İnsansız Hava Araçlarının Operasyonel Ortamda Modellenmesi ve Simülasyonu

Modeling and Simulation of Unmanned Aerial Vehicles in an Operational Environment

Kadir Alpaslan Demir
Bilgisayar Mühendisliği Bölümü
Deniz Harp Okulu, İstanbul
kademir@dho.edu.tr

Süleyman Erten
Bilgisayar Mühendisliği Bölümü
Deniz Harp Okulu, İstanbul
serten@dho.edu.tr

Erdi Dönmez
Bilgisayar Mühendisliği Bölümü
Deniz Harp Okulu, İstanbul
edonmez@dho.edu.tr

Halil Cicibaş
Komuta Kontrol Muhabere Bilgisayar ve İstihbarat (C4I) Sistemleri Programı
Deniz Bilimleri ve Mühendisliği Enstitüsü, İstanbul
hcicibas@dho.edu.tr

Öz

Özellikle otonom veya yarı otonom sistemlerin tasarımında ve geliştirilmesinde bulundukları operasyonel ortam şartlarının dikkate alınması önemli bir husustur. Maliyetli bir sistem geliştirme sürecine girmeden önce, bu tip sistemlerin içinde bulunacakları ortamlarda karşılaşacakları problemleri önceden görüp gerekli tasarım değişikliklerine gidebilmek ileride doğabilecek hataları azaltacaktır. Bunun yanında, gerçek ortamlarındaki görev başarı oranlarının önceden kestirilebilmesi sistem ortaya konduktan sonra yapılacak değişikleri azaltacak ya da tamamen ortadan kaldıracaktır. Bu bildirimizde daha önceden önerilen bir modelleme ve simülasyon yaklaşımını özellikle son yıllarda popülerliği gittikçe artan insansız hava araçları sistem geliştirmesine uyguladığımız durum çalışmasını anlatacağız. Bu modelleme ve simülasyon yaklaşımı sistemlerin ve ortamlarının modüler ve tekrar kullanılabilir şekilde Öznitelikli Olay Dili (Attributed Event Grammar) ile modellenmesi ve Ayrık Olay Simülasyonu (Discrete Event Simulation) ile simüle edilmesinden oluşmaktadır. Elde ettiğimiz sonuç, bu modellenme ve simülasyon yaklaşımının özellikle bu tip sistemler için etkin bir şekilde uygulanabildiği ve tasarımcılara sistemin operasyonel ortamındaki etkinliğine dair önemli bilgileri kısa bir süre içinde ortaya koyabildiğidir. Bildiride öncelikle bu yaklaşım kısa bir şekilde anlatılacak ve daha sonra yaklaşıma bağlı kalarak geliştirdiğimiz hava aracı simülasyonuna ilişkin sonuçlar sunulacaktır.

Abstract

During the development of autonomous and semi-autonomous of systems, the operational environment should be considered as well. Before undergoing a costly development cycle, to be able to foresee the problems of these systems in their operational environments and undergo the necessary modifications and the be able to predict the mission effectiveness will enable to eliminate or reduce changes after the system is deployed. In this paper, we explain a case study in which we apply a modeling and simulation approach to an unmanned air vehicle system development. This modeling and simulation approach consists of developing models with Attributed Event Grammar and simulating with discrete event simulation in a modular and reusable way. The results show that this approach is applicable to these types of systems and it quickly provides important information on the effectiveness of the system in its operational environment. In the paper, first we explain the approach and then we present the case study of simulating an unmanned air vehicle and the results of the study.

1. Giriş

Sistemlerin geliştirilmesinde konsept geliştirilmesi, gereksinim analizi ve sistem mimari tasarımı aşamalarında tespit edilecek bir hatanın giderilme maliyeti, sistemin çalışması esnasında doğacak hatanın giderilme maliyetinden çok daha azdır. Bu nedenle, sistemlerin operasyonel ortamlarda çalışmaları esnasında oluşabilecek hataların önceden tespit edilmeye çalışılması ve daha tasarım ve geliştirme aşamalarında bertaraf edilmesi ya da en aza indirilmesi idealde hedeflenmektedir. Bu husus, karmaşık, gerçek zamanlı, yüksek güvenirlilik isteyen, görev kritik otonom ve yarı otonom sistemlerde daha da önemlidir. Bu tip sistemlerin tasarımında oluşabilecek hataların sonuçları vahim olabilir. Örneğin bir şehir üzerinde güvenlik maksadıyla uçuş yapan bir insansız

hava aracının oluşan bir arıza nedeniyle düşmesi telafisi mümkün olmayan sonuçlar doğurabilir.

Bu tip sistemleri modellemek için literatürde birçok yöntem bulunmaktadır. UML [5], Petri Nets [6], SysML [10] bu örneklerden sadece bir bazılarıdır. Bu modelleme yöntemleri, sistem ortamını modelleden çok sistemleri modelleme üzerine yoğunlaşmaktadır. Sistemlerin statik ve dinamik özelliklerinin modellenmesi üzerine sayısız çalışma olup, sistemlerin çalışma ortamlarının ve ortam sistem etkileşimlerinin modellenmesi üzerine olan çalışmalar sınırlıdır. [3]'de önerilen yaklaşım ile sistem mimarisi ve sistemin içinde bulunduğu operasyonel ortam hızlı bir şekilde modellenebilmekte ve simüle edilebilmektedir. Bu yaklaşım modelleme ve simülasyon olmak üzere iki safhadan oluşmaktadır. Bu yaklaşımda modelleme ve simülasyonda modülerlik, yeniden kullanılabilirlik ve çabuk prototip geliştirmeye vurgu yapılmaktadır. Modelleme safhasında Öznitelikli Olay Dili (Attributed Event Grammar); simülasyon safhasında ise Ayrık Olay Simülasyonu (Discrete Event Simulation) kullanılmıştır [3]. Simülasyon ortamı olarak OMNET++ 4.0 seçilmiştir. Öznitelikli Olay Dili, Ayrık Olay Simülasyonu ve OMNET++ 4.0 simülasyon ortamı hakkında detaylı bilgi sonraki bölümlerde bulunmaktadır. Ancak şunu da belirtmekte fayda vardır ki, bu yaklaşım vurgu yapılan özellikleri desteklemek kaydıyla diğer modelleme ve simülasyon teknolojilerinin kullanılmasına da açıktır.

