Makale Gönderilme Tarihi: 03-03-2021
Makale Kabul Tarihi: 22-12-2021

Türkiye'deki Yazılım Organizasyonlarının Mikroservis Tabanlı Mimaride Uyguladığı Analiz ve Tasarım Yöntemleri Üzerine Bir Araştırma

A Survey on Analysis and Design Practices of Turkish Software Organizations for Microservice Based Architectures

Hüseyin Ünlü, Burak Bilgin, Onur Demirörs

Bilgisayar Mühendisliği Bölümü
İzmir Yüksek Teknoloji Enstitüsü
{huseyinunlu, burakbilgin, onurdemirors}@iyte.edu.tr

Özet

Günümüzde esnek, güvenilir ve duyarlı (responsive) yazılımlara olan talebin artması ve bu talebe karşılık verebilen internet altyapısının olması yeni nesil bulut uygulamalarının gelişmesine öncülük etmiştir. Bunun sonucunda, bulut tabanlı dağıtık web uygulamalarının birlikte çalışarak oluşturduğu mikroservis tabanlı mimari popülerlik kazanmıştır. Mikroservis tabanlı mimari oldukça yeni olup bu mimariyi kullanarak yazılım geliştiren firmaların kültürlerini değiştirme gerekliliği doğmuştur. Ancak, literatürde mikroservis tabanlı mimaride analiz ve tasarım konusunda uygulanan yöntemleri ele alan çalışma sayısı çok azdır. Bu çalışmada, Türkiye'deki yazılım organizasyonlarının mikroservis tabanlı proje geliştirirken başvurdukları analiz ve tasarım yöntemlerini ele alan bir anket düzenlenmiştir. Anket sonuçları, mikroservis tabanlı proje konusunda tecrübesi olan yazılım organizasyonlarının analiz ve tasarım konusundaki bakış açılarını ortaya çıkarmaktadır. Elde edilen sonuçlar mikroservis kullanan organizasyonlar ile ilgili genel tabloyu göstermekte ve araştırmacılar için çalışma yapılabilecek konuları önermektedir.

Anahtar kelimeler: mikroservis, yazılım analizi, yazılım tasarımı, anket, bulut platformu

Abstract

The demand for more flexible, responsive and reliable software applications and the availability of internet infrastructure to respond to this demand, led to a new generation of cloud-based web applications. As a result, cloud-based distributed web applications working together in a microservice-based architecture has gained popularity. The concept of microservice-based architecture is quite new and software organizations need to transform their culture to develop applications in this fashion. However, there is a lack of research studies in the literature that explores the common practices for the analysis and design of microservices. Thus, we performed a survey to explore the organizational choices on software analysis and design when working with microservices. In this paper, we present the results from Turkey. The results provide a snapshot of the software industry that utilizes microservices and suggests a set of challenges researchers can focus on in the area.

Keywords: microservices, software analysis, software design, practitioner survey, cloud applications

1. Giriş

Günümüz yazılımları, ölçeklenebilir ve neredeyse sınırları olmayan kaynaklardan oluşan uygulamalar yönünde değişmektedir. Bulut platformu ise günümüz yazılımlarının ölçeklenebilir ve neredeyse sınırları olmayan kaynaklardan oluşabilme özelliklerini sağlamaya yöneliktir. Bulut platformu sayesinde sağlanan bu özellikler kullanılarak iş mantığı farklı platformlara dağıtılabilir [1]. Mikroservis tabanlı mimari, iş mantığını farklı platformlara dağıtabilmek için bir çözüm olarak yer alır. Mikroservis mimarisinde gevşek bağlı (loosely-coupled) servisler yüksek kohezyon içerisinde (cohesion) çalışır. Bu servisler birbirinden bağımsız olarak dağıtılabilir, ölçeklendirilebilir ve test edilebilir; aralarındaki iletişim ise REST benzeri arayüzler kullanarak ağ aracılığı ile gerçekleştirilir [1], [2].

Mikroservis mimarisinin yazılım endüstrisindeki popülerliği geçtiğimiz yıllarda daha da artmaya başlamıştır. Mikroservis tabanlı mimaride, servislerin birbirinden bağımsız olarak geliştirilebilmesi ve dağıtılabilmesi, bu servislerin sürdürülebilir ve ölçeklendirilebilir olmasını sağlar. Mikroservislerin aslında birer servis olmasına rağmen Servis Odaklı Mimariden temel karakteristik farklılıkları vardır. Bu farklılıklar boyut, sınırlı bağlam (bounded context) ve bağımsızlık olarak sıralanabilir [3]. İlk olarak, mikroservis boyutu bilinen servislere göre daha küçüktür; her bir servis genel olarak sadece bir iş mantığını sağlar. İkinci olarak, bir mikroservis ilgili bütün fonksiyonaliteleri tek bir iş mantığı çerçevesinde toplar. Son olarak ise, her bir mikroservis bağımsız birer servistir. Her bir mikroservisin tek sorumluluğu olduğu için değiştirilmesi için sadece bir neden olmalıdır ve diğer mikroservisler bu değişimden etkilenmeyecek şekilde dizayn edilmelidir.

Servis odaklı mimari ile yazılım geliştiren organizasyonlar mikroservisler ile benzer konseptleri kullandıklarını, DevOps uygulayan organizasyonlar ise otomatikleştirilmiş dağıtıma başladıklarını gözlemleyebilirler. Benzer olarak, çevik yazılım geliştiren organizasyonlar aslında kültürlerini mikroservis dünyasına uydurmaya başladıklarını gösterir [3]. Ancak, bu değişim göründüğü kadar kolay olmamakla birlikte görülmemiş zorlukları da beraberinde getirir. Mikroservis tabanlı mimari ile yazılım geliştirmeye başlayan organizasyonlar bir yandan yazılımda sorumluluğu dağıtma ve özerklik ile diğer yandan da yazılım takımlarının çıktılarını etkili bir şekilde yönetme ve entegre

etmek ile uğraşmaktadırlar. Bunun yanı sıra, geleneksel analiz ve tasarım tekniklerinin mikroservisler ile çalışırken etkili olmadığını görmektedirler. Bunun en önemli nedeni ise, en temel soyutlama kavramı olan "nesne"nin bile değişmek zorunda kalabilmesidir.

