Veri Dağıtım Servisi Tabanlı Dağıtık Sistem Test Yaklaşımı

Saniye Asena ARTAÇ[1], İlker KOYUNCU[2], Aylin HATIP IPEK[3]

[1] ROKETSAN A.Ş., saniyetokel@roketsan.com.tr

[2] ROKETSAN A.Ş., ikoyuncu@roketsan.com.tr

[3] ROKETSAN A.Ş., ahatip@roketsan.com.tr

Özet. Hızlı ve etkin bir biçimde gelişim gösteren veri iletişimi, geliştirilen sistemlerin özgün ve çevik tasarım ihtiyaçlarını ön plana çıkarmaktadır. İlgili sistemin kapsamı ve kabiliyeti arttıkça, ortamda bulunan çok sayıda donanım ve yazılım bileşeninin hızlı, doğru ve gerçek zamanlı veri paylaşımı gibi ihtiyaçları doğmaktadır. Veri Dağıtım Servisi (DDS) teknolojisi, dağıtık sistemlerin geliştirilmesinde kullanılan ve son dönemde pek çok sektörde yaygınlaşan bir ara katman standardıdır. Bu makale kapsamında, DDS tabanlı bir sistem için uygulanan test yaklaşımı incelenmiştir. "Veri Dağıtım Servisi Tabanlı Dağıtık Sistem Test Yaklaşımı" olarak adlandırdığımız bu çalışma DDS'in sağladığı dinamik ortam tanıma yeteneği ile esnek ve genişlemeye müsait bir test alt yapısı sunmaktadır. Ayrıca ilgili sistemin yaşam döngüsü aşamaları model tabanlı oluşturulmuş olup, buna paralel yazılım test analiz ve test tasarım aşamaları da bu kapsamda model tabanlı olarak hazırlanmıştır. Bu çalışma, özellikle dağıtık ve gerçek zamanlı, birden fazla alt sistemin güvenilir bir biçimde veri iletişiminin sağlanması ve entegrasyon eforlarını aza indirmesi nedeniyle DDS altyapıları üzerine bir hibrid analiz sunmaktadır.

Anahtar Kelimeler: Yazılım Test, Test Alt Yapısı, Simülatör, Dağıtık Mimari, Veri Dağıtım Servisi.

Data Distribution Service Based Distributed System Test Approach

Saniye Asena ARTAÇ[1], İlker KOYUNCU[2], Aylin HATIP IPEK[3]

[1] ROKETSAN A.Ş., saniyetokel@roketsan.com.tr

[2] ROKETSAN A.Ş., ikoyuncu@roketsan.com.tr

[3] ROKETSAN A.Ş., ahatip@roketsan.com.tr

Abstract. The needs for fast and efficient data communication has been progressively bringing original and agile design needs of the developed systems today. As the scope and capability of the system increases, the need for rapid, accurate and real-time data sharing of a large number of hardware and software components in the environment arises. Data Distribution Service (DDS) technology is a middleware standard that has been used in the development of distributed systems and has become popular in many sectors in recent years. In this paper, the test approach applied to a DDS based system is discussed. This study, which we call "Data Distribution Based Distributed System Test Aprroach", presents a flexible and expandable test infrastructure with the ability of dynamic environment recognition provided by DDS. In addition, the life cycle stages of the related system are designed as model based, and software test analysis and test design phases are also prepared in model based. In this study, it will presents a hybrid analysis that reliably delivering data communication by reducing efforts especially in distributed and real-time systems.

Keywords: Software Test, Test Engine, Simulator, Distributed Architecture, Data Distribution Servisi, DDS.

1 Giriş

Çeşitli platformlar üzerinde kullanmak için geliştirilen komuta kontrol yazılımları birçok farklı aviyonik sistemler, mühimmatlar ve sensörler gibi birimlerle iletişim halinde bulunmaktadır. Birimler arası iletişimin gerçek zamanlı, hızlı ve güvenilir şekilde çalışmasına imkan sağlayacak şekilde kurulmuş olması gerekmektedir. Bu koşulların sağlanıp sağlanmadığının anlaşılması için her yazılım ve sistem için uygulanması gereken farklı seviye test aktivitileri bulunmaktadır. Oldukça karmaşık bir yapıya sahip olan bu sistemlerin test ve entegrasyon süreci uzun zaman alabilmektedir.

