FEDERE TABANLI KOŞUT VE DAĞITILMIŞ BENZETİM SİSTEMLERİNİN MİMARİ MODELLEMESİ

ARCHITECHTURAL MODELING OF FEDERATED PARALLEL AND DISTRIBUTED SIMULATION SYSTEMS

TURGAY ÇELİK

Hacettepe Üniversitesi

Lisansüstü Eğitim – Öğretim ve Sınav Yönetmeliğinin

BİLGİSAYAR Mühendisliği Anabilim Dalı İçin Öngördüğü

DOKTORA TEZİ

olarak hazırlanmıştır.

2013

Fen Bilimleri Enstitüsü Müdürlüğü'ne,

Bu çalışma jürimiz tarafından **BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI'nda DOKTORA TEZİ** olarak kabul edilmiştir.

Başkan	:….. (Doç. Dr., Halit OĞUZTÜZÜN)
Üye (Danışman)	:….. (Yrd. Doç. Dr. Kayhan İMRE)
Üye	:….. (Yrd.Doç.Dr. Harun ARTUNER)
Üye	:….. (Doç. Dr. Veysi İŞLER)
Üye	:….. (Yrd. Doç. Dr. Bedir TEKİNERDOĞAN)

ONAY

Bu tez Hacettepe Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliği'nin ilgili maddeleri uyarınca yukarıdaki jüri üyeleri tarafından ...../...../..... tarihinde uygun görülmüş ve Enstitü Yönetim Kurulunca ...../...../..... tarihinde kabul edilmiştir.

Prof. Dr. Fatma SEVİN DÜZ
Fen Bilimleri Enstitüsü Müdürü

FEDERE TABANLI KOŞUT VE DAĞITILMIŞ BENZETİM SİSTEMLERİNİN MİMARİ MODELLEMESİ

Turgay Çelik

ÖZ

Koşut ve dağıtılmış benzetim sistemlerinin gerçekleştirimini kolaylaştırmak için HLA, DIS ve TENA gibi değişik benzetim altyapıları ve Dağıtık Benzetim Mühendisliği ve Çalıştırma Süreci *(Distributed Simulation Engineering and Execution Process- DSEEP)* gibi geliştirme faaliyetlerini destekleyici süreçler tanımlanmıştır. DSEEP'te tanımlanan önemli işlerden birisi, benzetim sisteminin performansının tasarım aşamasında değerlendirilmesidir. Bir benzetim sisteminin başarımı, sistemi oluşturan uygulamaların kaynaklara yerleştirilme biçiminden önemli ölçüde etkilenir. Uygulamaların kaynaklara yerleştirilmesi genelde pek çok farklı yöntemle gerçekleştirilebilir. DSEEP yerleştirme seçeneklerinin değerlendirilmesi için somut bir yöntem tanımlamaz. Bunun yanı sıra, değişik DSEEP adımlarını gerçekleştirmek için tanımlanmış mevcut yaklaşımlar da bu amaca yönelik yeteri kadar destek sağlamamaktadır. Bu çalışmada benzetim sistemi ve mevcut donanım kaynaklarına göre uygun yerleştirme seçeneklerinin türetilmesi için somut bir yaklaşım geliştirilmiştir. İlk adımda benzetim bileşenleri ve donanım kaynakları tasarlanmaktadır. İkinci adımda, bu tasarım kullanılarak çalıştırma konfigürasyonu seçenekleri tanımlanır. Son adımda ise, önceki iki adımda oluşturulan tasarım ve çalıştırma konfigürasyonu seçenekleri kullanılarak uygun bir yerleştirme seçeneği algoritmik olarak türetilebilir. Bu çalışmada geliştirilen yaklaşımı destekleyecek bir araç ailesi de geliştirilmiştir. Araç ailesi ile benzetim ortamı tasarımının yapılması ve uygun yerleştirme konfigürasyonlarının otomatik olarak türetilmesi işlemleri gerçekleştirilebilir. Geliştirilen yaklaşım endüstriyel karşılığı olan geniş ölçekli iki farklı benzetim sistemine uygulanmıştır.

Anahtar Kelimeler: Koşut ve Dağıtık Benzetimler, DSEEP, Yazılım Mimarisi, Model Güdümlü Mühendislik, Model Dönüştürme, Meta-modelleme, Yazılım Mimarisi Eniyileme

Danışman: Yrd. Doç. Dr. Kayhan İMRE, Hacettepe Üniversitesi, Bilgisayar Mühendisliği Bölümü

ARCHITECHTURAL MODELING OF FEDERATED PARALLEL AND DISTRIBUTED SIMULATION SYSTEMS

Turgay Çelik

ABSTRACT

Parallel and distributed simulations (PADS) realize the distributed execution of a simulation system over multiple physical resources. To realize the execution of PADS, different simulation infrastructures such as HLA, DIS, and TENA have been defined and different processes such as the Distributed Simulation Engineering and Execution Process (DSEEP) that supports the mapping of the simulations on the infrastructures has been proposed. An important recommended task in DSEEP is the evaluation of the performance of the simulation systems at the design phase. In general, the performance of a simulation is largely influenced by the allocation of member applications to the resources. Usually, the deployment of the applications to the resources can be done in many different ways. DSEEP does not provide a concrete approach for evaluating the deployment alternatives. Moreover, current approaches that can be used for realizing various DSEEP activities do not yet provide adequate support for this purpose. We provide a concrete approach for deriving feasible deployment alternatives based on the simulation system and the available resources. In the approach, first the simulation components and the resources are designed. The design is used to define alternative execution configurations, and based on the design and the execution configuration a feasible deployment alternative can be algorithmically derived. Tool support is developed for the simulation design, the execution configuration definition and the automatic generation of feasible deployment alternatives. The approach has been applied within two different large scale industrial case studies.

Keywords: Parallel and Distributed Simulations, High Level Architecture (HLA), DSEEP, Software Architecture, Model Driven Engineering (MDE), Model Transformations, Metamodeling, Software Architecture Optimization

Advisor: Associate Professor Kayhan İMRE, Hacettepe University, Department of Computer Engineering

TEŞEKKÜR

Yazar, bu çalışmanın gerçekleşmesinde katkılarından dolayı, aşağıda adı geçen kişi ve kuruluşlara içtenlikle teşekkür eder.

Sayın Yard.Doç.Dr. Kayhan İmre (tez danışmanı), yazarı benzetim dünyasıyla tanıştırmış, tez çalışmasının çalışmanın sonuca ulaştırılmasında, karşılaşılan güçlüklerin aşılmasında yön gösterici olmuştur.

Sayın Yard.Doç.Dr. Bedir Tekinerdoğan, tez çalışmasının sınırlarının belirlenmesinde, tezin katkısının somutlaştırılmasında ve sonuca ulaştırılmasında yön gösterici olmuştur.

Sayın Yard.Doç.Dr. Harun Artuner ve Doç.Dr. Veysi İşler tez çalışmasını takip etmiş, gerekli yönlendirmeleri yapmıştır.

Sayın Doç.Dr. Halit Oğuztüzün tez içeriğinin iyileştirilmesine ve tezin anlatım dilinin düzeltilmesinde yardımcı olmuştur.

Yazarın doktora çalışması boyunca çalışanı olduğu MilSOFT Yazılım Teknolojileri A.Ş, yazarın doktora çalışmalarını devam ettirmesi için gerekli esnekliği sağlamıştır.

MilSOFT Yazılım Teknolojileri A.Ş Genel Müdürü Sayın İsmail Başyiğit, yazarın doktora çalışmasını desteklemiştir.

Sayın Yavuz Tuğcu ve Güral Vural tez içeriğinin iyileştirilmesine ve tezin anlatım dilinin düzeltilmesinde yardımcı olmuşlardır.

Sayın Dr. Eda Yücel ve Hakan Serçe tez kapsamında geliştirilen kavramların olgunlaşmasında ve tezin anlatım dilinin düzeltilmesinde yardımcı olmuşlardır.

Son olarak, hayatımın her aşamasında beni destekleyen ve bana her zaman inanan aileme en içten teşekkürlerimi sunuyorum.

İÇİNDEKİLER DİZİNİ

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR

3B	3 Boyutlu
AAA	Anti Aircraft Artillery
API	Application Programming Interface
ATL	Atlas Transformation Language
BOM	Base Object Model
BVDM	Benzetim Veri Değişim Modeli
CGF	Computer Generated Forces
CMDS	Counter Measure Dispenser System
COM	Component Object Model
CRC	Central RTI Component
CTAP	Capacitated Task Allocation Problem
CW	Continuous Wave
C2	Command Control
DDM	Data Distribution Management
DDS	Data Distribution Service
DEVS	Discrete Event System Specification
DIS	Distributed Interactive Simulation
DIRCM	Directed Infra-Red Counter Measure
DMSO	Defense Modeling and Simulation Office
DoD	Department of Defense
DSEEP	Distributed Simulation Engineering and Execution Process
DSL	Domain Specific Language
ED	Elektronik Destek
EH	Elektronik Harp
EMF	Eclipse Modeling Framework
EO	Elektro-Optik

ET	Elektronik Taarruz
FAMM	Federation Architecture Metamodel
FEDEP	Federation Development and Execution Process
FOM	Federation Object Model
GEF	Graphical Editing Framework
GME	Generic Modeling Environment
GMF	Graphical Modeling Framework
GRIM	Guidance, Rationale, and Interoperability Modalities *(for the Real-time Platform Reference Federation Object Model - RPR FOM)*
HH	Havadan Havaya
HLA	High Level Architecture
HOMM	HLA Object Model Metamodel
HS	Havadan Satha
IRCM	Infra-Red Counter Measure
I/O	Input / Output
JET	Java Emitter Templates
KAMA	Kavramsal Modelleme Aracı
KKIAP	Kapasite Kısıtlı İş Atama Problemi
LAN	Local Area Network
LRC	Local RTI Component
LROM	Logical Range Object Model
LSC	Live Sequence Chart
LWR	Laser Warning Receiver
MAWS	Missile Approach Warning System
MDA	Model Driven Architecture
MDE	Model Driven Engineering
MOF	Meta Object Facility

M2M	Model to Model (Transformation)
M2T	Model to Text (Transformation)
OCL	Object Constraint Language
OMG	Object Management Group
OMT	Object Model Template
PDU	Protocol Data Unit
PIM	Platform Independent Model
PSM	Platform Specific Model
QVT	Query/View/Transformation
RAHAT	Rafta Hazır Ticari
RF	Radio Frequency
RPR-FOM	Real-time Platform Reference Federation Object Model
RTI	Runtime Infrastructure
RUP	Rational Unified Process
RWR	Radar Warning Receiver
SAM	Surface to Air Missile
SDEM	Simulation Data Exchange Model
S-IDE	Simulation–IDE
SOM	Simulation Object Model
SH	Satıhtan Havaya
SS	Satıhtan Satha
T2M	Text to Model (Transformation)
TAP	Task Allocation Problem
TÇB	Taktik Çevre Benzetimi
TES	Tactical Environment Simulation
UML	Unified Modeling Language
VV&A	Verification, Validation and Accreditation

WAN	Wide Area Network
XML	eXtensible Markup Language
XMI	XML Metadata Interchange

1 GİRİŞ

Koşut ve Dağıtılmış benzetimler *(Parallel and Distributed Simulations-PADS)* [Fujimoto 1999], bir benzetimin birden çok fiziksel kaynak üzerine dağıtılarak çalıştırılabilmesine olanak sağlarlar. Koşut ve Dağıtılmış benzetim sistemleri doğaları gereği karmaşık sistemlerdir ve geliştirilen sisteme özel gereksinimlerin yanı sıra, aşağıda birkaçı sıralanan genel teknik seviye gereksinimlerin adreslenmesini gerektirmektedir:

- Veri isteklerinin tanımlanması (*Declaration Management*)
- Veri değişimi (*Data Exchange*)
- Zaman yönetimi (*Time Management*)
- Keşfetme mekanizmaları (*Discovery Mechanisms*)
- Veri dağıtım yönetimi (*Data Distribution Management*)

Koşut ve dağıtılmış benzetim sistemlerinin geliştirilmesini kolaylaştırmak için *Distributed Interactive Simulation (DIS)* [IEEE 1998], *Discrete Event System Specification (DEVS)* [Zeigler 2003], *High Level Architecture (HLA)* [Kuhl 1999; IEEE 2010a], ve *Test and Training Enabling Architecture (TENA)* [Noseworthy 2008] gibi ortak standart mimari ve altyapılar tanımlanmıştır. Bunlar arasında, HLA hem koşut hem de dağıtılmış benzetimleri desteklemek için ortak bir mimari sağlayan önemli bir IEEE ve NATO standardıdır [Kuhl 1999; Perumalla ve Fujimoto 2003].

Koşut ve dağıtılmış bir benzetim sistemi birden çok alt uygulamadan oluşur. Örneğin koşut ve dağıtılmış bir trafik benzetimi gerçekleştiriminde, sistem araçları, sürücüleri, yayaları, sinyalizasyon öğelerini, çevreyi ve benzeri katılımcıları modelleyen alt uygulamalara bölünebilir. Sistemin alt parçalara bölünmesi, genellikle performans, tekrar kullanılabilirlik ve birlikte çalışabilirlik gibi kalite faktörlerinin sağlanması için uygulanır [Fujimoto 1999]. Sistemin alt parçalara bölünmesi ve parçaların fiziksel kaynaklara yerleştirilmesi pek çok farklı yöntemle gerçekleştirilebilir. Bir benzetim sisteminin başarımı, alt parçalarının mevcut fiziksel kaynaklara atanma konfigürasyonundan önemli ölçüde etkilenir.

HLA uyumlu benzetim sistemlerinin geliştirilmesi sürecini desteklemek için IEEE Federasyon Geliştirme ve Çalıştırma Süreci - FEDEP (*Recommended Practice for High Level Architecture Federation Development and Execution Process*) tanımlanmıştır [IEEE 2003]. FEDEP temel alınarak, yakın zamanda IEEE Dağıtılmış Benzetim Mühendisliği ve Çalıştırma Süreci- DSEEP (*Distributed Simulation Engineering and Execution Process*) standardı tanımlanmıştır [IEEE 2010d]. DSEEP benzetim sistemleri geliştirmek için önerilen işleri içeren bir grup adım ve aktiviteden oluşur.

DSEEP soyut bir süreçtir ve bilinçli olarak süreç adımlarının gerçekleştirim detayları için belirli bir yöntem tanımlamaz ve gerçekleştirim yöntemi için herhangi bir kısıtlama getirmez. DSEEP'te önerilen önemli işlerden birisi benzetim sisteminin başarımının tasarım aşamasında değerlendirilmesidir. Literatürde DSEEP'i destekleyen kavramsal modelleme yaklaşımları [Karagöz ve Demirörs 2007; SISO 2006], araçlar [Parr 2003; VT MAK 2010a], ve benzetim tasarlama yaklaşımları [Topçu et al. 2008; Çetinkaya ve Oguztüzün 2006] gibi çeşitli çalışmalar mevcuttur.

Literatürdeki mevcut çalışmalar DSEEP'te önerilen farklı işlerin gerçekleştirilmesi açısından faydalı olsa da, şu ana kadar yerleştirme seçeneklerinin belirlenmesi ve değerlendirilmesi için yeterli ve net destek sağlanmamıştır. Bu eksiklikten dolayı yerleştirme tasarımının değerlendirilmesi ve başarım değerlendirmesi genellikle ya geliştirme aşamasına ertelenir ya da tasarım aşamasında uzman değerlendirmesi (*expert judgment*) ile yapılır.

DSEEP'te tasarım aşamasında yapılması önerilen bu işlerin geliştirme aşamasına ertelenmesi, etkin olmayan gerçekleştirimlerin yapılmasına sebep olabilir. Bu durumda tasarım ve detaylı tasarım, gerçekleştirim, sınama elemanları, belgeler gibi ilişkili proje yaşam döngüsü elemanlarının gereksiz yere defalarca güncellenmesi gerekebilir. Benzer biçimde, uygun bir yerleştirme çözümü arama süreci kolaylıkla proje takviminde gecikmelere neden olabilir ve gereksiz iş gücü kaybından dolayı ürün geliştirme maliyetleri ciddi biçimde artabilir.

Uzman değerlendirmesi yöntemi sürecin gerçekleştirilmesini destekleyebilir ama sistemle ilgili tüm alanlarda bilgisi olan uzmanların bulunması kolay değildir. Daha

da önemlisi, uzman değerlendirmesi yöntemi küçük ve orta ölçekli sistemler için uygulanabilir olsa da, sistem büyüyüp karmaşıklaştıkça bu yöntemin uygulanabilirliği giderek azalır. Bu bağlamda, uygun yerleştirme seçeneklerinin aranması için daha sistematik ve biçimsel bir yaklaşıma ihtiyaç duyulduğu aşikârdır.

Bu çalışmada, benzetim sistemi ve mevcut fiziksel kaynaklara göre uygun bir yerleştirme seçeneklerinin türetilmesine yönelik somut bir yaklaşım sağlanmaktadır. Yaklaşımın ilk adımda benzetim bileşenleri ve fiziksel kaynaklar tasarlanır. Sonraki adımda benzetim tasarımı kullanılarak çalışma konfigürasyonu seçenekleri tanımlanır. Çalışma konfigürasyonu seçeneklerinde, ilişkili tasarım elemanlarının olgu sayısı ve parametreleri detaylandırılır. Benzetim tasarımı ve çalıştırma konfigürasyonu kullanılarak uygun yerleştirme seçenekleri algoritmik olarak türetilebilir. Bu çalışma kapsamında önerilen yaklaşımı desteklemek için, benzetim tasarımı, çalışma konfigürasyonu tanımlanması ve uygun yerleştirme seçeneklerinin otomatik olarak üretilmesini sağlayan araçlar geliştirilmiştir. Geliştirilen yaklaşım geniş ölçekli endüstriyel bir Elektronik Harp benzetimi ve bir trafik benzetimi örnek durum çalışmaları kullanılarak geçerlenmiştir. Yapılan çalışmanın sağladığı katkı somut olarak aşağıdaki gibi tanımlanabilir:

- Erken tasarım aşamasında mevcut fiziksel kaynaklar da dikkate alınarak uygun benzetim yerleştirme seçeneklerinin türetilmesi için sistematik bir analiz yaklaşımı geliştirilmiştir. Geliştirilen yaklaşım, uygun yerleştirme seçeneklerinin türetilmesinin yanı sıra, aşağıdaki gibi değişik amaçlarla da kullanılabilir:
    - Yeni benzetim bileşenlerinin sisteme eklenmesinin etkisinin analizi,
    - Mevcut tasarıma göre seçilen fiziksel kaynakların uygunluğunun analiz edilmesi,
    - Yayımlama/üye olma ilişkilerindeki değişikliklerin etkisinin analizi.
- DSEEP, tasarım aşamasında tasarım seçeneklerinin değerlendirilmesi ve benzetimin performansının tahmin edilmesini önerir, fakat bu önerilen işler için DSEEP'te bilinçli olarak detaylı bir süreç ve gerçekleştirim detayı

sağlanmamıştır. Bu çalışmada geliştirilen yaklaşım, bu önemli iş tanımlarının gerçekleştirimini desteklemek için detaylı bir süreç ve araç desteği sağlar. Geliştirilen yaklaşımın adımları, DSEEP'in ilgili adımlarıyla bütünleştirilmiş ve bu adımlarının gerçekleştirimi sağlanmıştır.

- Geliştirilen yaklaşımın adımların desteklemek için bir araç ailesi geliştirilmiştir. Geliştirilen araçlar, benzetim sisteminin ve fiziksel kaynakların tasarlaması ve tasarımın analiz edilerek uygun yerleştirme seçeneklerinin otomatik olarak üretilmesini sağlar. Araç ailesi yine bu çalışma kapsamında geliştirilen bir grup metamodel temel alınarak geliştirilmiştir.

- Literatürde pek çok çalışmaya konu olan Kapasite Sınırlı Görev Atama Problemi *(Capacitated Task Assignment Problem-CTAP)* çözüm yöntemlerinin koşut ve dağıtık benzetim sistemlerinin tasarım sürecinde kullanılmasına yönelik yeni bir yaklaşım ve araç desteği geliştirilmiştir.

Tez metninin geri kalanı şu şekilde organize edilmiştir:

- 2. bölümde temel bilgiler ve bağlam verilmektedir. Bu kapsamda, Koşut ve Dağıtılmış Benzetim Sistemleri, DSEEP ve Model Güdümlü Mühendislik hakkında bilgi verilmiştir.

- 3. bölümde, ilerleyen kesimlerde kullanılacak Elektronik Harp örnek durum çalışması tanımlanmaktadır.

- 4. bölümde problem tanımı yapılmaktadır.

- 5. bölümde tasarım seçeneklerinin değerlendirilmesi için geliştirilen yaklaşım anlatılmaktadır. Bu kesimde ayrıca yaklaşımın adımlarını destekleyen modeller ve algoritmik çözümler açıklanmaktadır.

- 6. bölümde yaklaşımı destekleyen araçlar anlatılmaktadır.

- 7. bölümde, geliştirilen yaklaşımın Elektronik Harp Benzetimine uygulanması ele alınmaktadır.

- 8. bölümde geliştirilen yaklaşımın farklı bir tipte benzetime uygulanabilirliğini değerlendirmek için ikinci bir örnek durum çalışması olan trafik benzetimi tanımlanmakta ve geliştirilen yaklaşım bu benzetim sistemine uygulanmaktadır.

- 9. bölümde geliştirilen yaklaşımın başarımının değerlendirmesi yapılmaktadır.

- 10. bölümde geliştirilen yaklaşım farklı açılardan ele alınarak tartışılmaktadır.

- 11. bölümde ilişkili çalışmalar anlatılmaktadır.

- 12. bölümde çalışmanın sonuçları ve gelecek çalışmalar açıklanmaktadır.

2 TEMEL BİLGİLER VE BAĞLAM

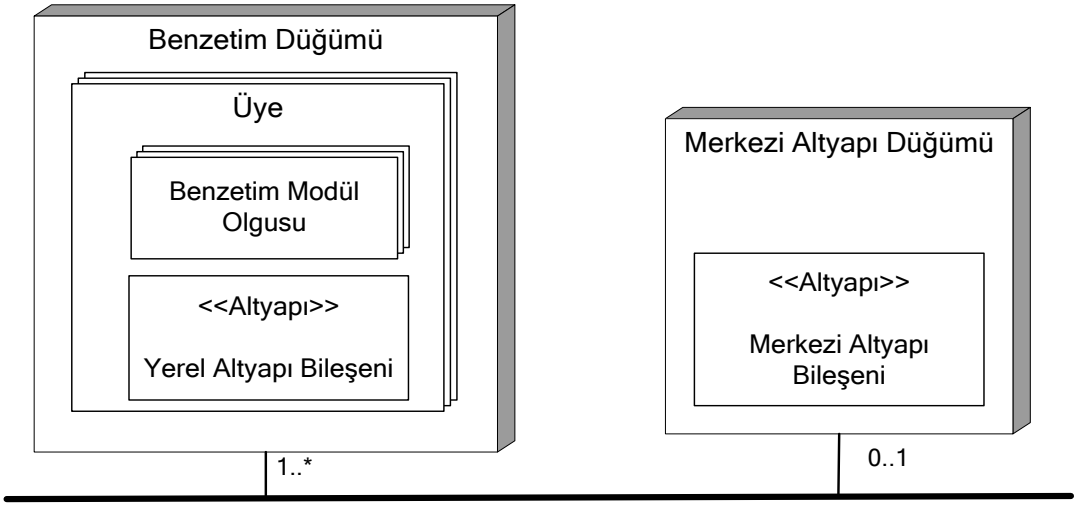
Bu kesimde, çalışma kapsamında geliştirilen yaklaşımın anlaşılabilmesi için gerekli temel bilgiler verilmiştir. 2.1 kesiminde Koşut ve Dağıtılmış Benzetimler için ortak referans mimari açıklanmıştır. 2.2 kesiminde temel DSEEP adımları anlatılmıştır. 2.3 kesiminde model güdümlü mühendislik hakkında bilgi verilmiştir.

2.1 Koşut ve Dağıtık Benzetim Sistemleri için Referans Mimari

Mevcut Koşut ve Dağıtılmış Benzetim mimarileri incelendiğinde, genel olarak ortak kavramları içerdikleri anlaşılmaktadır. DIS [IEEE 1998], HLA [Kuhl 1999; IEEE 2010a] ve TENA [Noseworthy 2008] gibi benzetim mimarileri incelenerek Şekil 2-1'de gösterilen referans mimari türetilebilir.

Tipik bir benzetim sistemi bir grup *Benzetim Düğümü (Simulation Node)* üzerine yerleştirilir. Her *Benzetim Düğümü* bir veya daha çok *Üye (Member)* içerir. *Üyeler* bir araya geldiklerinde benzetim koşumunu oluşturan uygulamalardır. Her *Üye,* belirli sayıda *Benzetim Modül Olgusu* (*Simulation Module Instance)* ve bir adet *Yerel Altyapı Bileşeni (Local Infrastructure Component)* içerir*. Benzetim Modül Olguları* benzetimdeki nesneleri ve olayları modelleyen benzetim bileşenleridir. Örneğin etmen tabanlı benzetim sistemlerinde her bir etmen bir benzetim modül olgusuna karşılık gelebilir. *Yerel Altyapı Bileşeni* veri değişimi ve ortak çalışmayı sağlamak için üyeler arasında çift yönlü etkileşimi sağlar*.*

Zorunlu olmamakla birlikte, benzetim sistemleri katılımcıların *Yerel Altyapı Bileşenlerine* ek olarak bir *Merkezi Altyapı Bileşeni* içerebilirler. *Merkezi Altyapı Bileşeni*, bulunduğu durumda benzetim yaşam döngüsünün yönetimi, zaman yönetimi, uyumlandırma (*synchronization*) işlemleri ve katılımcılar arası keşfetme (*discovery*) işlemlerinin yönetimi gibi benzetim geneli işlemleri yönetir. Bu işlemler *Merkezi Altyapı Bileşeninin* olmadığı mimarilerde, *Yerel Altyapı Bileşenlerinin* ortak sorumluluğunda olur. Yerel ve Merkezi Altyapı bileşenleri benzer hizmetleri sağladıklarından dolayı her ikisi de Şekil 2-1''de <<Altyapı>> etiketi ile gösterilmişlerdir.



Şekil 2-1 Koşut ve Dağıtılmış Benzetimler için Referans Mimari

HLA standardı, Şekil 2-1'de gösterilen referans mimariye büyük ölçüde uyumlu olan bir belirtim tanımlar. HLA bağlamında, *Üye'ler Federe* olarak adlandırılırlar. HLA standardına uyumlu benzetim sistemleri geliştirmek için altyapı sağlayan uygulamalar Koşum Zamanı Altyapısı *(Runtime Infrastructure-RTI)* olarak adlandırılırlar. HLA RTI'ları referans mimaride <<Altyapı>> olarak işaretlenmiş olan *Yerel Altyapı Bileşeni*'ne karşılık Yerel RTI Bileşeni *(Local RTI Component-LRC)* adı verilen bileşeni sağlarlar. Aslında HLA standardında tanımlı olmasa da, çoğu RTI gerçekleştirimi referans mimaride *Merkezi Altyapı Bileşeni*'ne karşılık Merkezi RTI Bileşeni *(Central RTI Component- CRC)* adı verilen bileşeni sağlar. *Federe*'ler *LRC* üzerinden *CRC* ve diğer *federeler* ile etkileşirler. Referans mimaride *Benzetim Modül Olgusu* olarak gösterilen bileşenler HLA mimarisinde federeler içerisinde tanımlanmış benzetim mantığı kesimlerine karşılık gelir. Bir federe tek bir varlığı/olayı modelleyebileceği gibi, farklı varlıkları/olayları

modelleyen alt benzetim bileşenleri tek bir federe içerisine yerleştirilebilir. HLA standardında tanımlanan federasyon yönetimi *(Federation Management)*, beyan yönetimi *(Declaration Management)*, nesne yönetimi *(Object Management),* sahiplik yönetimi *(Ownership Management)*, zaman yönetimi *(Time Management)* ve veri dağıtım yönetimi *(Data Distribution Management)* servisleri [IEEE 2010b] RTI'ların CRC ve LRC bileşenleri tarafından birlikte sağlanırlar.

DIS ve TENA mimarileri büyük ölçüde HLA ile benzerdir fakat bu mimarilerin koşum altyapılarında genellikle Merkezi Altyapı Bileşeni bulunmaz. Bundan öte, bu mimarilerin sağladığı servis grupları da HLA'dan farklıdır. Örneğin HLA standardında tanımlanmış olan zaman yönetimi servisi DIS ve TENA'da net olarak bulunmamaktadır.

Bahsedilen farklılıklara karşın, her üç mimari de mantık olarak Yayımla/Üye Ol *(Publish/Subscribe)* örüntüsünü [Eugster et al. 2003] sağlarlar. Yayımla/Üye Ol örüntüsü sayesinde veri üretici ve tüketici uygulamalar - yani Şekil 2-1'de verilen referans mimarideki adıyla *üyeler* – birbirine doğrudan bağımlı olmadan geliştirilebilir. Bu örüntüde üyeler, ürettikleri veriyi kimin tüketeceğini veya tükettikleri veriyi kimin ürettiğini bilmek zorunda değildir. Bu yaklaşım, benzetim sistemleri açısından oldukça önemli olan tekrar kullanılabilirlik *(reusability)* ve birlikte çalışabilirlik *(interoperability)* kalite faktörlerinin sağlanmasını ciddi anlamda destekler.

Yayımla/Üye Ol etkileşim modeli için gerekli altyapı desteği Şekil 2-1'de verilen referans mimaride <<altyapı>> olarak işaretlenen bileşenler tarafından sağlanır. Benzetim koşumunun üyeleri, <<altyapı>> bileşenleri tarafından sağlanan servisleri kullanarak veri değişim modeli elemanlarını yayımlayabilirler ve üye olabilirler. DIS, TENA ve HLA mimarilerinin veri değişim modelleri birbirinden farklıdır. DIS'in veri değişim modeli sabitlenmiş olan bir grup protokol veri biriminden *(Protocol Data Unit – PDU)* oluşur. HLA, DIS'in aksine, sabit bir veri değişim modelinin kullanımını zorunlu tutmaz. HLA standardının bir parçası olan Nesne Model Şablonu [IEEE 2010c], benzetim sisteminin ihtiyaçlarına uygun olarak Federasyon Nesne Modeli *(Federation Object Model – FOM)* ve Benzetim Nesne Modeli *(Simulation Object Model – SOM)* adı verilen veri değişim modellerini tanımlamaya olanak sağlar. TENA belirtimi ise, *Logical Range Object*

Model – LROM adı verilen genişletilebilir ve güncellenebilir ön tanımlı bir nesne modeli sağlar [Noseworthy 2008].

2.2 Dağıtık Benzetim Mühendisliği ve Koşum Süreci – DSEEP

Daha önce de belirtildiği gibi, DSEEP soyut bir süreçtir ve bilinçli olarak herhangi bir gerçekleştirim yöntemi sağlamaz. DSEEP süreci aşağıda verilen adımlardan oluşur [IEEE 2010d]:

(1) Benzetim Ortamı Amaçlarını Tanımla (Define Simulation Environment Objectives) – Benzetim ortamı için amaç ve hedefler paydaşlar tarafından tanımlanır.

(2) Kavramsal Analiz Yap (Perform Conceptual Analysis) – İlk önce senaryolar ve kavramsal model geliştirilir, ardından benzetim ortamı gereksinimleri ve yetenekleri tanımlanır.

(3) Benzetim Ortamını Tasarla (Design Simulation Environment) – Mevcut tekrar kullanılabilir üyeler belirlenir, yeni üyeler tanımlanır ve yetenekler üyelere atanır. Son olarak geliştirme planı tanımlanır.

(4) Benzetim Ortamını Geliştir (Develop Simulation Environment) – Veri değişim modeli tanımlanır, yeni üyeler geliştirilir ve tekrar kullanılacak bileşenler üzerinde gerekli güncellemeler yapılır.

(5) Benzetim Ortamını Entegre Et ve Sına (Integrate and Test Simulation Environment) – Benzetim bileşenleri bütünleştirilir ve sınanır.

(6) Benzetimi Koştur (Execute Simulation) – Benzetim sistemi çalıştırılır ve çıktısı bir sonraki adım için ön işlemeden geçirilir.

(7) Verileri Analiz Et ve Sonuçları Değerlendir (Analyze Data and Evaluate Results) – Bir önceki adımın sonucu olan koşum verisi analiz edilir ve benzetim amaçları ile karşılaştırılarak değerlendirilir.

Yukarıda tanımlanan adımları gerçekleştirmek için çeşitli çalışmalar yapılmıştır. Genellikle sağlanan çözümler bir adıma veya belirli bir grup adıma odaklanmıştır. Örneğin Base Object Model (BOM) [SISO 2006] ve [Karagöz ve Demirörs

2007]'de önerilen çözüm, DSEEP'in 2. adımındaki kavramsal model geliştirme aşamasına odaklanır. VR Forces [VT MAK 2010b] ve VR Vantage XR [VT MAK 2010c] araçları 4. adımı desteklerken, MAK Data Logger [VT MAK 2010a] ve Pitch Commander [Pitch Technologies 2009] araçları 5. adımı destekler.

Bu tezde anlatılan çalışma DSEEP'in 3. adımına odaklanır.

2.3 Model Güdümlü Mühendislik

Klasik, model güdümlü olmayan yazılım geliştirme yaklaşımlarında, uygulama kodu ile üst seviye tasarım modelleri arasında biçimsel bir bağlantı yoktur. Bu yaklaşımlarda uygulama kodu, tasarıma uygun olarak geliştiriciler tarafından el ile tanımlanır. Uygulamada ihtiyaç duyulan güncellemeler uygulama koduna el ile yansıtılır. Bu bağlamda tasarım modelinin de el ile güncellenmesi gerekmektedir. Uygulama koduna ek olarak tasarım modelini de sürekli el ile güncellemek bakım-idame maliyetleri anlamında etkin bir çözüm değildir ve pratikte tasarım ile kod bir süre sonra uyumsuz hale gelir. Model güdümlü mühendislik (Model-driven Engineering MDE) yaklaşımının tanımlanmasındaki temel motivasyonlardan birisi, bakım-idame maliyetlerini düşürmek ve uygulamanın maliyet etkin biçimde güncellenebilmesini sağlamaktır [OMG 2003; Frankel et al. 2004; Bezivin 2005; Schmidt 2006]. MDE bu hedefleri yerine getirmek için model ve metamodeller tanımlanmasına odaklanan bir yaklaşım tanımlar. MDE yaklaşımında modeller arası dönüştürümler için otomasyon desteği sağlanır. OMG, modelleri M0, M1, M2 ve M3 olarak seviyelendirir [OMG 2011b, OMG 2011c]. Çizelge 2-1'de model seviyeleri ve açıklamaları verilmiştir.

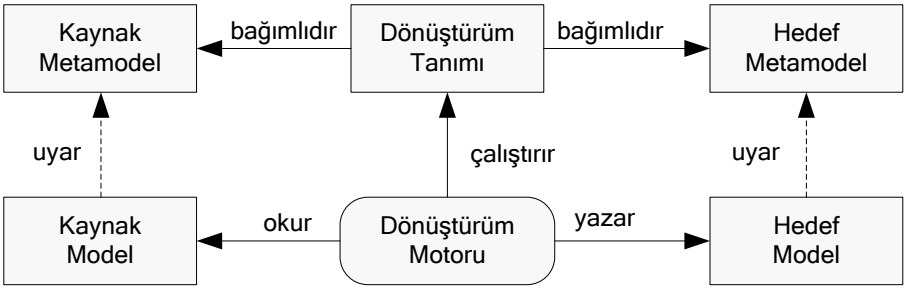
MDE yaklaşımda değişen bir gereksinimi uygulamaya yansıtmak için hem uygulama kodunu hem de modeli değiştirmek yerine sadece model güncellenerek uygulama kodu modelden tekrar üretilir. Böylece bakım-idame maliyetleri önemli ölçüde düşürülür.

MDE yaklaşımını gerçekleştirmek için OMG tarafından Model Güdümlü Mimari (Model Driven Architecture – MDA) tanımlanmıştır [OMG 2003]. MDA, UML [Fowler 2003; OMG 2011b], XMI [OMG 2011a], MOF [OMG 2011c] gibi diğer OMG standartları üzerine inşa edilmiş bir mimaridir [Bezivin 2005].

Çizelge 2-1 Model Seviyeleri ve Açıklamaları

Seviye	Model Seviyesi	Açıklama
M0	Sistem	Geliştirilen sistemin olgusudur.
M1	Model	Geliştirilen sistemin tasarımıdır. M0 Seviyesinin modelidir.
M2	Metamodel	Geliştirilen sistemin tasarımında kullanılan modellerin metamodelidir. M1seviyesinin modelidir.
M3	Meta-metamodel	Sistem tasarımında kullanılan modellerin metamodellerinin uyduğu metamodeldir. M2 seviyesinin modelidir. Örnekler: • UML metamodelinin metamodeli olan MOF [OMG 2011c] • Eclipse Modelleme ortamının metamodeli ECore [Gronback 2009] • ISIS Generic Modeling Environment (GME) aracının metamodeli metaGME [ISIS 2012]

MDE yaklaşımında, hedef sistemin modelden (yarı) otomatik olarak türetilebilmesi için model dönüştürümleri kullanılır. Model dönüştürümleri için genel yaklaşım Şekil 2-2'de verilmiştir [Bezivin 2005; Czarnecki and Helsen 2006]. Şekilde gösterilen Kaynak Model, Dönüştürüm Motoru'na girdi olarak sağlanır. Dönüştürüm Motoru, öntanımlı Dönüştürüm Tanımı'nı kullanarak Hedef Model'i üretir. Kaynak ve Hedef modeller tanımlı metamodellerine uyarlar.



Şekil 2-2 MDE Model Dönüştürüm Yaklaşımı

Temel model dönüştürüm yöntemleri aşağıdaki gibi özetlenebilir:

2.3.1 Modelden modele dönüştürüm (Model to model transformation – M2M)

Bu dönüştürüm modelinde [M2M 2012], adından da anlaşılacağı gibi, bir model amaca göre başka bir modele otomatik olarak dönüştürülür. Bu dönüştürüm yöntemi genellikle Model Güdümlü Mimari'de Platform Bağımsız Modellerden *(Platform Independent Model – PIM)* Platforma Özel Modellere *(Platform Specific Model – PSM)* otomatik geçiş için kullanılır [Bezivin 2005].

Modelden modele dönüşüm model kaynak dosyaları herhangi bir programlama dili ile işlenerek gerçekleştirilebileceği gibi, M2M dönüşümü için özel olarak tasarlanmış Eclipse Modelleme ortamının parçaları olan "Atlas Transformation Language (ATL)" [ATL 2012] veya "Query/ View/ Transformation (QVT)" [QVT 2012] gibi araçlar da kullanılabilir [Gronback 2009].

2.3.2 Modelden metne dönüştürüm (Model to text transformation – M2T)

Bu dönüştürüm yaklaşımı [M2T 2012], tasarım modelinden değişik hedef programlama dilleri (C++, Java, vs.) için otomatik kod üretimi veya belgeleme amaçlı metin üretme gibi ihtiyaçların karşılanması için tanımlanmıştır.

Modelden metne dönüşüm model kaynak dosyaları herhangi bir programlama dili ile işlenerek gerçekleştirilebileceği gibi, M2T dönüşümü için özel olarak tasarlanmış Eclipse platformunun parçaları olan Java

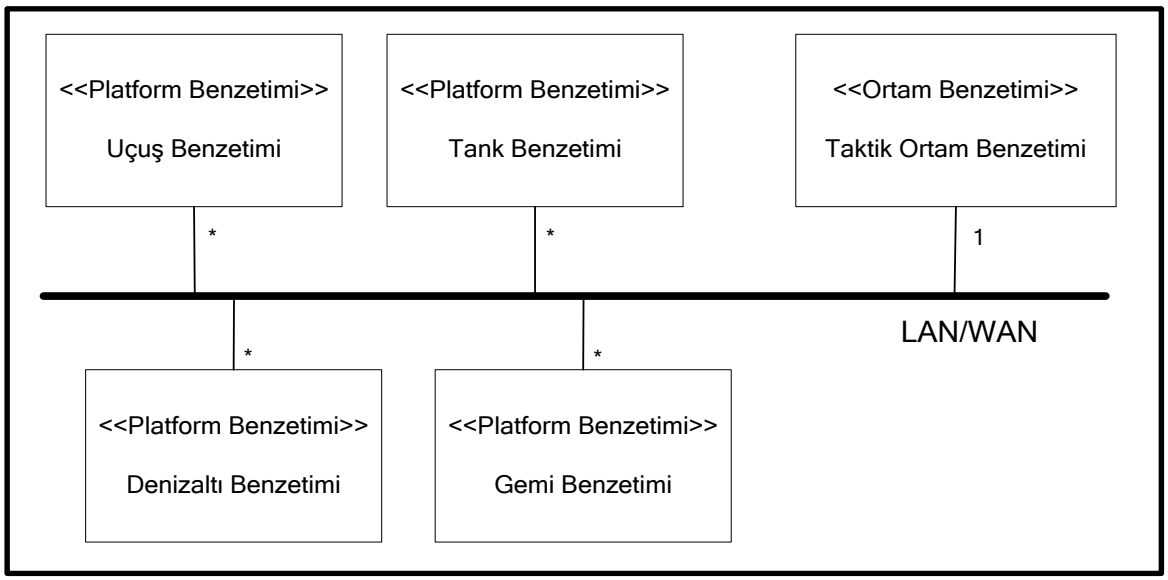
Emitter Templates (JET) [JET 2012] veya XPand [XPand 2012] gibi araçlar da kullanılabilir [Gronback 2009].

2.3.3 Metinden modele dönüştürüm (Text to model transformation – T2M)

T2M dönüştürümlerinde, metin dosyalarından belirli metamodellere uygun hedef modeller üretilebilir. XText [XText 2012] ve Gra2Mol (Grammar to Model Language) aracı [Gra2Mol 2012 ; Izquierdo and Molina 2009] T2M dönüştürümleri için kullanılabilecek ortamlara örnektir.

3 ÖRNEK DURUM ÇALIŞMASI - Geniş Ölçekli Bir Elektronik Harp Benzetimi

Bu kesimde tez çalışmasının ilerleyen kesimlerinde kullanılacak bir örnek durum çalışması anlatılmıştır. Örnek çalışma geniş ölçekli bir Elektronik Harp (EH) [Adamy 2001] benzetimi tanımlar. EH benzetimleri, gerçek tatbikatlarda uygulanması oldukça zor, pahalı ve tehlikeli olan durumların ve olayların sanal bir ortamında denenmesine olanak sağlamaları bağlamında savunma sanayi açısından oldukça önemlidir [Adamy 2006]. Örnek çalışma ilerleyen kesimlerde problem tanımının netleştirilmesi ve geliştirilen yaklaşımın kullanımının örneklenmesi için kullanılacaktır.



Şekil 3-1 Örnek Çalışmanın Mantıksal Görünümü

Örnek çalışmanın mantıksal görünümü Şekil 3-1'de verilmiştir. Örnek çalışmaya konu olan sistem, birbirleri ve bir taktik çevre benzetimi ile etkileşen platform benzetimlerinden oluşur. Bir platform benzetimi, bir platformun fiziksel ve davranışsal özelliklerini modelleyen bir sistemdir. Bir platform benzetimi, hareket modeli, 3 Boyutlu (3B) görselleştirme ve radar, iletişim cihazları gibi özel donanımların modellerini içerir. Bu örnek durum çalışmasında, uçuş, tank, denizaltı ve gemi olmak üzere dört değişik platform benzetimi türü vardır. Şekilde platform benzetimlerinin sayısı için net bir değer verilmemiştir, bunun yerine çokluk belirteci olarak '*' kullanılarak her platform türü için sıfır veya daha fazla sayıda benzetim olabileceği ifade edilmiştir. Platform benzetimlerinin net sayısı senaryolar arasında farklılık gösterebilir. Örneğin bu kesimin sonunda açıklanacak olan somut

senaryoda her platform benzetimi için net sayı verilmiştir. Taktik Çevre Benzetimi ortamdaki diğer platformlar (ör: uçak, gemi, tank, denizaltı, vs.), EH sistemleri (ör: radar, füzeler, vs.) ve çevresel koşullar (ör: hava durumu, gün saati, vs) için modeller içerir.

2.2 kesiminde de belirtildiği gibi, DSEEP'in ilk adımı benzetim ortamının amaçlarının tanımlanmasıdır. Örnek durum çalışması özelinde, bu adım gemi, uçuş, denizaltı, tank ve taktik çevre benzetimlerinin amaçlarının tanımlanması gibi alt adımları içerir.

Bu kesimin devamında öncelikle örnek durum çalışması için daha detaylı benzetim ortamı amaçları verilecek, sonrasında örnek bir senaryo tanımı yapılacaktır.

3.1 Örnek Durum Çalışması İçin Benzetim Ortamı Amaçlarının Tanımlanması

Bu kesimde, örnek durum çalışması için benzetim ortamı amaçları tanımlanacaktır. Benzetim sistemi aşağıdaki yetenekleri sağlamalı ve bu yetenekleri kullanarak değişik senaryolar tanımlamayı mümkün kılmalıdır:

- Benzetim sistemi, aşağıdaki alt benzetim modellerini içeren gemi benzetimleri sağlamalıdır:
    - Gemi dinamiklerini modellemek için bir gemi modeli
    - Gemi donanımlarını modellemek için aşağıdaki sistem modelleri:
        - Seyir Radarı,
        - Gözetleme Radarı,
        - Silah Kontrol Sistemi,
        - Satıhtan Satha (SS) RF/EO güdümlü füzeler,
        - Satıhtan Havaya RF/EO güdümlü füzeler,
        - Radar Elektronik Destek (ED) Sistemleri,
        - Link 11 tabanlı muhabere sistemleri.

        - Karşı Tedbir Atma Sistemi *(Counter Measure Dispenser System – CMDS)*

- Benzetim sistemi, aşağıdaki alt benzetim modellerini içeren uçuş benzetimleri sağlamalıdır:
    - Uçak dinamiklerini modelleyen bir uçuş modeli
    - Uçak donanımlarını modellemek için aşağıdaki sistem modelleri:
        - Takip radarı,
        - Silah Kontrol sistemi,
        - Aviyonik modeli,
        - Havadan Satha (HS) RF/EO güdümlü füzeler
        - Havadan Havaya (HH) RF/EO güdümlü füzeler,
        - Radar İkaz Alıcısı Sistemi *(Radar Warning Receiver – RWR)*,
        - Füze İkaz Sistemi *(Missile Approach Warning System – MAWS),*
        - Lazer İkaz Alıcı Sistemi *(Laser Warning Receiver – LWR)*,
        - Link 16 tabanlı muhabere sistemi,
        - Karşı Tedbir Atma Sistemi *(CMDS)*,
- Benzetim sistemi, aşağıdaki alt benzetim modellerini içeren denizaltı benzetimleri sağlamalıdır:
    - Denizaltı dinamiklerini modellemek için bir denizaltı modeli
    - Denizaltı donanımlarını modellemek için aşağıdaki sistem modelleri:
        - Sonar
        - Silah kontrol sistemi,

        - RF güdümlü HARPOON füzeleri,
        - Link 22 tabanlı muhabere sistemi.

- Benzetim sistemi, aşağıdaki alt benzetim modellerini içeren tank benzetimleri sağlamalıdır:
    - Tank dinamiklerini modellemek için bir tank modeli
    - Tank donanımlarını modellemek için aşağıdaki sistem modelleri:
        - Silah kontrol sistemi,
        - Kızıl Ötesi Karşı Tedbir Sistemi *(Infra-red Counter Measure – IRCM)*
        - Yönlendirilmiş Kızıl Ötesi Karşı Tedbir Sistemi *(Directed Infra-red Counter Measure – DIRCM)*
        - Füze İkaz Sistemi (MAWS)
        - Lazer İkaz Alıcı Sistemi (LWR)
        - Muhabere Sistemleri
- Benzetim sistemi, aşağıdaki alt benzetim modellerini içeren bir taktik çevre benzetimi sağlamalıdır:
    - RF Yayılım Modeli: Bu model tüm RF Elektromanyetik yayılımların benzetimini sağlamalıdır.
    - EO/Lazer Yayılım Modeli: Bu model tüm EO/Lazer yayılımların benzetimini sağlamalıdır.
    - Akustik Yayılım Modeli: Bu model tüm su altı akustik yayılımların benzetimini sağlamalıdır.
    - Hava durumu benzetimi modeli
    - Arazi benzetimi modeli

- Bilgisayar Üretimi Kuvvetler *(Computer Generated Forces – CGF)* Modeli: Bu model, gemi, denizaltı, uçuş ve tank benzetimlerinin etkileşeceği taktik ortamda bulunan platformlar/sistemler için benzetim modelleri sağlar. Taktik ortam platform/sistem modelleri aşağıdaki gibi listelenebilir:
    - Platformlar:
        - Hava Platformları
        - Füzeler
            - Havadan Havaya RF/EO Güdümlü,
            - Havadan Satha RF/EO Güdümlü,
            - Satıhtan Havaya RF Güdümlü,
            - Satıhtan Satha RF Güdümlü.
        - Kara araçları
        - Su üstü platformlar
        - Denizaltı modelleri.
    - Sistemler:
        - Radarlar
            - Gözetleme Radarı,
            - Sürekli Dalga *(Continuous Wave – CW)* Takip Radarı,
            - Darbeli Takip Radarı,
        - Elektronik Destek (ED) Sistemleri
            - Radar Elektronik Destek (ED) sistemi,

    - Muhabere Elektronik Destek (ED) Sistemi,
- Elektronik Taarruz Sistemleri
    - Radar Elektronik Taarruz (ET) Sistemi,
    - Muhabere Elektronik Taarruz (ET) Sistemi,
- İkaz Sistemleri
    - Radar İkaz Alıcı (RWR) Sistemi,
    - Füze İkaz Sistemi (MAWS) Sistemi,
    - Lazer İkaz Alıcı (LWR) Sistemi,
- Aldatma Sistemleri
    - Radar Dekoy,
    - Çekili *(Towed)* RF Dekoy,
- SAM *(Surface to Air Missile)* Sistemi,
- Komuta Kontrol Sistemi (C2),
- AAA *(Anti Aircraft Artillery)* Sistemi.

Bu kesimde benzetim ortamının amaçları ana başlıklarıyla anlatılmıştır. Bu noktada, gereksinimler detaylandırılmalı ve belgelendirilmelidir. Ortam amaçlarının detaylandırılması ve belgelendirilmesi esnasında Rational Unified Process – RUP [Kruchten 2003] gibi genel amaçlı süreçlerin ilgili kesimleri izlenebilir ve UML kullanım durumu çizenekleri *(Use Case Diagram)* [Fowler 2003] gibi çeşitli araçlar kullanılabilir. Klasik yazılım mühendisliği yaklaşım ve araçların kullanımına alternatif bir yöntem olarak, benzetim sistemlerinin görev uzayı seviyesinde *(Mission Space)* kavramsal modellerinin oluşturulması için özel olarak tasarlanmış yöntemler ve araçlar da kullanılabilir [Karagöz ve Demirörs 2007].

3.2 Örnek Çalışma İçin Örnek Bir Senaryo Tanımı Oluşturulması

Benzetim ortamının amaçlarının tanımlanmasından sonra, DSEEP'in bir sonraki adımı olan benzetim sisteminin kavramsal analizinin yapılmasına geçilir. Kavramsal analiz sürecinin önemli bir adımı örnek senaryoların tanımlanmasıdır. Bir senaryo daha önceden tanımlanan benzetim ortamı amaçlarına uygun olarak temel benzetim unsurlarının türleri ve sayılarını içerir.

Çizelge 3-1: Elektronik Harp Durum Çalışması için Örnek Bir Senaryo Tanımı

Benzetim Katılımcısı	**Sayı**	**Benzetim Katılımcısı Alt Unsurları**	**Toplam Unsur Sayısı**
Gemi Benzetimi	10	1 gemi modeli 8 EH sistemi 5 Satıhtan Havaya EO füze 10 Satıhtan Satha RF füze	240
Denizaltı Benzetimi	8	1 denizaltı modeli 10 EH sistemi 10 Satıhtan Satha RF füze	168
Uçuş Benzetimi	12	1 uçuş modeli 3 EH sistemi 5 Havadan Havaya EO füze 2 Havadan Satha RF füze	132
Tank Benzetimi	15	1 tank modeli 4 EH Sistemi	75
Taktik Çevre Benzetimi	1	40 uçak 20 gemi 10 denizaltı 100 tank 10 Komuta Kontrol (C2) Sistemi, 1 RF Yayılım Modeli 300 çeşitli türlerde füze 500 EH Sistemi (Radar, Elektronik Destek-ED sistemi, Elektronik Taarruz-ET sistemi, vs)	981

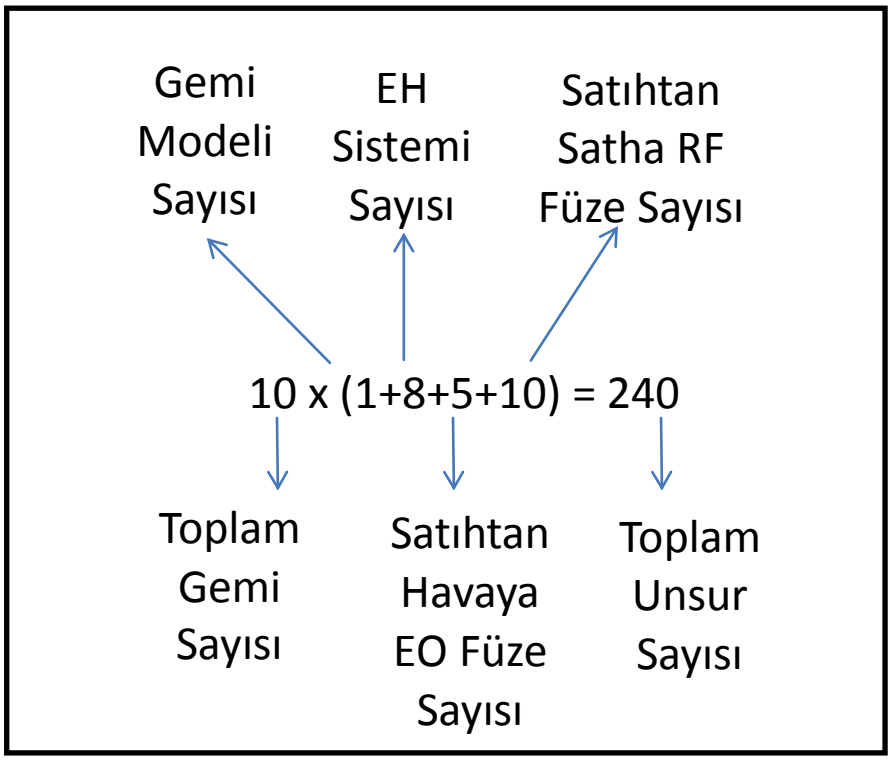
Çizelge 3-1'de örnek durum çalışması için örnek bir senaryo tanımı verilmiştir. Çizelgenin 'Benzetim Katılımcısı' sütunu, bir araya gelerek benzetim sistemini oluşturan katılımcıları gösterir.

Çizelgenin 'Sayı' sütunu ise, ilgili katılımcının örnek senaryoda kaç adet olgusu olduğunu gösterir. Örneğin Çizelge 3-1'de verilen senaryoda 10 adet gemi benzetimi olduğu tanımlanmıştır.

Çizelgenin 'Benzetim Katılımcısı Alt Unsurları' sütunu, ilgili benzetim katılımcısını gerçekleştirmek için gerekli olan alt unsur modellerini gösterir. Her benzetim katılımcısının kendine özel alt unsur modelleri vardır. Örneğin Çizelge 3-1'de verilen senaryoda, gemi benzetimi için alt unsur olarak 1 gemi modeli, 8 EH sistemi, 5 Satıhtan Havaya Elektro-Optik (EO) güdümlü füze ve 10 Satıhtan Satha Radyo Frekansı(RF) güdümlü füze içerir.

Çizelgenin son sütunu olan 'Toplam Unsur Sayısı' senaryodaki ilgili benzetim katılımcısının olgularının toplam alt unsur sayısını gösterir. Bu sayı benzetim katılımcı sayısı toplam Benzetim Katılımcısı Alt Unsur sayısı ile çarpılarak bulunur. Örneğin, 10 gemi benzetimi için toplam unsur sayısı 10 x (1+8+5+10) = 240 olarak hesaplanır. Bu hesabın detayları Şekil 3-2'te verilmiştir.

Çizelge 3-1'te her bir benzetim katılımcısı için toplam unsur sayısı verilmiştir. Senaryodaki toplam unsur sayısı tüm benzetim katılımcılarının toplam unsur sayısı toplanarak hesaplanabilir. Örneğin Çizelge 3-1'te tanımlanan senaryonun toplam unsur sayısı 240+168+132+75+981 = 1596'dır. Bu örnek senaryodan da gözlemlenebileceği gibi, bir benzetim sistemi koşum senaryosundaki toplam unsur sayısı oldukça fazla olabilir.

