Dokuz Eylül Üniversitesi-Mühendislik Fakültesi
Fen ve Mühendislik Dergisi
Cilt 19 Sayı 55.1 (Özel Sayı) Ocak 2017

Dokuz Eylul University-Faculty of Engineering
Journal of Science and Engineering
Volume 19 Issue 55.1. (Special Issue) January 2017

DOI: 10.21205/deufmd.2017195528

Veri Dağıtım Servisi Tabanlı Sistemlerde Enine Kesen İlgiler için İlgiye Yönelik Programlama Yaklaşımı

Ömer KÖKSAL[1], Mirun AKYÜZ[2]

[1] Wageningen Üniversitesi, Enformasyon Teknolojileri Bölümü, Wageningen, Hollanda
[2] TOBB Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Ankara

(Alınış / Received: 14.06.2016, Kabul / Accepted: 12.08.2016,
Online Yayınlanma / Published Online: 09.01.2017)

Anahtar Kelimeler
İlgiye Yönelik Programlama, Dağıtık Sistemler, Veri Dağıtım Servisi

Özet: Dağıtık sistemlerin geliştirilmesinde, veri dağıtım servisi arakatmanı yaygın olarak kullanılmaktadır. Pek çok sektörde kullanımı hızla artan DDS arakatmanı, veri merkezli yayınla-abone ol tabanlı haberleşme mekanizması ve ön tanımlı servis kalitesi parametreleriyle ölçeklenebilir ve güçlü sistemler geliştirmek için idealdir. Ancak, veri dağıtım sistemindeki ilgilerin, tek bir modül içinde toplanamayacak şekilde dağıtık olması yeniden kullanılabilirlik, program bakımı ve yönetimi ve benzeri yazılım kalite faktörleri için problem oluşturmaktadır. Bu makalede, DDS tabanlı sistemlerdeki enine kesen ilgi problemlerine, ilgiye yönelik programlama yaklaşımı ile çözüm bulunması konusu incelenmiştir. Örnek çalışma kapsamında geliştirdiğimiz mini komuta kontrol sistemi üzerinde kodladığımız ilgilerin DDS tabanlı sistemlerin geliştirilmesinde ne gibi faydalar sağlayacağı, geliştirme ve test sürelerini nasıl kısaltacağı tartışılmıştır.

Aspect Oriented Approach for Handling Cross-Cutting Concerns in Data Distribution Service Based Systems

Keywords
Aspect-Oriented Software Development, Distributed Systems, Data Distribution Service

Abstract: Data Distribution Service (DDS) is being widely used in developing distributed systems. The use of DDS middleware is being spread-out in many domains. With its data-centric publish-subscribe communication scheme and predefined Quality-of-Service parameters, DDS is ideal to develop scalable and robust distributed systems. On the other hand, cross-cutting concerns in DDS are hard to localize within a single software module causing maintenance, management and reusability problems. In this paper, we have discussed handling cross-cutting concerns in DDS based systems with Aspect-Oriented Programming (AOP) approach. As the case study, having implemented a mini command and control system and developing sample aspects on it, we have discussed the benefits of using AOP approach and how it shortens the development and test phases in the development of DDS based systems.

*Sorumlu yazar: omerkoksal@hotmail.com

1. Giriş

Günümüzde, dağıtık sistemlerin geliştirilmesinde, ana haberleşme çatısı olarak Veri Dağıtım Sistemi (Data Distribution Service - DDS) [13] kullanımı gittikçe yaygınlaşmaktadır. DDS, Object Management Grup (OMG) [12] tarafından yayınlanmakta olan bir standarttır. İlk sürüm 1.0, 2004 yılında yayınlanmıştır. Şu anki en son sürüm olan 1.4 ise 2015 yılında yayınlanmıştır.

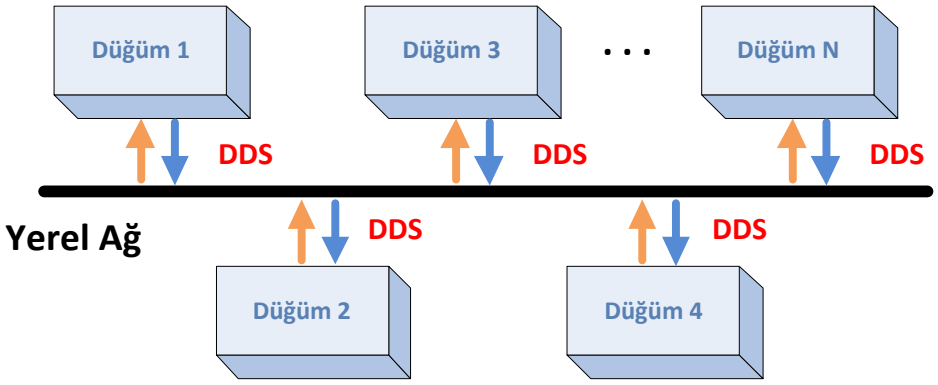
DDS, daha önce OMG tarafından tanımlanmış CORBA [22] arakatmanının aksine sunucu-istemci (client-server) mimarisi kullanmaz. Merkezi bir sunucu yerine, ağ üzerinde dağıtık bir sistem haberleşmesini temel alır. "DDS - dağıtık sistem mimarisi" Şekil-1'de verilmiştir. Bu mimari ile ağ üzerindeki düğümlerden (node) biri ya da birkaçı çalışmıyor/hasarlı olsa bile, diğer düğümler üzerinde haberleşme gerçekleşebilir.

