



**ÇEVİK YAZILIM GELİŞTİRMEDE İNSAN VE
SÜREÇ FAKTÖRLERİNİN PROJE BAŞARISINA
ETKİSİNDE ÇEVİK PRATİKLERİN
DÜZENLEYİCİ ROLÜNÜN ARAŞTIRILMASI**

Abdullah BALCI

**Doktora Tezi
Yönetim Bilişim Sistemleri Ana Bilim Dalı
Doç. Dr. Mustafa KESKİNKILIÇ
2024
Her Hakkı Saklıdır**

T.C.
ATATÜRK ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ
YÖNETİM BİLİŞİM SİSTEMLERİ ANA BİLİM DALI

Abdullah BALCI

ÇEVİK YAZILIM GELİŞTİRMEDE İNSAN VE SÜREÇ
FAKTÖRLERİNİN PROJE BAŞARISINA ETKİSİNDE ÇEVİK
PRATİKLERİN DÜZENLEYİCİ ROLÜNÜN ARAŞTIRILMASI

DOKTORA TEZİ

TEZ YÖNETİCİSİ
Doç. Dr. Mustafa KESKİNKILIÇ

ERZURUM- 2024



TEZ BEYAN FORMU

SOSYAL BİLİMLER ENSTİTÜSÜ MÜDÜRLÜĞÜNE

BİLDİRİM

Atatürk Üniversitesi Lisansüstü Eğitim ve Öğretim Uygulama Esaslarının ilgili maddelerine göre hazırlamış olduğum “Çevik Yazılım Geliştirmede İnsan ve Süreç Faktörlerinin Proje Başarısına Etkisinde Çevik Pratiklerin Düzenleyici Rolünün Araştırılması” adlı tezin tamamen kendi çalışmam olduğunu ve her alıntıya kaynak gösterdiğimi taahhüt eder, tezimin kâğıt ve elektronik kopyalarının aşağıda belirttiğim koşullarda saklanmasına izin verdiğimi onaylarım.

Gereğini bilgilerinize arz ederim *.

☐ Tezimin/Raporumun tamamı her yerden erişime açılabilir.

☒ Tezimin/Raporumun makale için altı ay, patent için iki yıl süreyle erişiminin ertelenmesini istiyorum.

20.12.2024

Ash ıslak imzalıdır

Abdullah BALCI

* LİSANSÜSTÜ TEZLERİN ELEKTRONİK ORTAMDA TOPLANMASI, DÜZENLENMESİ VE ERİŞİME AÇILMASINA İLİŞKİN YÖNERGE

ÜÇÜNCÜ BÖLÜM

Çeşitli ve Son Hükümler

Lisansüstü tezlerin erişime açılmasının ertelenmesi MADDE 6- (1) Lisansüstü teze ilgili patent başvurusu yapılması veya patent alma sürecinin devam etmesi durumunda, tez danışmanının önerisi ve enstitü anabilim dalının uygun görüşü üzerine enstitü veya fakülte yönetim kurulu iki yıl süre ile tezin erişime açılmasının ertelenmesine karar verebilir.

(2) Yeni teknik, materyal ve metodların kullanıldığı, henüz makaleye dönüşmemiş veya patent gibi yöntemlerle korunmamış ve internetten paylaşılması durumunda 3. şahıslara veya kurumlara haksız kazanç imkanı oluşturabilecek bilgi ve bulguları içeren tezler hakkında tez danışmanının önerisi ve enstitü anabilim dalının uygun görüşü üzerine enstitü veya fakülte yönetim kurulunun gerekçeli kararı ile altı ayı aşmamak üzere tezin erişime açılması engellenebilir.

Gizlilik dereceli tezler MADDE 7- (1) Ulusal çıkarları veya güvenliği ilgilendiren, emniyet, istihbarat, savunma ve güvenlik, sağlık vb. konulara ilişkin lisansüstü tezlerle ilgili gizlilik kararı, tezin yapıldığı kurum tarafından verilir. Kurum ve kuruluşlarla yapılan işbirliği protokolü çerçevesinde hazırlanan lisansüstü tezlere ilişkin gizlilik kararı ise, ilgili kurum ve kuruluşun önerisi ile enstitü veya fakültenin uygun görüşü üzerine üniversite yönetim kurulu tarafından verilir. Gizlilik kararı verilen tezler Yükseköğretim Kuruluna bildirilir.

(2) Gizlilik kararı verilen tezler gizlilik süresince enstitü veya fakülte tarafından gizlilik kuralları çerçevesinde muhafaza edilir, gizlilik kararının kaldırılması halinde Tez Otomasyon Sistemine yüklenir.



SOSYAL BİLİMLER ENSTİTÜSÜ

Graduate School of Social Sciences

TEZ KABUL TUTANAĞI

SOSYAL BİLİMLER ENSTİTÜSÜ MÜDÜRLÜĞÜNE

Doç. Dr. Mustafa KESKİNKILIÇ danışmanlığında, Abdullah BALCI tarafından hazırlanan bu çalışma 20/12/2024 tarihinde aşağıda isimleri yazılı jüri tarafından Yönetim Bilişim Sistemleri Anabilim Dalı'nda Doktora Tezi olarak kabul edilmiştir.

Başkan : Prof. Dr. Ömer ALKAN

İmza: Aslı Islak İmzalıdır

Jüri Üyesi : Doç. Dr. Mustafa KESKİNKILIÇ

İmza: Aslı Islak İmzalıdır

Jüri Üyesi : Doç. Dr. Ahmet Kamil KABAKUŞ

İmza: Aslı Islak İmzalıdır

Jüri Üyesi : Doç. Dr. Handan ÇAM

İmza: Aslı Islak İmzalıdır

Jüri Üyesi : Doç. Dr. Emine Demet EKİNCİ HAMAMCI

İmza: Aslı Islak İmzalıdır

Aslı Islak İmzalıdır

Prof. Dr. Hanifi ŞAHİN

Enstitü Müdürü

İÇİNDEKİLER

ÖZET.....	VII
ABSTRACT	VIII
KISALTMALAR DİZİNİ	IX
TABLolar DİZİNİ	XI
ŞEKİLLER DİZİNİ	XIV
ÖNSÖZ.....	XVII
GİRİŞ	1

BİRİNCİ BÖLÜM

YAZILIM ve YAZILIM MÜHENDİSLİĞİ

1.1. YAZILIM.....	3
1.1.1. Yazılım Özellikleri	4
1.1.2. Yazılım Türleri	7
1.1.3. Yazılım Sınıfları	8
1.1.4. Yazılım Uygulamaları	8
1.2. YAZILIM MÜHENDİSLİĞİ.....	10
1.2.1. Yazılım Mühendisliği Katmanları	12
1.2.2. Yazılım Mühendisliği Bilgi Alanları.....	14
1.2.3. Yazılım Mühendisliği ile İlişkili Disiplinler	15

İKİNCİ BÖLÜM

YAZILIM GELİŞTİRME METODOLOJİLERİ

2.1. YAZILIM GELİŞTİRME YAŞAM DÖNGÜSÜ	16
2.2. GELENEKSEL YAZILIM GELİŞTİRME YÖNTEMLERİ.....	21
2.2.1. Şelale Modeli (Waterfall Model)	22
2.2.2. Artımlı Geliştirme Modeli (Incremental Model)	24
2.2.3. Prototip Model (Prototyping Model).....	25
2.2.4. V Modeli (V Model)	27
2.2.5. Spiral (Sarmal) Model (Spiral Model)	28
2.3. ÇEVİK YAZILIM GELİŞTİRME YÖNTEMLERİ.....	30
2.3.1. Uç Programlama (Extreme Programming- XP)	35

2.3.2. Scrum	40
2.3.3. Kanban	46
2.3.4. Özellik Odaklı Geliştirme (Feature Driven Development)	51
2.3.5. Test Odaklı Geliştirme (Test Driven Development)	52
2.3.6. Yalın Yazılım Geliştirme (Lean Software Development).....	53
2.3.7. Hızlı Uygulama Geliştirme (Rapid Application Development).....	55
2.3.8. Uyarlanabilir Yazılım Geliştirme (Adaptive Software Development).....	55
2.3.9. Dinamik Sistem Geliştirme (Dynamic Systems Development).....	56
2.3.10. Kristal Aile Metodolojisi (Crystal Family Methodology).....	58
2.3.11. Scrumban.....	60
2.3.12. Scrum/XP Hibrit (Scrum/XP Hybrid)	61
2.4. ÇEVİK PRATİKLER.....	63
2.4.1. Kullanıcı Hikayesi (User Story).....	64
2.4.2. Kısa Yinelemeler (Short Iterations)	65
2.4.3. Kısa Sürümler (Short Releases) - Küçük Sürümler (Small Releases).....	65
2.3.4. Sprint	66
2.4.5. Artımlı Tasarım (Incremental Design).....	66
2.4.6. Basit Tasarım (Simple Design)	67
2.4.7. Metafor (Metaphor).....	68
2.4.8. Planlama Oyunu (Planning Game).....	68
2.4.9. Çapraz Fonksiyonel Ekipler (Cross-Functional Teams)	69
2.4.10. Ürün İş Listesi (Product Backlog).....	69
2.4.11. Sürüm ve Yineleme İş Listesi (Release and Iteration Backlog).....	70
2.4.12. Sprint İş Listesi (Sprint Backlog).....	70
2.4.13. Yineleme Planlama (Iteration Planning)	71
2.4.14. Sprint Planlama (Sprint Planning).....	71
2.4.15. Sürüm Planlama (Release Planning)	72
2.4.16. Ayaküstü Toplantı-Günlük Scrum (Stand-Up Meeting-Daily Scrum)	73
2.4.17. Yineleme Gösterimi (Iteration Demonstration)	74
2.4.18. Sprint Değerlendirme (Sprint Review)	74
2.4.19. Retrospektif (Retrospective)	74
2.4.20. Birlikte Oturma (Sit Together).....	75

2.4.21. Açık Çalışma Alanı (Open Workspace).....	76
2.4.22. Sürekli entegrasyon (Continuous Integration)	76
2.4.23. Kod Sahipliği (Code Ownership).....	77
2.4.24. Ortak (Kollektif) Kod Sahipliği (Collective Ownership).....	77
2.4.25. Kodlama Standartları (Coding Standards)	78
2.4.26. Yeniden Kod Düzenleme (Refactoring).....	79
2.4.27. Test Etme (Testing)	80
2.4.28. Önce Test Programlama (Test-First Programming).....	80
2.4.29. Kabul Testi Odaklı Geliştirme (Acceptance Test-Driven Development)	81
2.4.30. Birim Testi (Unit Test).....	82
2.4.31. Yerinde Müşteri (On-site Customer).....	82
2.4.32. Müşteri Katılımı (Customer Involvement).....	83
2.4.33. Eşli Programlama (Pair Programming)	83
2.4.34. Enerjik Çalışma (Energized Work)	84
2.4.35. Sürdürülebilir Tempo (Sustainable Pace)	85
2.4.36. Haftalık 40 Saat (40-Hour Weeks).....	85
2.4.37. On Dakikalık Derleme (Ten-Minute Build).....	85
2.4.38. Tüm Takım (Whole Team)	86
2.4.39. Bilgilendirici Çalışma Alanı (Informative Workspace)	86
2.4.40. İş İlerleme Grafiği- Yanma (Tüketim) Grafiği (Burn Chart).....	86
2.4.41. Kanban Panosu ve Scrum Panosu (Kanban Board and Scrum Board)	88

ÜÇÜNCÜ BÖLÜM

ÇEVİK YAZILIM GELİŞTİRME PROJE BAŞARISI ve KRİTİK BAŞARI FAKTÖRLERİ

3.1. YAZILIM PROJESİ.....	91
3.2. YAZILIM PROJE BAŞARISI.....	92
3.2.1. Kalite	95
3.2.2. Kapsam.....	96
3.2.3. Zaman.....	97
3.2.4. Maliyet	98
3.3. KRİTİK BAŞARI FAKTÖRLERİ.....	98
3.3.1. İnsan Faktörleri	101

3.3.1.1. Müşteri Katılımı	101
3.3.1.2. Takım Yeteneği	102
3.3.2. Süreç Faktörleri	103
3.3.2.1. Proje Tanımlama Süreci	104
3.3.2.2. Proje Yönetim Süreci	105
3.3.3. Organizasyon Faktörleri	106
3.3.4. Teknik Faktörler	107
3.3.5. Proje Faktörleri.....	107

DÖRDÜNCÜ BÖLÜM

LİTERATÜR ÖZETİ

4.1. ÇEVİK YAZILIM GELİŞTİRME PROJELERİNDE KRİTİK BAŞARI FAKTÖRLERİYLE İLGİLİ ÇALIŞMALAR	109
4.2. ÇEVİK PRATİKLERLE İLGİLİ ÇALIŞMALAR	128
4.3. ÇEVİK YAZILIM GELİŞTİRME PROJE BAŞARISIYLA İLGİLİ DİĞER ÇALIŞMALAR	143

BEŞİNCİ BÖLÜM

METODOLOJİ

5.1. ARAŞTIRMANIN AMACI VE ÖNEMİ	148
5.2. ARAŞTIRMANIN KISITLARI.....	151
5.3. ARAŞTIRMA YÖNTEMİ	152
5.4. ARAŞTIRMA MODELİ VE HİPOTEZLER	153
5.4.1. Müşteri Katılımına İlişkin Hipotezler	154
5.4.2 Takım Yeteneğine İlişkin Hipotezler	155
5.4.3 Proje Tanımlama Sürecine İlişkin Hipotezler	156
5.4.4. Proje Yönetim Sürecine İlişkin Hipotezler	157
5.4.5. Çevik Pratiklerin Düzenleyici Rolüne İlişkin Hipotezler	157
5.5. ARAŞTIRMANIN EVRENİ VE ÖRNEKLEMİ	159
5.6. VERİ TOPLAMA ARAÇLARI.....	161
5.6.1 Demografik Bilgiler	161
5.6.2 Çevik Başarı Faktörleri Ölçeği.....	161
5.6.3. Çevik Yazılım Geliştirme Projesinde Algılanan Başarı Ölçeği	162

5.6.4 Çevik Pratiklerin Kullanımı Ölçeği.....	163
5.6.5. Ölçme Araçlarının Türkçe 'ye Uyarlanması ve Pilot Uygulama	164
5.6.5.1. Ölçme Araçlarının Türkçe 'ye Uyarlanma Süreci	164
5.6.5.2. Pilot Uygulama	164
5.7. VERİ TOPLAMA SÜRECİ	168
5.8. VERİLERİN ANALİZİ	168
5.8.1. Veri Setinin Kontrolü ve Analize Hazırlanması.....	168
5.8.2.1. Veri Setinin Doğruluğu	168
5.8.2.2. Eksik Veri Analizi	169
5.8.2.3. Uç Değer.....	170
5.8.2.4. Normallik Testi.....	171
5.8.2.5. Çoklu Doğrusallık ve Tekillik	173
5.8.2. Güvenilirlik Analizi.....	173
5.8.3. Geçerlilik Analizi	175
5.8.4. Betimleyici Analizler	179
5.8.5. Korelasyon Analizi.....	179
5.8.6. Hipotez Testleri	180
5.8.6.1. Yapısal Regresyon Modeli Analizi.....	180
5.8.6.2. Düzenleyici Değişken Etkisinin Analizi	180

ALTINCI BÖLÜM

BULGULAR

6.1. VERİ SETİNİN KONTROLÜ ve ANALİZE HAZIRLANMASINA İLİŞKİN BULGULAR	184
6.1.1 Veri Setinin Doğruluğu ve Eksik Veri Analizi	184
6.1.2. Uç Değer Analizi.....	184
6.1.3. Normallik Testi	185
6.1.4. Çoklu Doğrusallık ve Tekillik.....	187
6.2 GÜVENİLİRLİK ANALİZİNE İLİŞKİN BULGULAR	188
6.2.1. Çevik Başarı Faktörleri Ölçeği Güvenilirlik Analizine İlişkin Bulgular	188
6.2.2. Çevik Pratiklerin Kullanımı Ölçeği Güvenilirlik Analizine İlişkin Bulgular	190
6.2.3. Çevik Yazılım Geliştirme Projesinde Algılanan Başarı Ölçeği Güvenilirlik Analizine İlişkin Bulgular	191

6.3. GEÇERLİLİK ANALİZİNE İLİŞKİN BULGULAR	192
6.3.1. Çevik Başarı Faktörleri Ölçeği Geçerlilik Analizine İlişkin Bulgular	192
6.3.2. Çevik Pratiklerin Kullanımı Ölçeği Geçerlilik Analizine İlişkin Bulgular	195
6.3.3. Çevik Yazılım Geliştirme Projesinde Algılanan Başarı Ölçeği Geçerlik Analizine İlişkin Bulgular	200
6.4. DEMOGRAFİK ÖZELLİKLERE İLİŞKİN BULGULAR.....	202
6.5. ÖLÇEK İFADELERİNE İLİŞKİN TANIMLAYICI BULGULAR	204
6.5.1. Çevik Başarı Faktörleri Ölçeğine İlişkin Tanımlayıcı Bulgular	204
6.5.3. Çevik Pratiklerin Kullanımı Ölçeğine İlişkin Tanımlayıcı Bulgular	205
6.5.3. Çevik Yazılım Geliştirme Projesinde Algılanan Başarı Ölçeğine İlişkin Tanımlayıcı Bulgular.....	207
6.6. KORELASYON ANALİZİNE İLİŞKİN BULGULAR	208
6.7. HİPOTEZ TESTLERİNE İLİŞKİN BULGULAR.....	210
6.7.1 Yapısal Regresyon Modeli Analizi	210
6.7.1.1. Ölçüm Modeli: Geçerlilik ve Güvenilirliğe İlişkin Bulgular	210
6.7.1.2. Yapısal Model: Hipotezlere İlişkin Bulgular.....	213
6.7.2. Düzenleyici Değişken Etkisinin Analizi ve Bulguları	216
6.7.2.1. Müşteri Katılımının Çevik Yazılım Geliştirme Proje Başarısı Üzerindeki Etkisinde Çevik Pratiklerin Düzenleyici Rolü.....	216
6.7.2.2. Takım Yeteneğinin Çevik Yazılım Geliştirme Proje Başarısı Üzerindeki Etkisinde Çevik Pratiklerin Düzenleyici Rolü.....	221
6.7.2.3. Proje Tanımlama Sürecinin Çevik Yazılım Geliştirme Proje Başarısı Üzerindeki Etkisinde Çevik Pratiklerin Düzenleyici Rolü.....	226
6.7.2.4. Proje Yönetim Sürecinin Çevik Yazılım Geliştirme Proje Başarısı Üzerindeki Etkisinde Çevik Pratiklerin Düzenleyici Rolü.....	231
6.7.3. Hipotez Test Sonuçlarının Özeti	237
SONUÇ VE TARTIŞMA.....	239
ÖNERİLER	252
KAYNAKÇA	255
EKLER.....	281
ÖZ GEÇMİŞ.....	288

ÖZET**DOKTORA TEZİ****ÇEVİK YAZILIM GELİŞTİRMEDE İNSAN VE SÜREÇ FAKTÖRLERİNİN PROJE BAŞARISINA ETKİSİNDE ÇEVİK PRATİKLERİN DÜZENLEYİCİ ROLÜ****Abdullah BALCI****Tez Danışmanı: Doç. Dr. Mustafa KESKİNKILIÇ****2024, 288 Sayfa****Jüri: Doç. Dr. Mustafa KESKİNKILIÇ (Danışman)****Prof. Dr. Ömer ALKAN****Doç. Dr. Ahmet Kamil KABAKUŞ****Doç. Dr. Handan ÇAM****Doç. Dr. Emine Demet EKİNCİ HAMAMCI**

Yazılım modern dünyanın tüm yönleri için önemli bir rol oynarken, yazılım geliştirme süreci kusursuz bir yapıya sahip değildir. Geleneksel yöntemlere kıyasla çevik yazılım geliştirme yöntemleri, yazılım geliştirme süreçlerinde daha iyi ve yenilikçi bir yol olarak ortaya çıkmıştır. Çevik yöntemler, yazılım projelerinde geleneksel yöntemlere göre daha yüksek başarı oranları sunmasına rağmen, bu projelerin önemli bir kısmı hala çeşitli sorunlarla karşılaşmakta ya da başarısızlıkla sonuçlanmaktadır. Bu durum, çevik yazılım geliştirme projelerinin başarısını artıracak kritik başarı faktörlerinin belirlenmesi ve incelenmesinin önemini vurgulamaktadır.

Bu çalışma, çevik yazılım geliştirme geliştirmede insan ve süreç faktörlerinin proje başarısına etkisinde çevik pratiklerin düzenleyici rolünü araştırmak amacıyla gerçekleştirilmiştir. Araştırma, nicel yöntemler kapsamında ilişkisel tarama deseni kullanılarak gerçekleştirilmiştir. Veriler, geçerliliği ve güvenilirliği literatürde kanıtlanmış olan “Çevik Başarı Faktörleri Ölçeği”, “Çevik Yazılım Geliştirme Projesinde Algılanan Başarı Ölçeği” ve “Çevik Pratik Kullanımları Ölçeği”ni içeren bir anket formu aracılığıyla elde edilmiştir. Araştırmanın örneklemini, İstanbul’daki Teknoloji Geliştirme Bölgelerinde faaliyet gösteren yazılım geliştirme firmalarında çalışan 411 katılımcı oluşturmaktadır. Veriler, kolayda örnekleme yöntemiyle toplanmış ve analizler SPSS 22, AMOS 26 ile Hayes Process Macro (v4.2) yazılımları kullanılarak gerçekleştirilmiştir.

Araştırma bulguları, müşteri katılımı ve takım yeteneğinin proje tanımlama süreci, proje yönetim süreçleri ve proje başarısı üzerinde anlamlı etkiler oluşturduğunu ortaya koymuştur. Bununla birlikte, proje tanımlama sürecinin proje yönetim sürecini olumlu yönde etkilediği ve proje yönetim sürecinin proje başarısını doğrudan belirlediği tespit edilmiştir. Ancak, proje tanımlama sürecinin proje başarısına doğrudan etkisi desteklenmemiştir.

Çevik pratiklerin düzenleyici etkileri incelendiğinde, müşteri katılımının proje başarısına etkisinde sosyal pratiklerin anlamlı bir düzenleyici rol oynadığı, ancak teknik pratiklerin etkisinin desteklenmediği görülmüştür. Bununla birlikte, sosyal ve teknik pratiklerin birlikte kullanımı, müşteri katılımının proje başarısına etkisinde güçlü bir düzenleyici etki oluşturmıştır. Takım yeteneğinin proje başarısına etkisinde hem sosyal hem de teknik pratiklerin ayrı ayrı anlamlı düzenleyici etkiler oluşturduğu, birlikte kullanıldığında ise daha güçlü bir düzenleyici etki sağladığı belirlenmiştir. Proje tanımlama sürecinin proje başarısına etkisinde sosyal pratiklerin anlamlı bir düzenleyici etkisi olduğu, ancak teknik pratiklerin etkisinin desteklenmediği saptanmıştır. Bununla birlikte, sosyal ve teknik pratiklerin birlikte kullanımı, bu ilişkide önemli bir düzenleyici etki oluşturmıştır. Son olarak, proje yönetim sürecinin proje başarısına etkisinde hem sosyal hem teknik pratiklerin anlamlı düzenleyici etkileri olduğu ve bu pratiklerin birlikte kullanıldığında daha güçlü bir düzenleyici etki oluşturduğu tespit edilmiştir.

Sonuçlar, çevik yazılım geliştirme projelerinde müşteri katılımı, takım yeteneği, proje tanımlama ve proje yönetim süreçlerinin birbirleriyle ilişkili olduğunu ve çevik pratiklerle bütünleştirildiğinde proje başarısının artırılmasında kritik bir rol oynadığını ortaya koymaktadır. Bu bulgular, çevik yazılım geliştirme literatürüne teorik katkı sunmanın yanı sıra, uygulamada başarıyı artırmaya yönelik pratik öneriler geliştirilmesi için de önemli bir temel sağlamaktadır.

Anahtar Kelimeler: Çevik Yöntemler, Çevik Proje Başarısı, Çevik Pratikler, Düzenleyici Rol, Yazılım Mühendisliği, Yönetim Bilişim Sistemleri

ABSTRACT**Ph.D DISSERTATION****INVESTIGATION OF THE MODERATING ROLE OF AGILE PRACTICES IN THE IMPACT OF HUMAN AND PROCESS FACTORS ON PROJECT SUCCESS IN AGILE SOFTWARE DEVELOPMENT****Abdullah BALCI****Advisor: Assoc. Prof. Dr. Mustafa KESKİNKILIÇ****2024, 288 Pages****Jury: Assoc. Prof. Dr. Mustafa KESKİNKILIÇ (Advisor)****Prof. Dr. Ömer ALKAN****Assoc. Prof. Dr. Ahmet Kamil KABAKUŞ****Assoc. Prof. Dr. Handan ÇAM****Assoc. Prof. Dr. Emine Demet EKİNCİ HAMAMCI**

While software plays a critical role in all aspects of the modern world, the software development process is not flawless. Compared to traditional methods, agile software development methods have emerged as a better and more innovative approach to software development processes. Although agile methods offer higher success rates in software projects than traditional methods, many of these projects still encounter various issues or result in failure. This highlights the importance of identifying and examining critical success factors that can enhance the success of agile software development projects.

This study aims to investigate the moderating role of agile practices in the impact of human and process factors on project success in agile software development. The research was conducted using a relational survey design within the scope of quantitative methods. Data were collected through a survey form comprising the “Agile Success Factors Scale”, the “Perceived Success in Agile Software Development Projects Scale” and the “Agile Practices Usage Scale” whose validity and reliability have been confirmed in the literature. The sample of the study consists of 411 participants working in software development firms operating in Technology Development Zones in Istanbul. Data were collected using a convenience sampling method and analyzed using SPSS 22, AMOS 26, and Hayes Process Macro (v4.2).

The findings of the study revealed that customer involvement and team capability have significant impacts on the project definition process, project management processes, and project success. Furthermore, the project definition process positively influences the project management process, which directly determines project success. However, the direct effect of the project definition process on project success was not supported.

Regarding the moderating effects of agile practices, social practices were found to have a significant moderating role in the impact of customer involvement on project success, while the effect of technical practices was not supported. However, the combined use of social and technical practices created a strong moderating effect on the impact of customer involvement on project success. In the relationship between team capability and project success, both social and technical practices independently had significant moderating effects, which became stronger when used together. Social practices had a significant moderating effect on the impact of the project definition process on project success, while the effect of technical practices was not supported. Nevertheless, the combined use of social and technical practices generated a significant moderating effect in this relationship. Finally, in the relationship between the project management process and project success, both social and technical practices had significant moderating effects, and their combined use created a stronger moderating effect.

The results demonstrate that customer involvement, team capability, project definition, and project management processes are interrelated in agile software development projects and play a critical role in enhancing project success when integrated with agile practices. These findings provide theoretical contributions to the agile software development literature and offer a solid foundation for developing practical recommendations to improve project success in practice.

Keywords: Agile Methods, Agile Project Success, Agile Practices, Moderating Role, Software Engineering, Management Information System

KISALTMALAR DİZİNİ

AGFI	: Adjustment Goodness Of Fit Index (Düzeltilmiş İyilik Uyum İndeksi)
AHP	: Analytical Hierarchy Process (Analitik Hiyerarşi Süreci)
AMOS	: Analysis of Moment Structures
AT	: Ayaküstü Toplantı
AVE	: Average Variance Extracted (Ortalama Açıklanan Varyans)
BT	: Bilgi Teknolojileri
BT	: Birim Test
C.R.	: Critical Ratio (Kritik Oran- t değeri)
CFI	: Comparative Fit Index (Karşılaştırmalı Uyum İndeksi)
CHAOS	: Comprehensive Human Appraisal for Originating Software
CR	: Composite Reliability (Bileşik Güvenilirlik)
ÇBF	: Çevik Başarı Faktörleri
ÇPK	: Çevik Pratiklerin Kullanımı
ÇYGPAB	: Çevik Yazılım Geliştirme Projesinde Algılanan Başarı
DFA	: Doğrulayıcı Faktör Analizi
DSGY	: Dinamik Sistem Geliştirme Yöntemi
GFI	: Goodness Of Fit Index (İyilik Uyum İndeksi)
HTMT	: Heterotrait-Monotrait
IEEE	: Institute of Electrical and Electronics Engineers
KBF	: Kritik Başarı Faktörleri
KFA	: Keşfedici (Açıklayıcı) Faktör Analizi
KS	: Kodlama Standardı
KYG	: Küresel Yazılım Geliştirme
MAR	: Missing at Random (Rastgele Kayıp)
MCAR	: Missing Completely at Random (Tamamen Rastgele Kayıp)
MK	: Müşteri Katılımı
ML	: Maximum Likelihood (Maksimum Olabilirlik)
MNAR	: Missing Not at Random (Rastgele Olmayan Kayıp)

NFI	: Normed Fit Index (Normlaştırılmış Uyum İndeksi)
NNFI	: Non-Normed Fit Index (Normlaştırılmamış Uyum İndeksi)
ÖOG	: Özellik Odaklı Geliştirme
PLS-SEM	: Partial Least Squares Structural Equation Modeling (Kısmi En Küçük Kareler Yapısal Eşitlik Modellemesi)
PMBOK	: Project Management Body of Knowledge
PMI	: Project Management Institute
PT	: Proje Tanımlama Süreci
PY	: Proje Yönetim Süreci
R	: Retrospektif
RMSEA	: Root Mean Square Error of Approximation (Yaklaşık Hataların Ortalama Karekökü)
SE	: Sürekli Entegrasyon
SH	: Standart Hata
SPSS	: Statistical Package for the Social Sciences
SRMR	: Standardized Root Mean Square Residual (Standardize Edilmiş Ortalama Hataların Karekökü)
SWEBOK	: Software Engineering Body of Knowledge
TDK	: Türk Dil Kurumu
TOG	: Test Odaklı Geliştirme
TY	: Takım Yeteneği
UP	: Uç Programlama
VIF	: Variance Inflation Factor
XP	: Extreme Programming
YEM	: Yapısal Eşitlik Modeli
YGYD	: Yazılım Geliştirme Yaşam Döngüsü
YKD	: Yeniden Kod Düzenleme
YT	: Yinelemeli Teslimat
YYG	: Yalın Yazılım Geliştirme
PB	: Proje Başarısı

TABLOLAR DİZİNİ

Tablo 1.1. Yazılım Mühendisliği Bilgi Alanları	14
Tablo 3.1. Çevik Kritik Başarı Faktörleri.....	100
Tablo 5.1. İstanbul’da Yer Alan Teknoloji Geliştirme Bölgelerinde Yazılım Geliştirme Faaliyetinde Bulunan Toplam Firma ve Çalışan Sayısı	160
Tablo 5.2. Araştırmada Kapsamında Kullanılacak İnsan ve Süreç Faktörlerine İlişkin Bilgiler	162
Tablo 5.3. ÇYGPAB Ölçeği ve Açıklamaları	162
Tablo 5.4. ÇPK Ölçeğinin Faktörleri ve Madde Sayısı.....	163
Tablo 5.5. Araştırmanın Pilot Çalışmasına Katılımcılara İlişkin Demografik Veriler	165
Tablo 5.6. ÇBF Ölçeği Madde Analizi.....	166
Tablo 5.7. ÇYGPAB Ölçeği Madde Analizi	167
Tablo 5.8. ÇPK Ölçeği Madde Analizi	167
Tablo 5.9. Güvenilirlik Katsayıları.....	174
Tablo 5.10. Uyum İyiliği İndeksleri ve Eşik Değerleri	178
Tablo 5.11. Korelasyon Katsayısının Değerlendirilmesi	180
Tablo 6.1. ÇBF Ölçeği Çarpıklık ve Basıklık Değerleri	185
Tablo 6.2. ÇPK Ölçeği Çarpıklık ve Basıklık Değerleri	186
Tablo 6.3. ÇYGPAB Ölçeği Çarpıklık ve Basıklık Değerleri	187
Tablo 6.4. Çoklu Doğrusallık ve Tekillik Test Sonuçları	188
Tablo 6.5. ÇBF Ölçeği Güvenilirlik Analizi Sonuçları	189
Tablo 6.6. ÇPK Ölçeği Güvenilirlik Analizi Sonuçları.....	190
Tablo 6.7. ÇYGPAB Ölçeği Güvenilirlik Analizi Sonuçları	191
Tablo 6.8. ÇBF Ölçeği DFA Uyum İyiliği Değerleri.....	192
Tablo 6.9. ÇBF Ölçeği DFA Regresyon Katsayıları	194
Tablo 6.10. ÇBF Ölçeği Güvenilirlik, Birleşim ve Ayrışım Geçerliliği	194
Tablo 6.11. ÇPK Ölçeği Birinci Düzey DFA Uyum İyiliği Değerleri.....	196
Tablo 6.12. ÇPK Ölçeği Birinci Düzey DFA Regresyon Katsayıları	196
Tablo 6.13. ÇPK Ölçeği Güvenilirlik, Birleşim ve Ayrışım Geçerliliği	197
Tablo 6.14. ÇPK Ölçeği İkinci Düzey DFA Uyum İyiliği Değerleri.....	197
Tablo 6.15. ÇPK Ölçeği İkinci Düzey DFA Regresyon Katsayıları	199

Tablo 6.16. ÇPK Ölçeği Güvenilirlik, Birleşim ve Ayrışım Geçerliliği (İkinci Düzey)	200
Tablo 6.17. ÇYGPAB Ölçeği DFA Uyum İyiiliği Değerleri	201
Tablo 6.18. ÇYGPAB Ölçeği DFA Regresyon Katsayıları	201
Tablo 6.19. Katılımcıların Demografik Özellikleri	202
Tablo 6.19. Katılımcıların Demografik Özellikleri (Devamı).....	203
Tablo 6.20. ÇBF Ölçeğine İlişkin Tanımlayıcı İstatistikler	205
Tablo 6.21. ÇPK Ölçeğine İlişkin Tanımlayıcı İstatistikler	206
Tablo 6.22. ÇYGPAB Ölçeğine İlişkin Tanımlayıcı İstatistikler.....	207
Tablo 6.23. Korelasyon Analizi Sonuçları	208
Tablo 6.24. Ölçüm Modeli Uyum İyiiliği Değerleri	210
Tablo 6.25. Ölçüm Modeli DFA Regresyon Katsayıları.....	212
Tablo 6.26. Ölçüm Modeli Güvenilirlik, Birleşim ve Ayrışım Geçerliliği	213
Tablo 6.27. Yapısal Model Uyum İyiiliği Değerleri	214
Tablo 6.28. Yapısal Modele İlişkin Bulgular	215
Tablo 6.29. Müşteri Katılımının Proje Başarısına Etkisinde Çevik Pratiklerin Düzenleyici Rolüne İlişkin Regresyon Analizi Sonuçları.....	217
Tablo 6.30. Müşteri Katılımının Proje Başarısına Etkisinde Çevik Pratiklerin Düzenleyici Rolüne İlişkin Özet Sonuçlar	217
Tablo 6.31. Müşteri Katılımının Proje Başarısına Etkisinde Çevik Pratiklerin Farklı Değerlerine Göre Durumsal Etkileri	218
Tablo 6.32. Takım Yeteneğinin Proje Başarısına Etkisinde Çevik Pratiklerin Düzenleyici Rolüne İlişkin Regresyon Analizi Sonuçları.....	221
Tablo 6.33. Takım Yeteneğinin Proje Başarısına Etkisinde Çevik Pratiklerin Düzenleyici Rolüne İlişkin Özet Sonuçlar	222
Tablo 6.34. Takım Yeteneğinin Proje Başarısına Etkisinde Çevik Pratiklerin Farklı Değerlerine Göre Durumsal Etkileri	223
Tablo 6.35. Proje Tanımlama Sürecinin Proje Başarısına Etkisinde Çevik Pratiklerin Düzenleyici Rolüne İlişkin Regresyon Analizi Sonuçları.....	226
Tablo 6.36. Proje Tanımlama Sürecinin Proje Başarısına Etkisinde Çevik Pratiklerin Düzenleyici Rolüne İlişkin Özet Sonuçlar	227

Tablo 6.37. Proje Tanımlama Sürecinin Proje Başarısına Etkisinde Çevik Pratiklerin Farklı Değerlerine Göre Durumsal Etkileri.....	228
Tablo 6.38. Proje Yönetim Sürecinin Proje Başarısına Etkisinde Çevik Pratiklerin Düzenleyici Rolüne İlişkin Regresyon Analizi Sonuçları.....	231
Tablo 6.39. Proje Yönetim Sürecinin Proje Başarısına Etkisinde Çevik Pratiklerin Düzenleyici Rolüne İlişkin Özet Sonuçlar	232
Tablo 6.40. Proje Yönetim Sürecinin Proje Başarısına Etkisinde Çevik Pratiklerin Farklı Değerlerine Göre Durumsal Etkileri.....	233
Tablo 6.41. Hipotez Test Sonuçlarının Özeti	237



ŞEKİLLER DİZİNİ

Şekil 1.1. Yazılım Bileşenleri	3
Şekil 1.2. Donanım Ürünü Arıza Oranı Eğrisi	4
Şekil 1.3. Yazılım Arıza Oranı Eğrisi	5
Şekil 1.4. Yazılım Türleri	7
Şekil 1.5. Yazılım Uygulamaları	9
Şekil 1.6. Yazılım Mühendisliği Katmanları	13
Şekil 1.7. Yazılım Mühendisliği ile İlişkili Disiplinler	15
Şekil 2.1. Kodla ve Düzelt	16
Şekil 2.2. Yazılım Geliştirme Yaşam Döngüsü	19
Şekil 2.3. Yazılım Geliştirme Modellerinin Zaman Çizelgesi	21
Şekil 2.4. Şelale Modeli	23
Şekil 2.5. Artımlı Geliştirme Modeli	25
Şekil 2.6. Prototip Model	26
Şekil 2.7. V Modeli	28
Şekil 2.8. Spiral Model	29
Şekil 2.9. Çevik Yöntemler	31
Şekil 2.10. Çevik Değerler, İlkeler ve Uygulamalar Arasındaki İlişki	32
Şekil 2.11. Şelale, Yinelemeli ve Uç Programlama Yazılım Geliştirme Süreci	36
Şekil 2.12. Uç Programlama Süreci	38
Şekil 2.13. Uç Programlama Değerleri	39
Şekil 2.14. Scrum Değerleri	41
Şekil 2.15. Scrum Süreci	43
Şekil 2.16. Ürün İş Listesi, Sprint İş Listesi ve Ürün Parçası	46
Şekil 2.17. Temel Kanban Tahtası	48
Şekil 2.18. Kanban Süreci ve Ayrıntılı Kanban Tahtası	49
Şekil 2.19. Özellik Odaklı Geliştirme Süreci	52
Şekil 2.20. Test Odaklı Geliştirme Süreci	53
Şekil 2.21. Hızlı Uygulama Geliştirme Süreci	55
Şekil 2.22. Uyarlanabilir Yazılım Geliştirme Süreci	56
Şekil 2.23. Dinamik Sistem Geliştirme Süreci	57
Şekil 2.24. Kristal Aile Metodolojisi	59

Şekil 2.25. Scrumban Süreci.....	61
Şekil 2.26. Scrum/XP Hibrit	62
Şekil 2.27. Aşağı Tüketim Grafiği	87
Şekil 2.28. Yukarı Tüketim Grafiği.....	87
Şekil 2.29. Kanban Panosu	89
Şekil 2.30. Scrum Panosu	89
Şekil 2.31. Scrum ve Kanban Panosunun Örneği – Benzerlikler ve Farklılıklar	90
Şekil 3.1. Demir Üçgen (Proje Yönetim Üçgeni)	93
Şekil 3.2. Yeni Dörtlü Kısıtlama	94
Şekil 5.1. Araştırma Modeli ve Hipotezleri	154
Şekil 5.2. Düzenleyici Modelin Kavramsal Gösterimi	181
Şekil 5.3. Düzenleyici Modelin İstatistiksel Gösterimi	182
Şekil 5.4. Process Makro Model 2 Kavramsal ve İstatistiksel Gösterimi.....	183
Şekil 6.1. ÇBF Ölçeği Birinci Düzey Çok Faktörlü DFA Modeli	193
Şekil 6.2. ÇPK Ölçeği Birinci Düzey Çok Faktörlü DFA Modeli	195
Şekil 6.3. ÇPK Ölçeği İkinci Düzey Çok Faktörlü DFA Modeli.....	198
Şekil 6.4. ÇYGPAÖ Ölçeği DFA Modeli.....	201
Şekil 6.5. Ölçüm Modeli.....	211
Şekil 6.6. Yapısal Model.....	214
Şekil 6.7. Müşteri Katılımının Proje Başarısına Etkisinde Sosyal Pratiklerin Düzenleyici Rolünün Grafıksel Gösterimi	220
Şekil 6.8. Müşteri Katılımının Proje Başarısına Etkisinde Sosyal ve Teknik Pratiklerin Birlikte Düzenleyici Rolünün Grafıksel Gösterimi.....	220
Şekil 6.9. Takım Yeteneğinin Proje Başarısına Etkisinde Sosyal Pratiklerin Düzenleyici Rolünün Grafıksel Gösterimi	224
Şekil 6.10. Takım Yeteneğinin Proje Başarısına Etkisinde Teknik Pratiklerin Düzenleyici Rolünün Grafıksel Gösterimi	225
Şekil 6.11. Takım Yeteneğinin Proje Başarısına Etkisinde Sosyal ve Teknik Pratiklerin Birlikte Düzenleyici Rolünün Grafıksel Gösterimi.....	225
Şekil 6.12. Proje Tanımlama Sürecinin Proje Başarısına Etkisinde Sosyal Pratiklerin Düzenleyici Rolünün Grafıksel Gösterimi	230

Şekil 6.13. Proje Tanımlama Sürecinin Proje Başarısına Etkisinde Sosyal ve Teknik Pratiklerin Birlikte Düzenleyici Rolünün Grafıksel Gösterimi.....	230
Şekil 6.14. Proje Yönetim Sürecinin Proje Başarısına Etkisinde Sosyal Pratiklerin Düzenleyici Rolünün Grafıksel Gösterimi	235
Şekil 6.15. Proje Yönetim Sürecinin Proje Başarısına Etkisinde Teknik Pratiklerin Düzenleyici Rolünün Grafıksel Gösterimi	235
Şekil 6.16. Proje Yönetim Sürecinin Proje Başarısına Etkisinde Sosyal ve Teknik Pratiklerin Birlikte Düzenleyici Rolünün Grafıksel Gösterimi.....	236



ÖNSÖZ

Tez çalışmamın her aşamasında değerli bilgi ve rehberlikleriyle bana yol gösteren, aynı zamanda doktora eğitimine başlamama vesile olan kıymetli danışman hocam Sayın Doç. Dr. Mustafa KESKİNKILIÇ'a en içten teşekkürlerimi ve minnettarlığımı sunarım.

Çalışmaya katkılarıyla ufkumu genişleten, yapıcı eleştirileri ve yönlendirmeleriyle tezin şekillenmesine katkı sağlayan değerli tez izleme komitesi üyesi hocalarım Sayın Prof. Dr. Ömer ALKAN ve Sayın Doç. Dr. Ahmet Kamil KABAKUŞ'a teşekkürlerimi sunarım.

Tez savunma jürimde yer alarak beni onurlandıran değerli hocalarım Sayın Doç. Dr. Handan ÇAM ve Sayın Doç. Dr. Emine Demet EKİNCİ HAMAMCI'ya kıymetli katkıları ve yapıcı eleştirileri için teşekkürlerimi sunarım.

Hayatım boyunca maddi ve manevi destekleriyle her zaman yanımda olan sevgili anneme, babama ve ablama, özellikle bu zorlu süreçte dualarıyla bana güç verdikleri için sonsuz teşekkürlerimi sunarım.

Hayat yolculuğumuzda ve doktora eğitimim süresince bana göstermiş olduğu sabrı, sevgisi ve anlayışıyla her zaman yanımda olan sevgili eşim Selcen BALCI'ya sonsuz teşekkürlerimi sunarım. Hayatıma anlam katan canım evlatlarım Utkan Taha, Aylin Beril ve Umut'a gönülden teşekkür ederim. Onların sevgisi, masumiyetleri ve varlıkları karşılaştığım zorlukların üstesinden gelmemde daima en büyük ilham kaynağım olmuştur. İyi ki varsınız.

Son olarak, akademik yolculuğum boyunca yanımda olan tüm dostlarıma, arkadaşlarıma ve bu süreçte bana destek olan herkese gönülden teşekkür ederim.

GİRİŞ

Bilgi çağında yazılım ürünleri eğitim, sağlık, finans ve iletişim gibi pek çok alanda hayatın vazgeçilmez bir parçası haline gelmiştir. Bireylerin, kurumların ve toplumların ihtiyaçlarını karşılayan bu ürünlerin geliştirilmesinde ise farklı metodolojiler uygulanmaktadır. Küreselleşme ve teknolojik gelişmelerin hızla ilerlemesi, yazılım ürünlerinin hızlı bir şekilde geliştirilmesini ve değişen ihtiyaçlara uyum sağlamasını zorunlu kılmaktadır. Bu gereklilik, yazılım geliştirme süreçlerinde esneklik ve adaptasyon sağlayan çevik yöntemlerin yaygınlaşmasını teşvik etmiştir. Çevik yöntemler, müşteri odaklılığı artırarak sürekli geri bildirim ve yinelemeli süreçler aracılığıyla yazılım geliştirme projelerinin daha etkin bir şekilde yönetilmesini sağlamaktadır (Kirpitsas ve Pachidis, 2022).

Çevik yöntemlerle geliştirilen yazılım projeleri, geleneksel yöntemlere kıyasla daha yüksek başarı oranlarına sahip olmakla birlikte, projelerin yarısından fazlası hâlâ sorunlu (bütçe aşımı, gecikme veya özellik kaybı) ya da başarısızlıkla (iptal) sonuçlanmaktadır (Mersino, 2020). Bu durum, çevik yazılım geliştirme projelerinin başarısına katkı sağlayacak kritik başarı faktörlerinin araştırılmasının önemini ortaya çıkarmıştır.

Kritik başarı faktörleri, bir organizasyonun veya projenin genel başarısı için kritik öneme sahip sınırlı sayıda alan olarak tanımlanmaktadır (Chow ve Cao, 2008). Çevik yazılım geliştirme projelerinin kritik başarı faktörlerini belirlemek amacıyla literatürde çeşitli çalışmalar gerçekleştirilmiştir. İlgili çalışmalar, çevik yöntemlerin kritik başarı faktörlerini farklı yaklaşımlarla ele almıştır. Bazı araştırmalar, bu faktörleri geniş bir perspektiften değerlendirerek tüm faktörleri kapsayıcı bir analiz sunarken, diğerleri ise belirli faktörlere odaklanarak ayrıntılı bir şekilde incelemiştir.

Tez çalışmasının odağı, literatürde önemi sıklıkla vurgulanan insan ve süreç faktörlerinin proje başarısı üzerindeki etkilerini ve çevik pratiklerin bu etkilerde nasıl bir rol oynadığını detaylı bir şekilde incelemektir (Chow ve Cao, 2008; Litchmore, 2016; Stanberry, 2018). Araştırmada insan faktörleri müşteri katılımı ve takım yeteneği; süreç faktörleri ise proje tanımlama süreci ve proje yönetim süreci olarak ele alınmıştır. Çevik pratikler ise sosyal ve teknik pratikler olmak üzere iki grupta değerlendirilmiştir.

Bu kapsamda, tez çalışma altı ana bölümden oluşmaktadır.

Birinci bölümde, yazılım ve yazılım mühendisliğine ilişkin kavramsal çerçeve sunulmuştur.

İkinci bölümde, yazılım geliştirme metodolojileri ele alınmış; yazılım geliştirme yaşam döngüsü, geleneksel yazılım geliştirme yöntemleri ve çevik yazılım geliştirme yöntemleri detaylı bir şekilde incelenmiştir. Ayrıca çevik yazılım geliştirme sürecinde kullanılan pratiklere ilişkin bilgilere de yer verilmiştir.

Üçüncü bölümde, yazılım projeleri, yazılım proje başarısı ve kritik başarı faktörleri kapsamlı bir şekilde ele alınmıştır.

Dördüncü bölümde, tez konusuyla ilgili literatürdeki çalışmalara başlıklar halinde yer verilmiştir.

Beşinci bölümde, araştırmanın metodolojisi açıklanmış; araştırma yöntemi, amacı, önemi, kısıtları, modeli, hipotezleri, evreni ve örnekleme, veri toplama araçları, veri toplama süreci ve verilerin analiz süreci ayrıntılı olarak sunulmuştur.

Altıncı bölümde ise analizlerden elde edilen bulgular başlıklar halinde detaylandırılmıştır.

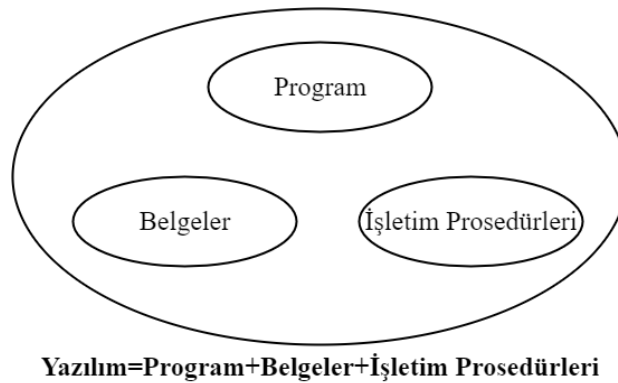
Sonuç bölümünde, araştırma bulguları özetlenmiş ve bu bulgulardan elde edilen sonuçlar değerlendirilmiştir. Çalışmanın literatüre ve uygulamaya katkıları tartışılmış ve gelecekteki araştırmalar için önerilere yer verilmiştir.

BİRİNCİ BÖLÜM

YAZILIM ve YAZILIM MÜHENDİSLİĞİ

1.1. YAZILIM

Yazılım “bir bilgisayarda donanıma hayat veren ve bilgi işlemde kullanılan programlar, yordamlar, programlama dilleri ve belgelerin tümü” olarak tanımlanmaktadır (Türk Dil Kurumu [TDK], t.y.). Pressman (2010) yazılımı; yürütüldüğü zaman istenen işlevi ve performansı bilgisayar programları ile sağlayan, programların bilgileri doğru şekilde işlemesi için kullanılan veri yapılarını içeren bir ürün olarak tanımlar. Sommerville (2021) ise yazılımı, belirli bir müşteri veya genel bir pazar için geliştirilebilen bilgisayar programları ve ilişkili belgeleme olarak ifade etmiştir. Elektrik ve Elektronik Mühendisleri Enstitüsü’nün (Institute of Electrical and Electronics Engineers – IEEE) yazılım mühendisliği alanında kullanılan terimlerin tanımlarını standartlaştırmak amacıyla 1983 yılında yayınlanan ve 1990 yılında revize edilen “IEEE Yazılım Mühendisliği Terimleri Standart Sözlüğü”nde (IEEE Standard Glossary of Software Engineering Terminology)” ise yazılım “bir bilgisayar sisteminin işletimine ilişkin bilgisayar programları, yordamlar ve bunlarla ilişkili belgeler ve veriler” olarak tanımlanmıştır (IEEE, 1990). Yazılım yalnızca program kodundan değil, aynı zamanda gereksinim spesifikasyon belgeleri, tasarım belgeleri, uygulama belgeleri, test belgeleri gibi tüm ilişkili belgelerin yanı sıra kullanıcı ve işletim kılavuzlarını içeren işletim prosedürlerinden (yordamlar) oluşmaktadır (Agarwal vd., 2010).



Şekil 1.1. Yazılım Bileşenleri

Kaynak: Agarwal vd. (2010)

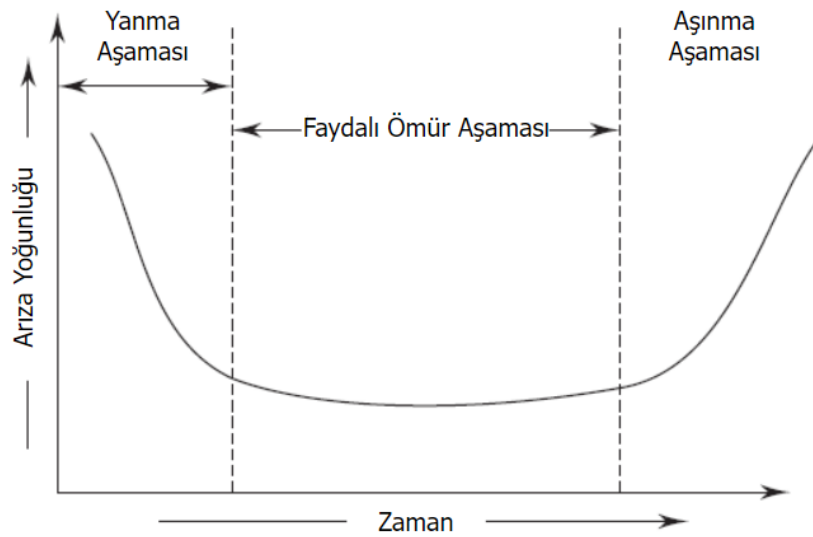
Yazılımlar yürüttüğü işlevler ve yetenekleri ile tanımlanmaktadır. Yetenekler yazılımın sağladığı özellikler ve sunduğu olanaklarla ilgilidir. Satış siparişi işleme için yazılan yazılım, farklı pazar segmentlerinden farklı satış siparişi türlerini işlemek için farklı işlevlere sahip olacaktır. Çoklu para birimi hesaplaması, yönetim bilişim sistemi raporlarında ve muhasebe defterlerinde ürün, satış ve vergi durumu güncellenmesinin ele alınması yazılımın sağladığı özelliklere örnek gösterilebilir. Olanaklar, satış siparişlerinin yazdırılması, müşterilere e-postalar, raporlar ve malları göndermeleri için mağaza departmanlarına tavsiyeler olabilir. Yazılımlarda olanaklar ve özellikler isteğe bağlı olabilir ve müşteri tercihlerine bağlı olarak değişkenlik göstermektedir. Ayrıca yazılımlar, platform olarak bilinen belirli donanım ve işletim sistemi hususları göz önünde bulundurularak geliştirilmektedir. Bu nedenle yazılımlar, yetenekleri ve çalıştırmak için gereken platform özellikleriyle birlikte açıklanmaktadır (Agarwal vd., 2010).

1.1.1. Yazılım Özellikleri

Yazılımların temel özellikleri aşağıdaki şekilde açıklanabilir.

i. Yazılım Yıpranmaz.

Donanım ürünleri için arıza oranını zamana bağlı olarak gösteren grafik Şekil 1.2’de yer almaktadır. Küvet eğrisi olarak ta adlandırılan eğride donanım ürünlerinin ömrünü belirtilen aşamalar yer almaktadır (Aggarwal ve Singh, 2005).

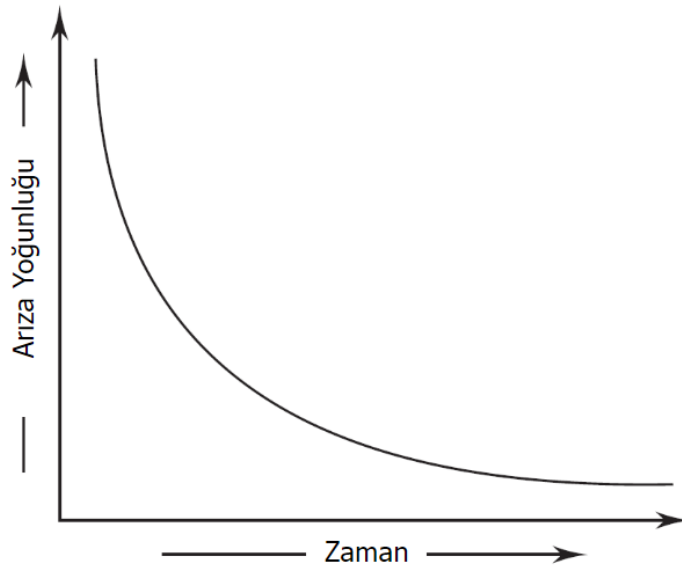


Şekil 1.2. Donanım Ürünü Arıza Oranı Eğrisi

Kaynak: Aggarwal ve Singh (2005)

Donanım ürünlerinin ömrü üç aşamadan oluşur. İlk aşama, donanımın ömrünün erken dönemlerinde nispeten yüksek arıza yoğunluğuna sahip olduğu yanma aşamasıdır. Bu aşamada, genellikle donanım ürünü test edilir ve tasarım ya da üretim hataları düzeltilir. Hataların test edilmesi ve düzeltilmesi sonucunda arıza yoğunluğu azalır ve arıza oranı belirli bir süre sonra sabit bir duruma ulaşır, bu da ikinci aşamayı oluşturur. İkinci aşama, arıza yoğunluğunun yaklaşık olarak sabit kaldığı ve bir ürünün kullanım ömrü olarak adlandırılan faydalı ömür aşamasıdır. Ancak zamanla donanım bileşenleri toz, titreşim, kötü kullanım, aşırı sıcaklıklar ve diğer çevresel etkenler nedeniyle yeniden arızalanmaya başlar. Bu süreç, aşınma aşaması olarak bilinen üçüncü aşamadır. Ancak, yazılım için bu aşama geçerli değildir çünkü yazılım fiziksel olarak yıpranmaz (Aggarwal ve Singh, 2005; Pressman, 2010).

Yazılım için arıza oranı eğrisi, Şekil 1.3'te gösterilen “idealize edilmiş eğri” biçimindedir. Kusurlar, bir programın ömrünün erken dönemlerinde yüksek arıza oranlarına neden olur. Ancak, bunlar düzeltilir ve eğri gösterildiği gibi düzleşir. İdealize edilmiş eğri, yazılım için gerçek arıza modellerinin aşırı basitleştirilmiş halidir. Önemli nokta, yazılımın zamanla yıpranmak yerine daha güvenilir hale gelmesidir. Yazılım, geliştirildiği ortam değiştiğinde eskimeye başlar. Bu nedenle, çevresel değişiklikler, yeni gereksinimler, yeni beklentiler vb. nedeniyle yazılım emekli edilebilir (Aggarwal ve Singh, 2005; Pressman, 2010).



Şekil 1.3. Yazılım Arıza Oranı Eğrisi

Kaynak: Aggarwal ve Singh (2005)

ii. Yazılım geliştirilir veya tasarlanır; klasik anlamda üretilmez.

Yazılım geliştirme ve donanım üretimi arasında bazı benzerlikler olsa da bu iki faaliyet temelde farklıdır. Her iki faaliyette de yüksek kalite iyi tasarımıyla sağlanır. Ancak donanımın üretim aşaması, yazılımda var olmayan veya kolayca düzeltilebilen kalite sorunlarına yol açabilir. Her iki faaliyet de insanlara bağlıdır, ancak uygulanan insan gücü ile elde edilen iş miktarı arasındaki ilişki tamamen farklıdır. Her iki faaliyet bir “ürün” inşasını gerektirir, ancak yaklaşımlar farklıdır. Her iki faaliyet bir “ürün” inşasını gerektirir, ancak yaklaşımlar farklıdır. Yazılım maliyetleri mühendislik sürecinde yoğunlaşır. Bu, yazılım projelerinin üretim projeleriymiş gibi yönetilemeyeceği anlamına gelmektedir (Pressman, 2010).

iii. Bileşenlerin Yeniden Kullanılabilirliği

Çoğu yazılım, mevcut bileşenlerden bir araya getirilmek yerine özel olarak oluşturulur. Çoğu yazılım özel olarak oluşturulmaya devam etse de son gelişmeler bileşen tabanlı olma eğilimindedir (Agarwal vd., 2010). Bir yazılım bileşeni, birçok farklı programda yeniden kullanılabilir şekilde tasarlanmalı ve uygulanmalıdır. Modern yeniden kullanılabilir bileşenler hem verileri hem de verilere uygulanan işlemeyi kapsülleyerek yazılım mühendisinin yeniden kullanılabilir parçalardan yeni uygulamalar oluşturmasını sağlar. Örneğin; günümüzün etkileşimli kullanıcı arayüzleri, grafik pencerelerinin, açılır menülerin ve çok çeşitli etkileşim mekanizmalarının oluşturulmasını sağlayan yeniden kullanılabilir bileşenlerle oluşturulmuştur. Arayüzü oluşturmak için gerekli veri yapıları ve işlem detayları, arayüz oluşturma için yeniden kullanılabilir bileşenlerden oluşan bir kütüphanede yer almaktadır (Pressman, 2010).

iv. Yazılım Esnektir.

Yazılım, neredeyse her görevi yerine getirecek şekilde geliştirilebilir. Bazı durumlarda, belirli bir özellik en değerli özellik olarak kabul edilerek çeşitli değişikliklere uyum sağlamayı mümkün kılabilir (Aggarwal ve Singh, 2005). Bu durum, yazılımın değişen gereksinimlere, kullanıcı ihtiyaçlarına veya teknolojik koşullara uyum sağlama kapasitesini ifade etmektedir. Bu uyum yeteneği, yazılım sistemlerinin esnekliğini ortaya koymaktadır (Agarwal vd., 2010; Pressman, 2010; Sommerville, 2021).

1.1.2. Yazılım Türleri

Bilgisayar yazılımları genellikle sistem yazılımları ve uygulama yazılımları olmak üzere iki kategoriye ayrılmaktadır.

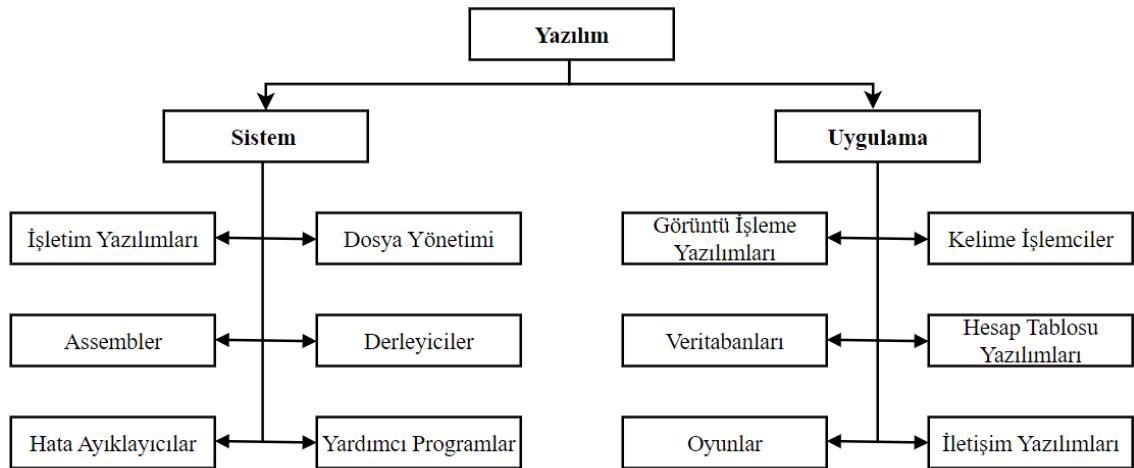
a) Sistem Yazılımları:

İşletim sistemini ve bilgisayarın çalışmasını sağlayan tüm yardımcı programları içerir. İşletim sistemleri, cihaz sürücüler ve yardımcı programlar bu tür yazılımların örnekleridir. Bu yazılımlar, donanım ve uygulama yazılımları arasında bir köprü görevi görmektedir (Silberschatz vd., 2014).

b) Uygulama Yazılımları:

Kullanıcılar için gerçek işler yapan programlardır. Ofis programları, medya oynatıcılar, grafik tasarım yazılımları gibi yazılımlardır. Uygulama yazılımları, günlük kullanımda en yaygın karşılaşılan yazılım türlerindendir (Pressman ve Maxim, 2015).

Sistem yazılımları, bilgisayarla çok temel düzeyde etkileşime giren ve işletim sistemini ve kaynakları yönetmek için yardımcı programları içeren düşük seviyeli programlardır. Uygulama yazılımları sistem yazılımlarının üstünde yer almaktadır. Çünkü çalışmak için sistem yazılımına ihtiyaç duymaktadır. Şekil 1.4 yazılım türlerine genel bir bakış sunmaktadır (Agarwal vd., 2010).



Şekil 1.4. Yazılım Türleri

Kaynak: Agarwal vd. (2010)

1.1.3. Yazılım Sınıfları

Yazılımlar genel amaçlı yazılımlar ve özel amaçlı yazılımlar olmak üzere iki ana sınıfa ayrılmaktadır.

a) Genel amaçlı yazılımlar:

Yazılım gereksinimlerinin çok yaygın, oldukça sabit ve yazılım mühendisleri tarafından iyi anlaşılan geniş bir müşteri pazarı için tasarlanan ürünlerdir. Bu ürünler açık piyasada satılmaktadır ve piyasada birkaç rakip ürün bulunabilir. Veri tabanı ürünleri, tarayıcılar, CAD/CAM paketleri, işletim sistemleri ve sistem yazılımları genel amaçlı yazılım örnekleridir (Aggarwal ve Singh, 2005).

b) Özel Amaçlı Yazılımlar:

Belirli bir müşterinin gereksinimlerine, ortamına ve alanına özgü olarak geliştirilen yazılımdır. Süreç kontrol sistemleri, trafik yönetim sistemleri, hastane yönetim sistemleri ve üretim süreç kontrol sistemleri gibi alanlarda özel amaçlı yazılımlar gerekmektedir.

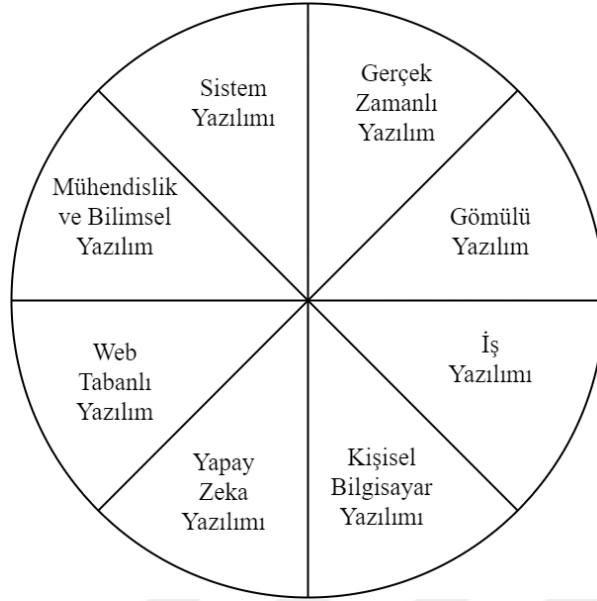
Genel amaçlı bir yazılımın gereksinimleri ve özellikleri geliştirici tarafından kontrol edilirken, özel amaçlı yazılımların gereksinimleri ve özellikleri müşteri tarafından kontrol edilir ve o endüstrinin uygulamaları tarafından şekillendirilmektedir (Agarwal vd., 2010).

1.1.4. Yazılım Uygulamaları

İnsan yaşamının çoğu alanının ayrılmaz bir parçası haline gelen yazılımlar kullanım alanları açısından Şekil 1.5'te belirtilen sekiz alanda gruplandırılmıştır (Aggarwal ve Singh, 2005; Agarwal vd., 2010).

i. Sistem Yazılımı:

Diğer yazılımlara hizmet sağlamak için kullanılan programların koleksiyonudur. İşletim sistemleri, derleyiciler, editörler, sürücüler, yardımcı programlar gibi altyapı yazılımları sistem yazılımına örnek verilebilir. Sistem yazılımı olmayan bir bilgisayar sistemi çalışamaz (Aggarwal ve Singh, 2005; Pressman ve Maxim, 2015).



Şekil 1.5. Yazılım Uygulamaları
Kaynak: Aggarwal ve Singh (2005)

ii. Gerçek Zamanlı Yazılım:

Bu yazılımlar, gerçek dünyadaki olayları meydana geldikçe izlemek, kontrol etmek ve analiz etmek için kullanılır. Örnek olarak, hava durumu tahmini için gerekli yazılım verilebilir. Bu tür yazılımlar, sıcaklık, nem ve diğer çevresel parametrelerin durumunu toplar ve işler, böylece hava durumunu tahmin eder (Aggarwal ve Singh, 2005).

iii. Gömülü Yazılım:

Bir ürün veya sistemin Salt Okunur Belleği (Read-Only Memory, ROM) içerisinde bulunan gömülü yazılımlar son kullanıcı ve sistemin kendisi için özellikleri ve işlevleri uygulamak ve kontrol etmek için kullanılır (Pressman ve Maxim, 2015). Ürün bir uçak, otomobil, güvenlik sistemi, sinyalizasyon sistemi, güç santrallerinin kontrol birimi vb. olabilir. Gömülü yazılım, donanım bileşenlerini yönetir ve aynı zamanda akıllı yazılım olarak da adlandırılır (Aggarwal ve Singh, 2005).

iv. İş Yazılımı:

İş süreçlerine yönelik tasarlanan yazılım, iş yazılımı olarak adlandırılır. İş yazılımları muhasebe paketleri, yönetim bilgi sistemleri, bordro paketleri ve envanter yönetimi gibi belirli işlemler için kullanılır (Aggarwal ve Singh, 2005; Agarwal vd., 2010).

v. Kişisel Bilgisayar Yazılımı:

Kişisel bilgisayarlarda kullanılan yazılımlar bu kategoride yer almaktadır. Kelime işlemciler, bilgisayar grafikleri, multimedya ve animasyon araçları, veritabanı yönetimi, bilgisayar oyunları vb. örneklerdir. Kişisel bilgisayar yazılımları çok yükselen bir alan olup, birçok büyük kuruluş burada yoğunlaşmaktadır çünkü geniş bir müşteri tabanına sahiptir (Aggarwal ve Singh, 2005).

vi. Yapay Zekâ Yazılımı:

Yapay Zekâ Yazılımı, sayısal olmayan algoritmalar kullanarak, doğrudan hesaplama veya basit analizle çözilemeyen karmaşık problemleri çözmek için kullanılır. Örnekler uzman sistemler, yapay sinir ağları, sinyal işleme yazılımı vb. olabilir (Aggarwal ve Singh, 2005).

vii. Web Tabanlı Yazılım:

Web uygulamalarıyla ilgili yazılımlar bu kategoriye girer. Bu ağ merkezli yazılım kategorisi, çok çeşitli uygulamaları kapsar ve hem tarayıcı tabanlı uygulamaları hem de mobil cihazlarda bulunan yazılımları kapsamaktadır (Pressman ve Maxim, 2015). Örnekler arasında CGI, HTML, Java, Perl vb. bulunmaktadır (Aggarwal ve Singh, 2005).

viii. Mühendislik ve Bilimsel Yazılım:

Bilimsel ve mühendislik uygulama yazılımları bu kategoride toplanır. Verileri işlemek için genellikle büyük hesaplama gücü gereklidir. Örnekler arasında CAD/CAM paketleri, SPSS, MATLAB, devre analizörleri vb. bulunmaktadır (Aggarwal ve Singh, 2005).

1.2. YAZILIM MÜHENDİSLİĞİ

NATO Bilim Komitesi, yazılım geliştirme ile ilgili kritik sorunları ele almak ve tartışmak amacıyla iki önemli konferans düzenlemiştir. İlk konferans, 1968 yılında Almanya'nın Garmisch kentinde, ikinci konferans ise 1969 yılında Roma'da gerçekleştirilmiştir. NATO konferansları, 1960'ların sonlarında yazılım sektöründe var olan sorunları ortaya koymuş ve bu sorunları tanımlamak için "yazılım krizi" terimi kullanılmıştır. Bu kriz, bütçe ve zaman aşımı gibi problemlerin yanı sıra teslim edilen yazılımların kalitesi ve güvenilirliği gibi sorunları da içeriyordu (O'Regan, 2021).

Konferanslar, yazılım üretiminin mevcut durumunu ve gelecekteki potansiyelini değerlendirmeyi amaçlamış ve yazılım geliştirme süreçlerinde karşılaşılan sorunlara yönelik çözümler sunmayı hedeflemiştir (Shaw, 1990). Bu bağlamda, yazılım mühendisliğinin bağımsız bir disiplin olarak ortaya çıkmasına ve programlamanın bilim ve matematikten farklı, mühendislik odaklı bir alan olduğunun anlaşılmasına yol açmıştır (O'Regan, 2021).

İlk kez 1968 yılında NATO tarafından düzenlenen konferansta kullanılan “yazılım mühendisliği” ifadesi, yazılım geliştiricilerin ilgisini çekmiş ve 1970’ler boyunca popülerlik kazanmıştır (Shaw, 1990). Konferansta Fritz Bauer, yazılım mühendisliğini “gerçek makinelerde güvenilir ve verimli çalışan, ekonomik olarak uygun yazılımlar elde etmek amacıyla sağlam mühendislik prensiplerinin oluşturulması ve uygulanması” şeklinde tanımlamıştır (Naur ve Randell, 1969). Günümüzde yazılım mühendisliği terimi, yazılım geliştirme süreçlerinde yönetsel süreçler, yazılım araçları ve tasarım faaliyetlerini kapsayan bir disiplin olarak kullanılmaktadır (Shaw, 1990).

Literatürde yazılım mühendisliği için çeşitli tanımlar bulunmaktadır. Bu tanımlar;

Schach, yazılım mühendisliğini “zamanında, bütçe dahilinde teslim edilen ve gereksinimlerini karşılayan kaliteli yazılım üretimi” olarak tanımlar (Schach, 1990; aktaran Aggarwal ve Singh, 2005).

Yazılım mühendisliği, profesyonellerin yüksek kaliteli bilgisayar yazılımları oluşturmaya olanak tanıyan bir süreci, bir dizi yöntemi (uygulamayı) ve bir dizi aracı kapsar (Pressman ve Maxim, 2015).

Yazılım mühendisliği yazılımın analizi, tasarımı, uygulanması, test edilmesi ve bakımı için genel olarak kabul görmüş bir dizi hedef bağlamında kullanılan sistematik bir prosedürü ifade eder (Leach, 2016).

Yazılım mühendisliği, yazılım üretiminin başlangıcındaki ilk fikirden işletim ve bakıma kadar olan tüm yönleriyle ilgilenen bir mühendislik disiplini (Sommerville, 2021).

Yazılım mühendisliği (1) “yazılımın geliştirilmesi, işletilmesi ve bakımı için sistematik, disiplinli, ölçülebilir bir yaklaşımın uygulanması yani mühendisliğin yazılıma

uygulanması”, (2) “(1)’deki gibi yaklaşımların incelenmesi” şeklinde tanımlanmıştır (IEEE, 1990).

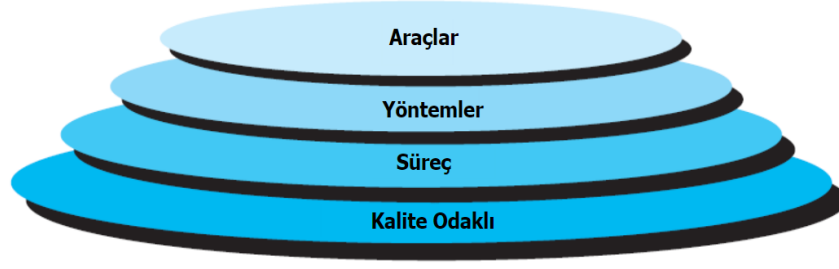
Yazılım mühendisliği, çeşitli yazarlar ve kuruluşlar tarafından tanımlanmış ve farklı bakış açılarıyla ele alınmıştır. Bu tanımları incelediğimizde, yazılım mühendisliğinin temel olarak yazılım geliştirme süreçlerini sistematik, disiplinli ve verimli bir şekilde yürütmeyi amaçlayan bir mühendislik disiplini olduğu sonucuna varılabilir. Tanımlarda öne çıkan ortak noktalar ise yazılımın kalitesi, süreçlerin sistematik yapısı, disiplinli yaklaşımlar ve yazılımın tüm yaşam döngüsünü kapsayan bir yapıya sahip olmasıdır.

Yazılım mühendisliği şunları içerir (O’Regan, 2021):

1. Müşterilerin ihtiyaçlarını karşılayacak yazılımların tasarlanması, geliştirilmesi ve test edilmesine yönelik metodolojiler.
2. Yazılım mühendisliği uygulanır. Yani yazılım ürünleri, mühendislik ilkelerine uygun olarak düzgün bir şekilde tasarlanır, geliştirilir ve test edilir.
3. Kalite ve güvenlik uygun bir şekilde ele alınır.
4. Yazılım ürünlerinin tasarımı ve doğrulamasında matematik kullanılabilir. Kullanılacak matematik seviyesi, ürünün güvenlik açısından kritik doğasına bağlıdır. Kaliteyi yazılıma entegre etmek için genellikle sistematik akran incelemeleri ve titiz testler yeterli olacaktır. Ağır matematiksel teknikler ise yalnızca güvenlik ve güvenilirlik açısından kritik yazılımlar için ayrılmıştır.
5. Sağlam proje yönetimi ve kalite yönetimi uygulamaları kullanılır.
6. Yazılımın desteklenmesi ve bakımı uygun bir şekilde ele alınır.

1.2.1. Yazılım Mühendisliği Katmanları

Pressman ve Maxim (2015), yazılım mühendisliğinin karmaşık bilgisayar tabanlı sistemlerin zamanında ve kaliteli bir şekilde inşa edilmesini sağlayan süreçleri, yöntemleri ve araçları kapsayan katmanlı bir teknoloji olduğunu belirtmiştir. Yazılım mühendisliğinin katmanları Şekil 1.6’da gösterilmiştir.



Şekil 1.6. Yazılım Mühendisliği Katmanları

Kaynak: Pressman ve Maxim (2015)

Modelde yazılım mühendisliği dört ana katmana ayrılmıştır.

i. Kalite Odaklı:

Yazılım mühendisliğini destekleyen temel yapı taşı kaliteye odaklanmaktır. Kalite, yazılım mühendisliği sürecinin her aşamasında sağlanması gereken en önemli unsurdur. Kalite yönetimi ve sürekli iyileştirme kültürü bu odakla ilişkilidir (Pressman ve Maxim, 2015).

ii. Süreç:

Yazılım mühendisliği teknolojisinin etkin bir şekilde sunulabilmesi için bir süreç oluşturulmalıdır. Yazılım süreci, yazılım projelerinin yönetim kontrolünün temelini oluşturur ve hangi teknik yöntemlerin uygulanacağı, üretilen iş ürünlerinin (modeller, belgeler, veri, raporlar, formlar vb.) bağlamını sağlar, kalite güvence altına alınır ve değişiklikler uygun şekilde yönetilir (Pressman ve Maxim, 2015).

iii. Yöntemler:

Yöntemler, yazılım geliştirme sürecinde kullanılacak teknik yöntemleri içerir. Bu yöntemler, gereksinim analizi, tasarım modelleme, program oluşturma, test ve destek gibi teknik faaliyetleri yönetir. Yazılım mühendisliği yöntemleri, teknolojinin her alanını yöneten ve modelleme faaliyetleri ile diğer tanımlayıcı teknikleri içeren bir dizi temel ilkeye dayanır (Pressman, 2010).

iv. Araçlar:

Yazılım mühendisliği araçları, süreç ve yöntemlerin uygulanmasına yönelik otomatik veya yarı otomatik destek sağlar. Araçları sayesinde, yazılım mühendisliği sürecinin etkinliği artırılır ve süreç içindeki bilgi entegrasyonu sağlanır (Pressman ve Maxim, 2015).

1.2.2. Yazılım Mühendisliği Bilgi Alanları

Yazılım Mühendisliği Bilgi Birikimi Kılavuzu (Software Engineering Body of Knowledge - SWEBOK), IEEE tarafından 1998 yılından beri yürütülen bir çalışma olup, yazılım mühendisliği alanında genel kabul gören bilgilerin tanımlandığı temel bir dokümandır. En son 2014 yılında yayımlanan 3. sürümüyle güncellenen SWEBOK Kılavuzu, yazılım mühendislerinin mesleki bilgi birikimine yönelik kapsamlı bir kaynak sunmaktadır (IEEE Computer Society, 2014).

Kılavuz, beş temel hedef doğrultusunda oluşturulmuştur. Bu hedeflerden ilki, dünya çapında yazılım mühendisliğine ilişkin tutarlı bir görüşün teşvik edilmesidir. İkinci hedef, yazılım mühendisliğinin kapsamını belirlemek ve bu disiplinin bilgisayar bilimi, proje yönetimi, bilgisayar mühendisliği ve matematik gibi diğer disiplinlerle olan ilişkisini açıklığa kavuşturmadır. Üçüncü hedef, yazılım mühendisliği disiplininin içeriğini karakterize etmek; dördüncü hedef, Yazılım Mühendisliği Bilgi Birikimi'ne konu bazında erişim sağlamaktır. Son olarak, beşinci hedef ise müfredat geliştirme, bireysel sertifikasyon ve lisanslama materyalleri için sağlam bir temel sunmaktır (SWEBOK, 2014).

SWEBOK Kılavuzu, yazılım mühendisliği alanındaki profesyonellere rehberlik eden önemli bir kaynak olarak işlev görmektedir. Tablo 1.1'de SWEBOK rehberinde yer alan 15 bilgi alanı yer almaktadır. Bu bilgi alanları, yazılım mühendisliğinin temel kapsamını oluşturan disiplinleri temsil etmektedir.

Tablo 1.1. Yazılım Mühendisliği Bilgi Alanları

Sıra No	Bilgi Alanı (Türkçe)	Bilgi Alanı (İngilizce)
1	Yazılım Gereksinimleri	Software Requirements
2	Yazılım Tasarımı	Software Design
3	Yazılım İnşası	Software Construction
4	Yazılım Testi	Software Testing
5	Yazılım Bakımı	Software Maintenance
6	Yazılım Yapılandırma Yönetimi	Software Configuration Management
7	Yazılım Mühendisliği Yönetimi	Software Engineering Management
8	Yazılım Mühendisliği Süreci	Software Engineering Process
9	Yazılım Mühendisliği Modelleri ve Yöntemleri	Software Engineering Models and Methods
10	Yazılım Kalitesi	Software Quality

Tablo 1.1. Yazılım Mühendisliği Bilgi Alanları (Devamı)

Sıra No	Bilgi Alanı (Türkçe)	Bilgi Alanı (İngilizce)
11	Yazılım Mühendisliği Mesleki Uygulaması	Software Engineering Professional Practice
12	Yazılım Mühendisliği Ekonomisi	Software Engineering Economics
13	Bilgisayar Temelleri	Computing Foundations
14	Matematik Temelleri	Mathematical Foundations
15	Mühendislik Temelleri	Engineering Foundations

Kaynak: SWEBOK (2014)

1.2.3. Yazılım Mühendisliği ile İlişkili Disiplinler

SWEBOK rehberi yazılım mühendisliğinin kapsamı belirlenirken, yazılım mühendisliği ile etkileşimde olan alanların tanımlanmasının önemli olduğunu vurgulamaktadır. Yazılım mühendisliği ile kesişen ve ilişkili olan disiplinler Şekil 1.7’de yer almaktadır (SWEBOK, 2014).

**Şekil 1.7.** Yazılım Mühendisliği ile İlişkili Disiplinler

Kaynak: SWEBOK (2014)

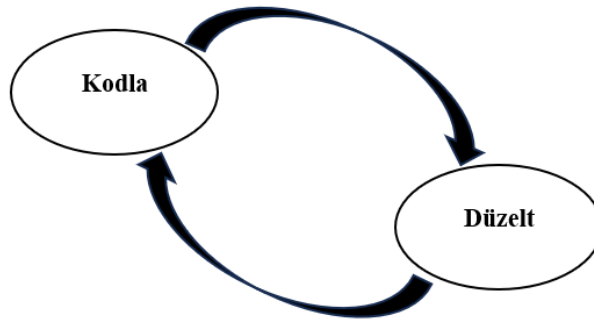
İKİNCİ BÖLÜM

YAZILIM GELİŞTİRME METODOLOJİLERİ

2.1. YAZILIM GELİŞTİRME YAŞAM DÖNGÜSÜ

Yazılım, bilgisayar programları ve ilişkili belgeler olarak tanımlanmakta olup, belirli bir müşteri veya genel bir pazar için geliştirilebilen ve yürütüldüğünde istenilen işlevi ve performansı sağlayan bir üründür (Pressman, 2001; Sommerville, 2015). Pressman (2001) ve Sommerville'in (2015) tanımları, yazılım geliştirme sürecinin, müşteri gereksinimlerini karşılayan işlevsel ve performans özelliklerini sağlamak üzere tasarlandığını vurgulamaktadır.

1950'li yıllarda yazılımlar, herhangi bir gereksinim veya tasarım çabası olmadan, yalnızca müşteri memnuniyetini sağlamak amacıyla gerektiği kadar yeniden düzenlenmiş bir ürün olarak geliştirilmekteydi. Kodla ve düzelt olarak bilinen bu yöntem, doğaçlama bir yaklaşımdır ve belirgin tanımlanmış adımları bulunmamaktadır. Temel olarak iki aşamadan oluşan bu modelde, ilk aşama kodun yazılması, ikinci aşama ise Şekil 2.1'de gösterildiği gibi kodun düzeltilmesidir. Düzeltme aşaması, hata giderme veya ek işlevsellik ekleme anlamına gelmektedir. Kodlama ve düzeltme aşamaları, birbirini takip eden bir döngü şeklinde tekrarlanır ve süreç, müşteri tatmini sağlanana kadar devam etmektedir (Aggarwal ve Singh, 2005; Gencer ve Kayacan, 2017).



Şekil 2.1. Kodla ve Düzelt

Kaynak: Aggarwal ve Singh (2005)

1960'ların sonlarına doğru, yazılım projelerinde belirlenen bütçe ve zaman sınırları içerisinde tamamlama zorluğunun yanı sıra kalite ve güvenilirlik sorunları da ortaya çıkmıştır. Bu sorunlar “yazılım krizi” olarak adlandırılmıştır (O'Regan, 2021). Yazılım geliştirme yaklaşımlarının ciddi eksiklikler içermesi ve kapsamlı bir değişikliğe ihtiyaç duyulması nedeniyle NATO Bilim Komitesi, yazılım geliştirmedeki kritik meseleleri tartışmak üzere iki önemli konferans düzenlemiştir. Bu konferanslarda, yazılım geliştirme süreçlerinde uygulanması gereken standartlar belirlenmiş ve günümüzde kullanılan yazılım süreç modellerinin temelleri atılmıştır (Gencer ve Kayacan, 2017; O'Regan, 2021).

Yazılım geliştirme süreci, bir yazılım sisteminin üretilmesi ile sonuçlanan etkinlikleri içeren karmaşık bir süreçtir (Nizam, 2015; Sommerville, 2015). Bu süreç, belirli bir zaman çizelgesine dayalı olarak planlanmış ve içerik bakımından bölünmüş aşamalardan meydana gelmektedir. Yazılımın işlevleri ile ilgili gereksinimlerin sürekli değişmesi ve genişlemesi nedeniyle, bu aşamalar sürekli bir döngü biçiminde tekrarlı bir yol izlemekte ve bir yaşam döngüsü oluşmaktadır. Bu nedenle, yazılım geliştirme sürecine yazılım geliştirme yaşam döngüsü (YGYD) adı verilmektedir (Keskinkılıç ve Kahveci, 2019). YGYD, yazılım geliştirme sürecini zaman içinde tamamlamak ve yazılımın kalitesini korumak için sistematik bir yaklaşımdır (Rastogi, 2015; Sarker vd., 2015).

Yazılım geliştirme süreçlerinde yaşam döngüsü modeli kullanmanın birçok önemli nedeni bulunmaktadır. Öncelikle, yaşam döngüsü modeli, yazılım geliştirme sürecinin tüm aşamalarını kapsamlı bir şekilde anlamayı sağlar. Bu model, geliştiricileri yapılandırılmış bir yaklaşım benimsemeye yönlendirerek sistematik bir süreç izlemelerini zorunlu kılar. Ayrıca, kaynakların önceden planlanmasına olanak tanır ve bu kaynakların daha sonraki aşamalarda kontrol edilebilmesini sağlar. Yönetim açısından bakıldığında, yaşam döngüsü modeli, proje ilerleyişinin etkin bir şekilde izlenmesine katkıda bulunarak, gelişim sürecinde gerekli müdahalelerin zamanında yapılmasına imkân verir. Bu özellikler, yazılım projelerinin başarıyla tamamlanmasına ve sürdürülebilir yazılım ürünlerinin üretilmesine katkıda bulunur (Agarwal vd., 2010).

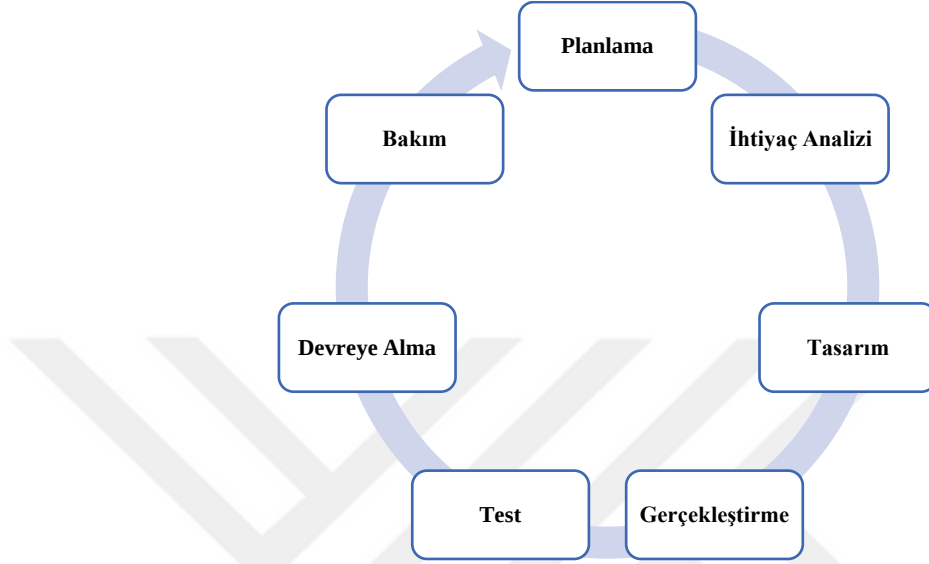
Yazılım yaşam döngüsü, bir yazılım ürününün ömrü boyunca geçirdiği belirli aşamaların veya süreçlerin tanımlanabilir bir dizisidir (Munassar ve Govardhan, 2010;

Thakur vd., 2015). YGYD modeli, yazılım yaşam döngüsünü açıklamak ve görselleştirmek için tanımlayıcı ve resimli bir temsil sunar (Sabale ve Dani, 2012). Bu model, bir yazılım ürününde incelemiden kullanımdan kaldırılıncaya kadar geçen farklı etkinlikleri haritalandırır (Alshamrani ve Bahattab, 2015). YGYD temelde yazılım geliştirme sürecini planlamak, yönetmek ve kontrol etmek için bir çerçeve sağlar. Bu çerçeve, bir bilgi sisteminin başlangıcından sonuna kadar olan aşamaları düzenler ve belirli bir sıra ile takip edilmesini sağlar (Kute ve Thorat, 2014).

YGYP, bir yazılım ürününün tasarlandığı andan, ürünün artık kullanılamaz hale geldiği ana kadar geçen süreci kapsamaktadır. Yazılım yaşam döngüsü genel olarak gereksinimler aşaması, tasarım aşaması, gerçekleştirme (uygulama) aşaması, test aşaması, kurulum ve kontrol aşaması, işletme ve bakım aşamalarını, bazen de emekliye ayrılma aşamasını içermektedir (IEEE, 1990). Ancak literatürde bu aşamalar için farklı isimlendirmeler ve eklemeler mevcuttur.