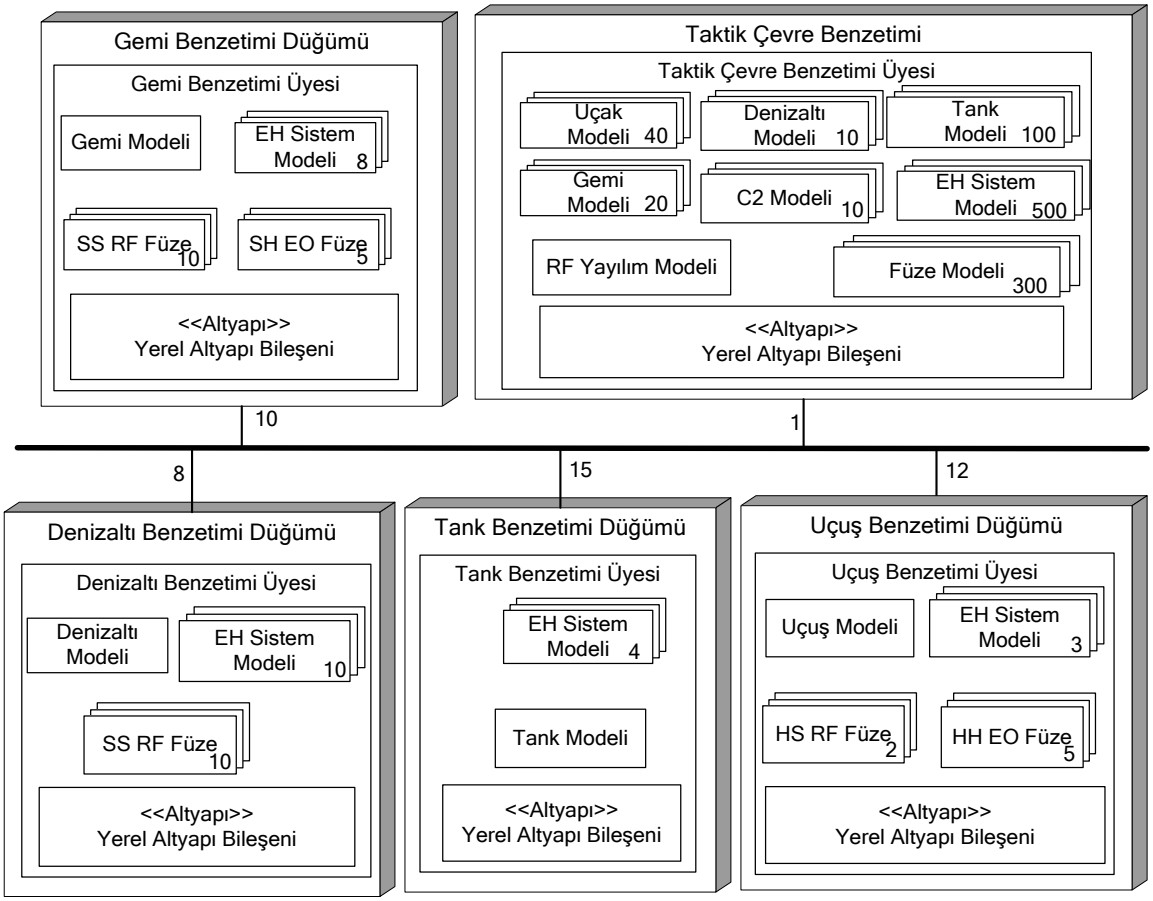


Şekil 3-2 Toplam Unsur Sayısı Hesabı Örneği

4 PROBLEM TANIMI

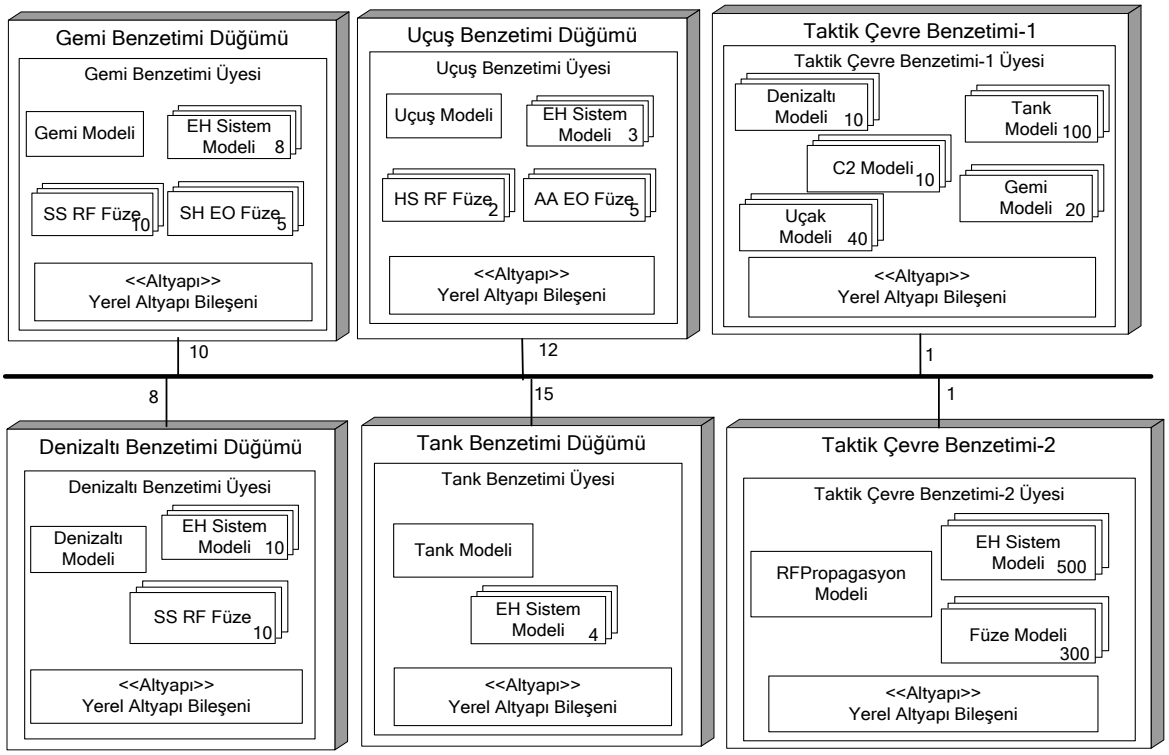
Senaryo tanımı yapılıp ve kavramsal analiz aşaması tamamlandıktan sonra, DSEEP'in 3. adımı olan benzetim sisteminin tasarlanmasına başlanabilir. Şekil 2-1'de verilen referans mimari ve Çizelge 3-1'de verilen senaryo kullanılarak çeşitli yerleştirme seçenekleri türetilebilir. Bir yerleştirme seçeneği, senaryodaki benzetim unsur (*entity*) modeli olgularının benzetim düğümlerine ve üyelere eşlenmesini tanımlar. Bu çalışma kapsamında, her benzetim düğümü için, üzerinde bu düğüme yerleştirilmiş tüm benzetim modülü olguları için yuva (*container*) görevi gören sadece bir adet üye tanımlandığı varsayılmıştır. Her düğüm için sadece bir üye tanımlanması yöntemi, özellikle çok sayıda benzetim modül olgusunun olduğu durumlarda performans açısından önemli olan işler arası iletişim yükünü azaltmak için seçilmiştir [Lees et al. 2007].

Örnek bir yerleştirme seçeneği Şekil 4-1'de verilmiştir. Bu yerleştirme seçeneğinde, her platform benzetimi ve taktik çevre benzetimi ayrı düğümlere yerleştirilmiştir. Her platform benzetimi düğümü platformun kendisini, taşıdığı EH sistemlerini ve diğer sistemleri modeller. Aslında bu seçenek her benzetimin ayrı bir düğüme yerleştirilmesi açısından mantıksal bölümlemeye uygundur. Bunun yanı sıra, tek bir merkezi taktik çevre benzetimi tanımlanmış olması, taktik çevre unsurları arasındaki iletişim maliyetini azaltıcı bir etkendir. Bu seçeneğin mantıksal bir bölümlemeye dayanmasından dolayı kolay anlaşılabilir olmasına karşın, her zaman etkin bir çözüm değildir. Örneğin platform benzetimleri ile taktik çevre benzetimine yerleştirilmiş EH sistem modelleri arasında sık iletişim olduğu durumda bu yerleştirme seçeneği etkinliğini kaybedecektir.



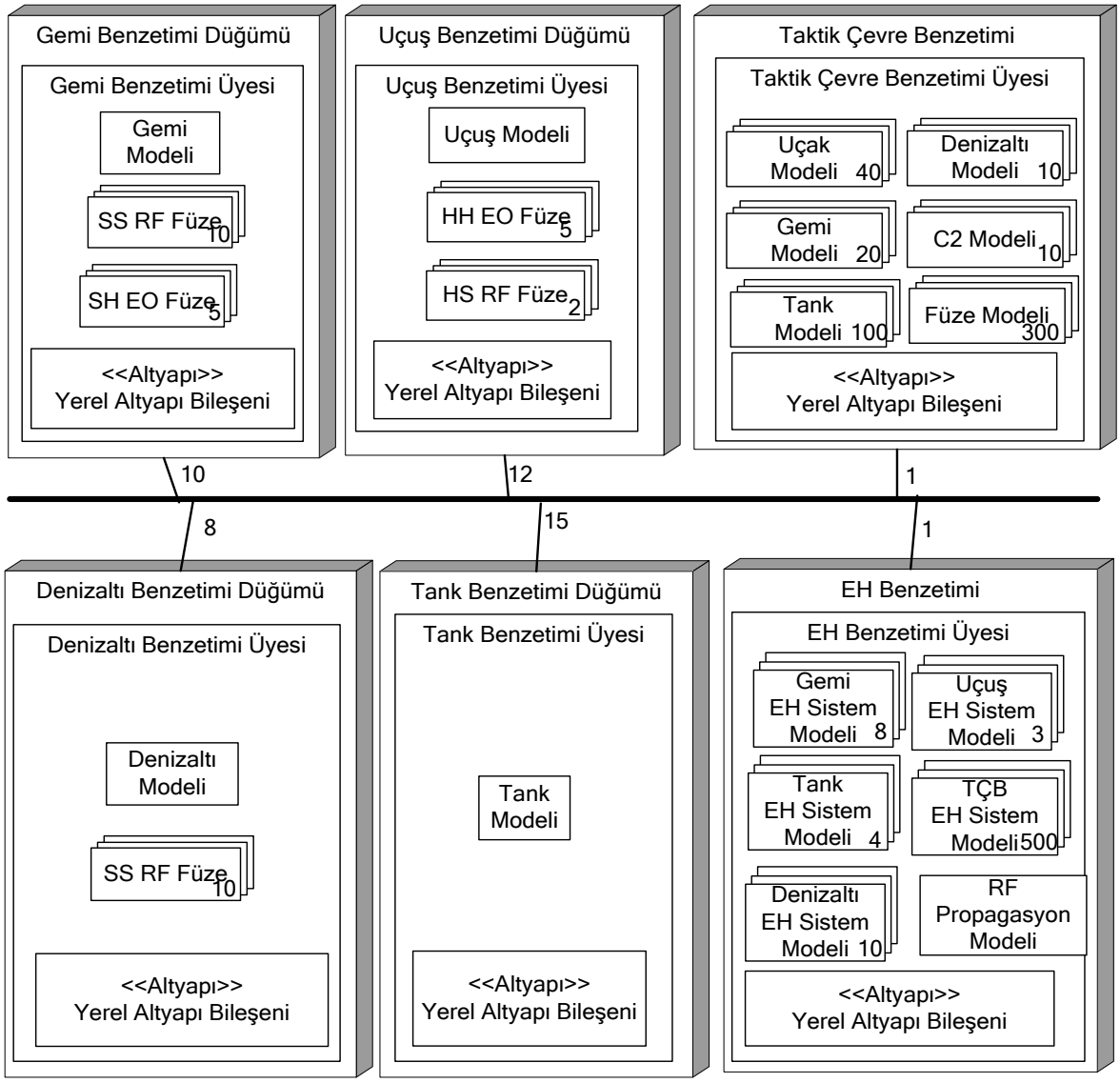
Şekil 4-1 Çizelge 3-1'de Verilen Senaryo için Yerleştirme Seçeneği – Bir Taktik Çevre Benzetimi ve Dağıtılmış EH Modelleri

İkinci bir örnek yerleştirme seçeneği Şekil 4-2'de gösterilmiştir. Bu seçenekte, öncelikle her platform benzetimi bir önceki seçenekte olduğu gibi ayrı bir düğüme yerleştirilmiştir. Bir önceki seçenekten farklı olarak, taktik çevre benzetimi yük bölümleme (*load partitioning*) amacıyla iki düğüme bölünmüştür. Aynen bir önceki yerleştirme seçeneğinde olduğu gibi, her bir platform benzetiminin ayrı düğüme yerleştirilmiş olması mantıksal açıdan kolay anlaşılır bir mimari ortaya çıkarmıştır. Taktik Çevre Benzetimi'nin (TÇB) ikiye bölünmüş olması, kullanılan işlem gücünü artırır, ama diğer taraftan farklı düğümlere yerleştirilmiş taktik çevre unsur modelleri arasındaki iletişim maliyetini de artırır.



Şekil 4-2 Çizelge 3-1'de Verilen Senaryo için Yerleştirme Seçeneği – İki Taktik Çevre Benzetimi ve Dağıtılmış EH Modelleri

Üçüncü bir örnek yerleştirme seçeneği Şekil 4-3'te gösterilmiştir. İlk iki yerleştirme seçeneğinde olduğu gibi bu yerleştirme seçeneğinde de platform benzetimleri ayrı düğümlere yerleştirilmiştir. İlk iki seçenekten farklı olarak, platform benzetimleri ve taktik çevre benzetimlerinin tüm EH sistemi modelleri ayrı bir düğümde toplanmıştır. Böylece, EH sistem modelleri arasındaki iletişim maliyeti azalmakta ama aynı zamanda EH sistem modellerinin platform benzetimleri ve taktik çevre benzetimi üzerine yerleştirilmiş olan diğer benzetim unsurlarıyla iletişim maliyeti artmaktadır.



Şekil 4-3 Çizelge 3.1'de Verilen Senaryo için Yerleştirme Seçeneği – Bir Taktik Çevre Benzetimi ve Tek Düğüme Yerleştirilmiş EH Modelleri

Yukarıda verilen üç seçenek gibi pek çok farklı yerleştirme seçeneği türetilebilir. Bu yerleştirme seçenekleri benzetim düğümü sayısı, benzetimlerin düğümlere eşlenmesi, benzetim unsurlarının düğümlere yerleştirilmesi gibi açılardan değişiklik gösterebilir. Farklı seçenek yerleştirmelerin sayısının oldukça büyük olacağı ve bu yerleştirme seçeneklerinin anlaşılabilirlik için mantıksal bölümleme, iletişim maliyetini asgariye indirme, fiziksel kaynakların kullanımını iyileştirme gibi farklı kalite faktörlerine göre etkinliklerinin değişik olacağı aşikârdır.

DSEEP'e göre, tasarım aşamasında yerleştirme seçeneklerinin belirlenmesi ve bunların performanslarının değerlendirilmesi önemlidir. Daha somut olarak,

DSEEP'in "Benzetim Ortamını Tasarla" adımında yapılması önerilen iki iş aşağıda verilmiştir [IEEE 2010d]:

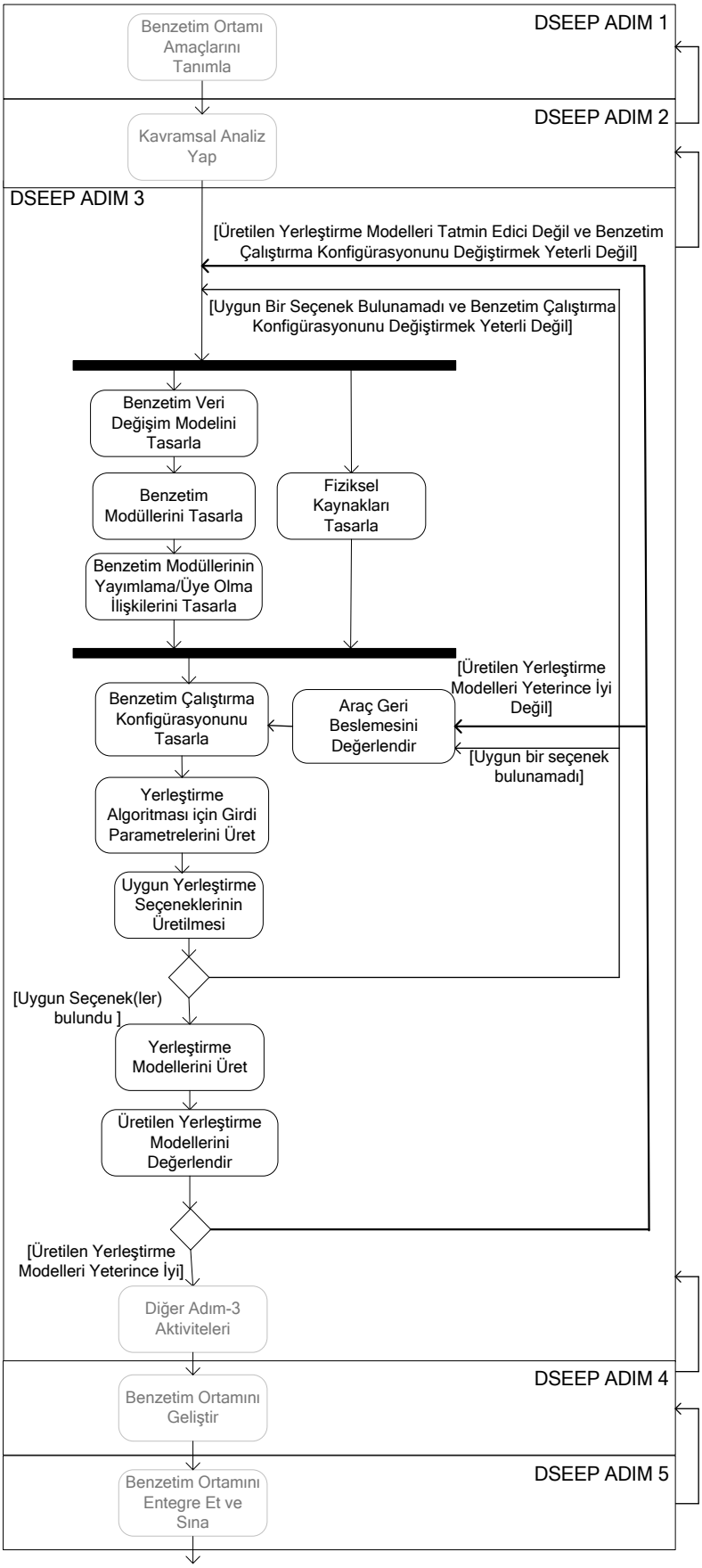
- Tasarım seçeneklerini değerlendir ve verilen gereksinimleri en iyi şekilde karşılayan benzetim ortamı tasarımını belirle.
- Benzetim ortamı başarımını tahmin et ve başarım gereksinimlerini karşılamak için herhangi bir işleme gerek olup olmadığına karar ver

DSEEP bilinçli olarak soyut bir süreç olarak tanımlandığından, yukarıda verilen işlerin ne şekilde gerçekleştirileceğine dair herhangi bir tanımlama ve kısıtlama içermez. Bundan öte, literatürde bu işleri yeteri ölçüde destekleyecek araç desteği henüz tanımlanmamıştır. Daha önce de belirtildiği gibi, tasarım seçeneklerinin değerlendirilmesi ve başarım kestirimi işleri ya geliştirme aşamasına ertelenir ya da tasarım aşamasında uzman değerlendirmesi ile gerçekleştirilir. Ne var ki bu işlerin geliştirme aşamasına ertelenmesi etkin olmayan gerçekleştirimlerin yapılmasına neden olabilir. Etkin olmayan gerçekleştirimler, tasarım ve ilişkili proje yaşam döngüsü elemanlarının (detaylı tasarım, gerçekleştirim, sınama elemanları, belgeleme, vs.) gereksiz yinelemeler ile tekrar tekrar güncellenmesine gerektirir. Bu durum, projenin gecikmesine ve maliyetinin yükselmesine sebep olur. Öte yandan, geliştirilen sistem karmaşıklaştıkça uzman değerlendirmesi giderek zorlaşır. Bu bilgiler ışığında, izleyen kesimde uygun yerleştirme seçeneklerinin bulunmasına yönelik sistematik bir yaklaşım anlatılacaktır.

5 UYGUN YERLEŞTİRME SEÇENEKLERİNİN TANIMLANMASI İÇİN BİR YAKLAŞIM

Bu kesimde uygun yerleştirme seçeneklerinin tanımlanması ve değerlendirilmesi için bu tez çalışması kapsamında geliştirilen ve DSEEP'e eşlenen somut bir yaklaşım anlatılmaktadır. Geliştirilen yaklaşım Şekil 5-1'deki aktivite çizeneğinde gösterilmiştir. Yaklaşım geliştirilirken DSEEP'in ilk iki adımının hâlihazırda literatürde mevcut olan yaklaşımlar kullanılarak gerçekleştirildiği varsayılmıştır (ör. [SISO 2006]). Bundan dolayı ilk iki adım şekilde silik gösterilmiştir. Daha önce de belirtildiği gibi, bu çalışma kapsamında DSEEP'in 3. Adımı olan tasarım aşamasına odaklanılacaktır. Bu noktada, nihai yerleştirme modelinin DSEEP'in 3, 4 ve 5. adımları üzerinde yapılacak birkaç yineleme (*iteration*) ile bulunabileceği belirtilmelidir. İlk yerleştirme modeli 3. adımda prototiplenir, 4. adımda bu yerleştirme modeline uygun geliştirme yapılır ve 5. adımda sistem test edilir. Geliştirme ve Test adımlarında elde edilen bulgular önceki adımlara geri besleme olarak aktarılır ve uygun yerleştirme modeli yinelemeli olarak iyileştirilir. DSEEP'in yinelemeli yapısı Şekil 5-1'de ana adımlar arasındaki ters yönlü oklarla gösterilmiştir.

Bu çalışma kapsamında geliştirilen yaklaşımın her bir alt adımı için somut bir gerçekleştirim yöntemi tanımlanmıştır. İlerleyen alt kesimlerde geliştirilen yaklaşımın alt adımlarını gerçekleştiren somut aktiviteler anlatılacaktır.



Şekil 5-1 Uygun Yerleştirme Modeli Bulma Akışı

5.1 Benzetim Veri Değişim Modelini Tasarla

Benzetim Veri Değişim Modelini tasarlama adımında, benzetim modülleri arasındaki veri paylaşımını tanımlamak için kullanılacak olan DSEEP'teki adıyla Benzetim Veri Değişim Modeli-BVDM (*Simulation Data Exchange Model- SDEM*) tanımlanır.

Aslında DSEEP'e göre BVDM'nin tanımlanması 4. Adım olan *Benzetim Ortamını Geliştir (Develop Simulation Environment)* adımının bir alt parçasıdır. DSEEP'ten farklı olarak, bu tez kapsamında tanımlanan süreçte BVDM'nin tanımlanması 3. adım olan tasarım aşamasına çekilmiştir çünkü BVDM'nin tanımlanmış olması benzetim modülleri arasındaki yayımla/üye ol ilişkilerinin tasarlanabilmesi için ön koşuldur. Aslında BVDM'nin tanımlanması için ihtiyaç duyulan temel bilgiler DSEEP'in 2. Adımı ve 3. Adımının alt adımlarında tanımlanmış olduğundan, BVDM'nin tanımlanmasına 3. Adımda başlanmasında herhangi bir sakınca yoktur.

BVDM tanımı olarak *Real-time Platform Reference Federation Object Model (RPR-FOM)* [SISO 2001a;2001b] gibi standart veri modeli tanımları kullanabileceği gibi, standart nesne modelleri genişleterek kullanabilir veya DSEEP'te tanımlanan yöntemlerden birisi olan sıfırdan BVDM geliştirme yaklaşımını seçilebilir. Bu bağlamda, HLA OMT [IEEE 2010c] standardı BVDM'lerin üretilmesi için standart bir metamodel tanımlar. BVDM elemanlarını tanımlamak için HLA OMT belirtimi (1) Nesne Sınıfları *(Object Classes)*, (2) Etkileşim Sınıfları *(Interactions)* ve (3) Veri Türleri *(Data Types)* olmak üzere üç temel eleman sağlar.

Örneğin EH örnek durum çalışmasında, Nesne Sınıfları deniz platformu, hava platformu, tank, radar gibi nesneleri tanımlamak için kullanılırlar. Etkileşim Sınıfları ise, benzetim katılımcıları arasındaki mesajlaşmaları tanımlamak için kullanılırlar. Örneğin, EH benzetimleri için mühimmat patlaması, sistem durum değişikliği (kapalı/açık gibi), füze ateşlenmesi gibi etkileşim sınıfları tanımlanabilir. Son olarak Veri Türleri, nesne sınıflarının niteliklerini ve etkileşim sınıflarının parametrelerini tanımlamak için kullanılırlar. Örneğin "*Platform*" Nesne Sınıfı *Position3D* adında platformun konumunu gösteren bir niteliğe sahip olabilir. Bir diğer Veri Türü örneği olarak, *SystemStatusChange* Etkileşim Sınıfının, sistemin hedef durumunu gösteren, değeri "açık" veya "kapalı" olabilen *systemState* parametresi gösterilebilir.

Bu tez çalışması kapsamında, Benzetim Veri Değişim Metamodeli tanımlamak için HLA OMT standardı [IEEE 2010c] temel alınarak genişletilmiştir. Benzetim Nesne Değişim Metamodelinin HLA OMT tanımlamasından farkı, “ArrayDatatype” sınıfına “averageSize” adında yeni bir niteliğin eklenmiş olmasıdır. Bu yeni nitelik, ilerde uygun atama seçeneklerinin analizi aşamasında değiştirilen nesnelerin boyutu hesaplanırken kullanılacaktır. Şekil 5-2’te HLA OMT şemasındaki [IEEE 2010c] “arrayDataType” türünün bu çalışma kapsamında güncellenmiş sürümü gösterilmiştir.

```
...
<xs:complexType name="arrayDataType">
        <xs:sequence>
                <xs:element name="name" type="IdentifierType" />
                <xs:element name="dataType" type="ReferenceType">
                        <xs:annotation>
                                <xs:documentation>identifies the datatype of an
element of this array</xs:documentation>
                        </xs:annotation>
                </xs:element>
                <xs:element name="cardinality" type="cardinalityType">
                        <xs:annotation>
                                <xs:documentation>contains the number of
elements that are contained in the array</xs:documentation>
                        </xs:annotation>
                </xs:element>
                <xs:element name="averageSize" type="xs:integer" />
                <xs:element name="encoding"
type="arrayDatatypeEncodingType">
                        <xs:annotation>
                                <xs:documentation>describe, in detail, the encoding
of the array datatype (e.g., the sequence of elements and the order of elements
in multi-dimensional arrays) as delivered to and received from the RTI
                                </xs:documentation>
```

```
                    </xs:annotation>
                </xs:element>
                <xs:element name="semantics" type="String" minOccurs="0" />
                    <xs:any namespace="##other" minOccurs="0" />
        </xs:sequence>
        <xs:attributeGroup ref="commonAttributes" />
</xs:complexType>

...
```

Şekil 5-2 HLA OMT şemasının tez kapsamında genişletilmesi

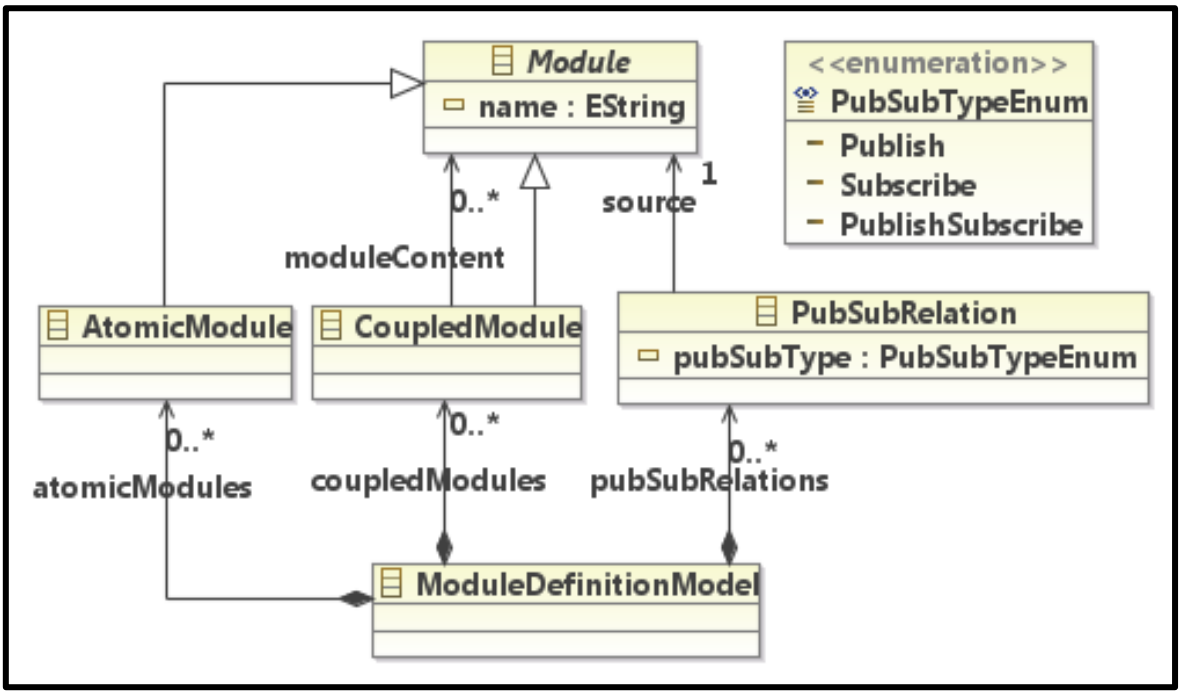
5.2 Benzetim Modüllerini Tasarla

Benzetim Modüllerini Tasarla adımında yapılan işlem, benzetim katılımcılarının alt benzetim modüllerinin tanımlanması olarak özetlenebilir. Benzetim Modülleri, sistemin belirli bir parçasını modellemekten sorumlu olan benzetim sistemi yapıtaşlarıdır. Çizelge 3-1'de verilen örnek senaryo ele alınırsa, benzetim modülleri Gemi, Tank, Uçak, Denizaltı, SH EO Füzeler, SS RF Füzeler, EH Modelleri, vs. olarak sıralanabilir.

Bu adımda, benzetim modüllerinin tanımlanmasının yanı sıra, benzetim modelleri arasındaki içerme *(composition)* ilişkilerinin de tanımlanması hedeflenmektedir. Örneğin bir Radar Modülü, bir verici *(transmitter)* modülü ve bir alıcı *(receiver)* modülü içerebilir. Bu çalışma kapsamında hem benzetim modüllerini ve hem de içerme ilişkilerini tanımlamaya olanak sağlayacak ortak bir metamodel (Şekil 5-3) tanımlanmıştır. *Discrete Event System Specification (DEVS)* [Zeigler 2003] modeline benzer olarak, metamodelde benzetim sistemlerinin yapıtaşları olacak atomik ve bağlı *(coupled)* modeller tanımlanmıştır.

Şekil 5-3'te gösterilen *ModuleDefinitionModel* sınıfı, metamodelin kök sınıfıdır ve bir modül tanımlama modelini temsil eder. Modül tanımlama modeli, kısaca modüller, modüller arası içerme ilişkileri ve yayımlama/üye olma ilişkilerini tanımlayan bir model olarak tanımlanabilir. Yayımlama/üye olma ilişkileri bir sonraki alt kesimde ele alınacaktır. Metamodelde gösterilen bir diğer sınıf olan *Module*, benzetim modül tanımlamalarının atası olan soyut sınıftır. *Module* sınıfı,

"*name*" adında modülün kimliğini tanımlayan bir nitelik içerir. Metamodeldeki *AtomicModule* sınıfı basit atomik benzetim modellerini temsil ederken *CoupledModule* sınıfı diğer atomik veya bağlı modülleri içerebilen karmaşık benzetim modellerini temsil eder. Bu içerme ilişkisi metamodelde *moduleContent* referansı ile gösterilmiştir.



Şekil 5-3 Modül ve Yayımla/Üye Ol İlişkilerini Tanımlama Metamodeli

5.3 Benzetim Modüllerinin Yayımlama/Üye Olma İlişkilerini Tasarla

Bu adımda, benzetim modüllerinin 5.1 adımında tasarlanan Benzetim Veri Değişim Modeli (BVDM) bileşenleriyle olan yayımlama/üye olma ilişkileri tanımlanır. Örneğin, *Radar* modülü *RadarBeam* nesne sınıfını yayımlarken *RFPropagation* modülü bu nesne sınıfına üye olabilir. Benzer şekilde, *Radar* modülü *SystemStatusChange* etkileşim sınıfına üye olabilir.

Benzetim Modüllerinin Yayımlama/Üye Olma İlişkilerini tanımlama adımının metamodeli, Benzetim Modüllerini Tasarla adımıyla ortak olarak Şekil 5-3'te tanımlanmıştır. Şekil 5-3'te gösterilen *PubSubRelation* sınıfı, bir benzetim modülü "*Module*" ile BVDM bileşeni "*ObjectModelElement*" arasında yayımlama/üye olma ilişkisi tanımlar. BVDM elemanlarının ata sınıfı olan *ObjectModelElement* sınıfı Benzetim Veri Değişim Metamodelinde tanımlanmış kök model sınıfıdır.

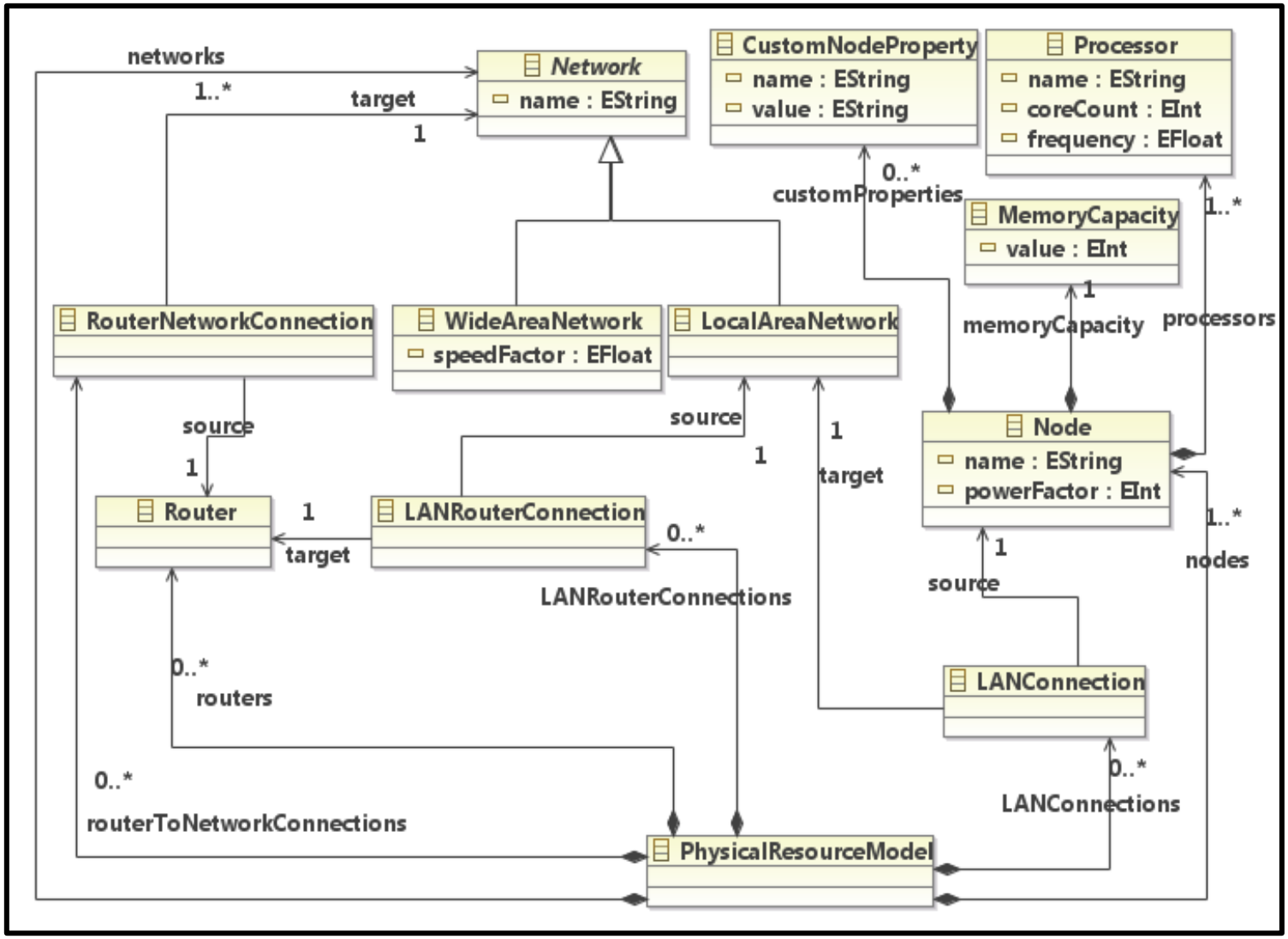
PubSubRelation sınıfı, ilişkinin türünü tanımlayan *pubSubType* niteliğine sahiptir. Bu niteliğin alabileceği değerler Şekil 5-3'te gösterilen *PubSubTypeEnum* sınıfı tarafından *Publish*, *Subscribe* ve *PublishSubscribe* olarak tanımlanmıştır.

5.4 Fiziksel Kaynakları Tasarla

Fiziksel Kaynakları Tasarla aktivitesinde, mevcut düğümlerin işlem gücü, bellek kapasitesi ve düğümler arasındaki ağ bağlantıları tanımlanır. Yazılımsal iş parçalarının tanımlanması ile fiziksel kaynakların tasarımı birbirinden bağımsızdır ve farklı ekipler tarafından yapılabilir. Sürecin etkinliğini arttırmak için Şekil 5-1'de verilen akışta da gösterildiği gibi, fiziksel kaynakları tanımlama aktivitesi yukarıda anlatılan üç yazılım tasarımı aktivitesine (Kesim 5.1, 5.2 ve 5.3) paralel olarak gerçekleştirilebilecek şekilde tanımlanmıştır.

Örneğin, fiziksel kaynak tasarımında benzetim katılımcılarının yerleştirilip çalıştırılacağı 25 düğüm tanımlanabilir. Tasarım detaylandırılarak, her düğümün 12280 MB bellek kapasitesine ve 2.3 MHz'lik 4 çekirdekli 2 işlemciye sahip olduğu tanımlanabilir. Düğümlerin tümü ortak bir yerel ağa bağlanarak düğümler arasında homojen bir bağlantı ağı tanımlanabilir. Bu örnekte düğümlerin kapasite ve güç özellikleri aynı olarak tanımlanmış olsa da, geliştirilen yaklaşımda düğümler değişik bellek kapasitesi ve işlem güçlerine sahip olabilir. Aynı şekilde geliştirilen yaklaşım bu örnekte tanımlananın aksine, aralarında farklı iletişim performansları olan yerel alan ve geniş alan ağları tanımlayıp düğümleri ağ üzerinde farklı noktalara dağıtarak heterojen bir bağlantı modeli de tanımlamayı mümkün kılar.

Şekil 5-4'de gösterilen Fiziksel Kaynak Metamodeli, mevcut fiziksel kaynak bileşenlerini tanımlamak için kullanılabilir.



Şekil 5-4 Fiziksel Kaynak Tanımlama Metamodeli

PhysicalResourceModel, bir fiziksel kaynak modelini temsil eden kök sınıftır. Bir fiziksel kaynak modelinde *Node* sınıfının olgularıyla temsil edilen bir veya daha çok düğüm olabilir. *Node* sınıfının *name* niteliği düğümün kimlik bilgisini içerir. *Node* sınıfının *powerFactor* niteliği ise, düğümün diğer düğümlerle göreceli olarak işlem gücünü tanımlar. Bir düğümün bir veya daha çok işlemci birimi olabilir. *Processor* sınıfı, *name, frequency* ve *coreCount* nitelikleriyle bir işlemci biriminin özelliklerini tanımlar. Bu sınıfın niteliklerinin açıklaması aşağıdaki gibidir:

- "*name*" niteliği, işlemci biriminin sembolik adıdır (Örneğin "Intel Core I7" gibi).
- "*coreCount*" niteliği, işlemci biriminin sahip olduğu çekirdek sayısını tanımlar. Günümüzde işlemciler birden çok çekirdeğe (1, 2, 4, 8, 10 gibi) sahip olabildiğinden dolayı bu nitelik gereklidir.
- "*frequency*" niteliği, işlemcinin frekansını MHz cinsinden tanımlar.

Bu noktada akla gelebilecek bir soru düğümler için neden hem *powerFactor* niteliği ile işlem gücü hem de *Processor* sınıfı ile açık olarak işlemci bilgisi tanımlandığıdır. Aslında bu metamodel tasarım kararı kullanıcıya esneklik sağlamak amacıyla alınmıştır. Kullanıcı isterse bir düğüme *powerFactor* niteliği ile doğrudan işlem gücü değeri tanımlayabileceği gibi, düğümün işlemcilerini detaylı olarak tanımlayıp uygun yerleştirme bulma aşamasında işlemci özelliklerine göre her düğümün işlem gücünü programatik olarak da hesaplama yolunu seçebilir.

Düğümlerin farklı bellek kapasiteleri olabilir. *MemoryCapacity* sınıfı *value* niteliği ile bir düğümün sahip olduğu bellek miktarını MegaByte cinsinden tanımlar.

Fiziksel kaynakları tanımlama metamodelinde her bir düğüm için açık olarak tanımlanan işlemci gücü, bellek kapasitesi gibi niteliklerin yanı sıra, kullanıcılara düğümlere özel nitelikler de tanımlayabilme yeteneği sağlamak önemli bir esneklik olacaktır. Örneğin kullanıcı işlemci gücü ve bellek kapasitesine ek olarak her düğüm için disk kapasitesi tanımlamak isteyebilir. *CustomNodeProperty* sınıfı, her düğüm için buna benzer ilave özellikler tanımlamayı mümkün kılar. İlave özellikler, metamodelde *CustomNodeProperty* sınıfının *name* ve *value* nitelikleriyle

gösterilen isim-değer ikilileri olarak tanımlanabilir. Örneğin bir düğümün disk kapasitesi "*diskCapacity"* adı ve "340 GB" değeri ikilisi ile tanımlanabilir.

Bir fiziksel kaynak modelinde, bir veya daha çok ağ tanımı olabilir. *Network* sınıfı, yerel alan ağlarını temsil eden *LocalAreaNetwork* (LAN) ve *WideAreaNetwork* (WAN) sınıflarının soyut ata sınıfıdır. *Network* sınıfının *name* niteliği, ağın sembolik adıdır (Örneğin: "ULAKBİM" gibi).

Yerel Alan Ağları ile Geniş Alan Ağlarının kapasiteleri doğal olarak aynı değildir. Bu durum, ağ üzerinde farklı noktalara yerleştirilmiş düğümler arasında iletişim başarımını etkileyecektir. Model seviyesinde ağlar arasındaki başarım farkını modellemek için geniş alan ağlarını temsil eden *WideAreaNetwork* sınıfına *speedFactor* adında ağın standart bir yerel alan ağına kıyasla hız çarpanını gösteren bir nitelik eklenmiştir. Örneğin bir geniş alan ağının hız faktörünün "0.25" olması, standart bir yerel alan ağından ortalama 4 kat daha yavaş olduğu anlamına gelmektedir.

LANConnection sınıfı, bir düğümün bir yerel alan ağına bağlantısını tanımlar. *Router* sınıfı, ağları birbirlerine bağlamak için kullanılan yönlendiricileri tanımlar. *Router* sınıfının "*name*" niteliği, yönlendiricinin sembolik adıdır. *LANRouterConnection* sınıfı bir yerel alan ağının yönlendiriciye bağlantısını gösterirken, *RouterNetworkConnection* sınıfı bir yönlendiricinin bir ağa bağlantısını temsil eder.

5.5 Benzetim Çalıştırma Konfigürasyonunu Tasarla

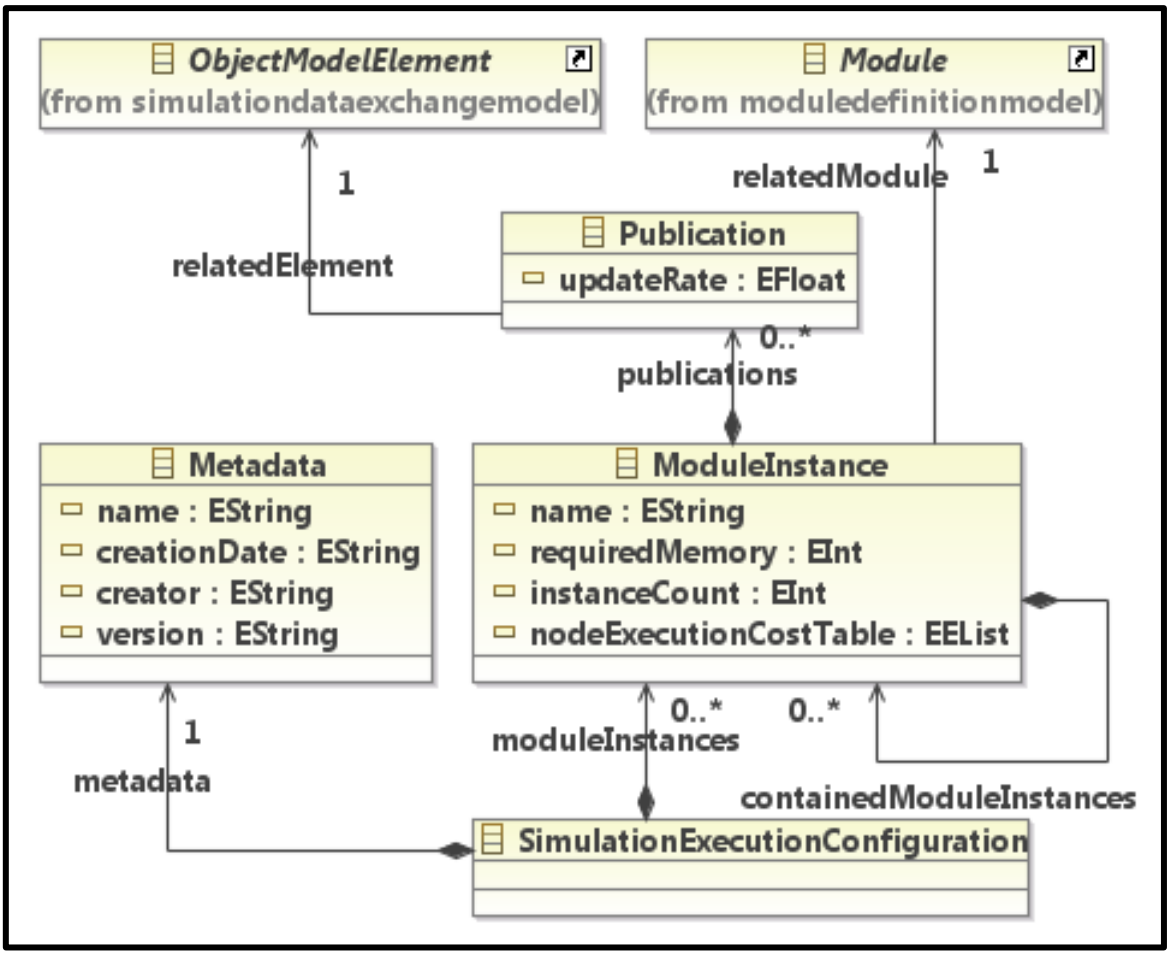
Sürecin bu aşamasına kadar benzetim sisteminin bileşenleri, veri değişim modeli, benzetim bileşenlerinin yayımlama/üye olma ilişkileri ve fiziksel kaynakların tasarlanması için gerekli tanımlamalar yapılmıştır. Bu aşamaya kadar yapılan tanımlamalar ile benzetim sisteminin yapısal *(structural)* özellikleri tasarlanabilir. Bir benzetim veya daha genel anlamda bir sistem için uygun bir yerleştirme bulunabilmesi için bu yapısal tanımlamaların yanı sıra, sistemde hangi bileşenden kaç adet bulunduğu, bileşenlerin veri modeli elemanlarını ne sıklıkta güncellediği *(update rate)* gibi koşum zamanına dair bilgilere de ihtiyaç vardır. Bu bağlamda, geliştirilen yaklaşımın bir parçası olarak benzetim çalıştırma konfigürasyonu tasarlama aktivitesi tanımlanmıştır.

Benzetim Çalıştırma Konfigürasyonu Tasarlama aktivitesinde, önceki adımlarda tanımlanan benzetim sistemi bileşenlerinin koşum zamanı özellikleri tanımlanır. Benzetim Çalıştırma Konfigürasyonu Tasarlama aktivitesinde, benzetim modül olgusu sayısı, her modül olgusunun her yayımlaması için güncelleme sıklığı, her modül olgusunun her bir hedef düğüm üzerindeki işletim maliyeti tanımlanır.

Bu tez kapsamında tanımlanan örnek durum çalışması ele alınırsa, uygun yerleştirme konfigürasyonu bulunurken kullanılacak çalıştırma konfigürasyonunda aşağıdaki gibi tanımlamalar olabilir:

- 300 Radar modülü olgusu olacaktır.
- Her bir radar olgusu saniyede 5 kere *RadarBeam* nesnesini güncelleyecektir.
- Her bir radar modülü olgusunun çalıştırma maliyeti ölçeklendirilmiş olarak bir düğüm için 10 üzerinden 6, başka bir düğüm için 10 üzerinden 4, vs. şeklinde tanımlanmıştır.
- 200 Uçak Modülü olgusu olacaktır.
- Her bir uçak olgusu saniyede 10 kere *AirPlatform* nesnesini güncelleyecektir.
- Her bir uçak modülü olgusunun çalıştırma maliyeti ölçeklendirilmiş olarak bir düğüm için 10 üzerinden 7, başka bir düğüm için 10 üzerinden 6, vs. şeklinde tanımlanmıştır.

Benzetim çalıştırma konfigürasyonlarını modellemek için gerekli elemanları içeren metamodel Şekil 5-5'da verilmiştir.



Şekil 5-5 Benzetim Çalıştırma Konfigürasyonu Metamodeli

SimulationExecutionConfiguration sınıfı, metamodelin kök sınıfıdır ve bir benzetim çalıştırma konfigürasyonunu tanımlar. Bir benzetim çalıştırma konfigürasyonu, bir *Metadata* ve bir grup *ModuleInstance* sınıfı olgusundan oluşur. *Metadata* sınıfı, isim *(name),* sürüm *(version),* oluşturan bilgisi *(creator)* ve oluşturulma tarihi *(creationDate)* gibi benzetim çalıştırma konfigürasyonunu tanımlayıcı nitelikleri içerir. *ModuleInstance* sınıfı, Benzetim Modülleri Tasarlama adımında tanımlanan benzetim modüllerinin olgularını temsil etmek için tanımlanmıştır. Bu sınıfın *name* niteliği modül olgusunun sembolik adını gösterir. Örneğin bir *"AirplaneModel"* benzetim modülü olgusunun adı "F-16 Blok 50" olabilir. Her benzetim modül olgusunun farklı düğümler üzerindeki çalıştırma maliyeti farklı olabilir. Bundan dolayı, *ModuleInstance* sınıfı *nodeExecutionCostTable* adında tablo türünde bir nitelik içerir. Bu nitelik ilgili benzetim modül olgusunun fiziksel kaynak modelinde tanımlı her düğüm için çalıştırma maliyeti bilgisini tutar. Çalıştırma maliyeti bilgisinin benzetim modülleri tasarlanırken bir defaya mahsus olarak tanımlanması

yerine çalıştırma konfigürasyonu tanımlama aşamasına bırakılmasının nedeni, çalıştırma maliyetinin seçilen çalıştırma konfigürasyonuna bağımlı olmasıdır. Örneğin, bir radar modülü olgusunun çalıştırma maliyeti, ilgili çalıştırma konfigürasyonundaki sinyal yansıma kaynaklarının (ör: hava platformları) sayısına göre değişir. Bu noktada akla gelebilecek bir soru, bir benzetim modülü olgusunun çalıştırma maliyetinin nasıl belirleneceğidir. Öncelikle, bir modül olgusunun çalıştırma maliyeti net olarak ancak modül geliştirildikten ve ilgili hedef düğümlerin üzerinde çalıştırıldıktan sonra ölçülebilir [Lauterbach et al. 2008]. Bu çalışma kapsamında geliştirilen yaklaşımın adımları temel olarak tasarım aşamasında işletileceği için, benzetim modülü çalıştırma maliyetleri de bu aşamada belirlenmelidir. Genel anlamda tasarım aşamasında yazılım bileşenlerinin çalıştırma maliyetlerini hesaplamak için karmaşıklık hesabı yöntemleri (Podgorelec ve Hericko [2007]) veya prototipleme gibi çeşitli yöntemler kullanılır. Benzetim modüllerinin çalıştırma maliyetlerini hesaplamak için de bu gibi genel yazılım mühendisliği yaklaşımları kullanılabilir. Bir benzetim modülü olgusunun çalıştırma maliyeti, benzetim çalıştırma konfigürasyonundaki diğer benzetim modülü olguları ve hedef düğümlerin işlem gücüne (5.4 kesiminde anlatılan düğümün *powerFactor* niteliği ile tanımlanır.) göre ölçeklendirilmiş olarak verilmelidir. Örneğin bir radar modülü olgusunun toplam işlem gücü 100 birim olan bir düğüm üzerindeki çalıştırma maliyeti 20 birimken, göreceli olarak daha basit olan bir Elektro Optik Füze Algılayıcısının çalıştırma maliyeti 5 birim olarak tanımlanmış olabilir.

ModuleInstance sınıfının bir diğer niteliği olan *requiredMemoryAmount*, benzetim modül olgusunun tahmin edilen bellek gereksinimini tanımlar. Bir benzetim modül olgusunun bellek gereksinimi, aynen çalıştırma maliyeti değeri gibi tasarım aşamasında genel yazılım mühendisliği yöntemleri uygulanarak tasarım aşamasında tahmin edilebilir. Herhangi bir yazılım bileşeni için geçerli olduğu gibi, bir benzetim modül olgusunun da bellek gereksinimi düğümden düğüme değişmeyeceğinden çalıştırma maliyeti değerinden farklı olarak, bellek gereksinimi değeri her düğüm için ayrı ayrı belirlenmek zorunda değildir ve tek bir değer belirlenmesi yeterlidir.

Bir benzetim çalıştırma konfigürasyonunda aynı benzetim modülünün birden çok olgusu olabilir. Örneğin örnek durum çalışması için tanımlanan bir benzetim

çalıştırma konfigürasyonunda onlarca radar modülü olgusu, yüzlerce uçak modülü olgusu ve binlerce IR güdümlü füze olgusu bulunabilir. Bu durumda kullanıcının her bir benzetim modülü olgusunu çalıştırma konfigürasyonuna teker teker eklemesi etkin bir yöntem değildir. Geliştirilen yaklaşım ve araçların kullanılabilir olması için bu gibi problemlerin yaklaşım geliştirilirken adreslenmesi şarttır. Bu bağlamda *ModuleInstance* sınıfına çalıştırma konfigürasyonunda ilgili modül olgusundan kaç adet bulunduğu bilgisini tutan *instanceCount* niteliği eklenmiştir.

ModuleInstance sınıfının *containedModuleInstances* içerme türü ilişkisi, ilgili modül olgusunun içerdiği diğer modül olgularını gösterir. Örneğin bu ilişki kullanılarak örnek durum çalışmasındaki uçak veya gemi gibi platformların üzerindeki radar, karıştırma cihazları, ikaz sistemleri, taşınan silahlar gibi alt sistemler gösterilebilir. Bu ilişki öz yineli olarak alt modül olguları için de geçerli olabilir. Örneğin bir platform olgusu üzerindeki bir füze olgusu füze algılayıcısı ve GPS gibi alt modül olgularını içerebilir.

ModuleInstance sınıfının bir diğer ilişkisi olan *relatedModule* ise, modül olgusunu 5.2 – Benzetim Modüllerini Tasarlama adımında tanımlanan benzetim modüllerinden birisiyle eşleştirir. Bu eşleştirme sayesinde ilerde uygun yerleştirme alternatifleri türetilirken ilgili modül olgusunun üye olduğu veri modeli elemanları tespit edilerek modül olgusuna birim zamanda dışarıdan gelecek veri miktarı hesaplanacaktır.

Bir *ModuleInstance* sınıfı, modül olgusunun veri yayımlama karakteristiğini gösteren bir veya daha çok *Publication* sınıfı olgusu içerebilir. *Publication* sınıfı, yayımlanan veri modeli elemanı ve güncelleme sıklığı niteliklerine sahiptir. *Publication* sınıfının yayımlanan veri modeli elemanı niteliği *relatedElement* ilişkisi ile gösterilir ve *Publication* sınıfını 5.1 adımında tanımlanan Benzetim Veri Değişim Modelindeki (BVDM) nesne veya etkileşim sınıflarından birisiyle ilişkilendirir. Bu sınıfın bir diğer niteliği olan *updateRate* ise, modül olgusunun ilgili BVDM elemanını saniyede kaç defa güncelleyeceği bilgisini tutar.

Aslında benzetim modüllerin yayımlama/üye olma ilişkileri 5.3 – Benzetim Modüllerinin Yayımlama/Üye Olma İlişkilerini tanımlama adımında her *Module* sınıfı için tanımlanmış ve her *ModuleInstance* sınıfı *relatedModule* ilişkisi ile bir

Module sınıfına bağlanmıştır. Buna rağmen benzetim çalıştırma konfigürasyonu tanımlama aşamasında tekrar yayımlama ilişkilerini gösteren *Publication* sınıfının tanımlanmasının sebebi, benzetim modülleri arasındaki iletişim maliyetini hesaplamak için yayımlanan/üye olunan elemanların yanı sıra, yayımlama sıklığı *(update rate)* bilgisine ihtiyaç duyulmasıdır. Bu noktada akla gelebilecek bir diğer soru, yayımlama sıklığı parametresini bir defaya mahsus olarak 5.3– Benzetim Modüllerinin Yayımlama/Üye Olma İlişkilerini tanımlama adımında belirlenmek yerine her çalıştırma konfigürasyonu için neden tekrar tanımlandığıdır. Bu çalışma kapsamında geliştirilen yaklaşım tasarlanırken, kullanıcıya mümkün olan en esnek ve değişik analizler yapabileceği ortamı sağlamak ana hedeflerden birisi olmuştur. Bu kapsamda yayımlama sıklığı değerinin tanımlanması, sistemin yapısal *(structural)* özelliklerinin tanımlandığı 5.1, 5.2 veya 5.3 adımları yerine benzetim çalıştırma konfigürasyonu tanımlama adımına bırakılarak kullanıcının aynı yapısal tasarım üzerinde farklı güncelleme sıklıklarının etkilerini analiz etmesi mümkün kılınmıştır. Örneğin çalıştırma konfigürasyonundaki gemi modülü olgularının pozisyon bilgilerini saniyede 10 defa güncellemeleri ile 20 defa güncellemeleri durumunda uygun yerleştirmenin ne şekilde değiştiği, hatta mevcut fiziksel kaynaklara göre halen uygun bir yerleştirmenin bulunup bulunamadığı bu esneklik sayesinde kolayca analiz edilebilir.

5.6 Yerleştirme Algoritması İçin Girdi Parametrelerini Üret

Yukarıda detaylı açıklamaları verilen adımlarla, benzetim katılımcılarının yapısal özellikleri (5.1, 5.2 ve 5.3 Kesimleri), fiziksel kaynak ortamı (5.4 Kesimi) ve çalıştırma konfigürasyonları (5.5 Kesimi) tanımlanarak benzetim ortamı tasarlandıktan sonra, uygun yerleştirme alternatiflerinin aranması süreci başlatılabilir.

5.6.1 Kapasite kısıtlı iş atama problemi

Çalışmanın bu aşamasına gelindiğinde, öncelikle benzetim ortamı tasarımından uygun yerleştirmelerin türetilmesi probleminin alternatif çözüm yöntemleri için literatür taraması gerçekleştirilmiştir. Yapılan literatür taramasında, aslında çözülmeye çalışılan yerleştirme probleminin pek çok çalışmaya konu olan İş Atama Problemi (*Task Allocation Problem – TAP*) [Stone 1977] alanına girdiği

anlaşılmıştır. İş Atama Problemi kısaca “mevcut kaynaklar ve işlerin tanımlarını girdi olarak alıp işlerin kaynaklara uygun biçimde atanması” olarak açıklanabilir.