DDS, üzerinde veri iletişiminde yayınla-abone ol yazılım örüntüsü kullanan bir arakatmandır. Yayınla-abone ol örüntüsü başka arakatmanlar tarafından kullanılsa da (ör. HLA), DDS, servis kalitesi (Quality of Service - QoS) parametreleri sağlayan tek arakatman olması ile öne çıkmaktadır. QoS parametreleri ile verinin sadece nereden nereye gideceği değil, hangi servis kaliteleri ile gönderileceği ön-tanımlı olarak belirlenebilmektedir. DDS, düğümlerinde, pek çok yayıncı (publisher) ve abone (subscriber) birbirleri ile DDS konuları (topic) üzerinden haberleşirler. DDS mimarisinin sağladığı avantajlardan birisi de yayıncıların ve abonelerin birbirlerini tanımasına, fiziksel adreslerini bilmelerine gerek olmamasıdır. Hatta haberleşmenin gerçekleşmesi için, aynı anda hayatta olmalarına bile gerek yoktur.
Dağıtık yazılımlar, DDS arakatmanı ile geliştirilerek daha modüler, daha dayanıklı (robust) ve ölçeklenebilir bir şekilde geliştirilebilirler. Ancak, dağıtık yazılımlar, yapıları itibariyle pek çok "enine kesen ilgi" (cross-cutting concern) bulundururlar. Yüzlerce düğüm üzerinde çalışabilen dağıtık yazılımlar için enine kesen ilgiler milyonlarca satır kod üzerine dağılmış olabilir. Bu durum, yazılımların bakımı, yönetimi ve geliştirilmesi açısından önemli bir problemdir.

Bu çalışmada, DDS tabanlı olarak geliştirilen dağıtık yazılımlardaki enine kesen ilgilerin oluşturduğu problemlerin "İlgiye Yönelik Programlama" (Aspect Oriented Programming) ile nasıl çözülebileceği konusu değerlendirilmiştir. Yazılım ve test mühendislerinin işlerini kolaylaştıracak, daha dayanıklı yazılım geliştirmeye yardımcı olacak ilgilere (aspect) odaklanılmıştır.

Örnek çalışma kapsamında, bir mini komuta kontrol yazılımı nesneye yönelik programlama ile geliştirildi. Daha sonra enine kesen ilgilerin yazılım üzerinde ne kadar dağıtık bir şekilde yer aldığı ve bunları yönetebilmek için ilgiye yönelik programlamanın nasıl kullanılabileceği gösterildi. Sadece nesneye yönelik programlama kullanıldığında, çok fazla kod değişikliği ile eklenebilecek bazı özelliklerin, ilgiye yönelik programlama ile minimum kodlama ile kolay bir şekilde yapılabileceği gösterildi.

Makalenin geri kalanı şu şekilde organize edildi: Bölüm 2, DDS hakkında artalan bilgileri verir. Örnek uygulama ve enine kesen ilgilerin belirlenmesi Bölüm 3'te anlatıldı. Bölüm 4 ilgiye yönelik programlama ile eklenen yeni özelliklerin detaylarını içerir. İlgili çalışmalar Bölüm 5'de, değerlendirmeler ve sonuç Bölüm 6'da yer almaktadır.



Şekil 1. DDS Dağıtık Sistem Mimarisi

2. Veri Dağıtım Servisi (DDS)

Veri Dağıtım Servisi (Data Distribution Service - DDS) [13], özellikleri Object Management Group (OMG) tarafından [12] standartlaştırılmış bir arakatman yazılımıdır. Dağıtık sistemlerde ana haberleşme çatısı olarak kullanılan DDS arakatmanının çalışma prensiplerini detaylı olarak anlatan pek çok referans bulunmaktadır [1, 2, 3, 8, 9, 10, 13]. Burada DDS'nin temel özelliklerinden bahsedilecektir.

2.1. DDS Bileşenleri

Bölge (domain): Bölge, birbiri ile iletişim kuracak uygulamaları bağlamak için kullanılan temel yapıdır. Tüm veri okuyucuları ve veri yazıcıları tanımlı bölge içerisinde iletişim kurabilirler. Dağıtık bir sistemde birden fazla bölge tanımlanıp verilerin sadece ilgili bölgelere gitmesi sağlanabilir.

Bölge İştirakçisi (domain participant): Bölge iştirakçisi, yazılım geliştiricinin varsayılan servis kalitesi (QoS) parametrelerini, ilgili bölgedeki tüm veri yazıcıları, okuyucuları, yayıncıları ve aboneleri için atamasını sağlar.

Veri Yazıcısı (data writer): Veri yazıcı, DDS veri bölgesindeki uygulamaların veri yayınlamasını sağlayan temel erişim noktasıdır.

Yayıncı (publisher): Yayıncı, veri yazıcıları gruplamak için kullanılır. Bir DDS bölge iştirakçisinin sadece bir yayıncısı olmasına rağmen bir yayıncının birden fazla veri yazıcısı olabilir.

Veri okuyucu (data reader): Veri okuyucu, DDS veri bölgesindeki uygulamaların veriyi almasını sağlayan temel erişim noktasıdır.

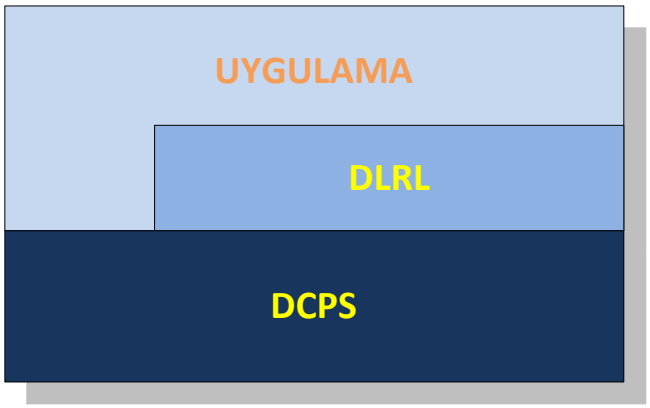
Abone (subscriber): Abone, veri okuyucuları gruplamak için kullanılır. Bir DDS bölge iştirakçi için sadece bir abone tanımlanabilirken bir abonenin birden fazla veri okuyucusu olabilir.

Konu (topic): Konu, yayıncı ile abone arasındaki iletişim noktasıdır. DDS sisteminde iletişimin sağlanabilmesi için yayıncının ve abonenin konusu aynı olmalıdır. Bu nedenle bir bölge içindeki konu isimleri eşsiz (unique) olarak belirlenmelidir

2.2. DDS – Katmansal Mimari

DDS, dağıtık yazılım geliştirme için uygun servisler ve özellikler sunmaktadır. Şekil 2'de verildiği gibi DDS, temel olarak iki ana katmandan oluşmaktadır:

- Veri tabanlı yayınla-abone ol (Data Centric Publish-Subscribe – DCPS) katmanı ve



Şekil 2. DDS – Katmansal Mimari

• Yerel veri yeniden inşa (Data Local Reconstruction Layer – DLRL) katmanı.