Örneğin; Nizam (2015), YGYD aşamalarını planlama, ihtiyaç analizi, tasarım, gerçekleştirme, test, devreye alma (teslim, entegrasyon) ve bakım olarak ifade etmektedir. Rather ve Bhatnagar (2015) ise bu aşamaları planlama, tanımlama, tasarım, inşa, test ve dağıtım olarak belirtirken; Despa (2014), planlama, tasarım, geliştirme, test, kurulum ve bakım şeklinde tanımlamaktadır. Alazzawi vd. (2023), ihtiyaç analizi, tasarım, gerçekleştirme, test ve bakım olmak üzere YGYD beş aşama olarak belirtmiştir. Thakur vd. (2015), yazılım geliştirme yaşam döngüsünde genel olarak takip edilen temel aşamaları belirlemiştir. Süreç, ilk olarak sorunun anlaşılması ile başlar; bu aşamada gereksinimlerin toplanması yoluyla yazılımın hedefleri ve ihtiyaçları tanımlanır. İkinci aşamada, toplanan gereksinimlere uygun çözüm planına karar verilir ve bu çerçevede yazılım tasarımı oluşturulur. Üçüncü aşamada, planlanan çözüm kodlanır ve yazılımın fonksiyonel hale getirilmesi sağlanır. Sonraki aşamada gerçek program test edilerek belirlenen gereksinimleri karşılayıp karşılamadığı değerlendirilir. Son olarak, ürünün dağıtımı ve bakımı yapılır; bu aşama, yazılımın kullanıma sunulması ve kullanım süresi boyunca performansının sürdürülebilirliğinin sağlanması için önemlidir. Thakur vd. (2015), bu aşamaların her yazılım geliştirme yaşam döngüsü modelinde ortak bir yapı sunduğunu vurgulamaktadır. Agarwal vd. (2010) ise yazılım geliştirme yaşam döngüsünün 5 ila 9 aşamaya bölünebileceğini, dolayısıyla en az beş, en fazla dokuz aşamadan oluşması gerektiğini ifade etmiştir. Ortalama olarak yedi veya sekiz aşamaya

sahip olması gerektiğini belirten yazarlar, YGYD aşamalarını proje başlatma ve planlama/ihtiyacın tanınması/ön araştırma, proje tanımlama ve seçimi/fizibilite çalışması, proje analizi, sistem tasarımı, kodlama, test etme, uygulama ve bakım olarak sıralamaktadır.



Şekil 2.2. Yazılım Geliştirme Yaşam Döngüsü

Kaynak: Agarwal vd. (2010); Nizam (2015)

Planlama, yazılım ürününü geliştirmek için tüm unsurların belirlendiği aşamadır (Despa, 2014). Yazılım geliştirme süreçlerinde “ön araştırma” olarak adlandırılan bu aşama, projenin veya sistem geliştirme yaşam döngüsünün başlangıç noktasıdır ve mevcut sistemin temel sorunlarının yanı sıra değişiklik ihtiyacının netleştirilmesine odaklanır. Planlama aşamasında, geliştirilecek sistem hakkında kapsamlı bir araştırma yapılır; bu süreç, proje için sağlam bir temel oluşturmanın yanı sıra, fizibilite çalışması için de gerekli verileri sağlar (Agarwal vd., 2010).

İhtiyaç analizi, yazılım ürününün müşteri tarafından gerçekleştirilmesi beklenen temel görev ve fonksiyonlarını belirleme sürecidir. Bu aşamanın temel amacı, sistemin işlevlerini ve kesin gereksinimlerini açıklığa kavuşturarak bunları belirli bir formatta belgelendirmektir. Bu süreçte, müşterinin doğru gereksinimlerini anlamak ve bunları doğal bir dilde, açık ve anlaşılır bir şekilde belgelemek önceliklidir (Nizam, 2015; Malik, 2017).

Tasarım, yapılan analizin yazılım geliştirme araçlarının imkân, kural ve sınırları dâhilinde yazılımın üst seviye modeline dönüştürülmesidir. Planlama ve ihtiyaç analizine göre tasarımın hazırlandığı aşamadır (Nizam, 2015).

Gerçekleştirme, müşteri talepleri doğrultusunda tasarımı gerçekleştirilen model için kodun yazıldığı ve yazılım uygulamasının fiilen inşa edildiği aşamadır (Nizam, 2015). Bu aşamada amaç, tasarımı mümkün olan en iyi şekilde uygulamaktır. Bu aşama hem test hem de bakım aşamalarını etkilemektedir. İyi yazılmış kod, test ve bakım çabasını azaltabilir. Bu nedenle, kodlama sırasında odak noktası, yalnızca yazılması basit programlar geliştirmek değil, okunması ve anlaşılması kolay programlar geliştirmektir (Agarwal vd., 2010).

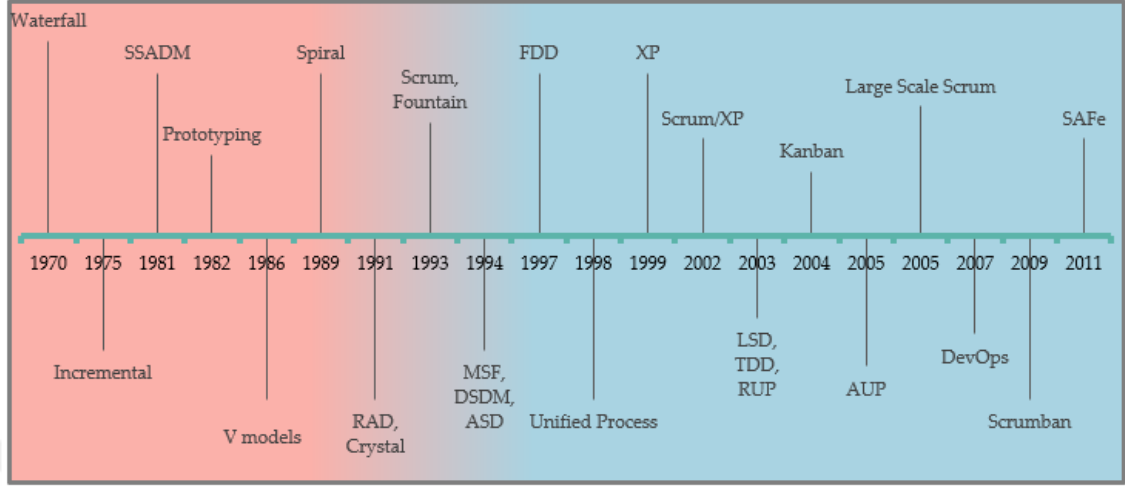
Test, programlama ve tasarım hatalarının belirlenip düzeltildiği aşamadır (Malik, 2017). Hedeflenen ve yapılanlar kontrol edilir. Bu işlem gerçekleştirilen yazılımın düzgün çalışmasını ifade eden doğrulama ve yapılanların istenileni karşılama ifade eden onaylama şeklinde gerçekleştirilir (Nizam, 2015).

Devreye Alma, yazılımın gerçek hayata geçtiği aşamadır. Bu aşamada kullanıcı eğitimi, pilot kurulum ve nihai kurulum yapılarak yazılım çalışır duruma getirilir (Nizam, 2015).

Bakım aşaması, yazılımın kodlama kısmında hataların düzeltilmesi, yazılımın yeteneklerinin güncellemeler yoluyla iyileştirilmesi ve yazılımın kullanım ömrü boyunca güncelliğini yitiren kısımlarının silinmesi işlemlerini içerir (Malik, 2017).

YGYD, sistemlerin oluşturulması veya değiştirilmesi sürecini ifade eden ve bu sürecin yönetiminde kullanılan çeşitli modeller ve metodolojileri kapsayan bir kavramdır. YGYD, birçok yazılım geliştirme metodolojisinin temelini oluşturur (Kute ve Thorat, 2014). Son elli yılda, yazılım geliştirme sürecini organize etmek için çok çeşitli modeller ortaya çıkmıştır. Bir anlamda, modeller farklı koşullar altında başarılı geliştirme için önceden test edilmiş kalıpları temsil etmektedir. Yazılım geliştirme modelleri genellikle benzer hedeflere sahiptir, ancak hedeflerine farklı yollardan ulaşırlar. Bir yazılım süreci modeli kavramı, geliştirme sürecinin belirli aşamalarında kullanılan yaklaşımlara veya araçlara atıfta bulunma eğiliminde olan bir yöntem veya teknikle ilgili fikirden daha geneldir (Deek vd., 2005). Bu modeller, yazılım geliştirme süreçlerini daha sistematik ve

verimli hale getirmek amacıyla, geleneksel yazılım geliştirme yöntemleri ve çevik yöntemler olmak üzere iki ana başlık altında sınıflandırılır.



Şekil 2.3. Yazılım Geliştirme Modellerinin Zaman Çizelgesi

Kaynak: Kirpitsas ve Pachidis (2022)

2.2. GELENEKSEL YAZILIM GELİŞTİRME YÖNTEMLERİ

Geleneksel yazılım geliştirme metodolojileri, yazılım geliştirme sürecinin yapılandırılmış ve belirli bir plana bağlı olarak aşamaların sıralı bir biçimde gerçekleştirildiği yöntemlerdir. Bu metodolojiler, yazılım geliştirme yaşam döngüsü boyunca ayrıntılı dokümantasyona önem verir. Dokümantasyon, işlevsel ve işlevsel olmayan gereksinimler, tasarım, kodlama, testler ve dağıtım planlarına ilişkin detayları kapsamaktadır (Kirpitsas ve Pachidis, 2022).

Geleneksel metodolojiler, çalışmanın başında kapsamlı bir gereksinim kümesinin belirlenmesi ve belgelenmesi ile başlamaktadır. Ardından, mimari ve üst düzey tasarım geliştirme ve inceleme süreçleri izlenir (Javanmard ve Alian, 2015). Bu bağlamda geleneksel yazılım geliştirme, kapsamlı dokümantasyonun önemsendiği, ayrıntılı ve kapsamlı bir planlamaya dayalı olarak yazılımın geliştirildiği, müşterinin geliştirme sürecine dâhil olmadığı ve gereksinim belirleme, tasarım, geliştirme ve test aşamalarının sırasıyla gerçekleştirildiği bir yöntem olarak tanımlanabilir (Awad, 2005; Javanmard ve Alian, 2015; Sommerville, 2015).

Geleneksel yöntemlerde, sürecin başında müşteri gereksinimleri ayrıntılı olarak netleştirilmekte, bu gereksinimlere dayalı bir tasarım oluşturulmakta ve yalnızca büyük

değişiklikler gerektiğinde müdahale edilerek yazılım geliştirilme süreci yürütülmektedir. Geleneksel yazılım geliştirme yöntemlerin belirleyici bir diğer özelliği de yazılım geliştirme döngüsündeki tüm aşamaların ardışık olarak gerçekleştirilmesidir. Bir sonraki aşamaya geçiş yapılabilmesi için önceki aşamalardaki tüm faaliyetlerin eksiksiz tamamlanmış olması gerekmektedir. Genellikle müşteri ile sadece yazılım gereksinimlerinin belirlendiği ilk aşamada temas kurulur; bu nedenle, süreçte daha sonra ortaya çıkabilecek ihtiyaç değişiklikleri göz önünde bulundurulmaz (Başar vd., 2015).

“Ağır” veya “plan güdümlü” modeller olarak da bilinen geleneksel yazılım geliştirme metodolojileri, müşteri gereksinimlerinin başlangıçta net olarak belirlenebildiği projeler için uygundur. Geliştirme süreci boyunca planın titizlikle uygulanmasını gerektiren bu yaklaşımlar, müşteri ihtiyaçlarının proje başında tam olarak belirlenemediği ya da süreç içerisinde değişiklik gerektiren projeler için uygun değildir (Başar vd., 2015; Sommerville, 2015).

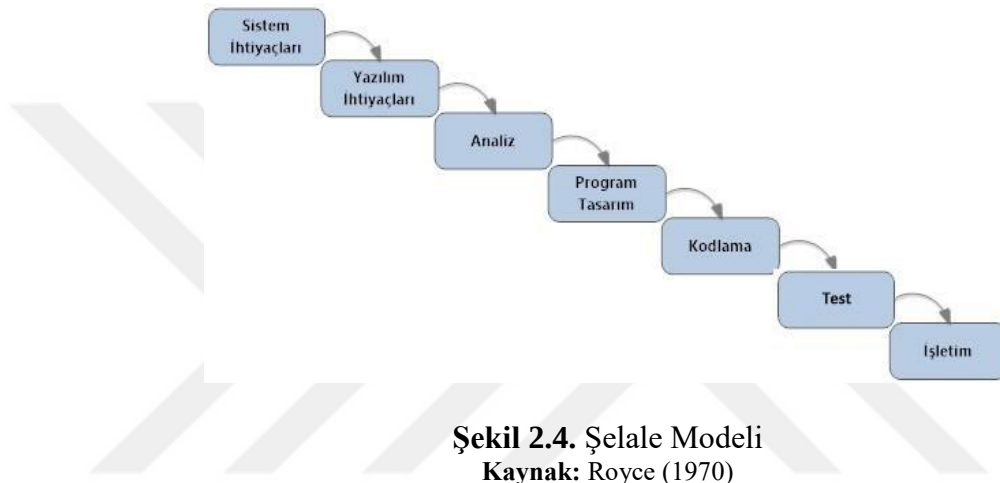
Geleneksel yazılım geliştirme metodolojileri, doğrusal modeller olarak bilinen Şelale ve V modellerinin yanı sıra, yinelemeli geliştirme modelleri olarak Artımlı, Evrimsel Geliştirme ve Spiral (Sarmal) modeli gibi çeşitler içermektedir. Doğrusal modeller, yazılım geliştirme sürecinin planlamadan devreye almaya kadar tüm aşamaların birbirini izlediği ve geri dönüşün sınırlı olduğu bir süreci öngörmektedir. Yinelemeli modeller ise yazılımın bir bütün olarak değil, çeşitli sürümlere bölünerek aşama aşama geliştirilmesini esas almaktadır (Nizam, 2015).

2.2.1. Şelale Modeli (Waterfall Model)

Şelale modeli süreç modellerinin ilk ve en etkili olanlarından biridir. Winston tarafından 1970’lerde ortaya çıkan bu model, sonraki modellerin evriminde önemli bir tarihsel etkiye sahip olmuş ve çoğu yazılım geliştirme standardının temeli haline gelmiştir (Deek vd., 2005). Şelale modeli, yazılım geliştirme sürecini ardışık aşamalar halinde gerçekleştiren klasik bir modeldir. Her aşama, tamamlandığında ve doğru bir şekilde sonuçlandığında, bir sonraki aşamaya geçiş yapılır. Bu model, yazılım geliştirme sürecindeki her aşamanın sonunda planlanan bir kontrol gözden geçirmeyi içerir. Bu gözden geçirme sürecinde elde edilen sonuçlara göre, bir sonraki aşamaya geçme veya gerekirse düzenleme yapma kararı alınır (Royce, 1970). Modelin adının kökeni de her

aşamanın tamamlanmasıyla bir sonraki aşamaya kademeli bir geçişi simgeleyen bir “şelale” görüntüsünden gelmektedir (Kirpitsas ve Pachidis, 2022).

Şelale Modeli, yazılım mühendisliğindeki klasik modellerden biri olarak kabul edilir. Model, özellikle devlet destekli projeler ve birçok büyük şirket tarafından yaygın bir şekilde benimsenmiştir. Modelin erken aşamalarda planlamaya vurgu yapması, tasarım kusurlarını gelişmeden önce tespit etmeyi amaçlar. Ayrıca, yoğun dokümantasyon ve planlama süreçleri, kalite kontrolünün önemli olduğu projelerde etkili bir şekilde çalışmasını sağlar (Kute ve Thorat, 2014).



Şelale Modelinin Başlıca Avantajları:

1. Basit ve anlaşılması kolay bir yapıya sahiptir.
2. Her seviyenin tamamlanmasının ardından bir sonraki aşamaya geçilmesini sağlayan doğrusal bir ilerlemeye sahiptir.
3. Hedeflerin net bir şekilde belirlenmesini sağlar.
4. Ekipler arasındaki iletişimi güçlendiren ayrıntılı bir plan ve dokümantasyon sağlar.

Şelale Modelinin Başlıca Dezavantajları:

1. Analiz ve tasarım aşamalarında dikkate alınmayan değişiklikleri uygulamak zordur.
2. Geliştirme sürecinin başında belirsizliklerin bulunması.
3. Yazılımın teslimi sürecin sonunda gerçekleşir.
4. Test aşaması sürecin sonunda yapılır.
5. Kullanıcılar sürece aktif olarak dâhil edilmez (Kirpitsas ve Pachidis, 2022).

2.2.2. Artımlı Geliştirme Modeli (Incremental Model)

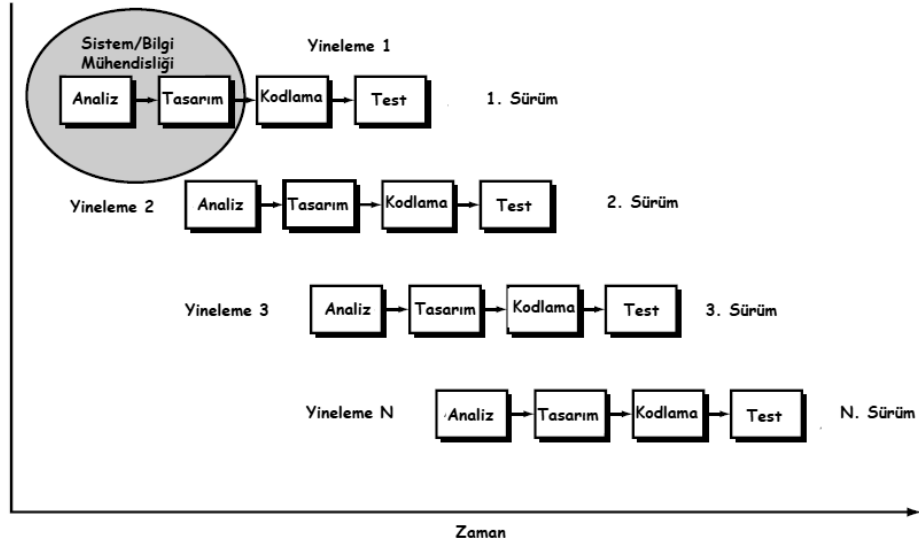
Artımlı ya da yinelemeli model, şelale modelinin tekrarlı olarak uygulandığı gelişmiş halidir. Her tekrarda, yazılımın teslim edilebilir bir “artımını” üretir ve her artışta müşterilere yazılım ürününün daha fazla işlevsel hali sunulur (Kute ve Thorat, 2014; Thakur vd., 2015).

İhtiyaç analizinin büyük ölçüde belirlenmesinden sonra yazılım geliştirme sürecindeki işlemlerin sürümlere bölünerek, her sürümde ihtiyaçların bir kısmının gerçekleştirilmesidir (Nizam, 2015). Artımlı bir model kullanıldığında, ilk artış genellikle bir temel üründür. Temel gereksinimler karşılanır, ancak birçok ek özellik sağlanmadan kalır. Temel ürün, müşteri tarafından kullanılır veya ayrıntılı incelemeye tabi tutulur. Kullanım ve/veya değerlendirme sonucunda bir sonraki artış için plan oluşturulur. Plan, müşterinin ihtiyaçlarını daha iyi karşılamak ve ek özellik ve işlevsellik sağlamak için temel üründe yapılan değişiklikleri ele almaktadır. Bu süreç, her bir artışın tesliminden sonra, ürünün tamamı üretilene kadar tekrarlanır (Agarwal vd., 2010).

Bu yaklaşım, projeyi daha küçük ve yönetilebilir parçalara ayırarak müşteri geri bildirimlerini daha etkili bir şekilde değerlendirme ve projeyi esnek bir şekilde yönetme imkânı sağlar. Artımlı model, yazılım araçları ve tekniklerinin kullanımıyla sonraki yıllarda ortaya çıkan prototipleme modelinin temelini oluşturmuştur (Kirpitsas ve Pachidis, 2022). Artımlı model, tüm sistemi bir kerede inşa etmenin riskli olduğu, ürünün en kısa sürede pazara sunulmasının gerektiği veya yeni bir teknolojinin test edilmesi gerektiği durumlarda kullanılır (Modi vd., 2017).

Artımlı Geliştirme Modelinin Başlıca Avantajları:

1. Her yineleme sonunda işlevsel bir ürün teslim edilir.
2. Yazılım ürününün önemli işlevleri hızlıca kullanıma sunulabilir.
3. Değişen gereksinimlere uyum sağlar.
4. Yinelemeli modelde yazılım erken geliştirilir. Bu durum müşteri değerlendirmesi ve geri bildirim almayı kolaylaştırır.
5. Proje ilerlemesi kolayca ölçülebilir.
6. Bir yinelemede karşılaşılan sorunlar, zorluklar ve riskler bir sonraki yinelemede dikkate alınır (Shylesh, 2017).



Şekil 2.5. Artımlı Geliştirme Modeli
Kaynak: Agarwal vd. (2010)

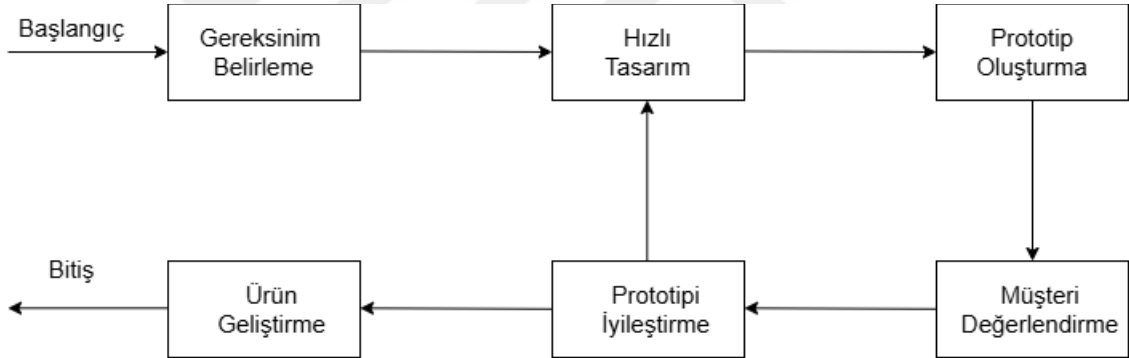
Artımlı Geliştirme Modelinin Başlıca Dezavantajları:

1. Kaynakların uzun süre boyunca tahsis edilmesini gerektirir.
2. Güçlü bir değişiklik yönetim kontrol süreci gerektirir.
3. Hem problemin hem de çözümün iyi anlaşılmış olması gerekir.

2.2.3. Prototip Model (Prototyping Model)

Prototipleme yaklaşımının temelinde, bir tasarım veya kodlama sürecine başlamadan önce gereksinimleri kesinleştirmek yerine, gereksinimleri daha iyi anlamak amacıyla geçici bir prototip oluşturmak fikri yatmaktadır. Bir prototip, mevcut gereksinimler doğrultusunda geliştirilmektedir ve müşteri, prototiple etkileşim kurarak sistemin “gerçek hissini” deneyimleme fırsatı bulmaktadır. Böylece müşteri istenen sistemin gereksinimlerine dair daha somut bir anlayış geliştirebilir. Özellikle, karmaşık ve büyük çaplı sistemlerde veya mevcut bir sistemin bulunmadığı durumlarda prototipleme, gereksinimlerin belirlenmesi sürecine katkı sağlayacak etkili bir yöntem olarak öne çıkar. Prototipler genellikle tamamlanmış bir sistem niteliğinde değildir ve birçok ayrıntıyı içermez; esas amaç, genel işlevselliğe sahip bir sistem sunarak temel gereksinimlerin değerlendirilmesini sağlamaktır (Shah ve Dhiman, 2016).

Prototipleme süreci, geliştirici ve müşterinin bir araya gelip yazılımın genel hedeflerini tanımlaması, mevcut gereksinimleri belirlemesi ve daha fazla tanımlama gerektiren alanları belirlemesiyle başlar. Bu aşamanın ardından hızlı bir tasarım süreci gerçekleşir. Hızlı tasarım, müşteriye veya kullanıcıya görünür olacak yazılım bileşenlerinin (örneğin, giriş yöntemleri ve çıktı formatları) temsil edilmesine odaklanır. Hızlı tasarımın ardından prototipin oluşturulması gelir. Oluşturulan prototip, müşteri veya kullanıcı tarafından değerlendirilerek geliştirilecek yazılımın gereksinimlerinin netleşmesine katkı sağlar. Prototipin, müşterinin ihtiyaçlarını karşılayacak şekilde düzenlenmesiyle yinelemeli bir süreç başlar ve bu süreç aynı zamanda geliştiricinin yapılması gerekenleri daha iyi anlamasına imkân tanır. İdeal olarak, prototip, yazılım gereksinimlerinin belirlenmesinde bir araç işlevi görür. Çalışan bir prototip geliştirildiğinde, geliştirici mevcut program parçalarını kullanmaya çalışır veya hızlı bir şekilde çalışan programlar oluşturmayı sağlayan araçlardan yararlanır (Agarwal vd., 2010).



Şekil 2.6. Prototip Model

Kaynak: Shah ve Dhiman (2016)

Prototip Geliştirme Modelinin Başlıca Avantajları:

1. Geliştirme sürecine aktif kullanıcılar dâhil edilir.
2. Modelde çalışan bir sistem sunulduğundan, kullanıcılar geliştirilen sistem hakkında daha net bir fikir sahibi olur.
3. Yanlış anlamalar çok daha erken aşamada fark edilebilir.
4. Kullanıcı geri bildirimi hızlı bir şekilde alınarak daha iyi çözümler üretilir.
5. İşlevsellikle ilgili sorunlar kolayca tespit edilebilir.

Prototip Geliştirme Modelinin Başlıca Dezavantajları:

1. Model, sistemlerin önce geliştirilip ardından düzeltilmesi gibi bir sürece yol açabilir.
2. Pratikte, sistem kapsamının beklenenden fazla genişlemesi durumunda sistemin karmaşıklığı da artabilir.
3. Eksik bir uygulama, sistemin tamamlanmış haliyle kullanılamamasına neden olabilir.
4. Sorunun eksik veya yetersiz analiz edilmesi riski bulunur (Kute ve Thorat, 2014).

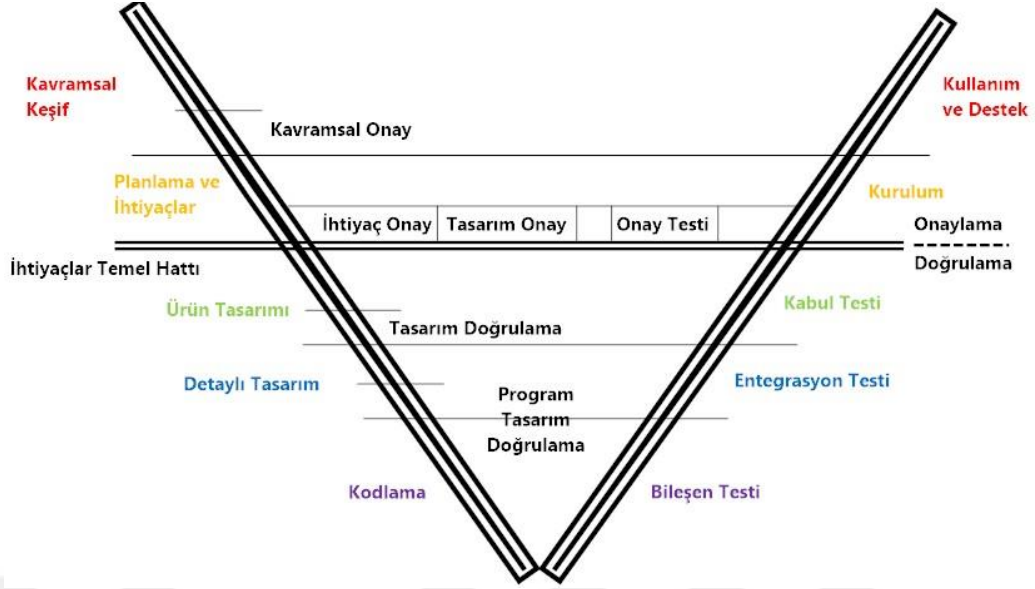
2.2.4. V Modeli (V Model)

V Modeli, şelale modelinin daha organize bir kontrol safhalarına sahip hali olup, doğrulama (verification) ve geçerleme (validation) süreçlerine dayanan bir modeldir (Boehm, 1977). Her aşama kendi kontrol aşamasıyla eşleşerek “V” harfine benzer ve bu yapısından dolayı adını alır (Nizam, 2015). Tıpkı şelale modelinde olduğu gibi, V-model yaşam döngüsünde süreçler sıralı bir şekilde yürütülür ve bir aşama tamamlanmadan sonraki aşamaya geçilmez.

Bu modelde her geliştirme aşamasına karşılık gelen bir test süreci bulunmaktadır ve test süreci geliştirme aşamalarına paralel olarak ilerlemektedir (Kute ve Thorat, 2014). Genellikle, modelin kodlama tasarımı her iki yanda bağlantılıdır; bir tarafta doğrulama (verification) yapılırken diğer tarafta geçerleme (validation) gerçekleştirilir. Bu yapı, test sürecinin modelin her iki tarafında paralel olarak ilerlemesini sağlar. V-model tasarımı Şekil 2.7’de gösterilmektedir (Alazzawi vd., 2023).

V Modelinin Başlıca Avantajları:

1. Her aşama bir önceki tamamlandıktan sonra başladığı için süreci düzenli ve kontrollü bir şekilde ilerletir. Disiplinlidir.
2. Aşamaları sıralıdır.
3. Küçük projeler için mükemmeldir.
4. Anlaması ve kullanması kolaydır.
5. Model yönetimi çok kolaydır.



Şekil 2.7. V Modeli

Kaynak: Boehm (1979)

V Modelinin Başlıca Dezavantajları:

1. Karmaşık ve nesne yönelimli projeler için uygun değildir.
2. Uzun vadeli ve sürekli projeler için uygun değildir.
3. Orta ve yüksek düzeyde değişiklik gerektiren projeler için kullanışlı değildir.
4. Test aşamasına gelindiğinde, işlevlerine geri dönüp değişiklik yapmak mümkün değildir (Alazzawi vd., 2023).

2.2.5. Spiral (Sarmal) Model (Spiral Model)

Spiral modeli, Boehm tarafından geliştirilmiş olup, şelale modelindeki sıralı akışın aksine yazılım geliştirme sürecinin yinelemeli bir yapıda ve aşamalı olarak gerçekleştirilmesine dayanan bir yöntemdir (Boehm, 1988). Bu modelde yazılım, tekrarlayan aşamalardan geçerek kademeli olarak ilerler ve her aşamada risk analizi gibi önemli değerlendirmeler yapılarak projenin daha güvenli ve etkin bir şekilde tamamlanması hedeflenir. Her yinelemede proje, önceki aşamalarda elde edilen bilgiye dayalı olarak genişletilmekte ve derinleştirilmektedir.

Spiral modelin planlama, risk analizi, mühendislik ve değerlendirme olmak üzere dört temel aşaması vardır. Modelin başlangıç aşamasında iş gereksinimleri toplanır ve risk değerlendirmesi yapılır. Risk analizinin odakta olduğu bu aşama, projede

2.3. ÇEVİK YAZILIM GELİŞTİRME YÖNTEMLERİ

Günümüzde küresel ve hızla değişen bir çevrede faaliyet gösteren şirketler yeni fırsatlara, pazarlara ve değişen koşullara hızla adapte olma gerekliliği içerisinde. Neredeyse tüm iş faaliyetlerinin bir parçası haline gelen yazılımlar, yeni iş fırsatlarının değerlendirilmesi, rekabet baskılarıyla başa çıkılması ve değişen koşullara uyum sağlanmasında etkin bir rol oynamaktadır. Bu nedenlerle şirketlerin ihtiyaç duyduğu yazılımların hızlı bir şekilde geliştirilmesi önem taşımaktadır. Ancak geleneksel yöntemlerle geliştirilen yazılımlar, yazılım geliştirme sürecinde değişen gereksinimlere ve teknolojilere yeterince hızlı cevap verememektedir. Plan odaklı birçok geleneksel geliştirme yönteminde projeler, kapsamlı bir gereksinim setinin toplanması ve belgelenmesi ile başlamaktadır. 1990'lı yılların ortalarından itibaren bu yaklaşım zorlayıcı hale gelmiştir. Hızla değişen teknoloji ve iş çevresi, başlangıçta belirlenen gereksinimlerin ve proje planlarının, hatta kısa vadeli projelerde bile geçerliliğini yitirmesine yol açmıştır. Müşteriler de ihtiyaçlarını başlangıçta net bir şekilde tanımlamakta giderek zorlanmıştır (Williams ve Cockburn, 2003; Sommerville, 2015).

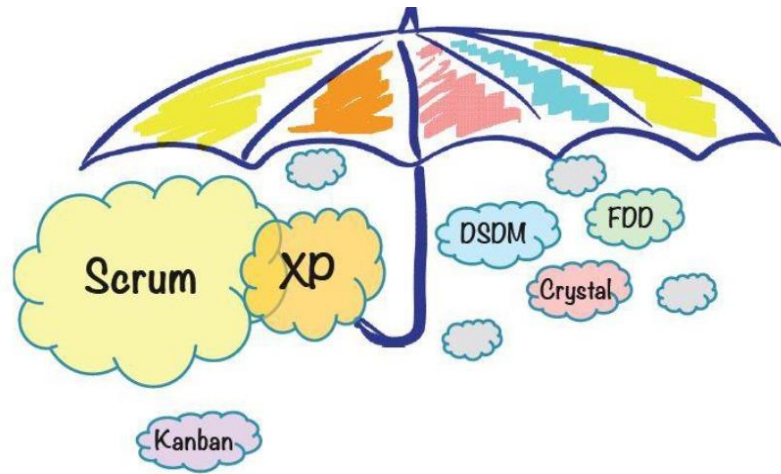
Bu durum karşısında, uygulayıcılar değişim hızını benimseyen metodolojiler ve pratikler geliştirmiştir. Bu metodolojiler, üç farklı kıtada ortaya çıkmıştır: Avrupa'da Dinamik Sistem Geliştirme Metodu, Avustralya'da Özellik Odaklı Geliştirme ve ABD'de Uç Programlama, Kristal, Uyarlanabilir Yazılım Geliştirme ve Scrum. Pratikleri ve felsefeleri temel benzerlikler taşısa da bu metodolojiler bağımsız olarak geliştirilmiştir (Williams ve Cockburn, 2003).

Şubat 2001'de, değişime toleranslı metodolojiler geliştirmiş 17 yazılım mühendisliği danışmanı, o dönemde "hafif yöntemler" olarak adlandırdıkları kendi metodolojilerindeki ortak yönleri tartışmak üzere bağımsız olarak Snowbird, Utah'ta bir araya gelmiştir (Agile Alliance, 2001a; Williams ve Cockburn, 2003; Williams, 2010). Çalışma alışkanlıklarının büyük ölçüde örtüştüğünü fark eden bu grup, süreçlerindeki hafifliğin müşteri memnuniyeti ve yüksek kalite gibi önemli hedeflere ulaşmada bir araç sağladığını kabul etmiştir. Metodolojilerini, esnek üretim uygulamalarında yaklaşık on yıldır kullanılan bir terim olan "çevik" olarak adlandırmışlardır (Agile Alliance, 2001a; Williams ve Cockburn, 2003). "Çevik" terimi, metodolojilerin değişime hızlı ve esnek bir yanıt verme yeteneklerini vurgulamaktadır. Danışmanlar ayrıca Çevik Birliğini (Agile

Alliance) kurarak, çevik yaklaşımın temelinde yatan değeri tanımlayan “Çevik Yazılım Geliştirme Manifestosu” ve “Çevik Manifesto’nun Ardındaki İlkeler”i yayınlamışlardır (Agile Alliance, 2001a; 2001b; 2001c). Çevik Birliği tarafından ilk olarak benimsenen metodolojiler arasında uyarlanabilir yazılım geliştirme, kristal, dinamik sistem geliştirme yöntemi, uç programlama, özellik odaklı geliştirme ve scrum yer almaktadır (Williams, 2010).

Çevik yazılım geliştirme metodolojileri, başlangıç aşamasında tam olarak belirlenmemiş gereksinimlerle yazılım geliştirme ihtiyacına yanıt olarak ortaya çıkmıştır. Çevik metodolojiler, yazılım geliştirme sürecinde ortaya çıkan değişikliklere ve değişen ihtiyaçlara uyum sağlama esnekliği sunmaktadır. Geleneksel, plan odaklı metodolojilere kıyasla daha hafif ve esnek olan çevik metodolojiler, geliştiriciler ile kullanıcılar arasında iş birliğini ön planda tutar ve kapsamlı planlar, ayrıntılı belgeler ve sıkı süreçler gereksinimini azaltmaktadır (Kirpitsas ve Pachidis, 2022).

Kelime olarak “kolaylık ve çabuklukla davranan, tetik, atik, atik tetik” anlamına gelen çevik, yazılımın hızlı ve etkili bir şekilde geliştirilmesine vurgu yapmaktadır (Sarker vd., 2015; TDK, t.y.). Çevik yöntemler, süreç ve belgeleme yerine doğrudan yazılımın kendisine yoğunlaşan, yinelemeli ve artırımı geliştirmeye dayanan tekniklerdir (Project Management Institute [PMI] ve Agile Alliance, 2017). Çevik yöntemler müşteri ile etkileşimi, değişen gereksinimleri benimseyen yöntemlerdir (Abrahamsson vd., 2002). Çevik yöntemler, çeşitli çerçeveleri ve yöntemleri kapsayan şemsiye bir terimdir (PMI ve Agile Alliance, 2017; Arora vd., 2021).



Şekil 2.9. Çevik Yöntemler
Kaynak: VersionOne (2008)

Çevik yöntemler, yazılım geliştirme sürecini küçük artırımlarla ilerletmekte ve her artımda yeni özelliklerin yayımlanmasını, iki veya üç haftalık aralıklarla sistemin müşteriye sunulmasını sağlamaktadır. Bu yöntemler, resmi olmayan iletişim yollarını tercih ederek dokümantasyonu en aza indirerek ve müşterilerin sürece aktif katılımını sağlamaktadır (Sommerville, 2015). Çevik yazılım geliştirme, yaşam döngüsü boyunca değişen gereksinimlere uyum sağlama, müşteri ile geliştirici arasındaki yakın iş birliğini destekleme ve yazılım özelliklerinin erken ve sık teslimini amaçlamaktadır (Babar vd., 2013).

Çeviklik, değerlerle tanımlanan, ilkelerle yönlendirilen ve çeşitli pratikler aracılığıyla ortaya çıkan bir zihniyet olarak ifade edilmektedir. Çevik yaklaşımları benimseyen uygulayıcılar, gereksinimlerine uygun pratikleri seçme esnekliğine sahiptir (Smith ve Sidky, 2009). Günümüzde kullanılan çeşitli çevik yöntemler çevik zihniyet, değerler ve ilkelerle ortak köklere dayanmaktadır (PMI ve Agile Alliance, 2017).



Şekil 2.10. Çevik Değerler, İlkeler ve Uygulamalar Arasındaki İlişki

Kaynak: Smith ve Sidky (2009)

Çevik yaklaşımın temelinde yer alan değerler yayımlanan manifesto ile tanıtılmış ve aşağıdaki şekilde ifade edilmiştir (Agile Alliance, 2001b).

“Bizler daha iyi yazılım geliştirme yollarını uygulayarak ve başkalarının da uygulamasına yardım ederek ortaya çıkartıyoruz. Bu çalışmaların sonucunda;

Süreçler ve araçlardan ziyade bireyler ve etkileşimlere
Kapsamlı dokümantasyondan ziyade çalışan yazılıma
Sözleşme pazarlıklarından ziyade müşteri ile işbirliğine
Bir plana bağlı kalmaktan ziyade değişime karşılık vermeye
 değer vermeye kanaat getirdik.

Özetle, sol taraftaki maddelerin değerini kabul etmekle birlikte, sağ taraftaki maddeleri daha değerli bulmaktayız.”

Çevik yaklaşımın temelini oluşturan dört değere ilişkin ayrıntılı inceleme aşağıda yer almaktadır (Williams, 2010).

Süreçler ve araçlardan ziyade bireyler ve etkileşimlere: Bu ilke, yazılım geliştirme süreçlerinin aşırı biçimlendirilmesinin ve esnek olmamasının, insan faktörünü ve pratik uygulamaları olumsuz etkileyebileceğini ve böylece başarı olasılığını düşürebileceğini ifade etmektedir. Çevik metodolojilerin ortak özelliği, rolleri yerine getiren bireylerin önemine vurgu yapmalarıdır. Süreçler ve araçlardan ziyade, projelerin başarısında en etkili unsurun bu bireyler olduğu kabul edilmektedir.

Kapsamlı dokümantasyondan ziyade çalışan yazılıma: Çevik yöntemlerde, takımın ilerlemesi çalışan yazılım ile değerlendirilir. Çevik yaklaşımlar ile ekip, müşterinin deneyimleyip geri bildirim sağlayabileceği kısmi bir sistemi hızla sunar. Geliştirme sürecini desteklemek ve müşterinin ihtiyaç duyduğu belgeleri sağlamak amacıyla yalnızca “yeterli” düzeyde dokümantasyon üretilir.

Sözleşme pazarlıklarından ziyade müşteri ile iş birliğine: Projenin başında gereksinimlerin sözleşmeyle belirlenmesi durumunda, iki temel varsayım yapılır: (1) geliştirme ekibinin daha fazla açıklamaya gerek olmaksızın bu gereksinimleri yerine getirebileceği ve (2) müşterinin proje başında ürünü net bir şekilde tanımlayabileceği. Çevik ekipler ise, yazılım yaşam döngüsü boyunca çalışan yazılımın gelişimi ile müşteri gerçek gereksinimlerini keşfettikçe “karşılıklı uyum” içinde çalışır.

Bir plana bağlı kalmaktan ziyade değişime yanıt vermeye: Çevik yöntemler, tüm yazılım yaşam döngüsü için kapsamlı bir plan hazırlamak yerine, genel bir sürüm planı ile her yinelemenin başında oluşturulan kısa vadeli yineleme planlarını içerir. Bu planlar, bir önceki yinelemeden bu yana gerçekleşen değişiklikleri, yeni keşifleri ve öncelikleri

yansıtabilir. Müşteriler, yazılımı “uygulamada” gördüklerinde taleplerini değiştirme eğiliminde olabilirler.

“Çevik Manifesto’nun Ardındaki İlkeler” ise şunlardır (Agile Alliance, 2001c):

1. En önemli önceliğimiz değerli yazılımın erken ve devamlı teslimini sağlayarak müşterileri memnun etmektir.
2. Değişen gereksinimler yazılım sürecinin son aşamalarında bile kabul edilmelidir. Çevik süreçler değişimi müşterinin rekabet avantajı için kullanır.
3. Çalışan yazılım, tercihen kısa zaman aralıkları belirlenerek birkaç haftada ya da birkaç ayda bir düzenli olarak müşteriye sunulmalıdır.
4. İş süreçlerinin sahipleri ve yazılımcılar proje boyunca her gün birlikte çalışmalıdırlar.
5. Projelerin temelinde motive olmuş bireyler yer almalıdır. Onlara ihtiyaçları olan ortam ve destek sağlanmalı, işi başaracakları konusunda güven duyulmalıdır.
6. Bir yazılım takımında bilgi alışverişinin en verimli ve etkin yöntemi yüz yüze iletişimidir.
7. Çalışan yazılım ilerlemenin birincil ölçüsüdür.
8. Çevik süreçler sürdürülebilir geliştirmeyi teşvik etmektedir. Sponsorlar, yazılımcılar ve kullanıcılar sabit tempoyu sürekli devam ettirebilmelidir.
9. Teknik mükemmeliyet ve iyi tasarım konusundaki sürekli özen çevikliği artırır.
10. Sadelik, yapılmasına gerek olmayan işlerin mümkün olduğunca arttırılması sanatı, olmazsa olmazlardandır.
11. En iyi mimariler, gereksinimler ve tasarımlar kendi kendini örgütleyen takımlardan ortaya çıkar.
12. Takım, düzenli aralıklarla nasıl daha etkili ve verimli olabileceğinin üzerinde düşünür ve davranışlarını buna göre ayarlar ve düzenler.

Çevik yazılım geliştirme yöntemleri, değişime uyum sağlamayı önceliklendirerek geliştirme süresini ve maliyetlerini azaltmayı amaçlayan çevik ilkeler doğrultusunda şekillenmektedir (Anwer vd., 2017). Bu yöntemler, belirli temel pratikler aracılığıyla çevik felsefenin temel değer ve prensiplerini benimsemektedir (Abrahamsson vd., 2002; Smith ve Sidky, 2009; PMI ve Agile Alliance, 2017). Çevik yöntemler arasında Uç

Programlama, Scrum, Kanban, Test Odaklı Geliştirme, Dinamik Sistem Geliştirme Modeli, Özellik Odaklı Geliştirme ve Kristal gibi çeşitli yaklaşımlar bulunmaktadır. Araştırmanın devamında çevik yöntemlere ilişkin bilgiler başlıklar halinde verilecektir. Yazılım geliştirmede yaygın kullanılan uç programlama, scrum ve kanban yöntemleri hakkında daha kapsamlı bilgilere yer verilirken; diğer çevik yöntemlere de değinilecektir.

2.3.1. Uç Programlama (Extreme Programming- XP)

Uç programlama (UP), 1990'ların sonunda Kent Beck, Ron Jeffries ve Ward Cunningham tarafından Chrysler şirketi için hazırlanan bir proje sonrasında ortaya çıkan yinelemeli ve artırımı bir çevik yöntemdir (Beck, 1999; Sommerville, 2015). Belirsiz ve değişen gereksinimlerle karşı karşıya kalan yazılım ekiplerinin ihtiyaçlarını karşılamak için geliştirilen UP, en basit kodlama çözümleri aracılığıyla yazılım kalitesini ve değişen müşteri gereksinimlerine yanıt verme yeteneğini iyileştirmeyi amaçlayan programlama odaklı bir yazılım geliştirme yöntemidir (Newkirk, 2002; Aitken ve Ilango, 2013).

Uç programlama, geliştirme sürecini yönlendiren beş temel değeri benimsemektedir. Bunlar; iletişim, sadelik, geri bildirim, cesaret ve saygıdır (Beck ve Andres, 2004).

İletişim, programcıların müşteriler ve diğer programcılarla sürekli etkileşim kurmalarını sağlar.

Sadelik, tasarımların anlaşılır ve sade olmasını ifade eder.

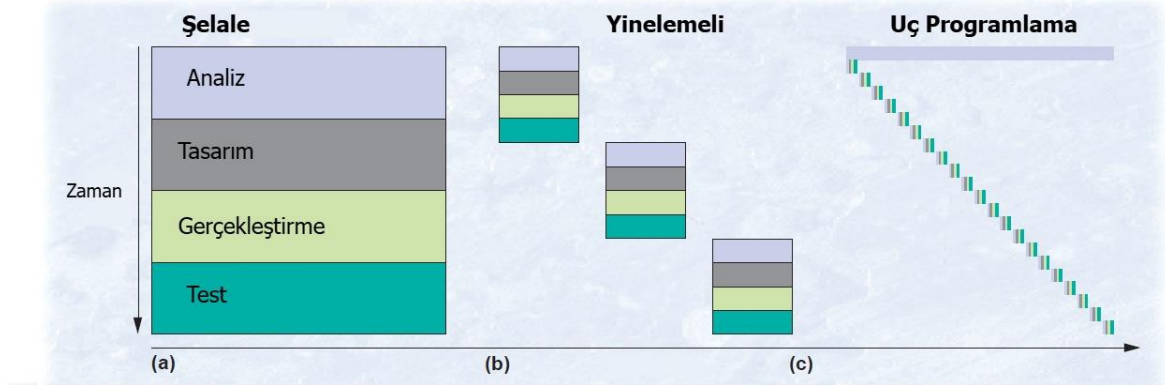
Geri bildirim, yazılımın ilk günden itibaren test edilmesiyle elde edilir; sistem mümkün olduğunca erken müşterilere teslim edilerek önerilen değişiklikler uygulanır.

Cesaret, programcıların değişen gereksinimlere ve teknolojiye cesaretle yanıt verebilmelerini sağlar.

Saygı, ekip üyelerinin birbirlerine ve projeye değer vermelerini gerektirir. Takımdaki herkesin bu değerlere odaklanması, uyumlu bir çalışma ortamı oluşturur ve ekibin kişisel tercihler yerine başarıya ulaşmasını sağlar (Beck ve Andres, 2004).

Uç Programlama, geleneksel yazılım geliştirme sürecinden farklı olarak, gelecekteki ihtiyaçlara yönelik planlama, analiz, tasarım, uygulama ve test aşamalarını tek seferde gerçekleştirmek yerine, tüm bu faaliyetleri “her bir seferde azar azar” olacak

şekilde yinelemeli ve artırımı geliştirme yapmaktadır (Beck, 1999). Şekil 2.11’de şelale modeli (a), yinelemeli geliştirme döngülerine sahip geliştirme modeli (örneğin spiral model) (b) ve uç programlama yazılım geliştirme süreci (c) yer almaktadır.



Şekil 2.11. Şelale, Yinelemeli ve Uç Programlama Yazılım Geliştirme Süreci
Kaynak: Beck (1999)

Uç Programlama, yazılım geliştirme süreci boyunca tüm bu faaliyetlerin azar azar gerçekleştirilmesini başarıyla sağlamak amacıyla çeşitli pratikler içermektedir (Beck ve Andres, 2004). Kent Beck, mevcut pratikleri uç noktalara kadar zorladığı için bu yöntemi “Uç Programlama” olarak adlandırmıştır (Conboy ve Fitzgerald, 2010; Sommerville, 2015).

UP’nin pratikleri, UP ekiplerinin günlük olarak yaptığı faaliyetlerdir. Pratikler, kendi başlarına anlam ifade etmez. Ancak değerlerle bütünleştirildiklerinde etkili ve anlamlı hale gelirler. Örneğin, eşli programlamanın yalnızca “yapılması gereken bir görev” olarak uygulanması, gerçek bir katkı sağlamaz ve verimsiz bir çabaya dönüşür. Ancak iletişimi güçlendirmek, geri bildirim almak, sistemi sadeleştirmek, hataları tespit etmek ve cesareti artırmak amacıyla eşli programlama yapılması, bu pratiğin değer odaklı bir yaklaşımla bütünleşmesini sağlayarak anlamlı hale getirir (Beck ve Andres, 2004). Uç programlama, 12 temel pratikle uygulanmaktadır (Newkirk, 2002). Pratikler ve pratiklere ilişkin tanımlamalar şu şekildedir (Beck, 1999; Abrahamsson vd., 2002; Newkirk, 2002; Beck ve Andres, 2004):

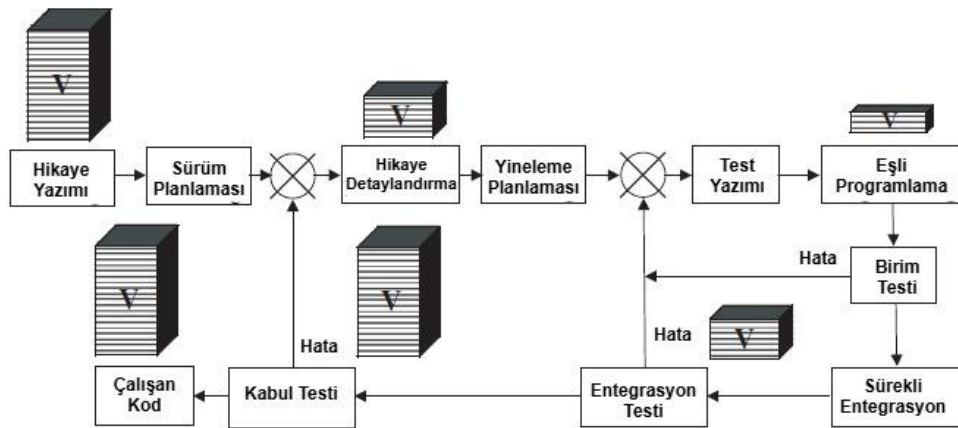
1. *Planlama Oyunu:* Müşteriler, programcılar tarafından sağlanan tahminlere göre sürümlerin kapsamını ve zamanlamasını belirler. Programcılar, yalnızca yinelemede hikayeler tarafından talep edilen işlevselliği uygular.

2. *Kısa Sürümler:* Sistem, tüm problemin çözülmesini beklemeden birkaç ay içinde üretime alınır. Yeni sürümler sık sık yapılır; günlükten aylığa kadar değişebilir. Gerçek yazılım üzerinde geri bildirim almak için kritik bir faktördür.
3. *Metafor:* Sistemin yapısı, müşteri ve programcılar arasında paylaşılan bir metafor veya metaforlar seti ile tanımlanır. Sistem özelliklerinin tanımı için “hikayeler” kullanılır. Her hikâye, genellikle bir kart üzerine yazılacak kadar kısa olan tek bir özelliği tanımlar. Metafor ve hikayeler, projenin gereksinimlerini oluşturur.
4. *Basit Tasarım:* Vurgu, o anda uygulanabilir olan en basit çözümü tasarlamaktır. Gereksiz karmaşıklık ve fazla kod hemen kaldırılır.
5. *Test:* Yazılım geliştirme, test odaklıdır. Testler, programcıların yazdığı birim testler ve müşterilerin yazdığı kabul testlerini içerir. Testler, tamamlanmanın göstergesidir.
6. *Yeniden Düzenleme:* Programcılar, mevcut yazılımın davranışını değiştirmeden tasarımını iyileştirmekle sorumludur. Yeniden düzenleme, programcılarının günlük faaliyetlerinin bir parçasıdır.
7. *Eşli Programlama:* Tüm üretim kodu, tek bir ekran/klavye/fare ile iki kişi tarafından yazılır.
8. *Ortak Sahiplik:* Her programcı, sistemin herhangi bir yerinde bir kodu iyileştirme fırsatı bulduğunda, herhangi bir kodu geliştirebilir.
9. *Sürekli Entegrasyon:* Yeni kod, en fazla birkaç saat içinde mevcut sistemle entegre edilir. Entegrasyon sırasında sistem sıfırdan derlenir ve tüm testlerin geçmesi gerekir; aksi halde yapılan değişiklikler iptal edilir.
10. *Haftada 40 saat:* Maksimum 40 saatlik çalışma haftası uygulanır. Arka arkaya iki hafta fazla mesai yapılmasına izin verilmez. Bu durum oluşursa, çözülmesi gereken bir problem olarak ele alınır.
11. *Yerinde Müşteri:* Müşteri, ekipte tam zamanlı olarak bulunur ve soruları yanıtlamakla görevlidir. Müşteri ayrıca kabul testlerini yazmaktan da sorumludur.
12. *Kodlama Standartları:* İletişim, temel bir değerdir. Kodlama standartlarının benimsenmesi, kodun sınıftan sınıfa tutarlı olması nedeniyle iletişimi geliştirir.

Uç programlama yazılım geliştirme süreci müşteri ve geliştirme ekibinin iş birliğiyle başlar. İlk olarak, müşteri gereksinimlerini yansıtan kullanıcı hikayeleri oluşturulur ve bu hikayeler öncelik sırasına göre düzenlenir. Müşteri, ekiple birlikte yineleme (1-2 hafta) ve sürüm (1-2 ay) planlamalarını yapar ve ilk yineleme için ele alınacak hikayeler seçilir. Bu hikayeler, ekibin her görevi gerçekleştirmek için gerekli iş gücünü ve kaynakları planlayabilmesi amacıyla daha küçük görevlere ayrılır.

Geliştirme ekibi, seçilen hikayeler doğrultusunda kodlamaya geçer. Kodlama süreci, UP'nin temel prensiplerinden olan eşli programlama ve test odaklı geliştirme ile yürütülür. Eşli programlama yöntemi ile iki programcı aynı bilgisayar üzerinde birlikte çalışarak kod yazarken, sürekli entegrasyon sayesinde kod her güncellemeden sonra tüm sistem ile entegre edilerek test edilir. Yineleme sürecinin sonunda, müşteriye çalışan bir sistem sunulur ve müşteri sistemin mevcut durumunu değerlendirerek ekibe geri bildirimde bulunur. Bu geri bildirimler doğrultusunda bir sonraki yineleme süreci planlanır.

Süreç içinde müşteri yeni gereksinimler belirlerse, bu gereksinimlere yönelik yeni hikayeler oluşturulur ve bu hikayeler de öncelik sırasına göre değerlendirilir. Yeni bir gereksinim yoksa mevcut hikayelerden en yüksek öncelikli olanlar seçilerek bir sonraki yinelemede geliştirilir. İlk sürümün sonunda, kullanıma hazır hale gelen yazılım müşteriye sunulur. Eğer daha fazla sürüm planlanmışsa, süreç aynı aşamalar izlenerek devam edecektir (Anderson, 2004; Sommerville, 2015). Uç programlama yazılım geliştirme süreci Şekil 2.12'te yer almaktadır.



Şekil 2.12. Uç Programlama Süreci

Kaynak: Anderson (2004)

Uç programlama, geliştirme sürecini yönlendiren beş temel değeri benimsemektedir. Bunlar; iletişim, sadelik, geri bildirim, cesaret ve saygıdır (Beck ve Andres, 2004).

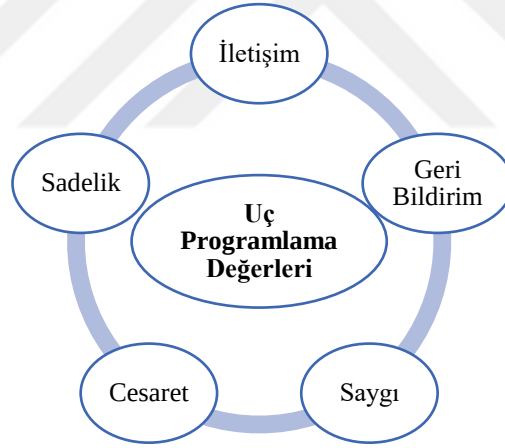
İletişim, programcılarının müşteriler ve diğer programcılarla sürekli etkileşim kurmalarını sağlar.

Sadelik, tasarımların anlaşılır ve sade olmasını ifade eder.

Geri bildirim, yazılımın ilk günden itibaren test edilmesiyle elde edilir; sistem mümkün olduğunca erken müşterilere teslim edilerek önerilen değişiklikler uygulanır.

Cesaret, programcılarının değişen gereksinimlere ve teknolojiye cesaretle yanıt verebilmelerini sağlar.

Saygı, ekip üyelerinin birbirlerine ve projeye değer vermelerini gerektirir. Takımdaki herkesin bu değerlere odaklanması, uyumlu bir çalışma ortamı oluşturur ve ekibin kişisel tercihler yerine başarıya ulaşmasını sağlar (Beck ve Andres, 2004).



Şekil 2.13. Uç Programlama Değerleri

Kaynak: Beck ve Andres (2004)

Uç programlama sürecinde, yazılım geliştirme ekibinin etkin çalışmasını sağlamak için belirli roller tanımlanmıştır (Williams, 2010). Uç programlama bulunan roller;

Yönetici, ekibi ve sorunlarını sahiplenir, ekibi kurar, kaynak sağlar, insanları ve problemleri yönetir ve dış gruplarla iletişim kurar.

Koç, ekibe UP süreci hakkında rehberlik eder, sorunlar olduğunda müdahale eder ve UP sürecinin takip edilip edilmediğini denetler. Koç genellikle bir programcıdır, yönetici değildir.

İzleyici, kullanıcı hikayelerinin ve kabul testi ilerlemelerinin durumunu düzenli olarak geliştiricilerden toplayarak görünür duvar grafikleri oluşturur. Bu kişi programcıdır, ancak yönetici veya müşteri değildir.

Programcı, test, tasarım ve kod yazar; kodu yeniden düzenler, görevleri ve hikayeleri belirler ve tahmin eder.

Test Uzmanı, müşterilere test yazma ve geliştirme konusunda yardımcı olur ve bu kişi aynı zamanda bir programcı olabilir.

Müşteri, hikayeleri ve kabul testlerini yazar; sürüm ve yineleme için hikayeleri seçer. Müşteri rolünün yalnızca müşteri organizasyonundan bir kişi tarafından oynanması gerektiği yönünde yaygın bir yanlış anlama olsa da bir müşteri grubu veya ürün yöneticisi gibi bir müşteri vekili de bu rolü üstlenebilir.

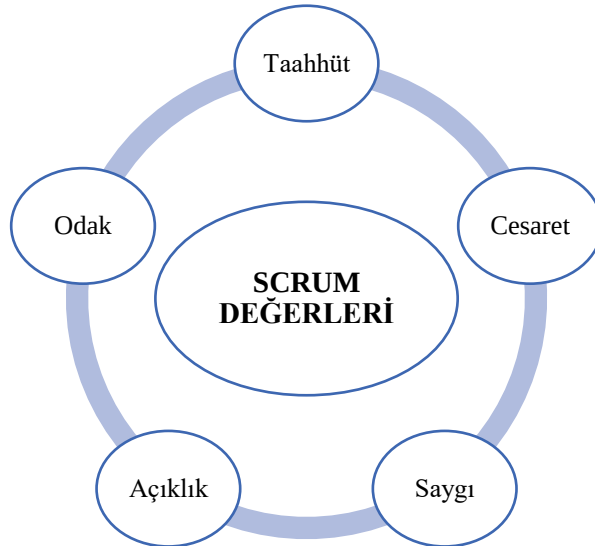
2.3.2. Scrum

Scrum, karmaşık yazılım sistemlerinin geliştirilmesinde en yüksek değere sahip ürünleri üretken ve yaratıcı bir şekilde sunmayı amaçlayan, yinelemeli ve artırıcı bir yönetim yaklaşımıdır (Baytam ve Kapısız, 2011; Salman vd., 2018; Schwaber ve Sutherland, 2020). Scrum, sürekli değişen bir ortamda esnek bir yazılım geliştirmek için ekip üyelerinin nasıl çalışması gerektiğine odaklanan bir proje yönetim süreci çerçevesi olarak tanımlanır (Aitken ve Ilango, 2013). Scrum, uç programlama gibi diğer çevik yöntemlere de uygulanabilen bir yazılım yönetim metodudur. Belirli bir yazılım geliştirme yaşam döngüsü yöntemine bağlı olmadığı için diğer yazılım geliştirme yaşam döngüsü yöntemleri için bir yönetim çerçevesi olarak kullanılabilir. Dolayısıyla Scrum, yazılımın geliştirilme şekliyle değil nasıl yönetileceğiyle ilgilenir (Anderson, 2004).

Scrum, şimdiye kadar en popüler çevik yöntem olarak kabul edilir ve ampirik süreç kontrol teorisine dayanır. İlk olarak 1986 yılında Hirotaka Takeuchi ve Ikujiro Nonaka tarafından otomobil ve tüketici ürünleri sektörlerinde yeni ürün geliştirme standardı olarak tanıtılmıştır (Takeuchi ve Nonaka, 1986). Scrum, adını her oyuncunun belirli bir

rol üstlendiği ve takım üyelerinin hızlıca stratejiler geliştirdiği rugby oyunundaki terminolojiden almaktadır. Bu bağlamda geliştirme ekibi, ortak bir amaca yönelik tek bir birim olarak kabul edilir (Schwaber, 1997; Zayat ve Senvar, 2020). 1993 yılında ise Ken Schwaber ve Dr. Jeff Sutherland tarafından yazılım geliştirme süreci için resmileştirilmiştir (Aitken ve Ilango, 2013). Scrum, ekiplerin yeni ürünleri mümkün olan en hızlı şekilde teslim etmelerini sağlamak için sürekli iyileştirme ve değişikliklere hızlı uyum sağlayan prensipler ve uygulamaları içeren bir çevik çerçevedir (Brezočník ve Majer, 2016). Günümüzde Yahoo, Microsoft, Google, Motorola, SAP ve Cisco gibi büyük ve küçük ölçekli şirketler tarafından kullanılmaktadır (Deemer vd., 2010).

Scrum taahhüt, cesaret, odak, açıklık ve saygı değerlerini içermektedir. Taahhüt, Scrum takımındaki her bir üyenin takım hedefini gerçekleştirmeyi üstlenmesi anlamına gelir. Cesaret, doğru olanı yapma ve zorlu sorunlar üzerinde çalışma kararlılığını ifade eder. Odak, takım üyelerinin sprint işine ve Scrum takımının hedeflerine yönelmesini sağlar. Açıklık, Scrum takımı ve paydaşlarının yapılacak işler ve karşılaşılan zorluklar hakkında açık olmayı kabul etmesini gerektirir. Saygı ise, takımın her bir üyesinin diğer üyeleri yetkin ve bağımsız bireyler olarak görüp değer vermesi anlamına gelir. Scrum uygulamalarının başarısının temelinde, Scrum takımının bu değerleri ustalıkla yaşatma becerisi yatmaktadır (Schwaber ve Sutherland, 2020).



Şekil 2.14. Scrum Değerleri
Kaynak: Schwaber ve Sutherland (2020)

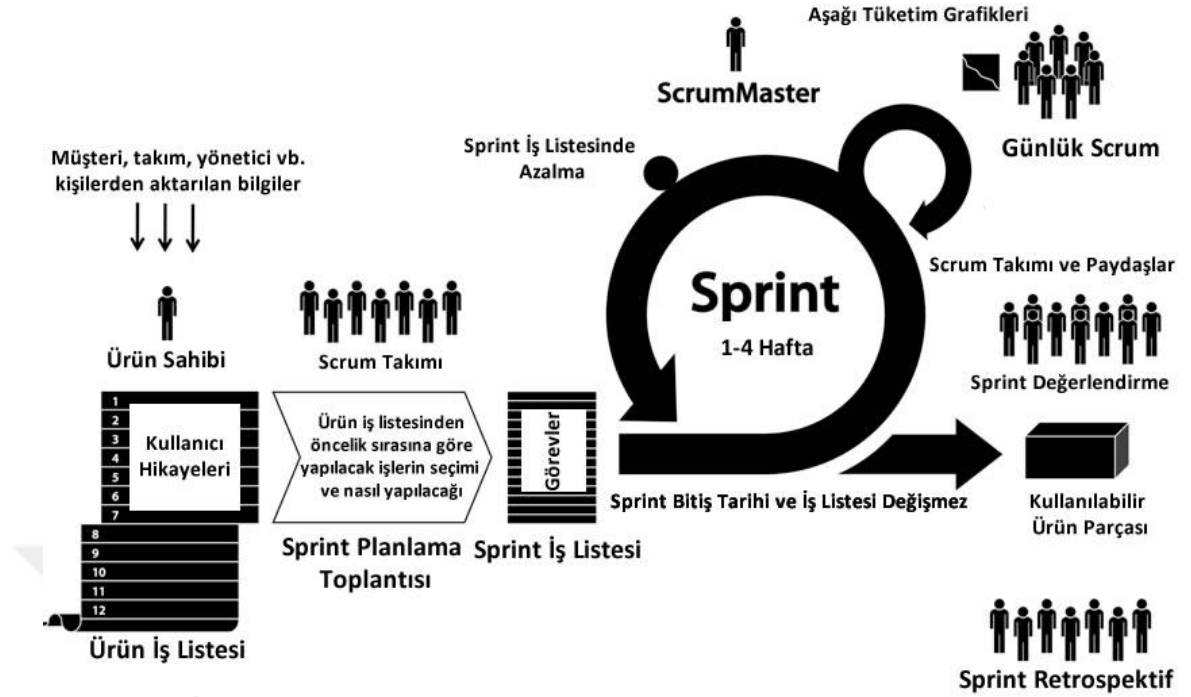
Scrum içerisinde tanımlı roller, etkinlikler ve eserler barındırmaktadır. Scrum’da ürün sahibi, geliştirme takımı ve scrum master olmak üzere üç rol bulunmaktadır (Sverrisdottir vd., 2014). Bir projedeki tüm yönetim sorumlulukları bu üç role bölünmüştür.

Ürün sahibi (Product Owner), projeye ve projenin sonucunda ortaya çıkan sisteme ilgi duyan herkesin çıkarlarını temsil etmekten sorumludur. Ürün sahibi, projenin başlangıç gereksinimlerini, yatırım getirisi hedeflerini ve sürüm planlarını oluşturarak proje için başlangıçta ve süresince fon sağlanmasını sağlar. Gereksinimlerin listesi ürün iş listesi olarak adlandırılır. Ürün sahibi, ürün iş listesini sıklıkla önceliklendirerek en değerli işlevlerin önce üretilmesine ve geliştirilmesine katkı sağlamaktadır. Ürün sahibi tek bir kişi olmalıdır.

Geliştirme takımı (Development Team), işlevselliği geliştirmekten sorumludur. Geliştirme takımı kendi kendini yönetir, kendi kendini organize eder ve çok işlevlidir. Ürün iş listesini bir yineleme içinde bir işlev artışına dönüştürmenin yollarını bulmaktan ve kendi işlerini yönetmekten sorumludurlar. Takım üyeleri, her bir yinelemenin ve projenin bir bütün olarak başarısından kolektif olarak sorumludur.

Scrum Master, Scrum sürecinden, projeye katılan herkese Scrum’ı öğretmekten, Scrum’ı organizasyonun kültürüne uyacak şekilde uygulamaktan ve beklenen faydaları sağlamaktan, ayrıca herkesin Scrum kurallarına ve pratiklerine uymasını sağlamaktan sorumludur (Schwaber, 2004; Schwaber ve Sutherland, 2020).

Scrum metodolojisinin kalbi sprint’tir (Webb, 2015; Yitmen ve Gürbüz, 2018). Sprint, sabit uzunluktaki bir yinelemeli geliştirme olup geliştirme sürecinin temel birimini oluşturur (Başar vd. 2015; Brezočnik ve Majer, 2016; Lei vd., 2017). Sprint bazen döngü, yineleme veya artış olarak adlandırılabilir ve amacı teorik olarak dağıtılabilir bir ürün ortaya koymaktır (Webb, 2015). Sprint süresi, scrum takımına bağlı olarak 1 ila 4 hafta arasında belirlenir. Yazılım geliştirme sürecinde sprint süresi sabittir, yani zaman kutulu (time-boxed) bir yapıya sahiptir (Başar vd. 2015; Lei vd., 2017). Aynı zamanda sprint içerisinde yapılacak işlerde herhangi bir değişiklik yapılamaz. Sprint’in sonunda kullanılabilir ve potansiyel olarak yayınlanabilir bir ürün parçası çıkmaktadır (Sutherland ve Schwaber, 2011; Yitmen ve Gürbüz, 2018). Şekil 2.15’te Scrum içerisinde tanımlı roller, etkinlikler ve eserlerin yer aldığı Scrum çerçevesi göstermektedir.



Şekil 2.15. Scrum Süreci
Kaynak: Sutherland ve Schwaber (2011)

Scrum, döngüsel ve yinelemeli bir geliştirme süreci olarak, belirli adımlar ve bileşenler aracılığıyla yazılım geliştirme projelerine sistematik bir yaklaşım sunmaktadır. Sürecin başlangıcında ürün sahibi müşteri, takım ve yönetim gibi çeşitli paydaşlardan gelen gereksinimleri toplar ve bu gereksinimleri ürün iş listesine ekler. Ürün iş listesi (product backlog), yapılması gereken işlerin önceliklendirilmiş bir şekilde yer aldığı kullanıcı hikayelerinden oluşur ve sürekli olarak güncellenir (Schwaber ve Sutherland, 2020).

Her bir Sprint'in başında, sprint planlama toplantısı düzenlenir. Bu toplantıda, scrum takımı, ürün iş listesinden öncelik sırasına göre yapılacak işleri seçer ve bu işlerin gerçekleştirilmesi için izlenecek yöntemi belirler. Sprint boyunca tamamlanması planlanan işlerin listesi olan sprint iş listesi (sprint backlog), bu aşamada oluşturulur. Sprint'in süresi, Scrum takımının yapısına ve projenin gereksinimlerine göre 1 ila 4 hafta arasında sabitlenir. Bu süre, Scrum'un "time-boxed" yani belirli bir zaman aralığına sıkıştırılmış yapısını korur ve Sprint boyunca sprint iş listesi'ne yeni bir iş eklenmesi veya listede değişiklik yapılması engellenir (Cole ve Scotcher, 2015; Schwaber ve Sutherland, 2020).

Sprint süresince, Scrum takımı her gün günlük scrum toplantıları düzenleyerek Sprint'in ilerleyişini değerlendirir ve karşılaşılan engelleri ele alır. Günlük scrum, takımın kendi içinde güçlü bir iletişim kurmasını ve Sprint süresince işlerin güncel durumunu düzenli olarak gözden geçirmesini sağlar. Ayrıca, ilerlemenin görselleştirilmesi amacıyla aşağı tüketim grafikleri (iş ilerleme grafikleri) kullanılır; bu grafikler, takımın işlerin zamanında tamamlanma durumunu izlemesine olanak tanır ve takımın iş yükünü değerlendirmesine katkıda bulunur (Sutherland ve Schwaber, 2011; Webb, 2015).

Sprint'in sonunda, Scrum takımı ve paydaşlar, yapılan çalışmaları gözden geçirmek amacıyla sprint değerlendirme (sprint review) toplantısı gerçekleştirir. Bu değerlendirme toplantısında, takım, Sprint süresince üretilen kullanılabilir ürün parçasını ürün sahibi ve diğer paydaşlara sunar. Bu ürün parçası, belirli bir kullanıcı hikayesini karşılayacak düzeyde tamamlanmış, test edilmiş ve dağıtımına hazır hale getirilmiştir. Sprint değerlendirme toplantısı, takımın projenin mevcut durumunu değerlendirmesi ve paydaşlardan geri bildirim alması için kritik bir fırsattır (Yitmen ve Gürbüz, 2018; Deemer vd., 2010).

Son olarak, her Sprint'in bitiminde, sprint retrospektif toplantısı düzenlenir. Bu toplantı, takımın Sprint süresince yaşanan süreçleri değerlendirerek nelerin iyi gittiğini, hangi alanlarda iyileştirmelere ihtiyaç duyulduğunu ve gelecekteki Sprint'ler için yapılması gereken değişiklikleri ele almasını sağlar. Retrospektif, Scrum'un sürekli iyileştirme ilkesinin bir parçasıdır ve Scrum sürecinin etkinliğini artırmayı hedefler (Aitken ve Ilango, 2013; Brezočnik ve Majer, 2016).

Sprint döngüsü, her sprintten önce, sırasında ve sonrasında gerçekleşen aşağıdaki etkinliklerden oluşmaktadır (Baytam ve Kapısız, 2011; Schwaber, 2014; Webb, 2015; Sutherland ve Schwaber, 2011; Schwaber ve Sutherland, 2020). Bunlar;

Sprint planlama, sprintten önce gerçekleştirilen toplantının amacı, kullanıcı hikayelerini geliştirmek ve sprint sonunda hangi hikayelerin teslim edileceği konusunda anlaşmaya varmaktır. İlgili sprintte yapılacak işlerin planlandığı tüm scrum takımıyla birlikte yapıldığı toplantıdır. Ürün sahibinin üründe ihtiyaç duyulan her şeyin sıralandığı kullanıcı hikayelerinde oluşan ürün iş listesinden ilgili sprintte kullanılacak sprint iş listesi seçilerek işlerin nasıl başarılacağı ve sprint sonunda teslim edilecek ürün parçası üzerine planlama yapılmaktadır.

Günlük scrum, bazen ayakta toplantı olarak da adlandırılır. Sprint süresi boyunca her günün başında aynı yer ve saatte scrum master ve geliştirme takımı üyeleri arasında gerçekleştirilen 15 dakikalık kısa süreli ayakta gerçekleştirilen toplantıdır. Toplantının amacı, Scrum’ın günlük olarak “incele ve uyum sağla” ilkesini uygulamaktır. Küçük problemler, mümkün olan en erken aşamada öğrenilir ve ele alınır, çözüm sprint boyunca yapılacak tüm gelecekteki görevlere dahil edilir.

Sprint değerlendirme, sprintin sonunda ortaya çıkan ürün parçasını görüp kontrol etmek ve gerekli olması durumunda ürün iş listesini uyarlamak için gerçekleştirilen scrum takımı ve paydaşlar arasında gerçekleştirilen toplantıdır. Amaç, ürün sahibi ve paydaşlardan, inceleme ve uyum sağlama sürecine katkı sağlamak amacıyla geri bildirim almaktır. Sprintten elde edilen teslim edilebilir ürünler, sprint değerlendirme toplantısından alınan geri bildirimlere dayalı olarak gelecekteki sprintlerde sürekli iyileştirmeye tabi tutulur.

Sprint retrospektif ise sprint değerlendirme toplantısı sonrasında scrum takımının kendini gözlemlemesi ve gözlemlerden yola çıkarak bir sonraki sprint için yapacakları iyileştirmeler için bir plan oluşturmalarını sağlar. Bu toplantının odak noktası teslim edilebilir ürün değil, takımın teslim edilebilir ürünü oluşturma sürecinin nasıl iyileştirilebileceğini belirlemektir.

Scrum, gözlem ve adaptasyon imkânı sağlamak için yapılan işi veya üretilen değeri ifade eden eserlere sahiptir (Schwaber ve Sutherland, 2020). Ancak Scrum eserlerini açıklamadan önce Scrum’ın bir parçası olan kullanıcı hikayeleri tanımlanacak ve ardından Scrum’ın eserleri olan ürün iş listesi, sprint iş listesi ve ürün parçası tanımlarına yer verilecektir.

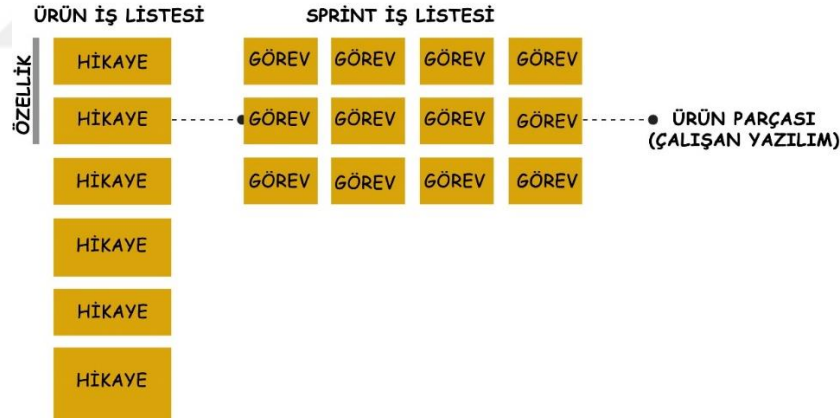
Kullanıcı hikayesi, Scrum ortamındaki tüm etkinlikler bir veya daha fazla kullanıcı hikayesi (bazen sadece “hikâye” olarak adlandırılır) ile başlar. Hikayeler ürünü tanımlar ve projenin varlık nedenini oluşturur. Her kullanıcı hikayesi, geliştirme ekibi için bir veya daha fazla belirli teslim edilebilir öğeyi tanımlar. Bir kullanıcı hikayesi, belirli bir görev türüdür ve şu şekilde ifade edilir: Bir [kullanıcı türü] olarak, [bir şey yapmak] istiyorum ki böylece [bir değer yaratılır]. Kullanıcılar her zaman bir tür paydaştır ya ürün sahibi ya da müşteridir.

Örneğin, bir fotoğraf yazılımı programı için bir müşteri aşağıdaki kullanıcı hikayesini belirtebilir: “Bir müşteri olarak, zaman ve stresi azaltmak için bir fotoğrafın web için boyutunu ve çözünürlüğünü tek bir tıklamayla ayarlamak istiyorum.” (Webb, 2015)

Ürün iş listesi, ürün ile ilgili yapılması gereken her şeyin yer aldığı tek listedir. Listede ürünün özellikleri, işlevleri, iyileştirmeleri ve düzenlemeleri yer almaktadır. Sürekli gelişen ve değişen dinamik bir yapıda olan ürün iş listesi, ürün sahibi sorumluluğunda oluşturulmakta ve kontrol edilmektedir. Listede ayrıca, her bir iş kaleminin önceliği, büyüklüğü, tahmini maliyeti ve değeri yer almaktadır (Schwaber ve Sutherland, 2020).

Sprint iş listesi, belirlenen sprint hedefine ulaşılması için tamamlanması gereken ürün iş listesinden önceliklere göre belirlenmiş hikayeleri ve sprint planlamada bu işlerin yapılması için gereken tüm görevleri içerir.

Ürün parçası, bir sprint sonunda tamamlanan sprint iş listesi sonucu müşteri için anlamlı ve fonksiyonel olan yazılıma denmektedir (Schwaber ve Sutherland, 2020).



Şekil 2.16. Ürün İş Listesi, Sprint İş Listesi ve Ürün Parçası
Kaynak: Yazar tarafından oluşturulmuştur.

2.3.3. Kanban

Kanban, çevik yazılım geliştirmede yalın bir yaklaşımı benimseyen ve Japonca’da “görsel kart – sinyal kart” anlamına gelen bir yöntemdir. İlk olarak 1940’larda Toyota’da, üretim aşamalarında iş ve envanteri etkin şekilde kontrol ederek “tam zamanında” üretimi desteklemek amacıyla Taiichi Ohno tarafından geliştirilmiştir (Jyothi ve Rao, 2012;

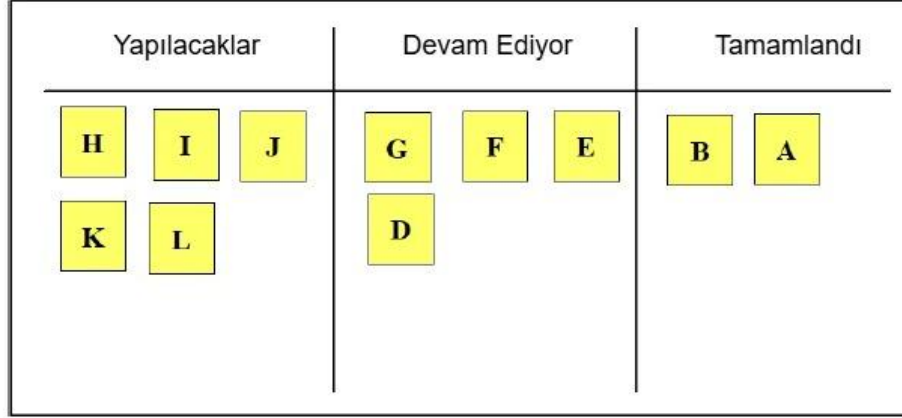
Webb, 2015). Kanban tekniđi, üretim sürecinin bir parçası olarak ihtiyaç duyulduğunda malzemeyi üretim alanına çekmek için “çekme” yöntemini kullanır. Böylece üretim sürecinin çeşitli aşamalarında mal arzını düzenler ve fazla stoklanmayı önler (Mahnič, 2013).

David Anderson, 2004 yılında Kanban sistemini ve yalın üretim felsefesini yazılım geliştirme süreçlerinde işlerin senkronizasyonu ve üretkenliđin artırılması amacıyla uyarlamıştır. Anderson, Kanban’ı “organizasyonlar için kademeli, evrimsel süreç ve sistem deđişimi yaklaşımı” olarak tanımlamıştır (Anderson, 2010). Kanban, üretkenliđi artırmak, teslim süresini kısaltmak ve israfı azaltmak için yazılım geliştirmeye yalın üretim uygulamaları getirmiştir. Kanban’ın ana odađı, yapılması gereken işleri ve bu işlerin zamanlamasını dođru belirleyerek iş akışını ve teslim süresini yönetmektir (Lei vd., 2017). Kanban’ın temel bileşenleri iş akışının görselleştirilmesi, üzerinde çalışılan işlerin sınırlandırılması ve teslim süresinin ölçülmesidir (Bougroun vd., 2014). Bu durum iş akışının anlaşılmasını, görünürlüğünü ve kontrolünü arttırarak yazılım projelerinin geliştirme sürecinde oluşabilecek darboğazları tespit etmesine yardımcı olabilir (Anderson, 2010; Alaidaros vd., 2021). Bu yöntem, sistem sürecindeki sorunları ortaya çıkarmak ve sistemi sürekli olarak iyileştirmek için iş birliđini teşvik etmek amacıyla çekme sistemini temel yaklaşım olarak kullanır (Anderson, 2010).

Kanban yönteminde iş akışını görselleştirmek, iş akışını yönetmek ve proje ilerlemesini izlemek için kanban tahtası (kanban board) kullanılır. Kanban tahtası, geliştirme sürecindeki faaliyetleri göstererek ve devam eden işleri kontrol altında tutarak proje takibini sağlar (Anderson, 2010). Ayrıca kanban tahtası geliştiricilerin belirli birkaç göreve odaklanabilmesine olanak tanır. Böylece, kanban tahtasında görevler arasında geçiş yapma sürecinden kaynaklanan kaynak ve zaman israfı azaltılır (Dennehy ve Conboy, 2017). Kanban tahtası, geliştirme sürecinin takip edilebilmesi açısından kanban yönteminin temel unsurlarından biridir (Alaidaros vd., 2021).

Bir kanban tahtası dikey yönde birkaç aşamaya ayrılmıştır. Her aşama bir görevin durumunu gösterir. Temel bir kanban tahtasında soldan sađa “yapılacaklar”, “devam ediyor” ve “tamamlandı” olmak üzere üç aşama bulunur (Nakazawa ve Tanaka, 2016). Basit bir kanban tahtası Şekil 2.17’de yer almaktadır. Şekil 2.17’de yer alan “Yapılacaklar” sütunu, henüz başlanmamış tüm görevlerin listelendiđi alandır. Genellikle

görevler, geliş sırasına göre sıralanmakla birlikte, bazen önceliklere göre de düzenlenebilir. “Devam Ediyor” sütunu, şu anda üzerinde çalışılan görevlerin bulunduğu bölümdür. “Tamamlandı” sütunu ise tamamlanmış görevleri gösterir (Zayat ve Senvar, 2020).



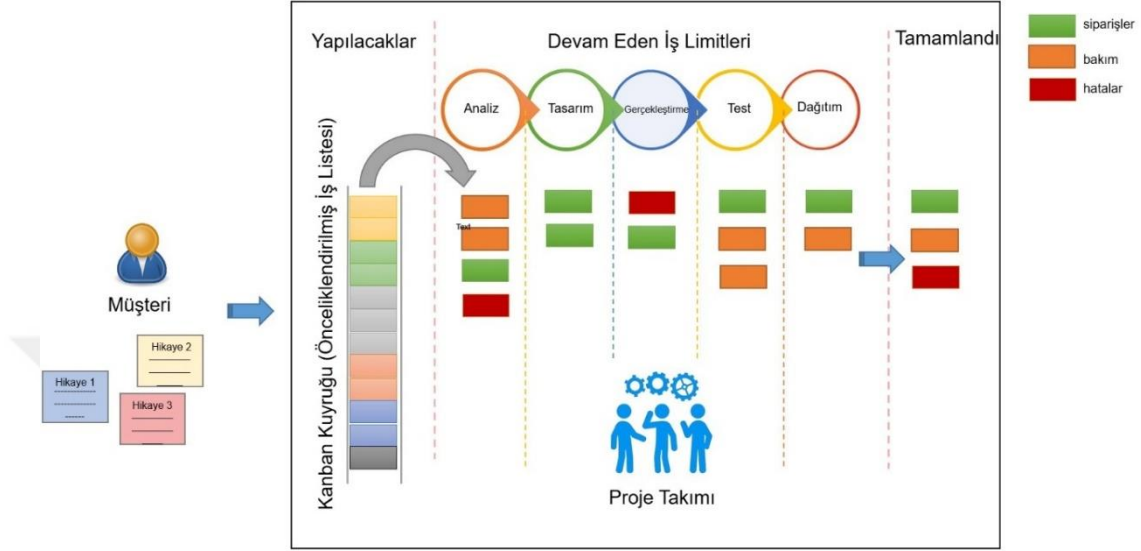
Şekil 2.17. Temel Kanban Tahtası

Kaynak: Alaidaros vd. (2021)

Ayrıntılı bir kanban tahtasında, “yapılacaklar,” “analiz,” “tasarım” ve “dağıtım” gibi çeşitli aşamalar yer almaktadır. Her bir görev, bir kart ile temsil edilir ve görevin mevcut durumuna göre uygun aşamaya yerleştirilir. Görevin durumu değiştikçe kart, Kanban tahtasında soldan sağa doğru ilerler. Ayrıca kanban yöntemi, her aşamada devam eden iş miktarını sınırlayan bir işlev tanımlar. Devam eden iş limiti sayesinde, geliştirme sürecinin akışı kontrol edilir ve kaynak israfı olmaksızın sürekli bir iyileşme sağlanır. Bu yaklaşımla Kanban yöntemi, geliştirme sürecinin genel akışını ve görevlerin durumunu görselleştirir ve sürecin sürekli iyileştirilmesine katkıda bulunur (Nakazawa ve Tanaka, 2016). Şekil 2.18’de kanban süreci ve ayrıntılı kanban tahtası yer almaktadır.

Kanban süreci, müşteri taleplerinin hikayeler (görevler) şeklinde oluşturularak kanban kuyruğuna (önceliklendirilmiş iş listesi) eklenmesiyle başlar. Bu aşamadan itibaren, her görev belirli bir iş akışı boyunca ilerler ve “analiz,” “tasarım,” “gerçekleştirme,” “test” ve “dağıtım” gibi aşamalardan geçer. Her bir aşama, işlerin durumu hakkında görsel bilgi sağlar ve devam eden işlerin sınırlarını belirleyerek süreçte oluşabilecek darboğazları tanımlamaya yardımcı olur. Görevler, belirli renklerle kategorize edilmiştir: yeşil siparişleri, turuncu bakım işlemlerini ve kırmızı hataları temsil etmektedir. Her bir görev kartı, sürecin bir aşamasından diğerine taşınarak tamamlanma

aşamasına ulaşır. Bu sistem, iş akışının izlenebilirliğini artırır ve tüm proje ekibinin işlerin durumunu kolayca gözlemlemesine olanak tanır. Kanban yönteminin bu yapılandırması, işlerin verimli bir şekilde ilerlemesini sağlayarak kaynak israfını azaltır ve proje yönetiminde iyileştirmelere olanak tanır (Zayat ve Senvar, 2020).



Şekil 2.18. Kanban Süreci ve Ayrıntılı Kanban Tahtası

Kaynak: Zayat ve Senvar (2020)

Kanban, scrum gibi bir süreç yönetim aracıdır ancak daha az kuralla ekibe esneklik tanır ve böylece verimliliği artırmayı hedeflemektedir. Scrum yönteminden farklı olarak, Kanban zaman kutulu değildir; anlık talepler ve değişiklikler işleme alınabilir. Ayrıca Kanban’da Scrum’daki gibi belirlenmiş roller (ürün sahibi, geliştirme takımı, Scrum master) ve sabit süreli döngüler bulunmamaktadır (Kniberg ve Skarin, 2010).

Kanban’ı uygulamak için temel olarak beş önemli ilke bulunmaktadır (Anderson, 2010). Bunlar işi baştan sona görsel olarak temsil eden bir akış oluşturmak, herhangi bir anda devam eden iş miktarına sınır koymak ve nihayetinde akışın verimliliğini ölçüp iyileştirmeye yönelik adımlar atmaktır (Cole ve Scotcher, 2015).

İş Akışını Görselleştirme: Yapılacak işten tamamlanmış işe kadar iş akışının görsel bir temsilini oluşturmaktır. Çoğu kişi, araya yalnızca “devam eden işler” adımı eklemeyi tercih ederken (Şekil 2.17), bazıları iş akışını planlama, tasarım, geliştirme, test ve dağıtım gibi prosedürel aşamalara böler; yapılacaklar ve tamamlananlar uçlarda (Şekil 2.18) yer almaktadır (Cole ve Scotcher, 2015). Kanban, iş öğelerinin (genellikle kullanıcı hikayeleri) bir kanban tahtasında sunulmasını gerektirir. Bu, işin geliştirme

sürecinin çeşitli aşamalarından nasıl aktığını gösteren görsel bir kontrol mekanizması görevi görür. Genellikle, yapışkan notlu bir beyaz tahta veya bir elektronik kart duvar sistemi kullanılır. Kanban tahtası, bir iş ögesinin geliştirme işlemi sırasında bulunabileceği çeşitli durumları temsil eden bir sütun dizisinden oluşur. İş geliştirme yaşam döngüsü boyunca ilerledikçe, kartlar son sütunda bitene kadar bir durumdan diğerine geçer. Her sütunun üstünde, tek seferde karşılık gelen iş akışı durumunda kaç kart olabileceğini gösteren bir devam eden iş sınırı vardır. Bir kart bir sütunda tamamlandığında bir sonrakine geçer, böylece mevcut sütununda açık bir alan oluşturur, bu da geliştirme ekibinin tamamlanmış bir kartı önceki bir sütundan almasını sağlar. Herhangi bir nedenle, bir sütundaki kartlar tamamlanamaz ve ilerletilemezse, bu sütun er ya da geç çalışılan iş sayısı sınırına ulaşır, bu da geliştirme ekibinden tamamlanmamış bir sürü işi biriktirmek yerine darboğazı düzeltmesini ister (Mahnič, 2014).