Literatürde, geleneksel tekniklerin yanı sıra mikroservis tabanlı mimari için birçok farklı modelleme notasyonu bulunmaktadır. Ancak, bu modeller incelendiğinde sistematik bir metodoloji sağlamadıkları ve dolayısıyla bu modelleme tekniklerinin anlaşılması güç olduğu görülmektedir. Ancak, kullanılan modelleme tekniklerinin anlaşılabilir olması organizasyonlar için büyük önem taşımaktadır [4]. 14. Ulusal Yazılım Mühendisliği Sempozyumu'nda (UYMS 2020) sunduğumuz bildirimizde Türkiye'deki organizasyonların bu zorluklar karşısındaki uyguladıkları yöntemleri ortaya çıkarmak amacıyla düzenlediğimiz anketin sonuçları gösterilmiştir [5].

Çalışmanın başlarında, mikroservis tabanlı mimaride analiz ve tasarım için başvurulan yöntemler ile ilgili sistematik literatür taraması yapılması amaçlanmıştır. Ancak, bu konuda literatürde çalışmalarda eksiklik olduğu görülmüştür ve metodoloji organizasyonların mikroservis tabanlı yazılım geliştirirken izlediği analiz ve tasarım tekniklerini ele alan bir anket çalışması olarak değiştirilmiştir. Çalışmada, Türkiye'deki yazılım organizasyonların mikroservis tabanlı projelerde uyguladığı analiz ve tasarım teknikleri ile ilgili sonuçları ele almak amaçlanmıştır.

Daha önce gerçekleştirdiğimiz çalışmada, mikroservis tabanlı projeler geliştiren organizasyonların geleneksel nesne tabanlı analiz ve tasarım yöntemlerini sıklıkla kullanmaya devam ettikleri görülmüştür [5]. Ancak, mikroservis tabanlı mimarinin sahip olduğu önemli karakteristik özellikler bulunmaktadır ve bu özellikler nesne tabanlı mimariye göre farklılık göstermektedir. Bu nedenle, mikroservis tabanlı projelerde başvurulan analiz ve tasarım yöntemlerinin bu karakteristik özellikleri sağlamaya yönelik yol gösterici olması gerekmektedir. Bu çalışmanın amacı yazılım organizasyonlarının mikroservis tabanlı projelerde başvurduğu analiz ve tasarım yöntemlerini ortaya çıkarmanın yanında mikroservis tabanlı mimarinin karakteristik özellikleri ile ilgili de okuyucuyu bilgilendirmektir.

Bu çalışmanın geri kalan kısımları şu şekilde organize edilmiştir. 2. Bölümde mikroservis tabanlı mimarinin karakteristik özellikleri tartışılmaktadır. Bölüm 3 ilgili çalışmaları, Bölüm 4 araştırma yöntemini vermektedir. Anket sonuçları 5. Bölümde verilmekte ve 6. Bölümde bulgular tartışılmaktadır. Son olarak, 7. Bölümde sonuçlar ve gelecekte planlanan çalışmalar ele alınmaktadır.

2. Mikroservis Mimarisinin Özellikleri

Mikroservis tabanlı mimarının sahip olduğu önemli karakteristik özellikler vardır. Bu özellikler; gevşek bağlılık, yüksek kohezyon, izolasyon, özerklik, birleştirilebilirlik, sınırlı bağlam, asenkron iletişim, tek sorumluluk, ölçeklenebilirlik ve hata toleransı olarak sayılabilir [2], [6]–[8]. Ancak, bu özellikler nesne tabanlı geleneksel mimariye göre farklılık göstermektedir. Çalışmamızın bu bölümünde yukarıda sayılan karakteristik özellikler ile ilgili bilgi vermek amaçlanmıştır.

- *Gevşek bağlılık:* Mikroservis tabanlı mimaride, modüllerin birbirine bağlılığının olabildikçe düşük olması beklenmektedir. Buna bağlı olarak, mimari tasarlanırken her bir iş yeteneğinin farklı bir servis tarafından ele alıncak şekilde ayrıştırılması gerekmektedir. Geleneksel mimaride (nesne tabanlı mimari) gevşek bağlılığın sınıflar arasında sağlanması hedeflenir. Ancak, mikroservis mimarisinde gevşek bağlılığın iş yetenekleri veya alt alanlara göre sağlanması gerekmektedir. Bunu sağlamanın bir diğer yolu ise farklı alt süreçlerin birbirinden ayrıştırılmasıdır. Mikroservislerin diğer önemli karakteristik özelliklerin sağlanabilmesinde mimarinin birbirine gevşek bağlı olan süreç ve iş yeteneklerine göre tasarlanmasının önemi büyüktür.

- *Yüksek kohezyon:* Mikroservis tabanlı mimaride belirli bir süreç ile ilgili olan tüm fonksiyonalitelerin bir arada olması beklenmektedir. Bu sayede, aynı fonksiyonalite ile ilgili elementlerin akışı bir arada sağlanabilir. Geleneksel mimaride ise yüksek kohezyonun sınıflar bazında olması hedeflenir. Bir sınıfa ait metodların fonksiyonel olarak birbirine bağlı olmasına rağmen ilgili süreçe ait tüm fonksiyonalitelerin bir arada tutmayı sağlaması her zaman mümkün olmayabilir. Bu bağlamda, mimari tasarımında ayrıştırmanın iş yetenekleri ve sınırlı bağlama göre yapılması, aynı sürece ait fonksiyonaliteleri bir arada tutmak için önemi büyüktür.

- *İzolasyon:* Mikroservis tabanlı mimari, her bir mikroservisin birbirini etkilemeden ve bağımsız olarak test edilebileceği, sürdürülebileceği ve dağıtılabileceği şekilde izole bir tasarım gerektirmektedir. Bu şekilde izole olarak tasarlanan mikroservisler, özerklik ve hata toleransını sağlamak için bir önkoşuldur. İzole mikroservislerin sağlanabilmesi için ortak veriye bağımlılığın olabildiğince engellenmesi gerekmektedir; dağıtık veri tabanı sistemleri sağlanmalıdır. Her bir mikroservisin kendisine ait veri koleksiyonu olmalıdır. Geleneksel mimaride ise genel olarak paylaşımlı veri yapıları kullanılmaktadır. Dağıtık veri tabanları olay günlüğü (event log) gibi yapılar kullanılarak sağlanılabilir. Olay günlüğü yapısında, her mikroservis kendisine ait veri koleksiyonuna sahip olur. Bu sayede, ortak veriye bağımlılık engellenebilir.

- *Özerklik:* Her bir mikroservisin, izole veritabanına ek olarak, tek bir iş kapasitesine ait tam sorumluluğu alıp özerk bir şekilde çalışması beklenmektedir. Tam sorumluluk ise veri depolama ile birlikte sunum, API istekleri ve ilgili iş mantığını kapsamaktadır. Geleneksel mimaride sistemin bir bütün olarak ayakta durması gerekmektedir. Ancak, mikroservis mimarisinde her bir mikroservis birbirinden bağımsız tasarlanarak farklı kullanıcı veya mikroservisler tarafından bağımsız olarak çalıştırılabilecek şekilde olmalıdır.

