FARKLI PROJELERDE GELİŞTİRİLEN GÖMÜLÜ YAZILIMLARIN TEST OTOMASYONU İÇİN DONANIM SİSTEMİ SİMÜLASYONU GELİŞTİRME DENEYİMİ

Ömer Faruk MORALIOĞLU[1] Önder CEZAYİRLİ [2] Murat YILMAZ[3]

[1,2,3] Radar Elektronik Harp ve İstihbarat Sistemleri (REHİS) Grubu, ASELSAN A.Ş. Ankara

[1]moralioglu@aselsan.com.tr, [2]ocezayir@aselsan.com.tr,
[3]muratyilmaz@aselsan.com.tr

Özet. REHIS bünyesinde geliştirilen Elektronik Harp projelerinin Yazılım Test faaliyetlerinde, geliştirilen yazılımların testi için donanım ihtiyaçları baş göstermektedir. Test Altındaki Gömülü Yazılımlar gerçek zamanlı olarak yüksek hızlı veri üreten tümleşik donanım sistemleri ile haberleşmektedir. Bu durum da hem Test Altındaki Gömülü Yazılımın, hem de test amaçlı geliştirilen yazılımların tümleşik donanım sistemleri ile bir arada kullanılması gerekliliğini doğurmaktadır. Bu çözüm yolu hem maliyet hem de yetenek açısından yazılım test mühendisini kısıtlamaktadır. Yazılım test mühendisinin yeterli olgunluğa ulaşmış tümleşik donanım sistemini edinmesi ya oldukça maliyetli bir çözüm olmakta, ya da bu sistemler yazılım ile eş zamanlı olarak geliştirildiğinden, temini mümkün olmamaktadır. Ayrıca bu sistem, benzetim ortamında hazırlanan tüm test senaryolarına uyum sağlayamamaktadır. Bu noktadan hareketle, laboratuar ortamında birebir donanım sistemi benzetimi yapan ve test senaryolarına uyum gösterebilen esneklikte bir gerçek zamanlı gömülü yazılım ihtiyacı doğmuştur. Bu bildiride, geliştirilen gömülü yazılımın arkasındaki motivasyon, gerçeklenmesine ilişkin teknik detaylar, bu yazılım ortamının test otomasyon sistemine [1] entegrasyonu ve yazılım ürün hattı yaklaşımı ile geliştirilmiş yazılımların testlerinde kullanılmasıyla elde edilen kazanımlar anlatılmaktadır.

Anahtar Kelimeler: Yazılım Test, Gerçek Zamanlı Gömülü Yazılım, Test Otomasyon, Simülatör, Donanım Sistemi Simülasyonu, Çevik Geliştirme, Yazılım Ürün Hattı

1 Giriş

REHIS bünyesinde geliştirilen Elektronik Harp projelerinin yazılım geliştirme faaliyetlerinde çevik yöntemler kullanılmakta, çevik geliştirme yöntemleri beraberinde yazılım test faaliyetlerinde de çevik yöntemlerin kullanılma zorunluluğunu getirmektedir. Çevik yöntemlerde, yazılım geliştirme ile birlikte test ve birimlerin bütünlenme faaliyetlerinin iç içe gerçekleştirilmesi gerekmektedir. Şekil 1'de çevik yöntemler kullanılan bir projenin yaşam döngüsü modellenmiştir.

Şekil 1. Çevik Yöntemler Kullanılarak Geliştirilen bir Projenin Yaşam Döngüsü

Görüldüğü gibi projenin her bir yineleme evresinde yazılım test işlemi bu evredeki geliştirme faaliyetine eşzamanlı olarak gerçekleştirilmelidir. Bu durum test alt yapısının her evrede ihtiyaçları karşılayabilecek yeterlilikte olma zorunluluğunu beraberinde getirmektedir.

Geliştirilen yazılımın hedef ortamı da testlerin niteliklerinde belirleyici rol oynamaktadır. Gömülü yazılımların temel geliştirilme amacı fiziksel dünya ile iletişimdir. Bu nedenle, gömülü yazılım geliştirme süreci denetim altındaki fiziksel ürün ile bütünlenmiş olmalıdır[2]. Bunun yanı sıra uygulama yazılımlarından farklı olarak, gömülü yazılım geliştirme süreci, gerçek zamanlı programlamaya ve gerçek zamanlı bir işletim sistemine ihtiyaç duyar. Gömülü yazılım geliştirme sürecine ait