Devam Eden İş Limitleri: Aynı anda çok fazla şey yapmaya çalışmak hem bireyler hem de ekipler için başarısızlığa yol açan kanıtlanmış bir yoldur. Kanban, herhangi bir zamanda yapılmasına izin verilen iş sayısını (devam eden iş) sınırlandırarak optimum verimliliği sağlar (Cole ve Scotcher, 2015). Kanban tahtasında her iş akışı durumunda (yani sütun) kaç ögenin (görev) olabileceğinin açık bir sınırıdır (Sienkiewicz, 2012). Kanban'ın temel amacı, devam eden iş miktarını sınırlayarak iş akışını en üst düzeye çıkarmak ve teslim süresini (yani bir ögeyi tamamlamak için ortalama süre) kısaltmaktır (Mahnič, 2014).

İş Akışını Yönetme: İş teslim süresinin yönetimini ifade etmektedir. Teslim süresi, iş ögesinin başlatıldığı tarihten tamamlanıp müşteri tarafından kabul edildiği ilan edilene kadar geçen toplam süreyi ifade eder. Teslim süresi, teslimatı tahmin etmek ve hizmet düzeyinde taahhütlerde bulunmak için kullanışlıdır (Mahnič, 2013). İş teslim süresi Şekil 2.18'de örneği bulunan kanban tahtasında analiz ile başlayıp kullanıcı kabul sürecine (tamamlandı sütunu) kadar uzanan sütunları içermektedir. Tüm süreçte israfi önlemek için tüm akışı mümkün olduğunca kısa / küçük olacak şekilde ayarlamaya çalışmaktır (Sienkiewicz, 2012).

Süreci Açık Hale Getirme: İşin nasıl yapıldığına dair net bir açıklama, herhangi bir nesnel değerlendirme için gereklidir. Ortak bir anlayış olduğunda, sorunları tarafsızca tartışmak ve iyileştirmeler konusunda fikir birliğine varmak daha kolaydır. Her adımın

sonunda doğal bir kontrol noktası bulunmalı ve bir sonrakine geçiş için net kurallar olmalıdır (Cole ve Scotcher, 2015). Tüm paydaşların süreç politikaları hakkında ortak bir anlayışa sahip olduğundan ve süreci, sorunları ve önerilen iyileştirmeleri ortak bir şekilde anlayabildiklerinden emin olmaktır (Sienkiewicz, 2012).

İş birliği içinde iyileştirme: Kanban yöntemi, iyileştirmeler oluşturmak için nicel bilimsel yaklaşımı benimser. Bu yöntem, Kısıtlar Teorisi’ni dikkate alarak israfi yönetmeye ve akışı kontrol etmeye yönelik modellere odaklanır (Alaidaros, 2021). Tüm paydaşlar süreci sürekli ve aşamalı olarak sürdürmeli ve iyileştirmelidir. İş, iş akışı ve süreçteki sürekli, artımlı ve evrimsel değişiklikleri uygulamak için bilimsel bir yaklaşım kullanmaktır (Sienkiewicz, 2012).

2.3.4. Özellik Odaklı Geliştirme (Feature Driven Development)

Özellik Odaklı Geliştirme (ÖOG), Jeff Luca, Peter Coad ve Stephan Palmer tarafından, Coad Yöntemi’nin bir evrimi olarak 1997 yılında geliştirilmiş, yinelemeli ve artımlı bir yazılım geliştirme yöntemidir. Model odaklı ve iş gereksinimlerine dayalı olan bu yaklaşım, süreç boyunca kaliteyi vurgulamakta ve yönetim seviyelerine anlamlı, zamanında ve doğru ilerleme raporları sunmaktadır (Aitken ve Ilango, 2013). ÖOG, yazılım geliştirmede doğru yazılımı, doğru ve zamanında oluşturma sorununu ele alır (Palmer ve Felsing, 2002). Yetenek, deneyim, geçmiş ve kültür açısından çeşitlilik gösteren büyük geliştirme ekiplerinin karmaşık yazılım sistemleri inşa etmesine yönelik bir yaklaşım olarak öne çıkan ÖOG, projede altı temel rol tanımlar: proje yöneticisi, baş mimar, geliştirme yöneticisi, baş programcı ve sınıf sahibi. Bu temel roller, ÖOG süreçleriyle birleşerek projede görevlerin ve sorumlulukların etkili bir şekilde dağıtılmasını sağlamaktadır. Ayrıca, ÖOG dolaylı olarak çeşitli destekleyici ve ek roller de önermektedir (Palmer ve Felsing, 2002).

ÖOG, genel model geliştirme, özellik listesi oluşturma, özelliklere göre planlama, özellik tasarımı ve özellik odaklı geliştirme olmak üzere beş süreçten oluşur. Bu süreçler, “Giriş-Görevler-Doğrulama-Çıkış” (Entry-Tasks-Verification-Exit, ETVX) adı verilen basit bir şablona göre tasarlanır (Palmer ve Felsing, 2002; Thakur ve Singh, 2014). İlk üç süreç, projenin başlangıcında gerçekleştirilir ve sistemin genel yapısını oluşturur. Son iki süreç ise, gereksinimlerdeki ve iş ihtiyaçlarındaki değişikliklere hızla adapte olabilen,

çevik geliştirmeyi destekleyen yinelemeli süreçlerdir (Award, 2005). Yineleme süreleri en fazla iki hafta olarak belirlenmiştir (Palmer ve Felsing, 2002).



Şekil 2.19. Özellik Odaklı Geliştirme Süreci

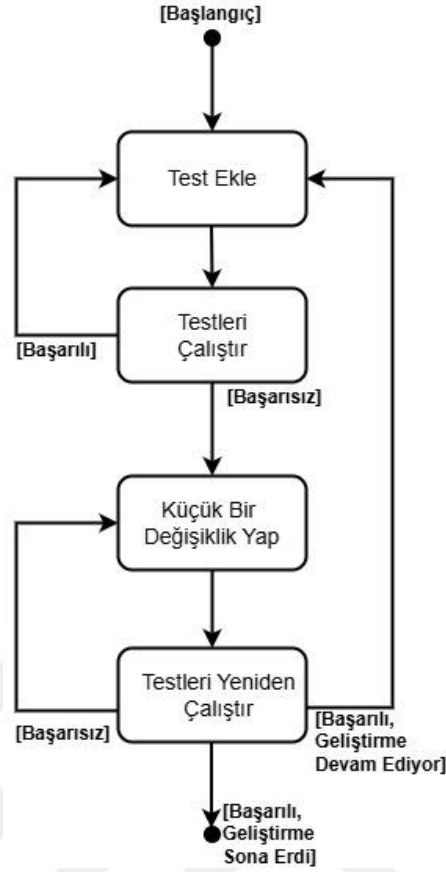
Kaynak: Palmer ve Felsing (2002)

2.3.5. Test Odaklı Geliştirme (Test Driven Development)

Test Odaklı Geliştirme (TOG), yazılımı küçük yinelemelerle geliştiren, önce otomatik test programları oluşturmaya, ardından bu testleri geçebilecek kodu yazmaya ve daha sonra kodun iyileştirilmesine (yeniden kod düzenleme) dayanan bir çevik yazılım geliştirme süreç modelidir (Anwer vd., 2017). TOG, geleneksel yazılım geliştirme yöntemlerinin aksine, kod tamamlandıktan sonra test yapmaktansa önce testi gerçekleştirme prensibine sahiptir (Al-Saqqa vd., 2020). TOG’nin amacı hata oranını azaltmak ve kod kalitesini artırmaktır. TOG, gereksinimleri karşılamak için hızlı geri bildirim olarak temiz kod yazmayı sağlar ve kodun daha öz hâle gelmesi için yeniden kod düzenlemesine imkân sağlamaktadır (Anwer vd., 2017).

TOG, bir yazılım geliştirme modeli olarak ilk kez 2002 yılında Kent Beck’in “Test Driven Development: By Example” adlı kitabında tanıtılmıştır. Uç programlama sürecinde kullanılan “yeniden (kod) düzenleme” ile birlikte “önce test” geliştirme tekniği, TOG süreç modelinin temelini oluşturmaktadır (Beck, 2002).

TOG ile geliştirici, uygulama kodunu yazmadan önce yeni işlevsellik için otomatik birim testleri oluşturur. Bu testler genellikle derlenemez durumda olduğu için, geliştirici testleri geçecek uygulama kodunu yazar. Geliştirici birkaç test senaryosu yazar ve kodu uygular; ardından birkaç test senaryosu daha yazar ve kodu uygular; bu döngüyü aynı şekilde tekrarlar. Bu süreç, geliştiricinin sürekli küçük tasarım ve uygulama kararları almasına olanak tanıyarak işlevselliğin tutarlı bir hızda artmasını sağlar. Yeni işlevsellik, yalnızca yeni birim testleri ve kod tabanı için yazılmış diğer tüm birim testleri başarıyla geçtiğinde tam olarak uygulanmış kabul edilir (Maximilien ve Williams, 2003). Test odaklı geliştirme süreci Şekil 2.20’de gösterilmiştir (Anwer vd., 2017).



Şekil 2.20. Test Odaklı Geliştirme Süreci
Kaynak: Anwer vd. (2017)

Test odaklı geliştirme süreci aşağıdaki şekilde özetlenebilir (Beck, 2002)

1. Küçük bir işlevsellik için yeni bir test durumu ekleyin.
2. Tüm test durumlarını çalıştırın ve yeni testin başarısız olup olmadığını kontrol edin.
3. Testi geçecek kodu yazın.
4. Tüm testleri tekrar çalıştırarak tüm testlerin başarılı olup olmadığını kontrol edin.
5. Tekrarları ortadan kaldırmak için kodu yeniden düzenleyin.

2.3.6. Yalın Yazılım Geliştirme (Lean Software Development)

Yalın yazılım geliştirme (YYG), Toyota ürün geliştirme sistemi ilkelerinin yazılım geliştirmeye uygulanmasıdır. 2003 yılında Poppendieck ve Poppendieck tarafından tanıtılan YYG, hızlı ve mümkün olan en düşük maliyetle yüksek kaliteli yazılımlar

üretmek amacıyla geliştirilmiştir (Poppendieck ve Poppendieck, 2003). YYG, israfi azaltmaya ve maksimum kazancı elde etmek için tüm süreci optimize etmeye odaklanan yinelemeli bir yöntemdir (Flora ve Chande, 2014).

YYG’de israfi ortadan kaldırmak, en temel yalın ilkedir. Müşteriye değer yaratmayan her şey israf olarak ifade edilmektedir. Diğer tüm ilkeler bu ilkeye dayanır. Bu nedenle, yalın geliştirmeyi uygulamanın ilk adımı, israfi görmeyi öğrenmektir. İkinci adım, en büyük israf kaynaklarını ortaya çıkarıp bunları ortadan kaldırmaktır. Bir sonraki adım, kalan en büyük israf kaynaklarını bulup onları ortadan kaldırmaktır. Ardından bu adımları tekrar etmektir. Bir süre sonra, temel gibi görünen şeyler bile yavaş yavaş ortadan kaldırılabilir (Poppendieck ve Poppendieck, 2003).

YYG, yedi temel ilkeyi teşvik etmektedir. Bu ilkeler; (Poppendieck, 2007; Flora ve Chande, 2014)

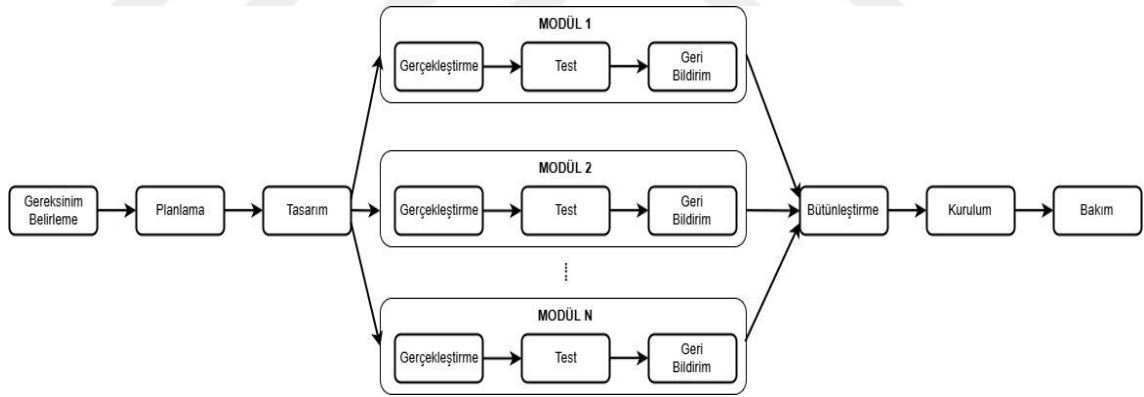
1. *İsrafi Ortadan Kaldırın:* Müşteriye değer katmayan her şeyin ortadan kaldırılmasıdır.
2. *Kaliteyi Oluşturun:* Tüm süreç boyunca varsayımların doğrulanmasıdır. Değeri olmayan bir metrik veya uygulama varsa bırakılmasıdır.
3. *Bilgi Oluşturun:* Kısa yinelemeli döngüler kullanarak doğru şeylere odaklanıldığından emin olmak için hızlı, sürekli geri bildirim sağlanmasıdır.
4. *Taahhüdü Erteleyin:* Yeterli bilgi edinilene kadar karar verilmemesi; sorunun ve potansiyel çözümlerin avantajlarının sağlam bir şekilde anlaşılması gerekliliğidir.
5. *Hızlı Teslim:* Bir iş sorununu tanımlama ve bunu ele alan bir sistem veya özellik teslim etme süresinin en aza indirilmesidir.
6. *İnsanlara Saygı Gösterin:* Ekip üyelerine saygı göstermenin başarıyı getirmesidir.
7. *Bütünü Optimize Edin:* Çapraz işlevli ekipler kullanarak, sorunun ve çözümün önemli, hatta kritik yönlerini gözden kaçırılmamasını sağlamaktır.

YYG yazılım geliştirme sürecinin proje yönetimi yönlerine odaklanmakta ve teknik uygulamalar belirtmemektedir. Yazılım geliştirme sürecinin teknik yönlerine daha çok odaklanan uç programlama gibi diğer çevik yöntemlerle bütünlük sağlamaktadır (Flora ve Chande, 2014).

2.3.7. Hızlı Uygulama Geliştirme (Rapid Application Development)

Hızlı uygulama geliştirme terimi ilk olarak James Martin tarafından “Rapid Application Development” adlı kitabında ortaya atılmıştır (Beynon-Davies vd., 1999). Martin kitabında, “Hızlı Uygulama Geliştirme, geleneksel yaşam döngüsüne göre çok daha hızlı geliştirme ve daha yüksek kaliteli sonuçlar sunmak için tasarlanmış bir geliştirme yaşam döngüsüdür. Yakın zamanda gelişen güçlü geliştirme yazılımlarından maksimum fayda sağlamak için tasarlanmıştır.” demektedir. Martin, hızlı geliştirmenin dört temel unsurunun araçlar, metodoloji, insanlar ve yönetim olduğunu belirtir (Daud vd., 2010).

Hızlı uygulama geliştirme, planlamadan daha çok geliştirmeye vurgu yapar. Geliştirme döngüleri zaman sınırlıdır ve aynı anda birden çok döngü geliştirilebilir. Geliştirme modüller şeklinde gerçekleştirilir. Her modül tamamlandıktan sonra proje sahibinin geri bildirimi alınır. Hızlı uygulama geliştirme metodolojisi, projelerin modüllere bölünmesi zorunluluğu ile küçük, orta ve büyük ölçekli projeler için uygundur (Despa, 2004).

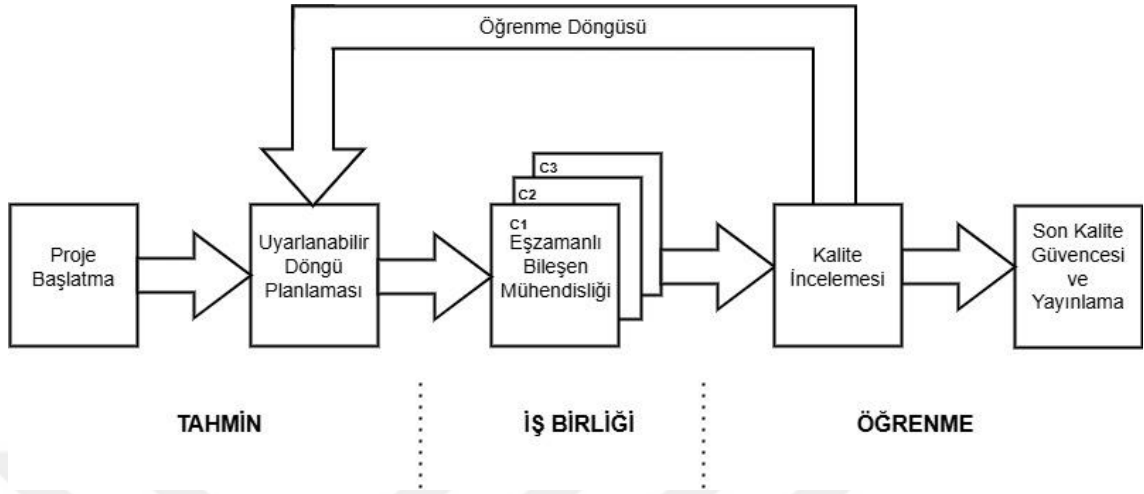


Şekil 2.21. Hızlı Uygulama Geliştirme Süreci
Kaynak: Despa (2004)

2.3.8. Uyarlanabilir Yazılım Geliştirme (Adaptive Software Development)

Uyarlanabilir yazılım geliştirme, 2000 yılında James A. Highsmith tarafından tanıtilen dinamik bir yazılım geliştirme yaşam döngüsüdür. Sürekli prototipleme ile yinelemeli ve artımlı geliştirme modelidir (Aitken ve Ilango, 2013). Highsmith, geleneksel statik planla-tasarla-inşa modelinin yerine, sürekli değişen proje

gereksinimlerine uyum sağlayabilen bir tahmin-iş birliği-öğren döngüsünü önermiştir (Chowdhury ve Huda, 2011).



Şekil 2.22. Uyarlanabilir Yazılım Geliştirme Süreci

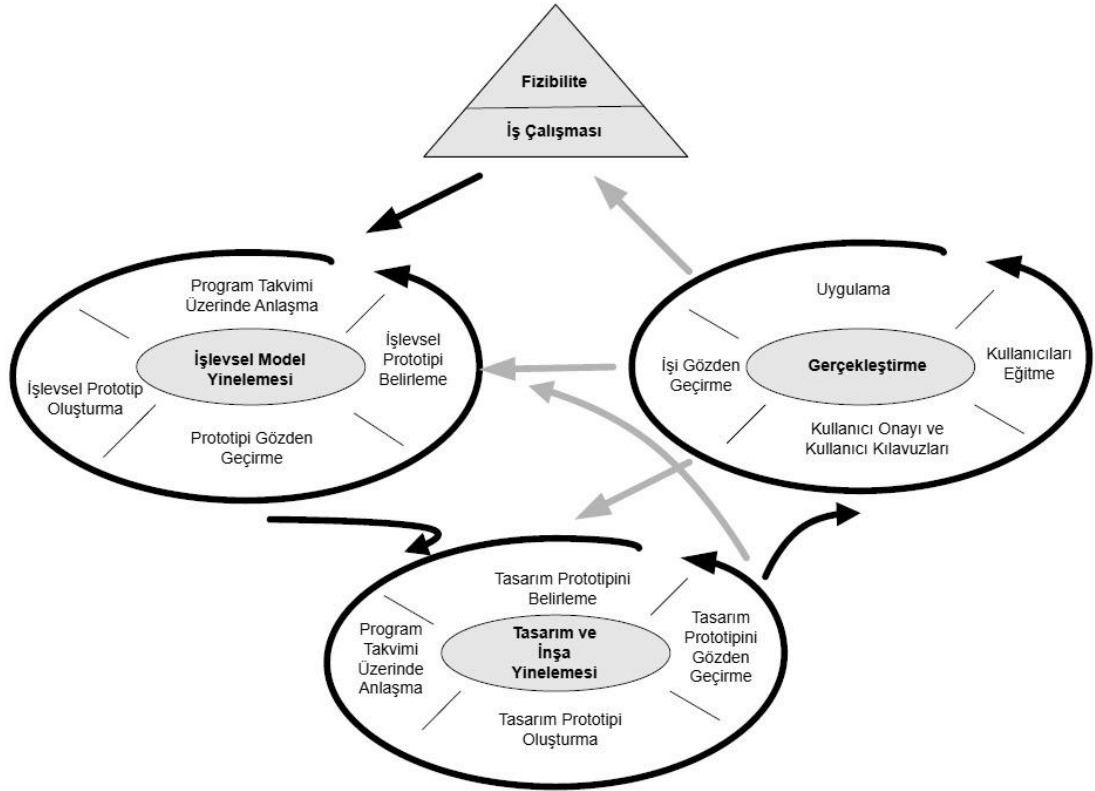
Kaynak: Highsmith (2000)

Süreç, proje hedeflerinin ve geliştirme döngüsü takvimlerinin belirlendiği tahmin aşaması ile başlar. Bu aşamada, projenin genel yönü çizilir ve geliştirme sürecinin ana hatları belirlenir. Ardından, iş birliği aşamasında, çeşitli bileşenler eşzamanlı olarak geliştirilir ve bu bileşenler sürekli olarak iyileştirilir. Bu nedenle, geliştirme döngüsünün planlaması, yinelemeli ve uyarlanabilir bir süreç olarak işlemeye devam eder. Sürecin son aşaması olan öğrenme aşamasında ise, projenin sonuçlarının hem müşteri perspektifinden hem de teknik açıdan kalitesi değerlendirilir. Bu aşama ayrıca kullanılan yöntemlerin, teslim ekibinin performansının ve projenin genel durumunun gözden geçirilerek öğrenilen derslerin kayıt altına alınmasını içerir (Highsmith, 2000; Chowdhury ve Huda, 2011).

2.3.9. Dinamik Sistem Geliştirme Yöntemi (Dynamic Systems Development Method)

Dinamik sistem geliştirme yöntemi (DSGY), 1994 yılında konsorsiyum tarafından bağımsız bir hızlı uygulama geliştirme çerçevesi geliştirme ve tanıtmaya amacıyla ortaya konulmuştur (Kirpitsas ve Pachidis, 2022). DSGY, işletmelerin ihtiyaçlarını derinlemesine anlamaya, işlevsel yazılım çözümleri sunmaya ve bu çözümleri mümkün olan en hızlı şekilde teslim etmeye odaklanan bir hızlı uygulama geliştirme metodolojisidir. DSGY, hızlı uygulama geliştirme süreçlerinde kontrol ve en iyi

uygulamalardan oluşan kapsamlı bir çerçeve sağlamaktadır (Stapleton, 1999). Bu yöntemin amacı, değişen gereksinimlere uyum sağlarken projeleri zamanında ve bütçe dahilinde tamamlamaktır. Bu, zaman ve kaynakların sabitlenmesiyle birlikte, belirlenen sınırlar dahilinde sunulacak işlevselliğin gerektiğinde ayarlanması yoluyla gerçekleştirilmektedir (Aitken ve Ilango, 2013). Ayrıca işletmeye gerçek faydaların erken aşamada teslim edilmesine odaklanmaktadır (Flora ve Chande, 2014).



Şekil 2.23. Dinamik Sistem Geliştirme Süreci

Kaynak: Stapleton (1997)

DSGY beş temel geliştirme aşamasından oluşmaktadır (Beynon-Davies vd., 1999).

1. *Fizibilite Çalışması:* Bu aşamada, projenin iş ve teknik açıdan uygulanabilirliği değerlendirilir ve projenin hızlı uygulama geliştirme yaklaşımına uygun olup olmadığı incelenir.
2. *İş Çalışması:* Bu aşama, projenin yüksek seviyedeki işlevsellik gereksinimlerini ve projeden etkilenecek ana iş birimlerini tanımlamaya odaklanır.
3. *İşlevsel Model Yinelemesi:* Bu aşamada, gereksinim duyulan işlevsellik, çalışan bir prototip üzerinden oluşturularak gösterilir ve doğrulanır.

4. *Sistem Tasarımı ve İnşa Yinelemesi*: İşlevsel prototip, özellikle işlevsel olmayan gereksinimlere uygun hale getirilmek üzere bu aşamada iyileştirilir ve nihai yapıya kavuşturulur.
5. *Gerçekleştirme*: Projenin kullanıcıya devredildiği ve başarısının değerlendirildiği son aşamadır. Bu aşamada, proje sonuçlarının gözden geçirilmesi de yapılır.

2.3.10. Kristal Aile Metodolojisi (Crystal Family Methodology)

Kristal, Alistair Cockburn tarafından 1990'ların başında geliştirilmiş ve yazılım projelerinde insan faktörünü ve iletişimi önceliklendiren bir çevik yöntemler ailesi olarak tanımlanmıştır (Highsmith, 2002; Anver vd., 2017). Cockburn, ekipler arasındaki iletişim eksikliğinin yazılım projelerinde başarısızlığın temel nedenlerinden biri olduğunu gözlemlemiş ve Kristal metodolojilerini bu eksikliği gidermek için modellemiştir (Cohen vd., 2004). Bu metodolojiler, yüz yüze etkileşimlerin yazılı dokümantasyona olan bağımlılığı azaltarak sistem teslimatını daha olası hale getirdiğini savunur. Cockburn, "Sistemin çalışan ve test edilmiş parçalarını sık teslim ederek yazılı vaat notlarına olan bağımlılığı azaltabilir ve başarı oranını artırabilirsiniz" diyerek bu yaklaşımı desteklemiştir (Cohen vd., 2004). Highsmith ise bu metodolojilerin süreçten çok insanlar, etkileşim ve topluluk gibi faktörlere odaklandığını, sürecin ise ikincil bir öneme sahip olduğunu vurgulamıştır (Highsmith, 2002).

Kristal metodolojileri, farklı proje türlerine uyarlanabilen esnek bir çerçeve sunar. Kristal metodolojisi ve metodoloji tasarımını etkileyen üç faktör; iletişim yükü (ekip büyüklüğü ile belirlenir), sistem kritiklik derecesi ve proje öncelikleridir (Highsmith, 2002). Metodolojiler, Şekil 2.24'te yer aldığı gibi artan karmaşıklık ve risk seviyelerini ifade eden renklerle kategorize edilmiştir. Örneğin, küçük ve düşük riskli projelerde Crystal Clear, orta büyüklükte projelerde Crystal Yellow, büyük projelerde Crystal Orange ve çok büyük kritik projelerde Crystal Red kullanılır (Anver vd., 2017; Kumar vd., 2019). Renk sistemi, projelerin farklı gereksinimlerine uygun metodolojinin seçilmesine olanak tanır ve projelerin başarı oranını artırmayı hedefler (Highsmith, 2002). Projelerin büyüklüğü ve karmaşıklığına göre en uygun metodoloji seçilebilir. Proje boyutu, ekip büyüklüğü ve sistemin kritiklik seviyesi gibi özellikler, bu seçimde belirleyici rol oynar (Anver vd., 2017).

Sistemin Kritiklik Seviyesi				
	L6	L20	L40	L80
	E6	E20	E40	E80
	D6	D20	D40	D80
	C6	C20	C40	C80
	Clear	Yellow	Orange	Red
	Proje Boyutu			

Şekil 2.24. Kristal Aile Metodolojisi

Kaynak: Abrahamsson vd. (2002)

Kristal metodolojilerinin temel değerleri arasında insanlar, etkileşim, topluluk, yetenekler, beceriler ve iletişim yer alır (Highsmith, 2002; PMI ve Agile Alliance, 2017). Metodolojinin insan odaklı yaklaşımı, ekiplerin iş birliğini ve iletişimini güçlendirmeyi hedefler. Bununla birlikte, Kristal metodolojilerinin başarısını artıran bazı temel özellikler bulunmaktadır. Artımlı geliştirme, sık teslimat, yansıtıcı iyileştirme ve osmotik iletişim bu özelliklerden bazılarıdır (Highsmith, 2002; Cohen vd., 2004). Osmotik iletişim, ekip üyeleri arasında doğal yollarla bilgi paylaşımını kolaylaştırarak iş birliğini artırmayı hedeflemektedir. Ayrıca, otomatik testler, yapılandırma yönetimi ve sürekli entegrasyon gibi modern yazılım geliştirme araçlarının kullanımını teşvik eder (Kumar vd., 2019).

Kristal metodolojileri, yalnızca süreç odaklı bir araç olmanın ötesinde, ekiplerin güven ve iş birliği içinde çalışmasını sağlayan bir yapı olarak tanımlanır (Highsmith, 2002). Bu metodolojiler, XP ve Scrum gibi diğer çevik yöntemlerle kolayca entegre edilebilir ve değişen proje gereksinimlerine uyum sağlayabilir (Cohen vd., 2004; Kumar vd., 2019). Projelerin farklı boyut ve risk seviyelerine göre özelleştirilebilmesi, Kristal metodolojilerini çevik yazılım geliştirme dünyasında önemli bir yere taşımaktadır. Özellikle iletişim odaklı yapısı sayesinde, yazılım geliştirme süreçlerinde başarı oranını artıran etkili bir çerçeve sunmaktadır (PMI ve Agile Alliance, 2017).

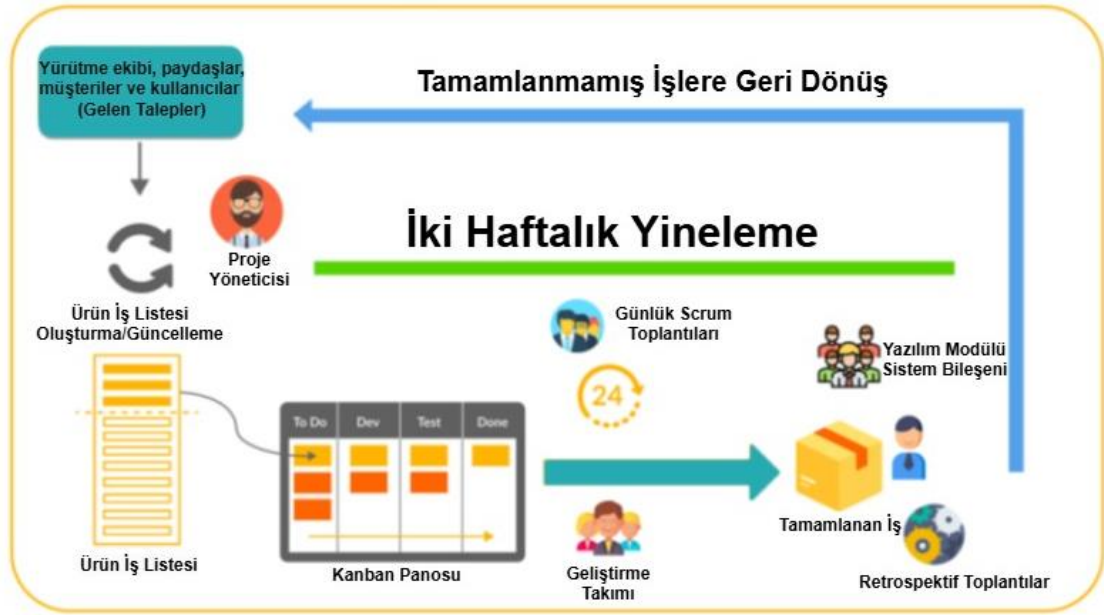
2.3.11. Scrumban

Scrumban, ilk olarak Corey Ladas tarafından 2009 yılında tanıtılan Scrum ve Kanban yöntemlerinin kombinasyonundan oluşan hibrit bir çevik metodolojidir (Fuior, 2019; Aini vd., 2021; Kirpitsas ve Pachidis, 2022). Bu yöntem, Scrum'un temel çerçeve yapısını ve Kanban'ın esneklik ve süreç iyileştirme odaklı yaklaşımını bir araya getirerek, dinamik müşteri gereksinimlerine ve yazılım geliştirme sürecindeki sık karşılaşılan problemlere çözüm sunmayı amaçlamaktadır (PMI ve Agile Alliance, 2017; Kumar vd., 2019). Scrumban, sprint tanımları yerine Kanban'ın çekme mekanizmasını kullanarak iş akışını düzenler. Görevler, kanban panosunda görselleştirilir ve devam eden iş sınırlarıyla yönetilir. Bu süreçte planlama, genellikle devam eden iş seviyesinin önceden belirlenmiş sınırların altına düşmesiyle tetiklenir (Kumar vd., 2019).

Scrumban, Scrum'dan farklı olarak belirli roller bulunmamaktadır. Günlük toplantılar ve kaizen etkinlikleri gibi süreç iyileştirme yöntemlerini teşvik eden Scrumban'da yineleme süreleri iki haftayı geçmemelidir. Ancak uzun süreli görevlerin birden fazla yineleme boyunca devam etmesine izin verir. Bu durum, yineleme sonunda tamamlanmamış bir ürünün oluşmasına yol açabilir. Tamamlanmamış özelliklerin nihai üründe yer almaması için ise Özellik Dondurma (Feature Freeze) konsepti uygulanır. Bu yaklaşım, yarım kalan işlerin teslim edilmesini önler ve artımlı sürüm sunumunu destekler (Fuior, 2019).

Scrumban yönteminin geliştirme süreci Şekil 2.25'te gösterilmektedir. Süreç, sistem gereksinimlerine dayalı bir iş listesi oluşturulmasıyla başlar. Ardından, bu iş listesi ekip tarafından zorluk seviyesi ve tamamlama hızına göre yeniden değerlendirilerek seçilen işler kanban panosundaki "Yapılacaklar" bölümüne eklenir. Her bir işin, iki hafta içinde veya yineleme süresi boyunca tamamlanması gerekmektedir. Ancak uzun süreli görevlerin birden fazla yineleme boyunca devam etmesine izin verir. Ekip, her gün düzenli olarak günlük scrum toplantıları gerçekleştirir. Bu toplantılarda, tamamlanan işler, karşılaşılan engeller ve gelecekteki iş planları tartışılır. Yineleme sonunda, ekip tarafından sistem demosu gerçekleştirilir ve sistemin yayınlanmaya hazır hale getirilmesi sağlanır. Scrumban yönteminin diğer yöntemlerden ayırt edici bir özelliği, tüm işler tamamlandıktan sonra retrospektif toplantı yapılmasıdır. Bu toplantıda ekip üyeleri, önceki yinelemeyi analiz eder ve bir sonraki yinelemenin daha etkili ve verimli olması

için yapılacak iyileştirme planlarını tartışır (Fuior, 2019; Kumar vd., 2019; Aini vd., 2021).



Şekil 2.25. Scrumban Süreci

Kaynak: Aini vd. (2021)

Scrumban, ekiplerin süreçlerini sürekli olarak iyileştirmelerini ve değişken gereksinimlere uyum sağlamalarını kolaylaştırarak, yazılım geliştirme projelerinde esneklik ve verimliliği artıran bir yöntem olarak öne çıkmaktadır (PMI ve Agile Alliance, 2017).

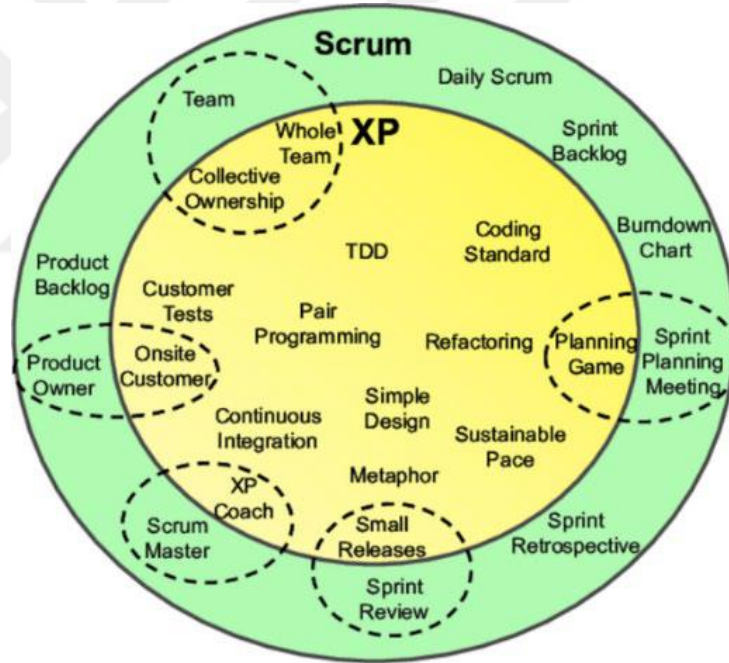
2.3.12. Scrum/XP Hibrit (Scrum/XP Hybrid)

Scrum/XP hibrit yöntemi, Scrum ve XP yaklaşımlarını birleştiren bir yöntem olarak 2002 yılında Mar ve Schwaber tarafından geliştirilmiştir (Fuior, 2019; Kirpitsas ve Pachidis, 2022). Bu metodoloji, Scrum'un çevik proje yönetimi çerçevesini ve XP'nin mühendislik odaklı pratiklerini bir araya getirerek (Şekil 2.26) yazılım geliştirme süreçlerinde karma bir yapı oluşturmayı hedeflemektedir (Fuior, 2019). İki yaklaşımın planlama süreçlerinde belirli örtüşmeler bulunsa da genel olarak birbirini tamamlayan özelliklere sahiptir (Kirpitsas ve Pachidis, 2022).

Scrum/XP hibriti, Scrum'un rehberlik ettiği çevik yönetim ilkelerini benimserken, XP'nin planlama, tasarım, geliştirme, test ve entegrasyon gibi teknik aşamalarındaki prensiplerini sürece entegre etmektedir (Fuior, 2019). Ayrıca her iki metodolojinin ortak

özellikleri arasında yinelemelerle çalışma, her yineleme sonunda bir ürün artışı teslim edilmesi, müşterinin bir sonraki yinelemede neyin geliştirilmesi gerektiğini yönlendirme rolü ve kendi kendini organize eden geliştirme ekipleri yer almaktadır (Kirpitsas ve Pachidis, 2022).

Scrum/XP hibrit metodolojisinde scrum master rolü korunmakta ve ekipler, Scrum ile XP pratiklerini etkili bir şekilde uygulamaya teşvik edilmektedir (Kirpitsas ve Pachidis, 2022). Aynı zamanda ürün sahibi, geliştirme ekibine neyin geliştirilmesi gerektiği konusunda rehberlik etme sorumluluğunu üstlenmektedir (Fuior, 2019). Böylelikle, bu metodoloji Scrum'un çeviklik odaklı yapısını ve XP'nin teknik mükemmeliyet anlayışını bir araya getirerek, yazılım geliştirme süreçlerini hem yönetim hem de mühendislik açısından daha verimli hale getirmektedir (Fuior, 2019; Kirpitsas ve Pachidis, 2022).



Şekil 2.26. Scrum/XP Hibrit
Kaynak: Fuior (2019)

2.4. ÇEVİK PRATİKLER

Çevik pratikler, yazılım geliştirme süreçlerinde çevik yöntemlerin temel ilkelerini ve değerlerini somutlaştıran, ekiplerin verimliliğini artıran ve yazılım kalitesini geliştiren faaliyetler ve tekniklerdir (Beck ve Andres, 2004; Abrantes ve Travassos, 2011; Arcos-Medina ve Mauricio, 2020). Bu pratikler, yazılım geliştirme sürecinde ekipler için bir rehber niteliği taşıırken, aynı zamanda müşteri ihtiyaçlarına hızlı yanıt verme ve değişen gereksinimlere uyum sağlama konusunda esneklik sunmaktadır (Jain ve Suman, 2016; Al-Sakkaf vd., 2017). Çevik pratikler yüksek esneklik, artan üretkenlik, müşteri memnuniyeti ve hızlı geliştirme sağlamakta; iletişim ve iş birliğini artırmakta, hata oranlarını azaltmakta ve gereksinimlerdeki geç değişiklikleri yönetmektedir (Jain ve Sumar, 2016). Çevik pratikler, yazılım geliştirme projelerinde sıkça karşılaşılan zorlukların üstesinden gelmek için hem sosyal hem de teknik boyutları bir araya getiren bir yaklaşımdır (Hummel vd., 2015).

Literatürde, çevik pratikleri genellikle teknik pratikler ve sosyal pratikler olmak üzere iki temel gruba ayırmaktadır (Anjani vd., 2021). Teknik pratikler, yazılım mühendisliği faaliyetlerine odaklanır ve yazılımın kalite, sürdürülebilirlik ve hata oranını azaltmaya yönelik araç ve yöntemleri içerir. Örneğin; birim testi, sürekli entegrasyon, yeniden kod düzenleme ve kodlama standartları gibi pratikler, yazılım geliştirme sürecinin teknik yönlerini optimize etmeye yöneliktir (Chow ve Cao, 2008; Gupta vd., 2019). Bu pratikler, kodun modüler ve sürdürülebilir olmasını sağlarken, geliştiricilerin değişikliklere daha kolay adapte olmasına da olanak tanır (Beck ve Andres, 2004).

Sosyal pratikler ise ekip içi iletişim, iş birliği ve müşteriyle etkileşim gibi insan odaklı süreçlere odaklanır. Günlük ayakta toplantılar, retrospektifler, yineleme planlaması ve müşteri erişimi gibi uygulamalar, ekip üyeleri arasındaki koordinasyonu güçlendirirken, aynı zamanda müşteri ihtiyaçlarına odaklanmayı kolaylaştırır (So ve Scholl, 2009; Hummel vd., 2015). Sosyal pratikler, yazılım geliştirme süreçlerinde bireyler arasındaki doğrudan iletişimi artırarak, projelerdeki bilgi akışını ve ekip verimliliğini iyileştirir.

Conboy ve Fitzgerald (2010), çevik pratikleri bir “araç takımı” olarak nitelendirmiş ve bu pratiklerin bağlamdan bağımsız olmadığını, ancak organizasyonların kendi ihtiyaçlarına ve bağamlarına göre özelleştirilebileceğini ifade etmişlerdir (Tripp vd.,

2016). Bu özelleştirme yeteneği, çevik pratiklerin esnek ve uyarlanabilir yapısını ortaya koymaktadır. Örneğin, bir ekip için kritik olan sürekli entegrasyon pratiği, başka bir ekip için bir öncelik olmayabilir. Bu durum, çevik pratiklerin bağlama özgü bir şekilde yeniden düzenlenebileceğini ve projenin gereksinimlerine uygun hale getirilebileceğini göstermektedir (Kurapati vd., 2012).

Çevik pratiklerin etkinliği, uygulandığı bağlama, ekibin pratiği ne kadar yoğun bir şekilde benimsediğine ve pratiğin süreçlere nasıl entegre edildiğine bağlıdır (Abrantes ve Travassos, 2011). Pratiklerin yalnızca bireysel düzeyde uygulanması, beklenen faydayı sağlamayabilir. Bu nedenle, çevik pratiklerin birbiriyle uyumlu bir şekilde birleşerek uygulanması, bireysel katkılarında daha büyük bir bütün oluşturur ve yazılım geliştirme sürecinin başarısını artırır (Beck ve Andres, 2004; Martin, 2014). Örneğin, sürekli entegrasyon, birim testleri ve yeniden kod düzenleme birlikte uygulandığında, yazılımın hata oranı düşürülerek kalite artırılabilir (Jain ve Suman, 2016).

Çevik yazılım geliştirmenin uygulanmasını destekleyen çok çeşitli pratikler bulunmaktadır (Anjani vd., 2021). Bu pratiklerinden öne çıkanlar aşağıda başlıklar halinde sunulmuştur.

2.4.1. Kullanıcı Hikayesi (User Story)

Kullanıcı hikayesi, müşterilere sunulacak değerin kısa bir tanımıdır (PMI ve Agile Alliance, 2017). Müşteriye görünür işlevsellik birimleri kullanılarak planlama yapılmasını sağlar ve kısa yazılı ya da görsel açıklamalarla desteklenir (Beck ve Andres, 2004). Kullanıcı hikayeleri; kullanıcıların talep ettiği işlevler, düzeltmeler veya işlevsel olmayan gereksinimlerin kaydedildiği araçlardır (Larman, 2003).

Kullanıcı hikayeleri, bir gereksinimle ilgili süregelen bir konuşmayı hatırlatıcı bir araç olup, müşteriler tarafından gereksinimlerin önceliklendirilmesi ve planlanması için kullanılır (Martin, 2014). Sistem kullanıcılarının veya müşterilerin talep ettikleri işlevsellikleri kısa ifadelerle tanımlarken, özelliklerin detaylandırılmasını teşvik eden bir araç olarak işlev görür (Williams, 2010). Çoğu çevik metodolojide kullanıcı hikayeleri, “Bir [kullanıcı rolü] olarak, [istediğim işlev] istiyorum ki [amacım] gerçekleşsin” formatında ifade edilir. Örneğin, bir müşteri tarafından, “Bir kahve makinesi sahibi

olarak, üç kahve çeşidinin tarifini sağlamak istiyorum ki müşteriye çeşitli kahveler sunabileyim.” şeklinde ifade edilebilir (Williams, 2010).

Bazı çevik yöntemlerde, tüm etkinlikler kullanıcı hikayeleri ile başlar. Bu hikayeler, projenin varlık nedenini oluşturur ve geliştirme ekibine belirli teslim edilebilir öğeleri tanımlama fırsatı sunar (Webb, 2015).

2.4.2. Kısa Yinelemeler (Short Iterations)

Yineleme, sabit bir zaman aralığında değer üretmek için tüm geliştirme faaliyetlerinin gerçekleştirildiği bir süreçtir (PMI ve Agile Alliance, 2017). Çevik ekipler genellikle 1 ila 4 hafta süren zaman kutulu yinelemelerle çalışır; bu süreçte süre sabit tutulur ve hedeflere ulaşılmadığında kapsam daraltılarak devam edilir (Williams, 2010). Her yinelemenin sonunda, tamamlanma kriterlerini karşılayan ve potansiyel olarak dağıtılabılır bir çalışır kod teslim edilir. Teslim edilen bu kod, müşteri ve diğer paydaşlardan alınan geri bildirimlerle geliştirilir (Williams, 2010; PMI ve Agile Alliance, 2017).

Yinelemeli geliştirme, sistemin adım adım büyütülmesini sağlayan bir yaklaşımdır. Her yineleme, gereksinim analizi, tasarım, programlama ve test gibi etkinlikleri içeren, kendi içinde tamamlanmış bir mini proje olarak yapılandırılır. Bir yinelemenin sonunda hedef, stabil, entegre edilmiş ve test edilmiş kısmi bir sistem olan bir “yineleme sürümünü” teslim etmektir (Larman, 2003). Bu yöntem, Scrum ve XP gibi çevik yöntemlerin temelini oluşturur. Çevik yaklaşımlarda, ekipler küçük ama tamamlanmış işlevsel özellikler üreterek her aşamada geri bildirim alır ve bu geri bildirimler doğrultusunda süreci iyileştirir (Larman, 2003; PMI ve Agile Alliance, 2017).

2.4.3. Kısa Sürümler (Short Releases) - Küçük Sürümler (Small Releases)

Yazılım geliştirme süreçlerinde kısa sürümler, ürünü müşteriye hızlıca ulaştırmayı ve geri bildirim alarak sürekli iyileştirmeyi hedeflemektedir. Bu yaklaşım, yazılımın desteklenen sürümler halinde her üç ayda bir veya daha kısa sürelerde müşterilere sunulmasını içerir. Kısa sürümler, özellikle rekabet avantajı sağlamak ve geliştirme sürecini müşteri geri bildirimleriyle yönlendirmek için kritik bir rol oynar (Williams, 2010).

Sistem, tüm problemin çözümü tamamlanmadan birkaç ay içinde üretime alınır ve sonraki sürümler (günlükten aylığa kadar değişen periyotlarla) sık aralıklarla gerçekleştirilir (Beck, 1999). Kent Beck, her sürümün mümkün olduğunca küçük olması ve en değerli iş gereksinimlerini içermesi gerektiğini vurgular (Beck, 2000). Kısa sürüm pratiği, ekibin kısa sürede sahaya sunulabilecek, işlevsellik açısından uyumlu gruplamaları göz önünde bulundurmasını gerektirir (Williams, 2010). Kısa sürümler ayrıca, uzun projelerde sıklıkla eksik olan başarı hissini sağlarken, ekiplerin sık ve anlamlı geri bildirim almasına da yardımcı olur (Highsmith, 2002).

2.3.4. Sprint

Kısa yinelemenin scrum'daki karşılığıdır (Williams, 2010). Sprint, scrum takımlarının işlevselliği geliştirmek veya yeni bir işlevsellik oluşturmak için kullandığı yinelemeli bir döngüdür (Jain ve Suman, 2016). Sprint, scrum metodolojisinin kalbini oluşturan bir süreçtir. Sprint'in amacı, sürecin sonunda teslim edilebilir bir ürün sunmaktır (Anderson, 2004). Bu ürün, küçük bir ürün olabilir ve yalnızca iş açısından gerçekten mantıklıysa teslim edilir. Ancak her durumda, bir kullanıcı hikayesini ele alır ve somut bir çıktı sunar. Sprintler, zamanla sınırlandırılan zaman kutulu (timeboxed) çalışmalardır. En uzun sprintler genellikle dört hafta ile sınırlıdır. Daha kısa sprintler, örneğin bir haftalık olanlar da zaman zaman görülse de ortalama süre genellikle iki hafta olarak kabul edilir (Webb, 2015).

2.4.5. Artımlı Tasarım (Incremental Design)

Artımlı tasarım, bir sistemin tasarımını aşamalı olarak geliştirmeyi ve her aşamada mevcut ihtiyaçlara en iyi şekilde uyum sağlamayı hedefleyen bir yaklaşımdır. Artımlı tasarım, sistemin tasarımına her gün küçük yatırımlar yapmayı ve mevcut ihtiyaçlara uyum sağlamak için sürekli iyileştirmeler yapmayı hedefler. Bu yaklaşım, ileriye dönük kapsamlı tasarımdan ziyade, tasarımın uygulamaya en yakın zamanda yapılmasının daha verimli olduğunu savunur. Tasarım sürecinde edinilen anlayışlara göre sistem kademeli olarak düzenlenir, bu da daha basit sistemler, erken ilerleme ve kolay test yazımı gibi avantajlar sağlar (Beck ve Andres, 2004).

Ekipler, genellikle “Yineleme 0” olarak adlandırılan aşamada sistemin mimari ve tasarıma yönelik temel bir anlayış geliştirir ve her yinelemede tasarımı seçilen özelliklere göre yeniden değerlendirir. Bu yöntem, geniş kapsamlı ön tasarım yatırımlarından kaçınarak, tasarım çalışmalarını her yinelemeye dağıtır ve daha çevik bir geliştirme süreci sağlar (Williams, 2010).

2.4.6. Basit Tasarım (Simple Design)

Basit tasarım, yazılım geliştirme süreçlerinde yalnızca mevcut gereksinimlere odaklanmayı ve bu gereksinimleri karşılamak için en sade ve etkili çözümleri sunmayı amaçlar (Highsmith, 2002; Larman, 2003). Bu yaklaşım, gelecekteki olası ihtiyaçlara yönelik spekülatif tasarımların getirebileceği karmaşıklıklardan kaçınmayı ve yalnızca bugünün ihtiyaçlarını karşılamak için özel çözümler geliştirmeyi savunur (Larman, 2003).

Basit bir tasarım, tüm testleri başarıyla çalıştırmalı, geliştiricilerin amaçlarını açıkça ifade etmeli, yinelenen kod içermemeli ve mümkün olan en az sayıda sınıf ve metoda sahip olmalıdır. Bu yaklaşım, “Her şeyi bir kez ve yalnızca bir kez söyle” olarak özetlenebilir (Beck, 1999).

Highsmith (2002), basit tasarımın iki temel unsura dayandığını ifade eder. İlk olarak, tasarım yalnızca mevcut işlevsellik için yapılmalı ve gelecekteki olası gereksinimlere yönelik spekülatif bir yaklaşım benimsenmemelidir. İkinci olarak, tasarım, mevcut işlevselliği en etkili şekilde karşılayacak en basit çözüm üzerine inşa edilmelidir. Bu yaklaşım, geleceği tahmin etmeye çalışmaktan kaçınıp, bugünün ihtiyaçlarına yönelik en iyi ve en sade tasarımı oluşturmayı hedeflemektedir.

Yazılım geliştirme sürecinde, gelecekteki olası ihtiyaçlar için spekülatif tasarım yapılmaması önemlidir. Mevcut gereksinimlere doğrudan hizmet etmeyen genel bileşenlerin oluşturulmasından kaçınılmalıdır. Tasarım, yalnızca mevcut ihtiyaçlara odaklanmalı; yinelenen kodların önlenmesi, sınıf ve metod sayısının minimal seviyede tutulması ve tasarımın sade ve anlaşılır olması hedeflenmelidir. Bu yaklaşım, gereksiz karmaşıklığı ortadan kaldırarak sistemin sürdürülebilirliğini artırır ve bakım süreçlerini kolaylaştırır (Larman, 2003).

2.4.7. Metafor (Metaphor)

Metafor, XP’de sistem tasarımını yönlendiren ve müşteri ile geliştiriciler arasında ortak bir anlayış sağlayan temel bir araçtır. Sistem, müşteri ve geliştiriciler arasında paylaşılan bir metafor ya da metaforlar kümesiyle şekillenir (Beck, 1999). Metafor, projenin genel çerçevesini tanımlarken, bireysel özelliklerin tanımlanması için hikayeler kullanılır ve projeye geniş bir perspektif sunar (Highsmith, 2002). Aynı zamanda, sistemin nasıl çalıştığını basit ve ortak bir hikayeyle açıklayarak taraflar arasında proje hakkında ortak bir anlayış oluşturur (Abrantes ve Travassos, 2011).

Metafor, genellikle sistem unsurlarına anlam kazandıran ve bu unsurlar arasındaki ilişkileri tanımlayan bir isimlendirme sistemi olarak işlev görür (Martin, 2014). Büyük resim niteliğindeki bu yaklaşım, her modülün sistemdeki yerini ve rolünü anlamayı kolaylaştırır. Eğer bir modül metaforla uyumlu değilse, bu genellikle o modülde bir sorunun varlığına işaret eder (Martin, 2014). Tüm geliştirme süreci, sistemin işleyişini yansıtan basit ve ortak bir metafor etrafında şekillenir (Beck, 2000).

2.4.8. Planlama Oyunu (Planning Game)

Planlama oyunu, müşteriler ile geliştiriciler arasında iş birliğini ve sorumluluk paylaşımını temel alan bir yöntemdir (Martin, 2014). Müşteriler, programcılarının sunduğu tahminlere dayanarak sürümlerin kapsamını ve zamanlamasını belirlerken, geliştiriciler yalnızca o yineleme için talep edilen işlevselliği uygular (Beck, 1999). Her yinelemenin başında müşteriler, yöneticiler ve geliştiriciler bir araya gelerek gereksinimleri ayrıntılandırır, tahmin eder ve önceliklendirir. Bu gereksinimler, tüm taraflarca anlaşılabilir bir dilde yazılarak hikâye kartlarına (kullanıcı hikayesi) kaydedilir (Cohen vd., 2004).

Planlama oyununda, müşteriler ve geliştirme ekipleri arasında sorumluluklar net bir şekilde tanımlanır. Müşteriler bir özelliğin ne kadar önemli olduğuna karar verirken, geliştiriciler ise bu özelliğin uygulanmasının ne kadar mal olacağını belirler. Her sürüm ve yineleme başlangıcında, geliştiriciler müşterilere son süreçte tamamlanan işlere bağlı olarak bir bütçe sunar. Müşteriler, bu bütçe dahilinde kalan hikayeleri seçer ve bu doğrultuda projeyi yönlendirir (Martin, 2014).

Bu yöntem, kısa yinelemeler ve sık sürümlerle projeye uyum sağlamayı kolaylaştırır. Müşteriler, geliştiricilerin ilerleme hızını gözlemleyerek projelerin süresi ve maliyeti hakkında daha gerçekçi tahminlerde bulunabilirler. Planlama oyunundaki aktif katılım, tarafların planı daha iyi anlamasını ve projeye yönelik ortak bir vizyon geliştirmesini sağlar (Martin, 2014; Highsmith, 2002).

2.4.9. Çapraz Fonksiyonel Ekipler (Cross-Functional Teams)

Çapraz fonksiyonel çevik ekipler, bir ürünün çalışır durumdaki versiyonunu üretmek için gerekli tüm yetkinliklere sahip bireylerden oluşur. Bu ekipler, tasarımcılar, geliştiriciler, test uzmanları ve diğer gerekli rollerle dışa bağımlılık olmadan mümkün olan en kısa sürede ve daha yüksek kalitede bitmiş iş teslim edebildikleri için kritik öneme sahiptir (PMI ve Agile Alliance, 2017; Schwaber ve Sutherland, 2017). Ekip üyeleri, işin tamamlanması için gereken tüm becerilere birlikte sahiptir ve işin sahipliğini kolektif olarak üstlenir. Düzenli aralıklarla yayınlanabilir ürünler sunar ve sık sık işlevsel ürün parçaları üretir. Bu bağımsızlık, müşteri değerini hızlı bir şekilde sunabilmelerini ve değişen ihtiyaçlara esnek bir şekilde uyum sağlamalarını sağlar (PMI ve Agile Alliance, 2017).

2.4.10. Ürün İş Listesi (Product Backlog)

Ürün iş listesi, yazılım geliştirme sürecinde nihai ürün için gerekli olan işlerin ve gereksinimlerin hikâye formatında sunulduğu sıralı listedir (Anderson, 2004; PMI ve Agile Alliance, 2017). Liste, işlevsel ve işlevsel olmayan gereksinimlerden oluşur ve ürün sahibinin sorumluluğunda şeffaf, görünür ve erişilebilir bir şekilde düzenlenir (Cole ve Scotcher, 2015). Kullanıcı hikayelerinden oluşan ürün iş listesi, geliştirme ekibinin teslim edilebilir çıktılara dönüştüreceği görevleri tanımlar (Webb, 2015).

Bu liste, süreç boyunca öğelerin eklenmesi, çıkarılması, güncellenmesi ve önceliklendirilmesiyle sürekli olarak değişir ve nihai üründe ihtiyaç duyulan her şeyi mevcut bilgilere dayanarak belirler (Abrantes ve Travassos, 2011). Ürün iş listesi, sürüm iş listesi ve yineleme iş listesi olarak alt parçalara ayrılır. Sürüm iş listesi, belirli bir sürüm için beklenen görevleri tanımlarken, yineleme iş listesi yalnızca belirli bir geliştirme yinelemesi için üzerinde anlaşılmış olan işleri içerir (Anderson, 2004).

Scrum gibi çevik yöntemlerde ürün iş listesi, tüm proje için eksiksiz bir liste oluşturmayı gerektirmez. İlk aşamada genel hatlar belirlenir ve her yinelemede detaylandırma yapılır (PMI ve Agile Alliance, 2017). Ürün sahibi, iş listesini düzenleme ve önceliklendirme sorumluluğunu taşıırken, bu liste ekip için net bir rehber görevi görür ve geliştirme sürecinin dinamik yapısına uyum sağlar (Cole ve Scotcher, 2015).

2.4.11. Sürüm ve Yineleme İş Listesi (Release and Iteration Backlog)

Ürün iş listesi, yazılım geliştirme sürecinde sürüm iş listesi ve yineleme iş listesi olarak iki alt bileşene ayrılmaktadır (Anderson, 2004).

Sürüm iş listesi, bir sürüm boyunca geliştirilmesi gereken iş ve teknik gereksinimlerden oluşan, sürekli güncellenen ve önceliklendirilmiş bir listedir. Her gereksinim için benzersiz bir kimlik numarası, kategori (örneğin, özellik, iyileştirme veya hata), durum, öncelik ve çaba tahmini gibi bilgiler içerir. Sürüm iş listesi, sürüm başlangıcında oluşturulur ve her yinelemede yeni hikayeler eklenebilir, çıkarılabilir veya mevcut hikayeler yeniden önceliklendirilir (Williams, 2010).

Yineleme iş listesi, mevcut yineleme için planlanmış iş, iyileştirmeler ve hataların bir listesidir. Yineleme iş listesi, yineleme boyunca ekibin tamamlamayı planladığı işlerin gerçek zamanlı ve görünür bir tablosunu sunar (Schwaber, 2004). Liste, ekip üyeleri tarafından günlük olarak güncellenir ve özellikler, müşteri veya ürün sahibi tarafından kabul edilip “tamamlanma kriterleri” karşılandığında iş listesinden çıkarılır. Yineleme iş listesi, sürüm iş listesindeki uzun vadeli hedeflerin kısa vadeli hedeflere dönüştürülmesini sağlar (Williams, 2010).

2.4.12. Sprint İş Listesi (Sprint Backlog)

Yineleme iş listenin (bkz. 2.4.11 Sürüm ve Yineleme İş Listesi) Scrum yöntemindeki karşılığıdır. Scrum takımının sprint sırasında tamamlanmak üzere belirlediği iş öğelerinin listesidir (PMI ve Agile Alliance, 2017). Sprint iş listesi, geliştiriciler tarafından oluşturulan ve onların kullanımına yönelik bir plan olup, sprint boyunca tamamlanması planlanan işlerin gerçek zamanlı ve görünür bir tablosunu sunar (Schwaber ve Sutherland, 2020).

2.4.13. Yineleme Planlama (Iteration Planning)

Yineleme planlama, bir çevik yöntemde her yeni yineleme başlamadan önce gerçekleştirilen temel bir süreçtir. Bu sürecin amacı, uygulanacak kullanıcı hikayelerini seçmek ve bu hikayeler için görevleri planlayarak ekibin kapasitesine uygun şekilde dağıtmaktır. Yineleme planlama genellikle 1 ila 3 hafta arasında süren yinelemeler için gerçekleştirilir (Larman, 2003).

Planlama sürecinde, müşteri ürün iş listesindeki hikayelerden öncelikli olanları belirleyerek ekip ile birlikte yineleme iş listesini oluşturur. Yineleme başladığında müşteri, ilgili yinelemedeki hikayelerin tanımını veya önceliğini değiştirmemeyi kabul eder (Martin, 2014). Ekip, hikâye boyutlarını dikkate alarak bir yineleme içinde tamamlayabileceklerinden daha fazla hikâyeye taahhütte bulunmamaya çalışır (PMI ve Agile Alliance, 2017). Ardından, ekip üyeleri bu hikayeleri daha küçük görevlere ayırır ve bu görevlerin süre tahminlerini yapar. Görevlerin tahmini süreleri genellikle yarım gün ile iki gün arasında olmalıdır. Daha uzun süreli görevler, daha küçük ve yönetilebilir birimlere bölünerek yeniden düzenlenir. Bu yöntem, görevlerin daha kolay izlenmesini ve ekip üyelerinin iş yükünü daha verimli bir şekilde yönetmesini sağlar. Görevlerin dağıtımını ise gönüllü olarak gerçekleştirilir (Larman, 2003).

2.4.14. Sprint Planlama (Sprint Planning)

Sprint planlama, yineleme planlama etkinliğinin (bkz. 2.4.13 Yineleme Planlama) Scrum yöntemindeki karşılığıdır. Bu etkinlik, her yeni sprintin başlangıcında gerçekleştirilen ve ekip üyelerinin ürün sahibi tarafından önceden gözden geçirilmiş, düzenlenmiş ve önceliklendirilmiş kullanıcı hikayelerini değerlendirdiği bir süreçtir (Jain ve Suman, 2016). Sprint planlamanın temel amacı, ekibin sprint süresine sığacak kadar işin seçilmesi ve bu işler üzerinde uzlaşmaya varılmasıdır. Etkinlik sırasında işin karmaşıklığı, riskler, boyut, çaba gereksinimleri, iş değeri ve iş ihtiyaçlarının ayrıntıları gibi unsurlar açıkça tartışılır. Amaç, ekibin üstleneceği işleri tam anlamıyla kavraması ve bu işlerde mutabık kalmasıdır (Cole ve Scotcher, 2015).

Sprint planlama, sprint boyunca gerçekleştirilecek işleri belirlemek ve planlamak için düzenlenir. Bu süreç üç temel konuya odaklanır. İlk olarak ürün sahibi, sprint'in ürünün değerini nasıl artıracığını önerir ve scrum ekibi, paydaşlara neden değerli

olduğunu açıklayan sprint hedefini belirler. Sprint hedefi, toplantı sonunda kesinleştirilir. İkinci olarak, geliştiriciler, ürün sahibi ile birlikte ürün iş listesinden sprint'e dahil edilecek maddeleri seçer. Geçmiş performans, kapasite ve tamamlanma tanımına ilişkin bilgi, seçilecek iş miktarının tahminini kolaylaştırır. Son olarak, geliştiriciler, seçilen iş maddelerini daha küçük parçalara ayırarak artımlar oluşturmayı planlar. Bu planlama süreci yalnızca geliştiricilerin inisiyatifindedir. Sprint planlama, scrum ekibi'nin iş birliğiyle etkili bir plan oluşturmasına olanak tanır (Schwaber ve Sutherland, 2020).

Sprint planlamada ekip, önceliklendirilmiş iş listesi üzerinden hangi işleri alacağına kendisi karar verir. Planlama etkinliğinde, sprint'e dahil edilmeden önce her kullanıcı hikayesinin tam olarak anlaşılması hedeflenir. İşe başlamadan önce iş gereksinimlerinin net bir şekilde belirlenmesi şarttır; herhangi bir belirsizlik önceden giderilmelidir. Bu süreçte, ürün sahibi işlerin netleştirilmesinden sorumludur. Ayrıca, belirsiz kullanıcı hikayeleri ya da açık olmayan taahhütlerle planlama oturumuna başlama hatasından kaçınılmalıdır. Kullanıcı hikayeleri sprint kapsamına alındıktan sonra scrum master ve ekip çalışmalara başlar. Bu noktadan itibaren düşünme ve planlama süreci sona erer ve gerçekleştirme süreci başlar (Cole ve Scotcher, 2015).

2.4.15. Sürüm Planlama (Release Planning)

Sürüm planlama, ileriye dönük teslimatları organize etmek amacıyla gerçekleştirilen düzenli toplantılardır. Sürümlerin belirli bir periyotla, örneğin iki haftada bir gerçekleştiği durumlarda, sürüm planlama faaliyetlerinin de bu düzenle uyumlu bir şekilde yürütülmesi gereklidir. Bu toplantıların temel odağı, gelecekteki sürümler için işlerin önceliklendirilmesi ve kaynakların etkili bir şekilde planlanmasıdır (Anderson, 2010). Sürüm planlamanın nihai amacı, müşteriye en yüksek değeri sunacak bir sonraki operasyonel sürümün kapsamını tanımlamaktır (Larman, 2003).

Sürüm planları, geliştiricilerin belirlediği bütçe doğrultusunda, müşteri tarafından seçilen ve önceliklendirilmiş kullanıcı hikayelerinden (sürüm iş listesi) oluşturulur. Geliştiriciler, önceki sürümlerde tamamladıkları iş miktarını göz önünde bulundurarak bir sonraki sürüm için bütçe belirler. Müşteri, bu bütçe sınırları dahilinde istediği sayıda kullanıcı hikayesini seçebilir ve bu hikayelerin sürüm içerisinde hangi sırayla

uygulanacağını belirleme hakkına sahiptir. Ayrıca, ekipler gerekli gördüklerinde sürümün ilk birkaç yinemesinde tamamlanacak işlerin planını da yapabilir (Martin, 2014).

Sürüm içerikleri kesin ve değişmez değildir; müşteri, sürecin herhangi bir aşamasında sürüm kapsamını yeniden değerlendirebilir. Bu değerlendirme kapsamında kullanıcı hikayeleri iptal edilebilir, yeni hikayeler eklenebilir veya mevcut hikayelerin öncelikleri değiştirilebilir (Martin, 2014).

2.4.16. Ayaküstü Toplantı-Günlük Scrum (Stand-Up Meeting-Daily Scrum)

Ayaküstü toplantılar, proje ilerlemesini izlemek, karşılaşılan sorunları vurgulamak ve günlük görevleri organize etmek amacıyla gerçekleştirilen yaklaşık 15 dakikalık kısa toplantılardır. Toplantının ayakta yapılması, katılımcıların dikkatli ve odaklanmış olmasını sağlamayı ve sürenin aşılmasını önlemeyi amaçlar (Abrantes ve Travassos, 2011).

Ayaküstü toplantılar, ekip üyelerinin önceki gün gerçekleştirdikleri işleri, o gün için planlarını ve karşılaştıkları veya öngördükleri engelleri paylaşmalarını içerir. Tipik bir ayaküstü toplantı, altı ila on iki kişiden oluşan bir ekip tarafından gerçekleştirilir ve genellikle şu üç temel sorunun yanıtlanmasını sağlar: “Dün ne başardınız?”, “Bugün ne yapacaksınız?” ve “Yardıma ihtiyaç duyduğunuz bir engel var mı?” (PMI ve Agile Alliance, 2017).

Kanban yönteminde ise ayaküstü toplantılar farklı bir formata evrilmiştir. Kanban panosu, kimin hangi iş üzerinde çalıştığını ve iş akışındaki değişiklikleri görsel olarak gösterir, bu da geleneksel üç sorunun sorulmasına olan ihtiyacı ortadan kaldırır. Toplantılar, iş akışına odaklanarak panodaki biletler üzerinden geriye doğru ilerler ve kolaylaştırıcı, gerektiğinde durum güncellemeleri veya ek bilgiler talep eder (Anderson, 2010).

Uç programlama, Scrum, Kanban, Scrumban, Scrum/XP hibrit vb. birçok çevik yöntemde yer bulan etkinlik yöntemlerde farklı şekillerde isimlendirilmektedir.

2.4.17. Yineleme Gösterimi (Iteration Demonstration)

Geliştirme ekibinin her yinelemenin sonunda tamamlanan işlevselliği müşteri, müşteri temsilcisi (örneğin, ürün yöneticisi) ve diğer paydaşlara sunmasıdır. Bu gösterim, geri bildirim almak, ekip motivasyonunu artırmak ve farkındalık yaratmak gibi üç temel amaca hizmet etmektedir. Geliştirilen yazılımın somut bir sürümünün müşteriye sunulması, geri bildirim sağlama sürecini kolaylaştırır ve bu geri bildirim, bir sonraki yineleme planlarına dahil edilebilir. Gösterim aynı zamanda ekibin tamamladığı çalışan kodu gururla sergileyerek motivasyonu artırmasını sağlar. Ayrıca, ürünle ilgili bilgi ve farkındalık oluşturmak amacıyla destek ekipleriyle etkileşim için bir fırsat sunar (Williams, 2010).

2.4.18. Sprint Değerlendirme (Sprint Review)

Sprint değerlendirme, her yinelemenin sonunda, sprintin çıktılarının incelendiği ve gelecekte yapılacak uyarlamaların belirlendiği bir toplantıdır. Scrum master tarafından düzenlenen toplantıya ekip üyeleri, ürün sahibi ve diğer paydaşlar katılır. Toplantıda, sprint boyunca tamamlanan kullanıcı hikayelerinden oluşan ürünün bir demosu sunulur (Larman, 2003; Webb, 2015; Schwaber ve Sutherland, 2020).

Toplantının amacı, paydaşlardan geri bildirim alarak sprint çıktıları ve çevresel koşullardaki değişiklikleri değerlendirmek ve ürün iş listesini yeni fırsatlara göre güncellemektir. Alınan geri bildirim, gelecekteki sprintler için iyileştirme sağlarken, bir sonraki sprint planlama toplantısına da rehberlik eder (Webb, 2015; Schwaber ve Sutherland, 2020). Sprint değerlendirme, bir aylık sprintler için en fazla dört saat sürer ve daha kısa sprintler için süre daha azdır (Schwaber ve Sutherland, 2020). Toplantıda, ürünün gösterilmesine odaklanır ve sunum araçlarının kullanımı önerilmez (Larman, 2003). Diğer çevik yöntemlerdeki yineleme gösterimi pratiğiyle aynı anlama gelmektedir.

2.4.19. Retrospektif (Retrospective)

Retrospektif toplantılar, sprint değerlendirme toplantısından sonra yalnızca ekip üyeleri arasında kaliteyi ve verimliliği artırmaya yönelik yöntemler planlamak için gerçekleştirilen özel değerlendirme etkinlikleridir (Tripp vd., 2016; Schwaber ve Sutherland, 2020). Bu toplantıların temel amacı, teslimattan ziyade ekip süreçlerini

geliştirmektir. Ekibin önceki çalışmalardan öğrenmesini ve küçük, uygulanabilir iyileştirmeler yapmasını sağlamak amacıyla gerçekleştirilen retrospektif toplantıları kısa ve öz tutulur (PMI ve Agile Alliance, 2017).

Toplantı sırasında her ekip üyesi, bir sonraki sprint döngüsü için iyileştirilebilecek bir veya iki görevle ilgili maddeyi belirler (Webb, 2015). Ayrıca ekip, önceki yinelemede kullanılan süreçleri değerlendirir, süreçle ilgili önerilerde bulunur ve bir sonraki yineleme için gerekli görülen iyileştirmeleri benimser (Tripp vd., 2016). Gerekirse ilgili iyileştirmeler bir sonraki sprint için sprint iş listesine eklenir (Schwaber ve Sutherland, 2020).

2.4.20. Birlikte Oturma (Sit Together)

Çevik ekiplerin tek bir odada ya da ortak bir alanda birlikte çalışabilecekleri mekânın oluşturulmasıdır (Williams, 2010; Martin, 2014; PMI ve Agile Alliance, 2017). Birlikte oturma pratiğini benimseyen ekipler, genellikle tek bir açık alanda veya sosyal/ortak alan olarak adlandırılan bir mekânda çalışmaktadır (Beck, 1999). Bu alanlarda masalar üzerinde birden fazla iş istasyonu yer almakta ve her iş istasyonunda eşli çalışacak programcılar için iki sandalye bulunmaktadır. Ayrıca duvarlarda durum tabloları, görevler ve UML diyagramları gibi ekip işlerini görselleştiren materyaller bulunmaktadır (Martin, 2014).

Birlikte oturma, ekip üyeleri arasındaki iletişimi kolaylaştırarak bilgi paylaşımını artırmayı hedeflemektedir. Bu tür alanlar, ekip üyelerinin aynı ortamda bulunmasını sağlayarak iletişimi kolaylaştırır ve iş birliğini artırır (Beck ve Andres, 2004). Ayrıca bu düzenlemelerin yazılım ekiplerinin verimliliğini iki katına çıkarabileceği belirtilmiştir (Williams, 2010).

Birlikte oturmanın yanı sıra ekip üyelerinin telefon görüşmeleri yapma, e-posta kontrolü veya sessiz bir ortamda çalışabileceği özel çalışma alanları da bulunmalıdır. Ekip üyelerinin periyodik olarak özel alana ihtiyaç duyabileceği göz önünde bulundurularak çalışma ortamı düzenlenmelidir. Bu nedenle ortak çalışma alanları ile sessiz veya bireysel çalışma alanları arasında bir denge kurulması önerilmektedir (PMI ve Agile Alliance, 2017; Williams, 2010).

2.4.21. Açık Çalışma Alanı (Open Workspace)

Açık çalışma alanı, birlikte oturma (bkz. 2.4.20 Birlikte Oturma) pratiğiyle eş anlamlıdır. Ekip üyeleri, çevresinde küçük bölmeler bulunan geniş bir odada çalışır. Geliştiriciler, odanın çevresine yerleştirilmiş bireysel çalışma istasyonlarında ve merkeze konumlandırılmış ortak geliştirme makinelerinde eşli programlama yaparak faaliyet gösterebilmektedir (Beck, 1999; Cohen vd., 2004).

2.4.22. Sürekli entegrasyon (Continuous Integration)

Sürekli entegrasyon, yazılım geliştirme sürecinde, ekip üyelerinin ürettikleri çıktıları düzenli aralıklarla birleştirerek otomatik testler ile doğrulamasını içeren bir uygulamadır. Bu pratikte, ekip üyeleri genellikle günde en az bir kez kodlarını sisteme entegre eder. Her entegrasyon, tüm otomatik testlerin çalıştırıldığı bir derleme süreciyle doğrulanır. Bu yaklaşım, entegrasyon hatalarının erken tespit edilmesini sağlar ve sistemdeki diğer bileşenlerle uyumluluğun korunmasını destekler (Beck, 1999; Martin, 2014; PMI ve Agile Alliance, 2017).

Sürekli entegrasyon süreci, geliştiricilerin kodlarını sıkça entegre etmesini ve entegrasyon sonrası sistemin sıfırdan inşa edilmesini içerir. Eğer entegrasyon sırasında testler geçilmezse, yapılan değişiklikler iptal edilir. Bu işlem, genellikle 7/24 çalışan otomatik bir döngü içinde gerçekleştirilir ve birim testleri ile kabul testlerinin tamamı veya büyük bir kısmı çalıştırılır. CruiseControl ve Anthill gibi Ant tabanlı açık kaynak araçlar, bu süreci destekleyen popüler araçlardandır (Larman, 2003).

Sürekli entegrasyonun temel faydası, entegrasyon sorunlarının riskini azaltmaktır (Williams, 2010). Ayrıca hata bulma ve düzeltme süreçlerinde, birleşik entegrasyonlar arasındaki farkların küçültülmesi sayesinde karmaşıklık azalır. Ancak, sistemin büyüklüğüne bağlı olarak tüm testlerin çalıştırılması zaman alıcı olabilir. Bu nedenle Beck, günlük derleme önerse de bazı durumlarda bu süreç haftalık olarak gerçekleştirilir (Beck, 1999; Xu, 2009).

2.4.23. Kod Sahipliği (Code Ownership)

Kod sahipliği, belirli bir kodun veya yazılım bileşeninin bir geliştirici tarafından sahiplenilmesini gerektirir ve genellikle geliştiricinin uzmanlık alanına göre atanır. Nesne yönelimli dillerde çalışan ekiplerde, sınıfların sahipliği genellikle belirli geliştiricilere verilir. Bir ekip üyesi, sahip olmadığı bir kodda değişiklik yapılması gerektiğinde, değişikliği gerçekleştirmek için kod sahibine başvurmalıdır (Williams, 2010). Kod sahipliği, yalnızca kaynak kodunun değil, aynı zamanda tasarım modelleri, dokümantasyon ve test planları gibi eserlerin kalitesinden de sorumlu olmayı içerir (Nordberg, 2003).

Kod sahipliği pratiğinin avantajları arasında net bir vizyon sağlama, uzmanlaşma, iş ürününe bağlılık ve teknik mükemmeliyet bulunurken; dezavantajları arasında kodla ilgili bilgi bir veya birkaç geliştiriciyle sınırlı olduğu için ekip, bu geliştiricilerin ayrılması durumunda büyük bir riskle karşı karşıya kalması ve kod sahibinin müsait olmaması durumunda yaşanan gecikmeler yer alır (Williams, 2010).

Kod sahipliğinin tersi olan ortak kod sahipliği, bir ekipteki tüm üyelerin herhangi bir kod parçasını düzenleme ve değiştirme hakkına sahip olduğu bir stratejidir. Çevik ekipler, proje gereksinimlerine ve ekip yapısına bağlı olarak kod sahipliği veya ortak kod sahipliği stratejisini tercih edebilirler (Williams, 2010).

2.4.24. Ortak (Kollektif) Kod Sahipliği (Collective Ownership)

Ortak kod sahipliği, tüm ekip üyelerinin proje iş ürünleri veya teslim edilebilirler üzerinde değişiklik yapma yetkisine sahip olduğu, sorumluluğu ekip genelinde paylaşmayı ve iş birliğini teşvik etmeyi amaçlayan bir tekniktir (PMI ve Agile Alliance, 2017). Kod tüm geliştiricilerin mülkiyetindedir ve ihtiyaç duyduklarında kodun herhangi bir yerinde değişiklik yapabilirler (Beck, 1999; Cohen vd., 2004). Ortak kod sahipliği, ekip üyelerine kodun herhangi bir bölümünde değişiklik yapma “yetkisi” verir ve ekip üyelerinin yüksek kaliteli tasarımlar, kod ve test senaryoları üretmek için iş birliği içinde çalışmasını sağlar (Highsmith, 2002).

Ortak kod sahipliği, tüm kodun ekip tarafından kolektif olarak sahiplenildiği bir yazılım geliştirme yaklaşımıdır. Bu pratik ile zaman çizelgeleri ve sorumluluklar, her ekip

üyesinin tüm alt sistemlere katkıda bulunma fırsatına sahip olmasını sağlayacak şekilde düzenlenir. Ekip üyeleri, ihtiyaç duyulduğunda alt sistemler arasında serbestçe çalışabilirler (Nordberg, 2003). Kolektif sahiplik, sürekli entegrasyon, birim testi, kodlama standartları ve eşli programlama gibi diğer XP pratikleriyle desteklenmelidir (Williams, 2010).

Bu yaklaşımın avantajları arasında, sistem bilgisinin ekip içinde dağıtılması, ekip içi etkileşimin artması, kritik ekip üyelerinin yokluğunda bile değişikliklerin hızlı yapılabilmesi ve değişen gereksinimlere etkili yanıt verme kabiliyeti yer almaktadır (Williams, 2010).

2.4.25. Kodlama Standartları (Coding Standards)

Kodlama standartları, yazılım geliştirme süreçlerinde ekip üyelerinin koda tutarlı bir şekilde katkı yapmasını sağlayan ve yazılım kalitesini artıran önemli bir pratiktir. Uç programlama gibi çevik yöntemlerde kodlama standartları, diğer pratiklerle entegre bir şekilde çalışarak eşli programlama ve kolektif kod sahipliği gibi yaklaşımların başarıyla uygulanmasını destekler. Bu standartlar, ekip üyelerinin ortak bir dil kullanarak çalışmasına olanak tanır, kodun okunabilirliğini artırır ve değişikliklerin izlenebilirliğini kolaylaştırır (Highsmith, 2002).

Kodlama standartları, tüm ekip üyelerinin sistemi tutarlı bir şekilde geliştirmesini ve kodun sürdürülebilirliğini sağlamayı amaçlar. Özellikle, farklı ekip üyelerinin sistemin farklı bölümlerini kodladığı, ortak bir kod tabanı üzerinde çalıştığı ve birbirlerinin kodunu gözden geçirdiği ekiplerde bu standartlar bir gereklilik haline gelir (Abrantes ve Travassos, 2011). Bu standartlar, sadece ekip içi iletişimi kolaylaştırmakla kalmaz, aynı zamanda kodun anlaşılabilirliğini ve entegrasyon süreçlerini de destekler (Highsmith, 2002).

Kodlama standartları, belirlenen kuralların mevcut kod tabanına uygulanması ve gelecekte eklenen her yeni kod parçasında bu kuralların geçerli olması gerektiği bir süreçle işler. Bu süreç, statik kod analizi araçları ve kod inceleme yöntemleriyle desteklenerek daha etkin hale getirilebilir (Boström vd., 2006). İyi bir kodlama standardı, “bir kez ve yalnızca bir kez” ilkesine dayanır ve gereksiz kod tekrarını önleyerek her gereksinim için en basit çözümü sunmayı hedefler (Beck, 2000).

Kodlama standartları, yazılım geliştirme ekiplerinin programlama tarzları arasındaki farklılıkların yol açtığı zorlukları en aza indirir. Ekibin uzlaşısıyla belirlediği kurallar sayesinde kod, kim tarafından yazıldığından bağımsız olarak herkes tarafından kolayca anlaşılabilir hale gelir. Ayrıca, standartların gönüllü olarak benimsenmesi ve uygulanması, ekip uyumunu ve iş birliğini artırarak yazılım geliştirme süreçlerinin başarısına önemli katkı sağlar (Beck, 2000; Abrantes ve Travassos, 2011).

2.4.26. Yeniden Kod Düzenleme (Refactoring)

Yeniden kod düzenleme, bir yazılımın dış davranışını veya işlevselliğini değiştirmeden iç yapısını iyileştirme amacıyla yapılan küçük, sürekli dönüşümlerden oluşan bir yazılım kalite tekniğidir (Abrantes ve Travassos, 2011; PMI ve Agile Alliance, 2017). Bu pratik, mevcut kodun tasarımını ve anlaşılabilirliğini artırarak bakım kolaylığını ve sürdürülebilirliğini geliştirir. Yeniden düzenleme sırasında yapılan her değişiklik küçük ve önemsiz gibi görünse de bir araya geldiklerinde sistemin genel mimarisi ve tasarımı üzerinde önemli bir etki yaratabilir (Martin, 2014).

Yeniden kod düzenleme süreci, karmaşık ifadelerin basitleştirilmesi, tekrar eden kodların kaldırılması ve ortak çözümlerin yeniden kullanılabilir kod olarak ele alınması gibi yöntemlerle gerçekleştirilir. Yeniden düzenleme sırasında, yazılımın mevcut birim testlerinin tümü başarılı bir şekilde çalışmaya devam etmelidir. Aksi takdirde yapılan düzenleme kodun güvenilirliğini riske atabilir (Abrantes ve Travassos, 2011). Yeniden kod düzenleme, yazılımın sürdürülebilir bir şekilde geliştirilmesine katkı sağlarken, değişime yanıt verme yeteneğini de artırır (Highsmith, 2002).

Yeniden kod düzenleme, yeni işlevsellik eklenmeden önce mevcut kodun iyileştirilmesini sağlayarak yeni işlevselliğin eklenmesini kolaylaştırır. Bu bağlamda, yeni bir işlevselliğin önerilmesi, genellikle yeniden düzenleme için bir motivasyon kaynağıdır (Highsmith, 2002). Artımlı tasarım pratiği uygulandığında, önceden yazılmış kodun yeniden düzenlenmesi, yazılımın tasarımını sürekli olarak geliştirmek için kritik öneme sahiptir (Williams, 2010). Yazılım geliştirme süreçlerinde kaliteyi artırmak ve uzun vadede bakım maliyetlerini düşürmek (bakım kolaylığı sayesinde) amacıyla kullanılan temel bir pratiktir.

2.4.27. Test Etme (Testing)

Yazılım testi, bir yazılım sisteminin spesifikasyonuna uygun olup olmadığını ve hedeflendiği ortamda çalışıp çalışmadığını belirlemek için çalıştırılması sürecidir (Whittaker, 2000). Testler, ürünün çalıştığından emin olmak için gerçekleştirilir (Xu, 2009). Çevik yazılım geliştirmede test sürecinin temel unsurlarından biri test pratiğidir. Yazılım testleri, yazılım geliştirme sürecinin hangi aşamasında gerçekleştirildiğine göre birim, entegrasyon ve sistem testi olarak sınıflandırılır (Whittaker, 2000). Sistem seviyesi testleri, yapı taşları için birim testleri kullanılmaktadır. Ara seviyelerde entegrasyon testlerinin gerekliliği değerlendirilerek, bu testlerin hangi aşamalarda ve hangi koşullarda uygulanması gerektiği analiz edilerek kullanılabilir. Testlerin ne zaman ve hangi durumda yapılacağını belirlemek, ürün kalitesini iyi bir performansla sürdürmeye yardımcı olur. Çevik ekipler, teslimat hızını korumak için otomatik testleri tercih etmektedir (PMI ve Agile Alliance, 2017).

Çevik yöntemlerin bazılarında test uzmanları bulunmaktadır (Williams, 2010). Projelerdeki test uzmanları, ürünün çalışacağına dair güven taşırlar ve bu güven, program ilerledikçe artar. Test uzmanları, ürünün kalitesiz olduğundan şikâyet etmek yerine, geliştiricilerle birlikte olası kusurları bulup düzeltmek için çalışırlar. Bu nedenle, test uzmanları ile geliştiriciler arasındaki iletişim çok iyidir ve aralarındaki geri bildirim süreci oldukça etkilidir. Projelerinde yazılan testler mümkün olduğunca izole olmalı ve otomatik hale getirilmelidir. İzole testler sayesinde test uzmanları veya geliştiriciler, kusurları anında ve doğrudan tespit edebilirler. Bu sayede, bir yığın çıktı mesajı arasından kusur aramaları gerekmez (Xu, 2009).

2.4.28. Önce Test Programlama (Test-First Programming)

Yazılım geliştirme süreçlerinde testler, genellikle kodlama işleminin ardından gerçekleştirilir. Birim testleri, işlevsellik kodlanırken programcılar tarafından uygulanır. Daha sonra, yeni eklenen veya değiştirilmiş bileşenlerin sistemle uyumluluğunu doğrulamak amacıyla entegrasyon testleri yapılır. Nihayetinde, müşterinin ihtiyaçlarına uygun bir nihai ürün sağlamak için sistem testleri gerçekleştirilir. Entegrasyon ve sistem testleri genellikle kalite güvencesi ekipleri tarafından tasarlanır ve yürütülür (Xu, 2009).

Çevik yöntemler ise bu geleneksel yaklaşımın aksine, yazılım geliştirme alanında devrim niteliğinde kodlamadan “önce test” uygulamasını benimser. Önce test programlama pratiğinde testler, tüm üretim kodunun başarısız olan birim testlerini geçmesini sağlamak amacıyla önceden yazılır. İlk olarak, yazılım sistemindeki herhangi bir kodu değiştirmeden başarısız olacak (test edilen işlev henüz var olmadığı için) bir birim testi oluşturulur. Daha sonra, bu testi geçmek için gerekli olan kod yazılır. Test yazma ve kod geliştirme döngüsü oldukça hızlıdır ve genellikle bir dakikadan kısa sürer. Sınırlı kapsamı nedeniyle, bu testler genellikle çok hızlı çalışır. Bu yöntem, testlerin kod geliştirme sürecine neredeyse eş zamanlı olarak dahil edilmesini sağlayarak, test senaryoları ile kodun birlikte evrimleşmesini mümkün kılar (Beck ve Andres, 2004; Martin, 2014).

2.4.29. Kabul Testi Odaklı Geliştirme (Acceptance Test-Driven Development)

Kabul testi, bir sistemin belirlenen kabul kriterlerini karşılayıp karşılamadığını değerlendirmek amacıyla gerçekleştirilen resmi bir test sürecidir. Bu süreç, kullanıcı, müşteri veya yetkilendirilmiş bir tarafın, sistemin ya da bileşenin gereksinimlere uygunluğunu doğrulayarak kabul edilip edilmeyeceğine karar vermesini sağlar (IEEE, 1990).

Kabul testi odaklı geliştirme (KTOG), kabul testlerinin ürün yöneticisi/müşteri, yazılım testçisi, yazılım geliştirici ve kullanıcı arayüzü tasarımcısının iş birliği içinde, yinelemenin başında gerçekleştirdiği bir pratiktir. Böylece istenen işlevsellik hakkında daha fazla bilgiyle geliştirme süreci ilerleyebilir (Williams, 2010). KTOG’de, tüm ekip bir araya gelerek bir iş ürünü için kabul kriterlerini tartışır. Daha sonra ekip, kriterleri karşılayacak yeterli kod ve otomatik testleri yazmalarını sağlayacak testleri oluşturur (PMI ve Agile Alliance, 2017). Bu testler, sistemin çalıştırılabilir bir parçası haline gelir ve yeni işlevselliğin daha önce çalışan işlevselliği bozup bozmadığını kontrol etmek için periyodik olarak çalıştırılabilir (Williams, 2010). Sistemin ya da bileşenin gereksinimlere uygunluğunu test edilerek kabul edilip edilmeyeceğine karar verilir.

