DÜZCE ÜNİVERSİTESİ | LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

# TÜRKİYE'DEKİ FONKSİYONEL VE OTOMASYON TESTLERİNİN AGİLE PROJELERDEKİ SÜREÇLERİ VE İŞLEYİŞ ANALİZİNİN DEĞERLENDİRİLMESİ

**HAYRİYE KATRANCI**

**YÜKSEK LİSANS TEZİ**
**ELEKTRİK-ELEKTRONİK VE BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI**

**DANIŞMAN**
**DOÇ. DR. SERDAR BİROĞUL**

**DÜZCE, 2024**

**T.C.**
**DÜZCE ÜNİVERSİTESİ**
**LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ**

# TÜRKİYE'DEKİ FONKSİYONEL VE OTOMASYON TESTLERİNİN AGİLE PROJELERDEKİ SÜREÇLERİ VE İŞLEYİŞ ANALİZİNİN DEĞERLENDİRİLMESİ

Hayriye Katrancı tarafından hazırlanan tez çalışması aşağıdaki jüri tarafından Düzce Üniversitesi Lisansüstü Eğitim Enstitüsü Elektrik-Elektronik ve Bilgisayar Mühendisliği Anabilim Dalı'nda **Yüksek Lisans TEZİ** olarak kabul edilmiştir.

**Tez Danışmanı**

Doç. Dr. Serdar BİROĞUL

Düzce Üniversitesi

**Jüri Üyeleri**

Doç. Dr. Serdar BİROĞUL
Düzce Üniversitesi _____

Prof. Dr. Ali ÇALHAN
Düzce Üniversitesi _____

Doç. Dr. Osman KARACA
Muğla Sıtkı Koçman Üniversitesi
_____

Tez Savunma Tarihi: 23/07/2024

# BEYAN

Bu tez çalışmasının kendi çalışmam olduğunu, tezin planlanmasından yazımına kadar bütün aşamalarda etik dışı davranışımın olmadığını, bu tezdeki bütün bilgileri akademik ve etik kurallar içinde elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara kaynak gösterdiğimi ve bu kaynakları da kaynaklar listesine aldığımı, yine bu tezin çalışılması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını beyan ederim.

Ağustos 2024

(İmza)

Hayriye KATRANCI

# TEŞEKKÜR

Yüksek lisans öğrenimimde ve bu tezin hazırlanmasında gösterdiği her türlü destek ve yardımdan dolayı çok değerli hocam Doç. Dr. Serdar BİROĞUL'a en içten dileklerimle teşekkür ederim.

Bu çalışma boyunca yardımlarını ve desteklerini esirgemeyen sevgili aileme, kıymetli eşime, sürekli çalışmama izin veren oğlum Mete Kayra ve kızım Alara Beren'e, motivasyon desteği için Funda BELSU'ya sonsuz teşekkürlerimi sunarım.

**Ağustos 2024** **Hayriye KATRANCI**

# İÇİNDEKİLER

# ŞEKİL LİSTESİ

**Sayfa No**

# ÇİZELGE LİSTESİ

**Sayfa No**

# GRAFİK LİSTESİ

**Sayfa No**

# KISALTMALAR

| | |
|---|---|
| **ISTQB** | International Software Testing Qualifications Board |
| **MVC** | Model-View-Controller |
| **QA** | Quality Assurance |
| **QTP** | Quick Test Professional |
| **V-Model** | Validation and Verification Model |
| **XP** | Extreme Programming |

# ÖZET

## TÜRKİYE'DEKİ FONKSİYONEL VE OTOMASYON TESTLERİNİN AGİLE PROJELERDEKİ SÜREÇLERİ VE İŞLEYİŞ ANALİZİNİN DEĞERLENDİRİLMESİ

Hayriye KATRANCI

Düzce Üniversitesi
Lisansüstü Eğitim Enstitüsü, Elektrik-Elektronik ve Bilgisayar Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi
Danışman: Doç Dr. Serdar BİROĞUL

Ağustos 2024, 102 sayfa

Bu araştırma, Türkiye'deki çevik yazılım geliştirme projelerinde fonksiyonel ve otomasyon test süreçlerinin uygulanışını ve bu süreçlerin projelerin başarısına etkilerini incelemektedir. Çalışma, çevik metodolojilerin adaptasyonu, özellikle Scrum ve Kanban yöntemlerinin benimsenmesi ve test süreçlerinin bu metodolojilere entegrasyonu üzerine yoğunlaşmaktadır. Araştırmanın amacı, fonksiyonel ve otomasyon test süreçlerinin çevik çerçevesinde nasıl optimize edildiğini ve bu optimizasyonların projelerin verimliliği ve başarısı üzerindeki etkilerini anlamaktır.

Katılımcılar arasında yazılım mühendisleri, proje yöneticileri ve kalite güvence uzmanları gibi çeşitli rollerden bireyler bulunmaktadır. Araştırma, demografik dağılım, iş ve eğitim geçmişi, çevik ve Scrum metodolojilerine aşinalık düzeyleri ile test tecrübelerini kapsayan anketler aracılığıyla gerçekleştirilmiştir. Bulgular, çevik modellerinin yaygın olarak benimsendiğini ve test süreçlerinin çoğunlukla otomasyon desteğiyle gerçekleştirildiğini ortaya koymaktadır.

Analiz sonuçları, çevik metodolojilerin ve özellikle Scrum'un, test süreçlerinin entegrasyonunda ve yönetiminde etkili olduğunu göstermektedir. Bu metodolojilerin, değişen gereksinimlere hızlı bir şekilde adapte olabilme ve sürekli iyileştirme fırsatı sunma avantajları vurgulanmıştır. Araştırma, Türkiye'deki yazılım geliştirme sektöründe çevik metodolojilerin daha etkin kullanılmasına yönelik stratejik öneriler sunarak, yazılım kalitesinin artırılmasına katkıda bulunmayı hedeflemektedir. Bu çalışma, hem akademik literatüre hem de sektörel uygulamalara yönelik değerli katkılar sağlamaktadır.

**Anahtar Sözcükler:** Agile Yazılım Geliştirme, Scrum, Kanban, Fonksiyonel Testler, Otomasyon Testleri

# ABSTRACT

## EVALUATION OF THE PROCESSES AND OPERATIONAL ANALYSIS OF FUNCTIONAL AND AUTOMATION TESTS IN AGILE PROJECTS IN TURKEY

Hayriye KATRANCI
Düzce University
Graduate School, Department of Electrical-Electronics and Computer Engineering
Master Thesis
Supervisor: Doç Dr. Serdar BİROĞUL

August 2024, 102 pages

This research investigates the implementation of functional and automation testing processes within Agile software development projects in Turkey and examines their impacts on project success. The study focuses on the adoption of Agile methodologies, particularly the integration of Scrum and Kanban practices, and how these practices enhance testing processes. The aim is to understand how functional and automation testing are optimized within the Agile framework and to identify the effects of these optimizations on project efficiency and success.

Participants include individuals from various roles such as software engineers, project managers, and quality assurance specialists. The research is conducted through surveys that cover demographic distribution, work and educational backgrounds, familiarity with Agile and Scrum methodologies, and testing experiences. The findings indicate a widespread adoption of Agile models and a predominant implementation of testing processes supported by automation.

The analysis highlights the effectiveness of Agile methodologies, especially Scrum, in integrating and managing testing processes. The advantages of these methodologies in adapting to changing requirements quickly and providing continuous improvement opportunities are emphasized. The research aims to contribute to enhancing software quality by offering strategic recommendations for more effective use of Agile methodologies in Turkey's software development sector. This study provides valuable contributions to both academic literature and industry practices.

**Keywords:** Agile Software Development, Scrum, Kanban, Functional Testing, Automation Testing

# 1. GİRİŞ

Günümüz yazılım geliştirme sektöründe, projelerin karmaşıklığı ve sürekli değişen müşteri ihtiyaçları, geleneksel yazılım geliştirme yöntemlerinin ötesinde daha esnek ve adaptif yaklaşımlara olan ihtiyacı artırmaktadır. Bu ihtiyaçtan doğan çevik (Agile) metodolojiler, proje yönetimi ve yazılım geliştirme süreçlerine yaklaşımı köklü bir şekilde değiştirmiştir. Özellikle Scrum, Kanban, Extreme Programming gibi çevik uygulamalar, projelerde müşteri memnuniyetini ve ürün kalitesini yükseltmek, üretkenliği artırmak ve değişen gereksinimlere hızlı bir şekilde yanıt vermek için yaygın olarak kullanılmaktadır. Bu dönüşüm, yazılım test süreçlerine de yansımış ve yazılım kalitesini güvence altına almak için fonksiyonel ve otomasyon testlerine olan talep artmıştır.

Fonksiyonel testler, yazılım ürünlerinin belirlenen işlevsel gereksinimleri karşılayıp karşılamadığını kontrol ederken, otomasyon testleri, bu süreci hızlandırmak ve tekrarlanabilirliği artırmak amacıyla manuel testlerden daha hızlı ve hatasız bir yaklaşım sunmaktadır. Bu çerçevede, çevik projelerde test süreçleri; hızlı teslimat, sürekli entegrasyon, küçük ve bağımsız geliştirme ekipleri gibi çevik yaklaşımların ihtiyaçlarına uyum sağlayacak şekilde yeniden düzenlenmiştir. Bu nedenle, fonksiyonel ve otomasyon testlerinin çevik projelerdeki rolü ve işleyişi, proje başarısı için kritik hale gelmiştir.

Bu tez çalışmasında, Türkiye'deki 52 tane yazılım geliştiren firmalar daki çevik projelerde fonksiyonel ve otomasyon test süreçlerinin detaylarını incelenerek bu süreçlerin projelerin işleyiş şekline etkilerinin analizi yapılmıştır. Çalışmanın kapsamı, Türkiye yazılım sektöründeki çevik uygulamalarının benimsenmesi, fonksiyonel ve otomasyon testlerinin süreçlere entegrasyonu ve bu entegrasyonun yazılım kalitesine katkıları üzerine derinlemesine bir analiz sunmaktır. Yarı yapılandırılmış görüşme formları aracılığıyla toplanan veriler, katılımcıların demografik bilgilerini, profesyonel deneyimlerini, çevik metodolojilerine yaklaşımlarını ve test süreçlerinde karşılaştıkları zorlukları detaylı bir şekilde ele alınmıştır. Bu kapsamda, çalışmanın sonuçları; yazılım firmaları, proje yöneticileri, test mühendisleri ve yazılım geliştiricileri için önemli

içgörüler sunması ve Türkiye'deki çevik projelerin test süreçlerinin daha verimli hale getirilmesi için yol gösterici olması amaçlanmıştır.

Bu çalışma aynı zamanda 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 olacaktır. Bu çalışma ile birlikte, çevik projelerdeki test süreçlerinin kalitesinin artırılması, sektör paydaşlarının test süreçlerine yönelik bilinçlendirilmesi ve firmaların test stratejilerini optimize etmelerine katkı sağlanması beklenmektedir.

Bu doğrultuda araştırmanın kuramsal temellerinin yer aldığı birinci bölümde, "Yazılım Geliştirmede Geleneksel Yöntemler" başlığı altında, Geleneksel Yazılım Geliştirme Süreci, Şelale (Waterfall) Modeli ve V-Modeli gibi klasik yaklaşımlar ele alınmıştır. Bu modellerin yapısı, uygulama alanları ve bu yaklaşımların sınırlılıkları ile sektörden aldığı eleştiriler detaylı bir şekilde incelenmiştir. Ardından, "Yazılım Geliştirmede Çevik Yöntemler" bölümünde, Çevik Metodolojinin Temelleri ve popüler çevik yaklaşımlar olan Scrum, Extreme Programming (XP) ve Kanban metodolojileri anlatılmıştır. Bu metodolojilerin avantajları ve karşılaştıkları zorluklar, çevik yöntemlerdeki roller ve sorumluluklar ile birlikte değerlendirilerek, yazılım kalitesi ve test stratejilerine olan etkileri değerlendirilmiştir. "Yazılımda Kalite ve Test" başlığı altında, kalitenin tanımı, kalite güvencesi ve yazılım testinin rolü gibi önemli konulara değinilmiş ve çeşitli test stratejileri sunulmuştur. "Yazılımda Test Seviyeleri ve Test Türleri" bölümünde ise, birim testleri, entegrasyon testleri, sistem testleri ve kabul testleri gibi farklı test seviyeleri ve fonksiyonel testler ile otomasyonun rolleri ele alınmıştır. "Otomasyon Testleri" bölümünde, otomasyon testlerinin tanımı, önemi, kullanılan araçlar, çevik projelere entegrasyonu ve bu süreçte karşılaşılan zorluklar anlatılmıştır. Son olarak "İlgili Araştırmalar" bölümünde ise, konuyla ilgili önceki çalışmalar ve elde edilen bulgular gözden geçirilmiştir.

Bu tez çalışmasının "Materyal ve Yöntem" bilgilerinin yer aldığı ikinci bölümde, araştırmanın yapısal çerçevesini ve kullanılan yöntemleri kapsamlı bir biçimde açıklanmıştır. "Materyal" bölümü altında, araştırmada kullanılan ana veri toplama aracı olan görüşme formunun içeriği, bu formda yer alan çevik metodolojileri üzerine sorular, Scrum ve Kanban metodolojilerinin detayları, fonksiyonel ve otomasyon test süreçleri

ile ilgili sorular derinlemesine ele alınmıştır. Bu sorular, katılımcıların çeşitli çevik metodolojiler ve test süreçlerine dair deneyimlerini ve perspektiflerini detaylı bir şekilde ortaya konmuştur. "Yöntem" başlığı altında ise, araştırmanın amacı ve önemi vurgulanarak, çalışmanın geniş kapsamı ve hedefleri belirtilmiştir. Araştırmanın evreni ve örneklemi, Türkiye'deki çevik projelerde çalışan yazılım profesyonelleri olarak tanımlanırken, bu örneklem üzerinden elde edilecek verilerin çeşitliliği ve temsil kabiliyeti tartışılmıştır. Verilerin toplanması süreci, katılımcılara dağıtılan çevrimiçi anketler ve yarı yapılandırılmış görüşmeler aracılığıyla yapılandırılmış olup, bu sürecin nasıl gerçekleştirildiği ve hangi tekniklerin kullanıldığı açıklanmıştır. Verilerin analizi, hem nitel hem de nicel veri analiz yöntemlerini içeren karma bir metodoloji kullanılarak, elde edilen bulguların nasıl değerlendirildiği ve yorumlandığı detaylarıyla ortaya konmuştur.

Araştırmanın üçüncü bölümünde elde edilen bulgulara ve bulguların tartışılmasına yer verilmiştir. Araştırmanın son bölümünde ise elde edilen bulgulara yönelik sonuç ve öneriler ifade edilmiştir.

# 2. KURAMSAL TEMELLER ve KAYNAK ARAŞTIRMASI

## 2.1. YAZILIM GELİŞTİRMEDE GELENEKSEL YÖNTEMLER

Geleneksel yazılım geliştirme yöntemleri, projenin başlangıcından sonuna kadar aşamalı bir yaklaşım benimser. Bu metodolojiler, genellikle "şelale" modeli olarak bilinen, gereksinimlerin belirlenmesi, tasarım, kodlama, test etme ve bakım gibi kesin ayrılmış evreleri içerir. Geleneksel yöntemler, özellikle iyi tanımlanmış ve değişiklik göstermeyen gereksinimleri olan projeler için uygundur. Bu yöntemler, projenin her aşamasında belirgin bir disiplin ve düzen gerektirir ve değişikliklere adapte olmada zorluklar yaşayabilir (Fertalj & Katic, 2008).

Yazılım geliştirme, karmaşık ve katmanlı bir süreçtir ve bu sürecin yönetimi, projenin başarısında kritik bir rol oynar. Geleneksel yazılım geliştirme süreci, planlama, gereksinim analizi, tasarım, uygulama, test ve bakım olmak üzere altı ana aşamadan oluşur. Her bir aşama, bir öncekinin sonuçlarına dayanarak sıralı bir biçimde ilerler. Bu yapılandırılmış yaklaşım, projenin öngörülebilir ve yönetilebilir olmasını sağlar. Özellikle, gereksinimlerin projenin başında açıkça belirlendiği ve belirsizliklerin minimal olduğu durumlarda, bu modelin uygulanması idealdir. Geleneksel model, her aşamanın sonunda elde edilen çıktıların, sonraki aşamaların temelini oluşturduğu kesin ve katı bir yapı sunar.

Fedorova, Moiseeva ve Poddubnaya (2018) tarafından geliştirilen araştırma, yazılım geliştirme sürecini daha da standartlaştırmayı hedefler. Bu yaklaşım, sürecin daha verimli ve etkili bir şekilde yönetilmesini sağlayarak yazılım kalitesini artırır. Fedorova, Moiseeva ve Poddubnaya'nın önerdiği metodoloji, geliştiricilerin ve müşterilerin karşılaşabileceği riskleri minimize ederken, yazılımın oluşturulma süresini ve maliyetlerini düşürmeyi amaçlar. Standartlaşma, süreç içerisindeki belirsizlikleri azaltır ve tüm paydaşların beklentilerinin daha net bir şekilde yönetilmesine olanak tanır. Böylece, geliştirme süreci boyunca ortaya çıkabilecek sorunlara karşı daha hazırlıklı bir yaklaşım sergilenir. Sonuç olarak, Fedorova, Moiseeva ve Poddubnaya'nın önerdiği metodoloji, yazılım geliştirme sürecinin organizasyonunu kolaylaştırarak hem geliştirici hem de müşteri için önemli avantajlar sunar. Yazılım projelerinde başarının

anahtarlarından biri olan süreç yönetimi, bu tür yenilikçi yaklaşımlar sayesinde daha da güçlenmektedir. Fedorova, Moiseeva ve Poddubnaya'nın önerdiği metodoloji, projelerin daha hızlı ve ekonomik bir şekilde tamamlanmasını sağlayarak, yazılım endüstrisindeki rekabet avantajını artırır. Aynı zamanda, yazılımın son kullanıcıya sunulduğu andaki kalitesini ve performansını da önemli ölçüde yükseltir. Bu nedenle, bu tür standartlaştırılmış yaklaşımların, özellikle büyük ve karmaşık yazılım projelerinde tercih edilmesi yararlı olacaktır.

### 2.1.1. Şelale (Waterfall) Modeli

Şelale modeli (Waterfall model), yazılım geliştirme süreçlerinde kullanılan geleneksel ve yapılandırılmış bir yaklaşımdır. 1970'lerin başında popülerleşen bu model, projenin net ve değişmez gereksinimlerle başladığı, büyük ve karmaşık olmayan yazılım projeleri için idealdir. Model, projenin başından sonuna kadar sıralı bir süreç izler; bu da her bir aşamanın önceki aşamanın tamamlanmasına bağlı olarak başlatıldığı anlamına gelir. Temel aşamaları sistem gereksinimlerinin belirlenmesi, yazılım tasarımı, uygulama, test, entegrasyon ve bakım olarak sıralanabilir. Şelale modeli, projenin başında tüm gereksinimlerin net bir şekilde anlaşılmasını gerektirir ve projenin ilerleyişinde az veya hiç değişiklik beklenmez.

Bassil (2012) tarafından yapılan araştırmalarda, Şelale modelinin, öngörülebilir ve az değişkenlik gösteren projelerde oldukça etkili olduğu belirtilmiştir. Bu model, başlangıçta belirlenen gereksinimlerin değişmediği ve projenin kapsamının genişlemeyeceği durumlarda, planlanan süre ve bütçe dahilinde başarılı yazılım teslimatını mümkün kılar. Ancak, gereksinimlerde meydana gelebilecek değişiklikler veya eklemeler Şelale modelinde zorluklara neden olabilir. Çünkü modelin doğası gereği esnek değildir ve geriye dönük değişikliklere izin vermez.

Hardyanto ve arkadaşları tarafından 2017 yılında yapılan bir çalışmada, Şelale modelinin MVC (Model-View-Controller) mimarisi ile nasıl genişletilir bileceği incelenmiştir. Bu araştırma, Şelale modelinin esnekliğini artırmak ve modern yazılım geliştirme ihtiyaçlarına uyum sağlamak amacıyla yapılmıştır. MVC mimarisi, uygulama içindeki veri (Model), kullanıcı arayüzü (View) ve uygulama mantığı (Controller) arasındaki işlevselliği ayrıştırarak yazılımın daha modüler ve yönetilebilir olmasını sağlar. Hardyanto'nun çalışması, bu üç bileşeni Şelale sürecine entegre ederek, modelin hem yapılandırılmış hem de adaptif özellikler taşımasını amaçlamaktadır.

Bu genişletme, özellikle veri değişikliklerinden kaynaklanan gecikmeleri azaltma ve yazılımın güvenlik yönlerini güçlendirme konularında önemli katkılar sağlar. MVC, veri güncellemelerinin ve kullanıcı arayüzü değişikliklerinin, uygulamanın diğer bölümlerine minimal etkiyle hızlı bir şekilde yapılmasına olanak tanır. Bu da, yazılımın bakım sürecini kolaylaştırır ve güncellemelerin hızlı bir şekilde uygulanmasını sağlar. Ayrıca, MVC mimarisi sayesinde, güvenlik açıklarının izole edilmesi ve sistematik bir şekilde ele alınması mümkün hale gelir.

Hardyanto ve ekibinin çalışması, Şelale modelinin modern yazılım geliştirme pratikleri ile nasıl uyumlu hale getirebileceğini göstermektedir. Bu yaklaşım, Şelale modelinin katı ve sıralı yapısını korurken, aynı zamanda MVC mimarisi aracılığıyla esneklik ve adaptasyon yeteneğini artırmaktadır. Bu genişletilmiş model, yazılım geliştirme sürecinin hem performansını hem de güvenilirliğini artırarak, değişen teknoloji ve iş ihtiyaçlarına daha iyi cevap verebilmektedir.

### 2.1.2. **V-Modeli**

V-Modeli, yazılım geliştirme süreçlerinde kullanılan ve Şelale modelinin bir varyasyonu olarak kabul edilen, sistemli bir test sürecini merkezine alan bir yaklaşımdır. Bu model, geliştirme sürecinin her aşamasında karşılık gelen bir test aşamasının yer almasını öngörür. Böylece geliştirme ve test süreçleri paralel ilerler. Geliştirme aşamaları modelin "sol bacağı" boyunca yukarıdan aşağıya doğru sıralanırken, test aşamaları "sağ bacak" üzerinde aşağıdan yukarıya doğru izlenir. Bu yapısal özellik, V-Modeli'nin en belirgin karakteristiğidir ve geliştirme sürecinin her safhasında derhal bir test sürecinin planlanmasını sağlar.

Mateen, Tabassum ve Rehan (2017) tarafından yapılan araştırmalar, V-Modeli'nin erken hata tespiti konusunda özellikle etkili olduğunu vurgular. Model, geliştirme sürecinin ilerleyişine paralel olarak test süreçlerini entegre ettiği için, hataların oluştuğu andan itibaren hızla tespit edilmesine ve müdahale edilmesine olanak tanır. Bu özellik, yazılımın genel kalitesini artırır ve projenin ilerleyişinde karşılaşılabilecek zaman kayıplarını ve maliyet artışlarını önemli ölçüde azaltır. Dolayısıyla, V-Modeli, özellikle hata kritikliği yüksek projelerde tercih edilen bir model haline gelmiştir.

V-Modeli, yazılım geliştirme süreçlerinde sistematik bir test entegrasyonu sağlayarak projelerin daha kontrollü ve yönetilebilir olmasına katkıda bulunur. Geliştirme ve test süreçlerinin birbiriyle bu kadar yakın ve sistemli bir şekilde entegre edilmesi, projenin

başlangıcından sonuna kadar yüksek bir kalite standardının korunmasını mümkün kılar. Bu model, özellikle değişkenlik göstermeyen ve açık gereksinimlerle belirlenen projelerde, risk yönetimi ve kalite kontrol süreçlerini güçlendirerek yazılım geliştirme alanında değerli bir araç olarak öne çıkar.

### 2.1.3. **Prototipleme Modeli**

Prototipleme modeli, kullanıcı gereksinimlerini daha iyi anlamak ve bu gereksinimlerin tam olarak karşılandığından emin olmak için kullanılan bir yazılım geliştirme yaklaşımıdır. Bu model, özellikle gereksinimlerin belirsiz veya tam olarak anlaşılmadığı durumlarda kullanışlıdır. Prototipleme süreci, bir prototipin (yani, sistemin işlevsel ancak eksik bir versiyonunun) oluşturulmasını içerir. Bu prototip, kullanıcılar tarafından test edilir ve geri bildirim sağlanır. Bu geri bildirimler, sistemin nihai versiyonunu geliştirmek için kullanılır.

Budde, Kautz, Kuhlenkamp ve Zullighoven (1992) tarafından yapılan araştırmalar, prototipleme modelinin kullanıcı gereksinimlerini daha iyi anlamak ve kullanıcı memnuniyetini artırmak için etkili bir yöntem olduğunu belirtmektedir. Prototipleme, kullanıcıların sistemin erken aşamalarında nasıl çalıştığını görmelerine ve gereksinimlerini daha net bir şekilde ifade etmelerine olanak tanır. Bu süreç, son ürünün kullanıcı gereksinimlerine daha uygun olmasını sağlar ve projede değişiklik yapılması gerektiğinde daha esnek bir yaklaşım sunar.

### 2.1.4. **Spiral Modeli**

Spiral modeli, risk analizi ve risk yönetimini merkeze alan, iteratif ve artımlı bir yazılım geliştirme modelidir. 1986 yılında Barry Boehm tarafından önerilen bu model, yazılım geliştirme sürecinin her aşamasında risklerin belirlenmesini ve yönetilmesini sağlar. Spiral model, her bir döngüsünde (spiral) planlama, risk analizi, mühendislik ve değerlendirme adımlarını içerir. Bu model, projelerin karmaşıklığına ve risk düzeyine bağlı olarak uyarlanabilir ve esneklik sağlar.

Boehm (1988) tarafından yapılan çalışmalar, spiral modelin özellikle büyük ve karmaşık projelerde etkili olduğunu göstermektedir. Bu model, projenin her aşamasında riskleri değerlendirdiği için, potansiyel sorunların erken tespit edilmesine ve projenin ilerleyen aşamalarında ciddi sorunların önlenmesine olanak tanır. Spiral model, sürekli olarak

proje hedeflerini ve gereksinimlerini yeniden değerlendirerek, proje yöneticilerine daha bilinçli kararlar almalarını sağlar.

### 2.1.5. Evrimsel Model

Evrimsel model, yazılımın aşamalı olarak geliştirilmesini ve her aşamada kullanıcı geri bildirimine dayalı olarak iyileştirilmesini sağlayan bir yazılım geliştirme yaklaşımıdır. Bu model, kullanıcı gereksinimlerinin zamanla değişebileceği ve bu değişikliklerin yazılım geliştirme sürecine yansıtılmasının önemli olduğu durumlarda kullanışlıdır. Evrimsel model, yazılımın her bir versiyonunun kullanıcı gereksinimlerine daha iyi uyum sağlamasını amaçlar.

Larman ve Basili (2003) tarafından yapılan araştırmalar, evrimsel modelin kullanıcı memnuniyetini artırma ve yazılımın kalitesini iyileştirme konularında etkili olduğunu göstermektedir. Bu model, yazılımın her bir versiyonunun kullanıcı gereksinimlerine dayalı olarak iyileştirilmesini ve geliştirilmesini sağlar. Bu süreç, yazılımın daha esnek ve kullanıcı dostu olmasını sağlar.

### 2.1.6. Geleneksel Yöntemlerin Sınırlılıkları ve Eleştirileri

Geleneksel yazılım geliştirme modelleri, özellikle planlama ve iş akışı süreçlerinde büyük ölçüde sıralı ve yapılandırılmış yaklaşımlar sunar. Bu modeller, projenin başlangıcında tüm gereksinimlerin detaylı bir şekilde belirlenmesini ve projenin bu gereksinimlere sıkı sıkıya bağlı kalarak ilerlemesini öngörür. Kumar ve Bhatia (2014) tarafından yapılan çalışmada belirtildiği üzere, bu tür bir yaklaşım, değişken gereksinimlerin ve teknolojik değişikliklerin hızlı adaptasyonunu zorlaştırır. Geleneksel modellerde, projede herhangi bir değişiklik gerektiğinde, bu genellikle kapsamlı yeniden planlamayı ve mevcut iş akışının gözden geçirilmesini gerektirir. Bu durum, projenin esnekliğini önemli ölçüde sınırlar ve dinamik pazar koşullarına hızlı bir şekilde adapte olma yeteneğini azaltır.

Bu modellerin bir diğer önemli kısıtlılığı, müşteri gereksinimlerindeki değişikliklere yavaş yanıt verebilmeleridir. Geleneksel yazılım geliştirme süreçleri, projenin başında müşterilerden alınan gereksinimlere dayalı olarak ilerler. Ancak projenin geliştirilmesi sırasında bu gereksinimler değişebilir. Geleneksel modeller, bu tür değişiklikleri kabul etme ve entegre etme konusunda genellikle yetersiz kalmaktadır. Bu, müşteri

memnuniyetini ve projenin pazar başarısını olumsuz etkileyebilir. Çünkü müşteri ihtiyaçlarına uygun çözümler üretilmesi gecikebilir veya yetersiz kalabilir.

Son olarak, geleneksel yazılım geliştirme yöntemleri, nihai ürün teslim edilmeden önce son kullanıcıların geri bildirimlerini entegre etme konusunda da yetersiz kalabilirler. Bu yaklaşımlar, genellikle ürünün son test aşamalarına kadar kullanıcı geri bildirimlerini aktif olarak talep etmezler. Bu durum, kullanıcı deneyimini iyileştirecek ve ürünün kullanıcı ihtiyaçlarına daha iyi uyum sağlamasını sağlayacak değerli içgörülerin göz ardı edilmesine neden olabilir. Kumar ve Bhatia'nın belirttiği gibi, son kullanıcı geri bildirimlerinin erken ve sürekli bir şekilde entegrasyonu, yazılımın başarısında kritik bir rol oynar ve bu, geleneksel modellerin çoğu tarafından yeterince desteklenmez. Bu eksiklikler, yazılım projelerinde yenilikçi ve kullanıcı odaklı çözümler geliştirmenin önündeki engellerden bazılarını oluşturur. (Kumar & Bhatia, 2014).

## 2.2. YAZILIM GELİŞTİRMEDE ÇEVİK YÖNTEMLER

Çevik yöntemler, yazılım geliştirme süreçlerinde esneklik ve hızlı adaptasyonu önceliklendiren iteratif bir yaklaşım sunar. Bu metodolojiler, 2001 yılında yayımlanan Çevik Manifesto ile popülerlik kazanmıştır ve geliştirme sürecini kısa iterasyonlar halinde, sürekli geliştirme ve düzenleme fırsatı sağlayarak yeniden şekillendirir. 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 kılar. Ayrıca çevik metodolojiler, planlamayı ve teslimatları daha yönetilebilir ve esnek hale getirerek, geliştirme sürecinin genel verimliliğini artırır.

Çevik metodolojilerin temelinde, takım çalışması ve sürekli iletişim yatar. Bu yaklaşımlar, proje takımlarının düzenli aralıklarla bir araya gelmesini ve ilerleme hakkında sürekli geri bildirimde bulunmasını teşvik eder. Takım üyeleri arasındaki açık iletişim, ortak hedeflere ulaşma konusunda koordinasyonu ve işbirliğini kolaylaştırır. Çevik yöntemler, müşteri geri bildirimlerini sürecin merkezine koyar. Müşteriler, geliştirme sürecinin her aşamasında aktif bir rol oynarlar ve düzenli olarak sağladıkları geri bildirimlerle ürünün şekillendirilmesine doğrudan katkıda bulunurlar. Bu sürekli etkileşim, son ürünün müşteri beklentileri ve ihtiyaçları ile daha uyumlu olmasını sağlar.

Çevik metodolojiler, yazılım geliştirme sürecinin hızını ve etkinliğini artırmak için tasarlanmıştır. Sürekli değişen gereksinimlere hızlı bir şekilde adapte olmayı sağlayan bu yaklaşım, yazılım geliştirme ekiplerine, projeleri üzerinde daha iyi kontrol sahibi olma imkânı tanırken, 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 (Kate, Bhalerao, & Sharma, 2023).

### 2.2.1. Çevik Metodolojinin Temelleri

Çevik metodoloji, yazılım geliştirme süreçlerinde esneklik ve hızlı adaptasyonu vurgulayan modern bir yaklaşım olarak ön plana çıkar. Bu metodoloji, özellikle pazar koşullarının ve müşteri ihtiyaçlarının hızla değişebildiği projelerde etkili bir şekilde kullanılır. Çevik yaklaşımın temelinde, projenin gereksinimlerine ve hedeflerine uyum sağlayabilmek için sürekli olarak geliştirme ve değerlendirme süreçlerinin gerçekleştirilmesi yatar. Bu süreç, geliştirme takımlarının değişen ihtiyaçlara hızla yanıt verebilmesini, böylece pazardaki değişimlere ve teknolojik yeniliklere dinamik bir şekilde adapte olabilmesini sağlar. Çevik metodoloji, bu esnek yapıyı desteklemek için kısa süreli planlamalar ve düzenli değerlendirmeler içeren iteratif döngüler kullanır.

Çevik metodolojiler, küçük, öz-yönetimli takımların kullanımını teşvik eder. Bu takımlar, projenin her aşamasında aktif rol alır ve kendi iş süreçlerini kendileri yönetirler. Öz-yönetimli takımlar, karar verme süreçlerinde daha hızlı hareket edebilir ve bu da genel proje sürecinin hızını artırır. Takımların kendi içinde daha etkin bir iletişim kurmaları ve sorumlulukları paylaşmaları, iş akışını optimize eder ve projenin gereksinimlerine daha iyi uyum sağlar. Çevik metodoloji içerisinde takımlar, sürekli geliştirme ve sürekli teslimatı hedefleyen kısa süreli iterasyonlar boyunca ilerler. Bu iterasyonlar, sürekli müşteri geri bildirimi ve sürekli iyileştirme imkanı sunarak, ürünün kalitesinin ve müşteri memnuniyetinin artırılmasına katkıda bulunur.

Çevik metodoloji, yazılım geliştirme sürecine dinamizm ve adaptasyon yeteneği kazandırır. Müşteri odaklı bu yaklaşım, değişen ihtiyaçlara hızlı bir şekilde yanıt verilmesini ve proje sonuçlarının sürekli olarak iyileştirilmesini sağlar. Kısa iterasyonlar ve öz-yönetimli takımlar sayesinde, çevik metodoloji kullanan projeler, geleneksel

yazılım geliştirme metodolojilerine göre daha yüksek başarı oranları ve daha iyi müşteri memnuniyeti sunar. Bu avantajlar, çevik metodolojinin yazılım geliştirme sektöründe giderek daha fazla benimsenmesine yol açmaktadır. Böylece projeler daha esnek, verimli ve yenilikçi bir şekilde ilerlemektedir.

### 2.2.2. Scrum

Scrum, çevik metodolojiler içinde özellikle kompleks yazılım projeleri için tercih edilen öncü bir yaklaşımdır. Bu metodoloji, belirgin roller (Scrum Master, Product Owner ve Development Team) ve özel süreçler ile karakterize edilir. Scrum'ın temelini, kısa süreç döngüleri olan sprintler oluşturur. Her bir sürat toplantısı genellikle birkaç hafta süren yoğun çalışma periyodlarıdır ve belirlenen hedeflere ulaşmak için yoğunlaşmış bir çaba gerektirir. Günlük scrum toplantıları ise, günlük olarak düzenlenen kısa toplantılardır ve takımın o gün üzerinde çalışacağı öncelikleri belirlemesine, bir önceki gün karşılaşılan sorunları çözmesine ve işbirliği içinde ilerlemesine olanak tanır. Trihardianingsih ve arkadaşları tarafından 2023 yılında yapılan araştırma, Scrum metodolojisinin, özellikle karmaşık yazılım projelerinde, süreçlerin etkili bir şekilde yönetilmesini sağladığını ve projenin başarılı bir şekilde tamamlanmasına katkıda bulunduğunu ortaya koymuştur.

Scrum metodolojisinin en önemli özelliklerinden biri, rol tanımlarının açıkça yapılmış olmasıdır. Scrum yöneticisi, takımın Scrum süreçlerini etkili bir şekilde uygulamasını sağlamakla sorumludur ve takımın önündeki engelleri kaldırarak iş akışını optimize etmeye çalışır. Ürün sahibi ise, ürünün vizyonunu ve projenin iş hedeflerini tanımlar ve ürün geliştirmenin önceliklerini belirler. Geliştirme Takımı, ürünün gerçekleştirilmesi için gerekli yazılımı tasarlar ve geliştirir. Bu roller, Scrum içinde açıkça tanımlanmış sorumluluklar ve yetkilerle donatılmıştır. Bu da her bir üyenin projeye katkıda bulunmasını ve işbirliği içinde çalışmasını kolaylaştırır. Scrum metodolojisi, esneklik ve hızlı adaptasyonu destekleyen dinamik bir yapı sunar. Sprintler ve sürekli geri bildirim döngüleri sayesinde, ekip projenin her aşamasında değişen gereksinimlere hızla yanıt verebilir ve müşteri ihtiyaçlarına uyum sağlayabilir. Bu süreç, projenin sürekli olarak güncellenmesini ve müşteri memnuniyetinin maksimize edilmesini sağlar. Trihardianingsih ve arkadaşlarının çalışması, Scrum'un bu özelliklerinin, karmaşık yazılım geliştirme projelerinde başarılı sonuçlar elde etmek için nasıl kritik öneme sahip olduğunu vurgulamıştır. Bu nedenle, Scrum, modern yazılım geliştirme ortamında çevik

ve etkin çözümler arayan organizasyonlar için vazgeçilmez bir araç haline gelmiştir (Trihardianingsih et al., 2023).

### 2.2.3. **Extreme Programming (XP)**

Olağanüstü Programlama, yazılım geliştirme süreçlerinde yüksek kalite ve müşteri memnuniyetini maksimize etmek amacıyla tasarlanmış bir çevik metodolojidir. XP, dinamik pazar koşulları ve sürekli değişen müşteri gereksinimleri karşısında esnek ve hızlı bir adaptasyon sağlayarak dikkat çekmektedir. Bu metodolojinin temel uygulamaları arasında sık sık versiyon yayınlama, sürekli kod incelemesi, çiftler halinde programlama ve müşteri geri bildiriminin sürekli entegrasyonu bulunur. Sık sık versiyon yayınlama, yazılımın kısa döngüler halinde, düzenli aralıklarla piyasaya sürülmesini içerir. Bu sayede müşteriler yazılımı erken aşamada deneyimleyebilir ve değerli geri bildirimler sağlayabilir. Sürekli kod incelemesi ve çiftler halinde programlama, yazılımın kalitesini artırarak hataların erken tespit edilmesini ve düzeltilmesini sağlar.

XP'nin bir diğer önemli özelliği, müşteri geri bildirimlerini sürecin merkezine koymasıdır. Müşteriler, geliştirme sürecinin her aşamasında aktif bir rol alır ve düzenli olarak sağladıkları geri bildirimlerle ürünün şekillendirilmesine doğrudan katkıda bulunurlar. Bu sürekli etkileşim, son ürünün müşteri beklentileri ve ihtiyaçları ile daha uyumlu olmasını sağlar. Müşteri geri bildirimlerinin sürekli entegrasyonu, XP metodolojisinin temel taşlarından biridir ve bu yaklaşım, projenin başarısını doğrudan etkileyen müşteri tatminini önemli ölçüde artırır. Trihardianingsih ve arkadaşlarının 2023 tarihli çalışması, XP'nin bu uygulamalarının, müşteri gereksinimlerine hızla uyum sağlama ve yazılım kalitesini sürdürme konusunda ne kadar etkili olduğunu göstermektedir.

XP, çevik yazılım geliştirme metodolojileri arasında, özellikle değişken ve talepkar müşteri gereksinimleriyle başa çıkmada öne çıkar. XP, müşteri odaklı yaklaşımı ve sürekli geliştirme süreci ile yazılım projelerinde esnekliği ve adaptasyon kabiliyetini artırır. Bu metodoloji, müşteri memnuniyetini ve yazılım kalitesini merkeze alarak, dinamik ve rekabetçi yazılım geliştirme ortamlarında başarıya ulaşmak için güçlü bir araç sunar. Trihardianingsih ve ekibinin araştırmaları, XP'nin yazılım geliştirme süreçlerindeki etkinliğini ve projelerdeki pozitif etkisini vurgulamıştır. Bu da metodolojinin geniş çapta benimsenmesine yol açmıştır (Trihardianingsih et al., 2023).

