Süleyman Demirel Üniversitesi
Fen Bilimleri Enstitüsü Dergisi
Cilt 29, Sayı 1, 189-200, 2025

Süleyman Demirel University
Journal of Natural and Applied Sciences
Volume 29, Issue 1, 189-200, 2025

DOI: 10.19113/sdufenbed.1624000

Scrum Metodolojisi Kullanılarak Yönetilen Projelerde Fonksiyonel ve Otomasyon Testlerinin Türkiye'de Yazılım Geliştiren Firmalardaki İşleyiş Analizi

Hayriye KATRANCI[1], Serdar BİROĞUL [*2]

[1] Yıldız Tech, İstanbul, Türkiye

[2] Düzce Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği, Düzce, Türkiye
Nahçıvan Devlet Üniversitesi, Mimarlık ve Mühendislik Fakültesi, Elektronik ve Bilgi Teknolojileri Bölümü, Nahçıvan, Azerbaycan

(Alınış / Received: 21.01.2025, Kabul / Accepted: 22.03.2025, Online Yayınlanma / Published Online: 25.04.2025)

Anahtar Kelimeler
Agile Yazılım Geliştirme, Scrum Metodolojisi, Fonksiyonel Testler, Otomasyon Testleri

Öz: Bu çalışmada, Türkiye'deki yazılım geliştirme firmalarında Scrum metodolojisinin proje yönetiminde nasıl uygulandığı ve bu bağlamda fonksiyonel ve otomasyon test süreçlerinin işleyişi analiz edilmiştir. Araştırma, Scrum metodolojisinin çalışanlar tarafından benimsenme düzeyine ve test süreçlerine entegrasyonuna odaklanmaktadır. Özellikle fonksiyonel ve otomasyon test süreçlerinin Scrum çerçevesinde nasıl optimize edildiği ve bu süreçlerin proje verimliliği ile başarısına etkileri incelenmiştir. Çalışmada, katılımcıların demografik profilleri, mesleki ve akademik geçmişleri, Scrum metodolojisine aşinalıkları ve test süreçlerindeki deneyimleri dikkate alınarak kapsamlı bir değerlendirme yapılmıştır. Elde edilen bulgular, araştırmaya dahil olan firmaların büyük çoğunluğunun Scrum metodolojisini benimsediğini ve test süreçlerinin ağırlıklı olarak otomasyon araçlarıyla desteklendiğini göstermektedir. Sonuçlar, Scrum metodolojisinin test süreçlerinin entegrasyonu ve yönetimi açısından etkili bir çerçeve sunduğunu ortaya koymaktadır.

Functional and Automation Testing in Projects Managed Using Scrum Methodology in Software Development Companies in Turkey

Keywords
Agile Software Development, Scrum Methodology, Functional Tests, Automation Tests

Abstract: This study analyzes the implementation of the Scrum methodology in project management within software development firms in Turkey, with a particular focus on the functioning of functional and automation testing processes. The research examines the extent to which Scrum methodology is adopted by employees and how testing processes are integrated into this framework. Specifically, it investigates how functional and automation testing processes are optimized within Scrum-based workflow management and assesses their impact on project efficiency and success. The study considers participants' demographic profiles, professional and academic backgrounds, familiarity with Scrum methodology, and experience in testing processes to provide a comprehensive evaluation. The findings indicate that the majority of the firms included in the study have embraced the Scrum methodology and that their testing processes are predominantly supported by automation tools. The results demonstrate that Scrum provides an effective framework for the integration and management of testing processes.

1. Giriş

Değişen teknolojiyle birlikte müşteri ihtiyaçlarının artması, değişmesi, çözülmesi ve yönetilmesi gereken problemlerin gittikçe daha da karmaşık ve zor hale gelmiştir. Bundan dolayı yazılım geliştirme süreçlerinde ve proje yönetiminde artık klasik metodolojiler yeterli kalmamaktadır. Çevik (agile)

*Sorumlu yazar: serdarbirogul@duzce.edu.tr, serdarbirogul@ndu.edu.az

yöntemler ile bu belirtilen handikap noktaları ortadan kaldırılmaya çalışılmaktadır. Scrum, kanban, extreme programming gibi çevik yaklaşımlar projelerde müşteri memnuniyetini, ürün kalitesini arttırmak, üretkenliği artırmak ve değişen gereksinimlere hızlı bir şekilde yanıt vermek için yaygın olarak kullanılmaktadır. Bu durum yazılım test süreçlerine de yansımış ve yazılım kalitesini güvence altına almak için de fonksiyonel ve otomasyon testlerine olan talep artmıştır. Bu nedenle fonksiyonel ve otomasyon testlerinin scrum metodolojisi ile yazılım geliştirilen projelerdeki önemi daha da artmıştır. Bu çalışmada Türkiye'deki 52 tane yazılım geliştiren firmaların scrum metodolojisi uygulanarak geliştirilen projelerdeki fonksiyonel ve otomasyon test süreçlerinin işleyişi analiz edilmiştir. Bu süreçlerin projelerin başarı düzeyine olan etkileri de gözlemlenmiştir. Çalışmanın kapsamı içerisinde fonksiyonel ve otomasyon testlerinin scrum süreçlere entegrasyonu ve yazılımda kalite üzerine etkilerinin araştırma verileri saha araştırması neticesinde analiz edilmiştir. Bu çalışma ile Türkiye'deki çevik metodolojilerin uygulanış biçimlerinin küresel normlar ve sektör standartlarıyla karşılaştırılması için de önemli bir fırsat yaratması öngörülmüştür. Elde edilen bulgular, yazılım geliştirme ve test süreçlerinde yaygın olarak kullanılan yöntemlerin zorluklarını, avantajlarını ve iyileştirme alanlarını aydınlatmaya yardımcı olması hedeflenmiştir. Bu çalışma ile birlikte çevik metodolojilerin uygulandığı projelerdeki test süreçlerinin kalitesinin artırılması, sektör paydaşlarının test süreçlerine yönelik bilinçlendirilmesi ve firmaların test stratejilerini optimize etmelerine katkı sağlanması hedeflenmiştir.

Sektör standartlarını belirleme açısından bu çalışmada Türkiye'deki yazılım geliştirme sektöründe çevik metodolojilerin uygulanma şekli ve etkinliği ortaya konmaya çalışılmıştır. Bu sayede firmalar, küresel trendlerle uyumlu olarak kendi süreçlerini gözden geçirme fırsatı bulabilir. Verimlilik ve etkinlik açısından test süreçlerinin daha verimli ve etkili bir şekilde tasarlanmasını sağlanmalıdır. Böylelikle çeşitli çevik metodolojileri ve test stratejileri arasındaki ilişkiler incelenerek uygun metodolojilerin ve tekniklerin başarılı olarak uygulanması sağlanabilir. Eğitim ve gelişim açısından bakıldığında eğitim programlarının ve profesyonel gelişim faaliyetlerinin daha etkili bir şekilde planlanmasına olanak tanıyarak yazılım testi alanında çalışan veya bu alana girmek isteyen profesyoneller için önemli tespitler çıkarılmıştır. Bu çalışma ile karar verme ve politika geliştirme noktasında firmaların kendi iç süreçlerini iyileştirmek, daha bilinçli kararlar almak ve risk yönetimini optimize etmek için kullanılabilir bilgiler oluşturulmuştur. Bu çalışmada, yazılım geliştirme sürecindeki test sürecinin önemi, kalite güvence sisteminin önemi, scrum metodoloji kullanan firmaların bu belirtilenleri ne kadar uyguladıkları ve süreçlerine adapte edebildiklerinin analizi saha verileri üzerinden ortaya konmuştur.

2. Yazılım Geliştirmede Scrum Metodolojisi

Çevik yöntemler, yazılım geliştirme süreçlerinde esneklik ve hızlı adaptasyonu önceliklendiren bir yaklaşım sunar. Bu metodolojiler, 2001 yılında yayımlanan Çevik Manifesto ile popülerlik kazanmış ve geliştirme sürecini kısa iterasyonlar halinde, sürekli geliştirme ve düzenleme fırsatı sağlayarak yeniden şekillendirmiştir. Çevik yaklaşımlar, projenin başlangıcında tüm gereksinimlerin tam olarak tanımlanmasını gerekli kılmaz. Bunun yerine, gereksinimler proje boyunca evrilebilir ve geliştirilebilir. Bu, teknolojik yeniliklere ve pazar koşullarındaki değişimlere hızlı bir şekilde yanıt verilmesini mümkün kılmaktadır. Çevik metodolojiler, planlamayı ve teslimatları daha yönetilebilir ve esnek hale getirerek, geliştirme sürecinin genel verimliliğini artırmaktadır. Çevik yöntemler, müşteri geri bildirimlerini sürecin merkezine koymakta ve geliştirme sürecinin her aşamasında aktif bir rol oynamalarını, düzenli olarak sağladıkları geri bildirimlerle ürünün şekillendirilmesine doğrudan katkıda bulunmalarını sağlamaktadır. Bu sürekli etkileşim, son ürünün müşteri beklentileri ve ihtiyaçları ile daha uyumlu olmasını sağlamaktadır. Sürekli değişen gereksinimlere hızlı bir şekilde adapte olmayı sağlayan bu yaklaşım, yazılım geliştirme ekiplerine, projeleri üzerinde daha iyi kontrol sahibi olma imkânı tanır. Müşteri memnuniyetini de maksimize etmeyi hedefler.

