//////////////////////////////////////////////////////////////////////////

void TickRTI(void)
{
   int eventsToProcess = 1;
   while (eventsToProcess)
   eventsToProcess = rtiAmb.tick();
}

//////////////////////////////////////////////////////////////////////////

int SiparisC(void)
{
   RTI::ObjectHandle id;
   try
   {
         id = rtiAmb.registerObjectInstance(urunCClassID);

         printf("C Siparisi Acildi\n");
         cout<<"CID:"<<id<<endl;
   }
   catch (RTI::Exception& x)
   {
      cerr << "C siparisi olusturulmasında hata ile karsilasildi: " << &x << endl;
      return 0;
   }
   return (int)id;
}

//////////////////////////////////////////////////////////////////////////

void CUrunuUret(int id)
{
         int CBasZaman, CBitZaman, gbasZaman, gbitZaman;
         Urun *obj;

         cout<<"ABitZaman:" << Urun::ABitZaman() << " BBitZaman:"
<<Urun::BBitZaman()<<endl;
         if(Urun::ABitZaman()>Urun::BBitZaman())
         {
                  fedZamani = Urun::ABitZaman();
```

```
	}
	else if (Urun::ABitZaman()<=Urun::BBitZaman())
	{
		fedZamani = Urun::BBitZaman();
	}
	CBasZaman = fedZamani;
	CBitZaman = CBasZaman + Urun::Miktar() * 6;

	gbasZaman = htonl(CBasZaman);
	gbitZaman = htonl(CBitZaman);

	RTI::AttributeHandleValuePairSet *isimDegerSeti;

	isimDegerSeti = RTI::AttributeSetFactory::create(2);
	isimDegerSeti->add(baszamanCID,(char*)&gbasZaman,sizeof(gbasZaman));
	isimDegerSeti->add(bitzamanCID , (char*)&gbitZaman, sizeof(gbitZaman));
	try
	{
		rtiAmb.updateAttributeValues(RTI::ObjectHandle(id),
*isimDegerSeti, NULL);

		obj->SetCBasZaman(CBasZaman);
		obj->SetCBitZaman(CBitZaman);
		fedZamani = Urun::CBitZaman();

		cout<<"Baslangiç ve Bitis Zamanlari Guncellendi: "
<<Urun::CBasZaman()<<"-"<<Urun::CBitZaman()<<endl;
	}
	catch (RTI::Exception&e)
	{
	  cerr<<"Başlangıç ve Bitiş Zamanlarını Güncellemede Hata: "<<&e<<endl;
	}
	isimDegerSeti->empty();
	delete isimDegerSeti;
}

//////////////////////////////////////////////////////////////////////////

void DosyayaYaz(void)
{
	ofstream dosyayayaz ("sonuc.dat", ios::app);
	if (!dosyayayaz)
	{
		cerr<<"Dosya acilamadı!"<<endl;
		exit(1);
	}
	dosyayayaz <<Urun::SipNo()<<"     "<<Urun::Miktar()<<"          "
	<<Urun::ABasZaman()<<"   "<<Urun::ABitZaman()<<"   "
	<<Urun::BBasZaman()<<"   "<<Urun::BBitZaman()
```

```
        <<"     "<<Urun::CBasZaman()<<"  "<<Urun::CBitZaman()<<endl;
}

//////////////////////////////////////////////////////////////////////////

void DosyadanListele(void)
{
        ifstream dosyadanlistele ("sonuc.dat", ios::in);
        if (!dosyadanlistele)
        {
                cerr<<"Dosya acilamadı!"<<endl;
                exit(1);
        }
        cout<<"SipNo  "<<"Miktar  "<<"[A]BasZam-BitZam  "
                <<"[B]BasZam-BitZam  "<<"[C]BasZam-BitZam  "<<endl;

        int sip, mik, abas, abit, bbas, bbit, cbas, cbit;

        while (dosyadanlistele>>sip>>mik>>abas>>abit>>bbas>>bbit>>cbas>>cbit)
                cout<<setiosflags(ios::left)<<setw(7)<<sip
                <<setiosflags(ios::left)<<setw(12)<<mik
                <<setiosflags(ios::right)<<setw(5)<<abas<<"- "
                <<setiosflags(ios::left)<<setw(11)<<abit
                <<setiosflags(ios::right)<<setw(5)<<bbas<<"- "
                <<setiosflags(ios::left)<<setw(11)<<bbit
                <<setiosflags(ios::right)<<setw(5)<<cbas<<"- "
                <<setiosflags(ios::left)<<setw(5)<<cbit
                <<endl;
}
```

EK C: rti_arayuz.h Dosyası

