ULAK-SPY: Ara Katman Entegrasyon Test Otomasyon Aracı

Gökhan Öztaş

Gömülü ve Gerçek Zamanlı Yazılım Tasarım Müdürlüğü, SST Sektör Bşk.
ASELSAN A.Ş.

goztas@aselsan.com.tr

Özet. Yazılımlar arası protokol ve entegrasyon testi manüel yöntemler ile yapıldığında test amaçlı simülatörlerin geliştirilmesi çok fazla zaman almaktadır [7]. Diğer yandan standart analiz araçlarının kullanımı durumunda, bu araçlar üzerinde yazılımlar arası protokolün tanımlanması ve görüntülemenin bu şekilde sağlanması ihtiyacı oluşmaktadır. Her iki yöntem de esnek olmadığı için protokolde değişiklik yapılmasını zorlaştırmaktadır. ULAK, ASELSAN Gömülü ve Gerçek Zamanlı Yazılım Tasarım Müdürlüğü bünyesinde sensör ve silah sistemleri yazılımları geliştiren ekip tarafından tasarlanmış ağ üzerinde çalışabilen bir ara katmandır. ULAK-SPY, ULAK ara katmanı için hazırlanmış ve yukarıda bahsedilen zorlukları aşmayı hedefleyen bir entegrasyon test ve otomatik simülatör oluşturma aracıdır. ULAK-SPY ULAK ara katmanını kullanarak haberleşen iki sunucu ya da istemcinin herhangi birisinin yerine geçerek kendisine gelen ara katman mesajlarının görüntülenmesine ya da karşı taraftaki sunucu ya da istemciye mesajların gönderilmesini ve bu sayede entegrasyon testlerin yapılmasını mümkün kılmaktadır. Araç yukarıda bahsedilen ara katman mesajları test yeteneklerinin yanı sıra kullanıcıya senaryo oluşturma ve oynatma altyapısı ve iki yazılım arası haberleşmeyi kayıtlama altyapısı da sunmaktadır. ULAK-SPY aracı, yukarıda bahsedilen yeteneklerin ara katman protokolünün tanımlanması ile beraber otomatik olarak oluşturulmasını sağlayarak esnek bir altyapı sunmakta ve bu sayede gömülü yazılımların geliştirilmesi sürecinde büyük bir işçilik ve zaman tasarrufu sağlamaktadır.

Anahtar Kelimeler: Gömülü Sistemler, Entegrasyon Testi, Ara katman Protokol Testi, Otomatik Simülatör Oluşturma, Senaryo Oluşturma

Abstract. Making Integration tests and testing protocol between software is time consuming when performed by manual methods due to development of simulators for testing purposes. On the other hand with the use of standard analysis tools protocol between software must be implemented on these tools to be able to observe communication. Since both methods are not flexible making

modifications to the protocol gets harder. ULAK developed in ASELSAN Embedded and Real Time Software Design Department by a team that develops software for sensor and weapon systems is a middleware that works on network. ULAK- SPY is an integration test and automatic simulator generation tool which aims to overcome the difficulties mentioned above developed for ULAK middleware. ULAK- SPY enables testing of protocols by replacing server or client that communicates with each other via ULAK middleware. Apart from the middleware message test capabilities ULAK-SPY provides tools for creating and running scenarios and logging communication between software. ULAK-SPY generates the capabilities mentioned above automatically with the creation of middleware protocol which leads to a flexible infrastructure and saves great deal of time during the development of embedded software.

