Bir Gömülü Yazılım Bileşeninde Birim Testlerin Google Birim Test Aracı'yla Ana Sistemde Yürütülmesi

Özgür KIZILAY, Yuşa ÖZ

REHİS Elektronik Harp Görev Yazılımları Müdürlüğü,
ASELSAN A.Ş., Ankara, Türkiye
{ozgurk,yusaoz}@aselsan.com.tr

Özet Gömülü yazılımlarda birim testlerin hedef platformda yürütülmesi, gömülü platformlar için hazır test çerçevelerinin kısıtlı olması, derleme-bağlama-koşturma döngüsünün pratik olmaması ve hedef platforma bağlı kalınması gibi sebeplerden dolayı sorunlu bir konu olmuştur. Test edilecek yazılım bileşeni gerekli koşulları yerine getiriyorsa birim testlerin hedef platform yerine ana sistemde yürütülebilmesi mümkün olup daha pratik bir yöntem olarak karşımıza çıkmaktadır. Bu bildiride, açık kaynaklı bir birim test aracı olan Google Test kullanılarak bir gömülü yazılım bileşeninin birim testleri ana sistemde yürütülmüş ve edinilen deneyimler aktarılmıştır.

Abstract Execution of unit tests on target platform in embedded software has been a problematic issue due to the causes such as limited ready made test frameworks for embedded platforms, unpractical compile-link-run loop and dependence to the target platform. If the software component to be tested fulfills necessary conditions, it may be possible that unit tests can be executed on host system instead of target platform and this may be considered as a more practical method. In this paper, using Google Test which is an open source unit test tool, unit tests of an embedded software component are executed on host system and gained experiences are introduced.

Anahtar Kelimeler: Birim test, gömülü yazılım, ana sistem, hedef platform

1 Giriş

Gömülü yazılımlar, gömülü sistemlerde bulunan özel amaçlı bilgisayarlarda koşan ve üzerinde koştuğu sisteme özel görevler yerine getiren yazılımlardır. Son yıllarda hayatın her alanında gömülü sistemlerin kullanımı artmıştır. Araç kontrol sistemleri, tüketici elektroniği, aviyonik sistemler, iletişim sistemleri, uzaktan algılama sistemleri, akıllı evler ve savunma sistemleri gömülü sistemlerin popüler uygulama alanlarından bazıları olarak sayılabilir [1]. Bu farklı alanlardaki uygulamaların farklı gereksinimleri vardır. Gömülü sistemlerin parçası olan gömülü

yazılımların da bu gereksinimleri karşılaması gerekmektedir. Gereksinimler, fonksiyonel ve fonksiyonel olmayan gereksinimler olmak üzere iki gruba ayrılabilir. Hesaplamalar, teknik detaylar, veri idaresi ve işlemesi fonksiyonel gereksinimlerin kapsamında ele alınır [2]. Fonksiyonel olmayan gereksinimler ise daha çok işleyişi etkileyen gereksinimlerdir [3]. Örneğin en kötü işletim zamanları ve kaynak kullanım kısıtlarını belirten gereksinimler fonksiyonel olmayan gereksinimlerdir.

Fonksiyonel olmayan gereksinimlerden kaynak kullanımıyla ilgili olanların karşılanması çoğu zaman yazılımın üzerinde koştuğu platforma bağlıdır. Dolayısıyla eğer bu gereksinimler test yapılarak doğrulanacaksa bu testlerin nihai hedef platformda yürütülmesi gereklidir. Fonksiyonel gereksinimlerin testlerle doğrulanması için ise gereksinimlerin gerçeklenme yöntemlerine bağlı olmakla birlikte çoğu zaman testlerin hedef platformda yürütülmesi gerekli değildir. Bu durum, fonksiyonel gereksinimlerin testlerinin ana sistemde yürütülmesine olanak sağlamaktadır.

