Bilgisayar Bilimleri ve Mühendisliği Dergisi

Cilt-16
Sayı-1
2023

DevOps İşletim Hattının Kalite Bakış Açısı ile İrdelenmesi ve ISO/IEC 25010 Boşluk Analizinin Gerçekleştirilmesi

Assessment of DevOps Pipeline with Quality Perspective and Application of ISO/IEC 25010 Gap Analysis

Çağrı ŞENKAL
Havelsan A.Ş.
Test Müh. Grup Müdürlüğü
Ankara/ Türkiye
csenkal@havelsan.com.tr
ORCID: 0000-0003-1583-2419

Burak YOLAÇAN
Havelsan A.Ş.
Test Müh. Grup Müdürlüğü
Ankara/ Türkiye
byolacan@havelsan.com.tr
ORCID: 0000-0002-1274-5654

Kadir HERKİLOĞLU
Havelsan A.Ş.
Test Müh. Grup Müdürlüğü
Ankara/ Türkiye
kherkiloglu@havelsan.com.tr
ORCID: 0000-0003-3258-7240

Asım Egemen YILMAZ
Havelsan A.Ş. / Ankara Üniversitesi
Test Müh. Grup Müdürlüğü / Yazılım Müh. Bölümü
Ankara/ Türkiye
(aeyilmaz)@havelsan.com.tr; @eng.ankara.edu.tr
ORCID: 0000-0002-4156-4238

Öz

Bu çalışmada, DevOps işletim hattı bütüncül bir yazılım ürünü olarak ele alınmakta; bugüne değin farklı araştırmacılar tarafından literatürde yayınlanmış olan ve DevOps işletim hattı üzerinde yürütülen süreçlerin başarımının izlenmesine yönelik olarak önerilmiş yazılım geliştirme ve ürün ölçütlerinin, ISO/IEC 25010 kalite bakış açısı ile hangi kavramlarla ilintilenebilir olduğu incelenmekte; bu kapsamda ISO/IEC 25010 kalite çerçevesinde hangi boyutlarda boşluklar kaldığına ve hangi hususlarda çalışmalar ve tanımlamalar yapılmasına ihtiyaç bulunduğu tespit edilmeye çalışılmaktadır.

***Anahtar sözcükler*:** Yazılım geliştirme süreci, DevOps, İşletim hattı, Yazılım kalitesi, Yazılım kalite güvencesi, ISO/IEC 25010

Abstract

In this study, DevOps pipeline is considered as an integrated software product; DevOps pipeline process performance metrics in the literature are examined with the perspective of ISO/IEC 25010. Considering the studies existing in the literature, an analysis is performed in order to identify the dimensions and aspects in which there exists some gaps and therefore, more effort is required.

Keywords: Software development process, DevOps, Pipeline, Software quality, Software quality assurance, ISO/IEC 25010

Gönderme, düzeltme ve kabul tarihi: 26.05.2022 - 25.10.2022 – 09..02.2023
Makale türü: Araştırma

1. Giriş – DevOps

Günümüzde, faaliyet alanından bağımsız olarak hemen her kuruluş günlük operasyonlarını yürütürken yoğun bir şekilde yazılım kullanmaktadır. Mevcut durumda, söz konusu operasyonların sorunsuz bir şekilde yürütülerek sürdürülebilmesi için her zaman güvenilir, güvenli ve kullanılabilir yazılımlara ihtiyaç bulunmaktadır. Dolayısıyla yazılım sektöründeki kuruluşlar ve firmalar için de sürekli olarak yenilikçi uygulama özellikleri içeren ve aynı zamanda yüksek kaliteli yazılım ürünleri sunabilmek, rekabetçilik açısından da çok kritik bir faktör haline gelmiş durumdadır.