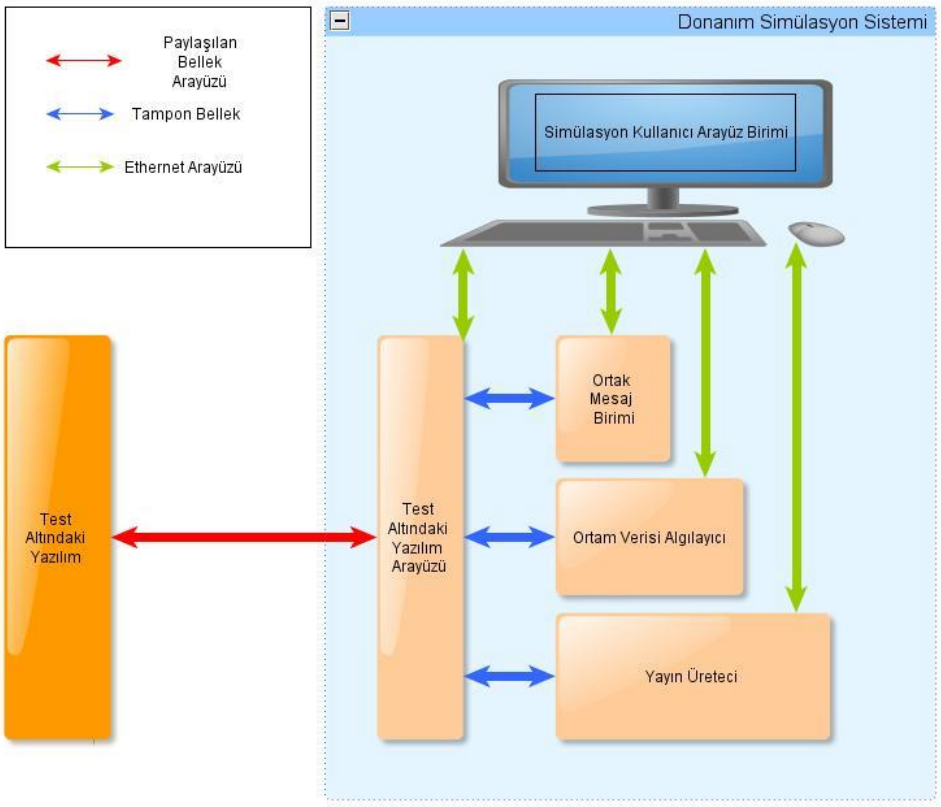
tüm bu nitelikler aynı zamanda yazılım test faaliyetlerine de benzer etkiler yapmaktadır.

Gömülü yazılım testleri, nihai ürün kalitesi ve güvenilirliği için detaylı bir şekilde gerçekleştirilmelidir[3]. Test altındaki yazılımın tüm çevre birimlerinin, *benzetim yapan yazılım birimleri* (Simulation Software Module) kullanılarak simüle edilmesi ve bu yazılım birimlerinin uygun donanım sistemleri üzerinde çalıştırılması gerekmektedir. Fakat bu süreç içerisinde, olası hedef platformların çeşitliliği, platforma bağlı kısıtlar ve test edilen yazılımın haberleştiği çevre birimin tümleşik bir donanım sistemi olması gibi çeşitli zorluklarla karşılaşılmaktadır.

REHIS bünyesinde, çevik yöntemler ile geliştirilmekte olan belirli tipteki gömülü yazılımların çevre birimlerinden mutlaka bir tanesi, tümleşik bir donanım sistemi olmakta ve bu durum, yine çevik yöntemlerin amaçlandığı yazılım test süreçlerinde de bu donanım sistemi ara yüzünün varlığını gerekli kılmaktadır. Yazılım Ürün Hattı yaklaşımı ile geliştirilmiş farklı projelere uyum sağlayabilecek, test edilecek gömülü yazılım ile eş zamanlı olarak geliştirilen donanım sistemine erişim zorunluluğunu ortadan kaldıracak, yazılım test otomasyon altyapısının bir parçası olacak ve test amaçlı farklı girdi verilerine uyum sağlayarak gerçek donanımın gösteremeyeceği esnekliği gösterecek bir simülatörün, gömülü yazılım ortamında gerçeklenmesi bu bildirinin konusunu oluşturmaktadır. Bildirinin ikinci bölümünde gömülü simülasyon yazımlının mimarisinden, alt birimlerinden ve bu birimlerin işlevsel özelliklerinden, üçüncü bölümünde gömülü simülasyon yazılımının otomatik test altyapısı ile entegrasyonundan bahsedilmiş, sonuç bölümünde ise bu deneyim ile birlikte elde edilen kazanımlar ve değerlendirmeler paylaşılmıştır.

2 Gömülü Simülasyon Yazılımı Mimari ve İşlevsel Özellikleri

Gömülü simülasyon yazılımı, tümleşik bir donanım sisteminin benzetimini yapmaktadır. Bu nedenle geliştirilen yazılım, donanım sisteminde olduğu gibi farklı işlevleri yerine getiren Dağıtık Yazılım Sistemi (Distributed Software System) mimari yapısındadır. Dağıtık bir yazılım sistemi, belli bir göreve odaklanmış, düşük maliyetli birçok birimin geliştirilmesine imkân sunar. Aynı zamanda, birimler arası iletişim altyapısının tasarlanması ve çoklu süreçlerin ele alınması gibi zorlukları da beraberinde getirir[4]. Bu bölümde karşılaşılan zorlukların nasıl ele alındığı açıklanmaktadır. Şekil 2'de dağıtık yazılım sistemi mimarisi ile geliştirilen Simülasyon yazılımının şeması bulunmaktadır.