Fonksiyonel gereklerin karşılandığının tespiti birim testlerle yapılabilmektedir. Birim test, yazılımda belli bir işlevi yerine getiren kaynak kod birimlerinin bu işlevi yerine getirdiğinin test edilmesi ve kullanıma hazır olduğunun tespitidir [4]. Gömülü yazılımlarda birim testlerin yürütülmesi sorunlu bir konu olarak karşımıza çıkmaktadır. Bu durumun sorunlu olarak nitelendirilmesindeki ilk neden, genel amaçlı bilgisayar platformları (Örnek:PC) için bulunan hazır birim test çerçevelerinin gömülü platformlar için bulunmasının zor olmasıdır. Gömülü platformlarda derleme-bağlama-koşturma döngüsünün pratik olmaması da ikinci neden olarak karşımıza çıkmaktadır. Birim testlerin hedef platformda yürütülmesinin sorunlu olarak değerlendirilmesine neden olan son sebep olarak hedef platforma bağımlı kalınması gösterilebilir. Bazı projelerde gömülü yazılımın üzerinde koşacağı platform, gömülü yazılımın geliştirilmeye başlamasından daha sonra hazır hale gelebilir. Böyle durumlarda yazılım testlerinin yürütülmesi gecikecek ve yazılım doğrulanmadan geliştirilmeye devam edecektir.

Bir gömülü yazılım bileşeni için oluşturulan birim testlerin yukarıda anlatılan muhtemel sorunların yaşanmaması için ana sistemde yürütülmesi sonucu edinilen tecrübeler ve kazanımlar bu bildirinin esas konusudur. İlerleyen bölümlerde Google Test Aracı tanıtılmış, neden bu aracı seçtiğimizden bahsedilmiş, test edilen gömülü yazılım bileşeni hakkında bilgi verilmiş, test yöntemi aktarılmış ve son olarak bu yöntemle elde edilen kazanımlardan bahsedilmiştir.

2 Google Birim Test Aracı Nedir?

Google birim test aracı [5], C++ programlama dili için 'xUnit' mimarisine dayanan bir birim test kütüphanesidir. 'C' ve 'C++' kaynak dosyalarında, modifikasyon gerekmeksizin birim test yapılabilmesine imkan vermektedir. Google birim test aracının testleri nasıl gerçekleştirdiği aşağıda kısaca anlatılmıştır. $TEST()$ makrosu, Google birim test aracına bir test yazıldığını belirtmektedir ve Google birim test aracında test edilecek birimler $TEST()$ makrosunun içine yazılmaktadır. Örnek bir test makrosu örnek olarak aşağıda verilmiştir.

```
TEST(CarpmaTest, CarpimDogruMu)
{
  EXPECT_TRUE(carp(3,3)== 9);
  EXPECT_FALSE(carp(4,3)== 12);
  EXPECT_EQ(10, carp(3,3));
}
```