DCPS, DDS'nin temel haberleşme katmanıdır. Yayınla-abone ol mimarisi ile çalışır. Uygulamaların içinde tanımlanan yayıncılar ve aboneler birbirleri ile konular üzerinden haberleşirler. Haberleşmede kullanılan verinin özellikleri arayüz tanımlama dili (Interface Definition Language - IDL) [17] ile belirtilir. Her bir iştirakçi (participant) veri yayınlayabilir ya da istediği verilere abone olabilir. Yayıncılar, olay (event) şeklinde veri üretirler ve bu veriler aboneler tarafından tüketilir.

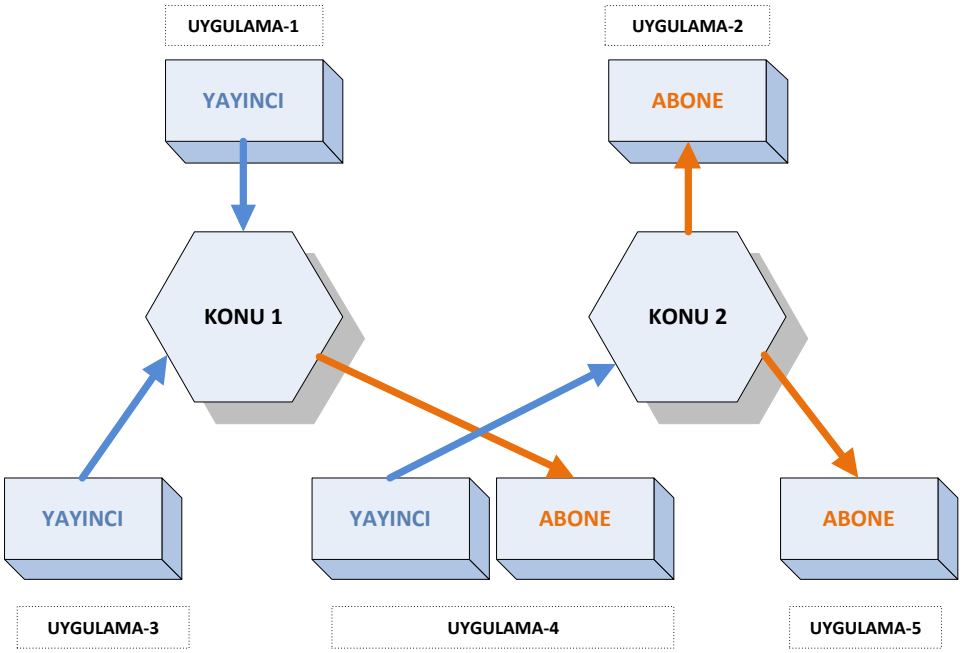
Bu mimarinin avantajı, yayıncı ve abonelerin birbirini tanımak zorunda olmamasıdır. Hatta yayıncı ve abone aynı anda hayatta olmak zorunda değildir. Bilmeleri gereken tek şey haberleştikleri konudaki verinin özellikleridir. Konu üzerinden gerçekleşen haberleşme bir gazete yazarı ile okuyucuları arasındaki haberleşme gibi düşünülebilir. Yazar, makalesini yazıp gazeteye gönderir ve makale yayınlanır. Bu haberleşmenin gerçekleşmesi için yazarın, aboneleri tanıması ve fiziksel yerlerini bilmesi gerekmez. Aynı durum gazete aboneleri için de geçerlidir.

DLRL'in temel görevi, nesneye yönelik programlamaya uygun olarak tasarlanmış bilgilerin, veri tabanlı yapıya çevrilmesi için (ve tersi akışlar için) çevirici görevi görmektir. DLRL katmanı isteğe bağlı olarak standarda eklenmiştir ve piyasadaki pek çok DDS sağlayıcı tarafından kullanılmamaktadır. DDS üzerinden veri iletişiminin nasıl gerçekleştiğini gösteren iletişim şeması Şekil 3'de [10] verilmiştir.

2.3. DDS –Servis Kalitesi

DDS, "Servis Kalitesi" (Quality of Service – QoS) isimli özel iletişim parametreleri kullanır. Yayınla – abone ol mimarisi kullanan başka arakatman yazılımları (ör. High Level Architecture - HLA) olmasına rağmen, DDS ön tanımlı QoS kullanımı ile dağıtık yazılım geliştirme sürecini kolaylaştırmakta ve hızlandırmaktadır.

QoS parametrelerine örnek olarak: yayınlama hızı (rate of publication), abone olma hızı (rate of subscription), verinin ne kadar süre ile geçerli olduğu (deadline), en son kaç adet verinin saklanacağı (history), gibi servis kaliteleri verilebilir.



Şekil 3. DDS İletişim Şeması

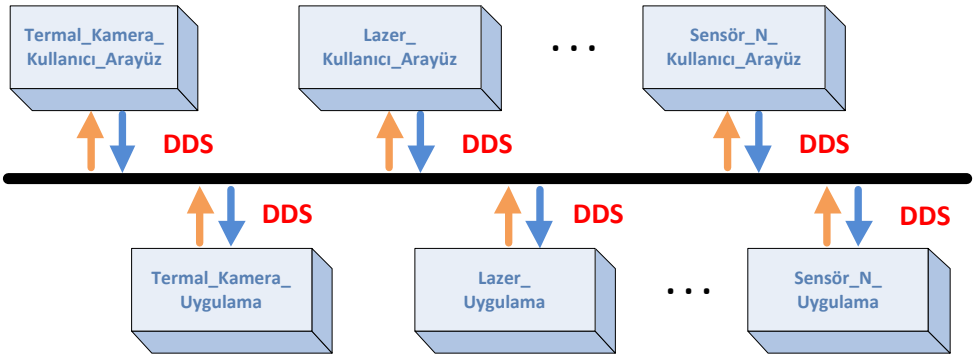
Ayrıca, sürekli gönderilen veriler için *continuous*, sisteme sonradan katılan abonelere de gönderilmesi gerekli veriler için *reliable*, veriyi DDS veri tabanında saklamak için *persistent*, sıklıkla kullanılan QoS parametrelerindendir. Burada not edilmesi gereken önemli bir konu da, veri iletişimin gerçekleşebilmesi için yayıncı, abone ve konu için tanımlanan servis kalitelerinin uyumlu olmasıdır. Aksi halde veri iletişimi gerçekleşmez.