Keywords: Embedded Systems, Integration Test, Middleware Protocol Test, Automatic Simulator Generation, Scenario Generation

1 Giriş

Protokol farklı bilgisayarlar üzerinde koşan yazılımlar arası haberleşmeyi sağlamak amacıyla verileri düzenlemeye yarayan standart olarak kabul edilmiş kurallar dizisidir [2]. İki yazılım arasında iletişim için kullanılan dili ve haberleşme kurallarını belirtir ve iki yazılım arasında haberleşme sırasında kullanılacak mesaj tanımları protokol kurallarına uyularak tariflenir. Yazılımlar arası haberleşme için kullanılacak mesajlar yazılımcı tarafından manüel olarak kodlanabileceği gibi bu iş için oluşturulmuş hazır bir ara katman yazılımı da kullanılabilir. ULAK, özellikle az sayıda yazılımdan oluşan gerçek zamanlı gömülü sistemlerdeki haberleşmeyi sağlamak için oluşturulmuş bir ara katmandır [1]. Haberleşmeyi, gecikmeyi minimuma indirerek kısıtlı kaynak kullanımı ile gerçekleştirmek, çoklu görev yürütümünün olduğu yazılımlarda görev bölünmesine karşı önlem almak ULAK'ın öncelikleri ve geliştirmesini tetikleyen nedenlerdendir. ULAK Sensör ve Silah sistemleri için yazılım-yazılım ve yazılım-donanım arası haberleşme ihtiyaçlarını çözebilmektedir. ULAK ara katmanı bu makale kapsamında tartışılmayacaktır, Karasoy ve Çınar "Dağıtık Sistemler İçin Haberleşme Otomasyon Ara Katmanı: ULAK" isimli makalelerinde ULAK ara katmanı hakkında detaylı bilgiyi vermektedir. ULAK-SPY; ULAK ara katmanı ile haberleşen yazılım ya da donanımların entegrasyon test altyapılarının otomatik olarak oluşturulabilmesi için geliştirilmiş bir yazılımdır ve bu makalenin ana odağını ULAK ara katmanı değil ULAK-SPY aracı oluşturmaktadır. ULAK ara katmanı ve ULAK-SPY aracı bugüne kadar yaklaşık 28 adet sensör, silah ve görüntü işleme sistemi için geliştirilmiş gömülü yazılımların geliştirilmesinde kullanılmıştır. Bir sonraki bölümde ULAK-SPY aracına ilişkin genel bilgiler verilecektir.

2 ULAK-SPY Genel Bakış

Yazılım testlerinin yazılım kalitesini artırıcı yöndeki etkileri görüldükçe çok farklı test seviyesinde yazılım testleri gündeme gelmiştir. Test seviyesi test edilmekte olan yazılım bileşenlerinin büyüklüğünü ve testlerin uygulanış sırasını ifade etmektedir [2]. Yukarıdaki tanıma göre test seviyeleri 5 farklı gruba ayrılabilir:

1. Birim Testi
2. Entegrasyon Testi
3. Yazılım Yeterlilik Testi
4. Sistem Entegrasyon Testi
5. Kullanıcı Kabul Testi

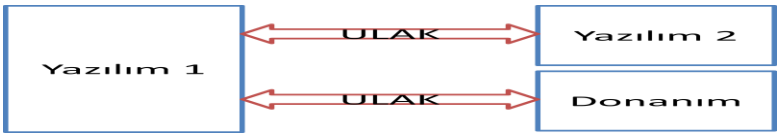
Yukarıda belirtilen test seviyelerinden entegrasyon testi, büyük sistemlerde birbirinden bağımsız geliştirilen bileşenlerin bütünleşik bir şekilde çalışabilir hale getirilmesinin ilk adımı olduğu için ciddi hataların tespit edilebileceği bir adımdır. IEEE Yazılım mühendisliği sözlüğüne göre entegrasyon testi, yazılım bileşenlerinin, donanım bileşenlerinin veya her ikisinin bir bütün halinde ele alınarak aralarındaki etkileşimin test edilmesidir [2]. Birbiriyle haberleşen ve veri alışverişinde bulunan pek çok yazılım bileşenin ya da donanımın bulunduğu büyük sistemlerde entegrasyon testleri büyük bir önem arz etmektedir. Yazılım geliştirme süreci boyunca yazılım hatalarının erken aşamalarda bulunabilmesi geliştirme ekibinin hataları daha erken giderebilmelerine, test ekiplerinin de hataları çözülen yazılım bileşenleri üzerinde tekrar test (regression test) yapabilmesine imkân sağlamakta ve yazılım kalitesini artırmaktadır [5]. Kullanıcı kabul testlerine yaklaşılan zamanlarda bulunan hata yoğunluğunun fazla olması müşteri memnuniyetini azaltmakta ve projenin başarısız olmasına sebep olmaktadır.