$carp(x, y)$ fonksiyonunun aldığı iki argümanı çarparak işlem sonucunu döndüğünü kabul edersek, verilen örnekte, çarpım işlemlerinin doğruluğunu test eden bir test durumu oluşturulmuştur. $TEST()$ makrosunun içine yazılan ilk argüman test durumunun adı, ikinci argüman ise testin adıdır. $EXPECT\_TRUE()$, içinde yazılan ifadenin doğruluğunu test etmekte ve bir sonraki cümleye geçmektedir. $EXPECT\_FALSE()$ cümlesi ise $EXPECT\_TRUE()$ cümlesinin mantıksal olarak tam tersi şeklinde işlemekte, test cümlesinde içine yazdığımız ifadenin yanlışlığı kontrol edilmekte ve bir sonraki cümleye geçilmektedir. $EXPECT\_EQ()$ cümlesi de aldığı iki argümanın birbirine eşit olup olmadığını kontrol etmektedir. İlk argüman beklenen değer, ikinci argüman ise gerçek değerdir. Bu cümleden sonra test bitmektedir. Google birim test aracı, test etme işleminde kaç tane test durumunun çalıştırıldığını ve kaç tane testin başarılı olup kaç testin başarısız olduğunu sonuç olarak göstermektedir. Başarılı testler "$PASSED$" olarak, başarısız testler ise "$FAILED$" olarak gösterilmektedir. Testlerin içinde bulunan başarısız test cümleleri için beklenen ve gerçek değer ekranda gösterilerek kullanıcıya yol gösterilmektedir. Ayrıca Google birim test aracı, test durumlarını koşturmadan önce test durumuna özel test koşullarının oluşturulması için kullanıcıya bir "fixture" sınıfı sunmaktadır. Bu sınıfta $SetUp()$ ve $TearDown()$ fonksiyonları sunulmuştur. $SetUp()$ fonksiyonu ilgili test koşmadan hemen önce ve $TearDown()$ fonksiyonu ise testin bitmesinden hemen sonra otomatik olarak çağrılmaktadır. Bu fonksiyonların içi kullanıcı tarafından doldurulmaktadır. Kullanıcı "fixture" sınıfı yardımıyla test için gerekli bağlamı yaratmakta ve test bittikten sonra da teste özel yarattığı bağlamı temizlemektedir.

Google birim test aracı, yukarıdaki testin içine yazılmış test cümleleri dışında birçok test cümleleri barındırmaktadır ve her test cümlesinin test planlamasındaki etkisi farklıdır. Örneğin; $ASSERT\_EQ()$ fonksiyonu iki argüman alır ve bu iki argümanın birbirine eşit olup olmadığını kontrol eder. Bu iki argüman birbirine eşit değil ise test başarısız olur ve diğer hiçbir test yapılmadan tüm test planlaması bitirilir. Bu gibi değişik işlevleri olan test cümlelerinin varlığı, kullanıcının yapacağı testlere işlevsellik katmakla beraber geliştiricinin özgün ve çeşitli test planlaması yapmasına olanak sağlamaktadır.

3 Neden Google Birim Test Aracı?

Birim testlerimizde Google birim test aracını seçmemizdeki nedenlerden ilki Google birim test aracının ücretsiz ve açık kaynaklı bir kütüphane olmasıdır. Ücretsiz olması, yazılım geliştiricisini ekonomik bir yükten kurtarmaktadır. Kütüp-

hanenin açık kaynaklı olması ise kütüphaneye erişimi ve kullanımı kolaylaştırmaktadır. Google birim test aracını seçmemizdeki diğer bir neden ise güvenilir bir birim test aracı olmasıdır. Google dahil dünyanın birçok büyük teknoloji ve bilişim firması birim testleri için bu aracı kullanmaktadır. Chrome tarayıcısının ve Chrome işletim sisteminin arkasındaki Chromium projeleri, LLVM derleyici, OpenCV kütüphanesi gibi projelerde Google birim test aracı kullanılmaktadır [5]. Bu durum da Google birim test aracının güvenilir olduğunun bir göstergesidir. Google birim test aracı, Linux, MAC OS X, Symbian gibi birçok platform üzerinde koşmaktadır. Bu özelliği ile değişik işletim sistemine sahip ana platformlar üzerinde de birim test işlemi gerçekleştirilebilmektedir. Ayrıca, XML test raporu üretebilmesi, parametrelerin değeri veya tipi için test yapabilmesiyle Google birim test aracı, birim testler için tercih edilen bir yazılım haline gelmiştir [5]. Google birim test aracında testleri çalıştırmak da oldukça kolaydır. Google birim test aracının kendi kütüphanesinde önceden tanımlı olan RUN_ALL_TESTS makrosu ile tüm testler koşturulabilmektedir. Testleri kolayca koşturma özelliği sayesinde yazılım geliştiricileri tarafından tercih sebebi olmaktadır [6].