### 2.2.4. **Kanban**

Kanban, iş akışını görselleştiren ve süreçler arasındaki geçişi düzene sokarak iş yükünü optimize etmek için tasarlanmış bir çevik yöntemdir. Bu metodoloji, özellikle sürekli iyileştirme ve iş akışının etkin yönetimi üzerine odaklanır. Kanban, işleri belirli aşamalara bölerek ve her bir aşamadaki işleri görsel olarak temsil eden Kanban tahtaları kullanarak çalışır. Bu görselleştirme, takım üyelerinin her aşamadaki iş yükünü net bir şekilde görmelerini sağlar ve sürecin tamamında neyin, ne zaman ve nasıl yapıldığını açıkça ortaya koyar.

Trihardianingsih ve arkadaşları tarafından 2023 yılında yapılan araştırma, Kanban'ın mevcut süreçleri kesintiye uğratmadan sürekli iyileştirmeler yapılmasını kolaylaştırdığını ve bu sayede iş akışının daha verimli hale geldiğini belirtmektedir.

Kanban metodolojisinin temel avantajlarından biri, iş yükünü dengelemeye ve süreç içindeki tıkanıklıkları azaltmaya yönelik olmasıdır. Takım üyeleri, Kanban tahtasındaki görsel öğeler aracılığıyla iş akışının durumunu anlık olarak takip edebilir ve gerektiğinde ayarlamalar yapabilir. Bu sistem, süreçler arası geçişleri düzenleyerek, her bir aşamada iş yükünün aşırı toplanmasını önler ve böylece süreçlerin daha düzgün ve hızlı işlemesini sağlar. Ayrıca, Kanban, takım üyelerine hangi işlerin öncelikli olduğunu ve hangi kaynakların nerede kullanılması gerektiğini belirlemede yardımcı olur, bu da genel verimliliği artırır.

Trihardianingsih ve arkadaşlarının çalışmaları, Kanban'ın iş verimliliğini artırma, süreç tıkanıklıklarını azaltma ve sürekli iyileştirme sağlama konularında ne kadar etkili olduğunu göstermektedir. Bu özellikler, Kanban'ı dinamik ve sürekli değişen iş ortamlarında değerli bir araç haline getirmiştir. Böylece organizasyonlar projeleri daha etkin ve verimli bir şekilde yönetebilmektedirler (Trihardianingsih et al., 2023).

Türkçe kaynaklardan da benzer görüşler ve bulgular elde edilmiştir. Örneğin, Serdar Kuzuloğlu, Kanban sisteminin iş süreçlerinin daha verimli yönetilmesi için etkili bir yöntem olduğunu ve özellikle yazılım geliştirme ve üretim süreçlerinde büyük avantaj sağladığını belirtmiştir (Kuzuloğlu, 2016).

Ercan Bozkurt ise, Kanban'ın iş akışını görselleştirerek iş süreçlerini optimize ettiğini ve sürekli iyileştirme sağladığını ifade etmiştir (Bozkurt, 2018). Bozkurt'un çalışmaları,

Kanban'ın iş süreçlerinde nasıl etkin kullanılabileceğini ve verimlilik artışını detaylı bir şekilde ele almıştır.

Ahmet Aksoy, Kanban'ın projelerin daha verimli yönetilmesi için kullanılan bir iş yönetim metodolojisi olduğunu ve iş akışının görselleştirilmesi ile darboğazların tespit edilmesinin önemini vurgulamaktadır (Aksoy, 2017).

Son olarak, Murat Yıldırım, Kanban'ın iş süreçlerinin yönetiminde etkin bir araç olduğunu ve özellikle üretim ve yazılım geliştirme alanlarında iş akışını düzenleyerek sürekli iyileştirme sağladığını belirtmiştir (Yıldırım, 2019).

## 2.3. Çevik Yöntemlerin Avantajları ve Zorlukları

Çevik metodolojiler, yazılım geliştirme sürecinde önemli avantajlar sunmaktadır. Bunlar arasında müşteri memnuniyetinin artırılması, süreç üzerinde daha fazla kontrol sağlanması ve ürünlerin hızlı bir şekilde piyasaya sürülmesi yer alır. Bu avantajlar, projelerin dinamik ve değişken gereksinimlere hızlı bir şekilde adapte olmasına olanak tanır ve böylece müşteri ihtiyaçlarına daha uygun çözümler sunulabilir. Ancak, Khairi, Kamaruddin ve Widyarto (2016) tarafından yapılan çalışmalarda, çevik metodolojilerin uygulanmasının bazı zorluklar içerebileceğini gösterilmiştir. Özellikle, organizasyonel kültürde gerekli değişikliklerin yapılması ve ekip üyelerinin bu yeni çalışma biçimlerine adapte olması, çevik metodolojilerin başarılı bir şekilde uygulanabilmesi için önemli engeller arasında yer alır.

Çevik metodolojilerin etkin bir şekilde uygulanması, öncelikle açık iletişim ve takım işbirliğine bağlıdır. Ekip üyeleri arasındaki sürekli ve açık iletişim, projenin şeffaf bir şekilde yönetilmesini ve herkesin projedeki rol ve sorumluluklarını net bir şekilde anlamasını sağlar. Ayrıca, takım işbirliği, çevik ortamlarda öncelikli hedeflerin belirlenmesi ve sürekli değişen gereksinimlere hızlı tepki verilmesi açısından kritik öneme sahiptir. Bu bağlamda, her bir takım üyesinin, ortak hedeflere ulaşmak için işbirliği yapma ve uyum sağlama yeteneği, projenin genel başarısını doğrudan etkileyebilir.

Çevik metodolojilerin başarıyla uygulanması için sürekli öğrenme ve adaptasyon gereklidir. Çevik süreçler, ekip üyelerinin sürekli olarak yeni beceriler öğrenmelerini, değişen teknoloji ve pazar koşullarına adapte olmalarını gerektirir. Bu sürekli gelişim, projelerin en güncel ve etkili yöntemlerle yürütülmesini sağlar ve böylece süreçlerin

iyileştirilmesine katkıda bulunur. Khairi, Kamaruddin ve Widyarto'nun çalışması, çevik metodolojilerin başarılı bir şekilde uygulanabilmesi için, takımların bu sürekli öğrenme kültürünü benimsemesi ve adaptasyon süreçlerine açık olmalarının ne kadar önemli olduğunu vurgulamaktadır. Bu durum, çevik metodolojilerin, sadece teknik bir yaklaşım olmanın ötesinde, aynı zamanda bir takım kültürü ve felsefesi olarak görülmesi gerektiğini gösterir (Khairi, Kamaruddin, & Widyarto, 2016).

## 2.4. ÇEVİK YÖNTEMDEKİ ROLLER VE SORUMLULUKLARI

Çevik metodolojiler, rollerin net tanımlandığı ve her bir rolün belirgin sorumlulukları olduğu bir yapı sunar. Bu roller Ürün Sahibi, Scrum ustası, Geliştirme Takımı ve Diğer Paydaşları içerir.

### 2.4.1. Ürün Sahibi (Product Owner)

Ürün Sahibi rolü, çevik yazılım geliştirme süreçlerinde kritik bir öneme sahiptir. Bu rol, ürün vizyonunun belirlenmesi, gereksinimlerin önceliklendirilmesi ve ürünün iş değerinin maksimize edilmesinden sorumludur. Ürün Sahibi, aynı zamanda müşteri ve kullanıcı ihtiyaçlarını temsil eder. Bu da onları projenin merkezine koyar. Bu rolün başarılı bir şekilde yerine getirilmesi, ürünün piyasadaki başarısını doğrudan etkileyebilir. Çünkü Ürün Sahibi, müşteri ihtiyaçlarını ve pazar taleplerini doğru bir şekilde yorumlayarak ürün stratejisini buna göre şekillendirir. Ürün Sahibi, aynı zamanda takım ile müşteriler arasında bir köprü görevi görür ve projenin hedeflerinin ve gereksinimlerinin tüm takım üyeleri tarafından anlaşılmasını sağlar.

Bass (2013) tarafından yapılan çalışmada, küresel ölçekteki projelerde Ürün Sahibi rollerinin nasıl uyarlandığı detaylı bir şekilde incelenmiştir. Küresel projelerde, Ürün Sahibi rolünün, farklı coğrafi konumlardaki takımlar ve müşteriler arasında etkili bir iletişim ve koordinasyon sağlaması gerekmektedir. Bu tür projelerde, kültürel ve operasyonel farklılıklar nedeniyle ek zorluklar ortaya çıkabilir ve Ürün Sahibi'nin bu farklılıkları yönetme becerisi projenin başarısını büyük ölçüde etkileyebilir. Bass'ın çalışmasında, Ürün Sahibinin, farklı bölgelerden gelen gereksinimleri nasıl önceliklendirdiği ve teknik yönetim sağladığı konusunda önemli içgörüler sunulmuştur. Bu, ürün geliştirme sürecinde karşılaşılan zorlukların üstesinden gelinmesinde ve tüm pazar segmentlerinin ihtiyaçlarının karşılanmasında hayati bir rol oynar.

Ürün Sahibi'nin rolü, küresel ölçekte yürütülen projelerde özellikle karmaşık bir hal alabilir. Etkili bir Ürün Sahibi, gereksinimleri uygun bir şekilde önceliklendirmeli ve müşteri ihtiyaçlarını doğru bir şekilde yansıtabilmelidir. Bass çalışmasında, bu rolün, projenin genel yönetim stratejisi ve teknik koordinasyonla nasıl bütünleştiğini ortaya koymuş ve bu entegrasyonun, projenin başarılı yürütülmesindeki önemini vurgulamıştır. Bu bağlamda, Ürün Sahibi'nin yetkinlikleri, projenin etkin yönetimi ve sonuçta elde edilen ürünün başarısı için kritik önem taşır. Bass'ın çalışması, ürün yönetimi pratiğinde Ürün Sahibi rolünün nasıl optimize edilebileceğine dair değerli bilgiler sağlar ve bu rolün, global çapta yürütülen yazılım geliştirme projelerindeki etkilerini anlamada yardımcı olur.

### 2.4.2. Scrum Yöneticisi

Scrum yöneticisi, çevik yazılım geliştirme süreçlerinde, takımın çevik prensiplere uygun ve verimli bir şekilde çalışmasını sağlayan temel bir roldür. Bu rol, takımın karşılaştığı engelleri ortadan kaldırmak, süreçleri optimize etmek ve çevik uygulamaların doğru bir şekilde yerine getirilmesini garanti etmekle sorumlu olan bir kişidir. Scrum yöneticisi aynı zamanda, takım ile diğer proje paydaşları arasındaki iletişimi kolaylaştırarak, proje hedeflerine ulaşılmasında koordinasyon ve işbirliğini destekler. Bu rol, takımın iç dinamiklerini anlamak ve her bir üyenin potansiyelini en iyi şekilde kullanarak toplam performansı artırmak için de kritik öneme sahiptir. Scrum yöneticisi, sürekli olarak takımın çevik metodolojileri nasıl daha etkin uygulayabileceği üzerine çalışır ve bu süreçte sürekli iyileştirme fırsatları arar.

Shastri, Hoda ve Amor (2017) çalışmalarında, çevik projelerdeki yöneticilerin üstlendiği çeşitli rolleri detaylı bir şekilde analiz etmiştir. Bu çalışmada, yöneticilerin mentor, koordinatör, müzakereci ve süreç uyarlayıcısı gibi rolleri nasıl üstlendikleri üzerinde durulmuştur. Mentor olarak yöneticiler, takım üyelerine rehberlik etmekte ve onların profesyonel gelişimini desteklemektedir. Koordinatör olarak, projenin farklı bileşenleri arasında uyum sağlamak ve iş akışını düzenlemekle sorumludurlar. Müzakereci rolünde ise, çevik takımlar ve diğer paydaşlar arasında etkili iletişim kurarak, tüm tarafların ihtiyaç ve beklentilerini dengelerler. Süreç uyarlayıcısı olarak, projenin gereksinimlerine göre çevik süreçleri uyarlar ve optimize ederler. Bu roller, çevik projelerde yöneticilerin çok yönlü ve esnek olmalarını gerektirir. Çünkü bu rollerin her biri projenin başarısında belirleyici bir etkiye sahiptir.

Scrum yöneticisinin rolü, çevik projelerde merkezi bir öneme sahiptir ve bu rolün etkin bir şekilde yerine getirilmesi, projenin genel başarısı için kritik önem taşır. Shastri, Hoda ve Amor (2017), Scrum Master'ın ve diğer çevik yöneticilerin projelerde üstlendikleri çeşitli rollerin ne kadar hayati olduğunu vurgulamıştır. Bu rollerin doğru bir şekilde uygulanması, takımın yüksek performans göstermesini, engellerin etkin bir şekilde aşılmasını ve projenin hedeflerine ulaşmasını sağlar. Shastri, Hoda ve Amor (2017)'un çalışması ayrıca, çevik yöneticilerin adaptasyon yeteneklerinin ve esnek roller üstlenmelerinin, çevik metodolojilerin başarılı uygulanması için temel olduğunu göstermektedir.

### 2.4.3. Geliştirme Takımı

Geliştirme Takımı, çevik yazılım geliştirme metodolojilerinde merkezi bir rol oynar ve bu takım ürünün tasarımı, geliştirilmesi ve test edilmesinden sorumlu bireylerden oluşur. Bu takımlar, kendi kendini yönetme yeteneğine sahiptir. Bu da onlara proje sürecinde daha fazla esneklik ve hızlı karar alma kapasitesi sağlar. Çevik metodolojilerin temel prensiplerinden biri olan iteratif geliştirme süreci, Geliştirme Takımının sürekli olarak ürün üzerinde çalışmasını ve düzenli aralıklarla iyileştirmeler yapmasını gerektirir. Bu sürekli geliştirme yaklaşımı, ürünün kalitesini artırmanın yanı sıra, pazarın ve müşterinin değişen ihtiyaçlarına hızlı bir şekilde uyum sağlamayı mümkün kılar.

Çevik metodolojiler, takım üyeleri arasında sürekli iletişim ve etkileşim olmasını teşvik eder. Bu sürekli iletişim, Geliştirme Takımının üyelerinin birbirleriyle ve diğer proje paydaşlarıyla düzenli olarak bilgi alışverişinde bulunmalarını sağlar. Böylece projenin her aşamasında şeffaflık ve açıklık korunur. Ayrıca, sürekli geri bildirim alma olanağı, takımın ürünü piyasa ve müşteri gereksinimlerine daha iyi uyum sağlamasına yardımcı olur. Bu geri bildirimler, projenin ilerleyişinde kritik öneme sahiptir. Çünkü takımın hedefler doğrultusunda ilerlemesini ve gerekli düzeltmeleri zamanında yapmasını sağlar. Böylece Geliştirme Takımı, ürünü daha etkin bir şekilde geliştirme ve pazar gereksinimlerine uygun hale getirme şansına sahip olur.

### 2.4.4. Diğer Paydaşlar

Çevik yazılım geliştirme süreçlerinde, "Diğer Paydaşlar" kategorisi, projenin başarısı üzerinde büyük bir etkiye sahip olan çeşitli grupları kapsar. Bu gruplar arasında müşteriler, kullanıcılar, şirket yöneticileri ve proje finansörleri bulunmaktadır.

Lárusdóttir ve diğerleri tarafından 2016 yılında yapılan araştırma da, bu paydaşların çevik geliştirme sürecine katılımının, yazılım projelerindeki kullanıcı deneyimini nasıl etkilediğini incelenmiştir. Çalışma, paydaşların aktif katılımının, yazılımın son kullanıcılar tarafından nasıl algılandığı ve kullanıldığı üzerinde doğrudan bir etkisi olduğunu göstermektedir. Paydaşların süreçteki rolleri, kullanıcı ihtiyaçlarının doğru bir şekilde anlaşılmasını ve bu ihtiyaçların yazılım çözümlerine entegre edilmesini sağlamada kritik bir öneme sahiptir. Bu katılım, yazılımın kullanıcı arayüzü ve işlevselliği üzerinde önemli düzenlemeler yapılmasına olanak tanır. Böylece son ürünün kullanıcı deneyimi önemli ölçüde iyileştirilir.

Lárusdóttir ve ekibinin çalışması, çevik metodolojilerde paydaş katılımının, sadece yazılımın teknik yönlerini değil, aynı zamanda estetik ve işlevsel yönlerini de iyileştirmede ne kadar etkili olduğunu ortaya koymaktadır. Paydaşlar, yazılımın geliştirilmesi sürecinde önemli bilgiler ve geri bildirimler sağlayarak geliştirme takımına rehberlik eder. Bu etkileşim, yazılımın pazarın ve son kullanıcının beklentilerine daha uygun hale gelmesini sağlar. Ayrıca, proje ilerledikçe paydaşların sürekli katılımı, yazılımın adaptasyonunu ve evrimini teşvik eder. Böylece dinamik pazar koşullarına ve teknolojik değişikliklere hızla uyum sağlaması mümkün olur. Bu bağlamda, Lárusdóttir ve diğerlerinin çalışması, çevik yazılım geliştirme projelerinde paydaş katılımının stratejik önemini ve bu katılımın kullanıcı deneyimi üzerindeki pozitif etkilerini vurgulamaktadır (Lárusdóttir vd., 2016).

## 2.5. YAZILIMDA KALİTE VE TEST

Yazılım kalitesi ve test, yazılım geliştirme sürecinin temel unsurlarıdır ve ürün kalitesini sağlamak, hataları azaltmak ve müşteri memnuniyetini artırmak için kritik öneme sahiptir.

### 2.5.1. Kalitenin Tanımı

Yazılım kalitesi, yazılımın belirli gereksinimleri ve beklentileri ne derece karşıladığını ölçen bir kriterdir ve bu, 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. Örneğin, yüksek performans ve güvenilirlik, özellikle kritik uygulamalar için

olmazsa olmaz özelliklerdir. Çünkü bu tür uygulamalar sık sık ve uzun süreli kullanımlara maruz kalır. Kullanılabilirlik ise, yazılımın son kullanıcı tarafından ne kadar kolay ve verimli kullanılabileceğini gösterir ve bu da kullanıcı memnuniyetini doğrudan etkiler.

Zhao, Hu ve Gong'un 2021 yılında 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. Bu durum, yazılım geliştiriciler için kaliteyi sürekli olarak ön planda tutmanın ve iyileştirmeler yapmanın önemini vurgular. Kalite, sadece yazılımın mevcut durumunu değil, aynı zamanda marka imajını ve müşteri sadakatini de güçlendirir.

Yazılım kalitesi, geliştirme sürecinin her aşamasında dikkate alınması gereken bir önceliktir. Geliştiriciler, yazılım kalitesini artırmak için sürekli olarak çaba göstermeli ve kalite kontrol mekanizmalarını entegre etmelidir. Bu, yazılımın gereksinimleri daha iyi karşılamasını, hataların azaltılmasını ve son kullanıcının deneyimini iyileştirmesini sağlar. Zhao, Hu ve Gong'un çalışması, yazılım kalitesine yapılan yatırımın, uzun vadede yazılımın pazar performansını ve kullanıcı memnuniyetini nasıl olumlu yönde etkilediğini göstermektedir. Bu nedenle, kalite odaklı bir yaklaşım, yazılım geliştirme süreçlerinde merkezi bir rol oynamalıdır (Zhao, Hu, & Gong, 2021).

### 2.5.2. **Kalite Güvencesi (KG)**

Kalite güvencesi (Quality Assurance, QA), yazılım geliştirme sürecindeki kalite standartlarının korunmasını ve iyileştirilmesini amaçlayan kritik bir süreçtir. KG, yazılımın tasarımından dağıtımına kadar olan tüm aşamalarında kalitenin sistemli bir şekilde denetlenmesi ve yönetilmesi için kullanılan metodolojileri ve araçları kapsar. Malik ve Singh (2017) tarafından yapılan araştırmada belirtildiği üzere, KG süreçleri, yazılımın geliştirilmesi sırasında ortaya çıkan riskleri yönetmeye, maliyetleri optimize etmeye ve ürün kalitesini maksimize etmeye odaklanır. Bu süreçler, yazılımın hata oranını azaltmak ve nihai ürünün belirlenen kalite standartlarını karşılamasını sağlamak için tasarlanmıştır. KG 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. Risk yönetimi, olası risklerin önceden tanımlanmasını, değerlendirilmesini ve bu risklere karşı önlemlerin alınmasını içerir. Değer mühendisliği ise, maliyet-etkin çözümler üreterek yazılımın değerini artırmayı hedefler. Kalite modelleri ve araçları, KG sürecinde kullanılan metodolojileri destekleyerek, yazılımın kalite hedeflerine ulaşmasını kolaylaştırır. Bu modeller, yazılımın kalite ölçütlerini tanımlar ve bu ölçütler doğrultusunda sürekli iyileştirmeler yapılmasını sağlar. Malik ve Singh çalışmasında, kalite güvencesinin, projelerin zamanında ve bütçe dahilinde tamamlanmasına yardımcı olacak şekilde tasarlandığını vurgulamıştır.

Kalite güvencesi sürecinin başarısı, sürekli iyileştirme anlayışına bağlıdır. KG ekipleri, sürekli geri bildirim toplayarak ve yazılımın her bir sürümünü sürekli olarak değerlendirerek, sürekli olarak kaliteyi artırmaya yönelik stratejiler geliştirmelidir. Bu, yazılımın kalitesini zaman içinde sürekli olarak artıracak ve müşteri gereksinimlerini daha iyi karşılayacak çözümler sunacaktır. Malik ve Singh'in araştırması, kalite güvencesinin, yazılım endüstrisindeki firmalar için rekabet avantajı sağladığını ve kalite yönetim sistemlerinin etkin bir şekilde uygulanmasının, şirketlerin piyasadaki konumlarını güçlendirmelerine olanak tanıdığını ortaya koymaktadır (Malik & Singh, 2017).

### 2.5.3. **Yazılım Testinin Rolü**

Yazılım testi, yazılım geliştirme sürecinin ayrılmaz bir parçasıdır ve ürünün hata oranını tespit etme, giderme ve yazılımın kalite gereksinimlerini karşıladığını doğrulama işlevini görür. Bu süreç, yazılımın piyasaya sürülmeden önce belirlenen kalite standartlarına uygunluğunu sağlamak için kritik öneme sahiptir. Mateen, Zhu ve Awan (2018) tarafından yapılan araştırma, etkili bir test stratejisinin, yazılımın güvenilirliğini artırdığını ve geliştirme maliyetlerini azalttığını belirtmektedir. 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. Manuel testler, belirli senaryolar ve kullanıcı davranışları simülasyonu yoluyla yazılımın işlevselliğini değerlendirir. Otomatik testlerde yazılımın belirli bölümlerinin tekrarlanabilir ve düzenli olarak test edilmesi için özel olarak geliştirilmiş araçlar ve scriptler kullanır. 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. Manuel testler, yazılımın kullanıcı dostu olup olmadığını değerlendirmede ve karmaşık kullanıcı etkileşimlerini test etmede yararlı olabilirken, otomatik testler büyük veri setleri ile çalışırken veya sık sık kod değişikliklerini kontrol ederken daha etkilidir. Ayrıca, otomatik testler, yazılım geliştirme sürecinin erken aşamalarında hataları tespit etmek ve düzeltmek için de kullanılabilir. Bu da hata düzeltme maliyetlerinin zamanla azalmasına yardımcı olur. Yazılım testi, yazılımın kalitesini doğrulamanın ve güvenilir bir ürün sunmanın temel yollarından biridir. Mateen, Zhu ve Awan'ın çalışması, yazılım testinin, hata oranlarını azaltma ve kullanıcı memnuniyetini artırma konusundaki önemini vurgular. Etkili bir test stratejisi, yazılımın başarılı bir şekilde piyasaya sürülmesini sağlamak için gereklidir ve bu strateji, hem manuel hem de otomatik test yaklaşımlarını kapsamlı bir şekilde entegre etmelidir. Böyle bir yaklaşım, yazılım geliştirme sürecinin verimliliğini ve son ürünün kalitesini önemli ölçüde artırabilir. (Mateen, Zhu, & Awan, 2018).

### 2.5.4. **Test Stratejileri**

Yazılım test stratejileri, yazılım geliştirme sürecinin temel bir bileşenidir ve kalitenin tüm geliştirme döngüsü boyunca sürekli olarak değerlendirilmesini sağlar. Garg (2015) tarafından yapılan araştırma, test stratejilerinin yazılımın farklı gelişim aşamalarında nasıl uygulanması gerektiğini detaylandırılmıştır. Bir test stratejisi, belirli test türlerinin, hangi sıklıkta ve hangi geliştirme aşamalarında uygulanacağını tanımlar. Bu çerçeveler, test süreçlerinin sistematik ve kapsamlı bir şekilde yürütülmesini sağlar. Böylece yazılımın baştan sona kalite standartlarına uygunluğu garanti altına alınır. Test stratejileri, hata tespiti, risk yönetimi ve kalite kontrol gibi önemli işlevleri yerine getirir. Aynı zamanda yazılım projelerinin zamanında ve bütçe dahilinde tamamlanmasına olanak tanır.

Etkili bir yazılım test stratejisi, yazılımın her evresine uygun test türlerini belirler ve bu testleri uygulamanın zamanlamasını planlar. Örneğin, bir strateji, geliştirme sürecinin erken aşamalarında birim testleri, orta aşamalarında entegrasyon testleri ve son aşamalarda kabul testlerini öngörebilir. Bu yaklaşım, yazılımın her bir parçasının, sonraki aşamaya geçmeden önce yeterli kalite standartlarını karşıladığından emin olunmasını sağlar. Her test türü, yazılımın farklı yönlerini değerlendirir. Birim testleri yazılımın temel bileşenlerinin doğru çalıştığını doğrular, entegrasyon testleri bileşenler arası etkileşimleri kontrol eder ve kabul testleri yazılımın genel işlevselliğini ve kullanıcı gereksinimlerine uygunluğunu test eder.

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 edilebilir. Test stratejisinin belirlenmesinde, projenin risk profili, teknolojik karmaşıklığı ve son kullanıcı beklentileri gibi faktörler dikkate alınır. Bu stratejik yaklaşım, test süreçlerinin verimli bir şekilde yönetilmesini sağlayarak, yazılım kalitesinin sürekli olarak izlenmesini ve iyileştirilmesini mümkün kılar.

Yazılım test stratejileri, yazılım geliştirme sürecinin ayrılmaz bir parçası olarak, ürün kalitesinin güvence altına alınmasında kritik bir rol oynar. Garg (2015) çalışmasında, bu stratejilerin, yazılımın gelişim aşamalarına uygun şekilde nasıl tasarlanıp uygulanması gerektiğini vurgulamıştır. Stratejik test planlaması, hataların erken aşamalarda tespit edilmesine, düzeltilmesine ve böylece yazılım geliştirme maliyetlerinin azaltılmasına katkıda bulunur. Ayrıca, test stratejileri yazılımın pazardaki başarısını ve kullanıcı memnuniyetini artırmada da önemli bir etkendir. Çünkü kaliteli bir yazılım, kullanıcı ihtiyaçlarını daha etkili bir şekilde karşılar ve genel olarak daha güvenilir bir kullanıcı deneyimi sunar (Garg, 2015).

## 2.6. YAZILIMDA TEST SEVİYELERİ VE TEST TÜRLERİ

Yazılım testi, çeşitli seviyelerde ve türlerde yapılarak yazılımın gereksinimleri karşıladığından ve yüksek kalitede olduğundan emin olmayı amaçlar.

### 2.6.1. Birim Testleri (Unit Testing)

Birim testleri, yazılım geliştirme sürecinin temel bir unsuru olarak kullanılmaktadır. Yazılımın en küçük test edilebilir birimlerinin doğruluğu ve işlevselliği bu testler aracılığıyla doğrulanmaktadır. Genellikle fonksiyonlar, metotlar veya modüller gibi ayrı yazılım bileşenlerine odaklanan bu testler, bileşenlerin beklenen çıktıları doğru bir şekilde üretip üretmediğini kontrol etmektedir. Birim testlerinin uygulanması, geliştirme sürecinin erken aşamalarında gerçekleştirilmekte ve bu sayede hataların mümkün olan en erken zamanda tespit edilmesi ve düzeltilmesi sağlanmaktadır. Erken tespit, yazılımın daha sonraki geliştirme aşamalarında meydana gelebilecek daha karmaşık sorunların önüne geçmekte ve geliştirme sürecinin genel verimliliğini artırmaktadır. Ayrıca, birim testleri, geliştiricilerin kodlarını daha güvenli bir şekilde değiştirmelerine olanak tanımakta ve yazılımın esnekliğini ve bakım kolaylığını da desteklemektedir (Srivastava, Kumar, & Singh, 2021).

Birim testlerinin etkin kullanımı, yazılım kalitesini büyük ölçüde artırmaktadır. Bu testler, yazılım bileşenlerinin izole bir ortamda, yani diğer bileşenlerden bağımsız olarak test edilmesine imkan tanımaktadır. Bu izolasyon, hataların kaynağının daha hızlı ve doğru bir şekilde belirlenmesini sağlar. Çünkü her test sadece bir bileşeni ve onun işlevlerini hedef almaktadır. Ayrıca, birim testleri otomatikleştirilebilmekte, bu da test süreçlerinin hızlı ve tekrar edilebilir olmasını sağlamaktadır. Otomatikleştirilmiş birim testleri, yazılım geliştirme sürecinin daha tutarlı ve hatasız ilerlemesine yardımcı olmakta ve geliştiricilerin daha büyük güvenlikle yeni özellikler eklemelerine veya mevcut kod üzerinde değişiklik yapmalarına olanak tanımaktadır. Birim testleri, yazılımın kalite güvencesi stratejisinin ayrılmaz bir parçası olarak kabul edilmekte ve yazılımın güvenilirliğini, sürdürülebilirliğini ve kullanıcı memnuniyetini artırmada kritik bir rol oynamaktadır.

### 2.6.2. Entegrasyon Testleri (Integration Testing)

Entegrasyon testleri, yazılım geliştirme sürecinde birim testlerinden sonra gerçekleştirilen önemli bir test aşamasıdır. Bu testler, birim testlerinden başarıyla geçen modüllerin veya bileşenlerin bir araya getirilip getirilmediğinde birbiriyle doğru şekilde çalışıp çalışmadığını doğrulamak amacıyla yapılmaktadır. Entegrasyon testlerinin temel amacı, modüller arası arayüzlerde ve etkileşimlerde oluşabilecek sorunları tespit etmektir. Bu testler, modüllerin veya sistem bileşenlerinin bir araya geldiğinde oluşan

bütünleşik yapı içerisinde, tasarımın ve gereksinimlerin doğru bir şekilde karşılanıp karşılanmadığını değerlendirmekte ve her bir bileşenin diğerleriyle olan etkileşimini test etmektedir (Srivastava, Kumar, & Singh, 2021).

Entegrasyon testleri, yazılımın farklı modüllerinin birlikte çalışabilirliğini ve sistem içindeki veri akışını doğrulamak için kritik öneme sahiptir. Bu testler sırasında, arayüzler, veri aktarım mekanizmaları ve diğer sistem etkileşimleri detaylı bir şekilde incelenmekte ve modüller arası geçişlerde veri kaybı veya hata olup olmadığı kontrol edilmektedir. Ayrıca, entegrasyon testleri, yazılımın gerçek dünya koşullarında ve genel sistem içerisinde nasıl performans göstereceği konusunda önemli bilgiler sağlamaktadır. Bu test aşaması, modüllerin birbirleriyle olan uyumunu ve sistem içindeki işlevselliğini değerlendirerek, yazılımın toplam kalitesinin artırılmasına yönelik önemli bir adım olarak görülmektedir (Srivastava, Kumar, & Singh, 2021).

### 2.6.3. **Sistem Testleri (System Testing)**

Sistem testleri, yazılım geliştirme sürecinin önemli bir aşamasını oluşturur ve tamamlanmış bir yazılım sisteminin fonksiyonel ve gereksinim koşullarına uygun olarak nasıl çalıştığını doğrulamak amacıyla yapılmaktadır. Bu testler, yazılımın tüm bileşenlerinin bir araya getirildiği ve entegre bir şekilde doğru çalıştığını test etmekte ve sistemin belirlenen gereksinimleri karşılayıp karşılamadığını değerlendirmektedir. Sistem testleri, yazılımın kullanıcı ihtiyaçlarını ve beklentilerini karşılayacak şekilde işlevsel ve güvenilir olduğunu doğrulamayı amaçlar (Srivastava, Kumar, & Singh, 2021).

Sistem testleri sırasında, yazılımın farklı işlevleri ve senaryoları üzerinde kapsamlı bir test süreci uygulanmaktadır. Bu süreçte, kullanıcıların gerçek dünya senaryolarına benzer durumlar simüle edilerek, yazılımın performansı ve işlevselliği incelenmektedir. Sistem testleri ayrıca, yazılımın hata toleransını ve güvenilirliğini değerlendirirken, aynı zamanda kullanıcı arayüzü ve kullanılabilirlik gibi faktörleri de değerlendirir. Bu test aşaması, yazılımın kalitesini artırmak ve son kullanıcıya güvenilir bir deneyim sunmak için kritik öneme sahiptir (Srivastava, Kumar, & Singh, 2021).

### 2.6.4. **Kabul Testleri (Acceptance Testing)**

Kabul testleri, yazılım geliştirme sürecinde son aşamayı oluşturur ve yazılımın son kullanıcı veya müşteri gereksinimlerini karşılayıp karşılamadığını doğrulamak amacıyla

yapılmaktadır. Bu testler, yazılımın son haliyle son kullanıcı veya müşteri tarafından kabul edilip edilmeyeceğini belirlemek için önemli bir rol oynar. Kabul testleri genellikle yazılımın üretim ortamına taşınmadan önce gerçekleştirilir ve bu süreçte yazılımın işlevselliği, performansı, güvenilirliği ve kullanılabilirliği gibi faktörler detaylı bir şekilde incelenir. Bu testlerin amacı, yazılımın son kullanıcı beklentilerini ve gereksinimlerini eksiksiz bir şekilde karşılayıp karşılamadığını belirleyerek, yazılımın kullanıma hazır olup olmadığını doğrulamaktır (Srivastava, Kumar, & Singh, 2021).

Kabul testleri sırasında, yazılımın farklı kullanım senaryoları üzerinde kapsamlı bir test süreci uygulanır ve yazılımın belirlenen gereksinimleri ne kadar başarılı bir şekilde karşıladığı değerlendirilir. Bu süreçte, kullanıcıların gerçek dünya senaryolarına benzer durumlar simüle edilir ve yazılımın performansı gerçek kullanım koşullarında test edilir. Ayrıca, Kabul testleri sırasında kullanıcı geri bildirimleri de önemli bir rol oynar ve yazılımın son kullanıcı ihtiyaçlarına uygunluğu ve kullanıcı deneyimi hakkında değerli bilgiler sağlar. Bu test aşaması, yazılımın son kullanıcı veya müşteri tarafından kabul edilip edilmeyeceğini belirlemek için kritik bir adımdır ve yazılımın piyasaya sürülmesinden önce yapılan son kontroldür.

### 2.6.5. **Fonksiyonel Testler ve Otomasyonun Rolü**

Fonksiyonel testler, yazılımın belirlenmiş işlevleri yerine getirip getirmediğini değerlendirmek için kullanılan kritik bir test türüdür. Bu testler, yazılımın işlevsel gereksinimlerini kontrol etmek ve doğrulamak amacıyla gerçekleştirilir. 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, fonksiyonel testlerin daha hızlı ve tutarlı bir şekilde gerçekleştirilmesini sağlayan önemli bir araçtır. Otomasyon, tekrar eden test senaryolarının otomatik olarak yürütülmesini ve sonuçların raporlanmasını mümkün kılar. Bu, test süreçlerinin hızlandırılmasına ve yazılımın işlevselliğinin daha kapsamlı bir şekilde değerlendirilmesine olanak tanır. Ayrıca, otomasyon, manuel hataların azaltılmasına ve test süreçlerinin daha verimli hale getirilmesine katkı sağlar. Otomasyonun kullanılmasıyla birlikte, yazılım geliştirme sürecindeki test aşamaları daha etkin bir

şekilde yönetilebilir ve yazılımın kalitesi artırılabilir. (Srivastava, Kumar, & Singh, 2021).

## 2.7. OTOMASYON TESTLERİ

Otomasyon testleri, yazılım test süreçlerini daha hızlı ve etkili hale getirmek için otomasyon araçları kullanarak testlerin gerçekleştirilmesidir. Bu testler, manuel test süreçlerini otomatize ederek tekrar eden görevleri azaltır ve test süreçlerinin kapsamını ve tutarlılığını artırır.

### 2.7.1. Otomasyon Testlerinin Tanımı ve Önemi

Otomasyon testleri, yazılımın farklı modüllerini kapsamlı ve etkin bir şekilde analiz eden test senaryolarının ve araçlarının uygulandığı bir süreç olarak öne çıkmaktadır. Bu testler sayesinde yazılım geliştirme sürecindeki hatalar daha hızlı tespit edilmekte ve düzeltilmektedir. Manuel testlere göre çok daha yüksek doğrulukta ve sürekli bir izleme sağlandığından, yazılım projelerinin toplamda daha güvenilir bir şekilde yürütülmesine olanak tanınmaktadır (Sultania, 2015).

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. Genişletilmiş test kapsamı ve sürekliliği ile otomasyon, manuel testlerle karşılaştırıldığında daha uygun maliyetli olabilmekte ve hata riskini minimize etmektedir. Dolayısıyla, otomasyon testleri, yazılım geliştirme döngüsünün vazgeçilmez bir parçası olarak değerlendirilmektedir (Sultania, 2015).

### 2.7.2. Otomasyon Test Araçları

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. Bu otomasyon araçları, test sürelerini önemli ölçüde azaltmakta ve daha kapsamlı bir test kapsamı sağlamaktadır. Selenium, QTP (Quick Test Professional) ve TestComplete gibi araçlar, farklı test senaryolarını ve platformları destekleyerek geniş bir yelpazede uygulamaların test edilmesine olanak tanımaktadır. Bu çeşitlilik, yazılım testlerinin daha esnek ve kapsayıcı bir şekilde gerçekleştirilmesine yardımcı olmaktadır (Kumar ve Rao, 2012).

Kumar ve Rao'nun (2012) belirttiği gibi, otomasyon test araçlarının kullanımı, test süreçlerinin daha etkin yönetilmesine ve maliyetlerin optimize edilmesine imkan vermektedir. Test süreçlerini hızlandıran ve daha az insan kaynağına ihtiyaç duyulan bu araçlar, yazılım projelerinin bütçe ve zaman açısından daha verimli bir şekilde tamamlanmasına katkıda bulunmaktadır. Bu nedenle, otomasyon araçlarının etkin kullanımı, yazılım geliştirme ve test süreçlerinin vazgeçilmez bir unsuru olarak kabul edilmektedir.

### 2.7.3. Otomasyon Testlerinin Çevik Projelere Entegrasyonu

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 sağlayarak, hataların erken aşamada tespit edilmesine ve düzeltilmesine imkan tanımaktadır. Ayrıca, 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'ın (2014) belirttiği gibi, bu hızlı adaptasyon ve iterasyon yeteneği, çevik projelerde başarının kritik faktörlerinden biri olarak görülmektedir. Otomasyon testleri, çevik yazılım geliştirme süreçlerinde hızlı geri bildirim alınmasını sağlayarak projelerin daha etkin bir şekilde yönetilmesine katkıda bulunmaktadır. Ekiplerin zamanında ve kaliteli yazılım teslim etmelerini destekleyen bu süreçler, yazılımın piyasaya sürülme sürecinde karşılaşılacak potansiyel riskleri minimize etmekte ve projenin genel başarısını artırmaktadır. Bu

nedenle, otomasyon testlerinin çevik metodolojiler içerisinde stratejik bir yere sahip olduğu ve sürekli entegrasyon ile sürekli teslimat süreçlerinin temel taşlarından biri olduğu ifade edilmektedir (Rajasekhar, 2014).

### 2.7.4. Otomasyon Testlerinin Zorlukları

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 caydırıcı bir faktör olabilmektedir. Bu yüksek maliyetler, otomasyon testlerinin yaygınlaşmasını sınırlamakta ve bazı projelerde manuel test yöntemlerinin tercih edilmesine neden olmaktadır (Saleem et al., 2014).

