



DevOps Dünyasında TMMi® (v1.1)

#TMMiileBaşarı

EDİTÖR: ERIK VAN VEENENDAAL

www.tmmi.org

TMMi, yaşam döngüsünden bağımsız olarak tasarlanmıştır. Bu belge, TMMi test iyileştirme modelinin DevOps bağlamında nasıl kullanılıp faydalı bir şekilde uygulanabileceğini açıklamaktadır. Seviye 2 ve seviye 3 TMMi süreç alanları ve hedefleri tek tek ele alınmaktadır. Bunların DevOps ile nasıl ilişkili olduğu ve uygulamaların genellikle nasıl olacağı belirtilmektedir. Bir uygulamanın DevOps bağlamında gerçekleştirilmesi beklenmiyorsa, çünkü katma değeri yoksa, bu da açıkça belirtilmektedir. Bu belgenin tam bir DevOps test müfredatı olması amaçlanmadığını, daha çok “sadece” TMMi hedeflerinin DevOps bağlamında nasıl yorumlanması gerektiğini gösteren ve daha ileri çalışmalar için fikir ve yönlendirmeler sunan bir belge olduğunu unutmayın. DevOps Dünyasında TMMi de tam TMMi modelinin yerini alan bir açıklama olarak tasarlanmamıştır. DevOps Dünyasında TMMi her zaman TMMi referans model belgesi ile birlikte kullanılmalıdır.

© TMMi Foundation 2011–25

Tüm hakları saklıdır. Bu yayının hiçbir bölümü, ilgili lisans belgelerinde belirtilen istisnalar dışında, TMMi Vakfı'nın önceden izni olmadan herhangi bir şekilde veya herhangi bir yolla tamamen veya kısmen ödünç verilemez, satılamaz, devredilemez, çoğaltılamaz veya iletilemez. İlgili lisans belgelerinin koşulları uyarınca herhangi bir şekilde kopyalanmasına izin verilmesi durumunda, bu bildirim tüm kopyalarda yer alması şartıyla izin verilir.

Ticari marka olduğunu düşündüğümüz kelimeler bu şekilde belirtilmiştir. Ancak, bu tür bir belirtimin varlığı veya yokluğu, herhangi bir ticari markanın yasal statüsünü etkilediği şekilde değerlendirilmemelidir.

TMMi, TMMi Vakfı'nın (İngiltere) tescilli ticari markasıdır.

Katkıda bulunanlar

Katalin Balla (Hungary)

Suresh Chandra Bose (USA)

Kari Kakkonen (Finland)

Mattijs Kemmink (The Netherlands)

Rik Marselis (The Netherlands)

Szilard Szell (Hungary)

Erik van Veenendaal (The Netherlands)

Revizyonlar

Bu bölüm, bu belgenin sürümleri arasındaki önemli değişiklikleri özetlemektedir.

Bu bölüm sadece bilgi amaçlıdır.

Serbest bırakma	Düzeltilme Notları
V1.0	"DevOps dünyasında test etme" belgesinin ilk versiyonu
V1.1	DevOps bağlamında TMMi seviye 3 süreç alanlarının, hedeflerinin ve uygulamalarının yorumlanması eklenmiştir. Ayrıca DevOps'a özgü terimler için bir sözlük eklenmiştir.

Teşekkür

DevOps Dünyasında TMMi® dokümanının Türkçeleştirme çalışmasına katkıda bulunan Yazılım Test ve Kalite Derneği çeviri çalışma grubu üyelerine burada tekrar teşekkür etmek isteriz. Çalışma grubu üyeleri (alfabetik sıraya göre):

- Deniz Onat
- Eda Civar
- L. Koray Yitmen
- Meltem Aydaş
- Özlemnur Bayram Dönmez
- Yazılım Test ve Kalite Derneği, Ocak 2026

Yazılım Test ve Kalite Derneği Hakkında

(Turkish Testing Board – www.turkishtestingboard.org) Yazılım Test ve Kalite Derneği, 2006 yılından bu yana Türkiye bilişim sektöründe yazılım testi farkındalığının artması ve gelişmesi için kâr amacı gütmeyen gönüllü bir şekilde aşağıdaki faaliyetleri gerçekleştirmektedir:

Uluslararası Sertifikasyon Sınavları

Dernek uluslararası ISTQB® sertifika sınavlarını gerçekleştirerek sınavlarda başarılı olan katılımcılara uluslararası geçerliliği olan sertifikalar vermektedir. 2006 yılından bu yana 7.000'den fazla test uzmanı adayı derneğe başvurarak sertifika sınavlarına girmiştir. Dernek bünyesinde düzenlenmekte olan sertifika sınavları:

- ISTQB® Uluslararası Sertifikalı Temel Seviye Yazılım Test Uzmanı Sınavı
- ISTQB® Uluslararası Sertifikalı Temel Seviye Çevik Test Uzmanı Sınavı
- ISTQB® Uluslararası Sertifikalı İleri Seviye – Test Analisti Sınavı
- ISTQB® Uluslararası Sertifikalı İleri Seviye – Teknik Test Analisti Sınavı
- ISTQB® Uluslararası Sertifikalı İleri Seviye – Test Yönetimi Sınavı
- ISTQB® Uluslararası Sertifikalı İleri Seviye – Test Otomasyon Mühendisliği Sınavı
- ISTQB® Uluslararası Sertifikalı Performans Testi Sınavı
- ISTQB® Uluslararası Sertifikalı Yapay Zeka Testi Sınavı
- ISTQB® Uluslararası Sertifikalı Mobil Uygulama Testi Sınavı
- ISTQB® Uluslararası Sertifikalı Otomotiv Yazılımı Testi Sınavı
- ISTQB® Uluslararası Sertifikalı Oyun Testi Sınavı
- ISTQB® Uluslararası Sertifikalı Üretken Yapay Zeka ile Test Sınavı

Uluslararası Testİstanbul Konferansları – www.testistanbul.org

Dernek, 2010 yılından bu yana Uluslararası Testİstanbul Konferanslarını düzenlemektedir. Geçtiğimiz 16 konferansta 40'dan fazla ülkeden, 80'den fazla konuşmacı ve 7.400'den fazla katılımcı ağırlanmıştır.

Paneller

Dernek, yazılım test sektörünün gelişimi için sektör veya konu bazlı paneller organize etmektedir. Bu panellere şu ana kadar 1.000'den fazla profesyonel katılım göstermiştir. Şimdiye kadar düzenlenen paneller:

- TestFintech,
- TestDefence,
- TestGames,
- TestInsurance,
- TestAnkara,
- Testİzmir,
- Test Finance

Turkey Software Quality Report (TSQR)

Dernek tarafından 2011 yılından itibaren yüzlerce bilişim profesyoneli ve akademisyeninin katılımıyla her yıl düzenlenen anket sonuçlarının değerlendirilmesiyle hazırlanan, Türkiye bilişim sektörüne yön verir nitelikte çıkarımların olduğu rapordur. İngilizce yayınlanan rapor tüm ISTQB® üye dernekleri aracılığıyla 100'den fazla ülkedeki bilişim profesyoneline ulaşmaktadır.

ISTQB® Worldwide Software Testing Practices Report (WSTPR)

ISTQB® tarafından 100'den fazla ülkeden binlerce bilişim profesyoneli ve akademisyeninin katılımıyla düzenlenen anket sonuçlarının değerlendirilmesiyle hazırlanan, dünya bilişim sektörüne yön verir nitelikte çıkarımların olduğu rapordur.

Türkçeleştirme Çalışmaları

Uluslararası yazılım test terminolojisinin ülkemize kazandırılması için dernek bünyesinde yer alan gönüllü çeviri grubu ISTQB® dokümanlarının çevirisi üzerinde çalışmaktadır. Şu ana kadar çevrilen dokümanlar:

- ISTQB® Uluslararası Sertifikalı Temel Seviye Yazılım Test Uzmanı Ders Programı v4.0
- ISTQB® Uluslararası Sertifikalı Temel Seviye Yazılım Test Uzmanı Örnek Sınav A v4.0
- ISTQB® Uluslararası Sertifikalı Temel Seviye Yazılım Test Uzmanı Örnek Sınav B v4.0
- ISTQB® Uluslararası Sertifikalı Temel Seviye Yazılım Test Uzmanı Örnek Sınav C v4.0
- ISTQB® Uluslararası Sertifikalı Temel Seviye Yazılım Test Uzmanı Ders Programı 2018
- ISTQB® Uluslararası Sertifikalı Temel Seviye Yazılım Test Uzmanı Ders Programı 2011
- ISTQB® Uluslararası Sertifikalı İleri Seviye – Test Analisti Ders Programı 2012
- ISTQB® Uluslararası Sertifikalı Mobil Uygulama Testi Ders Programı 2019
- ISTQB® Uluslararası Sertifikalı Yapay Zekâ Testi Ders Programı 2021
- ISTQB® Uluslararası Sertifikalı Otomotiv Yazılımı Test Uzmanı Ders Programı 2018
- ISTQB® Yazılım Testi Terimler Sözlüğü
- Bir Ejderhadan Yazılım Test Dersi • Çevik Bir Dünyada TMMi®
- TMMi® Genel Broşürü • TMMi® Lightning Scan Tool dokümanı
- TMMi® Kariyer Yolu Broşürü • TMMi® TPI Broşürü
- TMMi® Professional Sertifikasyon Broşürü • DevOps Dünyasında TMMi®
- Yapay Zekâ Sistemleri Testi ve TMMi®

Burslar

Dernek her yıl karından belli bir miktarı T.C. Üniversitelerinin Bilgisayar/Yazılım Mühendisliği ve Bilgisayar Programcılığı, Yönetim Bilişim Sistemleri ve bununla alakalı bölümlerinde okumakta olan başarılı ve ihtiyaç sahibi öğrencilere burs olarak aktarmaktadır. 2025 yılı itibarıyla burs sağlanan toplam bursiyer sayısı 100'ü geçmiştir.

İçindekiler

Katkıda Bulunanlar	2
Revizyonlar	3
Teşekkür	4
Yazılım Test ve Kalite Derneği Hakkında	5
İçindekiler	7
DevOps Dünyasında TMMi®	9
1.1 Giriş.....	9
1.2 DevOps ve DevSecOps.....	9
1.3 Test Olgunluk Modeli Entegrasyonu (TMMi)	10
1.4 DevOps Bağlamında Test	11
1.4.1 DevOps Bağlamında Test Etmenin Zorlukları	11
1.4.2 TMMi ve DevOps.....	12
1.4.3 ISTQB® ile DevOps Müfredatı Arasındaki İlişki.....	14
TMMi Seviye 2 - Yönetilen	14
2.1 Süreç Alanı 2.1 Test Politikası ve Stratejisi	14
2.1.1 SG1 Test Politikası Oluşturma.....	14
2.1.2 SG2 Test Stratejisi Oluşturma.....	16
2.1.3 SG3 Test Performans Göstergeleri Belirleme	19
2.2 Süreç Alanı 2.2 Test Planlama	20
2.2.1 SG1 Risk Değerlendirmesinin Gerçekleştirilmesi	20
2.2.2 SG2 Test Yaklaşımı Belirleme.....	21
2.2.3 SG3 Test Tahminlerini Belirleme	23
2.2.4 SG4 Test Planı Geliştirme	24
2.2.5 SG5 Test Planına Bağlılığı Sağlama	25
2.3 Süreç Alanı 2.3 Test İzleme ve Kontrol.....	26
2.3.1 SG1 Test İlerlemesini Plana Göre İzleme.....	26
2.3.2 SG2 Ürün Kalitesini Plan ve Beklentilere Göre İzleme	28
2.3.3 SG3 Düzeltici Faaliyetleri Kapanışa Kadar Yönetme	29
2.4 Süreç Alanı 2.4 Test Tasarımı ve Uygulaması	29
2.4.1 SG1 Test Tasarım Teknikleri Kullanılarak Test Analizi ve Tasarımı Gerçekleştirme	30
2.4.2 SG2 Test Uygulamasını Gerçekleştirme.....	31

2.4.3 SG3 Test Yürütme	32
2.4.4 SG4 Test Olaylarını Kapanışa Kadar Yönetme	34
2.5 Süreç Alanı 2.5 Test Ortamı	34
2.5.1 SG1 Test Ortamı Gereksinimlerini Geliştirme	34
2.5.2 SG2 Test Ortamı Uygulamasını Gerçekleştirme	35
2.5.3 SG3 Test Ortamlarını Yönetme ve Kontrol Etme	35
TMMi Seviye 3 - Tanımlanmış	36
3.1 Süreç Alanı 3.1 Test Organizasyonu	36
3.1.1 SG1 Test Organizasyonu Oluşturma	36
3.1.2 SG2 Test Uzmanları için Test Fonksiyonları Oluşturma	37
3.1.3 SG3 Test Kariyer Yollarının Oluşturulması	38
3.1.4 SG4 Test Süreci İyileştirmelerini Belirleme, Planlama ve Uygulama	38
3.1.5 SG5 Organizasyonel Test Sürecini Yaygınlaştırma ve Öğrenilen Dersleri Entegre Etme ..	39
3.2 Süreç Alanı 3.2 Test Eğitim Programı	40
3.2.1 SG1 Organizasyonel Test Eğitim Yetkinliği Oluşturma	40
3.2.2 SG2 Gerekli Test Eğitimlerini Sağlama	40
3.3 Süreç Alanı 3.3 Test Yaşam Döngüsü ve Entegrasyon	41
3.3.1 SG1 Organizasyonel Test Süreci Varlıklarını Oluşturma	41
3.3.2 SG2 Test Yaşam Döngüsü Modellerinin Geliştirme Modelleri ile Entegrasyonu	43
3.3.3 SG3 Ana Test Planı Oluşturma	43
3.4 Süreç Alanı 3.4 Fonksiyonel Olmayan Testler	44
3.4.1 SG1 Fonksiyonel Olmayan Ürün Risk Değerlendirmesi Yapma	44
3.4.2 SG2 Fonksiyonel Olmayan Test Yaklaşımı Oluşturma	45
3.4.3 SG3 Fonksiyonel Olmayan Test Analizi ve Tasarımı Gerçekleştirme	45
3.4.4 SG4 Fonksiyonel Olmayan Test Uygulamasını Gerçekleştirme	47
3.4.5 SG4 Fonksiyonel Olmayan Testleri Yürütme	47
3.5 Süreç Alanı 3.5 Eş Gözden Geçirmeler	48
3.5.1 SG1 Eş Gözden Geçirme Yaklaşımı Oluşturma	48
3.5.2 SG2 Eş Gözden Geçirmeleri Gerçekleştirme	50
DevOps'a Özgü Terimler Sözlüğü	52
Kaynakça	54

1 DevOps Dünyasında TMMi®

1.1 Giriş

Bu belge, yazılım geliştirme için DevOps yaklaşımını TMMi test süreci iyileştirme modeliyle birleştirmenin iş hedeflerinize ulaşmak için nasıl bir yol olabileceğini açıklamaktadır. TMMi'nin DevOps bağlamında nasıl yararlı bir şekilde kullanılabileceğini ve uygulanabileceğini ele almaktadır. TMMi uygulamalarını hayata geçiren geleneksel test yaklaşımlarına, kanıtlanmış DevOps tabanlı alternatifler sunulmaktadır. Bu belge, kuruluşların DevOps yolculuğunda ne kadar olgun olduklarına bakılmaksızın, TMMi'nin DevOps kuruluşlarına nasıl yardımcı olabileceğini ele almaktadır. Kuruluşlar büyüdükçe ve proje baskısı arttıkça sıklıkla gözden kaçan kritik test uygulamalarını hatırlatarak bu yardımı sağlamaktadır. Her bir TMMi süreç alanı ve belirli hedefleri tek tek ele alınmaktadır. Bunların DevOps ile nasıl ilişkili olduğu ve ilgili uygulamaların genellikle nasıl olacağı belirtilmektedir. Bir uygulamanın DevOps bağlamında gerçekleştirilmesi beklenmiyorsa, çünkü herhangi bir katma değeri yoksa, bu da açıkça belirtilmelidir.

Sürekli entegrasyon, sürekli teslimat ve otomasyonun yazılım geliştirmeyi yönlendirdiği günümüzün hızlı tempolu DevOps dünyasında, her aşamada kaliteyi sağlamak çok önemlidir. Test Olgunluk Modeli Entegrasyonu (TMMi), test uygulamalarını değerlendirmek ve iyileştirmek için yapılandırılmış bir çerçeve sağlar. DevOps ekipleri kaliteden ödün vermeden hızlı sürümler yayınlamak için efor harcarken, TMMi'yi entegre etmek test olgunluğunu artırmak, iş hedefleriyle uyum sağlamak ve sonuçlar sunmak için bir yol haritası sunar. TMMi ve DevOps arasındaki bu sinerji, yüksek kalite ve güvenilirlik standartlarını korurken geliştirmeyi hızlandırmak için dengeli bir yaklaşım oluşturur.

Bu belgenin çeşitli hedef kitleleri bulunur. Başlıca gruplar şunlardır:

- Süreç olgunluğunu koruyarak DevOps'a geçmek isteyen olgun geleneksel kuruluşlar
- Test olgunluğunu artırmak için TMMi'yi uygulamaya başlamak ve paralel olarak DevOps'a geçmek isteyen daha az olgun kuruluşlar
- Yapılandırılmış bir Çevik test sürecine sahip olan ve şimdi DevOps'a geçmek isteyen Çevik organizasyonlar
- Başarılı ve büyümekte olan DevOps kuruluşları. Sonuçta, Çevik olmanın avantajlarını korurken aynı zamanda bir miktar süreç olgunluğuna ihtiyaç duyarlar.

Bu belge, orijinal TMMi modelinden bağımsız olarak kullanılmak üzere tasarlanmamıştır. DevOps ortamında test süreci iyileştirmesi gerçekleştiren kişilere, çeşitli TMMi hedefleri ve uygulamalarının yorumlanması ve anlamı konusunda rehberlik sağlamak amacıyla hazırlanmıştır. Bu belge, orijinal TMMi modelini tamamlayıcı niteliktedir ve tamamlayıcı bir belge olarak kullanılmalıdır.

1.2 DevOps ve DevSecOps

DevOps, yazılım geliştirme ve BT operasyon ekipleri arasındaki uçurumu kapatmayı amaçlayan bir felsefe ve uygulama setidir. İşbirliği ve ortak sorumluluk kültürünü teşvik ederek, DevOps yazılımın geliştirilmesini, dağıtımını ve bakımını kolaylaştırmayı amaçlamaktadır. Temel fikir, geliştirme ve

operasyonlar arasındaki geleneksel engelleri yıkarak yazılımın daha verimli ve etkili bir şekilde teslim edilmesini sağlamaktır. DevOps, özünde otomasyon ve sürekli süreçleri vurgular. Otomasyon, manuel görevleri en aza indirerek hataları (error) azaltmak ve hızı artırmak için çok önemli bir rol oynar. Bu, Sürekli Entegrasyon (CI) ve Sürekli Dağıtım (CD) uygulamaları yoluyla kod değişiklikleri, testler ve dağıtım süreçlerinin entegrasyonunun otomatikleştirilmesini içerir. Bir diğer önemli vurgu ise, iş hedefleri ve müşterilerle yakın uyum içinde, özerklik, liderlik, iş birliği, sık denemeler, daha kısa geliştirme döngüleri, daha yüksek dağıtım sıklığı ve daha güvenilir sürümler içeren DevOps kültürüne geçişin gerekliliğidir.

DevOps yaklaşımı, pazara daha hızlı giriş, ekipler arasında daha iyi iş birliği, daha yüksek yazılım kalitesi ve güvenilirliği ile daha fazla ölçeklenebilirlik ve esneklik gibi birçok fayda sağlar. Sorunların daha hızlı tespit edilmesini ve çözülmesini sağlayan DevOps, aynı zamanda kesinti sürelerini azaltır ve sistem dayanıklılığını artırır. Sonuç olarak, DevOps teknoloji ve süreçler kadar kültürel değişimle de ilgilidir. Hem geliştirme hem de operasyon ekiplerinin ortak hedefler doğrultusunda çalışması, geri bildirimleri benimsemesi ve sürekli iyileştirme eforu içinde olması için zihniyet değişikliği gerektirir. Bu unsurları entegre ederek DevOps, kuruluşların yazılımları daha verimli, güvenilir ve güvenli bir şekilde sunmasına yardımcı olur.

DevSecOps

DevSecOps, Geliştirme, Güvenlik ve Operasyonlar'ın kısaltmasıdır ve güvenlik uygulamalarını DevOps sürecine entegre eden bir yaklaşımdır. Bu metodoloji, güvenliğin ilk geliştirmeden dağıtım ve ötesine kadar tüm BT yaşam döngüsü boyunca ortak bir sorumluluk olmasını sağlamayı amaçlamaktadır. DevSecOps, güvenli yazılımları daha hızlı ve daha verimli bir şekilde sunarak güvenlik ihlallerinin riskini azaltmayı amaçlamaktadır.

1.3 Test Olgunluk Modeli Entegrasyonu (TMMi)

TMMi çerçevesi, test süreçlerinin iyileştirilmesi için bir kılavuz ve referans çerçevesi olarak TMMi Vakfı tarafından geliştirilmiştir ve test yöneticileri, test mühendisleri, geliştiriciler ve yazılım kalitesi uzmanları için önemli olan konuları ele almaktadır. Test, TMMi içinde en geniş anlamıyla, tüm yazılım ürünü kalitesi ile ilgili faaliyetleri kapsayacak şekilde tanımlanmıştır. TMMi, süreç değerlendirme ve iyileştirme için olgunluk seviyeleri kavramını kullanır. Ayrıca, süreç alanları, hedefler ve uygulamalar da belirlenir. TMMi olgunluk kriterlerinin uygulanması, test sürecini iyileştirecek ve ürün kalitesi, test mühendisliği verimliliği ve döngü süresi eforu üzerinde olumlu bir etkiye sahip olduğu gösterilmiştir. TMMi, kuruluşların test süreçlerini değerlendirme ve iyileştirme konusunda destek olmak amacıyla geliştirilmiştir.

TMMi, süreç iyileştirme için aşamalı bir mimariye sahiptir. Bir kuruluşun test süreci, geçici ve yönetilmeyen bir süreçten yönetilen, tanımlanmış, ölçülen ve optimize edilmiş bir sürece doğru ilerlerken geçirdiği aşamaları veya seviyeleri içerir. Her aşamaya ulaşmak, o aşamadaki tüm hedeflerin gerçekleştirildiğini ve bir sonraki aşamanın temeli için iyileştirmelerin yapıldığını garanti eder. TMMi'nin iç yapısı, sistematik bir şekilde öğrenilip uygulanabilen ve aşamalı olarak gelişen yüksek kaliteli bir test sürecini destekleyen zengin test uygulamaları içerir. TMMi'de, olgunluk hiyerarşisini ve test süreci iyileştirme yolundaki evrimsel süreci belirleyen beş seviye vardır. Her seviye, bir kuruluşun o seviyede

olgunluğa ulaşmak için uygulaması gereken bir dizi süreç alanına sahiptir. TMMi'nin her olgunluk seviyesi için süreç alanları Şekil 1'de gösterilmektedir.

TMMi'nin temel ilkelerinden biri, çeşitli yaşam döngüsü modelleri ve ortamlarına uygulanabilen genel bir model olmasıdır. TMMi tarafından tanımlanan çoğu hedef ve uygulama, Çevik dahil olmak üzere hem sıralı hem de yinelemeli yaşam döngüsü modellerine uygulanabilir olduğu gösterilmiştir. Ancak, modelin en alt düzeyinde, sağlanan alt uygulamaların ve örneklerin çoğu, uygulanan yaşam döngüsü modeline bağlı olarak (çok) farklıdır. TMMi'de yalnızca hedeflerin zorunlu olduğunu, uygulamaların zorunlu olmadığını unutmayın. TMMi, TMMi Vakfı'nın web sitesinde ücretsiz olarak mevcuttur ve ayrıca basılı kitap formatında da mevcuttur. Model, İspanyolca, Fransızca ve Çince dahil olmak üzere birçok dile çevrilmiştir.



Şekil 1: TMMi olgunluk seviyeleri ve süreç alanları

1.4 DevOps Bağlamında Test

1.4.1 DevOps Bağlamında Test Etmenin Zorlukları

DevOps bağlamında test yapmak, sürekli entegrasyon ve sürekli teslimat (CI/CD) süreçlerinin doğası, geliştirme ve operasyon ekipleri arasındaki iş birliği ve geliştirme döngülerinin hızlı temposu nedeniyle benzersiz zorluklar ortaya çıkarır. Test yapmanın başlıca zorluklarından bazıları şunlardır:

- *Sürekli Test*: CI/CD süreçlerine ayak uydurmak için testler hızlı ve sık bir şekilde gerçekleştirilmelidir. Otomatik testlere büyük ölçüde güvenilmektedir ve bu da sağlam ve güvenilir test komut dosyaları ve çerçeveleri gerektirmektedir.