Yazılım geliştiren ve süreç olgunluğu yeterince yüksek herhangi bir kuruluş veya firmanın, yeterli zaman ve bütçe sağlanması durumunda mükemmele yakın yazılım uygulamaları ve hizmetleri sağlaması mümkündür. Bu anlamda, bir yazılım ürününün belirli bir kalite ile geliştirilme hızı, yazılım sektöründe başarının anahtarı haline gelmiş

durumdadır. Geliştirme (*Development* - Dev) ve Operasyonlar (*Operations* - Ops) sözcüklerinin bir araya getirilmesiyle oluşturulmuş DevOps kavramı, işte tam da bu anlamda geliştirme ve operasyon ekibini bir araya getirerek müşterilere belirli bir kalitenin üzerinde hızlı ve sürekli/düzenli teslimat yapılabilmesini hedeflemektedir. DevOps terimi, ilk olarak 2008 yılında Çevik Metodolojiler Konferansı'nda Patrick Debois tarafından çevik bir altyapı ihtiyacının yanı sıra geliştirme ve operasyon ekipleri arasındaki etkileşimin öneminin vurgulanması ile birlikte kullanılmıştır [1].

DevOps, yazılım ürün ve hizmetlerini hızlı, güvenilir ve yüksek kaliteli bir şekilde sunmak adına geliştiricilerin ve operasyonların iletişim kurduğu ve işbirliği yaptığı bir dizi faaliyet içeren bir yaklaşımdır. Yazılımın geliştirilmesinden dağıtıma, hatta dağıtım sonrası desteğe kadar tüm döngü boyunca görev ve sorumlulukların tüm ekip içerisinde dağıtılması; işletim hattı (*pipeline*) adı verilen belirli bir teknolojik altyapının da kullanımı sayesinde de tüm faaliyetlerin izlenebilir ve hesap verilebilirliğinin sağlanması esasına dayanmaktadır. Yazılım ürünlerinin boyut ve karmaşıklığının artmasından kaynaklanan yüksek kalite ihtiyaç ve talebinin, çevik metodolojilerin işbirlikçi bir kültür içinde uygulanarak karşılanabileceği fikri ile ortaya çıkmıştır.

Öte yandan, başta sorumluluk paylaşımı gibi hususları zorunlu kılan DevOps kültürünün oluşturulması ve Sürekli Teslimat ve Dağıtım (*Continuous Delivery & Deployment*-CD) gibi uygulamalarının gelenekselleştirilmesi, birçok Yazılım Mühendisliği faaliyetinin yenilikçi teknikler ve araçlar ile gerçekleştirilmesini gerektirmektedir. Ayrıca DevOps'un getirdiği dönüşüm ve yeni bakış açısı, temel iş değerlerinin (*business values*) çoğunlukla geliştirme tarafında gerçekleştirilmesini de zorunlu kılmaktadır. Bu da:

- Sürekli planlama,
- Sürekli entegrasyon ve test,
- Sürekli yayın ve dağıtım,
- Sürekli altyapı izleme ve optimizasyonu,
- İşbirliğine dayalı sürekli geliştirme,
- Sürekli kullanıcı davranışı izleme ve geri bildirimi

gibi DevOps yeteneklerini ön plana çıkarmaktadır. Söz konusu yetenekler gerek süreç, gerekse ara ve nihai ürün/hizmet kalitesine işaret etmekte olup bu yetenekler ancak çok olgun, kararlı ve gürbüz bir DevOps işletim hattı ile sağlanabilmektedir. Bütün bu bilgiler ışığında, Şekil-1'de de işlevsel olarak resmedilmiş olan DevOps işletim hattının Yazılım Kalitesi değerlendirmesi de kaçınılmaz ve kritik bir husus haline gelmektedir. Yazılım Kalitesi, ISO/IEC 9126 2001 standardında "belirtilen veya ima edilen ihtiyaçları karşılayan bir yazılım ürününün tüm özelliklerinin toplamı" olarak tanımlanmıştır [2]. IEEE SA 610.12 1990'a göre Yazılım Kalite Güvencesi, "yazılım geliştirme yaşam döngüsü sürecinin ve ürünlerinin, planlı ve sistematik bir dizi operasyon yoluyla endüstrinin gereksinimlerine, standartlarına ve prosedürlerine uygun olmasını sağlayan" bir kavramdır [3].

