Düzce Üniversitesi Bilim ve Teknoloji Dergisi

Araştırma Makalesi

DevOps Test Süreç Geliştirmede Yeni Bir Model Önerisi

Asım Kerem HANCI [a,*], Sevinç GÜLSEÇEN [b]

[a] *Enformatik Anabilim Dalı, Fen Bilimleri Enstitüsü, İstanbul Üniversitesi, İstanbul, TÜRKİYE*
[b] *Enformatik Anabilim Dalı, Fen Bilimleri Enstitüsü, İstanbul Üniversitesi, İstanbul, TÜRKİYE*
** Sorumlu yazarın e-posta adresi: asimkerem.hanci@ogr.iu.edu.tr*
DOI: 10.29130/dubited.1132368

Öz

Yazılım dünyasında ürünün hızlı bir şekilde pazara ulaşması için her geçen gün yeni yaklaşımlar belirlenmekte ve yeni metodolojiler benimsenmektedir. Son yıllarda bu konuda oldukça popüler olan DevOps metodolojisi, yazılım geliştirme yaşam döngüsünün diğer safhalarında olduğu gibi yazılım test aktivitelerinde ve test süreçlerinde köklü değişikliklerin oluşmasına ortam hazırlamıştır. Yazılım test aktivitelerinin yapısı büyük ölçüde değişime uğradığı için test süreç değerlendirmelerinde yeni modellere ihtiyaç duyulmaktadır. Gerçekleştirilen bu çalışmada, DevOps metodolojisinin test pratiklerini değerlendiren bir test süreç iyileştirme modeli sunularak bu ihtiyacı gidermek amaçlanmıştır. Model tasarımında akademi ve iş hayatındaki bilgilerden ortak olarak beslenmek adına literatürden ve deneyimli DevOps uzmanlarından yararlanılmıştır. Uzmanların katkısıyla DevOps test pratiklerinin değerlendirilmesi için önemli kriterler belirlenmiş ve bu kriterleri kapsayan yeni bir test süreç iyileştirme modeli oluşturulmuştur. Geliştirilen bu model DevOps pratiklerini gerçekleştiren bir firma üzerinde uygulanmıştır. Bu sayede hem modelin işlerliği hem de firmanın test olgunluğu değerlendirilmiştir. Değerlendirme süreci detaylı olarak gözlemlenmiş olup değerlendirme sonucu ve modelin uygulanabilirliği uzmanlar tarafından teyit edilmiştir.

Anahtar Kelimeler: *DevOps test süreçleri, DevOps Test süreç geliştirme, DevOps test olgunluk ölçümü*

A New Model Approach for DevOps Testing Process Improvement

Abstract

A new approach is designated and adopted for fast delivery of software products to market each passing day. DevOps methodology, the most popular approach in recent times, has the same great effect on testing about replacing the activities and processes as the other phases in SDLC. The new test process needs a new test process improvement model. Hence, it is intended to design a new test process improvement model which is convenient with DevOps test practices. In the design process, literature review and experienced DevOps experts were benefitted to combine the academic and business information. After the key elements of DevOps test practices were determined, the new test process improvement model was designated with the help of the experts. This new model was evaluated in terms of operability and maintability by being applied in an IT company. The evaluation process was observed in detail, the outcomes and applicability of the model were verified by the experts.

Keywords: *DevOps testing process, DevOps testing process improvement, DevOps testing maturity measurement*

Geliş:17/06/2022, Düzeltme: 27/12/2022, Kabul: 23/01/2023

I. GİRİŞ

Yazılım dünyasında ürünlerin daha hızlı piyasaya sürülmesi için yapılan optimizasyon arayışları, yeni yaklaşımların oluşmasına ortam hazırlamaktadır. 2000li yıllarda ortaya çıkan ve 2010lu yıllarda yaygınlaşan çevik yönetim modeli, ürünün müşteriye ulaşma hızını olumsuz etkileyen birçok faktörü ortadan kaldırarak yazılım dünyasında bir devrim olarak nitelendirilmiştir [1]. Çevik model ile geliştirme ekiplerinin üyeleri arasındaki iletişim artırılmış, ürünün gelişim safhasında değişebilen pazar ihtiyaçları göz önünde bulundurularak daha esnek bir yapı benimsenmiş ve ürünler bütüncül olarak değil, küçük parçalar halinde geliştirilmiştir [2][3].

Çevik yönetim birçok problemin çözüme kavuşmasını sağlamış olsa da operasyon ve geliştirme ekipleri arasındaki iş bölümü, büyük ürünlerin bakımının yönetimi gibi problemler bu modelde de devam etmiştir. Geliştirme ekipleri genel olarak bir yazılım ürününün ortaya çıkması için uygulanan yazılım geliştirme yaşam döngüsü faaliyetlerinden sorumlu tutulmaktadır. Geliştirilen ürünün canlı ortama alınması için oluşturulan talepler operasyon ekiplerine iletilmektedir. Operasyon ekipleri bu taleplerin gerçekleştirilmesinin yanında uygulanan yazılım geliştirme yaşam döngüsü modelinden tamamen bağımsız olarak altyapı erişiminin sürekliliği, canlı ortamın sürekli olarak izlenmesi gibi faaliyetlerden sorumlu tutulmaktadır [4]. Geleneksel yaklaşımlarda olduğu gibi çevik yönetim modelinde de geliştirme ve operasyon ekipleri birbirinden bağımsız olarak çalışmaktadır. Operasyon ekipleri canlı ortamların güncel ve sürekli erişilebilir tutulması konularına yoğunlaştıkları için geliştirme ekiplerinden gelen taleplere daha az öncelik vermektedir. Bu sebeple ürünler müşteri ile daha geç buluşmaktadır ve bu durum rekabet ortamında olumsuz sonuçlar doğurabilmektedir. Ayrıca her talep ile birlikte büyüyen altyapıyı ve gelişen uygulama katmanlarını yönetebilmek bir süre sonra imkansız hale gelmektedir [5][6]. Geliştirme ve operasyon ekipleri arasındaki bu sorunların çözülmesi amacıyla 2008 yılında Andrew Shafer ve Patrick Debois tarafından "DevOps" kavramı ortaya çıkarılmıştır [7]. DevOps sözcüğü, İngilizce'de geliştirme anlamına gelen "development" ve operasyon anlamına gelen "operations" kelimelerinin birleştirilmesi ile oluşturulmuştur [8]. Temel olarak geliştirme ve operasyon ekiplerinin dayanışma içerisinde çalışmasını öngören ve süreci iyileştirmek adına farklı yazılım geliştirme pratiklerini içinde barındıran bir metodoloji olarak karşımıza çıkmaktadır. DevOps ile birlikte geliştirme ekiplerine operasyon ekiplerinin bazı yetenekleri kazandırılarak operasyon ekiplerinin iş yükü azaltılabilmekte ve bu sayede ürünlerin canlı ortama alınmasına engel olan sebepler ortadan kaldırılabilmektedir [9].

Çevik yönetim modeli ile temelleri atılan ve DevOps metodolojisi ile daha da önem kazanan üretim hızı; sürekli entegrasyon (continuous integration) ve sürekli teslimat (continuous delivery) kavramlarının öne çıkmasını sağlamıştır. Sürekli entegrasyon, yapılan geliştirmelerin otomatik bir şekilde ürünün yapısına dahil edilmesini; sürekli teslimat da geliştirmelerin canlı ortama alınma faaliyetlerini içermektedir [10]. Entegrasyon ve teslimat süreçleri süreklilik için otomatik olarak ilerleyeceğinden dolayı kalite kontrol faaliyetleri de DevOps ile değişime uğramıştır [11]. Geliştirilen kodun testleri otomasyon araçları ile test edilerek yalnızca başarılı olmaları durumunda ürüne entegre edilebilmektedir. Test aktivitelerinin değişime uğraması ile birlikte test süreçlerinin değerlendirme kriterlerinin de farklılaşması, yeni test süreç değerlendirme modellerinin oluşturulması gerekmektedir. Bu ihtiyaç göz önünde bulundurularak, yapılan bu çalışma ile DevOps metodolojisinde test süreç iyileştirme modeli tasarlanmıştır. Geliştirilen bu yeni model ile birlikte;

- Test süreçlerinin güncel süreçlere uygun olarak değerlendirilmesi,
- DevOps metodolojisinde test aktivitelerinin öneminin belirlenmesi,
- Yazılım geliştirme odaklı firma ve kurumların test disiplinine yönlendirilmesi amaçlanmıştır.

Çalışmanın ikinci bölümünde 'DevOps' ve 'test süreç iyileştirme modeli' alanları ile ilgili literatür taraması yer almaktadır. Üçüncü bölümde bu çalışmada yararlanılan malzeme ve yöntemlere değinilmiştir. Dördüncü bölümde model tasarımının detayları yer almaktadır. Beşinci bölümde yapılan çalışmanın diğer modellerle olan karşılaştırmalı analizi ve sonuçları ele alınmıştır.