- *Ortam Yönetimi:* Uygun test ortamı yönetiminin sağlanması ve sürekli testler için gerçekçi test ortamlarının hazırlanması karmaşık olabilir. Üretim ve test ortamları arasında genellikle eşitsizlik vardır ve bu da tespit edilemeyen sorunlara yol açabilir. Bir başka zorluk da, birden fazla ortamda tutarlı yapılandırmaların yönetilmesi ve sürdürülmesidir. Ortamlar, genellikle talep üzerine otomatik olarak dağıtılmalı ve yapılandırılmalı, testlerden sonra da silinmelidir.
- *Test Verileri Yönetimi:* Test verileri de üretim verileri gibi olmalı, tutarlı olmalı ve talep üzerine dağıtılmalı, ancak gizlilik kurallarına uyum da bir zorunluluktur.
- *Entegrasyon ve Regresyon Testi:* Karmaşık karşılıklı bağımlılıkları olan ve bu nedenle test edilmesi zor olan birden fazla bileşen ve servisi entegre etmek. Ayrıca, yeni değişikliklerin mevcut fonksiyonelliği bozmamasını sağlamak, bu da kapsamlı regresyon test paketleri gerektirir.
- *Araçlar ve Altyapı:* Çeşitli test araçlarını CI/CD süreçleri ve diğer DevOps araçlarıyla entegre etmek ve test altyapısının artan yük ve karmaşıklığı kaldırabilecek şekilde ölçeklenebilir olmasını sağlamak.
- *Kültürel ve Organizasyonel Zorluklar:* Geleneksel olarak birbirinden bağımsız ve çoğu zaman çelişkili teşviklerle çalışan geliştirme, test ve operasyon ekipleri arasında iş birliğini teşvik etmek. Ekip üyelerinin otomatik test, sürekli entegrasyon ve dağıtım uygulamalarını yürütmek için gerekli becerilere sahip olmasını sağlamak.
- *Performans Testi:* Kontrollü bir test ortamında gerçek dünya yüklerini ve koşullarını simüle etmekle kalmayıp, performansı sürekli olarak izlemek ve optimize etmek için performans testini CI/CD süreçlerine entegre etmek.
- *Güvenlik Testi:* Hızlı ilerleyen süreçlerde, güvenlik testleri genellikle ikinci plana atılır. Geliştirme sürecini yavaşlatmadan güvenlik testlerini (DevSecOps) entegre etmek ve güvenlik açıklarının erken tespit edilmesini sağlamak, birçok ekip için zorlu bir görevdir. Güvenlik testleri, tercihen erken aşamada ve geliştirme döngüsü boyunca otomatik araçlar kullanılarak statik ve dinamik güvenlik testleri gerçekleştirilecek şekilde entegre edilmelidir.
- *Geri Bildirim ve Raporlama:* Test sonuçlarına dayalı olarak ekibe hızlı, net ve uygulanabilir geri bildirim sağlamak. Yapının kalitesi hakkında içgörüler sunan kapsamlı raporlar üretmek, ancak bu süreci Çevik yaklaşımla uyumlu tutmak.
- *Shift left uygulamaları:* Görevleri yaşam döngüsünün mümkün olduğunca erken aşamalarında sola kaydırarak kaliteyi artırın. Sola kaydırma testi, örneğin (kod) incelemeleri, statik analizler ve kapsamlı birim testleri yoluyla, sürecin daha erken aşamalarında test yapmak anlamına gelir.
- *Shift right uygulamaları:* Üretime mümkün olduğunca sık değişiklikler uygulamak çok önemli olduğundan, test stratejisi CI (Sürekli Entegrasyon) bölümündeki test faaliyetleri ile CD (Sürekli Teslimat)'deki üretimde sistem kullanımını gözlemlene arasında bir denge sağlamalıdır.
- *İzleme ve Geri Bildirim Döngüleri:* DevOps'ta testler, dağıtım ile sona ermez; sürekli izleme ve üretim ortamlarından geri bildirim toplama çok önemlidir. Etkili izleme araçları kurmak ve üretimden gelen verileri yorumlayarak kusurları yakalamak ve performansı optimize etmek genellikle göz ardı edilir veya yeterince geliştirilmez.

1.4.2 TMMi ve DevOps

TMMi ve DevOps yaklaşımlarının birbiriyle çeliştiği yönündeki inanış yanlıştır. DevOps yaklaşımları ve TMMi sadece bir arada var olmakla kalmaz, başarılı bir şekilde entegre edildiğinde önemli faydalar da sağlar. Testlere farklı bir bakış açısıyla yaklaşmak, testleri geliştirme sürecine tam olarak entegre etmek,

CI/CD süreçlerine sahip olmak ve bunların “test” iyileştirme programı bağlamında ne anlama geldiğini anlamak da bir başka zorluktur. TMMi'nin “i” harfinin, testlerin yazılım geliştirmenin entegre bir parçası olması ve tamamen ayrı bir şey olarak ele alınmaması gerektiğini ifade ettiğini unutmayın. TMMi modelini kullanmak, DevOps bağlamında sıklıkla “unutulan” kritik test uygulamalarını (ör. ürün risk analizi) hatırlatır. Bu belge, TMMi ve DevOps yöntemlerinin etkili bir şekilde birlikte çalışabileceğini örneklerle gösterecektir.

TMMi'yi uygularken, TMMi modelinin amacının bir kuruluşa bir dizi uygulamayı “dayatmak” olmadığı ve “uygunluğu kanıtlamak” zorunda olunan bir standart olarak uygulanmayacağı dikkate alınmalıdır. Uygun şekilde kullanıldığında, TMMi, iş hedefleri göz önüne alındığında değişimin değer sağlayabileceği belirli test alanlarını belirlemenize yardımcı olabilir. Bu, uygulanan yaşam döngüsü modelinden bağımsız olarak geçerlidir. TMMi uygulamalarının beklenen bir bileşen olduğunu, ancak tanımlanmış bir TMMi uygulamasına göre “alternatif” uygulama olarak da gerçekleştirilebileceğini her zaman hatırlamak önemlidir. Her zaman uygulamanın amacının ne olduğunu, gerekçesinin ne olduğunu ve işletmeye nasıl değer kattığını düşünün. DevOps ortamında genellikle amaç, alternatif bir uygulama yoluyla zaten gerçekleştirilmiştir. Tipik olarak, “herhangi bir” çözüm, iş ihtiyaçları tarafından yönlendirildiği sürece uyumludur! TMMi'yi kullanırken çok kuralcı olmayın, TMMi'nin ilk amacı bu değildi. TMMi hedeflerini ve uygulamalarını her zaman kendi durumunuzun bağlamına göre yorumlayın. Genel olarak, önce belirli iş bağlamanız içindeki süreç ihtiyaçlarını belirleyerek, süreç iyileştirme önceliklerini nasıl odaklayacağınız ve yönlendireceğiniz konusunda kararlar alınabilir.

TMMi ve DevOps arasında ortaya çıkan çatışmaların çoğu, ya “iyi uygulama”nın nasıl uygulanması gerektiği şeklindeki tarihsel TMMi görüşüne ya da DevOps çalışma şekli ve değerlerini nasıl desteklemeleri gerektiğine dayanan DevOps uygulamalarının mantığının yanlış anlaşılmasına dayanmaktadır. (Baş) değerlendiriciler dahil olmak üzere TMMi uzmanları, “iyi bir TMMi uyumlu” uygulamanın nasıl olması gerektiği konusunda yanlışlıkla paylaşılmış olabilecek mesajları yeniden gözden geçirmeli ve muhtemelen yeniden düşünmelidir. DevOps yaklaşımları TMMi süreçleriyle birlikte uygun şekilde uygulandığında, bu durum test uygulamalarının silinmesine değil, etkili bir şekilde uygulanmasına yol açacaktır. TMMi içindeki belirli (test) uygulamalarına ek olarak, genel uygulamalar da olduğunu unutmayın. Genel uygulamaların amacı, bir süreç alanının kurumsallaşmasını desteklemektir. Bu, etkili bir şekilde, yeni kişiler geldiğinde veya kuruluş içinde başka değişiklikler olduğunda, kuruluşun süreç alanını destekleyecek bir altyapıya sahip olmasını sağlamak anlamına gelir.

Geleneksel yazılım geliştirme yöntemlerinden DevOps'a geçiş, günümüzde tanımlandığı şekliyle süreçleri budama ve yalınlaştırma girişimini de beraberinde getirebilir. Bu şekilde, TMMi tabanlı kuruluşlar, DevOps'ta mevcut olan çevik ve yalın düşünme biçiminden faydalanacaktır. İnsanların TMMi modeline olmayan şeyleri ekleme ve böylece gereksiz, katma değer yaratmayan süreçler ve iş ürünleri oluşturma eğilimi olmuştur. Köklere ve iyileştirme hedeflerine geri dönerek ve TMMi modelini amaçlandığı gibi kullanarak, gerçek katma değer yaratan süreçlerin gerçek süreç ihtiyaçları ve hedefleriyle uyumunu destekleyebilirsiniz. Çevik/yalın bir zihniyetle süreçleri budamak ve yalınlaştırmak, insanların gerçekten ne yaptığını yansıtan süreçler ortaya çıkaracak ve yalnızca kullanılan verilerin toplanmasını sağlayacaktır. Bu zihniyet aynı zamanda işleri olabildiğince basit tutmaya odaklanmayı da beraberinde getirecektir ki bu genellikle kolay değildir, ancak TMMi uygulamasını uygulayanlar için fayda sağlayacaktır. DevOps'taki iyileştirmeler, Değişiklik ön süresi, Dağıtım sıklığı, Değişiklik başarısızlık yüzdesi ve Başarısız dağıtım kurtarma süresi [DORA16], [DORA24]

DORA performans metrikleri doğrultusunda, genellikle hızlı hareket edebilen küçük, yetkilendirilmiş ekipler aracılığıyla gerçekleşir; bu da TMMi'nin DevOps'tan faydalanabileceği bir diğer yoldur. TMMi kapsamlı olsa da kuruluşlar her TMMi unsurunu uygulayamaz. Başarılı olmak için, kuruluşlar kendi temel test uygulamalarını ve öncelik ve odaklanma gerektiren iyileştirmeleri belirlemelidir.

1.4.3 ISTQB® ile DevOps Müfredatı Arasındaki İlişki

TMMi ve ISTQB, Yazılım Test Uzmanlığı mesleğini birlikte daha fazla tanıtmak için bir ittifak kurmuştur. Ayrıca, bu belgenin geliştirilmesi sırasında, TMMi teknik komitesi ve ISTQB “DevOps'ta Kalite” müfredat çalışma grubu birlikte çalışmıştır. TMMi bir test iyileştirme modelidir ve bu nedenle TMMi modelinin yapısına dayalı önceliklerle önceden tanımlanmış bir iyileştirme yaklaşımı sunar. Bu belgede, DevOps bağlamında çalışanlar için TMMi iyileştirme hedeflerinin yorumlanması üzerinde durulmaktadır. İyi DevOps mühendislik uygulamalarının ayrıntılı ve tam bir açıklaması sunulmamaktadır. ISTQB “DevOps'ta Kalite” müfredatı, içerik tabanlı bir belgedir ve bu nedenle, kalite mühendisliği (test dahil) uygulamaları gibi iyi DevOps mühendislik uygulamalarının ayrıntılı bir açıklamasını sunmayı amaçlamaktadır. Bu nedenle, iki belge özünde birbirini tamamlayıcı niteliktedir: TMMi neyin iyileştirilmesi gerektiğini (hangi süreçlerin) belirtirken, ISTQB DevOps müfredatı işlerin nasıl yapılması gerektiğini (mühendislik en iyi uygulamaları) açıklamaktadır.

2 TMMi Seviye 2 – Yönetilen

2.1 Süreç Alanı 2.1 Test Politikası ve Stratejisi

Test Politikası ve Stratejisi süreç alanının amacı, test politikası ve test faaliyetlerinin (örneğin test türleri ve test seviyeleri) açık bir şekilde tanımlandığı, kuruluş genelinde veya program genelinde bir test stratejisi geliştirmek ve oluşturmaktır. Test performansını ölçmek, test faaliyetlerinin değerini belirlemek ve iyileştirilmesi gereken alanları ortaya çıkarmak için test performans göstergeleri kullanılır.

2.1.1 SG1 Test Politikası Oluşturma

Test iyileştirme projesine başlayan her kuruluş, öncelikle bir test politikası tanımlamalıdır. Test politikası, kuruluşun test ve test uzmanları ile ilgili genel test hedeflerini, amaçlarını ve stratejik görüşlerini tanımlar. Test politikasının, kuruluşun genel iş ve kalite politikası ile uyumlu olması önemlidir. Test iyileştirmeleri, açık iş hedefleri tarafından yönlendirilmelidir ve bu hedefler de test (iyileştirme) politikasında belgelenmelidir. Test politikası, bir kuruluş içindeki tüm paydaşlar arasında test ve test hedefleri konusunda ortak bir görüşe ulaşmak için gereklidir. Bu ortak görüş, kuruluş genelinde test (süreç iyileştirme) faaliyetlerini uyumlu hale getirmek için gereklidir. Test hedeflerinin asla kendi başlarına hedefler olmaması, çalışan yazılım ve ürün kalitesini sağlamak için daha üst düzey hedeflerden türetilmesi gerektiği unutulmamalıdır.

Yukarıdaki durum, DevOps kültürünü benimsemiş ve DevOps yazılım geliştirme uygulamalarını benimseyen kuruluşlar için de geçerlidir. Nitekim, birçok kuruluşta DevOps yazılım geliştirmede testlerin değişen rolü, testlerin bağımsızlığı, test otomasyonu ve profesyonel test uzmanları hakkında çok sayıda tartışma yapılmaktadır. Bu konular ve diğerleri genellikle yönetim ve diğer paydaşlarla yapılan tartışmalarda ele alınması ve test politikasında belgelenmesi gereken konulardır. DevOps uygulayanlar da dahil olmak üzere, test iyileştirme projesi başlatmak isteyen her kuruluş, böyle bir girişimin iş itici güçlerini ve ihtiyaçlarını belirlemeli ve tanımlamalıdır. Aksi takdirde, neden bir iyileştirme projesi başlatılır ki? Gerçek iş ihtiyaçlarını belirlemek için zaman ayırarak, (test) iyileştirme önceliklerinin nereye odaklanacağına, örneğin hangi süreç alanına odaklanacağına karar vermek için bir bağlam sağlanabilir. Test politikası genellikle organizasyon düzeyinde bir sayfalık yalın bir belge, web sayfası veya duvar tablosudur, proje düzeyinde bir belge değildir.

TMMi'nin özel hedefi olan, belirli uygulamaları da içeren bir Test Politikası Oluşturmak, DevOps yazılım geliştirme uygulayan kuruluşlar için tamamen geçerlidir. Ancak DevOps'ta, Geliştirme, Test ve Operasyonlar genellikle aynı süreç kapsamında ele alınır veya en azından birbirleriyle sıkı bir şekilde uyumludur. Bu nedenle, Test Politikası'nın da genel DevOps politikasının ayrılmaz bir parçası olması ve sadece testten çok kalite mühendisliğine odaklanması beklenir. DevOps'ta test, Shift Left ve Shift Right yaklaşımları arasında denge kurularak, iş birliğine dayalı uygulamalar ve "Test First" düşüncesi kullanılarak düzenlenir. DevOps'ta test faaliyetleri, DevOps El Kitabı'nda tanımlandığı gibi üç yolu da (akış, geri bildirim ve sürekli deneme ve öğrenme) desteklemelidir.

Belirtildiği gibi, test hedefleri Shift Left ve Shift Right arasında net bir dengeye odaklanmalıdır; hataları yakalamak için dağıtımdan önce (CI/CD iş hattı içinde ve çevresinde) gerekli kalite ve test faaliyetlerini oluşturmak için proaktif, tasarım ve test öncelikli düşünce ve bir hata kaçarsa üretimde servis kesinti süresini azaltmak için reaktif önlemler, örneğin kesinti süresini azaltmak için otomatik geri alma mekanizması, mavi-yeşil dağıtım (blue-green deployment) veya etkiyi azaltmak için kademeli (canary) sürüm. Test faaliyetlerinin amacı, kusurlu bir yazılımın üretime dağıtılma olasılığını azaltmak olsa da dağıtım stratejileri, herhangi bir arızanın etkisini azaltmak için uygulanır ve bu da genel kalitenin artmasına neden olur. Etkiyi azaltmak için, dağıtım iş hattının test faaliyetleri, akışı ve teslim sürelerini iyileştirmek için optimize edilmeli, ancak genel risk seviyeleri düşük tutulmalıdır.

Yazılım geliştirme sürecinde yapılan testlerin yanı sıra, Shift Right (üretimde test etme) de DevOps'ta yaygın bir uygulamadır ve bu, politikada tanımlanan test hedeflerinin tanımını da etkileyecektir. Bu uygulama, telemetri bilgilerine dayalı olarak üretimdeki sorunlara hızlı tepki vermeye odaklanarak, kesinti süresini azaltır, kullanılabilirliği artırır ve böylece kaliteyi iyileştirir. Site Güvenilirlik Mühendisliği (SRE) uygulamaları, sistemlerin, servislerin ve uygulamaların durumunun izlenmesini ve durum sorunlarına/sapmalara tepki vermek için kapalı döngü otomasyon mekanizmaları oluşturulmasını kapsar [Beyer]. Bu, kendi kendini iyileştirme olarak da adlandırılır.

DevOps'ta beklenen değişikliklerin sayısı nedeniyle, test hedefleri hem fonksiyonel hem de fonksiyonel olmayan yönlerden regresyon testlerine yoğun bir şekilde odaklanmaktadır. Bazı fonksiyonel olmayan yönler, dağıtımdan sonra üretimde (Shift Right) ölçülebilirken, diğerleri (örneğin güvenlik) üretime geçmeden önce test edilmelidir.

Test hedeflerine örnekler:

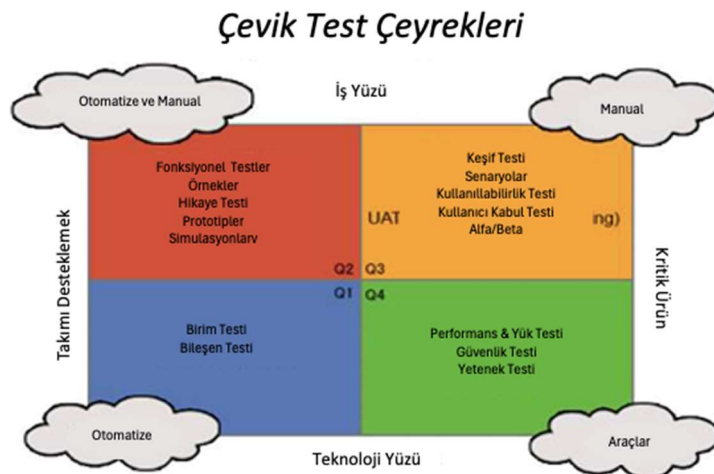
- Test, değişikliklerin beklenen değeri sağladığından (kabul testi) ve mevcut fonksiyonelliği bozmadığından (regresyon testi) emin olmaya odaklanır.
- Testler, kuruluşun değişiklik arıza oranının azaltılması veya dağıtım sıklığının artırılması gibi kuruluşun geliştirme hedeflerini desteklemelidir.

Ayrıca, bağımsızlık ve test politikası başına üst düzey süreç tanımı, DevOps bağlamında genellikle etkilenir. Bağımsızlık açısından, test etme, geliştirme organizasyonunda bir roldür ve çoğu zaman çift rolü temsil etmek için DevTest olarak adlandırılır. Bu nedenle, bağımsızlık genellikle düşüktür. Organizasyonlar genellikle, DevTest rollerinin kullanacağı ürün ve test otomasyon çözümlerinin test edilebilirlik yönlerini geliştirmek ve sürdürmek olan ana rolüne sahip Yazılım Geliştirme Mühendisleri (SDET) ile sözde platform ekibine güvenirlir. Ek olarak, fonksiyonel olmayan testler veya karmaşık uçtan uca senaryolar gibi daha karmaşık alanlar için bazı özel test ekipleri atanabilir. Bu durumda bağımsızlık daha yüksektir, ancak yine de iş birliği sıklıdır. Üst düzey test süreci tanımına gelince, testler artık geliştirme sürecinin veya CI/CD stratejisinin ayrılmaz bir parçasıdır, test süreci de öyle olacaktır.

Genellikle hafif/yalın bir belge olan test politikasının tüm organizasyona / değer akışına / Çevik sürüm trenine / ekibe dağıtılmasını sağlamak önemlidir. Testler, Çevik ve DevOps ekiplerinin çalışmalarının önemli bir parçası olduğundan ve ayrı bir organizasyon olmadığından, politika tüm ilgili ve etkilenen kişiler tarafından bilinmelidir.

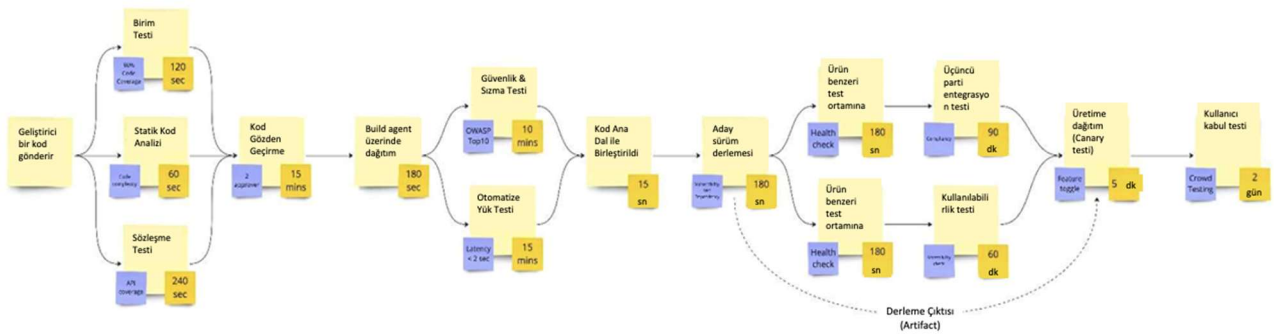
2.1.2 SG2 Test Stratejisi Oluşturma

Test stratejisi, test politikasını takip eder ve projelerdeki test faaliyetleri için bir başlangıç noktası görevi görür. Test stratejisi genellikle organizasyon genelinde veya program genelinde tanımlanır. Tipik bir test stratejisi, üst düzey bir ürün risk değerlendirmesine dayanır ve gerçekleştirilecek test türleri ve test seviyelerinin bir açıklamasını içerir. Örneğin, her proje içinde entegrasyon testi ve kabul testi yapılacağını belirtmek yeterli değildir. Birim ve kabul testlerinin ne anlama geldiğini, bunların genellikle nasıl gerçekleştirildiğini ve ana hedeflerinin neler olduğunu tanımlamamız gerekir. Deneyimler, bir test stratejisi tanımlandığında ve uygulandığında, çeşitli test faaliyetleri arasında daha az çakışma olasılığının olduğunu ve bunun da daha verimli bir test sürecine yol açtığını göstermektedir. Ayrıca, çeşitli test türleri ve seviyelerinin test hedefleri ve yaklaşımları uyumlu olduğundan, daha az boşluk kalması muhtemeldir, bu da daha etkili bir test sürecine ve dolayısıyla daha yüksek bir ürün kalitesine yol açar.



Şekil 2: Çevik Test Çeyrekleri

Test stratejisi, DevOps ortamında hayati öneme sahip bir belgedir. Bu belge, esas olarak CI/CD stratejisi kapsamında ekipler tarafından gerçekleştirilecek testleri, hangi test seviyeleri ve test türlerinin uygulanacağını ve genel yaklaşımı yüksek düzeyde tanımlar. Her test seviyesi ve test türü için hedefler, sorumluluklar, temel sürekli test görevleri, kabul kriterleri/kalite kapısı tanımlanır. Test stratejisi genellikle Çevik test çeyreklerini takip eder (bkz. şekil 2), ancak CI/CD iş hattı etrafında oluşturulur ve iş hattı içindeki faaliyetlerin akışını temsil eder. İş hattı, yalnızca bir CI/CD aracı tarafından yürütülen faaliyetler olarak değil, ekipler bir yapılacak iş (backlog) ögesi üzerinde çalışırken yaptıkları her şey olarak anlaşılmalıdır. Gözden geçirme ve keşif testi gibi test faaliyetleri CI/CD araçlarının dışında gerçekleşir, ancak akışın/iş hattının/değer akışının ayrılmaz bir parçasıdır. Süreç sadece teknoloji/araç olarak değil, faaliyetlerin, sürecin ve insanların görsel temsili olarak da kullanılabilir [bkz. şekil 3].



Şekil 3: Üst düzey bir iş hattı olarak temsil edilen bir test stratejisi

Test stratejisi, DevOps çalışma yöntemini uygularken de hayati öneme sahip bir belgedir. DevOps, birçok kuruluş için nispeten yeni bir kavramdır. Bu nedenle, kuruluş genelinde ortak bir test stratejisi, tüm ilgili kişilere rehberlik edecek ve testlerin iş hattında nerede ve nasıl gerçekleştirileceğini açıklayacaktır. Ayrıca, her projenin/iş hattının “tekerleği yeniden icat etmesini” de önleyecektir.

Testler, geliştirme iş hattının entegre bir parçası olduğundan, test stratejisi elbette geliştirme stratejisiyle tam olarak uyumlu olmalıdır. DevOps test stratejisi geliştirilirken dikkate alınması gereken bazı hususlar: (konuya göre düzenlenmiştir)

Test seviyeleri

- Test Seviyeleri, CI/CD iş hattında genellikle test faaliyetleri/işleri olarak temsil edilir.
- Test stratejisi, genellikle Davranış Odaklı Geliştirme (BDD) uygulamaları tarafından yönlendirilen, geliştirilen kullanıcı hikayelerinin kabul testleri etrafında oluşturulur. Bu, iş hattının her aşamasında sürekli test yapılmasını gerektirir.
- Test stratejisi, genellikle açık API'lerin kullanımının risklerini azaltmak ve Entegrasyon testlerini desteklemek için sözleşme testi (contract testing) gibi bir test seviyesi tanımlar.

Düşük seviye testler

- Çekme isteği (pull request), birleştirme (merge) ve kod gözden geçirme (code review) gibi düşük seviyeli geliştirme ve test uygulamaları dikkate alınmalıdır.

- Statik testler ve diğer düşük seviyeli test faaliyetleri genellikle CI/CD aracı tarafından zorunlu kılınır ve çoğu zaman kaçınılması da mümkün değildir (örneğin, çekme isteğinin bir parçası olarak yapılan otomatik kontroller ve testler gibi).
- Shift-left (sola kaydırma) paradigmasına dayanarak, birim testlerine, üzerinde uzlaşılan bir kod kapsama düzeyine ulaşılmasına ve entegrasyon testlerine de vurgu yapılmaktadır.

Test Otomasyonu

- Sürekli test, yazılım teslimat iş hattına ve DevOps araç zincirine sorunsuz bir şekilde entegre edilmiştir.
- Bir test otomasyon stratejisi, DevOps test stratejisinin kritik bir parçasıdır.