- *Birleştirilebilirlik:* Var olan mikroservisler, daha karışık veya yeni iş süreçlerini gerçekleştirebilecek şekilde yeniden birleştirilebilir olmalıdır. Mikroservisler, birbirleriyle işbirliği yaparak ve koordine olarak belirli bir süreci bir bütün olarak yönetmenin yanı sıra, harici mikroservisler ile de entegre olarak farklı kullanım senaryolarını da gerçekleştirebilir. Bu kompozisyonlar, var olan mikroservisler üzerinde ek değişikliklere neden olmamalıdır. Bu nedenle, SOA ilkeleri ve REST mimarisi ile uyum içinde arayüzlere sahip olmalıdırlar. Bu bağlamda, durumsuz (stateless) servislere ihtiyaç

duyulur. Geleneksel mimaride ise buna zıt olarak durumsal bir yapı bulunmaktadır.

- *Sınırlı bağlam:* Mikroservis boyutu bilinen servislere göre daha küçüktür; her bir servis genel olarak sadece bir iş mantığını sağlar. Geleneksel mimaride, sınırlar sınıf yapısına göre belirlenir; sınıf diyagramı sınırlı bağlamı belirlemek için kullanılan bir yoldur. Mikroservis tabanlı mimaride ise, sınırların belirlenmesinde iş mantığı ve ilgili veri önemli rol oynar. Bu nedenle, sınırların belirlenmesinde süreçler göz önünde bulundurulmalıdır.
- *Asenkron iletişim:* Mikroservis tabanlı mimaride, kaynakları daha verimli kullanmak ve daha iyi bir performans için asenkron iletişime ihtiyaç duyulmaktadır. Bu yapıyı sağlamak için en çok kullanılan yapılardan biri ise mesaj kuyruğudur. Bunu sağlayabilmek için olay tabanlı bir tasarıma ihtiyaç duyulmaktadır. Bu sayede, bir mikroservis bir olay tarafından tetiklenebilir ve görevini tamamladıktan sonra yeni bir olay oluşturabilir. Bu olaylar, mesaj kuyruklarında yığınlanarak işleme alınabilir. Geleneksel tasarım yöntemleri olay tabanlı olmadığı için bu yapıyı oluşturmada bir strateji sağlayamamaktadır.
- *Tek sorumluluk:* Mikroservis tabanlı mimaride her bir mikroservis tek bir iş sürecini gerçekleştirecek şekilde ayrıştırılmalıdır. Mikroservislerin bu özelliği geleneksel mimari ile benzerdir. Geleneksel mimaride ayrıştırma sınıflar bazında gösterilir. Her bir sınıf tek bir iş sürecini gerçekleştirecek şekilde tasarlanır. Ancak, bu kompozisyon olay tabanlı olmak yerine nesne tabanlıdır. Mikroservis tabanlı mimaride ise nesne kavramı anlamını yitirmekte; yerine süreç kavramı önem kazanmaktadır. Bu nedenle, mikroservisler tek bir iş sürecini bir bütün olarak ele alacak şekilde tasarlanmalıdır.
- *Ölçeklenebilirlik:* Mikroservis tabanlı mimarinin en önemli özelliklerinden biri de sistemin ölçeklenebilir olmasıdır. Geleneksel mimariden farklı olarak her bir iş süreci ile ilgili mikroservisin kapasitesi anlık olarak arttırılabilir veya düşürülebilir olmalıdır. Bunu sağlamak için her bir iş süreci birbirinden bağımsız olarak dağıtılabilmelidir. Bu nedenle, nesne tabanlı tasarımın ötesinde süreçlere bağlı bir tasarım gerekmektedir.
- *Hata toleransı:* Mikroservis tabanlı mimarinin sağladığı gevşek bağlılık, izolasyon ve özerklik özellikleri monolit ve servis odaklı mimarilere kıyasla daha fazla hata toleransı sağlar. Her mikroservis diğer mikroservislerin yokluğunda da çalışmaya devam edebilmelidir. Ancak, bu senaryoda herhangi bir veri kaybı olmamalıdır. Geri gelen mikroservis, verileri bir yerden alarak çalışmaya devam etmelidir. Geleneksel mimaride, sınıflar farklı alt-süreçler tarafından kullanılabileceğinden paylaşımlı metodlar ve veri tabanları bağlı olarak süreçler arası bağımlılık oluşturabilmektedir. Servis odaklı mimaride ise süreçlerin farklı servisler tarafından yönetilmesine rağmen bu servislerin ortak bir veri tabanına bağlı olmasından dolayı süreçler arası bağımlılık devam etmektedir. Ancak, mikroservis tabanlı mimaride her bir süreç bir mikroservis tarafından yönetilecek şekilde tasarlanmalıdır.

Mikroservis tabanlı mimarinin sağladığı özellikler geleneksel nesne tabanlı mimariye göre farklılık göstermektedir. Mikroservis tabanlı mimaride kullanılan analiz ve tasarım yöntemlerinin bu önemli özellikleri sağlaması önem göstermektedir. Bu nedenle, geleneksel analiz ve tasarım yöntemlerinin mikroservis tabanlı mimariler ile çalışırken etkili olmadığı görülmektedir. Literatürde farklı modelleme tekniklerinin mikroservislerde kullanıldığı görülmektedir. Ancak, bu tekniklerin anlaşılabilir olması önem göstermektedir.

3. İlgili Çalışmalar

Literatürdeki çalışmalar incelendiğinde sektördeki organizasyonların mikroservis tabanlı projelerin analiz ve tasarım aşamalarını nasıl gerçekleştirdiğini ele alan bir araştırma olmadığı görülmektedir. Ancak, mikroservis alanında farklı konuları ele alan sistematik literatür taramaları bulunmaktadır. Pahl ve Jamshidi [9] tarafından yapılan çalışmada mikroservis ile ilgili 33 çalışma metodoloji, mimari ve platform desteği başlıklarında kategorize edilmiştir. Bir diğer çalışmada, Di Francesco ve diğerleri [10] mikroservis mimarisindeki çalışmaları yayın trendi, araştırma odağı ve endüstriyel adaptasyon potansiyeli başlıkları ile ele almıştır. Bu çalışma, mikroservisler ile ilgili yayınların 2014 yılında başladığını ve 2015 yılında yayın sayısının en fazla olduğunu göstermiştir. Alshuqayran ve diğerleri [11] tarafından yapılan çalışmada ise mikroservis ile ilgili çalışmalar mimari zorluklar, kalite nitelikleri ve mimari diyagramlar yönünden ele alınmıştır. Çalışma sonuçları mikroservis mimarisi için en çok kullanılan diyagramların bileşen (component), sıralama (sequence), süreç (process), dağılım (deployment), sınıf (class) ve kullanım durumu (use case) diyagramları olduğunu göstermektedir. Son olarak, Taibi ve diğerleri [12] tarafından yapılan çalışmada ise mikroservis tabanlı mimari prensip ve kalıpları ele alınmış ve bu kalıplar avantaj ve dezavantajları ile birlikte tartışılmıştır.