Bu çalışmanın ana amacı, DevOps işletim hattının tamamının bütüncül bir yazılım ürünü olarak ele alınması; bugüne değin farklı araştırmacılar tarafından literatürde yayınlanmış olan ve DevOps işletim hattı üzerinde yürütülen süreçlerin başarımının izlenmesine yönelik olarak önerilmiş yazılım geliştirme ve ürün ölçütlerinin, ISO/IEC 25010 kalite bakış açısı ile nereye oturmakta olduğunun incelenmesi; bu kapsamda ISO/IEC 25010 kalite çerçevesinde nerelerde boşluklar kaldığının tespit edilmesidir. Dolayısıyla bir sonraki bölümde, öncelikle söz konusu kalite çerçevesinin kapsam ve içeriği ele alınmaktadır. Üçüncü bölümde ise DevOps'a ve işletim hattına ilişkin mevcut çalışmalarda önerilmiş ölçütlerin ISO/IEC 25010 kalite çerçevesinde hangi başlık veya alt başlık altına bağıntılanabileceği tartışılarak bir boşluk analizi (*gap analysis*) gerçekleştirilmekte; dördüncü ve son bölümde ise söz konusu boşlukların doldurulmasına yönelik öneriler getirilmektedir.

2. ISO/IEC 25010 Modeli

SQuaRE (*Systems and Software Quality Requirements and Evaluation*) modeli olarak da bilinen ISO/IEC 25010 kalite çerçevesi, Şekil-2'de de görüldüğü üzere herhangi bir yazılım ürününün kalitesinin 8 ana başlık altında bulunan toplam 34 farklı bakış açısı ile değerlendirilmesi esasına dayanır [4]. 2021 yılı itibarı ile Türk Standardı olarak da kabul edilmiş olan bu yaklaşım [5], ISO/IEC 9126'nın yerini almıştır.

Bir kuruluşta DevOps süreç başarımının değerlendirilmesine yönelik olarak 3 farklı bakış açısı ile soru sorulması, literatürde sıklıkla karşılaşılan bir yaklaşımdır [6]:

- Geliştirilen temel iş değerinin (*business value*) müşteriye / kullanıcıya aktarımına yönelik bakış
- Geribildirim mekanizmalarına ve bunların ele alınmasına yönelik bakış
- Sürekli öğrenme ve deneysel gelişime yönelik bakış

DevOps süreç başarımına ilişkin olarak bugüne değin literatürde sorgulanması önerilen hususların söz konusu bakış açıları ile ilişkilendirilmiş bir hali, Çizelge-1'de sunulmaktadır. DevOps işlem hattı mimarisinin ve işletiminin başarım değerlendirmesine yönelik olarak literatürdeki çalışmalarda bugüne değin sorgulanması önerilen hususlar ve ölçütler de Çizelge-2'de sunulmaktadır.

3. Boşluk Analizi

Çizelge-1'den de görülebileceği üzere önceki bölümde anılan 3 farklı bakış açısının da ele alınmasına yönelik olarak literatürde yeterli sayıda (ve bakış açıları arasında belirli bir denge oluşacak şekilde) sorgulama ve tanımlama yapılmış olduğu görülmektedir.

Öte yandan, DevOps işlem hattı mimarisinin ve işletiminin başarım değerlendirmesine yönelik olarak yapılmış tanımlar incelendiğinde, Çizelge-2'den görülebileceği üzere bugüne değin İşlevsel Elverişlilik, Güvenilirlik ve Başarım Verimliliği boyutlarında birçok husus irdelenmiş ve ölçüt tanımlanmıştır. Kullanılabilirlik, Güvenlik ve İdame Edilebilirlik boyutlarında henüz yeterince çalışma yapılmamış olduğu, bu boyutlardaki çalışmaların daha olgunlaşması gerektiği görülmektedir. Örneğin Kullanılabilirlik boyutu altında:

- İşletim hattının kullanılabilirliğine dair ölçütlerin ve anahtar performans göstergelerinin tanımlanmış olup olmadığı,

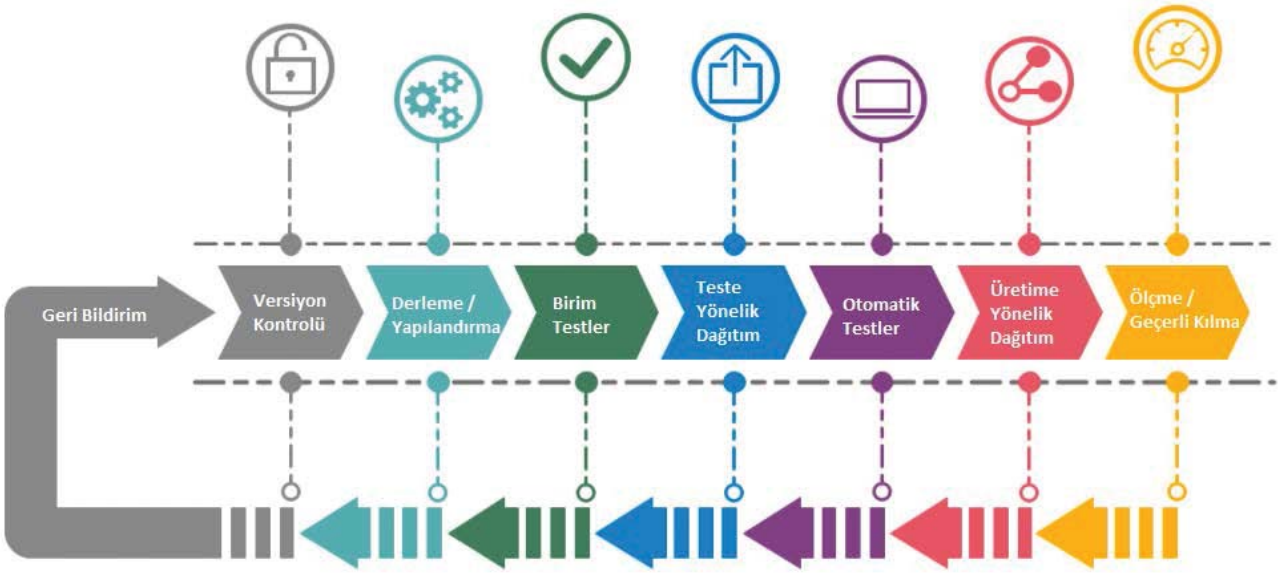
(örneğin veri analitiği, vb.) ilişkin yaklaşım ve yöntemlerin tanımlanmış olup olmadığı gibi yeni soruların da Çizelge-2'ye dahil edilebileceği tarafımızca değerlendirilmektedir.

Uyumluluk ve Taşınabilirlik boyutlarında ise, büyük olasılıkla DevOps işletim hattının söz konusu özellikleri sağlamasının gerekli olmadığı düşüncesiyle herhangi bir araştırmacı grup tarafından bir değerlendirmenin yapılmadığı görülmektedir. Oysa Şekil-2'de de görülebileceği üzere Uyumluluk boyutu altında özellikle Beraber İşletilebilirlik, Taşınabilirlik boyutu altında ise bilhassa Uyumlandırılabilirlik ve Yeri Doldurulabilirlik bakış açıları ile birtakım ölçütlerin geliştirilmesi gerektiği tarafımızca değerlendirilmektedir.