Yukarıda bahsedilen sebeplerden dolayı entegrasyon testlerine mümkün olduğunca erken başlayabilmek proje takviminin sonlarına doğru karşılaşılacak hata yoğunluğunun azaltılabilmesi için önemlidir. Birbirinden bağımsız ekip ya da kişiler tarafından geliştirilen yazılım bileşenleri ve donanımların her zaman beklenilen zamanda entegrasyon test takvimine uygun bir şekilde geliştirilmesi mümkün olamayabilmektedir[6]. Farklı bileşenlerin geliştirme takvimlerindeki gecikmeler entegrasyon testlerinin de gecikmesine sebep olmaktadır. Yazılım bileşenin etkileşimde bulunduğu diğer yazılım bileşenleri ya da donanımlardan bağımsız bir test ortamında koşturulabilmesi entegrasyon testleri sırasında görülebilecek kritik hataların çok daha önceden görülebilmesini sağlayacağı ve diğer yazılım ve donanım bileşenlerinin geliştirme takvimlerindeki gecikmelerin entegrasyon testlerinde gecikmeye sebep olması nedeni ile karşılaşılabilecek kritik hataların çok daha önceden çözülebilmesini sağlayacağı için önem taşımaktadır [3]. Bunu sağlayabilmek için yazılım bileşeninin beraber çalıştığı bütün yazılım bileşenleri ya da donanımların simüle edilmesi ihtiyacı ortaya çıkmaktadır. ULAK-SPY ulak ara katmanı kullanılarak tanımlanmış yazılım ya da donanım ara yüzleri için otomatik olarak simülatör oluşturma altyapısı sunarak yazılım entegrasyon testleri öncesi test

ortamının oluşturulabilmesini sağlar. ULAK-SPY yazılımı yeteneklerini yazılım entegrasyon testleri için faydası açısından 3 başlık altında toplayabiliriz.

2.1 ULAK-SPY Otomatik Simülatör Oluşturma Yeteneği

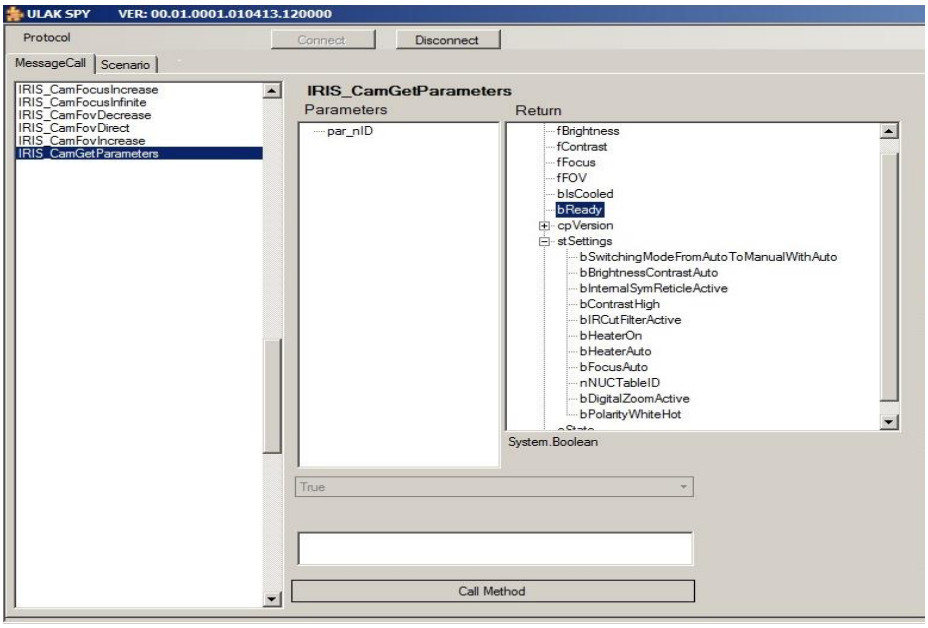
ULAK iki farklı yazılımın birbiri ile ya da yazılım ve donanımın birbiri ile haberleşmesini sağlamaktadır. Şekil 1'de görülen konfigürasyon ile haberleşen yazılım ve donanımlar için hataların projenin erken aşamalarında tespit edilebilmesi için yazılım geliştirme süreci boyunca entegrasyon ihtiyacı bulunmaktadır.