Şekil 2. Tümleşik Donanım Simülasyon Sistemi

Geliştirilen gömülü yazılım, birden fazla işlevsel alt birimden oluşmaktadır. Bu nedenle, bu birimlerin işlevsel öncelikleri ile iletişim yöntemleri tasarlanmalı ve gerçeklenmelidir. Şekil 2'deki şemadan da görüleceği üzere, birimlerin iletişim yöntemleri konusunda birden fazla çözüm yöntemi kullanılmıştır.

***Tampon Bellek* (Memory Buffer);** yazılım sisteminin test altındaki yazılım ile haberleşmesinde kullanılan bir araç görevi görmektedir. Örneğin, Ortak Mesaj Birimi göndermesi gereken anlamlı veriyi, daha önceden kararlaştırılmış bir tampon belleğe işlemekte ve karşılığında bu bellek alanının *işaretçi* (Pointer) bilgisini dönmektedir.

Ethernet Arayüzü; Simülasyon Kullanıcı Arayüzü biriminin diğer birimler ile haberleşmesi amacı ile kullanılmaktadır.

***Test Altındaki Yazılım* (Software Under Test) Arayüzü**, şema üzerinde bir birim gibi görünmesine rağmen, test altındaki yazılım ile haberleşme sırasında, ardı ardına aynı işlemlerin gerçekleştirildiği bir görev prosedürü olarak tanımlanabilir. Bu

prosedür ise, işaretçi bilgisi ile gösterilen anlamlı verinin Test Altındaki Yazılım ile paylaşılmakta olan bellek alanına işlenmesi ve *Paylaşılan Bellek* (Shared Memory) üzerinde belirlenmiş adrese *kesme* (Interrupt) çıkarılması şeklinde gerçekleşmektedir.

Önceliklendirme; kesme temelinde gerçekleştirilmektedir. Gelen kesme işareti doğrultusunda işlenecek olan görev öncelik kazanmakta ve görev sonlanana kadar da başka bir işlem gerçekleştirilmemektedir [5]. Gelen kesme sinyali sonrasında ise birimlerde tanımlanmış *Kesme Hizmet Usul* (Interrupt Service Routine) programları aracılığı ile gerekli işlemler gerçekleştirilmektedir.

Dağıtık mimari ile tasarlanmış simülasyon yazılımını oluşturan birimlerin işlevleri ilerleyen alt başlıkların konusunu oluşturmaktadır.

2.1 Simülasyon Kullanıcı Arayüzü

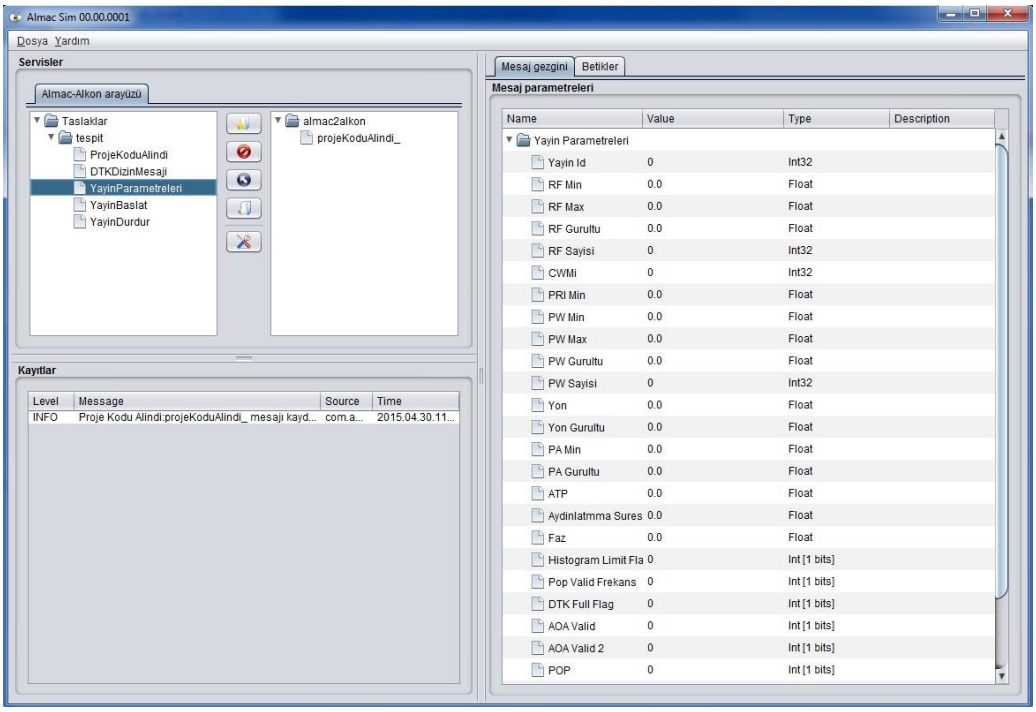
Simülasyon Kullanıcı Arayüzü birimi, testi gerçekleştiren kullanıcının gömülü simülasyon yazılımı ile her türlü etkileşimini gerçekleştiren birimdir. Kullanıcı bu birim ile, simülasyon yazılımının kurulum ayarlarını yapabilmekte, simülasyon yazılımına gönderebileceği mesajları listeleyip istediklerini belirli bir dizine kaydedebilmekte, gönderecek olduğu mesajın tüm parametrelerini görüntüleyip bu parametrelerin test koşum esnasında düzenlemelerini gerçekleştirebilmektedir. Aynı zamanda, simülasyon yazılımının gömülü birimleri ile olan haberleşmesinde gelen ve giden tüm mesajların başlık ve içeriklerini görüntüleyebilmekte ve simülasyon yazılımı ile ilgili durum bilgilerine de erişilmesini sağlamaktadır. Bunlarla birlikte Simülasyon Kullanıcı Arayüzü birimi ile önceden tanımlı bazı test pratikleri, *betikler* (Script) aracılığı ile kaydedilebilmekte ve betikler ile tanımlı test senaryolarının tekrar koşumları gerçekleştirilebilmektedir. Bu şekilde, test senaryolarının Yazılım Ürün Hattı yaklaşımı ile geliştirilmiş farklı yazılımların testlerinde mümkün olduğunca az bir değişiklik ile kullanılması sağlanmıştır.