4 Test Edilen Yazılım Modülü

Birim testlerinin ana sistemde yürütülmesi deneyimlenen yazılım bileşeni olan veri yönetimi bileşeni, büyük bir elektronik harp projesinin elektronik taarruz alt sistemi gömülü kontrol biriminde yer alan ve özel formatlı veri tablolarının yazılımdaki organizasyonundan sorumlu olan yazılım bileşenidir. Veri yönetimi bileşeni veri tablolarının okunması, saklanması ve veri tabloları üzerinde sorguların yürütülmesi işlevlerini yerine getirir. Bu bileşen 40'a yakın farklı tablo tipini ve birden fazla tabloyu girdi olarak alabilen 40'a yakın sorguyu barındırmaktadır. Bu bileşenin birim testlerini ana sistemde yürütmeyi tercih etmemizin sebepleri; bu bileşenin içerdiği veri tabloları okuma ve sorguların hataya açık kısımlar olmakla birlikte yoğun birim testlere tabi tutulmaları gerekliliği, bileşenin modüler yapıda olması ve bileşenin hedef platforma bağlı olmamasıdır.

5 Birim Test Ortamı

Veri yönetimi bileşeni açık kodlu Eclipse geliştirme ortamında C++ diliyle geliştirilmiştir. Eclipse için Aselsan bünyesinde geliştirilmiş bir eklenti, kaynak kodlarımızı hedef platformda kullandığımız gerçek zamanlı işletim sistemi olan VxWorks araç zinciriyle kolayca derlememizi sağlamaktadır. Bu eklentinin yardımıyla aynı geliştirme ortamında hem hedef platform için hem de ana sistem olarak kullandığımız Windows işletim sistemine sahip bilgisayar için derleme yapma yeteneğine sahip olmaktayız. Aynı çalışma alanında yer alan projelerden veri yönetimi bileşeni projesi hedef platform için derlenmekte, veri yönetimi birim test projesi ise ana sistem için derlenmektedir. Birim test projesine, bileşenin kaynak kodları referans olarak eklenmektedir. Böylece orijinal kaynak kodlar, müdahale riski olmadan ana sistem için derlenebilmektedir. Bileşen bünyesinde

geliştirilen fonksiyonlar test için hazır hale geldikçe geliştirmeye paralel olarak testler de yürütülmüştür.

6 Kazanımlar

Veri yönetimi bileşeninin birim testlerinin hedef platform yerine ana sistemde yürütülmesi, biz geliştiricilere çeşitli faydalar sağlamıştır. Sağlanan en büyük fayda zaman kazancı olmuştur.

Bir gömülü sistem yazılımında derleme-bağlama ana sistemde yapılmakta, koşturma hedef platformda yapılmaktadır. Derleme-bağlama sonucu oluşan yürütülebilir dosyalar hedef platforma yüklenir ve hedef platformda yürütülür. Yazılım, hedef platform üzerinde istenilen şekilde çalışmıyorsa hatalar giderildikten sonra hedef platformda tekrar yürütülmelidir. Gerek birim testler yürütülürken gerekse de birim test sonuçları uyarınca tespit edilen hataların takibi/düzeltmesi yapılırken kod defalarca derleme-bağlama-koşturma döngüsüne girmektedir. Bu döngü sırasında platform değiştirmenin getirdiği zaman kaybı, bu döngünün birçok kez tekrarlandığı göz önüne alındığında ciddi boyutlara ulaşmaktadır. Veri yönetimi bileşeni birim testlerinde kullanılan yöntem, bizi bu maliyetten kurtararak bize zaman kazandırmıştır. Birim testlerin yürütülmesi için yaşanan derleme-bağlama-koşturma döngüsü bütünüyle ana sistemde yapılmıştır.

