İletişim Katmanı Yazılım Mimarisi

(Communication Layer Software Architecture)

İbrahim Karaaslan, Tanın Afacan, Emrah Demircan, Özgür Başol, Erman Zaim

Aselsan A.Ş., Ankara, Türkiye
{ikaraaslan, tafacan, edemircan, obasol, ezaim}@aselsan.com.tr

Özet. Bu makalede, nesneye dayalı İletişim Katmanı Yazılım Mimarisi (İKYM) sunulmuştur. İKYM, iletişim katmanlarını ve protokollerini gerçekleyen yazılımların tasarımında kullanılmak üzere geliştirilmiş bir mimaridir ve her bir katman için genel ve modüler bir yapı önerir. Bu mimari, iletişim katmalarına kolayca uygulanabilirken, yazılım geliştirme sürecinin her evresindeki kazanımları sayesinde son ürün maliyetini azaltmayı hedefler.

Anahtar Kelimeler. Yazılım Mimarisi, İletişim Katmanları, İletişim Protokolleri, Tasarım Şablonu, Yazılım Kalitesi

1 Giriş

Kullanıcı gereksinimlerindeki artıştan dolayı gitgide büyüyen ve karmaşıklaşan yazılımlar için yapılan mimari tasarım çalışmaları, artık algoritma ve veri yapılarının tasarımı çalışmalarından daha öncelikli hale gelmiştir. Karmaşık yazılım sistemlerinin kaliteli olarak tasarlanması zorunluluğu, yeni bir takım problemleri de beraberinde getirmiştir. Bu problemlerin çözümü sürecinde sistem ve yazılım mimarisi, kalite nitelikleri, mimari kararlar ve yazılım şablonları gibi konular ön plana çıkmıştır.

Sistem mimarisi, karmaşık sistemlerin birbirleriyle ilişkili daha küçük parçalara bölünmesini ve bu parçalar arasındaki ilişkilerle daha kolayca ortaya çıkan ve daha belirgin bir biçimde görülebilen büyük resmin oluşturulmasını göz önünde bulundurur. Yazılım mimarisi ise, yazılım gereksinimleri ile gerçekleme arasında köprü görevini üstlenir. IEEE, yazılım mimarisini, bir sistemin temel yapısı, bileşenlerden oluşan, birbirleriyle ve çevreyle ilişkileri olan, sistemin tasarımını ve evrimini yöneten ilkeler olarak tanımlar [1].

Yazılım sistem gereksinimlerini sağlamak ve söz konusu sistem üzerindeki riskleri azaltmak için yazılım geliştirme sürecinin ilk aşamalarından itibaren kalite ölçütlerinin göz önünde tutulması gerekmektedir. Yazılım kalitesi, bir ürün veya hizmetin ima edilen veya belirlenen ihtiyaçlarını karşılamak için, yeteneğiyle ilişkilendirilen özellikler ve nitelikler şeklinde tanımlanır [2]. Riskleri ortadan kaldırmak ve tüm yazılım sistem başarısını kolaylaştırmak için yazılım kalite niteliklerinin yazılım geliştirme sürecinde çok önceden değerlendirilmesi gerektiğini vurgulamaktadır [3].

Mimari kararlar, yazılım sisteminin bütününü ya da bir veya birden çok çekirdek parçasını ilgilendiren tasarım kararlarıdır. Bu kararlar sistemin kalitelerini etkiler [4], [5]. Tipik kalite nitelikleri taşınabilirlik, bakım yapılabilirlik, uyarlanabilirliktir. Mimari kararlar, bir sistemin yazılım kalitesi nitelikleri gibi işlevsel olmayan gereksinimlerini dolaylı ya da dolaysız olarak etkileyebilmesinden dolayı çok önemlidir. Bu nedenle, tasarımcılar mimari kararların muhtemel yan etkilerini de dikkate almalıdır.