Simülasyon Kullanıcı Arayüzü ile ilerleyen bölümlerde detaylı anlatılan birimlerin işlevlerinin gerçeklenmesi için gerekli girdiler sağlanmakta ve simülasyon yazılımının test koşumu esnasında durum ve hata gibi bilgilerine de anlık olarak erişilebilmektedir. Şekil 3 ve 4'te bazı kullanım durumlarında Kullanıcı Arayüz biriminin alınmış görüntüleri paylaşılmıştır.

Simülasyon Kullanıcı Arayüz birimi, simülasyon yazılımının gömülü birimleri ile Ethernet üzerinden haberleşmektedir ve bu protokolde istemci olarak yer almaktadır. Haberleşme ve arayüzde kullanılan mesajların ve bu mesajların parametrelerinin tanımlanmasında Cobalt[6] kütüphanesinin servislerinden faydalanılmıştır.

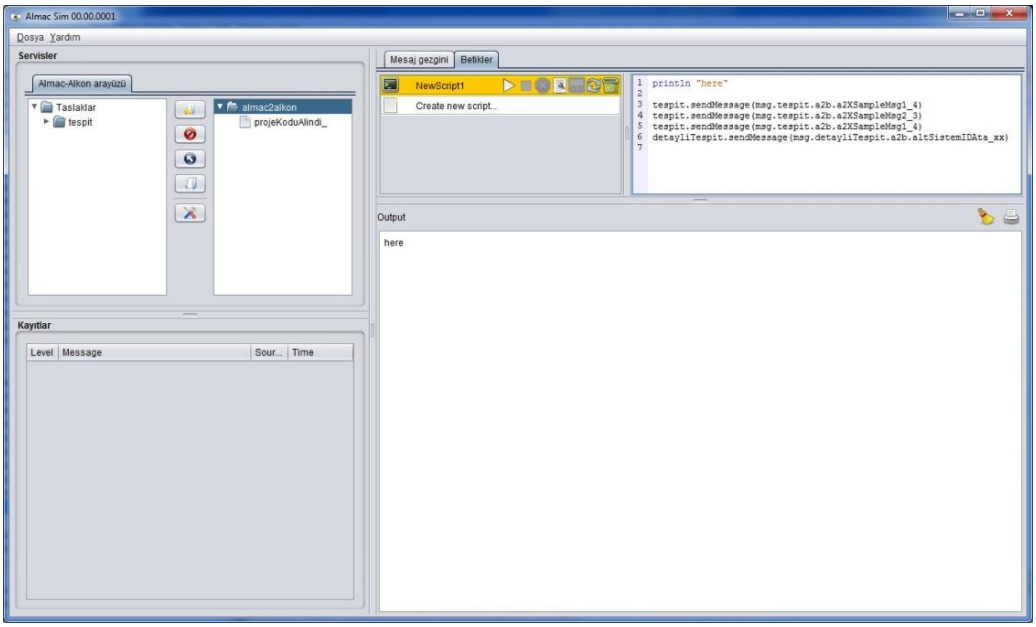
Alınan ve gönderilen mesajların tanımından sonra iletişim amaçlı olarak Alıcı ve Gönderici sınıfları oluşturulmuştur. Bu sınıflar, Cobalt kütüphanesinin yeteneklerini

kullanarak haberleşme protokolünden bağımsız bir şekilde mesaj iletiminin gerçekleştirilmesine imkân sunmaktadır[6].