2.4.30. Birim Testi (Unit Test)

Birim testi, yazılım geliştirme süreçlerinde bireysel yazılım bileşenlerinin veya bir bileşen koleksiyonunun belirlenen gereksinimlere uygunluğunu doğrulamak amacıyla gerçekleştirilen bir test türüdür (Whittaker, 2000). Birim testleri, işlevsellik kodlanırken programcılar tarafından uygulanır (Xu, 2009). Programcılar dakika dakika birim testleri yazarlar (Beck, 1999). Bu testlerde, test edilen bileşenlerin giriş ve çıkış alanlarına odaklanılırken, sistemin diğer kısımları göz ardı edilir (Whittaker, 2000).

Birim testi odaklı geliştirme, kısa süreli ve tekrar eden bir “test->kod” döngüsü izlenmesini teşvik etmektedir. Bu yaklaşımda, testler uygulama kodundan önce yazılır (Williams, 2010; Beck, 1999). Bu yöntem, yazılımın sürekli olarak doğrulanmasını sağlayarak kodun kalite ve güvenilirliğini artırır (Williams, 2010). Çevik metodolojilerin benimsenmesiyle bu pratik daha yaygın ve sistematik hale gelmiştir (Beck, 1999). Ayrıca birim testlerinin, yineleme süreçlerinde yazılımın ilerleyişini sürekli olarak değerlendirme ve onaylama olanağı sağladığı belirtilmektedir (Xu, 2009).

2.4.31. Yerde Müşteri (On-site Customer)

Yerde müşteri, yazılım geliştirme süreçlerinde müşterinin ekibin bir parçası olarak tam zamanlı katılımını ifade eder ve uç programlama yönteminin ve diğer çevik yöntemlerin temel unsurlarından biridir (Beck, 1999; Xu, 2009). Bu pratik, müşteri ile geliştiriciler arasında doğrudan iletişim sağlayarak gereksinimlerdeki belirsizlikleri veya sorunları hızla çözme olanağı sunar (Anderson, 2004).

Müşteri, ekibin bir parçası olarak sürekli soruları yanıtlamak, anlaşmazlıkları çözmek ve küçük ölçekli öncelikleri belirlemek gibi kritik roller üstlenir. Yerde müşteri, diğer müşteriler ve programcılar arasında bir köprü veya aracı görevi görebilir. Müşteri, projeyi yönetir ve başarıya ulaşmasını sağlar (Xu, 2009). Müşterinin projeye doğrudan dahil edilmesi, gereksiz çabayı azaltarak ekip içi güveni artırır ve sürekli iyileşmeyi teşvik eder (Beck ve Andres, 2004).

Yerde müşteri, tamamlanmamış kodun demolarını inceleyerek hızlı geri bildirim sağlayabilir. Bu durum, kullanıcı kabul testinin bir kaynak olarak kullanılmasını destekler

ve geliştiricilerin seviyesini yükseltir. Sonuç olarak, daha az kusur ve daha yüksek kaliteye sahip bir ürün olur ve bu da sistemin genel üretim hızını artırır (Anderson, 2004).

Gerçek bir müşteri ya da müşteri temsilcisinin eksikliği durumunda, kullanılmayan özelliklerin geliştirilmesi, kabul kriterlerini yansıtmayan testlerin belirlenmesi ve projenin farklı bakış açılarını birleştirme fırsatının kaybedilmesi gibi riskler ortaya çıkabilir (Beck ve Andres, 2004). Yerinde müşteri, müşteriye ekibin bir parçası haline getirilerek projeye güven sağlar ve yazılımın başarısını artırır (Beck ve Andres, 2004; Xu, 2009).

2.4.32. Müşteri Katılımı (Customer Involvement)

Müşteri katılımı, uç programlamadaki (bkz. 2.4.31 Yerinde Müşteri) yerinde müşteri pratiğinin diğer çevik yöntemler için kullanılan şeklini ifade etmektedir.

2.4.33. Eşli Programlama (Pair Programming)

Eşli programlama, iki yazılım geliştiricinin tek bir bilgisayar başında yan yana oturarak aynı iş ögesi üzerinde sürekli iş birliği yapmasını sağlayan bir yazılım geliştirme pratiğidir (PMI ve Agile Alliance, 2017). Tüm üretim kodu, bir ekran/klavye/farede iki kişi tarafından yazılır (Beck, 1999)

Eşli programlamada iki kişi aynı tasarım, algoritma, kod veya test üzerinde birlikte çalışır (Beck ve Andres, 2004). Eşlerden biri klavyeyi kullanarak kodu yazarken, diğeri yazılan kodu izleyerek hata ve iyileştirmelere dikkat eder. Bu iki kişi yoğun bir şekilde etkileşim içindedir ve ikisi de yazılım geliştirme sürecine tamamen dahil olur (Martin, 2014).

Eşli programlama, sürücü (driver) ve yol gösterici (navigator) olarak iki temel rolü barındırır. Sürücü, bilgisayarda kodlama veya tasarım kaydı yaparken; yol gösterici hem taktiksel hem de stratejik hataları tespit etmekle görevlidir. Taktiksel hatalar, sözdizimsel veya yazım yanlışları gibi küçük hataları kapsamaktadır. Stratejik hatalar, uygulamanın veya tasarımın hedeflerine ulaşmasını engelleyebilecek daha büyük sorunlardır. Yol gösterici, programcı çiftin daha uzun vadeli düşüneni olarak, işe daha objektif bir açıdan bakabilir ve işin yönü hakkında daha stratejik düşünme yeteneğine sahiptir. İki programcı

sürekli olarak fikir paylaşımında bulunarak alternatif yaklaşımları ve çözümleri tartışılır (Williams, 2010).

Eşli programcılar çeşitli görevler üstlenirler. Birbirlerini görevde tutarak odaklanmayı sağlayarak sürecin verimliliğini artırır. Sistem iyileştirmeleri için beyin fırtınası yaparak yenilikçi çözümler geliştirirler. Fikirleri netleştirerek anlaşmazlıkları ve belirsizlikleri en aza indirirler. Eşleri bir sorunla karşılaştığında inisiyatif alarak sürecin duraksamasını önler ve hayal kırıklığını azaltırlar. Ayrıca, takım pratiklerine bağlı kalınması konusunda birbirlerini sorumlu tutarak yazılım geliştirme standartlarının korunmasını sağlarlar (Beck ve Andres, 2004).

Eşli programlama pratiği, on yıllardır zaman zaman uygulanan bir teknik olmakla birlikte, XP gibi çevik metodolojilerle daha yaygın hale gelmiştir. Eşli programlama, yazılım geliştirme süreçlerinde ürün kalitesini artırma, ekip içi iş birliğini geliştirme, ekip ruhunu geliştirmek, bilgi yönetimini desteklemek ve ürün riskini azaltmak gibi önemli faydalar sağlar (Beck ve Andres, 2004; Martin, 2014).

2.4.34. Enerjik Çalışma (Energized Work)

Enerjik çalışma pratiği, bireylerin ve ekiplerin aşırı fazla mesai yapmaktan kaçınmasını ve sürdürülebilir bir çalışma temposunu benimsemesini önerir (Beck ve Andres, 2004; Williams, 2010). Enerjik çalışma pratiği, kodun yüksek kalitede tutulmasını sağlamak, yorgunluk nedeniyle oluşabilecek hataları önlemek ve çalışan memnuniyetini artırarak ekip içi sirkülasyonu azaltmak amacı taşır (Williams, 2010).

Fazla mesai, yalnızca gerekli olduğunda ve izole durumlarda uygulanmalıdır. Ancak, art arda iki hafta fazla mesai yapmak kabul edilemez bir durumdur. Sıkça tekrarlanan fazla mesai, ekip içinde daha derin yapısal sorunların bir göstergesi olabilir ve mutlaka ele alınmalıdır (Beck, 1999). Enerjik çalışma pratiği, yalnızca bireylerin üretken olabileceği kadar saat çalışmasını ve bu temponun sürdürülebilir olmasını önerir. Verimsiz bir şekilde fazla çalışmak yerine, mevcut enerjiyi etkili bir şekilde kullanarak ekibin uzun vadeli verimliliği korunur. Bugün fazla çalışıp kendinizi tüketmek, sonraki günlerin verimini düşürerek ekip için zararlı olabilir (Beck ve Andres, 2004).

Enerjik çalışma, farklı kaynaklarda “sürdürülebilir tempo” ve “haftalık 40 saat” olarak ta isimlendirilmiştir

2.4.35. Sürdürülebilir Tempo (Sustainable Pace)

Sürdürülebilir tempo pratiği, enerjik çalışma (bkz. 2.4.34 Enerjik Çalışma) pratiğinin eş anlamlısıdır (Williams, 2010).

2.4.36. Haftalık 40 Saat (40-Hour Weeks)

Haftalık 40 saat pratiği de benzer şekilde enerjik çalışma (bkz. 2.4.34 Enerjik Çalışma) pratiğinin eş değeridir. Haftalık 40 saatten fazla çalışılmamasını belirtmektedir (Beck, 1999).

2.4.37. On Dakikalık Derleme (Ten-Minute Build)

10 dakikalık derleme pratiği, yazılım geliştirme süreçlerinde tüm sistemin derlenmesi ve tüm otomatik testlerin 10 dakika içinde çalıştırılabilmesini hedefleyen bir yaklaşımdır. Bu pratik, geliştiricilerin yaptıkları son değişikliklere hızlı geri bildirim almasını sağlar. Derleme sürecinin hızlı olması, geliştiricilerin testleri daha sık çalıştırmalarına ve yeni kodları daha sık entegre etmelerine olanak tanır. Bu sayede, yeni kodun etkileri hızlı bir şekilde görülür ve entegrasyon süreci kolaylaşır (Williams, 2010).

Pratiğin temel amacı, yazılım sisteminin entegrasyon sürecinde zaman kaybını en aza indirmektir. Tüm sistemi otomatik olarak derlemek ve tüm testleri 10 dakika içinde çalıştırmak, geri bildirim alma süresini optimize eder. Ancak, derleme süreci 10 dakikadan uzun sürerse, bu süreç daha az kullanılabilir hale gelir ve geri bildirim alma fırsatları azalır (Beck ve Andres, 2004).

On dakikalık derleme pratiğinin uygulanması için üç önemli ipucu sunulmaktadır. Öncelikle, tüm sistemi otomatik olarak derleyecek bir süreç oluşturulmalıdır. Ardından, yalnızca değiştirilmiş olan sistem kısmını derlemek için iyileştirmeler yapılabilir. Son olarak, yalnızca yapılan değişiklikler nedeniyle riske giren sistem bölümlerini kapsayan testler çalıştırılabilir (Beck ve Andres, 2004). Bu yaklaşımlar, derleme sürecini daha verimli hale getirerek yazılım geliştirme süreçlerini hızlandırır ve hata oranını düşürür.

2.4.38. Tüm Takım (Whole Team)

Tüm takım pratiği, projede gerekli tüm beceri ve kaynakların bir arada bulunmasını sağlamak ve ekip içinde bir bütünlük hissi yaratmak için geliştirilmiş eski bir kavramdır (Beck ve Andres, 2004). Çevik yöntemleri benimseyen ekipler, ürün geliştirme sürecinde çapraz fonksiyonlu bir yapıyı benimseyerek, geliştiriciler, test uzmanları, ürün yöneticileri, proje yöneticileri, kullanıcı arayüzü tasarımcıları ve dokümantasyon uzmanlarını aynı takımda bir araya getirir. Bu yaklaşım, “biz-onlar” ayrımını ortadan kaldırarak ekip üyelerinin ortak hedeflere ulaşmak için iş birliği yapmasını sağlar (Williams, 2010).

2.4.39. Bilgilendirici Çalışma Alanı (Informative Workspace)

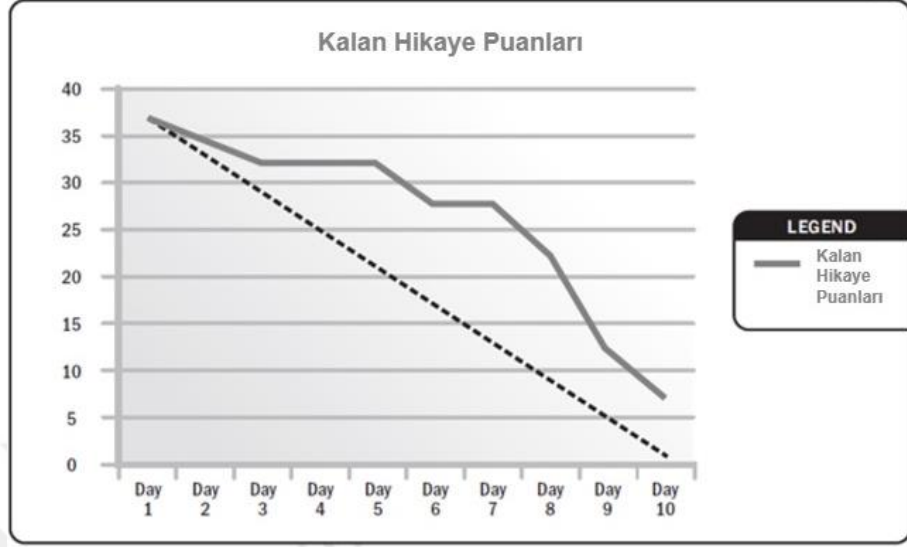
Bilgilendirici çalışma alanı, ekip üyelerinin veya ilgili bir gözlemcinin ekip alanına girdiğinde projenin ilerleyişine dair hızlı bir genel fikir edinmesini sağlayan bir yöntemdir. Bu uygulama, çalışma alanını projeye ilgili bilgilere odaklayarak, ekip üyelerinin proje durumunu kolaylıkla anlamalarını hedefler (Beck ve Andres, 2004). Etkili bir bilgilendirici çalışma alanında, proje durumunu ifade eden bilgiler açık, görünür ve kolay erişilebilir olmalıdır (Williams, 2010). Birçok ekip, bu pratiği hikâye kartlarını bir duvara yerleştirerek, yanma grafikleri ve yineleme durum panoları kullanarak bilgilendirici çalışma alanlarını oluşturmaktadır (Beck ve Andres, 2004; Williams, 2010).

2.4.40. İş İlerleme Grafiği- Yanma (Tüketim) Grafiği (Burn Chart)

Projenin zaman içinde nasıl ilerlediğini izlemek amacıyla kullanılan grafiklerdir. Çevik ekipler aşağı tüketim grafiği (burndown chart) ve yukarı tüketim grafiği (burnup down) olarak iki farklı şekilde kullanabilirler (PMI ve Agile Alliance, 2017).

Aşağı tüketim grafikleri, kalan iş miktarını zamana karşı grafiksel bir biçimde temsil ederek projenin ilerleme hızını ve tamamlanması gereken işleri görselleştirir. Dikey eksenle kalan iş miktarını, yatay eksenle ise zamanı gösteren bu grafik, ekip üyelerinin projenin genel durumunu kolayca kavramasını sağlar (Williams, 2010). Şekil 2.27’de ekibin 37 hikâye puanı teslim etmeyi planladığı bir aşağı tüketim grafiği örneği gösterilmektedir. Hikâye puanları, bir gereksinim veya hikâyenin göreceli iş yükü, risk ve karmaşıklığını değerlendirir. Birçok çevik ekip, çabayı tahmin etmek için hikâye

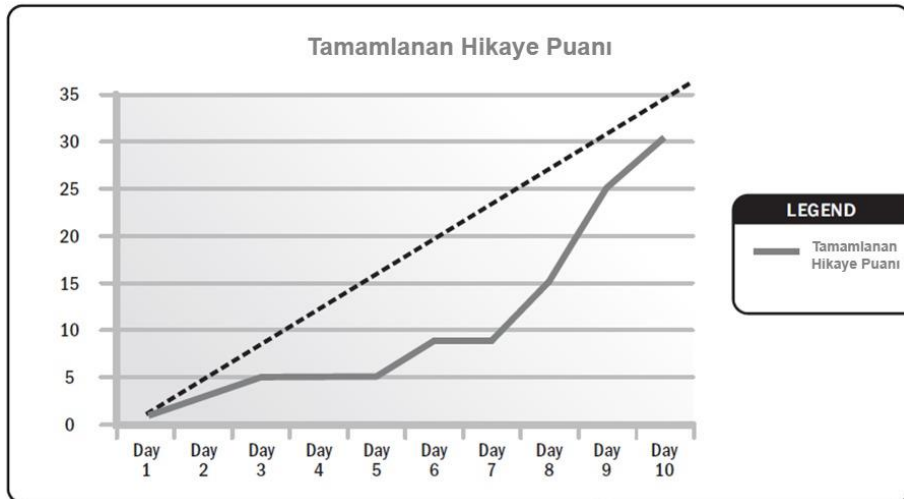
puanlarını kullanır. Kesikli yakma çizgisi, planı gösterir. Ekip 3. gün itibarıyla teslimat için risk altında olduklarını görebilir (PMI ve Agile Alliance, 2017).



Şekil 2.27. Aşağı Tüketim Grafiği

Kaynak: PMI ve Agile Alliance (2017)

Yukarı tüketim grafikleri, tamamlanan işleri görselleştirir. Yukarı tüketim grafiğine bir örnek Şekil 2.28’de sunulmuştur. Şekil 2.27 ve Şekil 2.28’deki grafikler aynı verilere dayanmaktadır ancak farklı biçimlerde sunulmaktadır. Ekipler, verilerini nasıl görüntülemek istediklerine bağlı olarak bu grafik türlerinden birini tercih edebilirler (PMI ve Agile Alliance, 2017).



Şekil 2.28. Yukarı Tüketim Grafiği

Kaynak: PMI ve Agile Alliance (2017)

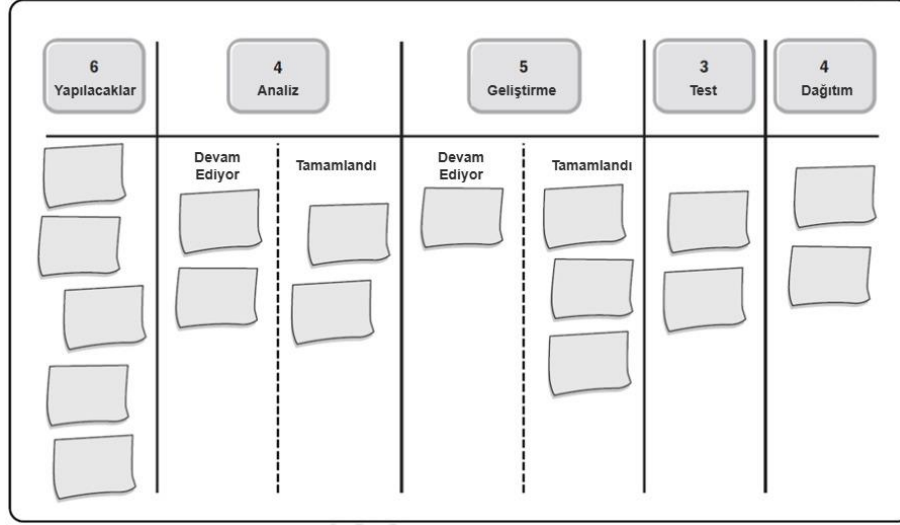
Ekipler, aşağı tüketim veya yukarı tüketim grafikleri kullanarak yineleme boyunca tamamladıkları işleri takip ederler. Yineleme sonunda, bu tamamlanan işlere dayanarak bir sonraki yineleme için kapasite ölçümlemlerini (kaç hikâye veya hikâye puanı tamamlanabileceğini) belirlerler. Bu süreç, ürün sahibinin bir sonraki yineleme için teslim edilme olasılığı daha yüksek olan işleri ekip ile birlikte yeniden planlamasını sağlar (PMI ve Agile Alliance, 2017).

2.4.41. Kanban Panosu ve Scrum Panosu (Kanban Board and Scrum Board)

Yineleme durum panoları, yineleme sürecinde tamamlanması gereken özelliklere ilişkin bilgileri sistematik bir şekilde organize etmek için kullanılan bir araçtır. Özellikler veya kullanıcı hikâyeleri genellikle “yapışkan notlar” üzerine yazılarak panoya eklenir. Henüz üzerinde çalışılmamış özellikler “Yapılacaklar” sütununda yer alırken, bir özelliğin geliştirme süreci başladığında, ilgili not “Devam Ediyor” sütununa taşınır. Çoğunlukla, özelliğin uygulanmasından sorumlu kişi, not üzerinde adını belirtir. Geliştirme tamamlandığında ise söz konusu not, “Tamamlandı” sütununa taşınarak sürecin durumu güncellenir (Williams, 2010).

Kanban yönteminde kullanılan Kanban panoları ve Scrum yönteminde kullanılan Scrum panoları, yineleme durum panolarının uygulamaları arasında yer almaktadır. Bu panolar, projelerde iş akışını görselleştirmek, görevlerin durumunu izlemek ve süreçlerin etkinliğini artırmak amacıyla kullanılan araçlardır (PMI ve Agile Alliance, 2017). Çevik yazılım geliştirme sürecinde kullanılan bu panolar görevlerin, gereksinimlerin veya kullanıcı hikayelerinin tamamının akışını kolay bir şekilde göstererek yazılım geliştirme yönetiminde maksimum görselleştirme sağlar (Sienkiewicz, 2012).

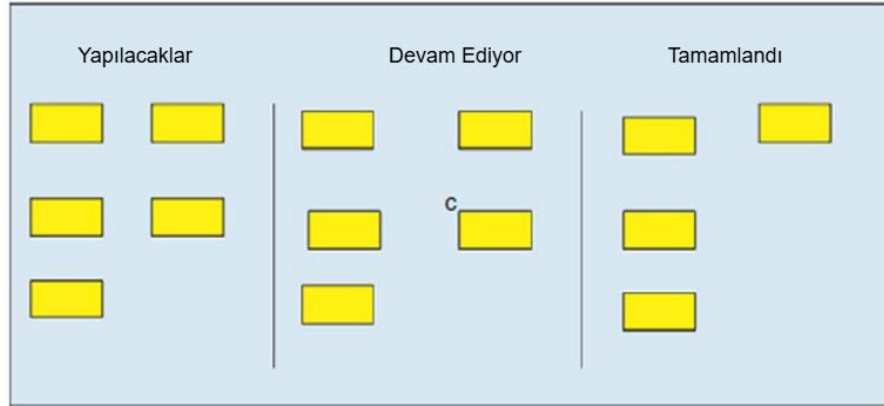
Kanban Panosu: İş akışını sürekli bir şekilde görselleştiren ve darboğazları ile iş yüklerini görünür hale getiren bir sistemdir. Bu özellikleriyle Kanban panoları, iş akışında iyileştirmeler yapılmasına ve süreçlerin optimize edilmesine olanak tanımaktadır. Ayrıca pano bir bilgi yayıcı olarak da işlev görür ve ekibin iş durumuna dair güncel bilgileri gören herkese sunmaktadır. Kanban panosu örneği Şekil 2.29’da sunulmuştur.



Şekil 2.29. Kanban Panosu

Kaynak: PMI ve Agile Alliance (2017)

Scrum Panosu: Ürün ve sprint iş listelerini yönetmek, iş akışını izlemek ve darboğazları tespit etmek için kullanılan bir bilgi yayma aracıdır. Özellikle yineleme odaklı süreçlerde görevlerin organizasyonunu sağlayarak ekiplerin sprint boyunca ilerlemelerini etkili bir şekilde yönetmelerine olanak sunar. Şekil 2.30'da sprint çalışmasının görselleştirilmesi sağlayan örnek bir scrum panosu sunulmuştur (Williams, 2010).



Şekil 2.30. Scrum Panosu

Kaynak: Williams (2010)

Scrum ve Kanban panolarının benzerlikleri ve farklılıklarını göstermek için Şekil 2.31'de bir örnek sunulmuştur. İki pano arasındaki temel farklılık, kanban panosuna eklenen ve “devam eden işler” (Work In Progress - WIP) olarak adlandırılan satırdır. WIP satırı, aynı anda üzerinde çalışılabilecek maksimum iş (görev) miktarını belirleyerek

optimum verimliliği sağlamayı amaçlamaktadır (Sienkiewicz, 2012; Cole ve Scotcher, 2015). Bu satırın, özellikle kanban yazılım geliştirme yöntemi içinde kritik bir öneme sahip olduğu belirtilmektedir (Sienkiewicz, 2012).

Scrum Panosu					Kanban Panosu				
Aşamalar	Ürün İş Listesi	Sprint İş Listesi			Aşamalar	Ürün İş Listesi	Sprint İş Listesi		
	Yapılacaklar	Başlanmadı	Devam Ediyor	Tamamlandı		Yapılacaklar	Başlanmadı	Devam Ediyor	Tamamlandı
Görevler	Görev		Görev	Görev	WIP			3	
	Görev	Görev	Görev	Görev	Görevler	Görev		Görev	Görev
	Görev	Görev	Görev	Görev		Görev	Görev	Görev	Görev
	Görev		Görev	Görev		Görev	Görev	Görev	Görev
	Görev					Görev		Görev	Görev

Şekil 2.31. Scrum Panosu ve Kanban Panosunun Örneği – Benzerlikler ve Farklılıklar
Kaynak: Sienkiewicz (2012)

ÜÇÜNCÜ BÖLÜM

ÇEVİK YAZILIM GELİŞTİRME

PROJE BAŞARISI ve KRİTİK BAŞARI FAKTÖRLERİ

3.1. YAZILIM PROJESİ

Proje, belirli bir süreyle sınırlı olarak, benzersiz bir ürün, hizmet veya sonuç oluşturmak amacıyla gerçekleştirilen geçici bir çabadır. Projeler, belirli amaçlara ulaşmak için teslim edilebilirler üreterek yürütülür. Amaç, çalışmaların yönlendirildiği hedeflenen bir sonuç olup, ulaşılması gereken stratejik bir konum, gerçekleştirilmesi gereken bir görev, elde edilmesi gereken bir çıktı, üretilmesi gereken bir ürün veya yerine getirilmesi gereken bir hizmet şeklinde tanımlanır. Teslim edilebilirler ise bir süreci, aşamayı veya projeyi tamamlamak için oluşturulması gereken benzersiz ve doğrulanabilir ürünler, sonuçlar veya hizmetler anlamına gelir. Teslim edilebilirler somut (örneğin, fiziksel bir ürün) veya soyut (örneğin, bir hizmet veya bilgi) nitelikte olabilir (Project Management Institute [PMI], 2017). Yazılım ürünleri, soyut nitelikte teslim edilebilirler arasında yer alır ve bu özellikleriyle projelerin dijital çerçevede benzersizliğini vurgular. Genel proje yönetiminde kullanılan birçok teknik, yazılım ürünlerinin oluşturulmasında da uygulanabilir (Hughes vd., 2011). Yazılım projeleri, diğer proje türlerinden ayrılan kendine özgü özelliklere sahiptir (Nizam, 2015). Bu özellikler; görünmezlik, karmaşıklık, uyumluluk ve değişkenlik olarak sıralanabilir (Hughes vd., 2011; Nizam, 2015).

Görünmezlik: Fiziksel projelerde, örneğin bir köprünün inşasında ilerleme gözle görülür bir şekilde izlenebilir. Ancak yazılım projelerinde ilerleme hemen görünmez. Bu nedenle yazılım projeleri, “görünmeyeni görünür hale getirme” süreci olarak tanımlanır.

Karmaşıklık: Yazılım ürünleri, birçok birbiriyle bağlantılı bileşenden oluştuğu için diğer mühendislik ürünlerine kıyasla daha fazla karmaşıklık içerir. Harcanan maliyet başına yazılımın sunduğu karmaşıklık seviyesi oldukça yüksektir.

Uyumluluk: Geleneksel mühendislik genellikle fiziksel malzemeler (örneğin, çimento ve çelik) ile çalışır ve bu malzemeler tutarlı fiziksel yasalarla yönetilir. Ancak yazılım geliştirme süreci, müşterilerin değişken gereksinimlerine uyum sağlamak zorundadır. Bu durum, insan faktörünün karmaşıklığını projeye taşır.

Değişkenlik: Yazılımın kolayca değiştirilebilmesi bir avantaj olarak görülse de yazılım projeleri, özellikle fiziksel veya organizasyonel sistemlerle entegre olduğunda değişim baskısına daha fazla maruz kalır. Bu durum, yazılımın diğer bileşenlere uyum sağlamak için sürekli değişime uğramasını gerektirir.

Yukarıda bahsedilen özellikler, yazılım projelerinin daha karmaşık ve dinamik bir yapıya sahip olmasına neden olmaktadır (Hughes vd., 2011). Yazılım projelerinin karmaşıklığı ve dinamik yapısı, yazılım projelerinin tanımını net bir şekilde yapmayı önemli hale getirmektedir.

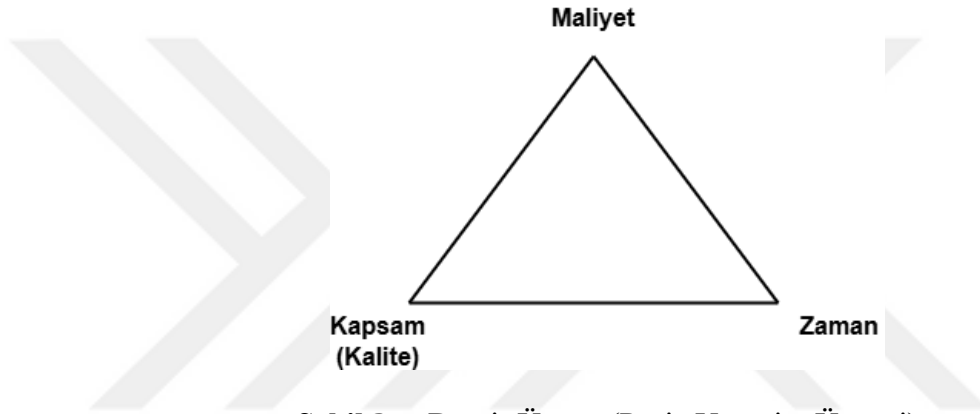
Yazılım projesi, belirli bir ihtiyaç veya sorunu analiz ederek, bunu elektronik ortamda karşılamak için geliştirilen, belirli bir geliştirme süresi olan ve daha önce var olmayan bir yazılım geliştirme çabasıdır (Nizam, 2015). Bir yazılım geliştirme projesi, birden fazla kişinin zaman, bütçe ve insan kaynağı gibi belirli kısıtlar çerçevesinde yürüttüğü, karmaşık ve organize bir çalışmadır. Bu tür projeler, mevcut bir iş sürecine ya da yeni oluşturulan bir sürece kayda değer iş değeri eklemeyi amaçlayan yeni ya da geliştirilmiş bilgisayar kodlarının üretilmesini içerir (Wysocki, 2006). Yazılım projelerinin geliştirilmesi sürecinde farklı yaklaşımlar ve metodolojiler kullanılmaktadır (bkz. İkinci Bölüm: Yazılım Geliştirme Metodolojileri). Bu metodolojiler, yazılım geliştirme sürecinin planlanması, yönetilmesi ve yürütülmesinde önemli bir rol oynamaktadır.

3.2. YAZILIM PROJE BAŞARISI

Proje başarısının yaygın olarak kabul edilen bir tanımı ve bir projenin ne zaman ve nasıl başarılı veya başarısız olarak adlandırılacağı konusunda fikir birliği yoktur (Ika, 2009; Hughes vd., 2011). Ancak proje başarısı geleneksel olarak demir üçgen (iron triangle) ile tanımlanmaktadır (Atkinson, 1999). Demir üçgen, literatürde altın üçgen (golden triangle), üçlü kısıtlama (triple constraint) veya proje yönetim üçgeni (project management triangle) olarak da adlandırılan, proje başarısını değerlendirmek için kullanılan temel bir kavramdır. Bu kavram, proje başarısının; belirlenen zaman çizelgesine uyum, bütçe sınırları dahilinde tamamlama ve kabul edilen kalite, performans veya kapsam seviyelerinin karşılanması ile ölçüldüğünü ifade eder. Demir üçgen, proje

performansının değerlendirilmesi için standart bir çerçeve olarak kullanılmaktadır (Drury-Grogan, 2014; Pollack vd., 2018; Pinto, 2020).

Demir üçgen, Şekil 3.1’de sunulmuştur. Bu modelde kapsam, müşterinin talepleri doğrultusunda şekillenerek projenin sınırlarını belirler. Maliyet, projenin gerçekleşmesi için gerekli insan kaynakları, bütçe, yazılım geliştirme araçları, donanım, mekân gibi ihtiyaçların toplam bedelini ifade eder. Zaman ise projenin tamamlanma ve teslim edilme süresini temsil eder. Kapsam, zaman ve maliyet, birbirine hassas bir dengeyle bağlıdır. Bu unsurlardan birindeki değişiklik, diğerlerinde de değişiklik yapılmasını gerektirir (Nizam, 2015).



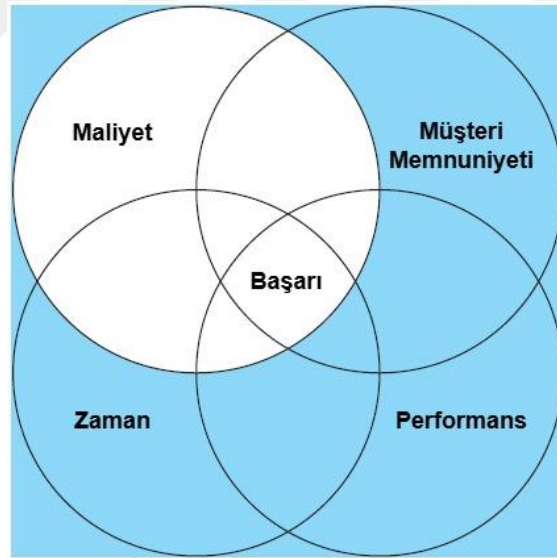
Şekil 3.1. Demir Üçgen (Proje Yönetim Üçgeni)
Kaynak: Atkinson (1999)

Şekil 3.1’de gösterildiği gibi, kapsam ve kalite, demir üçgen modelinde genellikle aynı unsur altında değerlendirilir. Chow ve Cao (2008) tarafından gerçekleştirilen çevik yazılım projelerindeki kritik başarı faktörlerine ilişkin araştırmada, başarı ölçütleri zaman, maliyet ve kapsamın yanı sıra kaliteyi de içerecek şekilde genişletilmiştir. Çalışmada kalite ayrı bir proje başarı unsuru olarak belirtilmiştir.

Pinto (2020) projeyi, belirli bir başlangıcı ve sonu olan, insanlar tarafından yürütülen ve belirlenmiş maliyet, zaman ve kalite parametreleri çerçevesinde belirlenen hedeflere ulaşmayı amaçlayan benzersiz bir girişim olarak tanımlanmıştır. Bu tanım, proje başarısında maliyet, zaman ve kalite yönetiminin kritik bir öneme sahip olduğunu vurgulamaktadır. Başarılı bir proje, başlangıçta belirlenen maliyet, zaman çizelgesi, kalite ve işlevsellik hedeflerini karşılamalı ve bunu yaparken kaynakların verimli bir şekilde kullanılmasını sağlamalıdır (Vithara vd., 2015). Henriksen ve Pedersen (2017) çevik yöntemlerin proje kapsamının proje süresince esnek bir şekilde tanımlanmasına imkân

sağladığını ve yazılım projelerinin başarısını değerlendirirken kapsam, zaman, maliyet ve kalitenin önemli kriterler olduğunu vurgulamıştır. Shakya ve Shakya (2020), yazılım projelerinin kalite, kapsam ve zaman çizelgesinden ödün vermeden sınırlı bir bütçe ile tamamlanmasının, proje başarısının anahtarı olduğunu belirtmektedir.

Günümüzde proje başarısının yalnızca geleneksel “üçlü kısıtlama” ile değerlendirilmesinin yetersiz olduğu belirtilmektedir. Üçlü kısıtlama; zaman, bütçe ve performans kriterlerinden oluşmaktadır. Performans kriteri, projenin kapsamını ve kalitesini ifade etmektedir. Pinto (2020), bu kriterlere müşteri kabulü (müşteri memnuniyeti) eklenerek daha kapsamlı bir değerlendirme yapılması gerektiğini belirtmiştir. Müşteri memnuniyeti, projelerin yalnızca teknik hedeflere ulaşmasını değil, aynı zamanda müşterilerin ihtiyaçlarını ve beklentilerini karşılamasını da zorunlu kılmaktadır. Böylece müşteri memnuniyeti, zaman, bütçe ve performansa eklenen dördüncü bir kriter olarak ele alınmıştır. “Yeni Dörtlü Kısıtlama” proje yönetiminde daha bütünsel bir yaklaşımı teşvik ederek, başarı kriterlerini daha kapsamlı ve sürdürülebilir bir çerçevede ele alır (Pinto, 2020).



Şekil 3.2. Yeni Dörtlü Kısıtlama

Kaynak: Pinto (2020)

Anjani vd. (2021) ise bir projenin belirlenen hedef zamandan daha erken veya zamanında teslim edildiğinde, bütçe dahilinde veya bütçenin altında tamamlandığında, kabul edilen kapsam ve gereklilikleri karşıladığında ve ortaya çıkan ürünün performansı kullanıcıların/müşterilerin beklentilerini karşıladığında başarılı olarak tanımlandığını

ifade etmektedir. Müşteri beklentilerinin karşılanması, müşteri memnuniyetini sağlayacak ve bu durum projenin kalitesiyle ortaya çıkmaktadır. Dolayısıyla müşteri memnuniyeti, proje başarısında kalite unsurunu vurgulamaktadır (Anjani vd., 2021). Müşteri memnuniyeti, müşteri gereksinimlerinin karşılanması ve üretilen ürünün kullanıma uygun olması anlamına gelmektedir (Gwangwadza ve Hanslo, 2024).

Proje Yönetimi Bilgi Birikim Kılavuzu (Project Management Body of Knowledge- PMBOK), bir projenin başarılı olup olmadığını belirlemenin proje yönetiminde karşılaşılan en yaygın zorluklardan biri olduğunu belirtmektedir. Ancak geleneksel olarak kalite, kapsam, zaman ve maliyet gibi proje yönetimi ölçütlerinin, bir projenin başarısını değerlendirmede en önemli faktör olduğuna katılmaktadır (PMI, 2017).

Yazılım projelerinde temel hedef, kararlaştırılan işlevselliğin, gerekli kalite seviyesine uygun şekilde, zamanında ve bütçe dahilinde tamamlanmasıdır (Hughes vd., 2011). Literatürde, çevik yazılım geliştirme projelerinin başarısının genellikle kalite, kapsam, zaman ve maliyet kriterleriyle değerlendirildiği görülmektedir (Chow ve Cao, 2006; Stankovic vd., 2013; Brown, 2015; Litchmore, 2016; Nguyen, 2016; Chiyangwa ve Mnkandla, 2017; Bergmann, 2018; Stanberry, 2018; Syeda, 2018; Anjani vd., 2021; Klotins vd. 2021; Kukreja vd., 2021; Malik vd., 2021; Mulugeta, 2021; Haskin, 2022; Fry, 2024). Yazılım proje başarısının farklı kriterlerle değerlendirildiği çeşitli çalışmalarda mevcuttur. Müşteri/paydaş memnuniyeti (Aldahmansh, 2018; Garousi vd., 2019; Shakya ve Shakya, 2020; Tam vd., 2020); Heper, 2023; Binboğa ve Gümüşsoy, 2024), süreç verimliliği (Misra vd., 2009; Garousi vd., 2019; Binboğa ve Gümüşsoy, 2024), organizasyonel ihtiyaçların karşılanması (Aldahmansh, 2018) proje risklerinin azaltılması (Shakya ve Shakya, 2020) vb. proje başarı kriterleri kullanılmıştır. Proje başarısını değerlendirmek için farklı kriterleri içeren araştırmalar gerçekleştirilmiş olsa da “demir üçgen” hâlâ proje başarısını ölçmede en yaygın kullanılan yöntemdir (Drury-Grogan, 2014). Araştırma kapsamında proje başarısı kalite, kapsam, zaman ve maliyet kriterleriyle değerlendirilmiştir.

3.2.1. Kalite

Tüm projeler, başlangıçta belirlenen teknik özelliklere uygun olarak geliştirilir. Projenin amacı ya da nihai ürünün işlevselliği proje başlamadan önce tanımlanır.

Performans deęerlendirmesi, tamamlanan ürünün bu özelliklere ne ölçüde uygun çalıştığını incelemeyi ifade eder. Proje müşterileri, ürünün beklentilerini karşılamasını bekler ve bu deęerlendirme süreci genellikle “kalite” olarak adlandırılır (Pinto, 2020). Kalite, tamamlanmış ürünün ne kadar iyi çalıştığını ifade etmektedir (Drury-Grogan, 2014). Kalite, proje bitiminde iyi bir ürün, hizmet ya da sonuç sunmayı ifade eder ve projenin başarısını gösteren önemli bir parametredir (Chow ve Cao 2008; Suvacı, 2019). Çevik yöntemlerde kullanılan bazı pratikler, yazılım projelerinin kalitesine önemli katkılar sağlamaktadır (Kurapati vd., 2012; Al-Sakkaf vd., 2017; Henriksen ve Pedersen, 2017; Klotins vd., 2021). Yazılım projelerinde kalite artışını destekleyen bu pratikler arasında artımlı tasarım, retrospektif toplantılar, görev panoları (ör. Scrum, Kanban), günlük ayakta toplantılar, önce test, yeniden kod düzenleme ve sürekli entegrasyon yer almaktadır (Al-Sakkaf vd., 2017; Henriksen ve Pedersen, 2017; Pantiuchina vd., 2017).

3.2.2. Kapsam

Kapsam, projede neyin yapılıp neyin yapılmayacağını belirleyen ve böylece projenin sınırlarını tanımlayan bir çerçevedir. Kapsamın adlandırılması, farklı alanlarda deęişiklik gösterebilir. Örneğin, mühendislik alanında kapsam yerine “iş tanımı” terimi tercih edilebilmektedir. Projeye ait sınırların doğru bir şekilde belirlenmesi, projenin başarıyla yürütülmesi ve hedeflerine ulaşması için kritik öneme sahiptir (Suvacı, 2019). Kapsam, müşterinin talepleri doğrultusunda şekillenerek projenin sınırlarını belirler. Yazılım ürününde yer alması gereken özellikler, fonksiyonlar ve işlevler, yazılımın kapsamını oluşturmaktadır (Nizam, 2015). Kapsam, yazılım projelerinde tüm gereksinim ve hedeflerin karşılanmasını ifade eder ve projenin başarısını belirleyen önemli bir parametredir (Chow ve Cao, 2008). Çevik yöntemlerde kullanılan bazı pratikler, yazılım projelerinin kapsam ölçütünün başarısına önemli katkılar sağlamaktadır (Kurapati vd., 2012; Al-Sakkaf vd., 2017; Henriksen ve Pedersen, 2017; Klotins vd., 2021; Mulugeta, 2021). Bu pratikler arasında yinelemeli geliştirme, sprint planlama, kullanıcı hikayeleri, ürün iş listesi, sprint deęerlendirme, artımlı tasarım ve sprint retrospektifi yer almaktadır (Al-Sakkaf vd., 2017; Henriksen ve Pedersen, 2017; Klotins vd., 2021). Ghimire ve Charters (2022), ayaküstü toplantılar, ürün iş listesi, birim testi, kullanıcı hikayeleri, scrumban ve retrospektifler gibi pratiklerin yüksek düzeyde kullanımının, müşteri

gereksinimlerinin daha net tanımlanmasını sağladığını, dolayısıyla projelerin kapsam parametresinin başarısına önemli katkılar sunacağını ifade etmiştir.

2.3.3. Zaman

Projeler, belirli bir süre içinde tamamlanması gereken faaliyetlerdir ve süresiz olarak devam etmesi beklenmez. Bu sebeple projenin belirlenen zaman çizelgesine uygun olarak, yani planlanan tarihte veya öncesinde tamamlanması gerekliliğidir (Pinto, 2020). Zaman, proje süresi ile başlangıç ve bitiş tarihlerini içeren bir parametre olup, görevlerin planlanması ve tamamlanma tarihlerini ifade eder. Bu zamanlamalar genellikle proje çıktısını kullanacak kurum içindeki birimler veya müşteri tarafından belirlenir (Drury-Grogan, 2014; Suvacı, 2019). Projenin belirlenen süre hedefleri doğrultusunda tamamlanması, zaman parametresi açısından projenin başarısını belirleyen önemli bir kriterdir ve proje başarısına doğrudan katkı sağlar (Ahimbisibwe vd., 2015). Chow ve Cao (2008), projeyi zamanında teslim etmenin çevik projeler için kritik bir başarı faktörü olduğunu belirtmiştir. Genel olarak, beklenen teslim tarihine bağlı iyi planlanmış bir takvim, projeyi başarılı bir şekilde zamanında teslim etmenin etkili bir yoludur (Litchmore, 2016). Yazılım proje başarısının diğer parametrelerine olduğu gibi, zaman ölçütünün başarısına katkı sağlayan çeşitli çevik pratikler kullanılmaktadır (Kurapati vd., 2012; Al-Sakkaf vd., 2017; Henriksen ve Pedersen, 2017; Klotins vd., 2021; Mulugeta, 2021). Bu pratikler arasında tüm takım, kullanıcı hikayeleri ve iş ilerleme grafiği yer almaktadır (Henriksen ve Pedersen, 2017; Klotins vd., 2021).

Kurapati vd. (2012), ayaküstü toplantılar, sprint/yineleme, sürekli entegrasyon, testler, sprint planlama toplantıları, ortak kod sahipliği, iletişim, kısa sürüm ve retrospektiflerin yazılım geliştirme süreçlerinde ekiplerin verimliliğini artırmak ve proje yönetim süreçlerini optimize etmek amacıyla yaygın olarak kullanıldığını belirtmiştir. Jain ve Suman (2016), sürekli entegrasyon, sprint, sprint planlama ve günlük scrum/ayaküstü toplantılarının proje ilerlemesini izlemek, iletişimi geliştirerek kalitesini artırmak, koordinasyonu güçlendirmek ve sorunların erken çözülmesine yardımcı olmak açısından oldukça etkili olduğunu ifade etmiştir. Ghimire ve Charters (2022) ise ayaküstü toplantılar, ürün iş listesi, birim testi, kullanıcı hikayeleri, Scrumban ve retrospektif gibi pratiklerin yüksek düzeyde kullanımının müşteri gereksinimlerinin daha net

tanımlanmasını sağladığını belirtmiştir. Dolayısıyla, bu pratikler proje başarısının zaman parametresine önemli katkılar sunmaktadır.

2.3.4. Maliyet

Tüm projeler için önemli bir kısıtlama, sınırlı bütçedir. Projelerin, kaynakları en verimli şekilde kullanarak belirlenen bütçe sınırları içinde tamamlanması gereklidir. Şirketler, projelere sınırsız kaynak ayırmaz ve proje sonuçlarını belirsiz bırakmaz. Bu nedenle, proje başarısını değerlendirmede temel kriterlerden biri, projenin bütçe sınırları dahilinde tamamlanıp tamamlanmadığıdır (Pinto, 2020). Maliyet, projenin gerçekleşmesi için gerekli insan kaynakları, bütçe, yazılım geliştirme araçları, donanım, mekân gibi ihtiyaçların toplam bedelini ifade eder (Nizam, 2015). Maliyet, proje başarısını tanımlayan önemli parametrelerden biridir ve proje için ne kadar harcama yapılması gerektiğini belirtir (Suvacı, 2019). Chow ve Cao (2008), projeyi tahmini maliyet ve çaba dâhilinde teslim etmenin çevik projeler için kritik bir başarı faktörü olduğunu belirtmiştir.

Mulugeta (2021) yinelemeli geliştirme, sprint incelemesi, kullanıcı hikayeleri, ürün iş listesi, ayakta toplantılar, görev panosu ve planlama oyunu çevik pratiklerinin proje başarısı üzerinde olumlu etkileri olduğunu belirtmiştir. Ghimire ve Charters (2022) ise ayaküstü toplantılar, ürün iş listesi, birim testi, kullanıcı hikayeleri, Scrumban ve retrospektif gibi pratiklerin yüksek düzeyde kullanımının müşteri gereksinimlerinin daha net tanımlanmasını sağladığını belirtmiştir. İlgili pratiklerin yazılım geliştirme sürecinde kullanılması, projelerin maliyeti hakkında daha gerçekçi tahminlerde bulunulmasını sağlayarak, proje başarısının zaman parametresine önemli katkılar sunmaktadır (Highsmith, 2002; Martin, 2014).

3.3. KRİTİK BAŞARI FAKTÖRLERİ

Günümüzde bilgi teknolojilerine olan talebin hızla artması, yazılımı modern dünyada kritik ve vazgeçilmez bir unsur haline getirmiştir (Shakya ve Shakya, 2020). Yazılım, devletler, kurumlar, şirketler ve toplumların çeşitli ihtiyaçlarına çözüm üretmeye devam etmektedir (Nizam, 2015; Sommerville, 2015). Ancak yazılım modern dünyanın tüm yönleri için bu kadar önemli bir rol oynarken, yazılım geliştirme süreci kusursuz bir yapıya sahip değildir. Geleneksel yöntemlere kıyasla çevik yazılım geliştirme yöntemleri,

yazılım geliştirme süreçlerinde daha iyi ve yenilikçi bir yol olarak ortaya çıkmıştır (Chow ve Cao, 2008).

Çevik yöntemlerle geliştirilen yazılım projelerinin, geleneksel yöntemlere göre üç kat daha başarılı olduğu belirtilmesine rağmen, bu projelerin yarısından fazlası önemli zorluklarla karşılaşmaktadır (Mersino, 2020). Yazılım geliştirme süreçlerinde karşılaşılan bu sorunlar; düşük kalite, zayıf proje yönetimi, yetersiz süreçler ve eksik gereksinim analizleri gibi faktörlere dayalı olarak hâlâ yüksek maliyetlere neden olmakta ve gecikmiş, başarısız olmuş, terk edilmiş ya da reddedilmiş yazılım projelerine yol açmaktadır (Cao, 2006).

Bu durum göz önünde bulundurulduğunda, yazılım geliştirme projelerinin başarı oranını artırmak ve proje yöneticilerinin projelerin başarılı olması için dikkate alması gereken temel değişkenleri belirlemek, akademik ve sektörel alanda kritik bir öneme sahiptir. Çevik yazılım geliştirme projelerinin başarıyı etkileyen kritik faktörlerinin araştırılması, bu gereksinimi karşılamak amacıyla ortaya çıkmıştır (Barros vd., 2024).

Kritik Başarı Faktörleri (KBF), bireylerin, departmanların veya organizasyonların başarılı bir rekabet performansı sergileyebilmesi için tatmin edici sonuçların elde edilmesi gereken sınırlı sayıdaki alanlardır. KBF'ler, bir işin gelişmesi ve yöneticilerin belirledikleri hedeflere ulaşabilmesi için “doğru şekilde ilerlemesi gereken” temel alanları ifade etmektedir (Bullen ve Rockart, 1981). Başka bir ifadeyle, başarı için vazgeçilmez olan başarı faktörlerinin bir alt kümesidir (Barros vd., 2024). KBF, doğru bir şekilde ele alındığında, proje başarısı olasılığını önemli ölçüde artıran konular olarak tanımlanır (Nasir ve Sahibuddin, 2011).

Chow ve Cao (2008), KBF'ler bir organizasyonun veya projenin genel başarısı için kritik öneme sahip sınırlı sayıda alandır. Reel (1999), proje yönetimi gibi birçok etkenin proje sonuçlarını etkilediğini belirtmiştir. Yazılım geliştirme bağlamında, kritik başarı faktörleri genellikle temel proje yönetim teknikleriyle veya iş stratejisi ve yazılım mühendisliğinin birleşimiyle ilişkilidir (Chow ve Cao, 2008; Reel, 1999). Bilişim teknolojileri (BT) departmanları ve BT projeleri, başarıya ulaşmak için ihtiyaç duyulan temel unsurları belirlemek amacıyla kritik başarı faktörlerinden faydalanır (Sudhakar, 2012).

Literatürde çevik yazılım geliştirme projelerine yönelik KBF'ler üzerine birçok çalışma bulunmaktadır (Cao, 2006; Misra vd., 2006; Chow ve Cao, 2008; Misra vd., 2009; Nasir ve Sahibuddin, 2011; Stankovic vd., 2013; Ahimbisibwe vd., 2015; Brown, 2015; Da Silva ve Dos Santos, 2015; Vithana vd., 2015; Demirel, 2016; Litchmore, 2016; Nguyen, 2016; Siddique ve Hussein, 2016; Silva ve Santos, 2016; Aldahmash vd., 2017a; Aldahmash vd., 2017b; Chiyangwa ve Mnkandla, 2017; Aldahmash, 2018; Bergmann, 2018; Stanberry, 2018; Syeda, 2018; Bergmann ve Karwowski, 2019; Ergüder, 2019; Garousi vd., 2019; Naslund ve Kale, 2020; Shakya ve Shakya, 2020; Tam vd., 2020; Tsoy ve Staples, 2020; Kukreja vd., 2021; Qatanani vd., 2021; Azhar ve Abdullah, 2022; Bogopa ve Marnewick, 2022; Fadilah vd., 2022; Greene, 2022; Haskin, 2022; Yeşilyurt, 2022; Heper, 2023; Ojha, 2023; Barros vd., 2024; Binboğa ve Gümüşsoy, 2024; Fry, 2024; Gwangwadza ve Hanslo, 2024; Junior vd., 2024; Zhang vd., 2024). Yapılan literatür incelemesi, çevik yazılım geliştirme projelerindeki KBF'lerin Chow ve Cao'nun (2008) çalışmasına sıkça referans verdiğini ve bu çalışmanın KBF'lerin tanımlanması açısından önemli bir temel oluşturduğunu göstermektedir. Özellikle Chow ve Cao'nun (2008) modeli, çevik yazılım geliştirme projelerinde başarıyı etkileyen kritik faktörlerin tanımlanmasında sıkça kullanılan bir referans model olarak öne çıkmaktadır. Bu model, çevik projelerin başarısını etkileyen faktörlerin sistematik bir şekilde incelenmesine olanak tanımış ve literatürde geniş bir kabul görmüştür.

Tablo 3.1. Çevik Kritik Başarı Faktörleri

Boyut	Faktör
Organizasyon	Yönetim Taahhüdü
	Organizasyon Ortamı
	Takım Ortamı
İnsan	Takım Yeteneği
	Müşteri Katılımı
Süreç	Proje Tanımlama Süreci
	Proje Yönetim Süreci
Teknik	Çevik Yazılım Mühendisliği Teknikleri
	Teslimat Stratejisi
Proje	Projenin Doğası
	Projenin Türü
	Dinamik Proje Takvimi

Kaynak: Chow ve Cao (2008)

Chow ve Cao (2008), çevik projelerde başarıyı etkileyen 12 potansiyel faktörü tespit etmiş ve bu faktörleri insan, süreç, organizasyonel, teknik ve proje olmak üzere beş ana kategori altında sınıflandırmıştır. İlgili faktörler Tablo 3.1’de sunulmuştur. Söz konusu faktörler, çevik yazılım projelerinin başarısını sağlamada kritik öneme sahip unsurlar olarak tanımlanmaktadır.

3.3.1. İnsan Faktörleri

İnsan faktörleri, projelerin başarısında en önemli unsurlardan biri olarak kabul edilmektedir (Ojha, 2023). İnsan faktörleri, bireysel takım üyelerinden yöneticilere, ekiplerden müşterilere kadar geniş bir kapsamı içermektedir (Cao, 2006). Yazılım geliştirme projelerinin başarısı, insanlar ve insan kaynakları faktörleriyle doğrudan ilişkilidir (Misra vd., 2006; Azhar ve Abdullah, 2022). Çevik yazılım geliştirme projelerinde ise bu ilişki daha belirgin hale gelmektedir. Bunun temel nedeni, çevik yöntemlerin bireyler ve etkileşimler, müşteri iş birliği ve müşterilerden gelen değişiklik önerilerine hızlı yanıt verme gibi değerlere öncelik vermesidir (Misra vd., 2009).

So ve Scholl (2009), çevik yöntemlerin yalnızca teknik ve süreçle ilgili konularla sınırlı kalmadığını, aynı zamanda insan ve sosyal boyutları kapsayan geniş bir perspektif sunduğunu ifade etmiştir. Çevik yazılım geliştirme projelerinde insan faktörleri, projelerin başarılı bir şekilde tamamlanmasında kritik bir rol oynamaktadır (Litchmore, 2016; Nguyen, 2016; Ojha, 2023). Bu faktörler, takım yeteneği ve müşteri katılımını içermekte ve projelerin başarısını doğrudan etkilemektedir (Chow ve Cao, 2008).

Tam vd. (2020), takım yeteneği ve müşteri katılımının çevik yazılım geliştirme projelerinin başarısına anlamlı katkılar sağladığını vurgulamıştır. Dolayısıyla, insan faktörlerinin etkin bir şekilde yönetilmesi, çevik projelerin başarıya ulaşmasında KBF olarak öne çıkmaktadır.

3.3.1.1. Müşteri Katılımı

Müşteri katılımı, çevik yazılım geliştirme süreçlerinde müşterinin ekibin bir parçası olarak tam zamanlı yer almasını ifade eder ve çevik yöntemlerin temel değerlerinden biri olarak kabul edilir (Agile Alliance, 2001b; Beck, 1999; Xu, 2009). Geleneksel yazılım geliştirme projelerinde müşteri ile geliştirme ekibi arasındaki etkileşim genellikle

sınırlıdır. Bu tür projelerde müşteriyle yalnızca gereksinimlerin belirlendiği başlangıç aşamasında ve proje sonunda geri bildirim alınırken iletişim kurulmaktadır (Hoda vd., 2011).

Çevik yazılım geliştirme yöntemlerinde ise müşteriye geliştirme sürecinin her aşamasına dahil ederek, kullanıcı hikayeleri yazma, ürün özelliklerini tartışma, özellik (iş) listelerini önceliklendirme ve düzenli geri bildirim sağlama süreçlerinde aktif bir rol üstlenmelerini teşvik eder (Hoda vd., 2011; Barros vd., 2024). Müşteri katılımı, gereksinimlerin ve önceliklerin hızla netleştirilmesine olanak tanırken, belirsizliklerin veya sorunların kısa sürede çözülmesini sağlar (Anderson, 2004; Şimşek ve Ateş, 2022). Bu durum, projenin etkili ve doğru bir şekilde ilerlemesine katkıda bulunur (Barros vd., 2024).

Müşteri, yalnızca gereksinimlerin belirlenmesinde değil, aynı zamanda projenin yönetiminde ve başarıya ulaşmasında da kilit bir rol oynar (Xu, 2009). Müşterinin projeye doğrudan dahil edilmesi, gereksiz çabaların önüne geçerken, ekip içindeki güveni artırır ve sürekli iyileştirme süreçlerini destekler (Beck ve Andres, 2004). Gerçek bir müşteri veya müşteri temsilcisinin eksikliği, kullanılmayan özelliklerin geliştirilmesi, kabul kriterlerini yansıtmayan testlerin tanımlanması ve projenin farklı bakış açılarını birleştirme fırsatının kaybedilmesi gibi riskler doğurabilir (Beck ve Andres, 2004).

Müşterinin ekip içinde yer alması, proje boyunca güven ortamı yaratırken yazılımın başarısını da artırır (Beck ve Andres, 2004; Xu, 2009). Ahimbisibwe vd. (2015), müşteri katılım düzeyinin yazılım geliştirme projelerinin başarısıyla doğrudan ilişkili olduğunu ve yüksek müşteri katılımı seviyelerine sahip projelerin genellikle daha başarılı olduğunu vurgulamıştır. Çevik yazılım geliştirme projelerinde müşterinin aktif katılımı, projelerin başarılı bir şekilde tamamlanması için kritik bir faktör olarak değerlendirilmekte ve bu durum literatürde geniş ölçüde desteklenmektedir (Perera ve Perera, 2019; Ojha, 2023). Çevik yazılım geliştirme projelerinde müşteri katılımı, önemli bir özellik ve temel bir başarı faktörü olarak görülmektedir (Aldahmash, 2018).

3.3.1.2. Takım Yeteneği

Takım yeteneği, insan faktörleri kapsamında ele alınan ve çevik yazılım geliştirme projelerinin başarısını etkileyen kritik bir başarı faktörü olarak tanımlanmaktadır. (Misra

vd., 2006; Chow ve Cao, 2008; Litchmore, 2016; Aldahmash, 2018; Tam vd., 2020; Tsoy ve Staples, 2020; Barros vd., 2024). Takım yeteneđi, yalnızca yüksek teknik yeterlilik ve ilgili alanlardaki uzmanlığı deđil aynı zamanda teknik beceriler (örneğin programlama dilleri ve metodolojiler), ekip çalışması, süreç bilgisi, alan bilgisi ve ürün bilgisi gibi unsurları da kapsamaktadır (Stankovic vd., 2013; Šmite vd., 2017). Bu bilgi ve beceriler, yazılım geliştirme süreçlerinde ekiplerin verimliliğini artırmaya ve proje yönetim süreçlerinin daha etkin bir şekilde optimize edilmesine katkı sağlamaktadır (Kurapati vd., 2012).

Literatürde, takım yeteneđinin proje başarısına olan etkisi sıklıkla vurgulanmıştır. Misra vd. (2009), yetkin bir ekibin, müşterinin gereksinimlerini karşılayan çalışan yazılımı hızlı bir şekilde teslim edebileceğini ifade etmektedir. Ahimbisibwe vd. (2015), takım yeteneđinin riskleri hızlıca anlamayı ve bu risklere yanıt vermeyi sağladığını, böylelikle proje başarısını artıran temel bir faktör olduğunu ileri sürmektedir. Takım yetenek düzeyinin artmasının, çevik yazılım geliştirme projelerinin başarı olasılığını önemli ölçüde artırdığına işaret edilmektedir (Barros vd., 2024).

Sudhakar (2012), takım faktörleri içerisinde yer alan takım yeteneđinin, proje yönetim faktörlerini (örneğin proje planlaması, net proje hedefleri, proje kontrol mekanizmaları) etkilediğini ifade etmektedir. Chow ve Cao (2008), takım yeteneđini çevik yazılım geliştirme projelerinde başarıyı etkileyen üç temel KBF'den biri olarak tanımlamaktadır. Ayrıca, Aldahmash (2018), yazılım geliştirme projelerinin ilerleyen aşamalarında takım yeteneđinin öneminin giderek arttığını belirtmektedir. Literatürdeki çalışmalar, takım yeteneđinin çevik yazılım projelerinin başarısına olan katkıları açıkça ortaya koymaktadır.

3.3.2. Süreç Faktörleri

Çevik yazılım geliştirme sürecinde başarıya ulaşmak için gereksinimlerin belirlenmesinden yönetim süreçlerine kadar tüm yaşam döngüsünün çevik bir yaklaşımla yürütülmesi gerekmektedir (Gwangwadza ve Hanslo, 2024). Proje süreci, yazılım geliştirme süreçlerinin ne derece etkin şekilde yürütüldüğünü tanımlamakla birlikte projenin zamanında, bütçe sınırları içinde ve belirlenen kapsam doğrultusunda teslim edilip edilmediğini ölçmektedir (Ahimbisibwe vd., 2015). Süreç faktörleri, proje işlevleri

veya süreçleriyle bağlantılı unsurlardır (Nguyen, 2016). Bu faktörler, çevik yazılım projelerinin başarıyla sonuçlanmasında kritik bir öneme sahiptir (Kukreja vd., 2021; Junior vd., 2024).

Chow ve Cao (2008), süreç faktörlerini proje tanımlama ve proje yönetim süreçleri olarak tanımlamıştır. Litchmore (2016), çevik yazılım geliştirme projelerinde başarıyı belirleyen temel unsurların kullanılan yöntemler (örneğin, Scrum, XP, Kanban) değil, insan ve süreç faktörleri olduğunu vurgulamıştır. Benzer şekilde, Farlik (2016), yazılım geliştirme süreçlerinin iyileştirilmesinin proje başarısına doğrudan katkı sağladığını ifade etmiştir. Yeşilyurt (2022), çevik projelerde süreç faktörünü en önemli başarı unsuru olarak değerlendirirken, Binboğa ve Gümüşsoy (2024), süreç faktörlerinin çevik projelerin başarısını etkileyen en önemli faktörlerden biri olduğunu belirtmiştir.

3.3.2.1. Proje Tanımlama Süreci

Proje tanımlama süreci, bir projenin hedeflerini, kapsamını ve kısıtlarını netleştirmek amacıyla yürütülen kritik bir aşamadır. Bu süreç, paydaşlar arasında ortak bir anlayış oluşturmayı ve proje başarısını sağlamayı hedefler (PMI, 2021). Aynı zamanda, sonraki tüm proje faaliyetleri için bir temel oluşturarak paydaş beklentilerinin yönetiminde önemli bir rol oynar (Kerzner, 2017).

Yazılım projeleri, bir ihtiyacın belirlenmesiyle başlayan bir süreçtir. Proje tanımlama, proje hedeflerine ulaşmak ve müşteri gereksinimlerini karşılamak amacıyla izlenecek yöntemlerin, gerçekleştirilecek işlerin, kullanılacak kaynakların ve oluşturulacak iş takvimlerinin proje kısıtları dikkate alınarak planlanmasını ifade eder (Nizam, 2015). Bu süreçte öncelikli olarak projenin hedeflerine uygun bir kapsam tanımlanmalı, ardından bütçe, zaman ve kaynak gibi kısıtlar belirlenmelidir. Ayrıca, projenin başarısını etkileyebilecek potansiyel riskler tanımlanmalı ve bu risklere yönelik önlemler değerlendirilmelidir (Wysocki, 2006; Hughes vd., 2011; Nizam, 2015; Kerzner, 2017).

Proje tanımlama sürecinin dikkatli bir şekilde yürütülmesi, kaynakların verimli kullanılmasını sağlarken, projenin belirlenen zaman ve bütçe sınırları içinde başarıyla tamamlanma olasılığını artırır (Kerzner, 2017). Geleneksel yazılım geliştirme projelerinde bu süreçte kapsam, maliyet ve zaman çizelgesi kesin ve ayrıntılı bir şekilde

tanımlanırken, çevik yazılım geliştirme projelerinde yalnızca yeterli düzeyde bir planlama yapılır. Çevik yöntemler, esnekliği artırmak ve değişen gereksinimlere uyum sağlamak amacıyla detaylı planlamayı yinelemeler boyunca kademeli olarak gerçekleştirir (Awad, 2005; Başar vd., 2015; Javanmard ve Alian, 2015; Sommerville, 2015; Kirpitsas ve Pachidis, 2022).

Çevik yazılım geliştirme projelerinde proje tanımlama süreci, müşteri ile yakın iş birliği içinde yürütülmekte ve çeşitli pratiklerle desteklenmektedir (Larman, 2003; Beck ve Andres, 2004; Martin, 2014; Webb, 2015). Bu iş birliği, ihtiyaçların doğru bir şekilde tanımlanmasına olanak tanırken, süreç boyunca ortaya çıkabilecek değişikliklere hızlı bir şekilde uyum sağlamayı kolaylaştırmaktadır.

Proje tanımlama süreci, çevik yazılım geliştirme projelerinde kritik bir başarı faktörü olarak değerlendirilmektedir (Stankovic vd., 2013; Brown, 2015; Litchmore, 2016; Stanberry, 2018). Proje tanımlama süreci proje hedeflerinin, kapsamının ve kısıtlarının net bir şekilde belirlenmesini sağlayarak projenin etkin yönetimini ve başarıya ulaşmasını desteklemektedir.

3.3.2.2. Proje Yönetim Süreci

Çevik yazılım geliştirme projeleri, herhangi bir yazılım projesinin tipik yönetim süreçlerini takip etmektedir. Bu süreçler, gereksinim yönetimi, proje yönetimi ve konfigürasyon yönetimi gibi süreçleri kapsamaktadır. Ancak, bu süreçler çevik bir yaklaşımla uygulanmaktadır. Örneğin, gereksinim yönetimi süreci açısından çevik projeler, “başlangıç gereksinimlerini çok genel bir düzeyde belirler, böylece proje ilerledikçe yorumlama ve uyarılama için geniş bir alan bırakır” (Koch, 2005, akt. Cao, 2006).

Çevik yazılım geliştirme projelerinin doğası gereği, projenin erken aşamalarında genellikle belirsizlikler bulunmaktadır. Bu durum, gereksinimlerin henüz net olmamasından ve proje süresince gelişmesinin beklenmesinden kaynaklanmaktadır. Değişikliklere uyum sağlayan ve birçok yineleme (iterasyon) sunan çevik projelerin bu yapısı, esnek bir proje yönetim yaklaşımını zorunlu kılmaktadır (Aldahmash, 2018). Çevik yazılım geliştirme projelerinin başarısında, uygun bir proje yönetim sürecinin seçimi önemli bir katkı sağlamaktadır (Chow ve Cao, 2008).

Literatürde yapılan çeşitli çalışmalar, çevik proje yönetim sürecinin proje başarısına önemli katkılar sağladığını ortaya koymaktadır (Chow ve Cao, 2008; Stankovic vd., 2013; Litchmore, 2016; Nguyen, 2016; Aldahmash, 2018; Stanberry, 2018; Tsoy ve Staples, 2020; Binboğa ve Gümüşsoy, 2024).

3.3.3. Organizasyon Faktörleri

Organizasyon faktörleri, çevik yazılım geliştirme projelerinin başarısında kritik bir öneme sahiptir (Ahimbisibwe vd., 2015; Vithana vd., 2015; Chiyangwa ve Mnkandla, 2017; Bergmann, 2018; Shakya ve Shakya, 2020; Kukreja vd., 2021; Junior vd., 2024). Chow ve Cao (2008), organizasyon boyutunda proje başarısına katkı sağlayan unsurları; kararlı bir sponsor veya yönetici, hiyerarşik yerine iş birliğine dayalı bir organizasyon kültürü, yüz yüze iletişime yüksek değer veren sözlü kültür, çevik metodolojinin evrensel olarak kabul edildiği organizasyonlar, tüm ekibin aynı lokasyonda bulunması, çevik çalışma tarzına uygun bir iş ortamı sağlayan tesisler ve çevik yaklaşıma uygun bir ödüllendirme sistemi olarak tanımlamıştır. Bu unsurlar, yönetim taahhüdü, organizasyon ortamı ve takım ortamı faktörlerini oluşturmaktadır (Stankovic vd., 2013).

Üst düzey yönetim veya yöneticiler, çevik metodolojilerin projelerde başarılı bir şekilde uygulanmasını desteklemek adına somut ve aktif katkılar sağlamalıdır. Bu destek, ekip üyelerine rehberlik edilmesi, danışmanlık hizmetlerinin sunulması, kullanıcıların eğitilmesi ve psikolojik, finansal ile manevi destek sağlanması gibi çeşitli biçimlerde gerçekleştirilmelidir (Chow, 2006).

Organizasyon kültürü, bir organizasyondaki çevik yazılım geliştirmeyi etkileyebilecek unsurların bütünü olarak tanımlanabilir (Aldahmash vd., 2017a). Çevik yöntemler, bürokratik yapıya sahip organizasyonlar için uygun değilken, dinamik ve hızlı değişim gösteren organizasyonlar için oldukça etkili bir yaklaşım sunmaktadır (Misra vd., 2006). Çevik yöntemlerde kullanılan bilgilendirici çalışma alanı, birlikte oturma ve açık çalışma alanı gibi çeşitli pratikler, çevik dostu takım ve organizasyon ortamının oluşturulmasına katkı sağlayacaktır (Gwangwadza ve Hanslo, 2024).

3.3.4. Teknik Faktörler

Çevik yazılım geliştirme projelerinin başarısında en önemli kritik başarı faktörlerinden biri teknik faktörlerdir (Chow ve Cao, 2008). Teknik faktörler, bir projenin işleyişini doğrudan etkileyen ve yazılım, donanım veya proje geliştirme sürecinde kullanılan teknolojiyle bağlantılı unsurları kapsamaktadır (Nguyen, 2016; Azhar ve Abdullah, 2022). Bu unsurlar, doğru zamanda etkinliklerin başlatılması, kodlama standartlarının uygulanması ve planlanmış uygun dokümantasyonun takip edilmesi gibi entegrasyon ve teslimat stratejilerini içermektedir (Greene, 2022).

Çevik yazılım geliştirme projelerinde teknik faktörler, çevik yazılım geliştirme tekniklerinin ve teslimat stratejilerinin etkili bir şekilde uygulanmasını sağlamaktadır (Chow ve Cao, 2008). Çevik yazılım geliştirme teknikleri, proje ekip üyelerinin seçilen yöntem veya pratiğin kavramlarını, terminolojisini ve temel ilkelerini anlamasını gerektirmektedir (Stankovic vd., 2013). Teslimat stratejisi ise düzenli olarak işlevsel yazılım üretimini ve en kritik özelliklerin öncelikli olarak sunulmasını ifade etmektedir. Bununla birlikte, ekip üyelerinin teslimat stratejisindeki rollerini ve bu stratejiyi nasıl uygulayacaklarını net bir şekilde bilmeleri gereklidir (Cao, 2006; Ojha, 2023).

Teknik faktörlerin, çevik yazılım geliştirme projelerinin başarısında merkezi bir role sahip olduğu çeşitli araştırmacılar tarafından doğrulanmıştır (Chow ve Cao, 2008; Brown, 2015; Nguyen, 2016; Shakya ve Shakya, 2020; Kukreja vd., 2021; Haskin, 2022; Heper, 2023).

3.3.5. Proje Faktörleri

Proje faktörleri, çevik yazılım geliştirme projelerinin başarısı için kritik başarı faktörleri arasında yer almaktadır (Stankovic vd., 2013; Brown, 2015; Chiyangwa ve Mnkandla, 2017; Stanberry, 2018; Shakya ve Shakya, 2020; Heper, 2023; Binboğa ve Gümüşsoy, 2024). Bu faktörler, projenin doğasını, proje türünü ve dinamik proje takvimini kapsamaktadır (Chow ve Cao, 2008).

Proje doğası, hayati risk taşımayan ancak iş açısından kritik bir yazılım projesi olarak tanımlanabilir. Hayati risk taşıyan projelere, ileri düzey silah sistemleri veya hava trafik kontrol sistemleri gibi örnekler verilebilir (Chow ve Cao, 2008). Çevik yazılım

geliştirme yöntemleri, hayati risk taşımayan projelerde daha uygun bir yaklaşım sunarken, hayati risk içeren projelerde genellikle geleneksel yazılım geliştirme yöntemleri tercih edilmektedir (Cohen vd., 2004; Nizam, 2015).

Proje türü faktörü, değişken kapsam ve ortaya çıkan gereksinimlere sahip projelere işaret etmektedir (Stankovic vd., 2013). Dinamik proje takvimi ise projelerin hedeflerine daha hızlı ulaşılmasını sağlamanın yanı sıra gereksinimlerin ve koşulların değişimine uygun olarak esnek bir şekilde güncellenmesini ifade etmektedir (Coa, 2006). Sürekli değişen ve karmaşık belirsizlik içeren dinamik takvime sahip projelerde çevik yöntemler oldukça etkili sonuçlar sunmaktadır (Highsmith, 2002). Kukreja vd. (2021), çevik yazılım geliştirme projelerinde başarıyı etkileyen en önemli kritik başarı faktörünün proje faktörleri olduğunu vurgulamıştır.

DÖRDÜNCÜ BÖLÜM

LİTERATÜR ÖZETİ

Çevik yazılım geliştirmede insan ve süreç faktörlerinin proje başarısına etkisinde çevik pratiklerin düzenleyici rolünü araştırmak amacıyla gerçekleştirilen çalışmada literatür özeti başlıklar halinde sunulmuştur.

4.1. ÇEVİK YAZILIM GELİŞTİRME PROJELERİNDE KRİTİK BAŞARI FAKTÖRLERİYLE İLGİLİ ÇALIŞMALAR

Chow ve Cao (2008) çevik yazılım projelerindeki başarıyı etkileyen kritik başarı faktörlerini ve bu faktörlerin birbirlerine göre önem düzeylerini belirlemek amacıyla çalışma gerçekleştirmiştir. Literatür taraması sonucunda, çevik projelerde başarıyı etkileyen 12 olası faktör tespit edilmiş ve bu faktörler insan, süreç, organizasyon, teknik ve proje olmak üzere beş ana kategori altında toplanmıştır. Proje başarısı kalite, kapsam, maliyet ve zaman kriterleri ile tanımlanmıştır. Nicel yöntemler kullanılarak gerçekleştirilen çalışmada, veriler araştırmacılar tarafından geliştirilen anket ile toplanmış ve çoklu regresyon analizi ile değerlendirilmiştir. Araştırmaya 25 farklı ülkeden 109 kişi katılmıştır. Araştırma sonuçlarına göre; takım ortamı (kalite), takım yeteneği (zaman ve maliyet), müşteri katılımı (kapsam), proje yönetim süreci (kalite), çevik yazılım mühendisliği teknikleri (kalite ve kapsam) ve teslim stratejisi (kapsam, zaman ve maliyet) kritik başarı faktörleri olarak belirlenmiştir. Araştırmada, çevik yazılım projelerinde başarıyı etkileyen üç ana kritik başarı faktörünün teslimat stratejisi, çevik yazılım mühendisliği teknikleri ve takım yeteneği olduğu tespit edilmiştir. Teknik boyutun çevik projelerdeki başarı üzerinde en büyük etkiye sahip olduğu, bunu insan, süreç ve organizasyonel boyutların izlediği; proje boyutunun ise başarı üzerinde anlamlı bir etkisinin olmadığı sonucuna ulaşılmıştır (Cao, 2006; Chow ve Cao, 2008).

Misra vd. (2006), çevik yazılım geliştirme projelerinin başarısını etkileyen faktörleri incelemek amacıyla bir araştırma gerçekleştirmiştir. Literatür taraması niteliğinde gerçekleştirilen araştırmada çevik yazılım geliştirme başarı faktörlerine dair daha önce yapılmış çeşitli çalışmalardan elde edilen bulgular derlenerek çevik yazılım geliştirme projelerinin başarıya ulaşmasında etkili olan organizasyon, insan ve teknik

faktörler incelenmiştir. Organizasyon faktörler başlığında müşteri bağlılığı, karar alma süresi, takım dağılımı, takım büyüklüğü, kurumsal kültür, planlama ve kontrol, güvenlik kritikliği ve değişim-belirsizlik yer almaktadır. İnsan faktörleri altında yetkinlik, kişisel özellikler, iletişim ve müzakere, ekip kompozisyonu (yapısı), toplumsal kültürü ve eğitim-öğrenme unsurları ele alınmıştır. Teknik faktörler ise gereksinimler, geliştirme ve test süreçlerini kapsamaktadır. Elde edilen bulgulara göre müşteri bağlılığı, karar alma süresi, takım dağılımı, yetkinlik, kişisel özellikler, iletişim ve teknik süreçlerin çevik yazılım geliştirme projelerindeki başarı üzerinde önemli bir etkiye sahip olduğunu göstermektedir. Özellikle müşteri ile sıkı iş birliği, hızlı karar alma yeteneği ve ekip üyelerinin yetkinliği başarıya ulaşmada kritik faktörler olarak öne çıkmıştır.

Misra vd. (2009), çevik yazılım geliştirme projelerinin başarısını etkileyen faktörleri belirlemek amacıyla kapsamlı bir araştırma gerçekleştirmiştir. Araştırmada başarıya etki eden faktörler organizasyon ve insan olmak üzere iki ana başlık altında incelenmiştir. İnsan faktörleri altında yetkinlik, kişisel özellikler, iletişim ve müzakere, toplumsal kültürü ve eğitim-öğrenme yer almaktadır. Organizasyon faktörler arasında müşteri memnuniyeti, müşteri iş birliği, müşteri bağlılığı, karar alma süresi, takım dağılımı, takım büyüklüğü, kurumsal kültür, planlama ve kontrol yer almaktadır. Araştırmada proje başarısı ise azalan teslimat süreleri, yatırım getirisi artışı, mevcut müşteri gereksinimlerinin karşılanabilme yeteneği, değişen müşteri gereksinimlerine uyum sağlama kapasitesi ve iş süreçlerinin iyileştirilmesi içeren beş değişken ile ölçülmüştür. Nicel yöntemlerle gerçekleştirilen çalışmada veriler anket yardımıyla 241 katılımcıdan elde edilmiştir. Anket verileri korelasyon ve regresyon analizi gibi istatistiksel yöntemlerle analiz edilmiştir. Araştırma sonuçları müşteri memnuniyeti, müşteri iş birliği, müşteri bağlılığı, karar alma süresi, kurumsal kültür, kontrol, kişisel özellikler, toplumsal kültür ve eğitim-öğrenme çevik yazılım geliştirme projelerinin başarısında önemli rol oynadığını ortaya koymuştur.

Nasir ve Sahibuddin (2011) tarafından gerçekleştirilen çalışma, yazılım projelerinde kritik başarı faktörlerini tanımlamak ve bu faktörleri önem sırasına göre karşılaştırmalı olarak değerlendirmek amacıyla yapılmıştır. Nitel yöntemlerle geliştirilen araştırma literatür taramasına dayanmaktadır. Çalışma kapsamında 1990-2010 yılları arasında yayınlanmış 43 makale incelenmiştir. Veriler, dört büyük çevrimiçi veri tabanı üzerinden (ScienceDirect, IEEE Xplore, SpringerLink, Emerald Insight) elde edilen

makalelerden toplanmıştır. Veri toplama aşamasında, yazılım projelerinin başarısını etkileyen faktörler literatürdeki makalelerden çıkarılarak içerik analizi yöntemi ile incelenmiştir. Her bir makalede bahsedilen kritik başarı faktörlerinin sıklıkları belirlenmiş ve bu sayede faktörlerin göreceli önemi hesaplanmıştır. Toplamda 26 kritik başarı faktörü tespit edilmiş ve insan (7), süreç (16) ve teknik (3) faktörler olmak üzere üç ana grupta toplanmıştır. Araştırmanın bulgularına göre, projelerin başarısında insan ve süreç odaklı faktörlerin, teknik faktörlere kıyasla çok daha belirleyici olduğu sonucuna ulaşılmıştır.

Sudhakar (2012), yazılım projelerinde kritik başarı faktörlerini tanımlamak ve bu faktörleri kategorize ederek bir kavramsal model geliştirmek amacıyla kapsamlı bir literatür taramasına dayalı bir araştırma gerçekleştirmiştir. Çalışma, özellikle iletişim, ekip, proje yönetimi ve ürünle ilgili faktörlere vurgu yaparak, bu faktörleri yedi ana kategoriye ayırmıştır: iletişim, teknik, organizasyonel, çevresel, ürün, takım ve proje yönetimi faktörleri. Literatürde sıkça tekrar eden ve önem derecesi yüksek olan faktörler arasından her bir kategori için beş kritik başarı faktörü seçilmiştir. Araştırmada, toplamda 35 kritik başarı faktörü tanımlanmıştır. Bu faktörler arasında; proje içi iletişim, üst yönetim desteği, net proje hedefleri, kullanıcı katılımı, ekip çalışması, çıktı güvenilirliği ve proje planlaması gibi unsurlar öne çıkmaktadır. Çalışma, bu faktörlerin yazılım projelerinin başarısını artırma potansiyeline sahip olduğunu ve proje yöneticileri için önemli bir rehber sunduğunu ifade etmektedir. Ayrıca araştırma teorik katkılar sağlarken, proje yöneticilerinin bu faktörlere odaklanarak projelerin başarı oranlarını artırabileceklerini vurgulamaktadır.

Stankovic vd. (2013), çevik yazılım projelerinde başarıyı etkileyen kritik faktörleri belirlemek ve önceki araştırmalara kıyasla elde edilen bulguları değerlendirmek amacıyla bir çalışma gerçekleştirmiştir. Araştırma, Chow ve Cao'nun (2008) yaptığı çalışmanın tekrarı niteliğinde olup, nicel yöntemler kullanılarak yürütülmüştür. Araştırmada, Chow ve Cao'nun (2008) geliştirdiği anket formu kullanılmış ve Güneydoğu Avrupa Bölgesi'nde (Sırbistan, Makedonya, Hırvatistan ve Bosna Hersek) faaliyet gösteren bilişim şirketlerindeki yönetici ve üst düzey geliştiricilerden oluşan 23 kişilik bir örneklem üzerinde uygulanmıştır. Verilerin analizi için çoklu regresyon yöntemi tercih edilmiştir. Araştırma sonuçlarına göre, proje yönetim ile proje tanımlama sürecinin maliyet üzerinde, projenin doğasının hem zaman hem de maliyet üzerinde ve dinamik proje takvimi maliyet üzerinde önemli bir etkisi olduğu belirlenmiştir. Bu bulgular, çevik

projelerde maliyet ve zaman yönetiminin, projelerin başarılı olmasında kritik rol oynadığını göstermektedir.

Ahimbisibwe vd. (2015) tarafından gerçekleştirilen bu çalışma, yazılım geliştirme projeleri için kritik başarı faktörlerini (KBF) belirlemek ve bu faktörlerin çevik ve geleneksel yöntemler üzerindeki etkilerini incelemek amacıyla yapılmıştır. Ayrıca, yazılım geliştirme projelerinin başarısında etkili olan faktörlerin, proje özellikleri ve kullanılan proje yönetimi metodolojisine (çevik ya da geleneksel) göre nasıl farklılık gösterdiğini açıklayan bir durumsal uygunluk modeli geliştirilmiştir. Çalışma, sistematik bir literatür taraması yöntemiyle gerçekleştirilmiş olup, 148 farklı yayından elde edilen veriler doğrultusunda yazılım projeleri için 37 kritik başarı faktörü tanımlanmış ve bu faktörler organizasyon, takım, müşteri ve proje faktörleri olmak üzere dört ana kategoriye ayrılmıştır. Araştırma, proje faktörlerini bir düzenleyici değişken olarak ele almış ve proje başarısı ile organizasyon, ekip ve müşteri faktörleri arasındaki ilişkileri düzenlediğini öne sürmüştür. Bu bağlamda, organizasyon, ekip ve müşteri faktörlerinin proje başarısı üzerinde genel olarak doğrudan olumlu bir etkiye sahip olduğu; buna karşılık, proje faktörlerinin proje başarısı ile organizasyon, ekip ve müşteri faktörleri arasındaki ilişkiler üzerinde düzenleyici bir etkiye sahip olduğu ve proje başarısı üzerinde potansiyel olarak olumsuz doğrudan etkileri olduğu varsayımıyla bir durumsal uygunluk modeli geliştirilmiştir.

Brown (2015) tarafından gerçekleştirilen doktora çalışması, Chow ve Cao'nun (2008) çevik yazılım projelerindeki kritik başarı faktörlerini belirleme çalışmasının bir tekrarı niteliğindedir. Çalışma, çevik yöntemler kullanılan yazılım projelerinde başarıyı sağlayacak kritik başarı faktörlerini ve bu faktörlerin göreceli önem düzeylerini incelemeyi amaçlamaktadır. Chow ve Cao'nun (2008) çalışmasında ABD'deki çevik yazılım geliştiricilerinin katılım eksikliğine dikkat çekilmiş ve bu nedenle Brown, çalışmasını ABD'de gerçekleştirmiştir. Nicel yöntemler kullanılarak yapılan çalışmada, veriler Chow ve Cao'nun (2008) geliştirdiği anket formu kullanılarak toplanmıştır. Araştırmaya, en az iki yıllık deneyime sahip 133 çevik proje profesyoneli (proje yöneticisi, kalite güvencesi, yazılım geliştiricisi, iş analisti veya konu uzmanı) katılmıştır. Verilerin analizi, çoklu regresyon ve hiyerarşik regresyon analizleri kullanılarak gerçekleştirilmiştir. Araştırma bulgularına göre, çevik projelerde başarıyı etkileyen kritik faktörleri proje türü, dinamik proje takvimi, proje doğası, yönetim taahhüdü, teslimat

stratejisi ve proje tanımlama süreci faktörleri olduğu sonucuna ulaşılmıştır. Proje türü, kalite, zaman, kapsam ve maliyet açısından kritik bir faktör olarak öne çıkmaktadır. Proje takvimi, zaman ve kapsam üzerinde önemli bir etkiye sahipken, proje doğası kalite, zaman ve maliyet açısından etkili olmuştur. Yönetim taahhüdü ise kalite ve maliyet üzerinde belirleyici bir rol oynamaktadır. Teslimat stratejisi, kapsam ve maliyet açısından önemli bir faktör olarak belirlenmiş, proje tanımlama süreci ise kalite ve zaman üzerinde etkili bulunmuştur. Ayrıca kritik başarı faktörlerinin her birinin projenin başarısına göreli önemini belirlemiştir. Sonuçlara göre, kalite için önemli faktörler proje türü, proje niteliği, yönetim taahhüdü ve proje tanımlama süreci olarak saptanmıştır. Kapsam için proje türü, teslimat stratejisi, proje takvimi ve proje niteliği önemli bulunurken, zaman açısından proje türü, proje niteliği, teslimat stratejisi ve yönetim taahhüdü kritik rol oynamıştır. Maliyet açısından ise proje türü, proje takvimi, yönetim taahhüdü, organizasyon ortamı, teslimat stratejisi ve proje doğasının etkili olduğu belirlenmiştir.

Darwish ve Rizk (2015) çalışmasında, çevik yazılım geliştirme projelerinde başarı faktörlerini çok boyutlu bir bakış açısıyla incelemeyi amaçlamıştır. Nitel yöntemlerden literatür taraması yöntemiyle gerçekleştirilen çalışmada çevik yazılım geliştirme projelerinde başarı faktörlerini organizasyonel, insan, süreç, proje ve teknik olmak üzere beş ana boyutta sınıflandırmıştır. Her bir ana faktör, alt faktörlerle desteklenmiş ve daha anlaşılır bir yapı oluşturulmuştur. Faktörlerin çevik projelerde uygulanabilirliğini değerlendirmek için bir teorik çerçeve ve anket tabanlı bir değerlendirme yaklaşımı önermiştir.

Vithana vd. (2015), Sri Lanka yazılım endüstrisinde çevik yazılım geliştirme projelerindeki başarı faktörlerini incelemek amacıyla bir araştırma gerçekleştirmişlerdir. Misra vd. (2009) tarafından geliştirilen modele dayanan çalışmada, insan ve organizasyon faktörlerinin proje başarısı ile olan ilişkisi araştırılmıştır. Nicel yöntemlerle gerçekleştirilen çalışmada anket aracılığıyla 64 katılımcıdan veri toplanmıştır. Veriler, korelasyon analizi yöntemiyle değerlendirilmiştir. Araştırma sonuçlarına göre, insan faktörleri arasında teknik yeterlilik; organizasyonel faktörler arasında ise müşteri memnuniyeti, ekip büyüklüğü, karar verme süresi, planlama ve kurumsal kültürün çevik projelerde başarıya önemli katkı sağladığı tespit edilmiştir.

Demirel (2016) tez çalışmasında, çevik yazılım geliştirme projelerindeki kritik başarı faktörlerinin projelerin başarısına olan etkisi incelenmiştir. Araştırmada insan, proje, organizasyon, süreç ve teknik olmak üzere beş ana kategori altında toplanan faktörler değerlendirilmiştir. Proje başarısı zaman ve kalite değişkenleri ile ölçülmüştür. Nicel yöntemlerin kullanıldığı bu çalışmada, veriler 179 katılımcıdan çevrimiçi anket yoluyla toplanmış ve PLS-SEM (Partial Least Squares Structural Equation Modeling - Kısmi En Küçük Kareler Yapısal Eşitlik Modellemesi) yöntemi ile analiz edilmiştir. Araştırma sonuçları, insan, proje ve teknik faktörlerin proje başarısı üzerinde en yüksek etkiye sahip olduğunu ortaya koymuştur. Projelerin büyüklüğüne göre yapılan analizlerde, küçük projelerde teknik faktörler en belirleyici faktör iken, büyük projelerde insan ve proje faktörlerinin daha fazla etkiye sahip olduğu saptanmıştır.