Son yıllarda yapılan yazılım mimarisi araştırmaları kapsamında, başta yazılım sistemlerinin genel yapısı olmak üzere, özellikle alt sistemler ile bileşenler arasındaki ilişkileri konu alan ilkesel çalışmalar yayınlanmıştır. Araştırmalar başlarda pratik yazılım çalışmaları olarak adlandırılırlarken, günümüze kadar olan süreçte karmaşık yazılım tasarımı ve geliştirmesi probleminin çözümünde somut bir yol gösterici görev üstlenmişlerdir. Bu çalışmaların yazılım dünyasında yer bulmasıyla beraber yazılım sistemlerinin geliştirilmesinde alınan mimari kararlar bu çalışmalarla eşgüdümlü hale gelmiştir. Güncelliğini koruyan çalışmalardan biri olan Şablon Tabanlı Yazılım Mimarisi de günümüzde çokça kullanılmakta olup, yazılım sistemleri tasarımında önemli bir rol oynamaktadır [6].

Yazılım şablonları yazılım mimarisinin anahtar kavramlarından biridir ve bu şablonlar kısaca bir problemin çözümü olarak ifade edilebilirler. Öyle ki bu şablonların yeniden kullanılması sayesinde genel bir ilkeye bağlı kalınarak problemlerin çözümü gerçekleştirilir. Böylece, şablonlar çeşitli sistem tasarımlarında benzeri görülebilecek tekrarlayan sorunlara rahatlıkla uygulanabilecek ortak bir çözüm sundukları için yazılım maliyetlerini düşürmektedirler. Örneğin, Gang of Four (GoF) tasarım şablonları, en çok kullanılan şablonlar arasında gösterilebilir [7].

Bu makalede anlatılan İletişim Katmanı Yazılım Mimarisi, iletişim katman ve protokollerini içeren bir yazılım sisteminin mimari tasarımını hedeflemektedir. Aynı kapsamdaki Protokol Yazılım Mimarisi konusunda çeşitli öncül çalışmalar da bulunmaktadır. Bu çalışmalardan [8], ortak protokol yapısını modelleyen tasarım şablonları sunar. Bu tasarım şablonlarından Protokol Sistem Şablonu protokol sistemini genel bir seviyede, Protokol Birim Şablonu sistemin aktif parçalarını ve Protokol Davranış Şablonu ise protokol sistem parçaları arasındaki iletişimi modeller. [9] şablon tabanlı protokol geliştirme yöntemleri ile ilgilenir.

Ancak, öncül çalışmalar, yazılım sistemlerini tasarım şablonu seviyesinde göz önüne almakta ve iletişim katmanlarının ve protokollerinin daha detaylı tasarımlarını sunmamaktadır. Bu nedenle, bu çalışmada, eksikliği hissedilen detayların da bulunduğu genel ve modüler bir mimari tasarım hedeflenmiş ve nesneye dayalı İletişim Katmanı Yazılım Mimarisi (İKYM) önerilmiştir.

Bu makale şu şekilde organize edilmiştir. 2. bölümde, iletişim katmanları genel olarak anlatılmış ve 3. bölüm'de, İKYM modeli sunulmuştur. 4. bölümde, İKYM'nin iletişim protokollerine nasıl uygulanacağı açıklanmış ve İKYM kullanılarak bazı protokoller modellenmiştir. Son bölümde ise çalışmanın sonuçları ve gelecekte yapılması düşünülen çalışmalar yer almaktadır.

2 İletişim Katmanları

International Standards Organization (ISO), iletişim ağlarındaki tasarım karmaşıklığını azaltmak üzere iletişim işini belli bir görevi üstlenmiş birçok basit katmana ayırmış ve üst üste yerleşen bu katmanlardan oluşan mimariyi Open System Interconnection (OSI) referans modeli olarak adlandırmıştır. Yedi Katman Referans Model olarak da tanımlanan bu model ağ cihazları arasında veri iletimi ve işlenmesini tanımlayan bir kavramdır. Öte yandan, The Defense Advance Research Projects Agency (DARPA) tarafından savunma ağlarını birbirine bağlamak için geliştirilmiş ve tanımlanmış olan TCP/IP Dört Katmanlı Referans Modeli de mevcuttur.

OSI ile TCP/IP arasındaki temel fark, OSI'de iletişim katman protokollerinin tanımlanmaması, TCP/IP'de ise modelin tanımlı protokoller içermesidir. Her iki modelin de ortak çıktısı iletişim katmanı kavramının kullanılmasıdır, ancak uygulamada çok yer bulan TCP/IP mimarisi yanında OSI modeli iletişim katmanı tanımları teorik anlamda daha yaygındır [10].