4. Araştırma Yöntemi

Bu çalışmada, organizasyonlarım mikroservis tabanlı projelerde uyguladıkları analiz ve tasarım yöntemleri hakkında bilgi edinebilmek amacıyla anket metodolojisine başvurulmuştur. Anket, belli kişilerin ya da grupların bir konu üzerindeki deneyimlerini kısa sürede anlayabilmek için sıklıkla başvurulan bir yöntemdir. Birçok farklı alanda başvurulduğu gibi, yazılım mühendisliği alanında da çeşitli konularda örnekleri bulunmaktadır [13]–[15]. Bu amaçla, anonim çevrimiçi anket yapılmasına karar verilmiştir. Bu seçiminin arkasında 3 ana neden bulunmaktadır. İlk olarak, anketin çevrimiçi olarak uygulanmasının nedeni bu sayede daha fazla katılımcıya ulaşabilmektir. İkinci olarak, anketin anonim olması ile katılımcıların şirket bilgilerini paylaşmadan tecrübelerini paylaşmaları sağlanmıştır. Son olarak ise elde edilen verinin daha kolay kategorize ve analiz edilmesini sağlanmıştır. Anket Google Forms uygulaması aracılığı ile uygulanmıştır.

Anketlerde olan ana problemlerden biri de anketin belirsiz sorular içerebilmesidir [16]. Olası belirsizlikleri kaldırmak amacı ile öncelikle bir pilot anket çalışması uygulanmış ve tamamlama süresi ile birlikte geri dönüşler alınmıştır. Bu uygulamadan sonra sorularda bazı düzeltmeler yapılmış ve anket bu düzeltmelerden sonra İzmir Yüksek Teknoloji Enstitüsü Etik Komitesi'nden onay alınarak uygulamaya başlanmıştır.

Anketin hedef grubu mikroservis tabanlı mimaride tecrübesi olan yazılım alanında çalışan uzmanlar olup bu

katılımcılara ulaşmak için çeşitli e-posta grupları, LinkedIn ve araştırmacıların çevresindeki bu konuda deneyimli uzmanlar ile anket paylaşılmıştır.

4.1. Amaç ve Araştırma Soruları

Bu çalışmada, organizasyonların mikroservis tabanlı projelerde başvurduğu analiz ve tasarım yöntemlerini belirlemek amacı ile araştırma soruları aşağıdaki şekilde belirlenmiştir:

- Organizasyonlar mikroservis tabanlı proje geliştirmede hangi analiz tekniklerini kullanmaktadır?
- Organizasyonlar mikroservis tabanlı proje geliştirmede hangi tasarım tekniklerini kullanmaktadır?

Tablo 1: Analiz ve tasarım soruları

Bölüm	Soru	Soru Tipi	
		Tek Yanıt	Çok Yanıt
5	Yazılım analizi için standart bir süreç kullanılıyor mu?	✓	
5	Şu anki (eğer bittiyse en son) mikroservis tabanlı projenizde fonksiyonel gereksinimlerini nasıl gösterdiniz?		✓
5	Şu anki (eğer bittiyse en son) mikroservis tabanlı projenizde problemi nasıl analiz ettiniz?		✓
5	Şu anki (eğer bittiyse en son) mikroservis tabanlı projenizde analiz için hangi notasyonları kullandınız?		✓
5	Şu anki (eğer bittiyse en son) mikroservis tabanlı projenizde problemi nasıl mikroservislere ayrıştırdınız?	✓	
6	Şu anki (eğer bittiyse en son) mikroservis tabanlı projenizde tasarımı göstermek için hangi notasyonları kullandınız?		✓
6	Şu anki (eğer bittiyse en son) mikroservis tabanlı projenizde hangi tasarım kalıplarını (pattern) kullandınız?		✓

4.2. Anket Tasarımı

Anket, mikroservis mimarisi ile ilgili 6 bölümden oluşmaktadır. İlk bölüm anket ile ilgili bilgi vermekte ve katılımcının onayını almaktadır. İkinci bölüm katılımcılar ve çalıştıkları organizasyonlar hakkında bilgi alma amaçlı demografik sorular içermektedir. Bu bölümün son sorusu katılımcının mikroservis tecrübesi olup olmadığını sormaktadır. Katılımcı bu soruda tecrübesi olmadığını belirttiği taktirde anket sona ermektedir. Üçüncü bölümde katılımcının mikroservis tecrübesi ile ilgili genel sorular sorulmaktadır. Dördüncü bölüm yazılım ölçümü ve efor kestirimi ile ilgili sorular içermektedir. Beşinci bölümde yazılım analizi ve altıncı bölümde is yazılım tasarımı ile ilgili sorular yer almaktadır. Bu çalışmada mikroservis tabanlı projelerde yazılım analizi ve tasarımı bölümleri ile ilgili sonuçlar ele alınacak olup bu iki bölümde yer alan sorular Tablo 1'de gösterilmektedir.

4.3. Doğrulama Kriterleri

Anketin hedef grubu mikroservis tecrübesi olan katılımcılar olduğu için mikroservis tabanlı proje geliştirmemiş katılımcıların cevapları sonuçlara yansımamalıdır. Bu nedenle, ikinci bölümde yer alan yedinci soruda katılımcının mikroservis tecrübesi olup olmadığı sorulmuştur. Bu soruya olumsuz cevap veren katılımcılarda anket sonlandırılmış ve diğer bölümlerdeki soruları yanıtlaması engellenmiştir.

5. Anket Sonuçları

Bu anket çalışmasına yazılım sektöründe çalışan 46 kişi katılmıştır, ancak bu katılımcıların 29'unun mikroservis ile yazılım geliştirme deneyimi bulunmaktadır ve bu başlık altında sadece deneyimi olan 29 katılımcının sonuçları göz önünde bulundurulacaktır.