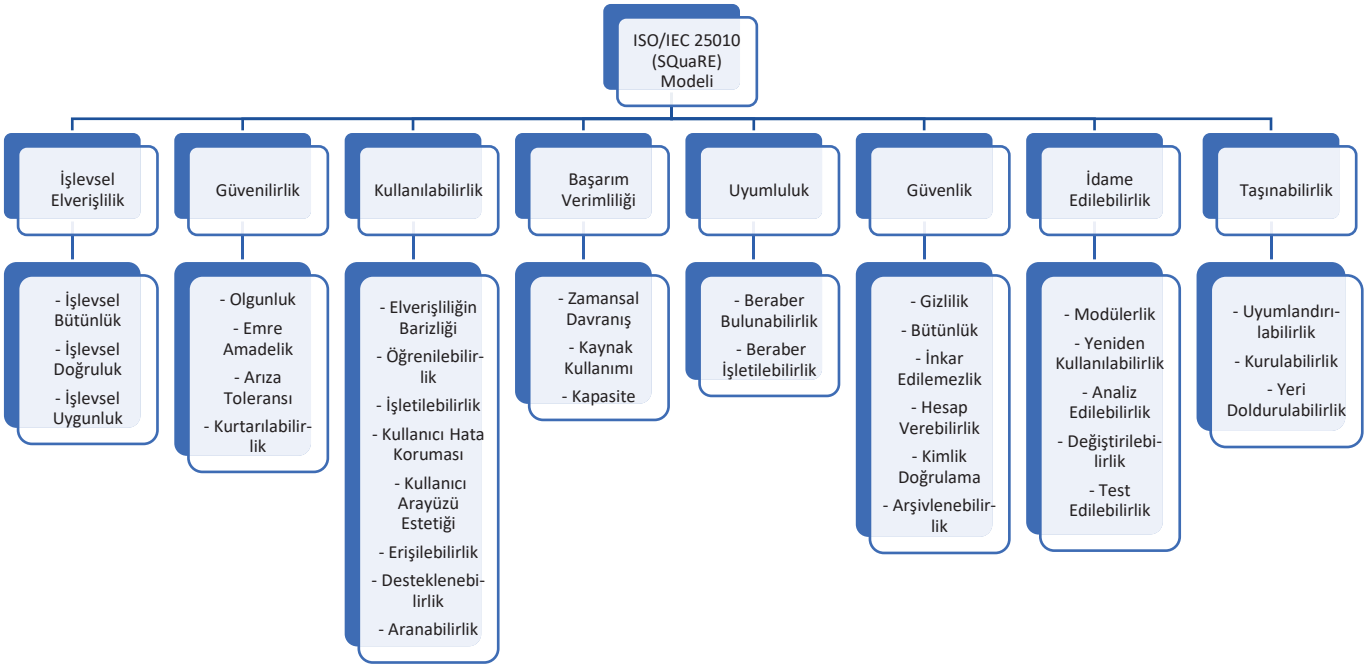
- Söz konusu ölçütlerin ve anahtar performans göstergelerinin ölçümüne (örneğin test, anket, vb.) ve değerlendirilmesine

4. Sonuçlar

Bu çalışmada, DevOps işletim hattı bütüncül bir yazılım ürünü olarak ele alınmış; literatürde bugüne değin yayınlanmış olan ve DevOps işletim hattı üzerinde yürütülen süreçlerin başarımının izlenmesine yönelik olarak önerilmiş yazılım geliştirme ve ürün ölçütlerinin, ISO/IEC 25010 kalite bakış açısı ile hangi kavramlarla örtüşmekte olduğu incelenmiştir. Bu kapsamda ISO/IEC 25010 kalite çerçevesinde hangi boyutlarda boşluklar kaldığına ve hangi hususlarda çalışmalar ve tanımlamalar yapılmasına ihtiyaç bulunduğu tespit edilmiştir. Bölüm 3'te de belirtildiği üzere Kullanılabilirlik, Uyumluluk ve Taşınabilirlik boyutlarının altında yeni tanımlamalar yapılmasına ihtiyaç duyulduğu sonucuna varılmıştır.



Şekil-1: DevOps işletim hattı ([7]'den uyarlanmıştır).