3. Örnek Uygulama

3.1. C2S Yazılımı

Makalede örnek uygulama olarak DDS tabanlı bir mini komuta kontrol sistemi (Command and Control System – C2S) geliştirildi. Enine kesen ilgilere odaklanabilmek için, C2S'nin tasarımı basit tutulmakla birlikte enine kesen ilgi problemlerini ve çözümleri gösterecek özellikler eklendi. Enine kesen ilgilerle neden karşılaşıldığı konusu üzerinde duruldu. C2S'nin geliştirilmesinde, OCI [20] tarafından geliştirilen, açık kaynak kodlu OpenDDS yazılımı [4] kullanıldı. Kodlar, C++ ve Java dilleri kullanılarak nesneye yönelik programlama yaklaşımı ile yazıldı. Geliştirme, Eclipse [7] yazılım geliştirme aracı üzerinde gerçekleştirildi.

Bu amaçla gerçekleştirilen C2S, örnek olarak, sadece termal kamera ve lazer sensörlerinden gelen bilgileri alıp işleyerek bazı bilgileri kullanıcı arayüzünde gösterecek şekilde tasarlandı. Kullanıcının, arayüz üzerinden yaptığı değişiklikleri ve sensör ayarlarını DDS üzerinden sensörlere iletme özelliği eklendi. DDS tabanlı C2S'nin iletişim altyapısı Şekil-4'te verilmiştir. Örnek çalışma kapsamında iki sensör kullanılmış olmasına rağmen, tasarlanan sistem mimarisi daha büyük sistemlere kolaylıkla ölçeklenebilir şekilde geliştirilmiştir. Benzer şekilde, komuta kontrol altyapısı ve ilerideki bölümlerde açıklanacak olan ilgiler gelişmiş ve karmaşık sistemler için de aynen kullanılabilir olarak tasarlanmıştır.



Şekil 4. Örnek Çalışma – C2S

3.2. Enine Kesen İlgilerin Belirlenmesi

Yukarıda belirtildiği gibi, örnek uygulama C2S nesneye yönelik programlama yaklaşımı geliştirilmiştir. C2S'deki uygulama ve kullanıcı arayüz sınıflarının benzer özellikler taşıyacağı açıktır. Örneğin, Şekil 4'te verilen her bir uygulama birimi tipik olarak veri doğrulama, veri işleme, veri kaydetme, sistem günlüğü tutma (logging) vb. fonksiyonları içerecektir. Bu ve benzeri fonksiyonların, yazılım içinde saçılmış (code scattering) ve dağıtık (code tangling) şekilde bulunması, yazılımın yönetilmesini ve idame ettirilmesini zorlaştırmaktadır. Örnek uygulamadaki enine kesen ilgiler, sadece nesneye yönelik programlamanın doğal yapısından kaynaklanmamakta, aynı zamanda kullanılan dağıtık mimari nedeniyle de yazılımda yer almaktadır. Örneğin, C2S'deki pek çok yayıncı ve abone için ayrı ayrı zamanlama ve iletişim kodları geliştirilmiştir. Bu ilgilerin tek bir yerde toplanması mümkün olmadığı için yazılım içinde dağıtık olarak yer almışlardır. Tanımlanan DDS servis kaliteleri için de durum benzerdir.

Enine kesen ilgiler problemleri için İlgiye Yönelik Programlama ile kolay ve hızlı çözümler üretilebilir [19]. Tanımlanan ilgiler ile yukarıda bahsedilen problemler için çözüm üretmek mümkündür. Örneğin, tüm sistem günlüğü tutma işlemlerinden önce başka bir fonksiyon çağırılmak isteniyorsa (ör. sabit diskte kalan yerin kontrol edilmesi gibi), yazılacak tek bir ilgi ile, bu değişiklik gerçekleştirilebilir.

Yukarıda bahsedilen enine kesen ilgiler, nesneye yönelik programlama yaklaşımı ile geliştirilen yazılımlarda tipik olarak bulunmaktadır. Bu makalede, DDS tabanlı sistemlerde karşımıza çıkan enine kesen ilgi problemleri üzerinde durulmuştur. Bu makalede, DDS tabanlı sistemlerin geliştirilmesinde, sıklıkla kullanılabilecek ilgilere yer verilmiştir:

DDS İzleme İlgileri (DDS Monitoring Concern): DDS tabanlı komuta kontrol sistemleri dağıtık mimaride tasarlanırlar ve yüzlerce düğüm içerebilirler. Tüm düğümlerde yayıncılar ve aboneler bulunabilir. Belli bir anda, tüm düğümlerdeki DDS bileşenlerinin (yayıncılar, aboneler, vs.) izlenmesi yazılım geliştirme ve test faaliyetlerinde önemli olduğu gibi yazılımın gerçek ortamda izlenmesi sırasında da önemlidir. Klasik yöntemlerle DDS bileşenlerinin izlenmesi çok zahmetli olabilir ve/veya büyük kod değişiklikleri gerektirebilir. Özellikle, kod satır sayısı

arttıkça, yazılım geliştirme faaliyetleri sırasında aktif olan ancak haberleşemeyen yayıncı ve abonelerin tespiti büyük problem teşkil eder. Bu nedenle, ilgiye yönelik programlama ile geliştirilecek DDS izleme ilgileri, yazılımın geliştirilmesinde ve test edilmesinde önemli rol oynamaktadırlar.

Servis Kalitesi Uygulama İlgisi (EnforceQoS Control Concern): Bölüm 2'de belirtildiği gibi, DDS sistemlerinde haberleşmenin gerçekleşebilmesi için yayıncı ve abonenin aynı konu üzerinden konuşması ve bu üç DDS bileşenin de (yayıncı, abone ve konu) servis kalitelerinin uyumlu olması gereklidir. Servis kaliteleri uyumlu olmazsa, yayıncı ve abone hayatta olduğu halde, konu isimleri aynı bile olsa veri iletişimi gerçekleşmeyecektir. QoS parametrelerinin izlenmesini ve değiştirilmesini sağlayacak ilgiler DDS tabanlı sistemlerin geliştirilmesinde önemlidir.