II. LİTERATÜR TARAMASI

Gerçekleştirilen literatür taraması 'DevOps', 'DevOps test süreçleri', 'yazılım süreç değerlendirme', 'yazılım test süreç geliştirme/iyileştirme modeli' ve DevOps ile ilgili test pratiklerini ele alan araştırmaları içermektedir. Bu sebeple literatür taraması üç ana başlık altında ele alınmıştır.

A. DEVOPS

DevOps ile ilgili literatür taramasında DevOps konusunu farklı açılardan ele alan birçok yayına rastlanmıştır. Bu yayınların en çok referans verilenlerinden biri Erich ve arkadaşları tarafından 2015 yılında yayımlanan ve 2017 yılında revize edilen literatür taramasıdır. Çalışmada DevOps prensiplerine değinilmiş ve altı farklı işletmede DevOps pratikleri uygulanarak çeşitli araştırmalar yapılmıştır. Bu araştırmalar sonucunda DevOps uygulamasında önemli olan alanların; dayanışma kültürü, otomasyon, ölçümleme, paylaşım, servisler, kalite güvencesi ve yönetim olduğu vurgulanmıştır [12]. França da Erich'in çalışmasına benzer şekilde DevOps kavramını prensipler bütünü olarak ele almıştır ve bu prensipleri altı alanda tanımlamıştır: sosyal yön, otomasyon, ölçümleme, paylaşım, kalite kontrol ve yalınlık. França ayrıca DevOps alanındaki akademik çalışmaların yeterli olgunlukta olmadığını belirterek 'gri literatür' olarak nitelendirilen blog, web siteleri ve teknik raporlardan da faydalanmıştır [13].

DevOps metodolojisinin benimsenmesindeki süreci ele alan birçok yayın bulunmaktadır. Lwakatare, 2019 yılında beş firma üzerinde yaptığı araştırmada DevOps'un benimsenmesiyle özellikle küçük ve orta büyüklükteki firmalarda süreçlerin çok daha hızlı ilerlediğini gözlemlemiştir [14]. Smeds, 2015 yılında yaptığı çalışmada DevOps ile yazılım geliştirmeye geçiş dönemini olumsuz etkileyen faktörleri 'yetenek', 'kültür' ve 'teknoloji' başlıkları altında ele almıştır [9]. Bucena, küçük firmalarda DevOps'un kolayca uygulanabilmesi için önerdiği pratikleri uluslararası bir firmanın şubesinde uygulayarak çeşitli çıkarımlarda bulunmuştur [15].

DevOps metodolojisi ile birlikte önem kazanan mikro servis mimarisi ile ilgili başlıca yayınlar Luz ve Balalaie tarafından literatüre kazandırılmıştır. Luz, mikro servis mimarisini detaylıca tanımlayarak monolitik mimariden farklarını bütün yönleriyle ortaya koymuştur. Ayrıca mikro servis mimarisinin işletilme sürecini üç farklı devlet enstitüsünde deneyimleyerek çıkarımlarını paylaşmıştır [16]. Balalaie ise monolitik yapıdaki kodların mikro servis mimarisine taşınması sırasında defalarca kullanılabilecek yöntemleri ele almıştır [17]. Literatür taraması bu noktaya kadar DevOps ile ilgili bir genel bilgi toplama amacı taşımaktadır. Oluşturulacak test olgunluk değerlendirme modeline temel oluşturmak amaçlanmıştır.

B. SÜREÇ GELİŞTİRME VE OLGUNLUK DEĞERLENDİRME MODELLERİ

DevOps ile ilgili temel bilgiler edinildikten sonra yazılım olgunluk değerlendirme modelleri incelenerek test olgunluk değerlendirme modeline dayanak oluşturulması amaçlanmıştır. Literatürde DevOps olgunluk değerlendirme modeli ile ilgili olarak öne çıkan birkaç araştırma bulunmaktadır. Zarour, DevOps olgunluk değerlendirme modelleri üzerinde bir literatür taraması gerçekleştirmiştir. Bu çalışmada yedi DevOps olgunluk değerlendirme modelinin karşılaştırmalı analizini gerçekleştirmiştir [18]. Bahrs tarafından tasarlanan IBM modeli 4 aşamadan oluşmaktadır ve plan/ölçüm, geliştirme/test, sürüm, izleme kategorilerinde değerlendirme yapmaktadır [19]. Buna benzer şekilde fakat değerlendirme ölçütleri farklı olmak üzere Mohamed [20], Menzel [21], Inbar [22] ve Eficode [23] tarafından aşamalı olarak tasarlanan yeni süreç değerlendirme modelleri önerilmiştir. Fejiter tarafından önerilen model ise diğerlerinden farklı olarak sürekli bir modeldir ve bir firmayı belli kritik değerlendirme alanlarına göre incelemiştir. Bu değerlendirme alanları 'kültür ve dayanışma', 'ürün, süreç ve kalite' ve 'temeller/dayanaklar' ana başlıkları altında toplanmıştır [24]. Bu modeller bütün yazılım geliştirme safhalarını kapsayan modellerdir ve test aktivitelerini tam anlamıyla

kapsamamaktadır. Fakat oluşturulması planlanan model yalnızca test safhasının detaylı değerlendirmesini güncel kriterleri kapsayacak şekilde oluşturulacaktır.

Literatür taramasının kalan kısmında yazılım testi olgunluk ölçüm ve test süreç iyileştirme modelleri araştırılmıştır. Bu araştırmada 2017 yılında Garousi ve arkadaşları tarafından ortaya atılan çok yönlü literatür taraması dikkat çekmektedir. Akademik çalışmalar dışındaki kaynakları da içeren araştırmada, 181 geçerli kaynak ve 58 ölçüm modeli tespit edilmiştir. Bu modellerin sınıflandırılması ve birbirleri ile olan ilişkileri kronolojik olarak ele alınmıştır [25]. Günümüzde test süreç iyileştirmede yaygın olarak kullanılan modeller TMMI [26] ve çevik türevleri [27][28] ve TPI [29] modelleridir. Garousi'nin literatür taramasında ele alınan modeller incelendiğinde DevOps metodolojisini kapsayıcı ve bütün pratiklerini içeren bir test süreç iyileştirme modeline rastlanmamıştır. Bu çalışma ile DevOps pratiklerini içeren ve firmaların test olgunluğunu değerlendiren bir model önerisi sunularak literatüre katkı sağlanması amaçlanmaktadır.

Ülkemizde de DevOps ve test süreç iyileştirme modelleri ile ilgili çeşitli çalışmalar gerçekleştirilmiştir. Arifoğlu, araştırmasında büyük ölçekli uygulamaların DevOps süreçlerine uyarlanması sürecinde yaşanan zorlukları bir kuruluş üzerinden örneklendirerek ele almıştır [30]. Kutluay ise çevik yönetim modeli kapsamında sürekli teslimat kavramının önemini incelemiştir [31]. Arkan, yazılım test servislerinin olgunluğunu ölçen bir model öne sürmüş ve bu modelin uygunluğunu iki firmada uygulayarak değerlendirmiştir [32]. Arkan'ın oluşturduğu modelde servislerin olgunluğu ölçülmekte iken yapılacak bu çalışma ile bütün test sürecinin değerlendirilmesi gerçekleştirilecektir.

C. DEVOPS TEST PRATİKLERİ

Literatür taramasının son aşamasında test süreç iyileştirme modelinin tasarlanması için uzmanlar ile yapılan görüşmeler sonrasında DevOps metodolojisinde gerekli olabilecek test pratikleri ile ilgili araştırmalar yapılmıştır. Uzmanlar tarafından önerilen bu pratiklerle ilgili literatür taraması aşağıdaki gibi özetlenebilir:

- ISO 25010 Kapsamında Ele Alınan Kalite Karakteristikleri: Bu özellikler; fonksiyonalite, kullanılabilirlik, güvenilirlik, verimlilik, sürdürülebilirlik, taşınabilirlik, uyumluluk ve güvenlik kategorileri olarak sıralanabilir [33].
- Test Güdümlü Geliştirme: Beck tarafından detaylıca ele alınmıştır. Kod yazılmadan önce geliştirmenin test kodları yazılır, ortada bir ürün olmadığından dolayı testler başarısız olur. Testler başarılı oluncaya kadar geliştirmeye devam edilir. Başarılı olduktan sonra kod diğer kodlar ile entegre olmaya hazır hale gelir [34].
- Davranış Güdümlü Geliştirme: Wynne ve arkadaşları tarafından detaylı olarak ele alınmıştır. Gereksinimler DSL (Domain Specific Langugage) bir formatta yazılır [35]. Gherkin bu formata örnek olarak verilebilir. Bu dil hem müşteri hem de teknik ekip tarafından rahatlıkla anlaşılır formattadır ("Bir banka müşterisi olarak bankanın mobil uygulamasına kullanıcı adımı ve şifremi yazdığımda uygulamaya girebilmeliyim" gibi). Yazılım mühendisleri Gherkin ile oluşturulan çalıştırılabilir gereksinimleri valide edecek ürün kodunu geliştirirler. Bu sayede hem paydaşlar arasında çatışma yaşanmamakta hem de kodlar gereksinimlere tamamen uygun olarak geliştirilmektedir.
- Sürekli Entegrasyon Testleri: Tamamı otomatik olarak geliştirilerek -başarılı olması durumunda- kodun manuel bir test sürecinden geçmeden derlenmesini sağlamaya yönelik birim, entegrasyon ve kontrat testlerinin tamamıdır [36][37].
- Sürekli Teslimat Testleri: Tamamı otomatik olarak geliştirilen fonksiyonel testlerdir. Ara yüz, servis ve veri tabanı gibi test nesnelerinin gereksinimlere uygunluğunu doğrulama amacını taşımaktadır [38].
- Mikro Servis Testleri: DevOps ile birlikte monolitik mimari yerine kullanılan mikro servis mimarisinin test aktiviteleridir. Entegre servislerin testleri için kontrat testleri, bu testlerin sanallaştırılmış servisler ile gerçekleştirilebilmesini ve bağımsız mikro servisler için servis fonksiyonel testlerini içermektedir [39].