Çevik yöntemlerin uygulanması, projelerde esneklik, hız ve verimlilik gibi kritik faktörleri ön plana çıkararak, geliştirme süreçlerinin daha dinamik ve yanıt veren bir yapıya kavuşmasını sağlar. Bu, yazılım endüstrisinde çevik metodolojilerin giderek daha fazla benimsenmesine yol açmaktadır. Bunlar arasında Scrum, Kanban ve Extreme Programming (XP) gibi yöntemler bulunmaktadır [1]. Scrum metodolojisi, çevik metodolojiler içinde özellikle kompleks yazılım projeleri için tercih edilen öncü bir yaklaşımdır. Scrum metodolojisinin temelini, kısa süreç döngüleri olan sprintler oluşturur. Her bir sprint (hızlı, sürat) toplantısı genellikle birkaç hafta süren yoğun çalışma periyodlarıdır ve belirlenen hedeflere ulaşmak için yoğunlaşmış bir çaba gerektirir. Trihardianingsih ve arkadaşları scrum metodolojisinin, özellikle karmaşık yazılım projelerinde süreçlerin etkili bir şekilde yönetilmesini sağladığını ve projenin başarılı bir şekilde tamamlanmasına katkıda bulunduğunu ortaya koymuştur [2]. Bu nedenle, scrum yaklaşımı modern yazılım geliştirme ortamında çevik ve etkin çözümler arayan organizasyonlar için vazgeçilmez bir araç haline gelmiştir.

3. Yazılımda Kalite ve Test

Yazılım kalitesi ve test, yazılım geliştirme sürecinin temel unsurlarıdır. Ürün kalitesini sağlamak, hataları azaltmak ve müşteri memnuniyetini artırmak için

kritik öneme sahiptir. Yazılım kalitesi, yazılımın belirli gereksinimleri ve beklentileri ne derece karşıladığını ölçen bir kriterdir. Bu durum yazılımın genel başarısını doğrudan etkileyen bir faktördür. Kalite, yazılımın işlevselliği, güvenilirliği, kullanılabilirliği, performansı ve bakım kolaylığı gibi bir dizi özniteliğe dayanarak değerlendirilir. Bu öznitelikler, yazılımın hedef kullanıcı kitlesi tarafından ne kadar iyi kabul göreceğini ve kullanılacağını belirler. Zhao ve arkadaşlarının yaptığı çalışmada, yazılım kalitesinin pazar başarısı ve kullanıcı memnuniyeti üzerindeki etkileri detaylı bir şekilde incelenmiştir. Araştırma, yüksek kaliteli yazılımların pazarda daha iyi performans gösterdiğini ve kullanıcılardan yüksek memnuniyet oranları aldığını ortaya koymuştur. Yazılımın kalitesi arttıkça, arızaların ve kullanıcı hatalarının azaldığı, dolayısıyla kullanıcıların yazılımdan aldıkları genel memnuniyetin ve ürün sadakatinin arttığı gözlemlenmiştir [3]. Bu durum, yazılım geliştiriciler için kaliteyi sürekli olarak ön planda tutmanın ve iyileştirmeler yapmanın önemini vurgulamaktadır. Kalite, sadece yazılımın mevcut durumunu değil aynı zamanda marka imajını ve müşteri sadakatini de güçlendirir. Malik ve Singh tarafından yapılan araştırmada, kalite ve kalite güvence süreçleri, yazılımın geliştirilmesi sırasında ortaya çıkan riskleri yönetmeye, maliyetleri optimize etmeye ve ürün kalitesini maksimize etmeye odaklanır [4]. Bu süreçler, yazılımın hata oranını azaltmak ve nihai ürünün belirlenen kalite standartlarını karşılamasını sağlamak için tasarlanmıştır. Kalite güvence faaliyetleri hata önleme, hata tespiti ve sürekli iyileştirme faaliyetlerini içerir. Böylece yazılım projelerinin başarısını artırma ve müşteri memnuniyetini sağlama konusunda temel bir rol oynar. Kalite güvencesi, aynı zamanda risk yönetimi ve değer mühendisliği gibi önemli kavramları da içine alır. Kalite güvencesi sürecinin başarısı, sürekli iyileştirme anlayışına bağlıdır.

Yazılım testi, yazılım geliştirme ve kalite güvence sürecinin ayrılmaz bir parçasıdır. Ürünün hata oranını tespit etme, giderme ve yazılımın kalite gereksinimlerini karşıladığının doğrulama işlevini görür. Bu süreç, yazılımın piyasaya sürülmeden önce belirlenen kalite standartlarına uygunluğunu sağlamak için kritik öneme sahiptir. Mateen ve arkadaşlarının yaptığı araştırmada, etkili bir test stratejisinin, yazılımın güvenilirliğini artırdığını ve geliştirme maliyetlerini azalttığını belirtmiştir [5]. Yazılım testi, genellikle yazılımın her sürümünün kullanıcı beklentilerini ve işlevsel gereksinimleri karşılayıp karşılamadığını değerlendirmek için kullanılır. Bu, hem son kullanıcı memnuniyetini hem de yazılımın pazardaki başarısını doğrudan etkileyen bir faktördür. Yazılım test süreci, genellikle manuel ve otomatik olmak üzere iki temel yaklaşımla gerçekleştirilir. Manuel testler, test uzmanları tarafından el ile yapılan ve yazılımın kullanıcı perspektifinden doğrulamasını içeren süreçlerdir. Bu tür testler, özellikle kullanıcı arayüzü ve kullanıcı deneyimi gibi insan faktörlerinin önemli olduğu alanlarda etkilidir. Otomatik testlerde ise yazılımın belirli bölümlerinin tekrarlanabilir ve düzenli olarak test edilmesi için özel olarak geliştirilmiş araçlar ve scriptler kullanılır. Otomatik testler özellikle büyük ölçekli projelerde, tekrar eden görevleri hızlı ve hatasız bir şekilde gerçekleştirme kapasitesi nedeniyle tercih edilir.

Etkili bir yazılım test stratejisi, hem manuel hem de otomatik test tekniklerinin dengeli bir şekilde kullanılmasını gerektirir. Yazılım testi, yazılımın kalitesini doğrulamanın ve güvenilir bir ürün sunmanın temel yollarından biridir. Etkili bir test stratejisi, yazılımın başarılı bir şekilde piyasaya sürülmesini sağlamak için gereklidir. Bu strateji, hem manuel hem de otomatik test yaklaşımlarını kapsamlı bir şekilde entegre etmelidir. Garg tarafından yapılan araştırmada test stratejilerinin yazılımın farklı gelişim aşamalarında nasıl uygulanması gerektiğini detaylandırılmıştır [6]. Bu çerçeveler, test süreçlerinin sistematik ve kapsamlı bir şekilde yürütülmesini sağlar. Böylece yazılımın baştan sona kalite standartlarına uygunluğu garanti altına alınır. Yazılım test stratejilerinin uygulanması, projenin gereksinimlerine ve karmaşıklığına göre değişiklik gösterir. Büyük ölçekli projeler genellikle daha kapsamlı ve katmanlı test stratejileri gerektirirken, küçük ölçekli projeler daha az yoğun test süreçleri ile idare edilebilmektedir. Test stratejisinin belirlenmesinde, projenin risk profili, teknolojik karmaşıklığı ve son kullanıcı beklentileri gibi faktörler dikkate alınmaktadır.