[3] ile rapor edilen çalışmada bu modelleme ve simülasyon yaklaşımı tanıtılmış ve denizaltıdan atılan homing bir torpidonun sualtı ortamındaki hedef arama ve imha problemi üzerinde durulmuştur.

Bu araştırmamızda ise iki temel hedef vardır:

1. Yukarıda bahsedilen yaklaşımın uygulanabilirliğini ve kısıtlarını tespit edebilmek için daha kapsamlı bir durum çalışması yapmak.
2. Bir insansız hava aracının operasyonel ortamda modellenmesini ve simülasyonunu gerçekleştirmek.

Bu çalışmamızda insansız hava aracının örnek çalışma olarak seçilmesinin nedenleri aşağıda sıralanmaktadır:

- Son yıllardaki artan popülerliği,
- Ortam ile sürekli etkileşimde olması,
- Yaklaşımın test edilebilmesi için sistemin yeterince karmaşık ve gerçekçi olması

Araştırmadan elde ettiğimiz sonuçlar göstermektedir ki özellikle otonom ve yarı otonom sistemlerin geliştirilmesinde sistemlerin ve ortamlarının birlikte modellenmesi ve sistem-ortam etkileşimlerinin simülasyonu sistemin gerçek ortam şartlarındaki çalışması hakkında önemli bilgiler vermektedir. Bu bilgiler sistem geliştiricilerine, seçecekleri mimari tasarımların etkinliği hakkında bilgiler vererek alternatif tasarımları hızlı bir şekilde test edebilmelerine olanak sağlayacaktır.

Bildirinin içeriği şu şekildedir: Bölüm 2'de yaklaşım kısaca tanıtılacak olup, bölüm 3'de insansız hava aracı sisteminin ve operasyonel ortamın modellenmesi anlatılacak ve bölüm 4'te geliştirilen simülasyondan bahsedilecektir. Analizler bölüm 5'te, sonuçlar ise bölüm 6'da yer almaktadır.

2. Sistemlerin ve Ortamlarının Modüler Hızlı Prototip Geliştirilmesi Yaklaşımı ve Kullanılan Teknoloji ve Araçlar

2.1. Yaklaşım

Özellikle çeşitli otonom sistemlerde sistemin ana kullanıcısı yani sistemle direk etkileşim halinde bulunan özne sistemin içinde bulunduğu ortamdır. Sistem ortamdaki olaylardan etkilenir ve bu olaylara istinaden çeşitli işlemler yapar ya da reaksiyon gösterir. Örnek olarak bir insansız hava aracı ortamdan aldığı sinyallerden bazılarının bir hedef sinyali olduğuna karar verirse arama yöntemini değiştirip hedefin olduğu bölgeye yoğunlaşabilir ya da ana kumanda merkezine bir uyarı mesajı gönderebilir. İşte burada sistem bir insan ya da sistem yerine ortamı ile etkileşim halindedir.

Yukarıdaki örnekte belirtildiği üzere, özellikle otonom ve yarı otonom sistemlerin gereksinim analizi ve geliştirilme safhalarında, ortamın ve ortamdaki olayların sistem üzerine etkileri iyi anlaşılmalıdır. Sistemin geliştirilmesinde sistem-ortam etkileşimi önemli bir rol oynamaktadır çünkü sistemin ana kullanıcısı ortam olmaktadır. İşte bu tip sistemlerin geliştirilmesinde sistemlerin daha etkin ve daha az hata ile geliştirilebilmesi ve sistemin çalışma ortamında yüksek hedef tespit başarı oranına ulaşılabilmesi için sistemlerin ve ortamlarının hızlı bir şekilde modellenmesi ve simülasyonu sistem ve yazılım geliştiricilerine ihtiyacı olan bilgileri sağlamakla beraber alternatif tasarımları da hızlı bir şekilde test edebilme olanağı sağlamaktadır. Aynı zamanda özellikle füze, torpido, insansız araç, uzay aracı gibi maliyetli otonom ve yarı otonom sistemlere baktığımızda, sistemlerin her seferinde sıfırdan geliştirilmesi yerine versiyonlar halinde sistemlerin zaman içerisinde evrime uğradığı, çeşitli sensör ve cihazların zamanla sisteme eklendiği, yeni görevler çerçevesinde sistemin gereksinimlerinin değiştiği ve sistemin genişletildiği, sistemlerin sistemleri (system of systems) gibi sistemlere entegre olduğu gözlemlenmektedir. Bu evrim sürecinde sistemin değiştirilmesi ve geliştirilmesinde çeşitli konsept ispatlarının (proof of concept) ve gereksinimlerin hızlı bir şekilde test edilebilmesi, sistem ve yazılım geliştiricilerine maliyetli yanlışlardan kaçınılmasına olanak sağlar. Bu

ihtiyaca cevap vermek maksadıyla bir modelleme ve simülasyon yaklaşımı [3]'de geliştirilmiştir. Yaklaşım şu aşamalardan oluşur:

1. Sistemin genel bir mimarisi oluşturulur.
2. Bu mimari oluşturulurken sistemin değişmesi ve gelişmesi muhtemel parçaları tespit edilir.
3. Sistem, geliştirme ortamına (domain) uygun bir modelleme teknolojisi ile özellikle değişmesi ve gelişmesi muhtemel parçalara yoğunlaşılarak modüler bir şekilde modellenir. Böylece daha sonra bu parçaların yerine gelecek yeni parçaların modelleri kullanılarak sistemin yeniden kısa bir zamanda modellenmesine imkan tanınır.
4. Üçüncü aşamadaki felsefeye uygun kalınarak aynı aşama ortam içinde tekrarlanır. Yalnız ortam için özellikle dikkat edilmesi gereken ortam hakkında elde bulunan bilgilerdir. Mesela bir sualtı ortamı için bazı ortam verileri iyi bilinirken bazı ortam verileri daha az doğrulukla biliniyor olabilir. İşte bu durumlarda özellikle ortamda iyi bilinmeyen kısımlar ayrı modellenmeli ki böylece yeni bilgilere sahip olunduğunda bu yeni bilgiler hızlı bir şekilde modele dökülüp, modüler bir şekilde ortam modeline entegre edilebilesin.
5. Geliştirilen modeller baz alınarak uygun bir simülasyon ortamı ile sistem ve ortamı simüle edilir. Çok çeşitli senaryoların otomatik bir şekilde çalıştırılması ile sistem-ortam etkileşimleri ile sistemin çalışma ortamında oluşabilecek hatalar tespit edilmeye çalışılır. Elde edilen geribeslemeler değişik alternatiflerin denenmesinde ve sistemin mimarisinin geliştirilmesinde kullanılır.
6. Geliştirilen modeller sistemin evrimi boyunca yeni versiyonlar ve benzer sistemler için tekrar kullanılır.