Bu çalışma kapsamında İş Atama Problemi eniyileme (*optimization*) yöntemlerinin uyarlanarak kullanılması bağlamında öncelikle [Stone 1977], [Lo 1988], [Ucar et al. 2005] gibi çeşitli kaynaklar incelenmiştir. İncelenen kaynaklarda, İş Atama Problemi çözüm yaklaşımlarının çalıştırma maliyeti *(execution cost),* iletişim maliyeti *(communication cost)*, bellek gereksinimi *(memory requirement),* ve disk Yazma/Okuma (I/O) maliyetleri gibi değişik parametreleri girdi olarak aldığı gözlemlenmiştir.

İncelenen kaynaklara dayanarak, İş Atama Problemi çözme algoritmalarının amacı şu şekilde ifade edilebilir:
“İşleri, işlemcilere çalıştırma maliyetleri ve iletişim maliyetlerinin toplamı asgari olacak şekilde ata. Eğer iki iş aynı işlemciye atanmışsa, aralarındaki iletişim maliyetinin sıfır olduğunu varsay.”
Bu tez çalışması kapsamında, uygun yerleştirme alternatifleri bulunurken iletişim maliyeti ve çalıştırma maliyeti parametrelerinin yanı sıra, işlemcilerin işlem gücü ve bellek miktarı kısıtlarını dikkate alınması gerekmektedir, fakat klasik İş Atama Problemi çözüm yöntemleri [Stone 1977; Lo 1988; Ucar et al. 2005] kaynak kısıtlarını dikkate almamaktadır. Bu bağlamda yapılan literatür taraması derinleştirilmiş ve bu tez çalışması kapsamında çözülmeye çalışılan yerleştirme probleminin aslında İş Atama Problemi’nin bir özelleştirilmiş bir hali olan Kapasite Kısıtlı İş Atama Problemi *(Capacitated Task Allocation Problem – CTAP)* [Pirim 2006; Mehrabi et al. 2009] kategorisine girdiği anlaşılmıştır. Klasik iş atama problemi çözüm yöntemlerinde, kaynaklarla ilgili herhangi bir kısıt yokken, kapasite kısıtlı iş atama problemi çözümlerinde kaynakların bellek miktarı, toplam işlem gücü gibi çeşitli kapasite sınırları vardır. Örneğin Mehrabi et al. [2009] çalışmasında iş ataması yapılacak kaynakların bellek kısıtlı olduğunu varsaymıştır.

Bu kesimde, TAP ve CTAP tanımlarına uygun olarak bu çalışma kapsamında kullanılan eniyileme problemi tanımlanacaktır. Problem tanımında, m adet iş vardır ve i işi m_i birim belleğe ihtiyaç duyar. Ortamda n adet birbirine denk olmayan işlemci vardır, p işlemcisinin toplam M_p birim belleği ve C_p birim işlem kapasitesi vardır. i işini p işlemcisi üzerinde çalıştırmanın maliyeti x_{ip}’dir. c_{ij} i ve j işleri

arasındaki toplam iletişim maliyetini gösterir. İletişim maliyetleri hesaplanırken iletilen veri boyutuna ek olarak işler arası iletişimin sıklığı dikkate alınmalıdır. *i* ve *j* işleri arasındaki yüksek iletişim sıklığı daha yüksek iletişim maliyeti c_{ij}'ye sebep olacaktır. Eniyileme problemindeki amaç, işleri işlemcilere bellek ve işlem gücü kısıtlarını aşmadan asgari iletişim ve işletim maliyetiyle yerleştirmektir. Problemin karar değişkeni:

a_{ip} = 1, eğer *i* işi *p* işlemcisine atanmışsa, diğer durumda 0.

Problem, Ucar et al. [2005]; Pirim [2006] ve Mehrabi et al. [2009] gibi çalışmalardakine benzer olarak bir 0-1 programı (**M**) olarak tanımlanabilir. Sıralanan çalışmalara benzer olarak, bu tez çalışmasına konu olan problemin de amacı, tüm işlerin bitme süresini ve iletişim maliyetini enküçüklemektir *(minimization)*.

Yukarıda verilen tanımlamalara dayanarak, amaç matematiksel olarak ikili karar değişkenleri olan bir eniyileme problemi *(optimization problem with binary decision variables)* olarak aşağıdaki gibi ifade edilebilir.

(**M**) Enküçükle $\sum_{i=1}^{m}\sum_{p=1}^{n} a_{ip}x_{ip} + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{p=1}^{n} a_{ip}(1-a_{jp})c_{ij}$

Şu kısıtlar dahilinde:

$$\sum_{p=1}^{n} a_{ip} = 1, \qquad \forall i = 1,.....,m$$

$$\sum_{i=1}^{m} m_i a_{ip} \le M_p, \quad \forall p = 1,.....,n$$

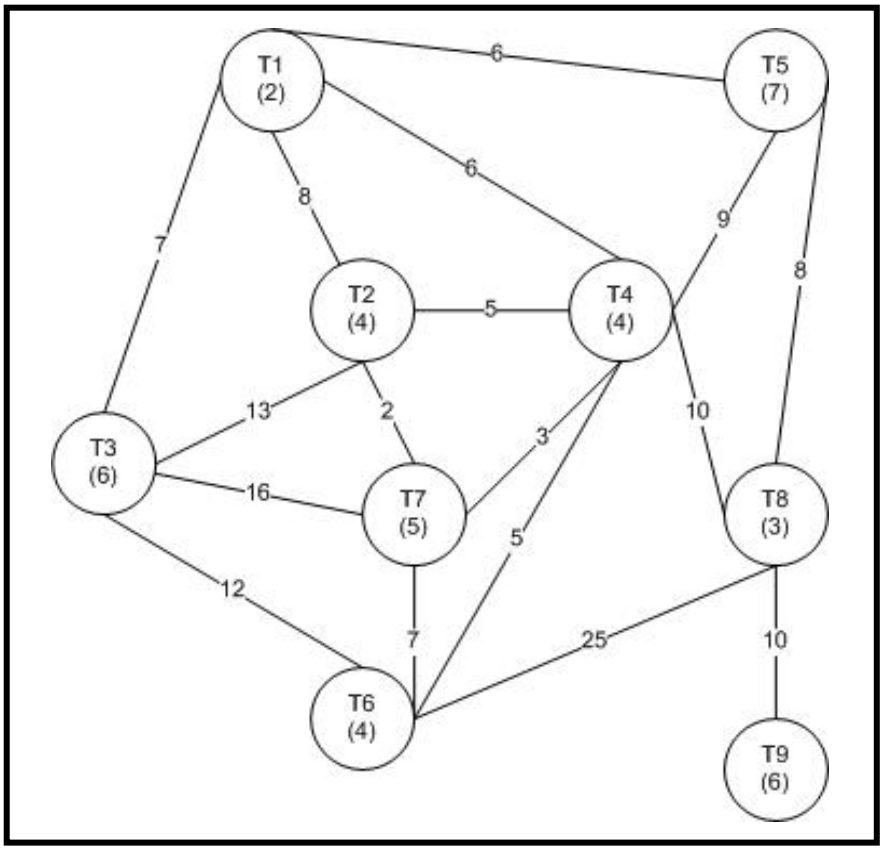
$$\sum_{i=1}^{m} x_{ip} a_{ip} \le C_p, \quad \forall p = 1,.....,n$$

$$a_{ip} = \{0,1\}, \qquad \forall i = 1,.....,m, \ \forall p = 1,.....,n$$

M matematiksel modelinin amacı şu şekilde açıklanabilir:
"İşleri, işlemcilere çalıştırma maliyetleri ve iletişim maliyetlerinin toplamı asgari olacak şekilde ata. Eğer iki iş aynı işlemciye atanmışsa, aralarındaki iletişim maliyetinin sıfır olduğunu varsay. Atamaları yaparken her işlemciye atanan işlerin toplam bellek ihtiyacının işlemcinin bellek miktarını aşmadığına ve toplam işlem gücü ihtiyacının da işlemcinin işlem gücünü aşmadığına emin ol."

5.6.1.1 Örnek bir kapasite kısıtlı iş atama problemi ve çözümü

Kapasite kısıtlı iş atama problemine örnek bir çalışma olarak, dokuz işin dört işlemciye atanması problemi tanımlanmış olsun. İşlerin özelliklerini ve aralarındaki iletişimleri gösteren çizge Şekil 5-6'de verilmiştir. Çizgedeki kenarların ağırlıkları işler arasındaki iletişim maliyetlerini gösterir. Örneğin T_2 ve T_3 arasındaki iletişim maliyeti 13 birimdir. Çizgedeki düğümlerin değerleri ilgili işin bellek gereksinimini gösterir. Örneğin T_7 işinin bellek gereksinimi 5 birimdir.



Şekil 5-6 Örnek Bir İş Atama Problemi Çizgesi

Şekil 5-6'de tanımlanan işlerin dört farklı işlemci üzerindeki çalıştırma maliyetleri Çizelge 5-1'de verilmiştir. Bu çizelgede gösterilen dört işlemcinin bellek kapasiteleri ise Çizelge 5-2'de tanımlanmıştır.

Çizelge 5-1 Örnek Dokuz İşin Dört İşlemci Üzerindeki Çalıştırma Maliyetleri

İş	**İşlemci Üzerindeki Çalıştırma Maliyeti**			
	P_1	**P_2**	**P_3**	**P_4**
T_1	8	7	8	9
T_2	13	12	14	13
T_3	9	7	8	9
T_4	12	10	11	13
T_5	16	14	13	15
T_6	17	15	16	14
T_7	14	12	11	10
T_8	3	2	4	3
T_9	6	7	6	7

Çizelge 5-2 Dört İşlemcinin Bellek Miktarları

İşlemci	**Bellek Miktarı**
P_1	20
P_2	12
P_3	12
P_4	10

Tanımlanan örnek problem literatürde tanımlı olan Kapasite Kısıtlı İş Atama Problemi çözüm yöntemleri ile çözülebilir. Örneğin [Mehrabi et al. 2009]'da tanımlanan genetik algoritma kullanılarak örnek problemi çözdüğümüzde Çizelge 5-3'te gösterilen sonuç ortaya çıkmıştır.

Çizelge 5-3 Örnek Bir İş – İşlemci Ataması

İşlemci	Atanan İşler
P_1	{ T_2, T_3, T_7 }
P_2	{ T_4, T_9 }
P_3	{ T_6, T_8 }
P_4	{ T_1, T_5 }

5.6.2 Kapasite kısıtlı iş atama problemi çözüm yöntemlerinin uygun yerleştirme bulma yaklaşımına bütünleştirilmesi

Kapasite kısıtlı iş atama probleminin matematiksel olarak ne şekilde çözülebileceği netleştirildikten sonra, mevcut çözüm algoritmalarının girdi ve çıktılarının bu tez kapsamında geliştirilen yaklaşıma bütünleştirilmesi ele alınmalıdır.

"Yerleştirme Algoritması için Girdi Parametrelerini Üret" adımında, Kapasite Kısıtlı İş Atama Problemi çözüm algoritmalarının girdileri olan parametreler önceki adımlarda oluşturulan benzetim ortamı tasarımından türetilir. Çizelge 5-4'te algoritmanın her bir parametresinin benzetim ortamı tasarımından nasıl türetildiği anlatılmıştır.

Çizelge 5-4 Kapasite Kısıtlı İş Atama Problemi Parametrelerinin Benzetim Ortamı Tasarımından Türetilmesi

Algoritma Parametresi	**Benzetim Ortamı Tasarımından Türetilmesi**
m adet iş	5.5 Benzetim Çalıştırma Konfigürasyonu Tasarlama adımında tanımlanan modül olgularından her biri bir işe karşılık gelir.
n adet işlemci	5.4 Fiziksel Kaynakları Tasarlama adımında tanımlanan düğümlerden *(node)* her biri bir işlemciye karşılık gelir.
M_p	*p* işlemcisinin bellek kapasitesi olan M_p, 5.4 Fiziksel Kaynakları Tasarlama adımında tanımlanan düğümlerin *(node) memoryCapacity* niteliğinden türetilir.
C_p	*p* işlemcisinin işlem gücü olan C_p, 5.4 Fiziksel Kaynakları Tasarlama adımında tanımlanan düğümlerin *(node) powerFactor* niteliğinden türetilir. Alternatif bir yöntem, düğümlerin işlemci *(processor)* tanımlarına göre işlem gücünün belirlenmesidir. 5.4 adımında verilen metamodel bu alternatifi destekleyecek şekilde esnek tanımlanmıştır.
m_i	*i* işinin ihtiyaç duyduğu bellek miktarı olan m_i, 5.5 Benzetim Çalıştırma Konfigürasyonu Tasarlama adımındaki *ModuleInstance* sınıfının gereken bellek miktarını gösteren *requiredMemory* niteliğinden türetilir.
x_{ip}	*i* işini *p* işlemcisi üzerinde çalıştırma maliyeti olan x_{ip}, 5.5 Benzetim Çalıştırma Konfigürasyonu Tasarlama adımındaki *ModuleInstance* sınıfının çalıştırma modül olgusu/düğüm üzerinde çalıştırma maliyeti ikililerini tutan *nodeExecutionCostTable* niteliğinden türetilir.

c_{ij}	*i ve j* işlerinin ayrı işlemcilere atanması durumunda sıfırdan farklı değer alan c_{ij} iletişim maliyeti, benzetim modülleri arasındaki yayımlama/üye olma ilişkilerine göre belirlenir. İletişim maliyetleri aşağıdaki tanımlamalar kullanılarak türetilir: - 5.5 Benzetim Çalıştırma Konfigürasyonu Tasarlama Adımında *Publication* sınıfı olgularıyla tanımlanan yayımlama özellikleri. - 5.3 Benzetim Modüllerinin Yayımlama/Üye Olma İlişkilerini Tasarlama adımında tanımlanan yayımlama ilişkileri. Yayımlama ilişkileri, *PubSubTypeEnum* türündeki niteliğinin değeri *Subscribe* veya *PublishSubscribe* olan *PubSubRelation* sınıf olguları sorgulanarak türetilir. - 5.1 Benzetim Veri Değişim Modelini Tasarlama adımında tanımlanan nesne modeli elemanları. Nesne modeli elemanları, yayımlama/üye olma ilişkileriyle tanımlanan veri değişimine konu olan yapıtaşlarıdır. Nesne modeli elemanlarının boyutları taşınan veri paketlerinin boyutunu belirler. Veri paketinin boyutu da, 5.5 Benzetim Çalıştırma Konfigürasyonu Tasarlama Adımında *Publication* sınıfı olgularında tanımlanan güncelleme sıklığı *(update rate)* niteliğiyle birlikte birim zamandaki iletişim maliyetlerinin hesaplanmasında kullanılır. İletişim maliyetlerinin hesaplanması 6. Kesimde anlatılan araç desteği kesiminde daha detaylı olarak anlatılacaktır.

5.7 Uygun Yerleştirme Seçeneklerinin Üretilmesi

Uygun Yerleştirme Seçeneklerinin Üretilmesi adımı, bir önceki adımın çıktıları olan parametreleri girdi olarak kullanarak bir Kapasite Kısıtlı İş Atama Problemi çözüm algoritması işletip uygun bir yerleştirme alternatifi türetmeye çalışır. Önceki kesimde de bahsedildiği gibi, literatürde Kapasite Kısıtlı İş Atama Problemini çözmek için değişik yaklaşım ve algoritmalar tanımlanmıştır [Lo 1988; Hamam et al. 2000; Chen and Lin 2000; Ucar et al. 2005; Pirim 2006; Mehrabi et al. 2009]. Literatürde tanımlı olan algoritmaların yanı sıra, eniyileme problemlerini çözmek için kullanılabilecek CPLEX [IBM 2010] gibi çeşitli Rafta Hazır Ticari (RAHAT) ürünler de mevcuttur. Örneğin 5.6 kesiminde tanımlanan **M** eniyileme problemi, kapasite kısıtlı ikinci dereceden bir problem *(quadratic problem with capacity constraints)* olarak ifade edilip CPLEX ile çözülebilir [Pirim 2006].

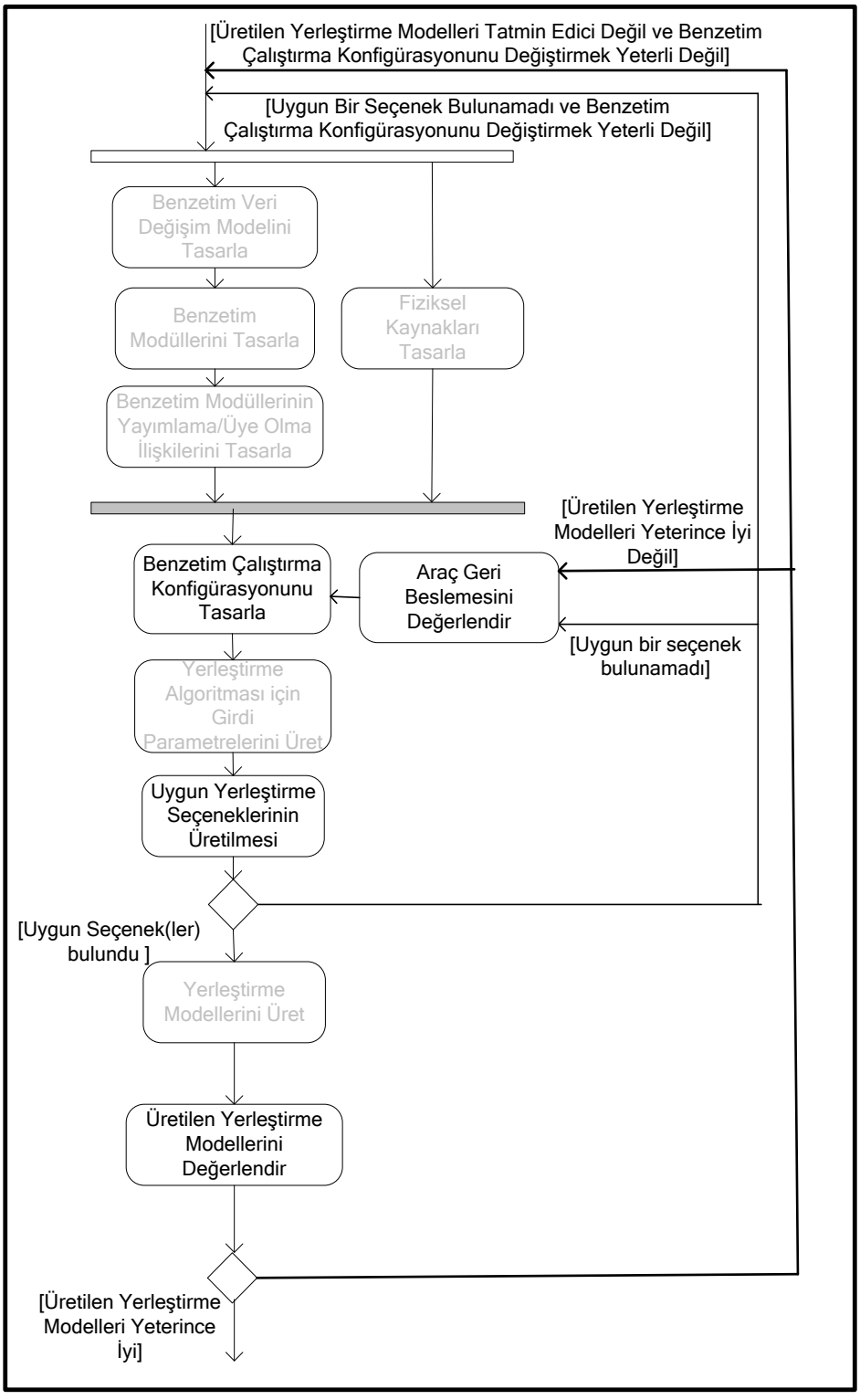
Ucar et al. [2005] İş Atama Probleminin çözülmesi için kullanılabilecek bir grup değişik algoritmanın karşılaştırmasını yapmıştır. Pirim [2006], değişik Kapasite Kısıtlı İş Atama Problemi çözüm algoritmalarını ve aynı zamanda CPLEX aracının başarımını ele almıştır.

Bu çalışma kapsamında, herhangi bir İş Atama Problemi çözüm yönteminin kullanılması zorlanmamaktadır. Tezin ilerleyen kesimlerinde daha detaylı bahsedileceği gibi, herhangi bir algoritmaya veya yönteme bağlı olmamak için İş Atama Problemi Çözüm kesimi uygulamadan OSGI tabanlı [McAffer et.al 2010; OSGI 2011; Equinox 2012] bir programlama seviyesi arayüz (*interface*) ile ayrılmıştır. Bu çalışma kapsamında örnek kullanım olarak Mehrabi et al. [2009] tarafından tanımlanan genetik algoritma tabanlı yöntem, ihtiyaç duyulan parametreleri ve kısıtları (işlemci gereksinimi, bellek gereksinimi ve iletişim maliyeti) kapsadığı ve de başarımı iyi olduğu için tercih edilmiştir.

Girdi parametreleri sağlanıp ilgili algoritma koşturulduktan sonra bir uygun bir yerleştirme alternatifi bulunursa, algoritma çıktı olarak Çizelge 5-3'te verilene benzer bir iş – işlemci atama tablosu üretecektir. Bu tablodan benzetim yerleştirme modelinin üretilmesi bir sonraki kesimde anlatılan Yerleştirme Modelini Üretme adımında detaylı olarak anlatılacaktır. Bu bölümün kalan kesiminde uygun bir yerleştirme algoritması bulunamaması durumunda izlenmesi gereken adımlar anlatılmıştır.

5.7.1 Uygun bir yerleştirme seçeneği bulunamaması durumunda uygulanabilecek genel yöntemler

Şekil 5-7'de, Şekil 5-1'de verilen ana akışta uygun yerleştirme bulunamaması durumunda izlenecek adımlar vurgulanmıştır.



Şekil 5-7 Uygun Bir Yerleştirme Bulunamaması Durumunda İzlenecek Akış

Şekil 5-7'de verilen akış şemasında da gösterildiği gibi, uygun yerleştirme bulma adımında bir yerleştirme alternatifi bulunamazsa araç desteği ile benzetim tasarımı analiz edilerek bir geri besleme raporu oluşturulacaktır. Araç desteği ile sağlanacak geri besleme aşağıdaki bilgileri içermektedir:

- Büyüklüğe göre sıralanmış olarak benzetim modül olguları arasındaki veri iletişim maliyetleri
- Boyutlarına göre sıralanmış olarak benzetim veri değişim modeli nesneleri
- Büyüklüğe göre sıralanmış olarak benzetim modül olgularının ihtiyaç duydukları bellek miktarları
- Kapasite kısıtlarına göre sıralanmış olarak fiziksel kaynak düğümleri.

Sağlanacak geri beslemeye göre öncelikle benzetim çalıştırma konfigürasyonu güncellenerek çözüm uygun bir yerleştirme seçeneği bulunmaya çalışılacaktır. Bu durumda, benzetim çalıştırma konfigürasyonu güncellenerek yerleştirme bulma algoritması yeniden işletilecektir. Uygun bir yerleştirme alternatifi bulunmadığı sürece benzetim çalıştırma konfigürasyonu güncellemeleri devam eder ve yerleştirme bulma algoritması tekrar işletilir. Benzetim çalıştırma konfigürasyonu yeterince iyileştirilmesine rağmen halen uygun bir yerleştirme alternatifi bulunamamışsa, Şekil 5-7'da gösterildiği gibi benzetim ortamı tasarımı sürecinin başlangıcına dönüp Benzetim Veri Değişim Modeli, Benzetim Modülleri, Yayımlama/Üye olma ilişkileri ve fiziksel kaynak tasarımı kesimleri iyileştirilmeye çalışılmalıdır.

Uygun bir yerleştirme seçeneği bulunamaması durumunda benzetim ortamı tasarımı iyileştirilirken temel amaç bellek, işlem gücü ve modüller arası iletişim gereksinimi gibi maliyet kalemlerini iyileştirmek olmalıdır. Koşut ve Dağıtılmış Benzetim sisteminin tasarımını iyileştirmek geliştirilen uygulamaya (bu tez çalışması kapsamında, örnek durum çalışması olarak seçilen Elektronik Harp benzetimi veya trafik benzetimi) sıkı sıkıya bağımlı olmasına rağmen, sağlanabilecek bir takım genel geçer eniyileme tavsiyeleri aşağıdaki gibi sıralanabilir:

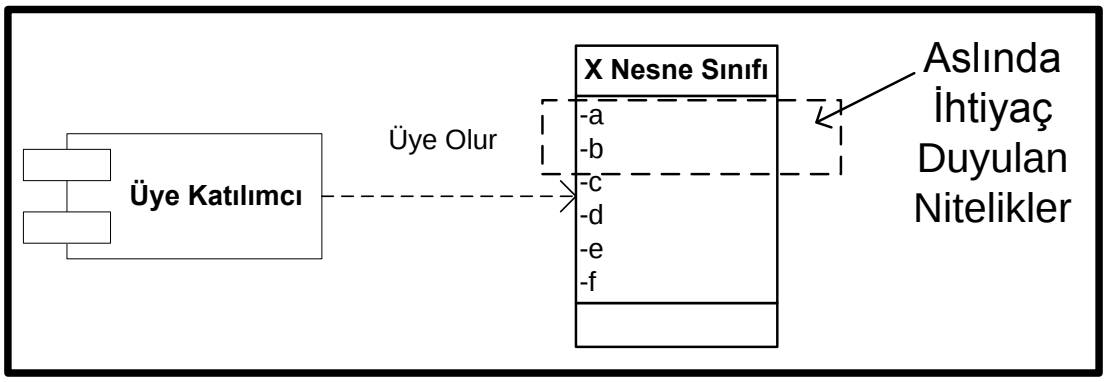
1. Benzetim çalıştırma konfigürasyonundaki güncelleme sıklıklarını düşür: Güncelleme sıklıkları, veri yayımlayıcıları ile üyeler arasındaki iletişim maliyetini etkiler. İletişim maliyetlerinin yanı sıra, daha sık veri güncelleme veriyi üreten kaynağın daha fazla işlem gücü ve bellek tüketmesine sebep olabilir. Bu bilgiler ışığında, benzetim ortamı tasarımını iyileştirmek için, benzetim çalıştırma konfigürasyonundaki bazı yayımlamaların güncelleme sıklığını düşürmek etkili olabilir. Örneğin bu tez kapsamında kullanılan örnek durum çalışmasında deniz platformları hava platformlarına nispeten daha yavaş hareket ettikleri için güncelleme sıklıkları düşürülebilir.

2. Üye olunan veri kümelerini ve veri modelini gözden geçir: Bir üyelik, üye olunan sınıfın olgularındaki güncellemelerin koşum zamanında üye katılımcıya iletilmesini gerektirir. Çoğu durumda üye olan katılımcı üye olduğu veri kümesinin aslında özel bir alt kümesine(örneğin konum ve oryantasyon bilgisi gibi) ihtiyaç duyar. Bu gibi durumları ele almak için benzetim ortamı tasarımı incelenerek öncelikle bu tip üyelikler tespit edilmeli ve daha net üyelik tanımları yapılması sağlanmalıdır. Örneğin bu tez kapsamında kullanılan örnek durum çalışmasında sinyal yayılım modeli platformların sadece konum ve oryantasyon bilgisine ihtiyaç duyduğu için veri modeli ağacında alt yapraklarda bulunan ve daha çok nitelik içeren hava platformu, deniz platformu, kara platformu gibi nesne sınıflarına üye olmak yerine bu sınıfların ata sınıfı olan ve pozisyon, oryantasyon gibi temel nitelikleri içeren Platform sınıfına üye olmalıdır.

   Benzer yayımlama/üye olma teknolojilerinden farklı olarak HLA mimarisi nitelik tabanlı üyeliğe (*attribute based subscription*) izin vermektedir. Bu tez kapsamında geliştirilen metamodellerde hedef sadece HLA teknolojisini destekleyen bir mimari yerine daha genel bir mimari geliştirmek hedeflendiğinden üye olma tanımlamalarının sınıf seviyesinde olmasına karar verilmiştir.

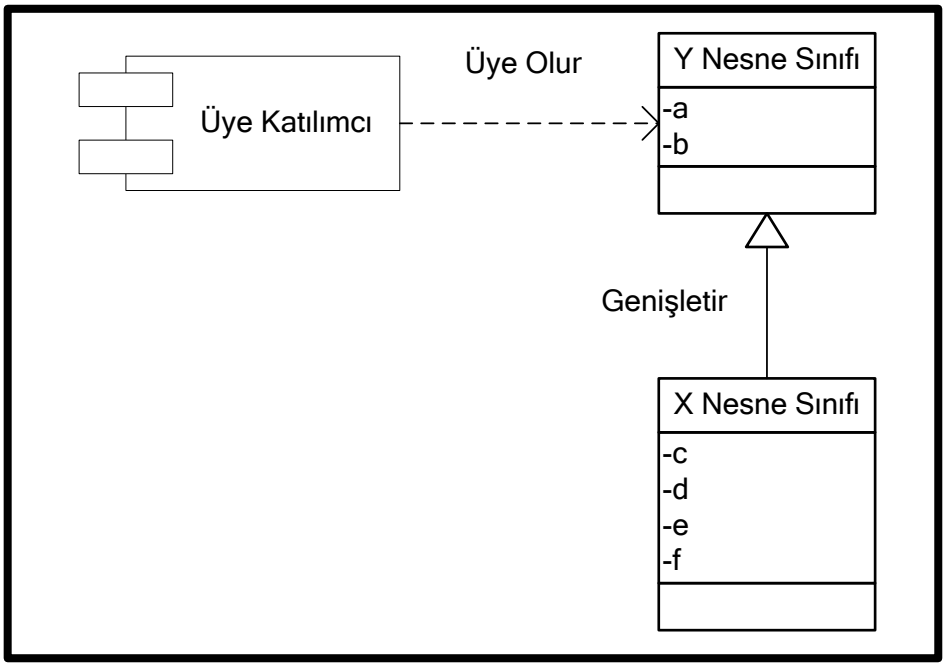
   Bazı durumlarda, ihtiyaç duyulan veri kümesi, aslında ilgili üyenin ihtiyaç duymadığı geniş bir veri kümesi ile aynı sınıf içerisinde tanımlanmış olabilir. Örneğin Şekil 5-8'da gösterilen *Üye Katılımcı*, aslında *X Nesne Sınıfının* sadece *a* ve *b* niteliklerine ihtiyaç duymasına rağmen sınıf seviyesi üyelik

zorunluluğundan dolayı X Nesne Sınıfına üye olur. Bu durumda, Üye Katılımcı, X Nesne Sınıfının olguları güncellendiğinde ilgilendiği *a* ve *b* niteliklerine ek olarak aslında ihtiyacı olmadığı halde *c, d, e* ve *f* niteliklerinin de güncellemelerini alır.



Şekil 5-8 Üye Olunan Veri Sınıfının İhtiyaç Duyulandan Fazla Veri İçermesi Durumu

Bu tez kapsamında adreslenen hedef mimari sadece HLA olsaydı, bu problem ilgili üyenin sadece ihtiyaç duyduğu veri kümesine üye olmasına yönelik metamodel seviyesinde destek sağlanarak problem çözülebilirdi, fakat sadece sınıf seviyesi üyeliği destekleyen mimarilerde bu durum mümkün değildir. Bu sorunu çözmek için izlenebilecek yöntemlerden birisi, üye olunan sınıfın bölünerek hiyerarşik bir yapı kurulmasıdır. Örneğin Şekil 5-8'da gösterilen problemli durum tasarım Şekil 5-9'da gösterildiği gibi güncellenerek çözülebilir. Şekilde de gösterildiği gibi, X Nesne Sınıfının a ve b nitelikleri ayrılarak veri modeline yeni eklenen Y Nesne Sınıfına taşınmıştır. X Nesne Sınıfı, Y Nesne Sınıfından türetilerek X Nesne Sınıfına üye olanların eskiden olduğu gibi a ve b niteliklerinin güncellemelerini almaları sağlanmıştır. Üye Katılımcının X Nesne Sınıfına üyeliği Y Nesne Sınıfı seviyesine taşınarak sadece ilgilendiği a ve b niteliklerinin güncellemelerini alması sağlanmıştır.



Şekil 5-9 İhtiyaç Duyulan Veri Kümesine Göre Veri Modelinin Ve Üyeliklerin Güncellenmesi

3. Paylaşılan Verilerin Güvenilirlik *(Reliability)* seviyesini gözden geçir: Yayımlama/Üye olma mimarisinde, veri modeli elemanları katılımcılar arasında iki farklı güvenilirlik seviyesinde paylaşılabilir:

    1. Reliable**:** Bu paylaşım yönteminde, yayımlanan verilerin tüm örnekleri ilgili veriye üye olan tüm katılımcılara kesinlikle ulaştırılır.

    2. Best Effort: Bu paylaşım yönteminde, yayımlanan verilerin tüm örneklerinin ilgili veriye üye olan tüm katılımcılara ulaştırılacağı kesin değildir.

    Paylaşılan verilerin tüm üyelere ulaşmasını garanti etmek için *reliable* yöntemde veri paketlerine bilgi başlıkları *(info header)* eklenmesi, geri besleme *(acknowledgement)* mekanizmalarının kurulması gibi çeşitli ek maliyetler vardır. Bundan dolayı iletişim maliyeti bağlamında verilerin *reliable* olarak paylaşılması, *best effort* paylaşımdan daha maliyetlidir. Bu bilgiler ışığında, iletişim maliyetlerini düşürüp uygun yerleştirme seçeneklerinin bulunabilmesine destek olmak için Benzetim Veri Değişim Modelindeki elemanların paylaşım güvenilirliği seviyeleri mümkün mertebe

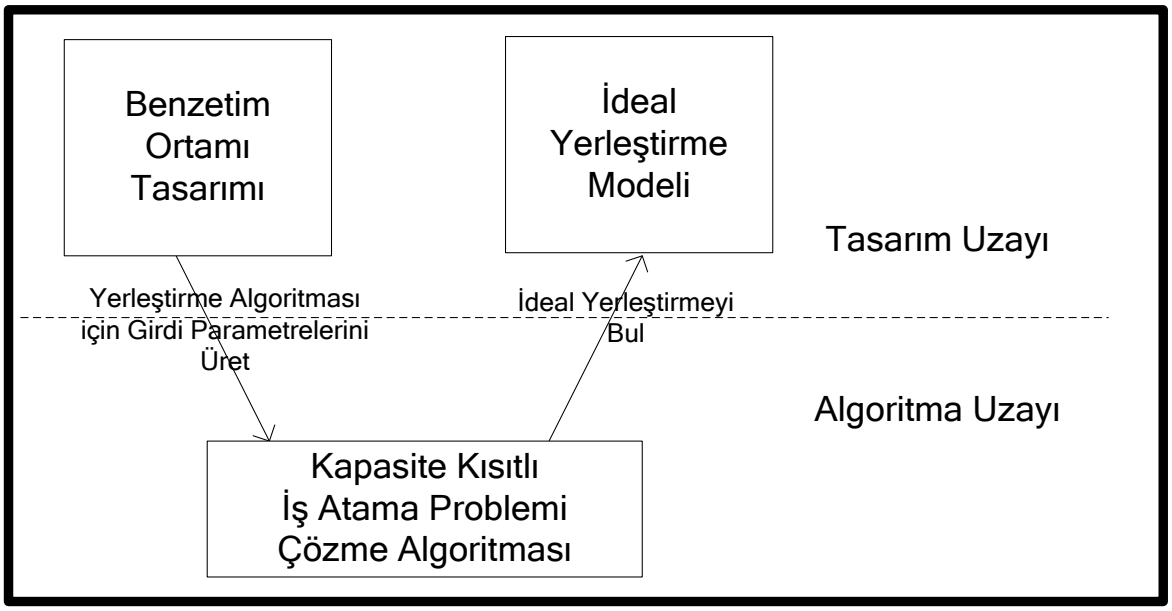
best effort olarak atanmalıdır. *Reliable* olarak tanımlanmış veri modeli elemanları incelenerek *best effort* olarak tanımlanıp tanımlanamayacakları irdelenmelidir. Örneğin bu tez çalışması kapsamındaki örnek durum çalışmasında platformların pozisyon bilgileri sıklıkla güncellenmektedir. Platform konum bilgisine üye olan potansiyel alıcıların pozisyon bilgisi üzerinde *dead reckoning* yöntemleri [Fujimoto 1999] uygulayacakları varsayılarak platformların pozisyon bilgilerinin *best effort* olarak paylaşılması değerlendirilebilir.

4. Fiziksel Kaynak Modelini Güncelle: Yukarıda sıralanan tüm öneriler benzetim ortamı tasarımı üzerinde değişiklik yapmaya dayanmaktadır. Bu önerilerin tümü uygulanmasına rağmen halen uygun bir yerleştirme alternatifi türetilemiyorsa, fiziksel kaynakları iyileştirmekten başka çare yoktur. Aslında bu durum endüstride sıklıkla karşılaşılan fiziksel kaynak ihtiyacının proje başında doğru kestirilemeyip yetersiz fiziksel kaynak alınması hatasını da büyük ölçüde engelleyecektir. Proje başlangıcında seçilen kaynakların yetersiz olması uzun vadede geliştirme sürecini baltalamakta ve projenin ilerleyen aşamalarında yeniden daha iyi konfigürasyonda kaynakların alınmasını gerektirmektedir. Bu durum hem zaman kaybına yol açmakta, hem de mali anlamda birden çok defa fiziksel kaynak satın almadan dolayı maliyetlerin yükselmesine sebep olmaktadır. Benzer şekilde, geliştirilecek sistemin kaynak ihtiyaçları projenin erken safhalarında kestirilememesinden dolayı, riski azaltmak adına proje başlangıcında gereğinden yüksek başarımlı fiziksel kaynakların alımı da izlenen yöntemlerden birisidir. Bu yöntem de aslında maliyetin yükselmesine sebep olmaktadır. Bu bağlamda bu tez kapsamında geliştirilen yaklaşım ve araçlar proje yaşam döngüsünün erken tasarım aşamasında ne düşük ne de yüksek kapasiteli olan, uygun fiziksel kaynakların seçilmesi için bilimsel destek sağlamaktadır.

5.8 Yerleştirme Modelini Üret

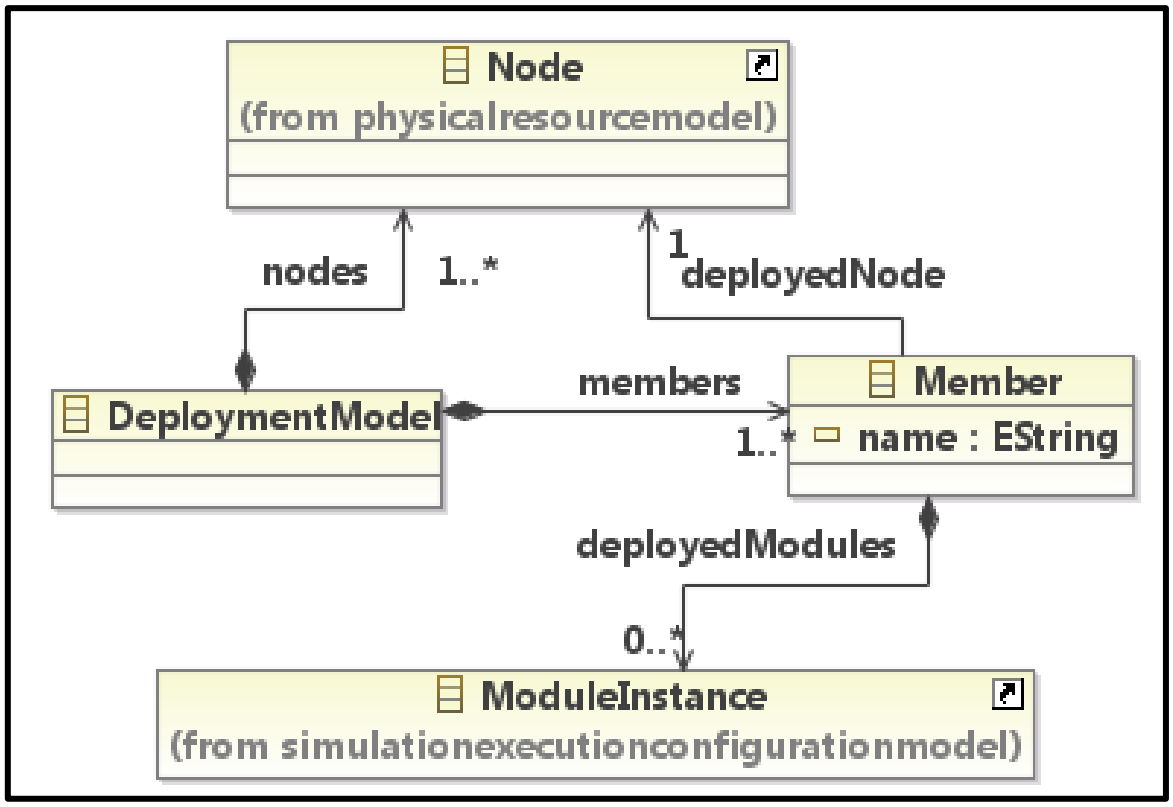
Bir önceki adımda uygun bir yerleştirme bulunduğu durumda Çizelge 5-3'tekine benzer bir çıktı tablosu elde edilecektir. Bu algoritma çıktısı, bu haliyle tasarım aşamasında kullanım için uygun değildir. Bu adımda algoritmanın çıktısı olan iş –

işlemci atama tablosu bir yerleştirme modeline (*deployment model*) dönüştürülecektir. Tasarım Uzayı ile Benzetim Uzayı arasındaki geçişler Şekil 5-10'de gösterilmiştir. Bir önceki adımda (Yerleştirme Algoritması için Girdi Parametrelerini Üret) problem tasarım uzayından algoritma uzayına taşınırken, Uygun Yerleştirmeyi Bulma adımında yeniden tasarım uzayına dönülmektedir.



Şekil 5-10 Tasarım Uzayı ve Algoritma Uzayı Arasında Geçişler

Uygun yerleştirme alternatiflerini göstermek için kullanılacak yerleştirme modelleri için bu çalışma kapsamında geliştirilen metamodel Şekil 5-11'de verilmiştir. Bir yerleştirme modelinde, benzetim çalıştırma konfigürasyonunda tanımlı olan modül olguları *(module instance)* üyelere (*Member*) yerleştirilir. Her bir üye de, şekilde gösterildiği gibi fiziksel kaynak modelindeki düğümlere (*Node*) yerleştirilir.



Şekil 5-11 Yerleştirme Metamodeli

5.9 Üretilen Yerleştirme Modellerini Değerlendir

Bu adımda, tasarımcı önceki adımda üretilen yerleştirme modellerinin kalitesini başka yerleştirme seçenekleriyle karşılaştırarak değerlendirir. Diğer yerleştirme seçenekleri uzman değerlendirmesiyle üretilmiş olabileceği gibi, bu tez çalışması kapsamında geliştirilen araç desteği ile de üretilmiş olabilir. Yaklaşımı destekleyen araç ailesi, yerleştirme seçeneklerini farklı kalite faktörlerine göre otomatik olarak karşılaştırmaya olanak sağlar. Tasarımcı, karşılaştırma sonuçlarına göre üretilen yerleştirme modelinin iletişim ve çalıştırma maliyetleri anlamında tatmin edici olup olmadığına karar verir. Tatmin edici bir yerleştirme seçeneği bulunduğu takdirde, uygun yerleştirme seçeneklerinin tanımlanması süreci sona erer. Aksi takdirde, tasarımcı araç desteği ile tasarım modeli analiz edilerek üretilen geri besleme raporunu inceler (geri besleme raporunun detayları 5.7.1 kesiminde anlatılmıştır). 5.7.1 kesiminde anlatıldığı gibi, geri besleme raporuna göre tasarımcı öncelikle benzetim çalıştırma konfigürasyonunu güncelleyerek tatmin edici bir yerleştirme alternatifi bulmaya çalışır. Benzetim çalıştırma konfigürasyonunu güncellemenin yeterli olmaması durumunda, tasarımcı benzetim tasarımı sürecinin başına dönerek benzetim tasarımını günceller.

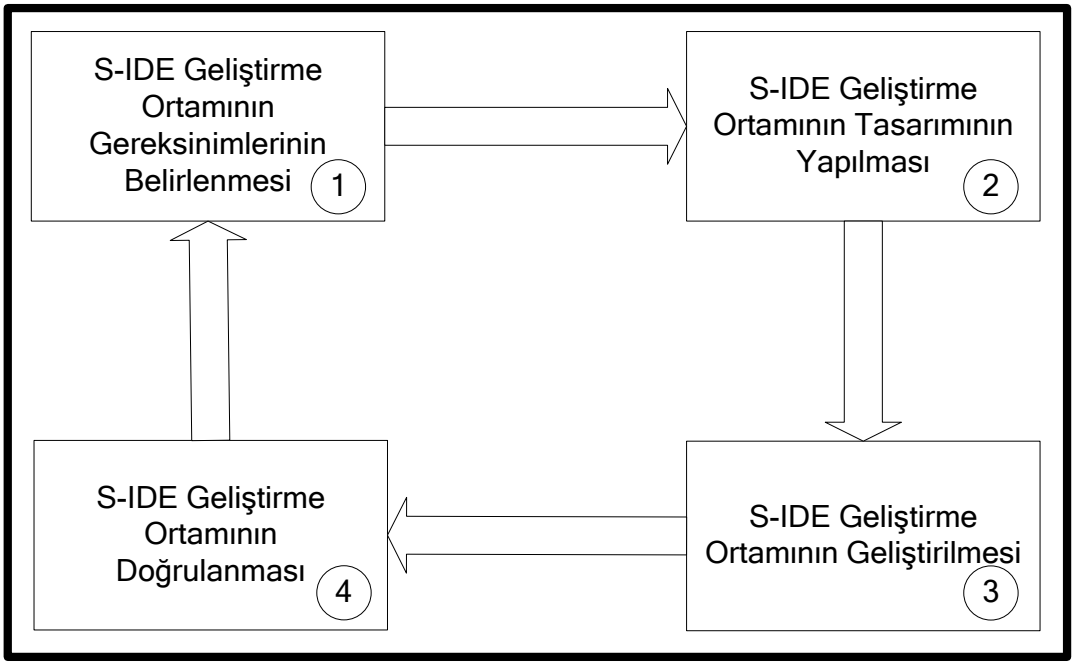
6 YAKLAŞIMI DESTEKLEYEN ARAÇ AİLESİ

Bir önceki kesimde uygun yerleştirme konfigürasyon seçeneklerini türetmek için geliştirilen yaklaşım anlatılmıştır. Genel anlamda, herhangi bir yaklaşımın kullanılabilir olması için yaklaşımı destekleyen ve uygulanmasını mümkün kılan araçların desteğine ihtiyaç vardır.

Bu kesimde, geliştirilen yaklaşımın adımlarının uygulanmasını desteklemek için bu tez çalışması kapsamında geliştirilen *S-IDE* (Simulation-IDE) bütünleşik geliştirme ortamı anlatılacaktır.

Önceki kesimde anlatılan yaklaşım araç desteği geliştirme bakış açısıyla incelendiğinde, bu yaklaşımı etkin bir biçimde destekleyecek araçların geliştirilmesinin basit, küçük çaplı bir yazılım geliştirme çalışması olmadığı anlaşılmıştır. Bu bağlamda, *S-IDE* geliştirme ortamının üretilmesinde yazılım mühendisliği prensiplerine uygun olarak gereksinim belirleme, tasarım, geliştirme ve doğrulama yaşam döngüsü adımları izlenmiştir.

Şekil 6-1'de gösterildiği gibi, üretim sürecinde yinelemeli bir yaşam döngüsü [Kruchten 2003] izlenmiştir.



Şekil 6-1 *S-IDE* Aracının Geliştirilmesinde İzlenen Yaşam Döngüsü Adımları

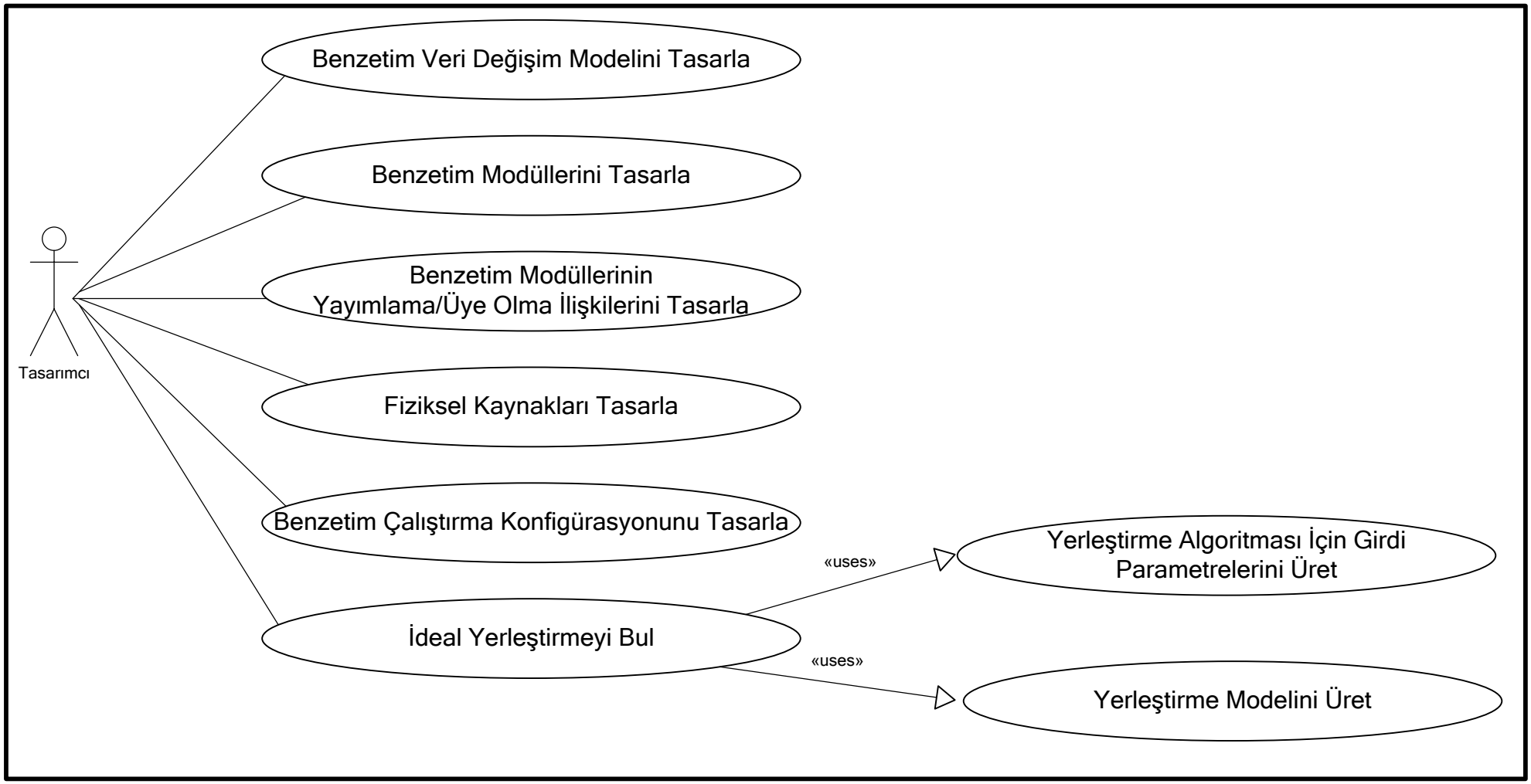
Bu bölümün kalan kesiminde *S-IDE* ortamının üretilmesinde izlenen yaşam döngüsü adımları sırasıyla açıklanacaktır.

6.1 *S-IDE* Geliştirme Ortamının Gereksinimlerinin Belirlenmesi

S-IDE geliştirme ortamının gereksinimleri, işlevsel ve işlevsel olmayan (örneğin ergonomi, başarım) gereksinimler olarak iki grupta ele alınmıştır.

6.1.1 *S-IDE* geliştirme ortamının işlevsel gereksinimleri

S-IDE geliştirme ortamının işlevsel gereksinimleri, 5. Kesimde detaylı olarak açıklanan yaklaşım adımlarından türetilmiştir. Yazılım mühendisliğinde bir sistemin gereksinimleri çıkarılırken sıklıkla kullanılan yöntemlerden birisi, sistemin aktörlerini ve yeteneklerini gösteren UML Kullanım Durumu Çizenekleri (*Use Case Diagram*) [Fowler 2003] oluşturmaktır. Şekil 6-2'de *S-IDE* aracının kullanım durumu çizeneği verilmiştir.



Şekil 6-2 *S-IDE* Aracı için UML Kullanım Durumu Çizeneği

Şekil 6-2'de gösterildiği gibi, sistemin "Tasarımcı" adı verilen tek bir aktörü vardır. Tasarımcı, yaklaşımın adımlarını sırasıyla *S-IDE* aracını kullanarak uygular. Diğer tüm adımlardan farklı olarak, "Yerleştirme Algoritması için Girdi Parametrelerini Üret" ve "Yerleştirme Modelini Üret" kullanım durumları "Tasarımcı" aktörü tarafından doğrudan tetiklenmezler. Bu iki kullanım durumu, "Tasarımcı" aktörü "Uygun Yerleştirmeyi Bul" kullanım durumunu tetiklediğinde otomatik olarak tetiklenirler.

6.1.2 *S-IDE* geliştirme ortamının işlevsel olmayan gereksinimleri

S-IDE aracının işlevsel gereksinimlerinin yanı sıra, kullanım ergonomisi ve başarım gibi alanlarda işlevsel olmayan gereksinimleri de vardır. Bu gereksinimler aşağıdaki gibi özetlenebilir:

1. *S-IDE* aracı, modelleme elemanlarının listelendiği bir öğe paleti (item palette) sağlamalıdır.

2. *S-IDE* aracı, modellemenin yapılacağı bir grafiksel düzenleme alanı sağlamalıdır.

3. *S-IDE* aracı, öğe paletindeki modelleme elemanlarının sürükle-bırak (drag-drop) yöntemiyle grafiksel düzenleme alanına eklenmesine olanak sağlamalıdır.

4. *S-IDE* aracı, modelleme alanına eklenen model elemanlarının hiyerarşik olarak listelendiği bir model elemanları ağacı sağlamalıdır.

5. *S-IDE* aracı, geliştirilen modellerin *XML Metadata Interchange – XMI* standardına [OMG 2011a] uyumlu bir biçimde saklanmasına olanak sağlamalıdır.

6. *S-IDE* aracı, kabul edilebilir bir sürede uygun yerleştirme seçeneklerini üretebilmelidir.

7. *S-IDE* aracı, tanımlanan kaynak düğümlerin toplam kapasitesi yerleştirilecek benzetim modülü olgularının toplam kaynak ihtiyacından düşükse, tasarımcıyı uyarmalıdır.

6.2 *S-IDE* Geliştirme Ortamının Tasarımının Yapılması

S-IDE Geliştirme Ortamının tasarımı yapılırken, tasarımın farklı perspektiflerle ele alınması uygun görülmüş ve aşağıda sıralanan alt tasarım adımları izlenmiştir:

(1) Geliştirme ortamının Model Güdümlü Mühendislik bakış açısıyla modellenmesi

(2) Geliştirme ortamı araçlarının tasarlanması ve çıktılarının belirlenmesi

(3) Uygun yerleştirme seçeneklerinin türetilmesinin algoritmik tasarımı

Bu kesimin kalanında yukarıda sıralanan tasarım ana başlıkları sırasıyla ele alınacaktır.

6.2.1 *S-IDE* geliştirme ortamının Model Güdümlü Mühendislik bakış açısıyla irdelenmesi

Bu kesimde, tez çalışması kapsamında geliştirilen metamodeller, metamodeller arasındaki bağımlılıklar ve gerçekleştirilen dönüştürümler ele alınacaktır. Şekil 6-3'te, *S-IDE* geliştirme ortamının model güdümlü bakış açısıyla üst seviye tasarımı verilmiştir. Bütünlüğü korumak ve tasarımın büyük resmini gösterebilmek amacıyla Şekil 6-3'te birçok kısaltma kullanılmıştır. Bu kısaltmaların açılımı Çizelge 6-1'de verilmiştir.