3.1. Otomasyon testlerinin çevik projelere entegrasyonu

Fonksiyonel testler, yazılımın belirlenmiş işlevleri yerine getirip getirmediğini değerlendirmek için kullanılan kritik bir test türüdür. Fonksiyonel testler genellikle kullanıcı senaryolarını temel alarak yazılımın farklı işlevlerini simüle eder ve bu işlevlerin beklenen sonuçları üretip üretmediğini kontrol eder. Bu testlerin amacı, yazılımın kullanıcı ihtiyaçlarını ve beklentilerini karşılayıp karşılamadığını doğrulamaktır. Otomasyon testleri fonksiyonel testlerin daha hızlı ve tutarlı bir şekilde gerçekleştirilmesini sağlayan önemli bir araçtır. Bu, test süreçlerinin hızlandırılmasına ve yazılımın işlevselliğinin daha kapsamlı bir şekilde değerlendirilmesine olanak tanır. Ayrıca otomasyon testleri, manuel hataların azaltılmasına ve test süreçlerinin daha verimli hale getirilmesine katkı sağlar. Otomasyon testlerinin kullanılmasıyla birlikte, yazılım geliştirme sürecindeki test aşamaları daha etkin bir şekilde yönetilebilir ve yazılımın kalitesi artırılabilir [7]. Otomasyon testleri, yazılımın farklı modüllerini kapsamlı ve etkin bir şekilde analiz eden test senaryolarının ve araçlarının uygulandığı bir süreç olarak öne çıkmaktadır. Manuel testlere göre çok daha yüksek doğrulukta ve sürekli bir izleme

sağlandığından, yazılım projelerinin toplamda daha güvenilir bir şekilde yürütülmesine olanak tanınmaktadır [8]. Bu test süreçleri, yazılımın piyasaya sürülme süresini önemli ölçüde kısaltmaktadır. Otomasyon testleri sayesinde test döngüleri tekrarlayan ve sistematik bir şekilde gerçekleştirilmekte, yazılımın farklı bileşenleri arasındaki uyumsuzluklar hızla ortaya çıkarılmaktadır. Bu durum, geliştiricilerin ve test uzmanlarının test sürelerini optimize etmelerine ve mevcut kaynakları daha etkili kullanmalarına olanak sağlamaktadır. Otomasyon testleri ayrıca test maliyetlerini azaltmaktadır. Bu test stratejisi, büyük ve karmaşık yazılım projelerinde kritik bir rol oynamaktadır. Otomasyon test araçları, test senaryolarını otomatik olarak çalıştırarak ve test sonuçlarını detaylı bir şekilde kaydederek yazılım test süreçlerinin standardizasyonuna ve verimliliğine katkıda bulunmaktadır. Bu araçlar, geliştiricilerin ve test mühendislerinin hata tespiti süreçlerini daha sistematik ve tekrarlanabilir hale getirmelerini sağlamaktadır. Böylece, yazılımın piyasaya sürülme sürecinde karşılaşılabilecek sorunlar daha aşamalı bir şekilde giderilmekte ve yazılımın kalitesi artırılmaktadır. Kumar ve Rao yaptıkları çalışmada otomasyon test araçlarının kullanımı, test süreçlerinin daha etkin yönetilmesine ve maliyetlerin optimize edilmesine imkan verdiğini belirtmiştir [9].

Çevik projelerde otomasyon testlerinin kullanımı, sürekli entegrasyon ve sürekli teslimat süreçlerinin ayrılmaz bir parçası olarak kabul edilmektedir. Bu testler, yazılım geliştirme sürecinin her aşamasında sürekli kalite kontrolünü sağlar. Böylelikle hataların erken aşamada tespit edilmesine ve düzeltilmesine imkan tanır. Otomasyonun sağladığı bu süreklilik, çevik metodolojilerin temel prensiplerinden olan hızlı ve esnek geliştirme döngülerine mükemmel bir şekilde uyum sağlamaktadır. Test otomasyonu, çevik ekiplerin daha hızlı iterasyonlar yapmasına ve projelerdeki değişikliklere çabuk adapte olabilmelerine olanak tanımaktadır. Bu sayede, ekipler yazılımın farklı sürümleri arasında sürekli olarak geçiş yapabilmekte ve her iterasyonda müşteriden alınan geri bildirimleri doğrudan sürece entegre edebilmektedir. Rajasekhar, hızlı adaptasyon ve iterasyon yeteneği, çevik projelerde başarının kritik faktörlerinden biri olduğunu ve otomasyon testlerinin, çevik yazılım geliştirme süreçlerinde hızlı geri bildirim alınmasını sağlayarak projelerin daha etkin bir şekilde yönetilmesine katkıda bulunduğunu belirtmiştir [10].

Otomasyon testlerinin uygulanması bazı önemli zorlukları da beraberinde getirmektedir. Bu zorluklardan ilki, yüksek başlangıç maliyetleridir. Otomasyon testlerinin kurulumu ve ilk aşamalarında gerekli altyapının ve araçların sağlanması ciddi bir yatırım gerektirmektedir. Bu durum, özellikle küçük ölçekli işletmeler ve kısıtlı bütçeye sahip projeler için zorlayıcı bir faktör olabilmektedir. Bu yüksek maliyetler, otomasyon testlerinin yaygınlaşmasını sınırlamakta ve bazı projelerde manuel test yöntemlerinin tercih edilmesine neden olmaktadır. İkinci bir zorluk ise test senaryolarının bakımının zor olmasıdır. Otomasyon test senaryoları yazılımın gelişimine bağlı olarak sürekli güncellenmeli ve revize edilmelidir. Bu süreç zaman alıcı ve karmaşık olabilmektedir. Özellikle yazılımın sürekli olarak değiştiği durumlarda senaryoların güncel ve doğru kalmasını sağlamak zorlaşmaktadır. Bu durum, test süreçlerinin etkinliğini doğrudan etkileyebilmekte ve ekstra kaynak gereksinimine yol açmaktadır. Üçüncü olarak otomasyon testlerinin etkinliği büyük ölçüde test senaryolarının kalitesine bağlıdır. İyi tasarlanmış ve kapsamlı test senaryoları olmadan, otomasyon araçları beklenen faydayı sağlayamamaktadır. Düşük kaliteli veya eksik senaryolar, yanıltıcı sonuçlar üretebilmekte ve test süreçlerinde düşük geri dönüş oranlarına sebep olabilmektedir. Bu nedenle etkili bir otomasyon stratejisi geliştirmek için test senaryolarının sürekli olarak gözden geçirilmesi ve iyileştirilmesi gerekmektedir [11].

4. Materyal ve Yöntem

Çalışmada kullanılan ana materyal, yazılım firmalarında çeşitli pozisyonlarda çalışan profesyonellerle yarı yapılandırılmış görüşme formlarıdır. Form, katılımcıların demografik bilgilerini, profesyonel deneyimlerini ve çevik metodolojileri ile ilgili uygulamalarını derinlemesine sorgulamaktadır. Türkiye'deki yazılım geliştiren firmalardaki çevik metodoloji uygulanan projelerdeki test süreçleri ve bu süreçlerin işleyişini inceleyen bu araştırmada karma bir metodoloji kullanılmıştır. Araştırma hem niceliksel hem de niteliksel veri toplama tekniklerini içermektedir. Araştırma formu, kişilerin fonksiyonel ve otomasyon test süreçlerine ilişkin deneyimlerini, süreçlerde karşılaştıkları zorlukları ve çözüm yollarını anlamak için detaylı sorular içermektedir. Bu araştırmanın evreni olarak Türkiye'de yazılım geliştiren 52 firma ve 116 kişiden alınan bilgilerden oluşmaktadır. Evrenin özellikleri, Türkiye'nin yazılım ve teknoloji sektöründeki çeşitliliği ile tanımlanır. Veri toplama yöntemi olarak, yazılım personellerine yapılandırılmış anketler uygulanmıştır. Veri toplama süreci, katılımcıların scrum metodolojisini kullanan çevik projelerdeki rol ve deneyimlerine dair kapsamlı bilgiler sağlaması amacıyla detaylı olarak planlanmıştır.

5. Bulgular ve Tartışma

Tablo 1.'de bu çalışmaya katılım göstermiş olan kişilerin demografik ve sektörel çalışma durumları gösterilmiştir.