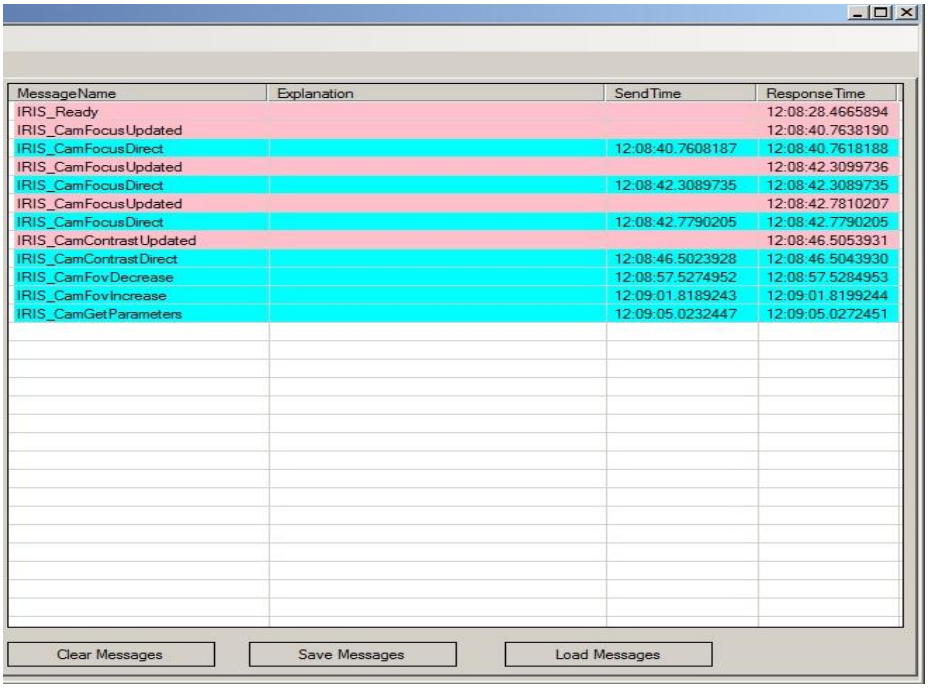
Şekil 1. Yazılım-Yazılım ve Yazılım-Donanım ULAK kullanımı

Entegrasyon ihtiyacını karşılamak için geliştirilen yazılım bileşeni ile haberleşen diğer yazılım ya da donanımların ara yüzlerini kontrollü bir şekilde gerçekleyen simülatör yazılımlar geliştirilmesi gerekmektedir. Geliştirilen bu simülatör yazılım bileşenleri ilgili yazılım ya da donanımın bütün yeteneklerini gerçekleştirmez, sadece test için gerekli olan ara yüzleri gerçekleştirir [7]. Bu simülatör yazılımlar her ara yüz için manüel olarak geliştirildikleri için test altyapısının oluşturulması çok fazla işgücü alabilmektedir. Geliştirilen yazılımın ULAK ara yüzlerinde proje yaşam döngüsü boyunca değişiklik yapılması ihtiyacı doğmakta ve buda ilgili ara yüze ait simülatör yazılımda değişiklikler yapılmasına sebep olmaktadır. Test altyapısının güncel tutulması çok fazla işgücü ve zaman kaybına yol açabilmektedir.