Çizelge 6-1 Şekil 6-3'de Kullanılan Kısaltmaların Açılımları

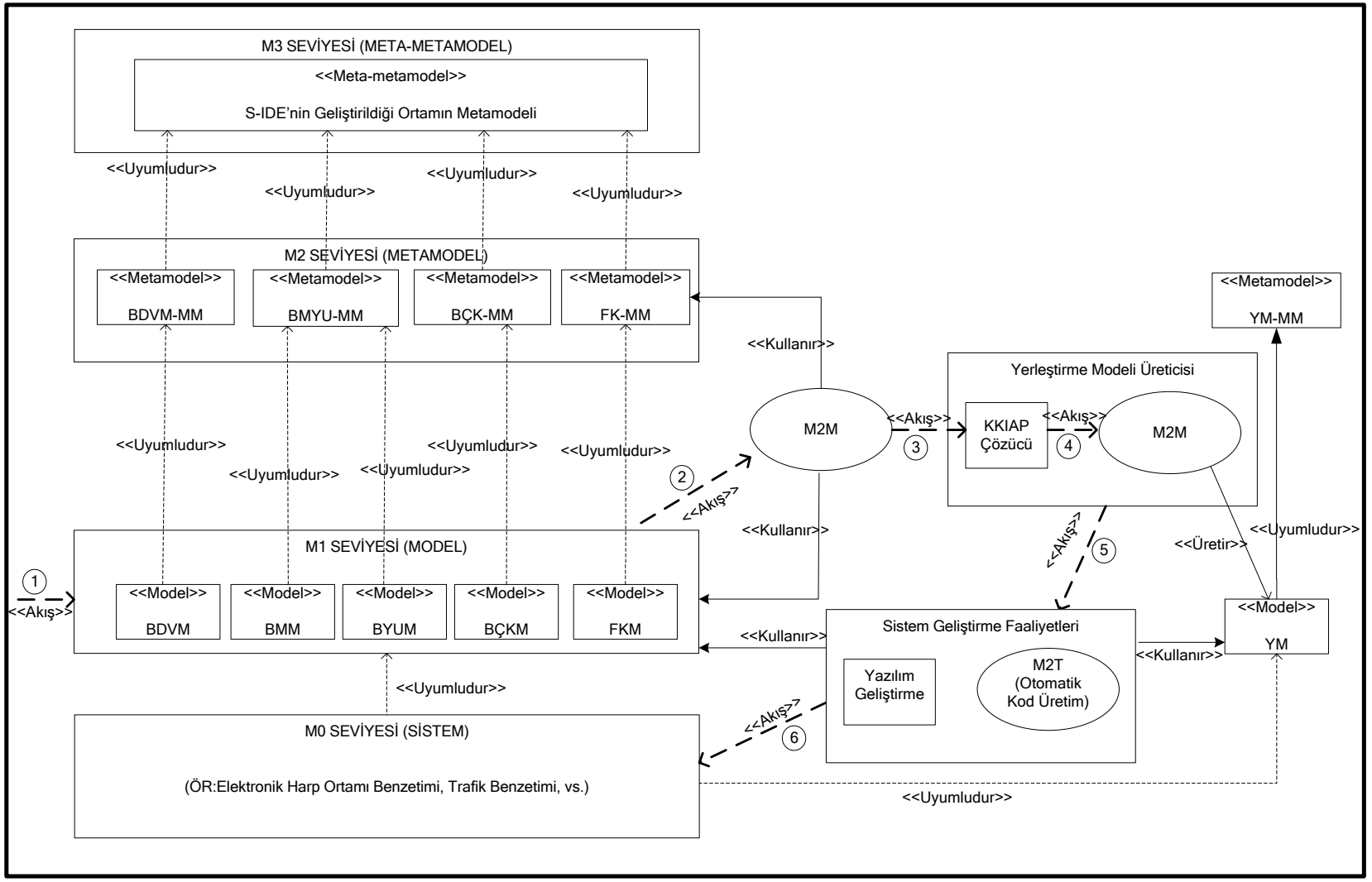
Kısaltma	Açılım
BVDM-MM	Benzetim Veri Değişim Metamodeli
BMYU-MM	Benzetim Modülleri ve Yayımlama/ Üye Olma Metamodeli
BÇK-MM	Benzetim Çalıştırma Konfigürasyonu Metamodeli
FK-MM	Fiziksel Kaynaklar Metamodeli
Y-MM	Yerleştirme Metamodeli
BVDM	Benzetim Veri Değişim Modeli
BMM	Benzetim Modülleri Modeli
BYUM	Benzetim Yayımlama/ Üye Olma Modeli
BÇKM	Benzetim Çalıştırma Konfigürasyonu Modeli
FKM	Fiziksel Kaynaklar Modeli
YM	Yerleştirme Modeli
KKIAP	Kapasite Kısıtlı İş Atama Problemi

Şekil 6-3'te sürecin akışı kesikli oklarla gösterilmiştir. Süreç M1 seviyesine karşılık gelen BVDM, BMM, BYUM, BÇKM (Kısaltmaların açılımları için Çizelge 6-1'ye bakınız) benzetim ortamı modellerinin oluşturulmasıyla başlar. M1 seviyesi bu modeller, 5. Kesimin alt başlıklarında açıklanan M2 seviyesi BVDM-MM, BMYU-MM, BÇK-MM,FK-MM metamodellerin olgularıdır. Benzetim ortamın tasarlanması ve uygun yerleştirme üretimi yetenekleri sağlayan *S-IDE* geliştirme ortamının da aslında bir şekilde tasarlanması gerekmektedir. M2 seviyesi metamodeller, *S-IDE* ortamının geliştirildiği ortamının Şekil 6-3'te M3 seviyesinde gösterilen meta-metamodelinin olguları olacaktır. İlerleyen kesimlerde *S-IDE* aracının geliştirilmesi anlatılırken M3 seviyesinde hangi ortamın kullanıldığı da açıklanacaktır.

1. adımda benzetim ortamının tasarımı yapıldıktan sonra, tasarımdan Kapasite Kısıtlı İş Atama Problemi çözüm algoritmasının girdilerinin otomatik olarak türetileceği 2. adıma geçilir. Bu adım Şekil 6-3'te modelden modele dönüştürüm (M2M) olarak gösterilmiştir. M2M dönüşümü girdi olarak M2 seviyesi metamodelleri ve M1 seviyesi benzetim ortamı tasarımını girdi olarak alıp iş atama parametrelerini türetir. M2M dönüşümünün gerçekleştirim detayları 6.2.3 kesiminde anlatılacaktır.

Benzetim ortamı tasarımından M2M dönüştürümü ile Kapasite Kısıtlı İş Atama Problemi çözüm algoritması parametreleri türetildikten sonra, Şekil 6-3'te "Yerleştirme Modeli Üreticisi" olarak gösterilen bileşen iş atama algoritmasını çalıştırır. Tasarımsal olarak *S-IDE* aracı herhangi özel bir iş atama algoritmasının kullanımını zorlamaz. Kullanılan algoritmanın seçimi *S-IDE* Geliştirme Ortamının Geliştirilmesi kesiminde anlatılacaktır.

3. adımda iş atama algoritması uygun bir yerleştirme bulabilirse, 4. adıma geçilir ve "Yerleştirme Modeli Üreticisi" bileşeni algoritmanın çıktısı olan İş-İşlemci eşleştirmesi tablosundan Yerleştirme Modeli üretmek için modelden modele (M2M) bir dönüştürüm başlatır. İş atama algoritmasının uygun bir yerleştirme bulamaması durumunda izlenecek yol ve algoritmanın ürettiği çıktının biçimi 5.6 – "Yerleştirme Algoritması İçin Girdi Parametrelerini Üret" kesiminde detaylı olarak anlatılmıştır.



Şekil 6-3 *S-IDE* Geliştirme Ortamının Model Güdümlü Bakış Açısıyla Tasarımı

Şekil 6-3'te 4. adımda iş atama algoritması çalıştırılıp çıktıları Yerleştirme Modeline dönüştürüldükten sonra, 5. Adım olan "Sistem Geliştirme Faaliyetleri" aşamasına geçilir. Bu adımın girdileri tasarımcının el ile geliştirdiği M1 seviyesi benzetim ortamı tasarımı ve otomatik olarak üretilen Yerleştirme Modelidir (Yerleştirme Modeli Şekil 6-3'te YM olarak gösterilmiştir). Bu adımda, günümüzde sistem geliştirme faaliyetleri kapsamında sıkça uygulanan Java, .NET gibi herhangi bir teknoloji ailesi kullanarak tasarımın gerçekleştirimi yöntemi uygulanabilir. Bu noktada süreci etkinleştirebilecek ve akademik anlamda katkı sayılabilecek bir çalışma, nihai sistem kodunun mümkün olan en çok kesimini modelden metne (M2T) dönüşüm yöntemleriyle otomatik olarak üretme yaklaşımının geliştirilmesidir. Bu adımın detayları ve gerçekleştirim yöntemi bu tez kapsamında ele alınmamış ve gelecek çalışmalara bırakılmıştır. Bu gelecek çalışma, tez metninin ilerleyen kesimlerinde gelecek çalışmaların anlatıldığı kesimde detaylı olarak ele alınacaktır.

Son adım olan 6. adımda, sistem geliştirme faaliyetleri sonucunda üretilen M0 seviyesi nihai sisteme geçiş yapılır. Bu tez çalışması kapsamında kullanılan Elektronik Harp Benzetimi örnek durum çalışması nihai sisteme örnek olarak gösterilebilir. M0 seviyesi nihai sistem, M1 seviyesindeki tasarımcının el ile geliştirdiği benzetim ortamı tasarımı ve otomatik olarak üretilen Yerleştirme modelinin olgusudur.

Farklı seviyelerdeki modeller ve model dönüştürümleri ele alındıktan sonra detaylandırılması gereken bir diğer nokta, *S-IDE* aracının çatısını oluşturan metamodellerin aralarındaki ilişkilerin belirlenmesidir. Bu ilişkiler, geliştirme aşamasında ilgili yazılım bileşenleri arasındaki ilişkileri belirleyecektir. İzleyen alt kesimde metamodeller arası bu bağımlılıklar ele alınacaktır.

6.2.1.1 Metamodeller arası bağımlılıkların tasarlanması

S-IDE geliştirme ortamının çatısı olan metamodeller ve aralarındaki bağımlılıklar Şekil 6-4'te gösterilmiştir. 5. Kesimde metamodeller detaylı olarak açıklandığından bu bölümde metamodellerin içerikleri tekrar açıklanmayacak, sadece bağımlılıkları ele alınacaktır.

Bu tez çalışması kapsamında geliştirilen metamodellerin bağımlılıkları ikiye ayrılabilir:

(1) Metamodellerin bu tez çalışması kapsamında geliştirilmeyen dış belirtimlere bağımlılıkları. Bu dış belirtimler, Şekil 6-4'te <<Dış Belirtim>> olarak işaretlenmişlerdir.

(2) Metamodellerin kendi aralarındaki bağımlılıkları.

Bu bölümün kalan kesiminde dış ve iç bağımlılıklar sırasıyla ele alınacaktır.

Metamodellerin Dış Bağımlılıkları:

Bu tez çalışması kapsamında yaklaşım ve araç desteği geliştirilirken DIS, HLA gibi herhangi bir standarda özel bir çözümden ziyade, koşut ve dağıtılmış benzetim sistemleri için genel bir çözüm oluşturulması hedeflenmiştir. Diğer alanlarda da olduğu gibi, benzetim sistemleri için de genel çözümler oluştururken, mevcut baskın standartları dikkate almak geliştirilen yaklaşımın kullanılabilirliği anlamında faydalı olacaktır. Örneğin benzetim sistemleri için genel bir kavramsal modelleme yaklaşımı olan Base Object Model (BOM) [SISO 2006] belirtimi altyapı olarak HLA OMT standardını kullanır [IEEE 2010c]. Benzer bir mantıkla, bu çalışma kapsamında geliştirilen metamodellerin bağımlı olduğu iki dış belirtim vardır:

(1) IEEE HLA 1516 standardının bir parçası olan HLA Nesne Modeli Şablonu (HLA Object Model Template – OMT) belirtimi [IEEE 2010c],

(2) *Discrete Event System Specification (DEVS)* [Zeigler 2003] belirtimi.

Benzetim Veri Değişim Metamodeli, 5.1 kesiminde detaylı olarak açıklandığı gibi, IEEE HLA 1516 Nesne Modeli Şablonunu genişletir. Bu ilişki Şekil 6-4'te Benzetim Veri Değişim Metamodeli ile HLA Nesne Modeli Şablonu arasındaki <<Genişletir>> ifadesi ile etiketlenmiş bir ilişki oku ile gösterilmiştir.

Benzetim Modülleri ve Yayımlama/Üye Olma Metamodeli, 5.2 Kesiminde anlatıldığı gibi, DEVS belirtiminde tanımlanan atomik ve bağlı *(coupled)* model kavramlarına benzer yapıtaşı tanımlamaları içerir. DEVS belirtimine olan bağımlılık, HLA OMT belirtimine olan bağımlılığa göre daha zayıftır. Bundan dolayı

Şekil 6-4'te Benzetim Modülleri ve Yayımlama/Üye Olma Metamodeli ile DEVS belirtimi arasındaki ilişki <<Kavramlarını Kullanır>> ifadesi ile etiketlenmiş ve UML gibi modelleme dillerinde daha zayıf ilişkileri göstermek için kullanılan kesikli ok gösterimi tercih edilmiştir.

Metamodellerin İç Bağımlılıkları:

Bu kesimde metamodeller arası iç bağımlılıklar geliştirilen yaklaşımda metamodellerin kullanım sıralamasına göre açıklanacaktır. Şekil 6-4'te metamodeller arası iç bağımlılıklar <<Bağımlıdır>> ifadesi ile etiketlenmiş oklar ile gösterilmiştir.

Benzetim Modülleri ve Yayımlama/Üye Olma Metamodeli, Benzetim Veri Değişim Metamodeline bağımlıdır. Veri Değişim metamodelinde tanımlı olan Nesne ve Etkileşim Sınıfları *(Object Class / Interaction Class)*, Benzetim Modülleri ve Yayımlama/Üye Olma metamodelinde benzetim modülleri arasındaki yayımlama/üye olma ilişkileri tanımlanırken kullanılır.

Fiziksel Kaynaklar Metamodeli, Şekil 6-4'te gösterildiği gibi herhangi bir metamodele bağımlı değildir.

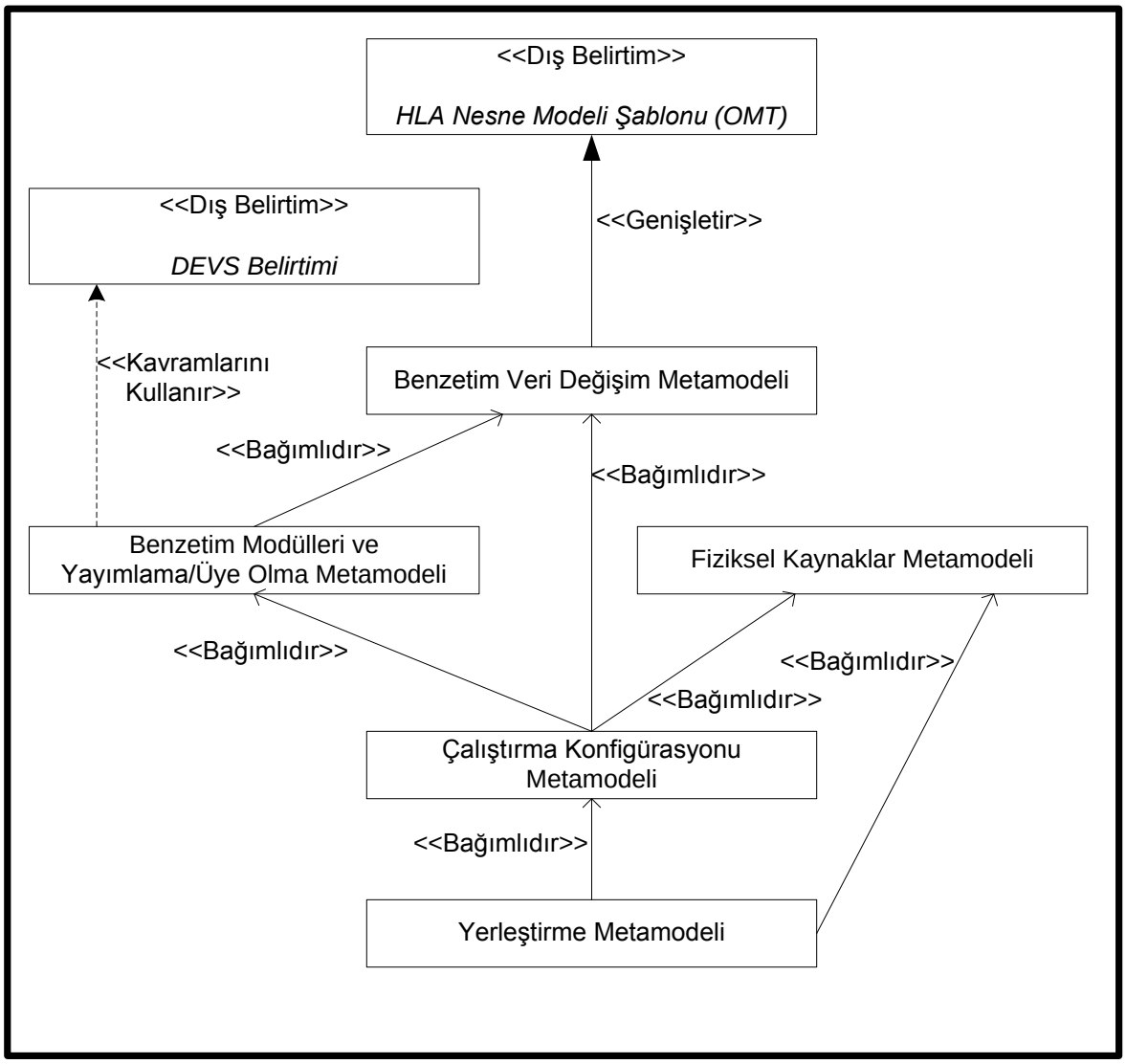
Metamodeller arasında en çok bağımlılığa sahip olan metamodel, Çalıştırma Konfigürasyonu Metamodelidir. Çalıştırma Konfigürasyonu Metamodeli;

- Benzetim modülü olgularının ilişkili oldukları modülleri tanımlamak için Benzetim Modülleri ve Yayımlama/Üye Olma Metamodeline,

- Benzetim modül olgularının yayımladıkları nesne modeli elemanlarını ne sıklıkta güncelleştiklerini tanımlamak için Benzetim Veri Değişim Metamodeline,

- Benzetim modül olgularının farklı düğümlerdeki çalıştırma maliyetlerini tanımlamak için de Fiziksel Kaynaklar Metamodeline bağımlıdır.

Son olarak Yerleştirme Metamodeli, mevcut düğümlerin bilgisi için Fiziksel Kaynak Metamodeline ve düğümler üzerine yerleştirilecek benzetim modül olgularının listesi için de Benzetim Çalıştırma Konfigürasyonu Metamodeline bağımlıdır.



Şekil 6-4 S-IDE Geliştirme Ortamı Metamodelleri ve Aralarındaki Bağımlılıklar

6.2.2 *S-IDE* geliştirme ortamı araçlarının tasarlanması ve çıktılarının belirlenmesi

Bu kesimde, bir önceki kesimde tanımlanan metamodeller, metamodeller arasındaki ilişkiler ve dönüştürümleri gerçekleştiren araçların tasarımı, ilişkileri ve çıktıları ele alınacaktır. Araçların açıklanmasının yanı sıra, bu bölümde tez kapsamında geliştirilen yaklaşımın araçlar kullanılarak uygulanmasının akışı da sağlanacaktır.

S-IDE Geliştirme Ortamı araçları, girdileri ve çıktılarıyla birlikte Şekil 6-5'te verilmiştir. Şekilde araçlar <<Araç>> ifadesi ile, araçların çıktıları ise <<Yapıtaşı>>

ifadesiyle gösterilmiştir. Numaralandırılmış çemberler yaklaşım uygulanırken araçların kullanım sırasını göstermektedir.

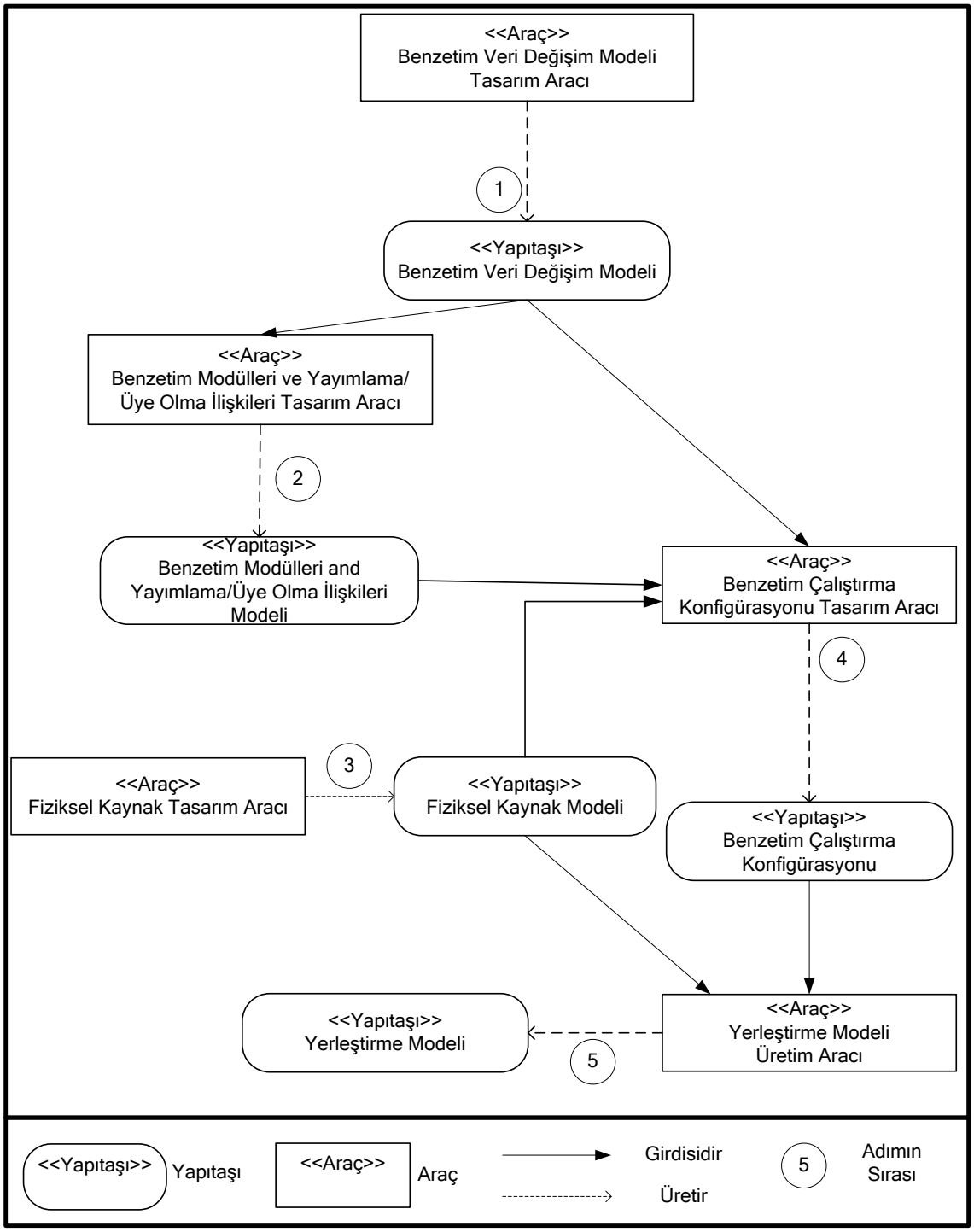
Benzetim Veri Değişim Modeli Tasarım aracı, Benzetim Veri Değişim Modellerini üretir. Üretilen bu veri modeli, Benzetim Modülleri ve Yayımlama Üye Olma İlişkileri Tasarım Aracı ve Benzetim Çalıştırma Konfigürasyonu Tasarım Aracının girdileridir.

Benzetim Modülleri ve Yayımlama Üye Olma İlişkileri Tasarım Aracı, Benzetim Çalıştırma Konfigürasyonu Tasarım Aracının bir diğer girdisi olan Benzetim Modülleri ve Yayımlama Üye Olma İlişkileri modelini üretir.

Fiziksel Kaynakları Tasarlama Aracı, Benzetim Çalıştırma Konfigürasyonu Tasarım Aracı ve Yerleştirme Modeli Üretim Aracının girdileri olan Fiziksel Kaynak Modelini üretir.

Benzetim Çalıştırma Konfigürasyonu Tasarım Aracı, Yerleştirme Modeli Üretim Aracının bir diğer girdisi olan Benzetim Çalıştırma Konfigürasyonunu tanımlamak için kullanılır.

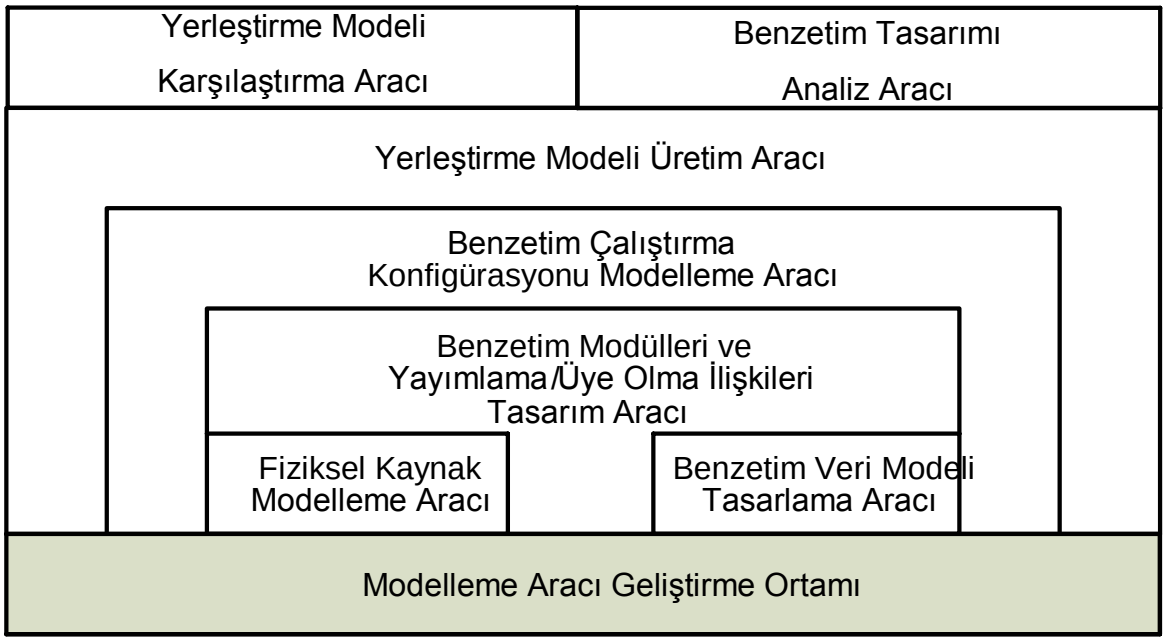
Son olarak Yerleştirme Modeli Üretim Aracı, kendisine sağlanan girdileri kullanarak uygun bir yerleştirme modeli türetmek için kullanılır.



Şekil 6-5 *S-IDE* Geliştirme Ortamı Araçları, Bağımlılıkları ve Yaklaşımın Araçlar Kullanılarak Uygulanmasının Akışı

Araçlar arasında Şekil 6-5'te <<Yapıtaşı>> ile etiketlenmiş tasarım modelleri kapsamında tanımlanan bağımlılıklar, araçların gerçekleştirim aşamasında da doğrudan bağımlılık oluşturacaktır. Günümüzde *S-IDE* Geliştirme Ortamı gibi modelleme ortamlarını sıfırdan geliştirmek oldukça maliyetli olduğundan bu tip

ortamların geliştirilebilmesi için tanımlanmış çeşitli modelleme aracı geliştirme ortamları mevcuttur. Bu bağlamda *S-IDE* geliştirme ortamının da bir araç geliştirme ortamında tasarlanıp üretilmesi sıfırdan geliştirilmesine göre oldukça maliyet ve zaman etkin bir yöntem olacaktır. Farklı araç geliştirme ortamları ve bu tez çalışması kapsamında kullanımları 6.3 kesiminde *S-IDE* geliştirme ortamının geliştirilmesi kapsamında irdelenecektir. *S-IDE* geliştirme ortamı araçları ve araç geliştirme ortamının katmanlı mimaride gösterimi Şekil 6-6'da gösterilmiştir. Modelleme Aracı Geliştirme Ortamı bu tez çalışması kapsamında geliştirilmemiş olması bağlamında diğer araçlardan farklı olduğu için şekilde de farklı renkle gösterilmiştir.



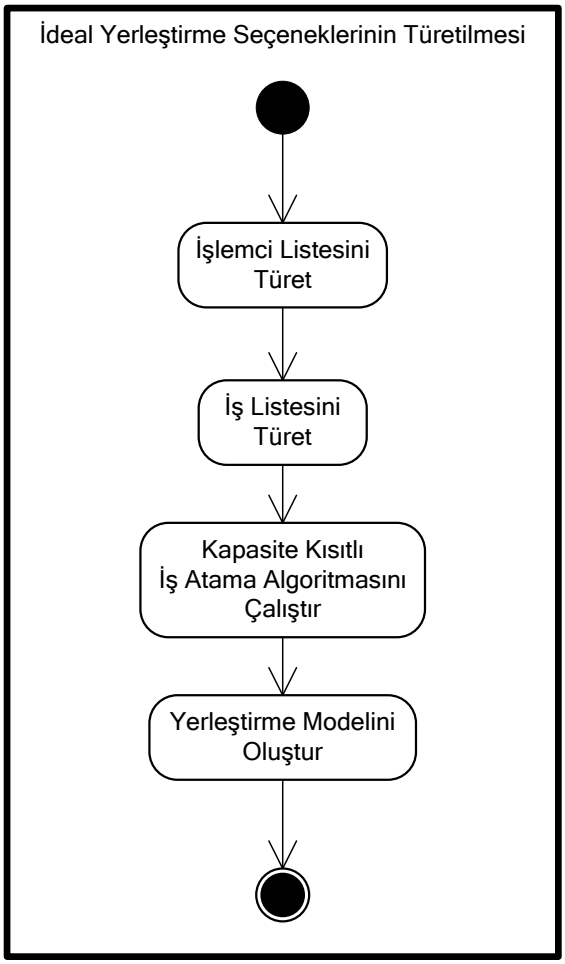
Şekil 6-6 *S-IDE* Geliştirme Ortamı Araçları ve Araç Geliştirme Ortamının Katmanlı Mimaride Gösterimi

6.2.3 Uygun yerleştirme seçeneklerinin türetilmesinin algoritmik tasarımı

Bu kesime kadar, *S-IDE* geliştirme ortamının tasarımı mimari bakış açısıyla ele alınmıştır. Bu kesimde ise, biraz daha detaya inilip *S-IDE* geliştirme ortamı kullanılarak uygun yerleştirme konfigürasyonu seçeneklerinin türetilmesi algoritmik açıdan ele alınacaktır.

Benzetim ortamı tasarlandıktan sonra, uygun yerleştirme alternatiflerinin algoritmik olarak türetilebilmesi için tasarımdan Kapasite Kısıtlı İş Atama Problemi çözüm

algoritmalarının ihtiyaç duyduğu parametrelerin türetilmesi, ilgili iş atama çözüm algoritmasının koşturulması ve algoritmanın sonuçları kullanılarak yerleştirme modelinin oluşturulması gerekmektedir. Bu akış, Şekil 5-10'de (Sayfa 56) tasarım uzayı ile algoritma uzayı arasında geçiş bakış açısıyla ele alınmıştır. Şekil 6-7'de ise, uygun yerleştirme seçeneklerinin türetilmesi algoritması üst seviyeden iş akışı bakış açısıyla ele alınmıştır. Şekil 6-7'de akış çizeneği verilen algoritmanın üst seviye kod taslağı Şekil 6-8'de verilmiştir.



Şekil 6-7 Üst Seviyeden Bakış ile Uygun Yerleştirme Seçeneklerinin Türetilmesi Algoritması – İş Akış Çizeneği

```
1. GENERATE_FEASIBLE_DEPLOYMENT (phy_resources,
   exec_config)
2.     processors ← EXTRACT_PROCESSORS (phy_resources)
3.     tasks ← EXTRACT_TASKS (exec_config)
4.     assignment_table ← EXECUTE_CTAP (tasks, processors)
5.     CREATE_DEPLOYMENT_MODEL (assignment_table)
```

Şekil 6-8 Uygun Yerleştirme Üretiminin Ana Algoritmik Akışı – Taslak Kod

Şekil 6-8'deki kod taslağının 1. satırında gösterildiği gibi, *GENERATE_ FEASIBLE_DEPLOYMENT* yöntemi, bir fiziksel kaynak modeli *(phy_resources)* ve bir çalıştırma konfigürasyonu modelini *(exec_config)* girdi parametreleri olarak alır. Kod taslağına dikkat edilirse, uygun yerleştirme bulma algoritmasının ihtiyaç duyduğu diğer girdiler olan (1) benzetim modülleri ve yayımlama/üye olma ilişkileri modeli ve (2) benzetim veri değişim modelleri *GENERATE_ FEASIBLE_DEPLOYMENT* yöntemine doğrudan parametre olarak aktarılmamaktadır. Bu modellerden ihtiyaç duyulan elemanlara benzetim çalıştırma konfigürasyonunun içerdiği referanslar üzerinden erişilecektir.

Şekil 6-7'deki akışın "İşlemci Listesi Üret" iş adımında ve Şekil 6-8'deki kod taslağının 2. satırında *EXTRACT_PROCESSORS* yöntemi ile gösterildiği gibi, uygun yerleştirme konfigürasyonu seçenekleri türetilirken öncelikle işlemci listesi türetilir. İşlemci listesi, Fiziksel Kaynak Tasarımındaki (kod taslağındaki adıyla *phy_resources*) düğümlerden türetilir. Bu adımda, temel olarak fiziksel kaynak modelindeki her düğüm için bir işlemci oluşturulmasıdır.

Sonraki adım, Şekil 6-7'deki akışın "İş Listesi Üret" iş adımında ve Şekil 6-8'deki kod taslağının 3. satırında *EXTRACT_TASKS* yöntemi ile gösterildiği gibi, iş listesi türetilir. İş listesinin türetilmesi kapsamında, işler arası iletişim maliyetlerinin hesaplanması da ele alınmaktadır. İş listesi türetilirken, Benzetim Çalıştırma Konfigürasyonu, Benzetim Modülleri ve Yayımlama/Üye Olma İlişkileri ve Benzetim Veri Modeli Tasarımları kullanılır. Bu adımda öncelikle benzetim çalıştırma konfigürasyonundaki her modül olgusu için bir "iş" oluşturulur, daha sonra modüller arası iletişim maliyetleri hesaplanır. Modüller arası iletişim maliyetlerinin hesaplanma yöntemi 6.2.3.1 kesiminde detaylı olarak açıklanacaktır.

Bir sonraki adım, Şekil 6-7'deki akışın "Kapasite Kısıtlı İş Atama Algoritmasını Çalıştır" iş adımında ve Şekil 6-8'deki kod taslağının 4. satırında gösterildiği gibi, iş atama algoritmasının çalıştırılmasıdır. Kod taslağında *assignment_table* değişkeni ile gösterilen algoritmanın çıktısı, işlerin işlemcilere eşleştirilmesini gösteren bir çizelgedir. Bu çizelge, aslında uygun yerleştirme seçeneğinin soyut bir belirtimidir.

Son adım, Şekil 6-7'deki akışın "Yerleştirme Modelini Oluştur" iş adımında ve Şekil 6-8'deki kod taslağının 5. Satırında gösterildiği gibi, bir önceki adımda çalıştırılan algoritmanın çıktısı olan iş-işlemci eşleştirmesi çizelgesinden yerleştirme modelinin üretilmesidir.

Şekil 6-8'de üst seviyeden taslak kodu verilen uygun yerleştirme üretimi algoritmasının taslak kodu Şekil 6-9'da verilmiştir.

```
1.  GENERATE_FEASIBLE_DEPLOYMENT (phy_resources, exec_config)
2.      processors ← EXTRACT_PROCESSORS (phy_resources)
3.      tasks ← EXTRACT_TASKS (exec_config)
4.      allocation_table ← EXECUTE_CTAP (tasks, processors)
5.      CREATE_DEPLOYMENT_MODEL (allocation_table)
6.
7.  EXTRACT_PROCESSORS (resources)
8.      create empty list processors
9.      for each node in resources do
10.         processor ← CREATE_PROCESSOR (node)
11.         append processor to processors list
12.     end for
13.     return processors
14.
15. EXTRACT_TASKS (exec_config)
16.     create empty list tasks
17.     for each source_module_inst in exec_config do
18.         task ← CREATE_TASK(source_module_inst)
19.         for each target_module_inst in exec_config do
20.             comm_cost←GET_COMM_COST (source_module_inst,
    target_module_inst)
21.             SET_COMM_COST(task, comm_cost, target_module_inst.ID)
22.         end for
23.         append task to tasks list
24.     end for
```

```
25.         return tasks
26.
27. CREATE_DEPLOYMENT_MODEL (allocation_table)
28.         for each task in allocation_table do
29.                 deployed_module ← CREATE_DEPLOYED_MODULE (task)
30.                 deployed_module.processor ← GET_PROCESSOR (allocation_table,
    task)
31.         end for
32.
33. GET_COMM_COST (source_module_inst, target_module_inst)
34.         communication_cost = 0
35.         if source_module_inst not equals target_module_inst then
36.                 for each publication in source_module_inst.publications do
37.                         data_exchange_object ← publication.data_exchange_object
38.                 if IS_SUBSRCIBER (target_module_inst,data_exchange_object)
39.                 then
40.                         communication_cost += CALCULATE_COST (publication)
41.                 end if
42.                 end for
43.         end if
44.         return communication_cost
45.
46. IS_SUBSCRIBER (target_module_inst,data_exchange_object)
47.         module ← target_module_inst.related_module
48.         if  module subscribes to data_exchange_object or a parent of it then
49.                 return TRUE
50.         end if
51.         else then return FALSE
52.         end else
53.
54. CALCULATE_COST (publication)
55.         data_exchange_object ← publication.data_exchange_object
56.         update_rate ← publication.update_rate
57.         return update_rate x SIZEOF (data_exchange_object)
58.
59. SIZEOF (data_exchange_object)
60.         size = 0
61.         if data_exchange_object has a parent_object then
62.                 size += SIZEOF (parent_object)
63.         end if
```

```
64.     if data_exchange_object is an Object Class then
65.         for each attribute in data_exchange_object.attributes do
66.             size += SIZEOF_DATATYPE (attribute.datatype)
67.         end for
68.     end if
69.     if data_exchange_object is an Interaction Class then
70.         for each parameter in data_exchange_object.parameters do
71.             size += SIZEOF_DATATYPE (parameter.datatype)
72.         end for
73.     end if
74.     return size
75.
76. SIZEOF_DATATYPE (parameter.datatype)
77.     size = 0
78.     if parameter.datatype is a BasicDatatype then
79.         size = ((BasicDatatype)parameter.datatype).size
80.     end if
81.     else then
82.         // simple, array, fixed record, enumerated ve
83.         // variant data türündeki elemanlarının boyutları basic türünde
84.         // elemanlara ulaşılıncaya kadar öz-yineli olarak hesaplanır.
85.         // Benzetim Veri Değişim modeli elemanlarının boyutlarının
            // hesaplanması 6.2.3.1 kesiminde detaylı olarak ele alınacaktır.
86.     end else
87.     return size
```

Şekil 6-9 Uygun Yerleştirme Üretiminin Algoritmik Akışı – Detaylı Taslak Kod

6.2.3.1 Benzetim modül olguları arasındaki iletişim maliyetlerinin hesaplanması

İki benzetim modül olgusu arasındaki toplam iletişim maliyeti, ilgili modül olgularının eşleşen yayımlama/üye olma ilişkilerinin maliyetleri toplanarak bulunabilir. Bu yöntem, matematiksel olarak Şekil 6-10'daki gibi ifade edilebilir. Şekil 6-10'daki ifadede kullanılan parametreler Çizelge 6-2'de açıklanmıştır.

Şekil 6-10'daki matematiksel ifade şu şekilde okunabilir:

"x modül olgusunun yayımladığı tüm BVDM elemanlarını dolaş.

Bu BVDM elemanlarından y modül olgusunun üye olduklarını iletişim maliyetine kat.

x modül olgusunun bir i BVDM elemanını yayımlamasının maliyeti, i elemanının boyutu ile x modül olgusunun i elemanını yayımlama sıklığının çarpımı ile hesapla.

x modülü için yapılan işlemin aynısını y modülü için de yaparak y modülünün yayımladığı ve x modülünün üye olduğu BVDM elemanlarından kaynaklanan iletişim maliyetini hesapla.

x ve y modül olguları için hesaplanan maliyetleri toplayarak iki modül olgusu arasındaki toplam iletişim maliyetini hesapla".

x ve y modül olguları arasındaki toplam iletişim maliyeti

$$\sum_{i=1}^{n} s_i u_{ix} a_{iy} + \sum_{j=1}^{m} s_j u_{iy} a_{jx}$$

Şu kısıtlar dahilinde:

$$(x, y) \in T$$
$$a_{iy} = \{0,1\}, i \in P_X$$
$$a_{jx} = \{0,1\}, j \in P_Y$$

(a_{iy} = 1, eğer *y* modül olgusu *i* elemana üye ise, aksi takdirde 0)

(a_{jx} = 1, eğer *x* modül olgusu *j* elemana üye ise, aksi takdirde 0)

Şekil 6-10 İki Modül Arasındaki İletişim Maliyetini Bulmanın Matematiksel İfadesi

Çizelge 6-2 İki Modül Arasındaki İletişim Maliyetini Bulmanın Matematiksel İfadesinin Parametrelerin Açıklamaları

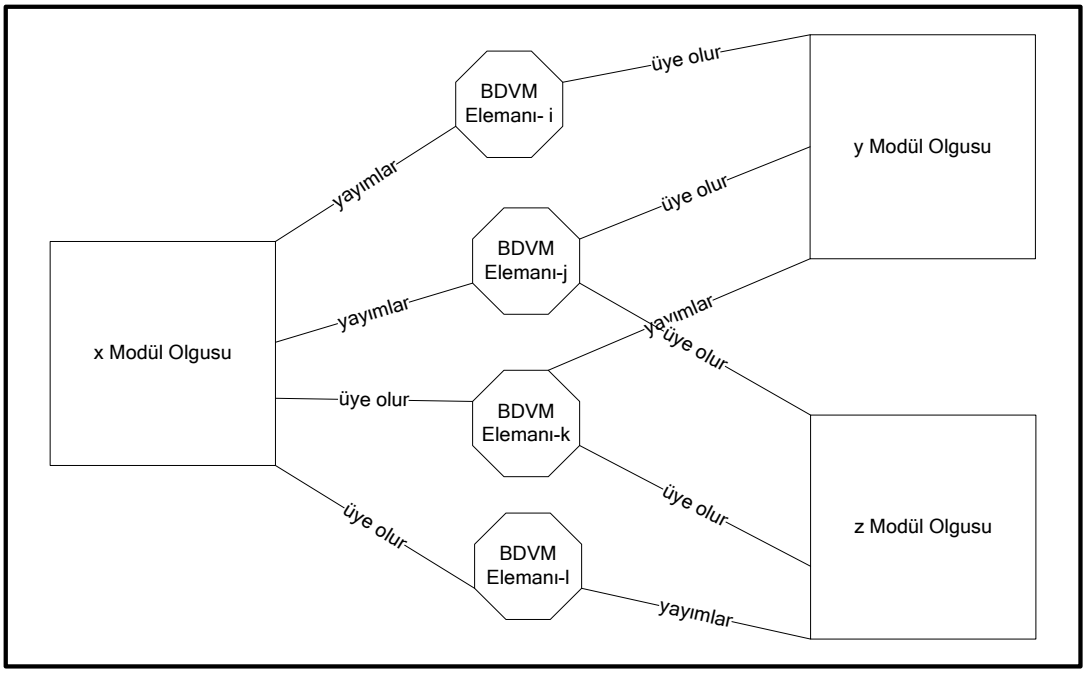
Parametre	**Açıklama**
T	Tasarımdaki modül olgusu kümesidir.
P_x	x modül olgusunun yayımladığı BVDM elemanlarının kümesidir. (1,2, … , n)
P_y	y modül olgusunun yayımladığı BVDM elemanlarının kümesidir. (1,2, … , m)
s_i	i BVDM elemanını temsil eden sınıfın bir olgusunun boyutudur. BVDM elemanlarının boyutlarının hesaplanması bu bölümün ilerleyen kesimlerinde ele alınacaktır.
u_{ix}	x modül olgusunun i BVDM elemanını güncelleme sıklığıdır.

Şekil 6-11'de örnek bir yayımlama/üye olma çizeneği verilmiştir. Örneğin bu çizenekteki x ve y modül olgularının arasındaki toplam iletişim maliyetinin hesaplanma yöntemini ele alalım. İki modül olgusu arasındaki veri alışverişi aşağıdaki kapsamda gerçekleşir:

- x modülünün i ve j BVDM elemanlarını yayımlaması ve y modülünün bu elemanlara üye olması
- y modülünün k BVDM elemanını yayımlaması ve x modülünün bu elemana üye olması

Bu bağlamda, x ve y modül olgularının arasındaki toplam iletişim maliyeti aşağıdaki formülle hesaplanmalıdır:

$$s_i u_{ix} + s_j u_{jx} + s_k u_{ky}$$



Şekil 6-11 Örnek Bir Yayımlama/Üye Olma Çizeneği

İzleyen kesimde, Şekil 6-10'da ve Çizelge 6-2'de gösterilen parametrelerden i Benzetim Veri Değişim Modeli elemanının boyutunu temsil eden si nin nasıl hesaplandığı henüz açıklanmamıştır. İzleyen alt kesimde, Benzetim Veri Değişim Modeli elemanlarının boyutlarının hesaplanması ele alınacaktır.

Benzetim Veri Değişim Modeli Elemanlarının Boyutlarının Hesaplanması

Bir benzetim veri değişim elemanının boyutu, ilgili elemanın alt elemanlarının boyutu özyineli olarak toplanarak hesaplanabilir. Örneğin RPR-FOM [SISO 2001a;2001b]'da tanımlı olan Designator nesne sınıfının bir olgusunun boyutunun hesaplanmasını ele alalım. Designator nesne sınıfı, RPR-FOM nesne modelinde Şekil 6-12'de gösterildiği gibi tanımlanmıştır.

```
(Class (ID 23)
    (Name "EmbeddedSystem")
    (Attribute (Name "EntityIdentifier")
            (DataType "EntityIdentifierStruct")
            ...
    )
     ...
 )
 (Class (ID 24)
    (Name "Designator")
    ...
    (SuperClass 23)
    ...
    (Attribute (Name "DesignatedObjectIdentifier")
            (DataType "RTIObjectIdStruct")
            ...
    )
    (Attribute (Name "DesignatorCode")
            (DataType "DesignatorCodeEnum16")
            ...
    )
    (Attribute (Name "DesignatorEmissionWavelength")
            (DataType "float")
            ...
    )
    (Attribute (Name "DesignatorOutputPower")
            (DataType "float")
            ...
    )
    ...
 )
```

Şekil 6-12 RPR-FOM Designator Nesne Sınıfı Tanımı

Designator nesne sınıfı Şekil 6-12'de gösterildiği gibi EmbeddedSystem nesne sınıfından türemektedir. Bundan dolayı bir *Designator* nesne sınıfının olgusunun boyutu, *Designator* ve *EmbeddedSystem* sınıflarının tüm niteliklerinin boyutlarının toplamına eşittir.

Designator sınıfının float türünde olan *DesignatorOutputPower* niteliği gibi temel veri türünde (Basic Datatype) olan niteliklerin boyutları doğrudan hesaplanabilir (ör: integer 4 byte, long 8 byte gibi).

Temel veri türünde olmayan niteliklerin hesaplanmasında özyineli olarak temel veri türlerine ulaşıncaya kadar tüm alt elemanlar için boyut hesaplaması yapılmalıdır. Örneğin *EmbeddedSystem* sınıfının *EntityIdentifier* niteliğinin türü *EntityIdentifierStruct*'tır. *EntityIdentifierStruct* veri türünün RPR-FOM tanımı Şekil 6-13'te verilmiştir. *EntityIdentifier* niteliğinin boyutu, *EntityIdentifierStruct*'ın *FederateIdentifier* ve *EntityNumber* alanlarının boyutlarının toplamına eşittir. Aynı özyineli boyut hesaplama mantığı temel bir veri türünde olmadıkları için *FederateIdentifier* ve *EntityNumber* alanları için de geçerlidir. Bu tip elemanlar için temel veri türlerine ulaşıncaya kadar boyut hesaplama algoritması çalışmalıdır.

```
(ComplexDataType (Name "EntityIdentifierStruct")
    (ComplexComponent (FieldName "FederateIdentifier")
        (DataType "FederateIdentifierStruct")
                        …
    )
    (ComplexComponent (FieldName "EntityNumber")
        (DataType "unsigned short")
         …
    )
)
```

Şekil 6-13 *EntityIdentifierStruct* Sınıfının RPR-FOM tanımı

6.3 *S-IDE* Geliştirme Ortamının Geliştirilmesi

Bu kesimde, önceki kesimlerde gereksinimleri ve üst seviye tasarımı verilen *S-IDE* geliştirme ortamının geliştirilmesi ele alınacaktır. Geliştirme süreci, (1) kullanılacak

teknoloji ve araçların belirlenmesi ve (2) seçilen geliştirme ortamı kullanılarak araçların geliştirilmesi olmak üzere iki alt adımda ele alınacaktır.

6.3.1 *S-IDE* geliştirme ortamı geliştirilirken kullanılacak teknoloji ve araçların belirlenmesi

S-IDE geliştirme ortamında önceki kesimlerde tanımlanan metamodellere uygun olarak benzetim ortamı tasarımın yapılması ve bu tasarımdan otomatik olarak yerleştirme modelinin üretilmesi gereksinimlerini karşılamalıdır. Geliştirilecek aracın 6.1.2 kesiminde tanımlanan grafiksel modelleme ortamı ve sürükle-bırak desteği gibi kullanımı kolaylaştıracak gereksinimlerin gerçekleştirimi irdelendiğinde ihtiyaç duyulan aracın bu tip uygulamalar geliştirmek için tanımlanmış hazır ürünler kullanılmadan sıfırdan geliştirilmesinin oldukça maliyetli ve zaman alıcı olacağı anlaşılmaktadır.

Literatürde, *S-IDE* geliştirme ortamına benzer grafiksel modelleme ortamları geliştirmek için tanımlanmış çeşitli Model Güdümlü Mühendislik ortamları mevcuttur. Tez çalışmasında araç geliştirme aşamasına gelindiğinde, geliştirme ortamı seçeneklerinin güçlü ve zayıf yönleri değerlendirilerek Uygun bir geliştirme ortamı seçimi yapabilmek için çeşitli kıstaslar tanımlanmış ve tanımlanan kıstasların nihai değerlendirmedeki payları ağırlıklandırılmıştır (Çizelge 6-3).

Çizelge 6-3 Modelleme Aracı Geliştirme Ortamı Alternatiflerinin Değerlendirilmesi için Kıstaslar ve Ağırlıkları

Kıstas	Açıklama	Ağırlık Yüzdesi
Geliştirme Kolaylığı	Geliştirme ortamı kullanılarak yeni modelleme aracı geliştirme sürecinin kolaylığı, yardımcı araçların mevcut olması önemlidir. Bu kapsamda seçilecek ortamın model geliştirme sürecini etkinleştirecek çeşitli araçlar sağlaması önemlidir.	**20**
Yaygın Kullanım	Herhangi bir yazılım uygulaması gibi, bir modelleme aracı geliştirme ortamının yaygın kullanılıyor olması, ortamın hatalarının büyük ölçüde ayıklanmış olmasını sağlayacaktır. Ayrıca uzun vadede geliştirilen modelleme ortamının başka araçlarla birleştirilmesi ve genişletilmesi açısından da seçilecek ortamın yaygın kullanımı önemlidir.	**10**
Uygulama Programlama Arayüzü *(Application Programming Interface - API)* Desteği	Geliştirilen modellerin programatik olarak işlenip dönüştürülebilmesi ve yine programatik olarak yeni model oluşturulabilmesi için seçilecek ortamın yetenekli bir uygulama programlama arayüzü sağlaması önemlidir. Örneğin benzetim ortamı tasarımından Kapasite Kısıtlı İş Atama Algoritması parametrelerinin türetilebilmesi için benzetim ortamı modeli elemanlarının etkin biçimde sorgulanabilmesi ve iş atama algoritması sonuçlarına göre yerleştirme modelinin otomatik olarak üretilebilmesi gerekmektedir.	**25**

Kıstas	Açıklama	Ağırlık Yüzdesi
Kaynak Belge Yaygınlığı ve Camia Genişliği	Seçilen ortam ile modelleme aracı geliştirmesi yapılırken çeşitli sorunlarla karşılaşılacağı aşikârdır. Seçilen ortam için yeterli belgelemenin olması önemlidir. Bazı durumlarda, karşılaşılan sorun kaynak belgeler incelenmesine rağmen çözülemeyebilir. Bu durumda internet ortamında soru sorulabilecek destek forumlarının ve aktif bir kullanıcı camiasının olması önemlidir.	**15**
Lisanslama Modeli	Seçilecek modelleme ortamının lisansının ticari olmaması, geliştirilecek aracın kullanıcılar tarafından erişilebilirliğini arttıracaktır. Aynı zamanda modelleme ortamı geliştirilirken geliştirme lisans ücreti ödenmemesi önemlidir.	**25**
Kaynak Kodun Açık Olması	Modelleme ortamının kaynak kodunun açık olması gerektiği durumlarda geliştirme ortamı koduna müdahale edilmesini ve kodun incelenerek herhangi bir durumda ortamın sergilediği davranışının anlaşılmasını kolaylaştıracaktır.	**5**

Modelleme aracı geliştirme ortamı seçimi ile ilgili kıstaslar ve ağırlıkları Çizelge 6-3'teki gibi belirlendikten sonra, literatürde tanımlanmış olan araç ve ortamlar incelenmiş, alternatifler ön elemeden geçirilip üç aday geliştirme ortamının daha detaylı incelenmesine karar verilmiştir. Bu bölümün izleyen kesiminde önce bu araçlarla ilgili kısaca bilgi verilecek, ardından Çizelge 6-3'te verilen kıstaslar ve ağırlıklarına göre alternatif araçlar puanlandırılacaktır.

6.3.1.1 Alternatif – 1: ISIS Generic Modeling Environment – GME

Generic Modeling Environment - GME, Vanderbilt Üniversitesine bağlı ISIS enstitüsünde geliştirilen, alana özel bir modelleme ortamıdır [ISIS 2012].

GME aracının açık kaynak kodlu ve ücretsiz olması önemli bir avantajdır.

GME aracında, metamodelleme dili olarak UML [Fowler 2003; OMG 2011b], kısıtları ifade etmek için de OCL [OMG 2006a] kullanılır.

GME aracıyla üretilen modelleri programatik olarak işleyebilmek için Microsoft COM [Templeman and Mueller 2003] uyumlu eklentiler geliştirmek gerekmektedir.

6.3.1.2 Alternatif – 2: Eclipse Modeling Project

Eclipse Platformu [Aniszczyk and Gallardo 2007], Model Güdümlü Mühendisliğin endüstriyel anlamda uygulanmasını mümkün kılmak için "Eclipse Modeling Project" başlığı altında birçok eklenti ve araç sağlar [Gronback 2009].

Eclipse modelleme ortamı, açık kaynak kodlu ve ücretsizdir.

Eclipse modelleme ortamında, metamodelleme ortamı olarak EMF'in [Budinsky et al. 2003] Ecore modeli kullanılır [Gronback 2009].

Eclipse modelleme ortamında, modeller Query/View/Transformation (QVT) [QVT 2012] veya XPand [XPand 2012] gibi eklentilerle sorgulanıp işlenebileceği gibi, modeller üzerinde EMF uygulama çatısının sağladığı yeteneklerle doğrudan Java programlama dili ile her türlü sorgulama ve güncelleme işlemi yapılabilir.

6.3.1.3 Alternatif – 3: Microsoft Visual Studio Visualization & Modeling SDK

Microsoft, Alana Özel Diller (Domain Specific Language - DSL) ve Model Güdümlü Mühendislik alanlarında yetenek sağlamak için Visual Studio .NET ortamına

eklenti olarak Visualization & Modeling SDK aracını geliştirmiştir [Cook et al. 2007].

Microsoft Visual Studio Visualization & Modeling SDK ortamının en büyük sorunu ticari bir ürün olan Microsoft Visual Studio .NET ortamına bağımlı olmasıdır.

Microsoft Visual Studio Visualization & Modeling SDK ortamında, metamodel geliştirmek için temel olarak UML sınıf çizeneklerine benzeri tanımlamalar kullanılır.

Microsoft Visual Studio Visualization & Modeling SDK ortamında üretilen modeller temel olarak .NET ortamında işlenebileceği gibi XML olarak kaydedilip dış herhangi bir ortamda da kullanılabilir.

İncelenen üç modelleme aracı geliştirme ortamı seçeneği incelendikten sonra, bu araçların Çizelge 6-3'te verilen kıstaslar ve ağırlıklarına göre puanlandırması yapılmıştır. Çizelge 6-4'te gösterilen puanlandırma sonuçları çizelgesinde de gösterildiği gibi, Eclipse Modelleme ortamı açık ara en yüksek puanı almıştır.

İzleyen kesimde, *S-IDE* Geliştirme Ortamının Eclipse Modelleme Ortamında geliştirilmesi ele alınacaktır.

Çizelge 6-4 Modelleme Aracı Geliştirme Ortamı Alternatiflerinin Kıstaslara Göre Puanlandırılması

Kıstas (Ağırlık Yüzdesi)	**Araç Seçeneği**		
	ISIS GME	Eclipse Modeling Project	Microsoft Visual Studio Visualization & Modeling SDK
Geliştirme Kolaylığı (20)	15	18	15
Yaygın Kullanım (10)	3	9	5
Uygulama Programlama Arayüzü *(Application Programming Interface – API)* Desteği (25)	10	23	10
Kaynak Belge Yaygınlığı ve Camia Genişliği (15)	5	13	5
Lisanslama Modeli (25)	25	25	0
Kaynak Kodun Açık Olması (5)	5	5	0
Toplam Puan:	**63**	**93**	**35**

6.3.2 Seçilen geliştirme ortamı kullanılarak *S-IDE* geliştirme ortamının geliştirilmesi

S-IDE geliştirme ortamı Eclipse platformu [Aniszczyk and Gallardo 2007] üzerine inşa edilen bir grup eklentiden (*plug-in*) oluşur.

S-IDE kapsamında geliştirilen eklentiler, Eclipse Modeling Framework (EMF) [Budinsky et al. 2003], ve Graphical Modeling Framework (GMF) [Voelter et al. 2006] gibi bir grup Eclipse uygulama çatısı eklentileri üzerine kurulmuştur.

EMF, metamodel geliştirme (bir önceki kesimde geliştirilen yaklaşım kapsamında anlatılan metamodeller gibi) ve otomatik kod üretme için kullanılan temel Eclipse uygulama çatısı eklentilerinden birisidir.

GMF ise, grafiksel modelleme ortamları oluşturmak için kullanılan bir uygulama çatısıdır. GMF, EMF ve GEF *(Graphical Editing Framework)* uygulama çatıları üzerine kurulmuştur. GMF uygulama çatısı olmadan doğrudan GEF kullanarak modelleme araçları geliştirmek de mümkündür, fakat bu yöntemle araç geliştirmek oldukça karmaşıktır ve uzun süre almaktadır. GMF modelleme aracı geliştirme sürecini hızlandırmak ve kolaylaştırmak amacıyla tanımlanmış bir uygulama çatısıdır.

GMF modelleme aracı geliştirme sürecini doğrudan GEF kullanarak geliştirmeyle karşılaştırıldığında önemli ölçüde etkinleştirmektedir, fakat modelleme aracı geliştirmek halen kolay bir süreç değildir. Bu bağlamda, EMF ve GMF kullanarak modelleme aracı geliştirme sürecini etkinleştirmek ve kolaylaştırmak için çeşitli Eclipse eklenti ve araçları geliştirilmiştir.

Modelleme ortamı geliştirme sürecini kolaylaştıran ve bu tez çalışması kapsamında *S-IDE* geliştirme ortamı geliştirilirken de kullanılan örnek iki Eclipse eklentisi Emfatic ve EuGENia aşağıda kısaca açıklanmıştır.