Dil Değişim İlgisi (Switching Language Concern): Bir komuta kontrol programı kullanıcı arayüzünde ya da veri kayıt/saklama işlemlerinde farklı dillerde çalışmayı desteklemelidir. Çoklu dil desteği sağlamak için özellikle kullanıcı arayüzündeki etiketlere, menülere ve diğer arayüz bileşenlerine erişerek farklı dil desteği eklenmelidir. Özellikle kalıt kod (legacy code) kullanıldığı durumlarda, enine kesen ilginin dağıtık olması sebebiyle bu özelliği eklemek büyük kod değişiklikleri, hatta yapısal değişiklikler gerektirebilir. Ancak, bu amaçla geliştirilecek bir ilgi ile bu problem kolayca çözülebilir.

4. İlgiye Yönelik Programlama

Bölüm 3.1'de, örnek uygulama olarak geliştirilen C2S yazılımından ve Bölüm 3.2'de DDS tabanlı yazılımlardaki enine kesen ilgi problemlerinden bahsedilmişti. Bu bölümde ise belirlenen problemlere, ilgiye yönelik programlama ile nasıl çözüm bulunabileceğinden bahsedilecek ve geliştirilen ilgilerin detayları verilecektir.

İlgilerin kodlanmasında, açık kaynak kodlu AspectJ [5] kütüphanesi kullanılmıştır. AspectJ ve ilgiye yönelik programlama detayları için AspectJ programlama kılavuzuna bakılabilir [6].

4.1. DDS İzleme İlgisi (DDS Monitoring Concern)

Nesneye yönelik tasarım ile geliştirilmiş DDS tabanlı bir sistemde, belli bir anda aktif olan DDS bileşenlerini ve/veya sistemdeki verileri izleyebilmek için DDS bölgesindeki (domain) tüm yayıncılar, aboneler ve veriler taranmalıdır. Sonradan yazılıma eklenen modüllerde de izleme ile ilgili metotların kodlanması gereklidir. Ancak, ilgiye yönelik programlama ile sistemde aktif olan tüm yayıncı ve aboneler izlenebildiği gibi, bunların istenilen özellikleri kolayca listelenebilir. Ayrıca, yeni yazılım modülleri eklendiğinde, izleme fonksiyonlarının eklenmesine gerek kalmaz.

Aşağıda, C2S için geliştirilen dört farklı izleme ilgisi anlatılmış ve ilgiye yönelik programlama ile geliştirilen AspectJ kodları verilmiştir. Bu dört ilgi, modüler olması açısından ayrı ayrı geliştirilmiştir. Ancak, proje gereksinimlerine göre, bu dört ilgi, tek bir ilgi altında da toplanabilir.

```
pointcut pblshr(Subscriber sb, String s, QosType q) : execution
(Subscriber.new(String[],String,*,QosType)) &&
args(String[],s,CallBackDelegate,q) && target(sb);

after(Subscriber sb, String s, QosType q): pblshr(sb,s,q){
     String str = "\n--- List Subscribers ASPECT: \n";
     str += "new subscriber - class name: " + s +"\n";
     str += "new subscriber - type name: " + sb.typeName +"\n";
     str += "new subscriber - qos type: " + q.toString() + "\n";
    Monitoring.textArea.setText(Monitoring.textArea.getText() +
str);
}
```

Şekil 5. Abone Listele (*ListsSubscribers*) İlgisi

• **Abone Listele (ListsSubscribers) İlgisi:** ListSubscribers ilgisi ile DDS tabanlı bir sistemdeki aboneler ve özellikleri izlenebilir. Bu ilgi için geliştirilen pointcut ve tavsiye (advice) Şekil-5'de verilmiştir.

• **Yayıncı Listele (ListPublishers) İlgisi:** ListPublisher ilgisi kullanılarak, DDS tabanlı bir sistemdeki yayıncılar ve özellikleri listelenebilir. DDS'de yayıncı ve abone tanımlamaları farklı olduğu için, yayıncı izleme ilgisi yukarıda verilen abone izleme ilgisinden farklıdır (Şekil-6).

• **Abone Olunan Veri Tipini Al (GetTypeOfSubcribedData) İlgisi:** Abone olunan verinin tipinin ve gelen veri değerlerinin izlenmesi bu ilgi ile sağlanmaktadır Bu ilgide, abone olunan verinin sınıfı, özellikleri ve değerleri izlenmektedir. Bu ilgi, yanlış veri ayarlarından kaynaklanan hataların ayrıştırılmasında önemli rol oynamaktadır. Bu ilgi için geliştirilen pointcut ve tavsiye Şekil-7'de verilmiştir.

• **Abone Olunan Veri Tipini Al (GetTypeOfSubcribedData) İlgisi:** Abone olunan verinin tipinin ve gelen veri değerlerinin izlenmesi bu ilgi ile sağlanmaktadır Bu ilgide, abone olunan verinin sınıfı, özellikleri ve değerleri izlenmektedir. Bu ilgi, yanlış veri ayarlarından kaynaklanan hataların ayrıştırılmasında önemli rol oynamaktadır. Bu ilgi için geliştirilen pointcut ve tavsiye Şekil-7'de verilmiştir.