ULAK-SPY yazılımı, ULAK'ın hazırladığı ara yüze ilişkin detayların yer aldığı DLL dosyasını girdi olarak alır. ULAK ara katmanı C++ ve C# dillerini desteklemektedir. ULAK-SPY yazılımı C# için üretilen DLL dosyasını kullanmakta ve tüm ara yüz yeteneklerini çalışma zamanında otomatik çözümlemektedir.ULAK-SPY ".NET Framework Reflection" teknolojisinden faydalanarak ilgili ara yüze ilişkin sınıfların yer aldığı DLL dosyasını çözümleyerek haberleşme ara yüzü ile ilgili kullanıcı ara yüzünü dinamik olarak oluşturur.



Şekil 2. ULAK-SPY Otomatik Simülatör Mesaj Seçim Arayüzü

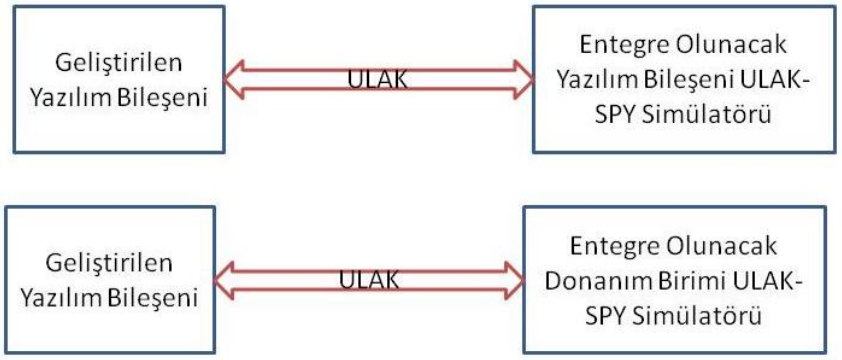


Şekil 3. ULAK-SPY Otomatik Simülatör Mesaj Listesi Arayüzü

Şekil 2 ve şekil 3'de ULAK-SPY tarafından otomatik olarak oluşturulan simülatör ara yüzü görülmektedir. Kullanıcı ilgili ULAK ara katman protokolü için sağlayıcı ya da istemci rolünü seçebilmektedir. Seçilen role göre ULAK ara katmanı ile haberleşen iki yazılım tarafının simülatörü de otomatik olarak oluşturulabilmektedir.

ULAK-SPY rolüne göre istemci olunan ara katman mesajları çalışma zamanında çözümlenerek bir liste halinde kullanıcıya sunulmaktadır. Kullanıcı bu listeden istediği bir mesajı seçtiğinde o mesajla ilgili doldurulması gereken parametreler dinamik olarak çözümlenerek kullanıcı ara yüzüne yansıtılmakta ve kullanıcı ilgili parametreler için değer girebilmektedir. Parametreleri için değer girilen mesaj karşı taraftaki yazılıma gönderilmekte ve ilgili mesaj kayıt listesine eklenmektedir. Yine aynı şekilde karşı taraftaki yazılım tarafından gönderilen ULAK ara katmanı mesajları çalışma zamanında çözümlenmekte ve mesaj listesine eklenmektedir. Gönderilen mesajlarda olduğu gibi alınan mesajlar içinde otomatik olarak kullanıcı ara yüzü oluşturulmakta ve kullanıcıya alınan mesaj içeriği sunulabilmektedir. ULAK-SPY mesaj listesindeki gönderilen ve alınan mesajlar analiz ve hata ayıklama amaçlı olarak dosyaya kaydedilebilmekte ve daha sonra dosyadan tekrar yüklenebilmektedir. Mesaj içeriği çalışma zamanında otomatik olarak çözümlenebildiği için proje sürecinde ortaya çıkan ULAK ara katman mesajları güncelleme ihtiyaçlarına çok hızlı bir şekilde cevap verilebilmektedir. Simülatörler otomatik olarak oluşturuldukları için entegrasyon test altyapısının idamesi için işçilik ve zaman boyutunda çok büyük bir kazanç sağlanabilmektedir.