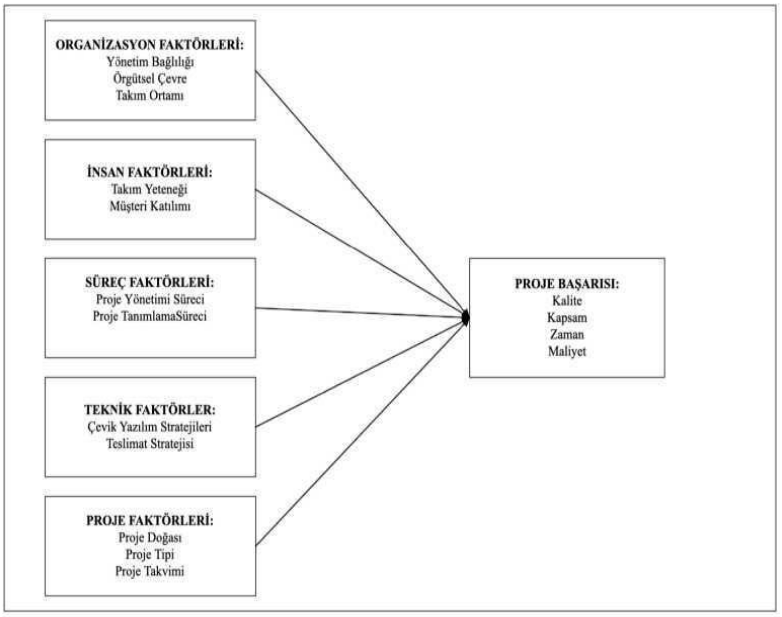
İkinci bir zorluk ise, test senaryolarının bakımının zor olmasıdır. Otomasyon test senaryoları, yazılımın gelişimine bağlı olarak sürekli güncellenmeli ve revize edilmelidir. Ancak, bu süreç zaman alıcı ve karmaşık olabilmekte, özellikle yazılımın sürekli olarak değiştiği durumlarda senaryoların güncel ve doğru kalmasını sağlamak zorlaşmaktadır. Bu durum, test süreçlerinin etkinliğini doğrudan etkileyebilmekte ve ekstra kaynak gereksinimine yol açmaktadır (Saleem et al., 2014).

Üçüncü olarak, otomasyon testlerinin etkinliği büyük ölçüde test senaryolarının kalitesine bağlıdır. Saleem ve arkadaşlarının (2014) belirttiği gibi, iyi 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 (Saleem et al., 2014).

## 2.8. İLGİLİ ARAŞTIRMALAR

Chow ve Cao 2008 yılında yaptığı “Çevik yazılım projelerindeki kritik başarı faktörlerine ilişkin bir anket çalışması” araştırmasında çevik yazılım projelerindeki başarı faktörlerini göz önünde bulundurarak başarılı bir çevik yazılım geliştirme projesine katkıda bulunacak en kritik faktörleri belirlemek için bir çalışma gerçekleştirmiştir. Bu araştırma, çevik yazılım geliştirme projelerinin kritik başarı

faktörleri üzerine yapılan bir anket çalışmasıdır. Çevik proje yönetimi, organizasyon, insan, süreç, teknik ve proje olmak üzere beş faktörden oluşmaktadır. Organizasyon faktörü; yönetim bağlılığı, örgütsel çevre, takım ortamı alt boyutları ile; insan faktörü; takım yeteneği ve müşteri katılımı alt boyutları ile; süreç faktörü; proje yönetimi süreci ve proje tanımlama süreci alt boyutları ile; teknik faktörü; çevik yazılım stratejileri ve teslimat stratejileri alt boyutları ile; proje faktörü; proje doğası, proje tipi ve proje takvimi alt boyutları ile ölçülmüştür. Chow ve Cao'nun çalışmasında proje başarısı kalite, kapsam, zaman ve maliyet olarak dört boyutta incelenmiştir. Chow ve Cao'nun çalışmasında her bir çevik proje yönetimi faktörünün alt boyutları ile proje başarı boyutları arasındaki ilişkinin incelenmesine yönelik hipotezler kurulmuştur (Şekil 2.1).

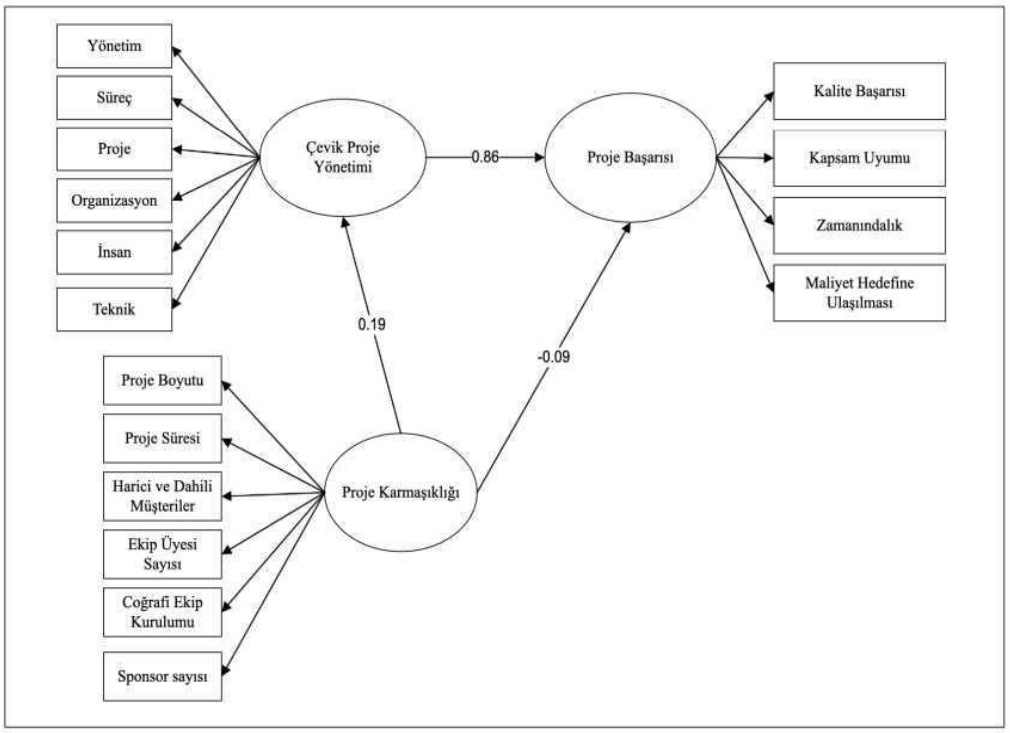



Şekil 2.1. Chow ve Cao (2008) araştırma modeli

Chow ve Cao'nun çalışmasında kalite, kapsam, zaman ve maliyet kritik başarı faktörlerini nihai bir sette birleştirmek için güvenilirlik analizi ve faktör analizi gerçekleştirilmiştir. Araştırmada; zaman ve maliyet başarı boyutlarını takım yeteneği ve teslim stratejisi alt boyutlarının etkilediği; kapsam boyutunu, çevik yazılım teknikleri, teslimat stratejisi ve müşteri katılımı alt boyutlarının etkilediği; son olarak, kalite boyutunu ise çevik yazılım teknikleri, ekip ortamı ve proje yönetim süreci alt boyutlarının etkilediği sonucuna ulaşılmıştır.

Bergmann'ın 2018 yılında yaptığı "Çevik Proje Yönetimi ve Proje Başarı Sonuçları arasındaki İlişki" araştırmasında incelenen temel ilişki, çevik proje yönetimi ile proje başarısı arasındaki ilişkidir. Bergmann'ın çalışmasında ayrıca proje karmaşıklığının

çevik proje yönetimi ve proje başarısı üzerindeki etkileri ve endüstri sektörlerinin proje başarısı üzerine etkileri de incelenmiştir. Çevik proje yönetimi, yönetim, organizasyonel, insan, süreç, teknik ve proje gözlenen değişkenden oluşan, gözlemlenemeyen bir değişkendir. Aynı şekilde, proje başarısı kalite başarısı, kapsam uyumu, zamanındalık ve maliyet hedefine ulaşılması alt boyutlarına ayrılmıştır. Proje karmaşıklığı; proje büyüklüğü, sektör, proje süresi, müşterilerin harici veya dahili olup olmadığı, ekip üyesi sayısı, coğrafi ekip kurulumu ve proje sponsorlarının sayısı alt boyutlarından oluşmaktadır. Yol katsayıları çok düşük olduğu ve sonuç olarak proje başarısı üzerindeki etkileri çok zayıf olduğu için beş endüstri sektörü (birincil, ikincil, üçüncül, dördüncül ve diğer sektörler) değişkenleri modelden kaldırılmıştır (şekil 2.2).

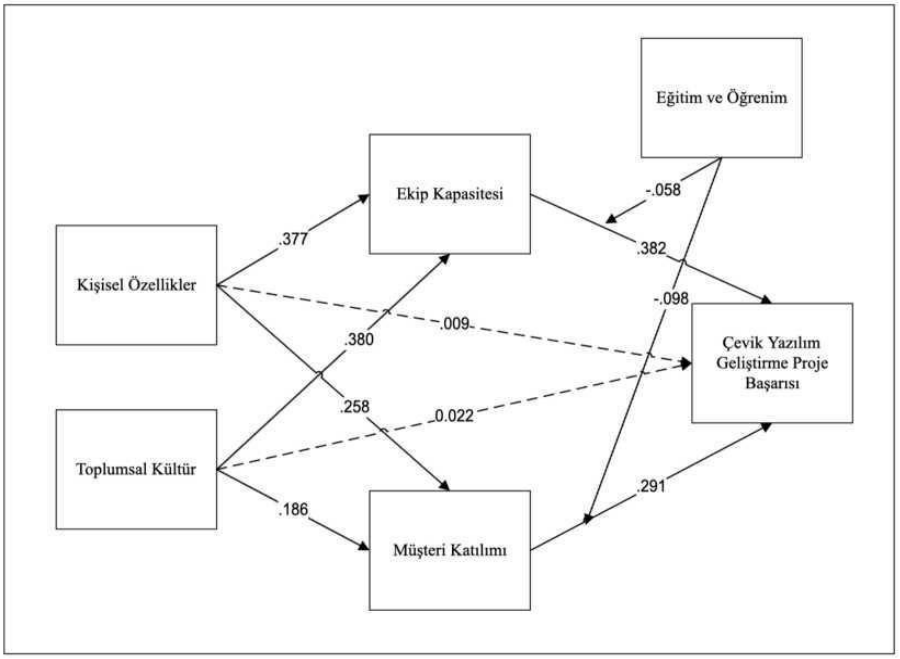



Şekil 2.2. Bergman (2008)'in revize yapısal eşitlik modeli.

Bergmann'ın çalışmasının sonucu olarak; süreç, organizasyon ve insan alt boyutlarının çevik proje yönetimi faktörünü en iyi açıklayan alt boyutlar olduğu görülmüştür. Ekip üye sayısı proje karmaşıklığını, kalite başarısı alt boyutu ise proje başarısı faktörünü ölçen en önemli alt boyut olarak belirlenmiştir. Çevik proje yönetiminin proje başarısı üzerinde güçlü ve anlamlı bir etkisi olduğu görülmüştür.

Carlos ve diğ. 2020 yılında "Devam eden çevik yazılım geliştirme projelerinin başarısını etkileyen faktörler" araştırmasında, Chow ve Cao (2008) ve Misra ve diğ. (2009) tarafından yapılan çevik yazılım geliştirme projelerinde başarıyla ilişkili olduğu

kanıtlanan çalışmalarda kullanılan beş insan faktörü seçilmiştir. "Kişisel Özellikler", "Eğitim ve Öğrenme", "Toplumsal Kültür", "Ekip Yeteneği" ve "Müşteri Katılımı" faktörleri kavramsal bir modelde birleştirilerek modelin geçerliliği test edilmiştir. Bu çalışmanın bağımlı değişkeni olan çevik yazılım geliştirme projesi başarısı, zaman, maliyet ve müşteri memnuniyeti alt boyutları ile tanımlanmıştır (şekil 2.3).




Şekil 2.3. Carlos ve diğ. (2020) araştırma modeli.

Carlos vd.'nin çalışmasının sonucunda Kişisel Özellikler ve Toplumsal Kültür faktörleri Ekip Kapasitesi ve Müşteri Katılımı faktörlerini açıklamada istatistiksel olarak anlamlı çıkmıştır. Ekip kapasitesi ve müşteri katılımı faktörleri çevik yazılım geliştirme projesi faktörünü açıklamada istatistiksel olarak anlamlı çıkmıştır. Çevik yazılım geliştirme projesi başarısını açıklamada kişisel özellikler ve toplumsal kültür faktörleri istatistiksel olarak anlamlı çıkmamıştır. Eğitim ve öğrenim, çevik yazılım geliştirme projesi başarısı arasındaki ilişkide moderatör bir değişken olarak değerlendirilmiştir. Eğitim ve Öğrenme özelliklerinin yüksek değerinin, çevik yazılım geliştirme projesi başarısı üzerindeki Müşteri Katılımının etkisini zayıflattığı görülmüştür.

Serrador ve Pinto'nun (2015) yaptığı "Çevik çalışıyor mu? Çevik proje başarısının nicel bir analizi" çalışmasında, organizasyonlarda çevik proje yönetimi kullanımının proje başarısının üzerindeki etkisi test edilmiştir. Proje başarısı boyutu; verimlilik, paydaş memnuniyeti, genel proje performansı alt boyutları ile ölçülmüştür. Çalışmanın sonucunda, bir projede kullanılan çeviklik düzeyinin, verimlilik, paydaş memnuniyeti

ve genel proje performansı algısı ile değerlendirildiğinde, proje başarısının her üç alt boyutu üzerinde de istatistiksel olarak anlamlı bir etkiye sahip olduğu gösterilmiştir. Önerilen diğer iki moderatörün (ekibin deneyimi ve projenin karmaşıklığı) çeviklik ile proje başarısı arasındaki ilişkiyi etkilemediği görülmüştür.

Lee ve Xia (2010) tarafından yapılan "Çevik gelişim sırasında kalite başarısını etkileyen faktörler" başlıklı çalışma, çevik yazılım geliştirme projelerinde kalite başarısını etkileyen faktörleri incelemiştir. Araştırma, takım yeteneği, müşteri katılımı ve teknik çeviklik faktörlerinin kalite başarısı üzerinde önemli etkileri olduğunu ortaya koymuştur. Çalışma ayrıca, çevik yöntemlerin doğru uygulanmasının kaliteyi artırmada kritik olduğunu vurgulamaktadır (Lee ve Xia, 2010).

Hoda, Noble ve Marshall (2011) tarafından yapılan "Çevik projelerdeki rol dinamikleri" başlıklı araştırma, çevik projelerdeki rol dinamiklerinin proje başarısına etkisini incelemiştir. Çalışma, takım içindeki rol netliğinin ve müşteri ile sürekli iletişimin proje başarısını olumlu yönde etkilediğini göstermiştir. Ayrıca, çevik projelerde liderlik ve takım içi koordinasyonun kritik olduğu sonucuna varılmıştır (Hoda, Noble ve Marshall, 2011).

Misra, Kumar ve Kumar (2009) tarafından yapılan "Çevik yazılım geliştirme projelerinde başarı faktörleri" başlıklı araştırma, çevik projelerdeki başarı faktörlerini detaylı olarak incelemiştir. Çalışma, takım yeteneği, müşteri katılımı, proje yönetim süreçleri ve teknik çeviklik faktörlerinin başarı üzerinde önemli etkileri olduğunu göstermiştir (Misra, Kumar ve Kumar, 2009).

# 3. MATERYAL ve YÖNTEM

## 3.1. MATERYAL

Bu çalışmada, Türkiye'deki firmalarda yapılan çevik (Agile) projelerde fonksiyonel ve otomasyon test süreçlerinin detaylarının ve işleyişinin incelenmesi amaçlanmıştır. Çalışmada kullanılan ana materyal, yazılım firmalarında çeşitli pozisyonlarda çalışan profesyonellerle yapılacak yarı yapılandırılmış bir görüşme formudur. Form, katılımcıların demografik bilgilerini, profesyonel deneyimlerini ve Çevik metodolojileri ile ilgili uygulamalarını derinlemesine sorgulamaktadır.

### 3.1.1. Anket Formunun İçeriği

Demografik Bilgiler;

- Cinsiyet
- Yaş
- Eğitim durumu
- Mevcut çalışma durumu (işsiz, tam zamanlı çalışan, yarı zamanlı çalışan, öğrenci vb.)
- Çalışılan firma ve pozisyon bilgileri

### 3.1.2. Çevik Metodolojileri Üzerine Sorular

- Kullanılan yazılım geliştirme modelleri
- Tercih edilen Çevik metodolojileri (Scrum, Kanban, Extreme Programming vb.)
- Ekip büyüklüğü ve çalışma modeli (uzaktan, hibrit, ofiste)

### 3.1.3. Scrum Metodolojisi Detayları

- Scrum panosu kullanımı
- Kullanılan araçlar (Jira, Miro vb.)
- Scrum etkinlikleri ve süreçlerinin memnuniyet düzeyi

- Scrum Master ve diğer rollerin işleyişi

#### 3.1.4. Kanban Metodolojisi Detayları

- Kanban panosu kullanımı ve tercih edilen araçlar
- Kanban metodolojisinde roller ve memnuniyet düzeyi

#### 3.1.5. Fonksiyonel Test Süreçleri

- Manuel ve otomasyon testlerinin yapılıp yapılmadığı
- Test süreçlerinde kullanılan araçlar ve dokümantasyon

#### 3.1.6. Otomasyon Test Süreçleri

- Otomasyon testlerinin yapılma durumu ve kullanılan araçlar
- Otomasyon testlerinde kullanılan senaryolar ve test süreçlerinin entegrasyonu

Araştırma formu, katılımcıların fonksiyonel ve otomasyon test süreçlerine ilişkin deneyimlerini, bu süreçlerde karşılaştıkları zorlukları ve çözüm yollarını anlamak için detaylı sorular içermektedir. Bu çalışmanın amacı, yazılım geliştirme ve test süreçlerinde en iyi uygulamaları belirleyerek, sektördeki standartları geliştirmek ve karşılaşılan problemlere çözüm önerileri sunmaktır.

## 3.2. YÖNTEM

Türkiye'deki firmalardaki çevik (Agile) projelerinde fonksiyonel ve otomasyon testlerinin süreçleri ve işleyişini inceleyen bu araştırma, karma bir metodoloji kullanarak tasarlanmıştır. Araştırma hem niceliksel hem de niteliksel veri toplama tekniklerini içermektedir. Öncelikle, yazılım geliştirme süreçlerine dahil profesyonellerden veri toplamak amacıyla yarı yapılandırılmış derinlemesine görüşmeler yapılmıştır. Bu araştırma formu, demografik bilgileri toplamak için tasarlanmış kısa anket soruları içermekte ve ardından Çevik metodolojileri, Scrum ve Kanban metodolojileri ile ilgili daha detaylı sorulara yer vermektedir. Anket soruları, fonksiyonel ve otomasyon test süreçlerine yönelik spesifik soruları da kapsamaktadır.

### 3.2.1. Araştırmanın Amacı

Bu araştırmanın amacı, Türkiye'deki firmalardaki çevik (Agile) projelerinde fonksiyonel ve otomasyon test süreçleri ile ilgili detaylı bir anlayış sağlamaktır. Araştırma, yazılım geliştirme ve test uygulamalarındaki mevcut trendleri, zorlukları ve en iyi uygulamaları belirlemeyi amaçlamaktadır. Katılımcılardan elde edilen demografik bilgiler, çalışma modelleri, kullanılan teknolojiler ve metodolojiler hakkında ayrıntılı bilgi toplanarak, sektördeki çeşitli rollerin test süreçlerine katılım düzeyleri ve tercihleri incelenmiştir. Bu bilgiler, Çevik metodolojilerin Türkiye'deki yazılım geliştirme ortamlarında nasıl uygulandığını ve bu süreçlerin yazılım kalitesi üzerindeki etkilerini daha iyi anlamak için kullanılmaktadır. Araştırma, Çevik projelerdeki test süreçlerinin işleyişini detaylandırarak, yazılım firmalarına, bilişim teknolojileri alanında çalışan profesyonellere ve insan kaynakları çalışanlarına rehberlik etmek, sektördeki uygulamaları geliştirmek ve optimizasyon yollarını sunmak için önemli bir kaynak olmayı hedeflemektedir.

### 3.2.2. Araştırmanın Önemi

Bu araştırma, Türkiye'deki firmalardaki Çevik projelerinde fonksiyonel ve otomasyon test süreçlerinin işleyişi ve etkinliği hakkında değerli bilgiler sağlaması açısından önemlidir. Yazılım geliştirme süreçlerinin kritik bir parçası olan test süreçleri, ürün kalitesini doğrudan etkileyerek yazılım hatalarının azaltılmasında ve kullanıcı memnuniyetinin artırılmasında önemli bir role sahiptir.

Sektör Standartlarını Belirleme: Araştırma, Türkiye'deki yazılım geliştirme sektöründe Çevik metodolojilerin uygulanma şeklini ve etkinliğini ortaya koyarak, sektörel standartların belirlenmesine yardımcı olabilir. Bu sayede firmalar, global trendlerle uyumlu olarak kendi süreçlerini gözden geçirme fırsatı bulabilir.

Verimlilik ve Etkinlik: Test süreçlerinin daha verimli ve etkili bir şekilde tasarlanmasını teşvik eder. Araştırma sonuçları, çeşitli Çevik uygulamaları ve test stratejileri arasındaki ilişkileri göstererek, hangi metodolojilerin ve tekniklerin daha başarılı olduğunu ortaya koyabilir.

Eğitim ve Gelişim: Araştırma, eğitim programlarının ve profesyonel gelişim faaliyetlerinin daha etkili bir şekilde planlanmasına olanak tanır. Öğrenilen bilgiler, yazılım testi alanında çalışan veya bu alana girmek isteyen profesyoneller için önemli öğrenme noktaları sunar.

Karar Verme ve Politika Geliştirme: Firmanın iç süreçlerini iyileştirmek, daha bilinçli kararlar almak ve risk yönetimini optimize etmek için kullanılabilir bilgiler sunar. Ayrıca, insan kaynakları yöneticileri için işe alım ve eğitim politikalarını şekillendirme konusunda faydalı veriler sağlayabilir.

Sektörel Benchmarking: Diğer firmaların uygulamalarıyla karşılaştırma yapma imkanı tanır. Bu sayede firmalar, kendi süreçlerini endüstri ortalamaları veya en iyi uygulamalarla karşılaştırabilir ve gerektiğinde iyileştirmeler yapabilir.

Bu nedenlerle, bu araştırma yazılım geliştirme ve test süreçlerinin sürekli iyileştirilmesine katkıda bulunarak teknoloji ve yazılım sektöründe kalite ve yenilikçiliği teşvik etme potansiyeline sahiptir.

### 3.2.3. Evren ve Örneklem

Bu araştırmanın evreni, Türkiye'deki firmalardan 52 tane firma ve 116 tane kişiden alınan bilgiler doğrultusunda Çevik projelerde çalışan ve fonksiyonel ile otomasyon test süreçlerine dahil olan yazılım profesyonelleridir. Bu geniş evren, çeşitli sektörlerdeki şirketleri ve çeşitli büyüklükteki işletmeleri içerir. Evrenin özellikleri, Türkiye'nin yazılım ve teknoloji sektöründeki çeşitliliği ile tanımlanır. Burada bankacılık, telekomünikasyon, e-ticaret gibi alanlarda faaliyet gösteren firmalar yer almaktadır.

Araştırmanın örneklemini, Türkiye'de Çevik metodolojileri kullanarak çalışan ve çeşitli fonksiyonel ve otomasyon test görevleriyle ilgilenen 52 adet farklı firmadaki 116 yazılım profesyonelinden oluşan bir grup oluşturmaktadır. Örneklem, farklı şehirlerden (özellikle İstanbul, Ankara ve İzmir gibi büyük şehirlerden), farklı rollerde (yazılım geliştiriciler, test mühendisleri, proje yöneticileri vb.) ve farklı tecrübe seviyelerinden bireyler içerir. Örneklem aynı zamanda farklı çalışma modellerini (uzaktan, hibrit ve ofiste) temsil eder. Bu da çalışmanın sonuçlarının geniş bir perspektifi yansıtmasını sağlar.

Bu örneklem seçimi, araştırma sorularına cevap verirken yazılım sektöründeki çeşitli deneyim ve perspektifleri kapsamlı bir şekilde temsil etmeyi amaçlar. Örneklemin boyutu ve çeşitliliği, elde edilen bulguların Türkiye'deki firmalardaki çevik projelerinde test süreçlerine dair genel eğilimler ve normlar hakkında güvenilir içgörüler sağlamasına yardımcı olur.

### 3.2.4. Verilerin Toplanması

Veri toplama yöntemi olarak, yazılım profesyonelleriyle yapılandırılmış anketler kullanılmıştır. Anket, katılımcıların demografik bilgilerini, iş ve eğitim geçmişlerini, çevik ve Scrum metodolojileri hakkındaki tecrübelerini ve fonksiyonel ile otomasyon test süreçlerine dair detaylı soruları içermektedir.

Anketler, katılımcılara çevrimiçi olarak dağıtılmış ve doldurulması için bir saat süre verilmiştir. Katılımcılara gizlilik ve anonimlik güvencesi verilerek, toplanan verilerin sadece bu araştırma çerçevesinde kullanılacağı belirtilmiştir. Ayrıca katılımcılardan verilen cevaplar üzerinden genel bir değerlendirme yapabilmek için, çeşitli iş ve eğitim durumlarından bireyler dahil edilmiştir.

Veri toplama süreci, katılımcıların çevik projelerdeki rol ve deneyimlerine dair kapsamlı bilgiler sağlaması amacıyla detaylı olarak planlanmıştır. Bu bilgiler, fonksiyonel ve otomasyon test süreçlerinin çevik metodolojilerle nasıl entegre edildiğini ve bu süreçlerin etkinliğini değerlendirmek için kullanılmıştır. Toplanan veriler, daha sonraki analizler için dijital bir veri tabanında saklanmıştır.

### 3.2.5. Verilerin Analizi

Verilerin analizi için, nitel veriler tematik analiz ile değerlendirilirken, nicel veriler ise istatistiksel analiz yöntemleri kullanılarak incelenmiştir. Bu karma yaklaşım, test süreçlerinin çeşitli yönlerini daha kapsamlı bir şekilde anlamayı ve projelerin başarısındaki rolünü daha detaylı bir şekilde irdelenmesine olanak tanıyacaktır. Araştırmanın sonuçları, sektördeki en iyi uygulamaları belirlemeye ve yazılım geliştirme süreçlerinde karşılaşılan zorluklara çözüm önerileri getirmeye katkıda bulunması beklenmektedir.

# 4. BULGULAR VE TARTIŞMA

## 4.1. DEMOGRAFİK BULGULAR

Çizelge 4.1. Demografik Bilgilere Yönelik Bulgular

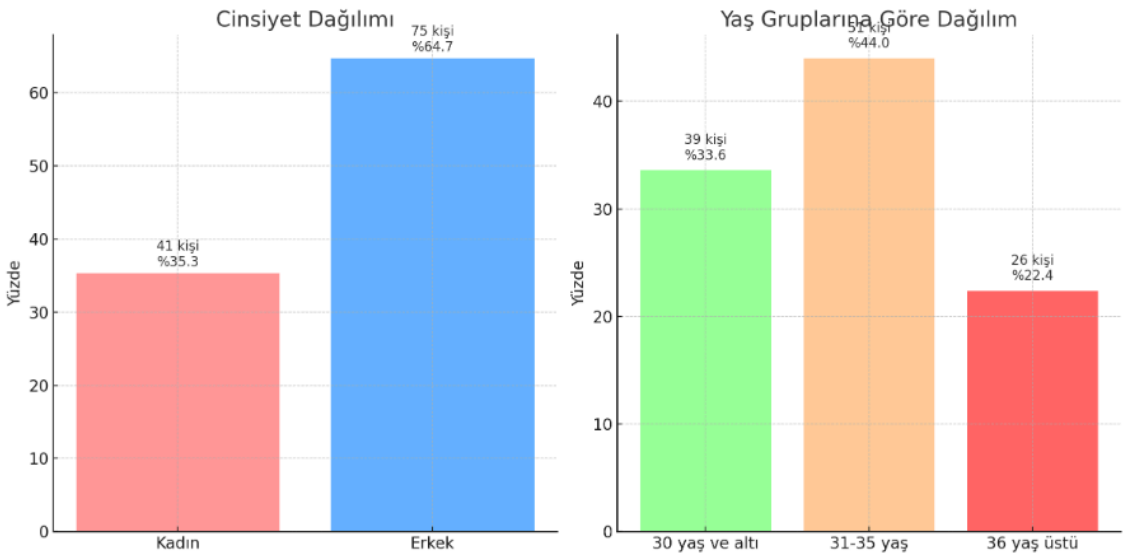
| | | n | % |
|---|---|---|---|
| Cinsiyet | Kadın | 41 | 35,3 |
| | Erkek | 75 | 64,7 |
| Kaç yaşındasınız? | 30 yaş ve altı | 39 | 33,6 |
| | 31-35 yaş | 51 | 44,0 |
| | 36 yaş ve üstü | 26 | 22,4 |
| Eğitim Durumunuz nedir? | Lisans | 96 | 82,8 |
| | Ön lisans | 2 | 1,7 |
| | Yüksek lisans | 18 | 15,5 |
| Şu an bir firmada ya da kurumda aktif olarak çalışıyor musunuz? | Evet | 111 | 100,0 |
| | Hayır | 0 | 0,0 |
| Şu an ki iş durumunuz nedir? | İşsiz ve iş arıyorum | 0 | 0,0 |
| | Öğrenciyim | 0 | 0,0 |
| | Tam zamanlı | 115 | 99,1 |
| | Yarı zamanlı | 0 | 0,0 |
| | Serbest | 1 | 0,9 |
| | Diğer | 0 | 0,0 |
| Şu anki çalışma modeliniz nedir? | Uzaktan | 34 | 29,3 |
| | Hibrit (Ofis + Uzaktan) | 78 | 67,2 |
| | Ofiste | 4 | 3,4 |
| | Diğer | 0 | 0,0 |
| Şu an ki iş yerinizin hangi şehirdedir? | İstanbul | 99 | 87,6 |
| | Ankara | 8 | 7,1 |
| | İzmir | 3 | 2,7 |
| | Diğer | 3 | 2,7 |
| İş yeri İstanbul ise İstanbul'un hangi yakasında bulunmaktadır? | Anadolu yakası | 62 | 63,9 |
| | Avrupa yakası | 35 | 36,1 |

| | | | |
|---|---|---|---|
| Firmadaki rolünüz nedir? | Proje Yöneticisi | 7 | 6,0 |
| | İş Analisti | 31 | 26,7 |
| | Teknik Analist | 0 | 0,0 |
| | Yazılım Geliştirici | 28 | 24,1 |
| | Teknik Lider | 12 | 10,3 |
| | Test Kalite Kontrol | 28 | 24,1 |
| | Ürün sahibi | 6 | 5,2 |
| | Diğer | 4 | 3,4 |
| İş tecrübeniz ne kadar? | 1 yıla yakın | 1 | 0,9 |
| | 1-2 yıl | 9 | 7,8 |
| | 2-3 yıl | 10 | 8,6 |
| | 3-4 yıl | 15 | 12,9 |
| | 4-5 yıl | 5 | 4,3 |
| | 5 yıl ve üzeri | 76 | 65,5 |
| Manuel yazılım test tecrübeniz oldu mu? Olduysa ne kadar süreliğine oldu? | Evet yaklaşık 6 ay kadar | 12 | 10,5 |
| | Evet yaklaşık 1 sene kadar | 12 | 10,5 |
| | Evet yaklaşık 2 seneden fazla | 29 | 25,4 |
| | Hayır hiç test tecrübem olmadı | 61 | 53,5 |
| | Diğer | 0 | 0,0 |
| Otomasyon yazılım test tecrübeniz oldu mu? Otomasyon test tecrübeniz ne kadar süreliğine oldu? | Evet yaklaşık 6 ay kadar | 3 | 2,7 |
| | Evet yaklaşık 1 sene kadar | 17 | 15,5 |
| | Evet yaklaşık 2 seneden fazla | 8 | 7,3 |
| | Hayır hiç test tecrübem olmadı | 82 | 74,5 |
| | Diğer | 0 | 0,0 |

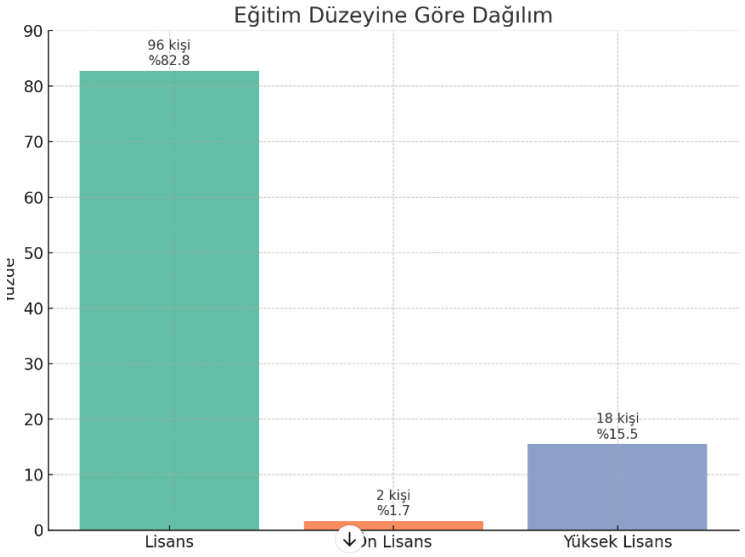
Çizelge 4.1. (Devamı) Demografik Bilgilere Yönelik Bulgular

| | | | |
|---|---|---|---|
| Şu an hangi sektör üzerine çalışıyorsunuz? | Banka | 35 | 33,3 |
| | Telekomünikasyon | 19 | 18,1 |
| | E-Ticaret | 23 | 21,9 |
| | İnsan Kaynakları | 0 | 0,0 |
| | Sağlık | 3 | 2,9 |

| | | | |
|---|---|---|---|
| | Diğer | 25 | 23,8 |




Grafik 4.1. Katılımcıların Cinsiyet ve Yaş Dağılımı




Grafik 4.2. Katılımcıların Eğitim Durumu Dağılımı

Çizelge 4.1'de görüldüğü üzere katılımcıların %35,3'ü (41 kişi) kadın, %64,7'si (75 kişi) erkektir. Yaş gruplarına bakıldığında, 30 yaş ve altı katılımcı sayısı 39 kişi (%33,6), 31-35 yaş aralığındaki katılımcı sayısı 51 kişi (%44,0), ve 36 yaş üstü katılımcı sayısı 26 kişi (%22,4) olmuştur. Eğitim düzeyine göre, lisans mezunları 96 kişi ile (%82,8) en büyük grubu oluştururken, ön lisans mezunları 2 kişi (%1,7) ve yüksek lisans mezunları 18 kişi (%15,5) ile temsil edilmiştir. Katılımcıların tamamı (%100, n=111) aktif olarak

çalışmakta, bunların %99,1'i (n=115) tam zamanlı olarak iş hayatına devam etmektedir. Çalışma şekline göre, %29,3'ü (n=34) uzaktan, %67,2'si (n=78) hibrit, ve %3,4'ü (n=4) ofiste çalışmaktadır. Coğrafi dağılım açısından İstanbul'da çalışanların %87,6'sı (n=99) bulunurken, Ankara'da %7,1 (n=8), İzmir'de %2,7 (n=3) ve diğer şehirlerde %2,7 (n=3) çalışmaktadır; İstanbul içinde %63,9'u (n=62) Anadolu, %36,1'i (n=35) Avrupa yakasında yer almaktadır. Meslek gruplarına göre, katılımcıların %6,0'ı (n=7) proje yöneticisi, %26,7'si (n=31) iş analisti, %24,1'i (n=28) yazılım geliştirici, %10,3'ü (n=12) teknik lider, %24,1'i (n=28) test kalite kontrol ve %5,2'si (n=6) ürün sahibidir. Ayrıca, %65,5'i (n=76) 5 yıl ve üzeri iş tecrübesine sahipken, %53,5'i (n=61) manuel test, %74,5'i (n=82) otomasyon test tecrübesine sahip değildir. Sektörel dağılım olarak, banka sektöründe çalışanların %33,3'ü (n=35), telekomünikasyon sektöründe %18,1'i (n=19), e-ticaret sektöründe %21,9'u (n=23) ve diğer sektörlerde %23,8'i (n=25) çalışmaktadır.

## 4.2. AGİLE METODOLOJİLERİ ÜZERİNE DETAYLI SORULAR

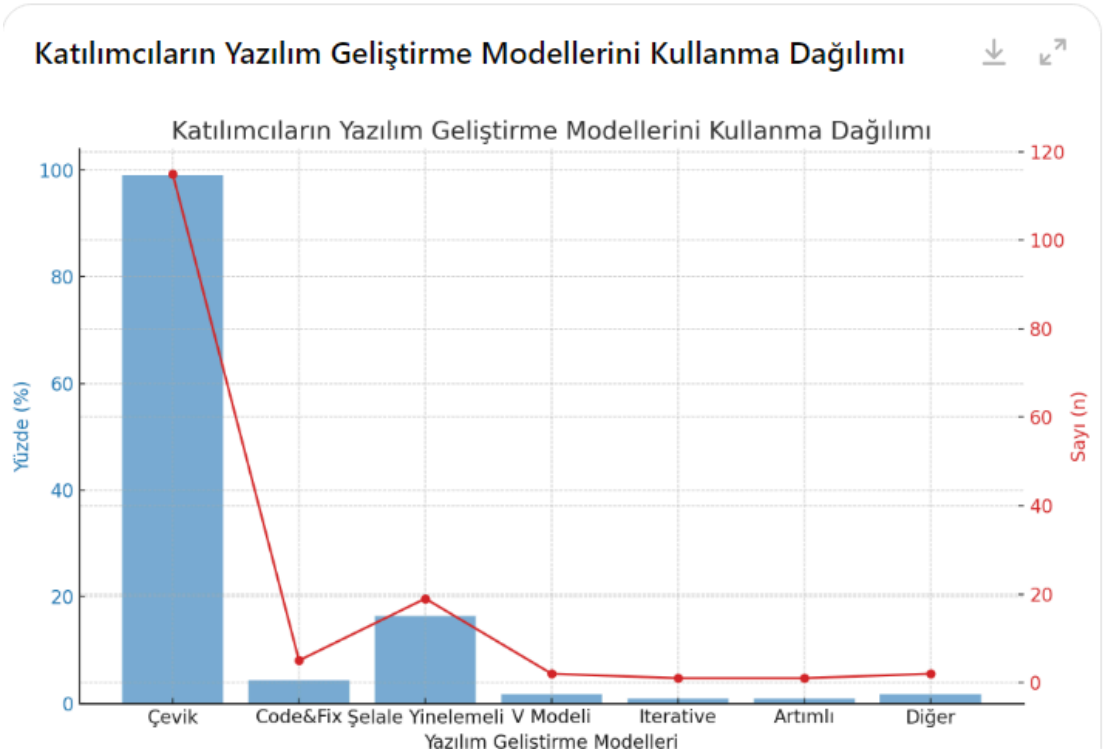
Çizelge 4.2. Yazılım Geliştirme Modellerinin Hangisinin Kullanıldığının Tespiti

| | | n | % |
|---|---|---|---|
| Yazılım geliştirme modellerinden hangilerini kullanıyorsunuz? | Agile (Çevik) Model | 115 | 99,1 |
| | Code&Fix Modeli | 5 | 4,3 |
| | Waterfall Iterative Model | 19 | 16,4 |
| | V Model | 2 | 1,7 |
| | Iterative Model | 1 | 0,9 |
| | Incremental Model | 1 | 0,9 |
| | Diğer | 2 | 1,7 |

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

Çizelge 4.2'de görüldüğü üzere Katılımcıların büyük bir çoğunluğu (%99,1, n=115) çevik modelini kullanmaktadır. Bunun yanı sıra, %4,3'ü (n=5) Code&Fix modelini, %16,4'ü (n=19) Şelale Yinelemeli modelini tercih etmektedir. V modelini kullananların oranı %1,7 (n=2), Iterative modelini kullananların oranı %0,9 (n=1), ve Artımlı modelini kullananların oranı da %0,9 (n=1) olarak belirlenmiştir. Ayrıca, %1,7'si (n=2) diğer yazılım geliştirme modellerini kullanmaktadır. 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 kanıtlarla desteklenmektedir (Chow ve Cao, 2008; Serrador ve Pinto, 2015; Bergmann, 2018; Carlos vd., 2020). Çevik modelin bu yaygın kabulü, projelerdeki başarı oranını artırmada önemli bir faktör olarak görülmektedir.