- Test Veri Yönetimi: Test aktiviteleri sırasında araçlar ile yapılandırılmış ve sistematik olarak yürütülen veri faaliyetleridir. Testler sırasında canlı ortamdan alınan verilerin maskelenmiş olarak kullanılmasını veya sentetik veri üretimi gibi faaliyetleri içermektedir [40].
- Hata Tahminleme: Geliştirme yapılan kodun mantıksal hatalarını çeşitli girdilere dayanarak makine öğrenmesi teknikleri ile tahmin eden algoritmaları kullanan modellerdir. Test sürecine; geliştirme yapılan alanın yaratacağı regresyon etkilerini belirleme ve bu alanların testlerinin gerçekleştirilmesi gibi yararlar sağlamaktadır [41].

III. MATERYAL, YÖNTEM ve BULGULAR

DevOps metodolojisinde test süreç iyileştirme modeli oluşturmak için literatür taramasının yanında firmalar/deneyimli çalışanlar tarafından oluşturulan bloglar, konferanslar, başarı hikayeleri de incelenerek öncelikle DevOps ve test kavramlarının yazarlar tarafından kapsamlıca anlaşılması hedeflenmiştir. Daha önce test süreç iyileştirme modeli öneren yayınlarda izlenen metotlar detaylıca araştırılmıştır. Yararlanılan kaynaklar doğrultusunda model oluşturulmadan önce bilişim sektöründe DevOps pratiklerini deneyimleyen uzmanlarla yapılandırılmamış görüşmeler gerçekleştirilmiştir. Üç farklı firmadan toplam yedi uzmanla yapılan görüşmelerde modelleme için gerekli fikri temeller atılmıştır. Bu temelleri daha da güçlendirmek ve daha fazla uzmana ulaşabilmek adına anket uygulamasının yapılmasına karar verilmiştir. Bu amaçla yapılandırılmamış görüşme gerçekleştirilen üç uzmana da danışılarak on sorudan oluşan bir anket düzenlenmiştir ve bu anket İstanbul Üniversitesi Sosyal ve Beşeri Bilimler Araştırmaları Etik Kurulu'na sunularak etik kurul onayı alınmıştır. Daha sonra anket çalışması 59 uzmana uygulanarak cevapları analiz edilmiştir. Anket cevapları, uzman görüşleri ve literatür bilgileri incelenerek DevOps metodolojisi için test iyileştirme modeli tasarlanmıştır.

A. UZMAN GÖRÜŞMELERİ

Türkiye'nin önde gelen firmalarında DevOps deneyimi bulunan yedi uzmana aşağıdaki araştırma soruları yöneltilmiştir:

- **S1**: Firmanızda DevOps'u ne zamandır uyguluyorsunuz?
- **S2:** DevOps süreçlerini ne kadar zamandır deneyimliyorsunuz?
- **S2**: DevOps ile birlikte test süreçlerinizde yaşadığınız en belirgin değişiklikler nelerdir?
- **S3**: DevOps test süreçlerini iyileştirme modeli/olgunluk ölçüm modeli üretmek mümkün olabilir mi?
- **S4:** Bir DevOps test süreç iyileştirme modeli üretseydiniz hangi DevOps ve test bileşenlerini ön planda tutardınız?

Uzmanların yanıtları özet olarak Tablo 1'de listelenmiştir.

B. ANKET UYGULAMASI

Çalışmanın daha efektif olması ve daha fazla uzmana danışabilmek adına uzman görüşmelerinde elde edilen bilgilerden yola çıkılarak on sorudan oluşan bir anket düzenlenmiştir. Bu anket Türkiye'nin farklı bilişim firmalarında çalışan 59 uzmana uygulanmıştır. Hazırlanan anket soruları aşağıdaki gibidir:

- **AS1:** Mesleki Pozisyonunuz?
- **AS2:** Firmanızda hangi yazılım geliştirme yaşam modeli uygulanıyor?
- **AS3:** Firmanızdaki yazılım test süreçlerini nasıl tanımlarsınız?
- **AS4:** Firmanızın yazılım test süreçlerinden sonra gerçekleştirilen son kullanıcı testlerinde tespit edilen hataların genelinin önemini nasıl sınıflandırırsınız?
- **AS5:** Firmanızın iş birimlerinin yazılım testi kalitesinden memnuniyetini nasıl yorumlarsınız?
- **AS6:** Yazılım test aktivitelerinde otomasyondan ne ölçüde faydalanıyorsunuz?
- **AS7:** Aşağıdakilerden hangisi/hangileri yazılım test süreçlerinin kalitesini olumsuz etkiler?

- **AS8:** Gereksinim yönetimi, analiz ve yazılım aşamaları sizce nasıl daha verimli hale getirilebilir?
- **AS9:** Test süreçlerinin verimini artırmak için aşağıdakilerin hangilerinden faydalanılabilir?
- **AS10:** Test süreçlerinin kalitesinin artırılması için sizce aşağıdaki yaklaşımlardan hangileri benimsenebilir?

Tablo 1. *Uzmanların Cevapları*

Uzmanlar	S1	S2	S2	S3	S4
U1	2 yıl	2 yıl	Otomasyon Oranının Artması	Evet	Early Testing (erken test), Otomasyon Oranı, Entegrasyon Testleri
U2	2 yıl	3 yıl	Birim, Servis ve Altyapı Testleri	Evet	TDD (Test Güdümlü Geliştirme), BDD (Davranış Güdümlü Geliştirme)
U3	2 yıl	2 yıl	Fonksiyonel Olmayan Testler	Evet	Performans, Güvenilirlik, Kullanılabilirlik, Bakım
U4	1 yıl	3 yıl	TDD	Evet	TDD, Birim Test, Entegrasyon Testi
U5	1 yıl	1 yıl	CI (Sürekli Entegrasyon), CD (Sürekli Teslimat) Test Aktiviteleri	Evet	Sanallaştırma Faaliyetleri (servis, bulut vb.), Hata Tahminleme
U6	1 yıl	1 yıl	BDD	Evet	BDD, Fonksiyonel Olmayan Testler
U7	1 yıl	2 yıl	Servis Testleri, Test Veri Yönetimi	Evet	TDD, BDD, Test Veri Yönetimi, Servis Testleri

Sorulara ilişkin açıklamalar ve her bir sorunun uzmanlar tarafından cevapları aşağıda belirtildiği gibidir.

AS1-Mesleki Pozisyonunuz: Katılımcıların yazılım geliştirmeyi hangi pozisyonda uyguladıklarını belirlemek amacıyla sorulmuştur.

Tablo 2'den de anlaşılacağı gibi katılımcıların %27,12'sini analistler, %8,47'sini yazılım mühendisleri, %59,32'sini test mühendisleri ve %5,08'ini bilişim teknolojileri yöneticileri oluşturmaktadır.

AS2-Firmanızda hangi yazılım geliştirme yaşam modeli uygulanıyor: Katılımcıların yazılım testini uyguladıkları yazılım geliştirme yaşam döngüsünü belirlemek amacıyla sorulmuştur.

Tablo 3'ten de anlaşılacağı gibi katılımcıların hiçbiri şelale modeli kullanmamakta iken %3,39'u V model, %93,22'si çevik model, %3,39'u diğer modelleri uygulamaktadır. Diğer model seçeneğini seçen katılımcıların cevapları incelendiğinde 'spiral' modelini belirttiği görülmüştür.

AS3-Firmanızdaki yazılım test süreçlerini nasıl tanımlarsınız: Katılımcıların firmalarındaki yazılım test süreçlerini hangi aşamada gördüklerini belirlemek amacıyla sorulmuştur.

Tablo 4'ten de anlaşılacağı gibi katılımcıların %67,80'i firmalarındaki test süreçlerinin yeterince olgunlaşmış olmadığını düşünürken, %28,81'i etkin olarak yönetilebildiğini, %3,39'unun da optimize bir test sürecine sahip olduğunu düşünmektedir.