Birbirleriyle çeşitli iletişim katmanları üzerinden konuşabilen makineler eşdüzey öğeler olarak tanımlanır. Bu öğeler, işlemler, donanım cihazları hatta insanlar bile olabilir. Eşdüzey öğeler katmansal modelin her bir katmanında üzerinde anlaşılmış bir protokol aracılığıyla iletişim kurarlar. Gerçekte veriler bir makinedeki katman n'den direkt olarak başka bir makinedeki katman n'ye iletilmez. Bunun yerine, her katman veri ve kontrol bilgilerini en alt katmana ulaşana kadar hemen altındaki katmana geçirir. Katman 1'in altında gerçek iletişimin gerçekleştiği fiziksel ortam vardır.

Her bir iletişim katmanı ETSI, ANSI, ITU, IETF gibi kuruluşlar tarafından tanımlanmış bir veya daha fazla protokolden oluşabilir. Protokoller standartlaştırılmış kurallar kümesidir ve bulunduğu katman ile birbiri yerine de kullanılabilir. Protokoller ve katmanların ortak özellikleri şu şekilde sıralanabilir:

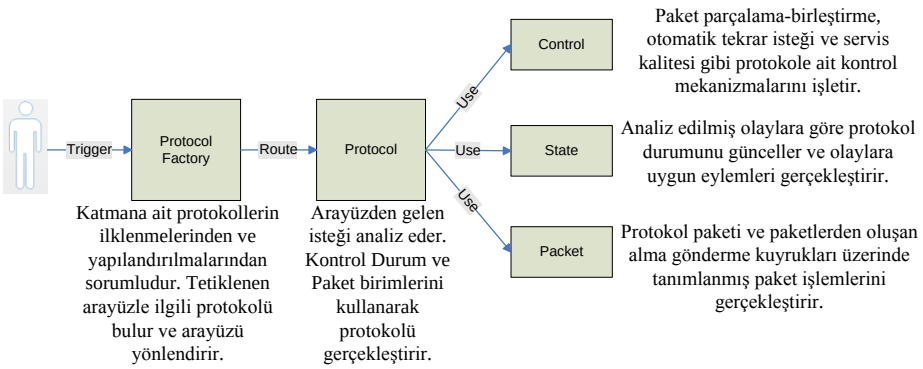
- Eşdüzey öğeler kontrol ya da kullanıcı verisi içeren mesaj veya paket alışverişiyle iletişirler.
- Bağlantılı veya bağlantısız hizmet sunabilirler.
- Veri iletimi için paket veya devre ağ anahtarlama yöntemleri kullanırlar.
- Yapılandırma parametrelerine sahiptirler.
- Hizmet almak veya sağlamak için bazı ara yüzleri vardır.
- Tetikleyici olayları uygun şekilde işleyebilmek için bir veya birden çok durumları olabilir.
- Öncelik verme, gecikme, hız, güvenilirlik gibi iletişim hizmet kalitesi gereksinimlerini çeşitli seviyelerde sağlayabilirler.
- Mesaj parçalama birleştirmeyi destekleyebilirler.
- Sıkışıklık ve akış kontrolü gibi hizmetleri sağlayabilirler.
- Otomatik Tekrar İsteği (ARQ), bütünlük kontrolü, hata bulma ve düzeltme gibi yöntemleri destekleyebilirler.

Yazılım sistemlerinde sıkça kullanılan iletişim yazılımları, katmansal model esas alınarak, katmanların ve katman protokollerinin yukarıda bahsi geçen genel kurallar,

gereksinimler ve özellikler çerçevesinde tasarlanması ve gerçeklenmesi ile oluşturulur.

3 İletişim Katmanı Yazılım Mimarisi Modeli

İKYM, iletişim katman ve protokollerini gerçekleyen yazılımların tasarımında kullanılmak üzere geliştirilmiş nesneye dayalı bir mimaridir ve her bir iletişim katmanı için genel ve modüler bir yapı önerir. Önerilen mimaride, iletişim katmanlarının ve bu katmanlardaki protokollerin daha önce bahsedilmiş olan ortak özellikleri kapsanmıştır. Bu nedenle, İKYM hem bir katman hem de bir protokol tasarım modelidir. İKYM Soyut Modeli Şekil 1'de verilmiştir.