Emfatic Modelleme Dili ve Geliştirme Ortamı:

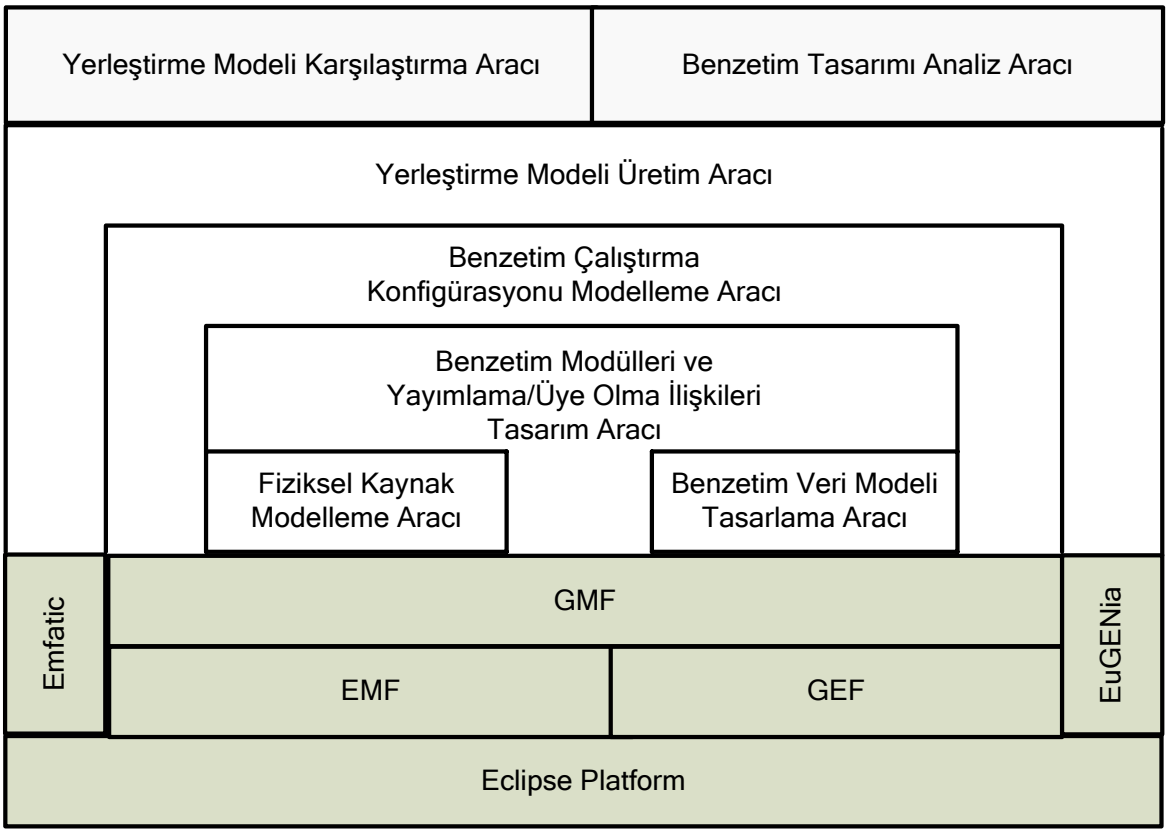
Emfatic [Daly 2004], EMF modellerini oluşturma ve güncelleme sürecini etkinleştirmek için tanımlanmış bir Eclipse eklentisidir. Emfatic, temel olarak metamodel oluşturmak için kullanılacak dil tanımı ve bu dil tanımı için bir geliştirme ortamı sağlar.

GMF Yardımcı Aracı EuGENia:

Epsilon projesinin [Kolovos et al. 2006] bir parçası olan EuGENia [Kolovos et al. 2010] eklentisi, GMF uygulama çatısının kullanımını kolaylaştırmak için çeşitli otomatik üretim araçları içerir. EuGENia eklentisi, Emfatic ile birlikte kullanılır. Emfatic dilinde metamodel tanımlaması yapılırken, metamodel elemanlarının olgularının nihai model editöründe nasıl görselleştirileceklerine dair tanımlamalar da yapılır. EuGENia eklentisi, bu görselleştirme tanımlamalarını kullanarak gerekli GMF kaynak dosyalarını otomatik olarak üretir.

Daha önce Şekil 6-6'da, henüz kullanılacak modelleme aracı geliştirme ortamı belirlenmeden önceki *S-IDE* geliştirme ortamı araç mimarisi verilmişti.

Şekil 6-14'te geliştirme ortamı olarak Eclipse modelleme ortamı seçimine göre güncellenmiş *S-IDE* geliştirme ortamı araç mimarisi verilmiştir.



Şekil 6-14 *S-IDE* Geliştirme Ortamı Araçları ve Araç Geliştirme Ortamının Katmanlı Mimaride Gösterimi

İzleyen kesimde, Eclipse modelleme ortamı kullanılarak modelleme aracı geliştirilmesi sürecinin detayları örnek bir proje ile anlatılmıştır.

6.3.2.1 Eclipse modelleme ortamı ile modelleme aracı geliştirme – Örnek Uygulama: Benzetim Veri Değişim Modeli Aracı

Bu kesimde Eclipse Modelleme Ortamı ile örnek bir araç geliştirilmesi süreci örnek bir uygulama ile ele alınacaktır. Anlatılan yöntemin Eclipse Modelleme ortamı ile araç geliştirmenin tek yöntemi olmadığı, bu tez çalışması kapsamında *S-IDE* geliştirme ortamı geliştirilirken izlenen yöntem olduğu not edilmelidir.

Bu tez kapsamında Eclipse Modelleme ortamında araçlar geliştirilirken izlenen sürecinin adımları kabaca aşağıdaki gibi sıralanabilir:

1. Yeni bir modelleme projesinin oluşturulması,

2. Geliştirilecek modelleme aracının metamodelinin (Eclipse terminolojisiyle, *Domain Model*'in) Ecore modeli olarak tanımlanması,

3. Metamodel elemanlarının görselleştirme biçimlerinin Emfatic dili ile tanımlanması,

4. Metamodelden EMF üretici modelinin (*EMF Generator Model - .genmodel*) üretilmesi,

5. Metamodelden EuGENia aracı ile GMF Grafiksel Tanımlama Modeli *(GMF Graphical Definition Model - .gmfgraph)*, Araç Tanımlama Modeli *(GMF Tooling Definition Model- .gmftool)* ve Eşleme Modelinin *(GMF Mapping Model - .gmfmap)* üretilmesi,

6. GMF Eşleme Modelinden GMF Çizenek Düzenleyicisi Üretici Modelin *(GMF Diagram Editor Gen Model - .gmfgen)* üretilmesi,

7. Çizenek Düzenleyici Modelde gerekiyorsa değişiklik yapılması (ör: diğer modelleme araçlarına referans verilmesi),

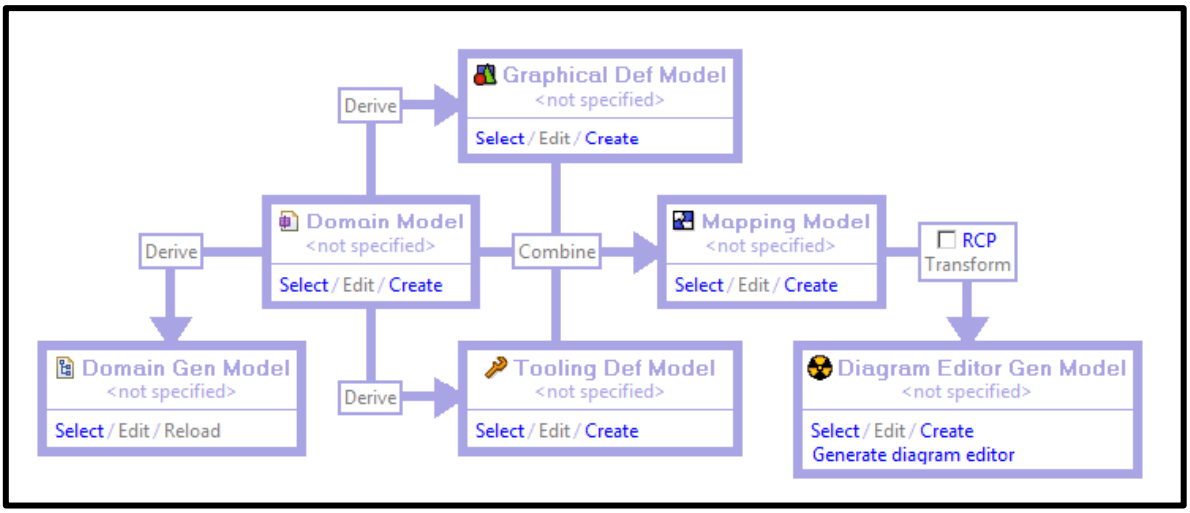
8. EMF üretici modelden modelleme aracının altyapısını oluşturacak Java projelerinin ve kaynak kodların üretilmesi,

9. GMF Çizenek Düzenleyicisi Üretici modelden grafiksel modelleme aracı Java projelerinin ve kaynak kodlarının üretilmesi,

10. Üretilen kaynak kod üzerinde gerekli uyarlamaların yapılması (Bu adım zorunlu değildir, uyarlama ihtiyacı varsa uygulanır).

Örneğin, Benzetim Veri Değişim Modeli aracı için yukarıda sıralanan adımların uygulanması aşağıda kısaca özetlenmiştir.

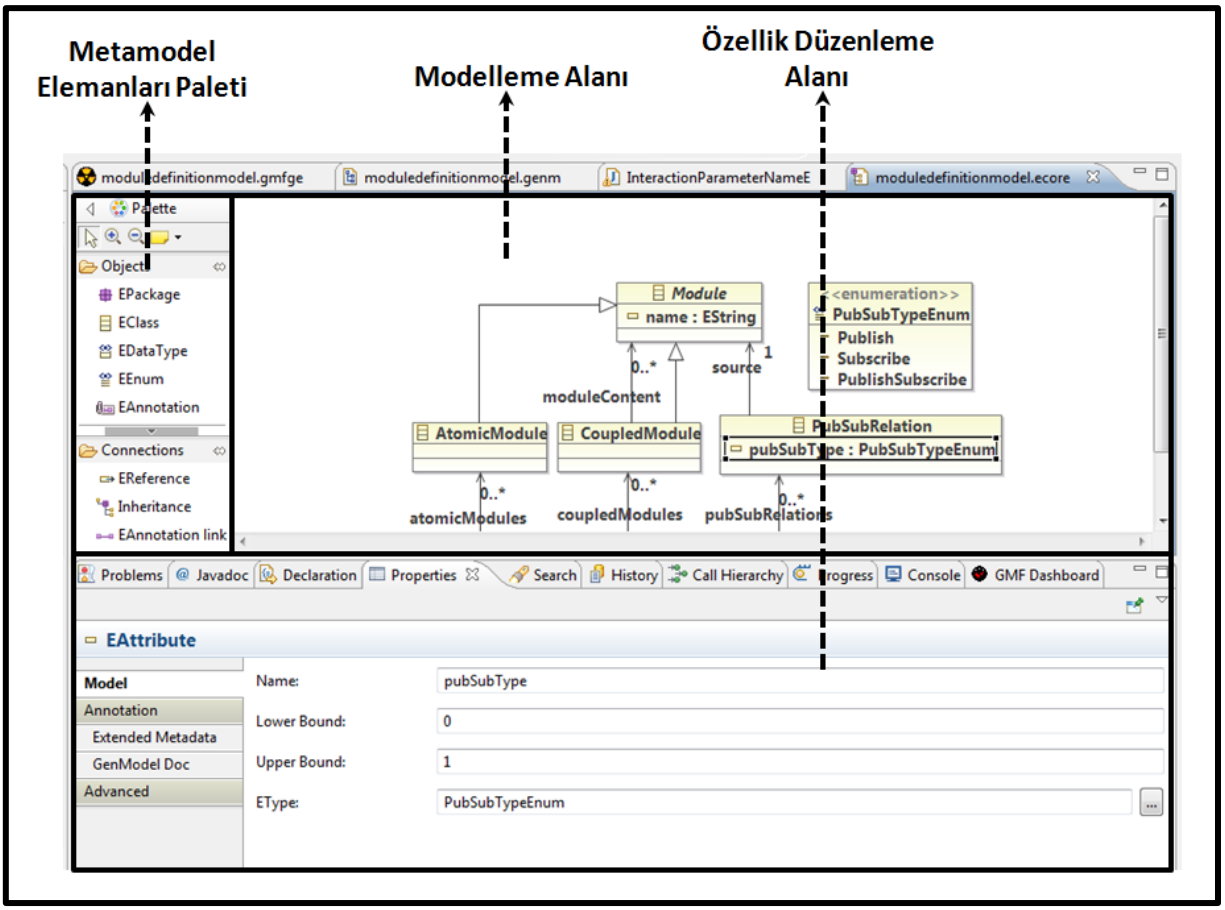
1. Eclipse platformu menüsünde *"File → New → Other → New GMF Project"* seçimi yapılır ve "SimulationDataExchangeModel" adında yeni bir proje oluşturulur. GMF eklentisi, Şekil 6-15'te gösterilen GMF Kontrol Paneli *(GMF Dashboard)* görünümünü açar. Bu şekil, aslında GMF ile modelleme aracı geliştirme adımlarını ve adımlar arasındaki ilişkileri anlatması açısından faydalıdır.



Şekil 6-15 Eclipse GMF Kontrol Paneli

2. Şekil 6-15'te gösterilen GMF Kontrol Panelinde *"Domain Model"* kesimindeki *"Create"* bağlantısı tıklanarak yeni bir metamodel oluşturulur. Metamodele "SimulationDataExchangeModel.ecore" adı verilir.

   Oluşturulan metamodel dosyasına sağ tıklanarak *"Initialize Ecore Diagram File"* seçimi yapılır ve metamodel için boş bir sınıf çizeneği oluşturulur. Sonraki adımda metamodel elemanları sınıf çizeneğine eklenir, özellikleri ve aralarındaki ilişkiler tanımlanır. Şekil 6-16'da Benzetim Veri Değişim Metamodelinin tanımlanması sürecinden bir ekran görüntüsü verilmiştir. Bu şekilde sol tarafta paket (*EPackage*), sınıf (*EClass*), referans (*EReference*) gibi metamodel oluşturma yapıtaşlarını içeren eleman paleti bulunmaktadır. Sağ üst tarafta ise, model oluşturma alanı gösterilmiştir. Sol taraftaki palette bulunan elemanlar sürükle-bırak yöntemi ile model oluşturma alanına taşınarak metamodel tanımlaması yapılır. Metamodel elemanlarının özellikleri şekilde sağ alt tarafta gösterilen özellikler (*Properties*) paneli kullanılarak güncellenebilir.



Şekil 6-16 Eclipse Grafiksel Metamodel Düzenleme Aracı

3. Bu adımda, önceki adımda oluşturulan metamodeldeki elemanların görselleştirme özellikleri Emfatic dili kullanılarak tanımlanacaktır. Öncelikle ecore biçiminde olan metamodeli Emfatic biçimine çevirmek gerekmektedir. Bunun için bir önceki adımda oluşturulan metamodel dosyası "SimulationDataExchangeModel.ecore" sağ tıklanarak *"Generate Emfatic Source"* seçimi yapılır.

   Açılan özel metin düzenleyicide, metamodel elemanlarının Emfatic dilindeki karşılıkları Şekil 6-17'deki gibi gösterilecektir. Metamodel elemanlarının görselleştirme biçimleri '@gmf' ile başlayan ek bilgi satırlarıyla belirlenecektir. Örneğin Şekil 6-17'de *"ObjectModelElement"* sınıfının olgularının modelleme ortamında etiketleri *"name"* nitelikleri olacak şekilde görselleştirileceği "@gmf.node(label="name")" ek bilgisiyle belirtilmiştir. Benzer şekilde, nesne sınıfları arasındaki kalıtım ilişkilerinin özellikleri de

“@gmf.link(target.decoration="closedarrow", style="solid")” ek bilgisi ile tanımlanmıştır.

```
1  @namespace(uri="objectmodel", prefix="objectmodel")
2  package objectmodel;
3
4  @gmf.diagram(foo="bar")
5  class ObjectModel {
6    val ObjectClass[*] objects;
7    val Interaction[*] interactions;
8    val DataType[*] dataTypes;
9  }
10
11 @gmf.node(label="name")
12 abstract class ObjectModelElement {
13   attr String name;
14 }
15
16 class ObjectClass extends ObjectModelElement {
17
18   @gmf.compartment(foo="bar")
19   val Attribute[*] attributes;
20
21   @gmf.link(target.decoration="closedarrow", style="solid")
22   ref ObjectClass objectInheritance;
23   attr SharingTypeEnum sharing;
24   attr String semantics;
25 }
26
```

Şekil 6-17 Emfatic Ortamında Metamodelin Düzenlenmesi

4. Bu adımda, ikinci adımda oluşturulan ve bir önceki adımda güncellenen metamodelden EMF üretici modeli (*EMF Generator Model*) oluşturulacaktır. Şekil 6-15’te gösterilen GMF Kontrol Panelinde *“Domain Model”* ile *“Domain Gen Model”* arasındaki türetme aracı “Derive” seçilerek üretici model oluşturma sihirbazı açılır. Açılan sihirbazdaki yönlendirmeler izlenerek üretici modelin oluşturulması tamamlanır. *“GMF Dashboard”* kullanmak yerine metamodel dosyası “SimulationDataExchangeModel.ecore” sağ tıklanarak *“New → Other → EMF Generator Model”* seçimi yapılarak da üretici model oluşturma sihirbazının açılması mümkündür.

5. Şekil 6-15’te gösterilen GMF Kontrol Panelinde, GMF Grafiksel Tanımlama Modeli (*Graphical Def Model - .gmfgraph*), Araç Tanımlama Modeli

(*Tooling Def Model - .gmftool*) ve Eşleme Modeli *(Mapping Model - .gmfmap)* üç adımda üretilmektedir. Bu çalışma kapsamında 2. adımda Emfatic ortamında yapılan görselleştirme tanımlamaları standart GMF yaklaşımında bu üç konfigürasyon modelinin içeriği oluşturulurken yapılmaktadır. Daha önce de bahsedildiği gibi, bu tez kapsamında EuGENia aracı kullanılarak GMF konfigürasyon modelleri otomatik olarak üretilmiştir. Bu adımda, metamodelden GMF konfigürasyon dosyaları oluşturulacaktır. "SimulationDataExchangeModel.ecore" metamodel dosyasına sağ tıklanarak "Eugeinia → Generate GMF tool, graph, and map models" seçimi yapılır ve ".gmfgraph", ".gmftool" ve ".gmfmap" uzantılı üç konfigürasyon modeli otomatik olarak üretilir. Bu modeller GMF'in kontrolü dışında üretildiğinden Şekil 6-15'te gösterilen GMF kontrol panelinde gösterilmezler. Bu araçları GMF Kontrol Panelindeki model kutularında gösterilen *"Select"* seçeneği ile seçerek kontrol paneline eklenebilirler.

6. Şekil 6-15'te gösterilen GMF Kontrol Panelinde eşleme modeli *"Mapping Model"* ile çizenek üretici *"Diagram Editor Gen Model"* model kutuları arasındaki dönüştürüm *(transform)* bağlantısı tıklanarak GMF Çizenek Düzenleyicisi Üretici Model üretilir. Bu işlem GMF Kontrol paneli kullanmak yerine *".gmfmap"* uzantılı eşleme modeline sağ tıklayıp *"Create Generator Model"* seçimi yapılarak da gerçekleştirilebilir.

7. Bu adımda, bir önceki adımda otomatik üretilen Çizenek Düzenleyici Üretici modelinde el ile yapılması gereken değişiklikler yapılır. Bu değişikliklerden en temeli, üretilecek modelleme aracının ilişkili olduğu diğer modelleme araçlarının tanımlanmasıdır. Böylece geliştirilen araç ortamında farklı modelleme araçlarıyla oluşturulmuş modeller arasında sürükle-bırak yöntemiyle model elemanı referansı eklemek mümkün olacaktır. Örneğin, Benzetim Veri Değişim Modeli tasarım aracıyla üretilen elemanlar Benzetim Modülleri ve Yayımlama/Üye Olma İlişkileri modelleme aracında kullanılacağından, bu iki model arasındaki ilişki tanımlanmalıdır. Çizenek Düzenleyici Üretici modeli *(.gmfgen)* çift tıklanarak model düzenleyicide açılır. Model ağacı açılarak *"Gen Editor Generator ... diagram → Gen Diagram …EditPart"* seçimi yapılır. Özellikler penceresinden, *"Diagram"*

özellikleri kesimindeki *"Contains Shortcuts To"* ve *"Shortcuts Provided For"* alanları ilgili modelleme aracının bağımlı olduğu ve ilgili modelleme aracına bağımlı olan modellere göre doldurulur.

8. Modelleme araçlarının tanımlanmasından sonraki bu adımda, modelleme araçlarının kodlarının otomatik olarak üretilmesine başlanacaktır. EMF üretici modeli *(.genmodel)* iki kere tıklanarak model düzenleyicide açılır. Düzenleyicide açılan model ağacındaki kök elemana sağ tıklanarak "Generate All" seçimi yapılarak EMF seviyesi kod üretimi tamamlanır.

9. GMF Kontrol Panelinden "Diagram Editor Gen Model" kutusundaki "Generate diagram editor" seçimi veya Çizenek Düzenleyici Üretici modeli *(.gmfgen)* sağ tıklanıp *"Generate Diagram Code"* seçimi yapılarak grafiksel model düzenleyici kodu üretilir.

10. Bu adımda, önceki iki adımda üretilen kaynak kodlar üzerinde ihtiyaç durumunda güncelleme yapılacaktır. Örneğin bir model elemanının isminin farklı görünmesi istendiği durumda bu adım işletilebilir. Eclipse Modelleme Platformu, ürettiği kodların otomatik üretim olduğunu belirtmek için yöntemleri "@generated" ifadesi ile işaretler. Bu yöntemler, bir sonraki otomatik kod üretiminde güncellenecektir. Uyarlama yapılan herhangi bir otomatik üretilmiş kod kesiminin bir sonraki modelden kod üretiminde ezilmemesi için ilgili elemanın işaretini "@generated not" olarak değiştirmek gerekmektedir.

    Örneğin Şekil 6-18'de Benzetim Veri Değişim Modeli Etkileşim Sınıflarının parametrelerinin modelleme aracındaki görünümünün nasıl değiştirildiği gösterilmiştir. Dikkat edilirse, yöntemin başındaki "@generated" ifadesi "@generated not" olarak güncellenmiştir. Yöntemin içeriği incelenirse, parametrelerin adı ile birlikte türlerinin de parametre adı ile türü ':' ile ayrılarak görünmesi için bir değişiklik yapılmıştır (ör: targetPos : Position3D şeklinde).

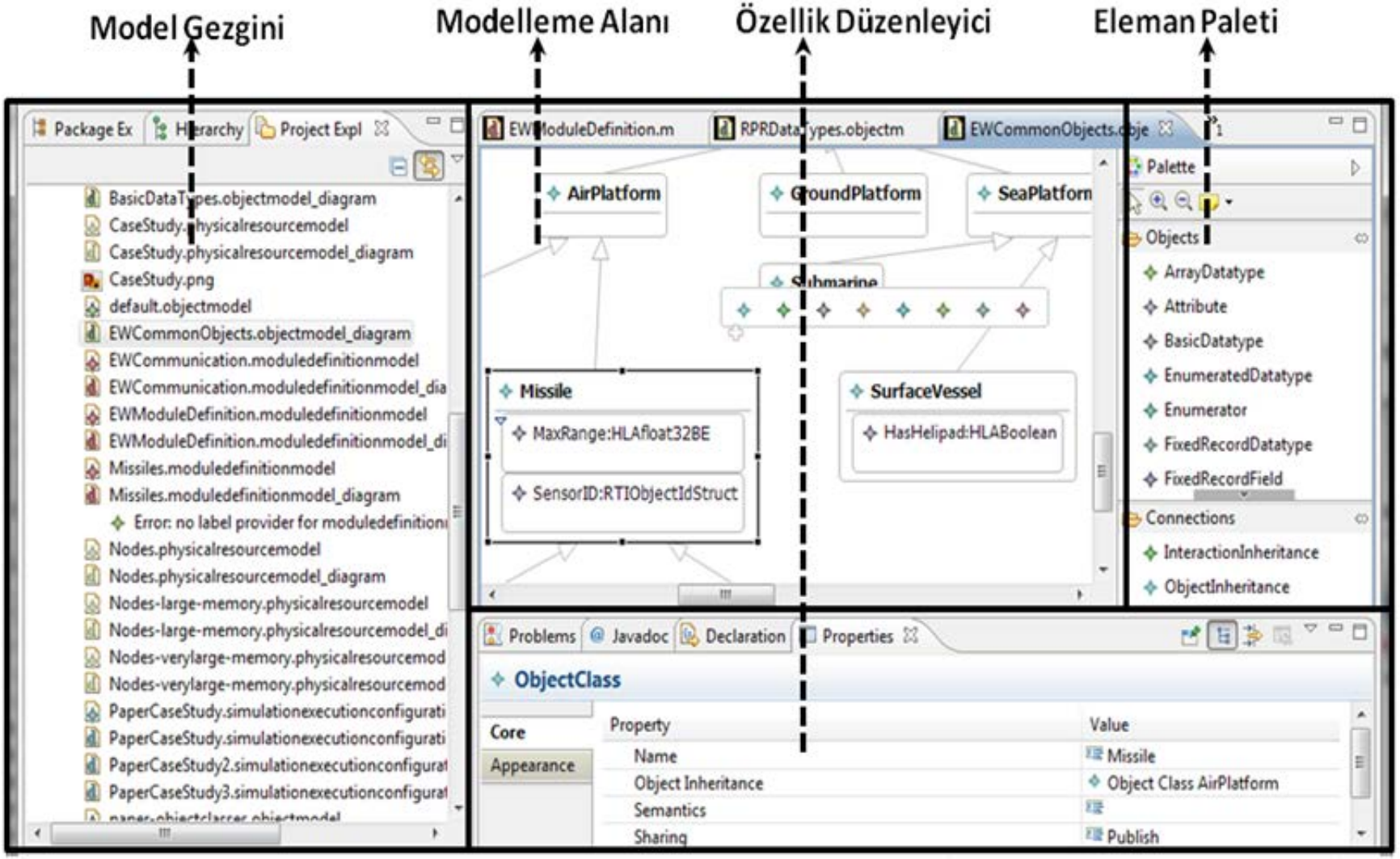
```
/**
 * @generated not
 */
protected String getLabelText() {
    String text = null;
    EObject parserElement = getParserElement();
    if (parserElement != null && getParser() != null) {
        text = getParser().getPrintString(new EObjectAdapter(parserElement), getParserOp
    }
    if (text == null || text.length() == 0) {
        text = defaultText;
    }

    if (parserElement != null && ((InteractionParameter) parserElement).getDataType() !=
        text += ":" + ((InteractionParameter) parserElement).getDataType().getName();
    }

    return text;
}
```

Şekil 6-18 Eclipse Modelleme Ortamının Otomatik Ürettiği Koda Müdahale Edilmesi

S-IDE ortamı, yukarıda adımları açıklanan yöntem ile geliştirilmiş çeşitli araçlardan oluşur. Bu araçların genel görünümü ortaktır (Şekil 6-19). Soldaki panel, mevcut modelleri ve bu modellerin elemanlarını gösteren Model Gezginini içermektedir. Ortadaki modelleme alanı, benzetim tasarımı için temel tasarım alanını oluşturur. Sağ taraftaki eleman paleti tasarım modellerinin oluşturulması için kullanılan nesneleri ve bağlantıları sağlar. Bu paletindeki elemanlar tasarım düzenleme alanına sürükle-bırak yöntemi ile eklenebilir. Alt taraftaki özellikler paneli, model düzenleme alanından veya model gezgininden seçilen tasarım elemanlarının özelliklerinin gösterilmesi ve güncellenmesini sağlar.



Şekil 6-19 *S-IDE* Araçlarının Genel Görünümü

7 GELİŞTİRİLEN YAKLAŞIMIN ELEKTRONİK HARP BENZETİMİ ÖRNEK DURUM ÇALIŞMASINA UYGULANMASI

Önceki kesimde, *S-IDE* Geliştirme Ortamının tasarımı ve gerçekleştirimi ele alınmıştır. Bu kesimde ise, *S-IDE* Geliştirme Ortamı kullanılarak tez çalışmasının örnek durum çalışmalarından ilki olan Elektronik Harp Benzetimi modellenecek ve uygun yerleştirme alternatifleri türetilecektir.

7.1 *S-IDE* Aracı Kullanılarak Elektronik Harp Benzetimi İçin Benzetim Ortamının Modellenmesi

Tezin önceki kesimlerinde de belirtildiği gibi, benzetim ortamı modeli, Benzetim Veri Değişim Modeli Tasarım Aracı, Benzetim Modül ve Yayımlama/Üye Olma İlişkileri Tasarım Aracı, Fiziksel Kaynak Tasarım Aracı ve Benzetim Çalıştırma Konfigürasyonu Tasarım Araçları kullanılarak oluşturulur.

Bu kesimde geliştirilen yaklaşımın akışına uygun olarak Elektronik Harp benzetim ortamı modelinin adım adım tasarlanması ele alınacaktır.

7.1.1 Elektronik Harp benzetimi veri değişim modelinin tasarlanması

Benzetim Veri Değişim Metamodelinin açıklandığı 5.1 kesiminde belirtildiği gibi, Benzetim Veri Değişim Metamodeli HLA Nesne Modeli Şablonunu (Object Model Template - OMT) [IEEE 2010c] genişletir. Bu metamodelin olguları olan Benzetim Veri Değişim Modelleri de doğal olarak büyük ölçüde HLA OMT belirtiminin olguları olan Federasyon Nesne Modeli (Federation Object Model – FOM) ve Benzetim Nesne Modeli (Simulation Object Model – SOM) [IEEE 2010c] tanımlamalarına benzeyecektir. Bu tez çalışması kapsamındaki örnek durum çalışmasının Benzetim Veri Değişim Modeli tasarlanırken *Real-time Platform Reference Federation Object Model (RPR-FOM)* [SISO 2001a;2001b] temel alınarak genişletilmiştir.

Benzetim Veri Değişim Modeli Tasarlanırken ilk önce HLA OMT standardında tanımlı olan temel veri türleri (*basic datatypes*) Şekil 7-1'de gösterildiği gibi tanımlanmıştır. Şekilde de gösterildiği gibi, integer, float, long, string, byte gibi temel veri türleri için gerekli durumlarda 16, 32 ve 64 bit tanımlamaları, mimari anlamda *Big Endian (BE) ve Little Endian (LE)* tanımlamaları HLA OMT standardına uygun olarak yapılmıştır.

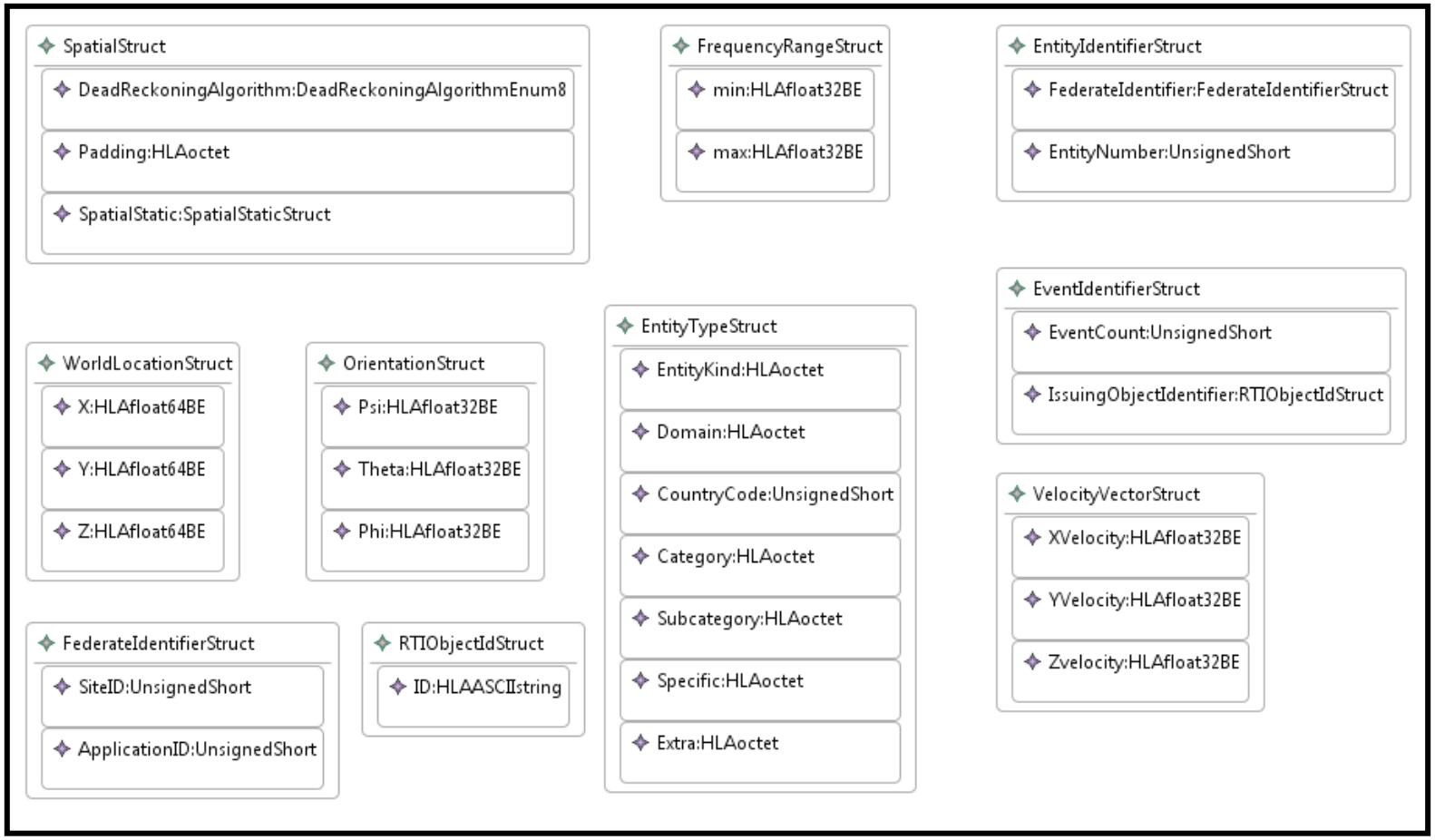
Şekil 7-1 HLA OMT Temel Veri Türlerinin *S-IDE* Aracıyla Tanımlanması

Temel veri türleri tanımlandıktan sonra, Benzetim Veri Değişim Modelinin türetileceği RPR FOM elemanları tanımlanmalıdır. İzleyen kesimde *S-IDE* geliştirme ortamında modellenen RPR FOM elemanlarının açıklamaları GRIM olarak bilinen *"Guidance, Rationale, and Interoperability Modalities for the Real-time Platform Reference Federation Object Model (RPR FOM)"* standardında mevcuttur [SISO 2001b].

Şekil 7-2, Şekil 7-3 ve Şekil 7-4'te RPR-FOM sabitlenmiş kayıtlarının (*fixed record*) bir kısmının *S-IDE* geliştirme ortamında tanımlanması gösterilmiştir. Şekillerdeki tanımlamalara dikkat edilirse veri türü olarak *HLAfloat32BE* gibi temel veri türlerinin yanı sıra, diğer sabitlenmiş kayıtlar veya numaralandırılmış veri türleri *(enumeration)* de kullanılmıştır.

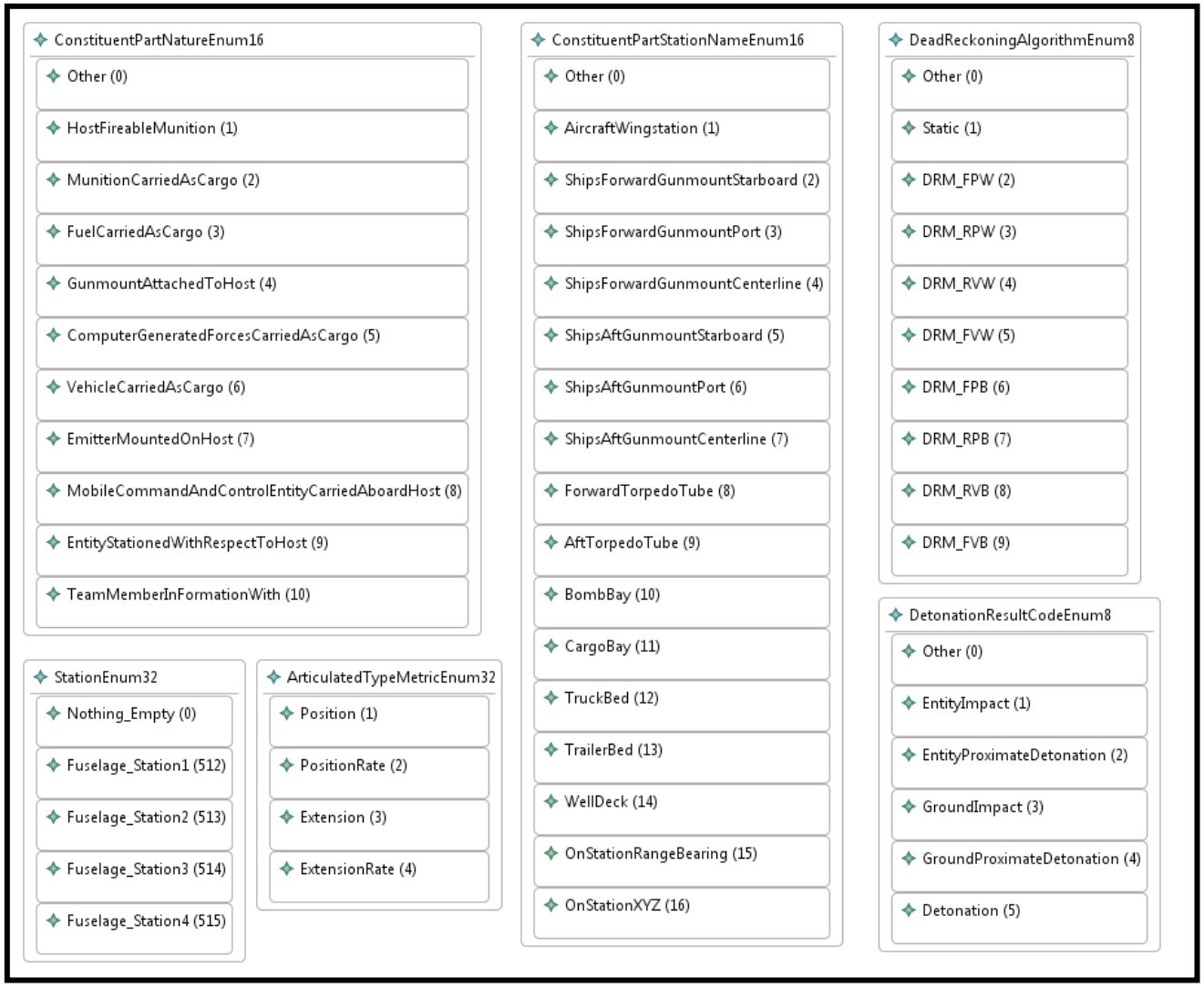
Benzetim Veri Değişim Modeli tanımlanırken kullanılan RPR-FOM numaralandırılmış veri türleri de örnek durum çalışması için benzetim ortamı modellenirken *S-IDE* geliştirme ortamında Şekil 7-5 ve Şekil 7-6'da gösterildiği gibi tasarlanmıştır.

Şekil 7-2 RPR-FOM Sabitlenmiş Kayıt *(Fixed Record)* Veri Türlerinin *S-IDE* Geliştirme Ortamında Modellenmesi – Kesim #1

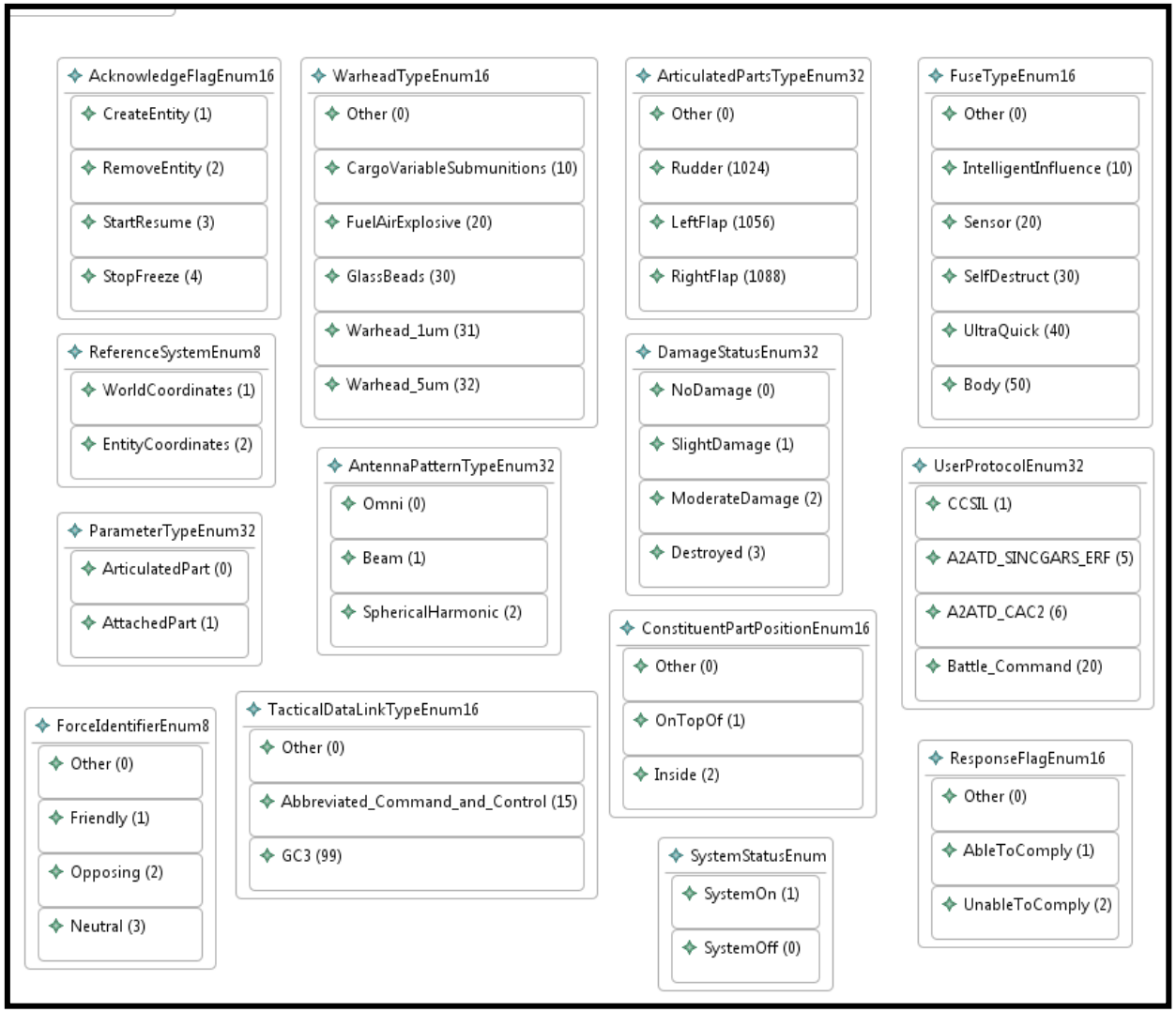


Şekil 7-3 RPR-FOM Sabitlenmiş Kayıt *(Fixed Record)* Veri Türlerinin *S-IDE* Geliştirme Ortamında Modellenmesi – Kesim #2

Şekil 7-4 RPR-FOM Sabitlenmiş Kayıt (Fixed Record) Veri Türlerinin S-IDE Geliştirme Ortamında Modellenmesi – Kesim #3



Şekil 7-5 RPR-FOM Numaralandırılmış Veri Türlerinin *(Enumerator) S-IDE* Geliştirme Ortamında Modellenmesi – Kesim #1



Şekil 7-6 RPR-FOM Numaralandırılmış Veri Türlerinin *(Enumerator) S-IDE* Geliştirme Ortamında Modellenmesi – Kesim #2

Gerekli veri türleri tanımlandıktan sonra, bu veri türlerini kullanarak Benzetim Veri Modeli modelinin nesne ve etkileşim sınıfları tanımlanabilir. Şekil 7-7'de, örnek durum çalışması için tanımlanan temel nesne sınıfları gösterilmiştir.

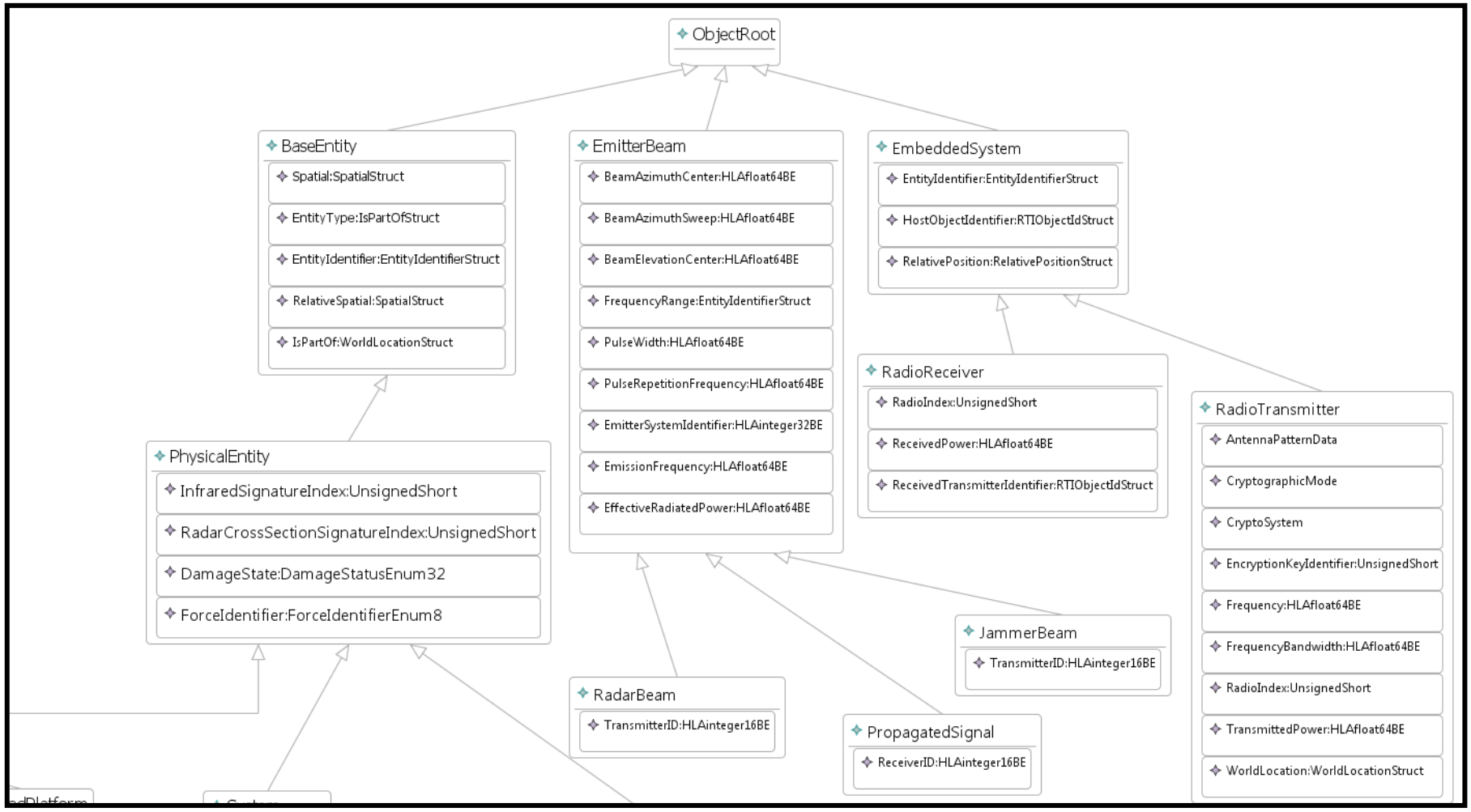
ObjectRoot nesne sınıfı, nesne modelindeki tüm diğer nesne sınıflarının doğrudan veya dolaylı olarak ata sınıfıdır. *ObjectRoot* nesne sınıfının kök sınıfı olduğu hiyerarşi dışında herhangi bir nesne sınıfı tanımlanmamalıdır. Şekil 7-7'de *ObjectRoot* nesne sınıfının alt sınıfları olarak tanımlanmış olan *BaseEntity, EmitterBeam, EmbeddedSystem, PhysicalEntity, RadarBeam, JammerBeam, RadioReceiver* ve *RadioTransmitter* sınıfları *RPR-FOM* nesne modelinden gelmektedir [SISO 2001a] ve açıklamaları RPR-FOM GRIM standardında verilmiştir [SISO 2001b].

Nesne sınıflarının nitelikleri gösterilirken, {Nitelik Adı} **:** {Nitelik Türü} biçimi kullanılmıştır. Örneğin Şekil 7-7'de bir unsurun konum bilgisini tutan "*BaseEntity"* sınıfının veri türü "*SpatialStruct*" olan *"Spatial"* niteliği *"Spatial**:**SpatialStruct"* şeklinde gösterilmiştir.

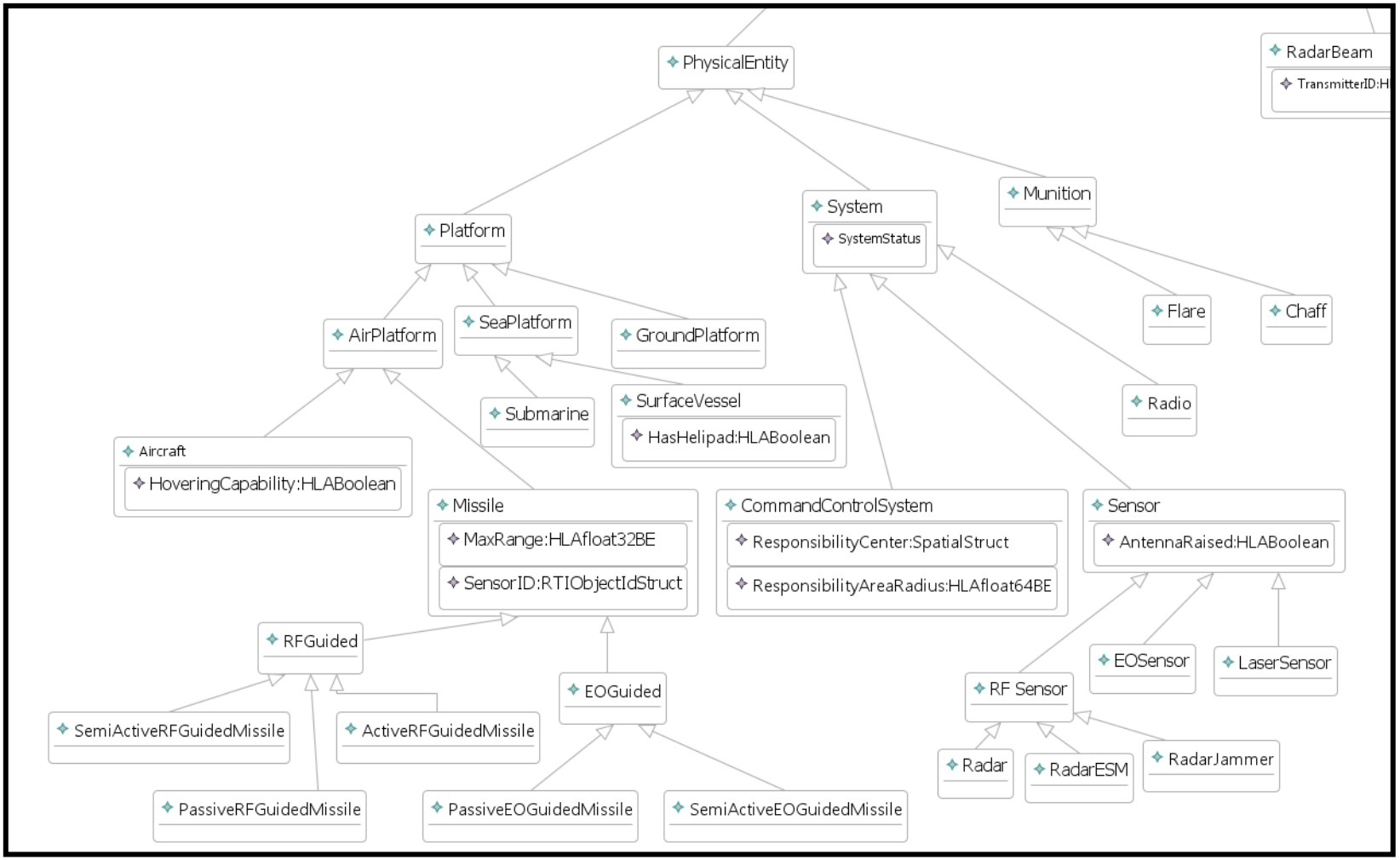
Temel *RPR-FOM* nesne sınıfı tanımlamaları yapıldıktan sonra, örnek durum çalışmasına özel nesne sınıfı tanımlamaları *RPR-FOM* elemanları genişletilerek tanımlanmıştır. Örneğin Şekil 7-7'de gösterilen *EmitterBeam RPR-FOM* sınıfından türeyen *PropogatedSignal* nesne sınıfı*,* örnek durum çalışması olan Elektronik Harp Benzetimindeki RF Sinyal yayılımlarını temsil eder.

Şekil 7-8'de Elektronik Harp Benzetimi örnek durum çalışmasındaki platform ve sistemlerin nesne sınıfı tanımları gösterilmiştir. Tüm platform ve sistemlerin kök sınıfı Şekil 7-7'de de gösterilen *RPR-FOM PhyisicalEntity* sınıfıdır.

Şekil 7-8'de her platform türü için alt nesne sınıfları (örneğin deniz platformları için *SurfaceVessel,* hava platformları için *AirPlatform* gibi), değişik türde füzeler (örneğin Aktif RF Güdümlü füzeler için *ActiveRFGuidedMissile*, Pasif EO güdümlü füzeler için *PassiveEOGuidedMissile* gibi) ve sistem tanımları (örneğin *Radar*, *RadarJammer*, *CommunicationDevice*, vs.)yapılmıştır.



Şekil 7-7 EH Benzetimi Örnek Durum Çalışması İçin Temel Nesne Sınıfı Hiyerarşisinin *S-IDE* Geliştirme Ortamında Modellenmesi



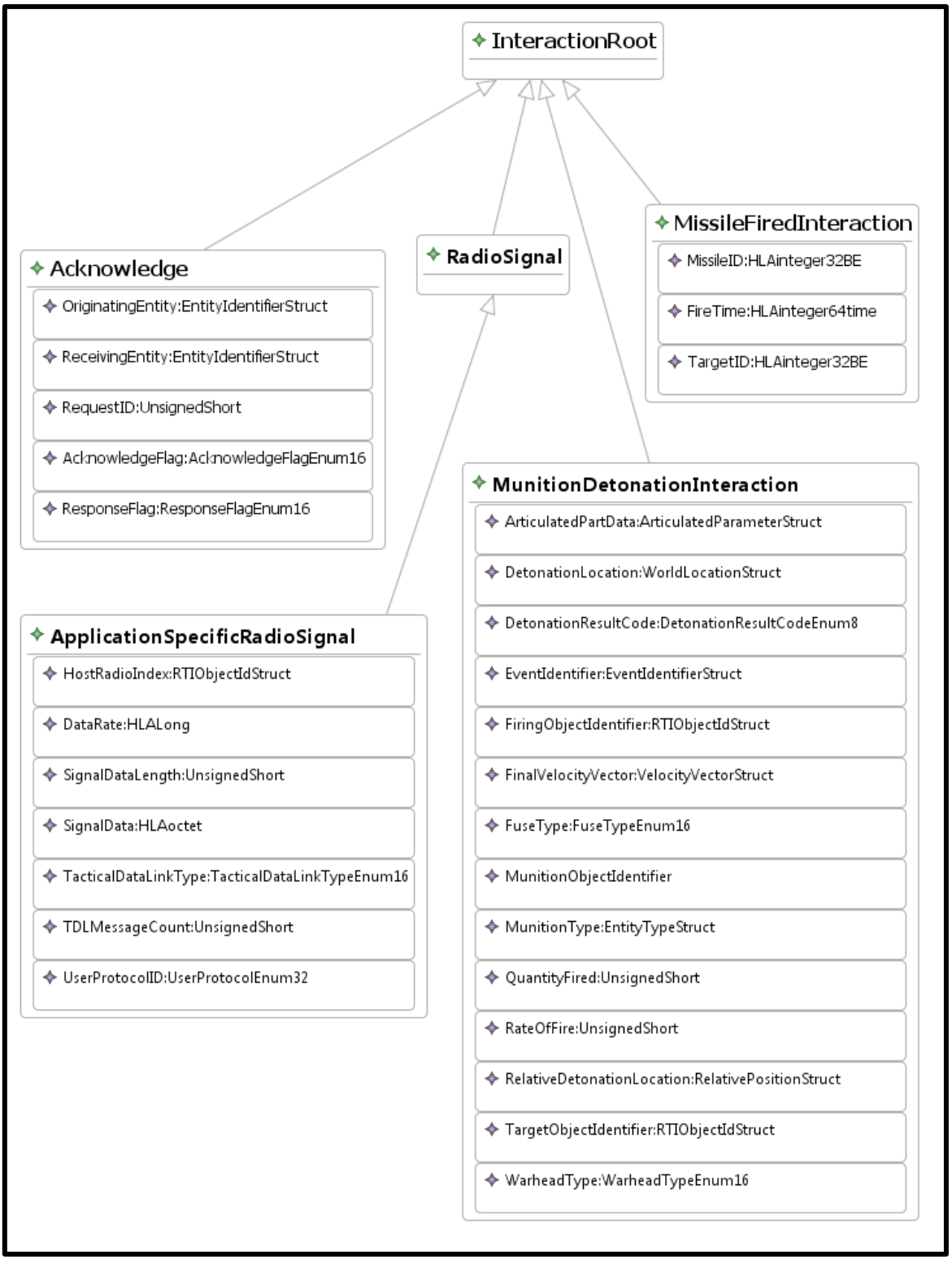
Şekil 7-8 EH Benzetimi Örnek Durum Çalışması İçin Platform ve Sistem Nesne Sınıfı Hiyerarşisinin *S-IDE* Geliştirme Ortamında Modellenmesi

Örnek durum çalışması için Benzetim Veri Değişim Modelinin tanımlanması kapsamında, nesne sınıflarına ek olarak Şekil 7-9'da bir bölümü gösterilen çeşitli etkileşim sınıfları da tanımlanmıştır. Aynen nesne sınıfı tanımlarında olduğu gibi, etkileşim sınıfı tanımlarında da RPR-FOM referans olarak alınmıştır.

Şekil 7-9'da gösterildiği gibi, nesne sınıflarının kök sınıfı olan *ObjectRoot* sınıfına benzer şekilde, etkileşim sınıflarının da türediği ortak bir kök sınıf vardır. Bu kök sınıf şekilde *InteractionRoot* olarak gösterilmiştir. Şekilde gösterilen *Acknowledge ve RadioSignal* etkileşim sınıfları *RPR-FOM* nesne modelinden gelmektedir [SISO 2001a] ve açıklamaları RPR-FOM GRIM standardında verilmiştir [SISO 2001b].