***Tablo 2**. Katılımcıların Mesleki Pozisyon Dağılımları*

Cevap Seçenekleri	%	Adet
Analist	27,12%	16
Yazılım Mühendisi	8,47%	5
Test Mühendisi	59,32%	35
Bilişim Teknolojileri Yöneticisi	5,08%	3
Toplam	**100,00%**	**59**

***Tablo 3**. Katılımcıların Uyguladıkları SDLC Modeli Dağılımları*

Cevap Seçenekleri	%	Adet
Waterfall (şelale) modeli	0,00%	0
V model	3,39%	2
Agile (çevik) model	93,22%	55
Diğer (lütfen belirtin)	3,39%	2
Toplam	**100,00%**	**59**

AS4-Firmanızın yazılım test süreçlerinden sonra gerçekleştirilen son kullanıcı testlerinde tespit edilen hataların genelinin önemini nasıl sınıflandırırsınız: Katılımcıların firmalarındaki yazılım test süreçlerinde yazılım test aktivitelerinde gözden kaçırılan hataların niteliğini belirlemek amacıyla sorulmuştur.

Tablo 5'ten de anlaşılacağı gibi katılımcıların %10,17'si test mühendislerinin gözünden kaçabilen hataların son kullanıcı tarafından fark edildiğini belirtmiştir. %45,76'sı son kullanıcı testini gerçekleştiren iş birimlerinin yalnızca minör veya kozmetik hatalar tespit edebildiğini belirtmiştir. %20,34'ü iş birimleri tarafından tespit edilen bulguların hata niteliğinde olmadığını belirtirken %23,73 oranında katılımcı iş birimleri tarafından hata olarak nitelendirilen noktaların genellikle gereksinim değişikliği içerdiğini belirtmiştir.

***Tablo 4**. Katılımcıların Test Süreç Değerlendirme Dağılımları*

Cevap Seçenekleri	%	Adet
Yeterince olgunlaşmış değil	67,80%	40
Etkin olarak yönetilebilen	28,81%	17
Optimize	3,39%	2
Toplam	**100,00%**	**59**

***Tablo 5**. Son Kullanıcı Tarafından Tespit Edilen Hataların Değerlendirme Dağılımları*

Cevap Seçenekleri	%	Adet
İş birimleri genellikle test mühendislerinin gözünden kaçabilen çok kritik hatalar tespit edebiliyor.	10,17%	6
İş birimleri yalnızca minör veya kozmetik hatalar tespit edebiliyor.	45,76%	27
İş birimlerinin bulguları genellikle hata niteliğinde değil.	20,34%	12
İş birimlerinin hata olarak nitelendirdiği noktalar genellikle gereksinim değişikliği içeriyor.	23,73%	14
Toplam	**100,00%**	**59**

AS5-Firmanızın iş birimlerinin yazılım testi kalitesinden memnuniyetini nasıl yorumlarsınız: Katılımcıların firmalarındaki son testleri gerçekleştiren iş birimlerinin yazılım test süreçleri konusundaki memnuniyetini belirlemek amacıyla sorulmuştur.

Tablo 6'dan da anlaşılacağı gibi katılımcıların %10,17'si iş birimlerinin test mühendisleri tarafından yapılan testlere güvenmediklerini belirtmişlerdir. Katılımcıların %66,10'u iş birimlerinin yazılım test aktiviteleri konusunda bir fikirlerinin olmadıklarını, standart son kullanıcı testleri gerçekleştirdiklerini belirtmişlerdir. Katılımcıların %23,73'ü de iş birimlerinin test mühendisleri tarafında gerçekleştirilen yazılım test aktivitelerine oldukça güvendiklerini, bu yüzden son kullanıcı testlerini çok gelişigüzel ve hızlı yaptıklarını belirtmişlerdir.

AS6-Yazılım test aktivitelerinde otomasyondan ne ölçüde faydalanıyorsunuz: Katılımcıların firmalarında test otomasyonundan ne ölçüde faydalanıldığını belirlemek amacıyla sorulmuştur. Tablo 7'den de anlaşılacağı gibi katılımcıların %6,78'i test süreçlerinde otomasyon bulunmadığını belirtirken, %86,44'ü tekrarlı testlerinde otomasyon kullandığını, %6,78'i de otomasyonun bütün test süreçlerinde yer aldığını belirtmiştir.

AS7-Aşağıdakilerden hangisi/hangileri yazılım test süreçlerinin kalitesini olumsuz etkiler: Yazılım testi süreçlerini olumsuz etkileyen etmenleri belirlemek amacıyla sorulmuştur. Bu soruda seçenekler arasında çoklu seçim yapma imkanı tanınmıştır.

Tablo 8'den de anlaşılacağı gibi katılımcıların %57,63'ü gereksinim yönetiminin dinamik olmamasının, %59,32'si analiz ve tasarım aşamalarının verimsiz olmasının, %69,49'u geliştirme süreçlerinin verimsiz olmasının, %93,22'si test sürelerinin çok kısa olmasının, %93,22'si yeni test yaklaşımları yerine geleneksel yöntemlerin kullanılmasının, %84,75'i yetersiz test veri yönetimi süreçlerinin, %88,14'ü de test alanında nitelikli personel eksikliğinin test süreçlerinin kalitesini olumsuz etkilediğini belirtmişlerdir.

***Tablo 6**. Son Kullanıcıların Yazılım Testine Bakış Açısını Değerlendirme Dağılımları*

Cevap Seçenekleri	%	Adet
Yazılım testine güvenmiyorlar, kendileri detaylı testler gerçekleştiriyorlar.	10,17%	6
Yazılım test aktiviteleri konusunda bir fikirleri yok, standart son kullanıcı testlerini gerçekleştiriyorlar.	66,10%	39
Yazılım testine oldukça güveniyorlar, son kullanıcı testleri çok gelişigüzel ve hızlı yapılıyor.	23,73%	14
Toplam	**100,00%**	**59**

***Tablo 7**. Katılımcıların Otomasyon Kullanım Oranlarını Değerlendirme Dağılımı*

Cevap Seçenekleri	%	Adet
Test süreçlerimizde otomasyon bulunmuyor, yalnızca manuel süreçleri işletiyoruz.	6,78%	4
Tekrarlı testlerimizde kullandığımız otomasyon araçlarımız bulunuyor, yeni geliştirmeleri manuel olarak test ediyoruz.	86,44%	51
Bütün test süreçlerimizde otomasyon yer almaktadır.	6,78	4
Toplam	**100,00%**	**59**

AS8-Gereksinim yönetimi, analiz ve yazılım aşamaları sizce nasıl daha verimli hale getirilebilir: Gereksinim yönetimi, analiz ve yazılım aşamalarının daha kaliteli hale getirilmesini sağlayan faaliyetleri belirlemek amacıyla sorulmuştur. Bu soruda seçenekler arasında çoklu seçim yapma imkanı tanınmıştır. Tablo 9'dan da anlaşılacağı gibi katılımcıların %59,32'si kullanıcı ile sürekli iletişim sağlanarak, %83,05'i ekip koordinasyonu ile dinamik iş paylaşımı yapılarak, %37,29'u birlikte çalışma ortamı yaratılarak, %93,22'si şeffaf bir çalışma ortamı oluşturularak, %91,55'i yeni yazılım geliştirme yaklaşımlarının benimsenerek, %84,75'i de manuel süreçler yerine daha teknolojik yapılar kullanılarak gereksinim yönetimi, analiz ve yazılım aşamalarının daha verimli hale getirilebileceğini belirtmişlerdir.

***Tablo 8**. Katılımcıların Test Süreci Kalitesini Değerlendirme Dağılımı*

Cevap Seçenekleri	%	Adet
Gereksinim yönetiminin dinamik olmaması	57,63%	34
Analiz ve tasarım aşamalarının verimsiz olması	59,32%	35
Geliştirme süreçlerinin verimsiz olması	69,49%	41
Test sürelerinin çok kısa olması	93,22%	55
Yeni test yaklaşımları yerine geleneksel yöntemlerin kullanılması	93,22%	55
Yetersiz test veri yönetimi süreçleri	84,75%	50
Nitelikli personel eksikliği	88,14%	52

***Tablo 9**. Katılımcıların Analiz ve Yazılım Süreçlerini İyileştirme Önerileri Dağılımı*

Cevap Seçenekleri	%	Adet
Kullanıcı ile sürekli iletişim	59,32%	35
Ekip koordinasyonu ile dinamik iş paylaşımı	83,05%	49
Birlikte çalışma ortamı	37,29%	22
Şeffaflık	93,22%	55
Yazılım geliştirmede yeni yaklaşımların benimsenmesi	91,55%	54
Manuel süreçler yerine daha teknolojik yapıların kullanılması	84,75%	50

AS9-Test süreçlerinin verimini artırmak için aşağıdakilerin hangilerinden faydalanılabilir: Test süreçlerinin daha kaliteli hale getirilmesini sağlayan faaliyetleri belirlemek amacıyla sorulmuştur. Bu soruda seçenekler arasında çoklu seçim yapma imkanı tanınmıştır.

Tablo 10'dan da anlaşılacağı gibi katılımcıların %69,49'u yeni yaklaşımlar için altyapılar benimsenerek, %89,93'ü sürekli entegrasyon sağlamaya yönelik test süreçlerin benimsenerek, %77,97'si sanallaştırma faaliyetlerini test süreçlerine dahil ederek, %81,36'sı test veri yönetimini etkin olarak kullanarak test süreçlerinin verimini artırabileceklerini belirtmişlerdir. %3,39 oranında "diğer" seçeneğini belirten katılımcılar tamamen otomatik süreçlerin uygulanmasını ve test mühendislerinin sürecin en başından dahil edilmesini önermişlerdir.