Şekil 1. İletişim Katmanı Yazılım Mimari Soyut Modeli

İKYM, bir taraftan taşınabilirlik, bakım yapılabilirlik, uyarlanabilirlik ve verimlilik gibi bazı kalite niteliklerini sağlamayı hedeflerken diğer taraftan da mimari kararların sonuçlarını ve muhtemel yan etkilerini de göz önüne alır.

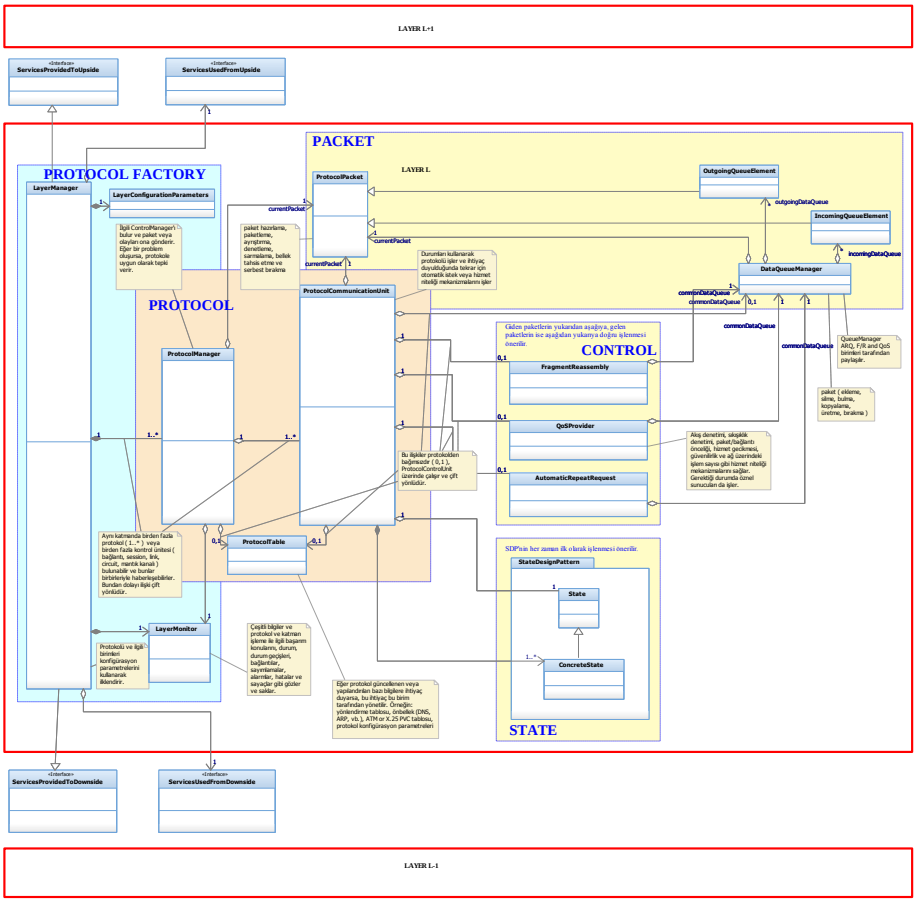
İKYM'nin tasarımında nesneye dayalı modelleme teknikleri kullanılmıştır. Mimari, Object Management Group (OMG) öncülüğünde desteklenen Model Temelli Mimari (Model Driven Architecture) temel alınarak Unified Modelling Language (UML) ile modellenmiştir.

Tasarımcılar İKYM'yi yeni ortamlara kolaylıkla taşıyabilir ve uyarlayabilir. İKYM, mimari tasarıma geç katılmış tasarımcılar için bile anlaşılabilirlik ve öğrenilebilirlik açısından kullanışlıdır. İKYM, Test veya Sistem Mühendisliği gibi tasarımcıyla çalışan gruplar için çözümlenebilirlik, değiştirilebilirlik ve test edilebilirlik açısından bakımı yapılabilirdir. Bu nedenle, nesneye dayalı İKYM kullanışlılık, bakım yapılabilirlik ve taşınabilirlik gibi yazılım kalite niteliklerini sağlar.