Grafik 4.3. Katılımcıların Yazılım Geliştirme Modellerini Kullanma Dağılımı

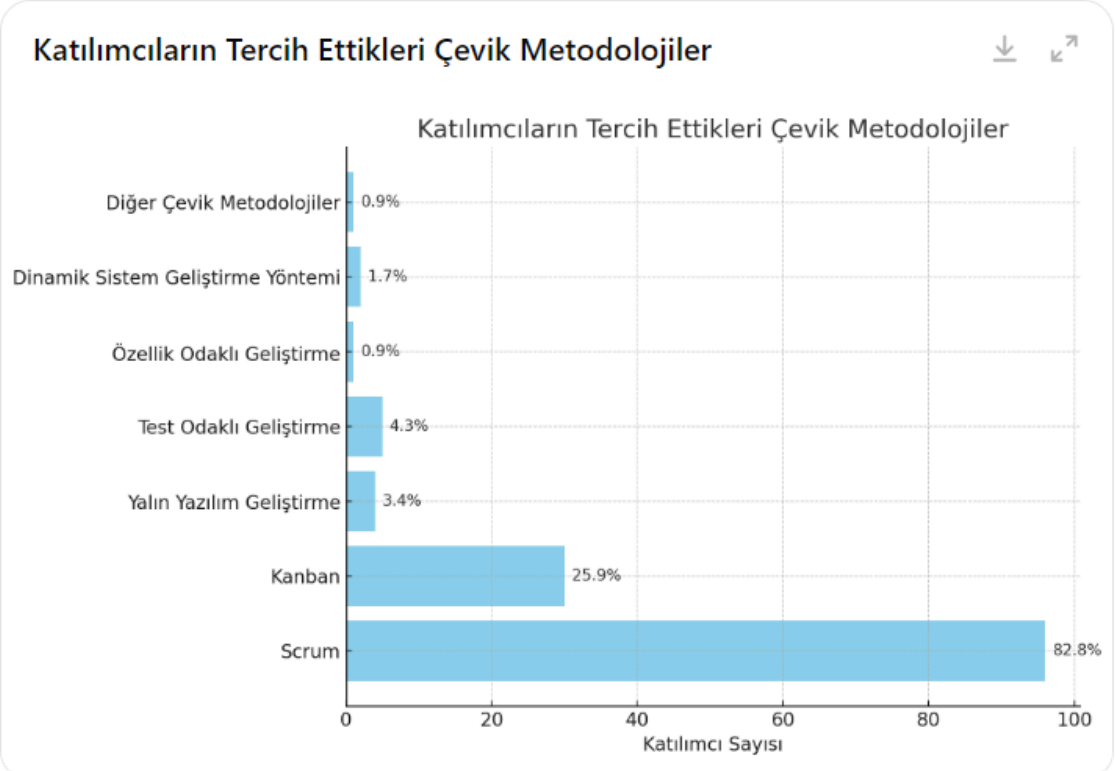
Çizelge 4.3. Şirkette Genel Olarak Hangi Çevik Metodolojilerinin Kullanıldığının Tespiti

| | | n | % |
|---|---|---|---|
| Şirkette genel olarak hangi çevik metodolojilerini kullanıyorsunuz? | Scrum | 96 | 82,8 |
| | Kanban | 30 | 25,9 |
| | Lean Software Development | 4 | 3,4 |
| | Test-Driven Development | 5 | 4,3 |

| | | | |
|---|---|---|---|
| | Feature Driven Development | 1 | 0,9 |
| | Dynamic Systems Development Method | 2 | 1,7 |
| | Diğer | 1 | 0,9 |

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

Çizelge 4.3'de görüldüğü üzere Katılımcıların büyük bir kısmı (%82,8, n=96) Scrum metodolojisini kullanmaktadır. Ayrıca, %25,9'u (n=30) Kanban metodolojisini, %3,4'ü (n=4) Yalın Yazılım Geliştirme metodolojisini, %4,3'ü (n=5) Test Odaklı Geliştirme metodolojisini tercih etmektedir. Özellik Odaklı Geliştirme metodolojisini kullananların oranı %0,9 (n=1), Dinamik Sistem Geliştirme Yöntemi kullananların oranı %1,7 (n=2) ve diğer çevik metodolojileri tercih edenlerin oranı da %0,9 (n=1) olarak belirlenmiştir. Bu veriler, katılımcıların çeşitli çevik metodolojilere olan yatkınlıklarını ve tercihlerini yansıtmaktadır.




Grafik 4.4. Katılımcıların Tercih Ettikleri Çevik Metodolojiler

Çizelge 4.4. Çalışılan Projede Ağırlıklı Olarak Kullanılan çevik Metodolojisinin Tespiti

| | n | % |
|---|---|---|

| | | | |
|---|---|---|---|
| Çalıştığınız projede ağırlıklı olarak hangi çevik metodolojisini kullanıyorsunuz? | Scrum | 90 | 77,6 |
| | Kanban | 8 | 6,9 |
| | Lean Software Development | 4 | 3,4 |
| | Test-Driven Development | 5 | 4,3 |
| | Feature Driven Development | 1 | 0,9 |
| Ekibiniz kaç kişiden oluşuyor? | 0-3 | 0 | 0,0 |
| | 3-10 | 69 | 59,5 |
| | 10 ve üzeri | 47 | 40,5 |

Çizelge 4.4'de görüldüğü üzere Çalışılan projelerde ağırlıklı olarak kullanılan çevik metodolojileri arasında %77,6 (n=90) ile Scrum öne çıkmakta, bunu %6,9 (n=8) ile Kanban, %3,4 (n=4) ile Yalın Yazılım Geliştirme, %4,3 (n=5) ile Test Odaklı Geliştirme ve %0,9 (n=1) ile Özellik Odaklı Geliştirme izlemektedir. Projelerdeki ekip büyüklüklerine bakıldığında, katılımcıların %59,5'i (n=69) 3-10 kişilik ekiplerde, %40,5'i (n=47) ise 10 ve üzeri kişilik ekiplerde çalışmaktadır. 0-3 kişilik ekiplerde çalışan katılımcı bulunmamaktadır. Bu durum, katılımcıların genellikle orta veya büyük ölçekli ekiplerde ve çoğunlukla Scrum metodolojisini kullanarak çalıştıklarını göstermektedir.

## 4.3. SCRUM METODOLOJİSİ ÜZERİNE DETAYLI SORULAR

Çizelge 4.5. SCRUM Metodolojisi Üzerine Detaylı Sorular

| | | n | % |
|---|---|---|---|
| İş takibini yapabilmek için Scrum panosu kullanıyor musunuz? | Evet | 107 | 94,7 |
| | Hayır | 2 | 1,8 |
| | Bilmiyorum | 4 | 3,5 |

| | | | |
|---|---|---|---|
| | Diğer | 0 | 0,0 |
| Scrum panosu için hangi toolu kullanıyorsunuz? | Jira | 87 | 77,7 |
| | Miro | 2 | 1,8 |
| | Bilmiyorum | 4 | 3,6 |
| | Diğer | 19 | 17,0 |
| Scrum metodolojisi ile ilerlemekten memnun musunuz? | Evet | 106 | 94,6 |
| | Hayır | 6 | 5,4 |
| | Bilmiyorum | 0 | 0,0 |
| | Diğer | 0 | 0,0 |
| Ekibinizde Scrum Master var mı? | Evet | 97 | 83,6 |
| | Hayır | 19 | 16,4 |
| | Bilmiyorum | 0 | 0,0 |
| | Diğer | 0 | 0,0 |

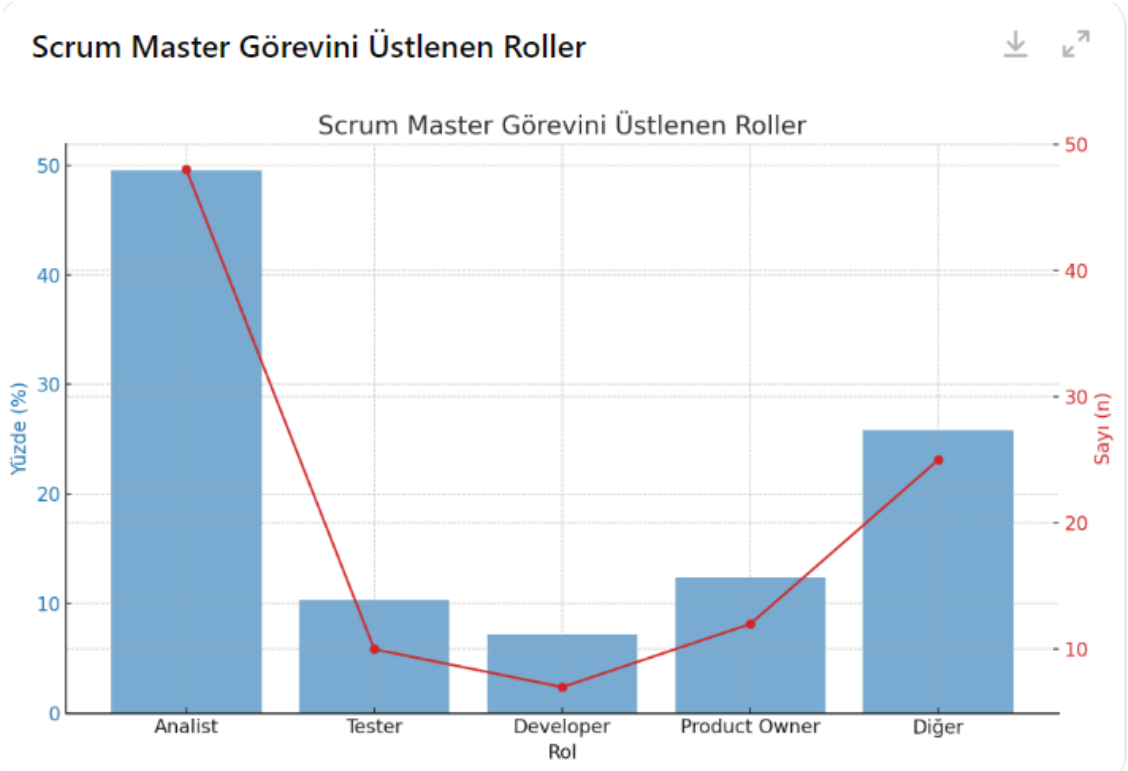
Çizelge 4.5’de görüldüğü üzere, katılımcıların %94,7'si (n=107) iş takibinde Scrum panosu kullandıklarını belirtmiş, %1,8'i (n=2) hayır, %3,5'i (n=4) ise bilmiyorum yanıtını vermiştir. Scrum panosu için kullanılan araçlara bakıldığında, %77,7'si (n=87) Jira, %1,8'i (n=2) Miro kullanırken, %17,0'i (n=19) diğer araçları tercih etmiş, %3,6'sı (n=4) ise bilmiyor. Scrum metodolojisiyle ilerleme konusunda katılımcıların %94,6'sı (n=106) memnun olduğunu ifade ederken, %5,4'ü (n=6) memnun olmadığını belirtmiştir. Ayrıca, ekibinde Scrum Master bulunanların oranı %83,6 (n=97), bulunmayanların oranı ise %16,4 (n=19) olarak kaydedilmiştir. Bu bulgular, Scrum metodolojisinin ve araçlarının yüksek bir kabul ve memnuniyet oranına sahip olduğunu göstermektedir.

Çizelge 4.6. Ekibinde Scrum Master Görevini Hangi Rolün Üstlendiği

| | | n | % |
|---|---|---|---|
| Ekibinizde Scrum Master görevini hangi rol üstleniyor? | Analist | 48 | 49,5 |

| | | | |
|---|---|---|---|
| | Tester | 10 | 10,3 |
| | Developer | 7 | 7,2 |
| | Product Owner | 12 | 12,4 |
| | Diğer | 25 | 25,8 |

Çizelge 4.6'da görüldüğü üzere ekibinde Scrum Master görevini hangi rolün üstlendiğine dair yapılan incelemede, %49,5 (n=48) oranla en yaygın olarak analistlerin bu görevi üstlendiği görülmüştür. Ayrıca, tester rolünde olanlar %10,3 (n=10), developerlar %7,2 (n=7), ve product ownerlar %12,4 (n=12) oranında Scrum Master görevini üstlenmektedirler. Diğer çeşitli rollerdeki katılımcılar ise bu görevi %25,8 (n=25) oranında üstlenmektedir. Bu dağılım, Scrum Master rolünün çeşitli iş pozisyonları tarafından nasıl paylaşıldığını göstermektedir.



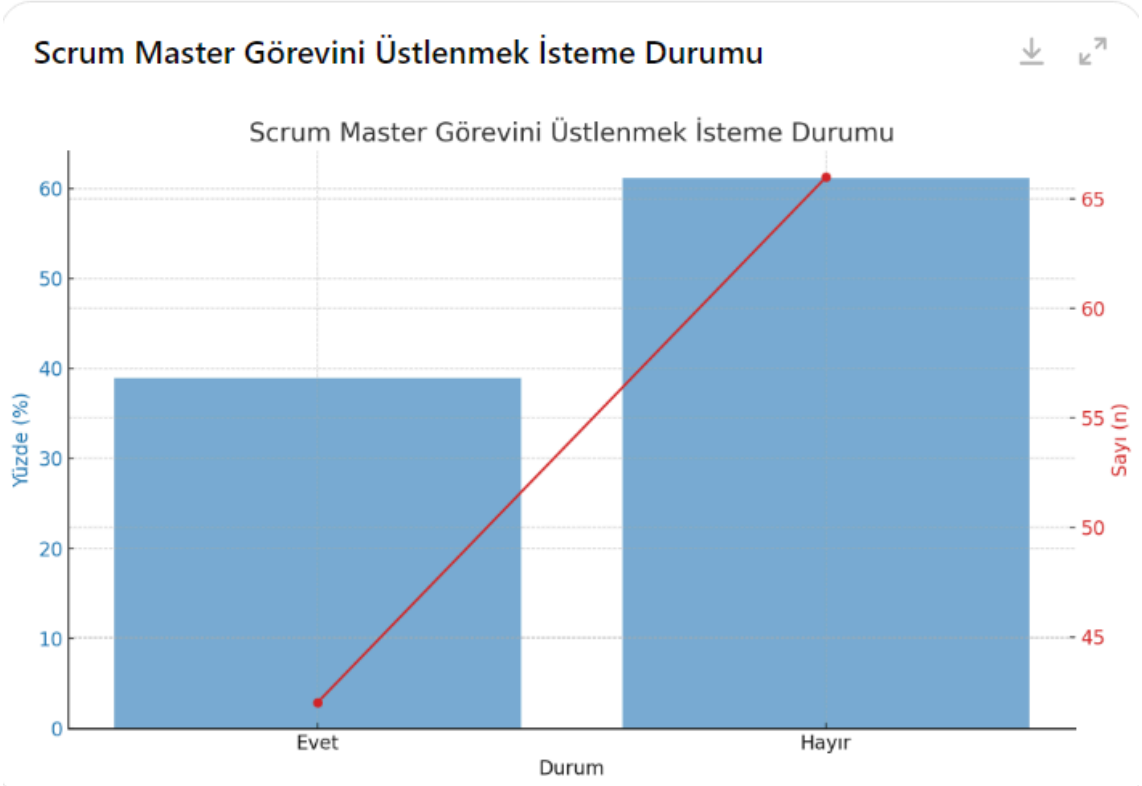

Grafik 4.5. Scrum Master Görevini Üstlenen Roller

Çizelge 4.7. Ekibindeki Scrum Master Görevini Üstlenmek İsteyip İstememe Durumu

| | | n | % |
|---|---|---|---|
| Ekibinizdeki Scrum Master görevini üstlenmek ister miydiniz? | Evet | 42 | 38,9 |

| | | | |
|---|---|---|---|
| | Hayır | 66 | 61,1 |
| | Bilmiyorum | 0 | 0,0 |
| | Diğer | 0 | 0,0 |

Çizelge 4.7'de görüldüğü üzere ekibindeki Scrum Master görevini üstlenmek isteyip istemedikleri sorulduğunda, katılımcıların %38,9'u (n=42) bu görevi üstlenmek istediklerini belirtmişken, %61,1'i (n=66) ise hayır yanıtını tercih etmiştir. Ankete katılanlar arasında bilgi sahibi olmayan veya diğer bir seçenek tercih eden katılımcı bulunmamaktadır. Bu sonuçlar, ekibindeki Scrum Master rolüne karşı genel bir isteksizliği veya bu role uygun olmadıklarını düşünen bir çoğunluğu göstermektedir.




Grafik 4.6. Scrum Master Görevini Üstlenmeyi İsteme Durumu

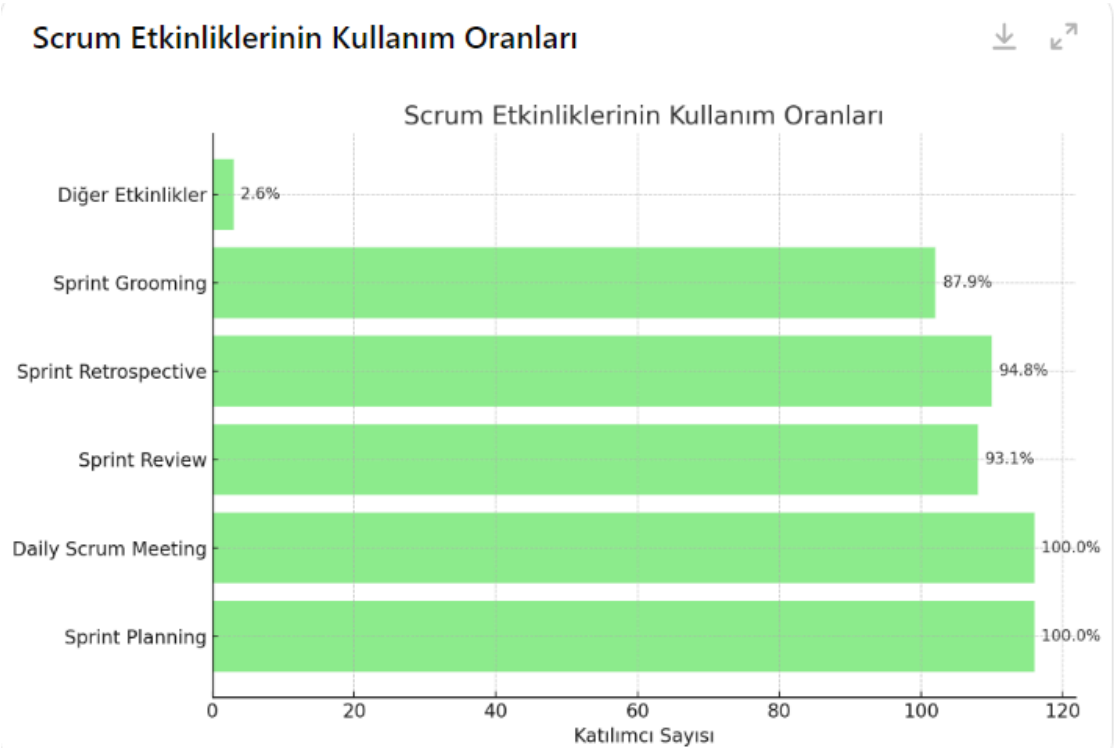
Çizelge 4.8. Projede Kullanılan Scrum Etkinlikleri

| | | n | % |
|---|---|---|---|
| Projenizde kullanılan Scrum etkinlikleri hangileridir? | Sprint Planing | 116 | 100,0 |
| | Daily Scrum Meeting | 116 | 100,0 |

| | | | |
|---|---|---|---|
| | Sprint Review | 108 | 93,1 |
| | Sprint Retro | 110 | 94,8 |
| | Sprint Grooming (Product Backlog Refinement) | 102 | 87,9 |
| | Diğer | 3 | 2,6 |

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

Çizelge 4.8'de görüldüğü üzere projede kullanılan Scrum etkinlikleri incelendiğinde, Sprint Planing ve Daily Scrum Meeting etkinlikleri katılımcıların tamamı tarafından (%100,0) kullanıldığı görülmektedir ve bu 116 kişiyi kapsamaktadır. Sprint Review etkinliği %93,1'lik bir oranla kullanılmakta olup, 108 kişiye ulaşırken, Sprint Retro etkinliği %94,8'lik bir oranla kullanılmakta ve 110 kişiyi temsil etmektedir. Sprint Grooming (Product Backlog Refinement) etkinliği ise %87,9'la 102 kişi tarafından kullanılmaktadır. Diğer etkinlikler ise sadece %2,6 oranında kullanılmakta ve 3 kişiyi temsil etmektedir. Bu sonuçlar, Scrum metodolojisinde temel etkinliklerin projelerde geniş kabul gördüğünü ve yoğun olarak kullanıldığını göstermektedir. Ancak Sprint Review, Retro ve Grooming gibi etkinliklerde tam katılım olmaması, belki de bu etkinliklerin proje gereksinimlerine veya takım dinamiklerine tam olarak uymadığını veya bu etkinliklerin anlam ve öneminin tam olarak anlaşılmadığını işaret edebilir. Diğer etkinliklerin düşük kullanım oranı ise, bu etkinliklerin daha az bilinen veya projelerin özel ihtiyaçlarına daha az hitap eden etkinlikler olabileceğini düşündürebilir.




Grafik 4.7. Scrum Etkinliklerinin Kullanım Oranları

Çizelge 4.9. Sprint Sürelerinin İncelenmesi

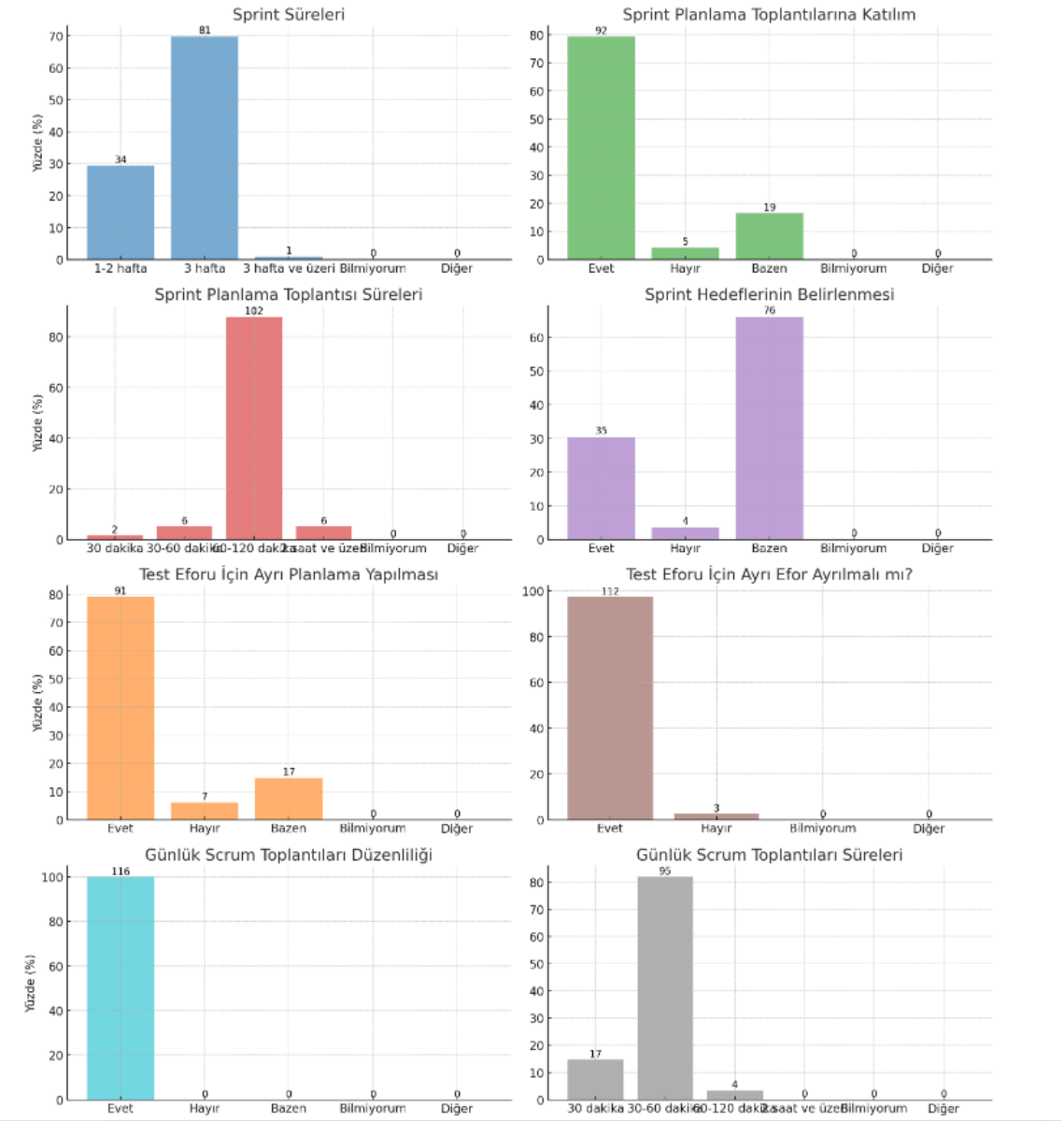
| | | n | % |
|---|---|---|---|
| Sprint süreleriniz ne kadar? | 1-2 hafta | 34 | 29,3 |
| | 3 hafta | 81 | 69,8 |
| | 3 hafta ve üzeri | 1 | 0,9 |
| | Bilmiyorum, fikrim yok. | 0 | 0,0 |
| | Diğer | 0 | 0,0 |

Çizelge 4.9. (Devamı) Sprint Sürelerinin İncelenmesi

| | | | |
|---|---|---|---|
| Sprint planlama toplantılarına tüm ekip katılım sağlıyor mu? | Evet | 92 | 79,3 |
| | Hayır | 5 | 4,3 |
| | Bazen tüm ekip katılım sağlıyor bazen sağlayamıyor | 19 | 16,4 |
| | Bilmiyorum, fikrim yok. | 0 | 0,0 |
| | Diğer | 0 | 0,0 |
| Sprint planlama toplantılarınız ne kadar sürüyor? | 30 dakika | 2 | 1,7 |

| | | | |
|---|---|---|---|
| | 30-60 dakika | 6 | 5,2 |
| | 60-120 dakika | 102 | 87,9 |
| | 2 saat ve üzeri | 6 | 5,2 |
| | Bilmiyorum, fikrim yok. | 0 | 0,0 |
| | Diğer | 0 | 0,0 |
| Sprint planlama toplantılarında ilgili sprint için sprint hedefleri belirliyor musunuz? | Evet | 35 | 30,4 |
| | Hayır | 4 | 3,5 |
| | Bazen belirleniyor bazen belirlenmiyor | 76 | 66,1 |
| | Bilmiyorum, fikrim yok | 0 | 0,0 |
| | Diğer | 0 | 0,0 |
| Sprint planlama toplantılarında görev planlaması yaparken test eforu için de ayrı bir planlama yapıyor musunuz? | Evet | 91 | 79,1 |
| | Hayır | 7 | 6,1 |
| | Bazen belirleniyor bazen belirlenmiyor | 17 | 14,8 |
| | Bilmiyorum, fikrim yok | 0 | 0,0 |
| | Diğer | 0 | 0,0 |
| Sizce sprint planlama toplantılarında görev planlaması yaparken test eforu için de ayrı bir efor verilmeli mi? | Evet | 112 | 97,4 |
| | Hayır | 3 | 2,6 |
| | Bilmiyorum | 0 | 0,0 |
| | Diğer | 0 | 0,0 |
| Günlük Scrum toplantıları (Dailyler) her gün düzenli olarak yapılıyor mu? | Evet | 116 | 100,0 |
| | Hayır | 0 | 0,0 |
| | Bazen belirleniyor bazen belirlenmiyor | 0 | 0,0 |
| | Bilmiyorum, fikrim yok | 0 | 0,0 |
| | Diğer | 0 | 0,0 |
| Günlük Scrum toplantılarında (daily) genel olarak ne kadar sürmektedir? | 30 dakika | 17 | 14,7 |
| | 30-60 dakika | 95 | 81,9 |
| | 60-120 dakika | 4 | 3,4 |
| | 2 saat ve üzeri | 0 | 0,0 |

| | | | |
|---|---|---|---|
| | Bilmiyorum, fikrim yok. | 0 | 0,0 |
| | Diğer | 0 | 0,0 |




Grafik 4.8. Sprint Sürelerinin İncelenmesi

Çizelge 4.9'da görüldüğü üzere Sprint sürelerinin incelenmesi sonucunda, katılımcıların %29,3'ü (n=34) 1-2 haftalık, %69,8'i (n=81) 3 haftalık ve %0,9'u (n=1) ise 3 hafta ve üzeri sürelerde sprintler düzenlediklerini belirtmişlerdir. Ekip katılımı açısından, %79,3'ü (n=92) tüm ekip sprint planlama toplantılarına katılırken, %16,4'ü (n=19) bazen tam katılım sağlanabiliyor, %4,3'ü (n=5) ise hiç katılmıyor. Sprint planlama toplantılarının süreleri genellikle %87,9 (n=102) tarafından 60-120 dakika olarak

sürdürülmektedir. Sprint hedeflerinin belirlenmesi konusunda, %66,1'i (n=76) bazen hedeflerin belirlendiğini, %30,4'ü (n=35) ise sürekli hedeflerin belirlendiğini, %3,5'i (n=4) ise hiç hedef belirlenmediğini ifade etmiştir. Görev planlaması sırasında test eforunun ayrıca planlanması gerektiğine %79,1'i (n=91) evet derken, %14,8'i (n=17) bazen, %6,1'i (n=7) ise hayır demiştir. Test eforu için ayrı bir efor ayrılması gerektiğine %97,4'ü (n=112) katılırken, sadece %2,6'sı (n=3) bu görüşe katılmamaktadır. Günlük Scrum toplantıları ise her gün %100 (n=116) katılım ile düzenli olarak yapılmakta ve bu toplantıların süreleri genellikle %81,9 (n=95) tarafından 30-60 dakika olarak belirlenmiştir. Bu bulgular, Scrum uygulamalarının genel kabul gördüğünü ve etkin bir şekilde kullanıldığını gösterirken, süreklilik ve tutarlılık konusunda bazı alanlarda iyileştirme gerektiğini de işaret etmektedir.

Çizelge 4.10. Günlük Scrum Toplantıları

| | | n | % |
|---|---|---|---|
| Günlük Scrum toplantılarında (daily) genel olarak aşağıdaki sorulardan hangilerini cevaplıyorsunuz? | Dün ne yaptım bugün ne yapacağım önümde engel var mı? | 115 | 99,1 |
| | “Son toplantıdan sonra ne yaptım?” | 96 | 82,8 |
| | “Bir sonraki toplantıya kadar ne yapmayı planlıyorum?" | 99 | 85,3 |
| | Konular genelde dağılıyor | 2 | 1,7 |

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

Çizelge 4.10’da görüldüğü üzere günlük Scrum toplantılarında katılımcıların %99,1'i genellikle "Dün ne yaptım, bugün ne yapacağım, önümde engel var mı?" sorularını cevaplamaktadır. Ayrıca, "Son toplantıdan sonra ne yaptım?" sorusu %82,8 oranında ve "Bir sonraki toplantıya kadar ne yapmayı planlıyorum?" sorusu ise %85,3 oranında cevaplanmaktadır. Bu, günlük toplantıların, ekip üyelerinin günlük işlerini düzenlemeleri ve engelleri belirtmeleri açısından etkin bir iletişim platformu olarak işlev gördüğünü göstermektedir. Ayrıca, toplantılarda konuların genellikle dağılmadığı (%1,7) ve odaklanmış bir şekilde ilerlediği belirtilmiştir. Bu durum, günlük Scrum toplantılarının amacına uygun şekilde yürütüldüğünü ve ekiplerin bu toplantılardan maksimum faydayı sağladığını işaret eder, bu da projelerin verimliliğini artırmada önemli bir rol oynamaktadır.

Çizelge 4.11. Sprint Sonlarında Düzenlenen Sprint Review Toplantıları

| | | n | % |
|---|---|---|---|
| Sprint sonlarında sprint review toplantısı yapıyor musunuz? | Evet | 97 | 84,3 |
| | Hayır | 12 | 10,4 |
| | Bazen belirleniyor bazen belirlenmiyor | 6 | 5,2 |
| | Bilmiyorum, fikrim yok | 0 | 0,0 |
| | Diğer | 0 | 0,0 |

Çizelge 4.11. (Devamı) Sprint Sonlarında Düzenlenen Sprint Review Toplantıları

| | | | |
|---|---|---|---|
| Sprint review toplantısı genel olarak ne kadar sürüyor? | 30 dakika | 0 | 0,0 |
| | 30-60 dakika | 4 | 4,0 |
| | 60-120 dakika | 27 | 27,3 |
| | 2 saat ve üzeri | 68 | 68,7 |
| | Bilmiyorum, fikrim yok. | 0 | 0,0 |
| | Diğer | 0 | 0,0 |
| Sprint review toplantısında ürünün ilgili sprint içerisinde tamamlanan geliştirmeleri ile ilgili bir demo yapıyor musunuz? | Evet | 23 | 22,3 |
| | Hayır | 18 | 17,5 |
| | Bazen belirleniyor bazen belirlenmiyor | 57 | 55,3 |
| | Bilmiyorum, fikrim yok | 5 | 4,9 |
| | Diğer | 0 | 0,0 |

Çizelge 4.11'de görüldüğü üzere Sprint sonlarında düzenlenen sprint review toplantılarına katılımcıların %84,3'ü her zaman katıldığını, %10,4'ü ise katılmadığını belirtirken, %5,2'si bazen katıldığını bazen katılamadığını ifade etmiştir. Toplantı sürelerine gelince, %68,7'si toplantıların genellikle 2 saat ve üzeri sürdüğünü, %27,3'ü 60-120 dakika, %4,0'ı ise 30-60 dakika arasında sürdüğünü belirtmiştir. Sprint review toplantılarında ürünün ilgili sprint içerisinde tamamlanan geliştirmelerinin demostrasyonunun yapılıp yapılmadığı sorulduğunda, %55,3'ü bazen yapıldığını bazen

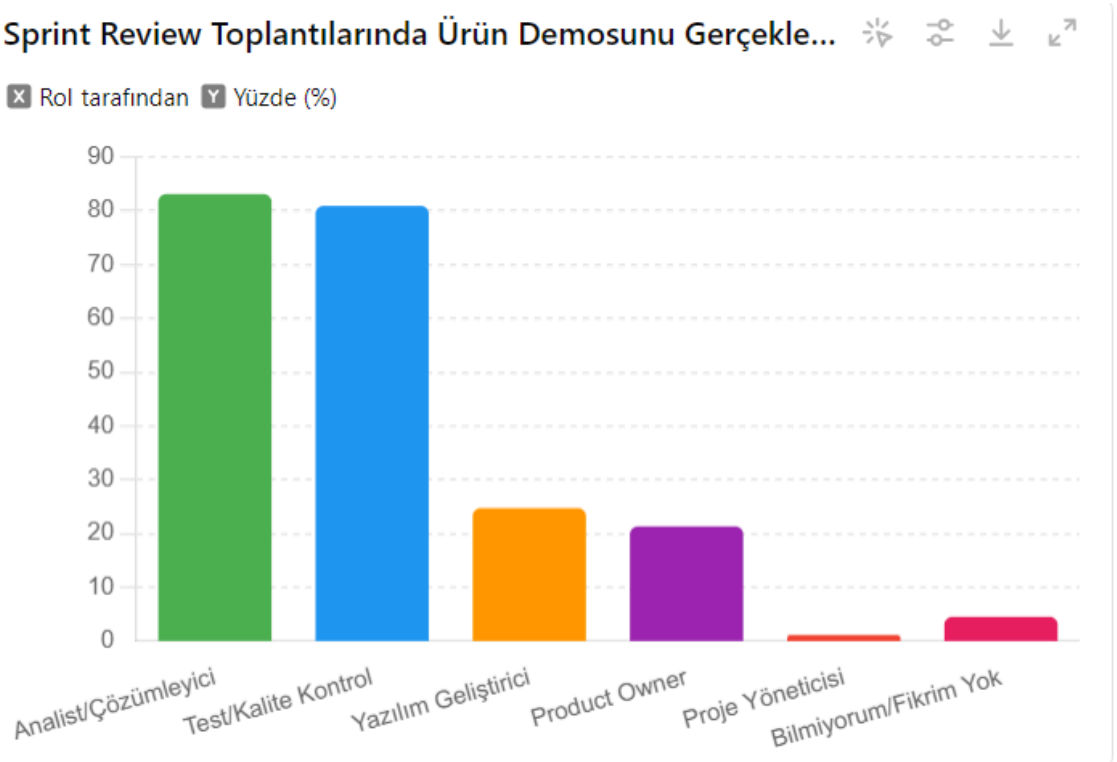
yapılmadığını, %22,3'ü her zaman yapıldığını, %17,5'i yapılmadığını ve %4,9'u ise bu konuda bilgi sahibi olmadığını ya da fikri olmadığını belirtmiştir. Bu veriler, sprint review toplantılarının önemli bir parçası olan ürün demostrasyonlarının tutarlı bir şekilde uygulanmadığını göstermektedir ve bu durum, sprint hedeflerine ulaşmada ve geri bildirim toplamada eksikliklere yol açabilir. Ayrıca, toplantı sürelerinin uzunluğu, bu toplantıların kapsamlı bir şekilde yürütüldüğünü ve önemli konuların ele alındığını göstermektedir.

Çizelge 4.12. Sprint Review Toplantılarında Ürün Demosunu Gerçekleştiren Takım Üyeleri

| | | n | % |
|---|---|---|---|
| Sprint review toplantılarında ürün demosunu hangi takım üyesi yapmaktadır? | Proje yöneticisi | 1 | 1,1 |
| | Analist/Çözümleyici | 74 | 83,1 |
| | Yazılım geliştirici | 22 | 24,7 |
| | Test/Kalite Kontrol | 72 | 80,9 |
| | Product Owner | 19 | 21,3 |
| | Bilmiyorum, fikrim yok | 4 | 4,5 |

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

Çizelge 4.12’de görüldüğü üzere Sprint review toplantılarında ürün demosunu gerçekleştiren takım üyelerine ilişkin yapılan ankette, katılımcıların %83,1'i analist/çözümleyicilerin, %80,9'u test/kalite kontrol ekibinin, %24,7'si yazılım geliştiricilerin ve %21,3'ü product ownerların demo yaptığını belirtmiştir. Proje yöneticileri tarafından demo yapılma oranı ise oldukça düşük, %1,1 olarak kaydedilmiştir. Ayrıca, %4,5'lik bir kesim bu konuda bilgi sahibi olmadığını veya fikri olmadığını ifade etmiştir. Bu veriler, sprint review toplantılarında ürün demosunun genellikle çok disiplinli bir yaklaşımla, çeşitli roldeki ekip üyeleri tarafından yapıldığını göstermektedir. Analistler ve test ekibi özellikle bu süreçte aktif roller üstlenirken, yazılım geliştiricilerin ve product ownerların daha az oranda bu görevi yerine getirdiği anlaşılmaktadır. Bu durum, sprint review toplantılarının işbirlikçi ve kapsayıcı bir doğaya sahip olduğunu ve her bir rolün projenin başarısına katkıda bulunma potansiyeline sahip olduğunu ortaya koymaktadır.