Örnek durum çalışması kapsamında tanımlanan etkileşim sınıflarına örnek olarak füze ateşlenmesi durumunda yayımlanan *MissileFiredInteraction* ile herhangi bir mühimmatın patlaması durumunda yayımlanan *MunitionDetonation* etkileşim sınıfı Şekil 7-9'da gösterilmiştir. Bu etkileşim sınıflarının her ikisi de doğrudan *InteractionRoot* kök etkileşim sınıfından türemektedir.

Nesne sınıflarının niteliklerine benzer şekilde, etkileşim sınıflarının da parametreleri {Parametre Adı} **:** {Parametre Türü} biçiminde gösterilmektedir. Örneğin, mühimmat patlamasını temsil eden *MunitionDetonationInteraction* etkileşim sınıfının patlama konumu gösteren *WordLocationStruct* türündeki parametresi olan *DetonationLocation* Şekil 7-9'da bu biçime uygun olarak *DetonationLocation**:**WordLocationStruct* şeklinde gösterilmiştir.



Şekil 7-9 EH Benzetimi Örnek Durum Çalışması İçin Etkileşim Sınıfı Hiyerarşisinin *S-IDE* Geliştirme Ortamında Modellenmesi

7.1.2 Elektronik Harp benzetiminin modüllerinin ve yayımlama/üye olma ilişkileri modelinin tasarlanması

Benzetim Veri Değişim Modeli tasarlandıktan sonra, Benzetim Modülleri ve Yayımlama/Üye Olma İlişkileri Tasarım aracını kullanarak öncelikle örnek durum çalışmasındaki benzetim modülleri ve bu modüllerin içerme ilişkileri tanımlanmıştır. Benzetim modüllerinin tanımlamasının ardından, benzetim modülleri arasındaki yayımlama üye olma ilişkileri Benzetim Veri Değişim Modeli elemanları kullanılarak tanımlanmıştır.

3. Kesimde verilen örnek durum çalışması tanımına göre Elektronik Harp Benzetim sistemi için benzetim modülleri ve yayımlama/üye olma ilişkileri Şekil 7-10, Şekil 7-11, Şekil 7-12 ve Şekil 7-13'te gösterildiği gibi modellenmiştir.

Benzetim modüllerinin yayımlama/üye olma ilişkilerini tanımlamak için bir önceki adımda tasarlanan Benzetim Veri Değişim Modelinin elemanları kullanılmaktadır. Benzetim Veri Değişim Modellerinden Şekil 7-10, Şekil 7-11, Şekil 7-12 ve Şekil 7-13'ye sürükle bırak yöntemiyle eklenen bu elemanların kenarlarında referans okları (↗) olacak şekilde gösterilmiştir.

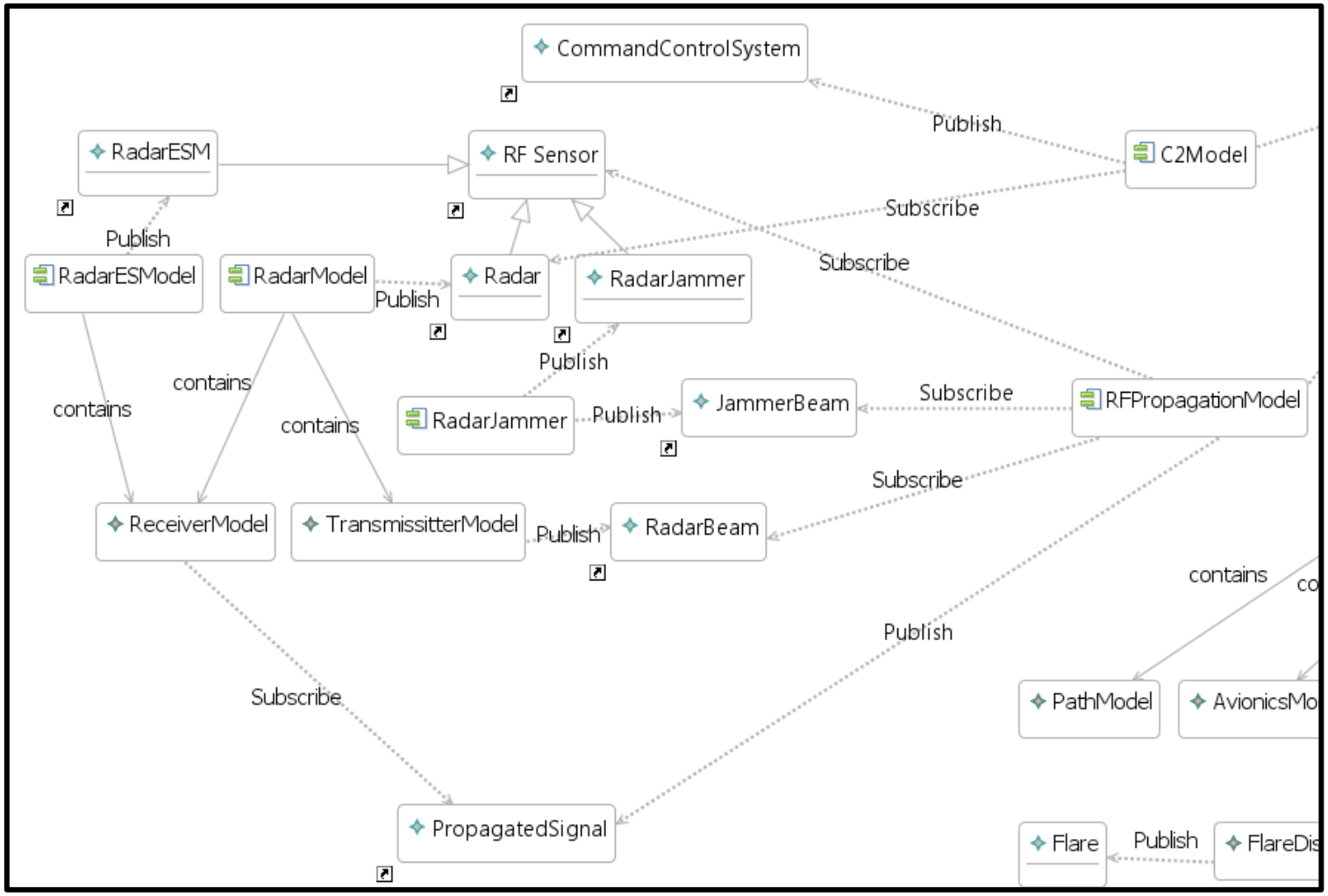
Şekil 7-10'da Elektronik Harp Benzetimindeki Radar ve Radar ile ilişkili sistemleri temsil eden benzetim modüllerinin tasarımı verilmiştir. Tasarımda bir radar sisteminin benzetimini sağlamaktan sorumlu modül *RadarModel* olarak gösterilmiştir. *RadarModel* benzetim modülü, radar alıcısını modelleyen *ReceiverModel* ve radar vericisini modelleyen *TransmitterModel* atomik benzetim modüllerini içeren bağlı *(coupled)* bir modüldür. *ReceiverModel* benzetim modülü, *PropagatedSignal* nesne sınıfına üye olur. *TransmitterModel* modülü ise *RadarBeam* nesne sınıfını yayımlar. *PropagationModel* benzetim modülü ise, radar alıcı ve verici modülleriyle etkileşimli olarak çalışmak için *RadarBeam* nesne sınıfına üye olur ve *PropagatedSignal* nesne sınıfını yayınlar. *PropagationModel* benzetim modülü aynı zamanda radar karıştırma etkilerini de modelleyebilmek için *RadarJammer* benzetim modülünün yayımladığı *JammerBeam* nesne sınıfına da üye olur. Şekil 7-10'da Radar ve Yayılım (*Propagation*) modüllerinin yanı sıra, Radar Elektronik Destek *(RadarESModel)*, Radar Karıştırma *(RadarJammer)* ve Komuta Kontrol Sistemi (C2Model) benzetim modülleri yayımlama/üye olma ilişkileriyle birlikte tanımlanmıştır.

Şekil 7-11'de Elektronik Harp Benzetimindeki Muhabere ve Muhabere ile ilişkili sistemleri temsil eden benzetim modüllerinin tasarımı verilmiştir. Bu şekilde Muhabere Cihazı, *CommunicationTransmitter* ve *CommunicationReceiver* atomik benzetim modüllerinden oluşan *CommunicationDeviceModel* adında bağlı (*coupled*) bir benzetim modülü olarak tanımlanmıştır. Şekilde ayrıca muhabere cihazlarının RF Yayılım benzetimini sağlayan *RFPropagationModel* benzetim modülü ile olan ilişkisi Şekil 7-10'da radar modeli için yapılan tanımlamaya benzer şekilde tanımlanmıştır.

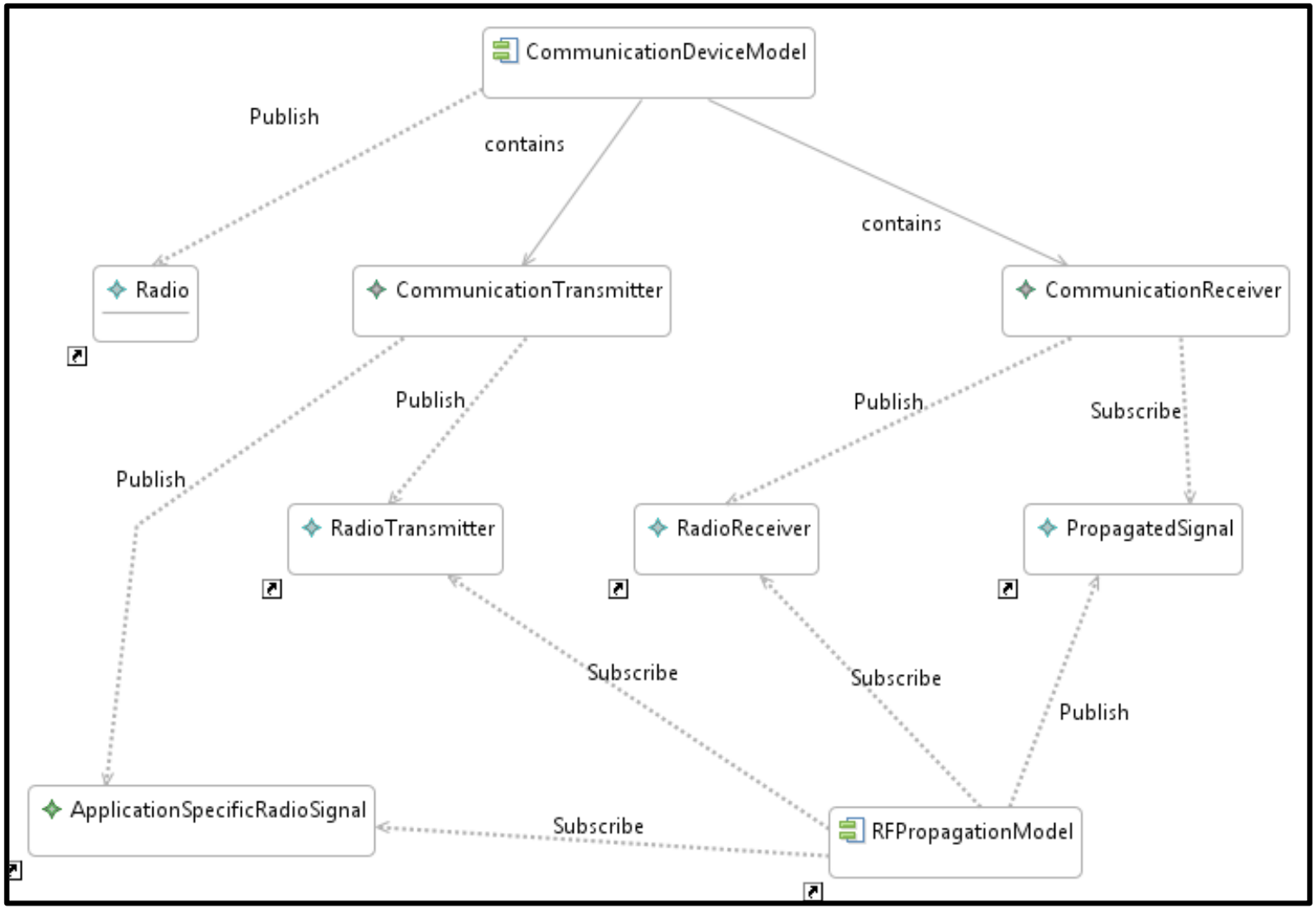
Şekil 7-12'de Elektronik Harp Benzetimindeki platformları temsil eden benzetim modüllerinin tasarımı verilmiştir. Şekilde temel olarak hava platformu *(AirplaneModel),* su üstü deniz platformu *(SurfaceVesselModel)*, denizaltı *(SubmarineModel)* benzetim modülleri ve yayımlama/üye olma ilişkileri gösterilmiştir. Her benzetim modülü, modellediği platform türünde bir nesne sınıfı yayımlar. Örneğin *AirplaneModel* benzetim modülü *AirPlatform* nesne sınıfını yayımlarken *TankModel* benzetim modülü *GroundPlatform* nesne sınıfını yayımlar.

Şekil 7-12'de verilen Elektronik Harp Benzetimi platform tasarımı kesimi ile Şekil 7-10 ve Şekil 7-11'de verilen sistem tasarımı kesimi arasındaki bağlantı *PropagationModel* benzetim modülü ile sağlanır. Daha önce sistem kesimindeki yayımlama/üye olma ilişkileri tanımlanan *PropagationModel* benzetim modülü, platform benzetim modüllerinin yayımladığı *AirPlatform* ve *GroundPlatform* gibi nesne sınıflarının ata sınıfı olan *Platform* sınıfına üye olarak platformlardan yansıyan sinyallerin Elektronik Harp sistemlerine olan etkisini modeller.

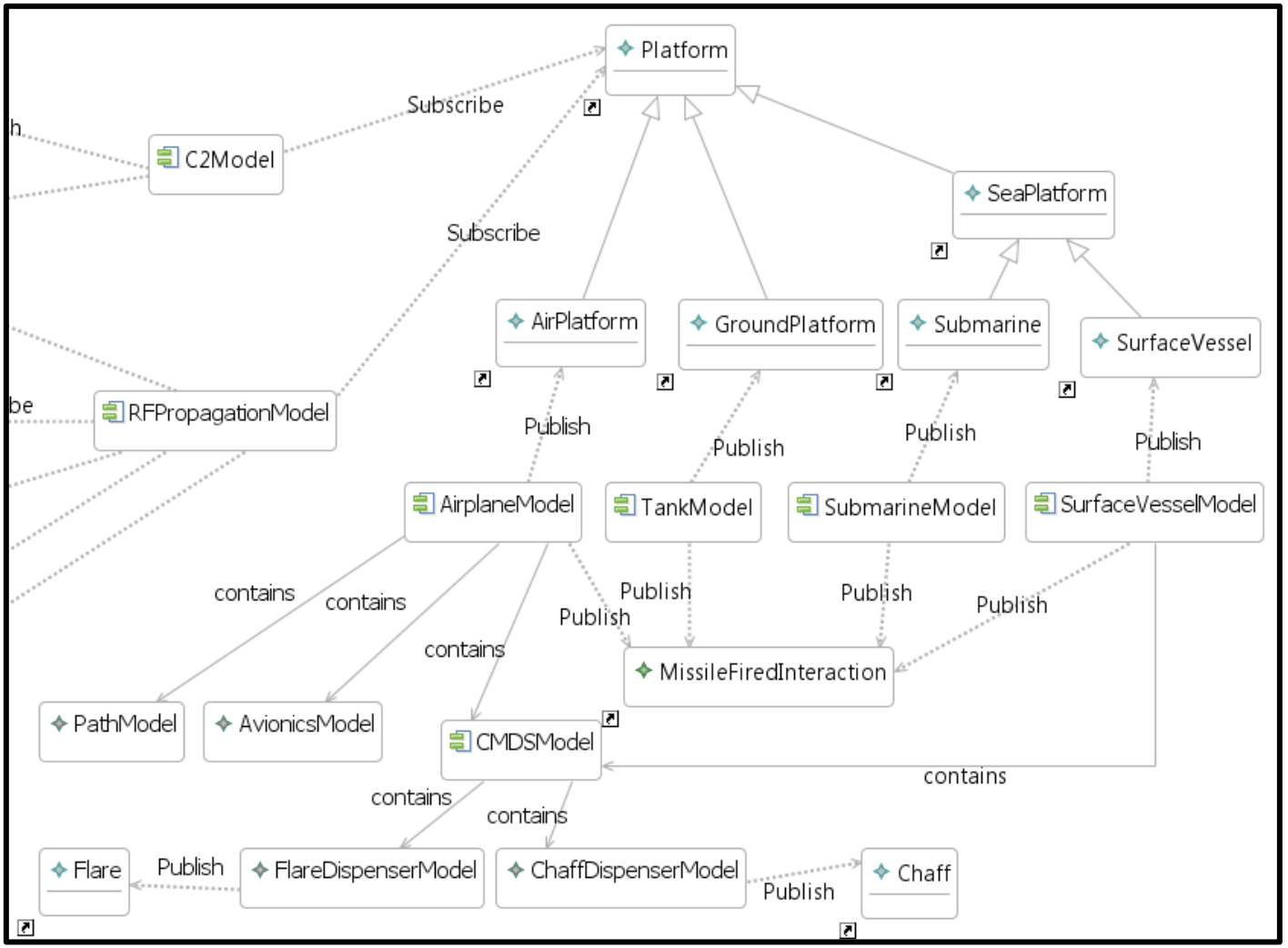
Şekil 7-12'de verilen tasarımda bağlı *(coupled)* modüller olan *AirplaneModel* ve *SurfaceVesselModel* benzetim modüllerinin, yine bağlı bir modül olan karşı korunma sistemi modeli *CMDSModel* benzetim modülünü içermektedirler. *CMDSModel* bağlı benzetim modülü, *FlareDispenserModel* ve *ChaffDispenserModel* atomik modüllerinden oluşur.



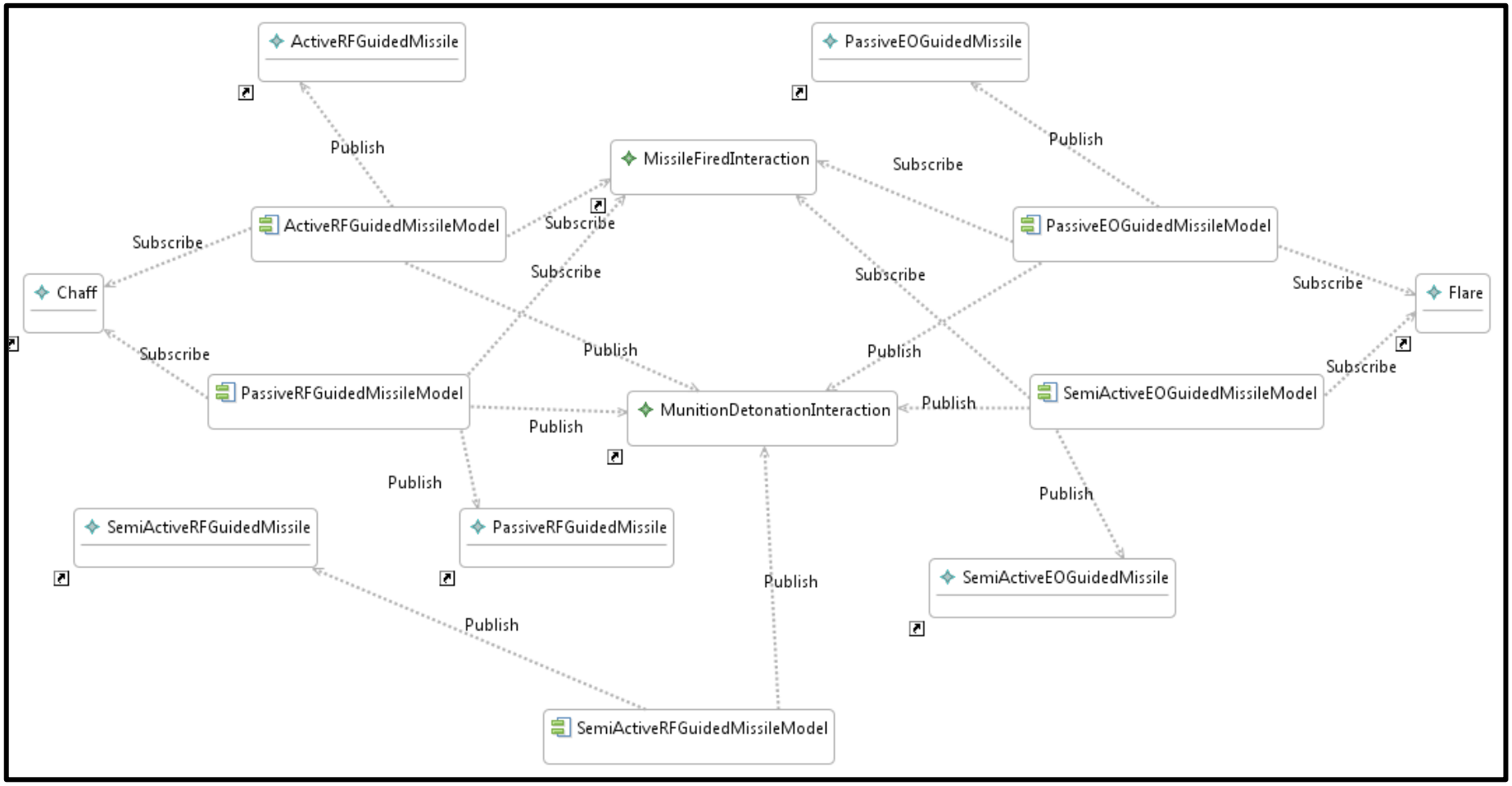
Şekil 7-10 EH Benzetimi Örnek Durum Çalışması için Benzetim Modüllerinin *S-IDE* Ortamında Modellenmesi – Kesim #1 Sistemler – Radarlar



Şekil 7-11 EH Benzetimi Örnek Durum Çalışması İçin Benzetim Modüllerinin *S-IDE* Ortamında Modellenmesi – Kesim #2 Sistemler – Muhabere



Şekil 7-12 EH Benzetimi Örnek Durum Çalışması İçin Benzetim Modüllerinin *S-IDE* Ortamında Modellenmesi – Kesim #3 Platformlar

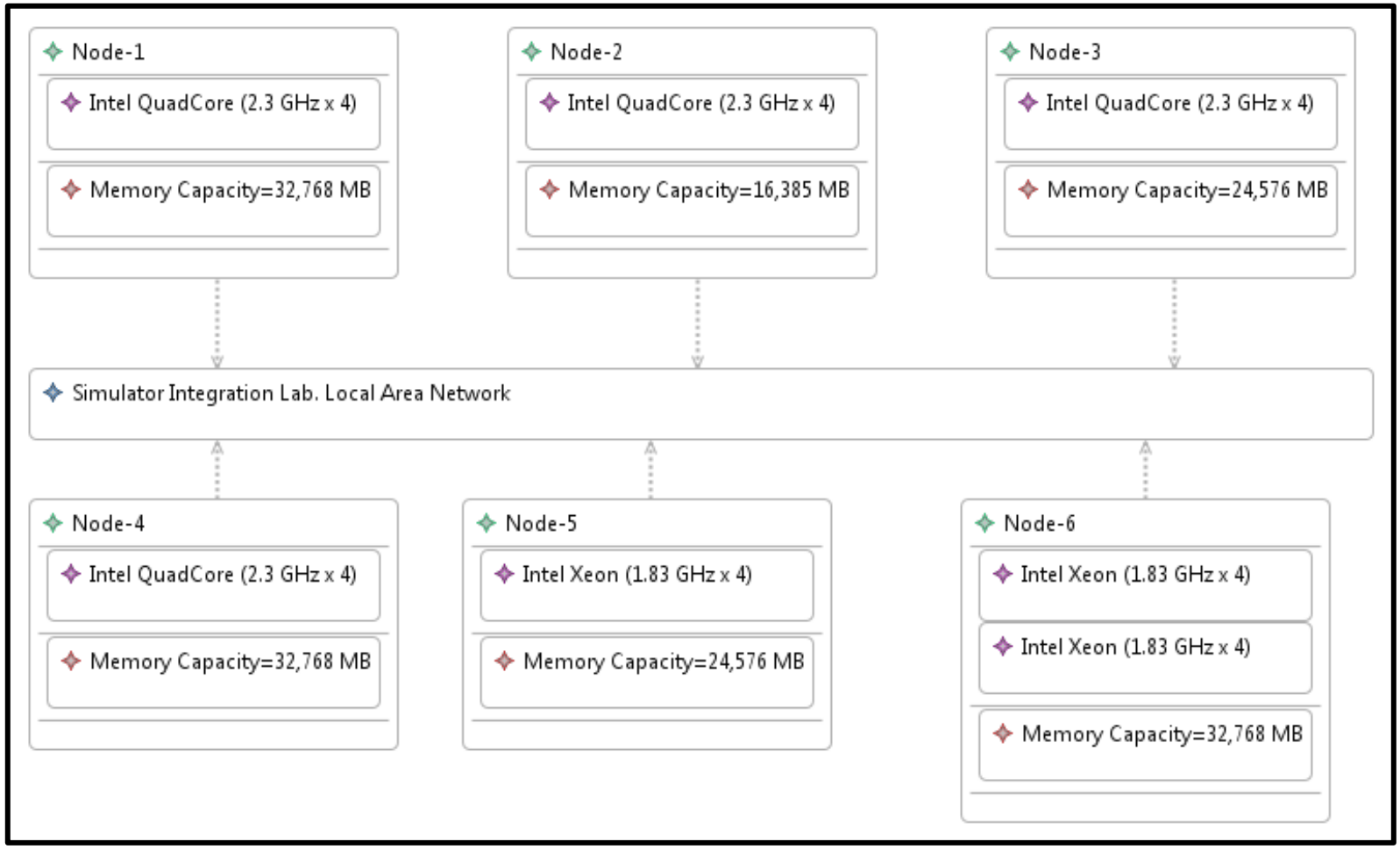


Şekil 7-13 EH Benzetimi Örnek Durum Çalışması İçin Benzetim Modüllerinin *S-IDE* Ortamında Modellenmesi – Kesim #4 Silah Sistemleri

Şekil 7-13'de Benzetim Modülleri ve Yayımlama/Üye Olma modelinin son kesimi olan Silah sistemlerinin tasarımı verilmiştir. Bu tasarımda silah sistemlerini modelleyen *ActiveRFGuidedMissileModel, SemiActiveRFGuidedMissileModel, PassiveRFGuidedMissileModel ve PasiveEOGuidedMissileModel* gibi benzetim modülleri ve yayımlama/üye olma ilişkileri modellenmiştir. Silah sistemi benzetim modülleri modelledikleri füzenin türüne göre füze nesne sınıfı olgularını yayımlarlar. Örneğin *ActiveRFGuidedMissileModel* benzetim modülü, *ActiveRFGuidedMissile* nesne sınıfını yayımlar. Silah sistemi benzetim modülleri, RF/EO olmalarına göre platformların CMDS benzetim modüllerinin (Şekil 7-12) yayımladığı *Chaff* ve *Flare* nesne sınıflarına üye olurlar. Silah sistemi benzetim modüllerinin hepsi füze ateşlenmesi durumunda yayımlanan *MissileFiredInteraction* etkileşim sınıfına ateşlenen füzenin modelledikleri füze olup olmadığını anlamak için üye olur. Benzer biçimde tüm silah sistemi benzetim modülleri patlama durumlarını ilgili diğer benzetim katılımcılarına bildirmek için *MunitionDetonationInteraction* etkileşim sınıfını yayımlarlar.

7.1.3 Elektronik Harp benzetimi için fiziksel kaynaklar modelinin tasarlanması

Fiziksel Kaynak Modeli Tasarlama aracı, *S-IDE* dışında farklı araçlarda kullanılabilecek nitelikte bir araç olduğu için diğer *S-IDE* araçlarından bağımsız olarak tasarlanıp geliştirilmiştir. Böylece gelecek çalışmalar kapsamında yaklaşımın farklı alanlara uygulanması kolaylaşacaktır. Geliştirilen Fiziksel Kaynak Modeli Tasarlama Aracı kullanılarak örnek durum çalışması için altı heterojen düğümden oluşan bir fiziksel kaynak modeli geliştirilmiştir. Geliştirilen fiziksel kaynak tasarımı Şekil 7-14'te verilmiştir. Bu örnek tasarımda düğümlerin işlemci sayısı, gücü ve bellek kapasitesi birbirinden farklıdır. Örneğin şekilde de görüldüğü gibi, *Node-6* düğümünün birden çok işlemcisi vardır. Şekil 7-14'te verilen örnek fiziksel kaynak modelinde düğümler ortak bir yerel ağa bağlıdır ve tüm düğümler arasındaki bağlantı hatlarının başarımlarının aynı olduğu varsayılmıştır. Bu örnek için düğümler arasındaki ağ bağlantısı homojen bir yerel alan ağı *(Local Area Network – LAN)* olarak tasarlanmış olsa da, Fiziksel Kaynak Tasarlama Aracı ile geniş alan ağlarını *(Wide Area Network – WAN)* da içeren, başarımları farklı ağlardan oluşan heterojen LAN/WAN ağ mimarileri tasarlamak da mümkündür.



Şekil 7-14 EH Benzetimi Örnek Durum Çalışması İçin Tasarlanmış Altı Düğümden Oluşan Bir Fiziksel Kaynaklar Modeli

7.1.4 Elektronik Harp benzetimi için çalıştırma konfigürasyonunun tasarlanması

Benzetim Çalıştırma Konfigürasyonu Tasarım Aracı, tasarımın bu aşamasına kadar oluşturulan Benzetim Veri Değişim Modeli (Şekil 7-7, Şekil 7-8 ve Şekil 7-9), Benzetim Modülleri ve Yayımlama/Üye Olma İlişkileri Modeli (Şekil 7-10, Şekil 7-11, Şekil 7-12 ve Şekil 7-13) ve Fiziksel Kaynak Modelini (Şekil 7-14) kullanarak benzetim çalıştırma konfigürasyonunun tanımlanmasına olanak sağlar.

Şekil 7-15'te Elektronik Harp Benzetim Sistemi örnek durum çalışması için tasarlanmış benzetim çalıştırma konfigürasyonunun bir kısmı gösterilmiştir. Bu benzetim çalıştırma konfigürasyonu 3.2 Kesiminde Çizelge 3-1'de verilmiş olan örnek senaryoya göre oluşturulmuştur.

Şekil 7-15'te, Çizelge 3-1'deki "Uçuş Benzetimi" katılımcılarına karşılık *FlightSimulator,* "Tank Benzetimi" katılımcılarına karşılık *TankSimulator* benzetim modül olguları tanımlanmıştır. Benzer şekilde Taktik Çevre Benzetimi katılımcısının uçuş benzetimlerine karşılık *TES_AirplaneSimulation* (TES kısaltması *"Tactical Environment Simulation"* yerine kullanılmıştır), gemi benzetimlerine karşılık *TES_ShipSimulation,* Komuta Kontrol Benzetimlerine karşılık *TES_CommandControlSimulation,* denizaltı benzetimlerine karşılık *TES_SubmarineSimulation,* tank benzetimlerine karşılık *TES_TankSimulation,* RF Yayılım Benzetimine karşılık *TES_PropagationModel,* EH sistemlerinden olan takip radarlarına karşılık *TES_TrackingRadarSimulation* benzetim modül olguları tanımlanmıştır.

Çalıştırma konfigürasyonunda ilgili benzetim modül olgusundan kaç adet bulunduğu bilgisi benzetim modülünün adının yanında parantez içinde "(x Olgu Sayısı)" şeklinde gösterilmiştir. Benzetim modül olgusu sayıları Çizelge 3-1'de verilen senaryo tanımına uygun olarak atanmıştır. Örneğin Çizelge 3-1'de 15 adet Tank Benzetimi katılımcısı olduğu için Şekil 7-15'te tank benzetim modül olgularını temsil eden bileşen "TankSimulator (x15)" olarak gösterilmiştir.

Şekil 7-15'te gösterilmeyen ama Çizelge 3-1'deki senaryoda tanımlı olan Gemi Benzetimi, Denizaltı Benzetimi, Taktik Çevre Benzetimi füze modelleri gibi diğer

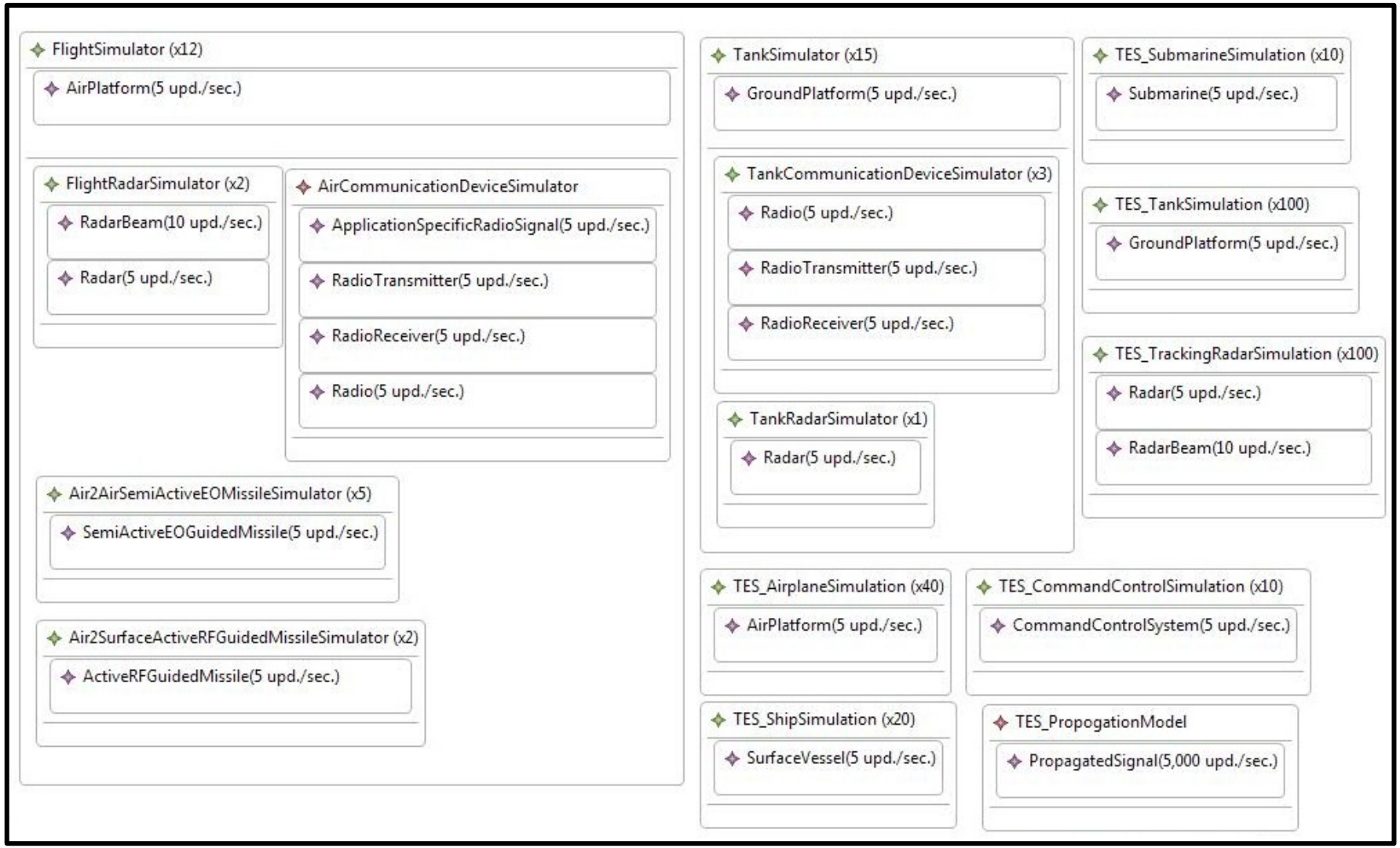
benzetim katılımcıları da benzetim çalıştırma konfigürasyonunda benzer biçimde tanımlanmıştır.

Şekil 7-15'te verilen benzetim çalıştırma konfigürasyonunun ana çatısını Çizelge 3-1'de tanımlı örnek senaryodaki tanımlamalar oluşturuyor olsa da, benzetim çalıştırma konfigürasyonu modelinde senaryo tanımının detay seviyesi arttırılmaktadır. Örneğin Çizelge 3-1'de uçuş simülatörü katılımcılarının üç adet Elektronik Harp cihazı olacağı belirtilmişken, Şekil 7-15'te verilen benzetim çalıştırma konfigürasyonunda bu tanımlama detaylandırılıp bir *FlightSimulator* benzetim modül olgusunun iki radar benzetim modül olgusu (şekilde *FlightRadarSimulator* olarak adlandırılmıştır) ve bir muhabere cihazı benzetim modül olgusu (şekilde *AirCommunicationDeviceSimulator* olarak adlandırılmıştır) içereceği tanımlanmıştır.

Benzetim çalıştırma konfigürasyonu modelinde senaryo tanımının detaylandırıldığı bir diğer nokta, benzetim modül olgularının veri yayımlama sıklıklarıdır. Örneğin *FlightRadarSimulator* benzetim modül olgusu *RadarBeam* nesne sınıfını saniyede on defa, *Radar* nesne sınıfını ise saniyede beş defa günceller. Bu bilgi Şekil 7-15'te ilgili benzetim modül olgusuna eklenen veri bileşenlerinde gösterilmektedir. Örneğin *FlightRadarSimulator* benzetim modül olgusunun içerisinde *RadarBeam* yayımlaması için ayrı bir bileşen tanımlanmış ve bu nesne sınıfının saniyede on defa güncellendiği bilgisi *RadarBeam(10 upd./sec.)* şeklinde tanımlanmıştır.

Bu noktada, aslında radar benzetim modülünün *Radar* ve *RadarBeam* benzetim veri değişim modeli elemanlarını yayımlayacağı bilgisinin Benzetim Modülleri ve Yayımlama/Üye Olma İlişkileri tasarlanırken tanımlandığını hatırlatmakta fayda vardır.

Senaryodaki benzetim modül olgularının başka benzetim modül olgularını içerme özellikleri Şekil 7-15'te iç içe bileşenler ile gösterilmiştir. Örneğin *TankSimulator* benzetim modül olgusu üç *TankCommunicationDeviceSimulator* ve bir *TankRadarSimulator* benzetim modül olgusu içermektedir.



Şekil 7-15 EH Benzetimi Örnek Durum Çalışması İçin Tasarlanmış Benzetim Çalıştırma Konfigürasyonu Modelinin Bir Kısmı

7.2 Elektronik Harp Benzetimi için Yerleştirme Modelinin Üretilmesi

Bu kesime kadar benzetim ortamının tasarımı fiziksel kaynakları da içerecek şekilde el ile yapılmıştır. El ile geliştirilen bu modeller kullanılarak uygun yerleştirme modeli otomatik olarak üretilebilir. Bu kesimde Elektronik Harp örnek durum çalışması için yerleştirme modelinin otomatik olarak üretilmesi ele alınacaktır.

Çizelge 3-1'de senaryosu verilen Elektronik Harp Benzetim Sistemi örnek durum çalışması için otomatik olarak üretilmiş uygun yerleştirme modeli Şekil 7-16'da verilmiştir. Bu modelde üst seviye yerleşimi görebilmek adına düğümlerin tam konfigürasyonları gösterilmemiştir. İzleyen şekillerde her düğüm için yerleştirilen tüm benzetim modül olguları sırasıyla gösterilmiştir. Sırasıyla Şekil 7-17, Şekil 7-18, Şekil 7-19, Şekil 7-20, Şekil 7-21 ve Şekil 7-22'de altı düğüm için tam konfigürasyon verilmiştir.

Aslında yerleştirme modeli Eclipse modelleme ortamında ve yaklaşım açıklanırken tanımlanan Yerleştirme Metamodeline (Şekil 5-11) uygun olarak üretilmiştir, fakat üretilen model çok sayıda unsur içerdiği için aslına uygun olarak tekrar çizilerek ilgili şekiller oluşturulmuştur.

Şekil 7-16'da gösterildiği gibi, otomatik üretilen yerleştirme modeli Şekil 7-14'e tanımlanan Fiziksel Kaynak modeline uygun olarak altı düğüm içermektedir. Şekil 7-15'de bir kısmı verilen benzetim çalıştırma konfigürasyonu modelindeki modül olguları bu altı düğüm üzerine çalıştırma maliyeti, iletişim maliyeti ve bellek ihtiyacı kalite faktörlerini eniyileyecek şekilde otomatik olarak yerleştirilmiştir.

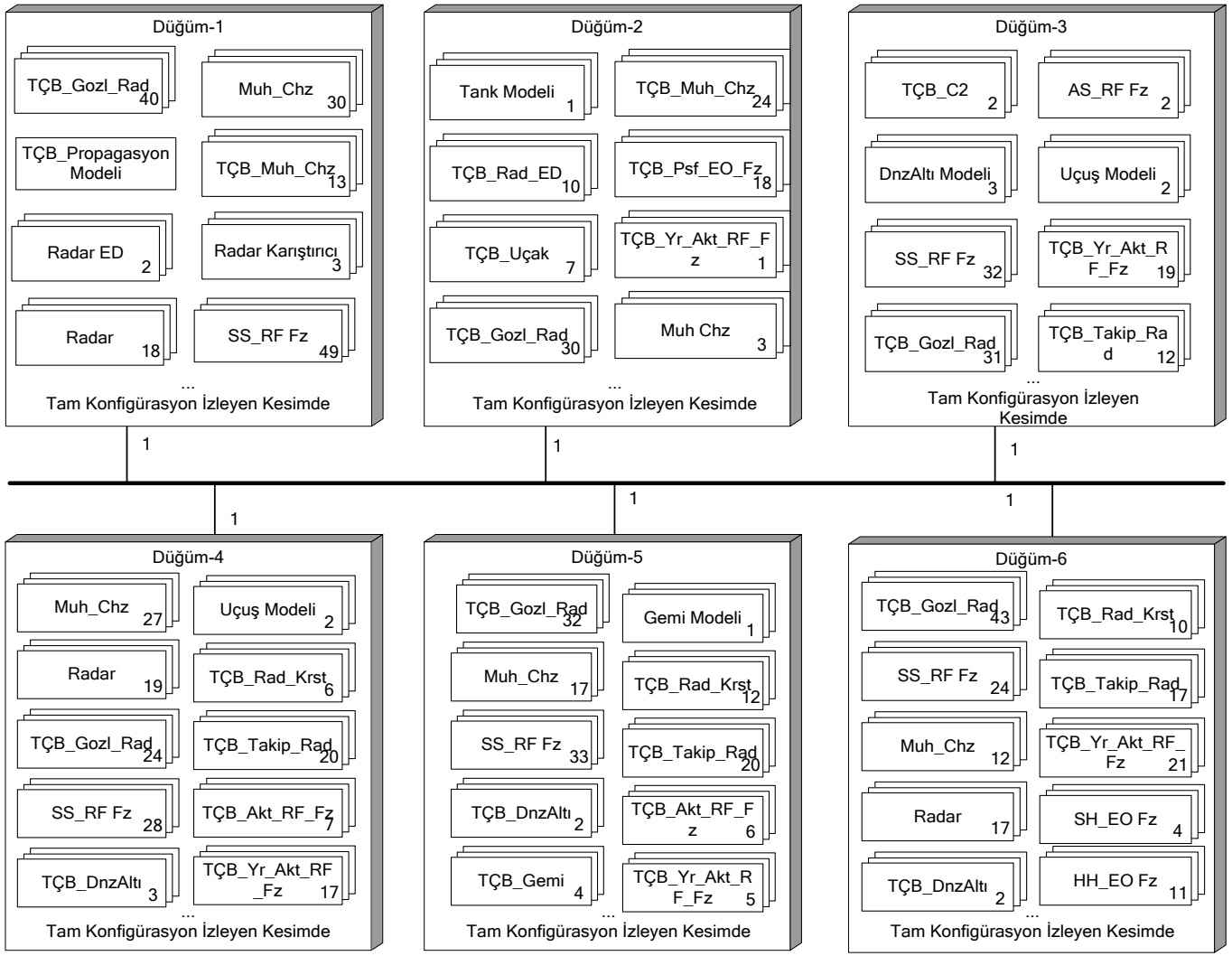
İzleyen alt kesimde, otomatik olarak üretilen yerleştirme modeli irdelenecektir.

7.2.1 Elektronik Harp benzetimi için üretilen yerleştirme modelinin irdelenmesi

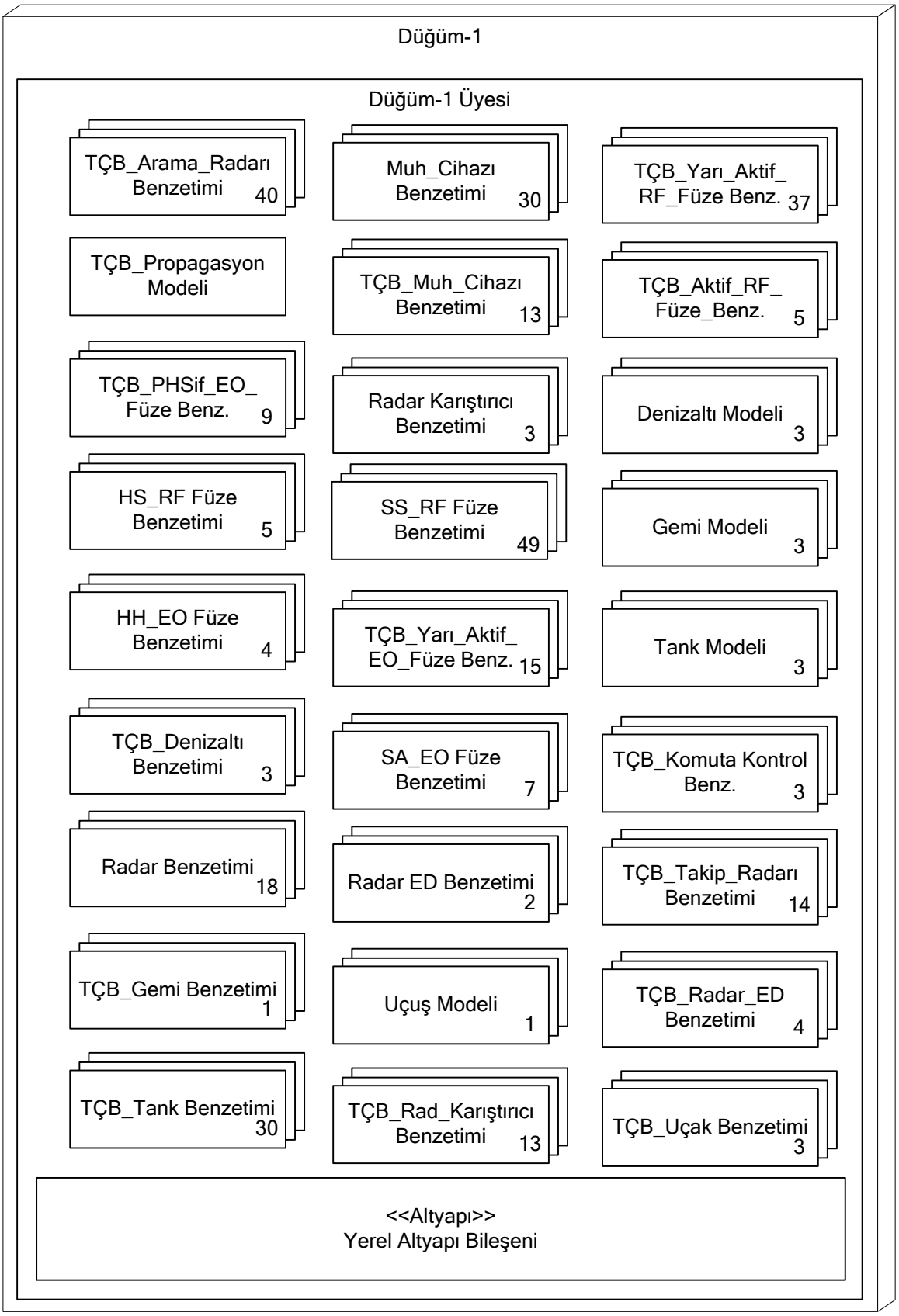
5.6 kesiminde uygun yerleştirme türetme algoritması açıklanırken tanımlanan gereksinimlerden birisi, düğümlerin bellek kapasitesinin aşılmamasıdır. Otomatik üretilen yerleştirme modeli bu bakış açısıyla yakından incelendiğinde, her düğüme yerleştirilen benzetim modül olgularının toplam bellek ihtiyacının ilgili düğümün bellek kapasitesini aşmadığı görünmektedir.

Üretilen yerleştirme modeli detaylı olarak incelendiğinde, kapasite kısıtlarına uyulmasının yanı sıra, kullanılan genetik algoritma tabanlı iş atama algoritmasının [Mehrabi et al. 2009] mantıklı bir sonuç ürettiği gözlemlenmiştir. Bu kapsamda, birbirleriyle sık haberleştiği bilinen benzetim modül olguları mümkün mertebe aynı düğüm üzerine yerleştirilmiştir. Örneğin 7.1.2 Kesiminde tanımlanan benzetim modülleri ve yayımlama/üye olma ilişkileri modeli incelenirse, RF Yayılım benzetim modülünün (*PropagationModel*), Radar, Muhabere Cihazı, Radar Karıştırıcı gibi sistemler ve uçak, tank, gemi, füzeler gibi platformlarla sıkı bir veri alışverişi içerisinde olacağı gözlemlenmektedir. Bu durumda uzman değerlendirmesi (*expert judgment*) ile bir yerleştirme modeli türetildiği durumda RF Yayılım benzetim modülü olgusunu ilişkili olduğu diğer benzetim modül olgularıyla mümkün mertebe aynı düğüme yerleştirmek mantıklı olacaktır. Şekil 7-16 incelendiğinde RF Yayılım modelinin bir numaralı düğüme yerleştirildiği anlaşılmaktadır. Şekil 7-17'de verilen bir numaralı düğüme yerleştirilmiş benzetim modül olgularının tam listesi incelendiğinde, bu düğüme RF Yayılım modeli ile birlikte 40 Arama Radarı, toplam 43 Muhabere Cihazı, 37 Yarı Aktif RF Güdümlü Füze, 5 Aktif RF Güdümlü Füze, 3 Radar Karıştırıcı, 3 Denizaltı modeli, 3 Gemi Modeli, 3 Tank Modeli, 3 Uçak Modeli, vs. şeklinde devam eden benzetim modül olgularının yerleştirildiği görünmektedir. Uzman gözüyle, bunun mantıklı görünen bir yerleştirme olduğu söylenebilse de, bu tez kapsamında geliştirilen yaklaşıma benzer sistematik bir yaklaşım olmadan bu şekilde hangi benzetim modül olgusundan kaç tanesinin hangi düğüme yerleştirileceği net olarak belirlenemez.

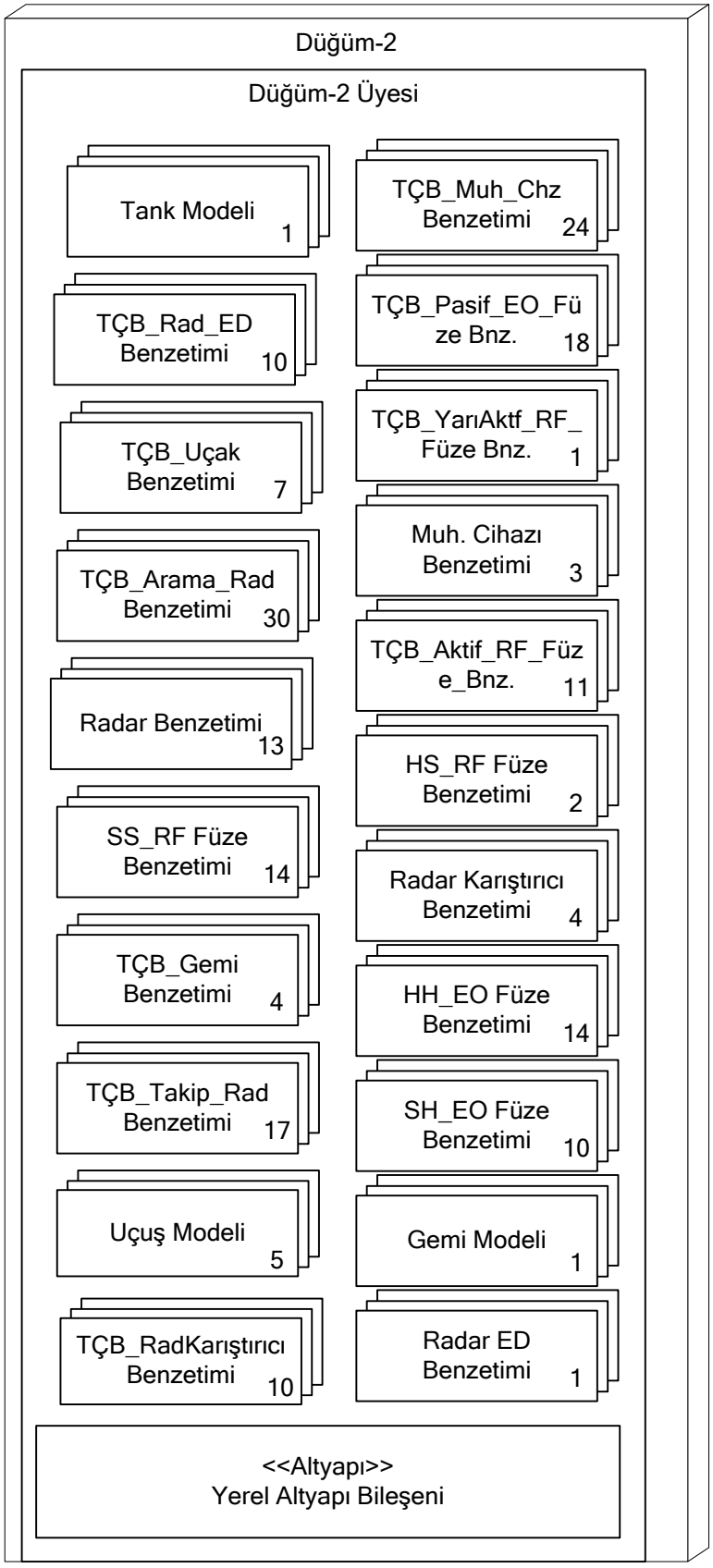
Bir numaralı düğüme benzer şekilde diğer düğümlerin yerleştirme konfigürasyonları da incelendiğinde, bu düğümlerin de konfigürasyonlarının kapasite kısıtları aşılmadan sık haberleşen benzetim modülü olgularını mümkün mertebe aynı düğüme yerleştirilmesi mantığıyla oluşturulduğu gözlemlenmektedir.