Bu yaklaşımda özellikle vurgulanan özellikler şunlardır:

- Modülerlik
- Yeniden kullanılabilirlik
- Model tabanlı mühendislik
- Hızlı prototip geliştirme
- Sistem, ortam ve sistem-ortam etkileşimi modellenmesi
- Gereksinim analizi ve konsept ispatı maksadıyla değişik senaryo ve alternatifler için çok sayıda simülasyonun otomatik koşturulması

2.2. Modelleme

Durum çalışmamızda sistemin ve ortamının modellenmesi maksadıyla öznitelikli olay dili (Attributed Event Grammar) kullanılmıştır. Öznitelikli olay dili, olay dilinin geliştirilmiş versiyonudur. Olay dili program testi, program çalışma gözlemi (monitoring), hata ayıklama (debugging) için kullanılmıştır [16], [17], [18]. Öznitelikli olay dili ortamların modellenmesi ve bu modellere dayanarak otomatik sistem testi maksadıyla başarıyla kullanılmıştır [1], [2]. [3]'te ise öznitelikli olay dili hem sistem hem de ortam modellenmesi için kullanılmıştır.

Otonom ve yarı otonom sistemlerin büyük bir kısmı gerçek zamanlı, reaktif ve görev kritik sistemler olarak düşünülebilir. Bu tip sistemlerde, sistem ortamı etkiler ve olay (event) yaratır. Aynı zamanda ortamdaki olaylardan da etkilenir ve reaksiyon gösterir. Öznitelikli olay dili özellikle bu tip sistemlerin modellenmesine doğal olarak destek sağlamaktadır.

Öznitelikli olay dilinde esas unsur olaylar ve onların öz nitelikleridir. Her olayın; tipi, başlangıcı, bitişi ve süresi gibi öznitelikleri mevcuttur. Bu dil kullanılarak tanımlanan olayların ortak özellikleri şunlardır:

- Olaylar kesiklidir.
- Olaylar birbirini takip ettiği gibi eşzamanlı da olabilir.
- Olaylar arasında sıra (öncelik) ilişkisi vardır.
- Bir olay diğer alt olayları kapsayabilir. Olaylar arasında hiyerarşik ilişki vardır.

Olay dili hakkında daha detaylı bilgi [1] ve [2]'de yer almaktadır.

Tablo 1'de öznitelikli olay dilindeki temel notasyonlar sunulmuştur.

Tablo 1: Öznitelikli Olay Dili Notasyonu

(…|…)	Olaylar arasında alternatif
(*….*)	Sıfır veya aha fazla sıralı olaylar dizisi
{*….*}	Sıfır veya daha fazla sırasız olaylar dizisi
{a , b}	Aralarında sıralama olmayan olaylar dizisi
/action/	Bir aksiyon, örneğin değişkene değer atama
(X..Y)	X..Y arasında rasgele bir değer kadar yineleme
P(Z)	Bir aksiyon ya da olayın oluşma olasılığı
RULE Event Type{…}	Olay tipini ve altındaki tanımlamaları gösterir
DELAY(t)	t kadar gecikme süresi
(*…*)(==X)	X sayısı kadar öteleme
ENCLOSING Parent_Event.Attribute	Üst sınıfın bir değişkenine veya özelliğini gösterme
(EVERY t)	Her t süresinde
WHEN(C)… ELSE…	C durumuna göre karar verme

2.3. Simülasyon

Yaklaşımdaki ikinci temel aşama modüler bir şekilde modellenenen sistemin ve ortamının modellere dayanarak

simülasyonlarının yapılmasıdır. Modelleri destekleyen ve hızlı bir şekilde simülasyon geliştirilmesine olanak sağlayan yani uygun bir simülasyon ortamının seçilmesi önemlidir. Hatta modelleri girdi olarak alıp onları derleyen ve otomatik olarak simülasyon kodu ortaya çıkaran bir simülasyon yazılım aracının kullanılması idealdir. Ancak günümüzde öznitelikli olay dili için böyle bir simülasyon yazılım aracı mevcut değildir. Dolayısıyla bu çalışmada bir önceki temel aşamada geliştirilen modeller simülasyon için koda dönüştürülmüştür. Auguston ve arkadaşları tarafından ortam modellerini otomatik olarak test senaryolarına dönüştürücü ve daha sonra bu senaryolar ile sistemi test etmeye yarayan bir derleyici geliştirilmiştir [1]. Ancak bu derleyici sistem geliştirildikten sonra test aşamasında kullanılabilmektedir.

Bu çalışmada sistem ve ortam modüler bir şekilde öznitelikli olay dili ile modellendikten sonra modellemenin olay tabanlı yaklaşımına sadık kalarak ayrık olay simülasyonu ile simüle edilmiştir. Ayrık Olay Simülasyonu (Discrete Event Simulation), simülasyonda gerçekleşen olayların ayrık olarak ele alınması esasına dayanır. Bu olaylar birbiriyle gerektiğinde ilişki içinde olabilen, sonlu sayıda ve belirli bir süresi olan olaylardır. Olaylar sistemin durumu üzerinde değişikliğe sebep olurken olaylar arasında sistem durumunda değişiklik olmaz. Birçok farklı disiplinde ayrık olay simülasyonu kullanılarak yapılan birçok çalışma bulunmaktadır [12,13,14]. Ayrık olay simülasyonu hakkında daha detaylı bilgi [8,9,15]'de yer almaktadır.