AS10-Test süreçlerinin kalitesinin artırılması için sizce aşağıdaki yaklaşımlardan hangileri benimsenebilir: Test süreçlerinin daha kaliteli hale getirilmesini sağlayan ve profesyonel olarak uygulanan yaklaşımları belirlemek amacıyla sorulmuştur. Bu soruda seçenekler arasında çoklu seçim yapma imkanı tanınmıştır.

Tablo 11'den de anlaşılabileceği gibi katılımcıların %86,44'ü TDD, %71,19'u BDD, %74,58'i sürekli entegrasyon/sürekli teslimat pratiklerini, %62,71'i statik test süreçlerini, %69,49'u da fonksiyonel olmayan testleri uygulayarak test süreçlerinin kalitesinin artırılabileceğini belirtmişlerdir.

C. YAZILIM TEST SÜREÇ İYİLEŞTİRME MODELİNİN TASARIMI

Önceki bölümlerde ele alınan literatür taraması, uzman görüşmeleri ve gerçekleştirilen anketler sonucunda, bir test süreç iyileştirme modelinin tasarımı için gerekli altyapı oluşturulmuştur. Model tasarımı yapılmadan önce literatürde kabul gören başlıca yazılım ve test modelleri incelenmiştir. Bu modellerde iki tip yapı dikkat çekmektedir:

- Firmaların test olgunluğunu değerlendirirken derecelendiren bir yaklaşım izleyen aşamalı modeller,
- Kritik değerlendirme alanlarını belirleyerek derecelendirmekten çok bu alanlara göre değerlendiren sürekli modeller

Tablo 10. *Katılımcıların Test Süreçlerini İyileştirme Önerileri Dağılımı*

Cevap Seçenekleri	%	Adet
Yeni yaklaşımlara uygun altyapının belirlenmesi	69,49%	41
Sürekli entegrasyon sağlamaya yönelik test süreçlerinin benimsenmesi	89,93%	53
Sanallaştırma faaliyetlerinin test süreçlerine dahil edilmesi	77,97%	46
Test veri yönetiminin etkin olarak gerçekleştirilmesi	81,36%	48
Diğer (lütfen belirtin)	3,39%	2

Tablo 11. *Katılımcıların Test Süreçlerini İyileştirme Pratikleri Önerileri Dağılımı*

Cevap Seçenekleri	%	Adet
TDD (Test Güdümlü Geliştirme)	86,44%	51
BDD (Davranış Güdümlü Geliştirme)	71,19%	42
Continuous Integration (Sürekli Entegrasyon)/ Continuous Deployment (Sürekli Teslimat)	74,58%	44
Statik test süreçlerinin kullanılması	62,71%	37
Fonksiyonel olmayan testlerin sürece entegre edilmesi	69,49%	41

Bu çalışma ile iki farklı yapının özelliklerini içeren hibrit bir model tasarımı amaçlanmıştır. Modelde 4 aşamalı bir yapı içerisinde her aşamaya özgü kritik değerlendirme alanları belirlenmiştir. İlgili değerlendirme alanlarının uygulanması o aşama için bu gereksinimin karşılandığı anlamına gelmektedir. Bir seviyede başarılı sayılabilmek için o seviyeye ait kritik değerlendirme alanlarının %80 oranında sağlanması gerekmektedir. Bu oran uzmanlarla yapılan görüşmeler ve literatürdeki diğer modellerin başarı oranları araştırılması neticesinde belirlenmiştir. Uzmanlara göre bir seviyede başarılı olabilmenin şartı o seviyeye ait fonksiyonların %80'inin yerine getirilmesi olarak yorumlanmıştır. Ayrıca TMMI gibi modellerde de başarı şartı %80-%85 oranında belirtilmiştir [26]. Tasarlanan modelin aşamaları ve her aşamaya özgü kritik değerlendirme alanlarının açıklamaları aşağıdaki gibidir.

C.1. Birinci Seviye: Başlangıç (Initial)

DevOps test süreç değerlendirmesinde 'başlangıç' seviyesinde değerlendirilen bir firmanın aktif bir test sürecinin olmadığı varsayılmıştır. Test aktiviteleri analist veya yazılım mühendisleri tarafından gayri resmi bir şekilde yapılmaktadır. Test süreçleri için ayrılan bir personel bütçesi bulunmamaktadır. Bu

safhada bulunan bir firma için amaç yazılım kodunun herhangi bir şekilde çalıştığını kanıtlamaktır. 'Başlangıç' seviyesi için herhangi bir kritik alan bulunmamaktadır.

C.2. İkinci Seviye: Yönetilebilen (Managed)

Bir firmanın 'yönetilebilen' seviyesinde test olgunluğuna sahip olabilmesi için test süreçlerinin minimum seviyede yönetilebildiği varsayılmıştır. Test stratejisi uygulanmakta, test planları bulunmakta, test metrikleri kullanılarak test raporları yayınlanmakta, test aktiviteleri için test mühendisi veya test analisti gibi unvanlara sahip olan personel bütçesi ayrılmaktadır. Bu bilgiler doğrultusunda 'yönetilebilen' seviyesi için belirlenen kritik alanlar aşağıdaki gibidir:

- **Test Stratejisi:** Testlerin uygulanması sırasında takip edilen yöntemleri ifade etmektedir.
- **Test İlkeleri:** Firmanın yazılım testinde projelerden bağımsız olarak takip ettiği politikayı ifade etmektedir.
- **Test Planı:** Her bir proje için özel olarak hazırlanan ve içerisinde projenin test aktivitelerine, kaynaklarına, sürelerine ait detayları içeren test dokümanını ifade etmektedir.
- **Test Raporları:** Projenin test aşamasında paydaşlara raporlanan metrikleri ifade etmektedir.
- **Test Personeli:** Yalnızca yazılım test aktivitelerini gerçekleştiren firma çalışanlarını ifade etmektedir.

Bu kritik alanlar belirlenirken -uzman görüşmeleri ve anket sonuçlarından da faydalanılarak- testin temel fonksiyonlarının farkındalığının artırılması, böylece test sürecinin oluşması, dolaylı olarak gereksinim, analiz ve tasarım aşamalarındaki kalite problemlerinin test aracılığıyla daha görünür olması ve son kullanıcı testlerinde kullanıcının ürün memnuniyetinin bu sayede artırılması amaçlanmıştır.

C.3. Üçüncü Seviye: Etkili (Effective)

Bir firmanın 'etkili' seviyesinde test olgunluğuna sahip olabilmesi için hem etkin bir test yönetiminin olması hem de DevOps metodolojisine hitap eden test aktivitelerini içermesi gerektiği varsayılmıştır. Bu sebeple TDD, BDD gibi yaklaşımların benimsenmiş, fonksiyonel testlerin yanında kullanılabilirlik, performans, güvenilirlik gibi fonksiyonel olmayan kalite karakteristiklerinin de test ediliyor olması gerekmektedir. Minimum seviyede statik testlerin de gerçekleştirilmesi beklenmektedir. Ayrıca test aktivitelerinin etkin bir şekilde kullanılması için gerekli olan test yönetimi, hata yönetimi gibi araçların kullanılması gerekmektedir. Bu bilgiler doğrultusunda 'etkili' seviyesi için belirlenen kritik alanlar aşağıdaki gibidir:

- **Test Araçları:** Senaryo yönetimi, hata yönetimi, metriklerin yönetimi gibi çeşitli amaçlarla kullanılan test araçlarını ifade etmektedir.
- **Statik Test Teknikleri:** Kod çalıştırılmadan önce analiz, tasarım gibi çeşitli dokümanların gözden geçirilmesini veya kodun gözden geçirilmesini sağlayan test tekniklerini ifade etmektedir.
- **Fonksiyonel Olmayan Testler:** ISO 25010 kapsamında ele alınan ve fonksiyonalite dışındaki kullanılabilirlik, güvenilirlik, verimlilik, sürdürülebilirlik, taşınabilirlik, uyumluluk ve güvenlik kategorilerindeki kalite karakteristiklerinin testlerini ifade etmektedir.
- **Test Güdümlü Geliştirme:** Test güdümlü geliştirme faaliyetlerinin uygulanmasını ifade etmektedir.
- **Davranış Güdümlü Geliştirme:** Davranış güdümlü geliştirme faaliyetlerinin uygulanmasını ifade etmektedir.

Bu kritik alanlar ile birlikte firmanın DevOps metodolojisine uygun olarak test etki gücünü belirlemek amaçlanmıştır. Anket sonuçları, uzman görüşmeleri ve literatür incelendiğinde DevOps test süreçlerine TDD, BDD, kalite karakteristikleri, statik testler ve etkin araç kullanımının sıklıkla uygulanan ve tercih edilen pratikler olduğu görülmüştür.