Grafik 4.9. Sprint Review Toplantılarında Ürün Demosunu Gerçekleştiren Takım Üyeleri

Çizelge 4.13. Sprint Sonlarında Düzenlenen Sprint Retro Toplantıları

| | | n | % |
|---|---|---|---|
| Sprint sonlarında sprint retro toplantısı yapıyor musunuz? | Evet | 102 | 88,7 |
| | Hayır | 3 | 2,6 |
| | Bazen yapılıyor bazen yapılmıyor | 6 | 5,2 |
| | Bilmiyorum, fikrim yok | 4 | 3,5 |
| | Diğer | 0 | 0,0 |

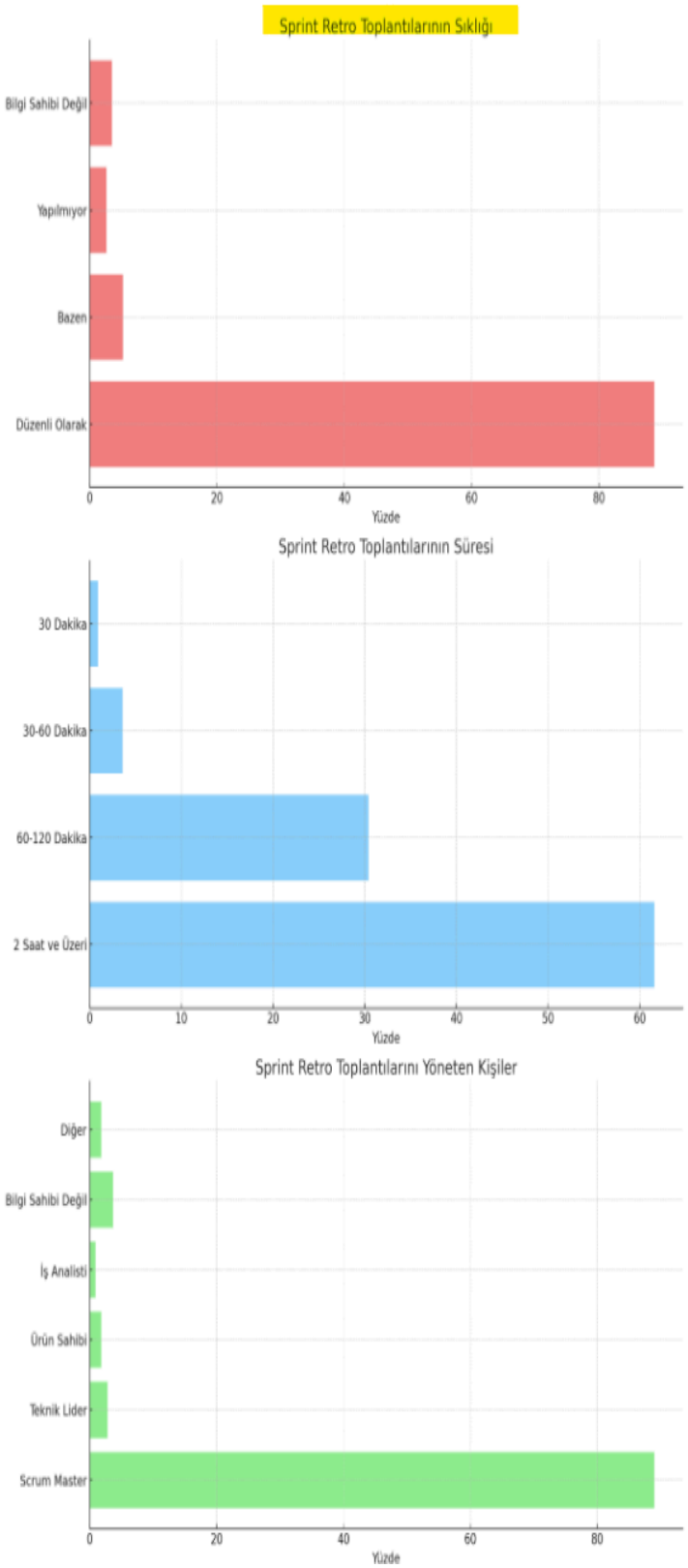
Çizelge 4.13.(Devamı) Sprint Sonlarında Düzenlenen Sprint Retro Toplantıları

| | | | |
|---|---|---|---|
| Sprint retro toplantısı genel olarak ne kadar sürüyor? | 30 dakika | 1 | 0,9 |
| | 30-60 dakika | 4 | 3,6 |
| | 60-120 dakika | 34 | 30,4 |
| | 2 saat ve üzeri | 69 | 61,6 |

| | | | |
|---|---|---|---|
| | Bilmiyorum, fikrim yok. | 4 | 3,6 |
| | Diğer | 0 | 0,0 |
| Sprint Retro toplantısını kim yönetiyor? | Scrum Master | 97 | 89,0 |
| | İş Analisti | 1 | 0,9 |
| | Teknik Analist | 0 | 0,0 |
| | Yazılım Geliştirici | 0 | 0,0 |
| | Test Kalite Kontrol | 0 | 0,0 |
| | Ürün sahibi | 2 | 1,8 |
| | Teknik Lider | 3 | 2,8 |
| | Bilmiyorum, fikrim yok | 4 | 3,7 |
| | Diğer | 2 | 1,8 |

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

Çizelge 4.13'te görüldüğü üzere Sprint sonlarında düzenlenen sprint retro toplantılarına katılımcıların %88,7'si düzenli olarak yapıldığını, %5,2'si bazen yapıldığını, %2,6'sı ise yapılmadığını belirtmiş; %3,5'i ise bu konuda bilgi sahibi olmadığını ifade etmiştir. Toplantı sürelerine bakıldığında, %61,6'sı toplantıların genellikle 2 saat ve üzeri sürdüğünü, %30,4'ü 60-120 dakika, %3,6'sı 30-60 dakika, ve %0,9'u 30 dakika sürdüğünü belirtmiştir. Sprint retro toplantılarını yöneten kişilere ilişkin olarak ise, %89,0'lık büyük bir çoğunluk Scrum Master'ı işaret etmiş, %2,8'i teknik lideri, %1,8'i ürün sahibini, ve %0,9'u iş analistini belirtmiştir. Bilgi sahibi olmayan veya fikri olmayanların oranı %3,7 iken, diğer yanıtları verenler %1,8 oranında olmuştur. Bu bulgular, sprint retro toplantılarının çoğunlukla Scrum Master tarafından yönetildiğini ve bu toplantıların projelerde önemli bir yere sahip olduğunu, ekiplerin süreçlerini değerlendirme ve iyileştirme fırsatı olarak gördüğünü göstermektedir. Bu toplantıların uzun süreler alması, derinlemesine değerlendirmeler ve tartışmalar yapıldığını işaret etmektedir.




Grafik 4.10. Sprint Sonlarında Düzenlenen Sprint Retro Toplantıları

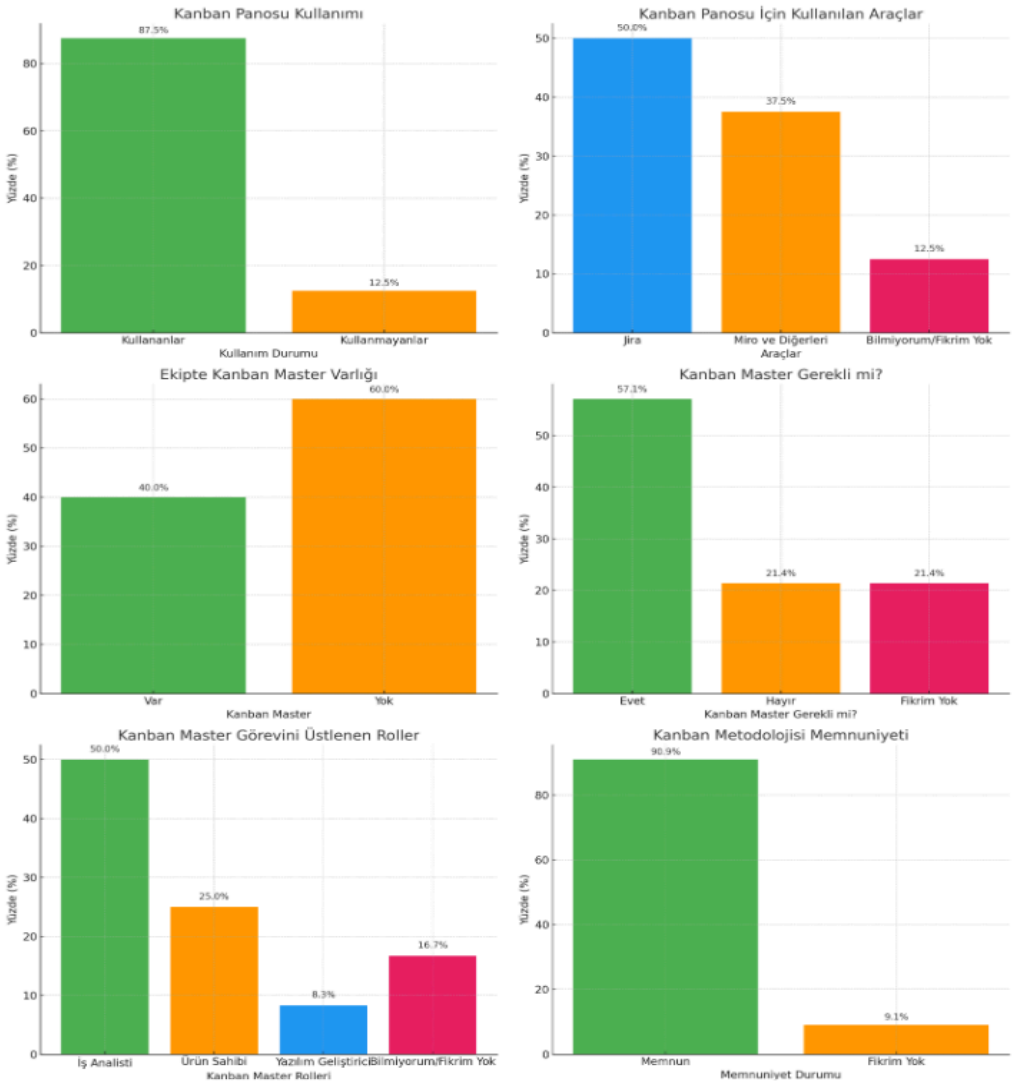
## 4.4. KANBAN METODOLOJİSİ ÜZERİNE DETAYLI SORULAR

Çizelge 4.14. KANBAN Metodolojisi Üzerine Detaylı Sorular

| | | n | % |
|---|---|---|---|
| İş takibini yapabilmek için Kanban panosu kullanıyor musunuz? | Evet | 14 | 87,5 |
| | Hayır | 2 | 12,5 |
| | Bilmiyorum , fikrim yok | 0 | 0,0 |
| | Diğer | 0 | 0,0 |
| Kanban panosu için hangi toolu kullanıyorsunuz? | Jira | 8 | 50,0 |
| | Miro | 0 | 0,0 |
| | Bilmiyorum, fikrim yok | 2 | 12,5 |
| | Diğer | 6 | 37,5 |
| Ekibinizde Kanban Master var mı? | Evet | 6 | 40,0 |
| | Hayır | 9 | 60,0 |
| Sizce ekibinizde Kanban Master olmalı mıdır? | Evet | 8 | 57,1 |
| | Hayır | 3 | 21,4 |
| | Bilmiyorum , fikrim yok | 3 | 21,4 |
| | Diğer | 0 | 0,0 |
| Ekibinizde Kanban Master görevini hangi rol üstleniyor? | İş analisti | 6 | 50,0 |
| | Teknik analist | 0 | 0,0 |
| | Ürün sahibi | 3 | 25,0 |
| | Yazılım Geliştirici | 1 | 8,3 |
| | Tester | 0 | 0,0 |
| | Bilmiyorum , fikrim yok | 2 | 16,7 |
| | Diğer | 0 | 0,0 |
| Kanban metodolojisi ile ilerlemekten memnun musunuz? Olumlu olumsuz yanları nelerdir? | Evet | 10 | 90,9 |

| | | | |
|---|---|---|---|
| | Hayır | 0 | 0,0 |
| | Bilmiyorum , fikrim yok | 1 | 9,1 |
| | Diğer | 0 | 0,0 |

Çizelge 4.14’de görüldüğü üzere katılımcıların %87,5'i iş takibinde Kanban panosu kullandıklarını belirtmiş, %12,5'i ise kullanmadıklarını ifade etmiştir. Kanban panosu için kullanılan araçlara gelince, %50,0'lık bir kesim Jira'yı tercih ederken, %37,5'i Miro gibi diğer araçları kullanmaktadır ve %12,5'i hangi aracı kullandıklarını bilmediklerini veya fikirlerinin olmadığını belirtmiştir. Ekibinde Kanban Master bulunanların oranı %40,0 iken, bulunmayanların oranı %60,0'dır. Katılımcıların %57,1'i ekibinde Kanban Master olması gerektiğine inanırken, %21,4'ü hayır ve %21,4'ü ise bu konuda bir fikri olmadığını belirtmiştir. Kanban Master görevini üstlenen roller arasında %50,0 ile iş analistleri, %25,0 ile ürün sahipleri öne çıkmış, %8,3'ü yazılım geliştiriciler ve %16,7'si bu konuda bilgi sahibi olmadığını belirtmiştir. Katılımcıların %90,9'u Kanban metodolojisiyle ilerlemekten memnun olduklarını ifade ederken, %9,1'i bu konuda bir fikre sahip olmadıklarını belirtmiştir. Bu bulgular, Kanban metodolojisinin yüksek bir kabul gördüğünü ve etkin bir iş takibi aracı olarak kullanıldığını göstermektedir. Ancak Kanban Master'ın varlığı ve rolleri konusunda farklı görüşler mevcut olup, bu role daha fazla dikkat çekilmesi ve rollerin netleştirilmesi gerekebileceğini işaret etmektedir.




Grafik 4.11. KANBAN Metodolojisi Üzerine Detaylı Sorular

### 4.4.1. **Projelerdeki Fonksiyonel Testler ve İşleyiş Süreçleri**

Çizelge 4.15. Projelerdeki Fonksiyonel Testler ve İşleyiş Süreçleri

| | | n | % |
|---|---|---|---|
| Projelerinizde manuel olarak fonksiyonel testler yapılıyor mu? | Evet | 114 | 98,3 |
| | Hayır | 2 | 1,7 |
| | Bilmiyorum , fikrim yok | 0 | 0,0 |
| | Diğer | 0 | 0,0 |
| Fonksiyonel yazılım testleri için destek aldığınız firmanızdan bağımsız bir test ekibiniz var mı? (Dış kaynak gibi) | Evet | 54 | 47,8 |

| | | | |
|---|---|---|---|
| | Hayır | 59 | 52,2 |
| | Bilmiyorum , fikrim yok | 0 | 0,0 |
| | Diğer | 0 | 0,0 |
| Projenizde analist var mı? | Evet | 62 | 98,4 |
| | Hayır | 1 | 1,6 |
| | Bilmiyorum , fikrim yok | 0 | 0,0 |
| | Diğer | 0 | 0,0 |

Çizelge 4.15. (Devamı) Projelerdeki Fonksiyonel Testler ve İşleyiş Süreçleri

| | | | |
|---|---|---|---|
| Analist yazmış olduğu analizin geliştirmesi tamamlanınca biten işi kendisi mi test ediyor ? | Evet | 26 | 38,8 |
| | Hayır | 26 | 38,8 |
| | Bilmiyorum, fikrim yok | 0 | 0,0 |
| | Diğer | 15 | 22,4 |
| Size göre analistler yazmış oldukları analizin geliştirmesi tamamlanınca testleri kendileri mi yapmalı yoksa test ekibinden testler için destek mi almalılar? | Analistler kendileri test yapmalı | 2 | 2,9 |
| | Test için test ekibinden destek almalı | 66 | 95,7 |
| | Bilmiyorum, fikrim yok | 0 | 0,0 |
| | Diğer | 1 | 1,4 |
| Developer kendi yazdığı kodun testini yapıyor mu? Biraz detaylarından bahsedebilir misiniz? | Evet | 58 | 84,1 |
| | Hayır | 11 | 15,9 |
| | Bilmiyorum, fikrim yok | 0 | 0,0 |
| | Diğer | 0 | 0,0 |
| Projenizdeki test süreçleri için dokümantasyon yapılıyor mu? | Evet | 52 | 76,5 |
| | Hayır | 15 | 22,1 |
| | Bilmiyorum, fikrim yok | 1 | 1,5 |
| | Diğer | 0 | 0,0 |

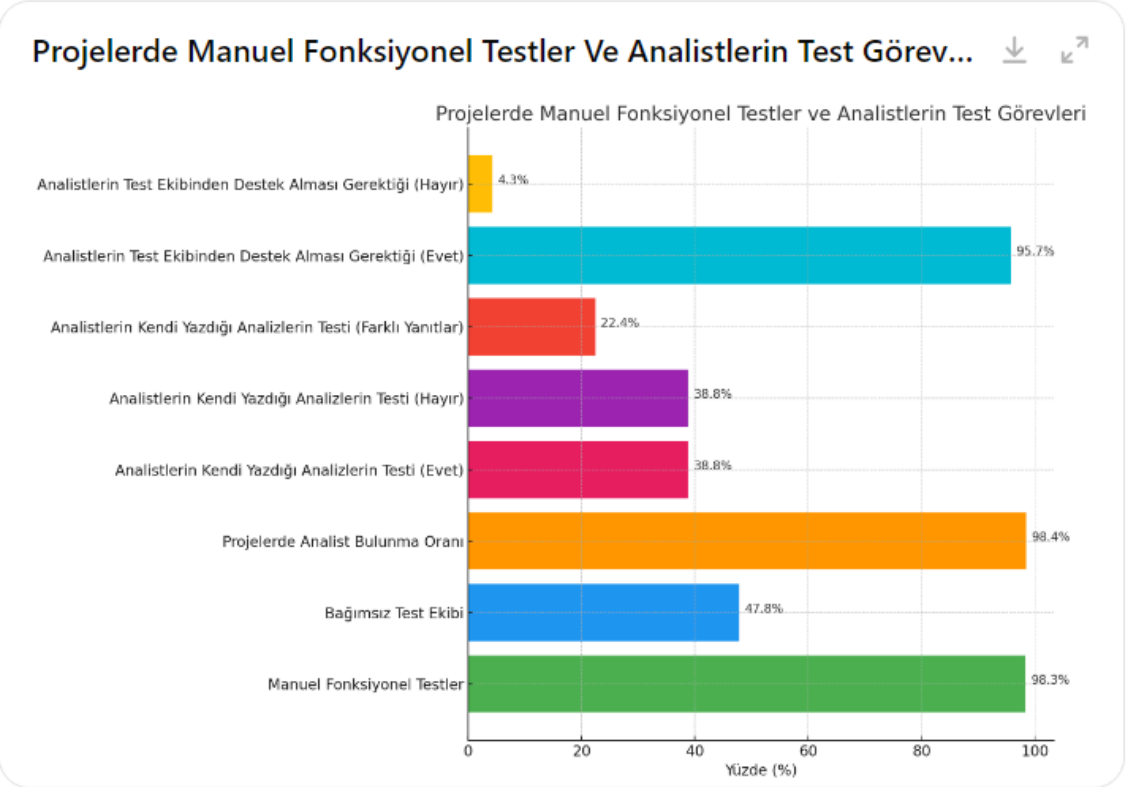
| | | | |
|---|---|---|---|
| Projenizde test senaryolarını yazmak için hangi toolu kullanılıyor? | Jira | 40 | 64,5 |
| | HP ALM | 5 | 8,1 |
| | Excel | 5 | 8,1 |
| | Bilmiyorum, fikrim yok | 6 | 9,7 |
| | Diğer | 6 | 9,7 |
| Yazılan test senaryoları review ediliyor mu? | Evet | 5 | 8,5 |
| | Hayır | 52 | 88,1 |
| | Bilmiyorum, fikrim yok | 2 | 3,4 |
| | Diğer | 0 | 0,0 |
| Test yapan kişiler testler esnasında karşılaştığı hata kayıtlarını tutabilmek için hangi toolu kullanıyorlar? | Jira | 44 | 67,7 |
| | HP ALM | 7 | 10,8 |
| | Excel | 3 | 4,6 |
| | E-mail | 3 | 4,6 |
| | Şirket içi sohbet iletişim aracı | 1 | 1,5 |
| | Bilmiyorum fikrim yok | 3 | 4,6 |
| | Diğer | 4 | 6,2 |
| Developer bu hata toolunu kullanabiliyor mu? | Evet | 51 | 96,2 |
| | Hayır | 1 | 1,9 |
| | Bilmiyorum, fikrim yok | 1 | 1,9 |
| | Diğer | 0 | 0,0 |
| Test ekibinin test yöneticisi var mı? | Evet | 78 | 76,5 |
| | Hayır | 24 | 23,5 |
| | Bilmiyorum, fikrim yok | 0 | 0,0 |
| | Diğer | 0 | 0,0 |
| Test ekibiniz kaç kişiden oluşuyor? | 0-2 | 6 | 5,8 |
| | 2-5 | 34 | 33,0 |

| | | | |
|---|---|---|---|
| | 5-10 | 53 | 51,5 |
| | Bilmiyorum, fikrim yok | 0 | 0,0 |
| | Diğer | 10 | 9,7 |
| Projenizin genelinde dokümantasyon için hangi araçlar kullanılmaktadır? | Jira | 31 | 29,8 |
| | Confluence | 42 | 40,4 |
| | Excel | 8 | 7,7 |
| | Bilmiyorum, fikrim yok | 0 | 0,0 |
| | Diğer | 23 | 22,1 |
| Projenizde kullanılan belirli bir doküman formatınız var mı? | Evet | 41 | 69,5 |
| | Hayır | 18 | 30,5 |
| | Bilmiyorum, fikrim yok | 0 | 0,0 |
| | Diğer | 0 | 0,0 |

Çizelge 4.15. (Devamı) Projelerdeki Fonksiyonel Testler ve İşleyiş Süreçleri

| | | | |
|---|---|---|---|
| Şirket içinde kullanılan standart doküman formatları varsa bunlar hangileridir? Genel olarak hangi isimlendirmeler kullanılmaktadır? | Fonksiyonel Analiz Dokümanı | 25 | 29,1 |
| | Gereksinim Analiz Dokümanı | 19 | 22,1 |
| | Teknik Analiz Dokümanı | 29 | 33,7 |
| | Bilmiyorum, fikrim yok. | 5 | 5,8 |
| | Diğer | 8 | 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.)




Grafik 4.12. Projelerdeki Fonksiyonel Testler ve İşleyiş Süreçleri

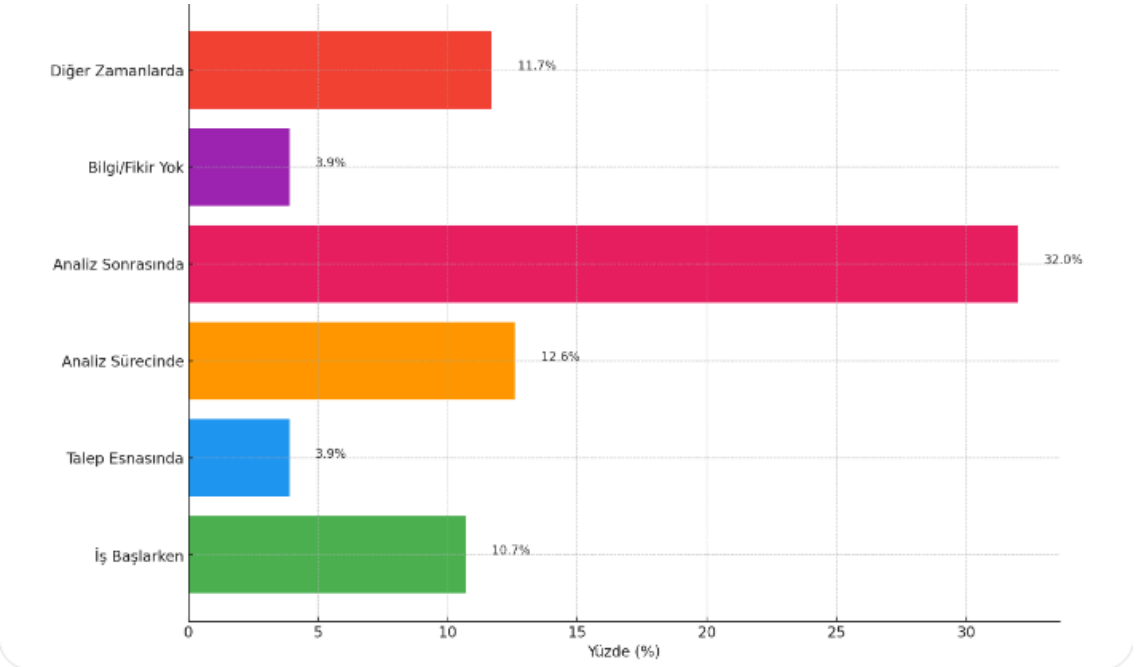
Çizelge 4.15'de görüldüğü üzere katılımcıların büyük bir çoğunluğu (%98,3) projelerinde manuel fonksiyonel testlerin yapıldığını belirtmişken, bu testler için bağımsız bir test ekibi bulunduranların oranı %47,8 olarak belirlenmiş, %52,2'si ise böyle bir ekibe sahip olmadıklarını ifade etmiştir. Projelerde analist bulunma oranı %98,4 gibi yüksek bir seviyede iken, analistlerin kendi yazdıkları analizlerin geliştirmesi tamamlandığında test işlemini kendilerinin yapması gerektiği konusunda görüşler bölünmüş; %38,8'i evet derken, %38,8'i hayır demiş ve %22,4'ü farklı yanıtlar vermiştir. Genel görüş olarak, %95,7'lik kesim analistlerin geliştirme sonrası testleri test ekibinden destek alarak yapmaları gerektiğini savunmuştur. Yazılım geliştiricilerin kendi kodlarını test etme oranı %84,1 olarak belirlenirken, projelerde test süreçleri için %76,5 oranında dokümantasyon yapıldığı, ancak yazılan test senaryolarının sadece %8,5 oranında review edildiği belirtilmiştir. Hata kayıtları için kullanılan araçlar arasında %67,7 ile Jira, %10,8 ile HP ALM öne çıkmış, developerların bu araçları kullanma oranı %96,2 olarak yüksek bir seviyede olmuştur. Test ekibi büyüklüğü çoğunlukla 2-10 kişi arasında değişmekte, test yöneticisi bulundurma oranı ise %76,5 olarak kaydedilmiştir. Dokümantasyon araçları olarak %29,8 ile Jira, %40,4 ile Confluence ve %22,1 ile diğer araçlar kullanılmakta; belirli doküman formatları ise

%69,5 oranında kullanılmakta ve bunlar arasında Fonksiyonel Analiz, Gereksinim Analiz ve Teknik Analiz Dokümanları öne çıkmaktadır. Bu bulgular, yazılım geliştirme ve test süreçlerinde yüksek düzeyde standartlaşma ve organizasyon olduğunu gösterirken, bazı alanlarda, özellikle test senaryolarının review edilmesi gibi, geliştirilmesi gereken yerler olduğunu da ortaya koymaktadır.

Çizelge 4.16. Katılımcıların Projelere Dahil Olma Zamanları

| | | n | % |
|---|---|---|---|
| Test ekibi işin hangi aşamasında işe dahil oluyor? | Yazılım Başlarken | 11 | 10,7 |
| | Talep Esnasında | 4 | 3,9 |
| | Analiz Edilirken | 13 | 12,6 |
| | Analiz Sonrası | 33 | 32,0 |
| | Bilmiyorum, fikrim yok | 4 | 3,9 |
| | Diğer | 12 | 11,7 |

Çizelge 4.16’da görüldüğü üzere katılımcıların projelere dahil olma zamanlarına ilişkin verilere göre, %10,7'si iş başlarken, %3,9'u talep esnasında, %12,6'sı analiz sürecinde, %32,0'si ise analiz sonrasında işe dahil olduklarını belirtmişlerdir. Bunun yanı sıra, %3,9'u bu konuda bilgi sahibi olmadıklarını veya fikirlerinin olmadığını ifade etmiş, %11,7'si ise projeye diğer zamanlarda katıldıklarını belirtmiştir. Bu veriler, çoğu katılımcının projenin daha ileri aşamalarında, özellikle analiz sonrasında dahil olduğunu göstermekte ve bu durum, projelerin başlangıcında yer almayan bireylerin projenin erken aşamalarında oluşan karar süreçlerinden ve yön belirlemelerden uzak kaldıklarını işaret edebilir. Bu, projenin ilerleyen safhalarında karşılaşılan zorlukların ve uyumsuzlukların bir nedeni olabilir, bu yüzden erken aşamalarda daha fazla katılımcının dahil edilmesi faydalı olabilir.



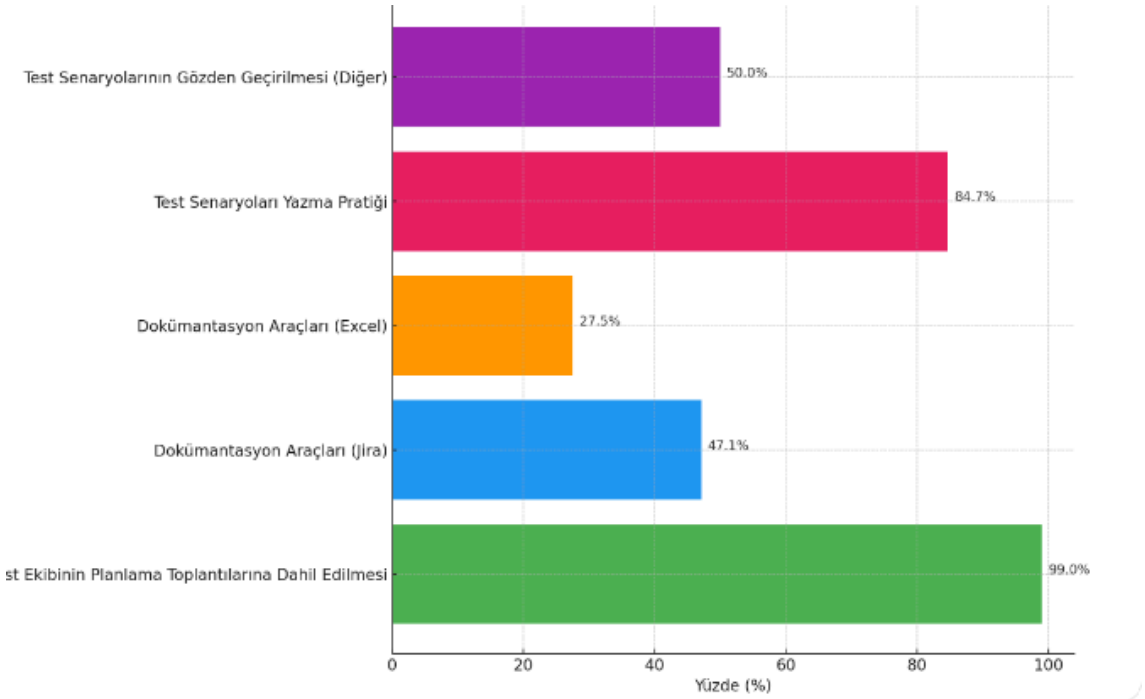

Grafik 4.13. Katılımcıların Projelere Dahil Olma Zamanları

Çizelge 4.17. Scrum Metodolojisini Kullanan Projeler

| | | n | % |
|---|---|---|---|
| Eğer Scrum çalışıyorsanız test ekibi planlama toplantılarına dahil ediliyor mu? | Evet | 101 | 99,0 |
| | Hayır | 1 | 1,0 |
| | Bilmiyorum, fikrim yok | 0 | 0,0 |
| | Diğer | 0 | 0,0 |
| Projenizdeki test süreçleri için dokümantasyon yapılıyor mu? Bu dokümantasyon için hangi araçlar kullanılmaktadır? | Jira | 24 | 47,1 |
| | HP ALM | 2 | 3,9 |
| | Excel | 14 | 27,5 |
| | Bilmiyorum, fikrim yok | 3 | 5,9 |
| | Diğer | 8 | 15,7 |
| Projenizde test senaryoları yazılıyor mu? | Evet | 50 | 84,7 |
| | Hayır | 2 | 3,4 |
| | Bilmiyorum, fikrim yok | 7 | 11,9 |

| | | | |
|---|---|---|---|
| | Diğer | 0 | 0,0 |
| Yazılan test senaryoları hangi rol tarafından review ediliyor? | Evet | 2 | 8,3 |
| | Hayır | 3 | 12,5 |
| | Bilmiyorum, fikrim yok | 7 | 29,2 |
| | Diğer | 12 | 50,0 |
| Şirket içi iletişim için hangi araç kullanılmaktadır? | Cisco Jabber | 14 | 13,3 |
| | Microsoft Teams | 53 | 50,5 |
| | Skype | 9 | 8,6 |
| | Slack | 18 | 17,1 |
| | Bilmiyorum, fikrim yok. | 0 | 0,0 |
| | Diğer | 11 | 10,5 |

Çizelge 4.17’de görüldüğü üzere Scrum metodolojisini kullanan projelerde, katılımcıların %99'u test ekibinin planlama toplantılarına genellikle dahil edildiğini doğrulamış, bu durum test süreçlerinin başlangıç aşamasından itibaren entegre edilmesinin önemini vurgulamaktadır. Projelerde test süreçleri için dokümantasyon yapılıyor ise, en yaygın kullanılan araçlar %47,1 ile Jira ve %27,5 ile Excel olarak belirlenmiştir, bu da çeşitli araçların test süreçlerindeki farklı ihtiyaçları karşıladığını göstermektedir. Test senaryoları yazma pratiği yüksek oranda (%84,7) uygulanmakta olup, ancak bu senaryoların gözden geçirilmesi konusunda belirsizlikler mevcut; katılımcıların %50'si bu durumu "Diğer" olarak tanımlamıştır, bu da revizyon süreçlerinin netliğinin artırılması gerektiğini işaret etmektedir. Şirket içi iletişim araçları olarak, %50,5 ile Microsoft Teams ve %17,1 ile Slack en yaygın kullanılanlar arasında yer alırken, Cisco Jabber ve Skype gibi diğer araçlar daha az tercih edilmektedir. Bu durum, modern iletişim araçlarının iş yerinde iletişim ve işbirliğini desteklemede nasıl bir rol oynadığını ve tercih edilen platformların projelerin ve ekiplerin ihtiyaçlarına nasıl uyum sağladığını göstermektedir.




Grafik 4.14. Scrum Projelerinin Test Durumları

Çizelge 4.18. Test Ekiplerinin Test Süreçleri Sırasında Karşılaştıkları Hataları Kaydetmek İçin Kullandıkları Araçlar

| | | n | % |
|---|---|---|---|
| Test ekibinin testler esnasında karşılaştığı hata kayıtlarını tutabilmek için hangi araçları kullanıyorlar? | Jira | 38 | 32,8 |
| | Excel | 11 | 9,5 |
| | Mail | 10 | 8,6 |
| | Şirket içi sohbet iletişim aracı | 4 | 3,4 |
| | Diğer | 5 | 4,3 |

Çizelge 4.18'de görüldüğü üzere Test ekipleri, test süreçleri sırasında karşılaştıkları hataları kaydetmek için çeşitli araçlar kullanmaktadır. En yaygın kullanılan araç %32,8 ile Jira iken, %9,5'lik bir oranla Excel de tercih edilmektedir. Ayrıca, %8,6 oranında katılımcılar hata kayıtlarını e-posta yoluyla tutarken, %3,4'ü şirket içi sohbet uygulamalarını kullanmayı tercih etmektedir. %4,3'lük bir kesim ise "Diğer" seçeneği altında toplanarak, standart dışı farklı araçlar kullanmayı seçmiştir. Bu çeşitlilik, farklı test ekiplerinin farklı ihtiyaç ve tercihlere sahip olabileceğini göstermektedir ve bu da hata kayıtlarının tutulmasında verimlilik ve erişilebilirlik açısından önemli bir faktör

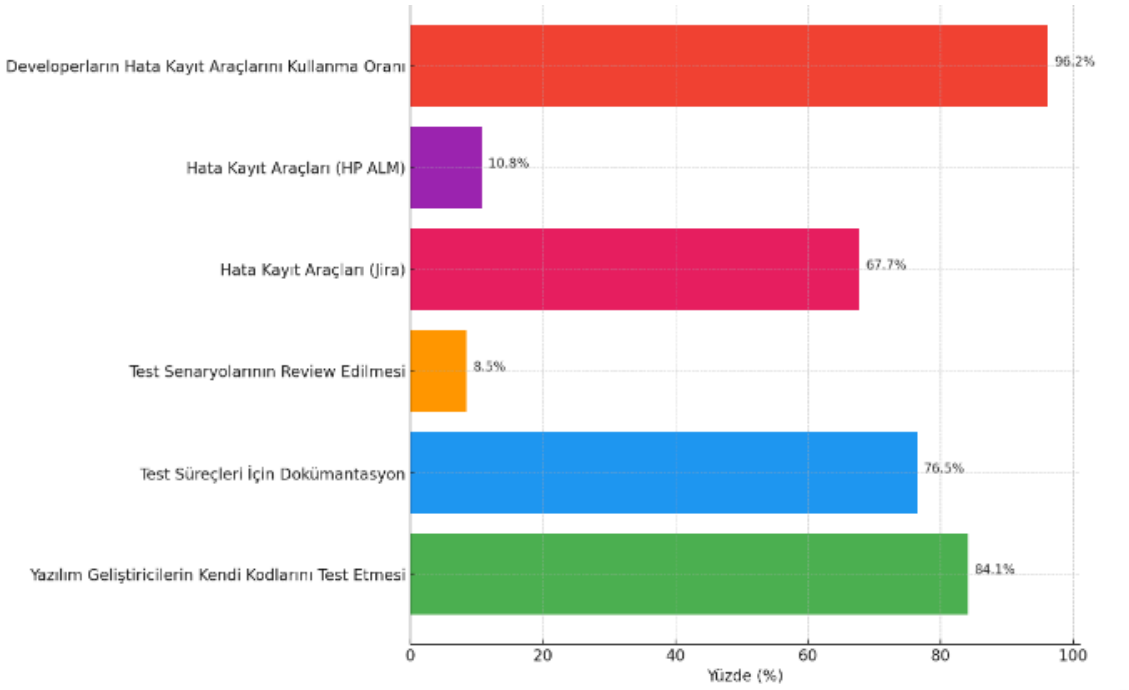
olabilir. Bu durum, araç seçiminin özellikle hata izleme süreçlerinde nasıl stratejik bir önem taşıdığını ve test süreçlerinin etkinliğini artırma potansiyeline sahip olduğunu vurgulamaktadır.

Çizelge 4.19. Yazılımcıların Kullandıkları Hata Araçları

| | | n | % |
|---|---|---|---|
| Yazılımcılar bu hata toolunu nasıl kullanıyor? | Hata detaylarına erişmek için kullanıyorlar | 41 | 35,3 |
| | Hatayı çözdükten sonra işin statüsünü ilerletiyorlar | 38 | 32,8 |
| | Developerlar genelde hata toolunu aktif olarak kullanamamaktadır. | 11 | 9,5 |
| | Developerlar genelde hatanın detayına erişmek için o hatayı açan kişi ile iletişime geçiyorlar. | 12 | 10,3 |

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

Çizelge 4.19'da görüldüğü üzere yazılımcılar genellikle hata detaylarına erişmek için hata araçlarını kullanmaktadır (%35,3), bu da hatayı anlamak ve çözmek için doğrudan bu araçlara başvurduklarını göstermektedir. Ayrıca, hataları çözdükten sonra işin statüsünü güncellemek amacıyla da bu araçları aktif olarak kullanmaktadırlar (%32,8). Ancak, bazı durumlarda (%9,5) geliştiricilerin hata araçlarını aktif olarak kullanamadıkları görülmekte, bu da belirli koşullar altında erişim veya kullanım zorluklarına işaret edebilir. Diğer bir yaklaşım olarak, %10,3 oranında geliştiriciler hatanın detaylarına erişmek için hata araçları yerine hatayı bildiren kişiyle doğrudan iletişime geçmeyi tercih etmektedir. Bu, 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 olduğunu, ancak geliştiricilerin zaman zaman daha kişisel ve doğrudan iletişim yöntemlerine başvurduklarını göstermektedir.