Şekil-2: SQuaRE modeli olarak da bilinen ISO/IEC 25010 Yazılım Kalite Modeli

Çizelge-1: DevOps süreç başarımı değerlendirmesine yönelik sorular / ölçütler

Bakış Açısı (Ana Soru)	**Alt Sorular ve Değerlendirme Ölçütleri**	**Kaynak(lar)**
Geliştirilen temel iş değerinin (*business value*) müşteriye / kullanıcıya aktarımına yönelik bakış	Aylık yayın (*release*) sayısı ne kadardır?	[12]
	Ayda ortalama kaç adet özellik (*feature*) yayınlanmaktadır?	[12]
	Yayın başına maliyet ne kadardır?	[15]
	Bir özelliğe ilişkin ortalama teslim süresi ne kadardır?	[10]
	Bir özelliğe ilişkin ortalama döngü süresi ne kadardır?	[18]
	Ortalama değişiklik gerçekleştirme ve teslim süresi ne kadardır?	[11, 17]
	Minimum özellik sağlama ve teslim süresi ne kadardır?	[12]
	Kullanıcı hikâyesi (*user story*) başına mevcut gelir ne kadardır?	[15]
	Hâlihazırda geliştirilmiş ve test edilmiş ancak henüz üretime dağıtılmamış olan en eski özellik ne kadar eskidir?	[12]
	Tespit edilen bir hatanın düzeltilmesi için geçen ortalama süre ne kadardır?	[17]
	Hatalı veya başarısız değişiklik/düzeltme (*change/correction failure*) oranı ne kadardır?	[17]
	Ortalama DevOps rozet boyutu (*badge size*) ne kadardır?	[10]
	Takımlar/ekipler, ürün veya KPI (*Key Performance Indicator* – Anahtar Performans Göstergesi) odaklı olarak mı oluşturulmuştur?	[14]
Geribildirim mekanizmalarına ve bunların ele alınmasına yönelik bakış	Bir hatanın tespit edilebilmesi için gerekli ortalama süre ne kadardır?	[10]
	Geliştirme ekibinin, hata ayıklamaya (*debug*) yönelik olarak üretim sisteminin günlüklerine (*logs*) ve yığın izlerine (*stack traces*) erişimi bulunmakta mıdır?	[11]
	Her bir özelliğin yayınlanmasından sonra beliren ortalama husus/olay sayısı ne kadardır?	[18]
	İşlem hattının güvenilirliği (*reliability*), emre amadeliği (*availability*) ve başarımına ilişkin ölçütler tanımlı mıdır?	[11]
	İşlem hattı izleme sisteminin, oluşan hatalar hakkında ilgili kişileri uyarma mekanizması bulunmakta mıdır?	[11]
	Kurumdaki herkesin, işlem hattının tüm bileşenlerinin görselleştirilmiş geri bildirimlerine ve ölçümlerine erişimi bulunmakta mıdır?	[9, 11, 16]
	Müşteri temas noktaları (*touch points*) arasındaki ortalama ve maksimum süreler ne kadardır?	[18]
Sürekli öğrenme ve deneysel gelişime yönelik bakış	Kurum bazında ortak bir anlayış oluşturmak adına kod gözden geçirme faaliyetleri gerçekleştirilmekte midir?	[11]
	Bir özelliğin yayınlanmasının ardından bozup yeniden yapma (*rework*) faaliyetleri için ne kadar zaman harcanmaktadır?	[18]
	Süreç kapsamında edinilen yeni bir bilginin depolanarak içselleştirilmesi için ne kadar zaman harcanmaktadır?	[18]

	Süreç kapsamında önceden edinilerek depolanmış olan bir bilginin kullanımı için ne kadar zaman harcanmaktadır?	[18]
	Her bir koşu (*sprint*) sonrasında çalışmaların yansıtılması için ne kadar zaman harcanmaktadır?	[18]
	Çalışma süreçlerinin yansıtılması için ne kadar zaman harcanmaktadır?	[16, 18]
	Deneysel amaçlı kasıtlı olarak oluşturulup enjekte edilen hataların yüzde kaçı tespit edilmektedir?	[18]
	Ekip/takım sınırlarının ötesinde işbirliğine yönelik bir kültür bulunmakta mıdır?	[9, 16]