Giriş ve Çıkış Kriterleri

- Giriş kriterleri, DevOps ekipleri için Hazırlık Tanımı (Definition of Ready - DoR) olarak temsil edilir. DoR, test için girdi işlevi gören yüksek kaliteli bir test temelini zorunlu kılmalıdır; örneğin, ekip tarafından ortaklaşa tanımlanan ve genellikle "Given-When-Then" biçiminde ifade edilen kabul kriterleri gibi. "Hazır" durumuna ulaşmadan önce risklerin ve test hedeflerinin net bir şekilde tanımlanmış ve üzerinde uzlaşılmış olması önemlidir; böylece tahminler test ihtiyaçlarını da kapsar.
- Çıkış kriterleri ise DevOps ekipleri için Tamamlanma Tanımı (Definition of Done - DoD) ve CI/CD iş hattındaki her test işine entegre edilmiş Kalite Kapısı (Quality Gate) olarak ikiye ayrılır. DoD, farklı yapılacak iş (backlog) seviyeleri (epik, özellik, hikaye) için farklı şekilde tanımlanabilir ve her seviyede/test aşamasında ulaşılan test tamamlama ve güven düzeyini temsil eder. Başarılı kabul testine ek olarak, DoD genellikle tüm yapılacak iş öğeleri için bir kısıtlama görevi gören fonksiyonel olmayan gereksinimlerin karşılanmasına yönelik beklenen test faaliyetlerini de içerir. DevOps ekiplerinde, bir yapılacak iş ögesi yalnızca üretim ortamına dağıtıldığında ve sistem/servis izleme sonuçlarına göre çalıştığında "Tamamlandı" (Done) durumuna ulaşmış kabul edilir.

Test Türleri: Güvenlik Testi

- Başarı için kritik öneme sahip olan fonksiyonel olmayan özellikleri (örneğin güvenlik, performans, güvenilirlik ve kullanılabilirlik) ve bunlara yönelik test yaklaşımı belirlenir.
- Güvenliğin önemine bağlı olarak, Statik Uygulama Güvenlik Testi (SAST), Dinamik Güvenlik Analiz Aracı (DAST), Etkileşimli Uygulama Güvenlik Testi (IAST), Yazılım Bileşimi Analizi (SCA), Sızma Testleri (Penetration Testing) ve API Güvenlik Testleri (API Security Testing), güvenlik test yaklaşımının bir parçası olarak belirtilebilir.

Test Ortamları

- Test ortamları, CI/CD iş hattındaki çeşitli test aşamaları için gösterilir (örneğin, geliştirme, hazırlık, ön üretim) ve her biri farklı bir odak noktasını temsil eder. Genellikle, tanımlanmış çeşitli test seviyeleriyle de uyumludurlar (örneğin, birim, entegrasyon testi, sistem testi ve kabul testi). DevOps, üretim ortamları için dağıtım ve yapılandırma uygulamalarının yoğun otomasyonuna odaklandığından, Kod Olarak Yapılandırma (Configuration-as-Code) gibi

uygulamaları uygularken, test ortamlarını tasarlarken ve oluştururken de aynı uygulamaları kullanmak açıktır. Bulut tabanlı ortamların kullanımıyla, talep üzerine bir test ortamı oluşturmak ve kullanımdan sonra silmek yaygın bir uygulamadır ve bu da standartlaştırılmış, kontrollü, istikrarlı bir test ortamı sağlar. Blue-Green(Mavi-Yeşil) gibi dağıtım stratejileri kullanıldığında, ürünümüzün en son sürümünün test ortamı, kalite kapıları ve DoD'nin başarıyla yerine getirilmesinden sonra üretim ortamı haline gelebilir. Bu durumda, önceki üretim ortamı bir sonraki test ortamı olarak yeniden kullanılır (mavi ve yeşil ortamlar arasında amaç değiştirme).

Yalın test stratejisi belgesi, DevOps organizasyonu için genellikle iyi bir çözümdür ve ayrıntılı (proje) test planlarından kurtulmanın bir yoludur. TMMi'nin belirli hedefi olan Test Stratejisi Oluşturma, belirli uygulamaları da içeren, DevOps yazılım geliştirmeyi uygulayan organizasyonlar için tamamen geçerlidir. Test politikasında olduğu gibi, test stratejisinin tüm organizasyona / değer akışına / Çevik sürüm trenine / ekibe dağıtılmasını sağlamak önemlidir

2.1.3 SG3 Test Performans Göstergelerini Belirleme

Test politikasında tanımlanan test iyileştirme iş hedefleri, bir dizi temel test performans göstergesine dönüştürülmelidir. Test politikası ve buna eşlik eden performans göstergeleri, beklenen ve elde edilen test performans düzeylerini iletmek için net bir yön ve araç sağlar. Performans göstergeleri, testlerin ve test süreci iyileştirmesinin paydaşlar için değerini göstermelidir. Süreç iyileştirmeye yapılan yatırımlar uzun vadeli yönetim desteği gerektirdiğinden, motivasyonlarını korumak için iyileştirme programının faydalarını nicel olarak ölçmek çok önemlidir. Bu TMMi özel hedefinin, sınırlı sayıda (örneğin 2 veya 3) test performans göstergesini tanımlamakla ilgili olduğuna dikkat edilmelidir. Bu, tam bir ölçüm programı oluşturmak ve uygulamakla ilgili değil, testin değerinin zaman içinde ve farklı teslimat ortamlarında nasıl değiştiğini gösteren bir dizi temel gösterge tanımlamakla ilgilidir.

Çevik yaklaşımda olduğu gibi, DevOps'ta da odak noktası daha çok ekip temelli ve sistemsel düşünce olacaktır. Bu, göstergelerin yalnızca testin kendisinin özellikleriyle sınırlı kalmak yerine, geliştirme ve dağıtım süreci performans göstergelerine doğru genişlemesine neden olabilir. Ana DevOps performans göstergeleri genellikle DevOps Araştırma ve Değerlendirme (DORA) Metriklerini takip eder. (Test) kapsamı yerine, üretim aşamasına geçiş sırasında değişiklik başarısızlık oranı (başarısız geçişlerin yüzdesi) ölçülür.

DORA metriklerine örnekler:

- Yazılım teslim performansı metrikleri: Dağıtım sıklığı (Deployment Frequency), Değişikliklerin teslim süresi (Lead Time for Changes), Servisi geri yükleme süresi (Time to Restore Service), Değişiklik arıza oranı (Change Failure Rate)
- Operasyonel performans: Güvenilirlik metrikleri, örneğin: Ortalama Kurtarma Süresi (MTTR – Mean Time to Restore) ve Ortalama Arıza Arası Süre (MTTF – Mean Time to Failure)

Performans göstergeleri, CI/CD faaliyetlerinin ayrılmaz bir parçası olmalı ve veriler neredeyse gerçek zamanlı olarak ve kuruluşun tüm çalışanları tarafından erişilebilir olmalıdır. DORA metriklerinin yanı

sıra, Değer Akışı Yönetimi/optimizasyonunu desteklemek için akış metrikleri de kullanılır. Test faaliyetleri, katkılarının anlaşılabilir olmasını sağlamak için yukarıdaki metriklere göre ölçülebilir.

Testle İlişkili Değer Akışı Yönetimi (Value Stream Management) için Akış Metriklerine Örnekler:

- Test yürütme teslim süresi (lead time), her kod değişikliğinde CI/CD süreci içindeki test faaliyetlerinin başarı veya arıza oranı
- Akış metrikleri, test yürütme teslim süresinin toplam teslim süresine olan katkısı

Buradaki zorluk, DevOps yaklaşımı ve sistem düşüncesiyle ilgili göstergelerin uygun karışımını tanımlarken, TMMi bazlı bir test iyileştirme programının performans başarılarını iyi bir şekilde göstermeyi başarmak olacaktır. TMMi'nin özel hedefi olan Test Performans Göstergelerini Belirlemek, özel uygulamaları da dahil olmak üzere, seçilen ve uygulanan performans göstergeleri sadece testlerle ilgili olmaktan daha geniş bir kapsamda ve daha geniş olsalar da tamamen uygulanabilir. Elbette, bu durum performans göstergelerinin analizini ve yorumlanmasını daha zor hale getirecektir. Aslında, kullanılan performans göstergeleri test performans göstergesi olarak adlandırılmayabilir. Testle ilgili unsurlar içerdiği ve test iyileştirme çalışmalarında kaydedilen ilerlemeyi değerlendirmek için kullanıldığı sürece, bu durum TMMi bağlamında sorun teşkil etmez.

2.2 Süreç Alanı 2.2 Test Planlama

Test Planlamasının amacı, belirlenen risklere ve tanımlanan test stratejisine dayalı bir test yaklaşımı tanımlamak ve test faaliyetlerini gerçekleştirmek ve yönetmek için sağlam temellere dayanan planlar oluşturmak ve sürdürmektir.

Başarılı test planlamasının anahtarının, ilgili test planını (“belge”) tanımlamakta değil, önceden düşünmekte (“faaliyet”) olduğunu unutmayın.

DevOps yaşam döngüleri için genellikle iki tür planlama yapılır: sürüm planlama ve yineleme planlama. TMMi seviye 2'deki Test Planlama süreç alanı, hem sürüm hem de yineleme planlamasında testle ilgili faaliyetlere odaklanır. Sürüm planlama, projenin başlangıcında bir ürünün sürümünü öngörür. Sürüm planlama, tanımlanmış bir ürün yapılacak işi gerektirir ve daha büyük kullanıcı hikayelerini daha küçük hikayelerden oluşan bir koleksiyona dönüştürmeyi içerebilir. Ancak, DevOps ve Çevik yaklaşımın genellikle sabit zaman çizelgeleri ve esnek içeriklerle çalıştığını unutmayın. Yalnızca test yaklaşımı ve DoD'ye uygun olarak test edilmiş yapılacak iş öğelerinin yayınlandığından emin olmak önemlidir. Sürüm planlaması, tüm yinelemeleri kapsayan bir test yaklaşımı ve test planı için temel sağlar. Sürüm planları üst düzeydir ve ayrıntılı görevlerden ziyade genel strateji veya yaklaşıma odaklanır. Sürüm planlaması tamamlandıktan sonra, ilk yineleme için yineleme planlaması başlar. Yineleme planlaması, tek bir yinelemenin sonunu öngörür ve yineleme yapılacak işleyle ilgilenir.

2.2.1 SG1 Risk Değerlendirmesinin Gerçekleştirilmesi

Yazılımı %100 test etmek imkansızdır ve her zaman seçimler yapılmalı ve öncelikler belirlenmelidir. Bu nedenle, bu TMMi hedefi DevOps projeleri için de geçerlidir. DevOps projeleri için, sürüm

planlamasında ürün vizyon belgesi veya üst düzey ürün hedefleri seti temel alınarak üst düzey bir ürün risk değerlendirmesi yapılmalıdır. Her yineleme için, yineleme planlama oturumunun bir parçası olarak, kullanıcı hikayeleri veya o yineleme için diğer gereksinimler temel alınarak daha ayrıntılı bir ürün risk oturumu yapılmalıdır. Bazen bu, kullanıcı hikayelerinin tartışıldığı ve riskler açısından analiz edildiği iyileştirme oturumlarında zaten yapılmaktadır. DevOps projesindeki ürün risk değerlendirmesi süreci, sıralı yaşam döngüsü modelini izleyen geleneksel bir projede uygulananlara kıyasla çok daha hafif bir formata sahip olacaktır. Kullanılacak hafif ürün risk değerlendirme tekniklerine örnek olarak risk pokeri [Veenendaal12] ve ROAM (Çözülen, Sahip Olunan, Kabul Edilen, Azaltılan) [Baah] verilebilir. Sürüm planlama veya program artış planlama (PI planlama) sırasında, sürümdeki özellikleri bilen iş temsilcileri, fonksiyonel olmayan gereksinimleri kısıtlamalar olarak kullanarak geliştirilecek fonksiyonelliğin genel bir özetini sunar ve test uzmanları da dahil olmak üzere tüm ekip risklerin belirlenmesi ve değerlendirilmesine yardımcı olur.

Yineleme planlaması sırasında DevOps ekibi, yaklaşan yinelemede uygulanacak kullanıcı hikayelerine dayalı olarak ürün risklerini belirler ve analiz eder. Tercihen Çevik ekibinin tüm üyeleri ve mümkünse diğer bazı paydaşlar da ürün riski oturumuna katılır. Sonuç, test için kritik alanları belirleyen, öncelikli ürün riski öğelerinin listesidir. Bu da her bir riski yeterli testlerle kapsamak için ayrılacak uygun test eforunun belirlenmesine ve yapılacak test çalışmalarının etkinliğini ve verimliliğini optimize edecek şekilde bu testlerin sıralanmasına yardımcı olur. Tahmin edilen görevler, bunlarla ilişkili ürün risk düzeyine göre önceliklendirilebilir. Daha yüksek risklerle ilişkili görevler daha erken başlamalı ve daha fazla test eforu gerektirmelidir. Daha düşük risklerle ilişkili görevler daha geç başlamalı ve daha az test eforu gerektirmelidir.

Birçok kuruluşta güvenliğin önemi göz önüne alındığında, tehdit modellemesi yukarıdakilere ek olarak güvenlik gereksinimlerini belirlemek, güvenlik tehditlerini ve potansiyel güvenlik açıklarını tespit etmek, tehdit ve güvenlik açıklarının kritikliğini ölçmek ve azaltmak için düzeltme yöntemlerini önceliklendirmek için kullanılabilir. Güvenlik tehditleri, her bir tehdidin olasılığını ve etkisini ölçmek için kılavuzlar kullanılarak önceliklendirilir ve ciddiyet derecesi belirlenir.

2.2.2 SG2 Test Yaklaşımı Belirleme

Tanımlanan ve önceliklendirilen ürün risklerini azaltmak için bir test yaklaşımı tanımlanır. Belirli bir yineleme için, test edilecek öğeler ve özellikler yineleme planlaması sırasında belirlenir. Bu faaliyet aynı zamanda ürün riski oturumunun sonucuna da dayanır. Test edilecek öğelerin önceliklendirilmiş listesi genellikle bu yinelemede test edilecek kullanıcı hikayeleriyle ilgilidir. Özellikler genellikle, diğerlerinin yanı sıra, test edilecek çeşitli yazılım kalite özellikleriyle ilgilidir. Yineleme sırasında yeni ürün riskleri ortaya çıkabilir ve bu riskler ek testler gerektirebilir. Ek testler gerektiren yeni ürün riskleri gibi konular genellikle günlük ayakta (standup) toplantılarında tartışılır.

DevOps ile test yaklaşımı, CI/CD iş hattı aracılığıyla Sürekli Test temelinde tanımlanır. Sürekli Test, teslimat hattının her aşamasına uygun, eyleme geçirilebilir geri bildirimlerle otomasyonu kullanarak erken ve sık test yapmak anlamına gelir. Sürekli Entegrasyon (CI) ve Sürekli Dağıtım (CD) araçları genellikle kod taahhüdü üzerine otomatik olarak çağrılır ve derleme, entegrasyon, test, güvenlik, uyumluluk ve dağıtım faaliyetlerini koordine eder. CI/CD iş hattındaki test faaliyetleri genellikle Çevik

Test Çeyrekleri kullanılarak belirlenen test seviyelerine dayanır ve ekipler tarafından sürekli olarak optimize edilir.

Riskleri azaltmak için yinleme düzeyinde veya yapılacak iş ögesi düzeyinde tanımlanan test yaklaşımı, örneğin kullanıcı hikayelerinin ve kabul kriterlerinin ek olarak gözden geçirilmesini, risk düzeyine orantılı test eforunu, riskin düzeyine ve türüne göre uygun test tekniklerinin seçilmesini kapsayabilir. Ayrıca, yalnızca dağıtımdan sonra gerçekleştirilen test faaliyetlerini de önerebilir; örneğin, kullanıcı deneyimi hakkında geri bildirim toplamak için üretimde topluluk testi (crowd testing) veya gizli dağıtım (dark launch) kullanarak gerçek kullanıcı trafiği ile performans testi. DevOps bağlamında, Davranış Odaklı Geliştirme (BDD) ve Kabul Testi Odaklı Geliştirme (ATDD), test senaryolarının tanımlanmasını ve tasarlanmasını yönlendirmek için popüler tekniklerdir. Sürüm düzeyindeki test yaklaşımı çok daha yüksek bir düzeyde olacak ve program veya organizasyon düzeyinde tanımlanmış test stratejisine dayalı olacaktır. Genellikle bir test yaklaşımı, ekip/proje wiki'sinde tutulur veya görüntülenir.

Yinelemeli geliştirmede her zaman görülen önemli bir risk, regresyon riskidir. Test yaklaşımı, regresyon riskinin nasıl yönetileceğini tanımlamalıdır. Genellikle bu, tercihen otomatikleştirilmiş belirli bir regresyon test seti oluşturularak yapılır. Bu bağlamda test otomasyon piramidi yardımcı olur [Cohn]. Piramidin en alt seviyesindeki birim testlerinden başlayarak servis seviyesi testlerine geçerek regresyon test otomasyonunun değerini nasıl en üst düzeye çıkarabileceğinizi gösterir. Kullanıcı arayüzü testleri en üstte yer alır. Birim testleri hızlı ve güvenilirdir. Servis katmanı, kullanıcı arayüzü (UI) tarafından engellenmeden API veya servis düzeyinde iş mantığını test etmeye olanak tanır. Düzey ne kadar yüksekse, testler o kadar yavaş ve kırılgan hale gelir. Otomasyon, DevOps'un önemli bir parçası olduğundan, test otomasyonu DevOps test yaklaşımının ana odak noktasıdır ve sadece regresyon testlerini desteklemekle kalmaz. Regresyon riskini ele almanın bir başka yolu da sistemin gözlemlenebilirliğine güvenmek, durumunu ve kullanımını sürekli izlemek ve riski önlemek için otomatik mekanizmalar oluşturmaktır, örneğin önceki, kararlı sürüme geri dönmek.

Giriş kriterleri (Hazırlık Tanımı olarak da adlandırılır), normalde tanımlanmış bir test yaklaşımının (özel uygulama 2.3) bir parçasıdır ve DevOps geliştirmede farklı şekilde ele alınabilir. DevOps yazılım geliştirmede, testler ekip sürecinin ayrılmaz bir parçası ve sürekli bir faaliyettir. Eskiden kullanılan giriş kriterleri kontrol listesi, Hazırlık Tanımı arasında bölünmüştür; örneğin, Kabul Kriterleri, Given-When-Then ifadeleri kullanılarak test senaryoları olarak tanımlanmıştır ve CI/CD makinesi içindeki kontroller, örneğin bir çekme isteği içindeki kontroller, daha büyük bir test yürütme faaliyetine başlamadan önce yeni dağıtılmış bir test ortamında duman testi (smoke test) çalıştırma veya gerekli test verilerinin de dağıtıldığını kontrol etme. Sonuç olarak, testin başlatılıp başlatılamayacağını belirlemek için CI/CD iş akışında belirli bir kontrol listesi veya geçit uygulanmalı ve diğer hatalar gibi uygunsuzluk durumunda yapı başarısız olmalıdır. Bu, DevOps ekibinde bir test aşamasından (ör. birim testi) diğerine (ör. kabul testi) geçen bir bileşen için de geçerlidir.

Test çıkış kriterleri (özel uygulama 2.4), CI/CD iş hattında gerçekleştirilen kalite kontrollerine büyük ölçüde dayanan, Tamamlanma Tanımı (DoD) kapsamında yer alır. DoD'nin, test kapsamı ve ürün kalitesi gibi testlerle ilgili özel kriterlere sahip olması önemlidir. Yinleme, üzerinde anlaşmaya varılan kullanıcı hikayelerinin uygulanmasıyla sonuçlanmalı ve DoD'da tanımlanan (test) çıkış kriterlerini karşılamalıdır. DevOps'ta "Tamamlandı (Done)" genellikle, yapılacak iş öğeleriyle ilgili kod değişikliklerinin CI/CD iş hattındaki tüm kalite kontrol noktalarından geçerek üretime dağıtıldığı anlamına gelir. Sürüm talep

üzerine gerçekleşiyorsa ve Özellik Kontrolü (Feature Toggles) gibi teknolojiler kullanılarak Dağıtımdan ayrılmışsa, birden fazla yinelemeyi kapsayabilecek sürüm düzeyinde bir DoD de vardır, ancak DevOps'un amacı, mümkün olduğunca küçük parçalar halinde sık sık sürüm yayınlamaktır. Sürüm düzeyindeki DoD yine tipik olarak kapsam ve ürün kalitesi ile ilgili kriterlere sahiptir ve genellikle, topluluk testi veya canary testi gibi üretimde gerçekleştirilen test faaliyetlerine dayalı olarak, beklenen değerler kullanıcıya sunulduğunu doğrulamaya odaklanır.

Özel uygulama 2.5 Askıya alma ve yeniden başlatma kriterlerini tanımlamak, DevOps yaşam döngüleri ile ilgili olmayabilir. Testler, DevOps yazılım geliştirme sürecinin ayrılmaz bir parçası olduğundan, elbette diğer yineleme faaliyetlerinden ayrı ve bağımsız bir faaliyet olarak ele alınmayacaktır. Testlerin ilerlemesine potansiyel veya gerçek tehditler olarak değerlendirilebilecek engelleyici sorunlar olduğunda, bunlar günlük ayakta toplantısında tartışılır. Bu forumda ekip, sorunları çözmek için alınması gereken önlemleri (varsa) belirler. Bu nedenle, resmi askıya alma ve yeniden başlatma kriterleri gerekli olmayacak ve tanımlanmayacaktır; bu konuyla ilgili sorunlar normal DevOps rutininin bir parçası olarak ele alınır. DevOps rutini, bu özel uygulama için alternatif bir uygulama görevi görür.

2.2.3 SG3 Test Tahminlerini Belirleme

DevOps ekipleri için, testlere ilişkin ayrıntılı tahminler yineleme planlaması sırasında yapılır. Üst düzey (test) tahminler, sürüm planlaması sırasında ve muhtemelen yapılacak işlerin iyileştirilmesi oturumlarında da yapılır. Tüm tahminler elbette, her bir hikayeyi teslim etmek için gereken tüm eforu içeren bir ekip tahmini olarak yapılır. Tahmin oturumları sırasında test faaliyetlerinin gerçekten dikkate alındığından emin olmak önemlidir. Bu, test faaliyetlerinin ayrı görevler olarak tanımlanması ve ardından her bir test görevinin tahmin edilmesi veya yapılacak testlerin açıkça belirtilmesi ve böylece dikkate alınmasıyla her bir kullanıcı hikayesinin tahmin edilmesi yoluyla yapılabilir. DevOps'ta neredeyse her şeyin otomasyonu beklendiği gibi, test otomasyonu, test ortamı dağıtım otomasyonu, test verisi dağıtım otomasyonu ve yineleme için belirlenen test yaklaşımına uygun olarak iş hattının iyileştirilmesi de dahil olmak üzere testle ilgili faaliyetlerin otomasyonu da tahmin edilmelidir. Tahmin sırasında, DevOps ekibinin bir parçası olan bir test uzmanı tahmin oturumlarına katılmalıdır. Planlama Pokeri (Planning Poker), T-shirt boyutlandırma (T-shirt sizing), yüzme kulvarı boyutlandırma (swim lane sizing) / (kova boyutlandırma) (bucket sizing) ve nokta oylama (dot voting) DevOps yazılım geliştirmede kullanılan tipik tahmin teknikleridir.

Bir sonraki yineleme için yapılacak işler genellikle kullanıcı hikayeleriyle tanımlanır. Kullanıcı hikayelerinin tahmin edilebilir olması için boyutlarının küçük olması gerekir. Tahmin edilebilirlik, INVEST tarafından tanımlanan kullanıcı hikayesi kriterlerinden biridir ve bir yinelemenin parçası olacak kullanıcı hikayeleri için geçerlidir. Yineleme planlaması sırasında Çevik ekipler genellikle testle ilgili görevleri belirler ve tanımlar. Test görevleri, diğer geliştirme görevleriyle birlikte bir görev panosunda kaydedilir. Geriye dönük değerlendirmede, ekipler tahminlerinin doğruluğunu (eğer bir sorun olarak tanımlanmışsa) gözden geçirmeli ve yinelemede yapılan bireysel görevlerin boyutundan öğrendiklerine dayanarak bunu iyileştirmelidir.

Sürüm planlamasında, kullanıcı hikayeleri veya epikler genellikle daha üst düzeyde tanımlanır ve henüz belirli görevlere ayrıntılı olarak bölünmez. Bu durum elbette tahminleri daha zor ve daha az doğru hale getirir. Belirtildiği gibi, DevOps projeleri tahminler için bir temel olarak iş bölümü yapısı oluşturmaz,

ancak sürüm düzeyinde, gerekli tahmin doğruluğunu sağlamak için yeterli olan, yani yapılacak iş ögesinin önceliklendirilmesini mümkün kılan, yapılacak iş ögesinin ne ve nasıl geliştirileceğine dair bilgilerden yararlanabilir. DevOps'ta amaç değer akışını optimize etmek olduğundan, tahminler bir sonraki adımda neyin geliştirileceğine dair kararı desteklemeli ve tam kapsamlı bir proje zaman çizelgesi oluşturmamalıdır.

DevOps ekibi nispeten gayri resmi bir şekilde tahminlerde bulunsa da, tahminlerin gerekçesi açık olmalıdır (yani hangi faktörlerin dikkate alındığı). Gereğe dayalı tartışmalar, tahminlerin doğruluk seviyesini artırır. Genellikle DevOps projelerinde tahminler, büyüklük (hikaye puanları kullanılarak) veya efor (tahmin birimi olarak ideal adam-günler kullanılarak) üzerine odaklanır. DevOps projelerinde maliyetler normalde tahmin oturumlarının bir parçası olarak ele alınmaz. DevOps bağlamında altyapı maliyetlerinin genellikle yüksek olduğunu ve maliyet tahminlerinin genellikle ekip tahmin oturumlarında değil, özel tahmin oturumlarında belirlendiğini unutmayın. Altyapı maliyetlerinin, örneğin genel bulut kullanımının yakından izlenmesi ve kontrol edilmesi gerekir ve bu, otomatik raporlama ile CI/CD iş hattına entegre edilebilir. (Örneğin, yürütülen her GitHub Action işi, kullanılan vCPU dakikaları olarak sunulur).