Grafik 4.15. Hata Kayıt Araçları

Çizelge 4.20. Test Ekibinin Performansını Ölçmek İçin Kullanılan Metrikler

| | | n | % |
|---|---|---|---|
| Test ekibinin performansında test metrikleri kullanılıyor mu? Projenizde bulunan test özelinde çalışan ekibin ya da kişinin performansı hangi metriklerle ölçülüyor? | Evet, kullanılıyor. Testerların yazdığın case sayısı | 14 | 12,1 |
| | Evet, kullanılıyor. Testerların açtığı bug sayısı | 9 | 7,8 |
| | Evet kullanılıyor. Canlı ortamdan dönen hata sayıları | 10 | 8,6 |
| | Hayır, kullanılmıyor | 80 | 69,0 |
| | Bilmiyorum, fikrim yok | 16 | 13,8 |
| | Diğer | 5 | 4,3 |

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

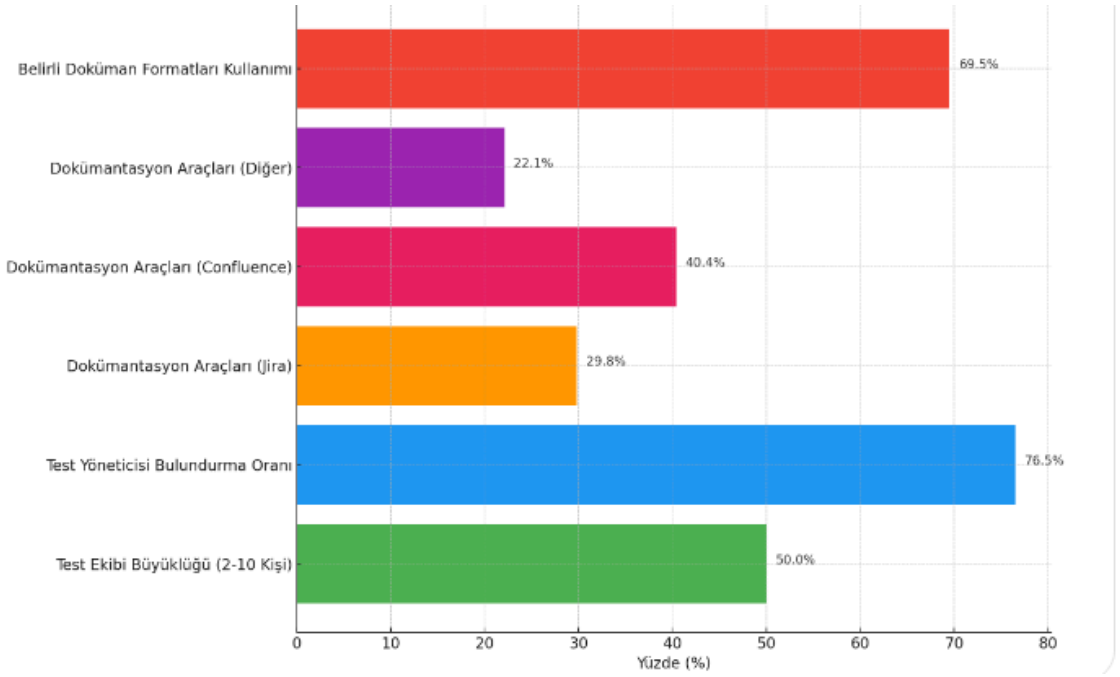
Çizelge 4.20'de görüldüğü üzere katılımcıların çoğunluğu (%69,0) test ekibinin performansını ölçmek için test metriklerinin kullanılmadığını belirtmiştir. Metriklerin kullanıldığı azınlık durumlarında ise, en yaygın ölçüm yöntemleri arasında "Testerların yazdığı test case sayısı" (%12,1), "Testerların açtığı hata sayısı" (%7,8) ve "Canlı ortamdan dönen hata sayısı" (%8,6) gibi metrikler bulunmaktadır. Bununla birlikte, %13,8'lik bir kesim test performansı ölçümü hakkında bilgi sahibi olmadıklarını ifade ederken, %4,3'ü "Diğer" seçeneğini işaretlemiştir ki bu durum, daha az bilinen veya özel metriklerin kullanıldığını düşündürebilir. 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 karşımıza çıkmaktadır.

Çizelge 4.21. Test Ekibinin Performans Değerlendirmeleri

| | | n | % |
|---|---|---|---|
| Test ekibinin performans değerlendirmelerini kim yapıyor? | Test Yöneticisi | 33 | 64,7 |
| | Proje Yöneticisi | 6 | 11,8 |
| | Yazılım Ekip Lideri | 2 | 3,9 |
| | Direktör | 2 | 3,9 |
| | Bilmiyorum, fikrim yok | 6 | 11,8 |
| | Diğer | 2 | 3,9 |

Çizelge 4.21'de görüldüğü üzere katılımcıların çoğunluğu (%64,7), test ekibinin performans değerlendirmelerinin "Test Yöneticisi" tarafından yapıldığını belirtmiştir. Ayrıca, performans değerlendirmelerini yapan diğer önemli roller arasında %11,8 ile "Proje Yöneticisi" ve yine %11,8 ile "Bilmiyorum, fikrim yok" cevapları dikkat çekmektedir. Bu, bazı katılımcıların kimin tarafından değerlendirildiklerinden habersiz olduğunu göstermektedir. "Yazılım Ekip Lideri" ve "Direktör" gibi rollerin performans değerlendirmeleri yapma oranları ise daha düşük olup, her biri %3,9'luk bir oranla daha az yaygındır. 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.




Grafik 4.16. Testler İle İlgili Genel Bilgiler

## 4.5. GENEL BULGULAR

Çizelge 4.22. Genel Bulgular

| | | n | % |
|---|---|---|---|
| Projede bulunan testerların, test dışında aldığı farklı sorumluluklar aşağıdakilerden hangileridir? | Scrum Master | 29 | 25,0 |
| | Analiz Destek | 12 | 10,3 |
| | Hayır, yok. | 72 | 62,1 |
| | Bilmiyorum, fikrim yok | 3 | 2,6 |
| | Diğer | 4 | 3,4 |

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

Çizelge 4.22'de görüldüğü üzere katılımcıların %10,3'ü (12 tester) analiz destek görevini, %25'ü (29 tester) Scrum Master rolünü üstlenmişken, %62,1'i (72 tester) herhangi bir ek sorumluluk almamıştır. Ayrıca, %2,6'sı (3 tester) performans değerlendirme ile ilgili soruya "Bilmiyorum veya fikrim yok" yanıtını vermiştir. Diğer görevler kategorisinde ise %3,4 (4 tester) yer almaktadır. Bu veriler, çoğu testerın ek görevlerden bağımsız olarak test odaklı çalıştığını, ancak önemli bir kesimin projelerde Scrum Master gibi kritik rolleri de üstlendiğini göstermektedir. Testerların projelerde çeşitli roller üstlenmesi, onların çok yönlülüğünü ve test süreçlerinin ötesinde projeye katkı sağlama kapasitelerini ortaya koymaktadır.

Çizelge 4.23. Projelerde Yapılan Test Türleri

| | | n | % |
|---|---|---|---|
| Projenizde smoke testler yapılıyor mu? | Evet | 16 | 23,5 |
| | Hayır | 45 | 66,2 |
| | Bilmiyorum, fikrim yok | 7 | 10,3 |
| | Diğer | 0 | 0,0 |
| Projenizde kullanıcı kabul testleri yapılıyor mu? | Evet | 101 | 88,6 |
| | Hayır | 9 | 7,9 |
| | Bilmiyorum, fikrim yok | 4 | 3,5 |
| | Diğer | 0 | 0,0 |

Çizelge 4.23'te görüldüğü üzere projelerde yapılan test türleri hakkında elde edilen bulgulara göre, smoke testlerin uygulanma durumu incelendiğinde, katılımcıların %23,5'i (16 test) bu testleri yaptıklarını belirtmişken, %66,2'si (45 test) smoke test yapmadıklarını, %10,3'ü (7 test) ise bu konuda bilgi sahibi olmadıklarını ifade etmiştir. Diğer kategorisinde katılımcı bulunmamaktadır. Kullanıcı kabul testleri (UAT) konusunda ise, %88,6'sı (101 test) projelerinde bu tür testlerin yapıldığını, %7,9'u (9 test) yapılmadığını ve %3,5'i (4 test) ise bilgi sahibi olmadığını belirtmiştir. Buradan, kullanıcı kabul testlerinin projelerde geniş çapta uygulandığı ve büyük oranda kabul gördüğü anlaşılmaktadır, smoke testlerin ise daha az yaygın olduğu görülmekte, bu durum projelerin veya organizasyonların ihtiyaç ve önceliklerine bağlı olabilir.

Çizelge 4.24. Projede Smoke Testlerini Gerçekleştiren Roller

| | | n | % |
|---|---|---|---|
| Smoke testlerini projenizde hangi roller yapıyor? | Yazılım Geliştiriciler | 5 | 22,7 |
| | Analistler | 4 | 18,2 |
| | Son kullanıcılar | 1 | 4,5 |
| | Yazılım geliştiren firmadan bağımsız bir test ekibi | 7 | 31,8 |
| | Firma içinde kurulmuş ayrı test ekibi | 9 | 40,9 |
| | Bilmiyorum, fikrim yok | 1 | 4,5 |

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

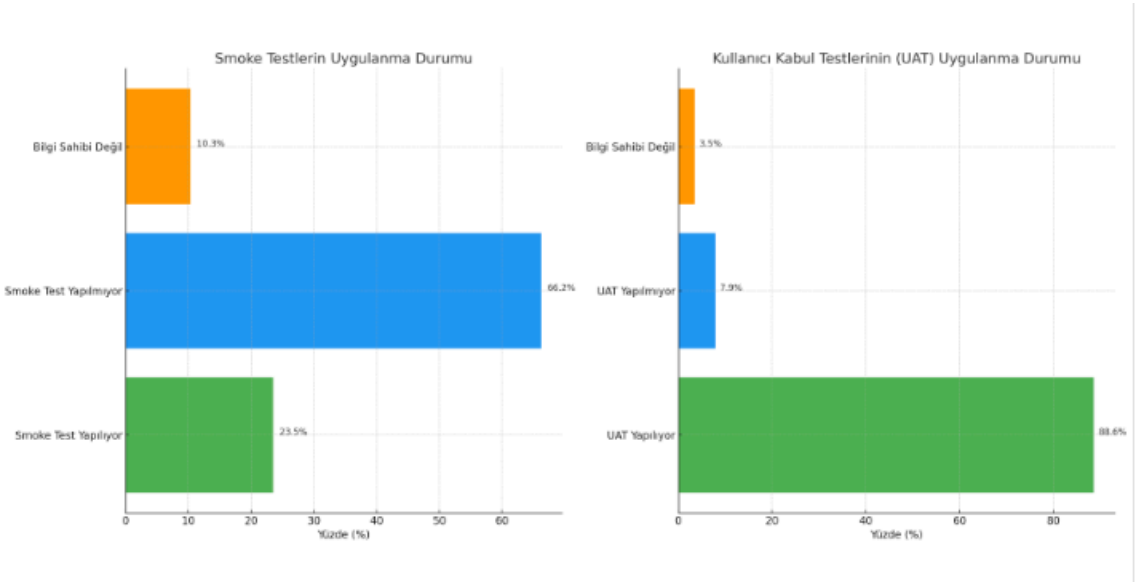
Çizelge 4.24'te görüldüğü üzere projede smoke testlerini gerçekleştiren roller üzerine yapılan incelemede, yazılım geliştiricilerden 5 kişi (%22,7) bu görevi üstlenmiş, analistlerden 4 kişi (%18,2) smoke testlerini yapmakta, son kullanıcılar arasından ise bu görevi gerçekleştiren sadece 1 kişi (%4,5) bulunmaktadır. Ayrıca, yazılım geliştiren firma dışından bağımsız bir test ekibi 7 kişi (%31,8) tarafından smoke testleri yürütülmekte, firma içinde kurulmuş ayrı bir test ekibi ise 9 kişi (%40,9) tarafından bu testler gerçekleştirilmektedir. Bilgi sahibi olmayan veya fikri olmayan kişi sayısı ise toplam katılımcıların %4,5'ini oluşturarak 1 kişi olarak belirlenmiştir. Bu veriler, 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.

Çizelge 4.25. Kullanıcı Kabul Testlerini Gerçekleştiren Roller

| | | n | % |
|---|---|---|---|
| Kullanıcı kabul testlerini hangi roller yapıyor? | İş Birimleri | 44 | 39,6 |
| | Developerlar | 5 | 4,5 |
| | Analistler | 54 | 48,6 |
| | Son kullanıcılar | 5 | 4,5 |
| | Yazılım geliştiren firmadan bağımsız bir test ekibi | 24 | 21,6 |
| | Firma içinde kurulmuş ayrı test ekibi | 9 | 8,1 |
| | Bilmiyorum, fikrim yok | 5 | 4,5 |
| | Diğer | 36 | 32,4 |

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

Çizelge 4.25'te görüldüğü üzere kullanıcı kabul testlerini gerçekleştiren roller üzerine yapılan analizde, iş birimlerinden 44 kişi (%39,6) bu görevi üstlenmiş, analistlerden 54 kişi (%48,6) bu testleri yapmaktadır, ki bu iki grup test sürecinde en aktif rolleri üstlenmektedir. Developerlar ve son kullanıcılar ise her biri 5 kişi (%4,5) ile bu testlerde daha az yer almaktadır. Yazılım geliştiren firma dışından bağımsız bir test ekibi 24 kişi (%21,6) tarafından, firma içinde kurulmuş ayrı bir test ekibi ise 9 kişi (%8,1) tarafından kullanıcı kabul testlerini gerçekleştirmektedir. Bilgi sahibi olmayan veya fikri olmayan kişilerin sayısı 5 (%4,5) olup, diğer roller ise 36 kişi (%32,4) tarafından bu testlerin yürütülmesinde rol almaktadır. Bu bulgular, kullanıcı kabul testlerinin projelerde geniş bir katılımcı yelpazesi tarafından yürütüldüğünü ve çeşitli departmanların bu süreçte önemli roller üstlendiğini göstermektedir, özellikle analistler ve iş birimleri bu testlerde merkezi bir role sahiptir.




Grafik 4.17. Kullanıcı Kabul Testleri ve Smoke Testler

### 4.5.1. **Projelerin Otomasyon Test Süreçleri ve İşleyişlerinin Analizi**

Çizelge 4.26. Projelerde Otomasyon Testlerinin Yapılıp Yapılmadığı

| | | n | % |
|---|---|---|---|
| Projelerinizde otomasyon testleri yapılıyor mu? | Evet | 73 | 62,9 |
| | Hayır | 37 | 31,9 |
| | Bilmiyorum, fikrim yok | 6 | 5,2 |
| | Diğer | 0 | 0,0 |
| Otomasyon testleri için destek aldığınız firmanızdan bağımsız bir test ekibiniz var mı? | Evet | 21 | 48,8 |
| | Hayır | 17 | 39,5 |
| | Bilmiyorum, fikrim yok | 0 | 0,0 |
| | Diğer | 5 | 11,6 |
| Otomasyon test ekibiniz yoksa otomasyon testlerini kim ya da hangi rol yapıyor? | Analist | 0 | 0,0 |
| | Developer | 3 | 75,0 |
| | Bilmiyorum, fikrim yok | 0 | 0,0 |
| | Diğer | 1 | 25,0 |

Çizelge 4.26. (Devamı) Projelerde Otomasyon Testlerinin Yapılıp Yapılmadığı

| | | | |
|---|---|---|---|
| Projenizdeki otomasyon test süreçlerindeki dokümantasyon nasıldır? | Biraz | 3 | 37,5 |
| | Çok | 3 | 37,5 |
| | Hiç | 1 | 12,5 |
| | Bilmiyorum, fikrim yok | 1 | 12,5 |
| | Diğer | 0 | 0,0 |
| Otomasyon testlerine başlamadan önce otomasyon için ayrı test senaryoları yazıyor musunuz? Yoksa fonksiyonel test ekibinin yazdığı senaryoları mı kullanıyorsunuz? | Evet | 29 | 82,9 |
| | Hayır | 6 | 17,1 |
| Projenizde otomasyonu data oluşturmak için kullanıyor musunuz? Biraz detaylandırabilir misiniz? | Evet | 16 | 64,0 |
| | Hayır | 9 | 36,0 |

Çizelge 4.26'da görüldüğü üzere projelerde otomasyon testlerinin yapılıp yapılmadığına ilişkin olarak, katılımcıların %62,9'u (73 kişi) evet yanıtını vermişken, %31,9'u (37 kişi) hayır demiş ve %5,2'si (6 kişi) bilgi sahibi olmadığını belirtmiştir. Otomasyon testleri için bağımsız bir test ekibi bulunduranların oranı %48,8 (21 kişi), bulundurmayanların oranı ise %39,5 (17 kişi) olarak tespit edilmiş, diğer kategorideki katılımcı sayısı 5 (%11,6) olarak kaydedilmiştir. Otomasyon test ekibi olmaması durumunda, bu testleri gerçekleştirenler arasında 3 developer (%75,0) ve 1 diğer kategoriden kişi (%25,0) bulunmaktadır. Otomasyon test süreçlerindeki dokümantasyon durumu incelendiğinde, katılımcıların %37,5'i biraz, %37,5'i çok, %12,5'i ise hiç dokümantasyon yapılmadığını belirtmiş, bilgi sahibi olmayanlar 1 kişi olarak kaydedilmiştir. Ayrıca, otomasyon testlerine başlamadan önce test senaryolarının yazılması konusunda %82,9 (29 kişi) evet derken, %17,1 (6 kişi) hayır demiştir. Otomasyonun data oluşturmak için kullanılıp kullanılmadığına dair, %64,0 (16 kişi) evet, %36,0 (9 kişi) ise hayır yanıtını vermiştir. Bu veriler, projelerde otomasyon testlerinin yaygın olarak kullanıldığını, ancak otomasyon test süreçleri ve dokümantasyon konusunda çeşitlilik gösterdiğini ve bazı alanlarda iyileştirme ihtiyacını ortaya koymaktadır.

Çizelge 4.27. Otomasyon Testlerinin Kullanım Alanları

| | | n | % |
|---|---|---|---|
| Genellikle hangi test türleri için otomasyon testini kullanıyorsunuz? | Regresyon | 35 | 85,4 |

| | | | |
|---|---|---|---|
| | Yük | 9 | 22,0 |
| | Performans | 9 | 22,0 |
| | Data oluşturma | 9 | 22,0 |
| | Diğer | 1 | 2,4 |

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

Çizelge 4.27'de görüldüğü üzere otomasyon testlerinin kullanım alanlarına bakıldığında, katılımcıların büyük bir çoğunluğu (%85,4, 35 kişi) regresyon testleri için otomasyonu tercih etmektedir, bu, otomasyonun tekrarlanabilir test süreçlerinde ne kadar değerli olduğunu göstermektedir. Yük testleri, performans testleri ve data oluşturma için otomasyon kullanımı ise her biri için %22,0 (9 kişi) oranında olup, bu test türlerinde otomasyonun daha az yaygın ancak önemli bir role sahip olduğunu belirtmektedir. Diğer test türleri için otomasyon kullanımı çok düşük bir oran olan %2,4 (1 kişi) ile sınırlı kalmıştır. Bu durum, otomasyonun çoğunlukla belirli ve yaygın test tiplerinde kullanıldığını, ancak bazı özel durumlar dışında genel kullanımının sınırlı olduğunu göstermektedir.

Çizelge 4.28. Otomasyon Süreçlerinin DevOps'a Entegrasyonu

| | | n | % |
|---|---|---|---|
| Otomasyon süreçleriniz devops a entegre midir? | Evet | 6 | 75,0 |
| | Hayır | 1 | 12,5 |
| | Bilmiyorum, fikrim yok | 1 | 12,5 |
| | Diğer | 0 | 0,0 |

Çizelge 4.28'de görüldüğü üzere otomasyon süreçlerinin DevOps'a entegrasyonu konusunda, katılımcıların büyük çoğunluğu (%75,0, 6 kişi) otomasyon süreçlerinin DevOps ile entegre olduğunu belirtmişken, sadece 1 kişi (%12,5) bu entegrasyonun olmadığını ifade etmiş ve aynı oranda (%12,5) bir kişi de bu konuda bilgi sahibi olmadığını veya fikri olmadığını dile getirmiştir. Diğer kategorisinde ise herhangi bir katılımcı bulunmamaktadır. Bu bulgular, katılımcıların çoğunun otomasyon süreçlerini DevOps kültürüyle uyumlu bir şekilde entegre ettiğini göstermekte, bu da süreçlerin daha hızlı ve verimli bir şekilde yürütülmesine olanak sağladığını işaret edebilir. Ancak,

tüm katılımcıların bu entegrasyonun farkında olmaması, bazı projelerde bu entegrasyonun henüz tam anlamıyla gerçekleşmediğini veya bu konudaki farkındalığın düşük olduğunu gösterebilir.

Çizelge 4.29. Projelerde Geliştirme ve Testler İçin Kullanılan Ortamlar

| | | n | % |
|---|---|---|---|
| Projenizde geliştirmeler ve testler için kaç ortam kullanılmaktadır? | Test ortamı | 14 | 100,0 |
| | Development ortamı | 8 | 57,1 |
| | Stable | 1 | 7,1 |
| | Preprod | 12 | 85,7 |
| | Prod | 12 | 85,7 |
| Otomasyon testlerini genellikle hangi ortamlar için çalıştırıyorsunuz? | Test ortamı | 6 | 42,9 |
| | Preprod | 8 | 57,1 |
| | Diğer | 4 | 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.)*

Çizelge 4.29'da görüldüğü üzere projelerde geliştirme ve testler için kullanılan ortamlar incelendiğinde, tüm katılımcılar (%100, 14 kişi) test ortamını kullanırken, %57,1'i (8 kişi) development ortamını, %7,1'i (1 kişi) stable ortamını, %85,7'si (12 kişi) preprod ve prod ortamlarını kullanmaktadır. Bu veriler, geliştirme ve test süreçlerinde farklı ortamlara ihtiyaç duyulduğunu ve her ortamın belirli amaçlarla kullanıldığını göstermektedir. Otomasyon testlerinin hangi ortamlarda çalıştırıldığına ilişkin olarak, %42,9 (6 kişi) otomasyon testlerini test ortamında, %57,1 (8 kişi) preprod ortamında çalıştırırken, %28,6 (4 kişi) diğer kategorilerde belirtilen ortamlarda bu testleri gerçekleştirmektedir. Bu bulgular, test ortamının ve preprod ortamının otomasyon testleri için yaygın olarak tercih edildiğini gösterirken, bazı katılımcıların 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.

Çizelge 4.30. Otomasyon Test Süreçleri İle İlgili Anket Sonuçları

| | | n | % |
|---|---|---|---|
| Otomasyon testleri için belirlenmiş olan regresyon senaryo setleriniz var mı? | Evet | 13 | 92,9 |
| | Hayır | 0 | 0,0 |
| | Bilmiyorum, fikrim yok | 1 | 7,1 |
| | Diğer | 0 | 0,0 |
| Belirlemiş olduğunuz bu regresyon setlerini ne sıklıkla koşuyorsunuz? | Her hafta | 1 | 20,0 |
| | Her ayın sonunda | 1 | 20,0 |
| | Her yeni kod çıkışı olduğunda | 1 | 20,0 |
| | Bilmiyorum, fikrim yok | 2 | 40,0 |
| | Diğer | 0 | 0,0 |
| Otomasyon test senaryolarını yazmak için bir tool kullanıyor musunuz? | Evet | 9 | 69,2 |
| | Hayır | 3 | 23,1 |
| | Bilmiyorum, fikrim yok | 1 | 7,7 |
| | Diğer | 0 | 0,0 |
| Otomasyon senaryolarını yazmak için hangi toolu'u kullanıyorsunuz? | Jira | 3 | 60,0 |
| | HP ALM | 0 | 0,0 |
| | Excel | 1 | 20,0 |
| | Bilmiyorum, fikrim yok | 0 | 0,0 |
| | Diğer | 1 | 20,0 |
| Otomasyon testlerini sıklıkla hangi kanallar için kullanıyorsunuz? | Web | 9 | 64,3 |
| | Mobilweb | 0 | 0,0 |
| | Application | 1 | 7,1 |
| | Bilmiyorum, fikrim yok | 4 | 28,6 |

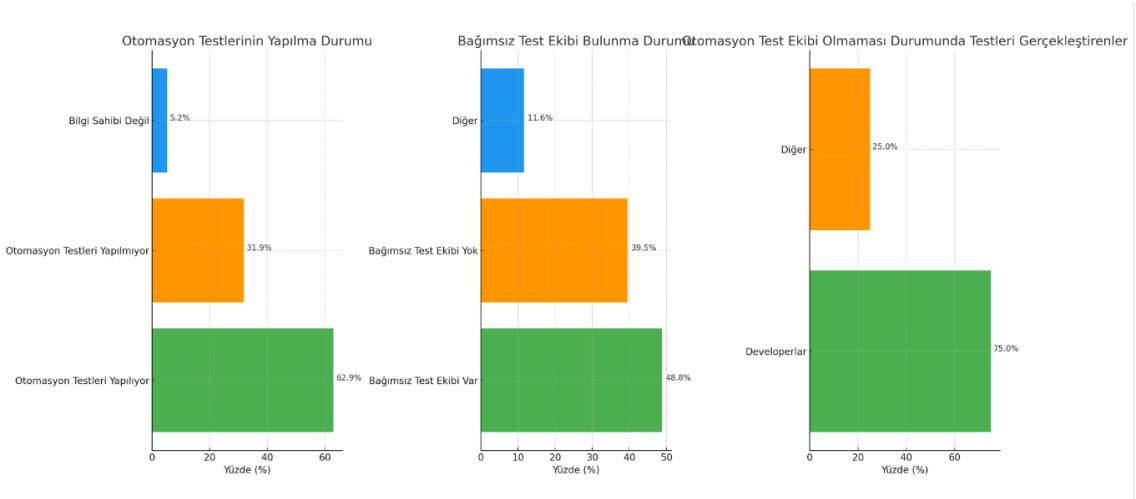
| | | | |
|---|---|---|---|
| | Diğer | 0 | 0,0 |
| Web otomasyon testleri için sıklıkla hangi aracı kullanıyorsunuz? | Selenium Framework | 9 | 69,2 |
| | Appium Framework | 0 | 0,0 |

Çizelge 4.30. (Devamı) Otomasyon Test Süreçleri İle İlgili Anket Sonuçları

| | | | |
|---|---|---|---|
| | Postman | 0 | 0,0 |
| | Soap | 0 | 0,0 |
| | Bilmiyorum, fikrim yok | 4 | 30,8 |
| | Diğer | 0 | 0,0 |
| Application otomasyon testleri için sıklıkla hangi aracı kullanıyorsunuz? | Selenium Framework | 6 | 42,9 |
| | Appium Framework | 4 | 28,6 |
| | Postman | 0 | 0,0 |
| | Soap | 0 | 0,0 |
| | Bilmiyorum, fikrim yok | 4 | 28,6 |
| | Diğer | 0 | 0,0 |
| Otomasyon testlerinde karşılaştığınız hataları raporlamak için bir araç kullanıyor musunuz? | Evet | 4 | 80,0 |
| | Hayır | 1 | 20,0 |
| | Bilmiyorum, fikrim yok | 0 | 0,0 |
| | Diğer | 0 | 0,0 |
| Otomasyon testlerinde karşılaştığınız hataları raporlamak için hangi aracı kullanıyorsunuz? | Jira | 2 | 66,7 |
| | HP ALM | 0 | 0,0 |
| | Excel | 0 | 0,0 |
| | Bilmiyorum, fikrim yok | 1 | 33,3 |
| | Diğer | 0 | 0,0 |

Çizelge 4.30’da görüldüğü üzere otomasyon test süreçleri ile ilgili anket sonuçlarına göre, katılımcıların %92,9'u (13 kişi) belirlenmiş regresyon senaryo setlerine sahip

olduklarını belirtmiş, sadece %7,1'i (1 kişi) bu konuda bilgi sahibi olmadığını ifade etmiştir. Bu regresyon setlerinin kullanım sıklığına gelince, her hafta, her ayın sonunda ve her yeni kod çıkışında bu testleri çalıştıranlar her biri %20 oranında (1'er kişi) olurken, bilgi sahibi olmayanların oranı %40 (2 kişi) olarak belirlenmiştir. Otomasyon test senaryolarını yazmak için kullanılan araçlar arasında, %69,2 (9 kişi) evet yanıtını verirken, %23,1 (3 kişi) hayır demiş ve %7,7'si (1 kişi) bilgi sahibi olmadığını belirtmiştir. Bu araçlar arasında Jira %60 (3 kişi), Excel %20 (1 kişi) ve diğer araçlar %20 (1 kişi) kullanım oranıyla yer almaktadır. Otomasyon testlerinin kullanım alanlarına bakıldığında, %64,3 (9 kişi) web için, %7,1 (1 kişi) application için kullanıldığı, %28,6'sının (4 kişi) bilgi sahibi olmadığı belirtilmiştir. Web otomasyon testlerinde %69,2 (9 kişi) Selenium Framework kullanılırken, application testlerinde %42,9 (6 kişi) Selenium ve %28,6 (4 kişi) Appium tercih edilmiştir. Ayrıca, otomasyon testlerinde karşılaşılan hataları raporlamak için kullanılan araçlarla ilgili olarak, %80 (4 kişi) evet, %20 (1 kişi) ise hayır yanıtı vermiştir. Bu bulgular, otomasyon test süreçlerinin kuruluşlar arasında yaygın olarak benimsendiğini ve çeşitli araçların etkin bir şekilde kullanıldığını göstermekte, özellikle regresyon ve web otomasyon testlerinde Selenium'un popüler bir tercih olduğunu ortaya koymaktadır.




Grafik 4.18. Otomasyon Testleriyle İlgili Genel Bilgiler

Çizelge 4.31. Projede Test Ekibinin Bir Test Yöneticisine Sahip Olup Olmama Durumu

| | | n | % |
|---|---|---|---|
| Otomasyon test ekibinin test yöneticisi var mı? | Evet | 22 | 56,4 |

| | | | |
|---|---|---|---|
| | Hayır | 17 | 43,6 |
| | Bilmiyorum, fikrim yok | 0 | 0,0 |
| | Diğer | 0 | 0,0 |
| Test yöneticisi yoksa test yönetimini kim ya da hangi rol yapmaktadır? | Test ekibinden biri | 0 | 0,0 |
| | Analist | 1 | 5,6 |
| | Yazılım Geliştirici | 0 | 0,0 |
| | Yönetici | 15 | 83,3 |
| | Bilmiyorum, fikrim yok | 2 | 11,1 |
| | Diğer | 0 | 0,0 |
| Projenizdeki otomasyon test süreçlerindeki dokümantasyon nasıldır? | Biraz | 17 | 54,8 |
| | Çok | 7 | 22,6 |
| | Hiç | 2 | 6,5 |
| | Bilmiyorum, fikrim yok | 5 | 16,1 |
| | Diğer | 0 | 0,0 |

Çizelge 4.31’de görüldüğü üzere projede test ekibinin bir test yöneticisine sahip olup olmama durumu incelendiğinde, katılımcıların %56,4'ü (22 kişi) test ekibinin bir test yöneticisine sahip olduğunu, %43,6'sı (17 kişi) ise sahip olmadığını belirtmiştir. Test yöneticisi bulunmayan durumlarda, test yönetimi görevini genellikle yöneticilerin %83,3'ü (15 kişi) üstlenmiş, analistlerden sadece 1 kişi (%5,6) bu rolü almış, yazılım geliştiriciler ve test ekibinden kimse (%0,0) bu sorumluluğu üstlenmemiştir. Bilgi sahibi olmayan veya fikri olmayanların oranı ise %11,1 (2 kişi) olarak belirlenmiştir. Otomasyon test süreçlerindeki dokümantasyon durumuna gelince, %54,8 (17 kişi) biraz dokümantasyon olduğunu, %22,6 (7 kişi) çok dokümantasyon olduğunu, %6,5 (2 kişi) ise hiç dokümantasyon olmadığını belirtmiş, %16,1'i (5 kişi) ise bu konuda bilgi sahibi olmadığını ifade etmiştir. Bu bulgular, projelerde test yönetimi ve dokümantasyon süreçlerinin farklı şekillerde yürütüldüğünü ve bu konularda bazı belirsizliklerin olduğunu göstermektedir.

Çizelge 4.32. Otomasyon Testlerinin Kullanım Alanları

| | | n | % |
|---|---|---|---|
| Genellikle hangi test türleri için otomasyon testini kullanıyorsunuz? | Regresyon | 32 | 82,1 |
| | Performans | 6 | 15,4 |
| | Data Oluşturma | 17 | 43,6 |
| | Bilmiyorum, fikrim yok. | 2 | 5,1 |
| | Hiçbiri | 2 | 5,1 |

Çizelge 4.32’de görüldüğü üzere otomasyon testlerinin kullanım alanları incelendiğinde, katılımcıların %82,1'i (32 kişi) regresyon testleri için otomasyon kullanırken, %15,4'ü (6 kişi) performans testleri için ve %43,6'sı (17 kişi) data oluşturma için otomasyon tekniklerinden yararlanmaktadır. Bu, otomasyonun çeşitli test süreçlerinde yaygın olarak kullanıldığını göstermektedir, özellikle regresyon testlerinde otomasyonun kritik bir rol oynadığını ortaya koymaktadır. Ayrıca, %5,1 (2 kişi) katılımcı bilgi sahibi olmadığını veya fikri olmadığını ifade etmiş, aynı oranda (2 kişi) hiçbir test türü için otomasyon kullanılmadığını belirtmiştir. Bu durum, otomasyonun test süreçlerindeki etkinliğini ve bazı durumlarda bilgi eksikliğinin veya otomasyonun uygulanamamasının varlığını gösterir.

Çizelge 4.33. Otomasyon Süreçlerinin DevOps'a Entegrasyonu

| | | n | % |
|---|---|---|---|
| Otomasyon süreçleriniz devops a entegre midir? | Evet | 11 | 52,4 |
| | Hayır | 8 | 38,1 |
| | Bilmiyorum, fikrim yok | 2 | 9,5 |
| | Diğer | 0 | 0,0 |
| Projenizde geliştirmeler ve testler için kaç ortam kullanılmaktadır? | Test | 58 | 96,7 |
| | Development | 36 | 60,0 |

| | | | |
|---|---|---|---|
| | Stable | 9 | 15,0 |
| | Preprod | 38 | 63,3 |
| | Production | 34 | 56,7 |
| | Diğer | 2 | 3,3 |

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

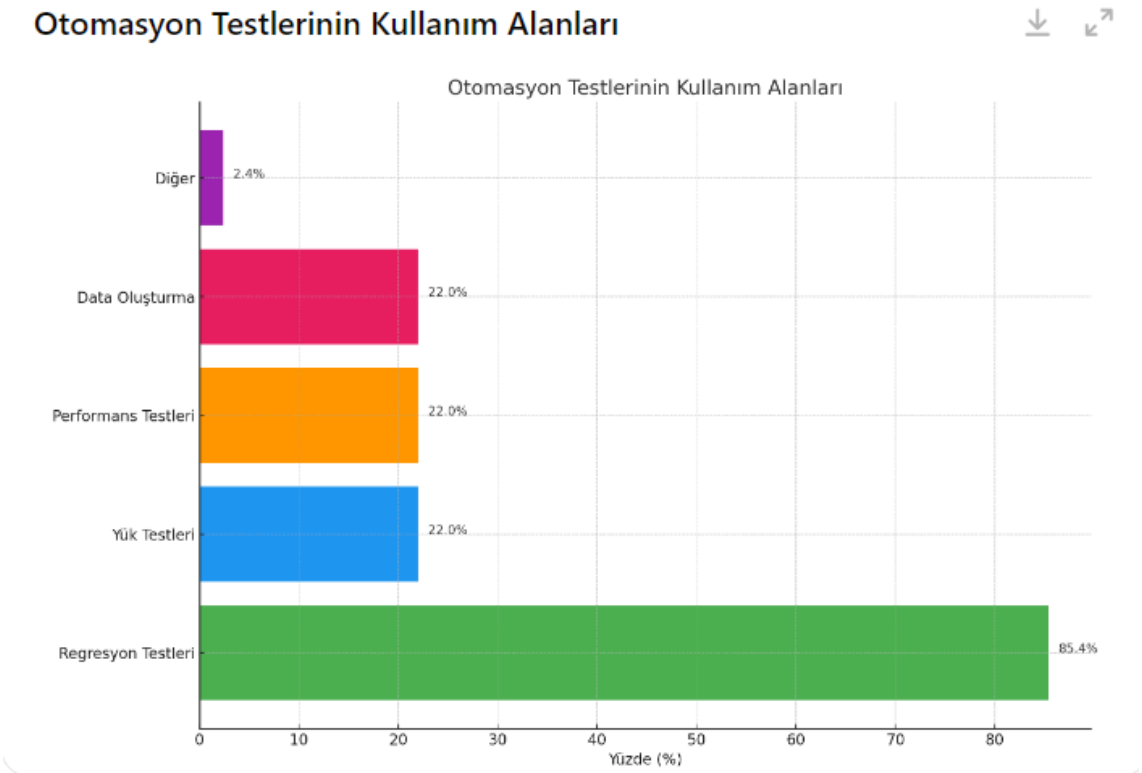
Çizelge 4.33'de görüldüğü üzere otomasyon süreçlerinin DevOps'a entegrasyonu konusunda katılımcıların %52,4'ü (11 kişi) bu süreçlerin entegre olduğunu belirtmiş, %38,1'i (8 kişi) entegre olmadığını ifade etmiş, ve %9,5'i (2 kişi) bu konuda bilgi sahibi olmadığını ya da fikri olmadığını dile getirmiştir. Proje geliştirmeleri ve testler için kullanılan çeşitli ortamlar arasında, %96,7'si (58 kişi) test ortamını, %60,0'ı (36 kişi) development ortamını, %63,3'ü (38 kişi) preprod ortamını ve %56,7'si (34 kişi) prod ortamını kullanmaktadır. Ayrıca, %15,0'ı (9 kişi) stable ortamı ve %3,3'ü (2 kişi) diğer ortamları kullanmaktadır. Bu veriler, çoğu projede birden fazla geliştirme ve test ortamının etkin bir şekilde kullanıldığını göstermekte, bu da farklı aşamalar ve ihtiyaçlar için çeşitlilik sağlamaktadır. Ayrıca, otomasyon süreçlerinin yarıdan fazlasının DevOps ile entegrasyonu, süreçlerin verimliliğini ve sürekliliğini artırmada önemli bir rol oynadığını işaret etmektedir.

Çizelge 4.34. Otomasyon Testlerinin Kullanıldığı Ortamlar

| | | n | % |
|---|---|---|---|
| Otomasyon testlerini genellikle hangi ortamlar için çalıştırıyorsunuz? | Test | 43 | 72,9 |
| | Development | 3 | 5,1 |
| | Stable | 2 | 3,4 |
| | Preprod | 33 | 55,9 |
| | Production | 3 | 5,1 |
| | Bilmiyorum, fikrim yok. | 1 | 1,7 |
| | Diğer | 1 | 1,7 |

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

Çizelge 4.34'de görüldüğü üzere otomasyon testlerinin kullanıldığı ortamlar hakkında elde edilen verilere göre, en yaygın olarak test ortamında otomasyon testleri yürütülmekte olup, bu ortamı kullananların oranı %72,9 (43 kişi) olarak belirlenmiştir. Preprod ortamı da %55,9 (33 kişi) ile sıkça kullanılan bir diğer ortam olarak öne çıkmaktadır. Buna karşın, development ve prod ortamlarında otomasyon testlerini çalıştıranların oranı her ikisi için de %5,1 (3'er kişi) olarak oldukça düşüktür. Stable ortamı için bu oran %3,4 (2 kişi) ile daha da azalmaktadır. Otomasyon testlerinin diğer ortamlarda kullanımı ve bilgi sahibi olmayanların oranı ise her biri %1,7 (1'er kişi) olarak tespit edilmiştir. Bu veriler, otomasyon testlerinin özellikle test ve preprod ortamlarında yoğunlaştığını göstermekte, ancak development ve prod gibi ortamlarda bu testlerin nadiren kullanıldığını ortaya koymaktadır. Bu durum, otomasyon testlerinin projenin farklı aşamalarında ve ihtiyaçlarına göre nasıl uygulandığına dair önemli içgörüler sunmaktadır.