Litchmore (2016) doktora tezinde, insan ve süreç faktörlerinin çevik yazılım projelerindeki başarıya etkisini, farklı çevik yöntemler çerçevesinde incelemeyi amaçlamıştır. Nicel yöntemlerin kullanıldığı araştırmada, veri toplama aracı olarak Chow ve Cao'nun (2008) tarafından geliştirilen ve geçerliliği sağlanmış anket formu kullanılmıştır. Araştırmaya, son beş yıl içinde çevik yazılım projelerinde çalışmış 174 katılımcı dâhil edilmiştir. Toplanan veriler Spearman Sıra Korelasyonu ve Kruskal-Wallis H Testi ile analiz edilmiştir. Araştırma sonuçları insan faktörlerinin (ekip yeteneği ve müşteri katılımı) ve süreç faktörlerinin (proje yönetimi ve proje tanımı) proje başarısında (maliyet, kalite, kapsam ve zaman) önemli bir rol oynadığını göstermiştir. Ancak, kullanılan çevik yöntemlerin (örneğin Scrum veya Kanban) proje başarısı üzerinde anlamlı bir fark yaratmadığı tespit edilmiştir. Bu bulgu, proje başarılarının, kullanılan yöntemlerden ziyade insan ve süreç yönetimi ile daha fazla ilişkili olduğunu ortaya koymaktadır.

Nguyen (2016) çevik yazılım geliştirme projelerinin başarısını etkileyen kritik faktörleri belirlemek amacıyla gerçekleştirmiştir. Nitel yöntemlerin kullanıldığı çalışmada veri toplama aracı olarak açık uçlu sorulardan oluşan e-posta anketleri kullanılmıştır. Veri toplama süreci, ABD'de faaliyet gösteren 10 çevik yazılım geliştirme proje yöneticisi veya lideri ile gerçekleştirilmiştir. Verilerin analizinde, Moustakas'ın modifiye edilmiş van Kaam yöntemi kullanılmıştır. Araştırma sonucunda çevik yazılım geliştirme projelerinin başarısında etkili olan kritik başarı faktörleri insan, süreç, teknik, organizasyon ile teknoloji ve geliştirme araçları olarak beş ana başlıkta toplanmıştır.

Ayrıca insan faktöründe güçlü müşteri katılımı; süreç faktöründe etkili çevik proje yönetim süreçleri; teknik faktörde çevik mühendislik teknikleri; teknoloji ve geliştirme araçları faktöründe iyi geliştirme araçları ve teknolojilerin kullanımı ve organizasyon faktöründe güçlü yönetici desteği belirleyici faktörler olarak öne çıkmıştır.

Siddique ve Hussein (2016), Norveç yazılım endüstrisinde çevik yazılım projelerindeki başarı kriterlerini incelemek amacıyla nitel bir araştırma gerçekleştirmiştir. Ayrıca, çevik tabanlı projeler ile şelale yaklaşımına dayalı projeler arasındaki başarı algısı ve yönetimi açısından benzerlikler ve farklılıklar araştırılmıştır. Araştırma kapsamında yarı yapılandırılmış görüşme formu kullanılarak 32 katılımcıdan toplanan veriler tematik analiz yöntemiyle analiz edilmiştir. Bulgular, her iki proje türünde de başarı kriterlerinin zamanında teslimat, bütçe uygunluğu ve gereksinimlerin karşılanması gibi proje yönetimi başarı ölçütlerine dayandığını göstermiştir. Bununla birlikte çevik projelerde başarı, her bir artıştan sonra düzenli olarak değerlendirilirken, müşteri memnuniyeti daha güçlü bir odak haline gelmiştir. Şelale projelerinde ise başarı genellikle proje sonunda tek bir değerlendirme ile ölçülmektedir. Bu, çevik projelerde başarının sürekli gözden geçirilmesi (başarı ölçümünün sıklığı) ve müşteri ile daha yakın iş birliği yapılması (başarı ölçümünün odağı) açısından farklılık ortaya çıkarmıştır.

Silva ve Santos (2016), çevik yazılım geliştirme projelerinin başarıya ulaşmasında etkili olan kritik başarı faktörlerini incelemek amacıyla sistematik bir inceleme gerçekleştirmiştir. Araştırmanın temel amacı, çevik yazılım geliştirme projelerinde başarıyı etkileyen faktörleri tespit etmek ve bu faktörleri insan, süreç ve teknoloji başlıkları altında sınıflandırmaktır. Araştırma kapsamında 2005-2015 yılları arasında yayımlanmış çeşitli elektronik veri tabanlarında yer alan çalışmalar taranmış ve 1387 çalışma arasından konu ile ilgili 23 ana çalışma seçilmiştir. Araştırma sonuçlarına göre 14 kritik başarı faktörü belirlenmiş ve bu faktörler üç ana kategoriye ayrılmıştır. İnsan faktörleri kategorisinde yetkinlik ve uzmanlık, yönetici desteği, ekip ve kullanıcı motivasyonu, ekip boyutu ve kullanıcı katılımı faktörleri bulunmaktadır. Süreç faktörleri arasında çevik pratiklerin kullanılması, en önemli özelliklerin önce teslim edilmesi, doğru miktarda dokümantasyon ve güçlü iletişim yer almaktadır. Teknoloji faktörleri ise uygun teknik eğitim, entegrasyon testi, basit tasarım, araç destekleri ve iyi tanımlanmış kodlama standartları olarak belirlenmiştir. Bulgular arasında dikkat çeken bir diğer unsur ise, seçilen 23 çalışmanın 9'unda Chow ve Cao'nun (2008) çevik projeler üzerine yaptığı

anket çalışmasının kullanılmış olmasıdır. Bu anket çalışması, çevik yazılım geliştirme süreçlerinde başarı ve başarısızlık faktörlerinin kapsamlı bir listesini sunmakta ve araştırmacılar tarafından önemli bir kaynak olarak değerlendirilmiştir. Araştırmanın bir diğer önemli bulgusu, çevik yöntemlerin (Scrum, XP, Kanban gibi) projelerde doğru şekilde uygulanmasının başarı oranlarını büyük ölçüde artırdığıdır. Ayrıca, ekip üyeleri arasındaki güçlü iletişim ve kullanıcı katılımı, projelerin başarılı bir şekilde yürütülmesinde kritik rol oynamaktadır.

Aldahmash vd. (2017a), çevik yazılım geliştirme projelerinin başarısında etkili olan kritik başarı faktörlerini incelemek amacıyla 2006-2016 yılları arasında gerçekleştirilen sekiz ampirik çalışmayı gözden geçirmiştir. İnceleme sonucunda, en az iki farklı çalışmada belirtilen sekiz kritik başarı faktörü belirlenmiş ve bu faktörler teknik, organizasyonel, insan ve süreç kategorileri altında sınıflandırılmıştır. Teknik kategorisinde teslimat stratejisi ve çevik geliştirme teknikleri faktörleri bulunmaktadır. Organizasyonel kategoride örgüt kültürü ve iletişim faktörleri yer almaktadır. İnsan kategorisinde takım yeteneği ve eğitimi, müşteri katılımı ve üst yönetim desteği faktörleri bulunmaktadır. Süreç kategorisinde ise proje yönetim süreci faktörü yer almaktadır.

Aldahmash vd. (2017b), çevik yazılım geliştirme projelerindeki kritik başarı faktörlerini incelemek amacıyla faktör analizine dayalı bir çalışma gerçekleştirmiştir. Araştırmada, daha önce yapılan bir literatür taramasına dayanarak belirlenen sekiz kritik başarı faktörü (Aldahmash vd., 2017a) değerlendirilmiştir. Nicel yöntemlerle yürütülen çalışmada, 28’den fazla ülkeden 131 katılımcıya anket uygulanmış ve katılımcılar, başarı faktörlerini “kesinlikle katılıyorum” ile “kesinlikle katılmıyorum” arasında değişen 7’li Likert ölçeği ile değerlendirmiştir. Faktör analizi sonuçlarına göre, kritik başarı faktörleri iki ana bileşende gruplandırılmıştır. Organizasyon ve insan faktörleri arasında örgüt kültürü, müşteri katılımı, üst yönetim desteği, takım yeteneği ve iletişim yer almaktadır. Teknik ve proje yönetim faktörleri ise proje yönetim süreci, teslimat stratejisi ve çevik yazılım geliştirme teknikleri bulunmaktadır. Ayrıca, organizasyon ve insan faktörleri bileşeninde en belirleyici faktörün örgüt kültürü, teknik ve proje yönetimi faktörlerinde ise en belirleyici faktörün proje yönetim süreci olduğunu ortaya koymuştur.

Chiyangwa ve Mnkandla (2017), Güney Afrika’daki çevik yazılım geliştirme projelerinde başarıyı etkileyen kritik başarı faktörlerini modellemeye yönelik bir çalışma

gerçekleştirmiştir. Kritik başarı faktörlerini belirlemek için organizasyon, insan, süreç, teknik ve proje olmak üzere beş ana faktör kullanılmıştır. Bu beş faktörün, performans beklentisine (algılanan fayda) ve proje başarısına etkileri incelenmiştir. Ayrıca performans beklentisinin proje başarısına etkisi de değerlendirilmiştir. Proje başarısı, kalite, kapsam, zaman ve maliyet değişkenleri ile tanımlanmıştır. Nicel veri toplama yöntemlerinin kullanıldığı çalışmada, çevik yazılım profesyonellerine yönelik çevrimiçi anket ile Güney Afrika'dan 460 katılımcıdan veri toplanmıştır. Toplanan veriler, yapısal eşitlik modellemesi ile analiz edilmiştir. Araştırma sonuçları insan, süreç, organizasyon ve proje faktörlerinin performans beklentisi (algılanan fayda) üzerinde önemli bir etkisinin olduğunu ve çevik yazılım projelerinin başarısında süreç faktörü ile performans beklentisinin belirleyici bir rol oynadığını ortaya koymuştur.

Aldahmash (2018) doktora tezinde, çevik yazılım geliştirme projelerindeki kritik başarı faktörlerini incelemeye yönelik kapsamlı bir araştırma gerçekleştirmiştir. Çalışmada, Aldahmash vd. (2017a) tarafından belirlenen sekiz kritik başarı faktörü kullanılarak, bu faktörlerin çevik yazılım projelerindeki önem sıralaması, proje ilerledikçe bu faktörlerin önem düzeylerinin değişip değişmediği ve bu faktörlerin gruplandırılması ele alınmıştır. Araştırma, 28'den fazla ülkeden 131 katılımcının yer aldığı bir anket çalışması ile yürütülmüştür. Katılımcılar, her bir faktörü “kesinlikle katılıyorum” ile “kesinlikle katılmıyorum” arasında değişen 7'li Likert ölçeği ile değerlendirmiştir. Araştırmada proje başarısı paydaş memnuniyeti, organizasyonel ihtiyaçların karşılanması, kapsam, maliyet ve zaman kriterleri ile tanımlanmıştır. Araştırma sonuçlarına göre; (1) Çevik yazılım projelerinde başarıyı etkileyen en önemli faktörler sırasıyla iletişim, müşteri katılımı, takım yeteneği ve eğitimi, üst yönetim desteği, organizasyon kültürü, çevik yazılım geliştirme teknikleri, teslim stratejisi ve proje yönetim süreci olarak belirlenmiştir. Sonuçlar, çevik geliştirme projelerinde iletişim, müşteri katılımı ve takım yeteneğinin en önemli başarı faktörleri olduğu sonucuna ulaşılmıştır. Dolayısıyla çevik projelerde organizasyon ve insan faktörleri ile ilgili faktörlerin en önemli başarı faktörleri olduğunu göstermektedir. (2) Araştırma bulguları, proje ilerledikçe proje yönetim süreci ve teslimat stratejisi faktörlerinin göreceli öneminin azaldığını ortaya koymaktadır. Buna karşın, iletişim, müşteri katılımı, organizasyon kültürü, çevik yazılım geliştirme teknikleri ve ekip yeteneği ve eğitimi gibi faktörlerin önemi proje süreci boyunca artış göstermektedir. Üst yönetim desteği

faktörünün ise proje ilerledikçe sabit bir öneme sahip olduğu, dolayısıyla proje sürecinde değişiklik göstermediği belirlenmiştir. (3) Aldahmash ve diğ. (2017b) yaptığı çalışma tezin diğer bir bulgusu olarak yer almaktadır. (4) Çevik yazılım geliştirme projelerindeki başarı ölçütleri önem sırasına göre paydaş memnuniyeti, organizasyonel ihtiyaçların karşılanması, kapsam, maliyet ve zaman olduğu sonucuna ulaşılmıştır.

Bergmann (2018) doktora tezinde, çevik proje yönetiminin benimsenmesinin proje yöneticileri tarafından algılanan proje başarısı üzerindeki etkilerini belirlemeyi ve aynı zamanda proje karmaşıklığının proje sonuçlarına olan etkisini araştırmayı amaçlamıştır. Araştırmada, çevik proje yönetimini değerlendirmek için başarı faktörleri kullanılmış ve Chow ve Cao'nun (2008) geliştirdiği modele dayanarak yönetim faktörü eklenmiştir. Çevik proje yönetimi; yönetim, süreç, proje, organizasyon, insan ve teknik faktörler çerçevesinde ele alınmıştır. Proje başarısı; kapsam, kalite, maliyet ve zaman boyutlarında incelenmiş, proje karmaşıklığı ise proje büyüklüğü (bütçe), proje süresi, müşterinin durumu (harici/dahili), ekip üye sayısı, ekip dağılımı (coğrafi) ve proje sponsor sayısı gibi unsurlarla ölçülmüştür. Çalışma, en az lisans derecesine sahip ve bir yıllık deneyimi olan 352 proje yöneticisine nicel yöntemler (anket) ile uygulanmıştır. Veriler, yapısal eşitlik modeli kullanılarak analiz edilmiştir. Araştırma sonuçları, çevik proje yönetimi ile proje başarısı arasında pozitif bir ilişki olduğunu, proje karmaşıklığı ile proje başarısı arasında ise zayıf bir negatif ilişki olduğunu ortaya koymuştur. Çevik proje yönetimini etkileyen altı faktör arasından süreç, organizasyon ve insan faktörleri en önemli unsurlar olarak öne çıkmıştır. Ayrıca proje karmaşıklığı açısından ekip üye sayısı en belirleyici faktör olarak tespit edilmiştir. Proje başarısında ise kalite en belirleyici faktör olarak bulunmuştur.

Stanberry (2018) doktora çalışmasını, Scrum metodolojisini kullanan büyük ve dağıtılmış çevik yazılım geliştirme projelerinde başarıyı etkileyen kritik başarı faktörlerini belirlemek amacıyla gerçekleştirmiştir. Nicel yöntemlerin kullanıldığı çalışmada, veri toplama aracı olarak Chow ve Cao (2008) tarafından geliştirilen anket formu uygulanmıştır. Araştırmaya, çeşitli rollerde görev yapan (ürün sahipleri, Scrum ustaları, yazılım geliştiricileri, iş analistleri ve/veya test uzmanları) toplam 132 profesyonel katılmıştır. Veriler, çoklu regresyon modeli ile analiz edilmiştir. Araştırma bulgularına göre, çevik yazılım geliştirme projelerinde başarıyı etkileyen kritik faktörler proje boyutlarına göre farklılık göstermektedir. Kalite boyutunda takım yeteneği ve teslim stratejisi; kapsam boyutunda takım yeteneği, teslim stratejisi ve proje doğası; zaman

boyutunda proje tanımlama süreci ve teslim stratejisi; maliyet boyutunda ise proje yönetim süreci ve proje tanımlama süreci kritik başarı faktörleri olarak tespit edilmiştir. Bulgulara göre en önemli kritik başarı faktörü teslim stratejisi iken, bunu takım yeteneği ve proje tanımlama süreci takip etmektedir. Çalışma, çevik projelerde başarının sağlanmasında insan ve süreç yönetiminin kritik bir rol oynadığını vurgulamaktadır.

Syeda (2018), doktora çalışmasında çevik sistem mühendisliği projelerinin kritik başarı faktörlerini belirlemek ve bu faktörlerin birbirlerine göre nispi önemini incelemek amacıyla bir araştırma gerçekleştirmiştir. Araştırma, nicel bir yöntemle yürütülmüş olup, çevrimiçi anket yoluyla 46 katılımcıdan veri toplanmıştır. Çalışmada, çevik sistem mühendisliği projelerindeki başarı faktörleri; teslimat stratejisi, değişim yönetimi, iletişim ve iş birliği, çevik metodoloji taahhüdü, ekip ortamı ve dinamikleri, proje yürütme ve gözetim, sürdürülebilirlik ve altyapı olmak üzere sekiz temel faktör olarak belirlenmiştir. Proje başarısı ise zaman, maliyet, kapsam ve müşteri beklentilerinin karşılanması kriterlerine göre değerlendirilmiştir. Verilerin analizinde Pearson korelasyon ve Mann-Whitney U testi kullanılarak çevik sistem mühendisliğinin kritik başarı faktörleri tespit edilmiştir. Araştırma sonuçları, teslimat stratejisi, değişim yönetimi, iletişim ve iş birliği, çevik metodoloji taahhüdü, ekip ortamı ve dinamikleri ile proje yürütme ve gözetim faktörlerinin çevik sistem mühendisliği projelerinde başarı için kritik öneme sahip olduğunu göstermiştir. Ayrıca, başarı faktörlerinin birbirlerine göre nispi önemini belirlemek amacıyla Analitik Hiyerarşi Süreci (AHP) yöntemi kullanılmış ve ekip ortamı ve dinamiklerinin en önemli faktör olduğu sonucuna ulaşılmıştır.

Bergmann ve Karwowski (2019) tarafından gerçekleştirilen çalışma, çevik proje yönetiminin proje başarısı üzerindeki etkilerini incelemeyi amaçlayan bir literatür taramasıdır. Araştırma, çevik proje yönetimini geleneksel proje yönetimiyle karşılaştırarak, proje karmaşıklığı faktörlerinin etkilerini ve proje başarısı için kritik başarı faktörlerini ele almıştır. Çalışma, nitel bir literatür taraması yöntemine dayanmaktadır. Araştırma sonuçları, günümüzün karmaşık ve hızla değişen iş ortamında geleneksel proje yönetiminin sınırlamalarını doğrularken, bu yaklaşımın karmaşıklığa ve değişen müşteri gereksinimlerine uyum sağlamak için esneklikten yoksun olduğunu göstermektedir. Çevik proje yönetimi, sürekli değişen gereksinimler ve proje yürütmede yakın müşteri katılımı ile yinelemeli bir planlama sürecine izin vermek için yeterli esneklik sağlamaktadır. Proje başarısı, zaman, maliyet ve kalite gibi orijinal proje başarı

boyutları, kapsam ve paydaş memnuniyeti ile genişletilmiştir. Literatür incelemesi sonucunda, çevik proje yönetimi boyutları yönetim, süreç, proje, organizasyon, insanlar ve teknik faktörler olmak üzere altı kategoride tanımlanmıştır.

Ergüder (2019) tez çalışmasında, farklı bilgi teknolojileri proje yönetimi yaklaşımlarındaki (geleneksel, çevik ve hibrit) kritik başarı faktörlerini belirlemeyi ve bu faktörler arasındaki benzerlik ve farklılıkları analiz etmeyi amaçlamıştır. Bu doğrultuda, literatürde yer alan kritik başarı faktörleri incelenmiş ve proje kritiklik seviyesi, üst yönetim desteği, son kullanıcı, proje ekibi, proje yönetim süreci, proje geliştirme süreci ve şirket kültürü olmak üzere yedi ana başlık altında değerlendirilmiştir. Proje başarısı ise kalite, zaman, maliyet ve müşteri memnuniyeti değişkenleri ile ölçülmüştür. Nicel yöntemlerle gerçekleştirilen araştırmada, veriler anket aracılığıyla toplanmış ve 178 farklı projeden elde edilen veriler çoklu regresyon analizi ile değerlendirilmiştir. Araştırma sonuçlarına göre, geleneksel proje yönetimi yaklaşımlarında başarıyı etkileyen temel faktörler proje ekibi ve proje geliştirme süreci olarak belirlenirken, çevik proje yönetimi yaklaşımlarında en etkili faktörlerin proje kritiklik seviyesi ve proje yönetim süreci olduğu tespit edilmiştir. Hibrit proje yönetimi yaklaşımlarında ise başarıyı etkileyen faktörler, geleneksel yöntemlerden proje ekibi ve çevik yöntemlerden proje yönetim süreci olarak belirlenmiştir.

Garousi vd. (2019) tarafından gerçekleştirilen bu çalışma, yazılım projelerinin başarısını etkileyen kritik başarı faktörleri (KBF) ile proje başarısı arasındaki korelasyonu incelemeyi amaçlamaktadır. Araştırmada, Ahimbisibwe vd. (2015) tarafından geliştirilen durumsal uygunluk modeli uyarlanarak ve genişletilerek uygulanmıştır. Çalışmada, yazılım proje başarı faktörleri olan organizasyon, takım, müşteri ve proje faktörlerinin, proje başarısı (süreç, ürün kalitesi ve paydaş memnuniyeti) ile olan ilişkisi incelenmiştir. Nicel yöntemlerle gerçekleştirilen çalışmada, veriler anket aracılığıyla 101 katılımcıdan toplanmış ve toplanan veriler üzerinde korelasyon analizi yapılmıştır. Araştırma bulguları, yazılım projelerindeki başarı ile en güçlü ilişkiye sahip üç kritik başarı faktörünün ekibin yazılım geliştirme metodolojileri konusundaki deneyimi (takım faktörü), ekibin görev konusundaki uzmanlığı (takım faktörü) ve proje izleme ve kontrol süreçleri (organizasyon faktörü) olduğunu ortaya koymuştur. Ancak müşteri faktörleri ve bazı proje faktörlerinin başarı ile olan ilişkisinin neredeyse önemsiz olduğu sonucuna ulaşılmıştır.

Naslund ve Kale (2020) çevik dönüşüm süreçlerinin başarısını etkileyen kritik faktörlerini belirlemek ve yapılandırılmış bir çerçeve sunmak amacıyla çalışma gerçekleştirmiştir. Sistematik literatür taraması yöntemine 2008-2018 yılları arasını kapsayan 207 makale belirlenmiş, bu makalelerin özetleri ve içerikleri gözden geçirilerek çalışma kapsamına en uygun 23 makale incelemeye dahil edilmiştir. Araştırmada çevik dönüşüm ile ilgili olarak 23 makalede toplam 103 kritik başarı faktörü tespit edilmiştir. Veri analizinde tematik analiz yöntemi kullanılmıştır. Literatürde yer alan kritik başarı faktörleri önce 13 ana kategoriye ayrılmış ve ardından bu kategoriler; amaç, süreç ve insan olmak üzere üç ana küme altında toplanmıştır. Araştırma bulguları, çevik dönüşüm süreçlerinde en önemli kritik başarı faktörlerinin üst yönetimin desteği, beceri ve yetkinliklerin geliştirilmesi, çalışanların eğitimi ve örgüt kültürü olduğunu ortaya koymaktadır.

Shakya ve Shakya (2020), Nepal yazılım endüstrisinde çevik metodolojinin kritik başarı faktörlerini incelemek amacıyla nicel bir araştırma gerçekleştirmişlerdir. Bu amaçla, yapılandırılmış anket kullanılarak 178 katılımcıdan veri toplanmıştır. Araştırmada organizasyon, çevre, teknik, proje, ekip ve müşteri faktörlerinin proje başarısına etkisini belirlemek amacıyla regresyon analizi yapılmıştır. Proje başarısını ölçmek için iyileştirilmiş ürün kalitesi, zamanında teslimat, müşteri memnuniyeti, proje risklerinin azaltılması, değişikliklere hızlı yanıt verme yeteneği ve yönetsel etkinliğin artırılması değişkenleri kullanılmıştır. Araştırmanın bulguları, çevre faktörleri hariç tüm diğer faktörlerin proje başarısına anlamlı bir katkı sağladığını ortaya koymuştur. Özellikle örgütsel yapı, iş süreçlerinin düzgün uygulanması, ekip yetkinliği ve müşteri katılımı gibi faktörler proje başarısının en önemli belirleyicileri olarak öne çıkmıştır.

Tam vd. (2020), devam eden çevik yazılım geliştirme projelerinin başarısını etkileyen insan faktörlerini incelemek amacıyla beş insan faktöründen oluşan bir model önermiştir. Çalışmada proje başarısı maliyet, zaman ve müşteri memnuniyeti açısından değerlendirilmiştir. Literatürdeki çalışmalardan geliştirilen anket formu ile 216 katılımcıdan veri toplanmıştır. Verilerin analizi, PLS-SEM yöntemiyle gerçekleştirilmiştir. Araştırma bulgularına göre, çevik yazılım projelerinin başarısına en fazla katkı sağlayan faktörler takım yeteneği ve müşteri katılımı olarak belirlenmiştir.

Tsoy ve Staples (2020), çalışmalarında çevik analitik projelerde başarıyı etkileyen kritik başarı faktörlerini incelemiştir. Özellikle çevik proje yönetimi yaklaşımının analitik projelerde uygulanabilirliği ve başarıya etkisi üzerinde durulmuştur. Araştırma, iki analitik projeyi inceleyen çoklu vaka incelemesi stratejisini kullanmıştır. Veri toplama aracı olarak proje ekip üyeleri, liderler ve müşterilerle yapılan derinlemesine mülakatlar kullanılmış; her biri yaklaşık 1,5 saat süren toplam 20 mülakat gerçekleştirilmiştir. Veriler, Chow ve Cao'nun (2008) başarı faktörlerine dayalı olarak kodlanmıştır. Chow ve Cao'nun (2008) çevik projeler için belirledikleri orijinal başarı faktörleri listesi, analitik projelere uyarlanarak güncellenmiştir. Listeye 10 yeni nitelik eklenmiş ve 3 özellik ise kaldırılmıştır. Veriler çapraz vaka analiziyle değerlendirilmiştir. Bulgular, Clover projesinin Pars projesine kıyasla neredeyse tüm başarı faktörlerinde daha güçlü performans gösterdiğini ortaya koymuştur. Özellikle yönetim desteği, müşteri katılımı, takım yeteneği ve proje yönetim süreçleri başarılı projelerde daha belirgindir. Verilerin kalitesi ve model doğrulama faaliyetlerinin önemi analitik projelerde kritik rol oynamaktadır. Ayrıca risk alma isteği ile organizasyonun bilgi işleme kapasitesinin uyumu, çevik projelerin başarısında önemli bir faktör olarak vurgulanmıştır.

Kukreja vd. (2021) tarafından gerçekleştirilen çalışma, çevik yazılım geliştirme projelerinde kritik başarı faktörlerini tanımlamak, değerlendirmek ve önemine göre sıralamak amacı taşımaktadır. Nicel yöntemlerin kullanıldığı çalışmada, veriler anket aracılığıyla Hindistan'da faaliyet gösteren ve çevik yazılım geliştirme uygulamalarını benimseyen 205 katılımcıdan toplanmıştır. Toplanan veriler üzerinde faktör analizi gerçekleştirilmiş ve başarı faktörleri, proje, organizasyon, teknik, insan ve süreç olmak üzere beş temel faktör altında toplanmıştır. Çalışmada proje başarısı, kapsam, kalite, maliyet ve zaman değişkenlerini içeren bir yapı olarak ele alınmıştır. Araştırmanın bulgularına göre, çevik yazılım geliştirme projelerinde en önemli başarı faktörü proje faktörü olduğu ve bunu sırasıyla organizasyon, teknik, insan ve süreç faktörlerinin izlediği sonucuna ulaşılmıştır.

Qatanani vd. (2021), COVID-19 pandemisi sırasında Ürdün'deki çevik yazılım geliştirme projelerinin başarısına katkıda bulunan kritik başarı faktörlerini incelemeyi amaçlamaktadır. Çalışma özellikle insan ve organizasyon faktörlerine odaklanmıştır. Araştırma, nicel yöntem kullanılarak, çevrimiçi anketler ile 114 katılımcıdan toplanan verilerle gerçekleştirilmiştir. Verilerin analizinde frekans ve tanımlayıcı istatistikler

kullanılmıştır. Araştırma sonuçlarına göre, takım yeteneği ve motivasyonun çevik projeleri olumlu etkileyebilecek en önemli başarı faktörleri olduğu tespit edilmiştir.

Azhar ve Abdullah'ın (2022) çalışması, çevik yazılım geliştirme projelerinin başarısına katkıda bulunan kritik başarı faktörlerini belirlemeyi ve bu faktörlerin proje performansını nasıl etkilediğini anlamayı amaçlamaktadır. Çalışma, nitel araştırma yöntemleri ile gerçekleştirilmiş olup, 2013-2021 yılları arasında yayımlanmış 47 çalışmanın literatür taraması yoluyla incelenmesine dayanmaktadır. Analiz edilen kritik başarı faktörleri organizasyon, teknik, süreç, takım ve proje olmak üzere beş ana boyut altında sınıflandırılmıştır. Bulgular, özellikle organizasyonel ve insan faktörlerinin çevik geliştirme projelerinin başarısında önemli bir rol oynadığını ortaya koymaktadır. Müşteri katılımı, en sık tekrar edilen ve çevik ilkelerle uyumlu olan faktörlerden biri olarak öne çıkmıştır. Ayrıca, takım yeteneği ve takım bağlılığı, başarıyı etkileyen önemli faktörler olarak belirlenmiştir. Bunun yanı sıra, gelişmiş teknoloji kullanımı ve takım motivasyonu da başarıya önemli katkılarda bulunmuştur. Organizasyon ortamı, organizasyon boyutunda en yüksek değere sahip olup, bu ortamın çevik uygulama sürecinin olumlu sonuçlanmasına katkı sağladığı görülmektedir. Ayrıca, yönetsel bağlılık, etkin iletişim ve eğitim sağlama gibi faktörler de yaygın olarak vurgulanmış olup, Çevik süreç iyileştirmenin organizasyonun desteği ile daha etkin bir şekilde benimsenebileceği sonucuna ulaşılmıştır.

Bogopa ve Marnewick (2022) çalışmasında, yazılım geliştirme projelerindeki kritik başarı faktörleri incelenmiştir. Çalışmada kritik başarı faktörleri insan, süreç ve teknik boyutlarıyla ele almıştır. Proje başarısı ise bütçe, kalite ve zaman ölçütleri ile değerlendirilmiştir. Araştırma, nicel yöntem kullanılarak gerçekleştirilmiş ve yapılandırılmış bir anket aracılığıyla 212 yazılım geliştirme uzmanından veri toplanmıştır. Veriler, SPSS yazılımı kullanılarak analiz edilmiş ve özellikle korelasyon analiziyle kritik başarı faktörleri ile proje başarı ölçütleri arasındaki ilişkiler incelenmiştir. Bulgular, projelerde en kritik başarı faktörlerinin “bağlı ve motive bir ekip”, “kullanıcı/müşteri katılımı”, “iyi liderlik” ve “açık gereksinim ve hedefler” olduğunu ortaya koymuştur.

Fadilah vd. (2022), Endonezya'daki çevik yazılım geliştirme projelerindeki insan faktörlerinin başarı üzerindeki etkisini incelemek amacıyla çalışma gerçekleştirmiştir.

Çalışmada bireysel özellikler, toplumsal kültür, takım yetenekleri ve müşteri katılımı değişkenlerinin çevik yazılım geliştirme projelerinin başarısına etkisi araştırılmıştır. Nicel yöntemlerle gerçekleşen çalışmada veriler anket yoluyla 48 katılımcıdan veri elde edilmiştir. Veriler, PLS-SEM ile analiz edilmiştir. Araştırma sonuçları takım yeteneklerinin proje başarısını önemli ölçüde etkilediğini, ancak bireysel özellikler, toplumsal kültür ve müşteri katılımının doğrudan bir etkisinin olmadığını göstermektedir. Bununla birlikte, bireysel özelliklerin takım yetenekleri ve müşteri katılımı üzerinde dolaylı bir etkisi olduğu tespit edilmiştir.

Greene (2022) doktora çalışmasında, çevik bilgi teknolojisi projelerinde organizasyonel, teknik ve süreç faktörlerinin proje başarısına olan etkisinde takım üyelerinin ortalama yaşının düzenleyici rolünü araştırmıştır. Nicel yöntemlerle gerçekleştirilen çalışmada veriler, Chiyangwa ve Mnkandla'nın (2017) çalışmalarından uyarlanarak hazırlanan anket aracılığıyla 103 çevik proje yöneticisinden toplanmıştır. Veriler regresyon analizi ile değerlendirilmiştir. Organizasyonel, teknik ve süreç faktörleri arasında yüksek düzeyde korelasyon görülmesi nedeniyle bu üç faktör analizlerde birleştirilmiş ve tek bir faktör (toplam faktör) olarak ele alınmıştır. Araştırma sonuçları, toplam faktörünün (organizasyonel, teknik ve süreç) çevik bilgi teknolojisi projelerinde başarı üzerinde anlamlı bir etkiye sahip olduğunu göstermiştir. Ancak, takım üyelerinin ortalama yaşının bu ilişki üzerinde düzenleyici bir etkisinin olmadığı tespit edilmiştir.

Haskin (2022) doktora çalışmasını, çevik yazılım geliştirme projelerinde başarıyı etkileyen kritik başarı faktörlerini araştırmak amacıyla gerçekleştirmiştir. Nicel yöntemlerin kullanıldığı çalışmada, veri toplama aracı olarak Chow ve Cao (2008) tarafından geliştirilen anket formu kullanılmıştır. Araştırma kapsamında 82 katılımcıdan toplanan veriler regresyon analizi ile değerlendirilmiştir. Araştırma sonuçlarına göre, çevik yazılım geliştirme projelerinde başarıyı etkileyen en önemli üç kritik başarı faktörü sırasıyla takım yeteneği, çevik yazılım mühendisliği teknikleri ve güçlü yönetim taahhüdü olarak belirlenmiştir.

Yeşilyurt (2022) tez çalışmasında, çevik proje yönetim yaklaşımlarının proje başarısı üzerindeki etkileri incelenmiştir. Proje başarısı, kalite, zaman, maliyet ve kapsam olmak üzere dört ana boyutta değerlendirilirken; çevik proje yönetimi süreç,

organizasyon, yönetim, insan, proje ve teknik olmak üzere altı boyutta ele alınmıştır. Nicel araştırma yöntemlerinin kullanıldığı çalışmada veriler, Chow ve Cao (2008)'nın çalışmasından uyarlanan ve Bergmann (2018) tarafından “yönetim” boyutu eklenerek revize edilen anket formu aracılığıyla toplanmıştır. Anket çevrimiçi olarak 104 katılımcıya uygulanmış ve elde edilen verilerin analizi yapısal eşitlik modeli (YEM) kullanılarak gerçekleştirilmiştir. Araştırma sonuçları, çevik proje yönetim yaklaşımları ile proje başarısı arasında pozitif ve güçlü bir ilişki olduğunu ortaya koymuştur. Çevik proje yönetiminde süreç boyutunun en etkili faktör olduğu, insan faktörünün de önemli bir rol oynadığı belirlenmiştir. Buna karşın, teknik faktörün çevik proje yönetimi üzerinde anlamlı bir etkisinin bulunmadığı saptanmıştır. Proje başarısı açısından ise kalite ve zaman boyutları en belirleyici unsurlar olarak öne çıkmıştır.

Heper (2023) doktora tez çalışmasında, çevik proje yönetiminde başarıyı etkileyen kritik başarı faktörlerini araştırmıştır. Araştırmada, çevik başarı faktörleri yönetim ve iklim, proje takımı, müşteri, yetkin çalışanlar, bilgi sistemleri ve proje olmak üzere altı ana kategori altında değerlendirilmiştir. Proje başarısı ise kalite, kapsam, zaman, maliyet ve paydaş memnuniyeti değişkenleri ile ölçülmüştür. Nicel araştırma yöntemleri kullanılarak 210 katılımcıdan toplanan veriler, hiyerarşik regresyon ve YEM kullanılarak analiz edilmiştir. Araştırma bulguları, çevik proje yönetimi yaklaşımlarında yönetim ve iklim ile proje faktörlerinin kritik başarı faktörleri olduğunu ortaya koymuştur.

Ojha (2023), çevik yazılım geliştirme projelerinde başarıya katkıda bulunan kritik başarı faktörlerini belirlemek ve incelemek amacıyla çalışma gerçekleştirmiştir. Nitel yöntemlerle gerçekleştirilen çalışmada sistematik literatür taraması yöntemi kullanılarak, 14 makale çalışmanın analizine dahil edilmiştir. Veriler, tematik analiz yöntemi ile incelenmiş ve kritik başarı faktörleri teknik, insan, organizasyonel ve süreç faktörleri olarak dört ana kategoriye ayrılmıştır. Bulgular, çevik yazılım projelerinde başarıyı etkileyen sekiz temel faktörün öne çıktığını göstermektedir. Bunlar; çevik geliştirme teknikleri, teslim stratejisi, proje takımı eğitimi, müşteri ve çalışan katılımı, yönetim desteği, organizasyon kültürü, iletişim ve proje yönetim süreçleridir. Bu faktörler arasında özellikle müşteri katılımı ve takım eğitimi projelerin başarısına önemli katkılar sağlarken, organizasyon kültürü ve yönetim desteğinin de çevik projelerin başarısında kilit rol oynadığı vurgulanmaktadır.

Barros vd. (2024), çevik yazılım geliştirme projelerinde insan faktörlerinin başarı üzerindeki kritik rolünü incelemiştir. Araştırma çevik projelerde başarıya etki eden dört temel insan ilişkili faktörü ele almıştır. Bunlar; takım yetkinliği, müşteri katılımı, psikolojik güvenlik ve takım özerkliğidir. Proje başarısı ise proje maliyet, zaman, müşteri memnuniyeti ve takım üyeleri memnuniyeti değişkenleriyle ölçülmüştür. Nicel yöntemlerle gerçekleştirilen çalışmadan veriler, 177 katılımcıdan anket aracılığıyla toplanmış ve PLS-SEM yöntemi ile analiz edilmiştir. Araştırma sonuçları takım yetkinlikleri ve müşteri katılımının çevik projelerin başarısı üzerinde doğrudan en güçlü etkiye sahip olduğunu göstermiştir. Ayrıca, psikolojik güvenliğin, takım yetkinlikleri ve müşteri katılımı aracılığıyla dolaylı bir etkiye sahip olduğu bulunmuştur. Takım özerkliği ise psikolojik güvenlik üzerinde ters yönde etkili olduğu sonucuna ulaşılmıştır.

Binboğa ve Gümüşsoy (2024) çalışmasında, çevik yazılım geliştirme projelerinde kritik başarı faktörlerinin modellenmesi ve belirlenmesi amaçlanmıştır. Araştırma kapsamında çevik başarı faktörleri, müşteri, ekip, organizasyon, süreç, teknik ve proje olmak üzere altı ana kategori altında değerlendirilmiştir. Proje başarısı ise süreç verimliliği, sürdürülebilir yazılım kalitesi ve paydaş memnuniyeti boyutlarıyla ölçülmüştür. Veriler, çevrimiçi anket aracılığıyla 596 çevik uygulayıcıdan toplanmış ve YEM kullanılarak analiz edilmiştir. Araştırma bulgularına göre, müşteri, ekip, süreç ve proje faktörlerinin çevik proje başarısını belirleyen en önemli unsurlar olduğu tespit edilmiştir. Özellikle çevik süreç ve müşteri faktörlerinin, sürdürülebilir yazılım ürün kalitesi üzerinde diğer faktörlere kıyasla daha güçlü bir etkiye sahip olduğu bulunmuştur. Buna karşın, teknik faktörlerin paydaş memnuniyeti üzerinde; organizasyonel faktörlerin ise sürdürülebilir yazılım kalitesi üzerinde anlamlı bir etkisi olmadığı sonucuna varılmıştır. Bu sonuçlar, çevik yazılım projelerinde müşteri ilişkileri ve süreç yönetiminin başarıyı artırmada kilit bir rol oynadığını ortaya koymaktadır.

Fry (2024) doktora çalışmasında, çevik yazılım geliştirme projelerinde teknik ve sosyal faktörlerin proje başarısına etkilerini incelemiştir. Bu amaç doğrultusunda yazılım geliştirme ekip büyüklüğü, ekip deneyimi ve çaba tahmini doğruluğu faktörlerinin çevik yazılım geliştirme projelerinin başarısına etkisini araştırmıştır. Proje başarısı zaman, maliyet, kapsam ve kalite boyutlarıyla ele alınmıştır. Nicel yöntemlerle gerçekleştirilen çalışmada veriler Chow ve Cao'nun (2008) çalışmasında tanımlanan çevik yazılım projeleri başarı kriterlerine dayandırılarak hazırlanan anket aracılığıyla 61 katılımcıdan

elde edilmiştir. Elde edilen veriler regresyon analiziyle değerlendirilmiştir. Araştırma sonuçları, yazılım geliştirme ekip büyüklüğü, deneyim düzeyi ve çaba tahmini doğruluğunun, proje başarısındaki varyansın %9'unu açıkladığını, ancak bu değişkenlerin istatistiksel olarak anlamlı bir etkisinin olmadığını göstermektedir.

Gwangwadza ve Hanslo'nun (2024) çalışması, Güney Afrika yazılım geliştirme sektöründe çevik yazılım geliştirme projelerinin başarıya ulaşmasını sağlayan kritik başarı faktörlerinin belirlenmesi ve önceliklendirilmesini amaçlamaktadır. Araştırma, nitel bir yaklaşımla gerçekleştirilmiş ve vaka çalışması yöntemi kullanılmıştır. Çalışmada veriler, çevik yazılım geliştirme projelerinde yer alan yazılım geliştiriciler, kullanıcı deneyimi tasarımcıları, proje yöneticileri ve teknik liderler gibi farklı pozisyonlardan 15 uygulayıcı ile yapılan yarı yapılandırılmış mülakatlar aracılığıyla toplanmıştır. Mülakatlardan elde edilen veriler tematik analiz yöntemiyle incelenmiş ve yazılım kullanılarak kodlanmıştır. Sonuç olarak, 25 kritik başarı faktörü tespit edilmiş ve bu faktörler organizasyonel, ekip, müşteri, süreç, teknik ve proje olmak üzere altı ana kategoriye ayrılmıştır. Araştırma sonuçları örgüt kültürünün en önemli başarı faktörü olduğunu ve bunu yönetim desteği ile ekip yetkinliklerinin takip ettiğini göstermektedir. Çalışma, ayrıca Güney Afrika bağlamında toplumsal kültür gibi dışsal faktörlerin başarı üzerindeki etkisinin görece daha düşük olduğunu ortaya koymuştur. Araştırmanın sonunda, organizasyon yapısının ve ekip dinamiklerinin çevik projelerin başarısı üzerindeki kritik önemi vurgulanmıştır.

Junior vd. (2024) çevik yazılım geliştirme projelerinde başarıyı etkileyen kritik başarı faktörlerini belirlemek ve bu faktörlerin proje başarısı üzerindeki etkilerini incelemek amacıyla çalışma gerçekleştirmiştir. Çalışma, literatürdeki mevcut bulgulara dayalı olarak yapılandırılmış bir nitel tarama süreci ile başlayarak, çevik yazılım geliştirme alanında deneyimli profesyonellerden elde edilen nicel verilerle desteklenmiştir. Araştırma kapsamında 121 katılımcıdan toplanan veriler, PLS-SEM kullanılarak analiz edilmiştir. Çalışmanın bulguları, proje, insan, organizasyon, süreç ve kültürel faktörlerin, çevik yazılım geliştirme projelerinin başarısında kritik bir rol oynadığını ortaya koymuştur. Ayrıca faktörlerin bir arada ele alındığında daha büyük bir etki yarattığı sonucuna ulaşılmıştır.

Moustafa ve Abdulrasoul (2024) çalışması, çevik yazılım geliştirme takımlarının başarısını etkileyen faktörleri incelemeyi amaçlamaktadır. Araştırmanın temel amacı, Scrum yöntemini kullanan yazılım geliştirme takımlarının başarıya ulaşmasında kritik rol oynayan faktörleri belirlemektir. Bu doğrultuda, kapsamlı bir literatür taraması gerçekleştirilmiş ve tarama sonucunda 23 kritik başarı faktörü tespit edilmiştir. Çalışmada, bu başarı faktörleri takım ortamı, takım iletişimi, takım üyeleri ve proje özellikleri olarak dört ana başlık altında sınıflandırılmıştır.

Zhang vd. (2024) çalışması, çevik yazılım geliştirme projelerindeki kritik başarı faktörlerini incelemeyi ve bu faktörleri sınıflandırmayı amaçlamaktadır. Kapsamlı bir literatür taraması yoluyla çevik yazılım geliştirme için belirlenen 34 kritik başarı faktörü organizasyon, insan, süreç, proje, takım ve müşteri olarak altı boyutta toplanmıştır. Araştırma sonuçları, insan, proje, ekip ve organizasyon boyutlarının çevik geliştirme projelerinin başarısında önemli bir rol oynadığını ortaya koymuştur. Ayrıca yönetim desteği, aktif iletişim, müşteri katılımı ve takım yetenekleri gibi unsurların da çevik yazılım projelerinin başarısında kritik olduğu belirlenmiştir.

4.2. ÇEVİK PRATİKLERLE İLGİLİ ÇALIŞMALAR

Pikkarainen vd. (2008) çalışmalarında, çevik pratiklerin yazılım geliştirme süreçlerindeki iletişim üzerindeki etkilerini incelemeyi amaçlamışlardır. Çalışmada kullanılan çevik pratikler arasında sprint planlama, ayaküstü toplantılar, açık ofis alanı, hikâye ve görev panoları, küçük sürümler, retrospektif, ürün ve sprint iş listesi, eşli programlama, sürekli entegrasyon ve sprint incelemesi yer almıştır. Çalışma, F-Secure Corporation'da yürütülen iki yazılım geliştirme projesinde bir vaka çalışması olarak gerçekleştirilmiştir. Araştırma, nitel yöntemler kullanılarak yapılmış ve veriler yarı yapılandırılmış görüşmeler ve gözlemler yoluyla toplanmıştır. Araştırma bulgularına göre, çevik pratikler hem iç iletişim (ekip içindeki iletişim) hem de dış iletişimi (yazılım ekibi ile diğer paydaşlar arasındaki iletişim) geliştirmiştir. Özellikle Scrum ve XP yöntemlerinin kullanımının, ekip içi ve paydaşlar arası iletişimdeki bağımlılıkları yönetmede olumlu etkileri olduğu görülmüştür. Ancak, dış paydaşlarla iletişimde bazen eksikliklerin yaşandığı ve çevik pratiklerin bazı durumlarda iletişim mekanizmalarının yetersiz kalmasına neden olabileceği de tespit edilmiştir.

Maruping vd. (2009) çalışması, kontrol teorisi perspektifinden çevik metodolojilerin kullanımını ve değişen kullanıcı gereksinimlerinin yazılım projeleri üzerindeki etkilerini incelemeyi amaçlamaktadır. Araştırmanın temel amacı, proje yöneticilerinin çevik geliştirme metodolojilerini en etkin şekilde nasıl kullanabileceklerini ve bu süreçte kullanıcı gereksinimlerindeki değişikliklerle nasıl başa çıkabileceklerini anlamaktır. Çevik metodolojilerin kullanımı, altı temel pratik çerçevesinde ele alınmıştır. Bu pratikler çift programlama, sürekli entegrasyon, yeniden kod düzenleme, birim testi, ortak kod sahipliği ve kodlama standartlarıdır. Araştırmada proje başarısı, hata şiddeti, bileşen karmaşıklığı, koordinatif karmaşıklık ve dinamik karmaşıklık gibi dört ana kriter ile ölçülmüştür. Nicel bir yöntemle gerçekleştirilen bu çalışmada, çevrimiçi anketler aracılığıyla 862 yazılım geliştiriciden veri toplanmıştır. Verilerin analizinde PLS-SEM yöntemi kullanılmıştır. Araştırma sonuçları, çevik metodolojilerin özellikle sürekli entegrasyon ve test odaklı geliştirme gibi pratiklerin proje kalitesini artırdığını göstermektedir. Ayrıca, yüksek gereksinim değişikliği koşullarında çevik metodolojilerin kaliteyi iyileştirdiği, ancak özerklik gibi kontrol mekanizmalarının bu ilişkiye olumlu ya da olumsuz etkiler yapabileceği tespit edilmiştir.

Abrantes ve Travassos (2011) çalışmalarında, yazılım geliştirme süreçlerinde kullanılan çevik pratiklerin yaygınlığını ve etkisini sistematik olarak incelemeyi amaçlamışlardır. Araştırma, nitel bir araştırma olup, sistematik literatür taraması yöntemiyle gerçekleştirilmiştir. 1998-2010 yılları arasında yayımlanmış çalışmalardan veri toplanmıştır. Araştırmanın veri toplama aşamasında, 441 çalışma incelenmiş ve bu çalışmalardan çevik süreçlerle ilişkili 51 pratik belirlenmiştir. Araştırmacılar, bu pratikler arasından en yaygın kullanılan 17 çevik pratik üzerinde yoğunlaşmıştır. Veriler, sistematik tarama kriterlerine göre filtrelenerek analiz edilmiş ve çevik pratiklerin kullanımı üzerine elde edilen bulgular değerlendirilmiştir. Analiz sonuçlarına göre, en sık kullanılan çevik pratikler arasında test odaklı geliştirme, sürekli entegrasyon, eşli programlama, planlama oyunu, müşteri katılımı, ortak kod sahipliği, küçük sürümler, metafor kullanımı, yeniden kod düzenleme, sürdürülebilir hız, basit tasarım ve kodlama standartları yer almaktadır. Çalışmanın bulguları, bu pratiklerin yazılım süreçlerinde yaygın olarak uygulandığını ve yazılım geliştirme süreçlerinin iyileştirilmesinde önemli rol oynadığını göstermektedir.

Jalali ve Wohlin (2011) çalışmalarında, KYG ile çevik pratiklerin bir arada kullanımını incelemek amacıyla sistematik bir literatür taraması yapmışlardır. Araştırmanın amacı, çevik yöntemlerin KYG bağlamında nasıl uygulandığını ve bu uygulamaların karşılaştığı zorlukları analiz etmektir. Çalışma, nitel bir araştırma olup, 1999-2009 yılları arasında yayımlanan 81 hakemli çalışma incelenmiş ve çevik pratiklerin KYG'deki etkili kullanımına dair deneysel raporlar ve vaka çalışmaları değerlendirilmiştir. Araştırma sonuçlarına göre, KYG bağlamında en yaygın kullanılan çevik pratikler arasında ayaküstü toplantıları, yinelemeler/sprintler, sürekli entegrasyon, sprint planlama ve retrospektif toplantılar (geriye dönük değerlendirme) yer almıştır. Çalışma, KYG ve çevik yöntemlerin başarılı bir şekilde bir arada kullanılabildiğini göstermiş, ancak bu süreçte iletişim, zaman dilimi farkları ve kültürel uyumsuzluklar gibi zorlukların öne çıktığını vurgulamıştır. Ayrıca, XP ve Scrum'ın dağıtık ekiplerde başarıyla uygulandığı tespit edilmiştir.

Kurapati vd. (2012) çalışmalarında, yazılım endüstrisinde kullanılan çevik pratiklerin benimsenme durumunu ve bu pratiklerin ne ölçüde uyum sağladığını incelemeyi amaçlamışlardır. Nicel bir yöntemle çalışmada çevrimiçi anket aracılığıyla 109 katılımcıdan veri toplanmıştır. Araştırmada, çevik pratiklerin ne sıklıkla kullanıldığı, pratiklerin birlikte nasıl kullanıldığı ve bu pratiklerin Scrum ve XP metodolojilerine ne ölçüde uyum sağladığı analiz edilmiştir. Veriler, betimsel istatistikler ve hiyerarşik kümeleme analizi ile değerlendirilmiştir. Araştırma bulgularına göre, en sık kullanılan çevik pratikler arasında ayaküstü toplantıları, sprint/yineleme, sürekli entegrasyon, testler, sprint planlama toplantıları, ortak (kolektif) kod sahipliği, iletişim, kısa sürüm döngüleri, iş takibi, konfigürasyon ve değişiklik yönetimi ve retrospektifler yer almıştır. Bu pratikler, yazılım geliştirme süreçlerinde ekiplerin verimliliğini artırmak ve proje yönetim süreçlerini optimize etmek amacıyla yaygın olarak kullanılmaktadır. Daha az sıklıkla kullanılan pratikler arasında ise kodlama standartları, yeniden kod düzenleme, basit tasarım, çift programlama ve planlama oyunu bulunmuştur. Sonuçlar, çevik pratiklerin çoğunlukla birlikte kullanıldığını ve özellikle Scrum metodolojisinin XP'ye kıyasla daha yaygın olarak benimsendiğini göstermiştir. Katılımcılar genel olarak çevik pratiklerin müşteri ve çalışan memnuniyeti açısından olumlu bir etki yarattığını belirtmişlerdir.

Diel vd. (2015) çalışmasında, Brezilyalı bilişim teknolojileri profesyonellerinin çevik yöntemler hakkındaki anlayışlarını, hangi pratikleri kullandıklarını ve bu yöntemlerle çalışmak için gerekli becerileri incelemeyi amaçlamışlardır. Araştırma, iki aşamalı bir süreçten oluşmaktadır: İlk aşamada nitel bir araştırma yapılmış, ikinci aşamada ise anket yöntemiyle nicel veriler toplanmıştır. Toplamda 173 profesyonelden elde edilen veriler, katılımcıların çevik yöntemlerin temel ilkelerini iyi anladıklarını ve en çok kullanılan yöntemlerin Scrum ve XP olduğunu göstermektedir. Araştırma bulgularına göre, çevik yöntemlerle çalışmak için en gerekli beceriler arasında kendini yönetim ve iletişim becerileri öne çıkmıştır. Sık kullanılan çevik pratikler arasında sprint planlama toplantısı, ürün iş listesi yönetimi ve sürüm planlaması yer alırken, çevik yöntemlerin benimsenmesinde karşılaşılan en büyük zorluklar arasında kültürel değişim ve müşteri iş birliği bulunmuştur.

Jain ve Suman (2016) çalışmalarında, küresel yazılım geliştirme (KYG) projelerinde çevik pratiklerin etkinliğini araştırmayı amaçlamışlardır. Araştırma, iki aşamalı bir süreç izleyerek gerçekleştirilmiştir. İlk aşamada, ilgili literatür incelenerek çevik pratiklerin KYG projelerinde getirdiği faydalar ve zorluklar belirlenmiştir. İkinci aşamada ise bu bulguların doğruluğunu test etmek amacıyla anket yöntemi kullanılmıştır. Toplamda 39 dağıtılmış çevik projeden veri toplanmış ve elde edilen bulgular, çevik pratiklerin KYG projelerine uyumunu değerlendirmeye odaklanmıştır. Veri toplama aracı olarak anket kullanılmış ve veriler betimsel istatistiksel analiz yöntemiyle analiz edilmiştir. Araştırmanın bulgularına göre, en sık kullanılan çevik pratikler arasında sürekli entegrasyon, sprint, sprint planlama ve günlük scrum/ayaküstü toplantılar yer almaktadır. Bu pratikler, proje ilerlemesini izlemek, iletişimi geliştirerek kalitesini artırmak, koordinasyonu artırmak ve sorunların erken çözülmesine yardımcı olmak açısından oldukça etkili bulunmuştur. Bununla birlikte, zaman dilimi farkları, kültürel farklılıklar ve iletişimdeki gecikmeler, bu pratiklerin uygulanmasında karşılaşılan başlıca zorluklar arasında yer almıştır. Çalışma, KYG projelerinde çevik pratiklerin başarılı bir şekilde uygulanabilmesi için ek iletişim araçları ve destekleyici pratiklerin kullanılması gerektiğini vurgulamaktadır.

Licorish vd. (2016) çalışmalarında, Brezilya, Finlandiya ve Yeni Zelanda'da yazılım geliştirme yöntemlerinin ve pratiklerinin benimsenmesini ve bu yöntemlerin proje ve süreç yönetimi açısından uygunluğunu araştırmayı amaçlamışlardır. Araştırma,

nicel yöntemler kullanılarak gerçekleştirilmiş olup, veri toplama aracı olarak çevrimiçi anket kullanılmıştır. Çalışma kapsamında 184 yazılım profesyonelinden toplanan veriler, betimsel istatistikler ve Pearson Ki-Kare testleri kullanılarak analiz edilmiştir. Araştırma bulguları, katılımcıların en yaygın olarak Scrum ve Kanban yöntemlerini kullandığını göstermiştir. Bununla birlikte, katılımcılar arasında kodlama standartları, basit tasarım, yeniden kod düzenleme, sürekli entegrasyon ve yineleme planlaması gibi çevik pratiklerin de yaygın olarak benimsendiği belirlenmiştir. Ayrıca, bu pratiklerin %84,8'inin proje yönetimi için uygun olduğu, %76,6'sının ise süreç yönetimini yeterince desteklediği belirtilmiştir. Ancak, daha büyük ekiplerde ve geleneksel yöntemleri katı bir şekilde uygulayan firmalarda, çevik pratiklerin proje yönetimi ihtiyaçlarını yeterince karşılamadığı sonucuna varılmıştır.

Tripp vd. (2016) çalışmalarında, çevik pratiklerin iş karakteristikleri üzerindeki etkisini inceleyerek, iş tatmini ile olan ilişkisini araştırmayı amaçlamışlardır. Çalışmada, 252 yazılım geliştirme profesyonelinden anket yöntemiyle veri toplanmış ve veriler, PLS-SEM yöntemiyle analiz edilmiştir. Araştırmada çevik pratikler, çevik proje yönetimi pratikleri ve çevik yazılım geliştirme pratikleri olmak üzere iki ana başlık altında incelenmiştir. Çevik proje yönetimi pratikleri arasında günlük ayakta toplantılar, yinelemeli teslimatlar, retrospektifler ve iş ilerleme grafikleri yer alırken; çevik yazılım geliştirme pratikleri arasında birim testleri, otomatik oluşturma, sürekli entegrasyon, kodlama standartları, yeniden kod düzenleme ve eşli programlama bulunmaktadır. Araştırma sonuçlarına göre, çevik proje yönetimi ve yazılım geliştirme pratiklerinin kullanımı, çalışanların iş karakteristiklerine ilişkin algılarını olumlu yönde etkilemiş ve doğrudan iş tatmini ile pozitif bir ilişki göstermiştir. Çalışanlar, bu pratikler aracılığıyla daha fazla özerklik kazanmış, geri bildirim süreçlerinin iyileştirilmesi ile iş tatminleri artmıştır. Ayrıca, çevik proje yönetimi pratikleri ile iş özerkliği ve geri bildirim arasındaki etkileşim etkisi pozitif bulunmuştur.

Al-Sakkaf vd. (2017), Suudi Arabistan'daki yazılım girişim şirketlerinde uygulanan çevik pratiklerin bağımlılıklarını belirleyerek, bu pratikleri kümeler halinde organize etmeyi amaçlamışlardır. Nicel araştırma yönteminin kullanıldığı çalışmada veriler anket aracılığıyla 76 yazılım profesyonelinden toplanmıştır. Araştırmada 20 çevik pratiğin uygulanma düzeyleri 5'li Likert ölçeği ile değerlendirilmiş ve hiyerarşik kümeleme analizi yapılmıştır. Analiz sonucunda proje yönetimi, kalite güvence, ekip iletişimi ile

yinelemeli ve artımlı geliştirme olmak üzere dört küme tespit edilmiştir. Bulgular, ekip iletişimi (günlük toplantılar, kısa yinelemeler, yineleme planlaması ve retrospektif) kümesinin en yaygın olarak benimsenen çevik pratik kümesi olduğunu, proje yönetimi ve kalite güvence kümelerinin ise daha az tercih edildiğini ortaya koymuştur. Çalışma, yazılım girişim şirketlerinin çevik yöntemlerden maksimum faydayı sağlayabilmeleri için pratikleri kümeler halinde uygulamalarının daha etkili olduğunu önermektedir.

Henriksen ve Pedersen (2017), çevik yazılım projelerinde kullanılan pratiklerin proje başarısı üzerindeki etkilerini incelemek amacıyla çalışma gerçekleştirmişlerdir. Çevik proje başarısı kalite, kapsam ve zaman boyutlarıyla değerlendirilmiştir. Araştırma, nitel bir durum çalışması olarak tasarlanmış olup, veri toplama sürecinde yarı yapılandırılmış görüşmeler ve katılımcı gözlem yöntemleri kullanılmıştır. Görüşmeler, çevik ekiplerde farklı rollerde görev alan 9 katılımcıyla gerçekleştirilmiştir. Araştırmacılar, tematik analiz yöntemi ile verileri analiz etmiş ve çevik pratiklerin proje başarısına olan katkılarını belirli temalar altında incelemiştir. Bu kapsamda, toplamda 23 çevik pratik belirlenmiş ve bunlardan 15 önemli çevik pratiğin proje başarısı üzerinde kritik bir rol oynadığı tespit edilmiştir. Araştırma sonuçlarına göre; (1) Özellikle yinelemeli geliştirme, sprint değerlendirme, artımlı tasarım, sprint retrospektifi ve iş birliği ile iyileştirme proje başarısına önemli katkılar sağlamaktadır. (2) Kapsam açısından proje başarısına katkı sağlayan 8 pratik şunlardır: yinelemeli geliştirme, sprint planlama, kullanıcı hikayeleri, ürün biriktirme listesi, sprint değerlendirme, artımlı tasarım, sprint retrospektifi ve takım bazlı tahminleme. Bu pratikler proje kapsamını yönetmeye ve müşteri ihtiyaçlarına daha uygun çözümler üretmeye yardımcı olur. (3) Kalite açısından, proje başarısına katkı sağlayan 6 pratik şunlardır: artımlı tasarım, sprint retrospektifi, iş birliği ile iyileştirme, görev panosu, günlük ayakta toplantılar ve ürün sahibi. Bu pratikler, proje kalitesini artırarak daha iyi bir ürün teslim edilmesine katkı sağlamaktadır. (4) Zaman açısından ise tüm takım pratiği, proje teslim sürelerini kısaltarak zaman yönetimine katkıda bulunmuştur. Bulgular, çevik pratiklerin proje kapsamı, zaman ve kalite yönetiminde olumlu etkiler yarattığını ortaya koymuştur. Bunun yanında, günlük ayakta toplantılar ve görev panosu gibi pratiklerin ekip içi iletişimi ve şeffaflığı artırdığı tespit edilmiştir. Genel olarak, çalışma çevik pratiklerin esneklik, müşteri geri bildirimi ve müşteri ihtiyaçlarına uyum sağlama açısından proje başarı kriterlerini olumlu yönde etkilediğini ortaya koymaktadır.

Pantiuchina vd. (2017), yazılım girişim şirketlerinin çevik pratikleri nasıl uyguladığını ve yalın girişim yaklaşımının çevik pratiklerin kullanımına etkisini araştırmayı amaçlamaktadır. Nicel yöntemlerle gerçekleştirilen çalışmada anket aracılığıyla 1526 katılımcıdan veriler toplanmıştır. Çalışmada düzenli yeniden kod düzenleme, önce test, sık sürüm yayınlama, çevik planlama ve günlük ayakta toplantılar olmak üzere beş temel çevik pratik sorgulanmıştır. Toplanan veriler, keşifsel faktör analizi ve ki-kare testi ile analiz edilmiştir. Araştırma bulguları, yazılım girişim şirketlerinin hız odaklı çevik pratikleri (sık sürüm yayınlama ve çevik planlama) daha yaygın kullandığını, buna karşın kalite odaklı pratiklerin (düzenli yeniden kod düzenleme ve önce test) daha az tercih edildiğini göstermiştir. İletişim odaklı pratikler (günlük ayakta toplantılar) ise en az kullanılan pratikler arasında yer almıştır. Ayrıca, yalın girişim yaklaşımını benimseyen şirketlerin kaliteyi hızla feda etmediği ve özellikle önce test, sık sürüm yayınlama ve günlük ayakta toplantılar gibi pratikleri daha sık uyguladıkları sonucuna ulaşılmıştır.

Recker vd. (2017) çalışması, çevik pratiklerin müşteri duyarlılığı ve yazılım geliştirme başarısı üzerindeki etkilerini incelemeyi amaçlamaktadır. Araştırmada müşteri duyarlılığı yazılım ekibinin yanıt kapsamlılığı ve yazılım ekibi yanıt verimliliği olarak değerlendirilirken; proje başarısı müşteri memnuniyeti, süreç performansı ve yazılım işlevselliği olarak üç ana değişkenle ölçülmüştür. Çevik pratikler ise yönetim pratikleri (ayakta toplantılar), geliştirme pratikleri (çift programlama) ve çevik standartlar ve normlar (ortak kod sahipliği) olmak üzere üç ana kategoride incelenmiştir. Araştırma kapsamında, 71 katılımcıdan anket yöntemi ile veri toplanmış ve elde edilen veriler PLS-SEM yöntemi kullanılarak analiz edilmiştir. Araştırmanın bulguları, çevik yönetim pratiklerinin müşteri taleplerine yanıt verme verimliliğini artırdığını, ancak geniş kapsamlı yanıtlar sunma yeteneğini olumsuz etkileyebileceğini ortaya koymuştur. Geliştirme pratiklerinin ise geniş kapsamlı yanıtları desteklediği, ancak verimliliği doğrudan etkilemediği belirlenmiştir. Çevik standartlar ve normlar ise proje başarısına doğrudan katkı sağlamakla birlikte, müşteri yanıt sürecini etkilememektedir. Çalışma, çevik pratiklerin müşteri taleplerine yanıt verme ve proje başarısına olan etkilerini derinlemesine inceleyerek literatüre katkı sağlamaktadır.

Paez vd. (2018) çalışmalarında, Latin Amerika'daki yazılım firmalarının teknik ve organizasyonel çevik pratikleri nasıl uyguladıklarını anlamayı amaçlamışlardır. Nicel

yöntemler kullanılarak gerçekleştirilen bu çalışmada, üç aşamalı bir anket yöntemi ile veri toplanmıştır. Araştırmaya toplamda 107 geçerli katılım sağlanmış olup, katılımcılar Agile konferanslarında anketi doldurmuşlardır. Anket, yazılım geliştirme süreçlerinde kullanılan 10 temel çevik pratiğin kullanım sıklığını ölçmeyi hedeflemiştir. Bu pratikler teknik (sürekli entegrasyon, çift programlama, test otomasyonu, test odaklı geliştirme, ortaya çıkan tasarım) ve organizasyonel (yinelemeli süreç, sık sürüm yayınlama, retrospektifler, kendi kendini örgütleme, ortak kod sahipliği) olarak iki ana kategoriye ayrılmıştır. Veriler, betimleyici istatistikler ve faktör analizi ile analiz edilmiştir. Bulgulara göre, organizasyonel pratiklerin teknik pratiklere kıyasla daha yaygın olarak uygulandığı, özellikle yinelemeli süreç ve sık sürüm yayınlamanın en çok kullanılan pratikler olduğu belirlenmiştir. Bununla birlikte, organizasyonel deneyim arttıkça teknik pratiklerin kullanımının da arttığı gözlemlenmiştir.

Sun ve Schmidt (2018) çalışmalarında, çevik yöntemlerin kullanımı ile yazılım geliştiricilerin iş algıları arasındaki ilişkiyi araştırmayı amaçlamışlardır. Araştırma, nicel yöntemler ile gerçekleştirilmiş olup, anket yoluyla 200 yazılım profesyoneline veri toplanmıştır. Elde edilen veriler, k-means kümeleme analizi ile değerlendirilmiştir. Araştırma sonuçlarına göre, yüksek çevik yöntem kullanımı, katılımcılar arasında daha düşük rol belirsizliği, daha düşük iş yükü ve daha yüksek profesyonel yeterlilik ile ilişkilendirilmiştir. En sık kullanılan çevik pratikler arasında eşli programlama, sürekli entegrasyon, yeniden kod düzenleme, birim test ve kodlama standartları yer almaktadır. Ayrıca, çevik pratiklerin yüksek düzeyde kullanımının iş memnuniyetini artırdığı ve iş tükenmişliği gibi olumsuz etkileri azalttığı tespit edilmiştir.

Vallon vd. (2018), KYG süreçlerinde çevik pratiklerin kullanımını incelemeyi amaçlamışlardır. Araştırma, sistematik bir literatür taraması yöntemiyle gerçekleştirilmiş olup, 1999-2016 yılları arasında yayımlanan çalışmaları kapsamaktadır. Toplamda 56 çalışmadan elde edilen veriler analiz edilmiştir. Veriler, nicel ve nitel analiz yöntemleri ile değerlendirilmiştir. Araştırma bulgularına göre, KYG projelerinde en sık kullanılan çevik pratikler arasında ayaküstü toplantıları, ürün iş listesi, sprint/yineleme planlama, retrospektifler, sprint inceleme ve sürekli entegrasyon yer almaktadır. Özellikle 2010-2016 döneminde bu pratiklerin kullanımında artış gözlemlenmiştir. Çalışma, çevik pratiklerin KYG projelerinde başarıyla kullanılabilceğini ancak iletişim, zaman dilimi farkları ve kültürel uyumsuzluklar gibi zorlukların hala var olduğunu ortaya koymuştur.

Gupta vd. (2019) çalışmalarında, bilgi teknolojileri (BT) departman kültürünün çevik yazılım geliştirme pratiklerinin kullanımına etkisini araştırmışlardır. Çalışma, dört farklı kültürel yapı ile sosyal ve teknik çevik pratiklerin benimsenmesi arasındaki ilişkileri test eden bir model önermiş ve bu modeli ampirik olarak sınamıştır. Araştırma, nicel bir yöntemle gerçekleştirilmiş olup, BT departmanlarında görev alan 179 katılımcıdan anket aracılığıyla veri toplanmıştır. Veriler, YEM yöntemi ile analiz edilmiştir. Araştırmada kullanılan çevik pratikler, teknik ve sosyal olarak ikiye ayrılmıştır. Teknik pratikler arasında sürekli entegrasyon, ortak kod sahipliği, kodlama standartları, birim testi ve yeniden kod düzenleme yer alırken; sosyal pratikler arasında eşli programlama, günlük ayaküstü toplantılar, retrospektifler ve müşteri erişimi bulunmaktadır. Araştırma bulgularına göre, hiyerarşik kültür hem sosyal hem de teknik pratiklerin kullanımını olumsuz yönde etkilerken, rasyonel kültür yalnızca sosyal pratiklerin kullanımını olumlu yönde etkilemiştir. Grup kültürü, sosyal pratiklerin benimsenmesini olumlu yönde etkilerken, gelişimsel kültür hem sosyal hem de teknik pratiklerin kullanımını olumlu yönde etkilemiştir. Sonuçlar, BT departmanlarında çevik pratiklerin benimsenmesinde organizasyonel kültürün önemli bir rol oynadığını göstermektedir.

Leal vd. (2019) çalışmalarında, yazılım girişim şirketlerinin kullandıkları çevik pratikler ve yazılım geliştirme araçlarını araştırmayı amaçlamışlardır. Araştırma, Brezilya'da faaliyet gösteren 100 yazılım uygulayıcıdan toplanan verilerle yürütülmüştür. Araştırmanın bulguları, girişim şirketlerinin çevik yöntemleri benimsediğini göstermektedir. En yaygın kullanılan pratikler arasında geliştirme çerçeveleri, sürekli teslimat, iş listesi yönetimi, kısa sürüm döngüleri, yazılım yeniden kullanımı ve sürekli entegrasyon bulunmaktadır. Girişim şirketlerinin hızlı büyüme ve müşteri geri bildirimlerine hızlı yanıt verme ihtiyacından dolayı çevik yöntemleri tercih ettikleri görülmüştür. Ayrıca, versiyon kontrol sistemleri arasında GitHub ve Bitbucket en yaygın kullanılan araçlar olarak belirlenmiştir. Proje yönetiminde ise Trello en çok tercih edilen araç olmuştur. Araştırma, girişim şirketlerinin başarılarını artırmak için doğru araç ve pratiklerin seçiminin önemini vurgulamaktadır.

Neto vd. (2019) çalışmasında, çevik pratiklerin benimsenmesinin yazılım geliştirme şirketlerindeki örgütsel öğrenme sürecine etkisini araştırmayı amaçlamışlardır. Araştırma, nicel bir yöntemle gerçekleştirilmiş olup, veri toplama aracı olarak anket

kullanılmıştır. Çalışma kapsamında 455 yazılım geliştirme profesyonelinden veri toplanmıştır. Veriler, YEM yöntemiyle analiz edilmiştir. Araştırma bulgularına göre, çevik yazılım geliştiriciler daha sık proje yönetim pratiklerini kullanmaktadır. Bu pratikler, ekiplerin projeleri daha verimli yönetmesine ve bilgi paylaşımını artırmasına katkıda bulunmuştur. Proje yönetimi pratikleri ve yazılım geliştirme pratikleri arasında ayırım yapılmış, yazılım geliştirme pratiklerinin bilgi yönetimi stratejilerini ve örgütsel öğrenmeyi doğrudan etkilediği, ancak proje yönetimi pratiklerinin bu süreçler üzerinde anlamlı bir etkisinin olmadığı gözlemlenmiştir. En yaygın kullanılan çevik pratikler arasında birim testleri, görev panoları, tek takım çalışması, sürekli entegrasyon, kolektif kod sahipliği ve kodlama standartları yer almaktadır. Çalışma, çevik pratiklerin bilgi paylaşımını teşvik ettiğini ve bu paylaşımın örgütsel öğrenmeye katkıda bulunduğunu ortaya koymuştur.

Arcos-Medina ve Mauricio'nun (2020) çalışması, yazılım geliştirme süreçlerinde organizasyonel ve kişisel faktörlerin çevik pratiklerin kullanımına etkisini araştırmaktadır. Çalışma, insan davranışları ve organizasyon teorilerinden türetilen altı kritik başarı faktörü ve dört ana çevik pratik kategorisinden oluşan bir model önermektedir. Bu dört kategori, takım çalışması, proje yönetimi, mühendislik ve test olarak sınıflandırılmıştır. Takım çalışması pratikleri; çift programlama, yerinde müşteri, ayakta toplantılar, kendini organize eden takımlar, çok işlevli takımlar, retrospektifler ve ortak kod sahipliği olarak belirlenmiştir. Proje yönetimi pratikleri; planlama oyunları, ilerleme takibi, haftalık 40 saat çalışma kuralı ve zaman kutulama pratiklerinden oluşmaktadır. Mühendislik pratikleri; küçük sürümler, basit tasarım, yeniden kod düzenleme ve kodlama standartları olarak sınıflandırılmıştır. Test pratikleri ise birim testleri ve sürekli entegrasyon olarak ele alınmıştır. Çalışmada toplamda 17 çevik pratik incelenmiş ve 13 faktör-pratik ilişkisi belirlenmiştir. Nicel yöntemle gerçekleştirilen bu çalışmada, anket yöntemi kullanılarak 146 katılımcıdan veri toplanmış ve elde edilen veriler güvenilirlik testi ve analizlerle değerlendirilmiştir. Araştırma bulguları, güven, öz-yeterlilik algısı, bilgi ve öğrenilmiş deneyimlerin bütünlüğü ve kullanılabilirliği, kullanılan medya gibi faktörlerin çevik takım çalışması pratikleri üzerinde önemli olumlu etkilere sahip olduğunu göstermektedir. Proje yönetimi pratiklerinin uygulanmasında, kullanılan medya olumlu bir etki sağlarken, stres olumsuz etkilemektedir. Mühendislik pratiklerinin uygulanmasında, güven ve stresin olumsuz etkiler yarattığı bulunmuştur.

Test pratiklerinin uygulanmasında ise stres ve çalışma ortamının kalitesi olumsuz etkiler yaratmaktadır. Çalışma, çevik pratiklerin yazılım geliştirme süreçlerindeki etkilerini detaylı bir şekilde ele alarak literatüre önemli katkılar sunmaktadır.

Paez vd. (2020) çalışmalarında, çevik yöntemlerin yazılım geliştirme pratiklerinin kullanımına etkisini ve teknik ile organizasyonel pratiklerin kullanımı arasında bir denge olup olmadığını araştırmayı amaçlamışlardır. Nicel yöntemler kullanılarak gerçekleştirilen çalışmada, HELENA anket verileri kullanılarak toplamda 36 yazılım geliştirme pratiği değerlendirilmiştir. Araştırmada teknik ve organizasyonel pratiklerin kullanım oranları karşılaştırılmış ve en çok kullanılan pratikler belirlenmiştir. Bulgulara göre, en yaygın kullanılan teknik pratikler arasında kodlama standartları, yeniden kod düzenleme, sürekli entegrasyon, birim test, kullanıcı hikayeleri ve uçtan uca test bulunmaktadır. Organizasyonel pratikler arasında ise sürüm planlaması, iş listesi yönetimi, yineleme/sprint değerlendirme, ayaküstü toplantılar ve yineleme planlaması öne çıkmıştır. Sonuç olarak, teknik ve organizasyonel pratikler arasında kullanım açısından önemli bir fark olmadığı belirlenmiştir, bu da çevik yöntemlerin her iki tür pratiğin kullanımını dengeli bir şekilde teşvik ettiğini göstermektedir.

Anjani vd. (2021), organizasyon kültürünün (hiyerarşik, rasyonel, grup ve gelişim kültürleri) çevik yazılım geliştirme projelerindeki başarıya olan etkisinde çevik pratiklerin aracılık rolünü incelemişlerdir. Araştırmada çevik proje başarısı kalite, kapsam, zaman ve maliyet boyutlarıyla değerlendirilmiştir. Çevik pratikler; çevik sosyal pratikler (müşteri erişimi, yineleme planlaması, ayakta toplantılar ve retrospektifler) ve çevik teknik pratikler (yinelemeli geliştirme ve sürekli entegrasyon) olarak proje başarısına olan katkıları bağlamında ele alınmıştır. Nicel yöntemlerle gerçekleştirilen çalışmada veriler anket aracılığıyla 101 katılımcıdan elde edilmiştir. Veriler PLS-SEM yöntemiyle analiz edilmiştir. Araştırma sonuçları rasyonel ve grup kültürlerinin çevik yazılım geliştirme proje başarısı üzerinde olumlu etkiler yarattığını; hiyerarşik kültürün ise olumsuz etkisi olduğunu ortaya koymuştur. Gelişim kültürü ise çevik pratikler aracılığıyla başarıya olumlu katkı sağlamaktadır. Çalışmada ayrıca çevik sosyal ve teknik pratiklerin organizasyon kültürü (rasyonel ve grup kültürü) ile proje başarısı arasındaki ilişkide kısmi aracı rol oynadığı tespit edilmiştir. Bu bulgular, çevik pratiklerin çevik yazılım geliştirme projelerinde başarıya ulaşmada organizasyon kültürü bağlamında farklı etkiler yarattığını göstermektedir.

Klotins vd. (2021) çalışması, girişim şirketlerinde çevik pratiklerin kullanımını ve bu pratiklerin proje başarısına olan etkilerini incelemeyi amaçlamaktadır. Araştırmanın temel hedefi, çevik pratiklerin bu dinamik ve hızlı büyüyen şirketlerde nasıl uygulandığını ve proje yönetiminde başarıya nasıl katkıda bulunduğunu ortaya koymaktır. Çalışma kapsamında, çevrimiçi anket yöntemi ile 181 yazılım geliştirme profesyonelinden veri toplanmış ve toplamda 56 farklı çevik pratik incelenmiştir. Verilerin analizinde PLS-SEM yöntemi kullanılmıştır. Proje başarısı kalite, kapsam, zaman ve maliyet değişkenleriyle ölçülmüştür. Araştırmanın bulguları, girişim şirketlerinde kullanılan çevik pratiklerin proje başarısına önemli katkılar sağladığını göstermektedir. Çalışmada en sık kullanılan çevik pratikler arasında ürün biriktirme listesi, versiyon kontrolü, yeniden kod düzenleme, birim test, kanban panosu ve kullanıcı hikayelerinin geliştirilmesi öne çıkmıştır. Ayrıca, çevik pratiklerin kaynak kod ve ürün kalitesi üzerinde doğrudan etkileri olduğu gözlemlenmiştir. Sürekli entegrasyon pratiği genel ürün kalitesini artırırken, kullanıcı hikayeleri proje zamanı ve kapsamı üzerinde etkili olmuştur. İlerleme grafiği, maliyet ve zaman yönetimine katkı sağlamış, otomatik derleme ise kaynak kod kalitesine katkıda bulunmuştur. Araştırma sonuçlarına göre, küçük ve dinamik girişim şirketlerinin, büyük ölçekli projelere kıyasla çevik pratikleri daha hızlı bir şekilde uyguladığı tespit edilmiştir.

Malik vd. (2021) çalışması, çevik pratiklerin proje performansı üzerindeki etkilerini incelemektedir. Araştırmanın temel amacı, çevik pratiklerin psikolojik güçlendirme aracılığıyla yenilikçi davranışları nasıl teşvik ettiğini ve bu süreçte proje performansına nasıl katkıda bulunduğunu açıklamaktır. Nicel yöntemle gerçekleştirilen bu çalışma, çevrimiçi anket yoluyla 202 yazılım geliştirme profesyonelinden veri toplamıştır. Veriler, PLS-SEM yöntemi ile analiz edilmiştir. Proje performansı, kalite, kapsam, zaman ve maliyeti içeren temel başarı kriterleriyle değerlendirilmiştir. Araştırma bulguları, takım özerkliği ve çevik iletişimin psikolojik güçlendirme yoluyla yenilikçi davranışları teşvik ettiğini ve bu davranışların proje performansını iyileştirdiğini ortaya koymaktadır. Buna karşın, takım çeşitliliği ve yinelemeli geliştirme gibi çevik pratiklerin yenilikçi davranışlar üzerinde doğrudan bir etkisi olmadığı tespit edilmiştir. Çalışma, çevik pratiklerin yenilikçi projelerde proje performansına olan etkisini psikolojik güçlendirme mekanizması ile açıklayarak literatüre önemli katkılar sunmaktadır.

Mulugeta (2021), çevik yazılım geliştirme projelerinde kullanılan pratiklerin proje başarısına olan etkilerini incelemek amacıyla nicel bir çalışma gerçekleştirmiştir. Çevik proje başarısı kapsam, kalite ve kısıt (maliyet ve zaman) boyutlarıyla değerlendirilmiştir. Araştırma kapsamında 45 geçerli anket toplanmış ve veriler çoklu regresyon analizi ile analiz edilmiştir. Araştırmanın sonuçları, çevik pratiklerin proje başarısına önemli ölçüde katkı sağladığını göstermiştir. Özellikle yinelemeli geliştirme, sprint incelemesi, iş birliği ile iyileştirme, kullanıcı hikayeleri, ürün iş listesi, ayakta toplantılar, görev panosu ve takım bazlı tahminleme çevik pratiklerinin proje başarısı üzerinde olumlu etkiler yarattığı saptanmıştır. Araştırma, çevik pratiklerin başarıya katkısının projeye özgü bağlama göre değişebileceğini de vurgulamaktadır.

Sandstø ve Reme-Ness (2021), çevik pratiklerin proje başarısına olan etkilerini incelemek amacıyla yürüttükleri nitel araştırmalarında, literatürde yaygın olarak bildirilen çevik uygulamaları ve bu uygulamaların proje başarısı üzerindeki etkilerini araştırmışlardır. Araştırma, iki aşamalı bir narratif literatür taraması yöntemine dayanmaktadır. İlk aşamada, yaygın olarak kullanılan çevik pratikler belirlenmiş; ikinci aşamada ise bu pratiklerin proje başarı koşulları üzerindeki olumlu ve olumsuz etkileri incelenmiştir. Araştırma sonucunda 12 yaygın çevik pratik tespit edilmiştir: birlikte çalışan ekipler, müşteri katılımı, kendini organize eden ekipler, sürekli ilerleme görselleştirmesi, artımlı sürümler, minimal dokümantasyon, retrospektif, değer öncelikli gereksinimler, sürekli kod entegrasyonu, eşli programlama, sprintler ve ayakta toplantılar. Çalışmada proje başarısına katkı sağlayan faktörler çeviklik, iletişim, motivasyon, şeffaflık, güven ve bilgi paylaşımı olarak belirlenmiştir. Çeviklik, kapsam genişliği ve verimlilik boyutlarıyla ele alınmış; kendini organize eden ekipler ve çift programlama uygulamaları bu faktörü etkilemiştir. İletişim, iç, dış, doğrudan ve dolaylı olmak üzere farklı kategorilerde incelenmiş; bu kategorilerde birlikte çalışan ekipler, sürekli ilerleme görselleştirmesi, artımlı sürümler, retrospektifler, değer öncelikli gereksinimler, sürekli kod entegrasyonu, çift programlama, sprintler ve ayakta toplantı pratiklerinin olumlu etkileri belirlenmiştir. Motivasyon üzerinde kendini organize eden ekipler, sprintler ve retrospektifler gibi uygulamaların hem olumlu hem olumsuz etkileri saptanmıştır. Şeffaflık ve güven faktörlerinde retrospektifler, sprintler ve ayakta toplantılar olumlu etkiler yaratırken, bazı olumsuz etkiler de raporlanmıştır. Son olarak, bilgi paylaşımı faktöründe birlikte çalışan ekipler, müşteri katılımı, sürekli ilerleme görselleştirmesi,

artımlı sürümler, retrospektifler, sprintler ve ayakta toplantılar pratiklerinin olumlu etkileri öne çıkmıştır. Çalışmanın bulguları, çevik pratiklerin proje başarısı üzerinde hem olumlu hem de olumsuz etkiler yaratabileceğini göstermiştir. Örneğin, sprintlerin bazı durumlarda iç iletişimi güçlendirdiği, ancak aynı zamanda uzun vadeli proje görünürlüğünü azaltarak iç iletişimi olumsuz etkileyebileceği tespit edilmiştir. Bu bağlamda, çevik pratiklerin proje bağlamına ve organizasyonel koşullara göre farklı sonuçlar doğurabileceği vurgulanmıştır. Sonuç olarak, araştırmacılar, çevik pratiklerin dikkatle değerlendirilmesi gerektiğini ve her bir projenin ihtiyaçlarına uygun çevik pratiklerin seçilmesinin önemli olduğunu belirtmişlerdir.

Dada ve Sanusi (2022) çalışmalarında, Scrum ortamında yazılım mühendisliği pratiklerinin benimsenme düzeyini ve kullanımını incelemeyi amaçlamışlardır. Araştırma, karma yöntem yaklaşımı ile gerçekleştirilmiş olup, nicel veriler anketler aracılığıyla 35 yazılım geliştirme profesyoneliinden, nitel veriler ise yarı yapılandırılmış mülakatlar ile 9 katılımcıdan toplanmıştır. Veriler, tanımlayıcı ve çıkarımsal istatistik yöntemleri kullanılarak analiz edilmiştir. Araştırmanın bulgularına göre, Scrum ortamında en yaygın kullanılan yazılım mühendisliği pratikleri arasında kullanıcı hikayeleri, sürekli entegrasyon, birim testleri, entegrasyon testleri ve ürün iş listesi bulunmaktadır. Ayrıca, Scrum'ın mühendislik pratiklerine yönelik doğrudan önerilerde bulunmamasına rağmen, "incele ve uyum sağla" ilkesinin uygulanmasının Scrum'ın başarılı bir şekilde benimsenmesini desteklediği sonucuna ulaşılmıştır. Çalışma, Scrum ekiplerinin planlama süreçlerini, kod incelemelerini ve geliştirme adımlarını sürekli gözden geçirip uyarlayarak yazılım mühendisliği pratiklerini daha verimli hale getirebileceklerini öne sürmektedir.

Ghimire ve Charters (2022) çalışmalarında, çevik yazılım geliştirme pratiklerinin proje çıktıları üzerindeki etkisini incelemeyi amaçlamışlardır. Araştırma, nicel yöntem ile gerçekleştirilmiş olup, Yeni Zelanda'daki yazılım geliştirme ekiplerinden 73 katılımcıyla bir çevrimiçi anket aracılığıyla veri toplanmıştır. Anket, çevik pratiklerin proje süresi, bütçesi ve ekip içi iletişim üzerindeki etkilerini değerlendirmeyi hedeflemiştir. Veriler, Pearson korelasyon katsayısı ile analiz edilmiştir. Bulgular, en sık kullanılan çevik pratiklerin ayaküstü toplantıları, ürün iş listesi, birim testi, kullanıcı hikayeleri, Scrumban ve retrospektifler olduğunu göstermektedir. Ayrıca çevik pratik kullanmanın yüksek düzeyde kullanılması ekip içi iletişim sorunlarını azalttığı ve müşteri gereksinimlerinin

daha net tanımlanmasını sağladığı tespit edilmiştir. Ancak, çevik pratiklerin proje süresi ve bütçesi ile anlamlı bir ilişkisi bulunmamıştır. Sonuç olarak, çevik pratiklerin etkin bir şekilde kullanılmasıyla proje çıktılarının iyileştirilebileceği vurgulanmaktadır.

Neumann (2022) tarafından gerçekleştirilen çalışmada, çevik yazılım geliştirme süreçlerinde kullanılan pratiklerin entegre bir listesinin oluşturulması amaçlanmaktadır. Nitel bir araştırma olarak tasarlanan bu çalışmada, çevik pratikler hakkında kapsamlı bir inceleme yapmak için sistematik haritalama ve literatür tarama yöntemleri kullanılmıştır. Çalışmaya, 2010-2021 yılları arasında yayımlanan toplamda 37 çalışma dahil edilmiştir. Bu çalışmaların listelerinde yer alan pratikler karşılaştırılarak, içerik analizi sonucunda 37 pratikten oluşan entegre bir liste oluşturulmuştur. Çevik pratikler, teknik, iş birliği, süreç, gereksinimler ve organizasyonel pratikler olmak üzere beş ana kategori altında sınıflandırılmış ve çevik yazılım geliştirme alanında hem araştırmacılar hem de uygulayıcılar için kapsamlı bir referans kaynağı sunulmuştur.

Estrada-Esponda vd. (2024) çalışmasında, yazılım geliştirme süreçlerinde çevik pratiklerin ve iyileştirme hedeflerinin önceliklendirilmesini araştırmışlardır. Çalışmanın amacı, organizasyonların ihtiyaçlarına en uygun çevik pratikleri ve iyileştirme hedeflerini belirlemek için AHP kullanarak bir değerlendirme yapmaktır. 40 profesyonelden toplanan veriler, dört yazılım geliştirme organizasyonundaki uzmanlar tarafından doldurulan anketler aracılığıyla elde edilmiştir. Verilerin analizi sonucunda, en kritik altı iyileştirme hedefi olarak zamanında karar verme, müşteri katılımının artırılması, iletişimin iyileştirilmesi, insan kaynakları yönetiminin geliştirilmesi, değişikliklerin etkin yönetimi ve teslimat gecikmelerinin önlenmesi öne çıkmıştır. Ayrıca, en önemli beş çevik pratik ise sık sık planlama toplantıları düzenlemek, takımın işini iyi tamamlanmış bir iş çıkarmaya odaklayarak organize etmek, işlerin görselleştirilmesi, sık sık teslimat yapılması ve bekleyen işlerin sürekli yönetimi olarak belirlenmiştir. Bu bulgular, çevik yöntemlerin uygulanmasında AHP gibi yapılandırılmış karar alma araçlarının kullanılmasının, organizasyonların verimliliğini artırmada etkili olduğunu göstermektedir.