Bu TMMi özel hedefi ve özel uygulamaları, özel uygulama 3.2 “Test yaşam döngüsünü tanımla” hariç, tamamen uygulanabilir. Yinelemeli ve DevOps yazılım geliştirmenin temellerinden biri, küçük parçalar halinde çalışmaktır. Bu nedenle, tanımlanan görevler genellikle (test) tahminine temel teşkil edecek kadar ayrıntılıdır. Bu nedenle, DevOps bağlamında, tahmin için ek bir temel teşkil edecek şekilde test faaliyetleri için bir yaşam döngüsü tanımlamak da gereklidir.

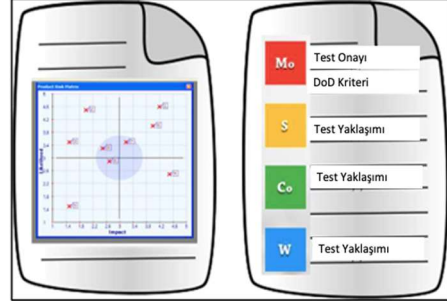
2.2.4 SG4 Test Planı Geliştirme

Test planlamasının önceden düşünme (“faaliyet”) ile ilgili olduğunu ve ilgili test planını (“belge”) tanımlamakla ilgili olmadığını tekrar belirtmekte fayda var. DevOps'ta, bu TMMi özel hedefi tarafından tanımlanan test planlama faaliyetlerinin çoğu, sürüm ve yineleme planlaması sırasında ve ayrıca yapılacak iş öğeleri iyileştirme oturumlarında gerçekleştirilecektir. Ancak, bu faaliyetlerin sonuçları genellikle bir test planında belgelenmez, özellikle de görev panosuna yansıtılabilecekleri yineleme planlaması için. Testler, sürüm ve yineleme planlamasının ayrılmaz bir parçası olarak gerçekleştirildiğinden, ortaya çıkan “çizelgeler” test faaliyetlerini de içerecektir. DevOps projesindeki zaman çizelgesi, sıralı yaşam döngüleriyle geliştirilen ayrıntılı bir zaman çizelgesinden ziyade, istenen iş değeri teslimatına dayalı olarak sürüm ve yineleme önceliklerini yansıtan kullanıcı hikayeleri (yapılacak iş öğeleri) ve görevlerin sıralamasına daha çok benzer. Test yürütmesinin esas olarak CI/CD iş hattında, kod taahhüdüyle tetiklendiğini unutulmamalıdır. Ek olarak, manuel testler veya Üretimde yapılan faaliyetler Yineleme panosunda planlanabilir. Burada destekleyici bir teknik olarak genellikle zihin haritaları kullanılır. Görev panosu, yineleme önceliklerini yansıtacaktır. Bu nedenle, açık bir (test) zaman çizelgesi oluşturulmaz; kullanıcı hikayeleri ve gerçekleştirilecek görevler, test görevleri de dahil olmak üzere, için net sürüm ve yineleme öncelikleri tanımlanması beklenir.

Proje düzeyinde, test personelinin seçimi ekibin oluşturulmasının bir parçasıdır; test veya çok becerili kaynaklara olan ihtiyaç önceden belirlenir. Projeler değiştikçe veya büyüdükçe, belirli bir ekip/projeye bir kişinin ilk seçilme nedeni kolayca unutulabilir; bu neden genellikle ekip/projeyle ilgili özel beceri ihtiyaçlarını veya deneyimi içerir. Bu, kişilerin beceri ihtiyaçlarını yazmak ve ekip/projeye neden

seçildiklerine ilişkin destekleyici bilgiler sağlamak için iyi bir gerekçe oluşturur. DevOps ekipleri tanımlandıktan sonra, test personelinin atanması aşağı yukarı sabitlenir. Yineleme planlaması sırasında, bir yinelemede testi gerçekleştirmek için gerekli olan (ek) kaynakların ve becerilerin, örneğin fonksiyonel olmayan testler için, tanımlanması, gerekirse, ekibin gerekli testleri gerçekleştirmek için yeterli test kaynaklarına, bilgiye ve becerilere sahip olmasını sağlamak için tartışılabilir.

İlk proje riski oturumu, sürüm ve yineleme planlamasının bir parçası olmalıdır. Yineleme sırasında ortaya çıkan diğer proje risklerinin belirlenmesi (ve yönetimi), günlük ayakta toplantılarının ve düzenli scrum-of-scrum toplantılarının bir parçasıdır ve genellikle engel günlüğü ile belgelenir. Test sorunlarının da engel günlüğüne kaydedilmesi önemlidir. Engeller, çözülene kadar günlük ayakta toplantısında veya ekibin bunları çözme yetkisi yoksa scrum-of-scrum toplantılarında tartışılmalıdır.



Şekil 4: Tek Sayfalık Bir Test Planı

Genellikle belirli ve ayrıntılı bir test planı geliştirilip belgelendirilmez, ancak 4.5 Test planını oluşturma uygulaması DevOps bağlamında hala geçerlidir. Test planlaması bağlamında yapılan tartışmaların sonucu, ancak hafif bir biçimde, muhtemelen bir zihin haritası veya sadece bir sayfalık bir belge (bkz. şekil 4), takım düzeyinde Çevik Test çeyrekleri ve CI/CD iş hattındaki test faaliyetleri şeklinde kaydedilebilir.

2.2.5 SG5: Test Planına Bağlılığı Sağlama

DevOps içinde, test yaklaşımı ve test planı geliştirme ve oluşturma süreci, muhtemelen bir test uzmanı (ekip üyelerinden biri) tarafından yönetilen, ekip tabanlı bir çalışmadır. Ürün kalitesi, ekibin sorumluluğundadır. Bu nedenle, ekip doğru süreci izlediği ve üzerinde anlaşmaya varılan test faaliyetleri CI/CD iş hattına entegre edildiği sürece, (test) yaklaşımına ve (test) planına bağlılık, bir ekip çalışması olduğu için zaten sürüm ve yineleme planlamasının dolaylı bir sonucudur. Bu, elbette, test uzmanının test planını hazırladığı ve ardından açık bir taahhüt alması gereken geleneksel bir ortamdaki çalışma şekliyle büyük bir fark oluşturur. CI/CD iş hattının Test Planını dağıtılmış test ortamları ve yürütülen otomatik testler şeklinde uygulamak için gerekli altyapı kaynakları dikkate alınmalı ve paydaşlar tarafından sağlanmalıdır.

DevOps bağlamında, ekip (ürün sahibi dahil) ürün risklerinin öncelikli listesini ve gerçekleştirilecek test azaltma eylemlerini anlamalı ve üzerinde anlaşmalıdır. Anlama ve taahhüt, örneğin, ürün risklerini, test yaklaşımını ve bunun gerekçesini ekibe açıklayan kısa bir sunumun ardından sürüm veya yineleme planlaması sırasında yapılacak bir tartışma yoluyla sağlanabilir.

Tahmin oturumu sırasında (bkz. SG 3 Test Tahminlerini Belirleme) iş yükü tahmin edilir. Bir ekibin (test) kaynakları, DevOps ekibinin yapısı tarafından belirlenir. Tahmin edilen ve geliştirilmek üzere seçilen kullanıcı hikayeleri, mevcut kaynakları başlangıç noktası (kısıtlama) olarak alır. Bu nedenle, iş ve kaynak düzeylerini yeniden uzlaştırmak anlamlı bir faaliyet değildir, ancak bulut bilişim/depolama/ağ kullanılabilirliği gibi teknik kaynaklara sürekli odaklanmak önemlidir ve bu kaynaklar ekiplere sağlanmalıdır. Bu nedenle, 3.2 İş ve kaynak düzeylerini uzlaştırma uygulaması, DevOps bağlamında

genellikle ilgili olmayan bir uygulamadır. Günlük ayakta toplantıları ve Scrum-of-Scrum toplantıları, yineleme sırasında ortaya çıkan kaynak sorunlarını ele almak ve uygun kaynakları hemen (yeniden) tahsis etmek veya bir yinelemeden bir teslimatı kaldırarak gelecekteki yineleme veya sürüm planlamasında uzlaştırmak için kullanılacaktır.

2.3 Süreç Alanı 2.3 Test İzleme ve Kontrol

Test İzleme ve Kontrolünün amacı, test sürecinin ilerleyişi ve ürün kalitesi hakkında bilgi sağlamak, böylece test sürecinin planından önemli ölçüde sapması ve ürün kalitesinin beklentilerden önemli ölçüde sapması durumunda uygun düzeltici önlemlerin alınabilmesini sağlamaktır.

DevOps bağlamında izleme ve kontrol, geleneksel projelere kıyasla bazı temel farklılıklar gösterir. İzleme ve kontrol, DevOps'un temel unsurları olsa da, bu, katı bir plana bağlı kalmanın amaç olduğu anlamına gelmez. Test izleme ve kontrol açısından bu, plan odaklı çalışmadığımız, ancak testlerden elde ettiğimiz ilerleme ve sonuçları sürekli olarak gözden geçirdiğimiz ve yeni ürün riskleri ortaya çıktıkça planımızı ve yaklaşımımızı uygun şekilde uyarladığımız anlamına gelir.

Genellikle ince ayarlı kabul kriterleri, yapılacak işler (risk düzeyleri dahil) ve görevler arasında dağıtılan test planı, canlı bir varlıktır ve yeni bilgiler elde edildikçe veya geri bildirimler alındıkça sürekli olarak gözden geçirilmesi ve güncellenmesi gerekir. Test faaliyetlerinin hedefleri, belirli bir yapılacak iş ögesi için kabul kriterleri kümesi, belirli bir yapılacak iş ögesi için güncellenen ekibin test yaklaşımı ve yapılacak iş düzeyinde belirli Tamamlanma Tanımı kriterleri ile tanımlanır. DevOps projeleri ayrıca “büyük resmi” göz önünde bulundurmalı ve CI/CD düzeyinde ve yineleme düzeyinde gözden geçirme ve kontrol yapılmalıdır.

Testlerin DevOps ekibinin genel sürecine tamamen entegre bir süreç olduğu göz önünde bulundurulduğunda, test izleme ve kontrolünün de DevOps ekibinin genel izleme ve kontrol mekanizmalarının ayrılmaz bir parçası olduğu unutulmamalıdır. Sonuç olarak, test uzmanları geleneksel projelerde olduğu gibi bir test yöneticisine değil, ekibe rapor verir.

2.3.1 SG1 Test İlerlemesini Plana Göre İzleme

DevOps ekipleri, testlerin ilerleyişini izlemek ve kaydetmek için çeşitli yöntemler ve araçlar kullanır. Örneğin, Çevik görev panosunda test görevlerinin ve hikayelerin ilerleyişi ve kalan iş (burndown) çizelgeleri. Bunlar daha sonra wiki panoları ve pano tarzı e-postalar gibi medya araçları kullanılarak veya stand up toplantıları sırasında sözlü olarak ekibin geri kalanına iletilebilir. Ekipler hem tüm sürümdeki hem de her yinelemedeki ilerlemeyi izlemek için kalan iş çizelgelerini kullanabilir. Bir kalan iş çizelgesi genellikle beklenen hız ve uygulanacak özelliklerin yapılacak iş yüküne göre ilerlemeyi gösterir (yani ekibin performansını gösterir). Ayrıca, DevOps'ta çevrimiçi panolarla sürekli izleme, CI/CD iş hattı boyunca altyapıyı, uygulamayı ve ağı otomatik bir şekilde izlemek için kullanılır. Test ortamı kaynakları, yineleme planlaması sırasında kararlaştırılanlara ve CI/CD iş hattı kullanımına göre sürekli olarak izlenir. Diğer bir yaygın uygulama, görev panosu aracılığıyla test ortamı sorunlarını izlemektir. Örneğin, hikaye kartlarına “engellenen test ortamı” yazılı etiketler yapıştırmak veya panoda, ortamlar tarafından engellenen tüm hikayelerin engel kaldırılana kadar bekletileceği ayrı bir sütun oluşturmak

gibi. Buradaki amaç, ilerlemeyi engelleyen test ortamının etkisini görünür kılmak ve ekibin dikkatini, diğer görevlere geçmeden önce bu sorunu çözmeye yöneltmektir.

Testlerin durumu da dahil olmak üzere ekibin ve CI/CD'nin mevcut durumunu anında ve ayrıntılı bir şekilde görsel olarak sunmak için ekipler genellikle görev panoları kullanır. Yineleme planlaması sırasında oluşturulan hikaye kartları, geliştirme görevleri, test görevleri ve diğer görevler görev panosunda kaydedilir. Yineleme sırasında, ilerleme bu görevlerin görev panosunda “yapılacaklar”, “devam edenler” ve “tamamlananlar” gibi sütunlara taşınmasıyla yönetilir. Akışı korumak için, Kanban Panosu'ndaki farklı durumlara genellikle İşlemde (Work in Progress - WIP) sınırları uygulanır ve bu sayede ekibin yeni görevlere başlamaktansa (devam eden) görevleri bitirmeye odaklanması sağlanır. DevOps ekipleri genellikle görev panolarındaki hikâye kartlarını korumak için araçlar kullanır ve bu araçlar gösterge panellerini ve durum özetini otomatikleştirebilir. Ancak, bazı ekipler bireysel faaliyetler için belirli görevler oluşturmazlar, sadece hikâye kartını kullanır ve bu kartta testler, referans araçları veya testlerin belgelendirilebileceği wiki'ler için yorumlar eklerler. Görev panosundaki test görevleri genellikle kullanıcı hikayeleri için tanımlanan kabul kriterleri ve her yapılacak iş ögesi için bir kısıtlama olan fonksiyonel olmayan gereksinimler ile ilgilidir. Bir test görevi için test otomasyon komut dosyaları ve keşif testleri geçme durumuna ulaştığında, görev görev panosunun “tamamlandı” sütununa taşınır.

Tamamlanma Tanımı (DoD), ilerlemenin ölçüldüğü çıkış kriterleri olarak işlev görür. DoD ayrıca test faaliyetleriyle de ilişkili olmalı ve bir kullanıcı hikayesinin geliştirilmesi ve test edilmesinden önce yerine getirilmesi gereken tüm kriterleri göstermelidir. Test görevleriyle ilgili test kriterlerinin, ekibin Tamamlanma Tanımı olarak tamamlamayı kabul ettiği şeyin sadece bir parçasını oluşturduğuna dikkat edin. Tamamlanma Tanımı kriterleri genellikle birden fazla düzeyde uygulanır, örneğin hikaye düzeyinde ve epik düzeyinde. Tüm ekip, görevlerin kabul edilebilir bir hızda ilerlediğinden emin olmak için, genellikle günlük ayakta toplantıları sırasında görev panosunun durumunu düzenli olarak gözden geçirir. Herhangi bir görev (test görevleri dahil) ilerlemiyorsa veya çok yavaş ilerliyorsa, bu durum, bu görevlerin ilerlemesini engelleyebilecek sorunların analiz edildiği ekip bazlı bir tartışmayı tetikler.

Günlük ayakta toplantısına, test uzmanları da dahil olmak üzere DevOps ekibinin tüm üyeleri katılır. Bu toplantıda, ekip üyeleri mevcut durumlarını ve gerçek ilerlemelerini ekibin geri kalanına bildirir. Geliştirme veya test sürecini engelleyebilecek her türlü engel, günlük ayakta toplantılarında bildirilir, böylece tüm ekip bu engellerin farkında olur ve buna göre hareket edebilir. Bu şekilde, proje risk yönetimi bu günlük toplantılara entegre edilir. Ekip, proje risklerinin durumunun izlenmesini desteklemek için düzenli olarak ROAM-lama egzersizi yapmaya karar verebilir [ROAM]. Test ortamlarının bulunmaması gibi testlerle ilgili olanlar da dahil olmak üzere tüm proje riskleri, günlük ayakta toplantısında iletilebilir ve ele alınabilir.

(Not: Ürün risklerinin izlenmesi, ürün kalitesinin izlenmesinin bir parçasıdır ve bu nedenle, aşağıda SP 2 Plan ve Beklentilere Göre Ürün Kalitesini İzleme özel hedefi ile birlikte ele alınacaktır.) Günlük görev yönetimi için günlük ayakta toplantıları ve görev rotasının düzeltilmesine ilişkin DevOps ekibi uygulamaları, Test İzleme ve Kontrol süreç alanındaki TMMi özel uygulamalarının amacına uygun, kanıtlanmış iyi tekniklerdir.

Yinelemenin sonunda, kararlaştırılan sprint hedeflerine göre bir sprint incelemesi yapılır. Ekip, neler başardığını, örneğin hangi hikayelerin veya epiklerin tamamlandığını gösterir. CI/CD iş hattı aracılığıyla entegrasyon elbette sürekli devam eder. Testlerin başarıları, örneğin Tamamlanma Tanımı'na göre, yineleme incelemesinin bir parçası olacaktır. Sunulan ürünün iş değeri ve kalitesini tartışmak için paydaşlarla demolar düzenlenir. Paydaşlar, yineleme planlaması, yineleme incelemeleri (demolar) ve geriye dönük değerlendirmelerde ürün sahibi tarafından temsil edilir. Ürün sahibi, ürün yapılacak işi hakkındaki tartışmalara katılır ve yineleme boyunca kullanıcı hikayelerinin detaylandırılmasına ve testlerin tasarımına geri bildirim ve katkı sağlar. Diğer paydaşlar, her yinelemenin sonunda yineleme incelemesi sırasında sürece dahil olur. Ürün sahibinin paydaşları temsil etmesi Çevik çalışma yöntemine dahil olduğundan, paydaşların katılımının özel olarak izlenmesi gerekmez.

2.3.2 SG2 Ürün Kalitesini Plan ve Beklentilere Göre İzleme

Ürün kalitesinin izlenmesi için, ilerlemenin izlenmesi ile büyük ölçüde aynı mekanizmalar kullanılır (yukarıdaki SG1'e bakın). Çevrimiçi gösterge panosu kullanılarak ürün kalitesinin izlenmesi, yalnızca ekipler tarafından geliştirme ve test aşamalarında yapılmaz, aynı zamanda yazılım ürünlerinin kalitesinin üretim aşamasında izlendiği sürekli bir süreçtir. Operasyon (Ops) rolleri, genellikle Servis Seviyesi Göstergeleri (SLI'ler) olarak tanımlanan gözlemlenebilirlik kullanarak ürünün kalitesini sürekli olarak izler. SLI'lerdeki regresyon veya sapma, kaliteyi ve kullanıcı deneyimini korumak için düzeltici eylemleri tetikler. DevOps'ta, ürün risk izleme için odak noktası, risklerin ayrıntılı belgelerinin incelenmesinden ziyade, düzenli toplantılarda güvenlik riskleri/tehditleri de dahil olmak üzere ürün riskleri listesinin gözden geçirilmesidir. Keşif testlerinin sonucu olarak yeni tespit edilen ürün riskleri veya değişen ürün riskleri tartışılır ve gerekli test eylemleri üzerinde anlaşmaya varılır. Gerekliğinde, ek bir ROAM-lama çalışması talep edilebilir [ROAM]. Çeşitli ürün risklerinin durumu genellikle gösterge tablosundaki grafiklerle gösterilir.

DevOps ekipleri, ürün kalitesini izlemek ve iyileştirmek için, test geçme/kalma oranları, hata bulma oranları, bulunan ve düzeltilen hatalar gibi geleneksel geliştirme metodolojilerinde kullanılanlara benzer hata bazlı metrikler kullanır. Bir yineleme sırasında bulunan ve çözülen hataların sayısı, sürekli entegrasyon faaliyetleri sırasında bulunan hataların sayısı ve bir sonraki yineleme için yapılacak işler listesine eklenmesi muhtemel çözülmemiş hataların sayısı, günlük ayakta toplantıları sırasında izlenmelidir. Genel ürün kalitesini izlemek ve iyileştirmek için birçok DevOps ekibi, ürünün müşteri beklentilerini karşılayıp karşılamadığına dair geri bildirim almak için müşteri memnuniyeti anketleri de kullanır.

Test kapsamı ve ürün kalitesi gibi test çıkış kriterleri, Tamamlanma Tanımı (DoD) kapsamında yer alır. Kararlaştırılan çıkış kriterlerine uyum genellikle görev panosu aracılığıyla izlenir; bu durumda bir hikâye, DoD kriterlerine uyduğu takdirde “tamamlandı” olarak işaretlenebilir. Bu kriterler genellikle, belirli bir yapılacak iş ögesinin üretime devredilmesi kriterini içerir (devredildiğinde “tamamlandı” olarak kabul edilir, tersi geçerli değildir). Genellikle, sürekli testler için belirli çıkış kriterleri tanımlanır. CI/CD iş hattı içindeki sürekli testlerin bir parçası olarak hatalar sürekli olarak izlenir. Üretimdeki ürünün beklenen kalite özelliklerinin yerine getirilmesi, gözlemlenebilirlik kullanılarak izlenir. Otomatik kalite geçişleri (tanımlanmış eşiklere dayalı olarak) genellikle DevOps iş hattının farklı aşamalarında uygulanarak ürün kalitesi beklentileri karşılamadığında dağıtımı engeller. Bu geçitler, test geçme oranı, performans karşılaştırmaları, kod kapsamı ve statik kod analizi sonuçları gibi ölçütleri değerlendirebilir.

Performans ve güvenilirlik genellikle önemli fonksiyonel olmayan kalite özellikleridir ve bu nedenle ürün kalitesinin izlenmesinin bir parçasıdır. Performans testleri (ör. yük ve stres testleri), beklenen kriterlere göre sistem performansını izlemek için iş hattına entegre edilebilir. Performans eşikleri aşıldığında uyarılar tetiklenir ve ekiplere sorunları derhal çözmeleri için bildirim gönderilir. Ayrıca (günlük) izleme araçları, üretimde gerçek zamanlı hata oranlarını, bellek sızıntılarını ve sistem çökmelerini izlemek için kullanılabilir. Bu sorunların erken tespit edilmesi, ürünün güvenilirlik beklentilerini karşıladığından emin olunmasına yardımcı olabilir. Günlük ayakta toplantı, ürün kalitesi incelemelerini neredeyse sürekli olarak gerçekleştirmek için kullanılan mekanizmadır. Doğrulama yapmak ve teslim edilen ürünün iş değeri ve kalitesini tartışmak için paydaşlarla demolar düzenlenir. Ürün kalitesi, tanımlanmış DoD kalite kriterlerine göre doğrulanır ve onaylanır. Test Planlama süreç alanını takiben, giriş, askıya alma ve yeniden başlatma kriterlerini izlemeye ilişkin belirli uygulamalar muhtemelen ilgili olmayacaktır. Daha fazla bilgi için, Test Planlama süreci alanının 2. Özel Hedefi olan Test Yaklaşımı Oluşturma'ya bakın. Burada, bu kriterlerin DevOps ortamında genellikle neden alakasız olabileceği açıklanmaktadır.

2.3.3 SG3 Düzeltici Faaliyetleri Kapanışa Kadar Yönetme

DevOps ekipleri, kalan iş çizelgesindeki beklentilerden sapmalar ve/veya görev panosundaki (test) görevlerin ve hikayelerin ilerlememesi gibi sorunları çok hızlı bir şekilde fark ederler. En önemlisi, üretimdeki yazılımın kalitesi düşüyorsa, bu durum anında fark edilir ve genellikle bilinen kararlı bir sürüme geri dönme gibi düzeltici eylemler tetiklenir. Bu ve diğer sorunlar, örneğin test ilerlemesini engelleyen sorunlar, günlük ayakta toplantılarında iletilir, böylece tüm ekip bu sorunların farkında olur. Bu, sorunların analiz edildiği ekip bazlı bir tartışmayı tetikler. Ekip, ürün sahibi ile birlikte alınacak düzeltici önlemler hakkında karar verir. DevOps tabanlı projelerde düzeltici önlemlerin yönetimi, öncelikle kendi kendini organize eden ekibin sorumluluğundadır. Ekip, uygun düzeltici önlemleri tanımlayıp uygulayabilir veya herhangi bir sorunu Scrum Master'a "engeller" olarak iletebilir. Ürün kalitesine yönelik düzeltici önlemler bazen otomatikleştirilir ve tekrar oluşmasını önlemek için kök nedenleri izlenir. Bu, DevOps ekibinin sürekli iyileştirme kültürünün bir parçasıdır. Scrum Master genellikle ekibi sorunları çözüme kavuşturmak için destekleme sorumluluğuna sahiptir. Düzeltici eylemleri tartışmak ve yönetmek için tipik etkinlikler günlük ayakta toplantıları ve geriye dönük toplantılardır. Kararlaştırılan düzeltici eylemler, ürün yapılacak işi aracılığıyla veya (yineleme içinde) görev panosu aracılığıyla "görevler" veya "yapılacak iş öğeleri" olarak çözüme kavuşturulabilir.

2.4 Süreç Alanı 2.4 Test Tasarımı ve Uygulaması

Test Tasarımı ve Yürütme'nin amacı, test tasarım spesifikasyonunu oluşturmak, test tasarım tekniklerini kullanmak, yapılandırılmış bir test yürütme süreci gerçekleştirmek ve test olaylarını kapanışa kadar yönetmek suretiyle test tasarım ve yürütme sürecinde test süreci yeteneğini iyileştirmektir.