Katılımcılarımızın lisans eğitimleri; Bilgisayar Mühendisliği, Yazılım Mühendisliği, Bilgisayar Bilimi, Matematik Mühendisliği, Elektrik ve Elektronik Mühendisliği ve Deniz Ulaştırma Mühendisliği, bölümlerinde iken, yüksek lisans eğitimleri; Bilgisayar Mühendisliği, Bilgisayar Bilimi, Bilgi Sistemleri, Elektrik ve Elektronik Mühendisliği ve Endüstri Mühendisliği bölümlerindedir. Katılımcılarımızın organizasyondaki rolleri ise; Yazılım Geliştirici (Developer), Kıdemli Yazılım Geliştirici (Senior Developer), Yazılım Geliştirme Yöneticisi, Yazılım Mimarı, Proje Yöneticisi ve Teknolojiden Sorumlu Yönetici (CTO)'dur. Katılımcılarımızın mevcut rollerindeki deneyimleri 1 yıldan daha az süre ile 15 yıl arasında değişmektedir ve ortalaması 4 yıldır. Yazılım sektöründeki deneyimleri ise 1 yıl ile 30 yıl arasında değişmektedir ve ortalaması 8 yıldır. Katılımcıların mikroservisler ile olan deneyimleri ise 1 yıl ile 10 yıl arasında değişmektedir ve ortalamaları 2,5 yıldır.

Katılımcıların çalıştıkları son mikroservis tabanlı projelerin bağlı olduğu sektörler; Otomasyon, Müşteri İlişkileri Yönetimi, E-ticaret, İnternet, Lojistik, Finans, Milli Yazılım, Mobil Yazılım, Telekom ve Kamu Hizmetleri'dir. Çalıştıkları son mikroservis projelerinin türlerine bakıldığında ise 13 katılımcı için (%44,8) monolitik mimari ile geliştirilmiş bir yazılımım mikroservis mimarisinde yeniden yazılması, 12 katılımcı için (%41,4) yeni bir mikroservis tabanlı proje yazılması, 3 katılımcı için (%10,3) bir sistemin bazı parçalarını monolitik mimaride bir proje ile beraber çalışmak amacıyla mikroservis olarak yazılması, 1 katılımcı için (%3,4) ise var olan bir mikroservis tabanlı sistemin düzenlenmesi olduğu görülmüştür. Mikroservisler ile çalışılırken başvurulan yazılım geliştirme modellerinde ise genellikle Çevik (Agile) yöntemlere başvurulduğu görülmüştür. 15 katılımcı (%51,7) sadece Scrum, 4 katılımcı (%13,8) sadece Kanban, 3 katılımcı (%10,3) Kanban ile Scrum, 2 katılımcı (%6,9) Kanban ile Ekstrem Programlama (EP), 2 katılımcı (%6,9) Kanban, Scrum ve EP, 2 katılımcı (%6,9) isim olarak bir yazılım geliştirme çerçevesi belirtmeden genel olarak Çevik (Agile) yazılım geliştirme, 1 katılımcı (%3,4) ise Şelale (Waterfall) modelini takip

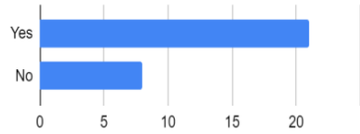
ettiklerini belirtmiştir. Katılımcıların çalıştıkları son mikroservis tabanlı projedeki personel sayısı minimum 2, maksimum 50, ve ortalama 9~ kişidir, organizasyonlarında paralel olarak devam eden mikroservis projesi sayısı ise; minimum 1, maksimum 75, ve ortalama 13'tür. Katılımcılarımız çoğunluğu (58.6%) aynı anda birden fazla mikroservis projesinde çalışmaktadır.

Katılımcıların organizasyonlarının çoğunluğunda (%82,8) DevOps uygulanmaktadır ve Jenkins en çok tercih edilen (%47,9) entegrasyon aracıdır. Mikroservis projelerinin test otomasyonuna bakıldığında ise, katılımcıların çoğunluğu (%37,9) 1-5 arası Likert ölçeğinde 4 olarak oylamıştır. Mikroservis tabanlı mimariler karşılaşılan en kritik problem sorulduğunda ise; katılımcıların çoğunluğu (%41,4) geçiş süreçlerin yönetimini, peşinden ise monolitik tabanlı bir yazılımı parçalamanın zorluğunu (29.3%), asenkron iletişimin zorluğunu (%18,9), veri tabanı ayrımlarının zorluğunu (%6,9), ve sürüm çıkılırken karşılaşılan kompleks adımların zorluğunu (%3,4) belirtmiştir.

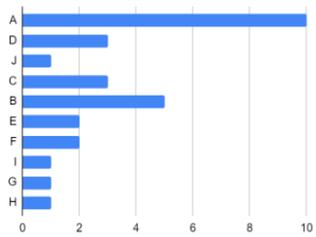
5.1. Yazılım Analizi

Bu başlık altında anketin yazılım analizi ile ilgili olan sonuçları verilmektedir. Verilen cevapları gösteren grafikler o seçeneğe cevap veren katılımcı sayısını göstermektedir.

Bu bölümde katılımcılarımızın cevapladığı ilk soru organizasyonlarında yazılım analizi için standart bir yazılım sürecinin olup olmadığıdır. Sonuçlar organizasyonların çoğunluğunun (%72,4) standart bir yazılım analiz sürecini takip ettiğini göstermektedir (Şekil. 1).



Şekil 1: Yazılım analizi için standart bir süreç kullanılıyor mu?

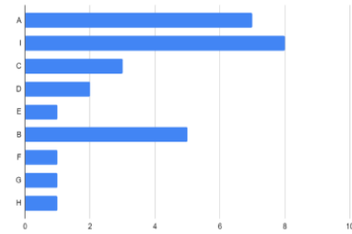


Şekil 2: Şu anki (eğer bittiyse en son) mikroservis tabanlı projenizde fonksiyonel gereksinimlerini nasıl gösterdiniz?

Sonuçlara göre, başka yöntemler ile beraber kullanılsa bile kullanıcı hikayesi yönteminin fonksiyonel gereksinimlerin belirtilmesinde organizasyonlar arasında en tercih edilen yöntem olduğu söylenebilir.