4.3. ÇEVİK YAZILIM GELİŞTİRME PROJE BAŞARISIYLA İLGİLİ DİĞER ÇALIŞMALAR

Livermore (2008) çevik yazılım geliştirme metodolojilerinin başarılı bir şekilde uygulanmasını etkileyen faktörleri incelemek amacıyla çalışma gerçekleştirmiştir. Eğitim, yönetim desteği ve katılımı, metodoloji uygulama stratejisi, model ve şablon kullanımı, dış kaynaklara erişim, internet/intranet yazılım geliştirme, şirket/ekip büyüklüğü ve geliştirme ekibinin aynı yerde olması faktörlerinin metodolojilerin uygulanmasındaki başarı üzerindeki etkileri incelenmiştir. Nicel yöntemlerle gerçekleştirilen bu çalışmada, veriler çevrimiçi anket aracılığıyla 112 katılımcıdan toplanmış ve korelasyon analizi yapılmıştır. Araştırma sonuçlarına göre, çevik metodolojilerin uygulanmasındaki başarı üzerinde eğitim, yönetim desteği ve katılımı ile dış kaynaklara erişim faktörleri etkili bulunmuştur. Diğer faktörlerin ise çevik yazılım geliştirme metodolojilerinin uygulanması üzerinde anlamlı bir etkisi olmadığı sonucuna varılmıştır.

Sheffield ve Lemétayer (2013), başarılı projelerde yazılım geliştirme çevikliğini etkileyen faktörleri incelemek amacıyla nitel ve nicel yöntemlerin kullanıldığı karma bir araştırma yürütmüştür. Araştırmanın nitel boyutunda, kart sıralama tekniği kullanılarak gerçekleştirilen yarı yapılandırılmış görüşmeler yapılmıştır. Bu görüşmeler, yazılım geliştirme çevikliğini etkileyen temel faktörleri belirlemek için ön bilgiler sağlamıştır. Kart sıralama süreci, katılımcıların farklı faktörleri önem sırasına göre düzenlemelerine olanak tanımış ve elde edilen bulgular, ankete dayalı nicel çalışmada kullanılan anket sorularının geliştirilmesine katkıda bulunmuştur. Araştırmanın nicel boyutunda ise uluslararası bir anket uygulanmıştır. Ankete katılan 106 katılımcıdan elde edilen veriler yapısal eşitlik modellemesi ile analiz edilmiştir. Bulgular, yazılım geliştirme çevikliğinin artırılmasında en önemli iki faktörün organizasyon kültürü ve proje ekibinin güçlendirilmesi olduğunu göstermiştir. Bu iki faktörün projelerin başarı oranını artırdığı ve yazılım geliştirme süreçlerinde çevikliğin sağlanmasında kritik rol oynadığı vurgulanmıştır.

Corvello ve Verteramo'nun (2014) çalışması, çevik yazılım geliştirme metodolojilerinin benimsenmesinde müşteri katılımının rolünü incelemeyi amaçlamaktadır. Araştırma, müşteri ile tedarikçi arasındaki iş birliği ve müşterinin

yazılım geliştirme süreçlerine aktif katılımının, organizasyonların çeviklik seviyesini nasıl etkilediğini araştırmaktadır. Nicel yöntemle gerçekleştirilen bu çalışma kapsamında, LinkedIn sosyal ağı üzerinden çevik metodolojileri kullanan firmalardan veri toplanmıştır. Çalışmada kullanılan anketler aracılığıyla, firmaların çevik uygulamaları kullanma sıklığı ve müşteri katılımı değerlendirilmeye alınmıştır. Toplamda 102 geçerli yanıtın incelendiği araştırmada, hiyerarşik çoklu regresyon analizi uygulanarak müşteri iş birliği tutumu ve müşterinin aktif katılımının organizasyon çevikliği üzerindeki etkileri analiz edilmiştir. Bulgular, müşteri ile kurulan uzun vadeli, iş birliğine dayalı ilişkilerin çevik yöntemlerin daha sistematik kullanılmasını sağladığını göstermiştir. Ayrıca müşterinin yazılım geliştirme sürecine aktif katılımı, çevik pratiklerin kullanım sıklığını artıran önemli bir etken olarak tespit edilmiştir.

Serrador ve Pinto (2015) çalışmasında, çevik yazılım geliştirme yöntemlerinin proje başarısına etkisinde proje hedeflerinin netliği, proje karmaşıklığı ve ekip deneyiminin düzenleyici rolü araştırılmıştır. Araştırmada proje başarısı, proje verimliliği ve paydaş memnuniyeti değişkenleri ile tanımlanmıştır. Nicel yöntemlerle gerçekleştirilen çalışmada veriler, 1002 farklı projeden çevrimiçi anket aracılığıyla toplanmış ve regresyon analizi ile değerlendirilmiştir. Araştırma sonuçları çevik yöntemlerin daha yüksek proje başarısı ile ilişkili olduğunu göstermiştir. Özellikle, proje hedeflerinin netliğinin, çevik yöntemlerin başarı üzerindeki etkisini artıran en önemli düzenleyici değişken olduğu belirlenmiştir. Buna karşılık proje karmaşıklığı ve ekip deneyiminin başarı üzerindeki etkisinde anlamlı bir düzenleyici rol oynamadığı tespit edilmiştir.

Farlik (2016) doktora çalışmasında, çevik yazılım geliştirme ortamlarında iletişim ve yazılım süreç iyileştirmenin proje başarısıyla ilişkisini incelemiştir. Nicel yöntemlerle gerçekleştirilen bu çalışmada, veriler çevrimiçi anket aracılığıyla 210 katılımcıdan toplanmıştır. Veriler, korelasyon analizi ile değerlendirilmiş ve proje başarısı; zaman, maliyet ve müşteri memnuniyeti kriterleriyle ölçülmüştür. Araştırma sonuçlarına göre, etkin iletişim, proje başarısının müşteri memnuniyeti boyutuna önemli katkı sağlamıştır. Yazılım süreç iyileştirme ise, proje başarısının zaman ve maliyet boyutlarında olumlu etkiler yaratmıştır. Bu bulgular, çevik projelerde iletişim ve süreç yönetiminin proje başarı kriterlerine doğrudan katkı sağladığını göstermektedir.

Kim vd. (2019) çalışması, çevik yazılım geliştirme projelerinde müşteri tarafının rolünü derinlemesine incelemeyi amaçlamaktadır. Araştırma, özellikle kullanıcı katılımı, gerçekçi olmayan müşteri beklentileri ve gereksinimlerin sürekli değişimi faktörlerin proje başarısı üzerindeki etkilerine odaklanmıştır. Nicel yöntemlerle gerçekleştirilen çalışmada veriler çevrimiçi anket yöntemiyle 213 katılımcıdan toplanmıştır. Verilerin analizi, sıralı lojistik regresyon yöntemiyle gerçekleştirilmiştir. Bulgular, kullanıcı katılımının çevik projelerin başarısı için önemli bir unsur olduğunu ortaya koymuştur. Ayrıca, gereksinimlerin sürekli değişmesinin çevik metodolojilerin doğasına uygun bir biçimde projelere olumlu katkı sunduğu tespit edilmiştir. Bununla birlikte, gerçekçi olmayan müşteri beklentilerinin proje başarısını olumsuz etkilediği sonucuna ulaşılmıştır. Çalışma, müşteri katılımının aktif bir şekilde yönetilmesinin ve beklentilerin uygun şekilde yönlendirilmesinin, çevik projelerin başarı olasılığını önemli ölçüde artırdığına dikkat çekmektedir.

Perera ve Perera (2019), Sri Lanka yazılım sektöründe çevik projelerin başarısına müşteri katılımının etkilerini incelemek amacıyla çalışma gerçekleştirmiştir. Araştırmada, nicel yöntem kullanılmış olup, çevrimiçi anketler aracılığıyla müşteri katılımı ile proje başarısı arasındaki ilişkinin boyutları incelenmiştir. Çalışmada, 380 katılımcıdan elde edilen verilere korelasyon ve regresyon analizleri yapılmıştır. Yapılan analizler sonucunda, müşteri katılımının çevik projelerdeki başarıya olumlu katkılar sağladığı tespit edilmiştir. Özellikle müşteri iletişim düzeyi, müşteri iş birliği, müşteri esnekliği ve müşteri sektör bilgisi gibi faktörlerin, çevik projelerin başarılı bir şekilde yürütülmesinde kritik bir rol oynadığı ortaya konulmuştur. Ayrıca örgüt kültürünün bu ilişkide düzenleyici bir etkisi olduğu belirlenmiştir. Diğer bir ifadeyle, güçlü bir örgüt kültürünün müşteri katılımının proje başarısına olan etkisini daha da artırdığı tespit edilmiştir. Bu bulgular müşteri ile sürekli iletişim ve iş birliği içinde olmanın çevik projelerin başarısında ne denli önemli olduğunu vurgulamaktadır.

Thompson (2019) doktora çalışmasında, çevik yazılım geliştirme projelerinde müşteri iş birliğinin etkilerini ve bu iş birliğinin teknoloji tabanlı girişim şirketlerde ürün başarı oranlarına nasıl katkı sağladığını incelemeyi amaçlamaktadır. Araştırma, özellikle Pacific Northwest bölgesindeki teknoloji girişimlerine odaklanmıştır. Müşteri katılımının çevik proje süreçleri üzerindeki etkileri, karar alma, esneklik ve yenilik gibi boyutlar açısından derinlemesine ele alınmıştır. Araştırma nitel bir yaklaşımla yürütülmüş olup,

vaka çalışması yöntemi kullanılmıştır. Katılımcılarla çevrimiçi görüşmeler ve anketler yoluyla veri toplanmış, bu görüşmelerde müşterinin proje süreçlerine dahil olma düzeyleri ve sağladıkları geri bildirimlerin yazılım geliştirme süreçlerine etkileri sorgulanmıştır. Araştırmaya dahil edilen katılımcılar, çevik yazılım geliştirme projelerinde yer alan girişim yöneticileri, proje liderleri ve yazılım geliştiricileridir. Verilerin analizinde tematik analiz yöntemi kullanılmış, müşteri katılımının projelerin başarı oranlarına etkisi çeşitli boyutlarda değerlendirilmiştir. Analiz sonuçları, müşteri iş birliğinin çevik projelerde başarıyı artıran önemli bir faktör olduğunu ortaya koymuştur. Özellikle müşteri geri bildirimlerinin projelere düzenli olarak entegre edilmesi, yazılım ürünlerinin müşteri ihtiyaçlarına daha uygun olmasını sağlamış ve ürün başarısını artırmıştır. Ayrıca, müşteri taleplerine zamanında yanıt verebilme yeteneği, projelere esneklik ve hız kazandırarak rekabet avantajı sunmuştur. Sonuç olarak, çalışma müşteri iş birliğinin çevik yazılım geliştirme projelerinin başarısındaki kritik rolünü vurgulamakta ve müşteri katılımının, proje çıktılarının kalitesine ve piyasaya sürülme hızına olan olumlu etkilerini ortaya koymaktadır.

Hamid vd. (2022) çalışmalarında, gelişmekte olan ülkelerdeki yazılım projelerinin başarısını etkileyen kritik faktörleri incelemişlerdir. Çalışmanın amacı, proje planlaması, insan kaynakları, zaman tahmini ve maliyet tahmini eksikliklerinin yazılım projeleri üzerindeki etkisini değerlendirmektir. Nicel yöntemlerle gerçekleştirilen çalışmada veriler anket yoluyla 339 kıdemli geliştirici ve proje yöneticisinden toplanmıştır. Verilerin analizinde YEM kullanılmıştır. Araştırma sonuçları yetersiz planlama, insan kaynaklarının eksikliği ve yanlış zaman ve maliyet tahminlerinin proje başarısı üzerinde olumsuz etkiler yarattığını ortaya koymuştur. Bu faktörler, projelerin gecikmesine, bütçe aşımına ve kaynakların verimsiz kullanılmasına neden olmuştur. Çalışma, başarılı yazılım projeleri için planlama, insan kaynaklarının etkin kullanımı ve doğru tahminlerin kritik önem taşıdığını vurgulamaktadır.

Şimşek ve Ateş (2022), bilişim sistemi geliştirme sürecinde insan faktörünün rolünü ve özelliklerini ortaya koymak amacıyla çalışma gerçekleştirmiştir. Sistematik literatür taramasıyla 21 çalışma incelenmeye uygun bulunmuştur. Bu incelemeler sonucunda toplanan veriler nitel olarak analiz edilmiştir. Araştırma bulgularına göre, bilişim sistemi geliştirme sürecindeki insan faktörü, yönetici, ekip üyesi ve müşteri rolleri şeklinde sınıflandırılmıştır. İnsan faktörlerinin süreçteki önemi sıralandığında, en çok vurgulanan

unsurlar arasında yönetim desteđi, proje ekibi, ekip çalışması ve etkili iletişim yer almaktadır. Ayrıca, müşteri katılımı ve lider yöneticilik gibi faktörlerin de süreç üzerinde önemli bir etkiye sahip olduğu belirlenmiştir.

Palopak ve Huang (2024), çevik ilkelerin çevik yazılım geliştirme projelerinin başarısına etkisini araştırmışlardır. Nicel bir araştırma yöntemiyle gerçekleştirilen çalışma kapsamında, 298 çevik proje uygulayıcısından çevrimiçi anket yoluyla veri toplanmıştır. Çalışmada PLS-SEM yöntemi kullanılarak veri analiz edilmiştir. Araştırma bulguları düzenli teslimat, teknik mükemmeliyet, ekip üyelerinin proaktifliği ve müşteri iş birliğinin çevik projelerde başarıyı en çok etkileyen faktörler olduğunu, buna karşın süreç basitliğinin anlamlı bir etki yaratmadığını ortaya koymuştur.



BEŞİNCİ BÖLÜM

METODOLOJİ

5.1. ARAŞTIRMANIN AMACI VE ÖNEMİ

Bilgi ve iletişim teknolojilerinin hayatımıza girmesiyle birlikte, yazılım ürünleri devletlerin, toplumların, şirketlerin ve kurumların çeşitli ihtiyaçlarına çözümler sunmaya başlamıştır (Nizam, 2015; Sommerville, 2015). Modern dünyanın pek çok alanında önemli bir rol oynayan yazılım, geliştirme süreçleri açısından kusursuz olamamıştır (Chow ve Cao, 2008). Yazılım geliştirme sürecinde çeşitli metodolojiler kullanılmasına rağmen düşük kalite, zayıf proje yönetimi, yetersiz süreçler ve eksik gereksinim analizi gibi nedenlerle geciken, başarısız olan, terk edilen veya reddedilen yazılım projeleri ortaya çıkmıştır (Cao, 2006).

Bu sorunlar, yazılım geliştirme süreçlerini daha verimli, etkili ve esnek hale getirmek için yeni yaklaşımların geliştirilmesini teşvik etmiştir. Bu doğrultuda, 2001 yılında yayınlanan Çevik Manifesto, yazılım geliştirme süreçlerine yenilikçi bir bakış açısı getirmiştir. Çevik yöntemler, müşteri memnuniyetini artırmayı, değişen gereksinimlere hızla uyum sağlamayı ve ürün kalitesini yükseltmeyi amaçlayan bir dizi ilke ve uygulamayı içermektedir (Agile Alliance, 2001a; 2001b; 2001c). Geleneksel yöntemlere kıyasla çevik yazılım geliştirme yöntemleri, yazılım geliştirme süreçlerinde daha iyi ve yenilikçi bir yol olarak ortaya çıkmıştır (Chow ve Cao, 2008).

Çevik yöntemler, yazılımın nispeten kısa bir süre içinde geliştirildiği ve birkaç yineleme sonucu kararlı bir yazılım sürümü elde edilen kavramsal bir çerçevedir (Lalsing vd., 2012). Çevik yöntemler, değişen gereksinimlere ve teknolojilere uyum sağlama yeteneği, yalnızca gerekli düzeyde dokümantasyon kullanımı ve yazılım teslimatını hızlı bir şekilde gerçekleştiren artımlı ve yinelemeli geliştirme süreçlerini benimsemesi ile öne çıkmaktadır (Kirpitsas ve Pachidis, 2022).

Günümüzde çevik yöntemler, yazılım geliştirme süreçlerinde en popüler ve en başarılı yaklaşımlar arasında kabul görmektedir (Lalsing vd., 2012). Ancak, Standish Group (2020) tarafından yazılım projelerinin başarısını ve başarısızlığını analiz etmek amacıyla oluşturulan CHAOS (Comprehensive Human Appraisal for Originating

Software - Yazılımın Kökenini Belirlemeye Yönelik Kapsamlı İnsan Değerlendirmesi) raporunda, çevik yazılım projelerinin %42'sinin başarıyla tamamlandığı, geriye kalan projelerin ise ya sorunlu (%47) ya da başarısız (%11) olduğu belirtilmiştir (Mersino, 2020; Portman, 2021).

Çevik yöntemler, diğer yöntemlere kıyasla daha başarılı sonuçlar elde etse de CHAOS raporunda belirtildiği üzere, projelerin yarısından fazlasının zorluklarla karşı karşıya olduğu görülmektedir (Mersino, 2020). Bu durum, yazılım geliştirme projelerinin başarı oranını artırmak ve yazılım şirketlerinin dikkate alması gereken temel değişkenleri belirlemenin önemini vurgulamaktadır. Çevik yazılım geliştirme projelerinin başarıya etkide bulunan kritik faktörlerinin incelenmesi, bu ihtiyaca yönelik önemli bir adım sunmaktadır (Barros vd., 2024).

Literatürde, çevik yazılım geliştirme projelerinin kritik başarı faktörlerini (KBF) belirlemek (Misra vd., 2006; Nasir ve Sahibuddin, 2011; Silva ve Santos, 2016; Aldahmash vd., 2017; Bergmann ve Karwowski, 2019; Azhar ve Abdullah, 2022; Naslund ve Kale, 2020; Şimşek ve Ateş, 2022; Ojha, 2023; Moustafa ve Abdulasoul, 2024; Zhang vd., 2024), KBF modelleri oluşturmak (Misra vd., 2006; Sudahakar, 2012; Ahimbisibwe vd., 2015) ve KBF'lerin proje başarısına etkisini incelemek (Chow ve Cao, 2008; Misra vd., 2009; Stankovic vd., 2013; Brown, 2015; Vithana vd., 2015; Demirel, 2016; Litchmore, 2016; Nguyen, 2016; Chiyangwa ve Mnkandla, 2017; Aldahmash, 2018; Bergmann, 2018; Stanberry, 2018; Syeda, 2018; Ergüder, 2019; Garousi vd., 2019; Kim vd., 2019; Perera ve Perera, 2019; Shakya ve Shakya, 2020; Tam vd., 2020; Tsoy ve Staples, 2020; Kukreja vd., 2021; Qatanani vd., 2021; Bogopa ve Marnewick, 2022; Fadilah vd., 2022; Greene, 2022; Haskin, 2022; Yeşilyurt, 2022; Heper, 2023; Barros vd., 2024; Binboğa ve Gümüşsoy, 2024; Gwangwadza ve Hanslo, 2024; Jonior vd., 2024) amacıyla pek çok çalışma gerçekleştirilmiştir.

Çevik yazılım geliştirme projelerinde kritik başarı faktörlerinin (KBF) proje başarısına etkisini inceleyen çalışmalar, farklı yaklaşımlar benimsemiştir. Bazı araştırmalar, tüm faktörleri ele alırken, bazıları yalnızca belirli faktörlere odaklanmıştır. Tüm faktörleri kapsayan çalışmalarda, çevik yöntemlerin proje başarısına etkisi daha geniş bir perspektifle ele alınmıştır. Bu bağlamda, Chow ve Cao (2008), Stankovic vd. (2013), Brown (2015), Demirel (2016), Nguyen (2016), Chiyangwa ve Mnkandla (2017),

Aldahmash (2018), Shakya ve Shakya (2020), Tsoy ve Staples (2020), Kukreja vd. (2021), Haskin (2022), Yeşilyurt (2022), Heper (2023), Binboğa ve Gümüşsoy (2024), Gwangwadza ve Hanslo (2024) ile Jonior vd. (2024) gibi çalışmalar, bu kapsayıcı yaklaşımı benimsemiştir.

Belirli faktörlere odaklanan çalışmalar ise daha derinlemesine analizler sunmaktadır. Örneğin, Misra vd. (2009), Vithana vd. (2015) ve Qatanani vd. (2021), organizasyon ve insan faktörlerinin proje başarısındaki önemini incelemiştir. Litchmore (2016), insan ve süreç faktörlerine odaklanırken, Tam vd. (2020), Fadilah vd. (2022) ve Barros vd. (2024) yalnızca insan faktörlerini ele almıştır. Bogopa ve Marnewick (2022), insan, süreç ve teknik faktörleri bir arada değerlendirmiş; Greene (2022) ise organizasyon, teknik ve süreç faktörlerinin rolüne dikkat çekmiştir. Teknik ve sosyal faktörlere odaklanan Fry (2024) ile müşteri faktörlerini ele alan Kim vd. (2019) ve Perera ve Perera (2019) ise daha spesifik bir odak sunmuştur.

Ayrıca, çevik yazılım geliştirme projelerinde KBF'lerin proje başarısına etkisini düzenleyen değişkenlerin ele alındığı araştırmalar da bulunmaktadır. Litchmore (2016), insan ve süreç faktörlerinin proje başarısına etkisinde farklı çevik yöntemlerin kullanımını incelemiştir. Bergmann (2018), KBF'lerin proje başarısına etkisinde proje karmaşıklığının düzenleyici bir rol oynadığını ortaya koymuştur. Garousi vd. (2019), proje faktörünün KBF'lerin proje başarısına etkisinde düzenleyici bir unsur olarak nasıl etkilendiğini ele alırken, Greene (2022), organizasyon, teknik ve süreç faktörlerinin proje başarısına etkisinde takım üyelerinin ortalama yaşının düzenleyici rolünü incelemiştir.

Literatürdeki çalışmalar, çevik yazılım geliştirme projelerinde kritik başarı faktörlerinin proje başarısı üzerindeki etkilerini anlamak için önemli katkılar sunmaktadır. Çevik yazılım geliştirmede insan ve süreç faktörlerinin proje başarısına katkısı sıkça vurgulanmıştır. Bununla birlikte, insan ve süreç faktörlerinin proje başarısına etkisinde çevik pratiklerin düzenleyici rolünü ele alan çalışmaya literatürde rastlanmamıştır. Çevik pratikler, yazılım geliştirme süreçlerinde çevik yöntemlerin temel ilkelerini somutlaştırarak ekiplerin verimliliğini artıran ve yazılım kalitesini iyileştiren tekniklerdir (Beck ve Andres, 2004; Abrantes ve Travassos, 2011). Müşteri ihtiyaçlarına hızlı yanıt verme, esneklik sağlama ve iletişimi güçlendirme gibi avantajlar sunan bu pratikler, sosyal ve teknik boyutları bir araya getirerek yazılım geliştirme projelerindeki

zorlukların üstesinden gelmede kritik bir rol oynamaktadır (Jain ve Suman, 2016; Hummel vd., 2015).

Bu doğrultuda, çevik yazılım geliştirme sürecinde insan ve süreç faktörlerinin proje başarısına etkisinde çevik pratiklerin düzenleyici bir rol üstlenip üstlenmediği önemli bir araştırma konusudur.

Bu çalışmanın amacı, çevik yazılım geliştirmede insan ve süreç faktörlerinin proje başarısına etkisinde çevik pratiklerin düzenleyici rolünü araştırmaktır. Elde edilecek bulgular, çevik projelerin yönetiminde stratejik rehberlik sağlayarak hem akademik literatüre hem de sektörel uygulamalara katkı sunmayı hedeflemektedir.

5.2. ARAŞTIRMANIN KISITLARI

Çevik yazılım geliştirmede insan ve süreç faktörlerinin proje başarısına etkisinde çevik pratiklerin düzenleyici rolünün araştırması amacıyla gerçekleştirilen çalışmaya ilişkin kısıtlar aşağıda yer almaktadır.

- a) Araştırma yalnızca İstanbul'daki Teknoloji Geliştirme Bölgelerinde (Teknoparklar) faaliyet gösteren yazılım geliştirme firmalarında çalışan çevik yazılım geliştiricileri kapsamaktadır. Bu durum, elde edilen sonuçların farklı bölgelerdeki teknoparklara veya genel yazılım sektörüne genellenmesini kısıtlamaktadır.
- b) Araştırma verileri Ağustos 2023 ile Aralık 2023 arasındaki dört aylık bir zaman diliminde toplanmıştır. Kesitsel araştırma olarak gerçekleştirilen çalışmanın sonuçları farklı zaman dilimine genellenmesini kısıtlamaktadır.
- c) Araştırma verilerinin toplanması dijital ortamda (internet) anket formu aracılığıyla gerçekleştirilmiştir. Veri toplama sürecinde kullanılan dijital ortamla ilgili teknik sorunlar veya engeller İstanbul'daki teknoparklarda faaliyet gösteren tüm yazılım geliştirme firmalarına veya firmalardaki çalışanlara ulaşmasını engelleyerek kısıtlayabilir.
- d) Araştırma verileri, kolayda örneklem tekniği kullanılarak toplanmıştır. Ancak bu teknikte elde edilen örneklem, araştırma evrenini temsil etme gücünü sınırlayabilir ve sonuçların genellenmesi açısından kısıtlamalar oluşturabilir.

- e) Çalışmada kullanılan anket sorularının, çevik yazılım geliştirme uygulamalarını tüm yönleriyle kapsamlı bir şekilde değerlendirme kapasitesi sınırlı olabilir. Ayrıca, soruların katılımcılar tarafından farklı yorumlanması sonuçları etkilemiş olabilir.
- f) Araştırma nicel yöntemlerle gerçekleştirilmiş ve veriler anket formu aracılığıyla toplanmıştır. Ancak, derinlemesine bilgi elde etmeye olanak tanıyan nitel veri toplama yöntemleri kullanılmamıştır. Bu durum, araştırmanın yöntemiyle ilgili önemli bir sınırlılığı ifade etmektedir.
- g) Katılımcıların, anketi zaman kaybı, sıkılma veya hızlı bir şekilde tamamlamak gibi nedenlerle gerçekte olduğundan farklı beyanlarda bulunmuş olma ihtimali vardır. Bu durum, araştırma sonuçlarının doğruluğunu olumsuz yönde etkileyebilir.

5.3. ARAŞTIRMA YÖNTEMİ

Çevik yazılım geliştirme projelerinde insan ve süreç faktörlerinin proje başarısına etkisinde çevik pratiklerin düzenleyici rolünü incelemeyi amaçlayan bu nicel araştırma, ilişkisel tarama deseniyle kesitsel olarak gerçekleştirilmiş ve veriler anket yoluyla elde edilmiştir.

Nicel araştırma yöntemleri, sayısal verilere dayanarak kesin ve genellenebilir sonuçlara ulaşmayı hedefleyen sistematik yaklaşımlardır. Bu yöntemler, nesnel gerçeklik varsayımını temel alır ve kişisel yorumlardan bağımsız, tekrarlanabilir gözlem ve ölçüm süreçlerini esas alır (Gürbüz ve Şahin, 2018; Tutar ve Erdem, 2022). Nicel araştırmalarda genellikle temsili büyük örneklemelerden veri toplanır ve bu veriler istatistiksel ve matematiksel yöntemlerle analiz edilerek hipotezlerin test edilmesine olanak tanınır (Gürbüz ve Şahin, 2018). Nicel yöntemler, genelleştirilebilir sonuçlar üretme, gruplar arası karşılaştırmalar yapma ve kuramsal çerçeveleri test etme açısından önemli avantajlar sunar (Lorcu, 2015).

Araştırma kapsamında kullanılan ilişkisel tarama deseni, değişkenler arasındaki ilişkilerin analiz edilmesine ve bu ilişkilerden elde edilen verilere dayanarak olası sonuçların tahmin edilmesine olanak tanır. İki veya daha fazla değişken arasındaki

ilişkileri belirlemek amacıyla yürütülen ve ilişkilerin derinlemesine analiz edildiği çalışmalardır (Gürbüz ve Şahin, 2018; Tutar ve Erdem, 2022).

Bu çalışma, belirli bir zaman dilimindeki olguların incelendiği kesitsel bir tasarımla gerçekleştirilmiştir. Kesitsel araştırmalar, bir olgu ya da örneklemin belirli bir andaki durumunu incelemeyi ve ilgili döneme ilişkin veri toplamayı amaçlamaktadır (Tutar ve Erdem, 2022).

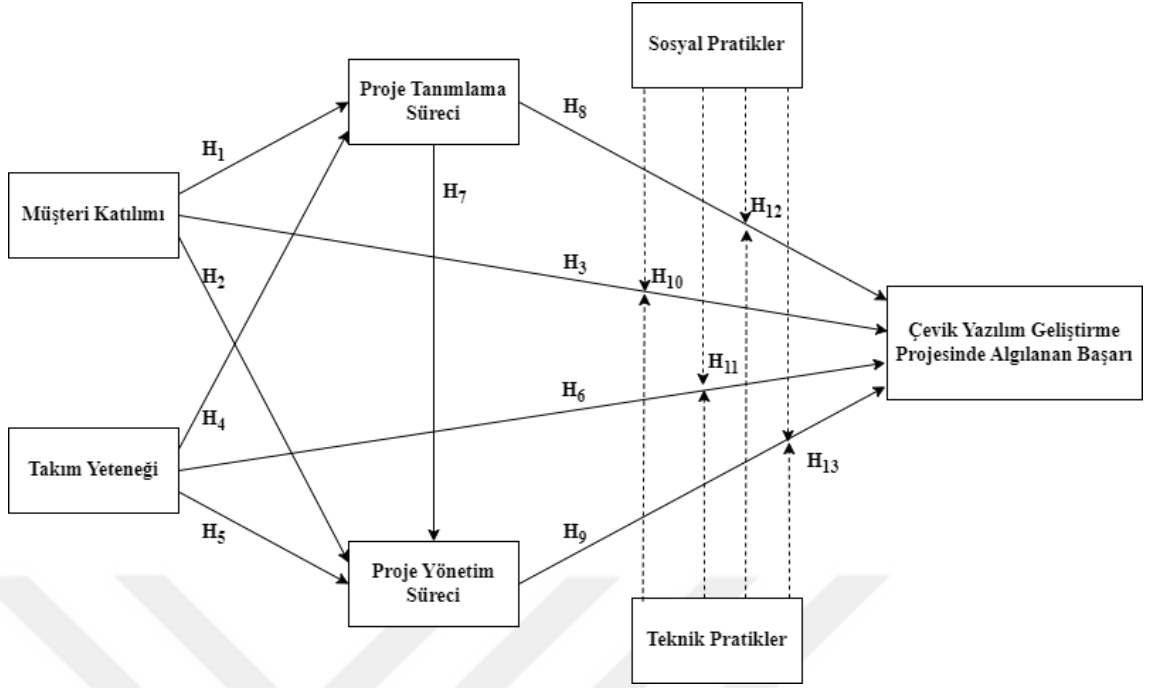
Araştırmada veri toplama aracı olarak anket kullanılmıştır. Anketler, önceden belirlenmiş bir sıralamada ve yapıda oluşturulan soruları içeren araçlardır (Coşkun vd., 2015). Sosyal bilim araştırmalarında yaygın bir şekilde kullanılan anketlerde, genellikle katılımcıların belirli değişkenlere yönelik tutumlarını, algılarını ve davranışlarını ölçen ölçekler yer alır (Gürbüz ve Şahin, 2018). Çeşitli ölçeklerden oluşan bu araçlar, araştırma sorularını yanıtlamak ve hipotezleri test etmek amacıyla tasarlanmıştır (Gürbüz ve Şahin, 2018). Posta, yüz yüze görüşme, telefon veya internet gibi çeşitli yöntemlerle uygulanabilen anketler, geniş bir veri toplama esnekliği sağlamaktadır (Karaca, 2021). Çalışmada veriler, üç farklı likert ölçeği ve demografik bilgileri içeren anket formu kullanılarak dijital ortamda (internet) toplanmıştır.

5.4. ARAŞTIRMA MODELİ VE HİPOTEZLER

Çevik yazılım geliştirmede insan ve süreç faktörlerinin proje başarısına etkisinde çevik pratiklerin düzenleyici rolünü araştırmak amacıyla gerçekleştirilen bu çalışmada, temel olarak aşağıdaki sorulara yanıt aranmaktadır:

1. Çevik yazılım geliştirme projelerinin başarısına katkıda bulunan insan ve süreç faktörleri nelerdir?
2. Çevik yazılım geliştirmede insan ve süreç faktörlerinin proje başarısına etkisinde çevik pratiklerin düzenleyici bir rolü bulunmakta mıdır?

Bu araştırma sorularından yola çıkılarak literatürdeki çalışmalar detaylı bir şekilde incelenmiş ve elde edilen bilgiler ışığında araştırma modeli ile bu modele ilişkin hipotezler geliştirilmiştir. Araştırma modeline ilişkin görselleştirme Şekil 5.1’de sunulmaktadır. Modele ilişkin hipotezler başlıklar halinde belirtilmiştir.



Şekil 5.1. Araştırma Modeli ve Hipotezleri

Kaynak: Chow ve Cao (2008) ve Sudhakar (2012) modeli temel alınarak yazar tarafından oluşturulmuştur.

5.4.1. Müşteri Katılımına İlişkin Hipotezler

Müşteri katılımı, çevik yöntemlerin temel değerlerinden biri olup, müşterinin yazılım geliştirme sürecine ekibin bir parçası olarak aktif şekilde dahil olmasını ifade etmektedir (Beck, 1999; Agile Alliance, 2001b). Müşteri katılımı, gereksinimlerin netleşmesini, belirsizliklerin çözülmesini ve projenin etkili bir şekilde ilerlemesini sağlamaktadır (Hoda vd., 2011; Anderson, 2004). Sudhakar (2012), müşteri katılımının proje yönetim süreci üzerinde doğrudan etkili olduğunu belirtmiştir. Literatürde, müşteri katılımının çevik yazılım geliştirme projelerinin başarısında kritik bir faktör olduğu ve yüksek müşteri katılımının projelerin başarısını olumlu yönde etkilediği geniş bir şekilde vurgulanmaktadır (Misra vd., 2006; Misra vd., 2009; Chow ve Cao, 2008; Ahimbisibwe vd., 2015; Silva ve Santos, 2016; Nguyen, 2016; Aldahmash, 2018; Bergmann ve Karwowski, 2019; Kim vd., 2019; Perera ve Perera, 2019; Thompson, 2019; Shakya ve Shakya, 2020; Tam vd., 2020; Tsoy ve Staples, 2020; Azhar ve Abdullah, 2022; Bogopa ve Marnewick, 2022; Ojha, 2023; Barros vd., 2024; Binboğa ve Gümüşsoy, 2024; Palopak ve Huang, 2024; Zhang vd., 2024).

Yukarıda belirtilenler göz önüne alındığında, aşağıdaki hipotezler öne sürülmektedir:

H₁: Müşteri katılımının proje tanımlama süreci üzerinde anlamlı bir etkisi vardır.

H₂: Müşteri katılımının proje yönetim süreci üzerinde anlamlı bir etkisi vardır.

H₃: Müşteri katılımının çevik yazılım geliştirme proje başarısı üzerinde anlamlı bir etkisi vardır.

5.4.2 Takım Yeteneğine İlişkin Hipotezler

Takım yeteneği, yüksek teknik yeterlilik ve uzmanlık, süreç bilgisi, ekip çalışması ve ürün bilgisi gibi unsurları kapsayan bir kavramdır (Stankovic vd., 2013; Šmite vd., 2017). Bu yetenekler, yazılım geliştirme süreçlerinde ekiplerin verimliliğini artırarak proje yönetim süreçlerinin etkinliğine katkıda bulunmaktadır (Kurapati vd., 2012). Literatürde, takım yeteneğinin çevik yazılım projelerinin başarısında kritik bir faktör olduğu sıklıkla vurgulanmıştır. Misra vd. (2009) yetkin bir ekibin müşteri gereksinimlerini hızlıca karşılayabileceğini ifade ederken, Ahimbisibwe vd. (2015), takım yeteneğinin riskleri algılama ve yanıt verme kapasitesini artırdığını belirtmektedir. Sudhakar (2012) ise takım faktörleri içerisinde yer alan takım yeteneğinin, proje yönetim faktörlerini (örneğin proje planlaması, net proje hedefleri, proje kontrol mekanizmaları) doğrudan etkilediğini vurgulamaktadır.

Chow ve Cao (2008), takım yeteneğini çevik projelerdeki üç temel kritik başarı faktöründen biri olarak tanımlarken, Aldahmash (2018), projenin ilerleyen aşamalarında bu faktörün öneminin arttığını vurgulamaktadır. Sonuç olarak, takım yeteneği, çevik yazılım geliştirme projelerinin başarısını etkileyen kritik bir başarı faktörü olarak tanımlanmaktadır (Misra vd., 2006; Chow ve Cao, 2008; Vithana vd., 2015; Litchmore, 2016; Aldahmash, 2018; Stanberry, 2018; Garousi vd., 2019; Shakya ve Shakya, 2020; Tam vd., 2020; Tsoy ve Staples, 2020; Qatanani vd., 2021; Azhar ve Abdullah, 2022; Fadilah vd., 2022; Haskin, 2022; Heper, 2023; Barros vd., 2024; Gwangwadza ve Hanslo, 2024; Zhang vd., 2024).

Yukarıda belirtilenler göz önüne alındığında, aşağıdaki hipotezler öne sürülmektedir:

H4: Takım yeteneğinin proje tanımlama süreci üzerinde anlamlı bir etkisi vardır.

H5: Takım yeteneğinin proje yönetim süreci üzerinde anlamlı bir etkisi vardır.

H6: Takım yeteneğinin çevik yazılım geliştirme proje başarısı üzerinde anlamlı bir etkisi vardır.

5.4.3 Proje Tanımlama Sürecine İlişkin Hipotezler

Proje tanımlama süreci, bir projenin hedeflerini, kapsamını ve kısıtlarını netleştirmek amacıyla yürütülen kritik bir aşama olup, paydaşlar arasında ortak bir anlayış oluşturmayı hedeflemektedir (PMI, 2021; Kerzner, 2017). Bu süreç, aynı zamanda sonraki tüm proje faaliyetleri için bir temel oluşturarak, paydaş beklentilerinin etkin bir şekilde yönetilmesinde önemli bir rol oynamaktadır (Kerzner, 2017).

Yazılım projelerinde proje tanımlama, müşteri gereksinimlerini karşılamak ve proje hedeflerine ulaşmak için yöntemlerin, iş takvimlerinin ve kaynakların proje kısıtları doğrultusunda planlanmasını kapsamaktadır (Nizam, 2015). Ayrıca, potansiyel risklerin tanımlanması ve bu risklere yönelik önlemlerin değerlendirilmesi, sürecin önemli bir parçasıdır (Wysocki, 2006; Hughes vd., 2011; Kerzner, 2017).

Çevik yöntemlerde proje tanımlama süreci, müşteri ile yakın iş birliği içinde yürütülmekte olup, süreç boyunca ortaya çıkan değişikliklere hızlı bir şekilde uyum sağlamayı kolaylaştırmaktadır (Beck ve Andres, 2004; Martin, 2014). Bu süreç, proje hedeflerinin, kapsamının ve kısıtlarının net bir şekilde belirlenmesini sağlayarak projenin etkin bir şekilde yönetilmesini ve başarıya ulaşmasını desteklemektedir. Literatürde, proje tanımlama süreci, çevik yazılım geliştirme projelerinde kritik bir başarı faktörü olarak değerlendirilmektedir (Sudhakar vd., 2012; Stankovic vd., 2013; Brown, 2015; Serrador ve Pinto, 2015; Litchmore, 2016; Stanberry, 2018; Haskin, 2022).

Yukarıda belirtilenler göz önüne alındığında, aşağıdaki hipotezler öne sürülmektedir:

H7: Proje tanımlama sürecinin proje yönetim süreci üzerinde anlamlı bir etkisi vardır.

H8: Proje tanımlama sürecinin çevik yazılım geliştirme proje başarısı üzerinde anlamlı bir etkisi vardır.

5.4.4. Proje Yönetim Sürecine İlişkin Hipotezler

Çevik yazılım projeleri, gereksinim yönetimi, proje yönetimi ve konfigürasyon yönetimi gibi süreçleri çevik yaklaşımla uygulamaktadır. Örneğin, gereksinimler başlangıçta genel bir düzeyde belirlenerek proje ilerledikçe yorumlama ve uyarlama imkânı sağlamaktadır (Koch, 2005, akt. Cao, 2006).

Projenin erken aşamalarındaki belirsizlikler ve değişen gereksinimler, çevik projelerde esnek bir proje yönetim süreci yaklaşımını zorunlu kılmaktadır (Aldahmash, 2018). Çevik yazılım geliştirme projelerinin başarısında, uygun proje yönetim süreçlerinin seçimi önemli bir katkı sağlamaktadır (Chow ve Cao, 2008; Stankovic vd., 2013).

Literatürde yapılan çeşitli çalışmalar, çevik proje yönetim sürecinin proje başarısına önemli katkılar sağladığını ortaya koymaktadır (Chow ve Cao, 2008; Sudhakar, 2012; Stankovic vd., 2013; Litchmore, 2016; Nguyen, 2016; Aldahmash, 2018; Stanberry, 2018; Ergüder, 2019; Garousi vd., 2019; Kim vd., 2019; Shakya ve Shakya, 2020; Tsoy ve Staples, 2020; Haskin, 2022; Ojha, 2023; Binboğa ve Gümüşsoy, 2024).

Yukarıda belirtilenler göz önüne alındığında, aşağıdaki hipotez öne sürülmektedir:

H₉: Proje yönetim sürecinin çevik yazılım geliştirme proje başarısı üzerinde anlamlı bir etkisi vardır.

5.4.5. Çevik Pratiklerin Düzenleyici Rolüne İlişkin Hipotezler

Çevik pratikler, çevik yöntemlerin temel ilkelerini somutlaştırarak ekiplerin verimliliğini artıran ve yazılım kalitesini iyileştiren faaliyetlerdir (Beck ve Andres, 2004; Abrantes ve Travassos, 2011). Bu pratikler, müşteri ihtiyaçlarına hızlı yanıt verme ve değişen gereksinimlere uyum sağlama esnekliği sunarken, aynı zamanda iletişim ve iş birliğini güçlendirir (Jain ve Suman, 2016; Al-Sakkaf vd., 2017). Çevik pratikler, yazılım geliştirme projelerinde sıkça karşılaşılan zorlukların üstesinden gelmek için hem sosyal hem de teknik boyutları bir araya getiren bir yaklaşımdır (Hummel vd., 2015).

Literatürde, çevik pratikler teknik ve sosyal olmak üzere iki ana gruba ayrılmaktadır (Anjani vd., 2021). Teknik pratikler, birim testi, sürekli entegrasyon ve kodlama standartları gibi uygulamalarla yazılımın kalite ve sürdürülebilirliğini artırmayı

hedeflerken, sosyal pratikler ekip içi iletişim ve müşteri etkileşimlerini desteklemektedir (Chow ve Cao, 2008; So ve Scholl, 2009). Neumann (2022), çevik pratiklerin teknik, iş birliği, süreç, gereksinim ve organizasyonel anlamda katkılar sağlayacağını belirterek pratikleri sınıflandırmıştır.

Çevik pratiklerin proje başarısına katkı sunduğunu gösteren çeşitli çalışmalar bulunmaktadır (Henriksen ve Pedersen, 2017; Recker vd., 2017; Anjani vd., 2021; Mulugeta, 2021; Sandstø ve Reme-Ness, 2021). Ghimire ve Charters (2022), ayaküstü toplantılar, ürün iş listesi ve retrospektiflerin iletişim sorunlarını azalttığını ve müşteri gereksinimlerinin daha net tanımlanmasını sağladığını belirtmiştir. Jain ve Suman (2016), sürekli entegrasyon ve sprint gibi pratiklerin proje ilerlemesini izlemek ve sorunların erken çözümüne katkıda bulunmak açısından etkili olduğunu vurgulamıştır.

Çevik pratiklerin etkinliği uygulama şekline ve süreçlere entegrasyon düzeyine bağlıdır. Leal vd. (2019), çevik pratiklerin seçiminin proje başarısına katkısını vurgulamıştır. Ayrıca, bu pratiklerin uyumlu bir şekilde bir arada uygulanması, yazılım geliştirme süreçlerinde başarıyı artırmaktadır (Beck ve Andres, 2004; Martin, 2014).

Yukarıda belirtilenler doğrultusunda, insan ve süreç faktörlerinin çevik yazılım geliştirme projelerinin başarısına etkisinde çevik pratiklerin düzenleyici bir rol oynayabileceği varsayımıyla aşağıdaki hipotezler öne sürülmektedir:

H₁₀: Müşteri katılımının çevik yazılım geliştirme proje başarısı üzerindeki etkisinde çevik pratiklerin düzenleyici rolü vardır.

H_{10a}: Müşteri katılımının çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal pratiklerin düzenleyici rolü vardır.

H_{10b}: Müşteri katılımının çevik yazılım geliştirme proje başarısı üzerindeki etkisinde teknik pratiklerin düzenleyici rolü vardır.

H_{10c}: Müşteri katılımının çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal ve teknik pratiklerin birlikte düzenleyici rolü vardır.

H₁₁: Takım yeteneğinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde çevik pratiklerin düzenleyici rolü vardır.

H_{11a}: Takım yeteneğinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal pratiklerin düzenleyici rolü vardır.

H_{11b}: Takım yeteneğinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde teknik pratiklerin düzenleyici rolü vardır.

H_{11c}: Takım yeteneğinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal ve teknik pratiklerin birlikte düzenleyici rolü vardır.

H₁₂: Proje tanımlama sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde çevik pratiklerin düzenleyici rolü vardır.

H_{12a}: Proje tanımlama sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal pratiklerin düzenleyici rolü vardır.

H_{12b}: Proje tanımlama sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde teknik pratiklerin düzenleyici rolü vardır.

H_{12c}: Proje tanımlama sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal ve teknik pratiklerin birlikte düzenleyici rolü vardır.

H₁₃: Proje yönetim sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde çevik pratiklerin düzenleyici rolü vardır.

H_{13a}: Proje yönetim sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal pratiklerin düzenleyici rolü vardır.

H_{13b}: Proje yönetim sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde teknik pratiklerin düzenleyici rolü vardır.

H_{13c}: Proje yönetim sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal ve teknik pratiklerin birlikte düzenleyici rolü vardır.

5.5. ARAŞTIRMANIN EVRENİ VE ÖRNEKLEMİ

Evren, araştırma bulgularının genellendiği ve araştırma örnekleminin seçildiği büyük grubu ifade etmektedir. Ortak özelliklere sahip tüm öğeler, bu evrenin bir parçası olarak değerlendirilir (Gürbüz ve Şahin, 2018). Araştırmanın evrenini İstanbul'da Teknoloji Geliştirme Bölgelerinde (Teknopark) yer alan yazılım geliştirme firmalarında

çeşitli rollerde (yazılım geliştirici, test sorumlusu, takım lideri, çevik koç vb.) çalışanlar oluşturmaktadır. İstanbul'da 9'u faaliyette, 2'si alt yapı çalışması devam eden olmak üzere 11 teknopark yer almaktadır (T.C. Sanayi ve Teknoloji Bakanlığı, 2021). Faaliyetine devam eden teknoparklarda iletişime geçilerek yazılım firma ve çalışan sayısı bilgisine ulaşılmıştır. Toplam 828 yazılım geliştirme firmasında çeşitli rollerde 17.540 çalışan yer almaktadır. İstanbul'da yer alan teknoloji geliştirme bölgelerinde yazılım geliştirme faaliyetinde bulunan toplam firma ve çalışan sayısı ile ilgili detaylı bilgi Tablo 5.1'de sunulmuştur.

Tablo 5.1. İstanbul'da Yer Alan Teknoloji Geliştirme Bölgelerinde Yazılım Geliştirme Faaliyetinde Bulunan Toplam Firma ve Çalışan Sayısı

Teknoloji Geliştirme Bölgesi	Kuruluş Yılı	Yazılım Firma Sayısı	Çalışan Sayısı
İstanbul Üniversitesi Teknopark	2003	60	600
İTÜ Arı Teknokent	2003	152	4383
Yıldız Teknik Üniversitesi Teknopark	2003	304	7214
Boğaziçi Üniversitesi Teknopark	2009	22	147
İstanbul Teknopark	2009	187	4070
Marmara Üniversitesi Teknopark	2014	37	215
Dudullu Organize Sanayi Bölgesi Boğaziçi Üniversitesi	2018	34	594
İstanbul Medeniyet Üniversitesi Teknopark	2018	14	202
Sağlık Bilimleri Üniversitesi Teknopark	2018	18	115
İstanbul Sebahattin Zaim Üniversitesi Teknopark*	2018	-	-
Esenler Akıllı Şehir Odaklı İhtisas Teknopark*	2021	-	-
TOPLAM		828	17540

Not: Tabloda yer alan veriler Aralık 2021 yılına aittir. * Altyapı çalışmaları devam ediyor.

Örneklem, belirli bir evren içerisindeki birimlerden, sistematik ve bilimsel yöntemlerle seçilen, evreni temsil etme yeterliliğine sahip daha küçük bir grubu ifade etmektedir (Gülbüz ve Şahin, 2018). Araştırmada kolayda örnekleme yöntemi kullanılarak veriler toplanmıştır. Kolayda örnekleme, araştırmacının kolay erişilebilen bireylerden veri topladığı ve ihtiyaç duyulan örneklem büyüklüğüne ulaşmayı amaçladığı bir örnekleme yöntemidir. Hızlı ve düşük maliyetli bir yaklaşım sunmasına karşın, bu yöntemle elde edilen örneklemin evreni temsil etme yeterliliği sınırlıdır. Bununla birlikte, sosyal bilimlerde bu yöntemin yaygın olarak kullanıldığına dair bulgular bulunmaktadır (Gülbüz ve Şahin, 2018).

Araştırma anakütlesini temsil edecek örneklem büyüklüğü genellikle anakütleden %95 güvenilirlik sınırları içerisinde %5’lik hata payı dikkate alınarak hesaplanmaktadır (Coşkun vd., 2015). Sekaran (2003) tarafından hazırlanan tabloya göre, %95 güven seviyesi ve %5 hata payı dikkate alındığında, 20.000 kişilik bir anakütle için gerekli minimum örneklem büyüklüğü 377 olarak belirtilmiştir. Araştırmada, kolayda örnekleme yöntemi kullanılarak 464 katılımcıdan veri toplanmıştır.

5.6. VERİ TOPLAMA ARAÇLARI

Nicel yöntemlerin kullanıldığı bu çalışmada, veriler üç farklı likert ölçeği ve demografik bilgileri içeren bir anket formu aracılığıyla toplanmıştır. Araştırma kapsamında kullanılan anket formu Ek 1’de sunulmaktadır.

5.6.1 Demografik Bilgiler

Demografik bilgi bölümü, literatürdeki çalışmalar incelenerek yapılandırılmıştır. Bu bölümde, katılımcılardan cinsiyet, yaş, eğitim düzeyi, çevik yazılım geliştirme alanındaki toplam deneyim süresi, çalıştıkları firmanın faaliyet gösterdiği sektör, mevcut pozisyonları ve kullandıkları çevik yöntemlere ilişkin verilerin toplanmasını amaçlayan yedi soru yer almaktadır.

5.6.2 Çevik Başarı Faktörleri Ölçeği

Çevik Başarı Faktörleri Ölçeği (ÇBF), ilk kez Cao (2006) tarafından doktora tezinde, çevik proje yönetimine yönelik geçmiş literatür taramalarına dayanılarak geliştirilmiştir. Söz konusu ölçek, çevik projelerdeki kritik başarı faktörlerini değerlendirmek amacıyla yapılan araştırmalarda sıklıkla kullanılan bir araçtır ve 12 faktörden oluşmaktadır (Chow ve Cao, 2008). Bu çalışma kapsamında, ölçeğin insan ve süreç faktörlerine ilişkin dört faktörüne ait toplam 17 madde kullanılmıştır. Ölçek maddeleri, katılımcıların görüşlerini ölçmek üzere 7’li Likert tipi bir formatta düzenlenmiştir. Katılımcıların katılım düzeylerini ifade etmelerine olanak tanıyan bu ölçek, (1) Kesinlikle katılmıyorum, (2) Katılmıyorum, (3) Biraz katılmıyorum, (4) Ne katılmıyorum ne katılıyorum, (5) Biraz katılıyorum, (6) Katılıyorum ve (7) Kesinlikle katılıyorum şeklinde tasarlanmıştır.

Tablo 5.2. Araştırmada Kapsamında Kullanılacak İnsan ve Süreç Faktörlerine İlişkin Bilgiler

Faktör	Soru Sayısı	Madde Özelliği
İNSAN FAKTÖRLERİ	Takım yeteneği	5
		Yüksek yetkinlik ve uzmanlığa sahip ekip üyeleri Yüksek motivasyona sahip ekip üyeleri Çeviklik konusunda bilgili yöneticiler Hafif dokunuşlu veya uyarlanabilir yönetim tarzına sahip yöneticiler Takıma uygun teknik eğitim
İNSAN FAKTÖRLERİ	Müşteri Katılımı	3
		İyi müşteri ilişkileri Güçlü müşteri bağlılığı ve varlığı Tam yetkiye sahip müşteri
SÜREÇ FAKTÖRLERİ	Proje Yönetim	6
		Çevik odaklı gereksinim yönetimi sürecini takip etmek Çevik odaklı proje yönetim sürecini takip etmek Çevik odaklı konfigürasyon yönetimi sürecini takip etmek İyi ilerleme izleme mekanizması Günlük yüz yüze toplantılarla güçlü iletişim odağı Düzenli çalışma programını onurlandırmak
SÜREÇ FAKTÖRLERİ	Proje Tanımlama	3
		Proje kapsamı iyi tanımlanmış Ön maliyet değerlendirmesi yapılan projeler Ön risk analizi yapılmış projeler

5.6.3. Çevik Yazılım Geliştirme Projesinde Algılanan Başarı Ölçeği

Çevik Yazılım Geliştirme Projesinde Algılanan Başarı Ölçeği (ÇYGPAB) Cao ve Chow (2008) çalışmasında yer alan proje başarısını maliyet, kapsam, zaman ve kalite açısından değerlendirmektedir. 4 madde içeren ölçek 7'li likert şeklindedir. Katılımcıların ölçek maddelerine ilişkin proje başarı algılarını 1- Kesinlikle katılmıyorum, 2- Katılmıyorum, 3- Biraz katılmıyorum, 4- Ne katılmıyorum ne katılıyorum, 5- Biraz katılıyorum, 6- Katılıyorum, 7- Kesinlikle katılıyorum şeklinde belirtmeleri istenmiştir.

Tablo 5.3. ÇYGPAB Ölçeği ve Açıklamaları

Madde	Açıklama
Kalite	İyi ürün veya proje çıktısı sunma
Kapsam	Tüm gereksinim ve hedeflerin karşılanması
Zaman	Zamanında teslim edilmesi
Maliyet	Tahmini maliyet ve çaba dâhilinde teslim edilmesi

5.6.4 Çevik Pratiklerin Kullanımı Ölçeği

Çevik Pratiklerin Kullanımı Ölçeği (ÇPK) Tripp vd. (2016) çalışmasında yer alan “Çevik Pratik Kullanımları” ölçeğinden oluşturulmuştur. Çalışmada yer alan pratikler; Türkiye 8. Çeviklik Raporu (2019), 14th Annual State Of Agile Report (2020) ve literatürde yaygın olarak kullandığı belirtilen pratikler (Jalali ve Wohlin, 2011; Diel vd., 2015; Licorish vd., 2016; Pantiuchina vd., 2017; Paez vd., 2018; Leal vd., 2019; Paez ve Oliveros, 2020) arasından seçilmiştir. Araştırmalarda çevik pratikler farklı şekillerde gruplandırılmıştır. Anjani vd. (2021) ve Gupta vd. (2019) sosyal ve teknik; Recker vd. (2017) standart ve normlar, yönetim ve geliştirme; Acros-Medina ve Mauricio (2020) takım çalışması, proje yönetimi, mühendislik ve test; Neumann (2022) iş birliği, teknik, süreç, organizasyonel ve gereksinim; Pantiuchina vd. (2017) kalite, hız, iletişim; Paez vd. (2018) teknik ve organizasyon; Tripp vd. (2016) proje yönetim ve yazılım geliştirme pratikleri şeklinde gruplandırmıştır. Gupta vd. (2019), çevik pratiklerin sosyal ve teknik olarak ayrılmasının akademik literatürde yaygın bir fikir birliğiyle kabul edildiğini belirtmiştir.

Araştırmada çevik pratikler sosyal ve teknik şeklinde ifade edilecektir. Sosyal pratikler, organizasyonda sosyal etkileşimi, iş birliğini ve doğrudan iletişimi kolaylaştıran uygulamaları ifade etmektedir. Teknik pratikler, kodlama ve test süreçlerine odaklanan yazılım mühendisliği uygulamalarını tanımlamaktadır (Gupta vd. 2019).

7 faktör ve 2 boyuttan oluşan ölçekte 22 madde yer almaktadır. Ölçekte yer alan maddeler 7’li likert şeklinde düzenlenmiştir. Ölçekte yer alan faktörlere ilişkin bilgiler tabloda belirtilmiştir.

Tablo 5.4. ÇPK Ölçeğinin Faktörleri ve Madde Sayısı

Çevik Pratik Sınıfı	Çevik Pratik	Madde Sayısı
Sosyal Pratikleri	Yinelemeli Teslimat (Sprint/Yineleme Planlama)	4
	Ayaküstü Toplantı	3
	Retrospektif	3
Teknik Pratikleri	Kodlama Standartları	3
	Birim Test	3
	Sürekli Entegrasyon	3
	Yeniden Kod Düzenleme	3

5.6.5. Ölçme Araçlarının Türkçe 'ye Uyarlanması ve Pilot Uygulama

5.6.5.1. Ölçme Araçlarının Türkçe 'ye Uyarlanma Süreci

Ölçek uyarlama yabancı bir dilde geliştirilerek güvenilirliği ve geçerliği sınanmış olan bir ölçeğin başka bir dilde ve kültürde güvenilirlik ve geçerlilik çalışmasının yapılarak kullanıma hazır hale getirilmesidir (Seçer, 2015).

Araştırmada kapsamında kullanılacak ölçekler, detaylı literatür taraması gerçekleştirilerek belirlendikten sonra İngilizce 'den Türkçe 'ye uyarlama yapılması amacıyla öncelikle ölçekleri geliştiren araştırmacılarla iletişime geçilmiştir. “Çevik Başarı Faktörleri Ölçeği” ve “Çevik Yazılım Geliştirme Projesinde Algılanan Başarı Ölçeği” için Dac-Buu Cao ile iletişime geçilerek izin istenmiştir. “Çevik Pratiklerin Kullanımı Ölçeği” için araştırmanın ilk yazarı Dr. John F. Tripp ile e-mail yoluyla iletişime geçilerek izin istenmiştir. Ayrıca “Çevik Pratiklerin Kullanımı Ölçeği”nde yer alan bir ifadeyle ilgili kendisinin görüşüne başvurulmuştur. Yazışmalar sonucu alınan izin doğrultusunda ölçeğin uyarlama sürecine başlanmıştır. İlgili izinler Ek 2’de yer almaktadır.

Ölçeklerin Türkçe 'ye çevrilmesi üniversitede yabancı dil eğitimi veren üç öğretim üyesi tarafından gerçekleştirilmiştir. Ölçeklerde yer olan teknik ifadelerle ilgili biri üniversitede konuyla ilgili araştırmalar gerçekleştiren öğretim üyesi, diğer özel sektörde çevik yöntemler konusunda tecrübeli iki alan uzmanının görüşüne başvurulmuştur. Türkçe 'ye çevrilen ölçekler formları karşılaştırılarak dil, kültür ve kuramsal açıdan değerlendirilmiştir. Bu aşamada çevrilen ölçeklerde yer alan ifadelerin anlaşılabilirliği üniversitede öğretim üyesi olarak görevli 2 Türk Dili uzmanı ve teknik terimler alan uzmanları tarafından kontrol edilmiştir. Araştırma kapsamında kullanılan ölçeklerin dil geçerliliği uzman görüşüne başvurularak gerçekleştirilmiştir. Türkçe'ye çevirisi yapılmış olan ölçeğe ilk şeklinin verilmesinin ardından pilot çalışma gerçekleştirilmiştir.

5.6.5.2. Pilot Uygulama

Pilot uygulamalar madde analizi ve maddelerin ölçeğin bütünüyle olan uyumunu incelemek amacıyla ölçek uyarlama ve ölçek geliştirme çalışmalarında kullanılmaktadır (Seçer, 2015). Seçer (2015), pilot uygulamanın ölçeğin iç tutarlılığının ve madde toplam

korelasyon uyumlarının incelenmesi amacıyla 50 civarında örneklem ile gerçekleştirilmesinin yeterli olacağını ifade etmiştir. Ölçeğin iç tutarlılığının 0,70 üzerinde ve madde toplam korelasyon değerinin 0,30 üzerinde olup olmadığı kontrol edilerek sorunlu maddelerin tespit edilerek düzenlenmesi gerekmektedir.

Tablo 5.5. Araştırmanın Pilot Çalışmasına Katılımcılara İlişkin Demografik Veriler

Değişken	Grup	Frekans (f)	Yüzde (%)
Cinsiyet	Erkek	42	82,4
	Kadın	9	17,6
Eğitim Düzeyi	Lise	2	3,9
	Ön lisans	3	5,9
	Lisans	35	68,6
	Yükseklisans	8	15,7
	Doktora	3	5,9
Çevik Deneyim	1 yıl ve az	12	23,5
	1-3 yıl	14	27,5
	4-6 yıl	12	23,5
	7-9 yıl	4	7,8
	10 ve üzeri	9	17,6
Firmanın Faaliyet Gösterdiği Sektör	Bankacılık/Finans	7	13,7
	Bilgi Teknolojileri	27	52,9
	Enerji	4	7,8
	E-Ticaret	1	2,0
	Savunma Sanayi	2	3,9
	Telekomünikasyon	6	11,8
	Yazılım Firması	3	5,9
	Endüstriyel Ürün	1	2,0
Firmadaki Görev	Proje Yöneticisi	5	9,8
	Takım Lideri	6	11,8
	Scrum Master	4	7,8
	Yazılım Geliştirici	24	47,1
	Müşteri/Ürün Sahibi	1	2
	Takım Üyesi	10	19,6
	Baş Teknoloji Danışmanı	1	2
Yazılım Geliştirmede Kullanılan Çevik Yöntemler	Scrum	42	82,4
	Kanban	14	27,5
	XP	9	17,6
	Scrum XP Karması	7	13,7
	Scrumban	4	7,8

Bu kapsamda anket formu Google Forms ile dijital ortama aktarılmıştır. Anket formunun linki Samsun, Sivas ve Tokat'ta yer alan Teknoparklarda çevik yazılım yöntemleri kullanan kişilere gönderilmiştir. Pilot çalışmaya 51 kişi katılmıştır. Katılımcılara ilişkin demografik bilgiler Tablo 5.5'te yer almaktadır.

ÇBF ölçeğine ilişkin madde analizi sonuçları Tablo 5.6'da yer almaktadır. Tablo incelendiğinde ölçek maddelerine ilişkin madde toplam korelasyon değerlerinin 0,549 ile 0,832 arasında değişmektedir. İç tutarlık değeri Cronbach's Alpha (α) ölçeğin alt boyutlarında 0,833 ile 0,867 arasında iken ölçek genelinde 0,946'dır. Bu sonuçlar ölçekte sorunlu bir maddenin yer almadığını ve ölçeğin güvenilir olduğunu göstermektedir.

Tablo 5.6. ÇBF Ölçeği Madde Analizi

Boyut	Madde	Madde Toplam Korelasyon Değeri	Madde Çıkarıldığında Cronbach's Alpha Değeri	Cronbach's Alpha Değeri
Takım Yeteneği	TK1	0,736	0,823	0,861
	TK2	0,771	0,820	
	TK3	0,746	0,815	
	TK4	0,623	0,846	
	TK5	0,629	0,862	
Müşteri Katılımı	MK1	0,600	0,855	0,833
	MK2	0,791	0,664	
	MK3	0,707	0,755	
Proje Yönetimi	PY1	0,637	0,849	0,867
	PY2	0,696	0,838	
	PY3	0,794	0,826	
	PY4	0,668	0,843	
	PY5	0,699	0,839	
	PY6	0,549	0,871	
Proje Tanımlama	PT1	0,601	0,893	0,848
	PT2	0,832	0,669	
	PT3	0,754	0,756	

ÇYGPAB ölçeğine ilişkin madde analizi sonuçları Tablo 5.7'de yer almaktadır. Tablo 5.7 incelendiğinde ölçek maddelerine ilişkin madde toplam korelasyon değerlerinin 0,450 ile 0,720 arasında değişmektedir. İç tutarlık değeri 0,780'dır. Bu sonuçlar ölçekte sorunlu bir maddenin yer almadığını ve ölçeğin güvenilir olduğunu göstermektedir.

Tablo 10. ÇYGPAB Ölçeği Madde Analizi

Boyut	Madde	Madde Toplam Korelasyon Değeri	Madde Çıkarıldığında Cronbach's Alpha Değeri	Cronbach's Alpha Değeri
Proje Başarısı	KALITE	0,719	0,680	0,780
	KAPSAM	0,720	0,690	
	ZAMAN	0,622	0,707	
	MALİYET	0,450	0,851	

ÇPK ölçeğine ilişkin madde analizi sonuçları Tablo 5.8’de yer almaktadır. Tablo 5.8 incelendiğinde ölçek maddelerine ilişkin madde toplam korelasyon değerlerinin 0,437 ile 0,868 arasında değişmektedir. İç tutarlık değeri ölçeğin alt boyutlarında 0,714 ile 0,884 arasında iken ölçek genelinde 0,932’dir.

Tablo 11. ÇPK Ölçeği Madde Analizi

Boyut	Madde	Madde Toplam Korelasyon Değeri	Madde Çıkarıldığında Cronbach's Alpha Değeri	Cronbach's Alpha Değeri
Yinelemeli Teslimat	SYT1	0,702	0,841	0,869
	SYT2	0,791	0,808	
	SYT3	0,742	0,825	
	SYT4	0,666	0,859	
Ayaküstü Toplantı	SAT1	0,738	0,868	0,884
	SAT2	0,846	0,774	
	SAT3	0,744	0,863	
Retrospektif	SR1	0,744	0,848	0,874
	SR2	0,729	0,857	
	SR3	0,831	0,753	
Birim Test	TBT1	0,656	0,767	0,817
	TBT2	0,723	0,696	
	TBT3	0,646	0,771	
Sürekli Entegrasyon	TSE1	0,427	0,783	0,730
	TSE2	0,624	0,592	
	TSE3	0,696	0,470	
Kodlama Standartları	TKS1	0,755	0,761	0,851
	TKS2	0,687	0,825	
	TKS3	0,728	0,788	
Yeniden Kod Düzenleme	TYKD1	0,448	0,733	0,714
	TYKD2	0,765	0,437	
	TYKD3	0,574	0,719	

Pilot uygulama sonrası ölçeklerin iç tutarlılığının 0,70 üzerinde ve madde toplam korelasyon değerinin 0,30 üzerinde olması nedeniyle herhangi bir sorunlu madde olmadığı sonucuna ulaşılmıştır.

5.7. VERİ TOPLAMA SÜRECİ

Atatürk Üniversitesi Sosyal ve Beşerî Bilimler Etik Kurul Başkanlığı'nın 21/11/2022 tarih ve E.88656144-000-2200383449 sayılı kararı doğrultusunda alınan etik kurul onayından sonra, Ağustos-Aralık 2023 (4 ay) tarihleri arasında veri toplama süreci gerçekleştirilmiştir. Etik kurul onay belgesi EK-3'te sunulmaktadır. Kolayda örnekleme yöntemi kullanılarak dijital ortamda (internet üzerinden) 464 katılımcıdan veri toplanmıştır.

5.8. VERİLERİN ANALİZİ

5.8.1. Veri Setinin Kontrolü ve Analize Hazırlanması

Araştırma kapsamında toplanan verilerin doğru bir şekilde analiz edilebilmesi için, öncelikle veri setinin detaylı bir kontrol sürecinden geçirilmesi ve analizlere uygun hale getirilmesi gerekmektedir (Tabachnick ve Fidell, 2013; Çokluk vd., 2021).

Tabachnick ve Fidell (2013) veri setinin kontrol ve analizlere hazırlama sürecinin, veri setinin doğruluğu, eksik veri analizi, uç değerler, normallik testi ile çoklu doğrusallık ve tekillik aşamalarını içerdiğini belirtmiştir.

5.8.2.1. Veri Setinin Doğruluğu

Veri setinin doğruluğunu sağlamanın en etkili yolu, orijinal verileri analiz için kullanılacak programa (SPSS, SAS vb.) aktarılan veri setiyle karşılaştırmak ve hatalı veri tespit edildiğinde bu hataları düzeltmektir (Tabachnick ve Fidell, 2013). Bu yöntem küçük veri setleri için kolaylıkla uygulanabilirken, büyük veri setlerinde birebir karşılaştırma ve düzeltme işlemleri genellikle mümkün olmamaktadır. Bu durumda, büyük veri setlerinin doğruluğu tanımlayıcı istatistikler kullanılarak değerlendirilmektedir. Nicel değişkenler için, tüm değerlerin olası sınırlar arasında bulunup bulunmadığına, ortalama ve standart sapma değerlerinin makul olup olmadığına dikkat edilir. Ayrıca, kategorik değişkenler

için tüm değerlerin kategori kodlarına karşılık gelip gelmediği kontrol edilerek doğruluk analizi gerçekleştirilir (Tabachnick ve Fidell, 2013; Çokluk vd., 2021).

Araştırma veri setinin doğruluğunu belirlemek amacıyla ilk olarak, orijinal verilerin programa girişi sırasında kodlamalarda olası hataları tespit etmek üzere karşılaştırma yapılacaktır. Nicel değişkenler için, değerlerin belirlenen olası sınırlar içinde bulunup bulunmadığını kontrol etmek amacıyla maksimum ve minimum değerlere göz atılacaktır. Ayrıca, her bir nicel değişkenin ortalama ve standart sapma değerleri incelenecek; standart sapmanın ortalamadan büyük olduğu durumlar belirlenerek gerekli düzeltmeler planlanacaktır. Kategorik değişkenler için ise, tüm değerlerin doğru kategori kodlarına karşılık gelip gelmediği kontrol edilecektir. Eğer hatalı bir kodlama tespit edilirse, bu hatalar düzeltilerek veri setinin doğruluğu sağlanacaktır.

5.8.2.2. Eksik Veri Analizi

Eksik (kayıp) veriler, veri analizindeki en yaygın sorunlardan biridir (Tabachnick ve Fidell, 2013). Anket sorularının belirli bir kısmının cevaplandırılmamış olması eksik veri olarak ifade edilir (Akalin, 2018). Eksik verilerin kaynağı araştırmacı veya katılımcı olabilir. Eksik veri sorununun ortaya çıkmasında katılımcının ilgili soru/sorulara cevap vermek istememesi veya unutması, araştırmacının hatasından kaynaklı olarak verilerin eksik kaydetmesinden kaynaklanabilir (Lorcu, 2015; Akalin, 2018).

Eksik veriler üç şekilde kategori edilmektedir. Bunlar, MCAR (Missing Completely at Random - Tamamen Rastgele Kayıp), MAR (Missing at Random -Rastgele Kayıp) ve MNAR (Missing Not at Random -Rastgele Olmayan veya Göz Ardı Edilemeyen Kayıp) olarak nitelendirilir. MCAR eksik verilerin dağılımı bilinmediği durumlarda ortaya çıkar ve eksik verinin en iyi durumudur. MAR eksik verinin deseninin diğer değişkenler kullanılarak tahmin edilebildiği durumken; MNAR eksiklik değişkenin kendisiyle ilgilidir ve bu nedenle göz ardı edilemez (Tabachnick ve Fidell, 2013; Lorcu, 2015).

Eksik verilerin ciddiyeti kayıp veri modeline, kayıp miktarına ve neden kayıp olduğuna bağlı olarak değişmektedir. Kayıp verilerin modeli, kayıp miktardan daha önemlidir. Eğer veri setinde rastgele dağılmış kayıp değerler bulunuyorsa daha az ciddi sorunlara yol açmaktadır. Öte yandan, rastgele olmayan kayıp değerler, ne kadar az olursa olsun ciddidir çünkü sonuçların genellenebilirliğini etkilemektedir. Ayrıca büyük bir veri

setinde rasgele bir düzende yalnızca birkaç veri noktası (örneğin %5 veya daha az) eksikse, eksik verilerden kaynaklanan sorunlar daha az ciddidir ve eksik verinin doldurulması için kullanılacak yöntemler (ortalama değerin kullanımı, regresyon vb.) benzer sonuçlar verecektir (Tabachnick ve Fidell, 2013).

Veri setlerinde eksik veriler farklı yöntemlerle tespit edilebilir. Bunlardan en kolayı, değişkenlerin SPSS programında frekans analizini gerçekleştirmektir (Gürbüz, 2021a). Araştırma kapsamında eksik verilerin tespit edilmesi için öncelikli olarak frekans analizi gerçekleştirilecektir. Eksik verilerin bulunması durumunda; eksik verilerin rasgele dağılıp dağılmadığı Little's MCAR Testi ile analiz edilecektir. Analizler sonucunda eksik verinin dağılımı ve miktarı değerlendirilerek, eksik verilerle ilgili işlemler gerçekleştirilecektir.

5.8.2.3. Uç Değer

Uç değer, veri setine uygun olmadığı düşünülen diğer bir ifade ile veri setinde bulunan değişkenlerin, sahip olmaları gerektikleri veri değerlerinden çok farklı değerler içermesi olarak tanımlanabilir (Lorcu, 2015; Akalın, 2018). Bir uç değerin varlığının dört nedeni vardır. Birincisi hatalı veri girişidir. Verilerin doğru girilip girilmediğini görmek için aykırı değerler dikkatle kontrol edilmelidir. İkincisi, eksik değer göstergelerinin gerçek veri olarak okunması için bilgisayar sözdiziminde eksik değer kodlarının belirtilmemesidir. Üçüncüsü, uç değerin örneklemeyi amaçladığınız popülasyonun bir üyesi olmamasıdır. Böyle bir durum tespit edildiğinde silinmelidir. Dördüncüsü, uç değerin amaçlanan örneklemden olması ancak örneklemin geri kalan kısmından farklı özelliğe sahip olmasıdır. Bu durumda uç değeri muhafaza edilir ancak değişkenlerin değeri dönüştürülerek muhafaza edilir. Veri girişindeki hatalar ve eksik değerlerin belirlenmesi kolaylıkla bulunup giderilebilse de üçüncü ve dördüncü alternatifler arasında, yani silme ile dönüştürülerek saklama arasında karar vermek zordur (Tabachnick ve Fidell, 2013).

Pek çok istatistiksel analiz ortalamadan sapmaların karesine dayanması nedeniyle veri setindeki uç değerler hem normal dağılımdan uzaklaşmaya hem de analiz sonuçlarının hatalı olmasına sebep olabilmektedir. Bu nedenle uç değerlerin dikkatle incelenmesi gerekmektedir (Çokluk vd., 2021; Gürbüz, 2021a).

Uç değerler, tek yönlü ve çok yönlü olarak iki açıdan incelenebilir. Tek yönlü uç değer, değişkenlerin her birinin uç değere sahip olup olmadığının incelenmesidir. Tek yönlü uç değerlerin belirlenmesinde verilerin frekans dağılımı, kutu grafiği ya da histogramlarının elde edilmesi gibi yollar kullanılabilir. Tek yönlü uç değerleri incelemenin diğer bir yolu ise ilgili değişkene ait z değerlerini kontrol etmektir. Standardize z değerleri $\pm 3,29$ üzerinde kalan veriler potansiyel uç değerler olarak kabul edilmektedir (Tabachnick ve Fidell, 2013; Gürbüz, 2021a).

Çok yönlü uç değer, iki ya da daha fazla değişkene ait değerlerin olağan dışı kombinasyonları olarak ifade edilebilir. Sosyal bilimlerde değişkenler ya da maddeler çoklu olarak analiz edildiğinden birlikte analiz edilecek değişkenlerin aykırı değerleri çok yönlü uç değer analizi ile kontrol etmek daha uygun bir yöntemdir (Gürbüz, 2021a). Çok yönlü uç değerleri tespit mahalanobis uzaklığı hesaplanarak yapılabilir. Mahalanobis uzaklığı, bir vakanın geri kalan vakaların ağırlık merkezine olan uzaklığıdır. Burada ağırlık merkezi, tüm değişkenlerin ortalamalarının kesişiminde oluşturulan noktadır (Tabachnick ve Fidell, 2013). Her bir denek için hesaplanan mahalanobis değeri, ki-kare tablo değeriyle karşılaştırılır. Ki-kare değeri, $p=0,001$ manidarlık düzeyi ve “Değişken Sayısı (K)-1” serbestlik derecesi için ki-kare tablosunda yer alan değerdir. Hesaplanan mahalanobis değerinin, ki-kare değerinden büyük olması durumunu uç değer olarak nitelendirilmektedir (Karagöz, 2019; Çokluk vd., 2021; Gürbüz, 2021a).

Araştırma verilerinde uç değerleri belirlemek amacıyla çok yönlü uç değer analizi gerçekleştirilecektir.

5.8.2.4. Normallik Testi

İstatistiksel analizlerde verilerin dağılımı önemlidir (Kalaycı, 2014). Verilerin nasıl dağılım gösterdiğini tespit etmek için gerçekleştirilen normal dağılım testleri, istatistik analizde en çok kullanılan testlerden biridir. Çünkü normallik testi, veri setine hangi analizlerin yapılacağını belirlemektedir (Sürücü vd., 2023). Parametrik testlerin uygulanmasında verilerin tek değişkenli normallik şartının sağlanması gerekirken, çok değişkenli parametrik testlerde çok değişkenli normallik şartının sağlanması gerekmektedir (Kalaycı, 2014). Tek değişkenli normallik, veri setinde bir değişkene ilişkin değerlerin normal dağılım göstermesini ifade etmektedir (Çokluk vd., 2021). Tek

değişkenli normalliği belirlemek amacıyla istatistiksel testler ve grafiksel yöntemler bulunmaktadır. Görsel ve en basit biçimde normallik varsayımını test etmek için histogram, gövde yaprak, kutu, Q-Q ve P-P grafikleri kullanılmaktadır (Kalaycı, 2014). Ayrıca örneklem sayısının 50'den az olduğu durumda Shapiro-Wilk Testi, örneklem sayısının 50'den fazla olduğu durumda Kolmogorav-Smirnov Testi ile tek değişkenli normal dağılım istatistiksel olarak incelenebilir. Ancak likert tipi ölçeklerde ve özellikle büyük örnekleme sahip verilerde bu testler sağlıklı sonuç vermemektedir. Bu sebeple sosyal bilimlerde yapılan araştırmalarda genellikle bu testler yerine değişkenlerin çarpıklık (skewness) ve basıklık (kurtosis) değerlerine göre normal dağılım tespit edilir. Çarpıklık ve basıklık değerlerinin $\pm 1,5$ arasında olması verinin normal dağılıma sahip olduğunu göstermektedir (Sürücü vd., 2023 s.22-23). Çarpıklık ve basıklık değerlerine ilişkin literatürde farklı yorumlar da mevcuttur. George ve Mallery (2010), çarpıklık ve basıklık değerlerinin $\pm 1,0$ aralığında olmasının mükemmel kabul edildiğini ancak $\pm 2,0$ arasında bir değerinde kabul edilebilir olduğunu belirtmektedir. Kline (2015) ise çarpıklık ve basıklık değerlerinin $\pm 1,0$ arasında olması gerektiğini ifade etmiştir.

Çok değişkenli normal dağılım, veri setindeki değerlerin (gözlemler), değişkenlerin tüm kombinasyonları açısından normal dağılım göstermesidir (Çokluk vd., 2021). Çok değişkenli normal dağılımı belirlemek içinde istatistiksel testler ve grafiksel yöntemler kullanılmaktadır (Kalaycı, 2014). Çok değişkenli normalliğin incelenmesi için çok değişkenli çarpıklık ve basıklık katsayıları kullanılabilir. Çok değişkenli normal dağılıma sahip olup olmadığını karar verebilmek için çok değişkenli basıklık (Mardia Katsayısı) ve çok değişkenli basıklığın normalleştirilmiş tahmini olan kritik oran değerine bakılmaktadır. Bu iki değerden kritik oran değeri çok önemlidir. Kritik oran değerinin 1,96'dan küçük olması çok değişkenli normallik varsayımının sağlandığını ifade etmektedir. Ancak literatürde kritik oran değerinin 5'ten küçük olması durumunda dağılımın normal olduğunu kabul edenlerde bulunmaktadır (Bayram, 2016; Karagöz, 2019).

Araştırmada tek değişkenli normallik için değişkenlerin çarpıklık ve basıklık değerleri; çok değişkenli normallik için mardia katsayısı ve kritik oran değeri incelenecektir.

5.8.2.5. Çoklu Doğrusallık ve Tekillik

Çoklu doğrusallık ve tekillik, değişkenler çok yüksek korelasyona sahip olduğunda ortaya çıkan korelasyon matrisiyle ilgili problemlerdir. Çoklu doğrusallık durumunda değişkenler oldukça yüksek korelasyona (0,90 ve üzeri) sahiptir. Tekillik ise değişkenlerden birinin, diğer değişkenlerden iki veya daha fazlasının birleşimi olmasıdır (Tabachnick ve Fidell, 2013). Değişkenler arasında korelasyonun 0,90 ve üzerinde olması çoklu doğrusallık olduğunu göstermektedir. Ayrıca varyans büyütme faktörü (Variance Inflation Factor-VIF) değerinin 5'ten küçük ve tolerans değerinin normal düzeyde ($>0,10$) olması çoklu doğrusallık sorununun olmadığını göstermektedir (Güriş ve Astar, 2015; Aksu vd., 2017). Kline (2015) ise üç veya daha fazla sürekli değişken arasındaki çoklu doğrusallığın (eş doğrusallığın) değişkenler arasında korelasyonun 0,90 ve üzerinde, tolerans değerinin 0,10'dan küçük ve VIF değerinin 10'dan büyük olduğunda ortaya çıktığını ifade etmiştir (Kline, 2015). Çoklu doğrusallık sorunu olup olmadığı gerçekleştirilecek korelasyon analizi, VIF ve tolerans değeri incelenerek belirlenecektir.

5.8.2. Güvenilirlik Analizi

Güvenilirlik, bir ölçme aracı ile aynı koşullarda tekrarlanan ölçümlerden elde edilen ölçüm değerlerinin kararlılığının göstergesidir (Lorcu, 2015). Güvenilirlik, ölçme aracının tutarlılığını ve kararlılığını ifade eder (Kimberlin ve Winterstein, 2008). Aynı koşullar altında tekrarlandığında, ölçme aracının benzer sonuçlar üretip üretmediğini gösterir. Güvenilir bir ölçme aracı, sonuçların istikrarını ve tekrarlanabilirliğini garanti etmektedir (Sürücü ve Maslakçı, 2020).

Ölçüm araçlarının güvenilir olması araştırma sonuçlarının sağlıklı yorumlanması için gerekli ön koşuldur. Sosyal bilimlerde ölçüm araçlarının güvenilirliğini test etmek için farklı yöntemler bulunmakla birlikte en çok kullanılan iç tutarlılık testleridir. İç tutarlılık ölçüm aracı bulunduran maddelerin güvenilirliği ile ilgilidir. Ölçüm aracı içindeki maddelerin tutarlılığını ölçer ve ölçüm aracının belirli bir niteliği veya davranışı ne kadar iyi ölçtüğünü sorgular. İç tutarlılık ölçüm aracını oluşturan her bir maddenin birbirleri ile olan korelasyonuna bağlıdır. İç tutarlılığı test etmek için en popüler yöntem Cronbach's Alpha(α)'dır. Cronbach's Alpha değeri 0 ile 1 arasında değer almaktadır. Cronbach's Alpha değeri 0,60 ile 0,70 arasında olan güvenilirlik ölçümleri, genellikle kabul

edilebilirliğin alt sınırı olarak değerlendirilmektedir (Hair vd., 2019). Bu değerin 1'e yaklaşması ölçüm aracının iç tutarlılığın yüksek olduğunu göstermektedir. Cronbach's Alpha değerinin yorumlanmasına yönelik tablo aşağıda yer almaktadır (Sürücü ve Maslakçı, 2020; Sürücü vd., 2023).

Tablo 5.9. Güvenilirlik Katsayıları

Cronbach's Alpha Değeri	Cronbach's Alpha Değerinin Yorumlanması
$\alpha \geq 0,90$	Ölçeğin iç tutarlılığı yüksektir
$0,90 > \alpha \geq 0,70$	Ölçeğin iç tutarlılığı vardır.
$0,70 > \alpha \geq 0,60$	Ölçeğin iç tutarlılığı kabul edilebilir düzeydedir.
$0,60 > \alpha \geq 0,50$	Ölçeğin iç tutarlılığı zayıftır.
$\alpha < 0,50$	Ölçeğin iç tutarlılığı yoktur.

Kaynak: Sürücü ve Maslakçı (2020); Sürücü vd. (2023)

Araştırmada kapsamında kullanılan ölçüm araçlarının güvenilirlik analizi iç tutarlılığı test etmek için en popüler yöntem olan Cronbach's Alpha ile gerçekleştirilecektir. Ölçeklerin güvenilirliği Tablo 5.9'da belirtilen güvenilirlik katsayıları dikkate alınarak değerlendirilecektir. Güvenirlik analizi gerçekleştirilirken maddelere ilişkin madde toplam korelasyon değeri ve madde çıkarıldığında cronbach's alpha değeri incelenecektir. Madde toplam korelasyon değeri, ölçekteki her bir maddenin ölçeğin toplam puanıyla olan korelasyonunu ifade etmektedir. Bu değer, her bir maddenin ölçeğin genel yapısına ne kadar uygun olduğunu ve ölçeğin iç tutarlılığını nasıl etkilediğini belirlemek için kullanılmaktadır. Yüksek madde toplam korelasyon değerleri, maddenin ölçeğin genel puanıyla iyi bir uyum gösterdiğini ve ölçeğin iç tutarlılığına katkıda bulunduğunu göstermektedir. Madde toplam korelasyon değerlerinin 0,50'nin üzerinde olması beklenmekte olup, kabul edilebilir minimum değer genellikle 0,30 ve üzeri olarak kabul edilmektedir (Eisinga vd., 2013; Hair vd., 2019).

Madde çıkarıldığında Cronbach's alpha değeri, bir maddenin ölçekten çıkarılması durumunda güvenilirlikteki değişimi gösterir (Sürücü vd., 2023). Eğer ölçekte bulunan bazı madde/maddeler çıkarıldığında ölçeğin güvenilirliği yükseliyorsa ilgili madde/maddeler güvenilirliği düşürmektedir ve çıkartılması gerekmektedir (Lorcu, 2015). Madde toplam korelasyonu 0,30'un altında olan veya iç tutarlılığı artıran maddelerin çıkarılması planlanmaktadır.

5.8.3. Geçerlilik Analizi

Geçerlilik, bir ölçüm aracının ölçmek istediği kavramı ne kadar doğru ölçtüğünü ifade etmektedir. Diğer bir ifadeyle bir testin sonuçlarının amaçlanan kullanımlar ve yorumlar için ne kadar uygun olduğudur (Kimberlin ve Winterstein, 2008). Geçerlilik, ölçme aracının ölçmek istediği davranışı veya özelliği ölçüp ölçmediğini ve ne kadar iyi bir şekilde ölçtüğünü belirten bir kavramdır. Araştırmanın sonuçlarının geçerli olabilmesi için kullanılan ölçme aracının doğru şeyi ölçmesi gereklidir. Geçerlilik, verilerin anlamlı ve uygun yorumlanabilir olmasını sağlamaktadır (Sürücü ve Maslakçı, 2020).

Ölçüm araçlarının geçerliliğini tespit etmek amacıyla literatürde farklı geçerlik türleri bulunmaktadır (Sürücü vd., 2023). Bunlar kapsam, ölçüt bağıntılı, eş zaman, yordama, yapı geçerliği vb. şeklinde listelenebilir (Seçer, 2015). Sosyal bilimlerde yapılan çalışmalarda ölçeğin hangi kavram veya özellikleri ölçtüğünün belirlenmesi amacıyla gerçekleştirilen yapı geçerliğinin test edilmesinde en çok faktör analizi kullanılmaktadır (Coşkun vd., 2015; Sürücü vd., 2023). Faktör analizi birbiriyle ilişkili çok sayıda değişkeni bir araya getirerek, kavramsal olarak anlamlı daha az sayıda yeni değişken bulmayı ve keşfetmeyi amaçlayan çok değişkenli bir istatistiksel yöntemdir (Seçer, 2015). Faktör analizi keşfedici (açıklayıcı) faktör analizi (KFA) ve doğrulayıcı faktör analizi (DFA) olmak üzere ikiye ayrılır (Çokluk vd., 2021). KFA araştırmacının değişkenlerin sayısı ve doğası hakkında bilgi sahibi olmak amacıyla gerçekleştirdiği analiz türüdür. Genellikle ölçek geliştirme çalışmalarında kullanılan KFA’da temel amaç yapıya ilişkin çok sayıda değişkenin incelenerek yapıyı açıklayan daha az değişkeni tespit etmek ve yapının açıklama gücünü açıklamaktır. DFA ise daha önceden geçerliliği test edilmiş bir ölçme aracının veya modelin doğruluğunu tespit etmektir. Sürücü vd. (2023), geçerliği önceden test edilmiş ölçeklerle yapılan araştırmalarda ölçeğin veya modelin doğruluğunu test etmek için öncelikle DFA yapılması gerektiğini ve yapılan DFA sonucunda yeterli uyum değerleri sağlanamazsa veya ölçüm aracının yapısı doğrulanamamışsa KFA yapılması gerektiğini belirtmişlerdir. Çokluk vd. (2021), geçerlilik ve güvenilirlik kanıtlarına sahip özgün araçların uyarlama çalışmalarında önceden keşfedilen kuramsal yapıyı tanımlayan modelin, uyarlanacak kültürde toplanan verilerle ne derece uyum gösterdiğini DFA kullanılarak incelenmesi gerektiğini ifade etmişlerdir. Benzer şekilde Seçer (2015), DFA ’nın KFA ile belirlenen örtük yapının

doğrulanması amacıyla kullanıldığını belirtmiştir. Bu nedenle farklı bir dilde geliştirilen bir ölçeğin aracının Türkçe 'ye uyarlama aşamasında ölçeğin orijinal dilinde sahip olduğu faktör yapısına uyumunun incelenmesinde DFA kullanımının en çok başvurulan yöntem olduğunu dile getirmiştir (Seçer, 2015). Çalışmada kullanılan ölçeklerin geçerlilik analizi AMOS (Analysis of Moment Structures) 26 programı aracılığıyla DFA ile gerçekleştirilecektir. AMOS programı yapısal eşitlik modellemesinde (YEM) kullanılan bir programdır (Meydan ve Şeşen, 2015).