Temel amaç ("riskleri azaltmak ve yazılımı test etmek") DevOps ve geleneksel sıralı projeler için aynı olsa da test etme yaklaşımı genellikle çok farklıdır. DevOps'ta esneklik ve değişime yanıt verebilme yeteneği önemli başlangıç noktalarıdır. Ayrıca, test analizi ve tasarımı, test uygulaması ve test yürütme, birbirini izleyen test aşamaları değil, hızlı geri bildirim döngüleriyle desteklenen, paralel, örtüşen ve yinelemeli olarak gerçekleştirilir. Oluşturulan test dokümantasyonunun ayrıntı düzeyi, test senaryosu otomasyon kodunun bir parçası olarak belgelere dayanan bir diğer önemli farktır. Manuel testlerde,

DevOps projelerinde genellikle tecrübeye dayalı ve hata bazlı teknikler kullanılarak test otomasyonu tamamlanır. Ek olarak, manuel testler, yazılımın üretime geçtikten sonra, önceden seçilmiş bir grup insana dağıtıldıktan sonra, bu grup bazı talimatları/test kurallarını takip ederek ve genellikle kullanılabilirlik veya erişilebilirlik gibi özel kalite özellikleri hakkında geliştirme ekibine ek geri bildirimler sağlayarak kitle testlerinde kullanılabilir. Spesifikasyon bazlı test tasarım teknikleri de, özellikle iş birliğine dayalı gereksinim analizinin bir parçası olarak, Kullanıcı Hikayesi kabul kriterlerini geliştirmek için olası örnek senaryoları listeleterek, hala uygulanabilir ve kullanılabilir. Birim ve entegrasyon testlerine daha fazla önem verilmesi ile birlikte, komut ve karar testi gibi beyaz kutu teknikleri ve API'lere karşı sözleşme testi de çok daha popüler hale gelmiştir. Son bir önemli fark, yüksek değişiklik oranından kaynaklanan regresyon riskinin düzeyidir; bu da çeşitli test düzeylerinde daha fazla regresyon testi yapılmasını gerektirir. İdeal olarak, regresyon testi yüksek düzeyde otomatiktir ve CI/CD iş hattı içinde, birden fazla test aşamasında otomatik olarak tetiklenir ve yürütülür. Birçok fark vardır, ancak sonuçta, Test Tasarımı ve Yürütme süreç alanı bağlamında her zaman testler oluşturmak, ürün risklerini azaltmak, testleri çalıştırmak ve hataları bulmak söz konusudur.

2.4.1 SG1 Test Tasarım Teknikleri Kullanılarak Test Analizi ve Tasarımı Gerçekleştirme

DevOps'ta test analizi ve tasarımı ile test yürütme, genellikle tüm ürün ekibinin Çevik çalışma yöntemine entegre olarak sürekli olarak yürütülen, birbirini destekleyen faaliyetlerdir. DevOps projelerinde, test uzmanları kullanıcı hikayelerini iş birliği içinde oluşturan ve geliştiren ekibin bir parçasıdır. Gereksinimler geliştirilirken, her bir kullanıcı hikayesi için kabul kriterleri de dahil olmak üzere sık sık gayri resmi incelemeler yapılır. Bu kriterler, iş temsilcileri, geliştiriciler ve test uzmanları arasında iş birliği içinde tanımlanmıştır. Genellikle, test uzmanının benzersiz bakış açısı, eksik ayrıntıları ve alternatif senaryoları belirleyerek ve bunları test edilebilir hale getirerek kullanıcı hikayesini iyileştirir. Test analizi bu nedenle açıkça ayrı bir faaliyet değil, test uzmanlarının iş birliğine dayalı kullanıcı hikayesi geliştirme sürecindeki rollerinin bir parçası olarak gerçekleştirdikleri örtük bir faaliyettir. Kullanıcı hikayelerinin analizine dayalı olarak test koşulları belirlenir. Test açısından, test temeli analiz edilerek nelerin test edilebileceği belirlenir – bunlar test koşullarıdır [Veenendaal24]. Tanımlanan kabul kriterleri yeterince ayrıntılı ve açık olduğu sürece, geleneksel test koşullarının rolünü üstlenebilirler. DevOps bağlamında, Gherkin sözdizimini kullanan Davranış Odaklı Geliştirme (BDD) ve Kabul Testi Odaklı Geliştirme (ATDD) gibi teknikler, senaryoları tanımlamaya ve test edilecek koşulları belirlemeye yardımcı olabilir ve bunları belirli bir kullanıcı hikayesinin kabul kriterleri olarak belgeleyebilir. Aynı şekilde, Fonksiyonel Olmayan Gereksinimler için de belirli test koşulları genellikle kabul kriterleri formatında tanımlanır. Test koşulları daha sonra, tanımlanmış ve üzerinde anlaşmaya varılmış kabul kriterlerinin kapsamını sağlamak için gerekli olan testlere dönüştürülür. İlginç test koşulları, keşif testlerinde kullanılmak üzere test şartnamelerinde de kaydedilebilir. Elbette, yukarıdaki bilgiler sadece hikaye düzeyinde geçerli değildir, aynı şey epik ve özellik düzeyinde testler için de geçerlidir. Epik ve özellik düzeyindeki test koşulları, genellikle kullanıcı hikayesi düzeyindeki test koşullarından daha soyuttur ve birden fazla kullanıcı hikayesini kapsar veya fonksiyonel olmayan gereksinimleri temsil eder. Spesifikasyon bazlı test tasarım teknikleri, genellikle kullanıcı hikayelerinden ve kabul kriterlerinden test koşullarını türetmede yardımcı olur. Ancak, DevOps bağlamında bu test tasarım teknikleri çoğunlukla açıkça değil, daha çok örtük olarak kullanılır. Bu durumda test uzmanları, deneyimlerine dayanarak teknikleri ustaca kullanır ve bağlama göre esnek bir şekilde uygulayabilirler. Test-öncelikli ilkesinin DevOps'ta uygulanmasıyla, test koşulları kümesini kapsayan testler, kodun geliştirilmesinden önce veya

en azından kod geliştirmeye paralel olarak belirlenecek (ve muhtemelen otomatikleştirilecektir). Bu yaklaşım bazen zaten yapılacak işlerin iyileştirilmesiyle başlar. Otomatik birim testleri için Test Odaklı Geliştirme (TDD) gibi bir yaklaşım düşünülebilir. Daha yüksek test seviyeleri için, daha önce de belirtildiği gibi Davranış Odaklı Geliştirme (BDD) ve Kabul Testi Odaklı Geliştirme (ATDD), test otomasyonu ile de yakından ilişkili olan popüler yaklaşımlardır. Çoğu manuel testte, testler ekip testlerin yürütülmesi sürecinde tanımlanacak/geliştirilecektir. Testler çoğu zaman geleneksel projelerde olduğu kadar ayrıntılı bir şekilde belgelenmez, daha çok keşif testleri kullanılarak test fikirleri formatında belgelenir. Karmaşık ve kritik alanlar için, test tasarımı ve test senaryosu geliştirme konusunda daha geleneksel bir yaklaşım (resmi test tasarım teknikleri kullanarak) riskleri karşılamak için en iyi yol olabilir. Ancak, bu yaklaşımda da, geleneksel sıralı yaşam döngüsü ortamında yapılan testlere kıyasla dokümantasyon miktarı sınırlı olacaktır. Testlerin önceliklendirilmesi, testlerin kapsadığı kullanıcı hikayesinin önceliklendirilmesine göre yapılır. Kullanıcı hikayelerinin önceliklendirilmesi iş değerine göre yapılır; en yüksek öncelik, en yüksek iş değeriyle ilişkilidir. Testlerin fonksiyonel ve fonksiyonel olmayan riskleri kapsayacak şekilde oluşturulması önemlidir, ancak özellikle DevOps'ta regresyon riskini de kapsamı özellikle önemlidir. Test önceliği, CI/CD iş hattı işlerine testler eklenerek uygulanır.

Test koşullarını ve testlerin yürütülmesini desteklemek için gerekli olan spesifik test verileri belirlenir. Ancak DevOps'ta, geleneksel ortamların aksine, gerekli test verileri genellikle test spesifikasyon belgesinin bir parçası olarak önceden belirtilmez. Test verileri, gerekli araçlar ve/veya işlevler hazır olur olmaz sağlanır. Test verileri anında oluşturulur ve böylece manuel testlerin yürütülmesine neredeyse anında başlanabilir. Ancak otomatik testler için verilerin genellikle önceden belirtilmesi gerekir. Test verileri genellikle test ortamlarıyla aynı şekilde yönetilir ve talep üzerine bu ortamlara dağıtılabilir. Test verisi hazırlama araçları, gizlilik ve GDPR (KVKK) düzenlemeleri gereği üretim verilerini gerektiğinde anonimleştirilebilir, rastgele hale getirebilir ve maskeleyebilir.

Gereksinimler, test koşulları ve testler arasında izlenebilirlik sağlanmalı ve sürdürülmelidir. Ekipler, testlerinin bir parçası olarak çeşitli kullanıcı hikayelerini ve kabul kriterlerini ele aldıklarını açıkça belirtmelidir. Bu nedenle ekibin, gereksinimlerinin (kullanıcı hikayeleri) her kullanıcı hikayesine tanımlayıcı atanmasını destekleyecek şekilde düzenlenmesi ve yönetilmesi için araç desteğine ihtiyacı vardır; böylece bu tanımlayıcılar, testlerin kararlaştırılan kriterlere göre tamamlanmasını sağlamak için kullanılabilir. Genellikle, bir araç içinde birbirine bağlı testler ve kullanıcı hikayeleri, gerçek zamanlı kapsama ölçümü olanakları sağlar. Testin hikaye izlenebilirliği, otomatik olarak oluşturulan CI/CD iş hattı test işi durumu ve günlüğü aracılığıyla görülebilir.

2.4.2 SG2 Test Uygulamasını Gerçekleştirme

Test uygulaması, testlerin yürütülmesine başlamak için gerekli olan her şeyi yerine koymakla ilgilidir. Genellikle, test yürütmeyi desteklemek için test belgelerinin geliştirilmesi en aza indirilir. Bunun yerine, otomatize edilmiş (regresyon) test komut dosyaları geliştirilir ve önceliklendirilir. Regresyon testi hazırlığı, diğer test faaliyetleriyle paralel olarak mümkün olan en kısa sürede başlar.

DevOps ortamında, manuel testler için ayrıntılı test prosedürleri yaygın bir uygulama değildir. Bilgili bir ekipte çalışırken, testler büyük olasılıkla çok daha yüksek bir soyutlama düzeyinde belgeleneyecektir, örneğin keşif testleri bulunan test başlatma belgeleri (test charters). Bu, testleri gerçekleştirenler için yeterli olacaktır, çünkü testleri gerçekleştirmek için gerekli düzeyde alan ve teknik bilgiye sahip oldukları beklenmektedir. Testleri gerçekleştiren kişiler ekip içinde çalışır, bu nedenle hem kodlamanın ne olduğu ve nasıl yapıldığı hem de fonksiyonelliğin amaca uygunluğu konusunda daha iyi bir anlayışa sahip

olurlar. Yinelemeler içindeki ve arasındaki değişiklik düzeyi genellikle yüksek olduğundan, ayrıntılı test prosedürleri geliştirmek de çok fazla bakım işi yaratacaktır. Genellikle, keşif testleri kullanılarak çok sayıda manuel test gerçekleştirilir. Keşif testlerinde, ayrıntılı test prosedürleri geliştirilmez, bunun yerine test fikirlerini ve test uzmanlarına yönelik kılavuzları açıklayan, üst düzey, tek sayfalık test başlatma belgeleri geliştirilir. Ancak çoğu DevOps ekibi otomatize testler kullanmaktadır. Kullanılan tipik yaklaşımlar TDD, BDD ve ATDD'dir. BDD'nin kendisi test otomasyonu olmasa da, kabul testlerini otomatize etmek için temel oluşturabilir. BDD'de (Gherkin gibi sözdiziminde) yazılan senaryolar genellikle test otomasyon çerçeveleriyle bağlantılıdır. Bu yaklaşımlardan bir veya daha fazlasını kullanarak, test uyarlama faaliyetinin bir parçası olarak, örneğin Gherkin senaryolarını ayrıntılı olarak açıklamak için otomatize test komut dosyaları oluşturulacaktır. Test senaryoları aynı zamanda test dokümantasyonu görevi görür, anlaşılması için yeterli yorum içerir ve iyi kodlama uygulamaları kullanılır. Genellikle, grupları geliştirilen test komut dosyalarını gruplandırmak için oluşturulur. Bu paketler daha sonra CI/CD iş hattındaki test işlerine bağlanır.

Test analizi ve tasarım faaliyeti sırasında belirtilen belirli test verileri, tercihen yeniden kullanılabilir formatta oluşturulur. Test Verileri bazen kod olarak ayarlanır; bu durumda test verileri şema dosyaları ve kurallar formatında tanımlanır. CI/CD iş hattı içinde bir alım testi (intake test) uygulanması belirtilmiştir. Bazen güven veya duman testi (confidence or smoke test) olarak da adlandırılan bu test, iş hattındaki her test yürütme işinin başında, test nesnesinin ayrıntılı ve daha ileri testler için hazır olup olmadığını belirlemek için kullanılır. Duman testi, testin başarısızlığı durumunda geliştirme ekibine ve paydaşlara anında geri bildirim sağlamak amacıyla tasarlanmıştır. Ayrıca, CI/CD iş hattı içindeki her test ortamı/aşaması için genellikle otomatik bir dağıtım sağlık kontrol listesi tanımlanır. Test verilerinin hazırlanması, test ortamındaki değişiklikler veya test otomasyonunun uygulanması ile ilgili görevler, yineleme planlaması sırasında ekiplerin görev panosunda tahmin edilir ve planlanır. Özel uygulama 2.4 Test yürütme programı geliştirme ile ilgili olarak, otomatik test yürütme, CI/CD ardışık düzenindeki herhangi bir kod değişikliği ile tetiklenir. Değişiklik durumunda test yürütülmesini başlatmak için tetikleyici senaryolar tanımlanır ve kodlanır (web-hooks). Manuel test uygulamaları, yineleme planlamasına uygun olarak programlanır, görev panosu aracılığıyla test görevleri olarak yönetilir ve günlük ayakta toplantılarında tartışılır.

2.4.3 SG3 Test Yürütme

Testler, her değişiklikte ve yineleme planını takiben ekip tarafından geliştirilen yeni hikayeler için belirtilen CI/CD iş hattı işlerine göre yürütülür. Test işlerinin sırası, hızlı geri bildirim sağlamak için tanımlanmıştır. Yeterli kaynak varsa, geri bildirim güçlendirmek için test işleri paralel olarak yürütülebilir. Bulut maliyetleri ve sürdürülebilirlik, CI/CD sürecini optimize etmek için çok önemlidir. Hatalar/olaylar raporlanır ve test günlükleri oluşturulur. Bir DevOps projesinde, yazılım, kod değişiklikleri tarafından tetiklenerek mümkün olduğunca sık üretime teslim edilir. Test, kaliteli bir ürün sunmak için iş birliği içinde çalışılan sürecin ayrılmaz bir parçasıdır. Duman testleri, daha büyük test setleri yürütülmeden önce CI/CD veri hatlarında tanımlanan test işlerinin bir parçası olarak yürütülür. Başarısızlık durumunda, iş hattının yürütülmesi bir sonraki işe devam edilmez ve geliştirme ekibine geri bildirim verilir. Arızanın nedeni bir hata olarak tespit edilirse, hata rapor edilir.

Test yürütme, yineleme planlaması sırasında belirlenen önceliklere uygun olarak gerçekleştirilir. Bazı testler belgelenmiş bir test prosedürü kullanılarak gerçekleştirilebilir, ancak genellikle birçok test, keşif ve oturum bazlı testler çerçevesini kullanarak gerçekleştirilir. Keşif amaçlı test oturumları önceden tanımlanmış test başlatma belgelerine dayanır ve süre sınırlaması vardır. Testler, yazılımın tüm

özellikleri sunulmadan önce üretim ortamında da gerçekleştirilebilir. Kitlesele testlerde, seçilmiş bir beta test uzmanı grubu, ürünün toplu olarak piyasaya sürülmesinden önce bazı kalite yönleri hakkında geri bildirim verir. Bu, özelliklerin kullanıcı veya coğrafi konum gibi önceden tanımlanmış ön koşullara göre açılıp kapatılabilmesini (özellik geçişi) gerektirir.

DevOps geliştirme ile birlikte, regresyon testlerini organize etme ve yapılandırma ihtiyacı artmaktadır. Özellikle birim ve entegrasyon testi düzeyinde regresyon testi, sürekli entegrasyon sürecinin bir parçasıdır. Temel olarak, her kod değişikliğinde gerçekleşen ve hataları erken ve hızlı bir şekilde tespit eden otomatik bir “oluşturma ve test etme” sürecidir. Sürekli entegrasyon, otomatik regresyon testlerinin düzenli olarak çalıştırılmasını ve kodun kalitesi ve elde edilen kod kapsamı hakkında ekibe hızlı geri bildirim sağlanmasını mümkün kılar. Testler ayrıca, kod değişikliği ile tetiklenen belirtilen CI/CD iş hattı işlerine göre de yürütülür. Bu otomatize test komut dosyaları test sonuçlarını ve test günlüklerini sağlayacaktır. Toplu test günlükleri genellikle epik, kullanıcı hikayesi, test senaryosu, test grubu ve test işi bazında ve tüm süreç için görünümle sağlanır.

Gerçek ve beklenen sonuçlar arasındaki farklılıklar hata olarak rapor edilir. Hata yönetimi genellikle geliştirme sisteminin ayrılmaz bir parçasıdır ve yürütme ve kod değişikliklerine tam izlenebilirlik sağlar. Ayrıca, testlerin yürütülmesi ile ilgili ayrıntıların kronolojik bir kaydını sağlamak için tüm sistemler aracılığıyla test günlükleri otomatik olarak sağlanır.

Ekip tarafından yeni kullanıcı hikayeleri geliştirilip test edilirken, Çevik projelerinde genellikle bulunan tüm hataların gerçekten kaydedilip kaydedilmeyeceği konusunda bir tartışma yaşanır. Özellikle, ekip aynı yerde çalışıyorsa, test uzmanları geliştiricilerle konuşmalı ve hemen düzeltilebilecek ve bir sonraki sürümde dahil edilebilecek hatalar kaydedilmesine gerek kalmayabilir, sadece düzeltilmeleri yeterlidir. Bazı ekipler yalnızca yinelemelerden kaçan hataları kaydeder, bazıları bugün düzeltilemeyen hataları kaydeder, bazıları ise yalnızca yüksek öncelikli hataları kaydeder. Bulunan tüm hatalar kaydedilmezse, hangi hataların kaydedilmesi ve hangilerinin kaydedilmemesi gerektiğini belirlemek için kriterler mevcut olmalıdır.

DevOps ekiplerinde de, test edilen öğelerin tanımlanmış kabul kriterlerini karşılayıp karşılamadığını ve gerçekten “tamamlandı” olarak etiketlenip etiketlenemeyeceğini belirlemek için test yürütme sırasında verileri günlüğe kaydetmek iyi bir uygulama olarak kabul edilir. Test verilerinin günlüğe kaydedilmesi, kullanılan otomasyon tarafından desteklenmelidir. Bu, keşif testi gibi deneyime dayalı teknikler uygulandığında da geçerlidir. Belgelenmesi yararlı olabilecek bilgilere örnek olarak şunlar verilebilir: test kapsamı (ne kadarının kapsandığı ve ne kadarının test edilmesi gerektiği), test sırasında yapılan gözlemler (örneğin, test edilen sistem ve kullanıcı hikayesi kararlı görünüyor mu), risk listesi (hangi riskler kapsanmış ve hangileri en önemli riskler arasında kalmış), bulunan hatalar ve diğer sorunlar ve olası açık sorular. Kaydedilen bilgiler, ekip ve paydaşların gerçekleştirilen tüm testlerin mevcut durumunu kolayca anlayabilecekleri şekilde, bir tür durum yönetimi aracına (örneğin, test yönetimi araçları, görev yönetimi araçları, görev panosu) aktarılmalı ve/veya özetlenmelidir. Elbette, ekibin durum bilgisine ihtiyaç duyduğu sadece test görevlerinin durumu değildir, kullanıcı hikayesinin genel durumunu da bilmesi gerekir. Bir yapılacak iş ögesi, Tamamlanma Tanımı'nı karşıladığında tamamlanmış sayılır. Bir DevOps ekibi Sürekli Dağıtım uygulamasını kullandığında, Tamamlanma Tanımı genellikle başarılı dağıtımı da bir kriter olarak içerir.

2.4.4 SG4 Test Olaylarını Kapanışa Kadar Yönetme

DevOps'ta “olay” terimi genellikle kullanıcıların etkilendiği gerçek bir üretim olayını belirtmek için kullanılır. Bu nedenle, bu paragrafın geri kalanında TMMi bağlamında bir test olayını belirtmek için “kusur” terimini kullanacağız. DevOps ekipleri Operasyonlardan da sorumlu oldukları için hızlı tepki vermelidirler. DevOps içindeki olay yönetimi, bir BT Servis Yönetimi (ITIL) sürecidir: “Olayların olumsuz etkisini, normal servis operasyonunu mümkün olan en kısa sürede geri yükleyerek en aza indirme uygulaması.”

DevOps'ta neredeyse her şey kodlandığından, bulunan tüm hatalar aslında bir yazılımdaki hatalardır. Bir ortam sorunuysa, ortamın Altyapı Kodu'nda (Infra as Code) bir hatadır; yanlış test verisiyse, bir test otomasyon kodu hatasıdır; başka bir yürütme güvenilirliği ise, bir işlem hattı kodu hatası olabilir vb. Önceki paragrafta belirtildiği gibi, DevOps ortamlarında bulunan tüm hatalar kaydedilmez. Bu özel amaç, yalnızca kaydedilen ve bu nedenle kapatılıncaya kadar yönetilmesi gereken hatalar için geçerlidir. Prensipte olarak, DevOps'ta olayları yönetmek basittir. Olay görev panosuna kaydedilirse, bir hikayenin ve görevlerinin tamamlanmasını engelleyen bir hata olarak görselleştirilir. DevOps'un standart çalışma yöntemi, ilerlemeyi engelleyen diğer tüm engeller veya görevlerde olduğu gibi uygulanacaktır. Ayakta toplantılarında tartışılacak ve ekip tarafından çözülmesi için görevlendirilecektir.

Bulunan bir hatanın başka bir yinelemeye ertelenmesine karar verilmesi durumunda, bunlar ürün için sadece başka bir istenen seçenek (veya değişiklik) haline gelir. Bunları yapılacak iş listesine eklenir ve uygun şekilde önceliklendirilir. Öncelik yeterince yüksek olarak belirlendiğinde, bunlar bir sonraki yinelemede ekip tarafından ele alınacaktır.

Günlük (standup - ayakta) toplantılarda ve geriye dönük toplantılarda, bir yineleme sırasında bulunan ve çözülen hata sayısının yanı sıra, bir sonraki yineleme için yapılacak işlerin bir parçası olabilecek çözülmemiş hata sayısının izlenmesi iyi bir uygulamadır.

2.5 Süreç Alanı 2.5 Test Ortamı

Test Ortamının amacı, test verilerini de içeren, testlerin yönetilebilir ve tekrarlanabilir bir şekilde yürütülmesini mümkün kılan yeterli bir ortam oluşturmak ve sürdürmektir.

Test Ortamı süreç alanı ile, testlerin yönetilebilir ve tekrarlanabilir bir şekilde yürütülmesinin mümkün olduğu, genel test verilerini de içeren uygun bir test ortamı oluşturulur ve sürdürülür. Elbette, bir DevOps yazılım geliştirme projesinde de uygun bir test ortamı vazgeçilmezdir. Her CI/CD iş hattı aşaması için bulut teknolojileri kullanılarak, kontrollü bir durumda talep üzerine bir ortam kurulabilir. Yinelemenin kısa döngü süreleri nedeniyle, test ortamının son derece kararlı ve kullanılabilir olması gerekir. Test ortamındaki sorunlar, örneğin CI/CD iş hattında, her zaman yinelemenin ilerleyişi ve sonuçları üzerinde hemen bir etki yaratacaktır. Bu nedenle, test ortamı ve test verilerinin yapılandırmasını ve değişikliklerini doğru bir şekilde yönetmek de son derece önemlidir.

2.5.1 SG1 Test Ortamı Gereksinimlerini Geliştirme

Test ortamı gereksinimlerinin belirlenmesi, projenin erken aşamalarında gerçekleştirilir ve ardından sürekli öğrenme sürecine dayalı olarak devam eder. Ortam gereksinimlerini belirlemek ve ardından

ortam uygulamasını yönlendiren gereksinimleri geliştirmek için ürün değer akışını, üretim ortamını, teknik ortamı ve kullanılan geliştirme araç zincirini anlamak önemlidir. Gereksinimlerin doğru, uygun, uygulanabilir ve “gerçek hayattaki” operasyonel ortamı doğru bir şekilde temsil ettiğini doğrulamak için gereksinimlerin belirlenmesi gözden geçirilir. Erken gereksinimlerin belirlenmesi, DevOps bağlamında sıklıkla kullanılan sanallaştırma, konteynerleştirme (containerization) ve diğer bulut bazlı unsurlar temelinde gerekli test ortamını edinmek ve/veya geliştirmek için daha fazla zaman sağlama avantajı sunar. Yapım iş hattındaki ortamlar (iş hattının CI kısmı) genellikle küçük ve basit bir ölçekte dağıtılır ve artan performans ve güvenilirlik beklentileri doğrultusunda sürekli olarak geliştirilir. Ancak, sürüm iş hattında (iş hattının CD kısmı) uçtan uca iş süreci testine ve buna bağlı olarak çok daha karmaşık bir test ortamına ihtiyaç duyulabilir.

Test ortamlarının ihtiyaçları, CI/CD kılavuzlarında veya platform ekibi tarafından tanımlanan Altyapı Kod Olarak (Infrastructure as Code) ve Yapılandırma Kod Olarak (Configuration as Code) uygulamaları ve standartları doğrultusunda, iyileştirme oturumları sırasında belirlenir ve belirtilir. Test ortamının önceliği ve veri gereksinimleri, ilgili yapılacak iş önceliğine göre belirlenir. Bazen, daha karmaşık ortam gereksinimlerini tanımlamak için kabul kriterleri içeren ayrı bir teknik deneme (spike) belirtilir. Gereksinimler genellikle, kuruluşun CI/CD uygulamaları ve terminolojisine uygun olarak, Altyapı/Yapılandırma-Kod uyumlu dillerde yazılır. Gereksinimler, değer akışı veya CI/CD iş hattı aşamalarına eşleştirilir. Geliştirme araç zincirleri, gerekli ve eksiksiz olmalarını sağlamak için operasyonel ve dağıtım teknolojisiyle uyumlu hale getirilir.