Sistemlerin sahip olduğu aviyonik birimler teknik ve fiziki açıdan değerlendirildiğinde, donanım bileşenlerinin kendisinin kullanımı her zaman uygun olmamaktadır. Bu durumlarda ilgili birime ait simülatör kullanımı tercih edilmekte ve bazı veriler sanal olarak oluşturulmaktadır. Ayrıca her alt sistemin farklı arayüz kullanması yerine ortak bir arayüzden konuşması tercih edilebilir bir özelliktir. Aralarında veri paylaşımı olan dağıtık mimariye sahip sistem ihtiyaçlarını karşılamak için, Object Management Grup (OMG) tarafından yayınlanan Veri Dağıtım

Servisi'nin (Data Distribution Service-DDS) gerçek zamanlı dağıtık sistemlerin geliştirilmesinde, ana haberleşme standardı olarak kullanımı yaygınlaşmaktadır [1, 2]. Dağıtık mimaride oluşturulmuş farklı fiziksel iletişim arayüzlerine sahip birimler ile komuta kontrol yazılımı arasındaki iletişimi modüler bir şekilde desteklemek amacıyla DDS ara katmanı kullanımı tercih edilmektedir. Aynı zamanda birçok sistemin aynı anda çalıştığı bir ortamda, veri iletiminin sistem performansını olumsuz yönde etkilememesi bir diğer önemli husustur. DDS haberleşme protokolü ile bahsedilen ihtiyaçlar için hem yazılım hem test alt yapısı olarak esnek ve uyarlanabilen mimariler oluşturulabilmektedir.

Bu çalışmada, sanal ve gerçek sistemlerin entegre bir şekilde kullanılabildiği, DDS altyapısı sayesinde farklı konumlarda bulunan alt sistemlerin kullanılarak testlerin gerçekleştirilebildiği, sistemin kullanılırken karşılaşabileceği durumların denenebildiği bir test ortamı ve yazılım test analiz, tasarım aşamalarının model tabanlı olarak hazırlanması çalışmaları özetlenmiştir.

2 Veri Dağıtım Servisi

Veri Dağıtım Servisi (DDS) temel olarak yayımla-abone ol ve veri merkezli mimari altyapısına sahip bir ara katman protokolüdür. Bu protokol, dağıtık ortamda çalışan gerçek zamanlı sistemlerin veri iletim ihtiyaçlarını karşılamak üzere geliştirilmiştir [3]. Yayımla-abone ol mimarisi ortamda bulunan sistemlerden yayıncı ve abone katılımcıların birbirini tanımasına, fiziksel adreslerini bilmelerine gerek yoktur. Bu yapı sayesinde sistemde bir değişiklik yapılmadan yeni birimlerin eklenmesi, ortamda bulunan verilere abone olunarak istenilen bilgiye erişilebilmesi gibi esneklik sunmaktadır. DDS servis kaliteleri(QOS) ile verinin nereden nereye hangi servis kalite parametreleri ile iletileceği önceden belirlenebilmektedir. DDS kullanımının sisteme sağladığı avantajlar;

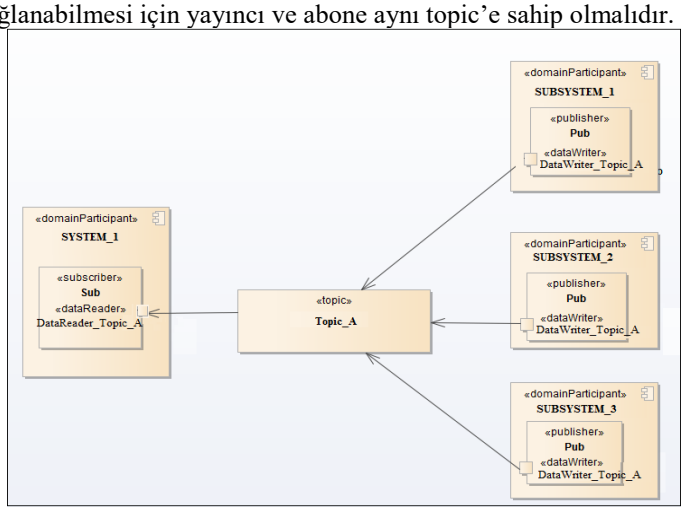
- Esnek ve uyarlanabilen haberleşme mimarisine sahip olması (bu özellik sayesinde mevcut alt sistemlerin alternatiflerinin entegrasyonunda, yazılım geliştirme ve test faaliyetlerinde ihtiyaç duyulacak zaman ve iş gücünün azalacağı değerlendirilmektedir.)
- Güvenilir veri paylaşımı
- Veri iletim bant genişliğinin verimli şekilde kullanılması
- Çoklu katılımcılı haberleşmeleri desteklediği gibi, teke tek haberleşmeye de imkân sağlaması