Tablo 1. Demografik Bilgilere Yönelik Bulgular

		%
Cinsiyet	Kadın	35,3
	Erkek	64,7

Yaş aralığı	30 yaş ve altı	33,6
	31-35 yaş	44,0
	36 yaş ve üstü	22,4
Eğitim durumu	Lisans	82,8
	Ön lisans	1,7
	Yüksek lisans	15,5
Firmadaki rol	Proje Yöneticisi	6,0
	İş Analisti	26,7
	Teknik Analist	0,0
	Yazılım Geliştirici	24,1
	Teknik Lider	10,3
	Test Kalite Kontrol	24,1
	Ürün sahibi	5,2
	Diğer	3,4
İş tecrübesi	1 yıla yakın	0,9
	1-2 yıl	7,8
	2-3 yıl	8,6
	3-4 yıl	12,9
	4-5 yıl	4,3
	5 yıl ve üzeri	65,5
Manuel yazılım test tecrübe süresi	6 ay kadar	10,5
	1 sene kadar	10,5
	2 seneden fazla	25,4
	Hiç test tecrübesi olmamak	53,5
Otomasyon yazılım test tecrübe süresi	6 ay kadar	2,7
	1 sene kadar	15,5
	2 seneden fazla	7,3
	Hiç test tecrübesi olmamak	74,5
Çalışılan sektör türü	Banka	33,3
	Telekomünikasyon	18,1
	E-Ticaret	21,9
	Sağlık	2,9
	Diğer	23,8

Tablo 2.'de görüldüğü üzere katılımcıların çok büyük bir çoğunluğu yazılım geliştirme süreçlerinde çevik metodolojiyi kullanmaktadır. Diğer model türleri de düşük oranda olmakla birlikte kullanılmaya devam etmektedir. Bu dağılım, katılımcıların çeşitli yazılım geliştirme yöntemlerine olan yatkınlıklarını ve tercihlerini göstermektedir. Çevik modelin yüksek oranda tercih edilmesi literatürde bu modelin çeşitli başarı faktörleri üzerinde olumlu etkiler yarattığına dair belirtimleri desteklemektedir.

Tablo 2. Yazılım Geliştirme Modellerinin Hangisinin Kullanıldığının Tespiti

		%
Yazılım geliştirme modellerinden kullanılan modeller	Agile (Çevik) Model	99,1
	Code&Fix Modeli	4,3
	Waterfall Iterative Model	16,4
	V Model	1,7
	Iterative Model	0,9
	Incremental Model	0,9
	Diğer	1,7

(Not: Bu sorunun cevapları çoklu olarak işaretlenmektedir. Yüzdelik değerler her bir cevabın toplam örneklem grubun içerisindeki oranını temsil etmektedir.)

Tablo 3.'de görüldüğü üzere çevik metodolojiler arasından scrum metodolojisinin kullanılma oranı yüksektir.

Tablo 3. Firmalarda Kullanılan Çevik Metodoloji Türleri

		%
Şirkette kullanılan çevik metodoloji türleri	Scrum	82,8
	Kanban	25,9
	Lean Software Development	3,4
	Test-Driven Development	4,3
	Feature Driven Development	0,9
	Dynamic Systems Development Method	1,7
	Diğer	0,9

(Not: Bu sorunun cevapları çoklu olarak işaretlenmektedir. Yüzdelik değerler her bir cevabın toplam örneklem grubun içerisindeki oranını temsil etmektedir.)

Tablo 4.'de görüldüğü üzere katılımcıların büyük bir çoğunluğu projelerinde manuel fonksiyonel testlerin yapıldığını belirtmiştir. Bu testler için bağımsız bir test ekibi bulunduranların oranı %47,8 olarak tespit edilmiştir. Projelerde analist bulunma oranı yüksek bir seviyede iken analistlerin kendi yazdıkları analizlerin tamamlandığında test işlemini kendilerinin yaptıkları da ortaya çıkmıştır. Genel görüş olarak, %95,7'lik kesim, analistlerin geliştirme sonrası testleri test ekibinden destek alarak yapmaları gerektiğini savunmuştur. Yazılım geliştiricilerin kendi kodlarını test etme oranları yüksek çıkmıştır. Projelerde test süreçleri için %76,5 oranında dokümantasyon yapıldığı, ancak yazılan test senaryolarının sadece %8,5 oranında gözden geçirildiğini belirtmiştir. Hata kayıtları için kullanılan araçlar arasında en yüksek kullanım oranı Jira olmuştur. Test ekibi büyüklüğü çoğunlukla 5-10 kişi arasında değişmekte, test yöneticisi bulundurma oranı ise %76,5 olarak tespit edilmiştir.. Dokümantasyon aracı olarak Confluence aracının diğer uygulamalara göre daha çok kullanıldığı ortaya çıkmıştır. Önceden belirlenmiş olan doküman formatlarının kullanılma eğilimi yüksektir. Bu bulgular yazılım geliştirme ve test süreçlerinde yüksek düzeyde standartlaşma ve organizasyon olduğunu göstermektedir.

Tablo 4. Projelerdeki Fonksiyonel Testler ve İşleyiş Süreçleri

		%
Manuel olarak fonksiyonel testler yapılma durumu	Var	98,3
	Yok	1,7
Fonksiyonel yazılım testler için bağımsız test ekibi olma durumu (dış kaynak da olabilir)	Var	47,8
	Yok	52,2
Projelerde analist olma durumu	Var	98,4
	Yok	1,6
Analistin kendi yazmış olduğu analizin geliştirmesi tamamlanınca biten işi kendisi test etme durumu	Var	38,8
	Yok	38,8
	Diğer	22,4
Analistlerin yazmış oldukları analizin geliştirmesi tamamlanınca testleri kendilerinin yapma veya test ekibinden destek alma durumları	Analistlerin kendi testlerini yapması	2,9
	Test için test ekibinden destek alınması	95,7

	Diğer	1,4
Kod geliştirenin kendi yazdığı kodu test etme durumu	Var	84,1
	Yok	15,9
Projelerdeki test süreçleri için dokümantasyon oluşturma	Var	76,5
	Yok	22,1
	Bilinmiyor, bir fikir yok	1,5
Projelerde test senaryolarının yazılmasın için kullanılan araç durumu	Jira	64,5
	HP ALM	8,1
	Excel	8,1
	Bilmiyorum, fikrim yok	9,7
	Diğer	9,7
Yazılan test senaryolarının gözden geçirilme durumu	Var	8,5
	Yok	88,1
	Bilinmiyor, bir fikir yok	3,4
Testi yapan kişilerin testler esnasında karşılaştığı hata kayıtlarını tutabilmek için kullandıkları araçlar	Jira	67,7
	HP ALM	10,8
	Excel	4,6
	E-mail	4,6
	Şirket içi sohbet iletişim aracı	1,5
	Bilmiyorum fikrim yok	4,6
	Diğer	6,2
Kod yazanların bu hata araçlarını kullanma durumları	Var	96,2
	Yok	1,9
	Bilinmiyor, bir fikir yok	1,9
Test ekibinde test yöneticisinin olma durumu	Var	76,5
	Yok	23,5
Test ekibi büyüklüğü	0-2	5,8
	2-5	33,0
	5-10	51,5
	Diğer	9,7
Projelerin genelinde dokümantasyon için kullanılan araçlar	Jira	29,8
	Confluence	40,4
	Excel	7,7
	Diğer	22,1
Projelerde kullanılan dokümanları belirli bir formatının olması durumu	Var	69,5
	Yok	30,5
Şirket içinde kullanılan standart doküman formatlarının isimleri	Fonksiyonel Analiz Dokümanı	29,1
	Gereksinim Analiz Dokümanı	22,1
	Teknik Analiz Dokümanı	33,7
	Bilmiyorum, fikrim yok.	5,8
	Diğer	9,3

(Not: Yukarıdaki sorular arasında çoklu olarak işaretlenen sorular mevcuttur. Bu sorular için yüzdelik değerler her bir cevabın toplam örneklem grubun içerisindeki oranını temsil etmektedir.)

Tablo 5. incelendiğinde üzere test ekibinin %32 oranla analiz sonrasında sürece dahil oldukları görülmektedir. Bundan dolayı projelerin başlangıcında yer almayan test ekibinin projenin erken aşamalarında oluşan karar süreçlerinden ve yön belirlemelerden uzak kaldıklarını göstermektedir. Dolayısıyla projenin ilerleyen safhalarında karşılaşılan zorlukların ve uyumsuzlukların bir nedeni bu durum olabilmektedir. Bu yüzden erken aşamalarda test ekibinin dahil edilmesi faydalıdır.

Tablo 5. Test Ekibinin Projelere Dahil Olma Zamanları

		%
Test ekibinin projeye dahil olma aşaması	Yazılım Başlarken	10,7
	Talep Esnasında	3,9
	Analiz Edilirken	12,6
	Analiz Sonrası	32,0
	Bilinmiyor, bir fikir yok	3,9
	Diğer	11,7

Tablo 6. incelendiğinde scrum metodolojisinin kullanıldığı projelerde test ekibinin planlama toplantılarına dahil edildiği görülmektedir. Projelerde test süreçleri için dokümantasyonda Jira uygulamasının daha yaygın olarak kullanıldığı tespit edilmiştir. Ayrıca daha farklı araçların test süreçlerindeki dokümantasyon işlemini karşıladığı görülmektedir. Test senaryoları yazma oranının yüksek olmasına rağmen bu senaryoların gözden geçirilmesinde belirsizliklerin olduğu görülmüştür. Test senaryolarında revizyon süreçlerinin netliğinin artırılması gerektiği ortaya çıkmaktadır. Test süreçlerinin işletilmesinde kişilerin birbirleriyle olan iletişim ve iletişim kanalları da önemli olmaktadır. Şirket içi iletişim araçları olarak, %50,5 ile Microsoft Teams ve %17,1 ile Slack en yaygın kullanılanlar arasında yer almıştır. Bu durum modern iletişim araçlarının iş yerinde iletişim ve işbirliğini desteklemede önemli bir rol oynadığını, tercih edilen platformlarında projelerin ve ekiplerin ihtiyaçları karşıladığını göstermektedir.