2.5.2 SG2 Test Ortamı Uygulamasını Gerçekleştirme

Dağıtım hattının ve üretim ve test ortamının uygulanması, ilk yinelemenin başlangıcında ortamların ilk sürümünün hazır olması için mümkün olan en kısa sürede başlatılmalıdır. Test ortamları, belirli CI/CD test aşamalarında ilgili test seviyelerini gerçekleştirmek için genel test verilerini içeren, üretim ortamlarına benzer ortamlardır. Verimli DevOps ekipleri, test ortamları için de üretim otomasyonunu yeniden kullanır ve bunları talep üzerine dağıtır. Ortam uygulamasıyla ilgili tüm görevler, diğer tüm bekleme listesi öğeleri gibi bekleme listesi içinde ele alınmalıdır.

DevOps bağlamında, Kod Olarak Altyapı (Infrastructure-as-Code) genellikle ortam bileşenlerini tanımlamak ve uygulamak için kullanılırken, Kod Olarak Yapılandırma (Configuration-as-Code) ise ortam değişkenlerini ve yapılandırmalarını hem otomatik belirler hem de işlem hatları aracılığıyla tanımlamak ve uygulamak için kullanılır. CI/CD iş hatları içinde kendi kendine servisleri kullanarak isteğe bağlı ortam sağlama ve keşif testi oturumları için de yeterliliğe sahip olmak genellikle önemlidir. Kodlama ortamları için genel test verileri, otomatize test verisi oluşturma ve yönetim araçları kullanılarak oluşturulur. Duman testi genellikle bir ortam örneği dağıtıldıktan sonra ve ortam ek etkinlikler için kullanılmadan önce iş hattı içinde çalıştırılır.

2.5.3 SG3 Test Ortamlarını Yönetme ve Kontrol Etme

Test ve üretim ortamlarının yönetimi ve kontrolü, DevOps projesi boyunca gerçekleştirilecektir. Ortamlar, ortamların ve verilerin oluşturulması, yapılandırılması, tahsis edilmesi ve kullanılmasına olanak tanıyan mevcut araçlar kullanılarak kesintisiz test yürütülmesine olanak sağlayacak şekilde yönetilir ve kontrol edilir. Test ortamlarında, provizyon, yapılandırma, tahsis, kullanım ve servisten çıkarma işlemlerini desteklemek için gerçekleştirilen sistem yönetimi, aşağıdaki uygulamalardan oluşur:

- Altyapı kod, otomasyon ve izleme yoluyla yönetmek

- Giriş bilgilerini sağlayarak test ortamına erişimi yönetmek
- Test yürütme sırasında ilerlemeyi engelleyen sorunlar hakkında teknik destek sağlamak
- Test sonuçlarını ve kusurların nedenlerini analiz etmek için kullanılacak izleme ve telemetri sağlar.

Ayrıca, test sürecini desteklemek için test verilerinin sağlanması, tahsisi, değiştirilmesi ve kullanımı mümkün olduğunca yönetilir ve otomatikleştirilir.

Test ortamlarının oluşturulması ve tahsis edilmesi, maksimum kullanılabilirlik ve verimlilik elde etmek için otomatikleştirilir, koordine edilir ve ölçülür. Özellikle kalıcı ortamlar için, ortam örneklerinin kendi kendine veya isteğe bağlı olarak sağlanması (başlatma/durdurma) ve izlenmesi uygulanır. Test ortamlarının oluşturulması veya kullanılması sırasında ortaya çıkan sorunlar kaydedilir ve DevOps ekipleri tarafından çözüme kavuşturulur. Test ortamı olayları, bir sorun gözlemlendiğinde, otomatik olarak başarısız testlerin bir parçası olarak veya ortam kullanıcıları tarafından kaydedilir.

Test Ortamı süreç alanı için ana sonuç olarak, belirli hedefler ve uygulamalar hala geçerlidir ve özünde bir değişiklik yoktur.

3 TMMi Seviye 3 - Tanımlanmış

3.1 Süreç Alanı 3.1 Test Organizasyonu

Test Organizasyonu süreç alanının amacı, testlerden sorumlu olan yüksek vasıflı kişileri belirlemek ve organize etmektir. Test grubu, testlerin yanı sıra, kuruluşun mevcut test süreci ve test süreci varlıklarının güçlü ve zayıf yönlerini kapsamlı bir şekilde anlayarak, kuruluşun test süreci ve test süreci varlıklarına yönelik iyileştirmeleri de yönetir.

3.1.1 SG1 Test Organizasyonu Oluşturma

Test Organizasyonu, sıklıkla yanlış anlaşılan bir süreç alanıdır. Birçok kişi bunu, TMMi'nin bağımsız test yapan bağımsız bir test grubu veya hatta departman gerektirdiği şeklinde yorumlar. DevOps ekipleri, akışı optimize etmeye ve geri bildirim artırmaya odaklanarak işlevler arası çalışacak şekilde tasarlandığından, bağımsız bir test organizasyonu bu kavramı bozar. Dolayısıyla, burada bahsedilen yüksek vasıflı test profesyonelleri grubu genellikle işlevler arası ekiplere ve farklı rollere dağılmıştır. Tipik olarak, bir test yetkinlik merkezi veya test birliği, beceriler ve kariyer yolları da dahil olmak üzere kurumsal test süreçlerini ve mesleği iyileştirmeye odaklanırken, uzun süreli Scrum ekipleri değer sunumunu optimize etmeye odaklanır. Her ikisi de iyileştirme mekanizmalarıdır.

Sözde test yetkinlik merkezinin tipik görevleri ve sorumlulukları, kuruluş genelinde standart olan test süreçlerini oluşturmak, yönetmek ve iyileştirmek ve test deneyimi, bilgisi ve becerisi olan üyeleri işlevler arası DevOps ekiplerine atamaktır. Bir test yetkinlik merkezi genellikle test metodolojileri ve süreçleri konusunda yetki sahibidir. Günlük olarak çalışan bir test uzmanı genellikle bir DevOps ekibinin parçasıdır ve bu nedenle CI/CD iş hattında sürekli testlere katılır, ancak bunun yanı sıra bir test uzmanının test mesleğine odaklanan test yetkinlik merkezi veya test birliği gibi bir test organizasyonuna ait olması da yaygındır.

Bir test organizasyonu, test ve kalite konularıyla ilgili alanlarda liderlik gerektirir. Ancak, testler değer akışının ayrılmaz bir parçası olarak görülmeli ve testlerle ilgili faaliyetler DevOps organizasyonunun

rollerine yayılmalıdır. Gerekli rollerde test becerilerini oluşturmak için, bir test uzmanı tarafından kalite desteği uygulanır. Test yetkinlik merkezi seçeneği DevOps ile uyumludur ve testlerin bir disiplin olarak korunmasını ve bu şekilde ciddiye alınmasını sağlar. DevOps bağlamında başka bir seçenek de, test organizasyonunun bir test loncası formatını almasıdır. Test loncasının üyeleri genellikle farklı DevOps ekiplerinden oluşan, gayri resmi, kendi kendini yöneten bir test uzmanları grubudur. Test loncasının önemli faaliyetleri arasında bilgi paylaşımı, grup halinde öğrenme, trendleri takip etme vb. yer alabilir. Lonca üyeliği gönüllülük esasına dayalı olabilir. Test birliği, test kariyer yolları oluşturmak, eğitim düzenlemek ve test süreci iyileştirmelerini belirlemek, planlamak ve uygulamak gibi daha resmi görevler de alabilir. Test birliğinin başarılı olması için önemli bir kısıtlama, üyelerine gerçekleştirilecek çeşitli test birliği faaliyetlerine ayrılacak zamanın tahsis edilmesidir.

3.1.2 SG2 Test Uzmanları için Test Fonksiyonları Oluşturma

Test, değerli bir meslek ve disiplin olarak kabul edilir. DevOps ekibi, test bilgisi ve becerilerine ihtiyaç duyar. Test uzmanı, test faaliyetlerini gerçekleştirirken bunları sağlar ve diğer ekip üyelerini eğitir. Tipik bir DevOps test uzmanı için gerekli bilgi ve beceriler, geleneksel bir organizasyondakinden farklıdır. Test uzmanı elbette “geleneksel” test bilgisi ve becerilerine de ihtiyaç duyacaktır, ancak buna ek olarak genellikle test otomasyonu için bilgi ve becerilere de ihtiyaç duyar ve test dışındaki görevleri, örneğin komut dosyası yazma veya gereksinim mühendisliği gibi görevleri de yerine getirebilir.

DevOps ile Davranış Odaklı Geliştirme (BDD) ve Kabul Testi Odaklı Geliştirme (ATDD) popüler yaklaşımlardır ve bu nedenle bunlar da DevOps test uzmanlarının beceri setinin bir parçası olmalıdır. Ek olarak, CI/CD iş hattı tasarımı ve yapılandırması, dağıtım otomasyonu, yapılandırma yönetimi ve dağıtım stratejileri de test uzmanlarının yetkinliklerinin bir parçası olmalıdır. DevOps test uzmanı bir ekip içinde çalıştığı için, teknik becerileri geliştirmek kadar sosyal beceriler de aynı derecede önemlidir. DevOps test uzmanı, test işlevlerinin tanımında da yansıtılması gereken, çok sayıda güçlü bilgi alanına sahip, T şeklinde (T-shape) bir test uzmanıdır. Bazı test uzmanları teknoloji konusunda daha güçlü olabilirken, diğerleri tasarım odaklı düşünme ve müşteri anlayışına daha fazla odaklanabilir. Bazı DevOps tabanlı kuruluşlar üç farklı türde test işlevi oluşturmuştur:

- Test Yazılım Geliştirme Mühendisi (SDET) – DevTest üyelerini destekleyen test ve dağıtım otomasyon çözümleri ile ürünün test edilebilirlik özelliklerini geliştirir.
- DevTest – ürünü test etmekten de sorumlu olan geliştiriciler, gereksinimleri analiz edebilen, testler tasarlayabilen, bunları uygulayabilen ve yürütebilen (genellikle SDET tarafından sağlanan mevcut test otomasyon araçlarını ve test edilebilirlik işlevlerini kullanarak) geliştiriciler.
- Test Uzmanı (veya Test Mimarı) – DevOps ekipleri arasındaki test faaliyetlerini denetler, gerektiğinde riskleri belirler ve yönetir, ürüne odaklanan özel testler de yürütür, ancak esas olarak genel test çalışmaları hakkında bilgilendirilir.

Test organizasyonu, hangi formatta olursa olsun, DevOps bağlamında iyi test uzmanları olmak için gerekli beceri ve motivasyona sahip kişilerden oluşur. Bu kişiler belirli bir test fonksiyonuna atanır (yukarıdaki örnekler). Özellikle DevOps bağlamında, tüm ekip test faaliyetlerini gerçekleştireceğinden, test faaliyetleri, rolleri, sorumlulukları ve gerekli test bilgi ve becerileri ile ilgili beklentileri içeren, testle ilgili olmayan diğer fonksiyonların açıklamalarını ayrıntılı olarak hazırlamak büyük önem taşır.

3.1.3 SG3 Test Kariyer Yollarının Oluşturulması

Test kariyer yolları, test uzmanlarının bilgi, beceri, statü ve ödülleri geliştirmelerine ve böylece kariyerlerinde ilerlemelerine olanak sağlamak için oluşturulmuştur. Örneğin, bir DevOps test mühendisinden DevOps test mimarı, test koçu veya ürün sahibi olmaya, müşteri anlama becerilerini geliştirmeye veya kişilerarası becerileri geliştiren Scrum Master olmaya kadar. Kariyer yolu, test alanıyla sınırlı olmak zorunda değildir; bu da uzun vadede tüm organizasyonda kalite bilincinin yayılmasına yardımcı olacaktır. Tanımlanan test kariyer yollarına dayalı olarak, test organizasyonunun her üyesi için kişisel test kariyer geliştirme planları oluşturulur ve sürdürülür. Bu özel hedef ve ona eşlik eden özel uygulamalar, DevOps bağlamında da tamamen geçerlidir. DevOps ekibi, zorlu işlerini anlayan, diğer ekip üyelerine koçluk yapabilen ve motivasyonu yüksek test uzmanlarına ihtiyaç duyar. Bunların çoğu, test kariyer yolları ile ele alınmaktadır.

Ancak, test kariyer yollarının gerçek tanımı, DevOps organizasyonunda, sıralı yaşam döngüsünü izleyen bir organizasyondakinden biraz farklı olacaktır. Geleneksel bir organizasyonda test kariyer yolları genellikle dikey büyümeye (test uzmanından test yöneticisine) odaklanırken, DevOps organizasyonundaki test kariyer yolları yatay büyümeye (junior test uzmanından senior test uzmanına, test mimarına ve belki de test koçuna) veya Scrum Master veya ürün sahibi gibi rollere büyümeye odaklanma eğilimindedir.

3.1.4 SG4 Test Süreci İyileştirmelerini Belirleme, Planlama ve Uygulama

Test iyileştirmeleri, DevOps organizasyonunun sürekli iyileştirme zihniyetinin ve yalın ilkeler ile sistem düşüncesine odaklanan Değer Akışı Yönetimi çerçevesinin temel bir parçasıdır.

Ana fikirler, akışı güçlendirmek, israfı ortadan kaldırmak, geri bildirim güçlendirmek, DevOps iş hattından toplanan gerçeklere dayanmak (örneğin Akış Metrikleri veya dört DORA Performans metriği) etrafında gruplandırılmıştır. Diğer iyileştirmeler gibi, test iyileştirmeleri de kuruluşun yapılacak işler listesinde kaydedilir, önceliklendirilir ve yönetilir.

Bu özel TMMi hedefi ve özel uygulamaları tarafından açıklanan iyileştirme süreci, büyük ölçüde DevOps ekibi tarafından gerçekleştirilen geriye dönük toplantılarla ve iş hattından elde edilen verileri kullanan veri ve gerçeklere dayanarak ele alınmaktadır. Geriye dönük toplantı, DevOps ekibinin kendisini denetlemesi ve iyileştirmeler için eylemler belirlemesi için bir fırsattır. Geriye dönük toplantılar genellikle yineleme incelemesinden sonra ve bir sonraki yineleme planlamasından önce gerçekleştirilir. İş hattının bir parçası olarak, tasarım metrikleri belirlenir ve veri toplama kurulumu, böylece gecikmelerin, kısıtlamaların veya ürün kalitesi sorunlarının nerede meydana geldiğini anlamak için değer akışı haritasında görselleştirilebilir. Bu metrikler, (test) iyileştirme fırsatlarının belirlenmesini sağlar. DevOps ekipleri, test yürütme süreleri, regresyon testi başarısızlık oranları veya test ortamı dağıtım ve yapılandırma süreleri gibi kendilerine daha yakın faaliyetleri ölçmeye odaklanırken, daha büyük organizasyonlar Akış Metrikleri ve dört DORA performans metriğine odaklanmaktadır.

Geriye dönük değerlendirme sırasında ekip, yinelemede nelerin iyi gittiğini, nelerin iyileştirilebileceğini ve bir sonraki yinelemede iyileştirilmek üzere nelerin taahhüt edileceğini (kısmen metriklerine dayanarak) tartışır. Problem çözme becerileri de yararlıdır, örneğin Ishikawa ile kök neden analizi veya problemleri analiz etmek için 5 neden tekniği. Sürekli öğrenme kültürü, bireyleri DevOps için sürekli olarak bilgi edinmeye ve deneyimlerini ve fikirlerini paylaşmaya teşvik eder. Her retrospektif sırasında, DevOps ekibi, iş süreçlerini iyileştirerek veya Hazırlık Tanımı veya Tamamlanma Tanımı kriterlerini ekip

veya değer akışı düzeyinde uyarlayarak ürün kalitesini artırmanın yollarını belirler ve planlar. Geriye dönük değerlendirmeler, test etkinliği, test verimliliği ve test kalitesine odaklanan testle ilgili iyileştirme kararlarına yol açabilir. Ayrıca uygulamaların, kullanıcı hikayelerinin, özelliklerin veya sistem arayüzlerinin test edilebilirliğini de ele alabilirler. Tüm ekip üyeleri, testçiler ve testçi olmayanlar, hem test hem de test dışı faaliyetler hakkında görüş bildirebilirler. Geriye dönük değerlendirmenin sonunda, DevOps ekibi bir sonraki yinelemede uygulayacağı (test) iyileştirmeleri belirlemiş olmalıdır. Bu iyileştirmelerin bir sonraki yinelemede uygulanması DevOps ekibinin kendisinin sorumluluğundadır, bu nedenle iyileştirmeler yapılacak işler olarak yönetilir. Test uzmanları genellikle test iyileştirmelerinin ekip içinde uygulanmasını sağlamak için bu iyileştirmelerin sorumluluğunu üstlenir. Geriye dönük değerlendirme, bir ekibin proje süresince sürekli olarak gelişmesini ve iyileşmesini sağlayan önemli bir mekanizmadır.

Kapsam genellikle yinelemeyle sınırlı olduğundan, küçük ama sık sık iyileştirmeler yapılır ve bunlar çoğunlukla belirli ekip sorunlarının çözülmesine odaklanır. Bu iyileştirmelerin odak noktası genellikle değer akışı çapında öğrenme ve iyileştirmelerin kurumsallaştırılması değildir. (Test) süreç iyileştirmesinin nasıl organize edildiğine ve yönetildiğine bakıldığında, organizasyonel düzeyde (test) süreç grubuna daha az odaklanıldığı ve ekiplerin kendi kendini yönetmesine daha fazla önem verildiği görülmektedir. Bu kötü bir şey olmasa da, başarılı olan yerel test iyileştirmelerinin organizasyonun geri kalanıyla paylaşıldığı ve DevOps ekibinin yetkisi dışındaki iyileştirme alanlarının da ele alınabileceği bir mekanizma oluşturmak önemlidir. Olası bir çözüm, test organizasyonunun bir temsilcisinin (test yetkinlik merkezi veya test guild) yerel retrospektif toplantılara katılması ve iyileştirmelerin uygun olduğunda organizasyon genelinde paylaşılması ve kurumsallaştırılmasını sağlaması olabilir.

Daha büyük kuruluşlar genellikle değer akışı düzeyinde özel geriye dönük etkinlikler düzenleyerek potansiyel sistem düzeyindeki sorunları belirler. Test süreci iyileştiricisi, kuruluş genelinde testleri etkileyen daha geniş kapsamlı temel sorunları veya girişimleri de ele alabilir ve bunları değer akışının yapılacak işlerine ekleyebilir.

3.1.5 SG5 Organizasyonel Test Sürecini Yaygınlaştırma ve Öğrenilen Dersleri Entegre Etme

Ayrıca, bir DevOps projesi için, organizasyonel standart test sürecine ve test süreci varlıklarına gerektiği kadar erişebilmek ve bunları yeniden kullanabilmek ve katma değer yaratmak önemlidir. Bunlar elbette DevOps çalışma şekliyle uyumlu olmalı, kalite güvencesi ve her seviyedeki yapılacak işlerin test edilmesine odaklanmalıdır. Test organizasyonu, bunların tüm ekiplere yayılmasını sağlayacaktır. (Organizasyonel standart test süreci ve test süreci varlıkları ile bunların DevOps projeleriyle ilişkisi hakkında daha fazla bilgi için Test Yaşam Döngüsü ve Entegrasyon süreci alanına bakın. Organizasyonun standart test sürecinin uygulanması ve DevOps projelerinde test süreci varlıklarının kullanımı, kendi kendini organize eden ekipler tarafından, örneğin geriye dönük değerlendirmelerde bu konuyu ele alarak izlenebilir. İleriye dönük harika bir yol, süreçleri ve şablonları araçlara yerleştirmek ve uygulamaları iş hatları tarafından zorunlu kılmaktır; örneğin, kapsama hedefleri yerine getirilmezse veya kararlaştırılan kodlama standartları ihlal edilirse, tüm çekme isteklerinin otomatik olarak reddedilmesi gibi. Test organizasyonunun bir temsilcisi, örneğin bir test süreci iyileştiricisi, bu toplantılara katılarak uygulama durumunu takip edebilir ve projeler arası sorunları ele alabilir. Bu kişi ayrıca DevOps ekibinden öğrendiği değerli dersleri ve test iyileştirmelerini test organizasyonuna geri götürebilir. Öğrenilen dersler ve test iyileştirmeleri daha sonra test süreci iyileştirme önerileri olarak sunulabilir ve buna göre yönetilebilir.

3.2 Süreç Alanı 3.2 Test Eğitim Programı

Test Eğitim Programı süreç alanının amacı, test görevlerinin ve rollerinin etkili ve verimli bir şekilde yerine getirilebilmesi için çalışanların bilgi ve becerilerinin geliştirilmesini kolaylaştıran bir eğitim programı geliştirmektir.

Test Eğitim Programının temel amacı, test uzmanlarına ve ekip içindeki ve dışındaki test sürecine dahil olan diğer kişilere gerekli (test) eğitimini sağlamaktır. Etkili bir eğitim programı, testlere katılanların test bilgilerini ve becerilerini geliştirmeye devam etmelerini ve testlerle ilgili gerekli alan bilgisini ve diğer bilgi ve becerileri edinmelerini sağlar. Kalite ve testler bir ekip sorumluluğu olduğundan, testlerle ilgili eğitimin sadece test uzmanlarına değil, tüm DevOps ekibine verilmesi beklenir.

3.2.1 SG1 Organizasyonel Test Eğitim Yetkinliği Oluşturma

Süreç, kuruluşun stratejik test eğitimi ihtiyaçlarının belirlenmesi ile başlar. Stratejik test eğitimi ihtiyaçları, güvenlik hususlarının ele alınmasını sağlarken, sürekli entegrasyon ve sürekli teslimat süreçleri içinde test uygulamalarını etkin bir şekilde hayata geçirmek için ekiplere gerekli beceri ve bilgileri kazandırmaya odaklanır. Test uzmanları için geleneksel ve Çevik eğitim ihtiyaçlarına ek olarak, örneğin test tasarım teknikleri, keşif testleri ve kapsama ölçümleri gibi, artık DevOps testlerinin özellikleri hakkında bilgi ve beceri edinme ihtiyacı da bulunmaktadır. Örnekler arasında test otomasyon araçları ve çerçeveleri, güvenlik testleri (örneğin, Statik Uygulama Güvenliği Testi (SAST), Dinamik Güvenlik Analizi Testi (DAST) ve sızma testi), Altyapı-kod, shift-left test uygulamaları, Test Odaklı Geliştirme (TDD), Davranış Odaklı Geliştirme (BDD) ve Kabul Testi Odaklı Geliştirme (ATDD) sayılabilir. Stratejik test eğitim ihtiyaçlarının belirlenmesi sadece test uzmanlarıyla sınırlı değildir; DevOps ekiplerinin diğer üyeleri için de testle ilgili eğitim ihtiyaçlarını belirlemek son derece önemlidir.

Diğer kuruluşlarda olduğu gibi, test eğitimi ihtiyaçları (hafif) bir eğitim planına dönüştürülecek ve muhtemelen bazı özel proje test eğitimi ihtiyaçları da ele alınacaktır. Eğitim planı gözden geçirilmeli ve taahhütler belirlenmelidir. Özellikle DevOps'ta bazı becerilerin gayri resmi araçlar (örneğin, işbaşı eğitimi, öğle yemeği eğitim oturumları, koçluk ve mentorluk) aracılığıyla etkili ve verimli bir şekilde aktarıldığı, diğer becerilerin ise resmi eğitim gerektirdiği unutulmamalıdır. Ayrıca, Bilgiyi Yerleştirme ve Sürekli Deneyimleme gibi DevOps uygulamaları, eğitim programlarının geliştirilmesi ve sunulması için izlenecek yaklaşımı belirleyecektir.

3.2.2 SG2 Gerekli Test Eğitimlerini Sağlama

Genel olarak, SG2 "Gerekli Test Eğitimi Sağlama" için belirli uygulamalar, geleneksel ve DevOps organizasyonlarında aynıdır. Organizasyonel eğitim planına göre, test görevlerini etkili bir şekilde yerine getirebilmek için eğitim alması gereken ekip üyeleri belirlenir. Ardından, gerekli kaynaklar da dahil olmak üzere eğitim planlanır ve gerçekleştirilir.

Yukarıda belirtildiği gibi, DevOps organizasyonlarında genellikle gayri resmi eğitim araçları kullanılır. Örneğin, resmi eğitime ek olarak, iş başında mentorluk ve koçluk genellikle sürekli olarak teşvik edilir ve uygulanır. Aslında, mentorluk herkesin sorumluluğudur ve organizasyonun tüm kademelerinde beklenir. İkili çalışma, ekiplerin beceri geliştirme ve bilgi aktarımı için sıklıkla kullandıkları bir başka araçtır.

Gerçekleştirilen test eğitiminin kayıtları, ister resmi ister gayri resmi olsun, oluşturulacak ve test eğitiminin etkinliği değerlendirilecektir. Katılımcıların bilgilerinin ve becerilerinin test görevlerini yerine getirmek için artık daha uygun olup olmadığına dair bir tartışma yapılan geriye dönük toplantının bir parçası olarak, katıldığı (test) eğitimi de değerlendirilebilir.

3.3 Süreç Alanı 3.3 Test Yaşam Döngüsü ve Entegrasyon

Test Yaşam Döngüsü ve Entegrasyonun amacı, kullanılabilir bir dizi organizasyonel test süreci varlığı ve çalışma ortamı standardı oluşturmak ve sürdürmek, ayrıca test yaşam döngüsünü geliştirme yaşam döngüsüyle entegre etmek ve senkronize etmektir. Entegre yaşam döngüsü, tüm değer akışı boyunca testlerin erken aşamada dahil edilmesini sağlar. Test Yaşam Döngüsü ve Entegrasyonunun amacı, belirlenen risklere ve tanımlanan test stratejisine dayalı olarak, birden fazla test düzeyinde (DevOps bağlamında CI/CD iş hattındaki aşamalar olarak temsil edilir) tutarlı bir test yaklaşımı tanımlamak ve tanımlanan test yaşam döngüsüne dayalı olarak genel bir test planı sağlamaktır.