Şekil 3. Kullanıcı Arayüz Birimi Görüntüsü (Mesaj Gezgini)

Simülasyon Kullanıcı Arayüz birimi, temel olarak verilerin ve çalışma durumlarının, görüntülenmesi/güncellenmesi için geliştirilen Grafiksel Kullanıcı Arayüzü bileşeni ve serileştirilmiş verilerin gönderimini ve gelen verinin ayrıştırılmasını sağlayan Gönderici ve Alıcı sınıfları bileşenlerinden oluşmakta ve bu bileşenlerin birbirleri ile etkileşimli çalışmaları ile işlev görmektedir. Bu yapı ve kullanım özelliği Kullanıcı Arayüz Biriminin, *Model Görünüm Denetleyici* (Model View Controller) *Tasarım Kalıbı* (Design Pattern) kullanılarak geliştirilmesi için uygun şartları sunmaktadır. Bu amaçla Denetleyici arayüzü geliştirilmiş ve bu arayüzü gerçekleyen sınıflar kodlanmıştır. Böylelikle Grafiksel Simülasyon Kullanıcı Arayüzü (Görünüm) ve Alıcı/Gönderici sınıflar (Model) birbirini etkilemeyecek şekilde geliştirilmiş ve verinin işlenmesi ve gösterim işlevlerinin soyutlanması sağlanmıştır. Bu sayede çevik yöntemler ile geliştirilmiş bir yazılımın testinde, gömülü simülasyon yazılımının, test altındaki yazılımda yapılan değişikliklere hızlıca uyarlanabilmesi sağlanmıştır.



Şekil 4. Kullanıcı Arayüz Birimi Görüntüsü (Betikler)

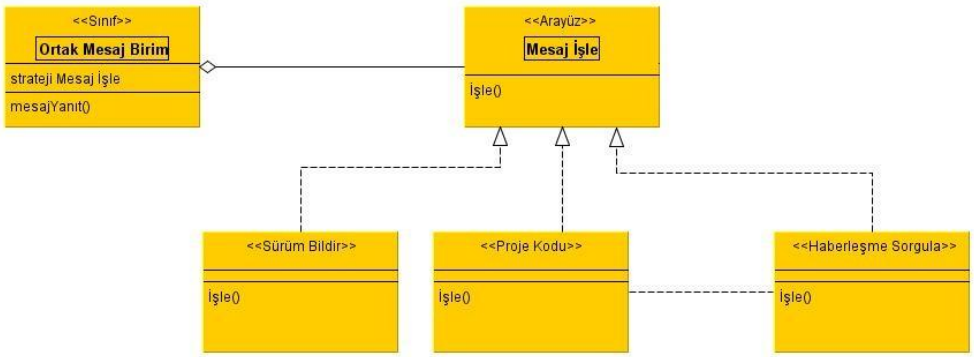
2.2 Ortak Mesaj Birimi

REHIS bünyesinde yürütülen Elektronik Harp projelerinde geliştirilmekte olan yazılım birimleri, yeniden kullanım amacı gözetilerek geliştirilmektedir. Farklı projelerdeki ortak ihtiyaçlar doğrultusunda olabildiğince tekrar kullanımı sağlamak adına çeşitli çalışmalar yapılmıştır. Bu sebeple geliştirme süreçlerinde, yeniden kullanılabilir bir platform ve ürün hattı mimarisinin geliştirildiği alan mühendisliği çalışması yürütülmüş, neticesinde Radar ve Elektronik Harp Fonksiyonel Referans Mimarisi (REFoRM) benimsenmiştir[7].

Test mühendisinin test etmekle yükümlü olduğu yazılım, farklı platformlar için tasarlanmış olmasına rağmen, birçok projede ortak işlevler görmektedir. Yazılımın farklı platformlar için hazırlanmış sürümlerinin etkin biçimde test edilebilmesi için, gömülü simülasyon yazılımı da yeniden kullanım amacı gözetilerek geliştirilmelidir.

Ortak Mesaj Birimi, yeniden kullanım özelliğini sağlamak ve yazılım ürün hattı yaklaşımının kazanımlarından faydalanmak adına geliştirilmiştir. Tüm projelerde ortak olacak şekilde alınıp verilen mesajlar için gerekli mesajlaşma altyapısını gerçekleme adına, Ortak Mesaj Birimi, ortak olan mesajların işlenip, karşılığında gerekli işlevlerin yerine getirildiği yazılım alt birimidir.

Bu birim temelde aynı prensibi, gelen farklı mesaj kodlarına göre değişecek şekilde farklı şekillerde uygulamaktadır. Görev tanımı dolayısı ile bu birimin tasarlanması esnasında "Strateji Tasarım Kalıbı"nın kullanılması uygun görülmüştür. Ortak Mesaj Birimi, projeler için ortak olan mesajların işlenmesi için geliştirilen algoritma ailesini değişmeli olarak kullanıp, test altındaki yazılıma gerekli işlevi sunmakla yükümlü birimdir. Bu birime ait UML sınıf diyagramı Şekil 5'te verilmiştir.