Tablo 6. Scrum Metodolojisinin Kullanıldığı Projelerde Test Süreci

		%
Scrum metodolojisinin uygulanması durumunda test ekibinin planlama toplantılarına dahil edilme durumu	Var	99,0
	Yok	1,0
Projelerdeki test süreçleri için dokümantasyon yapılması durumunda kullanılan araçlar	Jira	47,1
	HP ALM	3,9
	Excel	27,5
	Bilmiyorum, fikrim yok	5,9
	Diğer	15,7
Projelerde test senaryoları yazılma durumu	Var	84,7
	Yok	3,4
	Bilinmiyor, bir fikir yok	11,9
Yazılan test senaryolarının gözden geçirilme durumu	Var	8,3
	Yok	12,5
	Bilinmiyor, bir fikir yok	29,2
	Diğer	50,0
Şirket içi iletişim için kullanılan araç türleri	Cisco Jabber	13,3
	Microsoft Teams	50,5
	Skype	8,6
	Slack	17,1
	Diğer	10,5

Tablo 7.'de görüldüğü üzere test ekipleri, test süreçleri sırasında karşılaştıkları hataları kaydetmek için çeşitli araçlar kullanmaktadır. En yaygın kullanılan araç %32,8 ile Jira olmuştur. Diğer seçeneklerde göz önüne

alındığında bu çeşitlilik farklı test ekiplerinin farklı ihtiyaç ve tercihlere sahip olabileceğini göstermektedir. Bu da hata kayıtlarının tutulmasında verimlilik ve erişilebilirlik açısından önemli bir faktör olabilir. Bu durum, araç seçiminin özellikle hata izleme süreçlerinde nasıl stratejik bir önem taşıdığını ve test süreçlerinin etkinliğini artırma potansiyeline sahip olduğunu vurgulamaktadır.

Tablo 7. Test Ekiplerinin Test Süreçleri Sırasında Karşılaştıkları Hataları Kaydetmek İçin Kullandıkları Araçlar

		%
Test ekibinin testler esnasında karşılaştığı hata kayıtlarını tuttukları araçlar	Jira	32,8
	Excel	9,5
	Mail	8,6
	Şirket içi sohbet iletişim aracı	3,4
	Diğer	4,3

Tablo 8.'de görüldüğü üzere yazılımcılar genellikle hata detaylarına erişmek ve hataları çözdükten sonra işin statüsünü güncellemek amacıyla hata araçlarını kullanmaktadır. Bu da hatayı anlamak ve çözmek için doğrudan bu araçlara başvurduklarını göstermektedir. Ancak bazı durumlarda geliştiricilerin hata araçlarını aktif olarak kullanamadıkları da görülmektedir. Bu durum belirli koşullar altında erişim veya kullanım zorluklarının olabileceğini göstermektedir. Diğer bir yaklaşım olarak da geliştiricilerin hatanın detaylarına erişmek için hata araçları yerine hatayı bildiren kişiyle doğrudan iletişime geçmeyi tercih ettikleri tespit edilmiştir. Bu sonuç, hata çözüm sürecinde esnek bir iletişim yöntemi olarak karşımıza çıkmakta ve bazı durumlarda araç kullanımındaki zorlukları aşmak için alternatif bir yol sunmaktadır. Bu bulgular, yazılım geliştirme süreçlerinde hata yönetimi araçlarının kullanımının önemli olmasına rağmen geliştiricilerin bazen kişisel ve doğrudan iletişim yöntemlerine başvurduklarını göstermektedir.

Tablo 8. Yazılımcıların Kullandıkları Hata Araçları

		%
Yazılımcılar hata araçlarını nasıl kullandıkları	Hata detaylarına erişmek için	35,3
	Hatayı çözdükten sonra işin statüsünü ilerletmek için	32,8
	Hata aracı kullanılmamakta	9,5
	Hatanın detayına erişmede hata bildirimini yapan kişiye erişim için	10,3

(Not: Bu sorunun cevapları çoklu olarak işaretlenmektedir. Yüzdelik değerler her bir cevabın toplam örneklem grubun içerisindeki oranını temsil etmektedir.)

Tablo 9. incelendiğinde test ekibinin performansını ölçülmesinde herhangi bir metriğin çoğunlukla kullanılmadığı görülmektedir. Metriklerin kullanıldığı azınlık durumlarında ise test sorumlusunun yazdığı test durum sayısı, test sorumlusunun açtığı hata sayısı ve canlı ortamdan dönen hata sayısı gibi metrikler bulunmaktadır. Bu bulgular test metriklerinin kullanımının yaygın olmadığını ve test performansını ölçme pratiğinin genelde belirsiz veya standart dışı olduğunu göstermektedir. Ayrıca, test metriklerinin kullanılmaması test süreçlerinin değerlendirilmesinde ve iyileştirilmesinde potansiyel bir eksiklik olarak ortaya çıkmaktadır.

Tablo 9. Test Ekibinin Performansını Ölçmek İçin Kullanılan Metrikler

		%
Test ekibinin performansında test metrikleri kullanılma durumu, ve projelerde bulunan test özelinde çalışan ekibin ya da kişinin performansının ölçüm metriği	Kullanılıyor. Test yapanların yazdığı case sayısı	12,1
	Kullanılıyor. Test yapanların açtığı bug sayısı	7,8
	Kullanılıyor. Canlı ortamdan dönen hata sayıları	8,6
	Kullanılmıyor	69,0
	Bilinmiyor, bir fikir yok	13,8
	Diğer	4,3

Tablo 10.'da görüldüğü üzere test ekibinin performans değerlendirmelerinin çoğunlukla test yöneticisi tarafından yapılmaktadır. Yazılım ekip lideri ve yönetici rollerin performans değerlendirmeleri yapma oranlarının düşük olmaktadır. Bu durum test ekibinin performans değerlendirmelerinin büyük oranda test yönetimi odaklı yapıldığını, ancak bazı durumlarda bu sorumluluğun proje yönetimi veya üst düzey yönetim tarafından da üstlenildiğini göstermektedir.

Tablo 10. Test Ekibinin Performans Değerlendirmeleri

		%
Test ekibinin performans değerlendirmelerinin kim yaptığı	Test Yöneticisi	64,7
	Proje Yöneticisi	11,8
	Yazılım Ekip Lideri	3,9
	Direktör	3,9
	Bilmiyorum, fikrim yok	11,8
	Diğer	3,9

Tablo 11. incelendiğinde projelerde yapılan kullanıcı kabul testlerinin büyük oranda uygulandığı ve kabul gördüğü anlaşılmaktadır. Smoke testlerinin ise daha az yaygın olduğu görülmektedir. Bunun sebebinin projelerin veya organizasyonların ihtiyaç ve önceliklerine bağlı olduğu düşünülebilir.

Tablo 11. Projelerde Yapılan Test Türleri

		%
Projelerde smoke testlerin yapılma durumu	Var	23,5
	Yok	66,2
	Bilinmiyor, bir fikir yok	10,3
Projelerde kullanıcı kabul testlerinin yapılma durumu	Var	88,6
	Yok	7,9
	Bilinmiyor, bir fikir yok	3,5

Tablo 12. smoke testlerinin projelerde çeşitli roller tarafından uygulandığını göstermekte olup, firma içi ve dışı test ekiplerinin bu süreçte önemli bir rol oynadığını ve geniş bir katılım sağladığını ortaya koymaktadır.

Tablo 12. Projede Smoke Testlerini Gerçekleştiren Roller

		%
Smoke testlerinin projelerde hangi rollerin yaptığı durumu	Yazılım Geliştiriciler	22,7
	Analistler	18,2
	Son kullanıcılar	4,5
	Yazılım geliştiren firmadan bağımsız bir test ekibi	31,8
	Firma içinde kurulmuş ayrı test ekibi	40,9
	Bilinmiyor, bir fikir yok	4,5

(Not: Bu sorunun cevapları çoklu olarak işaretlenmektedir. Yüzdelik değerler her bir cevabın toplam örneklem grubun içerisindeki oranını temsil etmektedir.)

Tablo 13.'de kullanıcı kabul testlerinin projelerde geniş bir ekip tarafından yürütüldüğünü ve çeşitli departmanların bu süreçte önemli roller üstlendiği görülmektedir. Özellikle analistlerin ve iş birimlerinin bu testlerde merkezi bir role sahip oldukları da görülmektedir.