3.3.1 SG1 Organizasyonel Test Süreci Varlıklarını Oluşturma

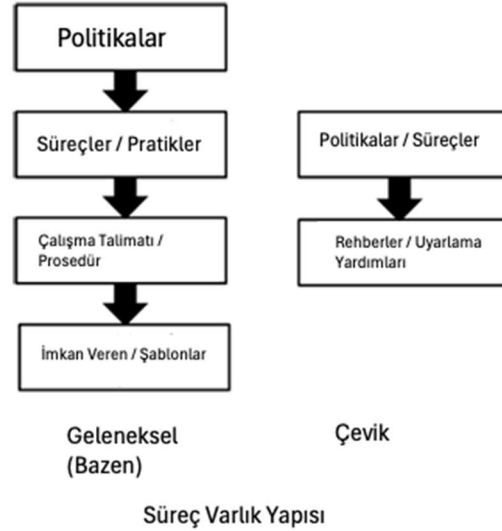
Test organizasyonunun önemli bir sorumluluğu, organizasyonun test politikası ve hedefleri doğrultusunda standart bir test süreci tanımlamak, belgelemek ve sürdürmektir. Diğerlerinin yanı sıra, bu süreç, yürütülecek çeşitli test seviyeleri ve test türlerinin bir açıklamasını içerir. TMMi, ayrıntılı olarak dokümanite edilmiş test süreçlerini zorunlu kılmadığından, genellikle örneklerle detaylandırılmış şablonlar, ortak bir çalışma yöntemi oluşturmak için harika bir yaklaşımdır [Galen]. Ekiplerin bir projede uygulayacakları test seviyeleri ve test türleri üzerinde anlaştıkları Çevik Test Çeyrekleri, üzerinde anlaşmaya varılan test sürecini ve stratejisini temsil etmenin basit ve üst düzey bir yoludur. Zorunlu test seviyeleri daha sonra “Tamamlanma Tanımı”nda kriterler olarak belirtilir. CI/CD iş hattındaki test aşamalarının üst düzey bir görünümü de iyi bir ortak belge görevi görür.

Bir şablon, yapısı içinde gösterilen bilgileri toplamak ve sonuçları belgelemek için gerekli faaliyetleri tanımlar. Bir süreç, faaliyetlerin katı bir sırası olmak zorunda değildir. Bağımlılıklar mevcutsa, bunlar şablondaki notlarla kaydedilebilir. Şablonlar, bir sürecin gerçek amacını aktaran pratik bir değere sahiptir. Şablonlar ayrıca, “laf kalabalığı” içeren süreç dokümanlarında sıklıkla görülen belirsizlikleri de ortadan kaldırır. DevOps organizasyonu olarak etkili süreçleriniz varsa, bunları basit bir şekilde belgelemeniz ve belki birkaç şey eklemeniz yeterli olabilir. Genel olarak, DevOps organizasyonları yoğun otomasyona dayanır ve süreçler ile kurallar genellikle araçlarda kaydedilir ve bu araçlar tarafından zorunlu kılınır. Bu nedenle, test süreçleri her zaman ayrı dokümanlar olarak mevcut olmayabilir, ancak yine de geliştirme ve dağıtım sürecinin önemli bir parçasıdır. Elbette, bu çalışma yönteminde de sürece dahil olan tüm tarafların uygulanan süreçler hakkında ortak bir anlayışa sahip olması önemlidir. DevOps organizasyonlarında süreçler oluşturulurken sıklıkla kullanılan bazı temel kılavuzlar:

- “Yapılması gerekenler” süreci, kılavuzlardan ayrı olarak paketlenmiştir.
- Bir süreç genellikle iki sayfadan fazla değildir.
- Süreçler, tüm projelerde, boyut ve ölçekten bağımsız olarak bunu zorunlu kılmaya karar verilmedikçe, “nasıl yapılır” bilgisi veya araç bilgisi içermez.
- Ayrı kılavuzlar, uyarılma/planlama seçenekleri ve “nasıl yapılır” bilgilerini içerir. “Nasıl yapılır” bilgileriyle, mevcut bir kitap veya makaleye atıfta bulunmaya da karar verilebileceğini unutmayın.

Büyük kuruluşlar, standart yapılacak iş akışları veya CI/CD iş hattı iş akışlarının zorunlu olduğu platform mühendisliği uygulamalarını kullanabilir. Bu, tanımlanan süreçlerin ekipler tarafından kullanılacak özel araçlarda uygulandığı anlamına gelir. Özelleştirme mümkünse, yaşam döngüsü aşaması isteğe bağlıdır.

Birçok büyük kuruluşta, politikalar, süreçler/uygulamalar, iş talimatları/prosedürler ve kolaylaştırıcılar/şablonlar gibi dört düzeyde süreç varlıkları görmek yaygın bir durumdur. Bunun nedeni TMMi'nin gerektirdiği bir şey değil, birçok büyük kuruluşun süreç varlıklarını uygulama şeklidir. Süreç varlıklarının seçimi her kuruluşa bağlı olmakla birlikte, birçok DevOps kuruluşu artık iki düzeyde süreç varlığının yeterli olduğunu deneyimlemektedir. Bu, bir politika beyanını, süreci yürütürken “yapılması gerekenleri” özetleyen ilgili süreç açıklamasıyla birleştirilerek gerçekleştirilir. İkinci seviye, süreci yürütmek ve uyarlama için “nasıl yapılır” kılavuzlarını içerir. Bu seviye, süreci uyarlama için bir yardımcı olarak görülebilir, ekibe tanımlanmış çerçeve içinde tam olarak nasıl çalışacağına karar verme özerkliği verir ve genellikle destekleyici şablonlar içerir. Çoğu DevOps kuruluşunda, adım adım prosedürler araç kılavuzları ve eğitim/mentorluk ile değiştirilmiştir. Yukarıda belirtildiği gibi, test şartı şablonu, test oturumu sayfası şablonu veya test planı şablonu gibi bir şablon, şablonda ima edilen gerekli süreç faaliyetlerini içeren bir süreç olarak servis edebilir. Bu, etkili Çevik süreç açıklamaları geliştirmek için yaygın olarak kullanılan bir tekniktir. [McMahon]



Kuruluşun standart test süreçleri, projeler veya ekipler tarafından özel olarak tanımlanmış süreçler oluşturmak üzere uyarlanabilir. SP 1.3 Uyarlama kriterleri ve kılavuzları oluşturma bağlamında, birçok kuruluş “uyarlama” yaklaşımını kullanır. Bu yaklaşım, insanların neden bir şeyin gerekli olmadığını açıklamak için efor sarf etmelerini gerektirir. Uyarlama yaparken, ne kadar ileri gidebilirsiniz? Tanımlanmış bir minimum var mı? Ancak, herkesin yapması gereken minimum setle başlarsanız, “yapılması gereken” bir şeyi uyarlama riskini ortadan kaldırırsınız. “Yukarı doğru uyarlama” yaklaşımı, Çevik ve DevOps yaklaşımlarıyla çok daha tutarlıdır ve TMMi ile tamamen uyumludur. Bir dizi basit kriter, yukarı doğru uyarlama yaparken doğru soruları sormaya yardımcı olabilir. Destekleyici kriterler olmadan, bazıları yukarı doğru uyarlama yapma eğilimindedir.

Standart test sürecinin uygulanmasını desteklemek için, genellikle bir wiki üzerinde bir test süreci varlık kütüphanesi oluşturulacak ve sürdürülecektir. Bu kütüphanede test uzmanları, günlük çalışmalarında kendilerine yardımcı olacak şablonlar, en iyi uygulamalar ve kontrol listeleri bulabilirler. Genellikle bu şablonlar, örneğin bir CI/CD iş hattı planı, zorunlu alanlar veya yapılacak iş öğesi yönetim aracındaki şablonlar gibi araç uygulamalarının bir parçasıdır. Ekiplerin test faaliyetleri sırasında sağlanan varlıkları kullanırken katma değer elde etmesi koşuluyla, bir test süreci varlık kitaplığı oluşturmak DevOps bağlamında da anlamlıdır. Aşağı yukarı aynı mantık, tahminler, kapsam, kalite ölçümleri gibi test verilerinin toplandığı ve ekiplerin kullanımına sunulduğu bir test süreci veri tabanı için de geçerlidir.

Toplanan ve kullanıma sunulan her şeyin ekipler arası öğrenmeye ve deneyim paylaşımına olanak tanıdığından ve ekipler için katma değer sağladığından emin olun.

Elbette, DevOps bağlamında da çalışma ortamı standartları, proje çalışma ortamlarının oluşturulmasında kılavuz olarak kullanılır.

3.3.2 SG2 Test Yaşam Döngüsü Modellerinin Geliştirme Modelleri ile Entegrasyonu

Standart test yaşam döngüsü modeli, çeşitli test düzeyleri için ana faaliyetleri ve çıktıları tanımlar. Standart test yaşam döngüsü modeli, aşamalar, kilometre taşları, çıktılar ve faaliyetler açısından test faaliyetlerini entegre etmek için geliştirme yaşam döngüsü modeliyle uyumludur. Yaşam döngüsü entegrasyonu, her bir yapılacak iş ögesi için testlerin projelere erken aşamada dahil edilmesi sağlanacak şekilde gerçekleştirilir. Elbette, DevOps ile geliştirme ve testler yaşam döngüsü içinde tamamen entegre edilmeli ve testler mümkün olduğunca erken aşamada gerçekleştirilmelidir. Test uzmanları ekibin bir parçası olarak yineleme planlamasına katıldıklarından, riskler önceden belirlenir ve test faaliyetleri üzerinde anlaşmaya varılır, tahminler ve taahhütler etkilenir ve testlerin yaşam döngüsüne entegrasyonu sağlanır.

Bu özel hedef, DevOps yazılım geliştirme yapan bir organizasyon için de geçerlidir. DevOps bağlamında, geliştirme ve testler elbette ekip düzeyinde tamamen entegre olmalı ve bunun nasıl yapılacağı konusunda ortak bir anlayış bulunmalıdır. Bu gibi durumlarda, ek bir şey yapılması gerekmez. Diğer durumlarda, entegrasyon ve erken katılım henüz tam olarak uygulanmıyorsa, bu özel hedef çok önemli bir iyileştirme alanıdır ve yinelemede gerçekleştirilen geliştirme ve test faaliyetlerinin çok daha iyi uyumlu hale gelmesini sağlamalıdır.

3.3.3 SG3 Ana Test Planı Oluşturma

TMMi seviye 3'te, testler, tüm test seviyelerinde test görevlerinin, sorumlulukların ve test yaklaşımının koordinasyonunu ele alan ana test planlaması ile ilgilidir. Bu, çeşitli test seviyeleri arasında gereksiz yinelemeleri veya testlerin atlanmasını önler ve genel test sürecinin verimliliğini ve kalitesini önemli ölçüde artırabilir. Ana test planı, gerçekleştirilecek belirli seviyeler ve bu seviyeler arasındaki ilişki dahil olmak üzere, belirli bir proje için test stratejisinin uygulanmasını açıklar. TMMi seviye 2'de, (test) planlama, Test Planlama süreç alanının bir parçası olarak hem sürüm hem de yineleme seviyesinde ele alınmaktadır. Bir DevOps ekibi genellikle birden fazla test seviyesi gerçekleştirir ve bunların tümü, sürüm ve yineleme planlamasının bir parçası olarak uygun şekilde ele alınmalıdır.

Ancak, DevOps yaşam döngüsü modeline göre çalışan bazı projeler, DevOps ekibi içindeki testlere ek olarak, bazı testlerin ekipler dışında organize edilmesi şeklinde, bir nevi hibrit bir yaklaşım benimsemiştir. Bu gibi durumlarda, ana test planı, DevOps ekiplerinden kendi kapsamları dışında gerçekleştirilen test düzeyleri veya türleri (örneğin, dış kaynaklı fonksiyonel olmayan testler (güvenlik, performans) veya dış taraflarla sistem entegrasyon testleri) ile olan ilişkiyi tanımlayacak ve yönetecektir. Bu genellikle yapılacak iş öğelerinin görev panosunda ek bir aşama olarak gösterilir. Tüm bu belirli hedeflerle ilgili özel uygulamalar da genellikle geçerli olacaktır.

Tüm testler DevOps ekipleri tarafından gerçekleştiriliyorsa, özel bir ana test planının ek bir değeri yoktur ve bu nedenle bu TMMi hedefi bu tür durumlarda muhtemelen geçerli olmayacaktır. Bu durumda, sürüm ve yineleme düzeyinde planlamanın gerekli tüm test yönlerini kapsamaması beklenir.

Ancak, ekibin üzerinde anlaşmaya varılmış ve sürekli olarak geliştirilen genel bir test yaklaşımı olup olmadığını kontrol etmek faydalıdır. Belirli test seviyelerini görmek için “Tamamlanma Tanımı”na, CI/CD veri hatlarına/iş akışlarına veya basit bir Çevik Test Çeyreği belgesine bakmak, bunlar da hafif bir ana test planı olarak algılanabilir.

3.4 Süreç Alanı 3.4 Fonksiyonel Olmayan Testler

Fonksiyonel Olmayan Test süreç alanının amacı, test planlama, test tasarımı ve yürütme aşamalarında fonksiyonel olmayan testleri de içerecek şekilde test süreç yeteneklerini geliştirmektir.

Fonksiyonel olmayan testlerin önemli olup olmadığı, uygulanan yaşam döngüsünden bağımsızdır. Geliştirilen ürün, test için hangi fonksiyonel olmayan yönlerin önemli olduğunu belirler. Bu elbette DevOps geliştirme için de geçerlidir. Fonksiyonel olmayan testler bu nedenle DevOps yazılım geliştirme için de geçerlidir.

Ancak, fonksiyonel olmayan özelliklerin niteliğine ve kullanılan test yaklaşımına bağlı olarak, performans ve güvenilirlik gibi bazı fonksiyonel olmayan özellikler, geleneksel yöntemler kullanılarak kısa bir yinelemede test edilemez. Bu nedenle, fonksiyonel olmayan özellikleri test etmek için genellikle farklı bir yaklaşım gerekir. Bu, DevOps ile daha iyiye doğru yapılan temel değişikliklerden biridir, çünkü kalite özellikleri sonuna kadar bırakılmadan erken aşamada test edilir. Tam bir fonksiyonel olmayan test yapmak için ilk yinelemeden itibaren tüm bileşenler/işlevler mevcut olmayacaktır, ancak bu testin sonuçları kalite sorunlarının erken belirtilerini vermek için kullanılabilir.

Kullanıcı hikayeleri, kullanıcılar ve müşteriler için doğru ürünün geliştirilmesini sağlamak amacıyla hem fonksiyonel hem de fonksiyonel olmayan unsurları ele almalıdır. ISO 25010 kalite özellikleri, gereksinimlerin yapılandırılmasına yardımcı olabilir ve test uzmanları bu fonksiyonel olmayan unsurları dikkate almalıdır [ISO25010]. DevOps ile, standart ISO 25101 kalite özelliklerine ek olarak, gözlemlenebilirlik (telemetri1 ile desteklenir) bir kalite özelliği olarak tanıtılmıştır.

3.4.1 SG1 Fonksiyonel Olmayan Ürün Risk Değerlendirmesi Yapma

DevOps projeleri için ürün riski oturumları ve teknikleri, örneğin risk pokeri, ROAM ve tehdit modelleme, Test Planlama süreç alanının bir parçası olarak belirlenen ve açıklanan belirli hedef SG1 Risk Değerlendirmesi Yapma kapsamında, artık fonksiyonel olmayan yönleri ve riskleri de içerecek şekilde açıkça genişletilecektir. DevOps ile genellikle gözlemlenebilirlik, güvenlik ve uyumluluğa özel bir önem verilir. Kuruluşlar daha dijital ve bulut tabanlı hale geldikçe, BT sistemleri artan riskler ve güvenlik açıklarıyla karşı karşıya kalmaktadır. Tehdit modelleme, özellik yaşam döngüsünün erken aşamasında, analiz/iyileştirme sırasında gerçekleştirilen ve risk değerlendirmesinin yapıldığı, tehditlerin belirlenmesi ve yapılandırılmış bir modelle azaltılması için yapılan bir süreçtir. Sonuç genellikle özellik düzeyinde fonksiyonel olmayan kabul kriterleri olarak yansıtılır ve bu da ek kullanıcı hikayelerinin oluşturulmasına veya özel mimari kararların alınmasına yol açabilir. Yineleme düzeyinde, risk değerlendirme, fonksiyonel olmayan gereksinimleri önemli bir girdi olarak tanımlayan ve güncellenmiş hikaye kabul kriterleri ile sonuçlanan hikaye iyileştirme sırasında kullanıcı hikayeleri düzeyinde gerçekleştirilir. Tercihen, ürün sahibi ve muhtemelen diğer bazı paydaşlar da dahil olmak üzere tüm ekip üyeleri ürün risk oturumlarına katılmalıdır. Bazı fonksiyonel olmayan alanlar için, uzmanların yardımı gerekebilir.

Ürün riski oturumları genellikle öncelikli fonksiyonel olmayan ürün risklerinin belgelenmiş bir listesiyle sonuçlanır ve özellik ve hikaye belgeleri güncellenir. Bu ürün riski listesi, CI/CD iş hattının tasarımını ve fonksiyonel olmayan test yaklaşımını yönlendirmekle kalmaz, aynı zamanda ürünün mimarisini ve

geliştirme için seçilen teknolojiyi de aktif olarak şekillendirir; örneğin, ölçeklenebilirlik ve güvenilirlik için konteynerleştirme kullanır. Test Planlama süreç alanında belirtildiği gibi, DevOps projelerinde kullanılan süreç ve sonuçta ortaya çıkan dokümantasyon, sıralı yaşam döngüsü modelini izleyen geleneksel projelere kıyasla çok daha hafif olacaktır.

3.4.2 SG2 Fonksiyonel Olmayan Test Yaklaşımı Oluşturma

Belirlenen ve önceliklendirilen fonksiyonel olmayan ürün risklerini azaltmak için, ilgili fonksiyonel olmayan kalite özelliklerine yönelik bir test yaklaşımı tanımlanır. CI/CD iş hattı, fonksiyonel olmayan testleri de kapsayacak şekilde tasarlanmalıdır. Belirli bir yineleme için, yineleme planlaması sırasında test edilecek ek fonksiyonel olmayan yönler belirlenirken, CI/CD iş hattındaki fonksiyonel olmayan testler her değişiklik için yürütülür. Ancak, fonksiyonel olmayan kalite özellikleri, gözlemlenebilirliğe dayanan telemetri kullanılarak üretimde dağıtımdan sonra da ölçülür. Servis Seviyesi Göstergeleri (SLI'lar) ve Servis Seviyesi Hedefleri (SLO'lar), DevOps'ta bunu yönetmenin tipik bir yoludur. Test edilecek fonksiyonel olmayan yönlerin önceliklendirilmiş listesi, genellikle yinelemede geliştirilen ve test edilen kullanıcı hikayeleriyle ilgilidir. Yineleme sırasında yeni fonksiyonel olmayan ürün riskleri ortaya çıkabilir ve bu da ek testler gerektirebilir. Ek test gerektiren yeni fonksiyonel olmayan ürün riskleri gibi sorunlar genellikle günlük ayakta toplantılarında tartışılır.

Fonksiyonel olmayan riskleri azaltmak için yineleme düzeyinde tanımlanan fonksiyonel olmayan test yaklaşımı, genellikle fonksiyonel olmayan risklerin düzeyine ve türüne göre uygun fonksiyonel olmayan test yöntemlerinin ve test tekniklerinin belirlenmesini kapsar. Genellikle, destekleyici araçların kullanımı ve fonksiyonel olmayan testler için test otomasyonu yaklaşımını da ele alır. Sürüm düzeyinde fonksiyonel olmayan testler için test yaklaşımı çok daha yüksek bir düzeyde olacak ve tanımlanmış CI/CD iş hattına dayalı olmalıdır. Hem sürüm hem de yineleme düzeyinde fonksiyonel olmayan test yaklaşımı, genel test yaklaşımının bir parçasıdır ve ekip/proje wiki'sinde tutulacak veya görüntülenecektir. Fonksiyonel olmayan testlerin sonuçları ürün panolarında sunulacaktır.

Fonksiyonel olmayan çıkış kriterleri, Tamamlanma Tanımı'nın (DoD) bir parçasıdır. DoD'nin, API yanıt süreleri veya "ön uç web sayfaları OWASP en önemli 10 risk listesi için test edilmiştir" gibi güvenlik beklentileri gibi fonksiyonel olmayan testlerle ilgili belirli kriterlere sahip olması önemlidir. CI/CD iş hattının bir parçası olarak dağıtımdan sonra ölçülen SLI'lar ve SLO'lar da çıkış kriterleri olarak işlev görür ve tipik olarak SLI'ların veya SLO'ların bozulmasını önlemek beklenir. Yineleme, üzerinde anlaşmaya varılan fonksiyonel olmayan kabul kriterleri ve muhtemelen kullanıcı hikayelerinin uygulanmasıyla sonuçlanmalı ve DoD'da tanımlanan fonksiyonel olmayan (test) çıkış kriterlerini karşılamalıdır. Yineleme düzeyinde fonksiyonel olmayanlar için bir DoD bulunmakla kalmaz, genellikle birden fazla yinelemeyi kapsayan özellik veya sürüm düzeyinde de bir DoD bulunur. Ayrıca DevOps bağlamında, dağıtımdan sonra SLI'lar ve SLO'lar kullanılarak sürekli izleme yoluyla üretimde fonksiyonel olmayan özellikleri ölçen Shift Right yaklaşımlarına daha fazla odaklanılır.

3.4.3 SG3 Fonksiyonel Olmayan Test Analizi ve Tasarımı Gerçekleştirme

Bu özel hedef, Test Tasarımı ve Yürütme süreç alanındaki SG1 Test Tasarım Tekniklerini Kullanarak Test Analizi ve Tasarımı Yapma özel hedefi ile büyük ölçüde aynı uygulamaları takip eder, ancak bu sefer fonksiyonel olmayan bir bakış açısıyla. Test analizi ve tasarımı sırasında, fonksiyonel olmayan testler için test yaklaşımı somut test koşullarına ve testlere dönüştürülür. DevOps'ta test analizi ve tasarımı ile test yürütme, genellikle bir yineleme boyunca paralel olarak yürütülen, birbirini destekleyen faaliyetlerdir.

Bu, çoğu fonksiyonel olmayan test için de geçerlidir. Fonksiyonel olmayan test analizi, bu nedenle açıkça ayrı bir faaliyet değil, test uzmanlarının (DevOps kalite mühendisleri) iş birliğine dayalı kullanıcı hikayesi geliştirme ve iyileştirme sürecindeki rollerinin bir parçası olarak gerçekleştirildikleri örtük bir faaliyettir.

Fonksiyonel olmayan kabul kriterlerinin ve kullanıcı hikayelerinin analizine dayalı olarak, fonksiyonel olmayan test koşulları belirlenir. Test koşulları temel olarak test edilmesi/kapsanması gereken “şeylerin” tanımlanmasıdır. Tanımlanan kabul kriterleri yeterince ayrıntılı ve net olduğu sürece, genellikle geleneksel test koşullarının rolünü üstlenirler. Kabul kriterleri daha sonra fonksiyonel olmayan testlere dönüştürülür. Fonksiyonel olmayan testlerde, sadece kullanıcı hikayeleri yerine daha yüksek bir düzeyde test analizi yapmak genellikle faydalıdır. Örneğin, bir özelliği, epik bir hikayeyi veya bir dizi hikayeyi analiz ederek, kullanıcı hikayesi düzeyindekinden daha soyut olan ve birden fazla kullanıcı hikayesini kapsayan fonksiyonel olmayan test koşullarını belirlemek. DevOps ile test öncelikli ilkesinin uygulanmasıyla, fonksiyonel olmayan test koşullarını kapsayan fonksiyonel olmayan testler, kodun geliştirilmesinden önce veya en azından kodun geliştirilmesiyle paralel olarak belirlenir (ve muhtemelen otomatikleştirilir).

Çoğu manuel fonksiyonel olmayan testlerde, testler ekip fonksiyonel olmayan testlerin yürütülmesi ilerledikçe tanımlanacak/geliştirilecektir. Testler çoğu zaman geleneksel projelerde olduğu kadar ayrıntılı olarak belgelenmez, daha çok keşif testleri kullanıldığında test fikirleri formatında belgelenir. Karmaşık ve kritik fonksiyonel olmayan alanlar için, resmi fonksiyonel olmayan test tasarım tekniklerini kullanarak test tasarımı ve test senaryosu geliştirmeye yönelik daha geleneksel bir yaklaşım, riskleri karşılamak için en iyi yol olabilir. Ancak, bu yaklaşımda da, geleneksel sıralı yaşam döngüsü ortamındaki fonksiyonel olmayan testlere kıyasla dokümantasyon miktarı sınırlı olacaktır. Fonksiyonel olmayan testlerin önceliklendirilmesi, genellikle kapsadıkları kullanıcı hikayesinin önceliklendirilmesini takip eder. Ancak, önceliklendirme, belirli bir fonksiyonel olmayan testi hazırlamak ve gerçekleştirmek için gereken süreye de bağlı olabilir.

Gözlemlenebilirlik ve telemetri gereksinimlerini ve hikayelerini belirlemek, üretimdeki fonksiyonel olmayan unsurları ölçmeye yardımcı olmak için fonksiyonel olmayan analiz ve tasarımın bir sonucu da olabilir.