ULAK-SPY rollerine göre Şekil 4 ve Şekil 5'te görülen test konfigürasyonları oluşturulabilmektedir. Şekil 4 teki konfigürasyonda geliştirilen yazılım olan Yazılım1 bileşenin arayüzü olduğu yazılım ve donanım bileşenleri için otomatik olarak simülatör oluşturulmuş ve entegrasyon testleri için otomatik olarak bir test altyapısı oluşturulmuştur. Şekil 5'te ise geliştirilen bir donanımın entegrasyon testinin yapılabilmesi için karşı taraftaki yazılımın simülatörü otomatik olarak oluşturulmuş ve ilgili donanımın entegrasyon testleri için bir test altyapısı oluşturulmuştur.



Şekil 4. "Yazılım-Yazılım ULAK-SPY simülatörü" ve "Yazılım-Donanım ULAK-SPY simülatörü" kullanımı konfigürasyonu

Şekil 5. "Yazılım ULAK-SPY Simülatörü-Donanım" kullanımı konfigürasyonu

2.2 ULAK-SPY Senaryo Yeteneği

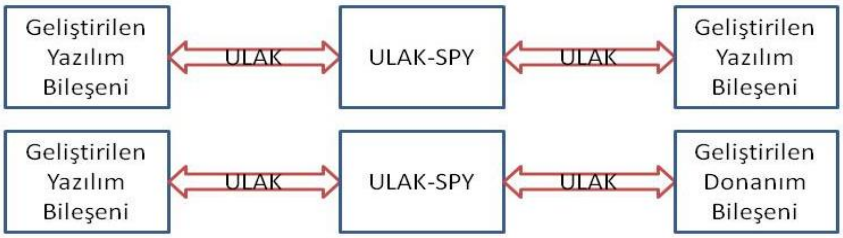
ULAK-SPY zaman veya mesajlar ile tetiklenen senaryolar tanımlanmasına ve senaryo tetiklenme koşulları sağlandığında içeriği belirlenebilen mesajların karşı taraftaki yazılıma otomatik olarak gönderilmesine imkân sağlamaktadır. Oluşturulan senaryolar kaydedilip daha sonra yüklenerek tekrar oynatılabilmektedir. Senaryo oluşturma yeteneğinin 2 ana kullanım alanı bulunmaktadır.

Periyodik Sorgu Yapabilme ve Cevap Verme İhtiyacı: Birbiri ile haberleşen iki yazılım ya da donanım arası haberleşme protokolünde tanımlanmış periyodik olarak gönderilmesi gereken mesajlar bulunabilmektedir. Belirli bir frekansta ya da karşı taraftan alınan bir mesaja cevap olarak sürekli olarak gönderilmesi gereken mesajların kullanıcı tarafından gönderilmesi çok zordur. Kullanıcı ULAK-SPY senaryo yeteneğini kullanarak periyodik mesajlar için gerekli senaryoları tanımlayıp koşturarak entegrasyon testi yaptığı yazılımı istediği duruma getirebilmekte ve diğer test senaryolarını koşturabilmektedir.

Entegrasyon Test Senaryolarının Gerçeklenmesi İhtiyacı: ULAK ara katmanını kullanarak haberleşen yazılım ya da donanımlar için entegrasyon testleri kapsamında tanımlanmış test tanımları senaryo yeteneği ile oluşturulabilmektedir. Oluşturulan bu test senaryoları koşturularak sonuçları kullanıcı tarafından analiz edilerek testlerin başarılı ya da başarısız olduğuna karar verilebilmektedir. Oluşturulan senaryolar tekrar koşturulabilir olduğu için yeni sürüm testlerinde (regression test) tekrar kullanılabilmektedir.