Sistem ve ortamı OMNET++ 4.0 simülasyon ortamında simüle edilmiştir. Çalışmamızda bu ortamı seçmemizin en temel nedeni olay tabanlı bir yaklaşımı desteklemesidir. Diğer nedenleri arasında açık kaynak kodlu ve ücretsiz olması ve geniş bir kullanıcı kitlesi tarafından kullanılmasıdır. OMNET++ 4.0 simülasyon ortamı C++ tabanlı başta ağ yapıları olmak üzere tüm dağıtık sistemlerin simüle edilmesine imkan tanıyan bir ayrık olay simülasyon aracıdır [7]. Bu özelliklerinin yanında OMNET++ kullanıcı dostu arayüzü sayesinde simülasyonun görsel olarak izlenebilmesini sağlamaktadır. OMNET++ 4.0 yazılımına ait daha geniş bilgi ilgili web sayfasında bulunmaktadır [11].

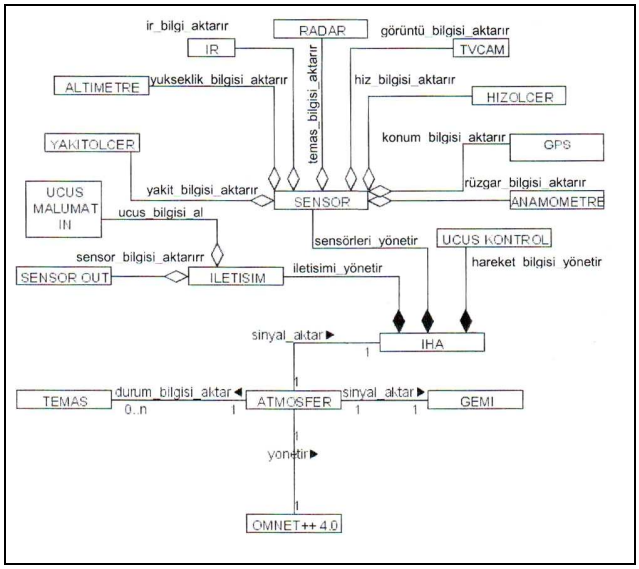
3. İnsansız Hava Aracı ve Operasyonel Ortam Modeli

Bu bölümde insansız hava aracı ve operasyonel ortam modelinin geliştirilmesi sürecinden bahsedilecektir.

3.1. İnsansız Hava Aracı Modeli

İnsansız hava aracı modelinin oluşturulmasında öncelikle sistemin kullanıcının ihtiyaçlarına, yapacağı görevlere ve çalışacağı ortama bağlı olarak sahip olması gereken fonksiyon ve özelliklerle beraber temel bir mimari ortaya konmuştur. Bu fonksiyon ve özellikleri sağlayacak altsistem ve cihazlar tespit edilmiştir. Altsistem entegrasyon sürecinde gelecekte sisteme yapılabilecek değişimler de öngörülerek açık mimari tasarım yaklaşımı izlenmiştir. Sonrasında ise her bir bileşen öznitelikli olay dili kullanılarak modellenmiştir. İnsansız hava aracının modüler tasarımı sayesinde sistemdeki herhangi bir bileşene yapılacak değişimlerden sistem etkilemeyecektir. Modüler tasarımın diğer bir avantajı ise her bir bileşenin birbirinden bağımsız olarak öznitelikli olay dili ile tanımlanabilmesini kolaylaştırmasıdır. Sisteme ait modüller ve bileşenler sonraki çalışmalarda oluşturulacak benzer sistemlerde kolaylıkla kullanılabilecek yapıdadır.

Sistemin ve ortamın kavramsal modeli Şekil-1'de sergilenmektedir. Bu modelde belirtilen bileşenler, insansız hava aracı tarafından gerçekleştirilecek görevlere bağlı olarak çoğaltılabilir ve modelleri kolayca sisteme entegre edilebilir.



Şekil 1: İnsansız hava aracı ve ortamın kavramsal modeli

Bu kısımda, sistem ve sistemin temel parçaları kısaca tanıtılmaktadır. İnsansız hava aracı uzaktan kontrol istasyonu ile iletişim modülü sayesinde haberleşmektedir. İletişim modülü; Sensör_Haberleşme, Uçuş_Bilgisi ve İletişim_Kernel olmak üzere 3 adet bileşene sahiptir. Her bir bileşen sistemdeki farklı fonksiyonları yerine getirmek üzere tasarlanmıştır. İnsansız hava aracı, sensör modülü sayesinde ortamdaki durum bilgisini elde eder. GPS, altimetre, hız ölçer, yakıt ölçer, radar, kamera ve Sensör_Kernel bileşenleri sensör modülünü oluşturmaktadır. İnsansız hava aracının diğer bir modülü ise uçuş kontrol modülüdür. Bu modül sayesinde uçuş planına ilişkin görevler ve kontroller gerçekleştirilir. Uçuş kontrol modülü 4 farklı çalışma moduna sahiptir. Bunlar kalkış, intikal, arama ve iniş modudur. İnsansız hava

aracının her modda izlemiş olduğu arama yöntemi modelde tanımlanmıştır.

Çalışma kapsamında temasların tespit olasılıklarının esas olarak bulut yüksekliğine ve temasın hareketlerine bağlı olarak değişeceği öngörüsünde bulunulmuştur. Bu kapsamda hedefler sabit kabul edilmiş ve insansız hava aracına entegre radarın çeşitli bulut yüksekliğindeki efektifliği belirlenerek modele dahil edilmiştir.

Çalışmada, yukarıda bahsedilen tüm sistem bileşenleri ve modüller öznitelikli olay dili kullanılarak modellenmiştir. Sistemin ve ortamın tüm modelini burada göstermek oldukça yer alacağından bildiride sadece örnek bir altsistemin modeli gösterilecektir. Şekil 2'de insansız hava aracındaki radar cihazının öznitelikli olay dili ile modellenmesi yer almaktadır. Radar cihazı ile ilgili önemli bir nokta, sistem temas yüzdelerinin radar performansı ile doğrudan ilişkili olmasıdır.