Fonksiyonel olmayan testlerin yürütülmesini desteklemek için gerekli olan belirli test verileri tanımlanır. DevOps'ta, gerekli test verileri genellikle test spesifikasyon belgesinde ilk olarak tam olarak belirtilmez, ancak genellikle kullanıcı hikayesinin bir parçası olarak, örneğin bir persona biçiminde kaydedilir. Bununla birlikte, gerekli araçlar ve/veya işlevler mevcutsa, fonksiyonel olmayan testleri desteklemek için gerekli test verileri, fonksiyonel olmayan manuel testlerin yürütülmesinin hızlı bir şekilde başlatılabilmesi için çoğu zaman anında oluşturulur. Buna karşılık, otomatikleştirilmiş fonksiyonel olmayan testler için verilerin genellikle önceden belirtilmesi gerekir. Ayrıca, fonksiyonel olmayan gereksinimlerin bazıları üretimde test edildiğinden, test verileri de üretim verileri olabilir. Tabii ki, bu durumda gizlilik ve GDPR çok önemlidir.

Fonksiyonel olmayan gereksinimler, test koşulları ve testler arasında izlenebilirlik sağlanmalı ve sürdürülmelidir. Ekipler, testlerinin bir parçası olarak çeşitli fonksiyonel olmayan kullanıcı hikayelerini ve kabul kriterlerini kapsadıklarını açıkça belirtmelidir. Çoğu DevOps kuruluşunda, kullanıcı hikayeleri bir dizi ilgili kabul kriteri ve ardından testler geliştirmek için temel oluşturur. Bu yaklaşım kullanılarak, gereksinimlerden teste yata izlenebilirlik sağlanabilir.

3.4.4 SG4 Fonksiyonel Olmayan Test Uygulamasını Gerçekleştirme

Test uygulaması, testlerin yürütülmesine başlamak için gereken her şeyi hazır hale getirmekle ilgilidir. Genellikle, test yürütülmesini desteklemek için test prosedürleri gibi test belgelerinin geliştirilmesi en aza indirilir. Bunun yerine, otomatikleştirilmiş (regresyon) test komut dosyaları geliştirilir ve önceliklendirilir. DevOps ile otomatikleştirilebilecek her şey otomatikleştirilmelidir.

Fonksiyonel olmayan test uygulaması, Test Tasarımı ve Yürütme süreç alanının bir parçası olarak belirli hedef SG2 Test Uygulamasını Gerçekleştirme'de daha önce açıklanan birçok uygulamayı takip edecektir. Neyin yapılması gerektiği ve fonksiyonel olmayan test uygulamasının nasıl yapılacağı, büyük ölçüde tanımlanan yaklaşıma, kullanılan tekniklere ve hangi fonksiyonel olmayan özelliklerin hangi düzeyde test edilmesi gerektiğine bağlıdır. Örneğin, daha az hazırlık gerektiren keşif testleri, kullanılabilirlik testleri için uygun olabilir, ancak genellikle kapsamlı güvenilirlik ve performans testleri için daha az uygundur.

Bazı fonksiyonel olmayan kalite özellikleri için, test verilerinin mevcudiyeti çok önemlidir ve test uygulaması sırasında oluşturulması gerekir. Bu, geleneksel projelerle büyük ölçüde aynıdır, ancak bunlar kısa bir şekilde belgelendirilebilir ve böylece regresyon testleri için yeniden kullanılabilir. Daha önce belirtildiği gibi, bazı fonksiyonel olmayan gereksinimler üretimde test edildiğinden, test verileri aynı zamanda üretim verileri de olabilir.

Elbette, test uygulaması ve hazırlığı diğer (test) faaliyetleriyle paralel olarak mümkün olan en kısa sürede başlayacaktır. Bu ayrı bir aşama değil, verimli ve etkili bir test yürütülmesi için gerçekleştirilmesi gereken bir dizi faaliyettir (görev panosunda listelenmiştir).

3.4.5 SG5 Fonksiyonel Olmayan Testleri Yürütme

Bu süreç alanındaki önceki hedefte olduğu gibi, büyük ölçüde bu belgenin önceki bölümlerinde ele alınan Test Tasarımı ve Yürütme süreç alanının ilgili özel hedefi SG 3 Test Yürütme bölümüne atıfta bulunacağız. DevOps ortamında fonksiyonel olmayan testlerin yürütülmesi, test kusurlarının raporlanması ve test günlüklerinin yazılmasına ilişkin uygulamalar, DevOps ortamında fonksiyonel testler için olanlarla temelde aynıdır. Bu işlemler genellikle geleneksel bir projede olduğundan çok daha az dokümantasyon ile gerçekleştirilir. Çoğu zaman ayrıntılı test prosedürleri ve test günlükleri oluşturulmaz. Fonksiyonel olmayan testlerin de dağıtımdan sonra test edildiğini/ölçüldüğünü ve bunun için Servis Seviyesi Göstergeleri (SLI'lar) ve Servis Seviyesi Hedefleri (SLO'lar) tipik bir yönetim yöntemi olduğunu unutmayın.

Fonksiyonel olmayan testlerin yürütülmesi, CI/CD sürecinin ayrılmaz bir parçasıdır ve yineleme planlaması sırasında belirlenen önceliklere uygun olarak gerçekleştirilir. Bazı testler belgelenmiş bir test prosedürü kullanılarak yürütülebilir, ancak genellikle birçok fonksiyonel olmayan test, keşif ve oturum bazlı testler çerçevesini kullanarak yürütülür. İteratif geliştirme ile regresyon testlerini organize etme ve yapılandırma ihtiyacı artar. Bu, otomatik regresyon testleri ve destekleyici araçlar kullanılarak yapılır. Regresyon testleri, elbette, test edilmesi önemli olarak belirlenen sistemin fonksiyonel olmayan yönlerine de uygulanır.

Test sırasında bulunan fonksiyonel olmayan kusurlar ekip tarafından kaydedilebilir ve raporlanabilir. DevOps projelerinde, bulunan tüm kusurların gerçekten kaydedilmesi gerekip gerekmediği konusunda genellikle tartışmalar yaşanır. Bazı ekipler sadece yinelemelerden kaçan kusurları kaydederken, bazıları

bugün düzeltilemeyen kusurları kaydeder, bazıları ise sadece yüksek öncelikli kusurları kaydeder. Bulunan tüm kusurlar kaydedilmiyorsa, hangi kusurların kaydedilmesi ve hangilerinin kaydedilmemesi gerektiğini belirlemek için kriterler mevcut olmalıdır. Bazen kusurlar, bir görevin tamamlanmasını ve görevlerin tamamlanmasını engellediğini görselleştirmek için görev panosuna bir görev üzerine yapıştırıcı olarak veya ayrı bir görev olarak kaydedilir. Bazıları, kusur yönetimi veya kusur izleme aracı gibi bir araç kullanarak bulunan kusurları kaydetmeyi ve belgelemeyi tercih eder. Böyle bir araç, mümkün olduğunca yalın bir şekilde kullanılmalı ve hiçbir veya çok az katma değeri olmayan fonksiyonel olmayan test kusur raporlarının sunulmasını zorlamamalıdır. Fonksiyonel olmayan gereksinimler üretimde ölçülürse, bozulma göstergesi (SLI, SLO'yu yerine getirmiyor) önceki sürüme otomatik olarak geri dönüşü tetikleyerek genel sistem performansını korur.

DevOps ekiplerinde de test edilen öğelerin tanımlanmış kabul kriterlerini karşılayıp karşılamadığını ve gerçekten “tamamlandı” olarak etiketlenip etiketlenemeyeceğini belirlemek için fonksiyonel olmayan testlerin yürütülmesi sırasında verilerin kaydedilmesi iyi bir uygulama olarak kabul edilir. Kaydedilen bilgiler, bir tür durum yönetimi biçiminde toplanmalı ve/veya özetlenmelidir. DevOps ekiplerinde de, test edilen öğelerin tanımlanmış kabul kriterlerini karşılayıp karşılamadığını ve gerçekten “tamamlandı” olarak etiketlenip etiketlenemeyeceğini belirlemek için fonksiyonel olmayan testlerin yürütülmesi sırasında verileri kaydetmek iyi bir uygulama olarak kabul edilir. Kaydedilen bilgiler, bir tür durum yönetimi biçiminde toplanmalı ve/veya özetlenmelidir.

3.5 Süreç Alanı 3.5 Eş Gözden Geçirmeler

Eş Gözden Geçirme sürecinin amacı, iş ürünlerinin belirtilen gereksinimleri karşıladığını doğrulamak ve seçilen iş ürünlerindeki kusurları erken ve verimli bir şekilde gidermektir. Bunun önemli bir yan etkisi, iş ürünleri ve önlenebilecek kusurlar hakkında daha iyi bir anlayış geliştirmektir.

3.5.1 SG1 Eş Gözden Geçirme Yaklaşımı Oluşturma

DevOps ekipleri, genellikle bir ürün hakkında geri bildirimde bulunmak için bir araya geldikleri tek bir zaman dilimi olmadığı için, resmi anlamda eş gözden geçirme yapmazlar. Ancak, geliştirme süreci boyunca sürekli olarak daha az resmi eş gözden geçirme yaparak eş gözden geçirmenin amacına ulaşırlar. Bu, birçok DevOps kuruluşunda yaygın bir uygulamadır. DevOps, akışa daha fazla odaklanarak yazılım geliştirmede yalın uygulamaları hayata geçirdiğinden, herhangi bir inceleme uygulaması bu hızlı gelişimi destekler ve faaliyetlerin akışında darboğaz veya gecikme yaratmaz. Daha küçük iş öğelerinin incelenmesi tipiktir, örneğin birkaç satırlık kodun sık sık incelenmesi ve yapılacak iş öğeleri için kabul kriterlerinin iş birliği içinde tanımlanması. İnceleme uygulamaları o kadar önemlidir ki, örneğin her çekme isteğinde inceleme zorunluluğu getirerek veya Hazır Tanımı'na uyumu zorunlu kılarak, süreçte kaçınılmaz hale getirilirler. Bu TMMi hedefi, bir proje içinde eş gözden geçirme yaklaşımı oluşturmaya yönelik uygulamaları kapsar. Bir inceleme yaklaşımı, inceleme faaliyetlerinin nasıl, nerede ve ne zaman gerçekleştirileceğini ve bu faaliyetlerin resmi mi yoksa gayri resmi mi olacağını tanımlar. Eş gözden geçirme yaklaşımı oluşturmak, DevOps projeleri için de geçerlidir. İnceleme yaklaşımının mümkün olduğunca çok kısmı, araçlarda uygulanarak kaçınılmaz hale getirilir, örneğin iş akışını yöneten görev panosunun aşamaları arasında, Pull Request'lerde veya CI/CD iş hattında kalite kapıları olarak.

DevOps projelerinde tipik olarak gerçekleştirilen eş gözden geçirmeye örnekler:

- yineleme boyunca düzenli olarak ekip ve iş paydaşlarıyla kullanıcı hikayeleri üzerinde yapılacak işlerin iyileştirilmesi oturumları düzenlemek

- geliştirilmekte olan iş ürünlerini (örneğin kod veya testler) tartışmak ve geri bildirim sağlamak için diğer ekip üyeleriyle günlük toplantılar yapmak
- ürünlerin müşterilere erken ve sık sık, en azından yineleme incelemesi sırasında yinelemenin sonunda gösterilmesi.
- Infrastructure-as-Code'un bir parçası olarak kullanıcı hikayeleri, çekme isteği yoluyla kod değişiklikleri, komut dosyaları ve yapılandırma dosyaları gibi iş ürünlerinin incelemeleri.
- yineleme sırasında herhangi bir faaliyette çalışan çift.

Yetersiz spesifikasyonlar genellikle proje başarısızlığının ana nedenidir. Spesifikasyon sorunları, kullanıcıların gerçek ihtiyaçlarını tam olarak anlamamaları, sistem için genel bir vizyonun olmaması, gereksiz veya çelişkili özellikler ve diğer iletişim sorunlarından kaynaklanabilir. DevOps geliştirmede, kullanıcı hikayeleri iş temsilcileri, geliştiriciler ve test uzmanlarının bakış açılarından gereksinimleri yakalamak için yazılır. Sıralı geliştirmede, bir özelliğin bu ortak vizyonu, gereksinimler yazıldıktan sonra resmi incelemelerle gerçekleştirilir; DevOps geliştirmede ise bu ortak vizyon, gereksinimler belirlenirken sık sık yapılan gayri resmi incelemelerle gerçekleştirilir. Bu gayri resmi inceleme oturumları genellikle yapılacak işlerin iyileştirme oturumları olarak adlandırılır. İyileştirme toplantıları sırasında ekip ve iş temsilcisi, ortak bir anlayış oluşturmak, uygulama için gerekli ayrıntı düzeyini bulmak ve açık sorunları netleştirmek için inceleme tekniklerini kullanır.

Gereksinim mühendisliği, gayri resmi incelemeler, adım adım incelemeler, denetimler veya perspektif tabanlı okumalar gibi yöntemlerle gereksinimlerin doğrulanmasını vurgularken, Çevik ve DevOps yöntemleri, değerli ürün artışlarını hızlı bir şekilde uygulamak için sık ve erken geri bildirim yoluyla gereksinimleri doğrulamaya çalışır. Prototipler veya üretime dağıtılan entegre ürün artışları şeklinde hızlı sonuçlar göstererek erken resmi doğrulama ihtiyacı azalır. Artış, paydaşların gereksinimlerini tam olarak karşılamıyorsa, fark yeni gereksinimler şeklinde ürün yapılacak işine geri konur ve diğer tüm yapılacak iş öğeleriyle birlikte önceliklendirilir. Yineleme incelemesi sırasında, bir yineleme sırasında gerçekte neyin oluşturulduğunun gösterilmesi, somut bir konu etrafında doğrulama temelli bir konuşmayı canlandırmanın çok etkili bir yoludur. Bir şeyin nasıl çalıştığını gerçekten görebilmek ve dağıtılan özelliklerin son kullanıcı davranışları üzerindeki etkisini anlamak, konuşmaya odaklanmayı sağlayan en iyi yoldur. Sunum, yineleme incelemesi sırasında gerçekleştirilen bir faaliyettir. Sisteme entegre edilen özellikler paydaşlara sunulur ve onlarla tartışılır, tartışmadan elde edilen yeni bilgiler yansıtılmak üzere ürün biriktirme listesine veya sürüm planına gerekli uyarlamalar yapılır.

Eş gözden geçirme kriterleri, otomatik bir şekilde veya iş akışı içinde bir kontrol listesi olarak tanımlanabilir. Kod incelemelerinde, zorunlu birim testi yürütme, kod karmaşıklığı ölçümleri, kod stil kılavuzuna uygunluk gibi giriş kriterleri tanımlanır. Bu, yalnızca incelenecek kodun giriş kriterlerine uygun olmasını sağlar. Çıkış kriteri olarak, kodun genellikle ekibin birden fazla üyesinin incelemesinden geçmesi gerekir. Aksi takdirde, kod güncellendikten sonra inceleme tekrarlanmalıdır. Bu genellikle kodun birleştirilemeyeceği anlamına gelir.

Yapılacak iş kalemlerinde, giriş kriterleri genellikle yalnızca resmi incelemelerde geçerlidir ve bu nedenle DevOps projeleri içindeki iyileştirme oturumları için uygun olmayabilir. Ancak, çıkış kriterlerinin kullanımı uygulanabilir ve katma değer sağlar. INVEST [Wake], DevOps projelerinde kullanıcı hikayelerini incelerken ve güncellerken yaygın olarak başvurulan bir dizi çıkış kriterine örnektir. INVEST, iyi bir kullanıcı hikayesini oluşturan aşağıdaki gereksinimleri kapsayan bir kısaltmadır:

I	independent	bağımsız
N	negotiable	tartışılabilir
V	valuable	değerli
E	estimable	saygı değer
S	small	küçük
T	testable	test edilebilir

Şekil 6: INVEST kullanıcı hikayesi kriterleri

- Bağımsız
- Tartışılabilir
- Değerli
- Saygıdeğer
- Küçük
- Test edilebilir

Elbette, kaliteyle ilgili başka çıkış kriterleri de mevcuttur ve bunlar da faydalı bir şekilde kullanılabilir.

İnceleme çıkış kriterleri, farklı iş ürünleri için Tamamlanma Tanımı'nda tanımlanmıştır. Kriterleri karşılamayan öğeler geri gönderilir ve bir sonraki aşamaya geçemez.

3.5.2 SG2 Eş Gözden Geçirmeleri Gerçekleştirme

Elbette, her yaklaşım gibi, bu da sadece tanımlanmış ve üzerinde anlaşmaya varılmış bir yaklaşım olarak var olmamalı, aynı zamanda uygulanmalıdır. Ekip, bir yineleme sırasında yapılacak iş iyileştirme oturumlarına önemli miktarda zaman ayırmalı, günlük rutinlerinin bir parçası olarak gayri resmi (ve muhtemelen resmi) incelemeler uygulamalı ve düzenli olarak paydaşlara demolar sunmalıdır. Paydaşlara/müşterilere en azından her yinelemenin sonunda demolar sunulması beklenir.

DevOps ekibinin bir parçası olan test uzmanı, inceleme oturumlarına katılmalıdır. Bu, özellikle yineleme boyunca testlerin temelini oluşturan iş ürünlerinin tartışıldığı ve incelendiği oturumlar için geçerlidir (örneğin, yapılacak işlerin iyileştirilmesi oturumları ve iş birliğine dayalı tasarım). Genellikle, test uzmanının benzersiz bakış açısı, eksik ayrıntıları veya fonksiyonel olmayan gereksinimleri belirleyerek kullanıcı hikayesini iyileştirir. Test uzmanı, özellikle belirli bir kullanıcı hikayesi için kabul kriterlerinin belirlenmesi ve tanımlanmasına destek olabilir. BDD ve ATTD gibi teknikler, kabul testleri formatında kabul kriterlerini tanımlamak için kullanılabilir ve böylece kullanıcı hikayesinin test edilebilirlik kriterini karşılayıp karşılamadığını belirleyebilir. Test uzmanı, iş temsilcilerine kullanıcı hikayesi hakkında açık uçlu “ya eğer?” soruları sorarak, kullanıcı hikayesini test etmek için yöntemler önererek ve kabul kriterlerini onaylayarak katkıda bulunur.

Özel uygulama 2.3 eş gözden geçirme verilerini analiz etmek, özellikle resmi incelemelerle ilgilidir. Resmi incelemelerde çok fazla efor harcanır. Bu eforun hem verimli hem de etkili bir şekilde harcanmasını sağlamak için, eş gözden geçirme verileri toplanır ve inceleme sürecini öğrenmek ve ayarlamak için ekibe iletilir. Belirtildiği gibi, resmi incelemeler DevOps projelerinde daha az yaygındır.

Veri toplama, resmi incelemelerin önemli bir parçası olsa da, gayri resmi incelemelerde çok daha az yaygındır. Eş gözden geçirme yaparken veri toplamaya karar verirken, aşağıdaki soruları sorun:

- Bu veriler toplanıyorsa, kimler kullanacak?
- Bu veriler iş hedeflerimizle nasıl bir ilişki içindedir?

Verileri kimse anlamlı bir şekilde kullanmayacaksa, değerli kaynakları bu verileri toplamak için boşa harcamayın. Ancak, DevOps'ta genellikle araçlar, inceleme sonuçlarını ve verileri toplamak ve bunları bir gösterge tablosunda bilgi olarak sunmak için kullanılır, bu da çok daha verimli hale getirir. Ayrıca, geriye dönük toplantılar sırasında, ekip inceleme verilerini ve bulgularını analiz ederek temel nedenleri anlamaya ve iyileştirme fırsatlarını belirlemeye çalışıldığında, bu veriler çok anlamlı hale gelir.

DevOps'a Özgü Terimler Sözlüğü

Mavi-yeşil dağıtım	Blue-green deployment	Aktif ve pasif olmak üzere iki özdeş üretim ortamının, yazılımın yeni sürümünü pasif ortamda test etmek için muhafaza edildiği ve doğrulandıktan sonra tüm trafiğin aktif ortamdaki aktarıldığı bir dağıtım stratejisi.
Canary testi	Canary testing	Değişikliklerin, ek aşamalı yaygınlaştırma öncesinde doğrulama yapılmasına olanak tanıyan küçük bir kullanıcı grubuna sunulduğu bir test yaklaşımı.
Değişiklik başarısızlık yüzdesi	Change fail percentage	Uygulanan değişikliklerin arızalara neden olan oranı.
Değişim süresi	Change lead time	Bir taahhüdün üretime ulaşması için gereken süre.
CI/CD iş hattı	CI/CD pipeline	Yeni bir yazılım sürümünü sunmak için gerekli adımlar dizisi.
Kapsayıcılık	Containerization	Bir uygulamayı ve bağımlılıklarını kapsayıcı adı verilen tek bir yürütülebilir birim halinde paketleyen hafif bir sanallaştırma biçimi.
Sürekli teslimat (CD)	Continuous delivery (CD)	Kod değişikliklerinin otomatik olarak oluşturulduğu, test edildiği ve üretime uygulanabilir hale getirildiği otomatik bir yazılım geliştirme prosedürü.
Sürekli dağıtım (CD)	Continuous deployment (CD)	Tüm testleri geçen kod değişikliklerinin manuel müdahale olmaksızın otomatik olarak üretime dağıtıldığı otomatik bir yazılım sürüm prosedürü.
Sürekli izleme	Continuous monitoring	Kullanıcı ve sistem davranışını gerçek zamanlı olarak anlamak için performans göstergelerinin otomatik olarak toplanması.
Çapraz fonksiyonel DevOps ekibi	Cross-functional DevOps team	Farklı ancak birbiriyle örtüşen bilgi, beceri ve yeteneklere sahip bir grup insanın ortak bir hedef doğrultusunda birlikte çalışması.
Gizli Dağıtım	Dark launch	Yeni bir özellik veya ürünün kamuya duyurulmadan piyasaya sürüldüğü ve ekiplerin üretimdeki performansı test edip izleyebildiği bir dağıtım stratejisi.
Dağıtım sıklığı	Deployment frequency	Üretime sunulan yazılım sürümlerinin oranı.
DevOps	DevOps	Bir çözümü etkin bir şekilde geliştirmek ve işletmek için gereken entegrasyon, otomasyon ve iş birliğini destekleyen bir zihniyet, kültür ve teknik uygulamalar kümesi.

Başarısız dağıtım kurtarma süresi	Failed deployment recovery time	Başarısız bir dağıtımdan sonra normal çalışmayı geri yüklemek için gereken süre.
Özellik kontrolü	Feature toggle	Kodu değiştirmeden sistem davranışını değiştiren bir mekanizma.
Geri bildirim	Feedback	Ürünleri, servisleri ve süreçleri iyileştirmek için sürekli bilgi alışverişi.
Akış	Flow	İlgili değer akışında iş ürünlerinin adım adım sorunsuz, doğrusal ve hızlı hareketi.
Altyapı olarak Kod (IaC)	Infrastructure as Code (IaC)	Makine tarafından okunabilir yapılandırma dosyalarını kullanarak BT altyapısını yönetme ve sağlama ve buna yazılım geliştirme uygulamalarını uygulama pratiği.
Gözlemlenebilirlik	Observability	Bir bileşenin veya sistemin durumunu, izleme, sorun giderme ve optimizasyon sağlamak için dış çıktısını analiz ederek anlama yeteneği.
Çekme isteği (PR)	Pull Request (PR)	Kod tabanında yapılan değişiklikleri ekip üyelerine bildirmeye, bu değişikliklerin incelenmesini, tartışılmasını ve ana kod deposuna birleştirilmesini önermeye olanak tanıyan bir mekanizma.
Servis Seviyesi Göstergesi (SLI)	Service Level Indicator (SLI)	Servis güvenilirliğinin ölçülebilir bir göstergesi.
Servis Seviyesi Hedefi (SLO)	Service Level Objective (SLO)	Servis seviyesi göstergesi için bir güvenilirlik hedefi.
Site Güvenilirliği Mühendisliği (SRE)	Site Reliability Engineering (SRE)	Otomatik geri bildirim döngüleri kullanarak servis güvenilirliği ile yeni özellik geliştirme hızı arasında denge kurarken, yazılım geliştirme uygulamalarını ve ilkelerini altyapı ve operasyonlara uygulayan bir yaklaşım.
Değer akışı	Value stream	Bir müşteriye ürün veya servis sunmak için gereken, bilgi ve malzeme akışı da dahil olmak üzere uçtan uca faaliyet dizisi.
Sanallaştırma	Virtualization	Diğer yazılımların çalıştığı yazılım ve/veya donanımın simülasyonu.

Kaynakça

[Baah] A. Baah (2017), Agile quality assurance, Bookbaby

[Beyer] B. Beyer, C. Jones, J. Petoff and N.R. Murphy (2016). Site Reliability Engineering: How Google Runs Production Systems, 1st. O'Reilly Media, Inc.

[Cohn] M. Cohn (2009), Succeeding with Agile: Software Development using Scrum, Addison-Wesley

[DORA16] A. Brown, N. Forsgren, J. Humble, N. Kersten and G. Kim (2016), 2016 State of DevOps Report, Puppet Labs, DORA (DevOps Research & Assessment]

[DORA24] DORA [2024], DORA's software delivery metrics: the four keys, <https://dora.dev/guides/dora-metrics-four-keys/>, DORA (DevOps Research & Assessment]

[Galen] R. Galen (2015), Three Pillars of Agile Quality and Testing: Achieving Balanced Results in your Journey Towards Agile Quality, RGCG, LLC

[ISO25010] ISO/IEC 25010 (2023), Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – Product quality model, International Organization for Standardization

[McMahon] Paul. E. McMahon (2011), Integrating CMMI with Agile Development – Case Studies and Proven Techniques for Faster Performance Improvement, Addison Wesley

[Veenendaal12] E. van Veenendaal (2012), PRISMA: Product Risk Assessment for Agile Projects, in: Testing Experience, Issue 04/12, December 2012

[Veenendaal24] Erik van Veenendaal, Rex Black, and Dorothy Graham (2024), Foundations of Software Testing - ISTQB Certification (5th edition), Cengage

[Wake] B. Wake (2003), INVEST in Good Stories, and SMART Tasks, xp123.com/articles/invest-in-good-stories-and-smart-tasks