• **Yayınlanan Veri Tipini Al (GetTypeOfPublishedData) İlgisi:** Yukarıda belirtilen GetTypeOfSubscribedData ilgisine benzer şekilde tasarlanmıştır. Bu ilgi, yayınlanan verinin ismini, sınıfını, tipini ve değerlerini izlemek için tasarlanmıştır. Özellikle yazılım geliştirme ve test aşamalarında sıklıkla ihtiyaç duyulan GetTypeOfPublishedData ilgisinin kodları Şekil-8'de belirtilmiştir.

```
pointcut pblshr(Publisher p, String s, QosType q) : execution
(Publisher.new(String[],String,QosType)) && args(String[],s,q) &&
target(p);

after(Publisher p, String s, QosType q): pblshr(p,s,q){
     String str = "\n--- List Publishers ASPECT ---\n";
     str += "new publisher - class name: " + s + "\n";
     str += "new publisher - type name: " + p.typeName + "\n";
     str += "new publisher - qos type: " + q.toString() + "\n";
     Monitoring.textArea.setText(Monitoring.textArea.getText() +
str);
}
```

Şekil 6. Yayıncı Listele (*ListPublisher*) İlgisi

```
pointcut p(GenericDataReaderListener g, String s, CallBackDelegate
c): execution(GenericDataReaderListener.new(String,
CallBackDelegate)) && args(s,c) && target(g);

after(GenericDataReaderListener g, String s, CallBackDelegate c):
p(g,s,c){
     String str = "\n--- subscribing data: \n";
     s +=" Call Back Class Name: " + s + "\n";
     s +=" Call Back Class Simple Name: " +
     c.getCallBackClass().getSimpleName()+"\n";
     try {
          s +=" Call Back Method: " +
c.getCallBackMethod().getName() +"\n";
     } catch(Exception e) {
          s +="Exception: " + e.toString() +"\n";
     }
     s +=" Call Back Object: " + c.getCallBackObject().toString()
+ "\n";
     Monitoring.textArea.setText(Monitoring.textArea.getText() +
str);
}
```

Şekil 7. Abone Olunan Verinin Tipini Al (*GetTypeOfSubscribedData*) İlgisi

4.2. Servis Kalitesi Uygulama (*Enforce QoS*) İlgisi

Bölüm 2.3'de belirtildiği gibi, DDS tabanlı sistemlerde, yayıncı, abone ve konu için tanımlanmış olan QoS parametreleri uyumsuzsa veri alış-verişi gerçekleşmez. Böyle bir durumun tespit edilmesi, geleneksel yazılım geliştirme yöntemleri ile bir hayli zordur. Bu nedenle, aynı konu üzerinden haberleşmeleri istenen yayıncıların ve abonelerin QoS değerlerinin izlenmesi, gerekirse değiştirilmesi, DDS tabanlı sistemlerin geliştirilmesinde önemlidir. Bu amaçla geliştirilen EnforceQoS ilgisi, yayıncı ve abonenin QoS parametrelerinin izlenmesini sağlar. QoS parametrelerinin dinamik olarak nasıl değiştirilebileceğini göstermek için EnforceQoS ilgisi, örnek uygulamamız C2S'de haberleşme için ihtiyaç duyulan Reliable QoS parametresi içim geliştirilmiştir. EnforceQoS ilgisi, QoS değerinin Reliable olarak tanımlı olup olmadığını kontrol eder, bu şekilde tanımlı değilse uyarı mesajı tanımlayarak QoS değerini Reliable yapar. EnforceQoS ilgisinin pointcut ve tavsiyeleri Şekil 9'da verilmiştir.

```
pointcut p(Object obj):
  execution(* *.publishData(Object)) && args(obj);

after(Object obj): p(obj){
  String str = "\n--- publishing data: \n";
  str += obj.getClass().getSimpleName() + " is publishing...\n";
  for (int i = 0; i < obj.getClass().getFields().length; i++){
    Object bl = new Object();
    try {
     if ((obj.getClass().getFields()[i].getType()
         .toString()) == "boolean")
       bl = (boolean)
         (obj.getClass().getFields()[i]. getBoolean(obj));
     else if
       ((obj.getClass().getFields()[i].getType().toString())
          == "int")
        bl = (int) (obj.getClass().getFields()[i]. getInt(obj));
     else if
        ((obj.getClass().getFields()[i].getType().toString())
          == "double")
         bl =  (double)
          (obj.getClass().getFields()[i]. getDouble(obj));
     else
         bl = obj.getClass().getFields()[i].get(obj);
    } catch (Exception e) {
         str += e.toString() + "\n";
    }
    str += obj.getClass().getFields()[i].getType().toString();
    str += " - " + obj.getClass().getFields()[i].getName();
    str += " - " + bl+"\n";
  }
  Monitoring.textArea.setText(Monitoring.textArea.getText()+str);
}
```

Şekil 8. Yayınlanan an Veri Tipini Al (*GetTypeOfPublishedData*) İlgisi

```
pointcut pblshr(QosType q) : execution
(Publisher.new(*,*,QosType)) && args(String[],String, q) &&
within(com.c2s.libdds.*);

void around(QosType q): pblshr(q){
    String str = "\n--- EnforceQoS ASPECT: " + q.toString();
    if(q != QosType.ReliableQos) {
         str += "do not use " + q.toString() + "type QoS\n";
         str += "instead use RELIABLE QoS\n";
         Monitoring.textArea.setText(Monitoring.textArea.
           getText() + str);
         q = QosType.ReliableQos;
         proceed(q);
    }else {
         str += q.toString() + "type QoS is true !\n";
         Monitoring.textArea.setText(Monitoring.textArea.
           getText() + str);
         proceed(q);
    }
}
```

Şekil 9. Servis Kalitesi Uygulama (*EnforceQoS*) İlgisi

```
privileged public aspect SwitchLanguage {
    declare parents: TrackControlPanelAbstract implements
         ActionListener;
    private JCheckBox
         TrackControlPanelAbstract.switchLanguageCheckBox;

     public void
     TrackControlPanelAbstract.actionPerformed(ActionEvent arg0)
     {
         if(switchLanguageCheckBox.isSelected()){
              isSendAllowedCheckBox.setText("Gonderme Acik");
              isOffsetModeCheckBox.setText("Ofset Modu");
              isControl1CheckBox.setText( "Kontrol");
         } else {
              isSendAllowedCheckBox.setText("Send Allowed");
              isOffsetModeCheckBox.setText("Offset Mode");
              isControl1CheckBox.setText("Control");
         }
     }
     after(TrackControlPanelAbstract r) :
         execution (TrackControlPanelAbstract.new(..)) &&
     this(r){
             r.switchLanguageCheckBox = new JCheckBox
               ("Switch Language");
             r.switchLanguageCheckBox.setBounds(20,100,120,25);
             r.add(r.switchLanguageCheckBox);
             r.switchLanguageCheckBox.addActionListener(r);
     }
}
```

Şekil 10. Dil Değişim (*SwitchingLanguage*) İlgisi

4.3. Dil Değişim (*SwithcingLanguage*) İlgisi

DDS tabanlı bir komuta kontrol programı kullanıcı arayüzünde ya da veri kayıt/saklama işlemlerinde farklı dillerde çalışmayı desteklemelidir. Özellikle kalıt kodların (*legacy code*) kullanıldığı bir geliştirme ortamında, kullanıcı arayüzündeki bileşenlere erişerek faklı dil desteği sağlamak büyük kod değişiklikleri ve zaman gerektirir.