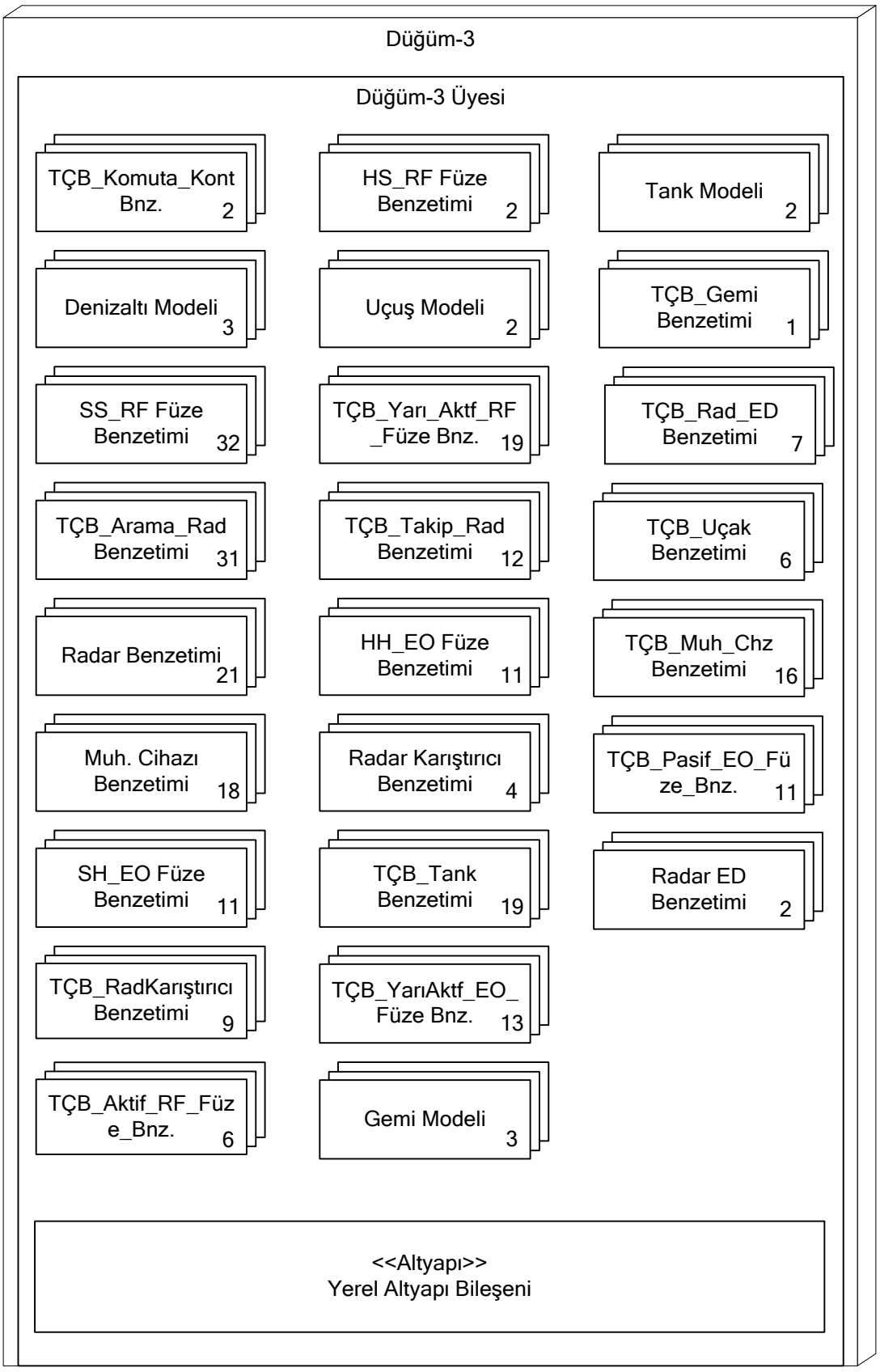
Şekil 7-16 EH Benzetimi Örnek Durum Çalışması için Otomatik Olarak Üretilmiş Uygun Yerleştirme Modeli



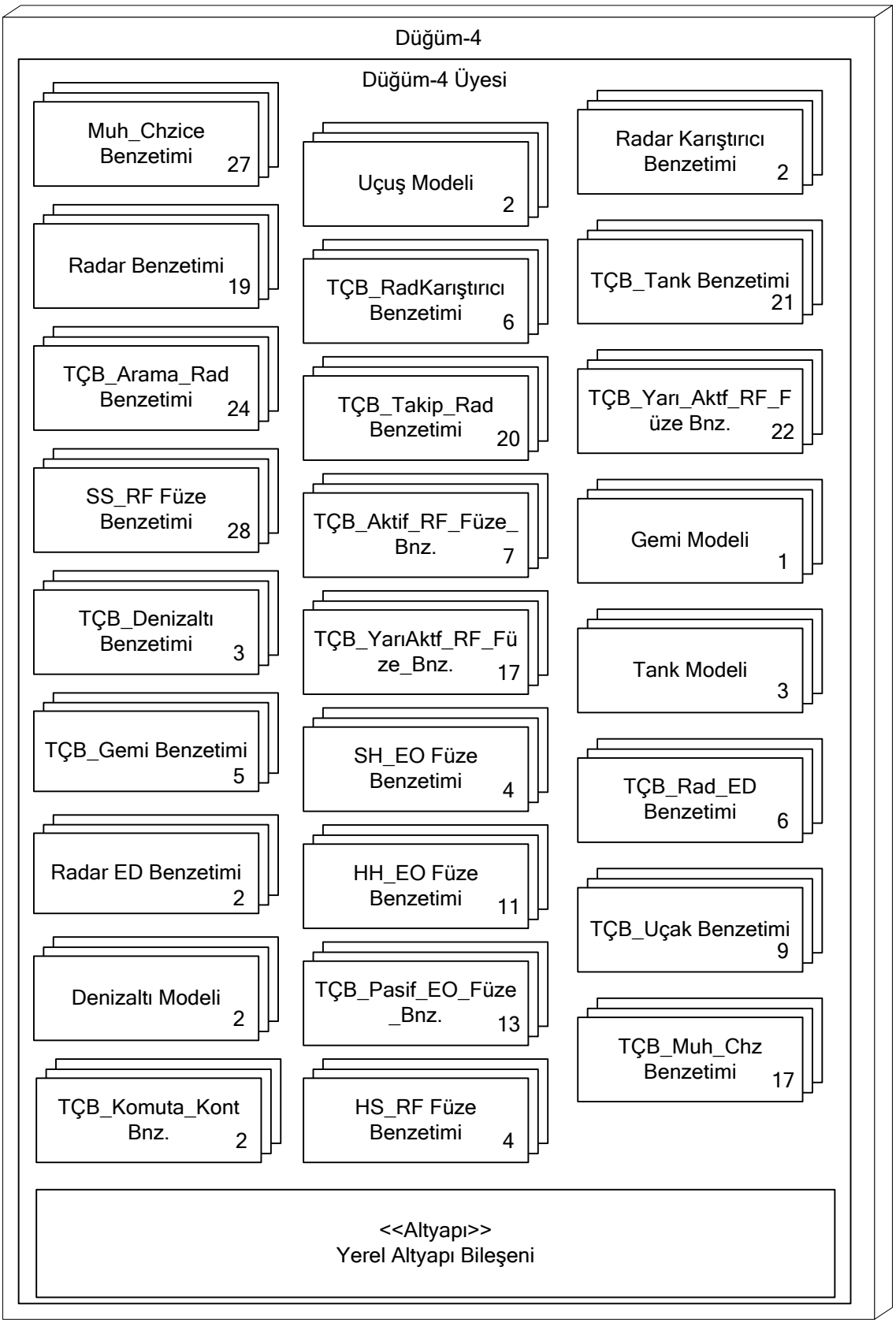
Şekil 7-17 EH Benzetimi Örnek Durum Çalışması için Otomatik Olarak Üretilmiş Uygun Yerleştirme Modeli – Düğüm#1



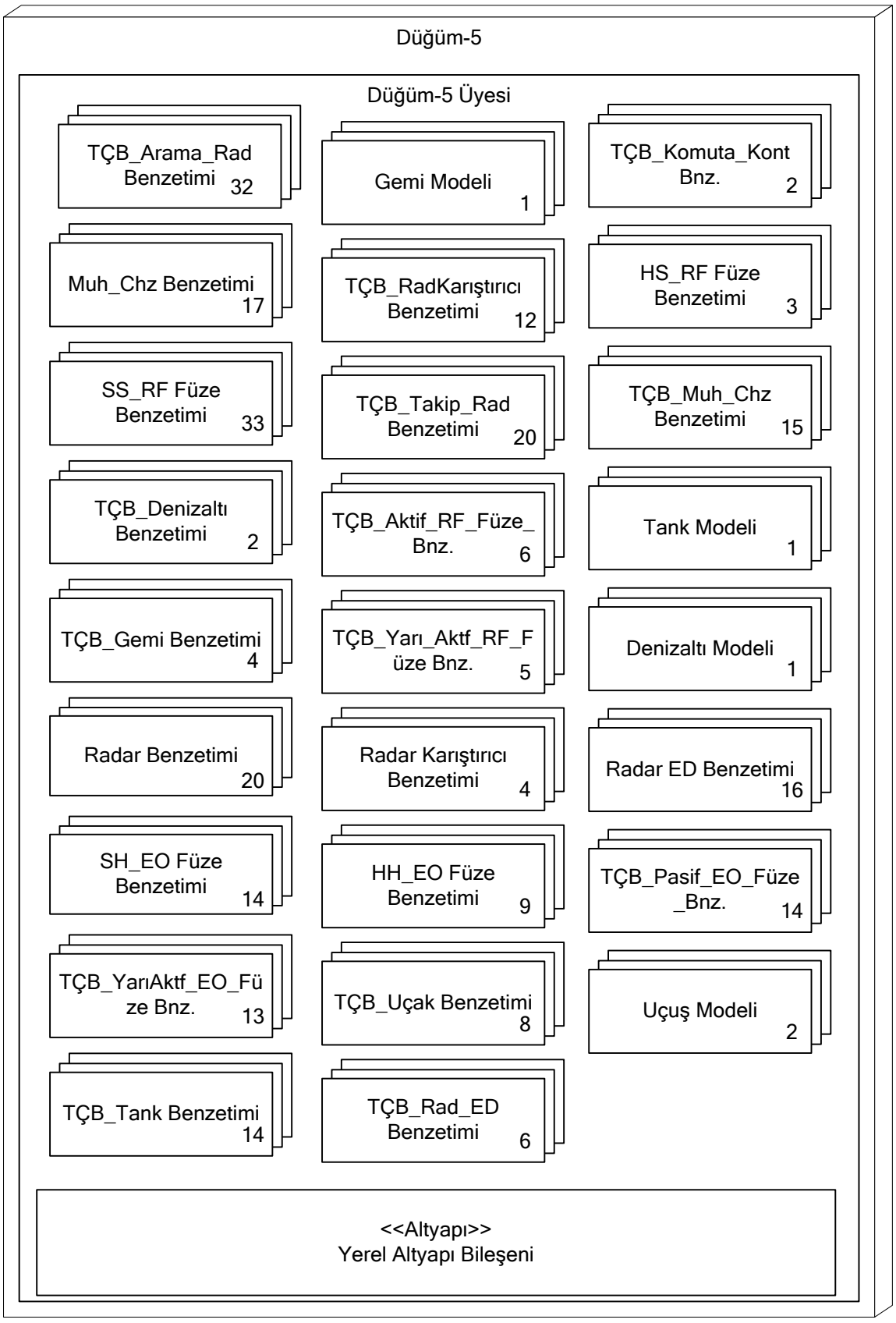
Şekil 7-18 EH Benzetimi Örnek Durum Çalışması için Otomatik Olarak Üretilmiş Uygun Yerleştirme Modeli – Düğüm#2



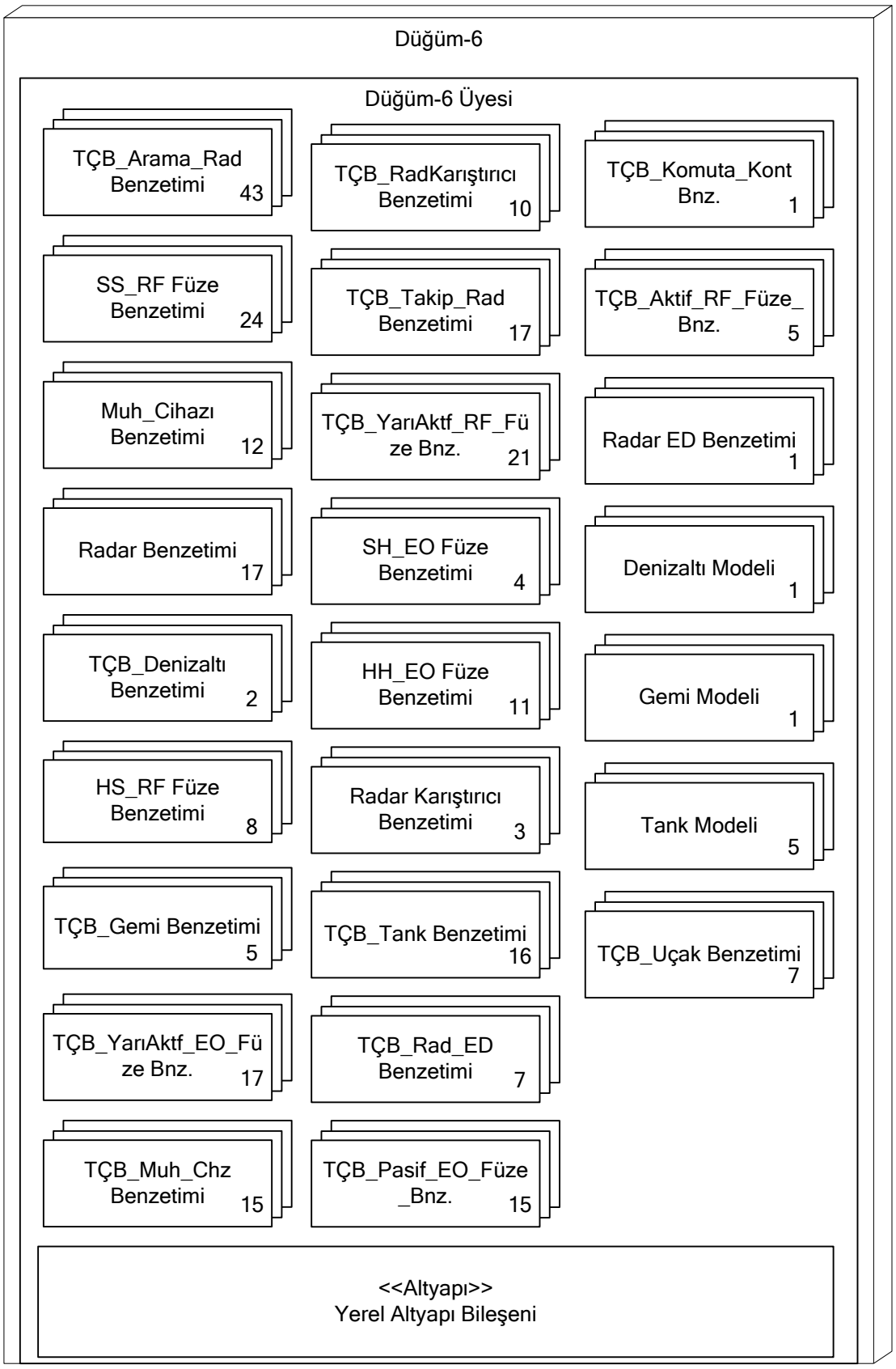
Şekil 7-19 EH Benzetimi Örnek Durum Çalışması için Otomatik Olarak Üretilmiş Uygun Yerleştirme Modeli – Düğüm#3



Şekil 7-20 EH Benzetimi Örnek Durum Çalışması için Otomatik Olarak Üretilmiş Uygun Yerleştirme Modeli – Düğüm#4



Şekil 7-21 EH Benzetimi Örnek Durum Çalışması için Otomatik Olarak Üretilmiş Uygun Yerleştirme Modeli – Düğüm#5



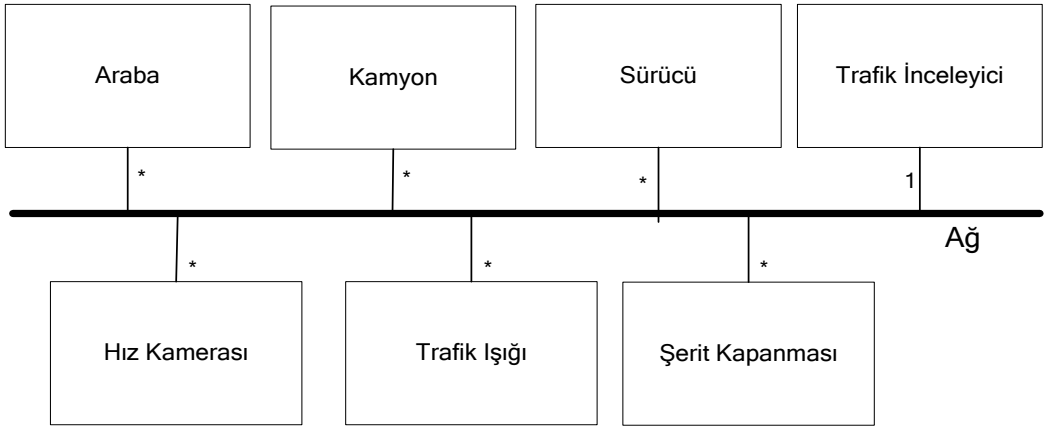
Şekil 7-22 EH Benzetimi Örnek Durum Çalışması için Otomatik Olarak Üretilmiş Uygun Yerleştirme Modeli – Düğüm#6

8 GELİŞTİRİLEN YAKLAŞIMIN TRAFİK BENZETİMİ ÖRNEK DURUM ÇALIŞMASINA UYGULANMASI

Önceki kesimde, *S-IDE* Geliştirme Ortamı kullanılarak tez çalışmasında geliştirilen yaklaşımın askeri bir benzetim sistemi olan bir Elektronik Harp Benzetimine uygulanması ele alınmıştır. Bu kesimde, geliştirilen yaklaşımın askeri benzetim sistemleri dışında farklı bir alanda denenmesi için ikinci bir örnek çalışma olarak bir trafik benzetimi ele alınacaktır.

8.1 Trafik Benzetiminin Kavramsal Tanımı

Bu örnek çalışmada ele alınan trafik benzetiminin temel amacı, trafiğin etkin akışını sağlamak için çeşitli trafik akış parametrelerinin analiz ve eniyilenmesinin desteklenmesidir. Trafik benzetim sisteminin üst seviye mimarisi Şekil 8-1'de verilmiştir.



Şekil 8-1 Trafik Benzetiminin Üst Seviye Mimarisi

Trafik benzetiminin katılımcıları, arabalar, kamyonlar, sürücüler, hız kameraları, trafik ışıkları, şerit kapanmaları ve bir trafik inceleyicisidir. Yaya geçidi, yayalar, sabit/hareketli hız radarları, kavşaklar ve hava koşulları gibi trafik akışını etkileyen diğer faktörler örnek çalışmanın basitliğini korumak için durum çalışmasına dahil edilmemiştir. Şekil 8-1'de trafik inceleyici katılımcısından bir adet olacağı belirtilmiş, diğer katılımcılar için net sayı verilmemiş, sıfır ya da daha çok katılımcı olabileceğini belirtmek için "*" kullanılmıştır. Katılımcıların net sayısı senaryo tanımı yapılırken belirlenir. Bir sonraki alt kesimde örnek bir senaryo tanımı yapılacaktır.

Tanımlanan trafik benzetiminde taşıt aracı olarak arabalar ve kamyonlar bulunacaktır. Bir taşıt aracının model yılı, motor gücü, mevcut sürücünün ehliyet

numarası gibi nitelikleri vardır. Sürücülerin trafik akışını etkileyen değişik fiziksel ve davranışsal nitelikleri vardır.

Bir sürücünün ehliyet numarası ve sosyo-demografik niteliklerine (yaş, cinsiyet, sürüş deneyimi, vs.) ek olarak, trafik benzetimi konusundaki literatüre göre sürüş tarzı (tedirgin, tehlikeli, öfkeli, yüksek hızlı, stresli, dalgın, sakin, dikkatli, vs.) [Taubman-Ben-Ari et al. 2004] ve kaza geçmişi [Chung, and Wong 2010] gibi ilave nitelikleri vardır.

Hız kameraları, trafik ışıkları ve şerit kapanmaları genel anlamda trafik akışını yavaşlatan unsurlardır. Bir hız kamerasının coğrafi konum ve hız limiti nitelikleri vardır. Bir trafik ışığının coğrafi konumu ve ışık durumu (kırmızı, sarı, yeşil) nitelikleri vardır. Bir şerit kapanmasının başlangıç konumu, bitiş konumu, şeridin kapalı kalma zaman aralığı ve şerit numarası (1. Şerit, 2. Şerit gibi) nitelikleri vardır.

Trafik inceleyici, taşıt araçları ve sürücüler gibi diğer katılımcılardan veri toplayıp çeşitli analizler yapan pasif bir katılımcıdır.

8.1.1 Trafik benzetimi için örnek bir senaryo

Üst kesimde trafik benzetim ortamı tanımlandıktan sonra, bu alt kesimde örnek bir benzetim senaryosu tanımlanacaktır. Çizelge 8-1'de trafik benzetimi için örnek bir senaryo verilmiştir.

Çizelge 8-1 Trafik Benzetimi Durum Çalışması için Örnek Bir Senaryo Tanımı

Benzetim Katılımcısı	Unsur Sayısı
Araba Benzetimi	600
Kamyon Benzetimi	80
Sürücü Benzetimi	680
Hız Kamerası Benzetimi	5
Trafik Işığı Benzetimi	15
Şerit Kapanması Benzetimi	4
Trafik İnceleyici Benzetimi	1

Çizelge 8-1'deki senaryo tanımında gösterildiği gibi, senaryoda 600 araba ve 80 kamyon katılımcısı vardır. Senaryo tanımında toplam 1386 benzetim katılımcısı vardır.

8.1.2 Trafik benzetimi örnek senaryosu için uzman değerlendirmesiyle yerleştirme modelleri oluşturulması

İlk örnek durum çalışması olan Elektronik Harp Benzetimi için yapıldığı gibi, trafik benzetimi senaryosu için de uzman değerlendirmesiyle çeşitli yerleştirme modelleri tanımlanabilir.

Örneğin, uzman değerlendirmesiyle Çizelge 8-1'de tanımlanan senaryo için tüm araba benzetim modüllerinin birinci düğüme, tüm kamyon benzetim modüllerinin ikinci düğüme, tüm sürücü benzetim modüllerinin üçüncü düğüme ve kalan benzetim modüllerinin dördüncü bir modüle yerleştirildiği bir model oluşturulabilir. Bu yerleştirme modeli farklı benzetim modülü türleri için ayrı birer düğüm tanımlamaya dayanan basit bir mantık izlemesi açısından kolay anlaşılır bir çözümdür. Bu çözümde, aynı türdeki benzetim modülleri (örneğin arabalar) aynı düğüme yerleştirildiğinden, kendi aralarındaki iletişim yükü düşecektir. Bu yerleştirme seçeneği, basit anlaşılır bir model olmasına karşın, her durumda etkin bir çözüm değildir, örneğin farklı düğümlere yerleştirilmiş benzetim modülleri (örneğin arabalar, kamyonlar ve sürücüler) arasında yoğun biçimde etkileşim olduğunda iletişim maliyetleri artacaktır.

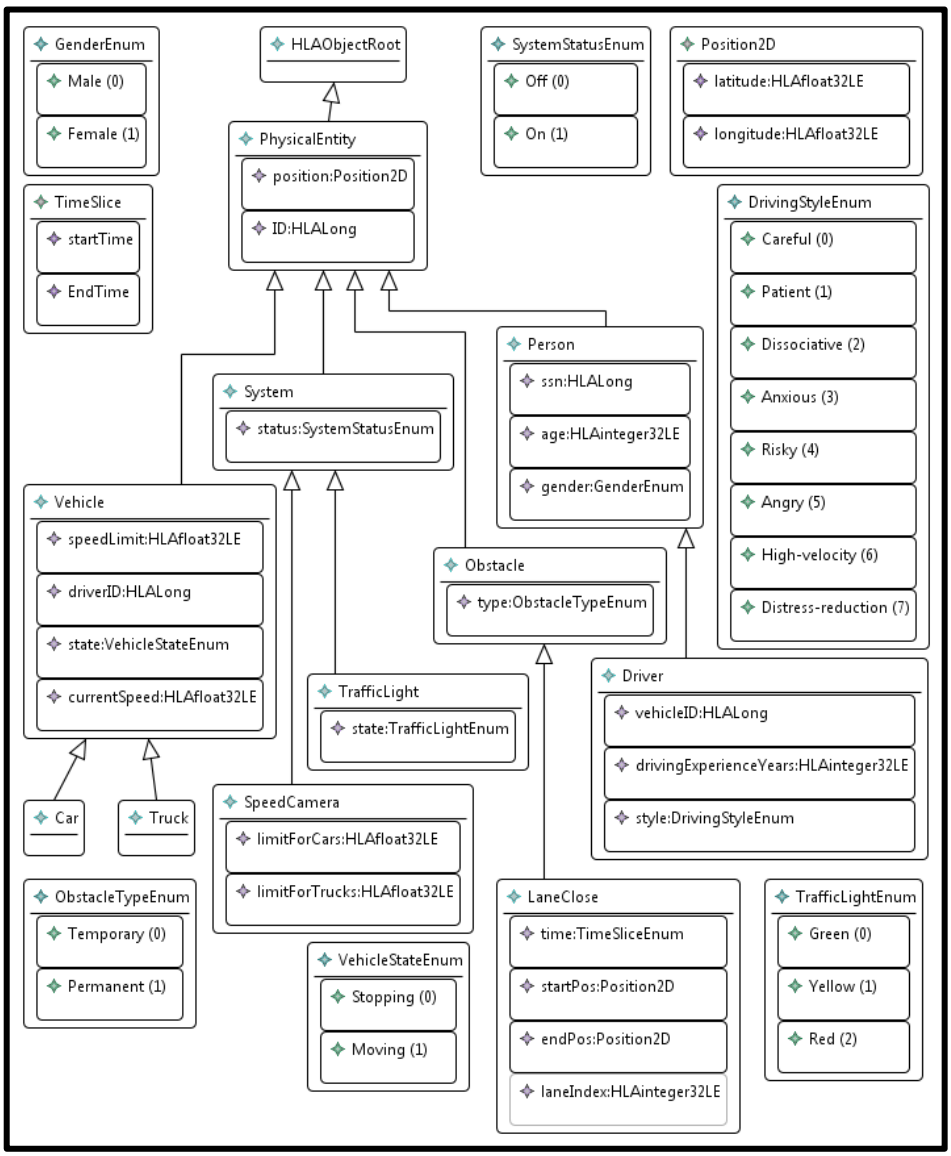
İkinci örnek yerleştirme modelinde, bir önceki seçenekten farklı olarak dört yerine üç düğüm tanımlanabilir. Bu seçenekte birbiriyle sık haberleşeceği öngörülen araba, kamyon ve sürücü benzetim modülleri birinci düğüme, hız kamerası, trafik ışıkları ve şerit kapanma modelleri ikinci düğüme, trafik izleyicisi modülü ise üçüncü düğüme yerleştirilir. Bu yerleştirme seçeneğinde, araba, kamyon ve sürücü modülleri arasındaki iletişim maliyeti aynı düğüme yerleştirildiklerinden azalır, fakat yine aynı sebepten ötürü bu düğümdeki bellek, işlemci gibi kaynakların yetersiz kalması durumu oluşabilir.

8.2 *S-IDE* Aracı Kullanılarak Trafik Benzetimi İçin Benzetim Ortamının Modellenmesi

Bu kesimde geliştirilen yaklaşımın akışına uygun olarak trafik benzetim ortamı modelinin adım adım tasarlanması ele alınacaktır.

8.2.1 Trafik benzetimi veri değişim modelinin tasarlanması

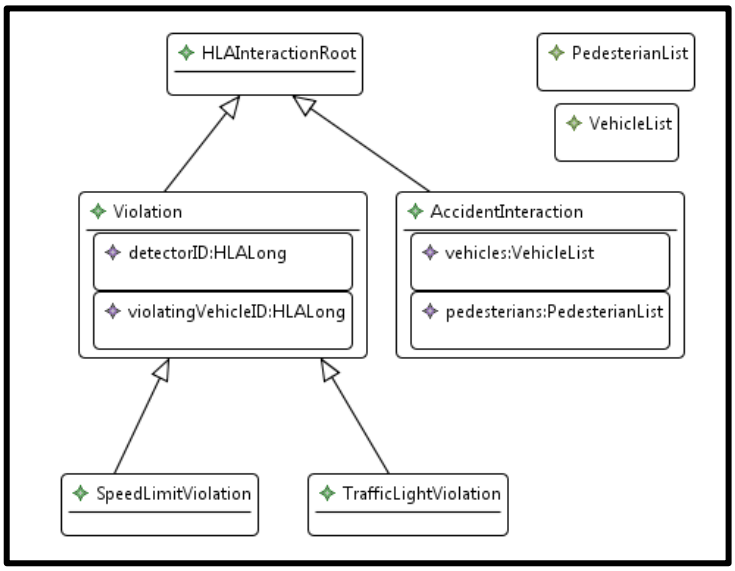
Şekil 8-2 ve Şekil 8-3'te *S-IDE* aracı ile tasarlanmış Trafik Benzetimi Veri Değişim Modelinin nesne sınıfları, etkileşim sınıfları ve gerekli veritürü tanımları gösterilmiştir.



Şekil 8-2 Trafik Benzetimi Veri Değişim Modeli – Nesne Sınıfları

Şekil 8-2'de *HLAObjectRoot* nesne sınıfı HLA OMT standardına uygun olarak diğer tüm nesne sınıfları için kök sınıf olarak tanımlanmıştır. *PhysicalEntity* nesne sınıfı kök sınıftan türer ve bir fiziksel unsurun iki temel özelliği olan konum ve kimlik bilgisi niteliklerini tanımlar. *PhysicalEntity* nesne sınıfının *Position* niteliğinin veri türü, konum bilgisini enlem ve boylam olarak tanımlayan *Position2D*'dir. *Car*, *Truck*, *TrafficLight*, *SpeedCamera* ve *Driver* nesne sınıfları da benzer biçimde ihtiyaç duydukları veritürleriyle birlikte tanımlanmıştır. Örneğin, sürüş tarzını tanımlamak için *DrivingStyleEnum*, trafik ışığı durumunu belirlemek için ise *TraficLightEnum* numaralandırılmış veritürleri tanımlanmıştır.

Trafik Benzetimi Veri Değişim Modelinin etkileşim sınıflarını tanımlayan Şekil 8-3'te, HLA OMT standardına uygun olarak tüm nesne sınıflarının kök sınıfı olarak *HLAInteractionRoot* sınıfı tanımlanmıştır. Hız limiti aşımı durumda gönderilecek olan *SpeedLimitViolation*, trafik ışığı ihlali durumunda gönderilecek olan *TrafficLightViolation*, kaza durumunda gönderilecek olan *AccidentInteraction* etkileşim sınıfları gerekli parametreleri ve veritürleriyle birlikte tanımlanmıştır.

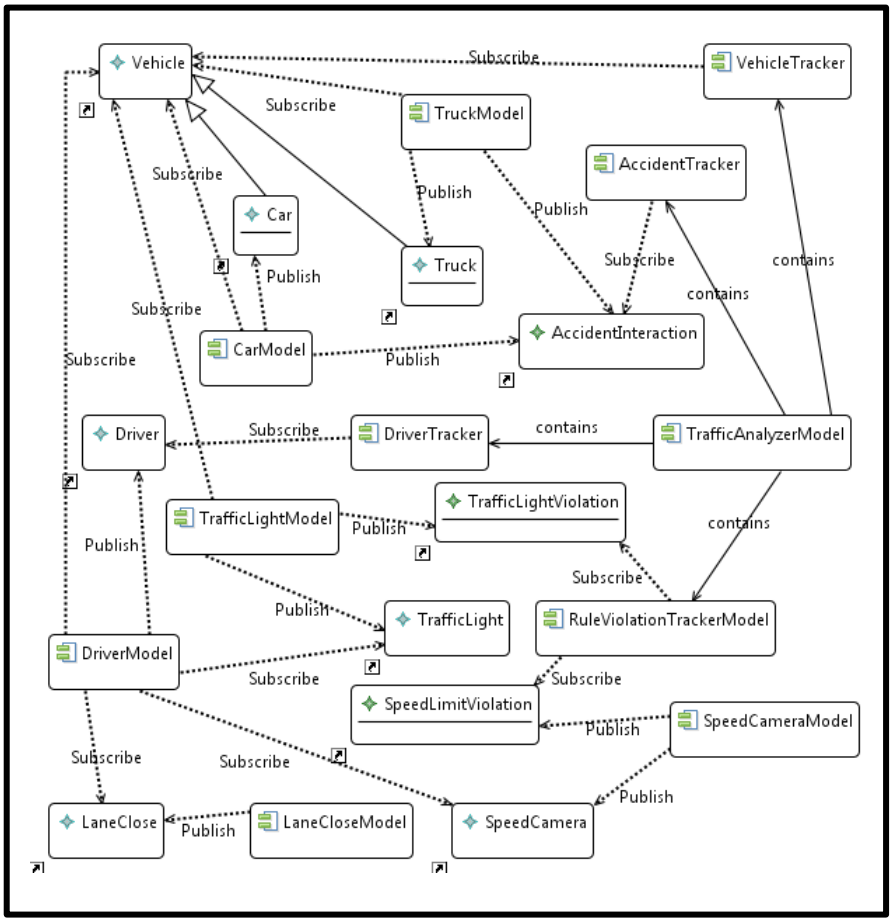


Şekil 8-3 Trafik Benzetimi Veri Değişim Modeli – Etkileşim Sınıfları

8.2.2 Trafik benzetiminin modüllerinin ve yayımlama/üye olma ilişkileri modelinin tasarlanması

Şekil 8-4'te trafik benzetiminin modülleri ve yayımlama/üye olma ilişkileri tanımlanmıştır. Yayımlama/Üye Olma ilişkileri, bir önceki kesimde tanımlanmış

olan veri değişim modeli (Şekil 8-3 ve Şekil 8-4) elemanları kullanılarak tanımlanmıştır.



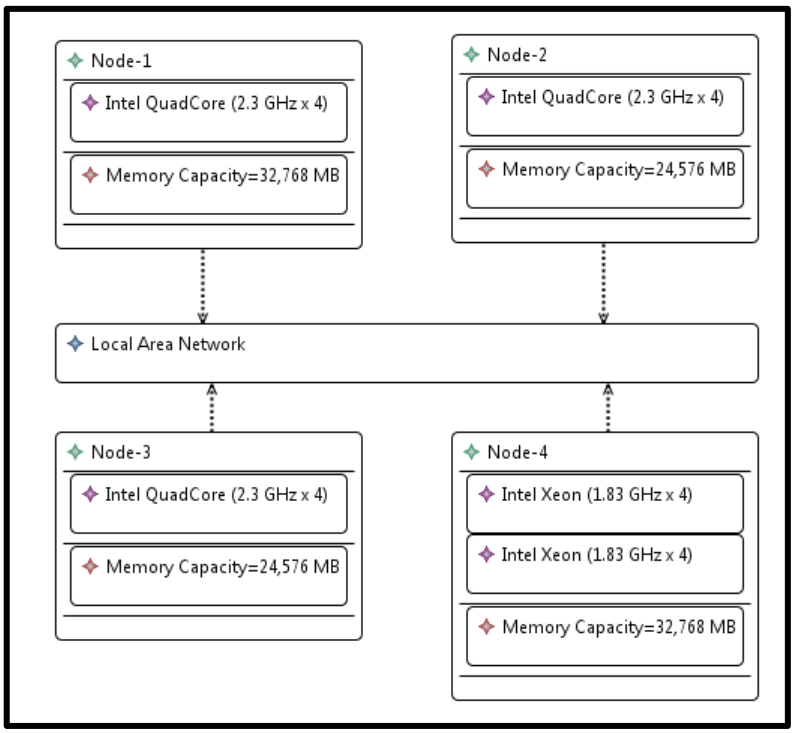
Şekil 8-4 Trafik Benzetimi Modülleri ve Yayımlama/Üye Olma İlişkileri Modeli

Şekil 8-4'te, örnek durum çalışmasına uygun olarak *CarModel*, *TruckModel*, *DriverModel*, *TrafficLightModel*, *LaneCloseModel* ve *SpeedCameraModel* benzetim modülleri tanımlanmıştır. *TrafficAnalyzer* benzetim modülü, detaylandırılarak *DriverTracker, VehicleTracker, AccidentTracker* ve *RuleViolationTracker* alt benzetim modülleri tanımlanmıştır. Bu alt modülleri içeren *TrafficAnalyzer* modülü birden çok atomik modülü içeren bağlı (*coupled*) bir modüldür (Bkz. Kesim 5.3, Atomik ve Bağlı modül tanımlamaları).

Her modül, modelleme sorumluluklarına göre çeşitli nesne ve etkileşim sınıflarını yayımlar. Örneğin, *CarModel* modülü, *Car* nesne sınıfını ve *AccidentInteraction* etkileşim sınıfını yayımlar. Modüller aynı zamanda ilgi duydukları nesne ve etkileşim sınıflarına üye olurlar. Örneğin, *DriverModel* modülü sürücünün davranışlarını modelleyebilmek için *TrafficLight* ve *Vehicle* nesne sınıflarına üye olur.

8.2.3 Trafik benzetimi için fiziksel kaynaklar modelinin tasarlanması

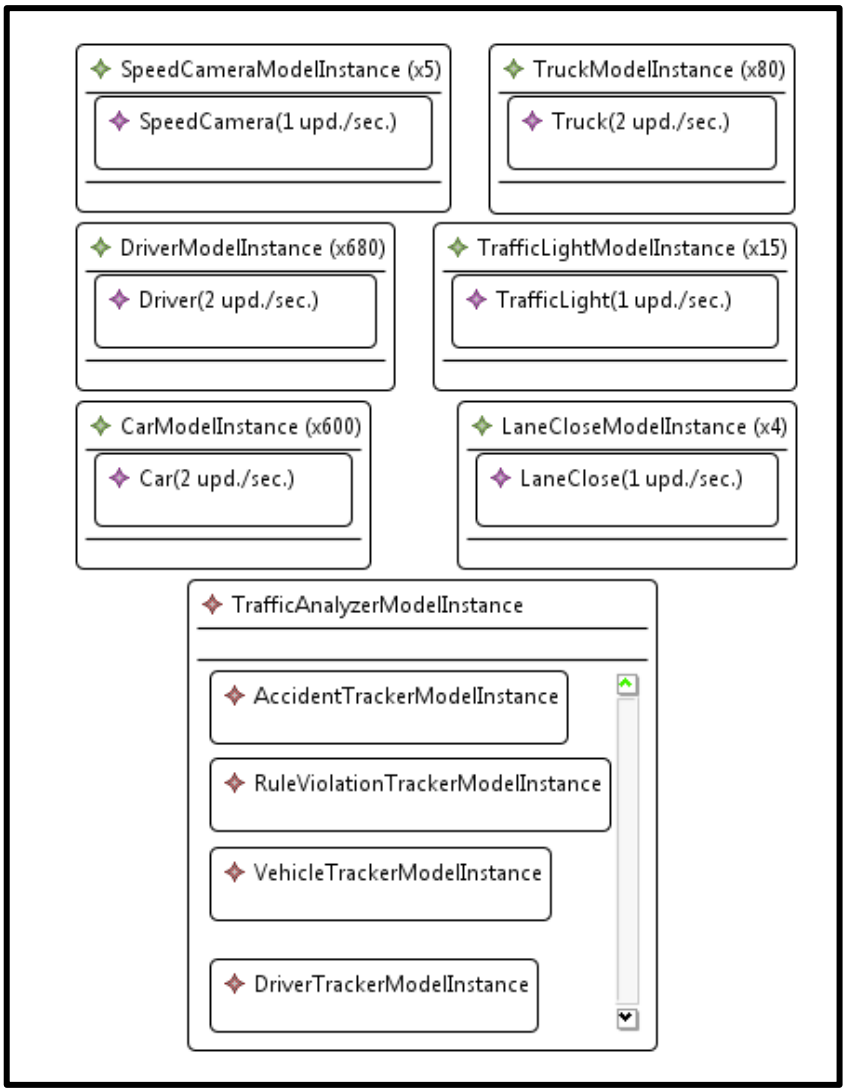
Trafik Benzetimi için örnek bir fiziksel kaynaklar modeli Şekil 8-5'te verilmiştir. Bu modelde değişik işlemci ve bellek kapasiteleri olan dört farklı düğüm tanımlanmıştır. Şekilde gösterildiği gibi *Node-4* gibi bazı düğümler birden çok işlemciye sahip olabilir.



Şekil 8-5 Trafik Benzetimi için Örnek Fiziksel Kaynak Modeli

8.2.4 Trafik benzetimi için çalıştırma konfigürasyonunun tasarlanması

Modül ve yayımlama/üye olma ilişkileri modeli (Şekil 8-4) ve fiziksel kaynak modeli (Şekil 8-5) kullanılarak Çizelge 8-1'de verilen trafik benzetimi senaryosu için örnek bir çalıştırma konfigürasyonu Şekil 8-6'daki gibi tanımlanabilir. Şekil 8-6'da gösterildiği gibi, Çizelge 8-1'e uygun olarak çalıştırma konfigürasyonunda 600 araba, 80 kamyon, 680 sürücü ve diğer gerekli benzetim modülleri tanımlanmıştır. Çizelge 8-1'den farklı olarak Trafik İzleyicisi modülü alt modülleriyle (*AccidentTrackerModel*, *VehicleTrackerModel*, vs.) birlikte daha detaylı olarak modellenmiştir. Şekil 8-6'da benzetim modüllerinin, veri değişim modeli elemanlarını yayımlama sıklıkları da tanımlanmıştır. Örneğin, *CarModelInstance* modülü *Car* nesnesini saniyede iki defa günceller.



Şekil 8-6 Trafik Benzetimi için Örnek Çalıştırma Konfigürasyonu

8.3 Trafik Benzetimi için Yerleştirme Modelinin Üretilmesi

Trafik Benzetimi için otomatik olarak üretilen bir yerleştirme seçeneği Şekil 8-7'de verilmiştir. Elektronik Harp Benzetimi için yerleştirme seçeneği üretilirken izlenilen yönteme benzer olarak, trafik benzetimi için de CTAP algoritması olarak Mehrabi et al.'ın [2009] tanımladığı algoritma kullanılmıştır.

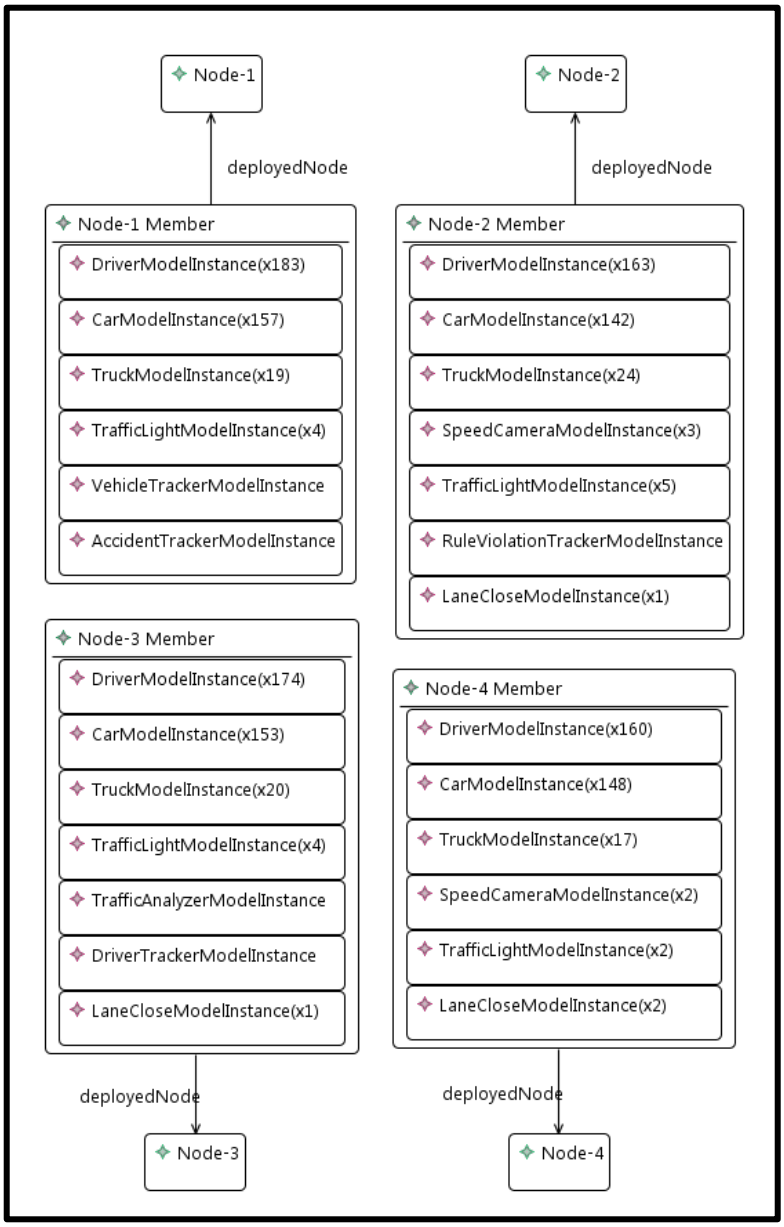
Şekil 8-7'de görüldüğü gibi, yerleştirme modeli Şekil 8-5'deki fiziksel kaynak modeline uygun olarak dört düğüm içerir. Yerleştirme modelindeki benzetim modül olguları da Şekil 8-6'da verilen çalıştırma konfigürasyonu ile uyumludur.

İzleyen alt kesimde, otomatik olarak üretilen yerleştirme modeli irdelenecektir.

8.3.1 Trafik benzetimi için üretilen yerleştirme modelinin irdelenmesi

Şekil 8-7'deki yerleştirme modeli detaylı olarak incelendiğinde, üretilen yerleştirme seçeneğinde her düğüme yerleştirilmiş olan benzetim modül olgularının toplam kaynak ihtiyaçları ilgili düğümün toplam kapasitesini aşmamaktadır.

Kullanılan CTAP algoritması, sık haberleşen modül olguları mümkün mertebe aynı düğüme yerleştirilmiştir. Örneğin trafik benzetimi tasarımında sık haberleştikleri görülen *VehicleTracker*, *CarModel, TruckModel* ve *DriverModel* modül olguları kapasite kısıtları da göz önünde bulundurularak mümkün mertebe aynı düğümlere yerleştirilmiştir.



Şekil 8-7 Trafik Benzetimi için Üretilmiş Yerleştirme Seçeneği

9 DEĞERLENDİRME

Otomatik yerleştirme modeli üretme aracının başarımı, seçilen İş atama algoritmasının başarımı ile yakından ilgilidir. İş atama algoritmaların başarımlarının doğrulanması ve karşılaştırılması literatürde çeşitli çalışmalar kapsamında detaylı olarak ele alınmıştır [Lo 1988; Ucar et al. 2005; Mehrabi et al. 2009]. Bu tez çalışması kapsamında geliştirilen *S-IDE* aracı, herhangi bir algoritmanın kullanılmasını zorunlu kılmaz, herhangi bir iş atama algoritması *S-IDE* ortamına entegre edilebilir. Bu çalışma kapsamında uygun yerleştirme modellerinin üretimi için örnek algoritma olarak Mehrabi et al. tarafından tanımlanmış olan genetik algoritma kullanılmıştır [2009].

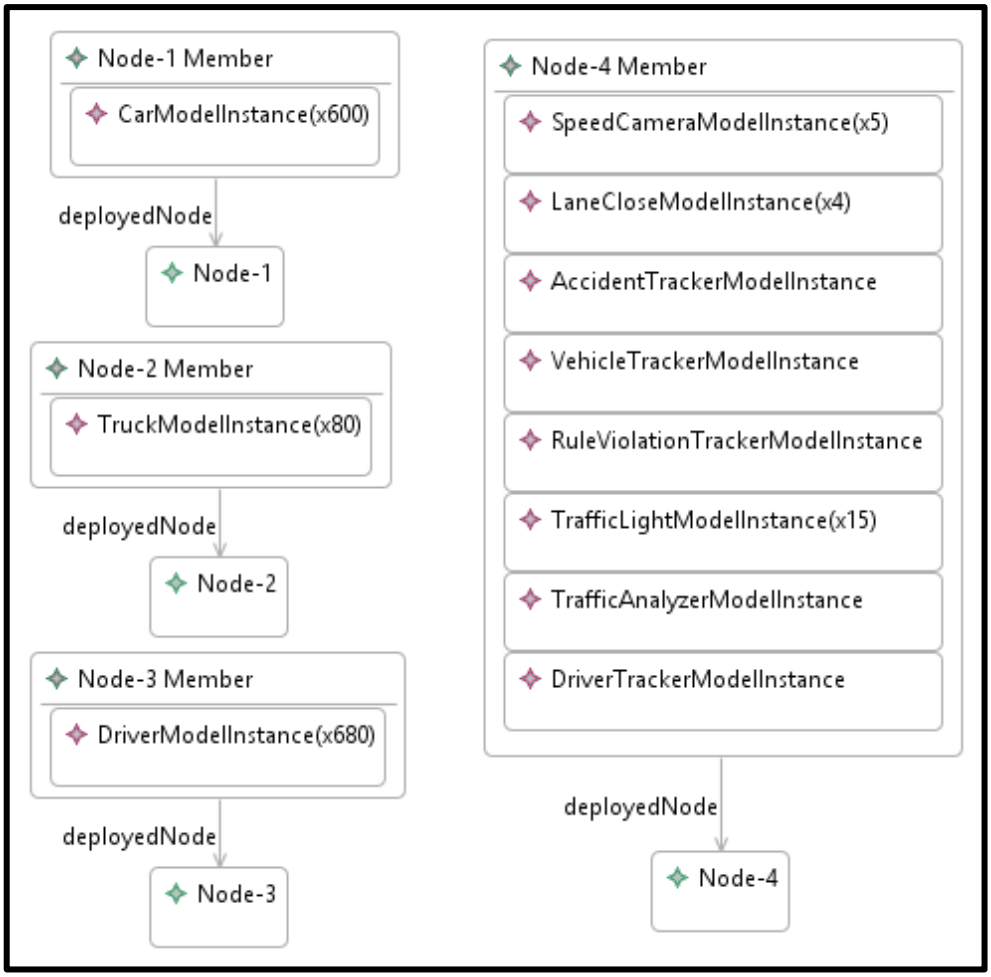
9.1 Üretilen Yerleştirme Modelinin Kalitesinin Değerlendirilmesi

Üretilen yerleştirme modelinin kalitesi değerlendirilirken iki farklı yaklaşım kullanılabilir.

Birinci yaklaşım, uzman incelemesine dayanan biçimsel olmayan bir yöntemdir. Bu yöntemde uzman üretilen yerleştirme modelini inceleyerek yerleşimin mantıklı olup olmadığına karar vermeye çalışır. Burada vurgulanması gereken nokta, yerleştirme alternatifinin uzman tarafından tanımlanmadığı ve otomatik olarak üretildiğidir. Elektronik Harp benzetimi örnek durum çalışması için otomatik olarak üretilen yerleştirme modeli, 7.2.1 kesiminde uzman gözüyle irdelenmiş ve yerleşimin mantıklı göründüğü kanaatine varılmıştır. Benzer bir incelemenin sonuçları trafik benzetimi örnek durum çalışması için 8.3.1 kesiminde verilmiştir.

Üretilen yerleştirme modelinin kalitesinin değerlendirilmesinde kullanılabilecek bir diğer yöntem, uzman incelemesine dayanan ilk yönteme göre daha biçimsel bir yöntem olan üretilen yerleştirme seçeneklerinin başka yerleştirme seçenekleriyle karşılaştırılmasıdır [Aleti et al. 2009a ; Malek et al. 2012]. *S-IDE* ortamı, iki yerleştirme seçeneğinin verilen benzetim çalıştırma konfigürasyonlarına göre karşılaştırılmasını sağlayan bir kalite değerlendirme aracı sağlar. Otomatik olarak üretilmiş bir yerleştirme modeli, (1) seçilen CTAP algoritmasının ürettiği diğer yerleştirme seçenekleriyle, (2) farklı CTAP algoritmalarının ürettiği yerleştirme seçenekleriyle, (3) uzman değerlendirmesi yöntemiyle el ile üretilmiş yerleştirme seçenekleriyle karşılaştırılarak göreceli kalitesi ölçülebilir. *S-IDE* ortamında sağlanan kalite değerlendirme süreci bu yöntemlerden herhangi birisiyle üretilmiş yerleştirme modellerinin karşılaştırılmasına olanak sağlar.

Yerleştirme modeli kalite değerlendirmesi örneği olarak, Şekil 8-7'de verilmiş olan trafik benzetimi için otomatik olarak üretilmiş yerleştirme modeli ile 8.1.2 kesiminde uzman değerlendirmesiyle tanımlanmış olan ilk örnek yerleştirme karşılaştırılacaktır. Karşılaştırma sürecinin ilk adımı olarak, uzman değerlendirmesiyle oluşturulmuş yerleştirme seçeneği için *S-IDE* ortamında el ile bir yerleştirme modeli oluşturulmuştur (Şekil 9-1). Şekilde görüldüğü gibi, uzman değerlendirmesine uygun olarak tüm araba modül olguları ilk düğüme, tüm kamyon modül olguları ikinci düğüme, tüm sürücü modül olguları üçüncü düğüme ve kalan modül olguları da dördüncü düğüme yerleştirilmiştir.



Şekil 9-1 Trafik Benzetimi için Uzman Değerlendirmesiyle Oluşturulmuş Yerleştirme Alternatifi

Otomatik üretilmiş yerleştirme modeli (Şekil 8-7) ile uzman değerlendirmesiyle oluşturulmuş yerleştirme modelinin (Şekil 9-1) karşılaştırması Çizelge 9-1'de verilmiştir.

Çizelge 9-1 Otomatik üretilmiş yerleştirme modeli ile uzman değerlendirmesiyle oluşturulmuş yerleştirme modelinin karşılaştırması

Modül	**Toplam İşletim Maliyeti**			**Toplam İletişim Maliyeti (MB/Saniye)**		
	Uzman Değ.	***S-IDE* ile Üretilmiş**	**İyileştirme (%)**	**Uzman Değ.**	***S-IDE* ile Üretilmiş**	**İyileştirme (%)**
DriverTrackerModelInstance (x1)	5,63	11,25	-100,00	0.0	0.0	0,00
TruckModelInstance (x80)	400	393,13	1,72	6,53	4,90	24,97
SpeedCameraModelInstance (x5)	6,25	8,50	-36,00	0,08	0,06	23,79
TrafficAnalyzerModelInstance (x1)	6,25	12,50	-100,00	0,00	0,00	0,00
CarModelInstance (x600)	2400	2331,00	2,88	29,34	22,00	25,03
RuleViolationTrackerModelInstance (x1)	5,63	9,00	-60,00	0,00	0,00	0,00
VehicleTrackerModelInstance(x1)	5,63	9,00	-60,00	0,00	0,00	0,00
AccidentTrackerModelInstance (x1)	6,25	10,00	-60,00	0,00	0,00	0,00
DriverModelInstance (x680)	4250	3317,50	21,94	0,11	0,08	25,59
LaneCloseModelInstance (x4)	7,50	10,50	-40,00	0,10	0,07	24,15
TrafficLightModelInstance (x15)	18,75	30,50	-62,67	0,18	0,13	25,13
Toplam Maliyet	7111,88	6142,88	**13,63**	36,34	27,25	**25,02**

Çizelge 9-1 her benzetim modül olgusu için işletim ve iletişim maliyetlerinin uzman değerlendirmesi ile oluşturulmuş yerleştirme modeli ile otomatik üretilmiş yerleştirme modelindeki değerlerini karşılaştırır. Çizelgenin sol sütununda, türlerine göre modül olguları listelenmiştir. Karşılaştırma için referans alınan benzetim çalıştırma konfigürasyonundaki modül olgusu sayısı her modül türü için parantez içinde gösterilmiştir. Örneğin, sol sütundaki TruckModelInstance (x80) ifadesi ilgili çalıştırma konfigürasyonunda 80 adet TruckModelInstance olduğunu gösterir. Toplam İşletim Maliyeti sütunu, uzman değerlendirmesi ve otomatik üretilmiş yerleştirme modelleri için her modül türünün toplam işletim maliyetini gösterir. Bu sütunda ayrıca üretilmiş yerleştirme modelinde sağlanan işletim zamanı iyileştirme yüzdesi gösterilmektedir. Benzer şekilde Toplam İletişim Maliyeti sütunu uzman değerlendirmesi ve otomatik üretilmiş yerleştirme modelleri için iletişim maliyeti değerlerini ve iyileştirme yüzdesini tanımlar.

Çizelgenin son satırında her yerleştirme seçeneği için toplam maliyetler ve iyileştirme yüzdeleri verilmiştir.

Çizelgede gösterildiği gibi, toplam işletim maliyeti ilgili örnek çalışma için % 13,63 oranında iyileştirilmiştir. Çizelgede verilen sonuçlarda ilginç olan bir nokta, bazı modül olguları (ör. *DriverTrackerModelInstance*, *TrafficAnalyzerModelInstance*) için işletim maliyetleri uzman değerlendirmesiyle oluşturulmuş yerleştirme modellerinde otomatik üretilen yerleştirme modeline göre daha iyidir. Bu durumun sebebi, yerleştirme modeli eniyileme yaklaşımının amacının toplam maliyetleri düşürmeyi hedeflemesidir. Bu amaca uygun olarak, toplam işletim maliyetleri yüksek olan *DriverModuleInstances*, *CarModuleInstances* ve *TruckModuleInstances* modül olgularının işletim maliyetleri sırasıyla % 21.94, % 2.88 ve % 1.72 oranında iyileştirilmiştir. Diğer türlerdeki modül olgularının işletim maliyetleri yükselse de, bu modüllerdeki iyileşme sayesinde işletim maliyetlerinde toplamda % 13,63 oranında iyileşme sağlanmıştır.

Toplam iletişim maliyeti, çizelgede gösterildiği gibi %25,02 oranında iyileştirilmiştir. Çizelgenin iletişim maliyetleri kesimi incelendiğinde, *DriverTrackerModelInstance* ve *TrafficAnalyzerModelInstance* gibi bazı modül olgularının aslında belirli veri modeli elemanlarına üye olmalarına rağmen toplam iletişim maliyetlerinin sıfır olduğu görülür. Bu durum, iletişim maliyetleri hesaplanırken çift hesap

yapılmaması için iletişim maliyetlerinin sadece veriyi yayımlayan modül olgularına yüklenmesinden kaynaklanmaktadır. Örneğin *DriverTrackerModelInstance* modül olgusu, *DriverModelInstance* tarafından yayımlanan *Driver* nesne sınıfına üye olur. Bu veri iletişiminin maliyeti sadece yayımlayıcı taraf olan *DriverModelInstance* modül olgusuna yüklenir. *DriverTrackerModelInstance* herhangi bir veri yayımlamadığı için iletişim maliyeti olarak sıfır değeri atanmıştır.

9.2 Otomatik Yerleştirme Modeli Üretme Aracının Başarımı

Uygun yerleştirme modeli üretme aracı Java programlama dili ile gerçekleştirilmiş ve denemeler 4 GB belleğe sahip olan Intel Core I-7 2.70 GHz 64-Bit bir bilgisayarda yapılmıştır. Farklı benzetim modülü olgusu ve düğüm sayıları için yerleştirme modeli üretme süreleri Çizelge 9-2'de verilmiştir. Bu çizelgedeki her satır için örnek senaryolar oluşturulmuş ve üretim süreleri ölçülmüştür.

Çizelge 9-2 Örnek Durumlar İçin Yerleştirme Modeli Üretim Süreleri

Benzetim No.	Benzetim	Benzetim Unsuru Sayısı	Düğüm Sayısı	Üretim Süresi (saniye)
1.	EH Benzetimi	5	4	20
2.	EH Benzetimi	17	4	50
3.	EH Benzetimi	81	4	182
4.	EH Benzetimi	141	5	325
5.	**EH Benzetimi**	**1596**	**6**	**360**
6.	Trafik Benzetimi	17	4	2
7.	Trafik Benzetimi	81	4	8
8.	**Trafik Benzetimi**	**1389**	**4**	**12115**
9.	Trafik Benzetimi	1389	12	4273

Çizelgedeki 5. satır, bu tez çalışması kapsamında tanımlanmış olan Elektronik Harp örnek durum çalışması için uygun yerleştirme modelinin üretilmesi için harcanan süreyi gösterirken, 8. Satır ise Trafik Benzetimi örnek durum çalışması için yerleştirme modeli üretim süresini göstermektedir.

Bu tez çalışması kapsamında geliştirilen yaklaşımın tasarım aşamasında uygulanacağı düşünülürse, Çizelge 9-2'de listelenen üretim sürelerinin kabul edilebilir rakamlar olduğu gözlemlenmektedir. Yaklaşık 1600 benzetim modülü içeren Elektronik Harp benzetimi örnek durum çalışması için 360 saniye gibi bir sürede uygun bir yerleştirme modeli üretilebilmiştir. Trafik benzetimi için elde edilen süreler, çok sayıda benzetim unsuru içeren durumlarda Elektronik Harp benzetimlerine nispeten oldukça yüksektir. Öncelikle bu sonuç tamamen seçilen CTAP algoritmasının karakteristiğiyle ilgilidir. Bu durum, trafik ve Elektronik Harp benzetimlerindeki işletim maliyeti ve iletişim karakteristiklerinin farklı olmasından kaynaklanmaktadır. Trafik benzetimi için ölçülen süreler nispeten yüksek olsa da, tasarım aşamasında kullanılacak bir araç için bu süreler kabul edilebilir rakamlardır.

Seçilen iş atama algoritmasının başarımı tasarım aşamasında kullanılacak bir yaklaşım için yeterli olsa da, Çizelge 9-2'de verilen zamanlar bazı durumlarda kabul edilebilir sınırların dışına çıkabilir. Örneğin bu tez çalışması kapsamında geliştirilen yaklaşım tasarım aşamasında değil de koşum zamanında (*runtime*) kullanılmak istenirse mevcut algoritmanın başarımı yetersiz kalacaktır.

Çizelge 9-2'de listelenen üretim süreleri incelenerek aşağıdaki çıkarsamalar yapılabilir:

1. Benzetimdeki modül olgusu sayısı arttıkça, yerleştirme modeli üretme süresi artmaktadır.

2. Benzetimdeki düğüm sayısı arttıkça, yerleştirme modeli üretme süresi azalmaktadır.

3. Özellikle çok sayıda modül içeren benzetimlerde modül olgularının iletişim ve işletim maliyeti karakteristikleri, yerleştirme modeli üretme süresini ciddi biçimde değiştirebilmektedir.