Tablo 13. Kullanıcı Kabul Testlerini Gerçekleştiren Roller

		%
Kullanıcı kabul testlerinin hangi roller tarafından yapıldığı	İş Birimleri	39,6
	Developerlar	4,5
	Analistler	48,6
	Son kullanıcılar	4,5
	Yazılım geliştiren firmadan bağımsız bir test ekibi	21,6
	Firma içinde kurulmuş ayrı test ekibi	8,1
	Bilinmiyor, bir fikir yok	4,5
	Diğer	32,4

Tablo 14'de incelendiğinde projelerde otomasyon testlerinin çoğunlukla yapıldığı, otomasyon testleri için bağımsız bir test ekibi bulundurulma oranın yüksek olduğu görülmektedir. Otomasyon test süreçlerindeki dokümantasyon durumunun istenen düzeyde olmadığı, otomasyon testlerine başlamadan önce test senaryolarının yazılması oranının ise yüksek olduğu görülmektedir. Ayrıca, otomasyonun veri oluşturmak için yüksek oranda kullanıldığı da tespit edilmiştir. Bu veriler, projelerde otomasyon testlerinin yaygın olarak kullanıldığını göstermektedir. Ancak otomasyon test süreçleri ve dokümantasyon konusunda çeşitlilik gösterdiğini ve bazı alanlarda iyileştirme ihtiyacının olduğunu da göstermektedir.

Tablo 14. Projelerde Otomasyon Testlerinin Yapılıp Yapılmadığı

		%
Projelerde otomasyon testlerinin yapılma durumu	Var	62,9
	Yok	31,9
	Bilinmiyor, bir fikir yok	5,2
Otomasyon testleri için bağımsız bir test ekibinin olması durumu	Var	48,8
	Yok	39,5
	Diğer	11,6
Otomasyon test ekibinin olmaması durumunda otomasyon testlerini kimin yaptığı	Analist	0,0
	Developer	75,0
	Diğer	25,0
Projelerin otomasyon test süreçlerindeki dokümantasyon durumu	Biraz	37,5
	Çok	37,5
	Hiç	12,5
	Bilinmiyor, bir fikir yok	12,5
Otomasyon testlerine başlanmadan önce otomasyon için ayrı test senaryolarının yazılma durumu	Var	82,9
	Yok	17,1
Projelerde otomasyonu veri oluşturmak için kullanılma durumu	Var	64,0
	Yok	36,0

Tablo 15.'de otomasyon testlerinin kullanım alanlarına bakıldığında yüksek oranla regresyon testlerinde kullanıldığı görülmektedir. Bu, otomasyonun tekrarlanabilir test süreçlerinde ne kadar değerli olduğunu göstermektedir. Genel değerlendirme yapıldığında otomasyonun çoğunlukla belirli ve yaygın test tiplerinde kullanıldığını, ancak bazı özel durumlar dışında genel kullanımının sınırlı olduğunu göstermektedir.

Tablo 15. Otomasyon Testlerinin Kullanım Alanları

		%
Test türlerine göre otomasyon testi kullanılma durumu	Regresyon	85,4
	Yük	22,0
	Performans	22,0
	Data oluşturma	22,0
	Diğer	2,4

(Not: Bu sorunun cevapları çoklu olarak işaretlenmektedir. Yüzdelik değerler her bir cevabın toplam örneklem grubun içerisindeki oranını temsil etmektedir.)

Tablo 16. incelendiğinde otomasyon süreçlerinin DevOps ile entegre olduğu görülmektedir. Bu bulgular katılımcıların çoğunun otomasyon süreçlerini DevOps kültürüyle uyumlu bir şekilde entegre ettiğini göstermektedir. Bu da süreçlerin daha hızlı ve verimli bir şekilde yürütülmesine olanak sağladığını göstermektedir.

Tablo 16. Otomasyon Süreçlerinin DevOps'a Entegrasyonu

		%
Otomasyon süreçlerinin devops'a entegre durumu	Var	75,0
	Yok	12,5
	Bilinmiyor, bir fikir yok	12,5

Tablo 17'de incelendiğinde projelerde geliştirme ve testler için kullanılan ortamlarda test ortamının mutlaka kullanıldığı görülmektedir. Bu tabloya göre geliştirme ve test süreçlerinde farklı ortamlara ihtiyaç duyulduğu ve her ortamın belirli amaçlarla kullanıldığı anlaşılmaktadır. Otomasyon testlerinin hangi ortamlarda çalıştırıldığına ilişkin değerlere bakıldığında preprod ortamında çalıştırmanın daha yüksek orana sahip olduğu görülür. Bu bulgular, test ortamının ve preprod ortamının otomasyon testleri için yaygın olarak tercih edildiğini gösterirken, bazı test ekiplerinin diğer ortamlarda da otomasyon testlerini gerçekleştirdiğini işaret etmektedir. Bu durum, projelerin farklı aşamalarında testlerin farklı

ortamlarda yürütülmesinin kalite güvencesi ve esneklik açısından önemli olduğunu yansıtmaktadır.

Tablo 17. Projelerde Geliştirme ve Testler İçin Kullanılan Ortamlar

		%
Projelerdeki geliştirmeler ve testler için kullanılan ortam türleri	Test ortamı	100,0
	Development ortamı	57,1
	Stable	7,1
	Preprod	85,7
	Prod	85,7
Otomasyon testlerinin hangi ortamlar için çalıştırılıyor olma durumu	Test ortamı	42,9
	Preprod	57,1
	Diğer	28,6

(Not: Yukarıdaki sorular arasında çoklu olarak işaretlenen sorular mevcuttur. Bu sorular için yüzdelik değerler her bir cevabın toplam örneklem grubun içerisindeki oranını temsil etmektedir.)

Tablo 18'de otomasyon test süreçleri hakkında yapılan incelemede belirlenmiş regresyon senaryo setlerine sahip olma oranı çok yüksek çıktığı görülmektedir. Otomasyon test senaryolarını yazmak için kullanılan araçlardan Jira'nın kullanılma oranının yüksek olduğu görülmektedir.

Tablo 18. Otomasyon Test Süreçleri Hakkında

		%
Otomasyon testleri için belirlenmiş olan regresyon senaryo setleri durumu	Var	92,9
	Yok	0,0
	Bilinmiyor, bir fikir yok	7,1
Belirlenmiş olan regresyon setlerini çalıştırılma sıklığı	Her hafta	20,0
	Her ayın sonunda	20,0
	Her yeni kod çıkışında	20,0
	Bilinmiyor, bir fikir yok	40,0
Otomasyon test senaryolarının yazılması için bir aracın kullanılma durumu	Var	69,2
	Yok	23,1
	Bilinmiyor, bir fikir yok	7,7
Otomasyon senaryolarının yazılması kullanılan araçlar	Jira	60,0
	HP ALM	0,0
	Excel	20,0
	Diğer	20,0
Otomasyon testlerinin sıklıkla kullanıldığı alanlar	Web	64,3
	Mobilweb	0,0
	Application	7,1
	Bilinmiyor, bir fikir yok	28,6

Tablo 19.'da projede test ekibinin bir test yöneticisine sahip olduğu, test yöneticisi bulunmayan durumlarda ise test yönetimi görevini genellikle yöneticilerin üstlenmiş olduğu görülmektedir.

Tablo 19. Projede Test Ekibinin Bir Test Yöneticisine Sahip Olup Olmama Durumu

		%
Otomasyon test ekibinde test yöneticisinin bulunma durumu	Var	56,4
	Yok	43,6
Test yöneticisinin olmadığı durumlar test yönetiminin yapan rol türü	Test ekibinden biri	0,0
	Analist	5,6
	Yazılım Geliştirici	0,0
	Yönetici	83,3
	Bilinmiyor, bir fikir yok	11,1

Tablo 20.'de incelendiğinde otomasyon test senaryolarının yazımında büyük oranla Selenium Framework'ün tercih edildiği ve bu aracın popülerliğinin yüksek olduğu görülmektedir. Appium Framework'ün ise %8,6 gibi daha düşük bir oranda tercih edilmesinin sebebi ise mobil uygulama testleri için özel bir gereksinim ihtiyacının olmasıdır. Postman'ın %25,9'luk oranla tercih edilmesi API testleri için bu aracın kullanımının yaygınlığını göstermektedir. Bu veriler, test otomasyon araçlarının çeşitliliğini ve farklı ihtiyaçları karşılayabilme kapasitesini yansıtmaktadır. Ancak popüler araçların hala önde olduğunu göstermektedir.