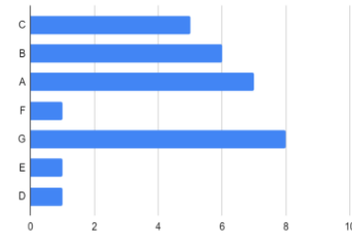
Üçüncü soruda katılımcılara mikroservis tabanlı projelerde problemi nasıl analiz ettikleri sorulmuştur. Sonuçların %24,1'i olay fırtınası (event storming) (A), %17,2'si Ad-hoc (B), %10,3'ü olay tabanlı (event-based) modellemeyi (C), %6,9'u olay fırtınası ve olay tabanlı modellemeyi (D), %3,4'ü Ad-hoc, olay fırtınası, ve olay tabanlı modellemeyi (E), %3,4'ü Ad-hoc ve olay fırtınası (F), %3,4'ü Ad-hoc ve olay tabanlı modellemeyi (G), %3,4'ü olay fırtınası, olay tabanlı model ve İş Süreci Model ve Gösterim (Business Process Model and Notation (BPMN)) (H), %27,6'sı ise hiçbirini (I) seçmiştir (Şekil. 3). Bu soruda hiçbiri seçeneğini seçen katılımcıların bu cevap ile problem analizi yapmadıkları mı yoksa kendileri için başka takımların mı problem analizi yaptığını bilmesek de en yüksek oranı bu cevabın almış olması dikkat çekicidir.

Sonuçlara göre, başka yöntemler ile beraber kullanılsa bile kullanıcı hikayesi yönteminin fonksiyonel gereksinimlerin belirtilmesinde organizasyonlar arasında en çok tercih edilen yöntem olduğu söylenebilir.

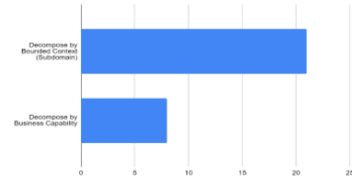


Şekil 3: Şu anki (eğer bittiyse en son) mikroservis tabanlı projenizde problemi nasıl analiz ettiniz?

Dördüncü soruda katılımcıların mikroservis tabanlı projelerin analizinde kullandıkları notasyonlar sorulmuştur. Katılımcıların %24,1'i sadece akış şeması (flow chart) (A), %20,7'si akış şeması ve aktivite diyagramı (B), %17,2'si sadece aktivite diyagramı (C), %3,4'ü akış şeması ve zihin haritası (mindmap) (D), %3,4'ü akış şeması, Aktivite diyagramı, değer akışı (value stream) haritaları ve BPMN (E), %3,4'ü eEPC (F), %27,6'sı ise hiçbir (G) notasyon kullanmadıklarını belirtmiştir (Şekil. 4). Önceki soruda olduğu gibi hiçbiri seçeneğini seçen katılımcıların bir notasyon kullanılmadığını mı yoksa kendilerinin kullanmadıklarını belirttiklerini bilmesek de mikroservis tabanlı projelerin analizleri sırasında bir notasyona ihtiyaç duyulduğunda akış şemasının popüler bir tercih olduğu söylenebilir.



Şekil 4: Şu anki (eğer bittiyse en son) mikroservis tabanlı projenizde analiz için hangi notasyonları kullandınız?



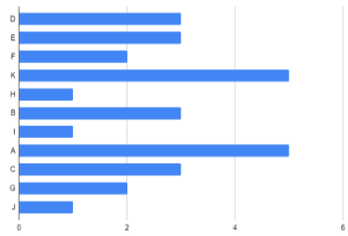
Şekil 5: Şu anki (eğer bittiyse en son) mikroservis tabanlı projenizde problemi nasıl mikroservislere ayrıştırdınız?

Son olarak katılımcılara problemi mikroservislere nasıl ayrıştırdıkları sorulmuştur. Sonuçlar bize organizasyonların çoğunlukla mikroservisleri sınırlı bağlama (bounded context) göre ayrıştırdıklarını (%72,4) ve bazı organizasyonların ise

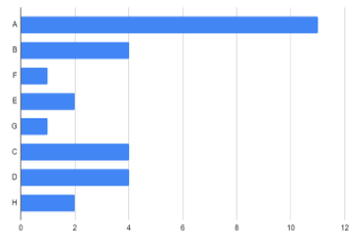
ayrıştırmayı iş yeteneklerine (business capability) göre (%27,6) gerçekleştirdiklerini göstermiştir (Şekil. 5).

5.2. Yazılım Tasarımı

Yazılım tasarımı ile ilgili olarak katılımcılara ilk soruda mikroservis tabanlı projelerinde tasarımı göstermek için hangi notasyonları kullandıkları sorulmuştur. Sonuçların %17,2'si ardıl işlem (sequence) diyagramı (A), %10,3'ü sınıf diyagramı (B), %10,3'ü ardıl işlem ve sınıf (class) diyagramı (C), %10,3'ü sınıf, ardıl işlem ve aktivite (activity) diyagramı (D), %10,3'ü sınıf, ardıl işlem, aktivite ve varlık ilişki modeli (Entity Relationship Diagram (ERD)) (E), %6,9'u ardıl işlem ve aktivite diyagramı (F), %6,9'u sınıf diyagramı ve ERD (G), %3,4'ü aktivite diyagramı (H), %3,4'ü ERD (I), %3,4'ü sınıf, ardıl işlem diyagramları ve ERD (J), %17,2'si ise hiçbirinin (K) kullanılmadığını göstermiştir (Şekil. 6). Bu cevaplardan organizasyonlar arasında yazılım tasarımı için bir diyagrama ihtiyaç duyulduğunda ardıl işlem ve sınıf diyagramlarının tercih edilen bir yansıtma sistemi olduğu çıkarılabilir.



Şekil 6: Şu anki (eğer bittiyse en son) mikroservis tabanlı projenizde tasarımı göstermek için hangi notasyonları kullandınız?



Şekil 7: Şu anki (eğer bittiyse en son) mikroservis tabanlı projenizde hangi tasarım kalıplarını (pattern) kullandınız?

İkinci soruda ise, katılımcılara son mikroservis tabanlı projelerinde kullandıkları tasarım kalıpları (design pattern) sorulmuştur. Sonuçların %27,6'sı devre kesici (circuit breaker) (A), %24,1'i olay kaynağını belirleme düzeni (event sourcing) (B), %13,8'i komut ve sorgu sorumluluğu ayrımı (Command Query Responsibility Segregation (CQRS)) (C), %13,8'i devre kesici, olay kaynağını belirleme düzeni ve CQRS (D), %6,9'u olay kaynağını belirleme düzeni ve CQRS (E), %3,4'ü devre kesici ve bölme perdesi düzeni (bulkhead) (F), %3,4'ü API ağ geçidi (gateway), servis paylaşımlı veri tabanı (shared database per service), log birleştirme (aggregation), performans metrikleri, dağıtılmış izleme (distributed tracing) (G), %6,9'u ise hiçbirini (H) kullanmadıklarını göstermiştir (Şekil. 7). Bu sonuçlar devre kesici modelinin mikroservis tabanlı mimariler sıkça tercih edilen bir model olduğunu, peşinden de olay kaynağını belirleme düzeni modelinin geldiğini göstermiştir.