Bu tez çalışması kapsamında geliştirilen yaklaşım, daha önce de belirtildiği gibi herhangi bir İş Atama Algoritmasının kullanımını zorunlu kılmamaktadır. İhtiyaç durumunda kullanılan İş Atama Algoritmasının yerine farklı algoritmalar kullanılarak daha kısa sürede uygun yerleştirme modelini üretmek veya farklı kalite

ihtiyaçlarına göre yerleştirme modeli oluşturmak mümkün olabilir. Farklı iş atama algoritmalarının başarımları literatürde tanımlı yöntemler [ör. Aleti et al. 2009a; Aleti et al. 2009b] kullanılarak değerlendirilip farklı karakteristikteki çalıştırma konfigürasyonları için en iyi çalışan algoritma belirlenebilir.

10 TARTIŞMA

Bu tez çalışmasında benzetim ortamının tasarımı ve mevcut fiziksel kaynak tanımlarını kullanarak koşut ve dağıtılmış benzetim sistemleri için uygun yerleştirme seçeneklerini türemeyi hedefleyen somut bir yaklaşım ve bu yaklaşım için araç desteği sağlanmıştır.

Geliştirilen yaklaşım, aslında Dağıtık Benzetim Mühendisliği ve Çalıştırma Sürecinin *(Distributed Simulation Engineering and Execution Process- DSEEP)* 3. adımı olan Benzetim Ortamının Tasarlanması aşamasında yapılması önerilen iki işin gerçekleştirimini destekler. Bu iki iş, temel olarak benzetim sistemi tasarımının başarımının değerlendirilmesi ve tasarım seçeneklerinin değerlendirilmesi üzerine kurgulanmıştır. Geliştirilen yaklaşımın gerekliliği ve değeri, aynı zamanda IEEE standardı olan DSEEP'te tanımlanan bu işlere dayanmaktadır. Geliştirilen yaklaşım ve ilgili araçlar tasarımcıya erken tasarım aşamasında sistemin tasarım seçeneklerinin başarımının değerlendirilmesi ve en iyi başarım için uygun yerleştirme modelini otomatik olarak üretilmesi konularında destek sağlar.

Bu bağlamda geçerli bir soru, önerilen yerleştirme modeli üretme aracının iyileştirilmeye çalışılan kalite metrikleri anlamında uygun bir sonuç üretme konusunda ne kadar başarılı olduğudur. Bu konuda geliştirilen yaklaşımın doğruluğunu ve tutarlılığını arttırmak adına, çalışma kapsamında yeni bir yaklaşım geliştirmek yerine, literatürde iyi bilinen ve uzun yıllardır çalışılan Kapasite Kısıtlı İş Atama Problemi *(Capacitated Task Assignment Problem - CTAP)* çözme teknikleri yaklaşıma bütünleştirilmiştir. Literatürde Kapasite Kısıtlı İş Atama Problemlerini çözmek için çok çeşitli alternatif çözüm yöntemleri, hatta IBM CPLEX gibi ticari araçlar mevcuttur. Kapasite Kısıtlı İş Atama Problemi çözüm yöntemlerinin olgun doğasından faydalanmak adına, bu tez kapsamında geliştirilen yaklaşım bu yöntemlerin kullanılabilmesine yönelik olarak tasarımdan algoritmaya ve tersi yönde algoritmadan tasarıma geçişler içermektedir. Aynı zamanda, özel bir Kapasite Kısıtlı İş Atama Problemi çözüm yöntemine bağımlı olmamak için yaklaşım geliştirilirken problem tanımı matematiksel olarak soyutlaştırılmış ve tasarım ilgili çözüm algoritması başka algoritmalarla kolayca değiştirilebilecek şekilde yapılmıştır. Geliştirilen yaklaşım, Kapasite Kısıtlı İş Atama Problemi çözüm algoritmasına net olarak tanımlanmış bir girdi kümesi sağlar ve bunun karşılığında

algoritmanın yine biçimi net olarak tanımlanmış bir çıktı üretmesini bekler. Böylece kullanılan iş atama problemi çözüm yönteminin yaklaşımdan tamamen soyutlanmış bir kara kutu *(black box)* olarak kullanılması sağlanmıştır. Kapasite kısıtlı iş atama algoritmalarının başarımları literatürde mevcut olan çeşitli çalışmalarda kanıtlandığı dikkate alınırsa, bu tez kapsamında geliştirilen yaklaşımın uygun bir çözüm üreteceği söylenebilir.

Çeşitli kapasite kısıtlı iş atama problemi çözüm yöntemlerinin ve araçların başarımlarının karşılaştırılması literatürde çeşitli çalışmalarda ele alınan bir konudur [Ucar et al. 2005; Pirim 2006]. Literatürdeki bu gibi çalışmaların ışığında, geliştirilen sistemin ihtiyaçlarına göre farklı kalite bileşenlerini eniyilemek için değişik kapasite kısıtlı iş atama problemi çözüm yöntemlerinin kullanımı değerlendirilebilir.

Uygun yerleştirme seçeneklerinin türetilmesiyle ilgili akla gelebilecek başka bir soru, tasarlanan sisteme ve tanımlanan fiziksel kaynaklara göre herhangi bir yerleştirme seçeneği üretilememesi durumunda nasıl bir yol izlenmesi gerektiğidir. Kapasite kısıtlı iş atama problemi çözüm algoritmalarının bakış açısıyla, bu durumda yapılması gereken maliyet parametrelerinin (ör: çalıştırma maliyeti, iletişim maliyeti vs.) iyileştirilmesidir. Benzetim tasarımı bakış açısıyla yapılması gereken, bu maliyet parametrelerinin türetildiği benzetim tasarımının iyileştirilmesidir. Benzetim tasarımın iyileştirilmesi büyük ölçüde geliştirilen benzetim sisteminin gereksinimlerine bağımlı olsa da, bu tez çalışması kapsamında tasarımın iyileştirilmesine yönelik çeşitli genel kurallar tanımlanmıştır. *S-IDE* aracı, benzetim tasarımını analiz ederek tasarımda performans anlamında sıkıntı yaratabilecek kesimlerle ilgili geri besleme sağlar. Tasarım analizi ve geri besleme raporu üretmenin yanı sıra, *S-IDE* aracı farklı yerleştirme seçeneklerinin başarımını karşılaştırmaya olanak sağlar.

Bu kesime kadar ele alınan algoritmik performansın ötesinde, bu tez kapsamında organizasyon seviyesinde performans kazancına odaklanılmıştır. Mevcut uygulamalarda benzetim sistemlerinin yerleştirme modelinin üretilmesi ya uzman değerlendirmesi (*expert judgment*) ile yapılmakta ya da geliştirme aşamasına bırakılmaktadır. Uzman değerlendirmesi belirli bir problem büyüklüğüne kadar yeterli olabilse de, genel anlamda el ile yapıldığı için yeterli değildir. Bu tez

çalışması kapsamında geliştirilen yaklaşımla bir adım öteye gidilerek kapasite kısıtlı iş atama problemi çözüm yöntemleri erken tasarım aşamasına bütünleştirilmiş ve uzmanların yerleştirme modeline karar verme süreci sistematik bir yaklaşımla desteklenmiştir. 4. Kesimde problem tanımı verilirken de belirtildiği gibi, yerleştirme modelinin üretilmesi işinin tasarım aşamasında yapılmayıp ürün gerçekleştirim aşamasına bırakılması kötü gerçekleştirimlerin ortaya çıkmasına neden olabilir. Ortaya çıkan kötü gerçekleştirimler, ürün geliştirme sürecinde mimari tasarım, detaylı tasarım, gerçekleştirim, sınama elemanları ve belgeleme gibi yaşam döngüsü elemanlarının gereksiz yere güncellendiği döngülerine sebep olur. Tüm yazılım projelerinde olduğu gibi koşut ve dağıtılmış benzetim sistemi geliştirme projelerinde de bu gereksiz döngüler gecikmelere ve maliyetlerin yükselmesine neden olur. Aslında DSEEP'in tasarım alternatiflerinin karşılaştırılması ve performanslarının ölçülmesi işlerinin erken tasarım sürecinde yapılmasını önermesinin altında yatan temel nedenin de bu önemli riskler olduğu rahatlıkla söylenebilir.

Tasarım aşamasında benzetim modül olgularının çalıştırma maliyetleri ve bellek gereksinimleri tahmin edilir, iletişim maliyetleri ise tanımlanan yayımlama/üye olma ilişkileri ve güncelleme sıklıklarına göre hesaplanır. Çalıştırma maliyeti ve bellek gereksinimleri ne kadar iyi tahmin edilirse üretilen yerleştirme modeli de o kadar uygun olacaktır. Tahmin değerleri daha önce geliştirilmiş benzer modeller incelenerek veya modellerin prototipleri geliştirilerek iyileştirilebilir. Bu aşamada yapılacak maliyet tahminleri en kötü duruma göre yapılabileceği gibi, tasarımcının tercihine göre ortalama gibi farklı maliyetler de kullanılabilir. Örneğin, belirli bir veriyi farklı sıklıkta güncelleyen bir benzetim modülü için işletim maliyeti hesaplanırken olası en sık güncelleme referans alınabilir. Bu bağlamda önemli olan, seçilen yöntemden ziyade (ortalama durum, en kötü durum, vs.), yöntemin tüm tahmin sürecinde tutarlı bir yaklaşım kullanılması önemlidir. Bu tez çalışması kapsamında benzetim modüllerinin çalıştırma maliyetlerini hesaplamak için biçimsel yöntemlerin kullanılması ele alınmamıştır. Örneğin, benzetim modüllerinin çalıştırma maliyetlerinin hesaplanmasında Genişletilmiş Kuyruk Ağı (*Extended Queuing Network - EQN*) modeli gibi biçimsel yöntemler kullanılabilir [Gianni et al. 2010]. Proje yaşam döngüsünün ilerleyen aşamalarında sistem şekillendikçe daha net bilgilere dayanarak çalıştırma maliyeti tahminleri güncellenebilir ve buna göre

uygun yerleştirme modeli iyileştirilebilir. Geliştirilen yaklaşımın hedeflenen temel kullanım süreci erken tasarım aşaması olsa da, yaşam döngüsünün herhangi bir aşamasında, hatta başarım iyileştirme amacıyla sistem geliştirmesi tamamlandıktan sonraki aşamada bile kullanılabilir. Bu tez kapsamında önerilen yaklaşımın sistem geliştirmesi tamamlandıktan sonraki aşamada kullanılması durumunda, benzetim modül olgularının tahmini maliyet değerleri yerine ölçülmüş maliyetleri kullanılarak tasarım aşamasına göre daha net olması beklenen uygun bir yerleştirme seçeneği bulunabilir. Benzetim modül olgularının çalıştırma maliyeti ve bellek gereksinimlerini koşum zamanında ölçmek için sistem koşumundan veri toplanması gerekmektedir. Farklı benzetim mimarileri için değişik veri toplama yöntemleri mevcuttur. Örneğin HLA bağlamında koşum zamanında veri toplamak için Yönetim Nesne Modeli *(Management Object Model – MOM)* servisleri [IEEE 2010b] kullanılabilir.

Bu tez kapsamında geliştirilen yaklaşım ile üretilen yerleştirme modelleri, 9. Kesimde anlatıldığı gibi alternatif yerleştirmeler ile karşılaştırılarak değerlendirilmiştir. Üretilen yerleştirme seçeneğinin başarımının benzetim sistemi geliştirilerek ölçülmesi bu tez çalışmasının kapsamı dışında tutulmuştur.

Bu tez çalışması kapsamında geliştirilen yaklaşımın uygulanabileceği hedef mimariler yayımla/üye ol modelini uygulayan HLA ve TENA gibi koşut ve dağıtılmış benzetim mimarileridir. Bunun yanında geliştirilen yaklaşım, benzetim sistemleri dışında yayımla/üye ol modelini uygulayan OMG DDS gibi mimarilere de uyarlanabilir [OMG 2006b]. Şekil 5-1'de verilen uygun yerleştirme türetme akışı, yaklaşım değişik mimarilere uyarlanırken ana hatlarıyla aynı kalacak, yaklaşımın adımlarında ve gerçekleştirimlerinde göreceli olarak küçük uyarlamalar yapılacaktır. Örneğin veri değişim modeli hedef mimarinin veri değişim modeli şablonlarına göre güncellenmeli ve buna bağlı olarak yayımlama/üye olma ilişkileri modeli gibi veri modeline bağımlı modeller güncellenmelidir. Bu konu gelecek çalışmalar kapsamında daha detaylı olarak ele alınacaktır.

Bu tez çalışması kapsamında yaklaşım ve araç ailesi geliştirilirken, HLA, DIS gibi belirli bir standarda özel bir çözüm geliştirmek yerine koşut ve dağıtılmış benzetim sistemleri için genel bir çözüm üretilmesi hedeflenmiş olsa da, benzetim alanında kabul görmüş ve yaygın olarak kullanılan yaklaşımlar göz ardı edilmemiştir. Bu

kapsamda, özellikle HLA ve DEVS gibi standartlardaki kavram ve yaklaşımlar tez çalışmasında ilgili kesimlerde referans olarak alınmıştır.

Bu tez çalışmasında geliştirilen yaklaşımın çıkış noktası endüstride büyük ölçekli dağıtılmış benzetim sistemleri geliştirilirken karşılaşılan gerçek problemlere dayanmaktadır [Çelik et al. 2012]. Benzer şekilde, yaklaşımı ve geliştirilen araçları doğrulamak için örnek durum çalışması olarak az katılımcılı, nesne modeli dar bir çalışma yerine, endüstride karşılığı olabilecek geniş ölçekli bir Elektronik Harp Benzetimi ve bir trafik benzetimi tanımlanmıştır.

Erken tasarım aşamasında uygun bir yerleştirme seçeneği üreterek koşut ve dağıtılmış benzetim sistemlerinin başarımını iyileştirmeye çalışmak yaklaşım olarak yeni olsa da, benzetim bileşenleri arasındaki iletişim maliyetlerini düşürmeye yönelik literatürde tanımlanmış çeşitli yaklaşımlar mevcuttur. Örneğin, HLA standardının bir parçası olan Veri Dağıtımı Yönetimi *(Data Distribution Management – DDM)* [IEEE 2010a], geniş ölçekli koşut ve dağıtılmış benzetimlerde koşum zamanında benzetim katılımcıları arasındaki iletişimi eniyilemek için kullanılan bir grup servis sağlamaktadır. Koşut ve dağıtılmış benzetim sistemlerinde DDM kullanıldığı durumda koşum zamanında katılımcılar arasındaki iletişim örüntüleri doğal olarak değişebilir. DDM katılımcılar arasındaki iletişimi eniyilerken katılımcılar arasında paylaşılan veri nesnelerinin niteliklerinin koşum zamanındaki değerlerine göre dinamik olarak ilgili verinin katılımcıya iletilip iletilmeyeceğine karar verir. Özetlenecek olursa, DDM koşum zamanında kullanılmakta ve veri değişim modeli nesnelerinin koşum zamanı özelliklerine ihtiyaç duymaktadır. DDM'in aksine, bu tez çalışması kapsamında net olarak tasarım aşamasında uygun yerleştirme seçeneklerinin türetilmesine odaklanılmıştır ve bu aşamada henüz veri değişim modeli nesnelerinin niteliklerinin değerleri bilinmediğinden yaklaşımda koşum zamanı veri değerlerine bağımlı hiçbir adım yoktur. Bu tez kapsamında geliştirilen yaklaşım ile DDM farklı aşamalarda ve farklı mantıklarla çalışırlar ama bu iki yaklaşım birbirlerinin alternatifi değildir. Bu tez kapsamında mimari tasarım seviyesinde iyileştirilmiş bir benzetim sisteminde koşum zamanında DDM uygulanarak katılımcılar arasındaki iletişim anlamında ilave iyileştirme sağlanabilir. DDM tanımlamaları bu tez kapsamında geliştirilen yaklaşımla bütünleştirilerek mevcut duruma nispeten daha

bütünleşik bir eniyileme yaklaşımı geliştirmek mümkündür. Bu konu gelecek çalışmalar kapsamında daha detaylı olarak ele alınacaktır.

Geliştirilen yaklaşımın ölçeklendirilebilirliği, benzetim çalıştırma konfigürasyonundaki modül olgularının sayısı, yayımlama/üye olma özelliklerine, tanımlanan fiziksel kaynaklara ve kullanılan iş atama algoritmasına göre değerlendirilebilir. Bu tez kapsamında seçilen algoritma, tanımlanan örnek durum çalışması için kabul edilebilir bir sürede uygun bir yerleştirme modeli üretebilmektedir (Çizelge 9-2). Çeşitli algoritmalar ve araçların yaklaşımla bütünleştirilerek yaklaşımı daha da iyileştirme konusu da gelecek çalışmalar kapsamında ele alınabilecek bir diğer başlıktır.

11 İLİŞKİLİ ÇALIŞMALAR

Bu kesimde literatürdeki ilişkili çalışmalar aşağıda sıralanan dört farklı bakış açısıyla ele alınacaktır:

(1) Koşut ve dağıtılmış benzetimlerin tasarlanması için tanımlanmış metamodeller,

(2) Model Güdümlü Mühendislik Yaklaşımları,

(3) DSEEP adımlarını destekleyen yaklaşım ve araçlar.

(4) Yerleştirme Modellerinin Eniyilenmesine Yönelik Yaklaşımlar

11.1 Koşut ve Dağıtılmış Benzetimlerin Tasarlanması için Tanımlanmış Metamodeller

Literatürde koşut ve dağıtılmış benzetimleri modellemek için geliştirilmiş çeşitli metamodeller vardır.

BOM (Base Object Model) Belirtimi [SISO 2006], benzetim sistemlerinin kavramsal modellerini tasarlamak için anlamsal *(semantic)* ve sözdizimsel *(syntactic)* tanımlamalar yapar. BOM kullanılarak tasarlanan kavramsal modeller, benzetim tasarımlarında, geliştirme aşamasında ve birlikte çalışabilir benzetim sistemlerinin genişletilmesinde kullanılabilir. BOM kavramsal modellerinin temel hedeflerinden birisi de kavramsal seviyede tekrar kullanılabilirlik *(reusability)* sağlamaktır.

Bir BOM, gerçek dünyadaki unsurları temsil eden kavramsal modeller ve kavramsal olaylardan oluşan platform bağımsız bir modeldir. Kavramsal modeller ve olayların ötesinde, BOM tanımlamaları benzetim unsurları arasındaki etkileşimleri birlikte koşma örüntüleri *(patterns of interplay)* ve durum makineleri *(state machines)* kullanarak ifade eder. Bir BOM tanımının diğer bir parçası da, arayüz tanımlamasıdır. Arayüz tanımlaması aslında bir nesne modeli tanımlamasıdır ve BOM belirtimi oluşturulurken yeni bir nesne modeli belirtimi tanımlamak yerine arayüz tanımlamaları için HLA Nesne Modeli Şablonunun *(Object Model Template - OMT)* [IEEE 2010c] kullanılmasına karar verilmiştir. Aslında bu tez çalışması kapsamında Benzetim Veri Değişim Modeli tasarlama

yöntemi tanımlanırken de benzer bir yaklaşım izlenmiş, HLA OMT belirtimi genişletilerek kullanılmıştır.

HLA OMT belirtimi, temel olarak nesne sınıfları, etkileşim sınıfları ve veri türü tanımlamalarından oluşur. Veri türü tanımlamaları, nesne ve etkileşim sınıflarının nitelikleri ve parametrelerini tanımlamak için kullanılır. Literatürde HLA OMT belirtimine uygun veri modelleri oluşturmak için tanımlanmış çeşitli metamodeller ve araçlar mevcuttur [Parr 2003; Çetinkaya and Oguztüzün 2006].

Tanımlanan kavramsal modellere (örneğin BOM ile oluşturulmuş olanlar) dayanarak benzetim sistemleri doğrudan UML [Fowler 2003] kullanılarak veya UML profilleri [Çelik 2005; Guiffard et al. 2006] kullanılarak tasarlanabilir.

FAMM çalışmasında [Topçu et al. 2008], Canlı Ardıl İşlem Çizenekleri *(Live Sequence Charts – LSC)* [Brill et al. 2004] kullanılarak HLA uyumlu benzetim sistemlerindeki federelerin davranışlarını modellemeye olanak sağlayan bir mimari tanımlanmıştır.

11.2 Model Güdümlü Mühendislik Yaklaşımları

Tolk, benzetim sistemlerinin geliştirilmesi sürecinde model güdümlü mühendislik yaklaşımların otomasyon amaçlı kullanılmasını tavsiye etmektedir [2002]. Bu bağlamda benzetim sistemlerinin geliştirilme sürecinde kullanılabilecek model güdümlü mühendislik yaklaşımlarından birisi Model Güdümlü Mimari yaklaşımıdır [Schmidt 2006; Frankel et al. 2004].

MDA yaklaşımında Platform Bağımsız Modeller *(Platform Independent Model – PIM)* ve Platforma Özel Modeller *(Platform Specific Model – PSM)* ayrılır. Platform Bağımsız bir model, sistemin gerçekleştirileceği platformdan soyutlanmış bir model sağlar. Platforma Özel Modeller ise, Platform Bağımsız Modellerin ilgili hedef platforma özel olarak detaylandırılması ve dönüştürülmesiyle oluşturulurlar. Platforma Özel bir Model, ilgili Platform Bağımsız Modelin tam veya yarı otomatik yöntemlerle dönüştürülmesi suretiyle oluşturulur. Benzer biçimde, Platforma Özel Modellerden hedef programlama ortamına özel otomatik kod üretimi sağlanabilir.

Model Güdümlü Mimari bakış açısıyla, bu tez çalışması kapsamında geliştirilen metamodeller benzetim sistemlerinin tasarımını HLA, DIS ve TENA gibi

mimarilerden soyutlayan Platform Bağımsız Modellerin tanımlanmasında kullanılabilir. Bu tez çalışması kapsamında geliştirilen tüm metamodeller platform bağımsızdır ve bu metamodel tanımları kullanılarak Platforma Özel Modeller otomatik veya yarı otomatik yöntemlerle türetilebilir.

11.3 DSEEP Adımlarını Destekleyen Yaklaşım ve Araçlar

Literatürde, DSEEP adımlarını desteklemek için kullanılabilecek kavramsal model geliştirme yaklaşımları, araç desteği ve benzetim sistemi tasarlama yaklaşımları gibi çeşitli çalışmalar mevcuttur. Bu çalışmalar, hedefleri / adımları desteklemek anlamında oldukça faydalı olsa da, yerleştirme modeli seçeneklerinin üretilmesi ve değerlendirilmesi için yeterli ve net destek henüz sağlanmamıştır.

Bu tez çalışması kapsamında geliştirilen yaklaşım, DSEEP'in 3. Adımı olan "Benzetim Ortamını Tasarlama" aşamasını destekler.

[Karagöz and Demirörs 2007]'de anlatılan çalışmada, DSEEP'in 1. Adımı olan "Benzetim Ortamı Amaçlarını Tasarlama" ve 2. Adım olan "Kavramsal Modelleme" adımlarını destekleyen bir yaklaşım geliştirmiştir.

BOM belirtimi [SISO 2006] DSEEP'in 2. Adımındaki kavramsal modelleme sürecini destekler.

HLA Object Model Metamodel – HOMM [Çetinkaya and Oguztüzün 2006] ve Federation Architecture Metamodel- FAMM [Topçu et al. 2008] çalışmaları DSEEP'in 3. Adımı olan "Benzetim Ortamının Tasarlanması" adımını destekler.

VR Forces [VT MAK 2010b], VR Vantage XR [VT MAK 2010c], ve [Adak et al. 2010] çalışmasında tanımlanan otomatik kod üretme aracı DSEEP'in 4. Adımı olan "Benzetim Ortamının Geliştirilmesi" adımını destekler.

[Parr 2003; Guiffard et al. 2006] çalışmalarında anlatılan kod üretimi ve tasarım araçları DSEEP'in hem 3. hem de 4. Adımlarını destekler.

Pitch Commander aracı [Pitch Technologies 2009] ve *MAK RTI* [VT MAK 2010d] ürününün *RTI Spy* bileşeni DSEEP'in 5. Adımı olan, "Benzetim Ortamını *Planla, Bütünleştir ve Sına*" adımını destekler.

Çelik et al. DSEEP'in 3, 4, 5, 6 ve 7. Adımlarını destekleyen bir uygulama çatısı tanımlar [2012]. Bu uygulama çatısında eksik olan yeteneklerden bir tanesi, bu tez kapsamında ele alınan uygun yerleştirme seçeneklerinin otomatik olarak üretilmesidir.

Çizelge 11-1'de DSEEP'i destekleyen metamodel tabanlı çeşitli yaklaşımlar ile bu tez çalışması kapsamında geliştirilen *S-IDE* ortamı desteklenen DSEEP adımları, ilgili metamodeller, uygulanan model dönüştürümleri ve yerleştirme seçeneklerinin otomatik olarak üretilmesini destekleme açılardan karşılaştırılmıştır. Çizelgede de gösterildiği gibi, *S-IDE* ortamı dışında yerleştirme seçeneklerinin otomatik olarak üretilmesini destekleyen herhangi bir yaklaşım yoktur.

Çizelge 11-1 Metamodel Tabanlı İlişkili Çalışmaların Karşılaştırılması

İlgili Çalışma	**Desteklenen DSEEP Adımları**	**İlgili Metamodeller**	**Model Dönüştürümleri**	**Yerleştirme Seçeneklerinin Otomatik Üretimi**
KAMA [Karagöz and Demirörs 2007]	1, 2	KAMA MM	Yok	Hayır
BOM (Base Object Model) [SISO 2006]	2	BOM MM	Yok	Hayır
Federation Architecture Metamodel-FAMM [Topçu et al. 2008]	3	FAMM	Yok	Hayır
FAMM based code generation tool [Adak et al. 2010]	4	FAMM	FAM'dan uygulama kodu üretimi (Java)	Hayır
HLA Object Model Development Tool [Çetinkaya and Oguztüzün 2006]	3	HLA OMT	Yok	Hayır
A Visual Tool to Simplify the Building of Distributed Simulations Using HLA [Parr 2003]	3, 4	HLA OMT UML MM	Benzetim Modeli (PIM) > Özelleştirilmiş Benzetim Modeli (PSM) > Uygulama Kodu	Hayır
An HLA-Based Tactical Environment Application Framework [Çelik et al. 2012]	3, 4, 5, 6, 7	HLA OMT	HLA OMT'den uygulama kodu üretimi (Java)	Hayır
S-IDE Ortamı	3	Bkz. 5. Kesim	Bkz. 6. Kesim	Evet

11.4 Yerleştirme Modellerinin Eniyilenmesine Yönelik Yaklaşımlar

Bu tez çalışması kapsamında geliştirilen yaklaşım, koşut ve dağıtılmış benzetim sistemlerinin yerleştirme modellerinin eniyilenmesinde kullanılabilir. Benzetim alanında bu yaklaşım yeni olsa da, farklı alanlarda benzer yöntemlerle yerleştirme modellerinin eniyilenmesine yönelik çalışmalar mevcuttur. Bu ilişkili çalışmaların temel motivasyonu olan "Yerleştirme mimarilerinin eniyilenmesine yönelik biçimsel yaklaşımlar tanımlamak" bu tez çalışmasının hedefleriyle örtüşmektedir.

Kugele et al., gömülü sistemlerin yerleştirme modellerinin eniyilenmesi için işlevsel olmayan gereksinimleri kullanan bir yöntem tanımlar [2009]. Kugele et al., bu tez çalışmasında "uzman değerlendirmesi" olarak adlandırılan işleri işlemcilere el ile atama yaklaşımını "kara büyü" olarak nitelendirmektedir. Bu çalışmada tanımlanması gereken işlevsel olmayan nitelikler *İşlem Gücü*, *Bellek* ve *Güç Durumu*'dur. İşlem gücü ve bellek nitelikleri bu tez çalışması kapsamında ele alınan problemle örtüşse de, gömülü sistemlere has bir kısıt olan güç durumu niteliği bu tez çalışması kapsamında kullanılmamaktadır. Kugele et al., bu tez çalışmasında tanımlanan yaklaşıma benzer olarak problemi bir eniyileme problemi olarak ifade eder [2009]. Bu çalışmada gerekli girdi parametreleri işlevsel olmayan gereksinimlerden türetilirken, bu tez çalışmasında girdi parametreleri benzetim tasarımından türetilir. Kugele et al. [2009], eniyileme problemini çözmek için tamsayı doğrusal programlama (integer linear programming- ILP) yöntemini kullanırken, bu tez çalışmasında tanımlanan yaklaşımda sabit bir eniyileme çözüm algoritmasının kullanılması zorunlu değildir, örnek bir gerçekleştirim olarak genetik bir algoritma kullanılmıştır.

Zheng et al., otomotiv endüstrisinde sıkı gerçek zamanlı bir ortamlarda iş yerleşimini ve sinyallerin mesaj paketlerine atanmasını eniyilemek için bir yaklaşım tanımlamıştır [2007]. Bu çalışmada eniyileme problemi toplam gecikmeyi asgariye indirecek (1) iş – işlemci ataması, (2) sinyal – mesaj ataması, (3) iş ve mesaj öncelikleri değerlerini sinyal gecikmesi ve mesaj boyutu kısıtlarını dikkate alarak bulma şeklinde tanımlanmıştır. Zheng et al. [2007] eniyileme problemini çözmek için CPLEX [IBM 2010] ortamında Karma Tamsayılı Doğrusal Programlama *(Mixed Integer Linear Programming - MILP)* yöntemlerini kullanmıştır.

Aleti et al. [2009a], iş-işlemci atamalarının eniyilemesinde yaygın olarak kullanılan genetik algoritmalar gibi yinelemeli yöntemler yerine yapısal (*constructive*) algoritmaların kullanımını gömülü sistemler alanında denemiştir. Yapısal algoritmalar, yinelemeli algoritmalara göre daha çabuk ve farklı sonuçlar üretir [Blum and Roli 2003]. Aleti et al. [2009a], Pareto-Ant Colony Optimization (P-ACO) algoritmasını [Doerner et al. 2004] çok amaçlı (*multi-objective*) bir yerleştirme eniyileme problemini çözmek için kullanmıştır. Çalışmada P-ACO algoritmasının performansı çok amaçlı bir genetik algoritma ile Archeopterix ortamında [Aleti et al. 2009b] karşılaştırılmıştır. Aleti et al. [2009a], eniyileme probleminin parametreleri olarak bellek gereksinimi, iletişim sıklıkları, mesaj boyutları, bellek kapasiteleri, ağ bant genişliği ve ağ gecikmesini kullanmaktadır. Bu parametrelerin çoğu bu tez çalışması kapsamında tanımlanan parametrelerle uyuşmaktadır. Bu tez çalışması kapsamında tanımlanan yaklaşımdan farklı olarak, Aleti et al. işlemci gücü parametresini tanımlamamış ve ağ bant genişliği ile ağ gecikmesi parametrelerini tanımlamıştır [2009a].

Malek et al. [2012] dağıtık sistemlerin yerleştirme mimarilerinin eniyilenmesi için genişletilebilir bir uygulama çatısı *(Deployment improvement framework - DIF)* tanımlar. Bu tez çalışması ve diğer benzer çalışmalardan farklı olarak, Malek et al. eniyileme problemi için sabit bir parametre kümesi (iletişim maliyeti ve işlem maliyeti gibi) tanımlamaz, parametrelerin tanımlanmasını da kullanıcıya bırakır [2012]. Bu çalışmanın genişletilebilir olarak nitelendirilmesi bu özelliğe dayanmaktadır. Malek et al. dört değişik eniyileme algoritmasını uygulama çatısı kapsamında gerçekleştirmiş ve bu algoritmalarda çeşitli iyileştirmeler yapmıştır [2012]. Malek et al. çalıştıkları en büyük senaryolarında yüzlerce yazılım bileşeni ve servis olduğunu belirtmektedir [2012]. Bu tez çalışmasında ise, benzetim alanının ihtiyaçlarından dolayı binlerce benzetim modeli ile çalışılması gerekmektedir.

Literatürde farklı alanlarda, farklı hedef mimariler için geliştirilmiş, farklı eniyileme parametreleri ile farklı çözüm yöntemleri sağlayan başka çalışmalar da mevcuttur [Metzner and Herde 2006; Matic et al. 2006].

12 SONUÇLAR

Koşut ve Dağıtılmış Benzetim Sistemlerinin gerçekleştirmesini sağlamak için literatürde HLA, DIS ve TENA gibi çeşitli altyapılar tanımlanmıştır. Koşut ve Dağıtılmış Benzetim Sistemlerinin geliştirilmesi sürecinde karşılaşılan önemli problemlerden birisi benzetimi oluşturan katılımcı uygulamaların mevcut kaynaklara uygun biçimde yerleştirilmesidir. Genellikle katılımcı uygulamalar mevcut kaynaklara çok farklı biçimlerde yerleştirilebilirler.

Bu tez çalışmasında tasarım aşamasında benzetim sistemi ve mevcut fiziksel kaynaklara göre benzetim sisteminin uygun yerleştirme modelini üreten somut bir yaklaşım geliştirilmiştir. Geliştirilen yaklaşım, tasarım aşamasında benzetim sisteminin başarımını ve tasarım alternatiflerini değerlendirmeyi öneren Dağıtık Benzetim Mühendisliği ve Çalıştırma Süreci olan DSEEP'e bütünleştirilmiştir.

Geliştirilen yaklaşımı anlatmak için bir Elektronik Harp Benzetim Sisteminin tasarlanması ve gerçekleştirilmesini konu alan gerçekçi bir örnek durum çalışması tanımlanmıştır. İkinci bir örnek durum çalışması olarak benzer şekilde bir trafik benzetimi tanımlanmıştır.

Geliştirilen yaklaşımın uygulanabilmesi için Benzetim Veri Değişim Modeli, Benzetim Modülleri ve Yayımlama/Üye Olma İlişkileri Modeli, Fiziksel Kaynak Modeli, Benzetim Çalıştırma Konfigürasyonu Modeli ve Yerleştirme Modeli olmak üzere toplam beş farklı model tanımlanmış ve bu modeller için metamodeller üretilmiştir.

Geliştirilen yaklaşımda uygun yerleştirme modelinin türetilmesi için Kapasite Kısıtlı İş Atama Problemi çözüm yöntemleri kullanılmıştır. İş Atama algoritmalarının girdi parametrelerinin tanımlanan benzetim modellerinden türetilmesi ve algoritma çıktısından yerleştirme modelinin üretilmesi için yöntemler geliştirilmiştir. Yerleştirme seçeneklerinin üretim sürelerinin tasarım aşamasında kullanılan bir yaklaşım için kabul edilebilir olduğu gözlemlenmiştir. Tez çalışması kapsamında geliştirilen yaklaşım, üretilen yerleştirme seçeneklerinin diğer yerleştirme seçenekleriyle karşılaştırılarak kalitesinin ölçülmesini destekler.

Geliştirilen yaklaşımın araç desteği olmadan yetersiz olacağı açıktır. Bu bağlamda benzetim ortamının tasarlanması, iş atama algoritmasının girdi parametrelerinin

tasarımdan otomatik olarak türetilmesi, algoritmanın çıktısına göre yerleştirme modelinin otomatik olarak üretilmesi ve üretilen yerleştirme modellerinin karşılaştırılmasını destekleyen *S-IDE* (Simulation-IDE) adında bütünleşik bir geliştirme ortamı tasarlanmış ve geliştirilmiştir [SIDE 2012]. Tez çalışması kapsamında, büyük ölçekli bir Elektronik Harp benzetimi ve bir trafik benzetimi için örnek durum çalışmaları *S-IDE* geliştirme ortamı kullanılarak tasarlanmış ve yerleştirme modelleri başarıyla üretilmiştir. *S-IDE* geliştirme ortamının, daha büyük örnek çalışmaları da destekleyeceği değerlendirilmektedir.

Gelecek çalışmalar kapsamında öncelikle aşağıdaki konular irdelenecektir:

1. Geliştirilen yaklaşım, bu tez çalışması kapsamında tanımlanmış olan çeşitli metamodellerin üzerine kurulmuştur. Gelecek çalışmalar kapsamında bu metamodelleri temel alarak benzetim geliştirme sürecinin otomasyonu konusuna odaklanılacaktır. Örneğin, HLA mimarisi için otomatik kod üretme konusu ele alınacaktır. Bu çalışma kapsamında benzetim modüllerinin kodlarının mümkün mertebe otomatik üretilmesi ve benzetim veri değişim modeli elemanlarının hedef platforma eşlenmesi ele alınacaktır.

2. Yaklaşıma benzetim sistemlerinin dinamik özelliklerini de tasarlayabilmek için davranışsal modelleme yeteneklerinin eklenmesi ele alınacaktır.

3. Geliştirilen yaklaşımın koşut ve dağıtılmış benzetimler dışında gerçek zamanlı sistemlere ve OMG DDS gibi farklı teknolojilere uyarlanması ele alınacaktır.

4. Bu tez kapsamında geliştirilen yaklaşım ve araç ailesi tasarım aşamasında kullanılmak üzere geliştirilmiştir. Gelecek çalışmalar kapsamında geliştirilmesi tamamlanmış bir sistemden koşum zamanında veri toplanarak üretilen yerleştirme seçeneğinin uygunluğunun değerlendirilmesi ele alınacaktır. Bu çalışma kapsamında gerekli durumlarda sistem tasarımın otomatik olarak güncellenmesi ve yeniden uygun yerleştirme konfigürasyonunun bulunması da ele alınacaktır.

5. Bu tez çalışması kapsamında benzetim modüllerinin kaynak gereksinimlerinin tahmin edilmesi için biçimsel bir yöntem

tanımlanmamıştır. Gelecek çalışmalar kapsamında biçimsel başarım kestirim yöntemlerinin geliştirilen yaklaşıma uyarlanması ele alınacaktır.

6. Bu tez çalışması kapsamında geliştirilen yaklaşım ve araç desteği herhangi bir CTAP çözüm algoritmasına bağımlı değildir. Çalışma kapsamında örnek CTAP çözüm algoritması olarak bir genetik algoritma kullanılmıştır. Gelecek çalışmalar kapsamında farklı CTAP çözüm algoritmalarının *S-IDE* ortamına entegre edilerek üretilen yerleştirme seçeneklerinin başarımlarının karşılaştırılması ele alınacaktır.

KAYNAKLAR DİZİNİ

ADAK, M., TOPÇU, O. AND OGUZTÜZÜN, H. 2010. Model-based code generation for HLA federates. *Software: Practice and Experience 40,* 2, 149-175.

ADAMY, D. 2001. *EW 101: A First Course in Electronic Warfare. 1st ed.* Artech House.

ADAMY, DAVID L. 2006. *Introduction to Electronic Warfare Modeling and Simulation.* SciTech Publishing.

ALETI, A., GRUNSKE, L., MEEDENIYA, I., MOSER, I. 2009a. Let the Ants Deploy Your Software - An ACO Based Deployment Optimisation Strategy. In *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering (ASE '09)*. IEEE Computer Society, Washington, DC, USA, 505-509.

ALETI, A., BJORNANDER, S., GRUNSKE, L., AND MEEDENIYA, I. 2009b. Acheopterix: An extendable tool for architecture ptimisation of AADL models. In *Model-Based Methodologies for Pervasive and Embedded Software, 2009. MOMPES '09.* ICSE Workshop. May 2009, 61-71.

ANISZCZYK, C. AND GALLARDO, D. 2007. Getting Started With The Eclipse Platform. IBM DeveloperWorks.

ATL. 2012. Eclipse Atlas Transformation Language Project Web Site. http://www.eclipse.org/atl/. Eclipse Foundation.

BEZIVIN, J. 2005. On the Unification Power of Models. *Journal of Software and System Modeling 4,* 171-188.

BLUM, C. AND ROLI, A. 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison, *ACM Comput. Surv. 35,* 3 (September 2003), 268–308.

BRILL, M., DAMM, W., KLOSE, J., WESTPHAL, B. AND WITTKE, H. 2004. *Live Sequence Charts: An Introduction to Lines, Arrows, and Strange Boxes in the Context of Formal Verification.* Lecture Notes in Computer Science, vol. 3147, Springer Verlag. 374-399.

BUDINSKY, F., STEINBERG, D., MERKS, E., ELLERSICK, R., and TIMOTHY J. GROSE. 2003. *Eclipse Modeling Framework.* Addison-Wesley Professional.

CHEN, W. AND LIN, C. 2000. A Hybrid Heuristic to Solve a Task Allocation Problem. *Comput. Oper. Res. 27,* 287-303.

CHUNG, Y S., AND WONG, J. T. 2010. Investigating Driving Styles and Their Connections to Speeding and Accident Experience. *Journal of the Eastern Asia Society for Transportation Studies 8*, 1944-1958.

COOK, S., JONES, G., KENT, S., WILLS, ALAN, C. 2007. *Domain-Specific Development with Visual Studio DSL Tools*. Addison-Wesley Professional.

CZARNECKI, K., AND HELSEN, S. 2006. Feature-based survey of model transformation approaches. *IBM Syst. J. 45,*3 (July 2006), 621–645.

ÇELİK, T. 2005. *A Software Modeling Tool for the High Level Architecture.* Master's Thesis. Hacettepe University Institute of Science, Ankara, Turkiye. Advisor(s) Kayhan Imre.

ÇELİK, T., GÖKDOĞAN, F.G., ÖZTÜRK, K., AND SARIKAYA, B. 2012. An HLA-Based Tactical Environment Application Framework. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology.* Advance online publication. Retrieved November 23, 2012. doi: 10.1177/1548512912465993.

ÇETINKAYA, D., AND OGUZTÜZÜN, H. 2006 A metamodel for the HLA object model. In *Proceedings of the 20th European Conference on Modeling and Simulation (ECMS)*, 207-213.

DALY, C. 2004. Emfatic Language Reference. http://www.eclipse.org/gmt/epsilon/doc/articles/emfatic/.

DOERNER, K., GUTJAHR, W. J., HARTL, R. F., STRAUSS, C. AND STUMMER, C. 2004. Pareto Ant Colony Optimization: A Metaheuristic Approach to Multiobjective Portfolio Selection. *Annals of Operations Research 131,* 1–4, 79–99.

ECLIPSE. 2012. Eclipse Project Web Site, www.eclipse.org. Eclipse Foundation.

EQUINOX. 2012. Equinox OSGI Project Web Site. http://www.eclipse.org/equinox. Eclipse Foundation.

EUGSTER, PATRICK TH., FELBER, PASCAL A., GUERRAOUI, R., AND KERMARREC, A. 2003. The many faces of publish/subscribe. *ACM Comput. Surv. 35*, 2 (June 2003), 114-131.

FOWLER, M. 2003. *UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed.* Addison-Wesley Professional.

FRANKEL, DAVID S., PARODI, J. AND SOLEY, R. 2004. *The MDA Journal: Model Driven Architecture Straight From The Masters.* Meghan Kiffer Pr.

FUJIMOTO, RICHARD M. 1999. *Parallel and Distributed Simulation Systems. 1st ed.* Wiley-Interscience.

GRA2MOL. 2012. Grammar to Model Language Project Web Site. http://modelum.es/trac/gra2mol/ Modelum Research Team, University of Murcia.

GIANNI, D., IAZEOLLA, G., AND D'AMBROGIO, A. 2010. A Methodology to Predict the Performance of Distributed Simulations. In *Proceedings of the*

2010 IEEE Workshop on Principles of Advanced and Distributed Simulation (PADS '10). IEEE Computer Society, Washington, DC, USA, 31-39.

GRONBACK, RICHARD C. 2009. *Eclipse Modeling Project: A Domain-Specific Language Toolkit.* Addison-Wesley.

GUIFFARD, E., KADI, D., MOCHET, J., AND MAUGET, R. 2006. CAPSULE: Application of the MDA methodology to the simulation domain. In *Proceedings of 2006 European Simulation Interoperability Workshop (SIW).*

HAMAM, Y. AND HINDI, K.S. 2000. Assignment of Program Modules to Processors: A Simulated Annealing Approach. *In Europian Journal of Operations Research 122,* 2 (April 2000), 509-513.

IBM 2010. Modeling with IBM ILOG CPLEX CP Optimizer – Practical Scheduling Examples, http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/.

IEEE 1998. IEEE STD 1278.1A-1998 Standard for Distributed Interactive Simulation – Application Protocols.

IEEE 2003. IEEE STD 1516.3-2003 Recommended Practice for HLA Federation Development and Execution Process (FEDEP).

IEEE 2010a. IEEE STD 1516-2010 Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules.

IEEE 2010b. IEEE STD 1516.1-2010 Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification.

IEEE 2010c. IEEE STD 1516.2-2010 Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Object Model Template (OMT) Specification.

IEEE 2010d. IEEE STD 1730-2010 Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP).

ISIS 2012. GME 12 Manual and User Guide. Institute for Software Integrated Systems, Vanderbilt University.

IZQUIERDO, J. L. C. AND MOLINA, J. G. 2009. A domain specific language for extracting models in software modernization. In *Proceedings of the 5th European Conference on Model Driven Architecture – Foundations and Applications* (ECMDA-FA '09), Richard F. Paige, Alan Hartman, and Arend Rensink (Eds.). Springer-Verlag, Berlin, Heidelberg, 82-97.

JET. 2012. Eclipse Java Emitter Transformations Project Web Site. http://www.eclipse.org/modeling/m2t/?project=jet#jet. Eclipse Foundation.

KARAGÖZ, A., AND DEMIRÖRS O. 2007. Developing Conceptual Models of the Mission Space (CMMS) - A Metamodel Based Approach. In *Proceedings of 2007 Spring Simulation Interoperability Workshop (SIW)*.

KOLOVOS, DIMITRIOS S., ROSE, LOUIS M., ABID, S., PAIGE, RICHARD F., POLACK, FIONA A. C., AND BOTTERWECK, G. 2010. Taming EMF and GMF Using Model Transformation. In *Proceedings of the 13th international conference on Model driven engineering languages and systems: Part I* (MODELS'10). Springer-Verlag, Berlin, Heidelberg, 211-225.

KOLOVOS, DIMITRIOS S, PAIGE, RICHARD F. AND POLACK, FIONA A. C. 2006. Eclipse Development Tools for Epsilon. In *Eclipse Summit Europe, Eclipse Modeling Symposium*.

KRUCHTEN, P. 2003. *The Rational Unified Process: An Introduction. 3rd ed.* Addison-Wesley Professional.

KUGELE, S., HABERL, W., TAUTSCHNIG, M., AND WECHS, M. 2009. Optimizing automatic deployment using non-functional requirement annotations. In *Leveraging Applications of Formal Methods, Verification and Validation Communications in Computer and Information Science Volume 17*. Springer Berlin Heidelberg, 400-414.

KUHL, F, WEATHERLY, AND R., DAHMANN, J. 1999. *Creating Computer Simulation Systems: An Introduction to the High Level Architecture. 1st ed.* Prentice Hall.

LAUTERBACH, C., LIN, MING C., DINESH. M., BORKMAN, S., LAFAVE, E., BAUER, M. 2008. Accelerating Line-of-sight Computations in Large OneSAF Terrains with Dynamic Events, In *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC).*

LEES, M., LOGAN, B. AND THEODOROPOULOS, G. 2007. Distributed simulation of agent-based systems with HLA. *ACM Trans. Model. Comput. Simul. 17,* 3, Article 11 (July 2007).

LO, V. M. 1988. Heuristic algorithms for task assignment in distributed systems. *IEEE Trans. Comput. 37,* 11 (November 1988), 1384-1397.

McAFFER, J., VANDERLEI, P., AND ARCHER, S. 2010. *Osgi and Equinox: Creating Highly Modular Java Systems (1st ed.).* Addison-Wesley Professional.

MALEK,, S, MEDVIDOVIC, N. MIKIC-RAKIC, M. 2012. An Extensible Framework for Improving a Distributed Software System's Deployment Architecture. *IEEE Trans. Softw. Eng. 38,*1 (January 2012), 73-100.

MATIC, S., GORACZKO, M., LIU, J., LYMBEROPOULOS, D., PRIYANTHA, B., ZHAO, F. 2006. Resource modeling and scheduling for extensible

embedded platforms. Technical Report MSR-TR-2006-176, Microsoft Reasearch, One Microsoft Way, Redmond, WA, USA.

MEHRABI, A., MEHRABI, S. AND MEHRABI, ALI D. 2009. An Adaptive Genetic Algorithm for Multiprocessor Task Assignment Problem with Limited Memory. In *Proceedings of the World Congress on Engineering and Computer Science 2009 Vol II.*

METZNER, A., AND HERDE, C. 2006. RTSAT--An optimal and efficient approach to the task allocation problem in distributed architectures. In *Proceedings of the 27th IEEE International Real-Time Systems Symposium* (RTSS '06). IEEE Computer Society, Washington, DC, USA, 147-158.

M2M. 2012. Eclipse Model to Model Transformation Project Website. http://www.eclipse.org/m2m/. Eclipse Foundation.

M2T. 2012. Eclipse Model to Text Transformation Project Website. http://www.eclipse.org/modeling/m2t/. Eclipse Foundation.

NOSEWORTHY, J. RUSSELL. 2008. The Test and Training Enabling Architecture (TENA) Supporting the Decentralized Development of Distributed Applications and LVC Simulations. In *12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*, Vancouver, BC, Canada, 259-268.

OMG, 2003. OMG Model Driven Architecture (MDA) Guide. Version 1.0.1.

OMG, 2006a. OMG Object Constraint Language (OCL) Version 2.0.

OMG, 2006b, Data Distribution Service for Real-Time Systems Ver 1.2.

OMG, 2011a. OMG MOF 2 XMI (XML Metadata Interchange) Mapping Specification. Version 2.4.1.

OMG, 2011b. OMG Unified Modeling Language (OMG UML) Infrastructure. Version 2.4.1.

OMG, 2011c. OMG Meta Object Facility (MOF) Core Specification. Version 2.4.1.

OSGI, 2011. OSGI Service Platform Core Specification Release 4, Version 4.3. OSGI Alliance.

PARR, S. 2003. A Visual Tool to Simplify the Building of Distributed Simulations Using HLA. *Information & Security Journal 12,* 2, 151-163.

PERUMALLA, KALYAN S., AND FUJIMOTO, RICHARD M. 2003. Scalable RTI-based Parallel Simulation of Networks. In *Proceedings of the seventeenth workshop on Parallel and distributed simulation* (PADS '03). IEEE Computer Society, Washington, DC, USA, 97-104.

PITCH TECHNOLOGIES. 2009. Pitch Commander. http://www.pitch.se/images/files/productsheets/PitchCommanderWeb_0904.pdf.

PIRIM, T. 2006. *A Hybrid Metaheuristic Algorithm for Solving Capacitated Task Allocation Problems as Modified Xqx Problems*. Ph.D. Dissertation. University of Mississippi, University, MS, USA. Advisor(s) Bahram Alidaee. AAI3259418.

PODGORELEC, V., AND HERICKO, M. 2007. Estimating software complexity from UML models. *SIGSOFT Softw. Eng. Notes* 32, 2 (March 2007), 1-5.

QVT. 2012. Eclipse Query/View/Transformations Project Web Site. http://www.eclipse.org/m2m/. Eclipse Foundation.

SCHMIDT, DOUGLAS C. 2006. Guest Editor's Introduction: Model-Driven Engineering. *IEEE Computer 39,* 2, 25-31.

SIDE 2012. SIDE Tool Framework Project Web Site. http://web.cs.hacettepe.edu.tr/~turgay/SIDE. Hacettepe University.

SISO, 2001a. Realtime Platform Reference (RPR) FOM Version 2 Draft 9.

SISO, 2001b. Guidance, Rationale, and Interoperability Modalities for the Real-time Platform Reference Federation Object Model (RPR FOM). Version 2.0D9v2.

SISO, 2006. Base Object Model (BOM) Template Specification. SISO-STD-003-2006.

STONE, H. S. 1977. Multiprocessor Scheduling with the Aid of Network Flow Algorithms. *IEEE Trans. Softw. Eng.* 3, 1 (January 1977), 85-93.

TEMPLEMAN, J., AND MUELLER, JOHN P. 2003. *COM Programming with Microsoft .NET.* Microsoft Press.

TOLK, A. 2002. Avoiding Another Green Elephant – A Proposal for the Next Generation HLA Based on the Model Driven Architecture. In *Proceedings of the 2002 Fall Simulation Interoperability Workshop (SIW).*

TOPÇU, O., ADAK, M., AND OGUZTÜZÜN, H. 2008. A metamodel for federation architectures. *ACM Trans. Model. Comput. Simul.* 18, 3, Article 10 (July 2008).

UCAR, B., AYKANAT, C., KAYA, K. AND IKINCI, M. 2005. Task assignment in heterogeneous computing systems. *J. Parallel Distrib. Comput.* 66, 1 (January 2006), 32-46.

VOELTER, M., KOLB, B., EFFTINGE, S., HAASE, A. 2006. From Front End To Code – MDSD in Practice, http://www.eclipse.org/articles/Article-FromFrontendToCode-MDSDInPractice/article.html.

VT MAK. 2010a. MAK Data Logger. http://www.mak.com/component/docman/doc_download/13-maek-data-logger--simulation-recording-a-playback.html.

VT MAK. 2010b. VR-Forces. http://www.mak.com/resources/white-papers/doc_download/15-vr-forces--the-complete-computer-generated-forces-solution-for-simulation.html.

VT MAK. 2010c. VR-Vantage XR. http://www.mak.com/component/docman/doc_download/24-vr-vantage-xr--common-operating-picture.html.

VT MAK. 2010d. MAK RTI. http://www.mak.com/component/docman/doc_download/14-maek-high-performance-rti.html.

XPAND. 2012. Eclipse XPand Project Web Site. http://www.eclipse.org/modeling/m2t/?project=xpand. Eclipse Foundation

XTEXT. 2012. Eclipse XText Project Web Site. http://www.eclipse.org/Xtext. Eclipse Foundation

ZEIGLER, B. P. 2003. DEVS today: recent advances in discrete event-based information technology. In 11th IEEE/ACM International Symposium on Modeling*, Analysis and Simulation of Computer Telecommunications Systems, MASCOTS 2003*, 148-161.

ZHENG, W., ZHU, Q., NATALE, M. DI, AND VINCENTELLI, A. S. 2007. Definition of Task Allocation and Priority Assignment in Hard Real-Time Distributed Systems. In *Proceedings of the 28th IEEE International Real-Time Systems Symposium (RTSS '07)*. IEEE Computer Society, Washington, DC, USA, 161-170.

ÖZGEÇMİŞ

Adı Soyadı : Turgay Çelik

Doğum Yeri : Siirt

Doğum Yılı : 1981

Eğitim ve Akademik Durumu:

Yüksek Lisans : 2003-2005 Hacettepe Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü

Lisans : 1999-2003 Hacettepe Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü

Lise : 1996-1999 Malatya Fen Lisesi

İş Tecrübesi:

2003- 2005 Araştırma Görevlisi, Hacettepe Üniversitesi Bilgisayar Mühendisliği Bölümü

2005- 2011 Yazılım Mühendisi, Uzman Yazılım Mühendisi, MilSOFT Yazılım Teknolojileri A.Ş.

2011- 2012 Uzman Yazılım Mühendisi, MilSOFT Bilişim İletişim Teknolojileri A.Ş.

2012 - Uzman Yazılım Mühendisi, MilSOFT Yazılım Teknolojileri A.Ş.

Yayınlar:

Dergi Yayınları:

1. **T.Celik**, B.Tekinerdogan, K.İmre, *"Deriving Feasible Deployment Alternatives for Parallel and Distributed Simulation Systems"*, ACM Transactions on Modeling and Computer Simulation – TOMACS (2013 Ocak'ta kabul edilmiştir).
2. **T.Celik,** B.Tekinerdogan, *"S-IDE: A Tool Framework for Optimizing Deployment Architecture of High Level Architecture Based Simulation Systems"*, ELSEVIER Journal of Systems and Software - Special Issue on

"Quality Optimisation of Software Architecture and Design Specifications". (2012 Haziran'da gönderilmiştir. 2012 Aralık'ta 1. Revizyon gönderilmiştir. Değerlendirme süreci devam etmektedir.)

3. **T.Celik**, G.Gokdogan, K.Ozturk, B.Sarikaya, "*An HLA Based Tactical Environment Application Framework*", Journal of Defense Modeling and Simulation-JDMS, 2012 Kasım. DOI: 10.1177/1548512912465993

Konferans Yayınları:

4. **T.Celik,** G.Vural, *"Savunma Sanayii Projelerinde Java Dili Ve Teknolojilerinin Kullanımının Değerlendirilmesi"*, SAVTEK'2012
5. **T.Celik**, H.Kutluca, G.Ozkan, *"Yüksek Seviye Mimari Uyumluluğu Dağıtık Benzetimlerin Kolay Birlikte Çalışabilirlik ve Tekrar Kullanılabilirliği için Yeterli midir?"*, USMOS'2011
6. H.Kutluca, **T.Celik**, G.Ozkan, *"Füze Sistemleri için Geliştirilecek Benzetimlerde Taktik Çevre Kullanımı"*, USMOS'2011
7. **T.Celik**, A.C.Cavdar, *"Benzetim Sistemleri ile Komuta Kontrol Sistemlerinin Birlikte Çalışabilirliğine Yönelik Örnek Çalışma"*, SAVTEK'2010
8. G. Gokdogan, G. Ozkan, **T. Celik**, F. Kucukyavuz, *"HLA Uyumlu Simülasyonlar için Entegre Geliştirme Ortamı"*, USMOS'2009
9. **T.Celik**, G. Gokdogan, *"HLA Uyumlu Taktik Çevre Uygulama Çatısı için Sistem Analiz Çalışması"*, USMOS'2009
10. G.Gokdogan, G.Ozkan, **T.Celik**, *"Ölçeklendirilebilir HLA Uyumlu Simülasyonlar için Uygulama Çatısı: SIMALT"*, USMOS'2007
11. R.Sutbas, **T.Celik**, K.Imre, *"HLA Uyumlu Benzetimlerin Ardıl-İşlem Çizenekleriyle Sınanması"*, UYMS'2005
12. **T.Celik**, R.Sutbas, K.Imre, *"HLA için Modelleme, Otomatik Kod Üretme, İzleme ve Sınama Araçları"*, USMOS'2005

Çalıştay Sunumları (Bildiri Özetli):

13. **T.Celik**, E.Serin, *"MilSOFT ICT Ağ Destekli Turkuaz Yazılım Ürün Hattı"*, UYMS'2012
14. **T.Celik,** B.Tekinerdogan, K.Imre, *"Koşut ve Dağıtılmış Benzetim Sistemleri İçin Uygun Yerleştirme Alternatiflerinin Türetilmesi"*, UYMS'2012

Diğer Sunumlar (Bildirisiz):

15. **T.Celik**, *"Müşterek Elektronik Harp Simülasyonu - MEHSİM"*, Aviyonik ve Sistem Entegrasyon Sempozyumu – ASES '2010

16. **T.Celik,** *"HLA Uyumlu Taktik Çevre Uygulama Çatısı"*, SSM Uçuş Eğitim Simülatörleri Çalıştayı'2010