Örnek C2S uygulamasında geliştirdiğimiz, dil değişim ilgisi ile DDS tabanlı sistemlerde çoklu dil desteğinin nasıl sağlanacağı gösterilmiştir. Dil değişim ilgisi, kullanıcı arayüzündeki etiketlerin, istenilen dilde gösterilmesini sağlar. Bu ilgi, mevcut kodlarda değişiklik yapmayı gerektirmediği için kolaylıkla kalıt kodlara da uyarlanabilir. Dil değişim ilgisinin pointcut ve tavsiyeleri Şekil-10'da verilmiştir.

3. İlgili Çalışmalar

Yazılımların modüler olarak geliştirilebilmesi, yeniden kullanım oranının artırılabilmesi, yönetim ve bakım çalışmalarının daha kolay yürütülebilmesi için Eric Gamma ve arkadaşları 1993'te [18], Buschmann ve arkadaşları 1996'da [11] yazılım tasarım örüntülerinin uygulanmasını önermişlerdir. Bütün bu faydalarına rağmen, yazılım örüntülerinin kullanılması, yazılımlardaki enine kesen ilgi problemlerini engelleyememektedir.

Nesneye yönelik programlama ile geliştirilen yazılımlardaki enine kesen ilgi problemlerinin çözümü için 1997'de Kiczales, "İlgiye Yönelik Programlama (Aspect Oriented Programming)" metodolojisini ortaya koymuştur [19]. Ancak bu çözümde, yazılım örüntüleri

gibi jenerik bir çözümdür ve doğrudan dağıtık sistemlerle ve DDS ile ilgili çözümler içermemektedir. Bu çalışmada ise özellikle DDS tabanlı sistemlerdeki enine kesen ilgi problemlerine odaklanılmıştır.

DDS arakatmanı geliştiren firmalar, yazılım ve test mühendislerine yardımcı olacak bazı geliştirme araçlarını da kendi ürünleri ile birlikte sunmaktadırlar. Object Computing Inc. (OCI) firmasının [20] geliştirdiği, OpenDDS [4] yazılımı ile birlikte sunulan DDS Monitoring Tool, Real Time Innovations (RTI) firmasının [15] geliştirdiği Connext DDS yazılımı ile birlikte sunulan SPY, PrismTech firması [16] tarafından geliştirilen Vortex yazılımı ile birlikte sunulan Tester, Tuner, Configurator yazılımları, milli DDS üreten MilSOFT firmasının geliştirdiği MilSOFT DDS yazılımı ile birlikte sunulan MilSOFT SPY yazılımı bu araçlara örnek olarak verilebilir. Bu araçlar sadece kendi firmalarının yazılımları ile çalışmaktadırlar. Yani, bu araçlar ile geliştirilmiş herhangi bir çözümün, diğer bir DDS üreticisinin kodlarına taşınması ve uygulanması mümkün değildir. Enine kesen ilgilere odaklanmış araçlar değillerdir, genelde izleme (monitoring) ve hızlı yazılım geliştirme amaçlı tasarlanmışlardır.

Bu makale, ön çalışma olarak gerçekleştirdiğimiz ilgiye yönelik programlama aracı [21] çalışmasının devamı olarak gerçekleştirilmiştir. Bu makalede, ön çalışmanın üzerinde ilerleyerek, DDS tabanlı sistemlerdeki, enine kesen ilgilere odaklanılmış ve ilgiye yönelik programlama yaklaşımının DDS tabanlı sistemler için kullanımı detaylandırılmıştır.

6. Bulgular ve Sonuçlar

Dağıtık sistemlerin geliştirilmesi uzun ve zorlu bir süreçtir. Bu süreçte pek çok yazılım ve test mühendisinin birlikte çalışarak proje ihtiyaçlarını gerçeklemesi gerekmektedir. DDS arakatmanı, dağıtık sistemler geliştirmek için son yıllarda yaygın olarak kullanılan açık standartlı ve gerçek zamanlı bir arakatman yazılımıdır. Dağıtık sistem geliştirmek için sağladığı güçlü mekanizmalara rağmen enine kesen ilgi problemleri DDS arakatmanının tasarımında çok fazla düşünülmüş bir konu değildir.

Bu makalede, veri dağıtım servisi (Data Distribution Service – DDS) tabanlı dağıtık sistemlerdeki enine kesen ilgilere çözüm üretebilmek için, ilgiye yönelik programlama (Aspect Oriented Programming – AOP) yaklaşımı anlatıldı. Öncelikle, DDS arakatmanının temel haberleşme mekanizması olarak kullandığı yayınla–abone ol (publish–subscribe) mimarisi ve servis kalitesi parametreleri (Quality of Service – QoS) hakkında temel bilgiler verildi.

İlgiye yönelik programlama yaklaşımının DDS tabanlı sistemlerde kullanımını detaylandırabilmek için, savunma sanayimizdeki pek çok projede geliştirmekte olduğumuz komuta kontrol sistemlerini temsil eden bir mini komuta kontrol sistemi (Command and Control System – C2S) örnek çalışma olarak kodlandı.

Nesneye yönelik programlama (Object Oriented Programming – OOP) ile geliştirilen C2S içindeki enine kesen ilgilerden örnekler verildi. Bu enine kesen ilgilerden, makale kapsamında sadece DDS arakatmanı ile ilgili 3 adet ilgiye odaklanıldı. Nesneye yönelik programlamanın ve kullanılan DDS arakatmanının, modüler ve ölçeklenebilir yazılım geliştirme için sağladığı güçlü mekanizmalara rağmen, enine kesen ilgiler için çabuk çözümler üretemediği görüldü. Diğer taraftan, ilgiye yönelik programlama yaklaşımı ile mevcut kodları değiştirmeden hızlı ve

faydalı çözümler üretilebileceği gösterildi. Belirlenen 3 adet ilgi için, C2S üzerinde kodlanan ilgiler detaylandırıldı ve AspectJ ile geliştirilen kodları verildi.