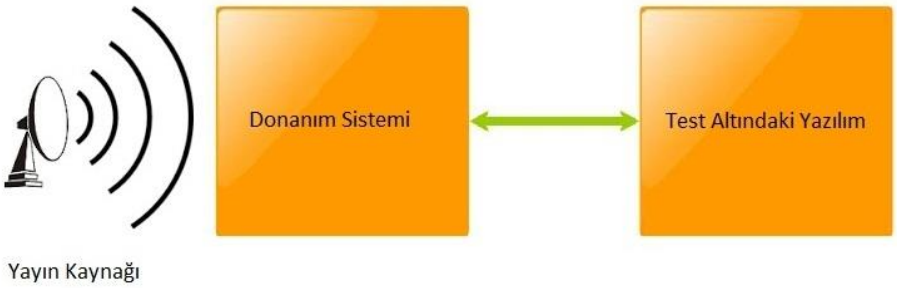


Şekil 5. Ortak Mesaj Birimi'ne ait UML Sınıf Diyagramı

Böylelikle, gömülü simülasyon yazılımı kullanılmakta olduğu projeden bağımsız olarak her ortak mesaja karşılık bu mesaja ait yanıtı verebilmektedir. Aynı zamanda kullanılan geliştirme yöntemi sayesinde aynı `Mesaj İşle` arayüzünün farklı yürütme işlevleri gerçeklenmiş ve birimin geliştirme ve idamesinde tasarımcıya kolaylık sunması sağlanmıştır. Böylece Yazılım Ürün Hattı yaklaşımının, test altındaki yazılımların geliştirilmesinde olduğu gibi gömülü simülasyon yazılımında da gerçeklenmesi sağlanmıştır.

2.3 Ortam Verisi Algılayıcı

Benzetimi yapılan tümleşik donanım sistemi, gerçek çalışma ortamında, etrafta var olan yayınları dinlemekte, önceden yapılandırılmış antenleri aracılığı ile yayınların, frekans, genlik, yön, polarizasyon gibi bilgilerini anlamlandırarak yazılım birimi ile olan arayüzüne iletmektedir. Dolayısıyla testi gerçekleştirilen yazılımın davranışlarında ortamda bulunan yayın kaynaklarının özellikleri de etkili olmaktadır. Bu yapı basitçe Şekil 6'daki gibi modellenebilir.



Şekil 6. Test Altındaki Yazılımın Gerçek Ortam Modellemesi

Test altındaki yazılımın, sahada operasyonel işlevlerini doğru bir şekilde gerçekleştirdiğinden emin olmak için, testlerinin masa başı ve laboratuar ortamında gerçeğe yakın senaryolar işletilerek yapılması önem taşımaktadır. Testlerin bu nitelikte tasarlanabilmesi için ise, Şekil 6'da görüldüğü gibi, ortamda var olabilecek yayın kaynaklarının benzetiminin yapılması zorunlu olmaktadır.

Bu motivasyon ile REHIS bünyesinde, bir alanda dağınık olarak yerleştirilecek olan alıcı platformun ve tehditlerin konumlarına, tehditlerin yayın tiplerine ve çalışma zamanlarına uygun şekilde RF yayın ya da *Darbe Tanımlayıcı Kelime* (Pulse Description Word, DTK) verilerini üretecek bir "Ortam Simülatörü" geliştirilmiştir. [8]

Gömülü simülasyon yazılımı iki farklı durumda çalışabilecek şekilde geliştirilmiştir. Bu çalışma durumlarının birincisinde, Ortam Simülatörü verisini algılayıp uygun DTK yapısına dönüştürmektedir. Çalışma Durumu bilgisi gömülü yazılıma Simülasyon Kullanıcı Arayüzü birimi tarafından Ethernet üzerinden iletilmektedir. Gömülü simülasyon yazılımı Ortam Verisi Algılayıcı alt birimi sayesinde, ortam simülatörünün üretmiş olduğu veriyi *ayrıştırıp* (Parse) test altındaki yazılımın anlayabileceği DTK yapısına dönüştürmektedir.

2.4 Yayın Üreteci

İkinci çalışma durumunda, gömülü simülasyon yazılımı, genlik, frekans ve periyot gibi parametrelerini kullanıcının belirlediği bir yayını üreterek DTK verisi oluşturabilmektedir. Bunun için kullanıcı, test senaryosunun başlatılmasından önce çalışma durumunu uygun olarak atamakta ve Test senaryosunun koşumu esnasında da, bir veya birden fazla yayını, bu yayınların her birine bir kimlik bilgisi atayarak, başlatabilmekte ve daha önceden başlatılmış olan yayınları durdurabilmektedir. Bu yetenek ile testi gerçekleştiren kullanıcının kendi belirleyeceği test senaryolarını

yaratması ve gerçek ortamda oluşturulması zor durumların laboratuar ortamında rahat bir şekilde gerçekleştirebilmesi sağlanmıştır.