YEM, faktör analizi ve regresyon analizinin birleşiminden oluşan, genellikle gözlenen ve örtük (gizil) değişkenlerin nedensel ve ilişkisel bir model içerisinde tanımlanmasına dayanan çok değişkenli istatistiksel yöntemdir (Meydan ve Şeşen, 2015; Hair vd., 2019; Gürbüz, 2021a). YEM gözlenen ve örtük değişkenler arasındaki doğrudan ve dolaylı etkilerin tek bir model içerisinde test edilmesini sağlamaktadır. Bu haliyle YEM ile model içinde aynı anda birden fazla regresyon analizi gerçekleştirilmektedir (Meydan ve Şeşen, 2015). YEM çalışmalarında genellikle DFA (Ölçüm Modeli) ve Yol Analizi (Yapısal Model) yapılmaktadır. DFA analizinde genellikle gözlenen değişkenlerle örtük değişkenler arasındaki ilişkileri test eden ölçüm modelleri test edilir. Burada bir ölçeğin ya da ölçme modelinin öngörülen kurumsal yapısı toplanan veri ile doğrulanmaktadır. Böylece ölçüm modelinin geçerliği test edilmektedir. Yapısal modelde ise örtük ya da gözlenen değişkenler arasındaki etkiler (doğrudan, aracı ve düzenleyici) sınanmaktadır (Gürbüz, 2021a). YEM testleri sınanmaya çalışan modelin, o model için toplanan verilerle ne derecede uygun olduğuna dair değerlendirme ölçütleri sunmaktadır. Bir modelin veri ile uyum ya da uyumsuzluğu test sonucu ortaya çıkan çeşitli uyum indeksleri değerlendirilerek yapılmaktadır (Meydan ve Şeşen, 2015). Tablo 5.10'da uyum indeksleri ve eşik değerleri yer almaktadır. Çalışmada kullanılan ölçeklerin veri ile uyum ya da uyumsuzluğu DFA sonucu ortaya çıkan uyum indeksleri değerlendirilerek gerçekleştirilecektir. DFA sonucunda maddelere ilişkin standardize faktör yükleri 0,50 veya daha yüksek ve ideal olarak 0,70 veya daha yüksek olmalıdır (Hair vd., 2019). Brown (2015) ise DFA sonucunda faktör yüklerinin en az 0,30 olması gerektiğini, ancak ideal olarak 0,50 veya daha yüksek olması gerektiğini belirtmektedir (Brown, 2015). Ayrıca p değerinin anlamlı olup olmadığı ($p < 0,05$) kontrol edilmelidir. Anlamlı olmayan p değeri veya faktör yükünün düşük olduğu durumda ilgili maddelerin analizden çıkarılması gerekmektedir (Gürbüz, 2021a). Collier (2020) faktör yükünün 0,70 eşğine

yakın veya altında olması durumunda AVE değeri hala 0,50 ve üzerinde ise maddenin korunmasını önermiştir (Collier, 2020).

Son yıllarda YEM temelli araştırmalarda ölçeklerin DFA sonuçları raporlanırken birleşim ve ayrışım geçerliklerine ilişkin ilave değerlere de yer verilmektedir. Birleşim (yakınsak) geçerliliği belirli bir yapı veya faktör altında yer alan göstergelerin (ölçüm maddelerinin) birbirleriyle yüksek düzeyde ilişkili olup olmadığını değerlendirmeyi amaçlamaktadır. Birleşim geçerliliği, aynı yapı veya faktörü ölçmesi beklenen değişkenlerin birlikte hareket etme derecesini ifade etmektedir (Yaşlıoğlu,2017; Hair vd., 2019). Birleşim geçerliliği için CR (Bileşik Güvenilirlik- Composite Reliability) ile AVE (Ortalama Açıklanan Varyans-Average Variance Extracted) değerleri kullanılmaktadır. CR, bir faktördeki maddelerin standardize yol katsayıları (faktör yükleri) ile hata varyanslarını dikkate almaktadır. Bu değer 0,70 veya üzerinde olması, göstergelerin yapı ile tutarlı bir şekilde ilişkili olduğunu ve birleşim geçerliğinin sağlandığını göstermektedir. AVE değeri ise, her bir yapı için açıklanan ortalama varyans miktarını temsil etmektedir. Bir yapının AVE değerinin 0,50 veya üzerinde olması, birleşim geçerliliği açısından yeterli kabul edilir. Bu, yapının toplam varyansının en az %50'sinin göstergeleri tarafından açıklandığını göstermektedir (Fornell ve Larcker, 1981). Özetle bir ölçüm modelinde faktörlerin birleşim geçerliğine sahip olması için CR değerinin 0,70 ve üzerinde, AVE değerinin 0,50 ve üzerinde ve CR değerinin AVE değerinden büyük olması ($CR > AVE$) gerekmektedir (Yaşlıoğlu,2017; Gürbüz, 2021a).

Ayrışım geçerliliği ise bir ölçüm modelindeki farklı yapıların (faktörlerin) birbirlerinden ne kadar bağımsız olduğunu belirlemeyi amaçlayan bir geçerlilik türüdür. Ayırt edici geçerlilik, bir yapının veya değişkenin diğer yapılardan veya değişkenlerden gerçekten farklı olma derecesidir (Hair vd., 2019). Henseler vd. (2015), Fornell-Larcker kriteri ve çapraz yükleme değerlendirmesinin ayrışım geçerliliği değerlendirmede düşük duyarlılığa sahip olduğunu vurgulamaktadır. HTMT (Heterotrait-Monotrait) oranı ile ayrışım geçerliği daha güvenilir ve hassas bir şekilde değerlendirmektedir. HTMT değeri, iki farklı yapının birbirleriyle ne kadar ilişkili olduğunu gösterir. Eğer HTMT değeri belirli bir eşik değerden (0,85 veya 0,90) küçükse, ayrışım geçerliği sağlanmış kabul edilmektedir (Henseler vd., 2015). Benzer şekilde Kline (2015), HTMT oranının ayrışım geçerliğini değerlendirmek için etkili bir yöntem olduğunu belirtmektedir.

Tablo 5.10. Uyum İyiliği İndeksleri ve Eşik Değerleri

Uyum İndeksi	Açıklama	İyi Uyum	Kabul Edilebilir Uyum	Kaynak
χ^2 (CMIN)	Modelin genel uyumunun belirlenmesi için kullanılır. Önerilen model ile verinin uyumunu test eder.	$p > 0,05$	-	Hair vd., 2019
χ^2/df	χ^2 değeri örneklem büyüklüğünden etkilendiği için serbestlik derecesine oranı daha güvenilir sonuç vermektedir.	< 2 < 3	$2 \leq \chi^2/df \leq 3$ $2 \leq \chi^2/df \leq 5$ $3 \leq \chi^2/df \leq 5$	Kline, 2015; Hair vd., 2019; Gürbüz, 2021a
GFI	Model uyumunu örneklem büyüklüğünden bağımsız olarak test etmektedir.	$\geq 0,95$	$\geq 0,90$	Hair vd., 2019; Tabachnick ve Fidell, 2013
AGFI	GFI'nın serbestlik derecesine göre ayarlanmış halini ifade etmektedir.	$\geq 0,95$	$\geq 0,90$ $\geq 0,85$ $\geq 0,80$	Hair vd., 2019; Kline, 2015; Tabachnick ve Fidell, 2013
CFI	Serbestlik derecesi ve örneklem büyüklüğü dikkate alınarak test edilen modeli temel modele göre kıyaslama gerçekleştirir.	$\geq 0,95$	$\geq 0,90$	Tabachnick ve Fidell, 2013; Kline, 2015; Hair vd., 2019
NFI	Modelin uyumunu değerlendirmek için kullanılan bir uyum indeksidir. NFI, önerilen modelin uyumunu, bağımsız (null) modelle karşılaştırır ve bağımsız model ile önerilen model arasındaki uyum iyileşmesini ifade etmektedir.	$\geq 0,95$	$\geq 0,90$ $\geq 0,85$ $\geq 0,80$	Hair vd., 2019; Kline, 2015; Tabachnick ve Fidell, 2013
TLI / NNFI	Modelin uyumunu bağımsız modelle karşılaştırır, NFI'nın serbestlik derecelerini dikkate alınarak hesaplanmış halidir.	$\geq 0,95$	$\geq 0,90$ $\geq 0,85$ $\geq 0,80$	Hair vd., 2019; Kline, 2015; Tabachnick ve Fidell, 2013
RMSEA	Modelin veriye ne kadar iyi uyum sağladığını değerlendirirken modelin karmaşıklığını ve serbestlik derecelerini dikkate alır. Bu indeks, modelin yaklaşık hata büyüklüğünü ölçer ve modelin belirli bir örnekleme uyumunu genel popülasyona genelleştirir.	$\leq 0,06$	$\leq 0,08$ $\leq 0,10$	Hair vd., 2019; Tabachnick ve Fidell, 2013 Kline, 2015;
SRMR	Modelin veriye ne kadar iyi uyum sağladığını değerlendirirken gözlemlenen ve tahmin edilen kovaryans matrisleri arasındaki ortalama farkın standardize edilmiş değeridir.	$\leq 0,08$	$\leq 0,10$	Hu ve Bentler, 1999; Hair vd., 2019

Not: χ^2 (CMIN)=Ki-Kare (Chi-Square), χ^2/df =Ki-Kare/Serbestlik Derecesi (Chi-Square/Degrees of Freedom), GFI=İyilik Uyum İndeksi (Goodness Of Fit Index), AGFI=Düzeltilmiş İyilik Uyum İndeksi (Adjustment Goodness Of Fit Index), CFI=Karşılaştırmalı Uyum İndeksi (Comparative Fit Index), NFI=Normlaştırılmış Uyum İndeksi (Normed Fit Index), TLI/NNFI=Normlaştırılmamış Uyum İndeksi (Non-Normed Fit Index), RMSEA=Yaklaşık Hataların Ortalama Karekökü (Root Mean Square Error of Approximation), SRMR=Standardize Edilmiş Ortalama Hataların Karekökü (Standardized Root Mean Square Residual)

5.8.4. Betimleyici Analizler

Betimsel istatistikler, veri setinin temel özelliklerini özetlemek ve tanımlamak için kullanılan istatistiksel yöntemlerdir (Hair vd., 2019). Verilerin dağılımlarının, aritmetik ortalama, standart sapma, çarpıklık ve basıklık katsayıları gibi istatistikler üzerinden incelenmesi, betimsel yöntemler olarak belirtilmektedir (Büyüköztürk, 2016). Veri toplama aracının demografik bilgiler bölümünde yer alan sorularla ilgili frekans ve yüzde betimsel istatistikleri kullanılacaktır. Ayrıca araştırmada kullanılan ölçeklerde yer alan ifadelerin en düşük ve en yüksek değerler, mod (en sık görülen değer), ortalama ve standart sapma değerlerini belirten temel tanımlayıcı istatistikler gerçekleştirilecektir.

Verilerin betimsel istatistikleri IBM SPSS (Statistical Package for the Social Sciences) 22 programı aracılığıyla gerçekleştirilecektir.

5.8.5. Korelasyon Analizi

Korelasyon, iki veya daha fazla değişken arasındaki ilişkinin yönünü ve gücünü belirleyen istatistiksel bir ölçüttür. Korelasyon, değişkenlerin birlikte nasıl hareket ettiğini ve bir değişkenin diğerine olan etkisini anlamamıza yardımcı olmaktadır. Korelasyon katsayısı “r” ile gösterilir, ± 1 arasında bir değer almaktadır. Bu değer, iki değişken arasındaki ilişkinin gücünü ve yönünü göstermektedir (Kline, 2015; Hair vd., 2019). Korelasyon katsayısının artı (+) değer olması değişkenler arasında pozitif yönlü bir korelasyonun olduğunu diğer bir ifadeyle bir değişken yüksek değerler aldıkça diğer değişkenin de yüksek değerler alacağını göstermektedir. Korelasyon katsayısının eksi (-) değer olması değişkenler arasında negatif yönlü bir korelasyonun olduğunu yani bir değişken yüksek değerler aldıkça diğer değişkenin daha düşük değerler alacağını göstermektedir. Korelasyon değerinin sıfır (0) olması ise değişkenler arasında doğrusal bir ilişkinin olmadığını göstermektedir (Sürücü vd., 2023).

Araştırma kapsamında kullanılan ölçeklerin faktörleri arasındaki ilişkiler Pearson korelasyon katsayısı ile belirlenecektir. Pearson korelasyon katsayısı, verilerin sürekli ve normal dağılıma sahip olduğu durumlarda iki değişken arasındaki doğrusal ilişkiyi ölçmek için kullanılmaktadır. Korelasyon katsayısının değerlendirilmesi Tablo 5.11’e göre gerçekleştirilecektir (Büyüköztürk, 2016).

Tablo 5.11. Korelasyon Katsayısının Değerlendirilmesi

Korelasyon Katsayısının Gücü		
Düşük $0 < r \leq 0,3$	Orta $0,3 < r \leq 0,7$	Yüksek $0,7 < r \leq 1,0$

Kaynak: Büyüköztürk (2016)

Not: Korelasyon katsayıları mutlak değer olarak belirtilmiştir.

5.8.6. Hipotez Testleri

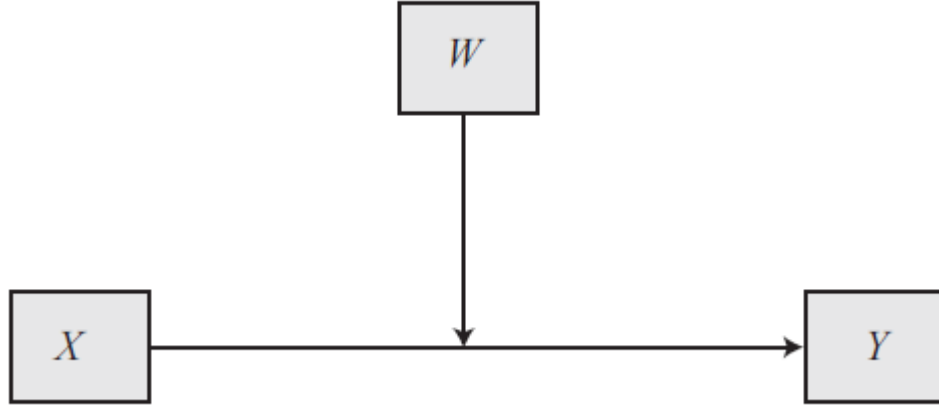
5.8.6.1. Yapısal Regresyon Modeli Analizi

Çoklu regresyon mantığıyla çalışan yapısal model analizi, örtük değişkenler arasındaki doğrudan ve dolaylı etkileri belirlemeye ve test etmeye odaklanmaktadır (Kline, 2015). Yapısal regresyon modeli olarak ta ifade edilen yapısal model analizinde her bir örtük (gizil) değişken kendisini oluşturan gözlenen değişkenlerle birlikte modele dahil edilerek analiz edilmektedir. Değişkenler arasındaki ilişkiler, gözlenen değişkenlerin ölçüm hatalarının modele dahil edilmesinden dolayı klasik regresyon analizine göre daha güçlü ve güvenilir şekilde ortaya konulmaktadır. Ayrıca aynı anda birden fazla bağımlı değişkenin test edilebilmesine de imkân sağlamaktadır (Gürbüz, 2021a). Yapısal regresyon modeli aynı zamanda DFA ve regresyon analizinin birleşimidir (Kline, 2015). Bu nedenle ilk olarak gizil değişkenlerin gözlemlenen değişkenler tarafından nasıl tanımlandığını gösteren ve gözlenen değişkenlerin ölçüm özelliklerini tanımlayan ölçüm modeli test edilir. Bu aşama genellikle DFA kullanılarak değerlendirilir (Hair vd., 2019). Ölçüm modeli doğrulandıktan sonra gizil değişkenler arasındaki nedensel ilişkilerin tanımlandığı yapısal model test edilir (Gürbüz, 2021a). Araştırma kapsamında oluşturulan hipotezlerin test edilmesinde yapısal regresyon modeli analizi AMOS 26 programıyla gerçekleştirilecektir.

5.8.6.2. Düzenleyici Değişken Etkisinin Analizi

Düzenleyici değişken, bağımsız değişkenin bağımlı değişken üzerindeki etkisinin yönünü veya büyüklüğünü değiştiren bir değişkendir (Baron ve Kenny, 1986). Moderatör veya ılımlatıcı değişken olarak ta adlandırılan düzenleyici değişken, bağımsız değişkenin bağımlı değişken üzerindeki etkisinin hangi durumlarda (ne zaman) veya koşullarda değişebileceğini ya da bu etkinin neye bağlı olduğunu gösteren değişkendir

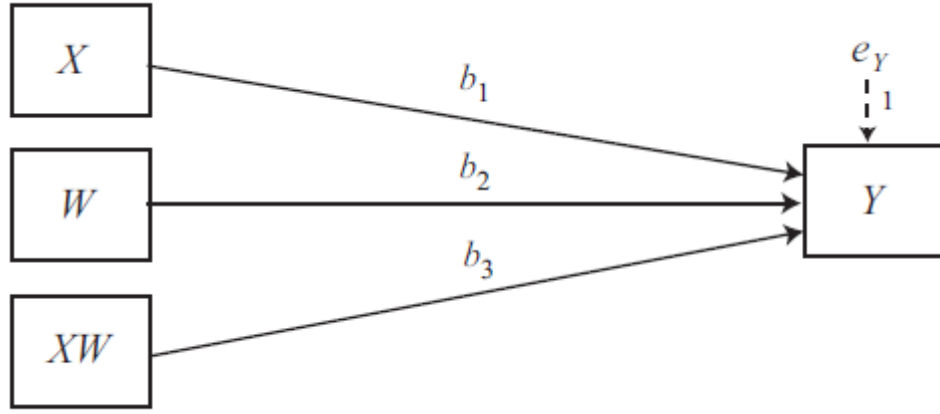
(Gürbüz, 2021b; Hayes, 2022; Sürücü vd., 2023). Düzenleyici değişkenler, bağımsız ve bağımlı değişkenler arasındaki ilişkinin bağlamını belirlemek için kullanılmaktadır. Genellikle bu ilişkinin belirli alt gruplarda (örneğin; erkek-kadın ya da düşük-orta-yüksek) nasıl farklılaştığını anlamak için kullanılır (Gürbüz, 2021b; Hayes, 2022). Düzenleyici değişken sürekli ya da süreksiz olabilmektedir. Basit bir düzenleyici modelin kavramsal gösterimi Şekil 5.2’de verilmiştir. Modelde X bağımsız, Y bağımlı ve W ise düzenleyici değişkeni ifade etmektedir. X ’in Y üzerindeki etkisi, W değişkenine bağlı olarak boyutu, işareti veya gücü değişiyorsa, W değişkeni X ’in Y üzerindeki etkisinin bir düzenleyicisi olarak kabul edilir. Bu durumda, W ’nin X ’in Y üzerindeki etkisini düzenlediği veya W ve X ’in Y üzerindeki etkisinde etkileşimde bulunduğu söylenir (Hayes, 2022).



Şekil 5.2. Düzenleyici Modelin Kavramsal Gösterimi

Kaynak: Hayes (2022)

Şekil 5.2’de kavramsal gösterimi verilen düzenleyici modelin istatistiksel gösterimi Şekil 5.3’te gösterilmektedir. İstatistiksel gösterim incelendiğinde bağımlı değişkeni Y yordayan değişkenler; bağımsız değişken X , düzenleyici değişken W ve bağımsız değişkenle düzenleyici değişkenin çarpımsal etkisiyle oluşturulan $X.W$ etkileşimsel değişkendir. Düzenleyici modele ilişkin regresyon modeli; $Y = i_Y + b_1.X + b_2.W + b_3.X.W + e_Y$ şeklindedir (Hayes, 2022). Modelde düzenleyici etkinin bulunup bulunmadığı etkileşimsel değişkenin anlamlılık durumuna bakılarak anlaşılmaktadır (Sürücü vd., 2023).



Şekil 5.3. Düzenleyici Modelin İstatistiksel Gösterimi

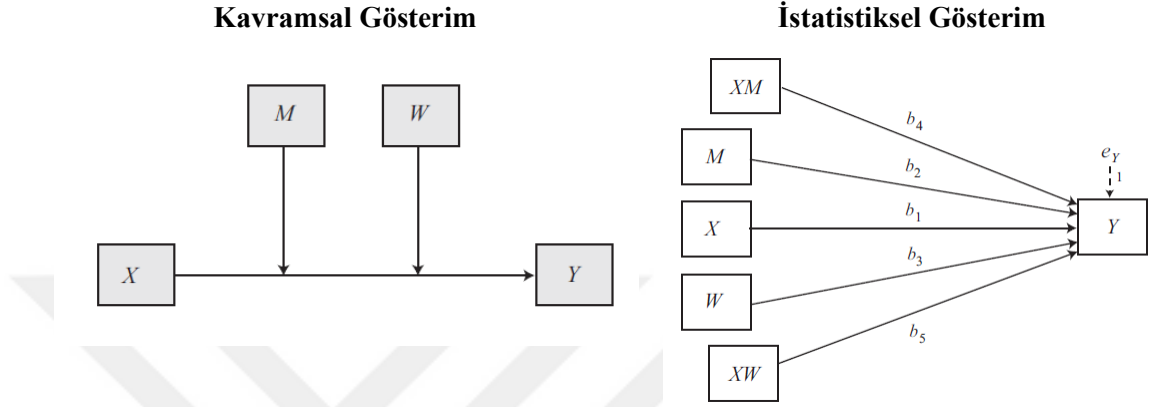
Kaynak: Hayes (2022)

Genel olarak düzenleyici etki analizinde aşağıda belirtilen adımlar uygulanmaktadır.

1. Bağımsız değişken X ve düzenleyici değişkene W ait değerler standardize edilir veya merkezileştirilir. Kategorik değişkenlerde gerek bulunmamaktadır. Buradaki temel amaç çoklu doğrusallık sorununu azaltmaktır. (Process Makro uygulaması bunu otomatik olarak yapmaktadır.)
2. Merkezileştirilmiş veya standardize edilmiş X ve W değişkenleri çarpılarak etkileşimsel değişken X.W oluşturulur. (Process Makro uygulaması bunu otomatik olarak yapmaktadır.)
3. SPSS, AMOS, LISREL vb. programlardan yararlanılarak analiz gerçekleştirilir. Analiz sonucunda etkileşimsel değişkenin bağımlı değişken üzerindeki etkisinin anlamlı olup olmadığı kontrol edilir. Düzenleyici etkinin anlamlı olması durumunda eğim (slope) analizi yapılarak sonuçlar metin ve grafik ile birlikte yorumlanır (Gürbüz, 2021b).

Düzenleyici etki analizleri SPSS, AMOS, LISREL, Mplus vb. birçok program vasıtasıyla yapılabilmektedir. SPSS, SAS ve R için geliştirilen PROCESS makrosu, düzenleyici ve aracılık analizleri için güçlü bir araç sunar ve kullanımı oldukça basittir. PROCESS makro bağımsız değişken, düzenleyici değişken ve bağımlı değişken arasındaki etkileşimlerin analizini kolaylaştırmaktadır (Hayes, 2022).

Araştırma kapsamında düzenleyici etki analizleri IBM SPSS 22 programında Hayes (2022) tarafından geliştirilen Process Makro (4.2) kullanılarak gerçekleştirilecektir. Araştırma modelinde sosyal ve teknik pratikler olmak üzere iki düzenleyici değişken yer aldığı için Model 2 kullanılacaktır. Process Makro Model 2'ye ilişkin kavramsal ve istatistiksel model gösterimi Şekil 5.4'te yer almaktadır.



Şekil 5.4. Process Makro Model 2 Kavramsal ve İstatistiksel Gösterimi
Kaynak: Hayes (2013)

ALTINCI BÖLÜM

BULGULAR

6.1. VERİ SETİNİN KONTROLÜ ve ANALİZE HAZIRLANMASINA İLİŞKİN BULGULAR

6.1.1 Veri Setinin Doğruluğu ve Eksik Veri Analizi

Araştırma kapsamında 464 katılımcıdan elde edilen veriler, öncelikle MS Excel 2016 programında kapsamlı bir ön incelemeye tabi tutulmuştur. Anket formlarındaki tüm ölçek maddelerine aynı yanıtı veren veya eksik kodladığı tespit edilen 53 katılımcıya ait veriler analiz dışı bırakılmıştır. Böylece, 411 katılımcıya ait geçerli veriler, analiz süreci için IBM SPSS 22 programına aktarılmıştır.

Veri setinin doğruluğunu sağlamak amacıyla, orijinal veri dosyası ile IBM SPSS 22 programına aktarılan veri seti karşılaştırılmıştır. Ardından nicel değişkenlerin değerlerinin ölçeğin tanımlı sınırları içinde (7'li Likert) yer alıp almadığı, ortalama ve standart sapma değerlerinin beklenen aralıklarda bulunup bulunmadığı incelenmiştir. Kategorik değişkenler için ise veri girişlerinin tanımlı kategori kodları ile uyumu değerlendirilmiştir (Tabachnick ve Fidell, 2013). Bu kapsamlı inceleme sonucunda veri setinde herhangi bir hata tespit edilmemiştir.

Eksik veri kontrolü amacıyla değişkenlere ilişkin frekans analizi gerçekleştirilmiştir (Gürbüz, 2021a). Yapılan analiz, veri setinde eksik gözlem bulunmadığını ortaya koymuştur.

6.1.2. Uç Değer Analizi

Veri setindeki uç değerlerin tespit etmek amacıyla çok yönlü uç değer analizi mahalanobis uzaklığı hesaplanarak gerçekleştirilmiştir. Her bir katılımcıya ait mahalanobis değeri, Ki-kare tablosunda $p=0,001$ manidarlık düzeyi ve $df=11$ serbest derecesindeki 31,264 değeri ile karşılaştırılmıştır (Tabachnick ve Fidell, 2013). Karşılaştırma sonucunda 3 katılımcıya ait verinin uç değer olduğu tespit edilmiştir. Veri setindeki uç değerler incelenmiştir. Veri setinde uç değerler bulunurken ($n=411$) yapılan

istatistiksel bulgular, uç değerler veri setinden çıkarıldıktan sonra (n=408) yapılan istatistiksel bulgular karşılaştırılmış ve anlamlı bir fark olmadığı görülmüştür. Bu nedenle uç değerler veri setinde korunmuştur.

6.1.3. Normallik Testi

Araştırma kapsamında toplanan verilerinin tek değişkenli normallik varsayımını sağlayıp sağlamadığını belirlemek amacıyla değişkenlerin çarpıklık ve basıklık değerleri incelenmiştir. Tek değişkenli normallik sağlanması durumunda çok değişkenli normallik testi gerçekleştirilmiştir.

Tablo 6.1’de ÇBF ölçeğindeki maddelere ilişkin çarpıklık ve basıklık değerleri yer almaktadır. Ölçekte yer alan maddelerin çarpıklık değerleri -0,242 ile -0,663 arasında; basıklık değerleri ise -0,235 ile -1,231 arasında değiştiği görülmektedir. Çarpıklık ve basıklık değerlerinin $\pm 1,5$ arasında (Tabachnick ve Fidell, 2013; Sürücü vd., 2023) olması verilerin tek değişkenli normal dağılıma sahip olduğunu göstermektedir.

Tablo 6.1. ÇBF Ölçeği Çarpıklık ve Basıklık Değerleri

Faktör	Madde	Çarpıklık	Basıklık
Takım Yeteneği	TY1	-0,242	-1,231
	TY2	-0,508	-0,487
	TY3	-0,528	-0,246
	TY4	-0,416	-0,235
	TY5	-0,663	-0,342
Müşteri Katılımı	MK1	-0,536	-0,449
	MK2	-0,515	-0,580
	MK3	-0,589	-0,475
Proje Tanımlama Süreci	PT1	-0,637	-0,498
	PT2	-0,452	-0,806
	PT3	-0,427	-0,868
Proje Yönetim Süreci	PY1	-0,590	-0,591
	PY2	-0,415	-0,732
	PY3	-0,369	-0,744
	PY4	-0,476	-0,758
	PY5	-0,399	-0,780
	PY6	-0,448	-0,764

Not: Mardia Katsayısı: -10,592 ve C.R.: -4,224

ÇBF ölçeğinde verilerin tek değişkenli normallik varsayımını sağlaması sonucunda çok değişkenli normallik testi gerçekleştirilmiştir. Analiz sonucunda çok değişkenli

basıklık (Mardia Katsayısı) değeri -10,592 ve çok değişkenli basıklığın normalleştirilmiş tahmini olan kritik oran (Critical Ratio- C.R.) değeri ise -4,224 olduğu görülmüştür. Kritik oran mutlak değerinin 5'ten küçük olması nedeniyle verilerin çok değişkenli normal dağılım varsayımını sağladığı sonucuna ulaşılmıştır (Bayram, 2016; Karagöz, 2019).

ÇPK ölçeğinin maddelere ilişkin çarpıklık ve basıklık değerleri Tablo 6.2'de yer almaktadır. Tablo 6.2 incelendiğinde ölçekte yer alan maddelerin çarpıklık değerleri -0,243 ile -1,018 arasında; basıklık değerleri ise 0,190 ile -0,611 arasında değiştiği görülmektedir. Çarpıklık ve basıklık değerlerinin $\pm 1,5$ arasında (Tabachnick ve Fidell, 2013; Sürücü vd., 2023) olması verilerin tek değişkenli normal dağılıma sahip olduğunu göstermektedir.

Tablo 6.2. ÇPK Ölçeği Çarpıklık ve Basıklık Değerleri

Faktör	Madde	Çarpıklık	Basıklık
Yinelemeli Teslimat	SYT1	-1,018	0,190
	SYT2	-0,385	-0,469
	SYT3	-0,384	-0,383
	SYT4	-0,351	-0,574
Ayaküstü Toplantı	SAT1	-0,369	-0,587
	SAT2	-0,322	-0,573
	SAT3	-0,363	-0,453
Retrospektif	SR1	-0,363	-0,422
	SR2	-0,441	-0,341
	SR3	-0,478	-0,230
Birim Test	TBT1	-0,243	-0,611
	TBT2	-0,548	0,030
	TBT3	-0,439	-0,167
Sürekli Entegrasyon	TSE1	-0,528	-0,226
	TSE2	-0,512	-0,024
	TSE3	-0,644	0,054
Kodlama Standartları	TKS1	-0,518	0,088
	TKS2	-0,582	0,095
	TKS3	-0,673	0,076
Yeniden Kod Düzenleme	TYKD1	-0,569	-0,168
	TYKD2	-0,538	-0,088
	TYKD3	-0,559	-0,008

Not: Mardia Katsayısı: 5,714 ve C.R.: 1,782

ÇPK ölçeğinde veriler tek değişkenli normallik varsayımını sağlaması sonucunda çok değişkenli normallik testi gerçekleştirilmiştir. Analiz sonucunda mardia katsayısı 5,714 ve C.R. ise 1,782 olduğu görülmüştür. C.R. değerinin 1,96'dan küçük olması

nedeniyle verilerin çok değişkenli normal dağılım varsayımını sağladığı sonucuna ulaşılmıştır (Bayram, 2016; Karagöz, 2019).

ÇYGPAB ölçeğinin maddelere ilişkin çarpıklık ve basıklık değerleri Tablo 6.3'te yer almaktadır. Tablo 6.3 incelendiğinde ölçekte yer alan maddelerin çarpıklık değerleri -0,749 ile -1,020 arasında; basıklık değerleri ise 0,039 ile 0,437 arasında değiştiği görülmektedir. Çarpıklık ve basıklık değerlerinin $\pm 1,5$ arasında (Tabachnick ve Fidell, 2013; Sürücü vd., 2023) olması verilerin tek değişkenli normal dağılıma sahip olduğunu göstermektedir.

Tablo 6.3. ÇYGPAB Ölçeği Çarpıklık ve Basıklık Değerleri

Faktör	Madde	Çarpıklık	Basıklık
Proje Başarısı	KALITE	-1,020	0,437
	KAPSAM	-0,749	0,039
	ZAMAN	-0,894	0,376
	MALİYET	-0,840	0,107

Not: Mardia Katsayısı: -0,727 ve C.R.: -1,064

ÇYGPAB ölçeğinde verilerin tek değişkenli normallik varsayımını sağlaması sonucunda çok değişkenli normallik testi gerçekleştirilmiştir. Analiz sonucunda mardia katsayısı -0,727 ve C.R. değeri ise -1,064 olduğu görülmüştür. Kritik oran mutlak değerinin 1,96'dan küçük olması nedeniyle verilerin çok değişkenli normal dağılım varsayımını sağladığı sonucuna ulaşılmıştır (Bayram, 2016; Karagöz, 2019)

Sonuç olarak araştırma kapsamında toplanan veriler tek değişkenli ve çok değişkenli normal dağılım göstermektedir. Tüm bu bulgulardan hareketle araştırma kapsamında parametrik ve çok değişkenli parametrik testler uygulanmıştır.

6.1.4. Çoklu Doğrusallık ve Tekillik

Değişkenler arasında çoklu doğrusallık ve teklik durumu bulunup bulunmadığını belirlemek amacıyla korelasyon katsayısı, VIF ve tolerans değerleri incelenerek sonuçlar Tablo 6.4'te sunulmuştur. Değişkenler arasında korelasyon katsayısının 0,90 ve üzerinde, VIF değerinin 5'ten büyük ve tolerans değerinin 0,10'dan düşük olması çoklu doğrusallık ve teklik sorunu göstergesidir (Tabachnick ve Fidell, 2013; Güriş ve Astar, 2015; Kline, 2015).

Tablo 6.4 incelendiğinde değişkenler arasında korelasyon katsayısının 0,90'dan düşük olduğu görülmektedir. Ayrıca değişkenlere ilişkin VIF değerleri 5'ten küçük ve tolerans değerleri 0,10'dan yüksektir. Bu sonuçlar değişkenler arasında çoklu doğrusallık ve tekliklik sorunu olmadığını göstermektedir.

Tablo 6.4. Çoklu Doğrusallık ve Tekillik Test Sonuçları

	TY	MK	PY	PT	YT	AT	R	BT	SE	KS	YKD	PB	Tolerans	VIF
TY	1												,258	3,881
MK	,753**	1											,331	3,018
PY	,790**	,728**	1										,289	3,466
PT	,785**	,753**	,749**	1									,298	3,354
YT	,379**	,356**	,459**	,364**	1								,246	4,073
AT	,321**	,308**	,378**	,308**	,770**	1							,340	2,939
R	,320**	,322**	,411**	,322**	,789**	,722**	1						,307	3,254
BT	,369**	,308**	,429**	,337**	,724**	,677**	,717**	1					,219	4,560
SE	,353**	,312**	,403**	,341**	,741**	,695**	,710**	,826**	1				,247	4,053
KS	,320**	,336**	,405**	,319**	,754**	,701**	,718**	,815**	,774**	1			,235	4,257
YKD	,396**	,334**	,432**	,369**	,754**	,710**	,715**	,794**	,788**	,801**	1		,244	4,100
PB	,602**	,574**	,603**	,593**	,408**	,336**	,389**	,415**	,410**	,425**	,441**	1	,528	1,894

Not: TY=Takım Yeteneği, MK=Müşteri Katılımı, PY=Proje Yönetim Süreci, PT=Proje Tanımlama Süreci, YT=Yinelemeli Teslimat, AT=Ayaküstü Toplantı, R=Retrospektif, BT=Birim Test, SE=Sürekli Entegrasyon, KS=Kodlama Standardı, YKD=Yeniden Kod Düzenleme, PB=Proje Başarısı

** simgesi $p < 0,01$ anlamlılık düzeyini göstermektedir.

6.2 GÜVENİLİRLİK ANALİZİNE İLİŞKİN BULGULAR

Araştırma kapsamında kullanılan ölçüm araçlarının güvenilirlik analizi iç tutarlılığı test etmek için en popüler yöntem olan Cronbach's Alpha ile gerçekleştirilmiştir (Hair vd., 2019). Ölçüm araçlarına ilişkin güvenilirlik sonuçları başlıklar halinde sunulmuştur.

6.2.1. Çevik Başarı Faktörleri Ölçeği Güvenilirlik Analizine İlişkin Bulgular

ÇBF ölçeğinin güvenilirlik analizine ilişkin detaylı bilgiler Tablo 6.5'te yer almaktadır. Ölçek maddelerine ilişkin madde toplam korelasyon değerleri; Takım Yeteneği faktöründe 0,685 ile 0,875 arasında, Müşteri Katılımı faktöründe 0,721 ile 0,804 arasında, Proje Yönetimi faktöründe 0,784 ile 0,812 arasında ve Proje Tanımlama faktöründe 0,750 ile 0,811 arasında değişmektedir. Ölçek genelinde ise madde toplam korelasyon değerleri 0,685 ile 0,812 arasında değişmekte olup, tüm maddelerin 0,50 değerinin üstünde olduğu tespit edilmiştir (Eisinga vd., 2013; Hair vd., 2019). Ayrıca,

herhangi bir madde çıkarıldığında Cronbach's Alpha değerinde önemli bir değişiklik gözlenmemektedir (Sürücü vd., 2023). Bu bulgular, ölçekte yer alan tüm maddelerin ilgili faktörlerle ve ölçeğin geneliyle yüksek uyum gösterdiğini ve ölçeğin iç tutarlılığına katkıda bulunduğunu ortaya koymaktadır.

Ölçeğin faktörlerine ilişkin Cronbach's Alpha (α) değerleri incelendiğinde; Takım Yeteneği faktöründe 0,890, Müşteri Katılımı faktöründe 0,912, Proje Yönetimi faktöründe 0,946 ve Proje Tanımlama faktöründe 0,906 olarak bulunmuş olup, tüm faktörlerin kabul edilebilir alt sınır olan 0,70'in üzerinde olduğu görülmektedir (Hair vd., 2019). Ölçeğin genel Cronbach's Alpha değeri 0,966 olarak hesaplanmış olup, bu durum ölçeğin yüksek derecede güvenilir olduğunu göstermektedir (Sürücü ve Maslakçı, 2020; Sürücü vd., 2023). Bu sonuçlar, Çevik Başarı Faktörleri Ölçeği'nin güvenilir bir ölçüm aracı olduğunu desteklemektedir.

Tablo 6.5. ÇBF Ölçeği Güvenilirlik Analizi Sonuçları

Faktör	Madde	Madde Toplam Korelasyon Değeri	Madde Çıkarıldığında Cronbach's Alpha Değeri	Cronbach's Alpha Değeri
Takım Yeteneği	TK1	0,875	0,962	0,890
	TK2	0,685	0,965	
	TK3	0,709	0,964	
	TK4	0,697	0,965	
	TK5	0,718	0,964	
Müşteri Katılımı	MK1	0,721	0,964	0,912
	MK2	0,795	0,963	
	MK3	0,804	0,963	
Proje Yönetim Süreci	PY1	0,784	0,963	0,946
	PY2	0,798	0,963	
	PY3	0,812	0,963	
	PY4	0,810	0,963	
	PY5	0,805	0,963	
	PY6	0,798	0,963	
Proje Tanımlama Süreci	PT1	0,750	0,964	0,906
	PT2	0,800	0,963	
	PT3	0,811	0,963	

Not: ÇBF Ölçeği Cronbach's Alpha (α)=0,966

6.2.2. Çevik Pratiklerin Kullanımı Ölçeği Güvenilirlik Analizine İlişkin Bulgular

ÇPK ölçeğinin güvenilirlik analizine ilişkin detaylı bilgiler Tablo 6.6'da yer almaktadır.

Tablo 6.6. ÇPK Ölçeği Güvenilirlik Analizi Sonuçları

Faktör	Madde	Madde Toplam Korelasyon Değeri	Madde Çıkarıldığında Cronbach's Alpha Değeri	Cronbach's Alpha Değeri
Yinelemeli Teslimat	SYT1	0,822	0,949	0,857
	SYT2	0,650	0,951	
	SYT3	0,631	0,952	
	SYT4	0,623	0,952	
Ayaküstü Toplantı	SAT1	0,626	0,952	0,791
	SAT2	0,582	0,952	
	SAT3	0,626	0,952	
Retrospektif	SR1	0,616	0,952	0,793
	SR2	0,625	0,952	
	SR3	0,640	0,952	
Birim Test	TBT1	0,609	0,952	0,856
	TBT2	0,656	0,951	
	TBT3	0,704	0,951	
Sürekli Entegrasyon	TSE1	0,720	0,951	0,864
	TSE2	0,711	0,951	
	TSE3	0,750	0,950	
Kodlama Standartları	TKS1	0,706	0,951	0,863
	TKS2	0,711	0,951	
	TKS3	0,725	0,951	
Yeniden Kod Düzenleme	TYKD1	0,722	0,951	0,860
	TYKD2	0,722	0,951	
	TYKD3	0,758	0,950	

Not: Çevik Pratiklerin Kullanımı Ölçeği Cronbach's Alpha (α)=0,953

Ölçek maddelerine ilişkin madde toplam korelasyon değerleri incelendiğinde sosyal faktörler olan; yinelemeli teslimat faktöründe 0,623 ile 0,822 arasında, ayaküstü toplantı faktöründe 0,582 ile 0,626 arasında ve retrospektif faktöründe 0,616 ile 0,640 arasında değişmektedir. Teknik faktörler olan; birim test faktöründe 0,609 ile 0,704 arasında, sürekli entegrasyon faktöründe 0,711 ile 0,750 arasında, kodlama standartları faktöründe 0,706 ile 0,725 arasında ve yeniden kod düzenleme faktöründe 0,722 ile 0,758 arasında değişmektedir. Ölçek genelinde ise 0,582 ile 0,822 arasında değişmekte olup, tüm maddelerin madde toplam korelasyon değerleri 0,50 değerinin üstünde olduğu tespit edilmiştir (Eisinga vd., 2013; Hair vd., 2019). Ayrıca, herhangi bir madde çıkarıldığında Cronbach's Alpha değerinde önemli bir değişiklik gözlenmemektedir (Sürücü vd., 2023).

Bu bulgular, ölçeğe yer alan tüm maddelerin ilgili faktörlerle ve ölçeğin geneliyle yüksek uyum gösterdiğini ve ölçeğin iç tutarlılığına katkıda bulunduğunu ortaya koymaktadır.

Ölçeğin faktörlerine ilişkin Cronbach's Alpha (α) değerleri incelendiğinde sosyal faktörler olan; yinelemeli teslimat faktöründe 0,857, ayaküstü toplantı faktöründe 0,791 ve retrospektif faktöründe 0,793 olarak bulunmuştur. Teknik faktörlerde ise Cronbach's Alpha (α) değerleri; birim test faktöründe 0,856, sürekli entegrasyon faktöründe 0,864, kodlama standartları faktöründe 0,863 ve yeniden kod düzenleme faktöründe 0,860 olup, tüm faktörlerin kabul edilebilir alt sınır olan 0,70'in üzerinde olduğu görülmektedir (Hair vd., 2019). Ölçeğin genel Cronbach's Alpha değeri 0,953 olarak hesaplanmış olup, bu durum ölçeğin yüksek derecede güvenilir olduğunu göstermektedir (Sürücü ve Maslakçı, 2020; sürücü vd., 2023). Bu sonuçlar, ÇPK ölçeğinin güvenilir bir ölçüm aracı olduğunu desteklemektedir.

6.2.3. Çevik Yazılım Geliştirme Projesinde Algılanan Başarı Ölçeği Güvenilirlik Analizine İlişkin Bulgular

ÇYGPAÖ ölçeğinin güvenilirlik analizine ilişkin detaylı bilgiler Tablo 6.7'de sunulmaktadır. Ölçek maddelerine ilişkin madde toplam korelasyon değerleri incelendiğinde, bu değerlerin 0,749 ile 0,870 arasında değiştiği ve tüm maddelerin madde toplam korelasyon değerlerinin 0,50'nin üzerinde olduğu tespit edilmiştir (Eisinga vd., 2013; Hair vd., 2019). Ayrıca, herhangi bir madde çıkarıldığında Cronbach's Alpha değerinde önemli bir değişiklik gözlenmemektedir (Sürücü vd., 2023). Bu bulgular, ölçeğe yer alan tüm maddelerin ölçeğin geneliyle yüksek uyum gösterdiğini ve ölçeğin iç tutarlılığına katkıda bulunduğunu ortaya koymaktadır.

Tablo 6.7. ÇYGPAÖ Ölçeği Güvenilirlik Analizi Sonuçları

Faktör	Madde	Madde Toplam Korelasyon Değeri	Madde Çıkarıldığında Cronbach's Alpha Değeri	Cronbach's Alpha Değeri
Proje Başarısı	KALITE	0,870	0,853	0,908
	KAPSAM	0,751	0,896	
	ZAMAN	0,804	0,877	
	MALİYET	0,749	0,896	

Ölçeğin Cronbach's Alpha değeri 0,908 olarak hesaplanmış olup, bu durum ölçeğin yüksek derecede güvenilir olduğunu göstermektedir (Sürücü ve Maslakçı, 2020; Sürücü vd., 2023). Bu sonuçlar, ölçeğin güvenilir bir ölçüm aracı olduğunu desteklemektedir.

6.3. GEÇERLİLİK ANALİZİNE İLİŞKİN BULGULAR

Araştırma kapsamında kullanılan ölçüm araçlarının yapı geçerliğini test etmek için DFA gerçekleştirilmiştir (Hair vd., 2019). AMOS 26 programı aracılığıyla gerçekleştirilen analizlerde verilerin normal dağılım göstermesi (tek değişkenli ve çok değişkenli normal dağılım) nedeniyle en yaygın ve çoğu YEM programında varsayılan olarak seçili durumda bulunan maksimum olabilirlik (Maximum Likelihood- ML) hesaplama yöntemi kullanılmıştır (Gürbüz, 2021a). Ölçüm araçlarına ilişkin geçerlilik sonuçları başlıklar halinde sunulmuştur.

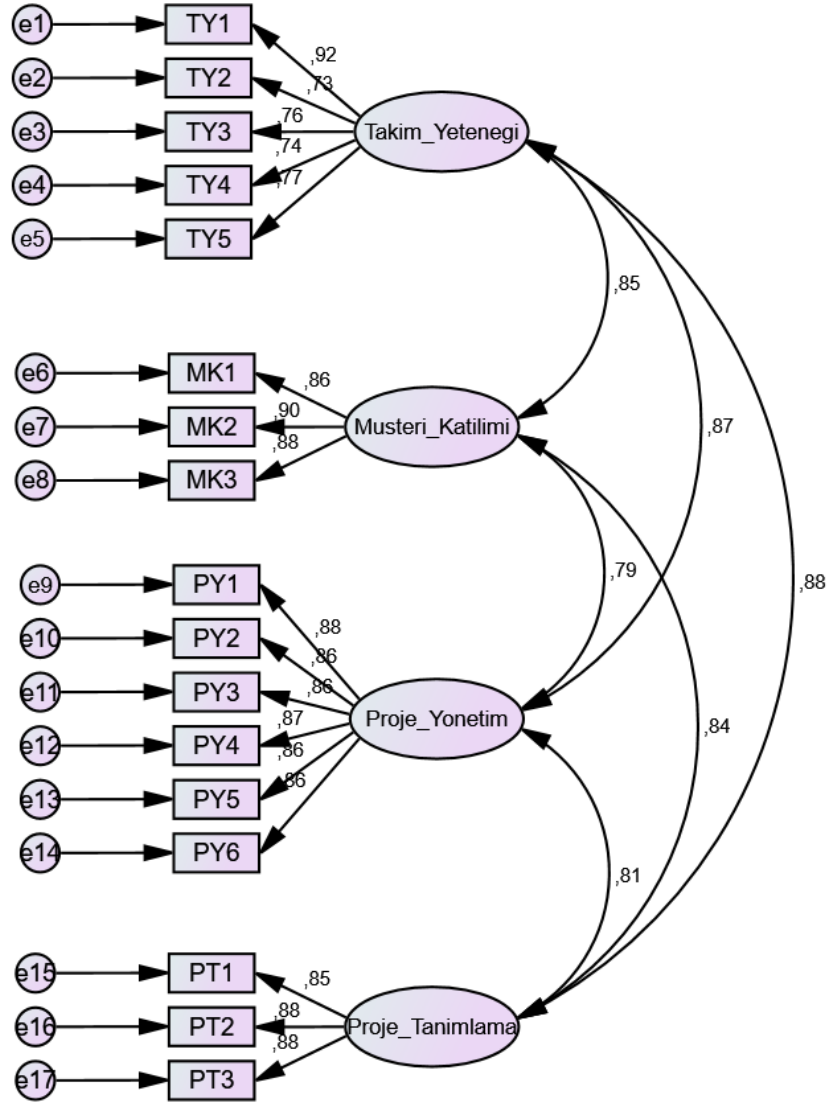
6.3.1. Çevik Başarı Faktörleri Ölçeği Geçerlilik Analizine İlişkin Bulgular

ÇBF ölçeğinin 4 faktörlü ve 17 maddeden oluşan birinci düzey çok faktörlü DFA modeli Şekil 6.1'de yer almaktadır. DFA sonucu elde edilen uyum iyiliği değerleri önerilen 4 faktörlü modelin veri ile uyumlu ve kabul edilebilir olduğunu göstermektedir. Modele ilişkin uyum iyiliği değerleri Tablo 6.8'de yer almaktadır. Ölçeğin genel anlamda veri ile iyi uyum gösterdiği görülmektedir.

Tablo 6.8. ÇBF Ölçeği DFA Uyum İyiliği Değerleri

Model	χ^2	df	χ^2/df	GFI	AGFI	CFI	NFI	TLI	RMSEA	SRMR
Birinci Düzey Çok Faktörlü	182,015	113	1,611*	0,952*	0,935**	0,989*	0,972*	0,987*	0,039*	0,0248*

Not: * simgesi "iyi uyum", ** simgesi "kabul edilebilir uyum" değerlerini ifade etmektedir.



CMIN=182,015; CMIN/DF=1,611; GFI=,952; AGFI=,935; CFI=,989; NFI=,972; TLI=,987; RMSEA=,039; SRMR=,0248

Şekil 6.1. ÇBF Ölçeği Birinci Düzey Çok Faktörlü DFA Modeli

DFA sonucunda maddelere ilişkin standardize faktör yüklerinin 0,50 veya daha yüksek ve ideal olarak 0,70 veya daha yüksek olması gerekmektedir. Ayrıca maddelere ilişkin p değerleri anlamlı olmalıdır (Gürbüz, 2021a; Hair vd., 2019). Ölçek maddelerine ilişkin standardize faktör yükleri ve p değerleri Tablo 6.9’da bulunmaktadır. Tablo 6.9 incelendiğinde ölçek maddelerine ilişkin standardize faktör yükleri 0,728 ile 0,923 arasında yer almaktadır, diğer bir ifadeyle ideal düzeydedir (Collier, 2020). Maddelere ilişkin p değerleri ise anlamlıdır. Bu sonuçlar ölçme aracında bulunan maddelerin ilgili faktörleri güçlü ve anlamlı bir şekilde temsil ettiğini göstermektedir.

Tablo 6.9. ÇBF Ölçeği DFA Regresyon Katsayıları

Faktör	Madde	β	b	SH	C.R.	p
Takım Yeteneği	TY1	0,923	1,188	0,057	20,731	***
	TY2	0,728	0,890	0,057	15,540	***
	TY3	0,758	0,928	0,057	16,312	***
	TY4	0,737	0,832	0,053	15,761	***
	TY5	0,768	1,000			
Müşteri Katılımı	MK1	0,863	1,014	0,043	23,673	***
	MK2	0,897	1,021	0,040	25,403	***
	MK3	0,878	1,000			
Proje Yönetim Süreci	PY1	0,876	1,075	0,044	24,209	***
	PY2	0,857	0,956	0,041	23,240	***
	PY3	0,856	0,970	0,042	23,157	***
	PY4	0,872	1,039	0,043	24,002	***
	PY5	0,858	0,974	0,042	23,298	***
	PY6	0,860	1,000			
Proje Tanımlama Süreci	PT1	0,852	0,978	0,042	23,333	***
	PT2	0,880	0,982	0,040	24,861	***
	PT3	0,884	1,000			

Not: β =Standardize edilmiş regresyon katsayısı, b=Standardize edilmemiş regresyon katsayısı, SH=Standart Hata, C.R.= Critical Ratio (Kritik Oran- t değeri), ***p<0,001

ÇBF ölçeğine ait güvenilirlik, birleşim geçerliliği ve ayrışım geçerliliği sonuçları Tablo 6.10'da yer almaktadır. Ölçekte yer alan faktörlerin bileşik güvenilirlik (CR) değerleri 0,70'in üzerinde olup, bu sonuçlar her bir faktörün yüksek iç tutarlılığa ve güvenilirliğe sahip olduğunu göstermektedir (Yaşlıoğlu, 2017; Gürbüz, 2021a; Sürücü vd., 2023). Faktörlere ait AVE değerlerinin 0,50'nin üzerinde ve CR değerlerinden düşük olması, faktörlerin birleşim geçerliliği kriterini sağladığını ortaya koymaktadır (Yaşlıoğlu, 2017; Collier, 2020; Gürbüz, 2021a). Ayrıca, faktörlere ait HTMT değerlerinin 0,90'nın altında olması, ayrışım geçerliliğinin bulunduğunu ve yapıların birbirinden farklı yapılar olduğunu doğrulamaktadır (Henseler vd., 2015; Kline, 2015; Collier, 2020). Bu bulgular, ÇBF ölçeğinin güvenilir, birleşim geçerliliğine ve ayrışım geçerliliğine sahip olduğunu göstermekte ve ölçek maddelerinin ilgili faktörleri tutarlı ve farklı yapılar olarak başarılı bir şekilde ölçtüğünü göstermektedir.

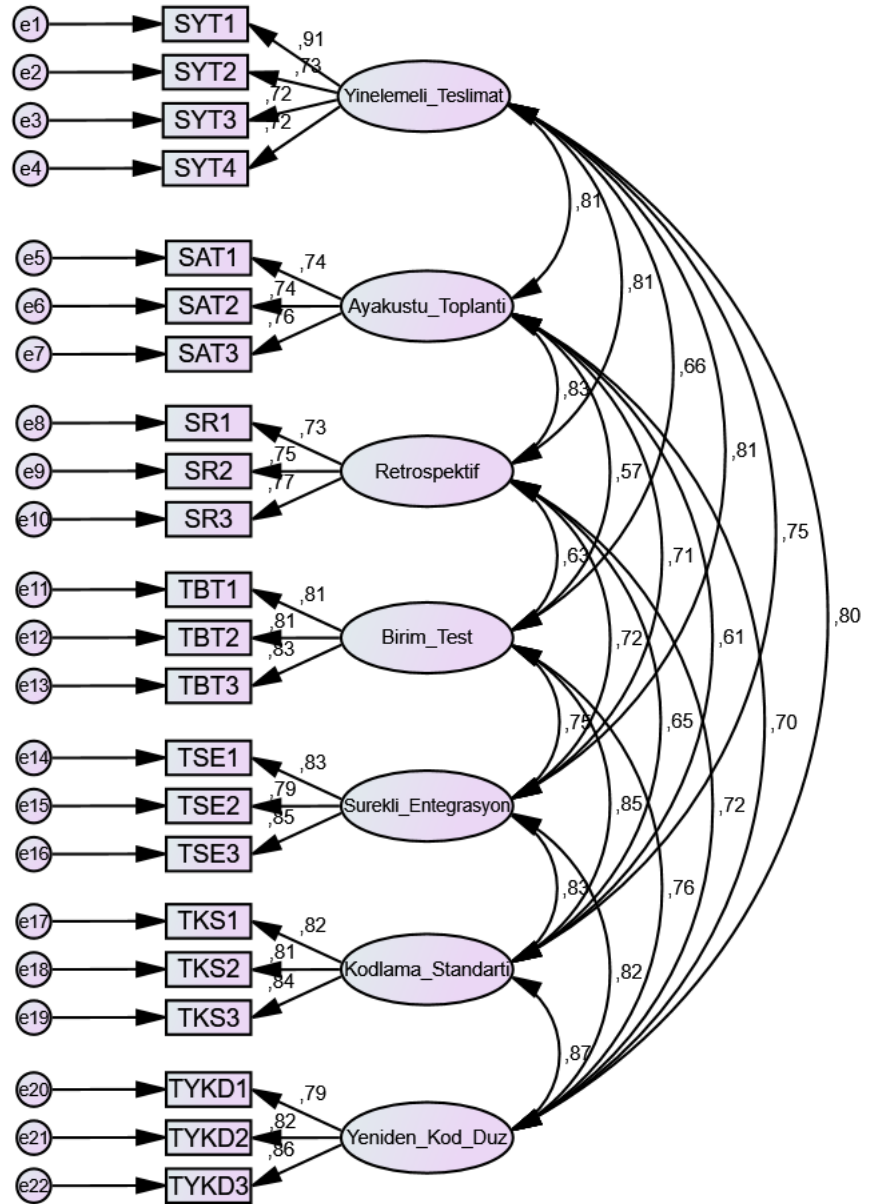
Tablo 6.10. ÇBF Ölçeği Güvenilirlik, Birleşim ve Ayrışım Geçerliliği

Faktörler	CR	AVE	HTMT			
			1	2	3	4
1- Takım Yeteneği	0,889	0,618	-			
2- Müşteri Katılımı	0,911	0,774	0,836	-		
3- Proje Yönetim Süreci	0,946	0,745	0,860	0,785	-	
4- Proje Tanımlama Süreci	0,905	0,760	0,874	0,830	0,811	-

Not: CR=Bileşik Güvenilirlik, AVE=Açıklanan Ortalama Varyans, HTMT= Heterotrait-Monotrait oranı

6.3.2. Çevik Pratiklerin Kullanımı Ölçeği Geçerlilik Analizine İlişkin Bulgular

ÇPK ölçeğinin 7 faktörlü ve 22 maddeden oluşan birinci düzey çok faktörlü DFA modeli Şekil 6.2’de yer almaktadır. DFA sonucu elde edilen uyum iyiliği değerleri önerilen 7 faktörlü modelin veri ile uyumlu ve kabul edilebilir olduğunu göstermektedir. Modele ilişkin uyum iyiliği değerleri Tablo 6.11’de yer almaktadır. Ölçeğin genel anlamda veri ile iyi uyum gösterdiği görülmektedir.



CMIN=283,159;CMIN/DF=1,506; GFI=,941; AGFI=,921; CFI=,984; NFI=,953; TLI=,980;RMSEA=,035; SRMR=0,0285

Şekil 6.2. ÇPK Ölçeği Birinci Düzey Çok Faktörlü DFA Modeli

Tablo 6.11. ÇPK Ölçeği Birinci Düzey DFA Uyum İyiliği Değerleri

Model	χ^2	df	χ^2/df	GFI	AGFI	CFI	NFI	TLI	RMSEA	SRMR
Birinci Düzey Çok Faktörlü	283,159	188	1,506*	0,941**	0,921**	0,984*	0,953*	0,980*	0,035*	0,0285*

Not: * simgesi “iyi uyum”, ** simgesi “kabul edilebilir uyum” değerlerini ifade etmektedir.

DFA sonucunda maddelere ilişkin standardize faktör yüklerinin 0,50 veya daha yüksek ve ideal olarak 0,70 veya daha yüksek olması gerekmektedir. Ayrıca maddelere ilişkin p değerleri anlamlı olmalıdır (Gürbüz, 2021a; Hair vd., 2019; Collier, 2020). Ölçek maddelerine ilişkin standardize faktör yükleri ve p değerleri Tablo 6.12’de bulunmaktadır. Tablo 6.12 incelendiğinde ölçek maddelerine ilişkin standardize faktör yükleri 0,717 ile 0,909 arasında yer almaktadır, diğer bir ifadeyle ideal düzeydedir. Maddelere ilişkin p değerleri ise anlamlıdır. Bu sonuçlar ölçme aracında bulunan maddelerin ilgili faktörleri güçlü ve anlamlı bir şekilde temsil ettiğini göstermektedir.

Tablo 6.12. ÇPK Ölçeği Birinci Düzey DFA Regresyon Katsayıları

Faktör	Madde	β	b	SH	C.R.	p
Yinelemeli Teslimat	SYT1	0,909	1,329	0,076	17,491	***
	SYT2	0,728	1,007	0,071	14,188	***
	SYT3	0,721	1,050	0,075	14,042	***
	SYT4	0,717	1,000			
Ayaküstü Toplantı	SAT1	0,739	0,995	0,069	14,365	***
	SAT2	0,738	1,021	0,071	14,332	***
	SAT3	0,764	1,000			
Retrospektif	SR1	0,729	0,895	0,063	14,272	***
	SR2	0,750	0,979	0,067	14,690	***
	SR3	0,767	1,000			
Birim Test	TBT1	0,807	1,068	0,059	18,119	***
	TBT2	0,813	1,033	0,057	18,281	***
	TBT3	0,826	1,000			
Sürekli Entegrasyon	TSE1	0,829	0,962	0,048	20,239	***
	TSE2	0,791	0,920	0,049	18,866	***
	TSE3	0,854	1,000			
Kodlama Standartları	TKS1	0,821	0,885	0,045	19,751	***
	TKS2	0,811	0,882	0,046	19,385	***
	TKS3	0,840	1,000			
Yeniden Kod Düzenleme	TYKD1	0,787	0,942	0,050	18,894	***
	TYKD2	0,823	0,957	0,047	20,266	***
	TYKD3	0,856	1,000			

Not: β =Standardize edilmiş regresyon katsayısı, b=Standardize edilmemiş regresyon katsayısı, SH=Standart Hata, C.R.= Critical Ratio (Kritik Oran- t değeri), ***p<0,001

ÇPK ölçeğine ait güvenilirlik, birleşim geçerliliği ve ayrışım geçerliliği sonuçları Tablo 6.13'te yer almaktadır. Ölçekte yer alan faktörlerin CR değerleri 0,70'ın üzerinde olup, bu sonuçlar her bir faktörün yüksek iç tutarlılığa ve güvenilirliğe sahip olduğunu göstermektedir (Yaşlıoğlu, 2017; Gürbüz, 2021a; Sürücü vd., 2023). Faktörlere ait AVE değerlerinin 0,50'nin üzerinde ve CR değerlerinden düşük olması, faktörlerin birleşim geçerliliği kriterini sağladığını ortaya koymaktadır (Yaşlıoğlu, 2017; Gürbüz, 2021a; Collier, 2020). Ayrıca, HTMT değerlerinin 0,90'nın altında olması, ayrışım geçerliliğinin bulunduğunu ve yapıların birbirinden farklı yapılar olduğunu doğrulamaktadır (Henseler vd., 2015; Kline, 2015; Collier, 2020). Bu bulgular, Çevik Pratiklerin Kullanımı Ölçeği'nin güvenilir, birleşim geçerliliğine ve ayrışım geçerliliğine sahip olduğunu göstermekte ve ölçek maddelerinin ilgili faktörleri tutarlı ve farklı yapılar olarak başarılı bir şekilde ölçtüğünü doğrulamaktadır.

Tablo 6.13. ÇPK Ölçeği Güvenilirlik, Birleşim ve Ayrışım Geçerliliği

Faktörler	CR	AVE	HTMT						
			1	2	3	4	5	6	7
1- Yinelemeli Teslimat	0,854	0,598	-						
2- Ayaküstü Toplantı	0,791	0,558	0,795	-					
3- Retrospektif	0,793	0,561	0,801	0,835	-				
4- Birim Test	0,856	0,665	0,630	0,573	0,628	-			
5- Sürekli Entegrasyon	0,865	0,681	0,786	0,713	0,717	0,748	-		
6- Kodlama Standartları	0,864	0,679	0,732	0,614	0,647	0,842	0,828	-	
7- Yeniden Kod Düzenle	0,862	0,677	0,790	0,704	0,718	0,773	0,830	0,880	-

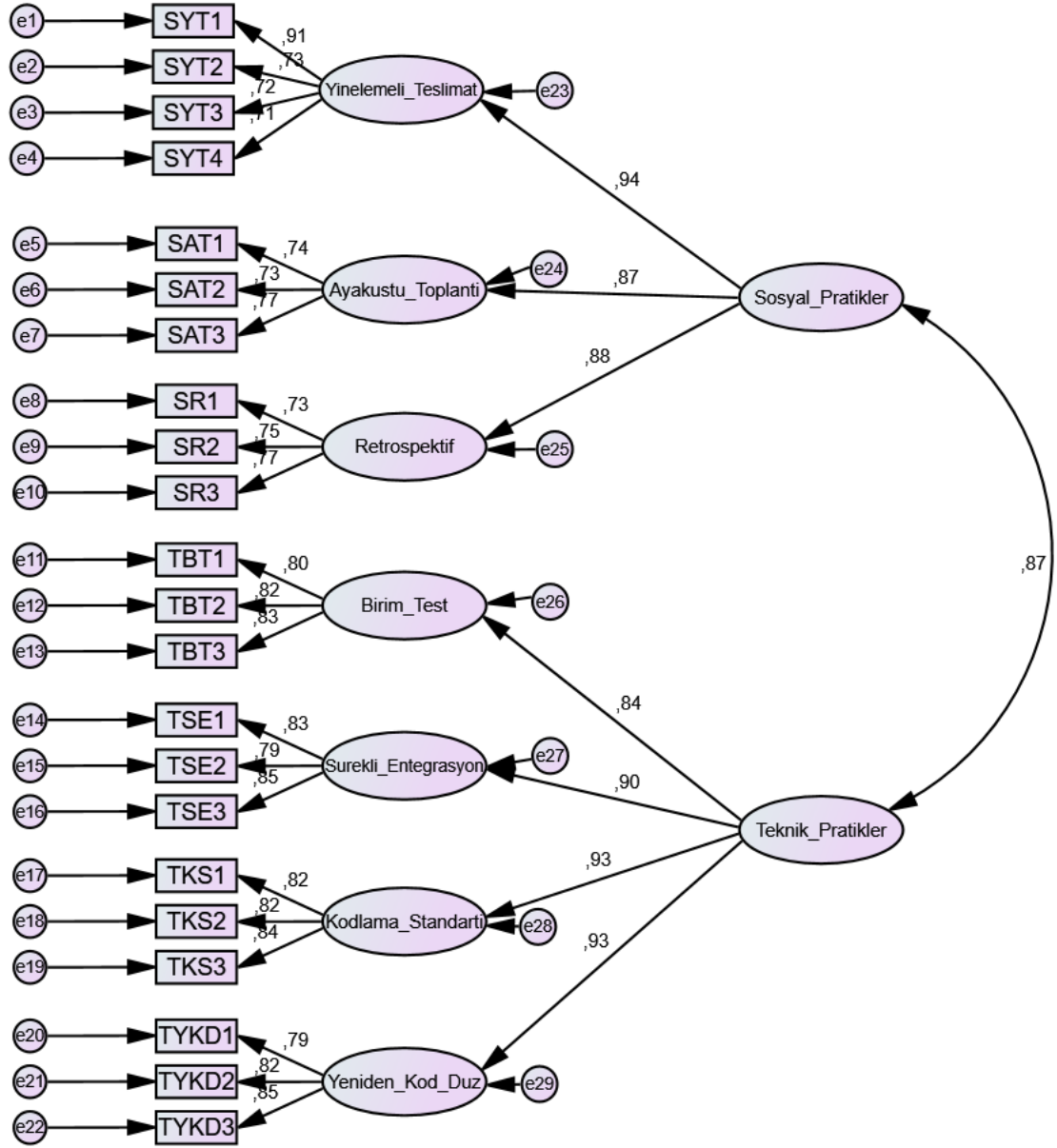
Not: CR=Bileşik Güvenilirlik, AVE=Açıklanan Ortalama Varyans, HTMT= Heterotrait-Monotrait oranı

ÇPK ölçeğinin ikinci düzey çok faktörlü DFA modeli Şekil 6.3'te yer almaktadır. DFA sonucu elde edilen uyum iyiliği değerleri önerilen modelin veri ile uyumlu ve kabul edilebilir olduğunu göstermektedir. Modele ilişkin uyum iyiliği değerleri Tablo 6.14'de yer almaktadır. Ölçeğin genel anlamda veri ile iyi uyum gösterdiği görülmektedir.

Tablo 6.14. ÇPK Ölçeği İkinci Düzey DFA Uyum İyiliği Değerleri

Model	χ^2	df	χ^2/df	GFI	AGFI	CFI	NFI	TLI	RMSEA	SRMR
İkinci Düzey Çok Faktörlü	338,404	201	1,684*	0,929**	0,911**	0,974*	0,944**	0,973*	0,041*	0,0359*

Not: * simgesi "iyi uyum", ** simgesi "kabul edilebilir uyum" değerlerini ifade etmektedir.



CMIN=338,404; CMIN/DF=1,684; GFI=.929; AGFI=.911; CFI=.976; NFI=.944; TLI=.973; RMSEA=.041; SRMR=0,0359

Şekil 6.3. ÇPK Ölçeği İkinci Düzey Çok Faktörlü DFA Modeli

DFA sonucunda maddelere ilişkin standardize faktör yüklerinin 0,50 veya daha yüksek ve ideal olarak 0,70 veya daha yüksek olması gerekmektedir. Ayrıca maddelere ilişkin p değerleri anlamlı olmalıdır (Gürbüz, 2021a; Hair vd., 2019; Collier, 2020). Ölçek maddelerine ilişkin standardize faktör yükleri ve p değerleri Tablo 6.15'te bulunmaktadır. Tablo 6.15 incelendiğinde ölçek maddelerine ilişkin standardize faktör yükleri 0,714 ile 0,941 arasında yer almaktadır, diğer bir ifadeyle ideal düzeydedir. Maddelere ilişkin p

değerleri ise anlamlıdır. Bu sonuçlar ölçme aracında bulunan maddelerin ilgili faktörleri güçlü ve anlamlı bir şekilde temsil ettiğini göstermektedir.

Tablo 6.15. ÇPK Ölçeği İkinci Düzey DFA Regresyon Katsayıları

Faktör	Madde	β	b	SH	C.R.	p
Sosyal Pratikler	SYT	0,941	0,944	0,076	12,435	***
	SAT	0,870	0,935	0,075	12,409	***
	SR	0,879	1,000			
Teknik Pratikler	TBT	0,845	0,865	0,056	15,356	***
	TSE	0,902	0,976	0,057	17,172	***
	TKS	0,931	1,052	0,061	17,329	***
	TYKD	0,930	1,000			
Yinelemeli Teslimat	SYT1	0,908	1,332	0,077	17,344	***
	SYT2	0,730	1,012	0,072	14,149	***
	SYT3	0,724	1,057	0,075	14,030	***
	SYT4	0,714	1,000			
Ayaküstü Toplantı	SAT1	0,741	0,992	0,069	14,328	***
	SAT2	0,732	1,008	0,071	14,152	***
	SAT3	0,768	1,000			
Retrospektif	SR1	0,731	0,897	0,063	14,211	***
	SR2	0,748	0,977	0,067	14,551	***
	SR3	0,767	1,000			
Birim Test	TBT1	0,804	1,064	0,059	17,937	***
	TBT2	0,816	1,037	0,057	18,258	***
	TBT3	0,826	1,000			
Sürekli Entegrasyon	TSE1	0,827	0,958	0,048	20,078	***
	TSE2	0,793	0,922	0,049	18,890	***
	TSE3	0,855	1,000			
Kodlama Standartları	TKS1	0,819	0,886	0,046	19,426	***
	TKS2	0,816	0,891	0,046	19,326	***
	TKS3	0,837	1,000			
Yeniden Kod Düzenleme	TYKD1	0,789	0,948	0,050	18,889	***
	TYKD2	0,823	0,960	0,048	20,144	***
	TYKD3	0,854	1,000			

Not: β =Standardize edilmiş regresyon katsayısı, b=Standardize edilmemiş regresyon katsayısı, SH=Standart Hata, C.R.= Critical Ratio (Kritik Oran- t değeri), ***p<0,001

ÇPK ölçeğinin ikinci düzey DFA analizi sonucuna ait güvenilirlik, birleşim geçerliliği ve ayrışım geçerliliği sonuçları Tablo 6.16'da yer almaktadır. Ölçekte üst faktörler olarak tanımlanan sosyal pratikler ve teknik pratikler faktörlerinin CR değerleri 0,70'in üzerinde olup, bu sonuçlar her iki faktörün yüksek iç tutarlılığa ve güvenilirliğe sahip olduğunu göstermektedir (Yaşlıoğlu, 2017; Gürbüz, 2021a; Sürücü vd., 2023). Faktörlere ait AVE değerlerinin 0,50'nin üzerinde ve CR değerlerinden düşük olması,

faktörlerin birleşim geçerliliği kriterini sağladığını ortaya koymaktadır (Yaşlıoğlu, 2017; Gürbüz, 2021a; Collier, 2020). Ayrıca, HTMT değerlerinin 0,85'nin altında olması, ayrışım geçerliliğinin bulunduğunu ve yapıların birbirinden farklı yapılar olduğunu doğrulamaktadır (Henseler vd., 2015; Kline, 2015; Collier, 2020). Bu bulgular, ÇPK ölçeğinin ikinci düzey faktörlü yapısının güvenilir, birleşim geçerliliğine ve ayrışım geçerliliğine sahip olduğunu göstermekte ve ölçek maddelerinin ilgili faktörleri tutarlı ve farklı yapılar olarak başarılı bir şekilde ölçtüğünü doğrulamaktadır.

Tablo 6.16. ÇPK Ölçeği Güvenilirlik, Birleşim ve Ayrışım Geçerliliği (İkinci Düzey)

Faktörler	CR	AVE	HTMT	
			1	2
1- Sosyal Pratikler	0,925	0,805	-	
2- Teknik Pratikler	0,946	0,815	0,811	-

Not: CR=Bileşik Güvenilirlik, AVE=Açıklanan Ortalama Varyans, HTMT= Heterotrait-Monotrait oranı

ÇPK ölçeğinin geçerlilik ve güvenilirlik analizleri kapsamında birinci ve ikinci düzey DFA gerçekleştirilmiştir. Birinci düzey DFA sonuçları, çok faktörlü yapının uyum iyiliği değerlerinin daha yüksek olduğunu göstermiştir. Ancak, ikinci düzey DFA sonuçları, sosyal pratikler ve teknik pratikler olmak üzere iki faktörlü bir üst yapının daha iyi birleşim ve ayrışım geçerliliği sağladığını ve daha yüksek iç tutarlılık ve güvenilirlik değerlerine sahip olduğunu ortaya koymaktadır. Bu bulgular, ölçeğin maddelerinin kendi faktörleri altında tutarlı bir şekilde toplandığını ve faktörlerin birbirinden bağımsız olarak iyi bir şekilde ayrıldığını göstermektedir. Sonuç olarak, birinci düzey DFA'nın uyum iyiliği değerleri daha iyi görünse de ikinci düzey DFA'nın birleşim ve ayrışım geçerliliği ile iç tutarlılık ve güvenilirlik açısından üstün olduğu belirlenmiştir. Araştırma modeli kapsamında düzenleyici değişkeni tanımlayan ÇPK ölçeğine ait ikinci düzey DFA yapısının kullanılmasına karar verilmiştir.

6.3.3. Çevik Yazılım Geliştirme Projesinde Algılanan Başarı Ölçeği Geçerlik Analizine İlişkin Bulgular

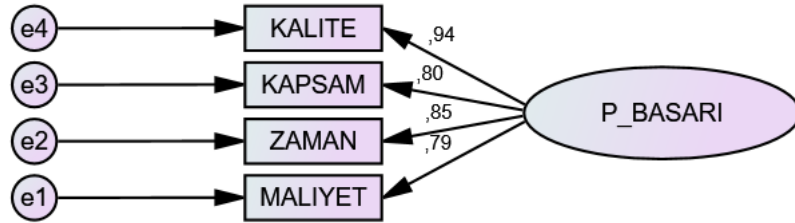
ÇYGPAB ölçeğinin tek faktörlü ve 4 maddeden oluşan birinci düzey tek faktörlü DFA modeli Şekil 6.4'de yer almaktadır. DFA sonucu elde edilen uyum iyiliği değerleri önerilen tek faktörlü modelin veri ile uyumlu ve kabul edilebilir olduğunu göstermektedir.

Modele ilişkin uyum iyiliği değerleri Tablo 6.17’de yer almaktadır. Ölçeğin genel anlamda veri ile iyi uyum gösterdiği görülmektedir.

Tablo 6.17. ÇYGPAB Ölçeği DFA Uyum İyiliği Değerleri

Model	χ^2	df	χ^2/df	GFI	AGFI	CFI	NFI	TLI	RMSEA	SRMR
Birinci Düzyen Tek Faktörlü	3,581	2	1,791*	0,996*	0,979*	0,999*	0,997*	0,996*	0,044*	0,0097*

Not: * simgesi “iyi uyum” değerlerini ifade etmektedir.



CMIN=3,581; CMIN/DF=1,791; GFI=.996; AGFI=.979; CFI=.999; NFI=.997; TLI=.996; RMSEA=.044; SRMR=0,0097

Şekil 6.4. ÇYGPAB Ölçeği DFA Modeli

DFA sonucunda maddelere ilişkin standardize faktör yüklerinin 0,50 veya daha yüksek ve ideal olarak 0,70 veya daha yüksek olması gerekmektedir. Ayrıca maddelere ilişkin p değerleri anlamlı olmalıdır (Gürbüz, 2021a; Hair vd., 2019; Collier, 2020). Ölçek maddelerine ilişkin standardize faktör yükleri ve p değerleri Tablo 6.18’de bulunmaktadır. Tablo 6.18 incelendiğinde ölçek maddelerine ilişkin standardize faktör yükleri 0,798 ile 0,940 arasında yer almaktadır, diğer bir ifadeyle ideal düzeydedir. Maddelere ilişkin p değerleri ise anlamlıdır. Bu sonuçlar ölçme aracında bulunan maddelerin ilgili faktörü güçlü ve anlamlı bir şekilde temsil ettiğini göstermektedir.

Tablo 6.18. ÇYGPAB Ölçeği DFA Regresyon Katsayıları

Faktör	Madde	β	b	SH	C.R.	p
Proje Başarısı	KALITE	0,940	1,271	0,058	21,816	***
	KAPSAM	0,798	0,983	0,055	17,966	***
	ZAMAN	0,847	1,070	0,055	19,446	***
	MALİYET	0,794	1,000			

Not: β =Standardize edilmiş regresyon katsayısı, b=Standardize edilmemiş regresyon katsayısı, SH=Standart Hata, C.R.= Critical Ratio (Kritik Oran- t değeri), ***p<0,001

ÇYGPAB ölçeğinin tek faktörlü yapısına ilişkin hesaplanan bileşik güvenirlik (CR) değeri 0,70'in üzerinde olup (CR=0,910), bu sonuç ölçeğin yüksek iç tutarlılığa ve güvenirliğe sahip olduğunu göstermektedir (Yaşlıoğlu, 2017; Gürbüz, 2021a; Sürücü vd., 2023). Ölçeğin AVE değerinin 0,50'nin üzerinde (AVE=0,717) ve CR değerinden düşük olması, tek faktörlü yapının birleşim geçerliliği kriterini sağladığını ortaya koymaktadır (Yaşlıoğlu, 2017; Gürbüz, 2021a; Collier, 2020). Ölçeğin tek faktörlü yapısı nedeniyle ayrışım geçerliliği analizi yapılmamıştır. Bu bulgular, ÇYGPAB ölçeğinin güvenilir ve birleşim geçerliliğine sahip olduğunu, ölçek maddelerinin ilgili faktörü tutarlı ve başarılı bir şekilde ölçtüğünü doğrulamaktadır.

6.4. DEMOGRAFİK ÖZELLİKLERE İLİŞKİN BULGULAR

Çevik yazılım geliştirmede insan ve süreç faktörlerinin proje başarısına etkisinde çevik pratiklerin düzenleyici rolünü incelemek amacıyla gerçekleştirilen araştırmaya, İstanbul Teknoloji Geliştirme Bölgelerinde (Teknopark) faaliyet gösteren yazılım geliştirme firmalarında çalışan 411 katılımcıdan veri toplanmıştır. Katılımcıların demografik özelliklerine ilişkin bulgular Tablo 6.19'de sunulmuştur.

Tablo 6.19. Katılımcıların Demografik Özellikleri

Değişken	Grup	Frekans (f)	Yüzde (%)
Cinsiyet	Erkek	323	78,6
	Kadın	88	21,4
Yaş	20-25	118	28,7
	26-30	69	16,8
	31-35	88	21,4
	36-40	63	15,3
	41-45	38	9,2
	46 ve üzeri	35	8,5
Eğitim Düzeyi	Lise	7	1,7
	Ön lisans	13	3,2
	Lisans	258	62,8
	Yüksek lisans	87	21,2
	Doktora	46	11,2
Çevik Yazılım Geliştirme Deneyimi	1 yıl ve az	61	14,8
	1-3 yıl	57	13,9
	4-6 yıl	70	17,0
	7-9 yıl	143	34,8
	10 ve üzeri	80	19,5

Tablo 6.19. Katılımcıların Demografik Özellikleri (Devamı)

Değişken	Grup	Frekans (f)	Yüzde (%)
Firmanın Faaliyet Gösterdiği Sektör	Bankacılık	26	6,3
	Bilgi Teknolojileri	34	8,3
	Danışmanlık	34	8,3
	Eğitim	35	8,5
	Enerji	39	9,5
	E-Ticaret	37	9,0
	İmalat	47	11,4
	İnşaat	41	10,0
	Sağlık/Tıp	44	10,7
	Savunma Sanayi	33	8,0
	Telekomünikasyon	41	10,0
Firmadaki Görev	Çevik Koç	40	9,7
	Müşteri/Ürün Sahibi	32	7,8
	Proje Yöneticisi	49	11,9
	Scrum Master	66	16,1
	Takım Lideri	47	11,4
	Takım Üyesi	22	5,4
	Yazılım Geliştirici	155	37,7
Yazılım Geliştirmede Kullanılan Çevik Yöntemler	Kanban	19	4,6
	Scrum	243	59,1
	Scrum/XP	64	15,6
	Scrumban	20	4,9
	XP	65	15,8

Araştırmaya katılan bireylerin %78,6'sı erkek, %21,4'ü ise kadındır. Yaş dağılımı incelendiğinde, katılımcıların en düşük yaşı 22, en yüksek yaşı ise 54 olarak belirlenmiştir. Yaş gruplarına göre dağılım şu şekildedir: %28,7'si 20-25 yaş aralığında, %16,8'i 26-30 yaş aralığında, %21,4'ü 31-35 yaş aralığında, %15,3'ü 36-40 yaş aralığında, %9,2'si 41-45 yaş aralığında ve %8,5'i 46 yaş ve üzerindedir. En fazla katılımcının 20-25 yaş aralığında (%28,7) olduğu görülmektedir.

Eğitim düzeyine bakıldığında, katılımcıların %1,7'si lise, %3,2'si ön lisans, %62,8'i lisans, %21,2'si yüksek lisans ve %11,2'si doktora mezunudur. Katılımcıların büyük çoğunluğu lisans mezunu (%62,8) olduğu belirlenmiştir.

Çevik yazılım geliştirme deneyimi açısından, katılımcıların %14,8'i 1 yıl ve az, %13,9'u 1-3 yıl, %17,0'ı 4-6 yıl, %34,8'i 7-9 yıl ve %19,5'i 10 yıl ve üzeri deneyime

sahiptir. Katılımcıların yarısından fazlasının 7 yıl ve üzeri (%54,3) deneyime sahip olduğu tespit edilmiştir.

Katılımcıların faaliyet gösterdiği sektörler arasında bankacılık (%6,3), bilgi teknolojileri (%8,3), danışmanlık (%8,3), eğitim (%8,5), enerji (%9,5), e-ticaret (%9,0), imalat (%11,4), inşaat (%10,0), sağlık/tıp (%10,7), savunma sanayi (%8,0) ve telekomünikasyon (%10,0) bulunmaktadır. En yüksek oran %11,4 ile imalat sektöründe yer almaktadır.

Firmadaki görev dağılımına bakıldığında, katılımcıların %9,7'si çevik koç, %7,8'i müşteri/ürün sahibi, %11,9'u proje yöneticisi, %16,1'i Scrum Master, %11,4'ü takım lideri, %5,4'ü takım üyesi ve %37,7'si yazılım geliştirici pozisyonundadır. Yazılım geliştirici pozisyonunda bulunan katılımcılar %37,7 ile en büyük grubu oluşturmaktadır.

Yazılım geliştirmede kullanılan çevik yöntemler arasında Kanban (%4,6), Scrum (%59,1), Scrum/XP (%15,6), Scrumban (%4,9) ve XP (%15,8) yer almaktadır. En yaygın olarak kullanılan çevik yöntem %59,1 ile Scrum olarak belirlenmiştir.

6.5. ÖLÇEK İFADELERİNE İLİŞKİN TANIMLAYICI BULGULAR

6.5.1. Çevik Başarı Faktörleri Ölçeğine İlişkin Tanımlayıcı Bulgular

Araştırmada kullanılan ÇBF ölçeğinin maddelerine ilişkin tanımlayıcı istatistikler Tablo 6.20'de sunulmuştur. Bu tabloda, ölçeği oluşturan her bir madde için en düşük ve en yüksek değerler, mod (en sık görülen değer), ortalama ve standart sapma değerlerini belirten temel tanımlayıcı istatistikler yer almaktadır.

Takım yeteneği faktörüne ilişkin maddelerde elde edilen ortalama değerler 5,348 (TK2) ile 5,212 (TK3) arasında değişmektedir. Maddelere verilen değerler 2 ile 7 arasında değişmekte olup, en yüksek moda sahip madde TK1 (Mod=7) iken, en düşük mod değeri 5 ile TK3 ve TK4 maddelerine aittir.

Müşteri katılımı faktörüne ilişkin maddelerde elde edilen ortalama değerler 5,127 (MK1) ile 5,010 (MK3) arasında değişmektedir. Maddelere verilen değerler 2 ile 7 arasında değişmekte olup, tüm maddelerde (MK1, MK2 ve MK3) mod değeri 5 olarak belirlenmiştir.

Proje yönetim süreci faktörüne ilişkin maddelerde elde edilen ortalama değerler 5,134 (PY4) ile 5,078 (PY6) arasında değişmektedir. Maddelere verilen değerler 2 ile 7 arasında değişmekte olup, en yüksek moda sahip madde PY1 (Mod=6) iken, en düşük mod değeri 5 ile PY2, PY3, PY4, PY5 ve PY6 maddelerine aittir.

Proje tanımlama süreci faktörüne ilişkin maddelerde elde edilen ortalama değerler 5,144 (PT3) ile 5,107 (PT2) arasında değişmektedir. Maddelere verilen değerler 2 ile 7 arasında değişmekte olup, en yüksek moda sahip maddeler PT1 ve PT2 (Mod=6) iken, en düşük mod değeri 5 ile PT3 maddesine aittir.

Tablo 6.20. ÇBF Ölçeğine İlişkin Tanımlayıcı İstatistikler

Faktör	Madde	En Düşük	En Yüksek	Mod	Ortalama	Standart Sapma
Takım Yeteneği	TK1	3,00	7,00	7,00	5,265	1,407
	TK2	2,00	7,00	6,00	5,348	1,335
	TK3	2,00	7,00	5,00	5,212	1,337
	TK4	2,00	7,00	5,00	5,251	1,235
	TK5	2,00	7,00	6,00	5,273	1,422
Müşteri Katılımı	MK1	2,00	7,00	5,00	5,127	1,468
	MK2	2,00	7,00	5,00	5,034	1,422
	MK3	2,00	7,00	5,00	5,010	1,423
Proje Yönetim Süreci	PY1	2,00	7,00	6,00	5,107	1,541
	PY2	2,00	7,00	5,00	5,085	1,400
	PY3	2,00	7,00	5,00	5,088	1,424
	PY4	2,00	7,00	5,00 ^a	5,134	1,496
	PY5	2,00	7,00	5,00	5,122	1,424
	PY6	2,00	7,00	5,00 ^a	5,078	1,460
Proje Tanımlama Süreci	PT1	2,00	7,00	6,00	5,134	1,517
	PT2	2,00	7,00	6,00	5,107	1,474
	PT3	2,00	7,00	5,00	5,144	1,495

Not: ^a Birden fazla mod mevcuttur. Küçük olan değer gösterilmiştir.

6.5.3. Çevik Pratiklerin Kullanımı Ölçeğine İlişkin Tanımlayıcı Bulgular

Araştırmada kullanılan ÇPK ölçeğinin maddelerine ilişkin tanımlayıcı istatistikler Tablo 6.21’de sunulmuştur. Tablo 6.21’de ölçek maddelerine ilişkin en düşük ve en yüksek değerler, mod (en sık görülen değer), ortalama ve standart sapma değerlerini yer almaktadır.

Tablo 6.21. ÇPK Ölçeğine İlişkin Tanımlayıcı İstatistikler

Faktör	Madde	En Düşük	En Yüksek	Mod	Ortalama	Standart Sapma
Yinelemeli Teslimat	SYT1	1,00	7,00	7,00	5,455	1,538
	SYT2	1,00	7,00	5,00	5,100	1,454
	SYT3	1,00	7,00	4,00	4,973	1,532
	SYT4	1,00	7,00	7,00	5,182	1,468
Ayaküstü Toplantı	SAT1	1,00	7,00	4,00	5,027	1,505
	SAT2	1,00	7,00	4,00	4,946	1,548
	SAT3	1,00	7,00	5,00	4,990	1,463
Retrospektif	SR1	1,00	7,00	4,00	4,981	1,460
	SR2	1,00	7,00	4,00	4,895	1,553
	SR3	1,00	7,00	5,00	5,002	1,550
Birim Test	TBT1	1,00	7,00	4,00	4,871	1,487
	TBT2	1,00	7,00	5,00	4,961	1,427
	TBT3	1,00	7,00	5,00	5,010	1,360
Sürekli Entegrasyon	TSE1	1,00	7,00	5,00	5,234	1,375
	TSE2	1,00	7,00	5,00	5,097	1,379
	TSE3	1,00	7,00	6,00	5,219	1,388
Kodlama Standartları	TKS1	1,00	7,00	5,00	5,073	1,341
	TKS2	1,00	7,00	5,00	5,207	1,354
	TKS3	1,00	7,00	5,00	5,144	1,482
Yeniden Kod Düzenleme	TYKD1	1,00	7,00	6,00	5,148	1,418
	TYKD2	1,00	7,00	5,00	5,200	1,375
	TYKD3	1,00	7,00	5,00	5,221	1,382

Yinelemeli teslimat faktörüne ilişkin maddelerde elde edilen ortalama değerler 5,455 (SYT1) ile 4,973 (SYT3) arasında değişmektedir. Maddelere verilen değerler 1 ile 7 arasında değişmekte olup, SYT1 ve SYT4 maddeleri en yüksek moda sahipken (Mod=7), SYT3 maddesi en düşük mod değeri olan 4'e sahiptir.

Ayaküstü toplantı faktörüne ilişkin maddelerde elde edilen ortalama değerler 5,027 (SAT1) ile 4,946 (SAT2) arasında değişmektedir. Maddelere verilen değerler 1 ile 7 arasında değişmekte olup, SAT1 ve SAT2 maddeleri en düşük mod değeri olan 4'e sahipken, SAT3 maddesinde en çok görülen değer 5 olmuştur.

Retrospektif faktörüne ilişkin maddelerde elde edilen ortalama değerler 5,002 (SR3) ile 4,895 (SR2) arasında değişmektedir. Maddelere verilen değerler 1 ile 7 arasında değişmekte olup, SR1 ve SR2 maddeleri en düşük mod değeri olan 4'e sahipken, SR3 maddesi 5 moduna sahiptir.

Birim test faktörüne ilişkin maddelerde elde edilen ortalama değerler 5,010 (TBT3) ile 4,871 (TBT1) arasında değişmektedir. Maddelere verilen değerler 1 ile 7 arasında

değişmekte olup, TBT1 maddesi en düşük mod değeri olan 4'e sahipken, TBT2 ve TBT3 maddeleri 5 moduna sahiptir.

Sürekli entegrasyon faktörüne ilişkin maddelerde elde edilen ortalama değerler 5,234 (TSE1) ile 5,097 (TSE2) arasında değişmektedir. Maddelere verilen değerler 1 ile 7 arasında değişmekte olup, TSE1 ve TSE3 maddeleri en yüksek moda sahipken (Mod=6), TSE2 maddesi 5 moduna sahiptir.

Kodlama standartları faktörüne ilişkin maddelerde elde edilen ortalama değerler 5,207 (TKS2) ile 5,073 (TKS1) arasında değişmektedir. Maddelere verilen değerler 1 ile 7 arasında değişmekte olup, tüm maddelerde (TKS1, TKS2 ve TKS3) mod değeri 5 olarak belirlenmiştir.

Yeniden kod düzenleme faktörüne ilişkin maddelerde elde edilen ortalama değerler 5,221 (TYKD3) ile 5,148 (TYKD1) arasında değişmektedir. Maddelere verilen değerler 1 ile 7 arasında değişmekte olup, TYKD1 maddesi en yüksek moda sahipken (Mod=6), TYKD2 ve TYKD3 maddeleri 5 moduna sahiptir.