Aynı sonuçları, sadece nesneye yönelik programlama ile elde etmek için büyük kod değişiklikleri gerekirken, ilgiye yönelik programlama da kodları hiç değiştirmeden, sadece basit ilgiler geliştirerek faydalı çözümler üretilebileceği vurgulandı. Bu ilgilerin kalıt kodlara (legacy code) da kolaylıkla uyarlanabileceği belirtildi. Bu kapsamda geliştirilen ilgiler, DDS tabanlı dağıtık sistemler geliştirmede yazılım ve test mühendislerinin işlerini kolaylaştırmakta ve ürün geliştirme süresini azaltmaktadır.

Burada not edilmesi gereken başka bir konu da ilgiye yönelik programlama için kullanılabilecek, pek çok açık kaynaklı ve ücretsiz kütüphane ve derleyicinin bulunmasıdır. Bu kütüphane ve derleyicilerin yardımı ile kolaylıkla geliştirilebilecek ilgiler ile kısa sürede faydalı sonuçlar elde etmek mümkündür. Alternatif olarak kullanılabilecek olan DDS üretici firmaların geliştirme araçlarının pek çoğu açık kaynak kodlu değildir ve oldukça yüksek maliyetlidir.

İlgiye yönelik programlamanın yukarıda belirtilen faydalarına rağmen, performans problemlerine yol açabileceği not edilmelidir. Çünkü ilgiye yönelik programlama araçları / kütüphaneleri, ilgilerin gerçekleştirmesinde yansıma (*reflection*) mekanizmaları kullanırlar. Bu, sistem performansını etkileyen bir yaklaşımdır. İlgiye yönelik programlama, sorun çözme ve test amaçlı kullanımda ideal olmasına rağmen, son ürünlerin içinde yer alması gerektiğinde performans problemlerini göz önünde bulundurmanın ve detaylı performans testleri yapmanın önemli olduğu düşünülmektedir.

Bu çalışma kapsamında geliştirilen örnek ilgilerin daha büyük DDS tabanlı sistemlerde, aynen ya da küçük değişiklerle kullanılabileceği, eklenmesi düşünülen yeni ilgiler için örnek teşkil edebileceği düşünülmektedir. Bu şekilde, DDS tabanlı dağıtık sistem geliştiren yazılım ve test mühendislerine kolaylık ve zaman tasarrufu sağlayacağı değerlendirilmektedir.

Teşekkür

Bu çalışmanın tamamlanmasındaki tüm katkıları için Prof. Dr. Bedir Tekinerdoğan'a (Wageningen Üniversitesi, Hollanda) teşekkür ederiz.

Kaynakça

[1] Castellote G.P., Farabaugh B. 2005. An Introduction to DDS and Data-Centric Communications: Teknik Rapor, Real Time Innovations.
[2] Köksal Ö. 2008. DDS Arakatmanı Nedir?: Teknik Rapor, ASELSAN.
[3] Köksal Ö., Bozkurt A. 2009. DDS Arakatmanı Çalışma Prensipleri: Teknik Rapor, ASELSAN.
[4] OpenDDS. http://www.opendds.org (Erişim Tarihi: 01.08.2016).
[5] AspectJ: http://eclipse.org/aspectj (Erişim Tarihi: 01.08.2016).
[6] AspectJ Programlama Klavuzu: http://www.eclipse.org/aspectj/doc/released/progguide/index.html (Erişim Tarihi: 01.08.2016).
[7] Eclipse: http://eclipse.org (Erişim Tarihi: 01.08.2016).
[8] Schmidt D.C., Corsaro A. ve Hag H.V. 2008. Addressing the Challenges of Tactical Information Management in Net-Centric Systems with DDS: The Journal of Defense Software Engineering, S. 24-29.
[9] Corsaro A. 2012. Quality of Service in Publish/Subscribe Middleware:

Emerging Communication, Cilt. 8, s. 79-97.

[10] Ryll M., Ratchev S. 2008. Towards a publish / subscribe control architecture for precision assembly with the Data Distribution Service: IFIP International Federation for Information Processing, Cilt. 260, s. 359-369.

[11] Buschmann, F., Meunier R., Rohnert, H., Sommerlad, P. ve Stal, M. 1996. Pattern-Oriented Software Architecture: A System of Patterns, Cilt. 1, s. : 476.

[12] Object Management Group: http://www.omg.org (Erişim Tarihi: 01.08.2016).

[13] Data Distribution Service: http://www.omg.org/spec/DDS (Erişim Tarihi: 01.08.2016).

[14] MilSOFT DDS, http://dds.milsoft.com.tr (Erişim Tarihi: 01.08.2016).

[15] Real Time Innovations DDS, https://www.rti.com/products (Erişim Tarihi: 01.08.2016).

[16] PrismTech, http://www.prismtech.com/vortex (Erişim Tarihi: 01.08.2016).

[17] Arayüz Tanımlama Dili (Interface Definition Language-IDL), http://www.omg.org/gettingstarted/omg_idl.htm (Erişim Tarihi: 01.08.2016).

[18] Gamma E., Helm R., Johnson R. E., ve Vlissides J. M. 1993. Design Patterns: Abstraction and Reuse of Object-Oriented Design: Proceedings of the 7th European Conference on Object Oriented Programming, s. 406-431.

[19] Kiczales G. 1996. Aspect Oriented Programming: ACM Computing Surveys, Cilt. 154, s. 28.

[20] Object Computing INC, http://www.ociweb.com (Erişim Tarihi: 01.08.2016).

[21] Köksal Ö. ve Akyüz M. 2011. Aspect Oriented Development Tool To Support Implementation of DDS Based Systems: Sunum, Fifth Turkish Aspect-Oriented Software Development Workshop.

[22] Common Object Request Broker Architecture (CORBA): http://www.omg.org/spec/CORBA (Erişim Tarihi: 01.08.2016).