```
#ifndef RTI_ARAYUZ_H
#define RTI_ARAYUZ_H
#include <RTI.hh>
#include "UrFederateAmbassador.h"
void OzellikTutamaciniAl(void);
void OzellikleriYayimlaAboneol(void);
void EtkilesimleriYayimlaAboneol(void);

void SipAcildiEtkilesimiGonder(void);
void BurunuUretildi(void);

int  YeniSiparis(void);
int  SiparisA(void);
void CUrunuUret(int id);
int SiparisC(void);

void SiparisSil(int SipID);
void SetMiktar( const double& miktar);
void MiktarGuncelle(int id);
void AUrunuUret(int id);
void DosyayaYaz(void);
void DosyadanListele(void);

void FederasyonaKatil(void);
void FederasyondanAyril(void);
void ZamanArtir(int zaman);
void TickRTI(void);

#endif
```

EK D: UrFederateAmbassador.cpp Dosyası

```
#include <stdio.h>
#include <stdlib.h>
#include <winsock2.h>
#include <process.h>
#include <windows.h>
#define getpid _getpid
#include "Urun.h"
#include "UrFederateAmbassador.h"
#include "rti_arayuz.h"
#include <iostream.h>
using std::endl;

UrunTable objects;

extern RTI::RTIambassador    rtiAmb;

int FedTime=0;

extern RTI::ObjectClassHandle urunClassID;
extern RTI::ObjectClassHandle urunAClassID;
extern RTI::ObjectClassHandle urunBClassID;
extern RTI::ObjectClassHandle urunCClassID;

extern RTI::AttributeHandle   siparisID;
extern RTI::AttributeHandle   miktarID;
extern RTI::AttributeHandle   baszamanAID;
extern RTI::AttributeHandle   bitzamanAID;
extern RTI::AttributeHandle   baszamanBID;
extern RTI::AttributeHandle   bitzamanBID;
extern RTI::AttributeHandle   baszamanCID;
extern RTI::AttributeHandle   bitzamanCID;

extern RTI::InteractionClassHandle  siparisAcildiID;
extern RTI::InteractionClassHandle  BurunuUretildiID;

extern RTI::ParameterHandle          siparisZamaniID;
extern RTI::ParameterHandle          BurunuBitisZamaniID;

//////////////////////////////////////////////////////////////////////
void FedAmb::startRegistrationForObjectClass(
   RTI::ObjectClassHandle        theClass)
throw (
  RTI::ObjectClassNotPublished,
  RTI::FederateInternalError)
{

}
```

```
////////////////////////////////////////////////////////////////////////
void FedAmb::discoverObjectInstance(
   RTI::ObjectHandle          theObject,
   RTI::ObjectClassHandle     theObjectClass,
   const char                 *theTag)
throw (
  RTI::CouldNotDiscover,
  RTI::ObjectClassNotKnown,
  RTI::FederateInternalError)
{
   cout<<"DISCOVER OBJECT INSTANCE KISMINA GIRILDI..."<<endl;
   cout<<"Object Handle No: "<<theObject<<endl;

    if (!objects.FindObject((int)theObject))
   {
      if (theObjectClass == urunClassID)
      {
          cout<<"Urun Handle No: "<<theObject<<endl;
         objects.AddObject((int)theObject,CLASS_URUN);
      }
   }
}
////////////////////////////////////////////////////////////////////////
void FedAmb::reflectAttributeValues(
   RTI::ObjectHandle                    theObject,
   const RTI::AttributeHandleValuePairSet& theAttributes,
   const char                           *theTag)
throw (
  RTI::ObjectNotKnown,
  RTI::AttributeNotKnown,
  RTI::FederateOwnsAttributes,
  RTI::FederateInternalError)
{
        cout<<"REFLECT ATTRIBUTES KISMINA GIRILDI..."<<endl;