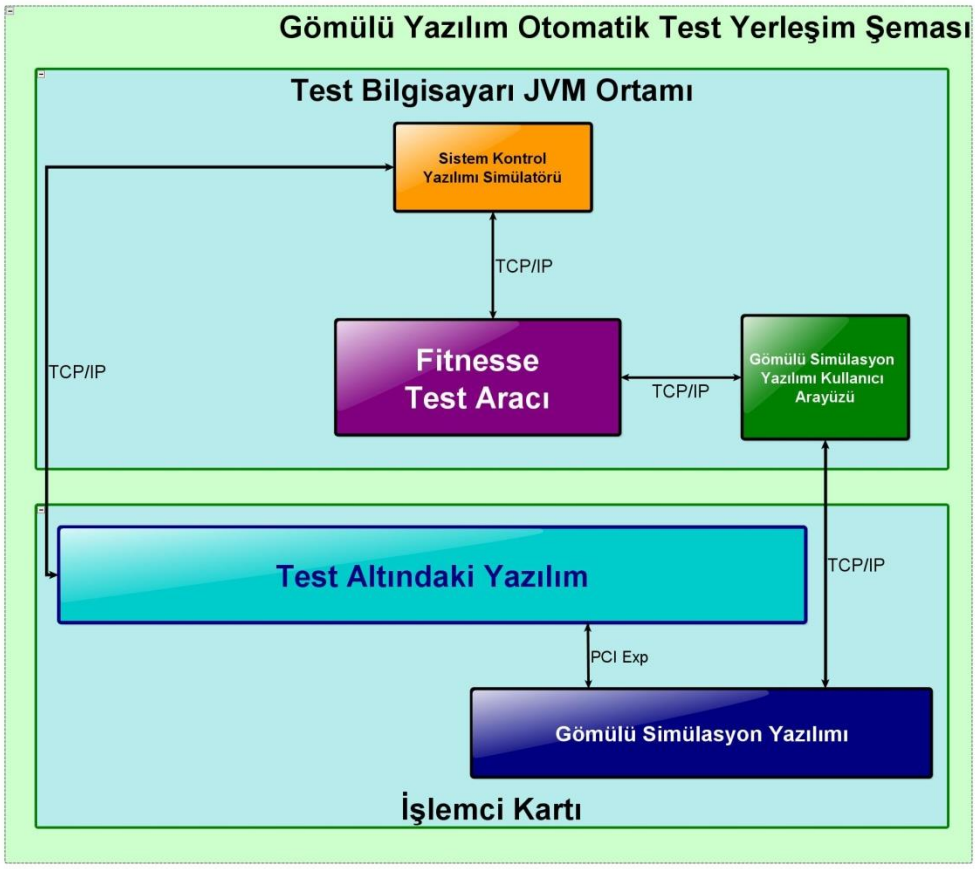
Kullanıcı ortamda olabileceğini tasarladığı yayına ait parametreleri belirledikten sonra bu bilgileri Ethernet protokolü üzerinden simülasyon yazılımına iletmektedir. Bu parametrelerin alınmasını takiben, Yayın Üreteci alt birimi aracılığı ile test altındaki yazılım ile olan arayüzde tanımlı DTK yapısına uygun veri üretilir. Yazılan her bir DTK verisine ait kimlik bilgisi ve yazıldığı zamana ait zaman etiketi de eklenir. Test altındaki yazılımın, üretilmiş bu DTK verilerinin kendisi ile olan mesaj arayüzüne aktarımını tetikleyecek mesajı atmasının ardından, gömülü simülasyon yazılımı, istenen özellikteki yayını var olan yayınların içinden çıkararak paylaşılan bellek arayüzüne aktarır. Böylece, test ortamında tasarlanmış yayın bilgilerini test altındaki yazılıma iletmiş olur. Koşu esnasında kullanıcı daha önceden yaratılmış yayınlardan herhangi bir veya birkaçını, kimlik bilgisini belirterek durdurabilir. Bu durumda durdurulan yayın bilgisinin test altındaki yazılım arayüzüne aktarımı son bulur.

Test altındaki yazılım ile olan arayüzde tanımlanmış yapıdaki DTK verisinin üretilip aktarılması projeye bağlı olarak değişkenlik göstermektedir ve gerçek sistemin nihai ortamında gerçekleştirdiği en önemli kullanım durumudur. Bu işlevin testleri kritik bir öneme sahiptir ve bu testlerin laboratuar ortamındaki esnek koşullar dâhilinde gerçekleştirilmesi açısından Ortam Verisi Algılayıcı ve Yayın Üreteci alt birimleri önemli rol oynamaktadır.

3 Otomatik Test Altyapısı ile Entegrasyon

Test altındaki yazılımın doğrulama sürecinin otomatik olarak yapılması hedeflenmiştir. Simülasyon yazılımı geliştirme çalışmaları da bu doğrultuda, otomatik test altyapısına imkân verecek şekilde planlanmıştır. Otomatik test aracı olarak Fitnesse[9] adlı araç kullanılmaktadır[1]. Fitnesse, açık kaynak kodlu, Wiki tabanlı bir test aracıdır.

Fitnesse testleri, test altındaki yazılıma mesaj aktarımını ve verilerin simülatörler üzerinden gönderilmesini tetikler. Test altındaki yazılım, ilgili işlevleri yerine getirdikten sonra, bağlı olduğu simülatörlere bu işlevin tamamlandığına dair mesaj ve veriyi gönderir. Fitnesse aracı da, simülatörlerden bu veri ve mesajları alarak değerlendirir. Eğer mesajlar ve veriler beklenen şekilde gelmiş ise test adımı başarılı, beklenenden daha farklı gelmiş ise test adımı başarısız olarak nitelendirilir. Başarısız bir test adımı varsa, hatanın kaynağı tespit edilip gerekli düzeltme yapılarak ilgili test tekrarlanır. Şekil 7'de tasarlanmış otomatik test yapısına ait şema görülmektedir.