İKYM UML model, sınıflar ve bileşim (composition), türeme (realization), birleşme (aggregation) tarzındaki ilişkiler gibi nesneye dayalı bileşenlerden oluşmuştur. Mimarideki, bileşenler ve ilişkilerin ne olduğu hakkındaki kararlar katman ve protokol terimlerinin nesneye dayalı söz dizimi ile tanımlanmasından elde edilebilir.

- *Katman*: *Üst ve altdaki katmanlara* açtığı *hizmetleri* **gerçekleştirir (realization)** ve onlar tarafından açılan *hizmetleri* **kullanır (aggregation/1)**. **Bir veya birden çok** *protokole* **sahiptir (composition/1..*)**. *Protokolleri* ilgilendiren *yapılandırma bilgileri birimini* **içerir (composition/1)**.
- *Protokol***: Bir veya birden çok** *protokol iletişim birimine* (connection, session, circuit, logical channel vs.) **sahip olabilir(aggregation/1..*)**. A**n itibariyla alınan** *paketi* **(aggregation/1)** kontrol eder ve gerekiyorsa ilgili *iletişim birimine* yönlendirir ve *protokol iletişim birimlerinden* gelen (**bidirectional**) bilgileri **sahip olduğu (composition/1)** *monitör birimine* bildirir**.** Standarda uygun şekilde oluşan olaylara göre **bir veya birden çok** *durum* **kullanarak (aggregation/1..*)** *protokolü* işletir. Standartta tanımlıysa, *protokolle* ilgili bilgilerin tutulduğu bir *protokol tablosu* **vardır (aggregation/1)**. Standartta tanımlıysa, paketlerin QoS gereksinimlerini sağlayan bir *Quality of Service (QoS) birimi* **vardır (aggregation/1)**. Standartta tanımlıysa, üst katmandan gelen büyük paketleri parçalayan veya alt katmandan gelen *protokol paketlerini* gerekiyorsa birleştirdiği Fregmantation Reassembly (FR) birimi **vardır (aggregation/1)**. Standartta tanımlıysa, güvenilir paket iletimini sağlayan *Automatic Repeat Request (ARQ) birimi* **vardır (aggregation/1)**. *QoS, FR, ARQ birimlerinin* katmana gelen *protokol paketlerini* **ortak** olarak işledikleri **bir** *veri paket kuyruk yöneticisi* **vardır(aggregation/1)**. *Veri paket kuyruk yöneticisinin* **sıfır veya daha çok sayıda hem alma hem gönderme yönünde** *kuyruk elemanları* **vardır (aggregation/*)**. *Alma ve gönderme kuyruk elemanı protokol paketlerini* biriktirdiğinden **aynı zamanda bir** *protokol paketidir* (**generalization**). *Veri paket kuyruk yöneticisi* **an itibariyle gelen** *protokol paketini* **kullanarak (aggregation/1)** gerekli işlemleri yapar.

İKYM'nin bileşenleri, ilişkileri ve ilişkilerin yönleri yukarıda anlatıldığı gibi belirlenir. Yukarıdaki tanımlarda, bileşenler ve ilişkilerini belirten kelimeler sırasıyla italik ve koyu yazılmıştır. Sonuç olarak, Rhapsody aracı kullanılarak Şekil 2'de gösterilen UML Sınıf Modeli elde edilmiştir.



Şekil 2. İletişim Katmanı Yazılım Mimari Modeli

Protokolün durum makinelerini temsil etmek için, Gang of Four tarafından tanımlanan Durum Tasarım Şablonu kullanılmıştır. Tasarım şablonları [7] problemlere ortak çözümlerdir. Fakat bu çalışmada, Durum Tasarım Şablonunda (SDP) ProtocolCommunicationUnit and ConcreteState birimleri arasında bileşim (composition) ilişkisi kullanarak bir değişiklik yapılmıştır.