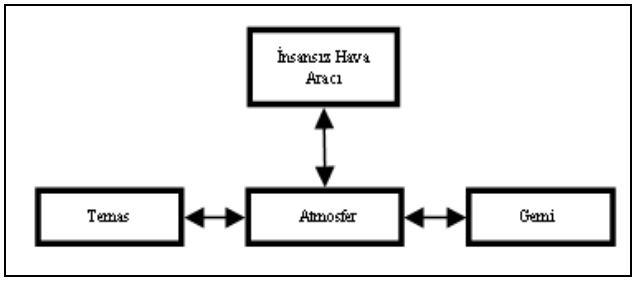
```
RULE RADAR{
        bool                    radar_calistir;
        int                     toplam_temas;
        bool                    temas_var;
        int                     temas_sayisi;
        int                     temaslar;
        int                     temas_ile_aradaki_aci[10];
        double                  temas_ile_aradaki_mesafe[10];
        radar_yayini            radar;
        gemi_radar_haberlesmesi temas;
}

RADAR:
        initialize(*radar_aktif_pasif,temas_belirle*)

initialize:
        /ENCLOSING RADAR.radar_calistir=false;
        ENCLOSING RADAR.temaslar=0;/

radar_aktif_pasif:
        WHEN(msg.getName( )==”inis”)
                /ENCLOSING RADAR.radar_calistir=false;
                delete(msg);/
        WHEN(msg.getName( )==”radar_calistir”)
                /ENCLOSING RADAR.radar_calistir=true;
                delete(msg);
                send_to_atmosfer(ENCLOSING RADAR.radar)
                ENCLOSING RADAR.temas_var=false;/
temas_belirle:
        WHEN(msg.getName( )==”radar”)
                /ENCLOSING RADAR.radar=msg;
                ENCLOSING
                RADAR.toplam.temas=ENCLOSING
                RADAR.radar.getToplam_temas( );
                ENCLOSING RADAR.temaslar=ENCLOSING
                RADAR.radar.getTemaslar( );
                ENCLOSING
                RADAR.temas_sayisi=ENCLOSING
                RADAR.radar.getTemas_sayisi( );
                parametreleri_degistir( );
                delete(ENCLOSING
                RADAR.toplam_temas=ENCLOSING
                RADAR.radar);
                bilgi_düzenle( );
                ENCLOSING
                RADAR.temas.setToplam_temas(ENCLOSING
                RADAR.toplam_temas);
                ENCLOSING
                RADAR.temas.setTemaslar(ENCLOSING
                RADAR.temaslar);
                ENCLOSING
                RADAR.temas.setTemas_sayisi(ENCLOSING
                RADAR.temas_sayisi);
                send_to_sensor_kernel(ENCLOSING
                RADAR.temas);/
```

Şekil 2: Radar cihazının öznitelikli olay dili ile tanımlanması

3.2. Operasyonel Ortam Modeli

Operasyonel ortam ile insansız hava aracının etkileşimde bulunduğu atmosfer, gemi ve temaslar kastedilmektedir. Sistemin sürekli olarak etkileşimde olacağı operasyonel ortamın modellemesi gerçek şartlar altında sistemin çalışması esnasında oluşabilecek muhtemel problemlerin tespitini kolaylaştıracak ve bu problemlerin ortadan kaldırılmasını sağlayacaktır. Örneğin atmosferik ortamdaki basınç, nem, rüzgar ve yağış gibi faktörlerin insansız hava aracının çalışma performansı üzerinde doğrudan etkisi vardır.

Ortam modelinde; yukarıda bahsedilen atmosferik şartlar, uzaktan kontrol istasyonu ve temaslar birbirinden bağımsız olarak modellenmiştir. Ortam modelindeki bileşenlerin ve insansız hava aracının oluşturduğu kavramsal yapı Şekil 3'de gösterilmiştir.



Şekil 3: Kavramsal Yapı

Operasyonel ortamda bulunan her bir bileşen bağımsız ve modüler olarak öznitelikli olay dili ile modellenmiştir. Örnek olarak atmosferik ortamın öznitelikli olay dili ile tanımlanması Şekil 4'de belirtilmektedir. Bu çalışma kapsamında, modeli oluşturulan her operasyonel ortam modülü benzer çalışmalarda üzerinde değişiklik yapılmasına gerek kalmadan kullanılabilecektir.

```
RULE ATMOSFER{
        double  ruzgarin_siddeti;
        double  ruzgarin_acisi;
        double  temas_sayisi;
        double  temas_koordinat_N[10][3];
        double  temas_koordinat_E[10][3];
        double  temas_yükseklik[10];
```

```
        double    temas_arasi_aci[10];
        double    temas_arasi_mesafe[10];
        double    iha_yukseklik;
        double    iha_N_koordinat [3];
        double    iha_E_koordinat [3];
        double    radarin_gorebilecegi_mesafe;
        bool      firtinali;
        bool      hafif_yagmurlu;
        bool      bulutlu;
        bool      acik;
        double    bulut_yuksekligi;
        double    tespit_orani;
        int       toplam_temas;
}
ATMOSFER:
        initialize (*gemi_iha_haberlesmesi,
        bilgilerin_ihaya_aktarimi, iha_hareketi*);
initialize:
        /ENCLOSING
        ATMOSFER.ruzgarin_siddeti=RAND(0..30);
        ENCLOSING                ATMOSFER.rüzgarin_acisi=
        RAND(0...360);
        ENCLOSING ATMOSFER.temas_sayisi=parametre("temas
        sayisi");
        ENCLOSING ATMOSFER.iha_yukseklik=10;
        ENCLOSING
        ATMOSFER.radarin_gorebilecegi_mesafe=5;
        ENCLOSING ATMOSFER.firtinali=parametre("firtinali");
        ENCLOSING
        ATMOSFER.hafif_yagmurlu=parametre("hafif_yagmurlu")
        ;
        ENCLOSING ATMOSFER.bulutlu=parametre("bulutlu");
        ENCLOSING ATMOSFER.acik=parametre("acik");
        ENCLOSING ATMOSFER.iha_N_koordinat [0]=38;
        ENCLOSING ATMOSFER.iha_N_koordinat [1]=0;
        ENCLOSING ATMOSFER.iha_N_koordinat [2]=0;
        ENCLOSING ATMOSFER.iha_E_koordinat [0]=25;
        ENCLOSING ATMOSFER.iha_E_koordinat [1]=0;
        ENCLOSING ATMOSFER.iha_E_koordinat [2]=0;
        atmosfer_hava_durumu();
        temas_mevkileri();/
gemi_iha_haberlesmesi:
        WHEN(msg.getName( )=="ucus_malumat")
                /send_to_ucus_malumat_modulu(msg);/
        WHEN(msg.getName( )=="temas")
                /send_to_gemi(msg);/
        WHEN(msg.getName( )=="gps")
                /send_to_gemi(msg);/
bilgilerin_ihaya_aktarilmasi:
        WHEN(msg.getName( )=="gps_calistir")
                /gpse_bilgi_gonder( );/
        WHEN(msg.getName( )=="altimetre_calistir")
                /altimetreye_bilgi_gonder( );/
        WHEN(msg.getName( )=="anomometre_calistir")
                /anomometreye_bilgi_gonder( );/
        WHEN(msg.getName( )=="radar")
                /radara_bilgi_gonder( );/
iha_hareketi:
        WHEN(msg.getName()=="iha") /ENCLOSING
        ATMOSFER.iha_N_koordinat [0]=msg.
        getN_mevkii(0);
        /ENCLOSING    ATMOSFER.iha_N_koordinat    [1]=msg.
        getN_mevkii(1);
        /ENCLOSING    ATMOSFER.iha_N_koordinat    [2]=msg.
        getN_mevkii(2);
        /ENCLOSING    ATMOSFER.iha_E_koordinat    [0]=msg.
        getE_mevkii(0);
        /ENCLOSING    ATMOSFER.iha_E_koordinat    [1]=msg.
        getE_mevkii(1);
        /ENCLOSING    ATMOSFER.iha_E_koordinat    [2]=msg.
        getE_mevkii(2);
        /ENCLOSING   ATMOSFER.yukseklik=msg.getYukseklik(
        );
        sensorlere_bilgi_gonder( );/
```