6. Tartışma

Bu çalışmada anketimizin Türkiye sonuçları verilmiştir. Anket sonuçlarına göre, katılımcıların mikroservis tabanlı projelerde analiz ve tasarım için genel olarak başvurdukları bir yaklaşım olmadığını göstermektedir.

Katılımcıların çoğunun çalıştığı organizasyonlarda yazılım analizinde standart bir süreç kullandıklarını belirtmişlerdir. Ancak, bu süreçte başvurdukları yol değişkenlik göstermektedir. Kullanıcı hikayelerinin bu konuda en çok başvurulan yöntem olduğu ve bunun yanında doğal dil ve kullanım durumu (use case) senaryolarının da sıklıkla kullanıldığı görülmektedir. Bu sonuç, nesne tabanlı analiz yöntemlerinin mikroservis tabanlı projelerde de fonksiyonel gereksinimlerin belirlenmesinde kullanıldığını göstermektedir. Katılımcıların mikroservis tabanlı projelerde başvurdukları problem analiz yöntemine bakıldığında ise olay fırtınası (event storming) yönteminin en çok ve ad-hoc path ile olay tabanlı (event-based) modellemenin de sıklıkla kullanıldığı görülmektedir. Problem analizinde olay tabanlı yöntemlerin sıklıkla kullanılmasına rağmen fonksiyonel gereksinimlerin çıkarılmasında nesne tabanlı yöntemlere başvurulması dikkat çekmektedir. Analiz sürecinde bu organizasyonların nesne tabanlı yöntemler ile başlayıp sonrasında ise olay tabanlı modelleme ile devam etmeleri ve bu geçişi nasıl sağladıkları incelenmesi gereken bir konudur. Analiz için kullanılan notasyonlarda ise yine nesne tabanlı yöntemler (akış şeması ve aktivite diyagramı) kullanıldığı göze çarpmaktadır. Sadece bir katılımcı olay tabanlı modelleme şekli olan eEPC kullandığını belirtmiştir. Analiz konusunda son olarak katılımcıların problemi mikroservislere nasıl ayrıştırdıkları sorulduğunda büyük çoğunluğun sınırlı bağlama (bounded context) göre gerçekleştirdikleri görülmektedir. Bu katılımcıların problemi olay bazlı yöntemlere başvurarak analiz ederken geleneksel olan nesne tabanlı notasyonlar kullanarak göstermesi sonucunda sınırlı bağlamı nasıl sağladıkları ele alınabilecek bir başka konudur.

Mikroservis tabanlı projelerde tasarım nesne tabanlı projelere göre farklılık göstermektedir. Bunun en önemli nedeni, mikroservislerde sınırlı bağlamın nesneler ile sağlanmamasıdır. Sonuçlara bakıldığında, nesne tabanlı tasarımın en önemli gösterim şekillerinden biri olan sınıf diyagramlarının mikroservis tabanlı projelerde de sıklıkla kullanıldığı görülmektedir. Ancak, mikroservis mimarisinde sınıf kavramı ve bu sınıfların arasındaki ilişki artık önemini yitirmekte ve farklı gösterim şekillerine ihtiyaç duyulmaktadır. Kullanılan tasarım kalıplarına bakıldığında ise yine olay tabanlı bir kalıp olan olay kaynağını belirleme düzeni (event sourcing) ikinci sırada görülmektedir. Bunun yanında kapalı devre (circuit breaker) en çok kullanılan tasarım kalıbıdır. Analiz ile paralel olarak, tasarım sürecinde de yine nesne tabanlı ve olay tabanlı yöntemlerin birlikte kullanıldığı görülmektedir. Ancak, nesne tabanlı gösterimden olay tabanlı tasarım kalıplarına nasıl geçiş yapıldığı yine incelenmesi gereken bir konudur.

Sonuç olarak, anketin Türkiye sonuçları organizasyonların, mikroservis tabanlı projelerde bir süreç izlediği ancak bu sürecin organizasyonlara göre farklılaştığı görülmektedir. Ancak, nesne tabanlı mimaride kullanılan yöntemler gibi kalıplaşmış bir yöntem bulunmamaktadır. Nesne tabanlı ve olay tabanlı yöntemler ile farklı kombinasyonlar kullanıldığı görülmektedir.

Mikroservis tabanlı projelerde başvurulan analiz ve tasarım yöntemlerinin mikroservislerin sahip olduğu

karakteristik özellikleri sağlayacak şekilde yol göstermesi gerekmektedir. Bu özellikler ise nesne tabanlı mimariye göre farklılık göstermektedir. Bu nedenle, nesne tabanlı analiz ve tasarım yöntemlerinin mikroservis tabanlı projelerin özelliklerini ne şekilde sağladığı bir soru işareti olarak kalmaktadır.

Literatür ile sektörü karşılaştırdığımızda; Alshuqayran ve diğerleri [11] tarafından yapılan çalışmada, mikroservis mimarisi için en çok kullanılan diyagramların bileşen (component), ardıl işlem (sequence), süreç (process), dağılım (deployment), sınıf (class) ve kullanım durumu (use case) diyagramları olduğu belirtilmektedir. Anket sonuçları da sektörde bu gösterimlerin sıklıkla kullanıldığını, nesne tabanlı yöntemlere yine sıklıkla başvurulduğunu göstermekte ve literatür ile paralel bir sonuç göstermektedir.

7. Sonuç

Mikroservis tabanlı mimari, yazılım geliştirmede popüler ve etkili bir yol olarak değerlendirilmektedir. Ancak, mikroservis yeni bir konsept olup yazılım organizasyonlarının kültürlerini değiştirmelerini gerektirmektedir. Yazılım analizi ve tasarımı teknikleri mikroservis tabanlı projelerde farklılaşmaktadır. Literatürde, mikroservis tabanlı projelerde analiz ve tasarım süreçlerini konu alan çalışmalarda eksiklik görülmektedir.

Bu çalışma kapsamında, yazılım sektöründeki organizasyonların mikroservis tabanlı proje geliştirirken başvurduğu analiz ve tasarım tekniklerini analiz etmek amaçlı bir anket düzenlenmiş ve bu anketin sonuçları organizasyonların bu konudaki tecrübelerini ortaya çıkarmak amacı ile sunulmuştur. Ankete Türkiye'den 46 kişi katılmıştır. Sonuçlar, ankete katılan organizasyonlarda mikroservis tabanlı projelerin analiz ve tasarımı için genel olarak benimsenen tutarlı bir yaklaşım olmadığını göstermektedir. Ankete katılan organizasyonlarda geleneksel nesne tabanlı analiz ve tasarım yöntemlerinin mikroservis tabanlı projelerin geliştirilmesinde de ağırlıklı olarak kullanıldığı görülmektedir. Ancak, geleneksel yöntemlerin mikroservis tabanlı projelerin analiz ve tasarım sürecinde ne derecede etkili olduğu tartışılması gereken önemli bir konudur.