Katman ve protokollerin ortak yönleri [8] ve [9] gibi bazı çalışmalar tarafından ele alınmıştır. Fakat önceki çalışmalar yazılım sistemlerini tasarım şablonu seviyesinde ele almamakta ve daha detaylı bir tasarım sunmamaktadırlar. İKYM, tasarım şablonlarına göre daha detaylı bir model sunmakta ve bu nedenle gerçeklemeye doğru adım adım ilerleyebilmemizi sağlamaktadır.

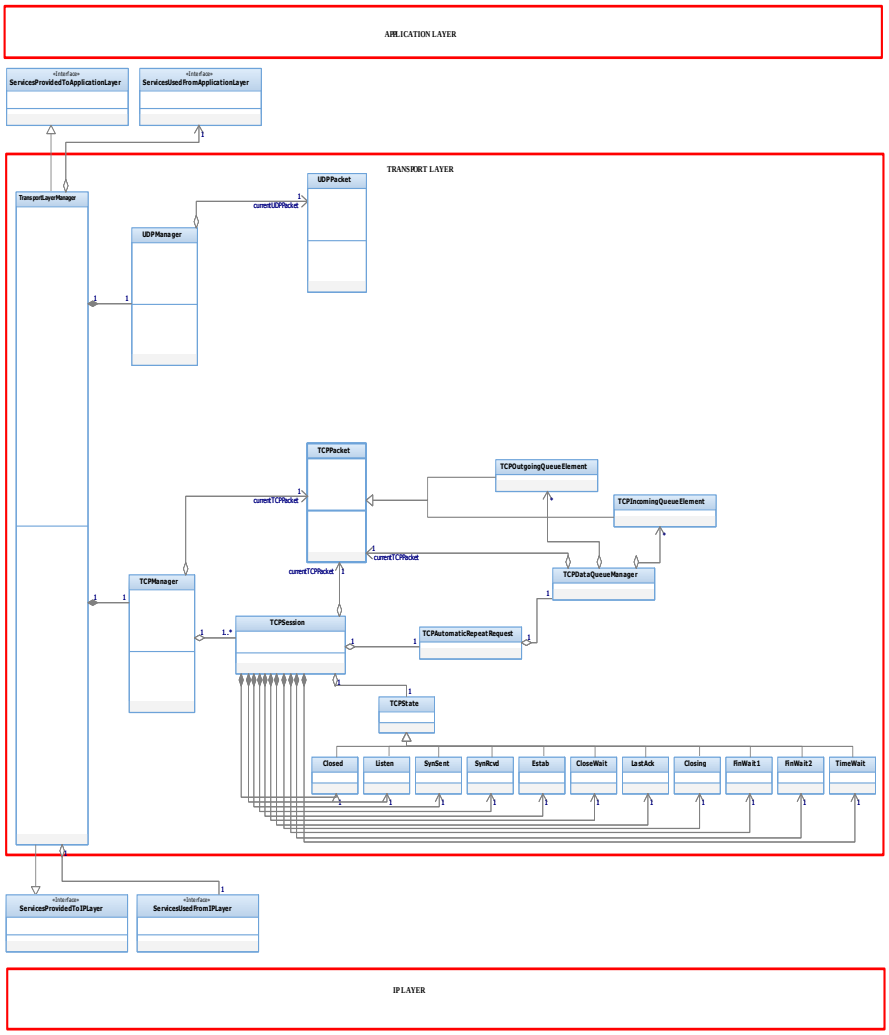
4 İKYM ile İletişim Katman Tasarımı

Tasarımcıların, standartları ve yazılım gereksinimlerini okuyarak söz konusu protokol ve katman gereksinimleri hakkında detaylı bir anlayışa sahip olmaları şarttır. Daha sonra, söz konusu katman için aşağıdakilerin uygulanabilir olup olmadığı tespit edilmelidir:

- Bileşen ve İlişkileri
- Arayüzler
- Yapılandırma Parametreleri
- Protokol ve Hizmet Gereksinimleri
- Protokol Durumları, Durum Makine Algoritması, Durum Olayları ve Geçişleri
- Protokol Paket Detayları ve Paket İşleme
- Test Örnekleri
- Katman ve Protokol Yönetim İşlemleri
- Yığın, Görev ve Zamanlama Yönetimi gibi İşletim Sistemi Gereksinimleri

Yukarıdakilere ve ilgili gereksinimlere dikkatlice karar verdikten sonra, tasarımcılar söz konusu mimarilerini, İKYM kullanarak modelleyebilirler. Daha sonra, bileşen ve ilişkilerin işlem ve değişken detaylarını tespit ederek detaylı tasarıma başlanabilir.