Şekil 7. Otomatik Test Yapısı

4 Kazanımlar ve Değerlendirme

Bu bölümde, gömülü simülasyon yazılımı geliştirilmeden önce, tümleşik donanım sistemi kullanılarak yapılan testlerde alınan ölçümler ile gömülü simülasyon yazılımı kullanılarak yapılan testlerde alınan ölçümler karşılaştırılacaktır. Alınan ölçümler ise, test altındaki yazılıma ait test kapsamı, yazılım seviyesindeki testlerde tespit edilen hata sayısı, adam saat bazında test yürütme süresi şeklinde olacaktır. Tablo 1' de A, B ve C projelerinde gömülü simülasyon yazılımı kullanılmadan önce ve kullanıldıktan sonra gerçekleştirilen yazılım seviyesindeki testlerde alınan ölçümler yer almaktadır.

	Test Kapsamı	Hata Sayısı	Yürütme Süresi
	Gömülü Simülasyon Yazılımı Kullanılmadan Önce		
A	% 68	43	344
B	% 74	37	432
C	% 71	40	392
	Gömülü Simülasyon Yazılımı Kullanıldıktan Sonra		
A	% 91	84	88
B	% 96	79	112
C	% 94	82	96

Tablo 1. Gömülü Simülasyon Yazılımının Kullanımı ile İlgili Alınan Ölçümler

Gömülü simülasyon yazılımının kullanımı, öncelikle testlerin laboratuar ortamında yapılabilirliğini sağladığından test kapsamında bir artışa sebep olmuştur. Yazılım seviyesi testlerdeki kapsamın artışı paralelde, tespit edilen hata sayısında da artışa sebebiyet vermiştir. Testlerin yürütme süresinde görülen ciddi azalma ise otomatik test altyapısı ile entegrasyon özelliğinin getirdiği sonuçtur. Gömülü simülasyon yazılımının geliştirilmesinin getirdiği en büyük maliyet, geliştirme süresinin uzunluğu olmuştur. Toplam 5 adam aylık bir sürede geliştirilmiş olan simülasyon yazılımının, maliyetini ileri aşamada kullanılması planlanan projeler ile birlikte telafi etmesi beklenmektedir.

5 Sonuç

Gömülü simülasyon yazılımı kullanılarak test altındaki yazılımın haberleştiği tümleşik donanım sisteminin temin edilme zorunluluğu ortadan kaldırılmış ve yazılım testlerinin tamamının laboratuar ortamında gerçekleştirilebilmesi mümkün olmuştur. Gerçek donanım kullanılarak test altındaki yazılıma gönderilemeyecek farklı veri girdileri gönderilerek test senaryolarının çeşitlendirilmiştir. Gömülü simülasyon yazılımı, test altındaki yazılımın testlerinin otomatize edilmesi için kurulan sistemin bir parçası olarak çalışmakta ve test senaryolarına paralel olarak güncellenebilecek esnekliği sunmaktadır. Model Görünüm Denetleyici tasarım kalıbı kullanılarak gömülü simülasyon yazılımı, çevik yöntemler ile geliştirilmiş bir yazılımın testinde, test altındaki yazılımda yapılan değişikliklere hızlıca uyarlanabilmiştir. Strateji Tasarım Kalıbı ve test betikleri kullanılarak Yazılım Ürün Hattı yaklaşımının, test altındaki yazılımların geliştirilmesinde olduğu gibi gömülü simülasyon yazılımında da gerçeklenmesi sağlanmıştır.

Gömülü simülasyon yazılımı, şu anda aktif olarak üç projede kullanılmaktadır, yakın gelecekte iki projede daha kullanılacaktır. Orta vadede, yeniden kullanılabilirlik özelliğinden faydalanarak bu sayının artması beklenmektedir.

Teşekkür

Yazarlar, işlevlerin gerçeklenmesi sırasında desteklerinden ötürü Ş. Fırat ADA, Eda GÜRLER ve Şafak ŞEKER'e, altyapı geliştirme esnasında cömert yardımlarından ötürü Uğur ZÖNGÜR ve S. Tuncer ERDOĞAN'a teşekkür eder.

Kaynaklar

1. E. Gürler, M. Yılmaz, "Büyük Ölçekli bir Gömülü Yazılımın Geliştirme ve Otomatik Test Deneyimi", UYMS'14
2. J. M. Morris," Software Industry Accounting", pp. 1-10, 2001.
3. E. Cota," Embedded software testing: what kind of problem is this?", PPGC, EDAA, 2010.
4. K. S. Mishra, A. K. Tripathi, "Some Issues, Challenges and Problems of Distributed Software System", IJCSIT, 2014
5. "https://en.wikipedia.org/wiki/Interrupt", Erişim Tarihi 22/04/2015
6. U. Zöngür, S.T. Erdoğan, "Cobalt: Test Uygulamaları için Protokol Kütüphanesi", UYMS'14
7. O. Aktuğ, "REFoRM'da Yazılım Ürün Hattı Mühendisliği Uygulamaları", Aselsan TTEK 2012
8. G.Ç. Aslan, K. Şen, "Radar Elektronik Harp Sistemleri için ED Ortam Simülatörü", Aselsan TTEK 2013
9. Object Mentor Group, Fitnesse, www.fitnesse.org, Erişim tarihi: 05/05/2015