Birim testlerin hedef platform yerine ana sistemde koşturulmasının zaman kazancından başka getirdiği bir avantaj ise hata ayıklama kolaylığıdır. Gömülü sistemler üzerinde çalışan yazılımlarda hata ayıklamak çoğu geliştirme ortamında mümkün olmamaktadır. Birim testler yürütülürken, PC ortamındaki geliştirme ortamlarında sunulan hata ayıklama işlevleri kullanılarak verimli bir hata tespit etme süreci gerçekleştirilmiştir.

Birim testlerin ana sistemde yürütülmesinin bize kazandırdığı en büyük faydalardan biri de test edilen kodun işletim kapsamını, yani test edilen kodun hangi kısımlarının koşturulduğunu, görmeye olanak sağlamasıdır. Özellikle çok fazla koşul ve döngü kısımları içeren uzun kodlarda her bir koşulun veya döngünün koşturulduğunu anlamak zor olmaktadır. Ana sistemde kullandığımız geliştirme ortamı olan Eclipse aracına entegre bir şekilde çalışan GCov [7] eklentisi, bize koşulun hangi kısımlarına girildiğini göstererek hazırladığımız testler ile kodun uygulanmasını beklediğimiz kısımların gerçekten uygulanıp uygulanmadığını bilmemize olanak sağlamaktadır. Ayrıca bu eklenti yardımıyla kodun işletim kapsamının yüzde bilgisi de kullanıcıya sağlanmaktadır. Böylece projenin ne kadarlık bir oranının teste tabi tutulduğu görülebilmektedir.

7 Sonuç

Birim testlerinin ana sistemde yürütülmesi 6. bölümde anlatılan faydalar göz önüne alındığında verimli bir pratik olarak tecrübe edilmiştir. Kullanılan yöntemle, gömülü yazılımlarda birim test süreci hem daha kolay uygulanır bir hale gelmekte hem de süreç daha az zaman almaktadır. Böylece gömülü yazılım geliştiricilerinin uygulamaktan imtina ettikleri birim test süreci uygulanabilir hale

gelmekte ve bu sürecin uygulanması yazılımın kalitesini arttıran bir faktör olarak yazılım sistemine katkı sağlamaktadır. Bu tecrübemiz sonrasında ekibimiz bünyesinde geliştirilen gömülü yazılımlarda, bu pratiğin uygulanmasına karar verilmiştir.

Bu çalışmadaki sonraki adım, hedef platforma bağlı kodlar bulunduran gömülü yazılım bileşenlerine ait birim testlerin de ana sistemde yürütülebilmesi için bir hedef platform soyutlama katmanının yazılmasıdır. Bu soyutlama katmanıyla, hedef platforma özel fonksiyonlar ana sistem ortamında taklit edilerek aynı işlevsel sonucu vermiş olacaklardır. Bu çalışmanın sonucu olarak birim testlerin ana sistemde yürütülmesi için bileşenlerin hedef platforma özel kod bulundurmaması gereği ortadan kalkmış olacaktır.

Kaynaklar

1. S.Edwards, L.Lavagno, A.Lee, E., Sangiovanni-Vincentelli, A.: Design of Embedded Systems: Formal Models, Validation and Synthesis. In: Proceedings of the IEEE. vol. 85, pp. 366–390 (1997)
2. United States Government US Army: Systems engineering fundamentals. (PDF), ISBN 978-1484120835 (2001)
3. Nixon, B., Chung, L., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Springer US (2000)
4. A.Kolawa, D.Huizinga: Automated Defect Prevention: Best Practices in Software Management, p. 426. Wiley-IEEE Computer Society Press, ISBN 0-470-04212-5 (2007)
5. Google Test (accessed May 30, 2016), https://github.com/google/googletest
6. A Quick Introduction to the Google C++ Testing Framework (accessed May 30, 2016), http://www.ibm.com/developerworks/aix/library/au–googletestingframework.html
7. GCov - A Test Coverage Program (accessed June 8,2016), https://gcc.gnu.org/onlinedocs/gcc/Gcov.html