Tablo 20. Otomasyon Test Senaryolarını Yazmak İçin Tercih Edilen Araçlar

		%
Web otomasyon testleri için kullanılan araçlar	Selenium Framework	65,5
	Appium Framework	8,6
	Postman	25,9
	Soap	5,2
	Diğer	13,8
	Bilinmiyor, bir fikir yok	1,7

(Not: Bu sorunun cevapları çoklu olarak işaretlenmektedir. Yüzdelik değerler her bir cevabın toplam örneklem grubun içerisindeki oranını temsil etmektedir.)

6. Sonuç

Yüzdesel oranlarla bir önceki bölümde yapılan değerlendirmeler neticesinde Türkiye'deki yazılım geliştirme sektöründe çevik metodolojilerinden özellikle scrum metodolojisinin yoğun bir şekilde benimsendiği görülmektedir. Fonksiyonel testlerin büyük çoğunluğunun manuel olarak yürütüldüğü ve dış kaynak kullanımı oldukça sınırlı düzeyde kaldığı tespit edilmiştir. Projelerde analistlerin aktif rol aldığı gözlemlenirken, bu analistlerin geliştirme sonrası kod testleri konusunda bölünmüş farklı görüşlere sahip oldukları ortaya çıkmıştır. Yazılımcılarım genellikle kendi kodlarını test ettikleri bir kez daha ortaya çıkmıştır. Yazılım geliştirme sürecinde test dokümantasyonu ve test senaryolarının incelenmesi gibi temel adımların sıklıkla ihmal edildiği görülmüştür. Test süreçlerinin yetersiz belgelenmesi ve test senaryolarının gözden geçirilmesinin atlanması projelerin genel kalitesini olumsuz yönde etkilediği aşikardır. Bu bulgular, scrum metodolojisi ile yürütülen yazılım projelerinde de test süreçlerinin etkinliğini artırma ve kaliteyi iyileştirme yönünde önemli adımların atılması gerektiğini işaret etmektedir. Bu iyileştirmeler arasında daha kapsamlı test dokümantasyonu, düzenli test senaryosu incelemeleri ve test ekiplerinin daha etkin yönetilmesi yer almaktadır. Test ekibinin planlama toplantılarına

yüksek oranda katılımının olması projenin ilerleyişi ve test faaliyetlerinin etkin şekilde planlanmasına katkı sağlamaktadır. Test süreçlerinin dokümantasyonu için yaygın olarak Jira ve Excel'in kullanılmakta olduğu, bu araçların test senaryolarının oluşturulması, izlenmesi ve test sonuçlarının raporlanması gibi temel işlevleri destekledikleri görülmektedir.

Yazılım projelerinde test ekibinin işlevlerini, kullandıkları araçları ve performans değerlendirme yöntemlerini kapsamlı bir şekilde incelenmiştir. Test süreçlerinde tespit edilen hataların kayıtlarının tutulması, hata yönetiminin organize edilmesi ve geliştirme sürecinde izlenebilirliğin arttırılması için Jira ve Excel gibi araçlar yaygın olarak kullanıldığı görülmektedir. Test ekibinin, projelerde çeşitli ek roller üstlendiği ve smoke testleri ile kullanıcı kabul testlerinin yönetilmesinde önemli görevler aldığı görülmektedir. Bu süreçlerdeki belirsizlikler ve bilgi eksiklikleri belirli test süreçlerinde önemli faktörler olarak dikkate alınmalıdır. Bu noktadan bakıldığında bu çalışma sayesinde test ekiplerinin projelerdeki rolleri ve sorumlulukları, kullanılan araçlar ve performans değerlendirme pratikleri detaylı bir şekilde ortaya konmuştur. Test süreçlerinin yönetimi ve optimizasyonu için daha belirgin ve sistemli bir yaklaşımın önemi bu çalışmada da tespit edilmiştir. Kullanıcı kabul testlerini gerçekleştiren rollerin incelenmesi neticesinde çeşitli paydaşların bu önemli sürece katılımı gösterdiğini ortaya koymaktadır. Çalışma sonucuna göre, kullanıcı kabul testlerini çoğunlukla analistler ve iş birimleri tarafından gerçekleştirildiği tespit edilmiştir. Bağımsız test ekiplerinin de kullanıcı kabul testlerine katıldıkları ve test sürecinin tarafsızlığını ve kalitesini artırma amacıyla dış kaynaklı uzmanların dahil edildiğini işaret etmektedir. Kullanıcı kabul testlerinin farklı roller tarafından gerçekleştirilmesi, test sürecinin çeşitli perspektiflerden değerlendirilmesine ve uygulamanın gereksinimlere uygunluğunun sağlanmasına yardımcı olmaktadır. Bu durum test sürecinin kapsamlı ve etkili bir şekilde yürütülmesini sağlamaktadır.

Test süreçlerindeki eksikliklerin temel nedenlerinin analiz edilmesi neticesinde aşağıda belirtilen noktalar ortaya çıkmıştır.

Elde edilen verilere göre, projelerde test süreçleri için dokümantasyon oluşturma oranı %76,5 olarak tespit edilmiştir. Ancak, yazılan test senaryolarının yalnızca %8,5'inin gözden geçirildiği görülmektedir. Bu durum, test senaryolarının doğruluğunun ve güncelliğinin sağlanamaması nedeniyle yazılım kalitesini olumsuz yönde etkilemektedir. Dokümantasyon eksikliğinin temel nedenleri arasında zaman kısıtlamaları, test süreçlerinin ikincil öneme sahip görülmesi ve test dokümantasyon araçlarının yetersiz kullanımı gösterilebilir.

Test ekiplerinin performansını ölçmek için metriklerin büyük oranda kullanılmadığı belirlenmiştir (%69). Kullanılan metrikler arasında test vakalarının sayısı, açılan hata sayısı ve canlı ortamdan dönen hata sayıları yer almaktadır. Ancak, yazılım geliştirme sürecinde test süreçlerinin etkinliğini ölçmek için kullanılan kapsamlı metriklerin eksikliği, kaliteyi artırma ve iyileştirme çabalarını sınırlamaktadır. Test metriklerinin kullanılmamasının nedenleri arasında bu metriklerin belirlenmesi ve uygulanması için gereken altyapı eksikliği, organizasyonel önceliklerin farklı olması ve test süreçlerinin yeterince değer görmemesi sayılabilir.

Veriler, test ekiplerinin %32 oranında analiz aşamasından sonra projelere dahil olduğunu göstermektedir. Bu durum, test süreçlerinin proje başında yeterince planlanamamasına ve erken aşamalarda test stratejilerinin oluşturulamamasına yol açmaktadır. Test ekiplerinin erken aşamalarda projeye dahil edilmemesi, yazılım geliştirme sürecinde sonradan ortaya çıkan hataların maliyetini artırmakta ve yazılım kalitesini olumsuz etkilemektedir.

Eksikliklerin proje başarısına olan etkileri;
Kalite Güvencesinin Zayıflaması: Test metriklerinin eksikliği, proje süreçlerinin değerlendirilmesini zorlaştırarak kalite güvence sistemlerinin etkin bir şekilde çalışmasını engellemektedir. Bu durum, yazılım ürünlerinde hata oranının artmasına ve müşteri memnuniyetinin azalmasına yol açabilir.

Proje Sürelerinin Uzaması: Test süreçlerinde dokümantasyon eksikliği ve test senaryolarının gözden geçirilmemesi, test süreçlerinde tekrarlanan hatalara ve gereksiz revizyonlara neden olarak proje teslim sürelerini uzatmaktadır.

Hata Yönetiminin Zayıf Olması: Test ekibinin projelere geç dahil olması, test süreçlerinde kritik hataların erken aşamalarda tespit edilmesini engellemekte ve bu hataların geliştirme sürecinin ilerleyen aşamalarında düzeltilmesini zorlaştırmaktadır.

Çevik yazılım geliştirme metodolojilerinde test süreçlerinin kritik bir rol oynadığı vurgulanmaktadır. Trihardianingsih ve arkadaşlarının [2] yaptığı çalışmada, scrum metodolojisinin test süreçlerine erken entegrasyonunun yazılım kalitesini artırdığı ve proje başarısını olumlu yönde etkilediği belirtilmiştir. Ancak, bu çalışmada elde edilen veriler, Türkiye'deki yazılım geliştirme firmalarında test süreçlerinin genellikle son aşamalara bırakıldığını ve test metriklerinin sistematik bir şekilde kullanılmadığını göstermektedir. Sektörel açıdan değerlendirildiğinde, dünya çapında yazılım firmalarının test süreçlerinde otomasyon oranını artırmaya yönelik yatırımlar yaptığı görülmektedir. Örneğin, Rajasekhar ve Shafi [10] tarafından yapılan araştırmada, yazılım projelerinde otomasyon testlerinin kullanılmasının

geliştirme sürelerini %30 oranında azalttığı ve hata oranlarını önemli ölçüde düşürdüğü belirtilmiştir. Ancak, bu çalışmada elde edilen veriler, Türkiye'deki firmaların yalnızca %62,9'unun otomasyon testlerini kullandığını ve bu testlerin genellikle regresyon testleriyle sınırlı kaldığını göstermektedir.