Şekil 4: Atmosferik ortamın öznitelikli olay dili ile tanımlanması

4. Simülasyon

Bir önceki bölümde insansız hava aracının ve operasyonel ortamın modellenmesi anlatıldı. Bu bölümde ise modellere dayanılarak geliştirilen simülasyon anlatılacaktır. Simülasyonda her bir modül modeli ayrı ayrı ele alınmış ve modeller temel alınarak kodlama yapılmıştır. Bu modüler yaklaşım ile sistem karmaşıklığının daha rahat bir şekilde ele alınması sağlanmıştır. Simülasyonlar OMNET++ 4.0 simülasyon ortamı kullanılarak geliştirilmiştir. Simülasyon safhasında OMNET++ haricinde faklı araçlar da kullanılabilir ancak çalışmamızda OMNET++ yazılımını seçmemizin birkaç sebebi bulunmaktadır. Öncelikle, OMNET++ yazılımının en önemli özelliklerinden birisi nesne tabanlı olan C++ dilini destekler olmasıdır. Dolayısıyla çok kısa bir süre içerisinde öğrenilebilmekte yani öğrenme eğrisi oldukça düşüktür. Bu aşamada elimizde bulunan modeller sayesinde kodlama oldukça hızlı ve basit bir şekilde yapılabilmiştir. Ayrıca OMNET++ sahip olduğu kullanıcı dostu arayüzü sayesinde zamana bağlı olayların takibini kolaylaştırmaktadır. Son olarak ise yazılım, açık kaynak kodu sayesinde eğitim ve araştırma maksatlı olarak ücretsiz bir şekilde kullanılabilmektedir.

Öznitelikli olay dili ile tanımlanan tüm modüller teker teker modellere bağlı kalınarak OMNET++ ortamında simülasyon koduna dönüştürülmüştür. Yani her bir model için ayrı bir C++ sınıfı yaratılmıştır. Henüz öznitelikli olay dili ile modellenen modelleri otomatik olarak simülasyon koduna dönüştürecek bir yazılım aracı mevcut olmadığından bu dönüşüm programcılar tarafından manuel olarak yapılmıştır. Ancak şunu da belirtmek gerekir ki öznitelikli olay dili modellerinden koda dönüşüm işlemi oldukça hızlı bir şekilde yapılabilmiştir. Öznitelikli olay dili modelleri incelendiğinde bir bilgisayar program koduna oldukça yakın olduğu gözlenecektir. Sekil 2'de modeli belirtilen radar cihazının simülasyon kodu Şekil 5'dedir.

```
#include <stdio.h>
#include <iostream>
#include <string.h>
#include <omnetpp.h>
#include "radar_yayini_m.h"
#include "gemi_radar_haberlesmesi_m.h"
class RADAR: public cSimpleModule{
        public:
                bool radar_calistir;
                int toplam temas;
                bool temas_var;
                int temas_sayisi;
                int temaslar;
                double temas_ile_aradaki_aci[10];
```