Bu çalışmada, sektördeki organizasyonların mikroservis tabanlı projeler için analiz ve tasarım konusundaki bakış açılarını ortaya çıkarmak amaçlanmıştır. Anket sonuçlarının istatistiksel analiz yöntemleri kullanılarak değerlendirilmesi ve sonuçlar arasındaki korelasyonun incelenmesi planlanmaktadır.

8. Kaynaklar

[1] A. R. Sampaio *et al.*, "Supporting Microservice Evolution," in *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Sep. 2017, pp. 539–543, doi: 10.1109/ICSME.2017.63.

[2] J. Thönes, "Microservices," *IEEE Softw.*, vol. 32, no. 1, Art. no. 1, Jan. 2015, doi: 10.1109/MS.2015.11.

[3] N. Dragoni *et al.*, "Microservices: Yesterday, Today, and Tomorrow," in *Present and Ulterior Software Engineering*, M. Mazzara and B. Meyer, Eds. Cham: Springer International Publishing, 2017, pp. 195–216.

[4] A. Dikici, O. Turetken, and O. Demirors, "Factors influencing the understandability of process models: A systematic literature review," *Inf. Softw. Technol.*, vol. 93, pp. 112–129, Jan. 2018, doi: 10.1016/j.infsof.2017.09.001.

[5] B. Bilgin, H. Ünlü, and O. Demirors, "Analysis and Design of Microservices: Results from Turkey," presented at the 14th Turkish National Symposium on Software Engineering (Ulusal Yazılım Mühendisliği Sempozyumu, UYMS), Turkish National Symposium on Software Engineering (Ulusal Yazılım Mühendisliği Sempozyumu, UYMS), Oct. 2020.

[6] J. Bonér, *Reactive Microservices Architecture*. O'Reilly Media, Inc., 2016.

[7] J. Bonér, *Reactive Microsystems*. O'Reilly Media, Inc., 2017.

[8] I. Nadareishvili, R. Mitra, M. McLarty, and M. Amundsen, *Microservice Architecture: Aligning Principles, Practices, and Culture*. O'Reilly Media, Inc., 2016.

[9] C. Pahl and P. Jamshidi, "Microservices: A Systematic Mapping Study.," in *CLOSER (1)*, 2016, pp. 137–146.

[10] P. Di Francesco, P. Lago, and I. Malavolta, "Architecting with microservices: A systematic mapping study," *J. Syst. Softw.*, vol. 150, pp. 77–97, Apr. 2019, doi: 10.1016/j.jss.2019.01.001.

[11] N. Alshuqayran, N. Ali, and R. Evans, "A Systematic Mapping Study in Microservice Architecture," in *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, Nov. 2016, pp. 44–51, doi: 10.1109/SOCA.2016.15.

[12] D. Taibi, V. Lenarduzzi, and C. Pahl, "Architectural patterns for microservices: a systematic mapping study," SCITEPRESS, 2018.

[13] V. Garousi, A. Coşkunçay, A. Betin-Can, and O. Demirörs, "A survey of software engineering practices in Turkey," *J. Syst. Softw.*, vol. 108, pp. 148–177, Oct. 2015, doi: 10.1016/j.jss.2015.06.036.

[14] V. Garousi, A. Coşkunçay, and O. Demirörs, "A survey of software testing practices in Turkey."

[15] D. Akdur, V. Garousi, and O. Demirörs, "A survey on modeling and model-driven engineering practices in the embedded software industry," *J. Syst. Archit.*, vol. 91, pp. 62–82, Nov. 2018, doi: 10.1016/j.sysarc.2018.09.007.

[16] F. Shull, J. Singer, and D. I. Sjøberg, *Guide to advanced empirical software engineering*. Springer, 2007.

Özgeçmişler

Hüseyin Ünlü lisans derecesini (2016) Orta Doğu Teknik Üniversitesi (ODTÜ) Kuzey Kıbrıs Kampusu Bilgisayar Mühendisliği programından, yüksek lisans derecesini (2019) ise yine aynı üniversitenin Sürdürülebilir Enerji ve Çevre Sistemleri programından aldı. Yüksek lisans eğitimi süresince yine aynı kurumda Bilgisayar Mühendisliği programında öğretim asistanı olarak görev aldı. Doktora eğitimine İzmir Yüksek Teknoloji Enstitüsü (İYTE) Bilgisayar Mühendisliği bölümünde Prof. Dr. Onur Demirörs danışmanlığında devam etmekte ve aynı bölümde araştırma görevlisi olarak çalışmaktadır. Araştırma alanları arasında yazılım büyüklük ölçümü, mikroservis tabanlı mimariler, yazılım süreç geliştirme, yazılım kalite yönetimi konuları bulunmaktadır.

Burak Bilgin lisans derecesini (2017) İzmir Yüksek Teknoloji Enstitüsü (İYTE) Bilgisayar Mühendisliği programından almıştır. Mezuniyet sonrası Türk Ekonomi Bankası (TEB) Bilgi Teknolojisi departmanında yazılım geliştiricisi olarak çalışmaya başlamıştır ve çeşitli web uygulamaları geliştirmelerinde rol almıştır. 2021 yılından itibaren TEB Arf bünyesinde çalışmaya devam etmektedir. Araştırma alanları arasında çecik yazılım geliştirme ve yazılım ölçümü konuları bulunmaktadır.

Prof. Dr. **Onur Demirörs** lisans diplomasını Orta Doğu Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü'nden aldı. 1990'da Ege Üniveristesi, 1993'te ABD'de Southern Methodist Üniversitesi Bilgisayar Mühendiliği Bölümleri'nden yüksek lisans dereceleri elde etti. 1995'te ise yine Southern Methodist Üniversitesi'nde doktora çalışmalarını tamamladı. Bu üniversitede ve sonrasında Dokuz Eylül ve Orta Doğu Teknik Üniversiteleri'nde geçen uzun ve verimli bir akademik deneyimin ardından, 2017 yılında İzmir Yüksek Teknoloji Enstitüsü Bilgisayar Mühendisliği bölümüne katıldı. Araştırma alanları arasında yazılım süreç geliştirme ve süreç modelleri: metodlar, araçlar, yöntemler, iş süreç yönetimi, yazılım kalite yönetimi, yazılım ölçme, yazılım proje yönetimi ve kavramsal modelleme konuları bulunmaktadır.