C.4. Dördüncü Seviye: Optimize (Optimized)

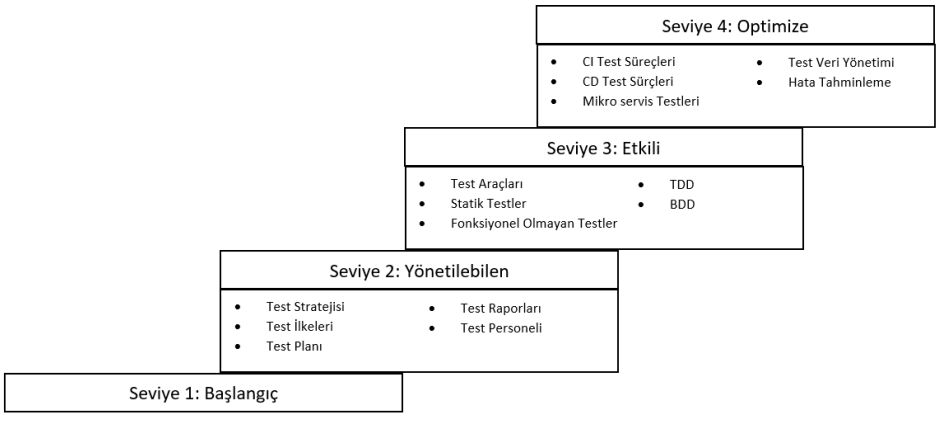
Bir firmanın 'optimize' seviyesinde test olgunluğuna sahip olabilmesi için test süreçlerinin tamamının DevOps ile entegre olabilmesi gerektiği varsayılmıştır. Bu sebeple DevOps yapısındaki sürekli bütünlük sağlamaya yönelik birim, entegrasyon ve kontrat testlerinin otomatik olarak gerçekleştirilebiliyor olması ve sürekli geliştirmeyi sağlamaya yönelik fonksiyonel testlerin gerçekleştirilebiliyor olması gerektiği düşünülmüştür. Test sürecinin alt disiplinlerinden olan test veri yönetimi, mikro servis testlerinde servis sanallaştırma kullanılması, akıllı hata tahminleme yöntemleri etkili olarak kullanılabilmektedir. Bu bilgiler doğrultusunda 'optimize' seviyesi için belirlenen kritik alanlar aşağıdaki gibidir:

- **Sürekli Entegrasyon Test Süreçleri:** Otomasyon ile gerçekleştirilen birim, entegrasyon ve kontrat testlerini ifade etmektedir.
- **Sürekli Teslimat Test Süreçleri:** Otomasyon ile geliştirilen fonksiyonel testleri ifade etmektedir [38].
- **Mikro Servis Test Aktiviteleri:** Mikro servis testleri için uygulanan kontrat testlerini ve gerekli olan durumlarda sanallaştırılmış servislerin kullanılmasını ifade etmektedir [39].
- **Test Veri Yönetimi:** Test yapılırken veri kullanmak için uygulanan maskeleme ve sentetik veri üretimi faaliyetlerini ifade etmektedir [40].
- **Hata Tahminleme:** Makine öğrenmesi teknikleri ile hata tahminleme faaliyetlerinin gerçekleştirilmesini ifade etmektedir [41].

Bu kritik alanlar belirlenirken firmanın optimum bir DevOps test süreci için mutlaka olması gereken faktörlerin belirlenmesi amaçlanmıştır.
Bütün bu seviyelendirme ve kritik değerlendirme alanları ile birlikte Şekil 1'deki gibi bir değerlendirme modeli ortaya çıkmaktadır.

D. YAZILIM TEST SÜREÇ İYİLEŞTİRME MODELİNİN UYGULAMASI

Oluşturulan test süreç iyileştirme modelinin fizibilitesini belirlemek amacıyla bu modelin, Türkiye'de faaliyet gösteren ve DevOps metodolojisini uygulayan bir teknoloji firmasında uygulanmasına karar verilmiştir. Firma yaklaşık 1100 kişilik bilişim personeline sahiptir ve geliştirme prosedürleri çevik yaklaşımları içermektedir. Ekipler çevik yönetime uygun bir şekilde geliştirme ekipleri olarak konumlandırılmıştır. Test süreç iyileştirme modelinde ele alınan değerlendirmeler bu firmanın bir geliştirme ekibi baz alınarak yapılmıştır.



Şekil 1. *DevOps Test Süreç İyileştirme Modeli*

Geliştirme ekibi 3 yazılım mühendisi, 4 analist ve 1 test mühendisi içermektedir. Çevik prosedürlere uygun olarak her sabah günlük toplantılar yapılmakta ve geliştirmeler belirli periyotlar ile düzenli aralıklarla canlı ortama taşınmaktadır. Her periyot sonunda gözden geçirme, retrospektif gibi aktiviteler

gerçekleştirilmektedir. Ekipteki yazılım mühendislerinden biri de DevOps'un gerektirdiği gibi ürünün canlı ortama geçişiyle ilgili operasyonel süreçleri üstlenmektedir.

İlgili ekibe ait test değerlendirmelerinin detayları aşağıda belirtilmiştir.

- **Test Süreci:** Firmanın çevik prosedürleri gereği, geliştirilen ürünün kalite kontrolünü içeren bir süreç bulunmaktadır.
- **Test Personeli:** Ekip içerisinde geliştirilen ürünün yalnızca testlerinden sorumlu bir personel bulunmaktadır.
- **Test Stratejisi:** Geliştirilen ürünün test aktivitelerinde dinamik test tekniklerinden olan sınır değer analizi, denklik paylarına ayırma, kullanım senaryosu gibi teknikler uygulanmaktadır. Bu da testlerin sistematik ve stratejik olduğunu kanıtlayan bir özelliktir.
- **Test İlkeleri:** Ekibin projelerden bağımsız genel test ilkeleri, merkezi test ekibi tarafından belirlenmiştir. Bu ilkelerden bazıları aşağıdaki gibi sıralanabilir:
  - Kalite testlerinde çok fazla hatası olan ürünün fonksiyonel testlerine başlanamaz.
  - Hiçbir ürün fonksiyonel testler tamamlanmadan kullanıcı testine verilemez.
  - Kullanıcı kabul testinden onay almamış bir ürün canlı ortama alınamaz.
- **Test Planı:** Ekibin proje bazında şablon olarak sahip oldukları bir test planı bulunmaktadır. Bu plan içerisinde görev alacak personeller, test edilecek ve edilmeyecek kısımlar, efor planlamaları bulunmaktadır.
- **Test Raporları:** Test mühendisi testlere başladığı andan itibaren paydaşları ürünün kalitesi konusunda çeşitli metriklerle bilgilendirmektedir. Bu metrikler arasında senaryo sayısı, kontrolü tamamlanan ve hatasız olan senaryolar, hatalı senaryolar, kritik hatalar, kozmetik hatalar gibi bilgiler bulunmaktadır.
- **Test Araçları:** Ekipte üç farklı test aracı kullanılmaktadır.

  - Test senaryolarının etkili yönetilebilmesi için ticari bir test yönetimi aracı kullanılmaktadır. Test mühendisi senaryolarını burada tasarlamakta ve koşumlarını bu araç üzerinde gerçekleştirmektedir. Ürün paydaşlarını bu araç üzerinden elde ettiği metrikleri raporlayarak bilgilendirmektedir.
  - Hataların yönetimi için başka bir ticari hata yönetimi aracı kullanılmaktadır. Test mühendisi senaryolarda tespit ettiği hataları bu araç üzerinden raporlayabilmektedir ve bu hataları test yönetimi aracı ile entegre edebilmektedir. Açılan hata raporları bu araç üzerinden analist ve yazılım mühendislerine iletilebilmektedir ve iletişim sağlanabilmektedir.
  - Senaryoların otomatik koşumları için başka bir ticari test otomasyon aracı kullanılmaktadır. Tekrarlı testlerde test mühendisi bu aracı senaryoları otomatik olarak koşabilmek için kullanmaktadır.

- **Statik Test Teknikleri:** Test mühendisi gereksinim belirleme toplantılarına katılmaktadır ve kod çalıştırılmadan önce hatalı durumların önüne geçmek adına review (gözden geçirme) yaparak gerektiğinde geliştirmeye müdahale edebilmektedir. Yazılan analiz dokümanlarında mantıksal hatalar olması durumunda hata raporu açılmasa da sözlü iletişim yoluyla hatanın önüne geçebilmektedir.
- **Fonksiyonel Olmayan Testler:** Ekip içerisinde ve ekip dışındaki personeller yardımıyla ISO 25010'da belirtilen fonksiyonel olmayan kalite karakteristiklerine göre testler yürütülebilmektedir.

  - Kullanılabilirlik testleri firma tarafından belirlenen kullanılabilirlik prensiplerine göre test mühendisi tarafından belirli periyotlarla gerçekleştirilmektedir.
  - Güvenilirlik ve güvenlik testleri farklı ekipler tarafından sürekli olarak koşulmaktadır.
  - Verimlilik testleri ihtiyaç halinde merkezi test ekibi tarafından gerçekleştirilmekte ve sonuçları bu ekip ile incelenmektedir.
  - Sürdürülebilirlik testleri için canlı ortam sürekli olarak takip edilmekte, müşteriler tarafından gelen şikayetlere acil müdahaleler gerçekleştirilmektedir.