olarak sıralanabilir.

2.1. DDS-Bileşenleri

Domain: Birbiri ile iletişim kuracak uygulamalar için genel veri alanını temsil etmektedir. Her "domain" bir tamsayı ID ile benzersiz şekilde tanımlanmaktadır. İki uygulamanın birbiriyle ile iletişim kurması için aynı "domain" de bulunması gerekmektedir.

Domain Participant: Uygulamaların bir domain'e katılmak için kullandığı yapıdır. Veri yayımlama ve abone olma işlemleri, veriler için varsayılan servis kalitesi parametrelerinin ayarlanması bu bileşen üzerinden sağlanmaktadır.
Publisher: Verilerin yayımlanması için kullanılır. Bu nesne bütün yayınlar için ortak bir ara yüz oluşturmaktadır.
Data Writer: DDS veri bölgesindeki uygulamaların veri yayınlamasını sağlayan temel erişim noktasıdır.
Subscriber: Verilere abone olmak için kullanılır. Bu nesne bütün abonelikler için ortak bir ara yüz oluşturmaktadır.
Data Reader: DDS veri bölgesindeki uygulamaların veriyi almasını sağlayan temel erişim noktasıdır.
Topic(Başlık): Yayıncı ile abone arasındaki iletişim noktasıdır. DDS sisteminde iletişimin sağlanabilmesi için yayıncı ve abone aynı topic'e sahip olmalıdır.

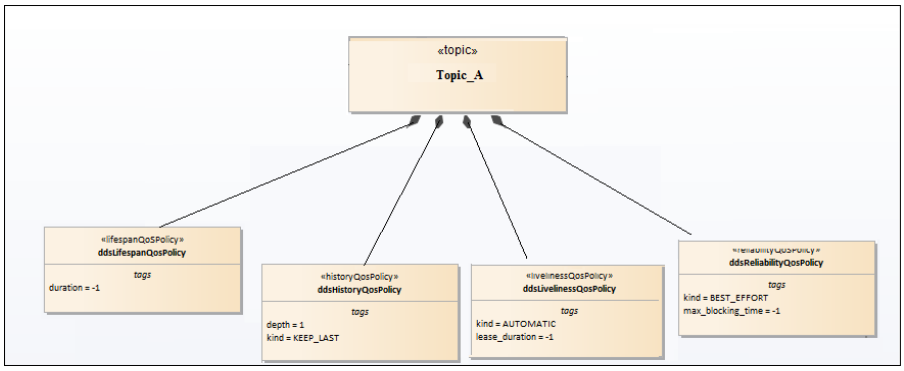


Şekil 1. DDS Bileşenleri

2.2. DDS-Servis Kalitesi(QOS) Özellikleri

DDS arakatmanın sahip olduğu servis kalite desteği, sistemlerin hataya dayanıklılık ve gerçek zaman ihtiyaçlarının karşılanmasına yardımcı olmaktadır. Aşağıda en önemli ve sık kullanılan servis kalite özellikleri listelenmiştir [4];
Güvenilirlilik (Reliability): Verinin abonelere dağıtımının ne seviyede garanti edildiğini belirleyen servis kalitesidir.
Kalıcılık (Durability): Verinin geç katılan alt sistemlere dağıtım biçimini belirleyen servis kalitesidir.
Geçmiş Bilgisi (History): Gönderilen verinin en son kaç örneğinin alt sistemler için erişilebilir olacağını tanımlayan servis kalitesidir.
Yaşam Ömrü (Life Span): Veriler için geçerlilik süresini belirleyen servis kalitesidir, geçerlilik süresi dolan verilere erişim sağlanamaz.