İKYM'nin iletişim katman ve protokollerine uygulanması, ilgili protokollerin ve bulundukları katmanların detaylı bir şekilde irdelenmesine bağlıdır. Ancak, farklı ekipler tarafından tasarlanan aynı katman mimarileri, hedeflenen standardın ilgili ekipler tarafından anlaşılması ve yorumlanması şekline göre farklılıklar gösterebilecektir. Bu nedenle, bu bölümde örneklendirilen TCP/IP taşıma katmanı modeli, farklılık yaratmayacak ölçüde basit tutulmuş ve bu kapsamda iki ana protokol, TCP ve UDP, için Şekil 3 TCP/IP Taşıma Katmanı Modeli'nde gösterilen mimari tasarım yapılmıştır.



Şekil 3. TCP/ IP Taşıma Katmanı Modeli

TCP/IP modeli taşıma katmanı, uygulama ve internet katmanı arasında bulunmaktadır. Örnek kapsamındaki internet katmanı IP protokolü ile gerçeklenmiş olup, taşıma katmanı ile IP protokolü hizmet alınan ve verilen ara yüzlerle birbirine bağlanmıştır. Diğer yandan, uygulama katmanı çok çeşitli uygulamalar barındırabilmesi sebebiyle genel ismiyle tanımlanmıştır. Taşıma katmanının uygulama katmanı ile bağlantısı yine hizmet alınan ve verilen ara yüzlerle yapılmıştır. Katmanlar arası iletişim, İKYM'de önerilen katman yöneticisi ile yapılmakta olup taşıma katmanı yöneticisi (TransportLayerManager) sınıfı olarak isimlendirilmiştir.

TCP/IP modeliyle uyumlu olarak belirtildiği üzere, bir iletişim katmanında bir ya da daha fazla protokol bulunabilmektedir. Şekil 3'de gösterilen modelde taşıma katmanı yöneticisine bağlanmış iki farklı taşıma katmanı protokolü (TCP ve UDP) bu-

lunmaktadır. Her iki protokol de kendi yöneticilerine (TCPManager, UDPManager) sahip olmakla beraber, uygulama katmanındaki olası bileşenlerden gelebilecek değişik gereksinimleri karşılayacak şekilde çalışmaktadır. TCP bağlantılı, güvenilir ve durum makinesi bulunan bir protokoldür. Bunun yanında, UDP bağlantısız, güvensiz ve durum makinesi bulunmayan basit bir protokoldür. Dolayısıyla, Şekil 3'de modellenmiş her iki protokol de özellikleri doğrultusunda seçilmiş İKYM bileşenleri kullanılarak tasarlanmıştır.

TCP'nin ana sınıfı olan TCPManager, TCP doğası gereği bu sınıfa bağlı kendi durum geçişleri olan oturumlarla etkileşim içindedir. TCP'nin bağlantılı bir protokol olması sebebiyle, sayısı gerçekleme yapılacak hedef platforma göre değişiklik gösterebilen oturumlara (TCPSession sınıfı) ihtiyaç duymaktadır. Ayrıca durum makinesi olan bir protokol olarak, her TCP oturumu, Durum Tasarım Şablonu esas alınarak tasarlanmış çeşitli TCP durum sınıfları ile bağlantılıdır. Protokolün güvenilirlik gereksinimleri, otomatik tekrar isteği mantığını çalıştıran TCPAutomaticRepeatRequest sınıfı ve bu kapsamda ihtiyaç duyulan paket kuyruklama amaçlı TCPDataQueueManager sınıfı ile sağlanır. Öte yandan, IP katmanından gelen TCP paketleri ve uygulama katmanından gelen bağlantı veya veri iletim isteklerinin işlenmesi, kodlanması veya çözülmesi işlemi yönetici, durum ve veri kuyruklama sınıflarına hizmet veren TCPPacket sınıfı tarafından yapılmaktadır.