- Taşınabilirlik testleri için farklı simüle ortamlara geçişler yapılmaktadır. Böylece canlı ortama geçildiğinde ortam değişimi ile ilgili riskler en aza indirilmektedir.
- Uyumluluk testleri için ürün entegrasyon testleri düzenli aralıklarla ekipte bulunan test mühendisi tarafından gerçekleştirilmektedir.

- **Test Güdümlü Geliştirme ve Davranış Güdümlü Geliştirme:** Yapılan geliştirmelerin gereksinimleri ekibe iş birimi tarafından DSL formatında iletilmektedir. Yazılım mühendisleri iletilen bu format için, test güdümlü geliştirme esaslarına uygun olarak, test senaryolarını hazırlamaktadır ve geliştirme kodunu daha sonra yazmaktadır. Böylece ürün için yazılan test senaryoları başarılı oluncaya kadar ve iş biriminin gereksinimine tam olarak cevap verilinceye kadar geliştirilmektedir ve hata oranı en az seviyeye indirgenmektedir.
- **Sürekli Entegrasyon ve Sürekli Teslimat Test Süreçleri:** Ekip içerisinde yapılan geliştirmenin otomatik olarak testlerini gerçekleştirerek direkt entegrasyonunu sağlayan bir süreç bulunmamaktadır. Sebebi incelendiğinde test mühendisinin kodlama konusunda gelişmeye ihtiyacı olduğu görülmüştür. Geliştirilen ürünler test mühendisi tarafından manuel olarak test edilmektedir.
- **Mikro Servis Test Aktiviteleri:** Ekip içerisinde mikro servis testleri için herhangi bir kontrat test süreci bulunmamaktadır. Servis sanallaştırma faaliyetleri gerektiğinde merkezi test ekibi tarafından sağlanmaktadır.
- **Test Veri Yönetimi:** Kişisel verilerin kullanımına uyumlanılması amacıyla canlı ortamdaki veriler simüle ortamlara maskeleme indirilerek kullanılmaktadır. Bunun dışında merkezi test ekibi tarafından sentetik veri üretimi faaliyetleri ekip ile çalışılarak gerçekleştirilebilmektedir.
- **Hata Tahminleme:** Makine öğrenmesi veya yapay zeka teknikleri ile hataları akıllı bir şekilde tahmin eden robotik bir süreç bulunmamaktadır.

Tablo 12. *Firmaya Ait DevOps Test Geliştirme Modeli Değerlendirme Sonuçları*

Seviye	**Değerlendirme Kriteri**	**Değerlendirme Sonucu**
1. Seviye: Başlangıç	Test Süreci	Başarılı
2. Seviye: Yönetilebilen	Test Personeli	Başarılı
	Test Stratejisi	Başarılı
	Test İlkeleri	Başarılı
	Test Planı	Başarılı
	Test Raporları	Başarılı
3. Seviye: Etkili	Test Araçları	Başarılı
	Statik Test Teknikleri	Başarılı
	Fonksiyonel Olmayan Testler	Başarılı
	Test Güdümlü Geliştirme	Başarılı
	Davranış Güdümlü Geliştirme	Başarılı
4. Seviye: Optimize	Sürekli Entegrasyon Test Süreçleri	Geliştirilmesi Gerekli
	Sürekli Teslimat Test Süreçleri	Geliştirilmesi Gerekli
	Mikro Servis Test Aktiviteleri	Geliştirilmesi Gerekli
	Test Veri Yönetimi	Başarılı
	Hata Tahminleme	Geliştirilmesi Gerekli

Değerlendirme kriterlerinin sonuçları Tablo 12'de gösterildiği gibidir. Tablodan da anlaşılacağı üzere ekibin DevOps test değerlendirme seviyesi %80 üzerinde bir oranda başarı kriterlerini sağladığından dolayı **3. Seviye: Etkili** olarak belirlenmiştir. Bu değerlendirme sonucuna göre bu firmanın etkin bir test yönetimi anlayışının olduğu, DevOps metodolojisine yatkın bir test altyapısına sahip olduğu sonucu çıkarılabilir. Testin optimum noktada uygulanabilmesi için sürekli entegrasyon, sürekli teslimat ve hata tahminleme alanlarının geliştirilmesi gerekmektedir. Bu noktada firma ile görüşüldüğünde uzmanlardan aşağıdaki öneriler alınmıştır.

- **Sürekli Entegrasyon ve Sürekli Teslimat Test Süreçleri İçin Öneriler:** Test mühendislerinin kodlama konusunda geliştirilebilir. Firmaya dahil edilecek yeni test personellerinin kodlama konusunda altyapısının olmasına özen gösterilebilir.
- **Mikro Servis Test Aktiviteleri İçin Öneriler:** Mikro servis mimarisinin ve kontrat test mantığının test mühendisleri tarafından kavranması ve uygulanması için eğitimler düzenlenebilir.
- **Hata Tahminleme İçin Öneriler:** Teknik test mühendisleri makine öğrenmesi ve yapay zeka konularında uzmanlaştırılabilir veya dış kaynak firmaları ile anlaşılarak hata tahminleme faaliyetlerini yürütecek personel kiralanabilir.

Uzmanlar tarafından verilen öneriler firma tarafından yapılacaklar listesine eklenmiştir ve ilerleyen zamanlarda değerlendirilecektir.

IV. TARTIŞMA ve SONUÇ

Bu çalışma ile DevOps metodolojisinde test süreçlerinin olgunluğunu ölçmek amacıyla test süreç iyileştirme modeli tasarlanmıştır. Bu kısımda daha önce geliştirilen modeller ile karşılaştırılması ve modelin uygulamalı değerlendirilmesi yer almaktadır.

Tasarlanan model, daha önce CMMI örnek alınarak geliştirilmiş olan TMMI ve türevleri ile benzerlikler göstermektedir [26][27][28]. Bu modellerde de aşamalı bir yapı söz konusudur. Bir seviyede başarılı olmanın koşulu o seviyenin gereksinimlerinin büyük oranda karşılanmasıdır. DevOps test süreç iyileştirme modelinin bu modellerden en büyük farkı her bir aşamanın kendine özgü kritik değerlendirme alanı içermesidir. Her bir seviyede bu değerlendirme alanlarının %80 oranında uygulanması gerekmektedir. Kritik değerlendirme alanları yaklaşımı, sürekli modellerde faydalanılan bir tekniktir ve TPI Next, CTP gibi sürekli modellerden ilham alınarak yapılmıştır [29]. Sürekli modellerde de aşamalı bir yapı söz konusu değildir. Bu çalışma ile geliştirilen model her iki tipe ait özellikler barındırdığı için hibrit bir yapı olarak nitelendirilmiştir.

DevOps test süreç iyileştirme modeli, daha önce tasarlanan ve yalnızca test süreçlerini değil, bütün DevOps süreçlerinin değerlendirmesini içeren modeller ile test bakımından benzerlik gösterebilmektedir. IBM DevOps Olgunluk Modeli'nde de bu çalışmada tasarlanan modelde olduğu gibi sürekli entegrasyon ve sürekli teslimat test aktiviteleri ön koşul olarak belirtilmiştir ve dört seviyede ele alınmıştır [19]. Fakat IBM modelinde ön koşul olarak belirtilen seviyeler farklılık göstermektedir. Bu durum geliştirme ihtiyaçları ile test ihtiyaçlarının aynı kategori altında değerlendirilmesinden kaynaklanabilir. Eficode Geliştirme Modeli'nin kalite güvencesi kategorisinde ele alınan birim test, statik kod analizi, otomasyon süreçleri gibi ön koşullar bu çalışmada geliştirilen model ile benzerlik göstermektedir [23]. Efficode modeli de dört seviyede ele alınan aşamalı bir modeldir. Diğer yazılım geliştirme modelleri olan Fejiter ve Mohamed tarafından ortaya konan DevOps geliştirme modeli, bu çalışma ile tasarlanan modelden oldukça farklıdır. Mohamed, otomasyon tabanlı ve aşamalı bir olgunluk değerlendirme model sunarken Fejiter kritik değerlendirme alanları sunan ve bu alanlar içerisinde test ile ilgili yalnızca otomasyon karakteristiğini içermektedir [20][24]. Bu çalışma ile Fejiter ve Mohamed'den farklı olarak yalnızca otomasyonu değil bütün test süreçlerini değerlendirmek amaçlanmıştır.

DevOps test süreç iyileştirme modelinin sektörde DevOps test süreçlerini değerlendirmeye yönelik kullanışlı bir model olacağı öngörülmekte ve hibrit yapısıyla kapsayıcı bir değerlendirme sağlayacağı düşünülmektedir. Bir firma ile yapılan örnek uygulamada da işlerliği kanıtlanmış ve firmadan ve tasarım sürecinde katkıda bulunan uzmanlardan olumlu geri bildirimler alınmıştır. Gelecek çalışmalarda bu modelin daha çok firmada uygulanarak analiz edilmesi ve sonuçlarının diğer yayınlar ile zenginleştirilmesi planlanmaktadır.

V. KAYNAKLAR

[1] S. Kiv, S. Heng, M. Kolp, and Y. Watelet, "Agile manifesto and practices selection for tailoring software development: A systematic literature review," *International Conference on Product-Focused Software Process Improvement*, 2018, pp. 12-30.