Bu çalışmada, yazılım geliştirme projelerinde scrum metodolojisinin yüksek oranda kullanıldığı ve otomasyon testlerinin uygulanışı ve etkinliği derinlemesine incelenmeye çalışılmıştır. Test süreçlerinde özellikle regresyon, yük ve performans testlerinin yanı sıra veri oluşturma için otomasyon test uygulamalarının tercih edildikleri de görülmektedir. Bu bulgular doğrultusunda otomasyon testlerinin scrum metodolojisinin uygulandığı yazılım geliştirme sürecinde de önemli bir rol oynadığı bu çalışmada da ispatlanmıştır. Bu testlerin sıklıkla ve düzenli aralıklarla uygulandığı, ancak performans testleri ve bazı test ortamlarının kullanımının iyileştirilmesi gerektiği anlaşılmaktadır. Bu araştırma çalışması neticesinde aşağıdaki belirtilenlerin literatürde belirtilenlerle birlikte dikkate alınması önemlidir.

- Scrum metodolojisinin sahip olduğu süreçlere otomasyon testlerinin entegrasyonu, test süreçlerinin verimliliğini artırabilecektir. Projelerin zamanında ve hatasız teslim edilmesine olanak tanıyabilecek şekilde otomasyon testleri derinleştirilmelidir. Bu bütünleşme, özellikle regresyon testlerinin otomasyonu üzerinden süreç hızının ve hata tespit kabiliyetlerinin geliştirilmesini sağlar nitelikte olmalıdır.
- Test süreçlerinin eksiksiz bir şekilde dokümante edilmesi, hata yönetimini ve kalite kontrol süreçlerini iyileştireceği saha uygulamaları neticesinde bir kez daha ispatlanmıştır. Test süreçlerinde gerçekleştirilen adımların, hata kayıtlarının ve çözüm süreçlerinin detaylı bir şekilde belgelenmesi, bu süreçlerin gelecekte daha etkin bir şekilde yönetilmesine olanak sağlayacaktır.
- Scrum metodolojisinin uygulandığı yazılım geliştirmede test süreçlerinin performansını ölçmek için metriklerin geliştirilmesi ve bu metriklerin sistemli bir şekilde kullanılması gerektiği ortaya çıkmaktadır. Bu durum, test süreçlerinin etkinliğini değerlendirmek ve sürekli iyileştirme yapmak için kritik öneme sahiptir. Test yöneticileri, bu metrikleri kullanarak süreçlerin verimliliğini artırma ve potansiyel problemleri erken tespit etme konusunda daha bilinçli kararlar alabilir noktaya gelecektir.
- Scrum metodolojisiyle uyumlu olacak şekilde sürekli test mekanizmaları oluşturulmalıdır.
- Otomasyon testleri her sprint sonunda otomatik olarak çalıştırılmalı ve sonuçları analiz edilmelidir.
- Test ekibi, geliştirme ekibi ile birlikte sprint planlama ve backlog refinement süreçlerine katılmalıdır.
- Test stratejisi sprint başında belirlenmeli ve test senaryoları geliştirme ile paralel olarak hazırlanmalıdır.
- Otomasyon testleri sadece regresyon testleriyle sınırlı kalmamalı, yük ve performans testleri de dahil edilmelidir.
- Selenium, Postman ve Appium gibi test araçlarının entegrasyonu sağlanarak test kapsamı genişletilmelidir.

Test süreçlerinde kaliteyi artırmak için metriklerin düzenli kullanımı büyük önem taşımaktadır. Test metriklerinin etkin kullanımını teşvik etmek için belirtilen adımlar uygulanabilir.

- Test kapsamı (% kaç testin tamamlandığı), hata yoğunluğu (hata başına düşen satır sayısı) ve hata geri dönüş oranı gibi metriklerin kullanılması teşvik edilmelidir.
- IEEE 829 Yazılım Test Standartları'na uygun bir test metriği seti oluşturulmalıdır [12].
- Scrum toplantılarında test metrikleri düzenli olarak ele alınmalı ve sprint bazlı test performansı ölçülmelidir.
- Metriklerin görünürlüğünü artırmak için Jira, TestRail gibi araçlarda otomatik raporlama mekanizmaları oluşturulmalıdır.
- Test mühendisleri ve yazılım geliştiriciler için metrik kullanımının önemi konusunda eğitim programları düzenlenmelidir.
- Ekiplerin metrikleri nasıl analiz edeceği ve yorumlayacağı konusunda rehber dokümanlar hazırlanmalıdır.

Test süreçlerinin verimli yürütülmesi için dokümantasyonun eksiksiz olması da gerekmektedir. Bu doğrultuda aşağıdaki iyileştirme adımları önerilmektedir.

- Gereksinim analizi, test planı ve test senaryoları için şablonlar oluşturulmalıdır.
- Test dokümantasyonunda IEEE 829 veya ISO/IEC 29119 standartlarına uyum sağlanmalıdır [13].
- Test senaryolarının güncelliğini sağlamak için Confluence, TestRail veya Zephyr gibi araçlar kullanılmalıdır.
- Dokümantasyonun güncellenme sıklığı artırılmalı ve her sprint sonunda gözden geçirilmelidir.
- Test dokümantasyonlarının belirli kalite kriterlerine göre düzenli olarak incelenmesi sağlanmalıdır.
- Test senaryolarının geçerliliğini kontrol eden iç denetim süreçleri uygulanmalıdır.

Etik Beyanı

Bu çalışmada, "Yükseköğretim Kurumları Bilimsel Araştırma ve Yayın Etiği Yönergesi" kapsamında uyulması gerekli tüm kurallara uyulduğunu, bahsi geçen yönergenin "Bilimsel Araştırma ve Yayın Etiğine Aykırı Eylemler" başlığı altında belirtilen eylemlerden hiçbirinin gerçekleştirilmediğini taahhüt ederiz.

Kaynakça

[1] Kate, V., Bhalerao, S., & Sharma, V. 2023. Exploring Agile methodologies in educational software development-A comparative analysis and project management insights. In 2023 IEEE International Conference on ICT in Business Industry & Government (ICTBIG), 1-11.

[2] Trihardianingsih, L., Istighosah, M., Alin, A. Y., & Asgar, M. R. G. 2023. Systematic literature review of trend and characteristic agile model. Jurnal Teknik Informatika, 16(1), 45-57.

[3] Zhao, Y., Hu, Y., & Gong, J. 2021. Research on international standardization of software quality and software testing. In 2021 IEEE/ACIS 20th International Fall Conference on Computer and Information Science (ICIS Fall) 56-62.

[4] Malik, V., & Singh, S. 2017. Tools, strategies & models for incorporating software quality assurance in risk oriented testing. Oriental Journal of Chemistry, 10(3), 603-611

[5] Mateen, A., Zhu, Q., & Afsar, S. 2018. Comparative analysis of manual vs automated testing for software quality. In Proceedings of the 7th International Conference on Software Engineering and New Technologies, 1-7

[6] Garg, P. 2015. Role of testing strategies in improving quality of software. International Journal of Research, 2(12), 800-805.

[7] Srivastava, N., Kumar, U., & Singh, P. 2021. Software and performance testing tools. Journal of International Electrical and Electronics Engineering, 2(1), 1-12

[8] Sultania, A. K. 2015. Developing software product and test automation software using Agile methodology. In Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT) 1-4.

[9] Kumar, P. S., & Siva, N. S. 2014. Automation of software testing in agile development-An approach and challenges with distributed database systems. International Global Journal for Research Analysis, 3(7), 102-104.

[10] Rajasekhar, P., & Shafi, R. M. 2014. Agile software development and testing: Approach and challenges in advanced distributed systems. Global Journal of Computer Science and Technology-GJCST-B, 14(1), 7-10

[11] Saleem, R. M., Qadri, S., ul Hassan, I., Bashir, R. N., & Ghafoor, Y. 2014. Testing automation in agile software development. International Journal of Innovation and Applied Studies, 9(2), 541

[12] IEEE (2008). IEEE Standard for Software Test Documentation (IEEE 829-2008). *IEEE Computer Society*.

[13] ISO (2013). ISO/IEC 29119-1:2013 Software and Systems Engineering – Software Testing – Part 1: Concepts and Definitions. *International Organization for Standardization*.