Grafik 4.19. Otomasyon Testlerinin Kullanım Alanları

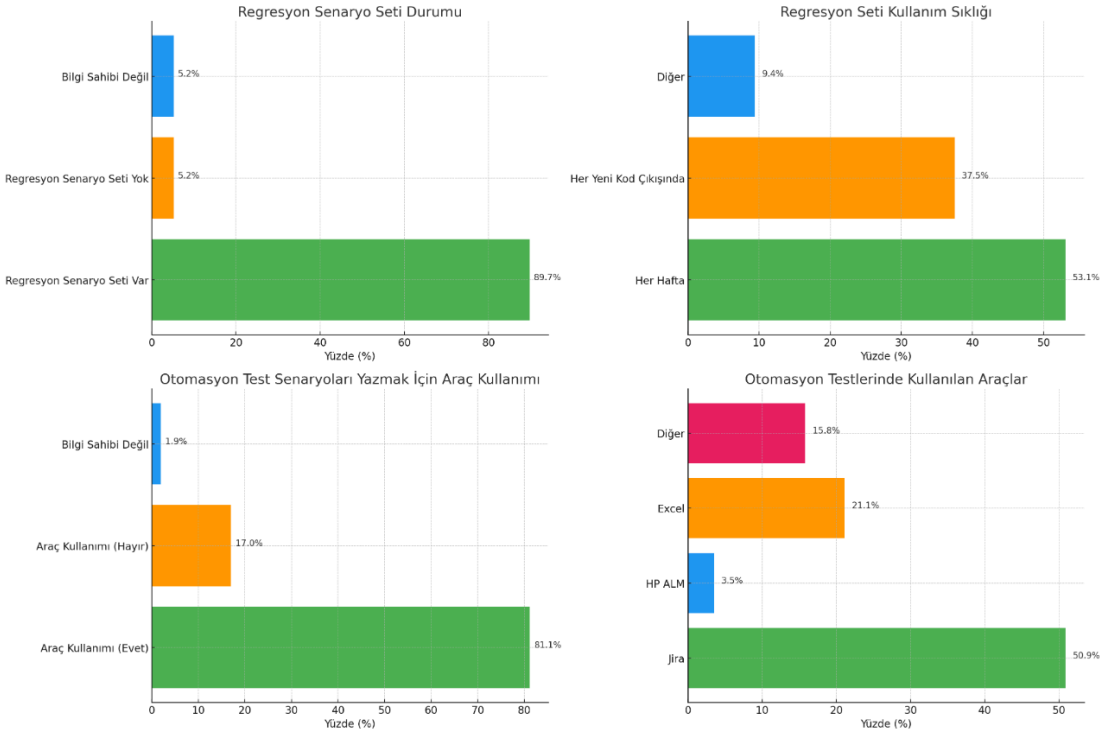
Çizelge 4.35. Otomasyon Testlerinin Planlanması ve Uygulanması

| | | n | % |
|---|---|---|---|
| Otomasyon testleri için belirlenmiş olan regresyon senaryo setleriniz var mı? | Evet | 52 | 89,7 |
| | Hayır | 3 | 5,2 |
| | Bilmiyorum, fikrim yok | 3 | 5,2 |
| | Diğer | 0 | 0,0 |
| Belirlemiş olduğunuz bu regresyon setlerini ne sıklıkla koşuyorsunuz? | Her hafta | 17 | 53,1 |
| | Her ayın sonu | 0 | 0,0 |
| | Her Yeni Kod çıkış Olduğunda | 12 | 37,5 |
| | Bilmiyorum, fikrim yok | 0 | 0,0 |
| | Diğer | 3 | 9,4 |
| Otomasyon test senaryolarını yazmak için bir tool kullanıyor musunuz? | Evet | 43 | 81,1 |
| | Hayır | 9 | 17,0 |
| | Bilmiyorum, fikrim yok | 1 | 1,9 |
| | Diğer | 0 | 0,0 |
| Otomasyon test senaryolarını yazmak için hangi toolu kullanıyor musunuz? | Jira | 29 | 50,9 |
| | HP ALM | 2 | 3,5 |
| | Excel | 12 | 21,1 |
| | Bilmiyorum, fikrim yok | 5 | 8,8 |
| | Diğer | 9 | 15,8 |

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

Çizelge 4.35'de görüldüğü üzere katılımcıların büyük çoğunluğu (%89,7) otomasyon testleri için belirlenmiş regresyon senaryo setlerine sahip olduklarını belirtmiş, %5,2'si bunların olmadığını ve aynı oranda (%5,2) bilgi sahibi olmadıklarını ifade etmiştir. Bu regresyon setlerini kullanma sıklığına gelince, %53,1'i bu setleri her hafta, %37,5'i her yeni kod çıkışında koştuğunu belirtmiş, %9,4'ü ise "Diğer" seçeneklerini işaretlemiştir. Otomasyon test senaryolarını yazmak için bir araç kullanımı sorusuna %81,1'lik bir kesim evet yanıtı verirken, %17,0'i hayır demiş ve %1,9'u bilgi sahibi olmadığını

belirtmiştir. Kullanılan araçlar arasında Jira %50,9 ile en popüler seçenek olup, diğerleri arasında HP ALM %3,5, Excel %21,1 ve "Diğer" %15,8 oranında kullanılmaktadır. Bu veriler, otomasyon testlerinin yaygın olarak regresyon senaryoları üzerinden yürütüldüğünü ve çoğunlukla Jira gibi yönetim araçlarının bu süreçlerde tercih edildiğini göstermektedir. Bu, otomasyon testlerinin planlanması ve uygulanmasında düzenli ve organize bir yaklaşımın benimsendiğini ortaya koymaktadır.



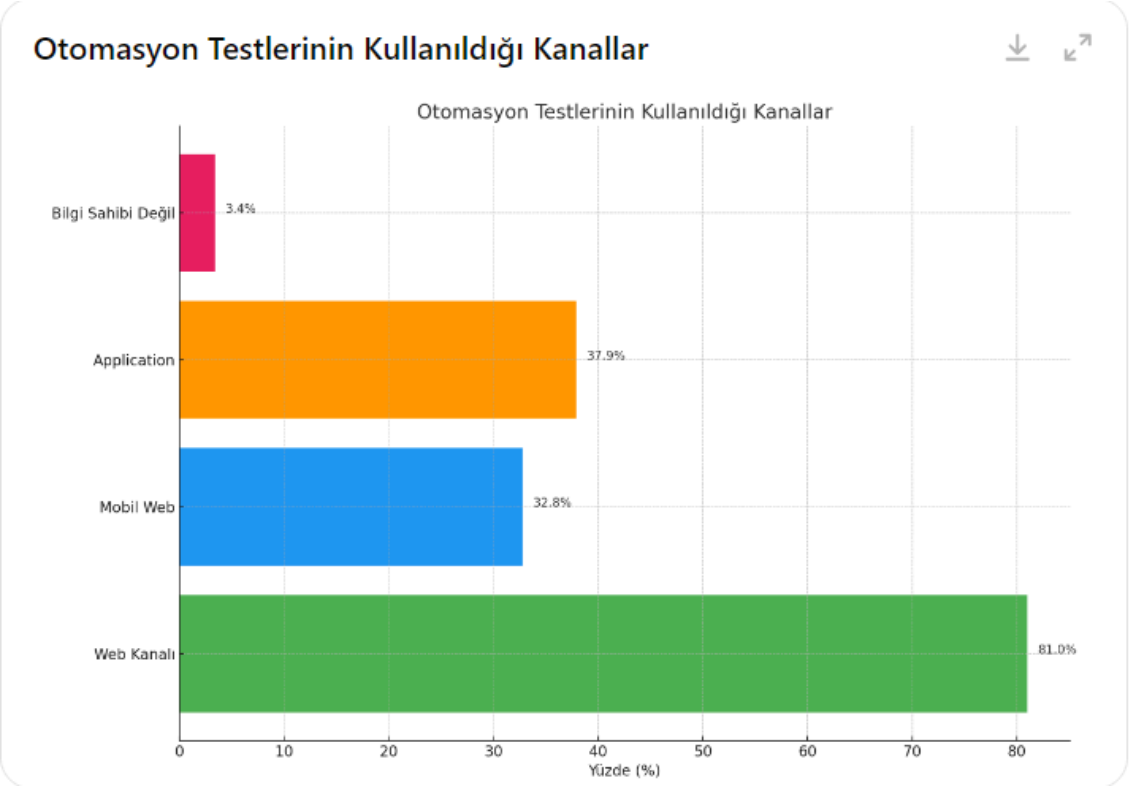

Grafik 4.20. Otomasyon Testleriyle İlgili Diğer Bilgiler

Çizelge 4.36. Otomasyon Testlerinin Kullanımı

| | | n | % |
|---|---|---|---|
| Otomasyon testlerini sıklıkla hangi kanallar için kullanıyorsunuz? | Web | 47 | 81,0 |
| | Mobilweb | 19 | 32,8 |
| | Application | 22 | 37,9 |
| | Bilmiyorum, fikrim yok | 2 | 3,4 |

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

Çizelge 4.36’da görüldüğü üzere otomasyon testlerinin kullanımı ile ilgili olarak, katılımcıların büyük bir çoğunluğu (%81,0) web kanalı için otomasyon testlerini kullandıklarını belirtmiştir. Ayrıca, %32,8'i mobil web ve %37,9'u application kanalları için otomasyon testleri yapmaktadır. Bununla birlikte, %3,4'lük bir kesim ise hangi kanallar için otomasyon testleri kullanıldığı konusunda bilgi sahibi olmadıklarını veya fikirlerinin olmadığını ifade etmiştir. Bu veriler, otomasyon testlerinin özellikle web ve uygulama geliştirme süreçlerinde yaygın olarak kullanıldığını göstermekte, aynı zamanda teknoloji alanındaki çeşitli platformlar üzerinde otomasyonun önemli bir rol oynadığını vurgulamaktadır.




Grafik 4.21. Otomasyon Testlerinin Kullanıldığı Kanallar

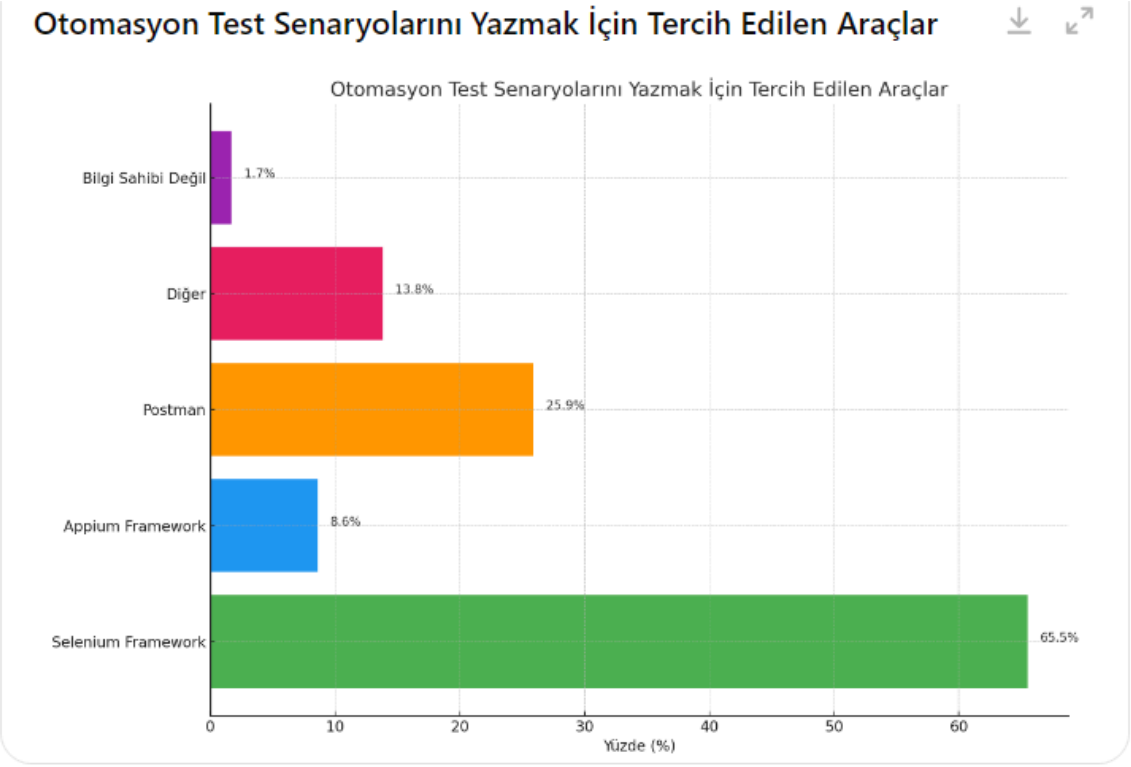
Çizelge 4.37. Otomasyon Test Senaryolarını Yazmak İçin Tercih Ettikleri Araçlar

| | | n | % |
|---|---|---|---|
| Web otomasyon testleri için sıklıkla hangi aracı kullanıyorsunuz? | Selenium Framework | 38 | 65,5 |
| | Appium Framework | 5 | 8,6 |
| | Postman | 15 | 25,9 |
| | Soap | 3 | 5,2 |

| | | | |
|---|---|---|---|
| | Diğer | 8 | 13,8 |
| | Bilmiyorum fikrim yok | 1 | 1,7 |

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

Çizelge 4.37'de görüldüğü üzere katılımcıların otomasyon test senaryolarını yazmak için tercih ettikleri araçlara ilişkin veriler, çeşitli araçların kullanım oranlarını göstermektedir. Özellikle, %65,5'lik büyük bir kesim Selenium Framework'ü tercih etmektedir, bu da bu aracın popülerliğini ve yaygınlığını vurgulamaktadır. Appium Framework'ün ise %8,6 gibi daha düşük bir oranda tercih edilmesi, mobil uygulama testleri için özel bir gereksinim olduğunda tercih edildiğini gösterebilir. %25,9'luk oranla Postman'ın tercih edilmesi, API testleri için bu aracın kullanımının yaygınlığını işaret ederken, %13,8'lik bir kesimin "Diğer" araçları tercih etmesi, test senaryolarını yazmak için çeşitli alternatif araçların kullanılabileceğini göstermektedir. Ancak, %1,7'lik bir kesimin bu konuda bilgi sahibi olmadığını belirtmesi, bazı katılımcıların test otomasyon araçları hakkında yeterli bilgiye sahip olmayabileceğini göstermektedir. Bu veriler, test otomasyon araçlarının çeşitliliğini ve farklı ihtiyaçları karşılayabilme kapasitesini yansıtmakta, ancak popüler araçların hala önde olduğunu göstermektedir.



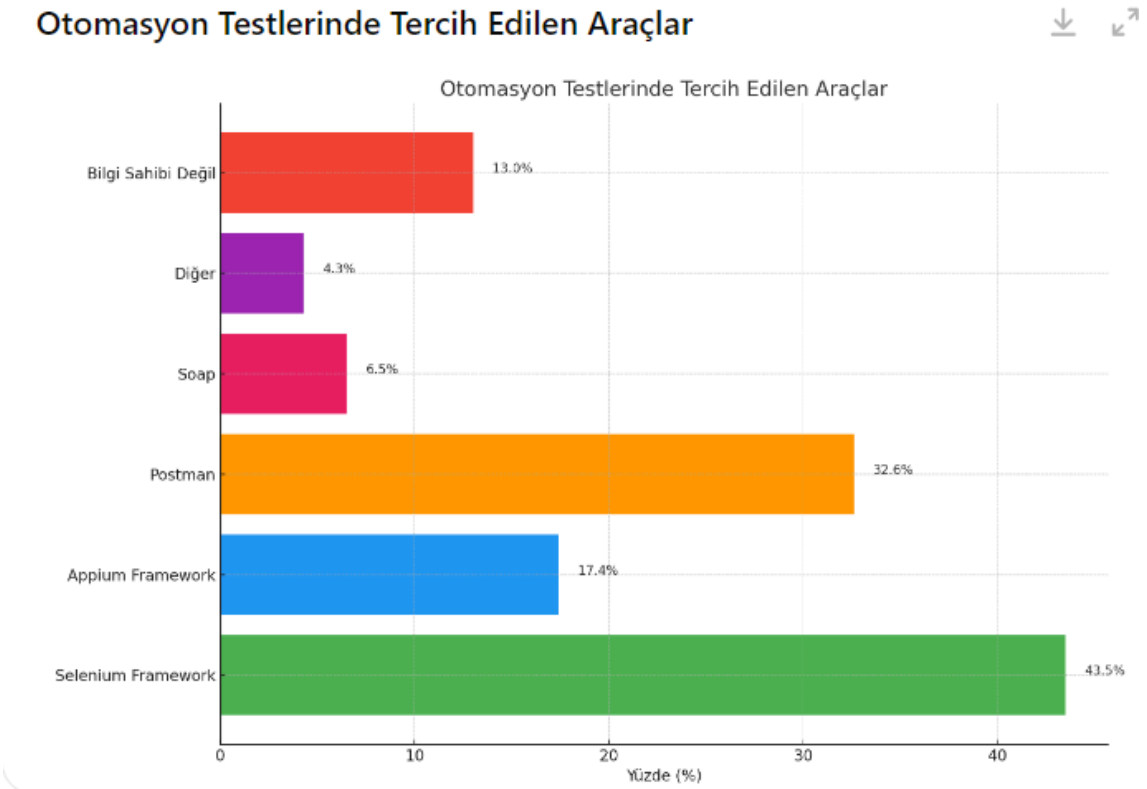

Grafik 4.22. Otomasyon test senaryolarını yazmak için tercih edilen araçlar

Çizelge 4.38. Otomasyon Testlerinde Tercih Edilen Araçlar

| | | n | % |
|---|---|---|---|
| Application otomasyon testleri için sıklıkla hangi aracı kullanıyorsunuz? | Selenium Framework | 20 | 43,5 |
| | Appium Framework | 8 | 17,4 |
| | Postman | 15 | 32,6 |
| | Soap | 3 | 6,5 |
| | Bilmiyorum fikrim yok | 6 | 13,0 |
| | Diğer | 2 | 4,3 |

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

Çizelge 4.38'de görüldüğü üzere katılımcıların otomasyon testlerinde tercih ettikleri araçlara ilişkin veriler, farklı test ihtiyaçlarını ve çeşitli araçların kullanımını göstermektedir. %43,5'lik oranla Selenium Framework'ün en yaygın tercih edilen araç olduğu görülüyor. Bu, Selenium'un web tabanlı uygulamalar için otomasyon testlerinde ne kadar yaygın ve etkili olduğunu göstermektedir. %17,4'lük oranla Appium Framework'ün de önemli bir kullanım oranına sahip olması, mobil uygulamalar için otomasyon testlerinde tercih edildiğini gösterebilir. %32,6'lık oranla Postman'ın tercih edilmesi, API testleri için bu aracın yaygın bir şekilde kullanıldığını göstermektedir. Bununla birlikte, %6,5'inin "Soap" aracını tercih etmesi ve %4,3'ünün "Diğer" araçları tercih etmesi, farklı test senaryoları için farklı araçların tercih edildiğini göstermektedir. Ancak, %13,0'luk bir kesimin bu konuda bir bilgisi olmadığını belirtmesi, bazı katılımcıların test otomasyon araçları hakkında yeterli bilgiye sahip olmayabileceğini göstermektedir. Bu veriler, test otomasyon araçlarının çeşitliliğini ve farklı ihtiyaçları karşılayabilme kapasitesini yansıtmakta, ancak popüler araçların hala önde olduğunu göstermektedir.




Grafik 4.23. Otomasyon Testlerinde Tercih Edilen Araçlar

Çizelge 4.39. Hataları Raporlamak İçin Kullanılan Araçlar

| | | n | % |
|---|---|---|---|
| Otomasyon testlerinde karşılaştığınız hataları raporlamak için bir araç kullanıyor musunuz? | Evet | 29 | 87,9 |
| | Hayır | 3 | 9,1 |
| | Bilmiyorum, fikrim yok | 1 | 3,0 |
| | Diğer | 0 | 0,0 |
| Otomasyon testlerinde karşılaştığınız hataları raporlamak için hangi aracı kullanıyorsunuz? | Jira | 12 | 41,4 |
| | HP ALM | 0 | 0,0 |
| | Excel | 9 | 31,0 |
| | Bilmiyorum, fikrim yok | 1 | 3,4 |
| | Diğer | 7 | 24,1 |

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

Çizelge 4.39’da görüldüğü üzere katılımcıların hataları raporlamak için çoğunlukla bir araç kullandıklarını gösteren %87,9'luk oran, hatanın tespit edilmesinden sonraki süreçte raporlanmanın önemini vurgular. Ancak %9,1'inin bir araç kullanmadığını belirtmesi, belki de daha basit veya daha az yapılandırılmış projelerde veya daha küçük ekiplerde bu sürecin manuel olarak yönetildiğini veya raporlanmadığını düşündürebilir. %3,0'luk bir kesimin bilgi sahibi olmadığını belirtmesi, bu sürecin belirsizliğini veya katılımcıların hataların raporlanma yöntemi hakkında yeterli bilgiye sahip olmadığını gösterebilir. Jira'nın %41,4'lük oranla en yaygın tercih edilen araç olması, bu tür süreçlerin proje yönetimi ve izleme araçlarına entegre edilmesinin önemini vurgular. Excel'in %31,0'luk oranla ikinci sırada olması, belki de daha esnek veya özelleştirilebilir bir yaklaşım sunması nedeniyle bazı projelerde tercih edildiğini gösterebilir. Diğer araçların daha düşük oranlarda tercih edilmesi, belki de bu araçların daha spesifik kullanım senaryolarına veya belirli endüstri standartlarına uyum sağlama kabiliyetine sahip olmamasından kaynaklanabilir. Bu bulgular, hata yönetiminin önemini ve farklı araçların kullanımının projenin gereksinimlerine ve özelliklerine bağlı olarak değişebileceğini vurgulamaktadır.

# 5. SONUÇ ve ÖNERİLER

Araştırmanın bulguları doğrultusunda elde edilen sonuçlar aşağıda ifade edilmiştir;

Araştırma bulguları, Türkiye'deki çevik yazılım geliştirme sektöründe çalışan profesyonellerin demografik özelliklerine dair detaylı bilgiler sunmaktadır. Katılımcıların çoğunluğu erkek (%64,7) olup, yaş olarak en yoğun grubu 31-35 yaş aralığı (%44,0) oluşturmaktadır. Eğitim düzeyi olarak, lisans mezunları (%82,8) hakimken, azımsanmayacak bir kısmı yüksek lisans yapmış (%15,5) ve neredeyse tümü aktif iş hayatında yer almaktadır (%100). Çalışma modeli olarak hibrit model (%67,2) tercih edilirken, uzaktan çalışma (%29,3) da önemli bir alternatif olarak karşımıza çıkmaktadır. İstanbul, özellikle Anadolu yakası, çalışanların büyük çoğunluğuna ev sahipliği yapmakta (%63,9). İş rolleri açısından iş analistleri (%26,7), yazılım geliştiriciler (%24,1) ve test kalite kontrol uzmanları (%24,1) en yaygın roller arasında yer almaktadır. İş tecrübesi açısından, katılımcıların çoğunluğu beş yıl ve üzeri deneyime sahipken (%65,5), otomasyon yazılım test tecrübesi olmayanların oranı (%74,5), manuel yazılım test tecrübesi olmayanlardan (%53,5) daha yüksektir, bu da otomasyon testlerinin daha az yaygın olduğunu göstermektedir. Sektörel dağılım itibarıyla, bankacılık (%33,3), telekomünikasyon (%18,1) ve e-ticaret (%21,9) sektörleri öne çıkmaktadır, bu da çevik metodolojilerin farklı alanlarda nasıl uyarlandığını yansıtmaktadır.

Araştırma sonuçları, Türkiye'deki yazılım geliştirme sektöründe çevik metodolojilerinin geniş çapta kabul gördüğünü ve uygulandığını ortaya koymaktadır; katılımcıların %99,1'i bu metodolojileri kullanmaktadır. Çevik metodolojilerin tercih edilme sebepleri arasında, esneklik, adaptasyon kapasitesi ve ekip içi işbirliğini teşvik eden yapısının önemli rol oynadığı anlaşılmaktadır. Diğer yandan, %16,4 oranında Şelale Iterative model kullanımı, bazı projeler için planlı ve aşamalı yaklaşımların hala tercih edildiğini göstermektedir. çevik metodolojilerin yaygınlığı, yazılım geliştirme süreçlerinde çevikliğin ve adaptasyonun giderek daha fazla önem kazandığını belirtirken, Code&Fix, V Model, Iterative ve Incremental gibi daha az popüler modellerin de kullanılması, farklı projelerin ve organizasyonların ihtiyaçlarına göre çeşitli yaklaşımların uygulanabilirliğini sürdürdüğünü göstermektedir. Bu durum, yazılım geliştirme eğitimleri ve pratiklerinin gelecekteki şekillenmesine yönelik değerli içgörüler sağlamaktadır.

Araştırma, Türkiye'deki yazılım geliştirme sektöründe çevik metodolojilerin, özellikle Scrum metodolojisinin, yoğun bir şekilde benimsendiğini göstermektedir. Katılımcıların %82,8'i Scrum'u tercih etmekte, bu durum Scrum'ın çeviklik, esneklik ve takım işbirliğine olan katkılarının şirketler tarafından ne kadar değerli bulunduğunu açıkça göstermektedir. Diğer yandan, Kanban metodolojisini kullanma oranı %25,9 ile bu yöntemin sürekli teslimat ve iş akışının düzenlenmesi gereken süreçlerde etkin bir alternatif olarak öne çıktığını belirtir. Lean Software Development ve Test-Driven Development gibi daha odaklı metodolojilerin daha düşük kullanım oranları, projelerin spesifik ihtiyaçlarına göre metodoloji seçimindeki çeşitliliği yansıtmaktadır. Bu çeşitlilik, çevik yaklaşımlarının adaptasyon yeteneğini ve yazılım geliştirme süreçlerinde sürekli iyileştirme ve adaptasyonun kritik önemini vurgular. Sonuç olarak, Türkiye'deki yazılım geliştirme sektöründe Scrum'un hakimiyeti yanı sıra Kanban ve diğer çevik pratiklerinin de ciddi bir yer tuttuğu gözlemlenmektedir, bu da çevik metodolojilerin çeşitli ve esnek uygulama potansiyelini göstermektedir.

Araştırma bulguları, Türkiye'deki yazılım geliştirme sektöründe Scrum metodolojisinin %77,6 oranında kullanılarak ne derece baskın ve etkili bir çözüm olduğunu ortaya koymaktadır. Bu oran, Scrum'un karmaşık projelerde takımların esneklik ve çeviklik ihtiyaçlarını karşılamada ne kadar başarılı olduğunu göstermektedir. Ekip büyüklükleri incelendiğinde, katılımcıların çoğunun 3 ila 10 kişilik ekiplerde çalıştığı (%59,5), büyük ekiplerin ise %40,5'ini oluşturduğu belirlenmiştir, bu durum çevik metodolojilerin farklı ekip büyüklüklerine olan uyum yeteneğini ve geniş kullanım alanını göstermektedir. Diğer çevik metodolojileri (Kanban, Lean Software Development, Test-Driven Development ve Feature Driven Development) daha düşük kullanım oranlarına sahip olmasına rağmen, belirli projeler ve özel ihtiyaçlar doğrultusunda tercih edildiklerini yansıtmaktadır. Bu çeşitlilik, çevik metodolojilerin projelerin gereksinimlerine göre nasıl esnek şekilde şekillendirilebildiğini vurgulamakta ve Scrum'un Türkiye'deki yazılım geliştirme sektöründe bir standart haline geldiğini, çevik pratiklerin sektördeki sürekli evrim ve gelişim ihtiyacına cevap verdiğini göstermektedir.

Araştırma sonuçları, Türkiye'deki yazılım geliştirme sektöründe Scrum metodolojisinin ve özellikle Scrum panosunun kullanımının yüksek oranda (%94,7) benimsendiğini göstermektedir. Bu durum, Scrum panosunun, işlerin durumunu açık bir şekilde izleyerek ekipler arasında iletişimi ve işbirliğini nasıl kolaylaştırdığını ve bu aracın çevik çalışma yöntemlerindeki önemini vurgulamaktadır. İş takibi için en yaygın

kullanılan araç olarak Jira (%77,7), çevik projelerin yönetimindeki tercih edilen seçenek olup, iş akışını düzenleyerek ve görsel olarak sunarak projelerin verimli yönetilmesine katkıda bulunmaktadır. Katılımcıların %94,6'sının Scrum metodolojisinden duyduğu memnuniyet, metodolojinin iş akışını optimize etme ve projeleri etkili bir şekilde yönetme kapasitesini yansıtmaktadır. Ancak, memnun olmayan azınlık (%5,4), Scrum'un her proje veya ekip yapısında ideal olmayabileceğine işaret etmektedir. Ayrıca, ekiplerin %83,6'sında Scrum Master varlığı gözlemlenmiş, bu da Scrum Master'ın projelerdeki kritik rolünü ve çevik süreçlerinin yönetimindeki önemini belirtmektedir. Sonuç olarak, bu bulgular, Scrum metodolojisinin Türkiye'deki yazılım geliştirme sektöründe nasıl etkin bir şekilde benimsendiğini ve yüksek memnuniyet oranlarıyla projelerin başarılı bir şekilde yönetilmesine olanak sağladığını ortaya koymaktadır.

Araştırma sonuçları, Scrum Master görevinin Türkiye'deki yazılım geliştirme ekiplerinde çeşitli roller tarafından üstlenildiğini, özellikle analistlerin bu rolü %49,5 oranında en sık üstlenen grup olduğunu göstermektedir. Bu bulgu, analistlerin analitik yetenekleri ve problem çözme becerilerinin, Scrum süreçlerinin yönetimine ne kadar uygun olduğunu ortaya koymaktadır. Diğer taraftan, Developer ve Tester gibi teknik odaklı rollerin bu görevi daha az üstlenmeleri, bu pozisyonlardaki profesyonellerin projelerdeki teknik ve operasyonel işlere daha fazla odaklanmalarından kaynaklanıyor olabilir. Ek olarak, katılımcıların %61,1'i Scrum Master rolünü üstlenmek istemediğini belirtmiş, bu durum bu rolün karşılaştığı zorlukları ve liderlik, organizasyon becerileri gibi gereklilikleri yansıtmaktadır. Ancak, %38,9'luk bir kesim bu rolü üstlenmeye istekli olduğunu ifade etmiş, bu da liderlik ve yönetim fırsatlarını değerlendirmeye açık olan profesyonellerin varlığını göstermektedir. Sonuç olarak, bu bulgular, Scrum Master rolünün, farklı iş disiplinlerinden bireyler tarafından üstlenebildiğini ve bu rolün ekiplerdeki işleyiş ve verimlilik üzerinde önemli bir etkiye sahip olduğunu göstermekte; aynı zamanda bu rolün getirdiği zorluklar ve sorumluluklar konusunda bir farkındalık olduğunu işaret etmektedir.

Araştırma, Türkiye'deki yazılım geliştirme projelerinde Scrum metodolojisinin etkinliklerinin geniş ölçekte benimsendiğini ve yüksek katılım oranlarına sahip olduğunu ortaya koymaktadır. Sprint Planning ve Daily Scrum Meeting gibi etkinliklerin %100 katılım oranıyla uygulanması, bu süreçlerin projelerde temel ve vazgeçilmez unsurlar olduğunu göstermektedir. Bu etkinlikler, ekiplerin günlük ve

sprint bazında hedeflerini sürekli gözden geçirmelerini ve güncellemelerini sağlayarak, iş akışının etkin yönetimine katkıda bulunur. Ayrıca, Sprint Review ve Sprint Retro gibi sürekli iyileştirme ve değerlendirme süreçlerine yüksek oranda (%93,1 ve %94,8) katılım gösterilmesi, projelerdeki sürekli gelişim çabalarının ve öğrenme döngülerinin önemini vurgular. Diğer yandan, Sprint Grooming (Product Backlog Refinement) etkinliğinin %87,9 oranında uygulanması, bu sürecin diğerlerine göre daha az yaygın olduğunu ve bazı projelerde arka plan düzenlemelerine daha az zaman ayırıldığını işaret edebilir. Daha az bilinen diğer Scrum etkinliklerinin çok düşük oranda (%2,6) uygulanması, çoğu ekibin standart Scrum süreçlerine sadık kaldığını göstermektedir. Bu bulgular, Scrum metodolojisinin projelerde ne kadar etkin bir şekilde kullanıldığını ve ekip işbirliği ile sürekli gelişim için sağladığı kritik yapıyı göstermekte, ayrıca projelerin başarılı bir şekilde yürütülmesine olanak tanıyan sistemli bir yaklaşım sunmaktadır.

Araştırma sonuçları, Türkiye'deki yazılım geliştirme projelerinde Scrum metodolojisinin uygulanışına dair detaylı bilgiler sunarak, bu süreçlere olan yoğun katılımı ve sistematik uygulamayı ortaya koymaktadır. Sprint süreleri genellikle üç hafta (%69,8) olup, bu uzunluk, projelerin geniş kapsamlı işleri yönetmesine imkan sağlar, ancak daha kısa sprintler (%29,3) daha hızlı teslimat ve çeviklik sunarken, uzun süreler (%0,9) nadiren tercih edilmektedir. Sprint planlama toplantılarına yüksek katılım (%79,3) ve toplantı sürelerinin çoğunlukla 60-120 dakika arasında olması, etkin planlama ve koordinasyon için yeterli zaman tanıdığını gösterir. Sprint hedeflerinin bazen belirlenmemesi (%66,1), bazı projelerde hedef odaklı olmayan planlamalara yol açabilirken, test eforunun planlamada özellikle dikkate alınması (%79,1), yazılım geliştirme süreçlerinde kalite kontrolünün entegrasyonunun ve öneminin arttığını belirtir. Günlük Scrum toplantılarının tam katılımla (%100) ve çoğunlukla 30-60 dakika süreyle düzenli olarak yapılması, ekipler arası iletişimin ve günlük ilerlemenin ne kadar önemli olduğunu vurgular. Bu bulgular, projelerin yönetiminde çeviklik ve adaptasyonun kritik önemini gösterirken, Scrum süreçlerinin projelerin başarılı yönetimi için ne kadar hayati olduğunu da ortaya koyar.

Araştırma sonuçları, günlük Scrum toplantılarının, katılımcıların projedeki günlük ilerlemeleri etkin bir şekilde takip etmelerini ve ekip içi işbirliğini desteklemelerini sağlayan kritik bir iletişim aracı olarak işlev gördüğünü göstermektedir. Toplantılarda, katılımcılar genellikle önceki gün ne yaptıklarını, o gün ne yapacaklarını ve

karşılaştıkları engelleri (%99,1) belirterek, günlük ilerlemelerini paylaşmakta ve takım içindeki sorunları çözmek için işbirliği yapmaktadırlar. Ayrıca, "Son toplantıdan sonra ne yaptım?" (%82,8) ve "Bir sonraki toplantıya kadar ne yapmayı planlıyorum?" (%85,3) sorularıyla, ekip üyeleri önceki toplantılardan gelen işleri ve sonraki adımları netleştirerek projenin sürekliliğini sağlamaktadırlar. Toplantıların odaklı ve etkin bir şekilde ilerlediğini (%1,7) belirten katılımcı yorumları, toplantıların projedeki günlük ilerlemelerin takibi için uygun ve verimli bir ortam sunduğunu vurgular. Bu bulgular, günlük Scrum toplantılarının, projelerde düzenli ilerleme izleme ve engellerin çözülmesine olanak tanıyan önemli bir rol oynadığını ortaya koymaktadır.

Araştırma sonuçları, Türkiye'deki yazılım geliştirme projelerinde sprint review toplantılarının genel uygulama durumuna dair farklılıklar göstermektedir. Bu toplantıları düzenleyen katılımcıların büyük çoğunluğu (%84,3) olmakla birlikte, bir kısmı (%10,4) bu toplantıları yapmamakta veya düzensiz olarak gerçekleştirmekte olduğunu belirtmiştir. Toplantıların süresi genellikle uzun olup (%68,7), 2 saatten fazla sürdüğü gözlemlenmiştir, bu durum sprint süresince yapılan çalışmaların detaylı bir şekilde gözden geçirildiğini ve kapsamlı geri bildirimlerin alındığını işaret etmektedir. Ayrıca, sprint içinde tamamlanan geliştirmelerin demo yapılarak gösterilmesi konusunda katılımcıların yarısından fazlası (%55,3) bazen demo yapıldığını, bazen yapılmadığını belirtmiştir. Bu, demo uygulamalarının her sprint sonunda tutarlı bir şekilde gerçekleştirilmediğini göstermektedir. Genel olarak, bu bulgular sprint review toplantılarının düzenlenme sıklığı, süresi ve içeriğinde çeşitlilik olduğunu gösterirken, bu toplantıların projelerde önemli bir rol oynadığını ve genellikle uzun süreler boyunca detaylı değerlendirmeler yapıldığını vurgulamaktadır.

Araştırma, sprint review toplantılarında ürün demosunun gerçekleştirilme biçimine dair önemli bulgular ortaya koymuştur. Toplantılarda, ürün demosunu gerçekleştirenlerin büyük bir çoğunluğunu analist/çözümleyiciler (%83,1) ve test/kalite kontrol ekipleri (%80,9) oluşturmaktadır, bu durum bu rol sahiplerinin işlevsel ve teknik detaylara olan hakimiyetlerinin bir göstergesi olarak değerlendirilebilir. Yazılım geliştiriciler (%24,7) ve product owner'lar (%21,3) daha az sıklıkla bu görevi üstlenirken, proje yöneticilerinin katılımı (%1,1) oldukça düşüktür. Bu rol dağılımı, sprint review toplantılarında demosunun çoğunlukla analistler ve kalite kontrol ekipleri tarafından yapıldığını ve bu süreçlerin ekipler tarafından işlevsel gereklilikler ve kalite standartları açısından detaylı bir değerlendirilme fırsatı olarak görüldüğünü göstermektedir. Ayrıca,

sprint review toplantılarının genellikle uzun süreli (%68,7'lik bir kesim 2 saatten uzun toplantıları tercih etmekte) ve zaman zaman ürün demosunun planlanmadığı veya yapılmadığı (bazı durumlarda belirsizlik %55,3) şekilde gerçekleştiği belirtilmektedir. Bu durum, süreçlerin düzenli olarak ve belirlenen standartlara uygun şekilde yürütülmesi gerektiğini ve süreç optimizasyonu için potansiyel alanların olduğunu işaret etmektedir. Bu bulgular, sprint review süreçlerinin projelerdeki kritik önemini ve ürün demosunun işlevsellik ile kalite kontrol odaklı ekipler tarafından yönetilmesinin önemini vurgulamaktadır.

Araştırma sonuçları, sprint retrospektif (retro) toplantılarının, Türkiye'deki yazılım geliştirme projelerinde genel olarak yaygın olarak uygulandığını, ancak bazı ekiplerin bu toplantıları düzensiz olarak ya da hiç yapmadıklarını göstermektedir. Katılımcıların büyük çoğunluğu (%88,7) bu toplantıları düzenli olarak gerçekleştirdiklerini belirtirken, %5,2'si bu toplantıların bazen yapıldığını veya yapılmadığını ifade etmiştir. Bu, tüm ekipler arasında bir konsensus eksikliğine işaret etmektedir. Toplantıların uzunluğu genellikle 2 saat ve üzerinde (%61,6) olduğu belirtilmiş, bu durum ekiplerin geçmiş sprintler üzerinde detaylı analizler yaparak, gelecekteki sprintler için iyileştirme fırsatlarını değerlendirmek için yeterli zaman ayırdıklarını göstermektedir. Toplantıları çoğunlukla Scrum Master'ın (%89,0) yönettiği, daha az oranda ise teknik liderler (%2,8) ve bazen ürün sahibi ile iş analistlerinin bu rolü üstlendiği görülmüştür. Bu bulgular, sprint retro toplantılarının projelerdeki önemini ve bu süreçlere ayrılan zamanın kritikliğini vurgulamakta, aynı zamanda süreklilik ve sistematik iyileştirme için daha düzenli bir yaklaşımın gerekliliğini ortaya koymaktadır.

Araştırma sonuçları, Kanban metodolojisinin iş takibi için Türkiye'deki yazılım geliştirme projelerinde yaygın olarak kullanıldığını ve katılımcıların büyük çoğunluğunun (%87,5) bu yöntemi iş süreçlerini görsel ve yönetilebilir kılmak amacıyla tercih ettiğini göstermektedir. Kanban panosu için en çok tercih edilen araç Jira (%50,0) olup, diğer araçlar da (%37,5) ekiplerin farklı ihtiyaç ve tercihlerine göre kullanılmaktadır. Katılımcıların %40'ı bir Kanban Master'a sahipken, %60'ı bu role sahip değildir. Bununla birlikte, katılımcıların çoğunluğu (%57,1) Kanban Master'ın varlığını önemli bulurken, %21,4'ü bu konuda kararsız ve %21,4'ü gerekli olmadığını düşünmektedir. Kanban Master genellikle iş analistleri (%50,0) tarafından üstlenilmekte, ancak ürün sahipleri (%25,0) ve yazılım geliştiriciler (%8,3) de bu rolü alabilmektedir. Bu rol dağılımı, Kanban Master'ın ekibin dinamiklerine ve ihtiyaçlarına

bağlı olarak değişkenlik gösterdiğini gösterir. Ayrıca, katılımcıların büyük bir çoğunluğu (%90,9) Kanban metodolojisinden memnun olduklarını belirtmiş, iş akışının görselleştirilmesi, işlerin önceliklendirilmesi ve sürekli iyileştirme fırsatları gibi olumlu yönlerine vurgu yapmışlardır. Ancak, küçük bir kesim (%9,1) bu konuda kararsızdır veya yeterli bilgiye sahip değildir, bu da metodolojinin tam potansiyelinin herkes tarafından tam olarak anlaşılmadığını işaret etmektedir.