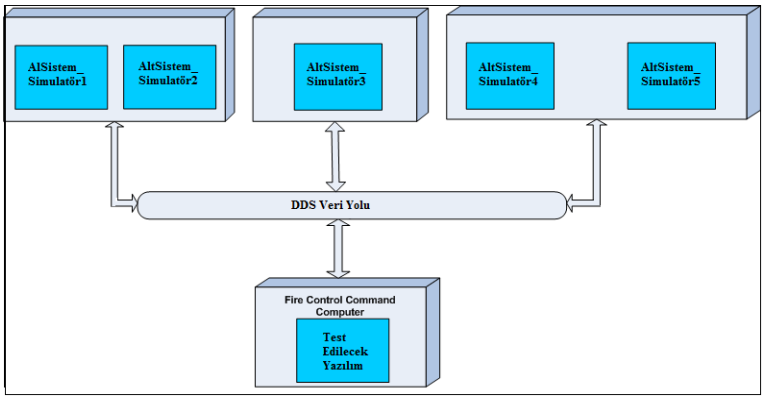
Sahiplik (Ownership): Bir alt sistemin aynı anda birden fazla farklı alt sistemden veri örneği alıp alamayacağını tanımlayan servis kalitesidir.



Şekil 2. Bir Başlığa Ait Örnek DDS Servis Kalitesi Özellikleri

3 Veri Dağıtım Servisi Tabanlı Dağıtık Sistem Test Mimarisi

Örnek sistemde yazılım doğrulama ve geçerleme çalışmaları kapsamında kullanılan test mimarisi **(b)**
Şekil 4'de verilmiştir. Yazılım test ortamı tanımlanan sistem mimarisi temel alınarak oluşturulmuştur.



Şekil 3. Test Mimarisi

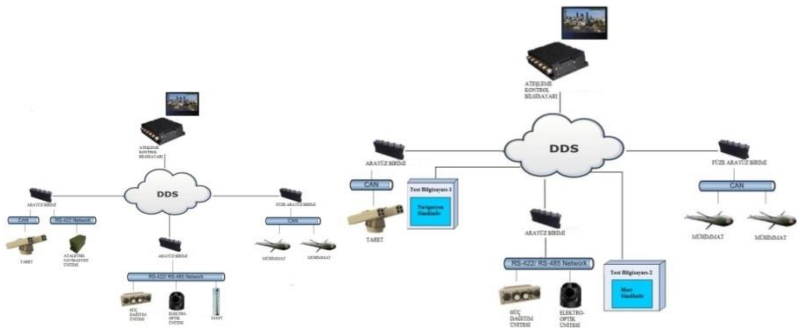
Test edilecek yazılım ve ilgili simülatörler aynı DDS domaine abone olarak veri alışverişini gerçekleştirmektedir. Bu yapı sayesinde yazılımların tek bir bilgisayar üzerinde bulunması yerine dağıtık ortamda farklı bilgisayarlar üzerinden aynı DDS domaini kullanarak haberleşme sağlanabilmektedir. DDS domain e katılacak her birim çalıştırıldığı zaman alacağı ve göndereceği başlıklara abone olur.

Verilerin doğru noktalara iletilmesi, zaman kontrolü, dağıtım biçimi, veriye erişim gibi DDS altyapısının sahip olduğu servis özellikleri kullanılarak başlıklara özel servis özellikleri belirlenmiştir.

Sisteme yeni bir yazılım kalemi eklenmesi ihtiyacı ortaya çıktığında yazılım kalemine ait altsistem/birim simülatörü sistemdeki diğer mevcut simülatörlerde değişiklik yapılmadan entegre edilebilmektedir.