Çizelge-2: DevOps işlem hattı mimarisinin ve işletiminin başarım değerlendirmesine yönelik sorular / ölçütler

Bakış Açısı (Soru)	**İlintili ISO/IEC 25010 Ölçütü**
İşlem hattı, tam otomatik dağıtımı (*fully automated deployment*) ne ölçüde desteklemektedir?	İşlevsel Elverişlilik [11]
Her bir eser (*artifact*) etiketlenip uygun bir şekilde yönetilmekte midir?	İşlevsel Elverişlilik [9, 13, 16]
Uygun araçlarla statik kod ölçümlemeleri gerçekleştirilmekte midir?	İşlevsel Elverişlilik [9, 11, 13, 16]
İşlem hattı mimarisi, dağıtım geri alma (*deployment roll-back*) işlemine izin vermekte midir?	İşlevsel Elverişlilik [13]
İşlem hattı mimarisi, kesintisiz dağıtım (*zero-downtime deployment*) işlemini desteklemekte midir?	İşlevsel Elverişlilik [8, 9, 11, 16]
İşlem hattı mimarisi, dağıtımlarla sürümler arasında bağlantının kesilebilmesini desteklemekte midir?	İşlevsel Elverişlilik [9, 10, 16]
Hatalı veya başarısız değişiklik/düzeltme (*change/correction failure*) oranı ne kadardır?	Güvenilirlik [17]
Veri tabanı, denetlenebilir bir şekilde otomatik olarak versiyonlanabilmekte midir?	Güvenilirlik [17]
Otomatik testler, yapı seviyesinde (*build level*) ve sürekli bir şekilde gerçekleştirilebilmekte midir?	Güvenilirlik [11]
Entegrasyon testleri sürekli bir şekilde gerçekleştirilebilmekte midir?	Güvenilirlik [9, 11, 16]
Başarım testleri sürekli bir şekilde gerçekleştirilebilmekte midir?	Güvenilirlik [9, 11]
Dağıtımlar, betikler (*scripts*) vasıtasıyla otomatik bir şekilde gerçekleştirilebilmekte midir?	Güvenilirlik [11]
Sistem testleri, her bir dağıtımın ardından otomatik bir şekilde gerçekleştirilebilmekte midir?	Güvenilirlik [11]
Altyapı, Versiyonlanmış Kod Altyapısı (*Versioned Infrastructure as Code*) olarak sınıflandırılabilmekte midir?	Güvenilirlik [9, 11]
Test sonuçları ve diğer ölçütler grafiksel olarak görüntülenebilmekte midir?	Kullanılabilirlik [11, 16]
Bir özelliğe ilişkin ortalama teslim süresi ne kadardır?	Başarım Verimliliği [10]
Bir özelliğe ilişkin ortalama döngü süresi ne kadardır?	Başarım Verimliliği [18]
Ortalama değişiklik gerçekleştirme ve teslim süresi ne kadardır?	Başarım Verimliliği [12]
Minimum özellik sağlama ve teslim süresi ne kadardır?	Başarım Verimliliği [17]
Yeni bir özelliğin ortalama dağıtım süresi ne kadardır?	Başarım Verimliliği [11]
Geliştirilmekte olan koddaki veya altyapıdaki her bir değişiklik, denetlenebilir durumda mıdır?	Güvenlik [11]
İşlem hattındaki her bir koşu, izlenebilir ve hesap verebilir durumda mıdır?	Güvenlik [9, 11, 16]
İşlem hattı mimarisi, bileşen temelli veya beraber işletilebilir (*orchestrated*) yapıda mıdır?	İdame Edilebilirlik [9, 16]
İşlem hattında, uygun kütüphane(ler) ve/veya API('lar) (*Application Programming Interface* – Uygulama Programlama Arayüzü) kullanılmakta mıdır?	İdame Edilebilirlik [16]

Kaynakça

[1] Forester Consulting, *Continuous Delivery: A Maturity Assessment Model*, Mar. 2012.