Araştırma, yazılım projelerindeki test süreçlerinin kritik yönlerini ve mevcut zorlukları ortaya koymaktadır. Fonksiyonel testlerin büyük çoğunluğu manuel olarak (%98,3) yürütülmekte ve dış kaynak kullanımı oldukça sınırlıdır (%52,2). Projelerde analistlerin aktif rol alması (%98,4) gözlemlenirken, bu analistlerin geliştirme sonrası kod testleri konusunda bölünmüş görüşlere sahip oldukları belirtilmektedir, bu da test süreçlerini yönetme ve test ekibi desteği arasında seçim yapma gereksinimi doğurmaktadır. Geliştiriciler genellikle kendi kodlarını test etmektedir (%84,1), ancak test dokümantasyonu ve test senaryolarının incelenmesi gibi temel adımlar sıklıkla ihmal edilmektedir. Jira, hata kayıtlarının tutulduğu en yaygın araç olarak öne çıkmakta (%67,7) ve geliştiriciler bu aracı etkin bir şekilde kullanabilmektedir (%96,2). Ancak, test süreçlerinin yetersiz belgelenmesi ve test senaryolarının gözden geçirilmesinin atlanması projelerin genel kalitesini olumsuz yönde etkileyebilir. Bu bulgular, yazılım projelerinde test süreçlerinin etkinliğini artırma ve kaliteyi iyileştirme yönünde önemli adımların atılması gerektiğini işaret etmekte; bu iyileştirmeler arasında daha kapsamlı test dokümantasyonu, düzenli test senaryosu incelemeleri ve test ekiplerinin daha etkin yönetilmesi yer almaktadır.

Araştırma, Scrum metodolojisi kullanılan yazılım projelerinde test süreçlerinin yönetimi ve iletişim altyapılarına dair önemli bilgiler sunmaktadır. Test ekibinin, planlama toplantılarına neredeyse tamamen (%99,0) katılımı, projenin ilerleyişi ve test faaliyetlerinin etkin şekilde planlanmasına katkı sağlamaktadır. Test süreçlerinin dokümantasyonu için yaygın olarak Jira ve Excel kullanılmakta, bu araçlar test senaryolarının oluşturulması, izlenmesi ve test sonuçlarının raporlanması gibi temel işlevleri desteklemektedir. Test senaryolarının yazılması yaygın bir uygulama olmakla birlikte (%84,7), bu senaryoların gözden geçirilmesi sürecinde belirsizlikler mevcuttur; katılımcıların yarısı (%50,0) bu konuda net bir bilgi vermemiştir, bu durum test kalitesi güvencesi süreçlerine daha fazla dikkat edilmesi gerektiğine işaret etmektedir. İletişimde ise, Microsoft Teams ve Slack gibi araçlar ekip işbirliğini ve bilgi paylaşımını

kolaylaştırarak ön plana çıkmakta, ancak Cisco Jabber ve Skype gibi diğer araçlar daha az tercih edilmektedir. Bu bulgular, yazılım projelerinde test süreçlerinin yönetimi ve iletişim için uygun araçların stratejik seçimi ve etkin kullanımının, projelerin genel kalitesini artırmak adına kritik öneme sahip olduğunu vurgulamaktadır.

Araştırma, yazılım projelerinde test ekibinin işlevlerini, kullandıkları araçları ve performans değerlendirme yöntemlerini kapsamlı bir şekilde incelemektedir. Test süreçlerinde Jira ve Excel gibi araçlar yaygın olarak kullanılmakta, bu araçlar hata kayıtlarının tutulması, hata yönetiminin organize edilmesi ve geliştirme sürecinde izlenebilirliğin artırılması için tercih edilmektedir. Test ekibi genellikle hata araçlarını etkin bir şekilde kullanırken, bazı durumlarda doğrudan iletişimi tercih ederek hata detaylarına erişim sağlamaktadır. Test süreçlerinin performans değerlendirmesi, genellikle Test Yöneticisi tarafından yapılırken, bazen Proje Yöneticisi veya diğer rollerin de bu değerlendirmeyi gerçekleştirdiği belirtilmiştir. Kullanılan test metrikleri arasında test case sayısı, hata sayısı ve canlı ortamdan dönen hata sayısı gibi ölçütler bulunmakta, ancak bu metriklerin kullanımı konusunda belirsizlikler mevcuttur. Test ekibi, projelerde çeşitli ek roller üstlenmekte ve smoke testleri ile kullanıcı kabul testlerinin yönetilmesinde önemli görevler ifa etmektedir. Ancak, bu süreçlerde belirsizlikler ve bilgi eksiklikleri, özellikle belirli test süreçlerinde önemli faktörler olarak dikkate alınmalıdır. Sonuç olarak, bu çalışma, test ekiplerinin projelerdeki rollerini ve sorumluluklarını, kullanılan araçları ve performans değerlendirme pratiklerini detaylı bir şekilde ortaya koymakta, test süreçlerinin yönetimi ve optimizasyonu için daha belirgin ve sistemli bir yaklaşımın önemini vurgulamaktadır.

Kullanıcı kabul testlerini gerçekleştiren rollerin incelenmesi, çeşitli paydaşların bu önemli sürece katılımını göstermektedir. Analizimize göre, kullanıcı kabul testlerini çoğunlukla analistler (%48,6) ve iş birimleri (%39,6) gerçekleştirmektedir. Bu sonuçlar, kullanıcı kabul testlerinin genellikle proje yönetimi ve iş gereksinimlerinin doğru bir şekilde karşılanmasından sorumlu olan analistler ve iş birimleri tarafından yürütüldüğünü göstermektedir. Ayrıca, bağımsız test ekipleri de (%21,6) kullanıcı kabul testlerine katılmaktadır. Bu, test sürecinin tarafsızlığını ve kalitesini artırma amacıyla dış kaynaklı uzmanların dahil edildiğini işaret etmektedir. Developerlar ve son kullanıcılar da kullanıcı kabul testlerine katılmaktadır, ancak bu rollerin katılım oranı daha düşüktür (%4,5). Bununla birlikte, bu rollerin test sürecine katılması, uygulamanın işlevselliği ve kullanılabilirliği açısından önemlidir. Sonuç olarak, kullanıcı kabul

testlerinin farklı roller tarafından gerçekleştirilmesi, test sürecinin çeşitli perspektiflerden değerlendirilmesine ve uygulamanın gereksinimlere uygunluğunun sağlanmasına yardımcı olmaktadır. Bu çeşitlilik, test sürecinin kapsamlı ve etkili bir şekilde yürütülmesini sağlamaktadır.

Araştırma, yazılım projelerinde otomasyon testlerinin uygulanışını ve etkinliğini derinlemesine incelemekte olup, katılımcıların çoğunluğunun (%62,9) otomasyon testlerini aktif olarak kullandığını göstermektedir. Test süreçlerinde özellikle regresyon, yük ve performans testlerinin yanı sıra veri oluşturma için otomasyon tercih edilmekte, bu süreçlerin çoğunlukla DevOps ile entegre edildiği belirtilmektedir. Otomasyon testleri için sıklıkla Jira, Excel ve Selenium Framework gibi araçlar kullanılmakta ve bu testler genellikle web ve uygulama kanallarında yoğunlaşmaktadır. Test ortamı, preprod ve prod ortamları en çok kullanılan test alanları olarak öne çıkmakta, ancak projelerin geliştirme ve stable ortamları daha az kullanılmaktadır. Katılımcıların büyük çoğunluğu otomasyon testlerinin yönetiminde belirlenmiş regresyon senaryo setlerine sahip olduğunu ifade etmiş, bu setlerin düzenli olarak çalıştırıldığı ve otomasyon süreçlerinin yönetimi için çeşitli araçların kullanıldığı belirtilmiştir. Bu bulgular, otomasyon testlerinin yazılım geliştirme süreçlerinde önemli bir rol oynadığını, test süreçlerinin verimliliğini ve etkinliğini artırdığını ve projelerdeki hata yönetimi ve kalite kontrol süreçlerine katkıda bulunduğunu göstermektedir. Ayrıca, bu testlerin sıklıkla ve düzenli aralıklarla uygulandığı, ancak performans testleri ve bazı test ortamlarının kullanımının iyileştirilmesi gerektiği anlaşılmaktadır.

Araştırmanın sonuçları doğrultusunda öneriler aşağıda sıralanmıştır;

Uygulayıcılar İçin Öneriler:

- Otomasyon Testlerinin ve Scrum Uygulamalarının Entegrasyonunun Optimizasyonu: Scrum süreçlerine otomasyon testlerinin entegrasyonu, test süreçlerinin verimliliğini artırabilecek ve projelerin zamanında ve hatasız teslim edilmesine olanak tanıyabilecek şekilde derinleştirilmelidir. Bu bütünleşme, özellikle regresyon testlerinin otomasyonu üzerinden, süreç hızının ve hata tespit kabiliyetlerinin geliştirilmesini sağlamalıdır.

- Kanban Uygulamalarında Araç Kullanımının İyileştirilmesi: Kanban metodolojisinde kullanılan araçlar, iş süreçlerini görselleştirmede ve yönetimde önemli role sahiptir. Jira ve benzeri araçların etkin kullanımı, süreçlerin şeffaflığını ve takip

edilebilirliğini maksimize edebilir. Kanban Master rolleri, bu araçları kullanarak iş akışını düzenlemede ve ekip üyeleri arasında sürekli bir iletişim ve işbirliği sağlamada kritik bir işlev görebilir.

- Test Süreçlerinin Dokümantasyonunun Geliştirilmesi: Test süreçlerinin eksiksiz bir şekilde dokümante edilmesi, hata yönetimini ve kalite kontrol süreçlerini iyileştirebilir. Özellikle, test süreçlerinde gerçekleştirilen adımların, hata kayıtlarının ve çözüm süreçlerinin detaylı bir şekilde belgelenmesi, bu süreçlerin gelecekte daha etkin bir şekilde yönetilmesine olanak sağlayabilir.

- Test Metriklerinin Geliştirilmesi ve Kullanılması: Test süreçlerinin performansını ölçmek için metriklerin geliştirilmesi ve bu metriklerin sistemli bir şekilde kullanılması gerekmektedir. Bu, 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.

Araştırmacılar İçin Öneriler:

- Otomasyon Testlerinin Etkinliği Üzerine Kapsamlı Araştırmalar Yapılması: Otomasyon testlerinin yazılım geliştirme süreçlerindeki etkilerini daha detaylı analiz etmek amacıyla kapsamlı çalışmalar yapılmalıdır. Özellikle, otomasyonun regresyon testleri ve DevOps entegrasyonu üzerindeki etkileri, bu süreçlerin verimliliğine ve projelerin başarısına olan katkıları açısından incelenmelidir.

- çevik ve Kanban Metodolojilerinin Entegrasyonu Üzerine Araştırmalar: çevik ve Kanban metodolojilerinin bir arada kullanılmasının avantajları ve zorlukları üzerine odaklanan araştırmalar yapılmalıdır. Hibrit modellerin etkinliği ve bu modellerin projeler üzerindeki etkileri, yazılım geliştirme süreçlerindeki verimlilik artışı bağlamında değerlendirilmelidir.

- Test Süreçlerinde İnovasyon ve İyileştirme Yöntemlerinin Geliştirilmesi: Test yönetiminde ve otomasyon süreçlerindeki inovasyon fırsatları üzerine yoğunlaşılmalıdır. Özellikle, test süreçlerinin standardizasyonu ve otomasyonun faydaları üzerine daha detaylı incelemeler yapılmalıdır. Bu incelemeler, test süreçlerinin daha etkin ve verimli bir şekilde yürütülmesine katkıda bulunabilir.

# KAYNAKÇA


Aksoy, A. (2017). *Proje yönetiminde Kanban kullanımı*. Proje Yönetimi Seminer Notları.

Bass, J. M. (2013, August). Agile method tailoring in distributed enterprises: Product owner teams. In *2013 IEEE 8th international conference on global software engineering* (pp. 154-163). IEEE.

Bassil, Y. (2012). A simulation model for the waterfall software development life cycle. *International Journal of Engineering & Technology (IJET)*, 2(5), 2049-3444.

Bergmann, T. (2018). *The relationship between Agile project management and project success outcomes* (Electronic Theses and Dissertations). University of Central Florida.

Bozkurt, E. (2018). *Kanban ve Yalın Düşünce*. İstanbul: Yalın Yayıncılık.

Chow, T., & Cao, D. B. (2008). A survey study of critical success factors in agile software projects. *Journal of Systems & Software*, 81, 961-971.

Fedorova, V. A., Moiseeva, T. A., & Poddubnaya, E. V. (2018). Methods of selecting acceptable strategies of software development. *Issues of Radio Electronics*, (11), 33-39.

Fertalj, K., & Katic, M. (2008). An overview of modern software development methodologies. In *Central European Conference on Information and Intelligent Systems* (p. 1). Faculty of Organization and Informatics Varazdin.

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

Graham, D., Veenendaal, E. V., Evans, I., & Black, R. (2008). *Foundations of software testing: ISTQB certification*. Intl Thomson Business Press.

Hardyanto, W., Purwinarko, A., Sujito, F., & Alighiri, D. (2017, March). Applying an MVC framework for the system development life cycle with waterfall model extended. In *Journal of Physics: Conference Series* (Vol. 824, No. 1, p. 012007). IOP Publishing.

Hardyanto, W., Purwinarko, A., Sujito, F., & Alighiri, D. (2017). Extending the waterfall model with model-view-controller architecture. *Journal of Software Engineering and Applications*, 10(5), 423-435.

Hoda, R., Noble, J., & Marshall, S. (2011). The impact of team communication and role clarity on agile software development project success. *Empirical Software Engineering*, 16(3), 335-373.

Kate, V., Bhalerao, S., & Sharma, V. (2023, December). 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)* (pp. 1-11). IEEE.

Khairi, A. M. M., Kamaruddin, M. Z., & Widyarto, S. (2016, May). Agile methodology software development: A brief review. In *Proceedings of the Informatics Conference* (Vol. 2, No. 2).

Kumar, G., & Bhatia, P. K. (2014, February). Comparative analysis of software engineering models from traditional to modern methodologies. In *2014 Fourth International Conference on Advanced Computing & Communication Technologies* (pp. 189-196). IEEE.

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).

Kuzuloğlu, S. (2016). Kanban sistemi ve iş yönetimi. *Teknoloji ve Yönetim Dergisi*.

Larusdottir, M., Cajander, Å., Gregory, P., Cockton, G., Salah, D., Kuusinen, K., & Nauwerck, G. (2016, October). Stakeholder involvement in agile software development. In *Proceedings of the 9th Nordic Conference on Human-Computer Interaction* (pp. 1-3).

Lee, G., & Xia, W. (2010). Toward agile: An integrated analysis of quantitative and qualitative field data on software development agility. *MIS Quarterly*, 34(1), 87-114.

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.

Mateen, A., Tabassum, M., & Rehan, A. (2017). Combining agile with traditional V model for enhancement of maturity in software development. *arXiv preprint arXiv:1702.00126*.

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

Misra, S. C., Kumar, V., & Kumar, U. (2009). Identifying some important success factors in adopting agile software development practices. *Journal of Systems and Software*, 82(11), 1869-1890.

Mitchell, J. L., & Black, R. (2015). *Advanced software testing-vol. 3: Guide to the ISTQB advanced certification as an advanced technical test analyst*. Rocky Nook, Inc.

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).

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.

Serrador, P., & Pinto, J. K. (2015). Does agile work?—A quantitative analysis of agile project success. *International Journal of Project Management*, 33(5), 1040-1051.

Shastri, Y., Hoda, R., & Amor, R. (2017, February). Understanding the roles of the manager in agile project management. In *Proceedings of the 10th Innovations in Software Engineering Conference* (pp. 45-55).

Sneha, K., & Malle, G. M. (2017, August). Research on software testing techniques and software automation testing tools. In *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)* (pp. 77-81). IEEE.

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

Sultania, A. K. (2015, February). 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)* (pp. 1-4). IEEE.

Szatmári, A., Gergely, T., & Beszédes, Á. (2023, April). ISTQB-based software testing education: Advantages and challenges. In *2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)* (pp. 389-396). IEEE.

Tam, C., da Costa Moura, E. J., Oliveira, T., & Varajão, J. (2020). The factors influencing the success of on-going agile software development projects. *International Journal of Project Management*, 38(3), 165-176.

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.

Yıldırım, M. (2019). *İş süreçlerinde verimlilik ve Kanban*. Verimlilik Dergisi.

Zhao, Y., Hu, Y., & Gong, J. (2021, October). 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)* (pp. 56-62). IEEE.

# EKLER

## Bilgisayar Mühendisliği Yüksek Lisans Tezi için Hazırlanan Görüşme Formu

**Demografik Sorular: *Katılımcıların demografik dağılım grafiği için hazırlanmış sorular**
1.Cinsiyetiniz nedir? (Kız, Erkek)
……..

2.Kaç yaşındasınız?
……..

3. Eğitim Durumunuz nedir?
……..

4.Şu an devam etmekte olan yüksek lisans/doktora eğitimleriniz var mı? Eğer var ise hangi üniversite hangi bölümde yapmaktasınız?
……..

5.Şu an bir firmada ya da kurumda aktif olarak çalışıyor musunuz?
……..

6.Çalışmakta olduğunuz firmanın ismi nedir? (Cevaplamak istemezseniz boş bırakabilirsiniz)
……..

7.Şu an ki iş durumunuz nedir? (İşsiz ve iş arıyorum, Öğrenciyim, Tam zamanlı, Yarı zamanlı, Serbest çalışanım, Emekliyim)
……..

**Öğrenci ise aşağıdaki soruyu cevaplayabilirsiniz. Öğrenci değilseniz bu soruyu atlayabilirsiniz.**

- **a)** Hangi bölümde okumaktasınız?
- **b)** Kaçıncı sınıftasınız?
- **c)** Okuldan mezun olduğunuzda hangi iş alanda ilerlemek istersiniz? (Analiz, Yazılım, Test)
- **d)** …….

8. Şu anki çalışma modeliniz nedir? (Uzaktan, Hibrit (Ofis+Uzaktan), Ofiste)
……..
9. Şu an ki iş yerinizin hangi şehirdedir?
……..

10.İş yeri İstanbul ise İstanbul'un hangi yakasında bulunmaktadır? (Anadolu -Avrupa Yakası mı?)
……..

11.Firmadaki rolünüz nedir? (Proje yöneticisi, Analist/Çözümleyici, Mimar- teknik lider, Yazılım Geliştirici, Test Kalite Kontrol, Product Owner, Scrum Master…)
……..

12. İş tecrübeniz ne kadar?
……..

13.Manuel yazılım test tecrübeniz oldu mu? Test tecrübeniz olduysa ne kadar süreliğine oldu?

a) Evet yaklaşık 6 ay kadar
b) Evet yaklaşık 1 sene kadar
c) Hayır hiç test tecrübem olmadı
d) Diğer

14.Otomasyon yazılım test tecrübeniz oldu mu? Otomasyon test tecrübeniz ne kadar süreliğine oldu?

a) Evet yaklaşık 6 ay kadar
b) Evet yaklaşık 1 sene kadar
c) Hayır hiç test tecrübem olmadı
d) Diğer

15. Şu an hangi sektör üzerine çalışıyorsunuz? (Banka, Telekomünikasyon, E-Ticaret, İnsan Kaynakları, Diğer)

……..

16.Daha önce deneyimlediğiniz farklı sektörler var mıdır varsa nelerdir?

……..

17. Daha önceden almış olduğunuz sertifikalar var mıdır, varsa bu sertifikalar nelerdir?

……..

18. Hangi sertifikaları almayı tercih edersiniz?

……..

19. Yaklaşık aylık gelir aralığınız nedir? (Cevaplamak istemezseniz boş bırakabilirsiniz)

……..

**AGİLE METODOLOJİLERİ ÜZERİNE DETAYLI SORULAR**

1. Yazılım geliştirme modellerinden hangilerini kullanıyorsunuz?
   **a)** Waterfall (Şelale) Modeli
   **b)** Code&Fix Modeli
   **c)** Waterfall Iterative Model
   **d)** V Model
   **e)** Iterative Model
   **f)** Incremental Model
   **g)** Evolutionary Model
   **h)** Sprital Model
   **i)** Rational Unified Process
   **j)** Agile (Çevik) Model
   **k)** ……..

2. Yazılım geliştirme modellerinden hangi modelin daha iyi olduğunu düşünüyorsunuz? Nedenini açıklayabilir misiniz?

   ……

3. Şirkette genel olarak hangi Agile metodolojilerini kullanıyorsunuz?
   **a)** Scrum
   **b)** Kanban
   **c)** Extreme Programing
   **d)** Lean Software Development
   **e)** Test-Driven Development
   **f)** Feature Driven Development
   **g)** Dynamic Systems Development Method
   **h)** ……..

4. Çalıştığınız projede ağırlıklı olarak hangi Agile metodolojisini kullanıyorsunuz?
   **a)** Scrum
   **b)** Kanban
   **c)** Extreme Programing
   **d)** Lean Software Development
   **e)** Test-Driven Development
   **f)** Feature Driven Development
   **g)** Dynamic Systems Development Method

**h)** …….

5. Size göre projeniz için hangi Agile metodolojisi daha uygun olurdu?
……

6. Ekibiniz kaç kişiden oluşuyor? (0-3, 3-10, 10 ve üzeri)
……..

**SCRUM METODOLOJİSİ ÜZERİNE DETAYLI SORULAR**

1. İş takibini yapabilmek için Scrum panosu kullanıyor musunuz?
……

2. Scrum panosu için hangi toolu kullanıyorsunuz? (Jira, miro ya da şirketin geliştirdiği şirket özelinde kullanılan proje yönetim modeli gibi)
……

3. Scrum metodolojisi ile ilerlemekten memnun musunuz? Olumlu olumsuz yanları nelerdir?
……

4. Ekibinizde Scrum Master var mı?
……..

5. Ekibinizde Scrum Master görevini hangi rol üstleniyor?
   **a)** Analist
   **b)** Tester
   **c)** Developer
   **d)** Product Owner
   **e)** ……

6. Ekibinizdeki Scrum Master görevini üstlenmek ister miydiniz?
……

7. Projenizde kullanılan Scrum etkinlikleri hangileridir?
   **a)** Sprint Planing
   **b)** Daily Scrum Meeting
   **c)** Sprint Review
   **d)** Sprint Retro
   **e)** Bilmiyorum, fikrim yok
   **f)** Hepsi
   **g)** ……

8. Sprint süreleriniz ne kadar?
   **a)** 1-2 hafta
   **b)** 3 hafta
   **c)** 3 hafta ve üzeri
   **d)** ……

9. Sprint planlama toplantılarına tüm ekip katılım sağlıyor mu?
   **a)** Evet
   **b)** Hayır
   **c)** Bazen tüm ekip katılım sağlıyor bazen sağlayamıyor
   **d)** ……..

10. Sprint planlama toplantılarınız ne kadar sürüyor?
    **a)** 30 dakika
    **b)** 30-60 dakika
    **c)** 60-120 dakika
    **d)** 2 saat ve üzeri
    **e)** Bilmiyorum, fikrim yok

**f)** …….

11. Sprint planlama toplantılarında ilgili sprint için sprint hedefleri belirliyor musunuz?
**a)** Evet
**b)** Hayır
**c)** Bazen belirleniyor bazen belirlenmiyor
**d)** Bilmiyorum, fikrim yok
**e)** …….

12. Sprint planlama toplantılarında görev planlaması yaparken test eforu için de ayrı bir planlama yapıyor musunuz?
**a)** Evet
**b)** Hayır
**c)** Bazen yapılıyor bazen yapılmıyor
**d)** Bilmiyorum, fikrim yok
**e)** …….

13. Sizce sprint planlama toplantılarında görev planlaması yaparken test eforu için de ayrı bir efor verilmeli mi?

……

14. Günlük Scrum toplantıları (Dailyler) her gün düzenli olarak yapılıyor mu?
**a)** Evet
**b)** Hayır
**c)** Bazen yapılır bazen yapılmaz
**d)** Bilmiyorum, fikrim yok
**e)** ……

15. Günlük Scrum toplantılarında (daily) genel olarak ne kadar sürmektedir?
**a)** 15 dakika
**b)** 15-30 dakika
**c)** 30-60 dakika
**d)** 60 dakika ve üzeri
**e)** Bilmiyorum, fikrim yok
**f)** ……

16.Günlük Scrum toplantılarında (daily) genel olarak aşağıdaki sorulardan hangilerini cevaplıyorsunuz?
**a)** Dün ne yaptım bugün ne yapacağım önümde engel var mı?
**b)** "Son toplantıdan sonra ne yaptım?"
**c)** "Bir sonraki toplantıya kadar ne yapmayı planlıyorum?"
**d)** Konular genelde dağılıyor
**e)** …….

17.Sprint sonlarında sprint review toplantısı yapıyor musunuz?
**a)** Evet
**b)** Hayır
**c)** Bazen yapılıyor bazen yapılmıyor
**d)** Bilmiyorum, fikrim yok
**e)** ……

18. Sprint review toplantısı genel olarak ne kadar sürüyor?
**a)** 15 dakika
**b)** 15-30 dakika
**c)** 30-60 dakika
**d)** 60 dakika ve üzeri
**e)** Bilmiyorum, fikrim yok
**f)** ……..

19. Sprint review toplantısında ürünün ilgili sprint içerisinde tamamlanan geliştirmeleri ile ilgili bir demo yapıyor musunuz?
**a)** Evet

**b)** Hayır
**c)** Bazen yapılıyor bazen yapılmıyor
**d)** Bilmiyorum, fikrim yok
**e)** …….

20. Sprint review toplantılarında ürün demosunu hangi takım üyesi yapmaktadır?
**a)** Proje yöneticisi
**b)** Analist/Çözümleyici
**c)** Mimar- teknik lider
**d)** Yazılım geliştirici
**e)** Test/Kalite Kontrol
**f)** Product Owner
**g)** Scrum Master
**h)** Hepsi
**i)** Bilmiyorum, fikrim yok
**j)** …….

21. Sprint sonlarında sprint retro toplantısı yapıyor musunuz?
**a)** Evet
**b)** Hayır
**c)** Bazen yapılıyor bazen yapılmıyor
**d)** Bilmiyorum, fikrim yok
**e)** …….

22. Sprint retro toplantısı genel olarak ne kadar sürüyor?
**a)** 15 dakika
**b)** 15-30 dakika
**c)** 30-60 dakika
**d)** 60 dakika ve üzeri
**e)** Bilmiyorum, fikrim yok
**f)** …….

23. Sprint Retro toplantısını kim yönetiyor?
**a)** Proje yöneticisi
**b)** Analist/Çözümleyici
**c)** Mimar- teknik lider
**d)** Yazılım geliştirici
**e)** Test/Kalite Kontrol
**f)** Product Owner
**g)** Scrum Master
**h)** Hepsi
**i)** Bilmiyorum, fikrim yok
**j)** …….

**KANBAN METODOLOJİSİ ÜZERİNE DETAYLI SORULAR**

1. İş takibini yapabilmek için Kanban panosu kullanıyor musunuz?

……

2. Kanban panosu için hangi toolu kullanıyorsunuz? (Jira, miro ya da şirketin geliştirdiği şirket özelinde kullanılan proje yönetim modeli gibi)

……

3. Ekibinizde Kanban Master var mı?

……

4. Sizce ekibinizde Kanban Master olmalı mıdır?

……

5. Ekibinizde Kanban Master görevini hangi rol üstleniyor?
**a)** Analist
**b)** Tester

**c)** Developer
**d)** Product Owner
**e)** ……..

6. Kanban metodolojisi ile ilerlemekten memnun musunuz? Olumlu olumsuz yanları nelerdir?
……

7. Kanban dışında hangi metodoloji ile çalışmayı tercih ederdiniz?
……

***Projelerin Fonksiyonel Test Süreçleri ve İşleyişinin analizi**
1.Projelerinizde manuel olarak fonksiyonel testler yapılıyor mu?
……..

2.Fonksiyonel yazılım testleri için destek aldığınız firmanızdan bağımsız bir test ekibiniz var mı?
……..

**Test ekibi var ise 14.sorudan devam ediniz. Test ekibi yok ise 3.sorudan devam edebilirsiniz.**

3.Test ekibiniz hiç yoksa yazılımın test edilmesi gereken noktalarda nasıl ilerliyorsunuz?
……..

4. Projenizde analist var mı? Kaç analist var?
……..

5. Analist yazmış olduğu analizin geliştirmesi tamamlanınca o işi kendisi test ediyor mu? Yoksa test için testerlardan mı destek alıyor?
……..

6. Sizce analistler yazmış oldukları analizin geliştirmesi tamamlanınca testleri kendileri mi yapmalı yoksa test ekibinden testler için destek mi almalılar?
……..

7. Developer kendi yazdığı kodun testini yapıyor mu? Biraz detaylarından bahsedebilir misiniz?
……..

8.Projenizdeki test süreçleri için dokümantasyon yapılıyor mu? Bu dokümantasyon için hangi araçlar kullanılmaktadır?
……..

9.Projenizde test senaryoları yazılıyor mu? Test senaryolarını yazmak için bir tool kullanıyor musunuz? (Jira, HP ALM, Excel)
……..

10.Yazılan test senaryoları review ediliyor mu? Review yapılıyorsa hangi rol tarafından yapılıyor?
……..

11.Test yapan kişiler testler esnasında karşılaştığı hata kayıtlarını tutabilmek için bir tool kullanıyorlar mı? (Jira, HP ALM, Excel, Mail, Şirket içi İletişim aracı)
……..

12.Developer aktif olarak bu hata toolu kullanabiliyor mu? Nasıl kullanıyor? (Hata detaylarına erişmek, hatayı çözdükten sonra hatanın tool üzerindeki statüsünü ilerletmek)
……..

13. Sizce ekibinizde bir tester olsa bu testerın görev tanımı ne olmalıydı?
……..

**Eğer test ekibi varsa aşağıdaki sorular sorulacak. Test ekibiniz yoksa bu bölümü atlayabilirsiniz.**

14.Test ekibinin test yöneticisi var mı? Eğer yoksa test yönetimini kim ya da hangi rol yapmaktadır?
……

15. Test ekibiniz kaç kişiden oluşuyor?
……..

16. Test ekibindeki testerlar domain bazında ayrılıyor mı her domaine bir tester gibi yoksa tester iş havuzundan gelen testin durumuna göre test kaynağı olarak gelen o işe atanıyor mu? Biraz test ekibinin çalışma modelinden kısaca bahsedebilir misiniz?
……..

17. Test için ayrılmış sadece 1 kaynağınız varsa test işlerini nasıl ilerletiyorsunuz? Bu kaynak bütün domainlerde test yapıyor mu yoksa onun kendi domaini mi oluyor?
……..

18.Projenizin genelinde dokümantasyon yapılıyor mu? Bu dokümantasyon için hangi araçlar kullanılmaktadır?
……..

19.Projenizde kullanılan belirli bir doküman formatınız var mı? Şirket içinde kullanılan standart doküman formatları varsa bunlar hangileridir?

- **a)** Teknik Analiz doküman formatı
- **b)** Gereksinim Analizi doküman formatı
- **c)** Functional Analiz dokümanı
- **d)** Test senaryo şablonu
- **e)** ……..

20. Şirket içinde kullanılan standart doküman formatları varsa bunlar hangileridir?

a) Fonksiyonel Analiz Dokümanı
b) Gereksinim Analiz Dokümanı
c) Teknik Analiz Dokümanı
d) Bilmiyorum, fikrim yok.
e) Diğer

21.Test ekibi işin hangi aşamasında işe dahil oluyor?

- **a)** Yazılım başlarken
- **b)** Talep esnasında
- **c)** Analiz edilirken
- **d)** Analiz sonrası
- **e)** ……

22.Eğer Scrum çalışıyorsanız test ekibi planlama toplantılarına dahil ediliyor mu?
……..

23.Projenizdeki test süreçleri için dokümantasyon yapılıyor mu? Bu dokümantasyon için hangi araçlar kullanılmaktadır?
……..

24.Projenizde test senaryoları yazılıyor mu? Test senaryolarını yazmak için bir tool kullanıyor musunuz? (Jira, HP ALM, Excel)
……..

25.Yazılan test senaryoları review ediliyor mu? Review yapılıyorsa hangi rol tarafından yapılıyor?
……..

26.Şirket içi iletişim aracı kullanılıyor mu? Hangi iletişim aracı kullanılmakta ve sıklıkla ne için kullanılmaktadır?

- **a)** Cisco Jabber
- **b)** Microsoft Teams

**c)** Skype
**d)** Slack
**e)** ……

27.Test ekibinin testler esnasında karşılaştığı hata kayıtlarını tutabilmek için bir tool kullanıyorlar mı? (Jira, HP ALM, Excel, Mail, Şirket içi İletişim aracı)
……..

28.Developer aktif olarak bu hata toolu kullanabiliyor mu? Nasıl kullanıyor? (Hata detaylarına erişmek, hatayı çözdükten sonra hatanın tool üzerindeki statüsünü ilerletmek)
……..

29.Test ekibinin performansında test metrikleri kullanılıyor mu? Projenizde bulunan test özelinde çalışan ekibin ya da kişinin performansı nasıl ölçülüyor? Testerların yazdığın case sayısı, testerların açtığı bug sayısı, productiondan dönen hata sayıları)
……..

30.Test ekibinin performans değerlendirmelerini kim yapıyor? (Test yöneticisi mi?)
……..

31.Projede bulunan testerların, test dışında aldığı farklı sorumluluklar var mı? (Scrum Master ya da daha farklı görevler)
……..

32.Sizce bir testerın görev tanımı nasıl olmalıdır?
……..

**GENEL**
1.Projenizde smoke testler yapılıyor mu? Smoke testlerini kim yapıyor?
**a)** Developerlar
**b)** Analistler
**c)** Son kullanıcılar
**d)** Yazılım geliştiren firmadan bağımsız bir test ekibi
**e)** Firma içinde kurulmuş ayrı test ekibi
**f)** Bilmiyorum fikrim yok
**g)** …….
2.Projenizde kullanıcı kabul testleri yapılıyor mu? Kullanıcı kabul testlerini kim yapıyor?
**a)** İş Birimleri
**b)** Developerlar
**c)** Analistler
**d)** Son kullanıcılar
**e)** Yazılım geliştiren firmadan bağımsız bir test ekibi
**f)** Firma içinde kurulmuş ayrı test ekibi
**g)** Bilmiyorum fikrim yok
**h)** ……
**i)**
3.Projenizde kullanıcı kabul testleri nasıl yapılmaktadır? İş birimi tek başına mı yapıyor, analist mi destek oluyor, analist önceden testleri yapıp daha sonra iş birimine sunum mu yapıyor, ekran görüntüleri ile mi yapılıyor?)

***Projelerin Otomasyon Test süreçleri ve İşleyişinin analizi**
1.Projelerinizde otomasyon testleri yapılıyor mu?
……..

**Cevap Hayır ise aşağıdaki sorular sorulmayacak. Görüşme sonlanacak.**
**Cevap Evet ise aşağıdaki sorulardan devam edilecek.**
2.Otomasyon testleri için destek aldığınız firmanızdan bağımsız bir test ekibiniz var mı?
……..

**Otomasyon test ekibi yoksa aşağıdaki sorudan devam edilecek.**

3.Otomasyon test ekibiniz yoksa otomasyon testlerini kim ya da hangi rol yapıyor?
……..

4.Projenizdeki otomasyon test süreçlerindeki dokümantasyon nasıldır?

**a)** Biraz
**b)** Çok
**c)** Hiç
**d)** Bilmiyorum fikrim yok
**e)** ……..

5.Otomasyon testlerine başlamadan önce otomasyon için ayrı test senaryoları yazıyor musunuz? Yoksa fonksiyonel test ekibinin yazdığı senaryoları mı kullanıyorsunuz?

**a)** Biraz
**b)** Çok
**c)** Hiç
**d)** Bilmiyorum fikrim yok
**e)** ……

6.Projenizde otomasyonu data oluşturmak için kullanıyor musunuz? Biraz detaylandırabilir misiniz?
……..

7.Genellikle hangi test türleri için otomasyon testini kullanıyorsunuz? (Regresyon, Yük, Performans, Data oluşturma)
……..

8.Otomasyon süreçleriniz devops a entegre midir?
……..

9. Projenizde geliştirmeler ve testler için kaç ortam kullanılmaktadır? (Test, Stable,Preprod,Prod…)
……..

10.Otomasyon testlerini genellikle hangi ortamlar için çalıştırıyorsunuz?
……..

11.Otomasyon testleri için belirlenmiş olan regresyon senaryo setleriniz var mı? Bunları ne sıklıkla koşuyorsunuz?
……..

12.Test senaryolarını yazmak için bir tool kullanıyor musunuz? Hangi tool'u kullanıyorsunuz? (Jira, Excel, HP ALM …)

**a)** Evet
**b)** Hayır
**c)** Bazen kullanılıyor bazen kullanılmıyor
**d)** ……
**e)**

13.Otomasyon testlerini sıklıkla hangi kanallar için kullanıyorsunuz? (Web, Mobilweb, App..)
……..

14.Web otomasyon testleri için sıklıkla hangi aracı kullanıyorsunuz?

**a)** Selenium Framework
**b)** Appium Framework
**c)** Postman
**d)** Soap
**e)** Diğer
**f)** Bilmiyorum fikrim yok
**g)** ……

15.Application otomasyon testleri için sıklıkla hangi aracı kullanıyorsunuz?

**a)** Selenium Framework

**b)** Appium Framework
**c)** Postman
**d)** Soap
**e)** Diğer
**f)** Bilmiyorum fikrim yok
**g)** ……

16.Otomasyon testlerinde karşılaştığınız hataları raporlamak için bir araç kullanıyor musunuz? Hangi araçları kullanıyorsunuz?
……..

**Otomasyon test ekibi var ise aşağıdaki sorular cevaplanacak.**
1.Otomasyon test ekibinin test yöneticisi var mı? Eğer yoksa test yönetimini kim ya da hangi rol yapmaktadır?
……..

2.Otomasyon test ekibi kaç kişiden oluşuyor?
……..

3.Projenizdeki otomasyon test süreçlerindeki dokümantasyon nasıldır?
**f)** Biraz
**g)** Çok
**h)** Hiç
**i)** Bilmiyorum fikrim yok
**j)** ……..

4.Otomasyon testlerine başlamadan önce otomasyon için ayrı test senaryoları yazıyor musunuz? Yoksa fonksiyonel test ekibinin yazdığı senaryoları mı kullanıyorsunuz?
**f)** Biraz
**g)** Çok
**h)** Hiç
**i)** Bilmiyorum fikrim yok
**j)** ……

5.Projenizde otomasyonu data oluşturmak için kullanıyor musunuz? Biraz detaylandırabilir misiniz?
……..

6.Genellikle hangi test türleri için otomasyon testini kullanıyorsunuz? (Regresyon, Yük, Performans, Data oluşturma)
……..

7.Otomasyon süreçleriniz devops a entegre midir?
……..

8. Projenizde geliştirmeler ve testler için kaç ortam kullanılmaktadır? (Test, Stable,Preprod,Prod…)
……..

9.Otomasyon testlerini genellikle hangi ortamlar için çalıştırıyorsunuz?
……..

10.Otomasyon testleri için belirlenmiş olan regresyon senaryo setleriniz var mı? Bunları ne sıklıkla koşuyorsunuz?
……..

11.Test senaryolarını yazmak için bir tool kullanıyor musunuz? Hangi tool'u kullanıyorsunuz? (Jira, Excel, HP ALM …)
**f)** Evet
**g)** Hayır
**h)** Bazen kullanılıyor bazen kullanılmıyor

**i)** ……

12.Otomasyon testlerini sıklıkla hangi kanallar için kullanıyorsunuz? (Web, Mobilweb, App..)
……..

13.Web otomasyon testleri için sıklıkla hangi aracı kullanıyorsunuz?

**h)** Selenium Framework
**i)** Appium Framework
**j)** Postman
**k)** Soap
**l)** Diğer
**m)** Bilmiyorum fikrim yok
**n)** ……

14.Application otomasyon testleri için sıklıkla hangi aracı kullanıyorsunuz?

**h)** Selenium Framework
**i)** Appium Framework
**j)** Postman
**k)** Soap
**l)** Diğer
**m)** Bilmiyorum fikrim yok
**n)** ……

15.Otomasyon testlerinde karşılaştığınız hataları raporlamak için bir araç kullanıyor musunuz? Hangi araçları kullanıyorsunuz?
……..

## ÖZGEÇMİŞ

### KİŞİSEL BİLGİLER

Adı Soyadı: Hayriye KATRANCI

Yabancı Dili: İngilizce

### ÖĞRENİM DURUMU

| **Derece** | **Alan** | **Okul/Üniversite** | **Mezuniyet Yılı** |
|---|---|---|---|
| Y.Lisans | Elektrik-Elektronik ve Bilgisayar Mühendisliği | Düzce Üniversitesi | 2024 |
| Lisans | Bilgisayar ve Öğretim Teknolojileri Öğretmenliği | Kahramanmaraş Sütçü İmam Üniversitesi | 2012 |
| Lise | Bilgisayar | Mersin Atatürk Anadolu Teknik Lisesi | 2008 |