   RTI::AttributeHandle attrHandle;
   unsigned long valueLength;
   Urun *obj;

   for (unsigned int i = 0; i < theAttributes.size(); i++)
   {
                attrHandle = theAttributes.getHandle(i);

                if (attrHandle== baszamanBID)
                {
                  int bbas, gbbas;

                  theAttributes.getValue(i, (char *)&gbbas, valueLength);
                  bbas = ntohl(gbbas);
```

```
                obj->SetBBasZaman(bbas);
                cout<<"B baslangic zamani alindi: "<<Urun::BBasZaman()<<endl;
               }

               if (attrHandle== bitzamanBID)
               {
                int bbit, gbbit;

                theAttributes.getValue(i, (char *)&gbbit, valueLength);
                bbit = ntohl(gbbit);
                obj->SetBBitZaman(bbit);
                cout<<"B bitis zamani alindi: "<<Urun::BBitZaman()<<endl;
               }
   }
}
////////////////////////////////////////////////////////////////////////
void FedAmb::receiveInteraction (
      RTI::InteractionClassHandle       theInteraction,
  const RTI::ParameterHandleValuePairSet& theParameters,
  const char                        *theTag)
throw (
  RTI::InteractionClassNotKnown,
  RTI::InteractionParameterNotKnown,
  RTI::FederateInternalError)
{
        cout<<"RECEIVE INTERACTION KISMINA GIRILDI..."<<endl;
        if (theInteraction == BurunuUretildiID)
        {
                cout << "B Urunu uretildi bilgisi alindi..."<<endl;
                Urun *obj;
                obj -> SetCUret(1);
        }
}
////////////////////////////////////////////////////////////////////////
void FedAmb::removeObjectInstance(
   RTI::ObjectHandle         theObject,
   const char               *theReason)
throw (
  RTI::ObjectNotKnown,
  RTI::FederateInternalError)
{

}
```

EK E: Urun.h Dosyası

```
#ifndef _URUN_TABLE_H
#define _URUN_TABLE_H
#include <iostream>
using std::endl;
#include <string.h>
#define MAX_OBJECTS 15
enum
{
  NOT_INITIALIZED,
  DISCOVERED,
  INITIALIZED
};

class UrunTable;

class Urun
{
   friend UrunTable;

   int    objectType;
   int    initialized;
   int    state;
   static int  sipno;
   static long miktar;
   static long abaszaman;
   static long abitzaman;
   static long bbaszaman;
   static long bbitzaman;
   static long cbaszaman;
   static long cbitzaman;
   static bool curet;

   public:

   Urun(void);
   ~Urun(void) {};
////////////////////////////////////////////////////////////
   int Class(void)
      { return objectType; };

   static int SipNo(void)
      { return sipno; };

   static bool CUret(void)
       { return curet; };

   static long Miktar(void)
```

```
    { return miktar; };

  static long ABasZaman(void)
    { return abaszaman; }

  static long BBasZaman(void)
     { return bbaszaman; }

  static long CBasZaman(void)
     { return cbaszaman; }

  static long ABitZaman(void)
     { return abitzaman; }

  static long BBitZaman(void)
     { return bbitzaman; }

  static long CBitZaman(void)
      { return cbitzaman; }

  int IsInitialized(void)
       { return (initialized==INITIALIZED); };
////////////////////////////////////////////////////////////
  void Initialize(void)
       { initialized = INITIALIZED; };

  void SetCUret(bool _curet)
       { curet = _curet; };

  void SetMiktar(long _miktar)
      { miktar = _miktar; };

  void SetABasZaman(long _abaszaman)
       { abaszaman=_abaszaman;};

   void SetBBasZaman(long _bbaszaman)
       { bbaszaman=_bbaszaman;};

   void SetCBasZaman(long _cbaszaman)
       { cbaszaman=_cbaszaman;};

   void SetABitZaman(long _abitzaman)
       { abitzaman=_abitzaman;};

   void SetBBitZaman(long _bbitzaman)
       { bbitzaman=_bbitzaman;};

   void SetCBitZaman(long _cbitzaman)
       { cbitzaman=_cbitzaman;};
```

```
    void SetSipNo(int _sipno)
        { sipno = _sipno; };
};

class UrunTable
{
   private:
   Urun objects[MAX_OBJECTS];
   public:

   Urun *FindObject(int handle);
   Urun *AddObject(int handle,int type);
   void DeleteObject(int handle);

   Urun *GetObject(int i);

   UrunTable(void);
   ~UrunTable(void) {};
};
enum
{
  CLASS_URUN,
};
enum
{
   OBJ_STATE_DEAD,
   OBJ_STATE_ALIVE
};
#endif
```

EK F: Urun.cpp Dosyası

```
#include <stdio.h>
#include <string.h>
#include <iostream>
#include "Urun.h"
using std::endl;
using std::cout;

bool Urun::curet = 0;
int  Urun::sipno = 0;
long Urun::miktar= 0;
long Urun::abaszaman=0;
long Urun::abitzaman=0;
long Urun::bbaszaman=0;
long Urun::bbitzaman=0;
long Urun::cbaszaman=0;
long Urun::cbitzaman=0;

////////////////////////////////////////////////////////////////////////
Urun::Urun(void)
{
   objectType  = 0;
   initialized = NOT_INITIALIZED;
   sipno  = 0;
}
////////////////////////////////////////////////////////////////////////
UrunTable::UrunTable(void)
{
   for (int i=0; i < MAX_OBJECTS; i++)
   {
      objects[i].initialized = NOT_INITIALIZED;
      objects[i].sipno = 0;
      objects[i].miktar = 0.0;
   }
}
////////////////////////////////////////////////////////////////////////
Urun *UrunTable::FindObject(int handle)
{
   for (int i=0; i < MAX_OBJECTS; i++)
   {
      if ((objects[i].sipno == handle) && (objects[i].initialized))
      {
         return &objects[i];
      }
   }
   return NULL;
}
////////////////////////////////////////////////////////////////////////
```

```
void UrunTable::DeleteObject(int handle)
{
   for (int i=0; i < MAX_OBJECTS; i++)
   {
      if (objects[i].sipno == handle)
      {
         objects[i].initialized = NOT_INITIALIZED;
         objects[i].sipno      = 0;
         return;
      }
   }
}
////////////////////////////////////////////////////////////////////////
Urun *UrunTable::AddObject(int handle,int type)
{
   for (int i=0; i < MAX_OBJECTS; i++)
   {
      if (objects[i].initialized == NOT_INITIALIZED)
      {
         objects[i].objectType  = type;
         objects[i].sipno       = handle;
         objects[i].initialized = DISCOVERED;
         return &objects[i];
      }
   }
   return NULL;
}

Urun *UrunTable::GetObject(int index)
{
   return &objects[index];
}
```

ÖZGEÇMİŞ

Özer Uygun ilk, orta ve lise eğitimini Sakarya'da yapmıştır. Üniversite eğitimini Sakarya Üniversitesi Mühendislik Fakültesi Endüstri Mühensiliği Bölümünde 1999 yılında tamamlamıştır. Yüksek Lisans Eğitimini de yine Sakarya Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği Anabilimdalında "Belediyeler için Yönetim Bilişim Sistemleri" konusunda yaptığı tez ile 2002 yılında tamamlamıştır.

Ocak 2000 – Aralık 2002 tarihleri arasında Marmara Üniversitesi Fen Edebiyat Fakültesi Bilgi ve Belge Yönetimi Bölümünde Öğretim Görevlisi olarak çalışmıştır. Ocak 2003 tarihinden itibaren ise Sakarya Üniversitesi Mühendislik Fakültesi Endüstri Mühendisliği Bölümünde Araştırma Görevlisi olarak çalışmaktadır.