2.3 ULAK-SPY Haberleşme Kayıtlama Yeteneği

Yazılım geliştirme sürecinde geliştirilen yazılım ya da donanımlar entegrasyon testi olgunluğuna geldiğinde gerçek yazılım ya da donanımlar ile entegrasyon testi yapılabilmektedir. Entegrasyon testi sırasında hatalar ortaya çıkabilmekte ve hatanın hangi yazılım ya da donanım kaynaklı olduğunun anlaşılabilmesi için bileşenler arası haberleşmenin izlenilebilmesi ihtiyacı doğmaktadır. ULAK-SPY Şekil 6'da verilen konfigürasyonda çalışarak ULAK ara katmanı ile haberleşen yazılım ya da donanımların arasına girerek iki taraflı gönderilen mesajları çalışma zamanında çözümleyip kayıt altına alarak liste halinde kaydedebilmektedir. Bu yetenek sayesinde entegrasyon testi sırasında test senaryosundaki hata görülebilmekte ve hatanın hangi yazılım ya da donanımda olduğu tespit edilebilmektedir.



Şekil 6. ULAK-SPY Haberleşme Kayıtlama kullanımı konfigürasyonu

3 Sonuçlar

ULAK-SPY yazılımı Sensör ve Silah Sistemleri gömülü yazılımları geliştirme sürecinde ortaya çıkan entegrasyon test altyapısını otomatik olarak oluşturulmasını sağlayan bir otomatik simülatör oluşturma yazılımıdır. Simülatörlerin otomatik olarak ULAK-SPY tarafından oluşturulması sayesinde simülatör geliştirme işçiliklerin den büyük oranda tasarruf edilmiştir. Otomatik simülatör oluşturma yeteneği yazılım geliştirme, donanım geliştirme ve sistem test ekiplerinin ortak bir ihtiyacını çözerek manüel olarak simülatör yazılımlarının hazırlandığı çözüme göre büyük bir işçilik ve zaman tasarrufu sağladığı görülmüştür. Test altyapısının ULAK-SPY tarafından otomatik olarak oluşmasının sağladığı esneklik yazılım geliştirme sürecinin her aşamasında yazılım ve donanım ara yüzlerinin ihtiyaç duyulması halinde hızlı bir şekilde değiştirilebilmesine imkân vermiştir. Entegrasyon faaliyetlerine diğer yazılım ya da donanımların geliştirilmesinden bağımsız bir şekilde başlanabilmesini sağladığı için ULAK-SPY kritik hataların yazılım geliştirme sürecinin erken aşamalarında tespit edilmesini sağlamıştır.

Kaynaklar

1. Karasoy, B., Çınar, S., "Dağıtık Sistemler İçin Haberleşme Otomasyon Ara Katmanı: ULAK", Proceedings of the 8th Turkish Natioanal Software Engineering Symposium UYMS2014, pp. 372-382, Güzelyurt, KKTC 2014
2. "IEEE Standard Glossary of Software Engineeering Terminology", IEEE Standards Board, 1990
3. Bennett, T.L., Wennberg, P.W., "Eliminating Embedded Software Defects Prior to Integration Test",The Journal of Defense Software Engineering, 2005.
4. Hetzel, W., "The Complete Guide to Software Testing", QED Information Sciences, 1984.
5. Sivera, P., Pasquato, M., "Software Integration and Test Techniques in a Large Distributed Project:Evolution, Process, Improvement, Results", Proceedings of ICALEPCS07, pp. 508-510, Knoxville, USA, 2007
6. Zhenyi, J., Offutt, A.J. "Coupling-based Criteria for Integration Testing", The Journal of Software Testing,Verification, and Reliability, pp. 133-154, 1998.
7. Copeland, L., "A Practitioner's Guide to Software Test Design", Artech House Publishers, 2007.