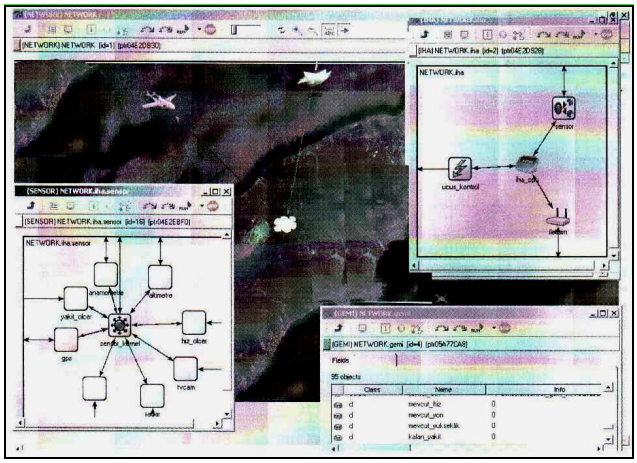
```
                    double temas_ile_aradaki_mesafe[10];
                    radar_yayini *radar;
                    gemi_radar_haberlesmesi *temas;
          protected:
                    virtual void intilialize();
                    virtual void handleMessage(cMessage *msg);
                    void parametreleri_degistir();
                    void bilgi_duzenle();
};
Define Module(RADAR);
void RADAR::intilialize(){
          radar_calistir=false;
          temaslar=0;
          WATCH(temas_var);
          WATCH(temas_ile_aradaki_aci);
          WATCH(temas_ile_aradaki_mesafe);
}
void RADAR::handleMessage(cMessage *msg){
          if(strcmp("inis",msg->getName())==0){
                    radar_calistir=false;
                    delete(msg);
          }
          else if(strcmp("radar_calistir",msg->getName())==0){
                    radar_calistir=true;
                    delete(msg);
                    radar_yayini *radar=new radar_yayini("radar");
                    send(radar,"radar_transmisyon_out");
                    temas_var=false;
          }
          else if (strcmp("radar",msg->getName())==0){
                    radar=check_and_cast<radar_yayini *>(msg);
                    toplam_temas=radar->getToplam_temas();
                    temaslar=radar->getTemaslar();
                    temas_sayisi=radar->getTemas_sayisi();
                    parametreleri_degistir();

                    delete(radar);
                    temas=new gemi_radar_haberlesmesi("temas");
                    bilgi_duzenle();

                    temas->setToplam_temas(toplam_temas);
                    temas->setTemaslar(temaslar);
                    temas->setTemas_sayisi(temas_sayisi);
                    send(temas,"cpu_out");
          }
}
void RADAR::parametreleri_degistir(){
          for(int i=0;i<temas_sayisi;i++){
          temas_ile_aradaki_aci[i]=radar->getTemasin_acisi(i);
          temas_ile_aradaki_mesafe[i]=radar->getAradaki_mesafe(i);
          }
}
void RADAR::bilgi_duzenle(){
          for(int i=0;i<temas_sayisi;i++){
temas->setTemas_ile_aradaki_aci(i,temas_ile_aradaki_aci[i]);
temas->setTemas_ile_aradaki_mesafe(i,temas_ile_aradaki_mesafe[i]);
          }
}
```

Şekil 5-Radar cihazına ait simülasyon kodu

Simülasyona ilişkin ekran görüntüsü şekil 6'da gösterilmektedir. Bu simülasyonlardaki temel amacımız değişik hava koşullarında ve değişik arama yöntemlerinde insansız hava aracımızın operasyonel ortama yerleştirmiş olduğumuz belirli sayıdaki hedefi radarı vasıtasıyla ne kadar başarıyla tespit edebileceğini bulmaktır.

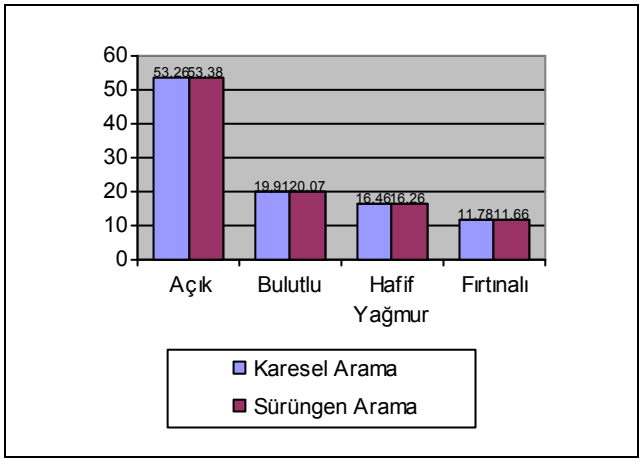
Simülasyon başlangıcında aşağıdaki bilgiler sisteme verilmiştir:

- Simülasyonun kaç defa çalışacağı,
- Atmosferik şartlar,
- Temas sayısı,
- Uçuşta takip edilecek noktalar,
- Arama yöntemi,
- Arama sahasının koordinatları



Şekil 6: Simülasyona ilişkin ekran görüntüsü

Simülasyon, 2 farklı arama yöntemi ve 4 farklı hava durumu olmak üzere 8 farklı senaryonun her biri 1000 defa olmak üzere çalıştırılmıştır. Simülasyon neticesinde elde edilen verilerin görsel olarak sergilenmesi, JAVA dilinde yazılan bir program ile sağlanmıştır. Bu program kullanılarak insansız hava aracının sahadaki temasları tespit yüzdeleri belirlenmiştir. Şekil 7'de hava şartlarına ve seçilen arama yöntemine göre hedef tespit yüzdeleri yer almaktadır.



Şekil 7: Temas yüzdeleri

Hedef tespit yüzlerini incelediğimizde daha önceden beklemediğimiz bir sonuç ile karşılaştık. Farklı arama yöntemlerinde temas tespit oranları birbirine çok yakın çıktı. Bunun nedeninin her iki arama yönteminin kaplama yüzdelerinin eşit olması ve hedefin sabit olmasından kaynaklandığı değerlendirdik. Diğer bir yandan daha önceden öngördüğümüz gibi hava durumu insansız aracımızın hedef tespit oranını ciddi oranda etkiledi. Simülasyon neticesinde elde edilen veriler kapsamında, temas tespit yüzdelerinin hava durumu ve arama yönteminin kaplama yüzdesine doğrudan bağlı olarak değiştiği tespit edildi.

5. Analiz

Çalışma kapsamında elde edilen tecrübeler aşağıda kısaca belirtilmiştir.

- Operasyonel ortam modelinin geliştirilmesi ile sistemin çalışma ortamındaki davranışları hakkında önemli bilgilere ulaşılmıştır.
- Modüler bir yaklaşımın kullanılması modelleme ve simülasyonun kısa sürede geliştirmesini sağlamıştır. Sistem karmaşıklığı üstesinden gelinir bir hale indirgenmiştir.
- Modellerin daha önceden geliştirilmesi simülasyonun kodlanmasını hızlandırmıştır.
- Olay tabanlı yaklaşım ile sistem-ortam etkileşimi daha anlaşılır ve efektif şekilde tanımlanabilmiştir.
- Simülasyon ile elde ettiğimiz prototip ile sistem analiz edilerek değiştirilmesi ve iyileştirmesi kısa sürede yapılabilmektedir.
- Simülasyonda ayrık olay simülasyonun kullanılması olaylar arasında hiyerarşik ve sıralı bir yapı oluşturma imkanını sağlamıştır.
- OMNET++ ortamının kullanıcı dostu arayüzüne sahip olması ve nesne-tabanlı programlamaya dayanması nedeniyle simülasyon başarılı bir şekilde gerçekleştirilmiştir.