Test edilecek yazılım/donanım kalemi, ortamdaki altsistemlerin istendiğinde gerçek, istendiğinde simülatör/emülatör şeklinde ayarlanabildiği bir ortamda kolaylıkla test edilebilmektedir. Örneğin navigasyon ünitesinin bulunmadığı bir ortamda yada farklı bir konumdaymış gibi test denemeleri yapılmak istendiği durumda ilgili birimin simülatör yazılımı sisteme entegre edilerek çalışmalar devam ettirilebilmektedir (**(b)**

Şekil 4). Bu yüzden kurulan yapıda her alt sistem için bir test simülatörü geliştirilmiştir ve DDS'in sağladığı dinamik ortam tanıma yeteneği ile, esnek, genişletilebilir bir test ve simülatör altyapısı sağlanmıştır.



(a) (b)

Şekil 4. (a) Altsistemler (b) Altsistemler ve Simülatörler

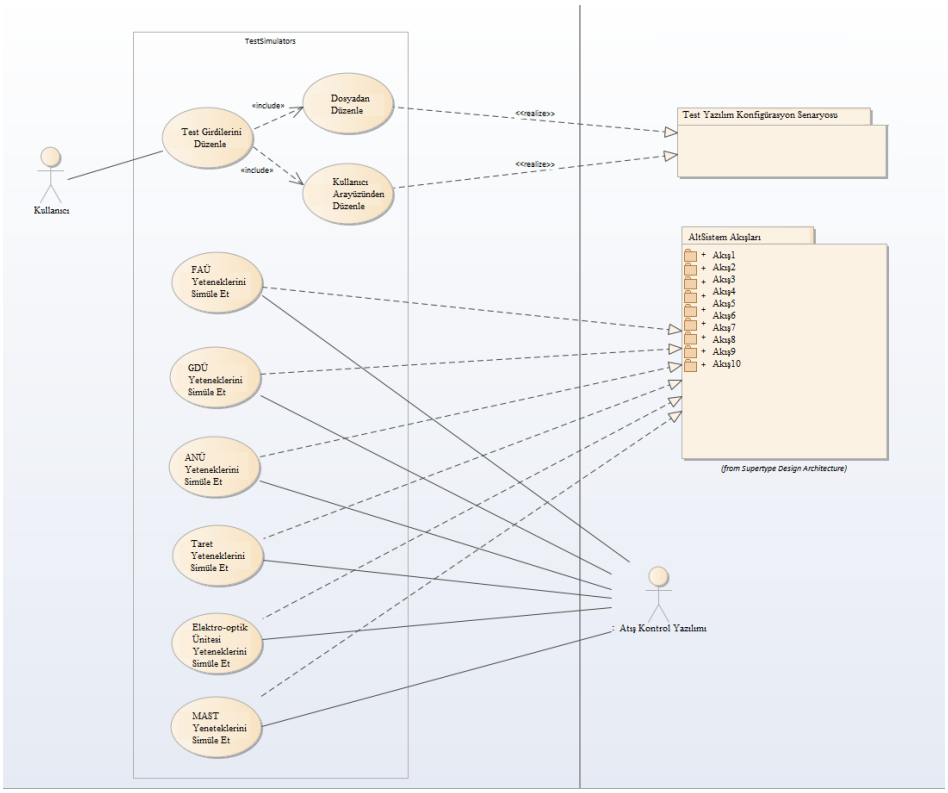
Yazılım test faaliyetlerinde öncelikle geliştirilen yazılımın tanımlanan yazılım gereksinimlerini karşıladığının doğrulanması, beklenen çıktılar ile elde edilen çıktılar arasındaki farkların tespit edilmesi amaçlanmıştır. Temel olarak gerçekleştirilecek koşullar önceden belirlenmiş test girdi formatına göre oluşturularak simülatörlere yüklenmektedir. DDS kütüphanesi sayesinde yayınlanacak verilerin ortama gönderilmesi, dinlenecek verilerin kaynaktan gelip gelmediğinin kontrolü gerçekleştirilerek testler icra edilmektedir. Aynı zamanda test yazılımlarının kullanıcı ara yüzlerinden başlık içeriklerinin değiştirilmesi ve alınan verilerin kontrolü sağlanabilmektedir. Test girdisi olarak, geliştirilen yazılımın ilgili görevi gerçekleştirmesine yönelik beklenen pozitif veriler ile alt sistemlerde meydana gelebilecek bir sorun sebebi ile sistemin gösterebileceği olumsuz davranışlar belirlenmiş ve kullanılmıştır.