UDP, TCP'den farklı olarak, sadece bir yönetici sınıfı ve bu sınıfa eşlik eden UDPPacket sınıfından oluşmaktadır. UDP'nin basit işlevselliği ve sınırlı yetenekleri nedeniyle, bu protokol az sayıda İKYM bileşeni ile modellenmiştir.

Sonuç olarak, çok basitten çok karmaşığa kadar çeşitli protokollerden oluşan iletişim katmanları, İKYM ve bu modelde kullanılan bileşenler kullanılarak kolaylıkla tasarlanabilir.

5 Sonuç

Yazılım mimarileri, yazılım sisteminin iç ve dış bileşenlerini, bileşenlerinin arasındaki ilişkileri tanımlar. Yanlış veya eksik mimari kararların ve mimarilerin çok maliyetli olduğu bilinmektedir. Tasarımcının doğru mimari kararlar aldığından ve yazılım gereksinimlerinin karşılandığından emin olmak için, modeller oluşturulur. Modeller kullanılarak da, var olan mimariler, tasarımlar analiz edilir, değişiklikler tartışılır ve paydaşlarla iletişim kurulur.

İKYM, protokol yazılım mühendislerinin ihtiyaçlarını karşılayacak bir iletişim katmanı tasarım modeli olmasının yanı sıra az deneyime sahip mühendisler için de belirsizlikleri ve karmaşıklığı ortadan kaldırarak faydalı olmayı hedefler. İKYM, mimari tasarım bileşenlerini, mimari tasarım kararlarını, iletişim protokollerinin ortak özelliklerini içerir.

Öncül çalışmalardan farklı olarak, İKYM, yazılım sistemini tasarım şablonu seviyesinde ele almakta ve daha detaylı bir mimari model önermektedir. İKYM, kolaylıkla uygulanabilir, taşınabilir, bakım yapılabilir ve uyarlanabilir bir mimari modeldir. Ayrıca, İKYM kullanılan yazılım tasarımlarında, adı geçen yazılım kalite nitelikleri-

nin sağlanması ve yazılım kalitesinin arttırılması hedeflenmektedir. İKYM kullanımının, yazılım geliştirme maliyetlerini büyük oranlarda azaltacağı düşünülmektedir.

Makalede, TCP ve UDP içeren TCP/IP taşıma katmanının, İKYM ile tasarlanabildiği gösterilmiştir.

Bir sonraki çalışmada ise İKYM'nin, yazılım geliştirme süreci ve yazılım kalitesi üzerindeki etkilerinin karşılaştırmalı olarak tartışılması yazarlar tarafından planlanmaktadır.

Kaynaklar

1. The Institute of Electrical and Electronics Engineers (IEEE) Standards Board. Recommended Practice for Architectural Description of Software-Intensive Systems (IEEE-Std-1471-2000) , September 2000
2. International Standards Organization: Information Technology - Software Product Quality - Part 1: Quality Model, ISO/IEC FDIS 9126-1
3. Francisca Losavio and Ledis Chirinos, Nicole Lévy and Amar Ramdane-Cherif, France "Quality Characteristics for Software Architecture" in Journal of Object Technology, vol. 2, no. 2, March-April 2003, pp. 133-150
4. Neil B. Harrison and Paris Avgeriou, "Leveraging Architecture Patterns to Satisfy Quality Attributes" F. Oquendo (Ed.): ECSA 2007, LNCS 4758, pp. 263–270, 2007 © Springer-Verlag Berlin Heidelberg 2007
5. Liliana Dobrica and Eila NiemelaÈ , Member, IEEE Computer Society, "A Survey on Software Architecture Analysis Methods" 0098-5589/02/$17.00. IEEE 2002
6. F. Buschmann, R. Meunier,H. Rohnert, P. Sornmerlad, M. Stal, "Pattern-Oriented Software Architecture, A system of Patterns", Volume1, February 2001
7. E. Gamma, R. Helm, R. Johnson and J. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software", Addson Wesley, 1995
8. Juha Parsinen and Markku Turunen , "Patterns for Protocol System Architecture", 2000
9. Youngjoon Byun, "Pattern-Based Design and Validation of Communication Protocols", 2003
10. Andrew S. Tanenbaum, "Computer Networks Ed.4", Prentice Hall, 2003