6. Sonuç

Bu bildiride özellikle otonom veya yarı otonom sistemlerin tasarımında ve geliştirilmesinde bulundukları operasyonel ortam şartlarının da dikkate alınarak geliştirilmesinin önemine vurgu yaparak, sistemlerin ve ortamlarının modüler hızlı prototip geliştirme yaklaşımı ile modellenmesi ve simülasyonu bir insansız hava aracı durum çalışması üzerinden anlatılmıştır.

Bu yaklaşımda vurgulanan özellikler arasında modülerlik, yeniden kullanılabilirlik, model tabanlı mühendislik, hızlı prototip geliştirme, sistem ve ortamın birlikte modellenmesi vardır. Bu modelleme ve simülasyon yaklaşımının kullanıldığı insansız hava aracı durum çalışmamızda, modelleme için öznitelikli olay dili (attributed event grammar), simülasyon için de ayrık olay simülasyonu (discrete event simulation) kullanılmıştır. Ayrık olay simülasyon aracı olarak da OMNET++ 4.0 simülasyon ortamından faydalanılmıştır.

Araştırmada elde ettiğimiz sonuçlar göstermektedir ki sistemlerin ve ortamlarının modüler hızlı prototip geliştirme yaklaşımı ile modellenmesi ve simülasyonu sistemin çalışma ortamındaki durumu hakkında oldukça önemli bilgileri kısa bir süre içerisinde elde etmemize olanak sağlayan etkili bir yazılım ve sistem geliştirme yöntemidir. Durum çalışmamız ile görülmüştür ki yaklaşım karmaşık sistemlere de rahatlıkla uygulanabilmekte dahası karmaşık sistemleri sistem ve yazılım geliştiricilerinin daha rahatlıkla anlayabileceği bir hale indirgenebilmesini sağlamaktadır. Ayrıca araştırma ile bu modelleme ve simülasyon yaklaşımının insansız hava aracı geliştirilmesinde rahatlıkla kullanılabileceği sonucuna varılmıştır.

Yukarıda faydalarından bahsettiğimiz yaklaşımın çeşitli kısıtları da bulunmaktadır. Bunların başında modellerden simülasyon koduna geçerken otomasyon olmamasıdır. Hernekadar sistem ve ortamın modellenmesi kodlama aşamasını oldukça hızlandırmakta ve hata oranını azaltmakta olsa bile ideal olan bu işlemin otomasyon ile yapılmasıdır. Gelecekte yapılabilecek çalışmaların başında bu otomasyonun sağlanması gelmektedir.

7. Açıklamalar

Bu bildirinin kapsamında olan direk veya dolaylı tüm fikir, yorum ve görüşler yazarların şahsi fikirleri olup, bağlı bulundukları hiçbir kurumun direk veya dolaylı olarak resmi veya gayriresmi hiçbir görüşünü temsil etmezler. Bağlı bulundukları kurumlar bu bildiriden doğan her türlü sorumluluktan muaftırlar.

8. Kaynaklar

[1] Auguston, M., Michael, J. B. and Shing, M., "Environment Behavior Models for Automation of Testing and Assessment of System Safety", *Information and Software Technology,* (48) 2006. pp. 971-980

[2] Auguston, M., Michael, J. and Shing, M., "New Directions in C2 Software Quality Assurance Automation Based on Executable Environment Models", Command and Control Research and Technology Symposium 2006, Naval Postgraduate School, Monterey, CA, USA, 2006.

[3] Demir K.A., "Modular Prototyping of Systems and Environments Using Models Developed with Attributed Event Grammar", Proc. of. Int. Conf. on Software Engineering Research and Practice 2009, Las Vegas,Nevada, USA, 2009.

[4] Auguston, M., "Program behavior model based on event grammar and its application for debugging automation", Workshop on Automated and Algorithmic Debugging, Rennes, France, 1995
[5] Selic B., "Using UML for Modeling Complex Real-Time Systems", Lecture Notes in Computer Science, Vol. 1474, Springer Berlin / Heidelberg, (1998) 250-260.
[6] Desel J., and Juhas G., "What is a Petri Net? Informal Answers for the Informed Reader", Lecture Notes in Computer Science, Vol. 2128, Springer Berlin / Heidelberg, (2001) 1-25.
[7] Varga A., "OMNeT++ Discrete Simulation System (Version 2.3) User Manual", Technical University of Budapest, Dept. of Telecommunications (BME-HIT), Hungary, Mar. 2002.
[8] Fishman, G.S. "Principles of Discrete Event Simulation." Wiley-Interscience, New York, 1978.
[9] Misra J, "Distributed Discrete Event Simulation", ACM Computing Surveys, March 1986
[10] Object Management Group, OMG SysML Specification, OMG Adopted Specification, OMG document ptc/06-05-04, Needham, MA, May 2006
[11] OMNETT++ Kurusal web sayfası, http://www.omnetpp.org/ Erişim: 24-07-2010
[12] Hoot N., Leblanc L., "Forecasting Emergency Department Crowding: A Discrete Event Simulation", Annals of Emergency Medicine, Vol. 52, No. 2., 2008.
[13] Semini M. and Fauske H.. "Applications of discrete-event simulation to support manufacture logistics decision-making: A survey". Proceedings of the 2006 Winter Simulation Conference, pages 1946–1953, 2006.
[14] Werker G, SaureA, French J, Shechter S. "The use of discrete-event simulation modelling to improve radiation therapy planning processes". Radiotherapy and Oncology 2009.
[15] Banks J., Carson J.S., Nelson B.L., Nicol D.M., "Discrete-Event System Simulation", fourth ed., Prentice-Hall, 2005.
[16] Auguston M., "A language for debugging automation, in Chang", Proc. Sixth Conf. on Software Engineering & Knowledge Engineering, June 1994
[17] Auguston M., "Lightweight semantics models for program testing and debugging information", Proc. Seventh Monterey Workshop: Modeling Software System Structures in a Fastly Moving Scenario, Santa Margherita Ligure, Italy, pp. 23-31, June 2000
[18] Auguston M., Jeffrey C., and Underwood S., "A framework for automatic debugging", in Proc. Seventeenth Int. Conf. on Automated Software Engineering, pp. 217-222 Edinburgh, Scotland, Sept. 2002