4 Model Tabanlı Yürütülen Test Faaliyetleri

IEEE Std 829-1998 (IEEE Standard for Software Test Documentation) Standardı Test aktivitelerini, gereksinim tanımlanmasından test icrasına kadar geçen süreçte testin tasarlanması, test durumları oluşturulması, test prosedürlerinin hazırlanması şeklinde tanımlamaktadır [5]. Şu an halen oldukça yaygın olarak doküman tabanlı geliştirme aktiviteleri kullanılıyor olsa da trendlerin model tabanlı geliştirme alanına doğru hızla kaydığı gözlenmektedir.

Model, gerçek dünyadaki bir sistemin yapı ve davranışının soyut ve biçimsel olarak gösterimidir. Bu yöntem ile karmaşık bir sistem ve yazılımın esnek bir şekilde gösterilmesi ve daha anlaşılır olması amaçlanmaktadır [6]. "Veri Dağıtım Servisi Tabanlı Dağıtık Sistem Test Altyapısı"nda yukarda bahsedilen standart aktiviteler çeşitli araçlar yardımıyla modellenerek oluşturulmuştur.

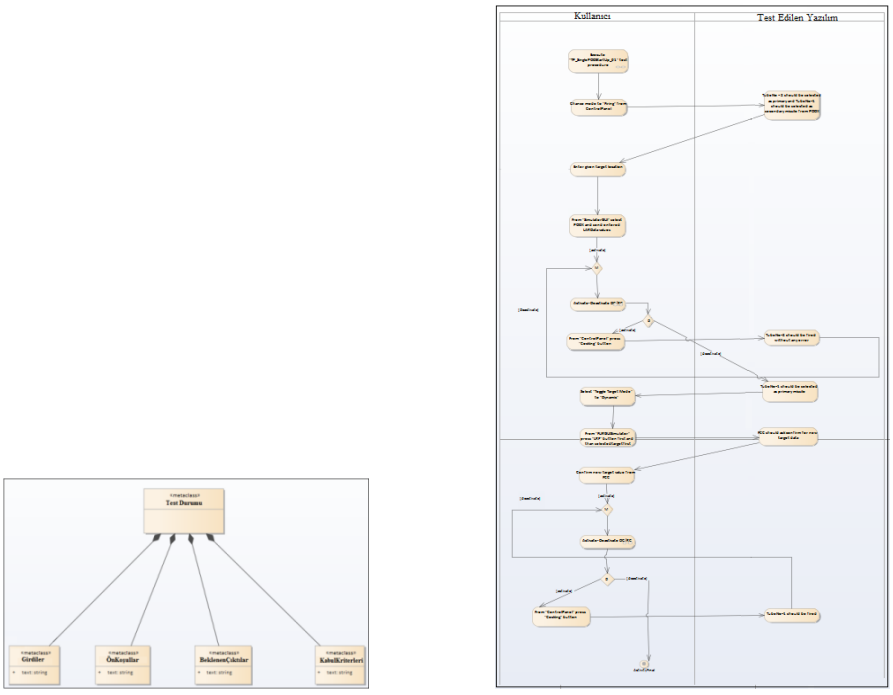
Bu yaklaşımda test yazılımlarına ait gereksinimler "Kullanım Durumu Diyagramı" ile tanımlanmıştır. **Şekil 5**'de verilen Kullanım Durumu Diyagramı ile simülatörlerden gerçekleştirmesi beklenen durumlar belirtilmiş ve alt sistem akışları kullanılarak gereksinimler belirlenmiştir.



Şekil 5. Kullanım Durumu Diyagramı

Tanımlanan test meta modeli yardımıyla test durumundan beklenen kabul kriterleri, beklenen sonuç, ön koşul ve girdi gibi değerler standartlaştırılabilmekte ve tüm modellere şablon oluşturulabilmektedir. Test meta modeli şablonunda yapılacak bir değişiklik aynı anda bu modelin kullanıldığı tüm diyagramlarda geçerli hale gelecek ve yayınlanacaktır. Her bir kullanım durumu için bu meta model şablonları yardımıyla hangi başlığın hangi değerleri alacağı ve hangi koşullarda test edileceği test durumlarına yansıtılmaktadır.