[2] International Organization for Standardization (ISO), *Software engineering — Product quality — Part 1: Quality model*, ISO/IEC 9126-1:2001.

[3] Institute of Electrical and Electronics Engineers, *IEEE Standard Glossary of Software Engineering Terminology*, IEEE Std 610.12-1990(R2002).

[4] International Organization for Standardization (ISO), *Systems and Software engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — System and Software Quality Models*, ISO/IEC 25010:2011.

[5] Türk Standardları Enstitüsü (TSE), *Sistemler ve Yazılım Mühendisliği, Sistemler ve Yazılım Kalite Gereksinimleri ve Değerlendirmesi (SQuaRE) - Sistem ve Yazılım Kalitesi Modelleri*, TS ISO/IEC 25010, Şub. 2021.

[6] König, L., Steffens, A. *Towards a Quality Model for DevOps, Continuous Software* Engineering & Full-Scale Software Engineering, pp. 37-42, 2018.

[7] Anonymous, *DevOps Deployment Pipeline - Traffic Sign Clipart* (#4306142), Çevrim İçi: http://www.pikpng.com, Son Erişim Tarihi: 29.03.2022.

[8] Forester Consulting, *Continuous Delivery: A Maturity Assessment Model*, Technical Report, Mar. 2013, Çevrim İçi: http://info.thoughtworks.com/rs/thoughtworks2/images/continuous_delivery_a_maturity_assessment_modelfinal.pdf , Son Erişim Tarihi: 29.03.2022.

[9] Gupta, A. *Continuous Integration, Delivery, Deployment and Maturity Model*, 2015, Çevrim İçi: http://blog.arungupta.me/continuous-integration-delivery-deployment-maturity-model/, Son Erişim Tarihi: 29.03.2022.

[10] Hüttermann, M. *DevOps for Developers: Books for Professionals by Professionals*, Apress, New York, NY, 2012.

[11] Juner, C., Benlian, A. *Praxisbasierte Capability-Modelle für DevOps-Einsaetze in Unternehmen*, HMD Praxis der Wirtschaftsinformatik, vol. 54, no. 2, pp. 230-243, 2017.

[12] Lehtonen, T, Suonsyja, S., Kilamo, T., Mikkonen, T. *Defining Metrics for Continuous Delivery and Deployment Pipeline*, Proc. 14th Symposium on Programming Languages and Software Tools (SPLST'15), pp. 16-30, 2015.

[13] Mills, R. *A Maturity Matrix for Continuous Delivery Pipelines*, 2014. Çevrim İçi: https://www.coveros.com/a-maturity-matrix-for-continuous-delivery-pipelines/, Son Erişim Tarihi: 29.03.2022.

[14] Parks, J. *The Solinea DevOps Maturity Model*, 2016. Çevrim İçi: https://solinea.com/blog/solinea-devops-maturity-model, Son Erişim Tarihi: 29.03.2022.

[15] Ravichandran, A., Taylor, K., Waterhouse, P. *DevOps for Digital Leaders*, Apress, Berkeley, CA, 2016.

[16] Rehn, A., Palmborg, T., Boström, P. The Continuous Deivery Maturity Model, 2013. Çevrim İçi: https://www.infoq.com/articles/Continuous-Delivery-Maturity-Model, Son Erişim Tarihi: 29.03.2022.

[17] Riley, C. *Metrics for DevOps*, 2015. Çevrim İçi: https://devops.com/metrics-devops/, Son Erişim Tarihi: 29.03.2022.

[18] Trienekens, J. J. M. *Towards a Metrics Model for DevOps: Results of a Case Study in an Industrial Company*, Proc. International Academy, Research, and Industry Association First International Conference on Fundamentals and Advances in Software Systems Integration (IARIA-FASSI 2015), pp. 1-6, 2015.