[2] K. Back, M. Beedle, and A.V. Bennekum. (2021, August 31). *Manifesto for Agile Software Development* [Online]. Available: https://agilemanifesto.org/

[3] O. Kalıpsız, A.B. Olcaysoy, ve G. Biricik, *Bilgisayar Bilimlerinde Sistem Analizi ve Tasarımı*, 4. baskı, İstanbul, Türkiye: Papatya Yayıncılık, 2018, böl. 4, ss. 45-47.

[4] E.B. Swanson, and C.M. Beath, "Departmentalization in Software Development and Maintenance," *Communications of the ACM*, vol. 33, pp. 658–667, 1990.

[5] B. Tessem, and J. Iden, "Cooperation between developers and operations in software engineering projects," *Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering*, Leipzig, Germany, 2008, pp. 105–108.

[6] P.A. Nielsen, T.J. Winkler, and J. Nørbjerg, "Closing the IT Development-Operations Gap: The DevOps Knowledge Sharing Framework," *16th International Conference on Prospectives in Business Informatics Research*. BIR 2017 – Aalborg University, Cophenagen, Denmark, 2017, pp. 1-16.

[7] I. Ozkaya, "The Deployment View," *IEEE Software*, vol. 37, no. 3, pp. 3-5, 2020.

[8] G.B. Ghantous, and A.Gill, "DevOps: Concepts, Practices, Tools, Benefits and Challenges," *Pacific Asia Conference on Information Systems*, Langkawi, Malaysia, 2017, pp. 96.

[9] J. Smeds, K. Nybom, and I. Porres, "DevOps: A Definition and Perceived Adoption Impediments," *International Conference on Agile Software Development*, 2015, pp. 166-177.

[10] M. Virmani, "Understanding DevOps & Bridging the gap from Continuous Integration to Continuous Delivery," *Fifth International Conference on the Innovative Computing Technology (INTECH 2015*), Galcia, Spain, 2015, pp. 78-82.

[11] J. Angara, S. Gutta, and S. Prasad, "DevOps with Continuous Testing Architecture and Its Metrics Model," *Recent Findings in Intelligent Computing Techniques*, Singapore, 2018, pp. 271-281.

[12] F. Erich, C. Amrit, and M. Daneva, "A Qualitative Study of DevOps Usage in Practice," *Journal of Software: Evolution and Process*, vol.29, no. 6, 2017.

[13] B.B.N. França, H. Jeronimo, and G.H. Travassos, "Characterizing DevOps by Hearing Multiple Voices," *30th Brazilian Symposium on Software Engineering* (SBES'16), Maringa, Brazil, 2016, pp. 53–62.

[14] L.E. Lwakatare, T. Kilamo, T. Karvonen, T. Sauvola, V. Heikkila, J. Itkonen, P. Kujava, T. Mikkonen, M. Oivo, and C. Lassenius, "DevOps in Practice: A Multiple Case Study of Five Companies," *Information and Software Technology*, vol. 114, pp. 217-230, 2019.

[15] I. Bucena, and M. Kirikova, "Simplifying the DevOps Adoption Process," *Workshops and Doctoral Consortium co-located with 16th International Conference on Perspectives in Business Informatics Research (BIR 2017)*, Copenhagen, Denmark, 2017, pp. 1-15.

[16] W. Luz, E. Agilar, M.C. Oliviera, C.E.R. Melo, G. Pinto, and R. Bonifacio "An Experience Report on the Adoption of Microservices in Three Brazilian Government Institutions," *XXXII Brazilian Symposium on Software Engineering*, Sao Carlos, Brazil, 2018, pp. 32-41.

[17] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture," *IEEE Software*, vol. 33, no. 3, pp. 42-52, 2016.

[18] M.I. Zarour, M. Alenezi, and N. Alhammad "A Research on DevOps Maturity Models," *International Journal of Recent Technology and Engineering*, vol. 8, no. 3, pp. 4854-4862, 2019.

[19] P. Bahrs. (2021, August 31). *Adopting the IBM DevOps Approach for Continuous Software Delivery* [Online]. Available: https://developer.ibm.com/articles/d-adoption-paths/

[20] S. Mohamed, "DevOps Shifting Software Engineering Strategy-Value Based Perspective," *International Journal of Computer Engineering*, vol. 17, no. 2, pp. 51–57, 2015.

[21] G. Menzel, and A. Macaulay. (2021, August 31). *DevOps - The Future of Application Lifecycle Automation* [Online]. Available: https://www.capgemini.com/2014/12/devops-the-future-of-application-lifecycle-automation/

[22] M. Gasparaite, S. Ragaisis, "Comparison of devops maturity models," *IVUS 2019. Proceedings of the International Conference on Information Technologies*, Kaunas, Lithuania, 2019, pp. 65-69.

[23] P. Laihonen, "Adoption of DevOps Practices in the Finnish Software Industry: an Empirical Study," M.S. thesis, School of Electrical Engineering, Aalto University, Espoo, Finland, 2018.

[24] R. Feijter, S. Overbeek, R. Vliet, E. Jagroep, and S. Brinkkemper, "DevOps Competences and Maturity for Software Producing Organizations," *International Conference on Evaluation and Modeling Methods for System Analysis and Development*, Tallinn, Estonia, 2018, pp. 244–259.

[25] V. Garousi, M. Felderer, and T. Hacaloğlu, "Software Test maturity assessment and test process improvement: A multivocal literature review," *Information and Software Technology*, vol. 85, pp. 16-42, 2017.

[26] T. Foundation. (2021, June 20). *Test Maturity Model Integration* [Online]. Available: https://www.tmmi.org/tmmi-model/

[27] G. Hongying, and Y. Cheng, "A Customizable Agile Software Quality Assurance Model," *The 5th International Conference in New Trends in Information Science and Service Science*, Macao, China, 2011, pp. 382-387.

[28] ATMM. (2021, June 20). *Agile Testing Maturity Model* [Onine]. Available: http://www.agile-testing-maturity-model.com/startseite.html

[29] J. Andersin, "TPI – a model for Test Process Improvement," *Seminar on Quality Models for Software Engineering*, Helsinki, Finland, 2004, pp. 1-13.

[30] Ö.F. Arifoğlu, "Büyük Ölçekli Uygulamaların DevOps Süreçlerine Uyarlanması ve Uygulanması," Yüksek Lisans tezi, Bilgisayar ve Bilişim Mühendisliği, Sakarya Üniversitesi, Sakarya, Türkiye, 2020.

[31] D. Kutluay, "Importance and Effects of Continuous Delivery on Agile Software Development Lifecycle," M.S. thesis, Mangement and Information Systems, Yeditepe University, Istanbul, Turkey, 2018.

[32] S. Arkan, "Assesing the Maturity of Software Testing Services: A Model and Its Industrial Evaluation," M.S. thesis, Computer Engineering, Cankaya University, Ankara, Turkey, 2016.

[33] M. Haues, A. Sellami, H.B. Abdallah, and L. Cheikhi, "A Guideline for Software Architecture Selection Based on ISO 25010 Quality Related Characteristics," *International Journal of System Assurance Engineering and Management*, vol. 8, pp. 886–909, 2017.

[34] K. Beck, *Test-Driven Development by Example*, 1st ed, London, England: Pearson Education, 2003, pp. 123-133.

[35] S. Rose, M. Wynne, and A. Hellesoy, *The Cucumber Book: Behaviour Driven Development for Testers and Developers*, 1st ed., North Carolina, USA: The Pragmatic Bookshelf, 2015, ch.3, pp. 31-45.

[36] D. Marijan, M. Liaaen, and S. Sen, "DevOps Improvements for Reduced Cycle Times with Integrated Test Optimizations for Continuous Integration," *2018 IEEE 42nd Annual Computer Software and Applications Conference*, Tokyo, Japan, 2018, pp. 22-27.

[37] H.G. Gross, and N. Mayer, "Built-In Contract Testing in Component Integration Testing," *Electronic Notes in Theoretical Computer Science*, vol. 82, no. 6, pp. 22-32, 2003.

[38] G. Schermann, D. Schöni, P. Leitner, and H.C. Gall, "Bifrost: Supporting Continuous Deployment with Automated Enactment of Multi-Phase Live Testing Strategies," *17th International Middleware Conference*, Trento, Italy, 2016, pp. 1-14.

[39] S.P. Ma, C.Y. Fan, Y. Chuang, I.H. Liu, and C.W. Lan, "Graph-Based and Scenario-Driven Microservice Analysis, Retrieval, and Testing," *Future Generation Computer Systems*, vol. 100, pp. 724-735, 2019.

[40] R.B. Jain, M. Puri, and U. Jain, "A Data Masking Utility to Secure Sensitive Production Data in Non-Production Environment," *International Journal of Knowledge Engineering and Data Mining*, vol. 6, no. 2, pp. 168-186, 2019.

[41] A. Hammouri, M. Hammad, M. Alnabhan, and F. Alsarayrah, "Software Bug Prediction using Machine Learning Approach," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 2, pp. 78-83, 2018.