Aynı şekilde test prosedürleri oluşturulurken kullanılan akışlarda da içerisinde birden fazla test durumu barındıran aktivite diyagramları yer almaktadır. Test durumu değiştirildiğinde bu test durumunu kullanan tüm prosedürlerde ilgili kısımlar otomatik olarak güncellenebilmektedir [7 , 8].



(a) (b)

Şekil 6. (a) Test Durumu (b) Test Prosedür Örnek

Oluşturulan altyapı ile sisteminde tanımlı her bir "Yapı" öğesi başlıkların hangi alanları içereceğini, her bir başlığın hangi test durumunda kullanılacağını, her bir test durumunun hangi kullanım durumunu karşılayacağını ve dolayısıyla ilgili gereksinimleri etkilediğinden izlenebilirliğin tüm depo kaynak üzerinden sağlanmasını mümkün kılmaktadır.

Klasik yöntemlerde uygulanan gereksinim bazlı testlere ek olarak, tanımlanan test prosedürleri senaryo bazında test tanımlarına sahiptir.

5 Sonuç

Bu makalede, veri dağıtım servisi tabanlı dağıtık bir sistemde uygulanan test alt yapısından bahsedilmiştir. Uygulanan yaklaşımlar toplu olarak değerlendirildiğinde;

- Entegrasyon açısından esnek ve genişlemeye müsait bir altyapı elde edilmesine,
- Yeni eklenecek veya değiştirilecek alt sistem/birimlerin mevcut sistem üzerindeki etkileri ile entegrasyon için harcanacak zaman ve efor en aza indirgenmesine,
- Geliştirilecek yazılımlar, henüz etkileşimde bulunacakları alt sistem unsurları ortada yokken rahatlıkla geliştirilip arayüzsel anlamda uyumluluklarının test edilebilmesine,
- "Yayımla-abone ol" mekanizması ile çalıştığı için karmaşık ağ yapılarına ihtiyaç duyulmadan test icrası yapılabilmesine,
- Aynı dili konuşan alt sistemlerin haberleşme ve veri iletimi sorunlarının azalmasına,
- Test mimarisinin, yazılım mimarisi ile olan bağını dokümanlar üzerindeki soyut izlenebilirlikten dinamik izlenebilir hale gelmesine,
- Model, sınıf, akış ve kullanım durumu ilişkilerinin test öğelerine de sirayet etmesine,
- Test durumu ve test prosedürlerinde kullanılacak öğelerin dinamik tanımlanması ile kolay güncellenebilir, güvenilir, yönetilebilir bir şekilde tasarlanmasına,

olanak sağlamıştır.

Kullanılan araçların kabiliyetlerindeki artış doğrudan tasarım sürelerini düşüreceği gibi yazılım mühendisliği yaklaşımlarında da yeni ihtiyaçların doğmasına öncülük edecektir.

Referanslar

1. OMG, (2006): "Data Distribution Service for Real-Time Systems Ver 1.2", http://www.omg.org.
2. P. Griffioen, TACTICOS CMS: exploitting the full DDS potential, DDS Information Day, 2006 , Washington, USA.
3. H. Kutluca, İ.E. Çetin, B. Bal (2007), "MilSOFT DDS Arakatmanı ve DDS'in Savaş Yönetim Sistemlerinde Simülasyon Amaçlı Kullanımı" USMOS 2007, Ankara, Türkiye
4. Valls M. G., Val P. B., Analyzing Point to Point DDS Communication
Over Desktop Virtualization Software, Computer Sciens and Interfaces, 2017
5. IEEE Std 829-1998, IEEE Standard for Software Test Documentation
6. J. Holt, S.Perry, "SysML for System Engineering: A Model-Based Approach", The Institution of Engineering and Technology.
7. B.M. Selvy, C. Claver, G. Angeli (2014), Using SysML for Verification and Validation Planning on the Large Synoptic Survey Telescope (LSST), LSST Project Offie, Tuscon, USA

8. R. Hauber (2007), From Use Cases to Test Cases, HOOD Group, Oberhaching, Germany