Sonuç olarak, çevik pratiklerin kullanımına ilişkin maddelere yapılan değerlendirmeler, katılımcıların algılarının genellikle orta ve yüksek düzeyde olduğunu göstermektedir. Özellikle yinelemeli teslimat ve sürekli entegrasyon gibi faktörlerde bazı maddelerin daha yüksek mod değerlerine sahip olması, belirli katılımcı gruplarının bu pratiklerde daha güçlü görüşlere sahip olduğunu işaret etmektedir.

6.5.3. Çevik Yazılım Geliştirme Projesinde Algılanan Başarı Ölçeğine İlişkin Tanımlayıcı Bulgular

Araştırmada kullanılan ÇYGPAB ölçeğinin maddelerine ilişkin tanımlayıcı istatistikler Tablo 6.22'de sunulmuştur.

Tablo 6.22. ÇYGPAB Ölçeğine İlişkin Tanımlayıcı İstatistikler

Faktör	Madde	En Düşük	En Yüksek	Mod	Ortalama	Standart Sapma
Proje Başarısı	KALITE	1,00	7,00	7,00	5,411	1,586
	KAPSAM	1,00	7,00	6,00	5,270	1,444
	ZAMAN	1,00	7,00	6,00	5,314	1,482
	MALİYET	1,00	7,00	6,00	5,324	1,477

ÇYGPAB ölçeğine ilişkin maddelerde elde edilen ortalama değerler 5,411 (KALITE) ile 5,270 (KAPSAM) arasında değişmektedir. Maddelere yapılan değerlendirmeler 1 ile 7 arasında değişmekte olup, KALITE maddesi en yüksek moda sahipken (Mod=7), KAPSAM, ZAMAN ve MALİYET maddelerinde modun 6 sonucuna ulaşılmıştır.

6.6. KORELASYON ANALİZİNE İLİŞKİN BULGULAR

Araştırma kapsamında kullanılan ölçeklerin faktörleri arasındaki ilişkileri diğer bir ifadeyle birlikte nasıl hareket ettiğini belirlemek amacıyla korelasyon analizi gerçekleştirilmiştir. Araştırma verilerinin normal dağılım göstermesi nedeniyle gerçekleştirilen pearson korelasyon analizi sonucu elde edilen bulgular Tablo 6.23'te sunulmuştur (Büyüköztürk, 2016).

Tablo 6.23. Korelasyon Analizi Sonuçları

	TY	MK	PY	PT	SP	TP	PB
TY	1						
MK	,753**	1					
PY	,790**	,728**	1				
PT	,785**	,753**	,749**	1			
SP	,371**	,359**	,454**	,362**	1		
TP	,390**	,350**	,453**	,370**	,850**	1	
PB	,602**	,574**	,603**	,593**	,412**	,459**	1

Not: TY=Takım Yeteneği, MK=Müşteri Katılımı, PY=Proje Yönetim Süreci, PT=Proje Tanımlama Süreci, SP=Sosyal Pratikler, TP=Teknik Pratikler, PB=Proje Başarısı

** simgesi $p < 0,01$ anlamlılık düzeyini göstermektedir.

Korelasyon analizi sonuçları, bağımsız değişkenler (takım yeteneği, müşteri katılımı, proje yönetim süreci ve proje tanımlama süreci) ve düzenleyici değişkenler (sosyal pratikler ve teknik pratikler) ile bağımlı değişken (proje başarısı) arasındaki ilişkileri ortaya koymaktadır.

Takım yeteneği ile müşteri katılımı arasında yüksek düzeyde pozitif ve anlamlı bir ilişki bulunmuştur ($r=0,753$; $p<0,01$). Bu ilişki, takım yeteneğinin müşteri katılımını artırma potansiyelini göstermektedir. Takım yeteneği ile proje yönetim süreci ($r=0,790$; $p<0,01$) ve proje tanımlama süreci ($r=0,785$; $p<0,01$) arasında da yüksek düzeyde pozitif ilişkiler tespit edilmiştir. Bu bulgular, takım yeteneğinin proje yönetimi ve tanımlama süreçleri ile güçlü ilişkilerinin olduğunu göstermektedir. Ayrıca, takım yeteneği ile sosyal

pratikler ($r=0,371$; $p<0,01$) ve teknik pratikler ($r=0,390$; $p<0,01$) arasında orta düzeyde pozitif ilişkiler bulunmaktadır. Takım yeteneği ile proje başarısı arasında orta düzeyde pozitif bir korelasyon ($r=0,602$; $p<0,01$) bulunmuştur. Bu sonuç, takım yeteneğinin proje başarısı ile olan pozitif ilişkisinin göstergesidir. Takım yeteneğinin yüksek olması durumunda proje başarısının da yüksek olacağını göstermektedir.

Müşteri katılımı ile proje yönetim süreci ($r=0,728$; $p<0,01$) ve proje tanımlama süreci ($r=0,753$; $p<0,01$) arasında yüksek düzeyde pozitif korelasyonlar bulunmuştur. Bu sonuçlar, müşteri katılımının proje yönetimi ve tanımlama süreçleri ile güçlü ilişkiler içinde olduğunu göstermektedir. Müşteri katılımı ile sosyal pratikler ($r=0,359$; $p<0,01$) ve teknik pratikler ($r=0,350$; $p<0,01$) arasında orta düzeyde pozitif korelasyonlar tespit edilmiştir. Müşteri katılımı ile proje başarısı arasında ise orta düzeyde pozitif bir korelasyon ($r=0,574$; $p<0,01$) bulunmaktadır. Bu sonuç, müşteri katılımının yüksek olması durumunda proje başarısının da yüksek olacağını göstermektedir.

Proje yönetim süreci ile proje tanımlama süreci ($r=0,749$; $p<0,01$) arasında yüksek düzeyde pozitif bir korelasyon bulunmaktadır. Bu ilişki, proje yönetim ve tanımlama süreçlerinin birbirleriyle güçlü bir ilişki içinde olduğunu göstermektedir. Proje yönetim süreci ile sosyal pratikler ($r=0,454$; $p<0,01$) ve teknik pratikler ($r=0,453$; $p<0,01$) arasında orta düzeyde pozitif korelasyonlar tespit edilmiştir. Ayrıca proje yönetim süreci ile proje başarısı arasında orta düzeyde pozitif bir korelasyon ($r=0,603$; $p<0,01$) bulunmuştur.

Proje tanımlama süreci ile sosyal pratikler ($r=0,362$; $p<0,01$) ve teknik pratikler ($r=0,370$; $p<0,01$) arasında orta düzeyde pozitif korelasyonlar bulunmuştur. Proje tanımlama ile proje başarısı arasında orta düzeyde pozitif ilişki ($r=0,593$; $p<0,01$) tespit edilmiştir.

Sosyal pratikler ile teknik pratikler arasında yüksek düzeyde pozitif bir ilişki ($r=0,850$; $p<0,01$) bulunmuştur. Sosyal pratikler ile proje başarısı ($r=0,412$; $p<0,01$) ve teknik pratikler ile proje başarısı ($r=0,459$; $p<0,01$) arasında ise anlamlı orta düzeyde pozitif korelasyonlar bulunmuştur. Bu bulgular, sosyal ve teknik pratiklerin proje başarısı ile olan pozitif ilişkilerini göstermektedir.

6.7. HİPOTEZ TESTLERİNE İLİŞKİN BULGULAR

Çevik yazılım geliştirmede insan ve süreç faktörlerinin proje başarısına etkisinde çevik pratiklerin düzenleyici rolünü incelemek amacıyla yapılan araştırmada hipotezler AMOS 26 programı kullanılarak yapısal regresyon modeli analizi ile test edilmiştir. Ayrıca düzenleyici etki analizleri IBM SPSS 22 programında Hayes (2022) tarafından geliştirilen Process Makro (4.2) kullanılarak gerçekleştirilmiştir. Hipotez testlerine ilişkin bulgular başlıklar halinde sunulmuştur.

6.7.1 Yapısal Regresyon Modeli Analizi

Yapısal regresyon modeli analizi ölçüm modeli ve yapısal model testi olmak üzere iki aşamadan oluşmaktadır. Öncelikli olarak gizil değişkenlerin gözlemlenen değişkenler tarafından nasıl tanımlandığını gösteren ve gözlenen değişkenlerin ölçüm özelliklerini tanımlayan ölçüm modeli test edilmektedir. Genellikle doğrulayıcı faktör analizi ile gerçekleştirilmektedir (Hair vd. 2019). Ölçüm modeli doğrulandıktan sonra gizil değişkenler arasındaki nedensel ilişkilerin tanımlandığı yapısal model test edilmektedir (Gürbüz, 2021a).

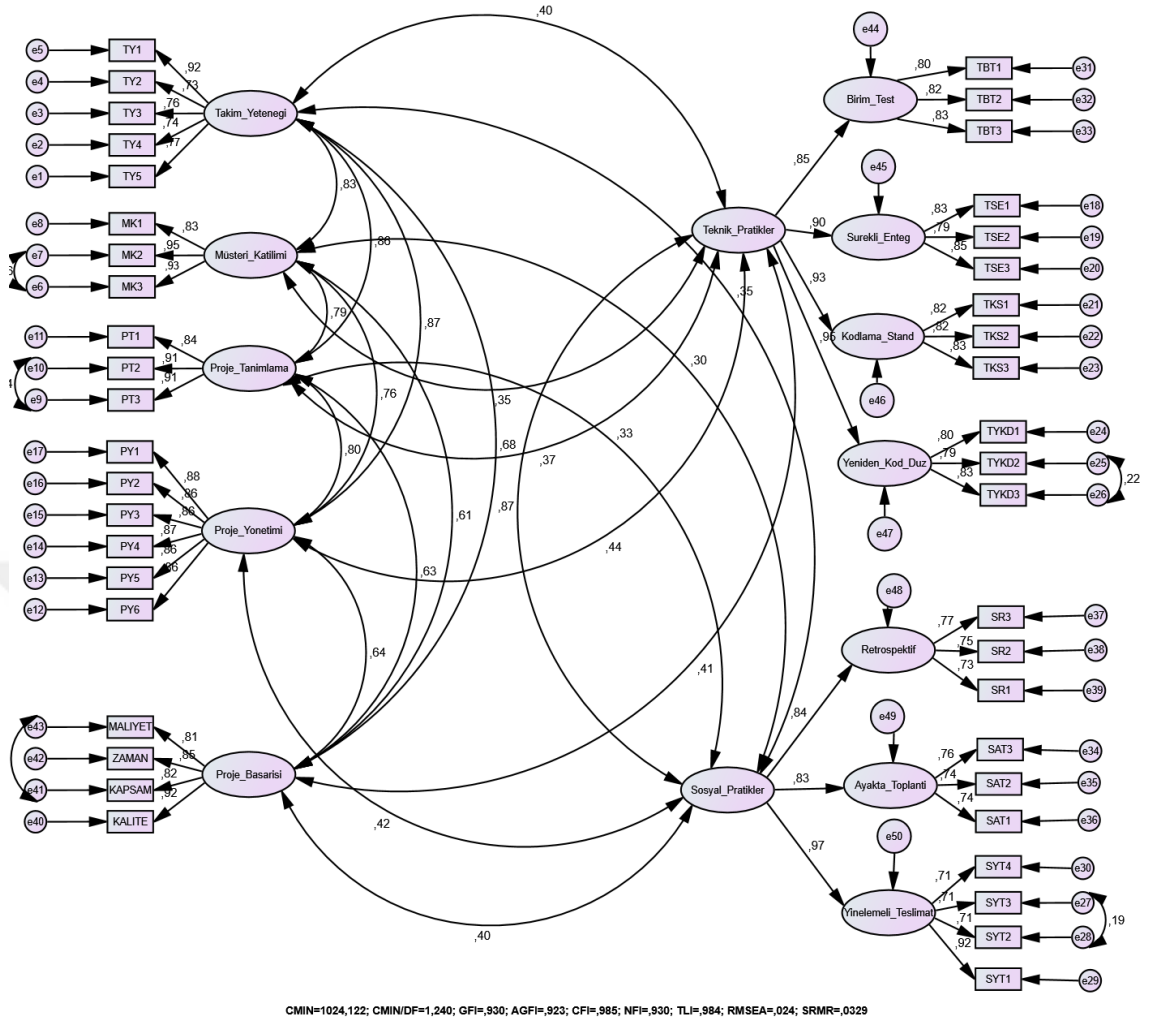
6.7.1.1. Ölçüm Modeli: Geçerlilik ve Güvenilirliğe İlişkin Bulgular

Gizil değişkenlerin gözlenen değişkenler tarafından nasıl tanımlandığını ve gözlenen değişkenlerin ölçüm özelliklerini (geçerlilik ve güvenilirlik) tanımlamak amacıyla oluşturulan ölçüm modeli Şekil 6.5'te yer almaktadır. DFA sonucu elde edilen uyum iyiliği değerleri ölçüm modelinin veri ile uyumlu ve kabul edilebilir olduğunu göstermektedir. Modele ilişkin uyum iyiliği değerleri Tablo 6.24'de yer almaktadır. Ölçeğin genel anlamda veri ile iyi uyum gösterdiği görülmektedir.

Tablo 6.24. Ölçüm Modeli Uyum İyiliği Değerleri

Model	χ^2	df	χ^2/df	GFI	AGFI	CFI	NFI	TLI	RMSEA	SRMR
Ölçüm Modeli	1024,122	826	1,240*	0,930**	0,923**	0,985*	0,930**	0,984*	0,024*	0,0329*

Not: * simgesi “iyi uyum”, ** simgesi “kabul edilebilir uyum” değerlerini ifade etmektedir.



Şekil 6.5. Ölçüm Modeli

Ölçüm modeli maddelerine ilişkin standardize faktör yükleri ve p değerleri Tablo 6.25'te bulunmaktadır. DFA sonucunda maddelere ilişkin standardize faktör yüklerinin 0,50 veya daha yüksek ve ideal olarak 0,70 veya daha yüksek olması gerekmektedir. Ayrıca maddelere ilişkin p değerleri anlamlı olmalıdır (Hair vd., 2019; Gürbüz, 2021a). Tablo 6.25 incelendiğinde ölçüm modeli maddelerine ilişkin standardize faktör yükleri 0,707 ile 0,966 arasında yer almaktadır, diğer bir ifadeyle ideal düzeydedir (Collier, 2020). Maddelere ilişkin p değerleri ise anlamlıdır. Bu sonuçlar ölçüm modelinde bulunan maddelerin ilgili faktörleri güçlü ve anlamlı bir şekilde temsil ettiğini göstermektedir.

Tablo 6.25. Ölçüm Modeli DFA Regresyon Katsayıları

Faktör	Madde	β	b	SH	C.R.	p
Takım Yeteneği	TY1	0,924	1,191	0,057	20,798	***
	TY2	0,728	0,891	0,057	15,553	***
	TY3	0,755	0,925	0,057	16,239	***
	TY4	0,737	0,833	0,053	15,773	***
	TY5	0,768	1,000			
Müşteri Katılımı	MK1	0,833	0,921	0,041	22,343	***
	MK2	0,947	1,015	0,041	24,847	***
	MK3	0,933	1,000			
Proje Yönetim Süreci	PY1	0,876	1,075	0,044	24,241	***
	PY2	0,857	0,956	0,041	23,268	***
	PY3	0,855	0,969	0,042	23,151	***
	PY4	0,873	1,039	0,043	24,058	***
	PY5	0,858	0,973	0,042	23,295	***
	PY6	0,861	1,000			
Proje Tanımlama Süreci	PT1	0,837	0,930	0,041	22,434	***
	PT2	0,911	0,984	0,040	24,398	***
	PT3	0,913	1,000			
Sosyal Pratikler	SYT	0,966	1,128	0,096	11,742	***
	SAT	0,830	1,000			
	SR	0,839	1,077	0,085	12,691	***
Teknik Pratikler	TBT	0,847	0,987	0,069	14,231	***
	TSE	0,901	1,000			
	TKS	0,930	0,997	0,063	15,844	***
	TYKD	0,954	1,054	0,067	15,668	***
Yinelemeli Teslimat	SYT1	0,920	1,307	0,076	17,136	***
	SYT2	0,710	0,954	0,063	15,167	***
	SYT3	0,707	1,000			
	SYT4	0,709	0,961	0,071	13,607	***
Ayaküstü Toplantı	SAT1	0,741	0,999	0,070	14,335	***
	SAT2	0,737	1,021	0,072	14,245	***
	SAT3	0,763	1,000			
Retrospektif	SR1	0,729	0,894	0,063	14,256	***
	SR2	0,749	0,977	0,067	14,649	***
	SR3	0,768	1,000			
Birim Test	TBT1	0,803	1,000			
	TBT2	0,818	0,978	0,055	17,732	***
	TBT3	0,825	0,940	0,052	17,909	***
Sürekli Entegrasyon	TSE1	0,827	1,000			
	TSE2	0,794	0,962	0,053	18,177	***
	TSE3	0,854	1,043	0,052	20,066	***
Kodlama Standartları	TKS1	0,819	1,000			
	TKS2	0,817	1,007	0,053	18,844	***
	TKS3	0,835	1,125	0,058	19,386	***
Yeniden Kod Düzenleme	TYKD1	0,799	1,000			
	TYKD2	0,790	0,959	0,056	17,054	***
	TYKD3	0,825	1,007	0,056	18,079	***
Proje Başarısı	KALITE	0,920	1,000			
	KAPSAM	0,818	0,809	0,036	22,431	***
	ZAMAN	0,850	0,862	0,035	24,917	***
	MALİYET	0,814	0,824	0,037	22,228	***

Not: β =Standardize edilmiş regresyon katsayısı, b=Standardize edilmemiş regresyon katsayısı, SH=Standart Hata, C.R.= Critical Ratio (Kritik Oran- t değeri), ***p<0,001

Ölçüm modeline ait güvenilirlik, birleşim geçerliliği ve ayrışım geçerliliği sonuçları Tablo 6.26’da yer almaktadır. Ölçüm modelinde yer alan faktörlerin CR değerleri 0,70’in üzerinde olup, bu sonuçlar her bir faktörün yüksek iç tutarlılığa ve güvenilirliğe sahip olduğunu göstermektedir (Yaşlıoğlu, 2017; Gürbüz, 2021a; Sürücü vd., 2023). Faktörlere ait AVE değerlerinin 0,50’nin üzerinde ve CR değerlerinden düşük olması, faktörlerin birleşim geçerliliği kriterini sağladığını ortaya koymaktadır (Yaşlıoğlu, 2017; Collier, 2020; Gürbüz, 2021a). Ayrıca faktörlere ait HTMT oranı değerlerinin 0,90’nın altında olması, ayrışım geçerliliğinin bulunduğunu ve yapıların birbirinden farklı yapılar olduğunu doğrulamaktadır (Henseler vd., 2015; Kline, 2015; Collier, 2020). Bu bulgular, ölçüm modelinin güvenilir, birleşim geçerliliğine ve ayrışım geçerliliğine sahip olduğunu göstermekte ve ölçüm modelinden bulunan maddelerinin ilgili faktörleri tutarlı ve farklı yapılar olarak başarılı bir şekilde ölçtüğünü doğrulamaktadır.

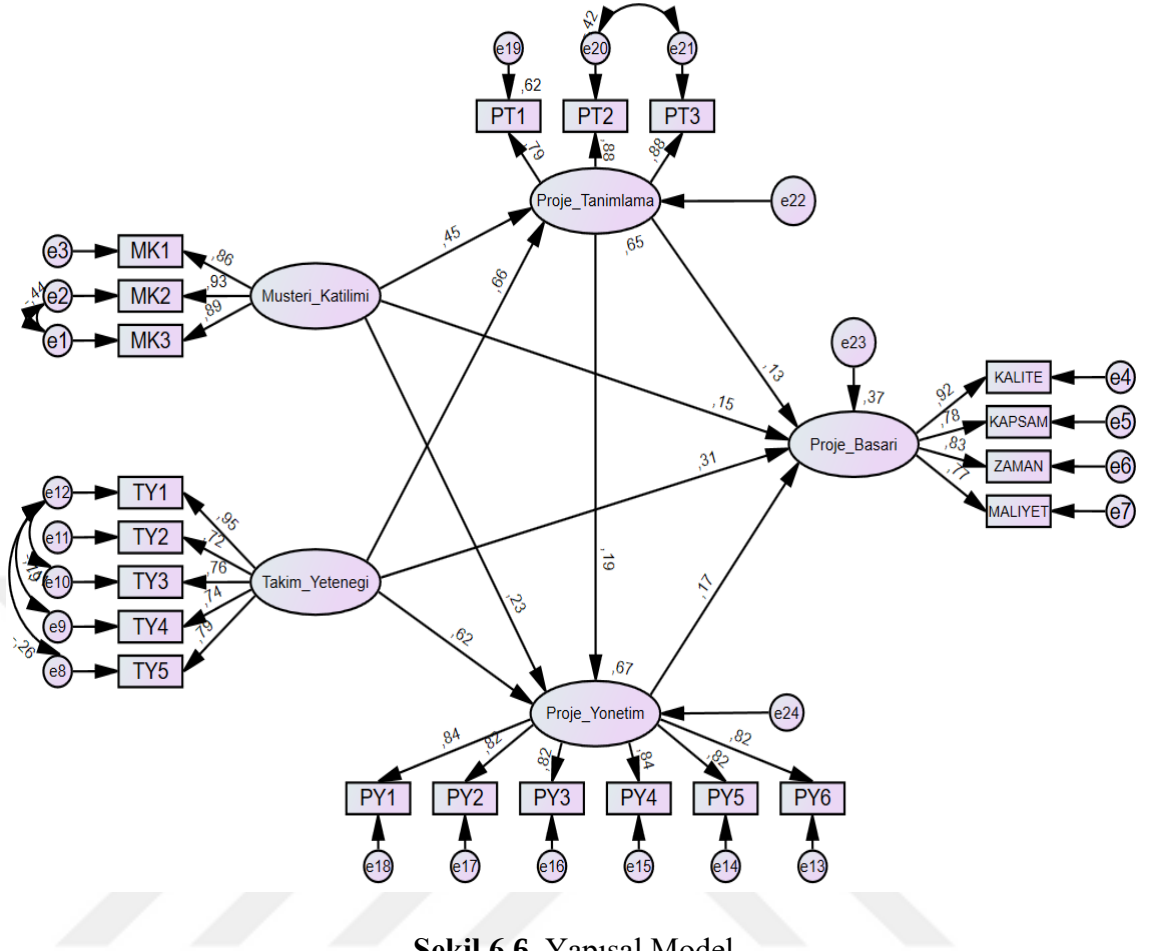
Tablo 6.26. Ölçüm Modeli Güvenilirlik, Birleşim ve Ayrışım Geçerliliği

Faktörler	CR	AVE	HTMT						
			1	2	3	4	5	6	7
1- Takım Yeteneği	0,889	0,617	-						
2- Müşteri Katılımı	0,932	0,820	0,836	-					
3- Proje Yönetim Süreci	0,946	0,745	0,860	0,785	-				
4- Proje Tanımlama Süreci	0,918	0,788	0,874	0,830	0,811	-			
5- Sosyal Pratikler	0,912	0,776	0,306	0,277	0,392	0,318	-		
6- Teknik Pratikler	0,950	0,826	0,373	0,330	0,433	0,368	0,811	-	
7- Proje Başarısı	0,913	0,725	0,668	0,632	0,651	0,655	0,371	0,414	-

Not: CR=Bileşik Güvenilirlik, AVE=Açıklanan Ortalama Varyans, HTMT= Heterotrait-Monotrait oranı

6.7.1.2. Yapısal Model: Hipotezlere İlişkin Bulgular

Yapısal modelin odak noktası yapılar arasındaki ilişkilerin incelenmesidir. Dolayısıyla bağımsız yapıların bağımlı yapıları nasıl etkilediğine veya daha karmaşık modellerde bağımlı değişkenlerin diğer bağımlı değişkenleri nasıl etkilediğini belirlemek amacıyla oluşturulmaktadır (Collier, 2020). Ölçüm modeli doğrulandıktan diğer bir ifadeyle geçerlilik ve güvenilirlik sağlandıktan sonra gizil değişkenler arasındaki nedensel ilişkilerin tanımlandığı yapısal model testi gerçekleştirilmiştir (Gürbüz, 2021a). Araştırma hipotezlerini (H₁-H₉ arası) test etmek amacıyla oluşturan yapısal model Şekil 6.6’da yer almaktadır.



Şekil 6.6. Yapısal Model

Yapısal modele ilişkin uyum iyiliği değerleri Tablo 6.27’de yer almaktadır. Tablo 6.27 incelendiğinde uyum iyiliği değerlerinin kabul edilebilir düzeyde diğer ifadeyle veri ile uyumlu ve modelin geçerli olduğu görülmektedir.

Tablo 6.27. Yapısal Model Uyum İyiliği Değerleri

Model	χ^2	df	χ^2/df	GFI	AGFI	CFI	NFI	TLI	RMSEA	SRMR
Yapısal Model	605,598	175	3,461*	0,905*	0,875*	0,944*	0,930*	0,933*	0,077*	0,0217*

Not: * simgesi “kabul edilebilir uyum” değerlerini ifade etmektedir.

Yapısal regresyon analizi sonucunda doğrudan etkilere ilişkin yapısal model bulguları Tablo 6.28’de yer almaktadır.

Tablo 6.28. Yapısal Modele İlişkin Bulgular

Değişkenler Arası İlişkiler	β	C.R.	p	Hipotez Sonucu
H ₁ : Müşteri Katılımı →Proje Tanımlama	0,453	10,131	***	Kabul
H ₂ : Müşteri Katılımı →Proje Yönetim	0,231	4,770	***	Kabul
H ₃ : Müşteri Katılımı →Proje Başarısı	0,146	2,325	0,020	Kabul
H ₄ : Takım Yeteneği →Proje Tanımlama	0,664	12,348	***	Kabul
H ₅ : Takım Yeteneği →Proje Yönetim	0,625	9,060	***	Kabul
H ₆ : Takım Yeteneği →Proje Başarısı	0,312	3,318	***	Kabul
H ₇ : Proje Tanımlama →Proje Yönetim	0,186	2,715	0,007	Kabul
H ₈ : Proje Tanımlama →Proje Başarısı	0,132	1,580	0,114	Ret
H ₉ : Proje Yönetim →Proje Başarısı	0,171	1,989	0,047	Kabul

Not: Proje Tanımlama (R^2) = 0,646; Proje Yönetim (R^2) = 0,671; Proje Başarısı (R^2) = 0,366; ***<0,001
C.R.= Critical Ratio (Kritik Oran- t değeri)

Yapısal regresyon analizi sonuçları müşteri katılımının proje tanımlama süreci üzerinde anlamlı ve pozitif bir etkiye sahip olduğunu göstermektedir (β = 0,453; p < 0,001). Müşteri katılımının proje yönetim süreci üzerindeki etkisi de anlamlı ve pozitif bulunmuştur (β = 0,231; p < 0,001). Ayrıca, müşteri katılımının proje başarısı üzerindeki etkisi anlamlı ve pozitif olup olduğu tespit edilmiştir (β = 0,146; p = 0,020). Bu bulgular H₁, H₂ ve H₃ hipotezlerini desteklemektedir.

Takım yeteneği, proje tanımlama süreci üzerinde anlamlı ve pozitif etkiye sahiptir (β = 0,664; p < 0,001) ve H₄ hipotezini desteklemektedir. Takım yeteneğinin proje yönetim süreci üzerindeki etkisinin de anlamlı ve pozitif olduğu (β = 0,625; p < 0,001) tespit edilmiştir. Bu bulgu H₅ hipotezinin kabul edilmesini sağlamaktadır. Benzer şekilde, takım yeteneğinin proje başarısı üzerindeki etkisi de anlamlı ve pozitif olarak bulunmuş (β = 0,312; p < 0,001) ve H₆ hipotezi desteklemektedir.

Proje tanımlama sürecinin proje yönetim süreci üzerindeki etkisi anlamlı ve pozitif olarak tespit edilmiştir (β = 0,186; p = 0,007). Bu sonuç H₇ hipotezini desteklemektedir. Ancak, proje tanımlama sürecinin proje başarısı üzerindeki etkisinin anlamlı olmadığı sonucuna ulaşılmıştır (β = 0,132; p = 0,114). Bu nedenle H₈ hipotezi reddedilmiştir.

Proje yönetim sürecinin proje başarısı üzerindeki etkisi anlamlı ve pozitif olarak bulunmuş (β = 0,171; p = 0,047) ve H₉ hipotezinin kabul edilmiştir.

Yapısal regresyon analizi sonuçlarına göre araştırmanın kuramsal çerçevesi doğrultusunda geliştirilen H_1 , H_2 , H_3 , H_4 , H_5 , H_6 , H_7 ve H_9 hipotezleri desteklenmiş, H_8 hipotezi reddedilmiştir.

Modelin açıklayıcı gücünün proje tanımlama süreci için %64,6 ($R^2 = 0,646$), proje yönetim süreci için %67,1 ($R^2 = 0,671$) ve proje başarısı için %36,6 ($R^2 = 0,366$) olduğu tespit edilmiştir. Bu sonuçlar, takım yeteneği ve müşteri katılımının proje tanımlama süreci, proje yönetim süreci ve proje başarısı üzerindeki etkilerini ortaya koymaktadır. Proje tanımlama sürecinin proje yönetim süreci üzerinde etkisi bulunurken, proje başarısı üzerinde anlamlı bir etkisinin bulunmaması ise dikkate değer bir bulgudur. Proje yönetim süreci de proje başarısını etkilemektedir. Bu bulgular çevik yazılım geliştirmede takım yeteneği, müşteri katılımı ve proje yönetim sürecinin başarılı proje sonuçlarına ulaşmada belirleyici faktörler olduğunu göstermektedir.

6.7.2. Düzenleyici Değişken Etkisinin Analizi ve Bulguları

Araştırmanın kuramsal çerçevesi doğrultusunda, çevik pratiklerin düzenleyici etkisini incelemek amacıyla geliştirilen H_{10} , H_{11} , H_{12} ve H_{13} hipotezlerinin analizleri IBM SPSS 22 programında Process Makro (v4.2) kullanılarak gerçekleştirilmiştir. Hipotez testlerine ilişkin bulgular başlıklar halinde sunulmuştur.

6.7.2.1. Müşteri Katılımının Çevik Yazılım Geliştirme Proje Başarısı Üzerindeki Etkisinde Çevik Pratiklerin Düzenleyici Rolü

Müşteri katılımının çevik yazılım geliştirme proje başarısı üzerindeki etkisinde çevik pratiklerin (sosyal pratikler ve teknik pratikler) düzenleyici rolünün tespitine yönelik yapılan regresyon analizi sonuçları Tablo 6.29'da yer almaktadır. Yapılan regresyon analizi, müşteri katılımının proje başarısına etkisini ve bu etkinin çevik pratiklerin kullanılması durumunda nasıl düzenlendiğini göstermektedir.

Tablo 6.29. Müşteri Katılımının Proje Başarısına Etkisinde Çevik Pratiklerin Düzenleyici Rolüne İlişkin Regresyon Analizi Sonuçları

Değişken	b	S.H.	t	p	LLCI	ULCI
Sabit	5,240	0,054	98,000	0,000***	5,135	5,345
Müşteri Katılımı (X)	0,463	0,040	11,549	0,000***	0,384	0,542
Sosyal Pratikler (W)	0,006	0,088	0,065	0,948	-0,168	0,179
X.W	0,252	0,066	3,846	0,000***	0,123	0,381
Teknik Pratikler (Z)	0,466	0,095	4,897	0,000***	0,279	0,653
X.Z	-0,082	0,067	-1,230	0,220	-0,213	0,049

Not: Model Özeti: R=0,663; R²= 0,439; F (5,405) =63,435; p<0,001

b= Standardize edilmemiş regresyon katsayısı, S.H.=Standart Hata, ***p< 0,001, **p< 0,01, * p< 0,05

LLCI=Alt Limit Güven Aralığı, ULCI=Üst Limit Güven Aralığı

Müşteri katılımı, proje başarısı üzerinde pozitif ve anlamlı bir etkiye sahiptir (b = 0,463; p <0,001). Sosyal pratiklerin proje başarısı üzerindeki etkisi ise istatistiksel olarak anlamsızdır (b = 0,006; p = 0,948). Bu sonuç, sosyal pratiklerin tek başına proje başarısı üzerinde belirgin bir etki yaratmadığını göstermektedir. Ancak, müşteri katılımı ile sosyal pratiklerin etkileşimi proje başarısı üzerinde pozitif ve anlamlı bir etkiye sahiptir (b = 0,252; p <0,001). Bu, müşteri katılımının proje başarısı üzerindeki etkisinin sosyal pratiklere bağlı olarak değiştiğini ve bu değişimin pozitif olduğunu göstermektedir.

Çevik yazılım geliştirmede teknik pratiklerin kullanımı proje başarısı üzerinde pozitif ve anlamlı bir etkiye sahiptir (b = 0,466; p <0,001). Teknik pratiklerin kullanımının artması, proje başarısını artırmaktadır. Müşteri katılımı ile teknik pratiklerin kullanımının etkileşimi proje başarısı üzerinde anlamlı bir etkiye sahip değildir (b = -0,082; p = 0,220). Bu, müşteri katılımının proje başarısı üzerindeki etkisinin teknik pratiklere bağlı olarak değişmediğini göstermektedir.

Tablo 6.30. Müşteri Katılımının Proje Başarısına Etkisinde Çevik Pratiklerin Düzenleyici Rolüne İlişkin Özet Sonuçlar

Etkileşim	R ² Değişimi	F	df1	df2	p
X.W	0,021	14,793	1,000	405,000	0,000***
X.Z	0,002	1,512	1,000	405,000	0,220
X.W ve X.Z (Birlikte)	0,034	12,223	2,000	405,000	0,000***

Not: X=Müşteri Katılımı, W=Sosyal Pratikler, Z=Teknik Pratikler, ***p<0,001, **p<0,01, * p<0,05

Müşteri katılımının proje başarısına etkisinde çevik pratiklerin düzenleyici rolüne ilişkin özet sonuçlar Tablo 6.30'da yer almaktadır. Tablo 6.30 incelendiğinde müşteri katılımı (X) ile sosyal pratiklerin (W) etkileşimi (X.W), proje başarısı üzerinde anlamlı

ve pozitif bir etkiye sahip olduğu ($F(1,405) = 14,793$; $p < 0,001$) ve bu değişimin %2,1'sini açıkladığı görülmektedir. Bu bulgu, sosyal pratiklerin müşteri katılımının proje başarısı üzerindeki etkisini güçlendirdiğini diğer bir ifadeyle sosyal pratiklerin düzenleyici bir etkiye sahip olduğunu göstermektedir ve H_{10a} hipotezi kabul edilmiştir.

Müşteri katılımı (X) ile teknik pratiklerin (Z) etkileşimi (X.Z), proje başarısı üzerinde anlamlı bir etkiye sahip olmadığı tespit edilmiştir ($F(1,405) = 1,512$; $p = 0,220$). Bu sonuç, teknik pratiklerin müşteri katılımının proje başarısı üzerindeki etkisini değiştirmediğini göstermektedir. Dolayısıyla müşteri katılımının proje başarısı üzerindeki etkisinde teknik pratiklerin düzenleyici rolü bulunmamaktadır ve H_{10b} hipotezi reddedilmiştir.

Müşteri katılımı ile hem sosyal hem de teknik pratiklerin birlikte etkileşimi (X.W ve X.Z), proje başarısı üzerinde anlamlı ve pozitif bir etkiye sahip olduğu tespit edilmiştir ($F(2,405) = 12,223$; $p < 0,001$) ve etkileşimsel değişkenin açıkladığı ilave varyans ise yaklaşık %3,4'tür. Çevik pratikler birlikte kullanıldığında müşteri katılımının proje başarısına olan etkisini artırdığını göstermektedir ve H_{10c} hipotezi kabul edilmiştir.

Gürbüz (2021b) düzenleyici etkinin anlamlı olması durumunda eğim (slope) analizi yapılarak sonuçların metin ve grafik ile birlikte yorumlanmasını önermektedir (Gürbüz, 2021b). Sosyal ve teknik pratikleri kullanmanın düşük, orta ve yüksek olduğu durumlarda müşteri katılımının proje başarısı üzerindeki etkilerinin anlamlı olup olmadığını gösteren eğim analizi sonuçları Tablo 6.31'de yer almaktadır.

Tablo 6.31. Müşteri Katılımının Proje Başarısına Etkisinde Çevik Pratiklerin Farklı Değerlerine Göre Durumsal Etkileri

Sosyal Pratikler	Teknik Pratikler	b	S.H.	t	p	LLCI	ULCI
Düşük	Düşük	0,277	0,061	4,539	0,000***	0,157	0,397
	Orta	0,192	0,082	2,328	0,020*	0,030	0,354
	Yüksek	0,107	0,139	0,765	0,445	-0,167	0,381
Orta	Düşük	0,548	0,080	6,857	0,000***	0,391	0,706
	Orta	0,463	0,040	11,549	0,000***	0,384	0,542
	Yüksek	0,378	0,080	4,723	0,000***	0,221	0,536
Yüksek	Düşük	0,820	0,138	5,942	0,000***	0,549	1,091
	Orta	0,735	0,080	9,175	0,000***	0,577	0,892
	Yüksek	0,650	0,058	11,188	0,000***	0,536	0,764

Not: b= Standardize edilmemiş regresyon katsayısı, S.H.=Standart Hata, *** $p < 0,001$, ** $p < 0,01$, * $p < 0,05$ LLCI=Alt Limit Güven Aralığı, ULCI=Üst Limit Güven Aralığı

Müşteri katılımının proje başarısına etkisinde çevik pratiklerin farklı kullanım seviyelerine göre durumsal etkileri incelendiğinde aşağıdaki sonuçlara ulaşılmıştır.

Sosyal pratikleri kullanmanın düşük ve;

- Teknik pratikleri kullanmanın düşük olduğu durumda müşteri katılımının proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,277$; $p < 0,001$). Bu etkiyi sosyal ve teknik pratikler düzenlemektedir
- Teknik pratikleri kullanmanın orta olduğu durumda müşteri katılımının proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,192$; $p = 0,020$). Bu etkide sosyal ve teknik pratiklerin düzenleyici rolü bulunmaktadır.
- Teknik pratikleri kullanmanın yüksek olduğu durumda müşteri katılımının proje başarısına olan etkisi anlamlı değildir ($b = 0,107$; $p = 0,445$). Dolayısıyla müşteri katılımının proje başarısına olan etkisinde düzenleyici rolü bulunmamaktadır.

Sosyal pratikleri kullanmanın orta ve;

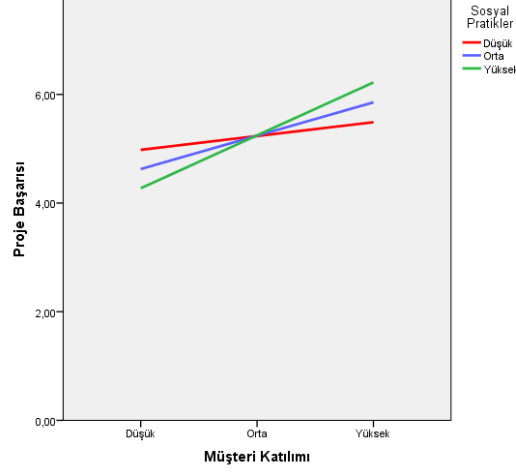
- Teknik pratikleri kullanmanın düşük olduğu durumda müşteri katılımının proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,548$; $p < 0,001$).
- Teknik pratikleri kullanmanın orta olduğu durumda müşteri katılımının proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,463$; $p < 0,001$).
- Teknik pratikleri kullanmanın yüksek olduğu durumda müşteri katılımının proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,378$; $p < 0,001$).

Sosyal pratikleri kullanmanın yüksek ve;

- Teknik pratikleri kullanmanın düşük olduğu durumda müşteri katılımının proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,820$; $p < 0,001$).
- Teknik pratikleri kullanmanın orta olduğu durumda müşteri katılımının proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,735$; $p < 0,001$).
- Teknik pratikleri kullanmanın yüksek olduğu durumda müşteri katılımının proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,650$; $p < 0,001$).

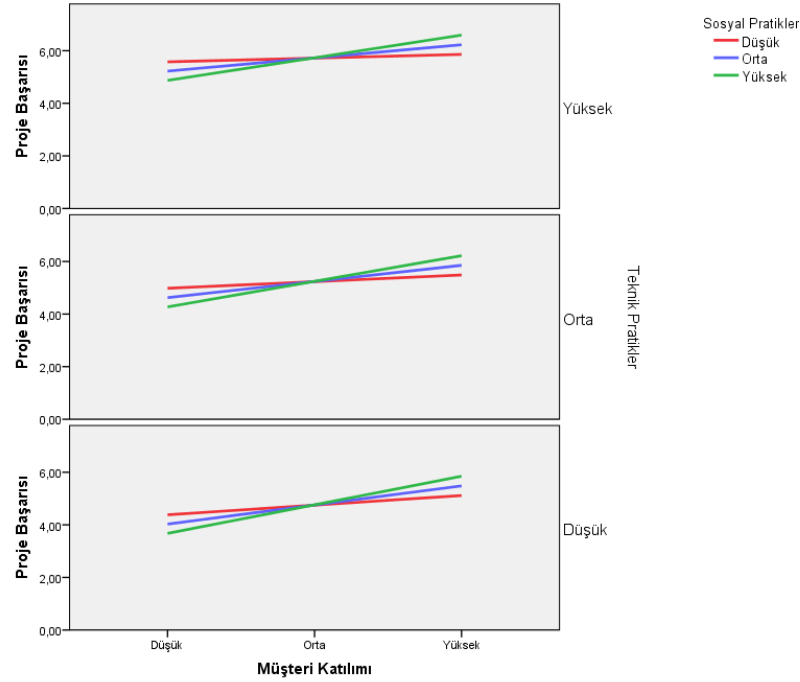
Müşteri katılımının proje başarısına etkisinde çevik pratiklerin düzenleyici rolüne ilişkin hipotezlerden H_{10a} ve H_{10c} kabul edilmiş, H_{10b} ise reddedilmiştir.

Müşteri katılımının çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal pratiklerin düzenleyici rolüne (H_{10a}) ilişkin grafiksel gösterim Şekil 6.7’de yer almaktadır.



Şekil 6.7. Müşteri Katılımının Proje Başarısına Etkisinde Sosyal Pratiklerin Düzenleyici Rolünün Grafiksel Gösterimi

Müşteri katılımının çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal ve teknik pratiklerin birlikte düzenleyici rolüne (H_{10c}) ilişkin grafiksel gösterim Şekil 6.8’de yer almaktadır.



Şekil 6.8. Müşteri Katılımının Proje Başarısına Etkisinde Sosyal ve Teknik Pratiklerin Birlikte Düzenleyici Rolünün Grafiksel Gösterimi

Şekil 6.8’deki grafikte yatay eksen “Müşteri Katılımı”, dikey eksen “Proje Başarısı” ve düzenleyici değişkenlerin farklı durumlarının (düşük-orta-yüksek) gösterimi yer almaktadır. Grafikte eğimin fazla olması etkinin de yüksek olduğunu ifade etmektedir (Sürücü vd., 2023) Grafik incelendiğinde sonuçların eğim (slope) analizindeki bulgularla örtüştüğü görülmektedir.

6.7.2.2. Takım Yeteneğinin Çevik Yazılım Geliştirme Proje Başarısı Üzerindeki Etkisinde Çevik Pratiklerin Düzenleyici Rolü

Takım yeteneğinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde çevik pratiklerin (sosyal pratikler ve teknik pratikler) düzenleyici rolünün tespitine yönelik yapılan regresyon analizi sonuçları Tablo 6.32’de yer almaktadır.

Tablo 6.32. Takım Yeteneğinin Proje Başarısına Etkisinde Çevik Pratiklerin Düzenleyici Rolüne İlişkin Regresyon Analizi Sonuçları

Değişken	b	S.H.	t	p	LLCI	ULCI
Sabit	5,222	0,052	99,629	0,000***	5,119	5,325
Takım Yeteneği (X)	0,575	0,046	12,484	0,000***	0,485	0,666
Sosyal Pratikler (W)	0,142	0,086	1,646	0,101	-0,028	0,311
X.W	0,447	0,075	5,953	0,000***	0,299	0,594
Teknik Pratikler (Z)	0,344	0,091	3,796	0,000***	0,166	0,523
X.Z	-0,205	0,077	-2,674	0,008**	-0,356	-0,054

Not: Model Özeti: R=0,693, R²= 0,481; F (5,405) =74,940, p<0,001

b= Standardize edilmemiş regresyon katsayısı, S.H.=Standart Hata, ***p< 0,001, **p< 0,01, * p< 0,05
LLCI=Alt Limit Güven Aralığı, ULCI=Üst Limit Güven Aralığı

Yapılan regresyon analizi, takım yeteneğinin proje başarısına etkisini ve bu etkinin çevik pratikler tarafından nasıl düzenlendiğini incelemektedir. Takım yeteneği, proje başarısı üzerinde pozitif ve anlamlı bir etkiye sahiptir (b = 0,575; p <0,001). Sosyal pratiklerin proje başarısı üzerindeki etkisi ise istatistiksel olarak anlamsızdır (b = 0,142; p = 0,101). Bu sonuç, sosyal pratiklerin tek başına proje başarısı üzerinde belirgin bir etki yaratmadığını göstermektedir. Ancak takım yeteneği ile sosyal pratiklerin etkileşimi proje başarısı üzerinde pozitif ve anlamlı bir etkiye sahiptir (b = 0,447; p <0,001). Bu bulgu takım yeteneğinin proje başarısı üzerindeki etkisinin sosyal pratiklere bağlı olarak değiştiğini ve bu değişimin pozitif olduğunu göstermektedir.

Çevik yazılım geliştirmede teknik pratiklerin kullanımı proje başarısı üzerinde pozitif ve anlamlı bir etkiye sahiptir ($b = 0,344$; $p < 0,001$). Teknik pratiklerin kullanımının artması, proje başarısını artırmaktadır. Takım yeteneği ile teknik pratiklerin kullanımının etkileşimi proje başarısı üzerinde anlamlı ve negatif bir etkiye sahiptir ($b = -0,205$; $p = 0,008$). Bu bulgu takım yeteneğinin proje başarısı üzerindeki etkisinin teknik pratiklere bağlı olarak değiştiğini ve bu değişimin negatif olduğunu göstermektedir.

Tablo 6.33. Takım Yeteneğinin Proje Başarısına Etkisinde Çevik Pratiklerin Düzenleyici Rolüne İlişkin Özet Sonuçlar

Etkileşim	R ² Değişimi	F	df1	df2	p
X.W	0,045	35,434	1,000	405,000	0,000***
X.Z	0,009	7,153	1,000	405,000	0,008**
X.W ve X.Z (Birlikte)	0,060	23,209	2,000	405,000	0,000***

Not: X=Takım Yeteneği, W=Sosyal Pratikler, Z=Teknik Pratikler, *** $p < 0,001$, ** $p < 0,01$, * $p < 0,05$

Takım yeteneğinin proje başarısına etkisinde çevik pratiklerin düzenleyici rolüne ilişkin özet sonuçlar Tablo 6.33'te yer almaktadır. Tablo 6.33 incelendiğinde takım yeteneği (X) ile sosyal pratiklerin (W) etkileşimi (X.W), proje başarısı üzerinde anlamlı ve pozitif bir etkiye sahip olduğu ($F(1,405) = 35,434$; $p < 0,001$) ve bu değişimin %4,5'ini açıkladığı görülmektedir. Bu bulgu, sosyal pratiklerin takım yeteneğinin proje başarısı üzerindeki etkisini güçlendirdiğini diğer bir ifadeyle sosyal pratiklerin düzenleyici bir etkiye sahip olduğunu göstermektedir ve H_{11a} hipotezi kabul edilmiştir.

Benzer şekilde takım yeteneği (X) ile teknik pratiklerin (Z) etkileşimi (X.Z), proje başarısı üzerinde anlamlı bir etkiye sahip olduğu tespit edilmiştir ($F(1,405) = 7,153$; $p = 0,008$). Etkileşim terimi modele eklendiğinde, açıklanan varyansta %0,9'luk bir artış gözlemlenmiştir. Bu sonuç, takım yeteneğinin proje başarısı üzerindeki etkisinde teknik pratiklerin düzenleyici bir role sahip olduğunu göstermektedir ve H_{11b} hipotezi kabul edilmiştir.

Takım yeteneği ile hem sosyal hem de teknik pratiklerin birlikte etkileşimi (X.W ve X.Z), proje başarısı üzerinde anlamlı ve pozitif bir etkiye sahip olduğu tespit edilmiştir ($F(2,405) = 23,209$; $p < 0,001$) ve etkileşimsel değişkenin açıkladığı ilave varyans ise yaklaşık %6'dır. Çevik pratiklerin birlikte kullanılmasıyla takım yeteneğinin proje başarısına etkisinde artış olacağını göstermektedir ve H_{11c} hipotezi kabul edilmiştir.

Gürbüz (2021b) düzenleyici etkinin anlamlı olması durumunda eğim (slope) analizi yapılarak sonuçların metin ve grafik ile birlikte yorumlanmasını önermektedir (Gürbüz, 2021b). Sosyal ve teknik pratikleri kullanmanın düşük, orta ve yüksek olduğu durumlarda takım yeteneğinin proje başarısı üzerindeki etkilerinin anlamlı olup olmadığını gösteren eğim analizi sonuçları Tablo 6.34'te yer almaktadır.

Tablo 6.34. Takım Yeteneğinin Proje Başarısına Etkisinde Çevik Pratiklerin Farklı Değerlerine Göre Durumsal Etkileri

Sosyal Pratikler	Teknik Pratikler	b	S.H.	t	p	LLCI	ULCI
Düşük	Düşük	0,307	0,071	4,314	0,000***	0,167	0,447
	Orta	0,094	0,094	0,996	0,320	-0,091	0,278
	Yüksek	-0,120	0,159	-0,753	0,452	-0,433	0,193
Orta	Düşük	0,789	0,093	8,523	0,000***	0,607	0,970
	Orta	0,575	0,046	12,484	0,000***	0,485	0,666
	Yüksek	0,362	0,092	3,947	0,000***	0,182	0,542
Yüksek	Düşük	1,270	0,159	8,009	0,000***	0,958	1,582
	Orta	1,057	0,092	11,451	0,000***	0,876	1,238
	Yüksek	0,844	0,068	12,435	0,000***	0,710	0,977

Not: b= Standardize edilmemiş regresyon katsayısı, S.H.=Standart Hata, ***p<0,001, **p<0,01, * p<0,05
LLCI=Alt Limit Güven Aralığı, ULCI=Üst Limit Güven Aralığı

Takım yeteneğinin proje başarısına etkisinde çevik pratiklerin farklı kullanım seviyelerine göre durumsal etkileri incelendiğinde aşağıdaki sonuçlara ulaşılmıştır.

Sosyal pratikleri kullanmanın düşük ve;

- Teknik pratikleri kullanmanın düşük olduğu durumda takım yeteneğinin proje başarısına olan etkisi pozitif ve anlamlıdır (b = 0,307; p <0,001). Bu etkiyi sosyal ve teknik pratikler düzenlemektedir.
- Teknik pratikleri kullanmanın orta olduğu durumda takım yeteneğinin proje başarısına olan etkisi anlamlı değildir (b = 0,094; p = 0,320). Dolayısıyla takım yeteneğinin proje başarısına olan etkisinde düzenleyici rolü bulunmamaktadır.
- Teknik pratikleri kullanmanın yüksek olduğu durumda takım yeteneğinin proje başarısına olan etkisi anlamlı değildir (b = -0,120 p = 0,452). Dolayısıyla takım yeteneğinin proje başarısına olan etkisinde düzenleyici rolü bulunmamaktadır.

Sosyal pratikleri kullanmanın orta ve;

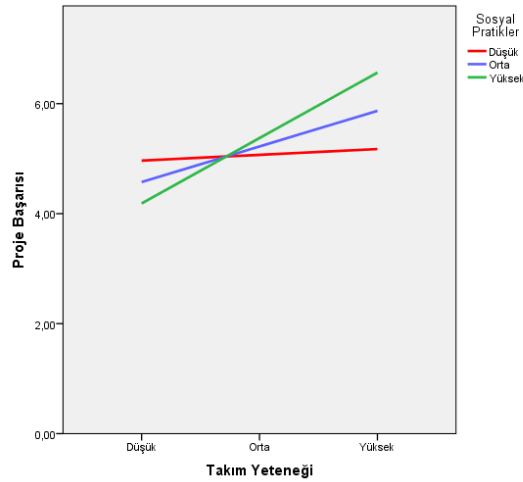
- Teknik pratikleri kullanmanın düşük olduğu durumda takım yeteneğinin proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,789$; $p < 0,001$).
- Teknik pratikleri kullanmanın orta olduğu durumda takım yeteneğinin proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,575$; $p < 0,001$).
- Teknik pratikleri kullanmanın yüksek olduğu durumda takım yeteneğinin proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,362$; $p < 0,001$).

Sosyal pratikleri kullanmanın yüksek ve;

- Teknik pratikleri kullanmanın düşük olduğu durumda takım yeteneğinin proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 1,270$; $p < 0,001$).
- Teknik pratikleri kullanmanın orta olduğu durumda takım yeteneğinin proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 1,057$; $p < 0,001$).
- Teknik pratikleri kullanmanın yüksek olduğu durumda takım yeteneğinin proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,844$; $p < 0,001$).

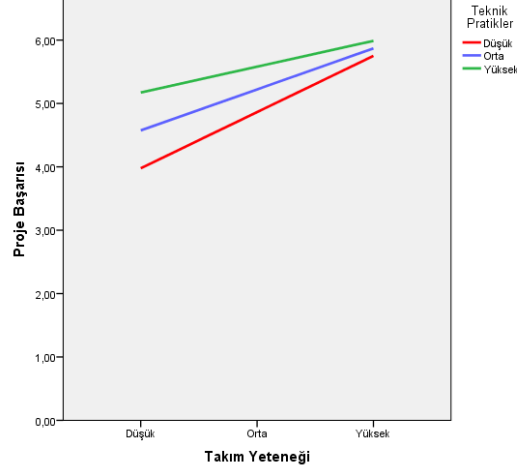
Takım yeteneğinin proje başarısına etkisinde çevik pratiklerin düzenleyici rolüne ilişkin H_{11a} , H_{11b} ve H_{11c} hipotezleri kabul edilmiştir.

Takım yeteneğinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal pratiklerin düzenleyici rolüne (H_{11a}) ilişkin grafiksel gösterim Şekil 6.9'da yer almaktadır.



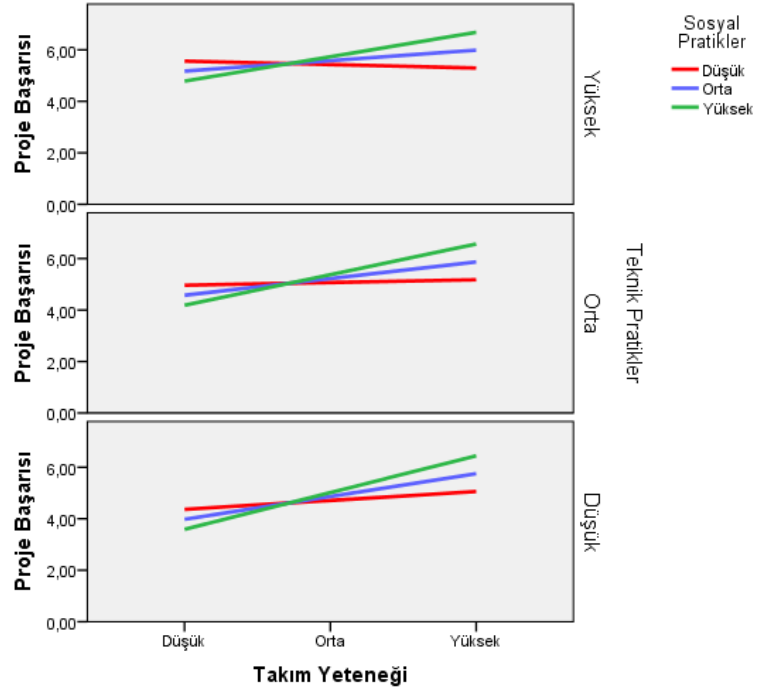
Şekil 6.9. Takım Yeteneğinin Proje Başarısına Etkisinde Sosyal Pratiklerin Düzenleyici Rolünün Grafiksel Gösterimi

Takım yeteneğinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde teknik pratiklerin düzenleyici rolüne (H_{11b}) ilişkin grafiksel gösterim Şekil 6.10'da yer almaktadır.



Şekil 6.10. Takım Yeteneğinin Proje Başarısına Etkisinde Teknik Pratiklerin Düzenleyici Rolünün Grafiksel Gösterimi

Takım yeteneğinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal ve teknik pratiklerin birlikte düzenleyici rolüne (H_{11c}) ilişkin grafiksel gösterim Şekil 6.11'de yer almaktadır.



Şekil 6.11. Takım Yeteneğinin Proje Başarısına Etkisinde Sosyal ve Teknik Pratiklerin Birlikte Düzenleyici Rolünün Grafiksel Gösterimi

Şekil 6.11'deki grafikte yatay eksen "Takım Yeteneği", dikey eksen "Proje Başarısı" ve düzenleyici değişkenlerin farklı durumlarının (düşük-orta-yüksek) gösterimi yer almaktadır. Grafikte eğimin fazla olması etkinin de yüksek olduğunu ifade etmektedir (Sürücü vd., 2023) Grafik incelendiğinde sonuçların eğim (slope) analizindeki bulgularla örtüştüğü görülmektedir.

6.7.2.3. Proje Tanımlama Sürecinin Çevik Yazılım Geliştirme Proje Başarısı Üzerindeki Etkisinde Çevik Pratiklerin Düzenleyici Rolü

Proje tanımlama sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde çevik pratiklerin (sosyal pratikler ve teknik pratikler) düzenleyici rolünün tespitine yönelik yapılan regresyon analizi sonuçları Tablo 6.35'de yer almaktadır.

Tablo 6.35. Proje Tanımlama Sürecinin Proje Başarısına Etkisinde Çevik Pratiklerin Düzenleyici Rolüne İlişkin Regresyon Analizi Sonuçları

Değişken	b	S.H.	t	p	LLCI	ULCI
Sabit	5,229	0,052	100,283	0,000***	5,126	5,331
Proje Tanımlama Süreci (X)	0,470	0,038	12,372	0,000***	0,395	0,545
Sosyal Pratikler (W)	0,063	0,086	0,731	0,465	-0,106	0,232
X.W	0,281	0,059	4,741	0,000***	0,165	0,398
Teknik Pratikler (Z)	0,422	0,092	4,603	0,000***	0,242	0,602
X.Z	-0,094	0,061	-1,535	0,126	-0,214	0,026

Not: Model Özeti: $R=0,683$, $R^2=0,466$; $F(5,405)=70,801$, $p<0,001$

b= Standardize edilmemiş regresyon katsayısı, S.H.=Standart Hata, *** $p<0,001$, ** $p<0,01$, * $p<0,05$

LLCI=Alt Limit Güven Aralığı, ULCI=Üst Limit Güven Aralığı

Yapılan regresyon analizi, proje tanımlama sürecinin proje başarısına etkisini ve bu etkinin çevik pratikler tarafından nasıl düzenlendiğini incelemektedir. Proje tanımlama süreci, proje başarısı üzerinde pozitif ve anlamlı bir etkiye sahiptir ($b = 0,470$; $p < 0,001$). Sosyal pratiklerin proje başarısı üzerindeki etkisi ise istatistiksel olarak anlamsızdır ($b = 0,063$; $p = 0,465$). Bu sonuç, sosyal pratiklerin tek başına proje başarısı üzerinde belirgin bir etki yaratmadığını göstermektedir. Ancak proje tanımlama süreci ile sosyal pratiklerin etkileşimi proje başarısı üzerinde pozitif ve anlamlı bir etkiye sahiptir ($b = 0,281$; $p < 0,001$). Bu bulgu proje tanımlama sürecinin proje başarısı üzerindeki etkisinin sosyal pratiklere bağlı olarak değiştiğini ve bu değişimin pozitif olduğunu göstermektedir.

Çevik yazılım geliştirmede teknik pratiklerin kullanımı proje başarısı üzerinde pozitif ve anlamlı bir etkiye sahiptir ($b = 0,422$; $p < 0,001$). Teknik pratiklerin kullanımının artması, proje başarısını artırmaktadır. Proje tanımlama süreci ile teknik pratiklerin kullanımının etkileşimi proje başarısı üzerinde anlamlı bir etkiye sahip değildir ($b = 0,094$; $p = 0,126$). Bu durum proje tanımlama sürecinin proje başarısı üzerindeki etkisinin teknik pratiklere bağlı olarak değişmediğini göstermektedir.

Tablo 6.36. Proje Tanımlama Sürecinin Proje Başarısına Etkisinde Çevik Pratiklerin Düzenleyici Rolüne İlişkin Özet Sonuçlar

Etkileşim	R ² Değişimi	F	df1	df2	p
X.W	0,030	22,482	1,000	405,000	0,000***
X.Z	0,003	2,355	1,000	405,000	0,126
X.W ve X.Z (Birlikte)	0,049	18,456	2,000	405,000	0,000***

Not: X=Proje Tanımlama Süreci, W=Sosyal Pratikler, Z=Teknik Pratikler, *** $p < 0,001$, ** $p < 0,01$, * $p < 0,05$

Proje tanımlama sürecinin proje başarısına etkisinde çevik pratiklerin düzenleyici rolüne ilişkin özet sonuçlar Tablo 6.36’da yer almaktadır. Tablo 5.36 incelendiğinde proje tanımlama süreci (X) ile sosyal pratiklerin (W) etkileşimi (X.W), proje başarısı üzerinde anlamlı ve pozitif bir etkiye sahip olduğu ($F(1,405) = 22,482$; $p < 0,001$) ve bu değişimin %3’ünü açıkladığı görülmektedir. Bu bulgu, sosyal pratiklerin proje tanımlama sürecinin proje başarısı üzerindeki etkisini güçlendirdiğini diğer bir ifadeyle sosyal pratiklerin düzenleyici bir etkiye sahip olduğunu göstermektedir ve H_{12a} hipotezi kabul edilmiştir.

Proje tanımlama süreci (X) ile teknik pratiklerin (Z) etkileşimi (X.Z), proje başarısı üzerinde anlamlı bir etkiye sahip olmadığı tespit edilmiştir ($F(1,405) = 2,355$; $p = 0,126$). Bu sonuç, teknik pratiklerin proje tanımlama sürecinin proje başarısı üzerindeki etkisini değiştirmedikçe göstermektedir. Dolayısıyla proje tanımlama sürecinin proje başarısı üzerindeki etkisinde teknik pratiklerin düzenleyici rolü bulunmamaktadır ve H_{12b} hipotezi reddedilmiştir.

Proje tanımlama süreci ile hem sosyal hem de teknik pratiklerin birlikte etkileşimi (X.W ve X.Z), proje başarısı üzerinde anlamlı ve pozitif bir etkiye sahip olduğu tespit edilmiştir ($F(2,405) = 18,456$; $p < 0,001$) ve etkileşimsel değişkenin açıkladığı ilave varyans ise %4,9’dur. Çevik pratikler birlikte kullanıldığında proje tanımlama sürecinin proje başarısına olan etkisini artırdığını göstermektedir ve H_{12c} hipotezi kabul edilmiştir.

Gürbüz (2021b) düzenleyici etkinin anlamlı olması durumunda eğim (slope) analizi yapılarak sonuçların metin ve grafik ile birlikte yorumlanmasını önermektedir (Gürbüz, 2021b). Sosyal ve teknik pratikleri kullanmanın düşük, orta ve yüksek olduğu durumlarda proje tanımlama sürecinin proje başarısı üzerindeki etkilerinin anlamlı olup olmadığını gösteren eğim analizi sonuçları Tablo 6.37’de yer almaktadır.

Tablo 6.37. Proje Tanımlama Sürecinin Proje Başarısına Etkisinde Çevik Pratiklerin Farklı Değerlerine Göre Durumsal Etkileri

Sosyal Pratikler	Teknik Pratikler	b	S.H.	t	p	LLCI	ULCI
Düşük	Düşük	0,264	0,056	4,727	0,000***	0,154	0,374
	Orta	0,167	0,075	2,236	0,026*	0,020	0,313
	Yüksek	0,069	0,127	0,545	0,586	-0,180	0,318
Orta	Düşük	0,568	0,074	7,628	0,000***	0,421	0,714
	Orta	0,470	0,038	12,372	0,000***	0,395	0,545
	Yüksek	0,372	0,074	5,047	0,000***	0,227	0,517
Yüksek	Düşük	0,871	0,127	6,857	0,000***	0,621	1,120
	Orta	0,773	0,074	10,421	0,000***	0,627	0,919
	Yüksek	0,676	0,055	12,379	0,000***	0,568	0,783

Not: b= Standardize edilmemiş regresyon katsayısı, S.H.=Standart Hata, ***p<0,001, **p<0,01, * p<0,05
LLCI=Alt Limit Güven Aralığı, ULCI=Üst Limit Güven Aralığı

Proje tanımlama sürecinin proje başarısına etkisinde çevik pratiklerin farklı değerlerine göre durumsal etkileri incelendiğinde aşağıdaki sonuçlara ulaşılmıştır.

Sosyal pratikleri kullanmanın düşük ve;

- Teknik pratikleri kullanmanın düşük olduğu durumda proje tanımlama sürecinin proje başarısına olan etkisi pozitif ve anlamlıdır (b = 0,264; p <0,001).
- Teknik pratikleri kullanmanın orta olduğu durumda proje tanımlama sürecinin proje başarısına olan etkisi pozitif ve anlamlıdır (b = 0,167; p = 0,026).
- Teknik pratikleri kullanmanın yüksek olduğu durumda proje tanımlama sürecinin proje başarısına olan etkisi anlamlı değildir (b = 0,069; p = 0,586). Dolayısıyla proje tanımlama sürecinin proje başarısına olan etkisinde düzenleyici rolü bulunmamaktadır.

Sosyal pratikleri kullanmanın orta ve;

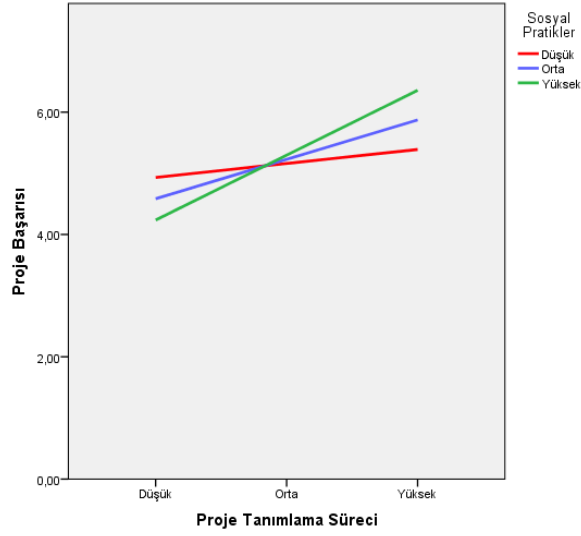
- Teknik pratikleri kullanmanın düşük olduğu durumda proje tanımlama sürecinin proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,548$; $p < 0,001$).
- Teknik pratikleri kullanmanın orta olduğu durumda proje tanımlama sürecinin proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,470$; $p < 0,001$).
- Teknik pratikleri kullanmanın yüksek olduğu durumda proje tanımlama sürecinin proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,372$; $p < 0,001$).

Sosyal pratikleri kullanmanın yüksek ve;

- Teknik pratikleri kullanmanın düşük olduğu durumda proje tanımlama sürecinin proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,871$; $p < 0,001$).
- Teknik pratikleri kullanmanın orta olduğu durumda proje tanımlama sürecinin proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,773$; $p < 0,001$).
- Teknik pratikleri kullanmanın yüksek olduğu durumda proje tanımlama sürecinin proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,676$; $p < 0,001$).

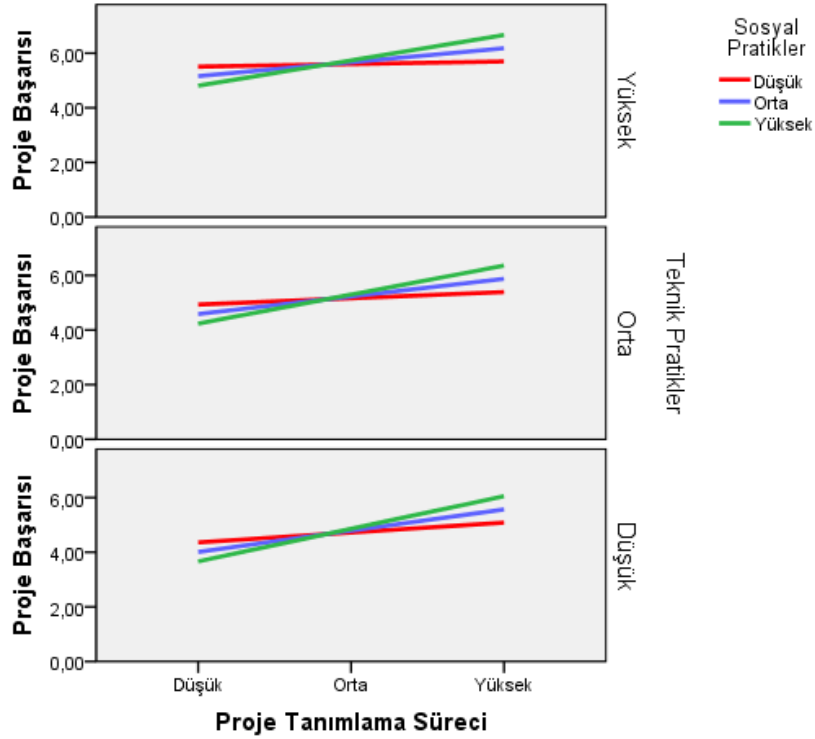
Proje tanımlama sürecinin proje başarısına etkisinde çevik pratiklerin düzenleyici rolüne ilişkin hipotezlerden H_{12a} ve H_{12c} kabul edilmiş, H_{12b} ise reddedilmiştir.

Proje tanımlama sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal pratiklerin düzenleyici rolüne (H_{12a}) ilişkin grafiksel gösterim Şekil 6.12’de yer almaktadır.



Şekil 6.12. Proje Tanımlama Sürecinin Proje Başarısına Etkisinde Sosyal Pratiklerin Düzenleyici Rolünün Grafiksel Gösterimi

Proje tanımlama sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal ve teknik pratiklerin birlikte düzenleyici rolüne (H_{12c}) ilişkin grafiksel gösterim Şekil 6.13'te yer almaktadır.



Şekil 6.13. Proje Tanımlama Sürecinin Proje Başarısına Etkisinde Sosyal ve Teknik Pratiklerin Birlikte Düzenleyici Rolünün Grafiksel Gösterimi

Grafikte yatay eksen “Proje Tanımlama Süreci”, dikey eksen “Proje Başarısı” ve düzenleyici değişkenlerin farklı durumlarının (düşük-orta-yüksek) gösterimi yer almaktadır. Eğimin fazla olması etkinin de yüksek olduğunu ifade etmektedir (Sürücü vd., 2023) Grafik incelendiğinde sonuçların eğim (slope) analizindeki bulgularla örtüştüğü görülmektedir.

6.7.2.4. Proje Yönetim Sürecinin Çevik Yazılım Geliştirme Proje Başarısı Üzerindeki Etkisinde Çevik Pratiklerin Düzenleyici Rolü

Proje yönetim sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde çevik pratiklerin (sosyal pratikler ve teknik pratikler) düzenleyici rolünün tespitine yönelik yapılan regresyon analizi sonuçları Tablo 6.38’de yer almaktadır.

Tablo 6.38. Proje Yönetim Sürecinin Proje Başarısına Etkisinde Çevik Pratiklerin Düzenleyici Rolüne İlişkin Regresyon Analizi Sonuçları

Değişken	b	S.H.	t	p	LLCI	ULCI
Sabit	5,216	0,055	94,102	0,000***	5,107	5,325
Proje Yönetim Süreci (X)	0,492	0,043	11,500	0,000***	0,408	0,576
Sosyal Pratikler (W)	0,082	0,090	0,915	0,361	-0,094	0,259
X.W	0,336	0,063	5,328	0,000***	0,212	0,460
Teknik Pratikler (Z)	0,378	0,092	4,111	0,000***	0,197	0,559
X.Z	-0,163	0,066	-2,455	0,015*	-0,293	-0,032

Not: Model Özeti: R=0,675, R²= 0,455; F (5,405) =67,760, p<0,001

b= Standardize edilmemiş regresyon katsayısı, S.H.=Standart Hata, ***p< 0,001, **p< 0,01, * p< 0,05

LLCI=Alt Limit Güven Aralığı, ULCI=Üst Limit Güven Aralığı

Yapılan regresyon analizi, proje yönetim sürecinin proje başarısına etkisini ve bu etkinin çevik pratikler tarafından nasıl düzenlendiğini incelemektedir. Proje yönetim sürecinin, proje başarısı üzerinde pozitif ve anlamlı bir etkiye sahiptir (b = 0,492; p <0,001). Sosyal pratiklerin proje başarısı üzerindeki etkisi ise istatistiksel olarak anlamsızdır (b = 0,082; p = 0,361). Bu sonuç, sosyal pratiklerin tek başına proje başarısı üzerinde belirgin bir etki yaratmadığını göstermektedir. Ancak proje yönetim süreci ile sosyal pratiklerin etkileşimi proje başarısı üzerinde pozitif ve anlamlı bir etkiye sahiptir (b = 0,336; p <0,001). Bu bulgu proje yönetim sürecinin proje başarısı üzerindeki etkisinin sosyal pratiklere bağlı olarak değiştiğini ve bu değişimin pozitif olduğunu göstermektedir.

Çevik yazılım geliştirmede teknik pratiklerin kullanımı proje başarısı üzerinde pozitif ve anlamlı bir etkiye sahiptir ($b = 0,378$; $p < 0,001$). Teknik pratiklerin kullanımının artması, proje başarısını artırmaktadır. Proje yönetim süreci ile teknik pratiklerin kullanımının etkileşimi proje başarısı üzerinde anlamlı ve negatif bir etkiye sahiptir ($b = -0,163$; $p = 0,015$). Bu bulgu proje yönetim sürecinin proje başarısı üzerindeki etkisinin teknik pratiklere bağlı olarak değiştiğini ve bu değişimin negatif olduğunu göstermektedir.

Tablo 6.39. Proje Yönetim Sürecinin Proje Başarısına Etkisinde Çevik Pratiklerin Düzenleyici Rolüne İlişkin Özet Sonuçlar

Etkileşim	R ² Değişimi	F	df1	df2	p
X.W	0,038	28,392	1,000	405,000	0,000***
X.Z	0,008	6,025	1,000	405,000	0,015*
X.W ve X.Z (Birlikte)	0,048	17,838	2,000	405,000	0,000***

Not: X=Proje Yönetim Süreci, W=Sosyal Pratikler, Z=Teknik Pratikler, *** $p < 0,001$, ** $p < 0,01$, * $p < 0,05$

Proje yönetim sürecinin proje başarısına etkisinde çevik pratiklerin düzenleyici rolüne ilişkin özet sonuçlar Tablo 6.39’da yer almaktadır. Tablo 6.39 incelendiğinde proje yönetim süreci (X) ile sosyal pratiklerin (W) etkileşimi (X.W), proje başarısı üzerinde anlamlı ve pozitif bir etkiye sahip olduğu ($F(1,405) = 28,392$; $p < 0,001$) ve bu değişimin %3,8’ini açıkladığı görülmektedir. Bu bulgu, sosyal pratiklerin proje yönetim sürecinin proje başarısı üzerindeki etkisini güçlendirdiğini diğer bir ifadeyle sosyal pratiklerin düzenleyici bir etkiye sahip olduğunu göstermektedir ve H_{13a} hipotezi kabul edilmiştir.

Benzer şekilde proje yönetim süreci (X) ile teknik pratiklerin (Z) etkileşimi (X.Z), proje başarısı üzerinde anlamlı bir etkiye sahip olduğu tespit edilmiştir ($F(1,405) = 6,025$; $p = 0,015$). Etkileşim terimi modele eklendiğinde, açıklanan varyansta %0,8’luk bir artış gözlemlenmiştir. Bu sonuç, proje yönetim sürecinin proje başarısı üzerindeki etkisinde teknik pratiklerin düzenleyici bir role sahip olduğunu göstermektedir ve H_{13b} hipotezi kabul edilmiştir.

Proje yönetim süreci ile hem sosyal hem de teknik pratiklerin birlikte etkileşimi (X.W ve X.Z), proje başarısı üzerinde anlamlı ve pozitif bir etkiye sahip olduğu tespit edilmiştir ($F(2,405) = 17,838$; $p < 0,001$) ve etkileşimsel değişkenin açıkladığı ilave varyans ise yaklaşık %4,8’dir. Çevik pratiklerin birlikte kullanılmasıyla proje yönetim

sürecinin proje başarısına etkisinde artış olacağını göstermektedir ve H_{13c} hipotezi kabul edilmiştir.

Gürbüz (2021b) düzenleyici etkinin anlamlı olması durumunda eğim (slope) analizi yapılarak sonuçların metin ve grafik ile birlikte yorumlanmasını önermektedir (Gürbüz, 2021b). Sosyal ve teknik pratikleri kullanmanın düşük, orta ve yüksek olduğu durumlarda proje yönetim sürecinin proje başarısı üzerindeki etkilerinin anlamlı olup olmadığını gösteren eğim analizi sonuçları Tablo 6.40'ta yer almaktadır.

Tablo 6.40. Proje Yönetim Sürecinin Proje Başarısına Etkisinde Çevik Pratiklerin Farklı Değerlerine Göre Durumsal Etkileri

Sosyal Pratikler	Teknik Pratikler	b	S.H.	t	p	LLCI	ULCI
Düşük	Düşük	0,299	0,065	4,618	0,000***	0,172	0,427
	Orta	0,130	0,083	1,572	0,117	-0,033	0,293
	Yüksek	-0,039	0,138	-0,285	0,776	-0,310	0,232
Orta	Düşük	0,662	0,081	8,213	0,000***	0,503	0,820
	Orta	0,492	0,043	11,500	0,000***	0,408	0,576
	Yüksek	0,323	0,082	3,949	0,000***	0,162	0,484
Yüksek	Düşük	1,024	0,134	7,627	0,000***	0,760	1,288
	Orta	0,855	0,078	10,974	0,000***	0,702	1,008
	Yüksek	0,685	0,060	11,389	0,000***	0,567	0,804

Not: b= Standardize edilmemiş regresyon katsayısı, S.H.=Standart Hata, *** $p < 0,001$, ** $p < 0,01$, * $p < 0,05$ LLCI=Alt Limit Güven Aralığı, ULCI=Üst Limit Güven Aralığı

Proje yönetim sürecinin proje başarısına etkisinde çevik pratiklerin farklı değerlerine göre durumsal etkileri incelendiğinde aşağıdaki sonuçlara ulaşılmıştır.

Sosyal pratikleri kullanmanın düşük ve;

- Teknik pratikleri kullanmanın düşük olduğu durumda proje yönetim sürecinin proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,299$; $p < 0,001$).
- Teknik pratikleri kullanmanın orta olduğu durumda proje yönetim sürecinin proje başarısına olan etkisi anlamlı değildir ($b = 0,130$; $p = 0,117$).
- Teknik pratikleri kullanmanın yüksek olduğu durumda proje yönetim sürecinin proje başarısına olan etkisi anlamlı değildir ($b = -0,039$ $p = 0,776$).

Sosyal pratikleri kullanmanın orta ve;

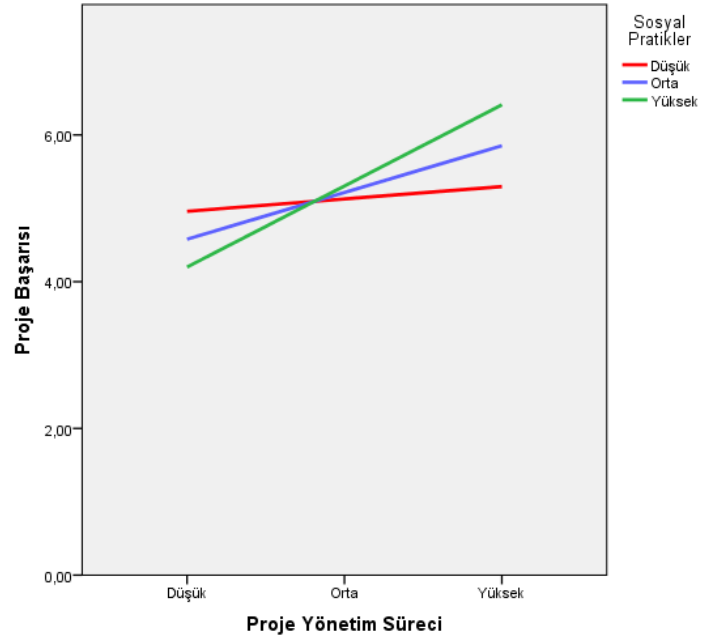
- Teknik pratikleri kullanmanın düşük olduğu durumda proje yönetim sürecinin proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,662$; $p < 0,001$).
- Teknik pratikleri kullanmanın orta olduğu durumda proje yönetim sürecinin proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,492$; $p < 0,001$).
- Teknik pratikleri kullanmanın yüksek olduğu durumda proje yönetim sürecinin proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,323$; $p < 0,001$).

Sosyal pratikleri kullanmanın yüksek ve;

- Teknik pratikleri kullanmanın düşük olduğu durumda proje yönetim sürecinin proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 1,024$; $p < 0,001$).
- Teknik pratikleri kullanmanın orta olduğu durumda proje yönetim sürecinin proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,855$; $p < 0,001$).
- Teknik pratikleri kullanmanın yüksek olduğu durumda proje yönetim sürecinin proje başarısına olan etkisi pozitif ve anlamlıdır ($b = 0,685$; $p < 0,001$).

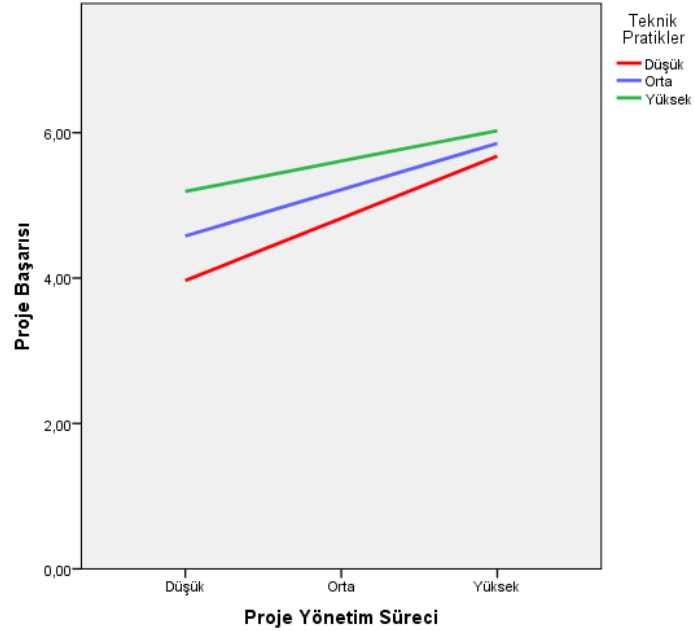
Proje yönetim sürecinin proje başarısına etkisinde çevik pratiklerin düzenleyici rolüne ilişkin H_{13a} , H_{13b} ve H_{13c} hipotezleri kabul edilmiştir.

Proje yönetim sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal pratiklerin düzenleyici rolüne (H_{13a}) ilişkin grafiksel gösterim Şekil 6.14'te yer almaktadır. Şekil 6.14'teki basit eğim grafiği incelendiğinde etkileşimsel etkinin düşük, orta ve yüksek olduğu durumlarda eğim görülmektedir. Grafikte eğimin bulunması düzenleyici etkinin varlığı göstermektedir ve eğim analizindeki bulgularla örtüşmektedir. Ayrıca grafikte en fazla eğimin etkileşimsel etkinin yüksek olduğu durumda olduğu görülmektedir. Eğimin fazla olması etkinin yüksek olmasını göstermektedir (Sürücü vd., 2023).



Şekil 6.14. Proje Yönetim Sürecinin Proje Başarısına Etkisinde Sosyal Pratiklerin Düzenleyici Rolünün Grafıksel Gösterimi

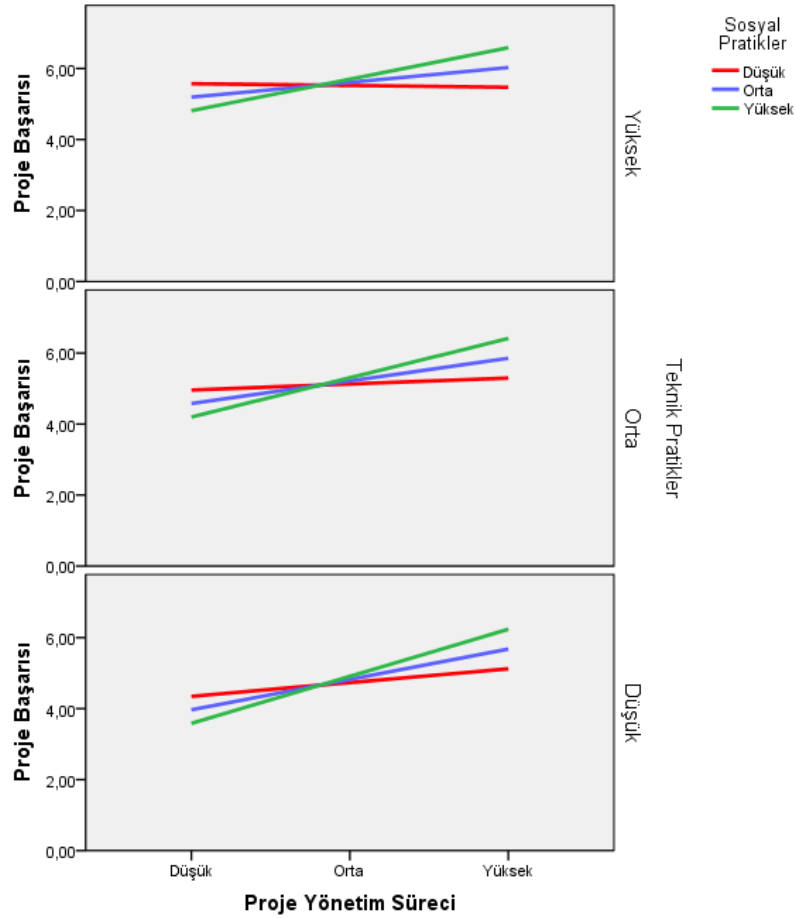
Proje yönetim sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde teknik pratiklerin düzenleyici rolüne (H_{13b}) ilişkin grafıksel gösterim Şekil 6.15'te yer almaktadır.



Şekil 6.15. Proje Yönetim Sürecinin Proje Başarısına Etkisinde Teknik Pratiklerin Düzenleyici Rolünün Grafıksel Gösterimi

Şekil 6.15'teki basit eğim grafiği incelendiğinde etkileşimsel etkinin düşük, orta ve yüksek olduğu durumlarda eğim görülmektedir. Grafikte eğimin bulunması düzenleyici etkinin varlığı göstermektedir ve eğim analizindeki bulgularla örtüşmektedir. Ayrıca grafikte en fazla eğimin etkileşimsel etkinin düşük olduğu durumda olduğu görülmektedir. Eğimin fazla olması etkinin yüksek olmasını göstermektedir (Sürücü vd., 2023).

Proje yönetim sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal ve teknik pratiklerin birlikte düzenleyici rolüne (H_{13c}) ilişkin grafiksel gösterim Şekil 6.16'da yer almaktadır. Grafikte yatay eksen "Proje Yönetim Süreci", dikey eksen "Proje Başarısı" ve düzenleyici değişkenlerin farklı durumlarının (düşük-orta-yüksek) gösterimi yer almaktadır. Eğimin fazla olması etkinin de yüksek olduğunu ifade etmektedir (Sürücü vd., 2023). Grafik incelendiğinde sonuçların eğim (slope) analizindeki bulgularla örtüştüğü görülmektedir.



Şekil 6.16. Proje Yönetim Sürecinin Proje Başarısına Etkisinde Sosyal ve Teknik Pratiklerin Birlikte Düzenleyici Rolünün Grafiksel Gösterimi

6.7.3. Hipotez Test Sonuçlarının Özeti

Çevik yazılım geliştirmede insan ve süreç faktörlerinin proje başarısına etkisinde çevik pratiklerin düzenleyici rolünü inceleme amacıyla gerçekleştirilen araştırmaya ilişkin hipotez test sonuçları Tablo 6.41’de yer almaktadır.

Tablo 6.41. Hipotez Test Sonuçlarının Özeti

Hipotez	Hipotez Sonucu
H₁: Müşteri katılımının proje tanımlama süreci üzerinde anlamlı bir etkisi vardır.	Kabul
H₂: Müşteri katılımının proje yönetim süreci üzerinde anlamlı bir etkisi vardır.	Kabul
H₃: Müşteri katılımının çevik yazılım geliştirme proje başarısı üzerinde anlamlı bir etkisi vardır.	Kabul
H₄: Takım yeteneğinin proje tanımlama süreci üzerinde anlamlı bir etkisi vardır.	Kabul
H₅: Takım yeteneğinin proje yönetim süreci üzerinde anlamlı bir etkisi vardır.	Kabul
H₆: Takım yeteneğinin çevik yazılım geliştirme proje başarısı üzerinde anlamlı bir etkisi vardır.	Kabul
H₇: Proje tanımlama sürecinin proje yönetim süreci üzerinde anlamlı bir etkisi vardır.	Kabul
H₈: Proje tanımlama sürecinin çevik yazılım geliştirme proje başarısı üzerinde anlamlı bir etkisi vardır.	Ret
H₉: Proje yönetim sürecinin çevik yazılım geliştirme proje başarısı üzerinde anlamlı bir etkisi vardır.	Kabul
H₁₀: Müşteri katılımının çevik yazılım geliştirme proje başarısı üzerindeki etkisinde çevik pratiklerin düzenleyici rolü vardır.	Kısmen Kabul
H_{10a}: Müşteri katılımının çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal pratiklerin düzenleyici rolü vardır.	Kabul
H_{10b}: Müşteri katılımının çevik yazılım geliştirme proje başarısı üzerindeki etkisinde teknik pratiklerin düzenleyici rolü vardır.	Ret
H_{10c}: Müşteri katılımının çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal ve teknik pratiklerin birlikte düzenleyici rolü vardır.	Kabul
H₁₁: Takım yeteneğinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde çevik pratiklerin düzenleyici rolü vardır.	Kabul
H_{11a}: Takım yeteneğinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal pratiklerin düzenleyici rolü vardır.	Kabul
H_{11b}: Takım yeteneğinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde teknik pratiklerin düzenleyici rolü vardır.	Kabul
H_{11c}: Takım yeteneğinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal ve teknik pratiklerin birlikte düzenleyici rolü vardır.	Kabul
H₁₂: Proje tanımlama sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde çevik pratiklerin düzenleyici rolü vardır.	Kısmen Kabul
H_{12a}: Proje tanımlama sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal pratiklerin düzenleyici rolü vardır.	Kabul
H_{12b}: Proje tanımlama sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde teknik pratiklerin düzenleyici rolü vardır.	Ret
H_{12c}: Proje tanımlama sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal ve teknik pratiklerin birlikte düzenleyici rolü vardır.	Kabul

Tablo 6.41. Hipotez Test Sonuçlarının Özeti (Devamı)

Hipotez	Hipotez Sonucu
H₁₃: Proje yönetim sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde çevik pratiklerin düzenleyici rolü vardır.	Kabul
H_{13a}: Proje yönetim sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal pratiklerin düzenleyici rolü vardır.	Kabul
H_{13b}: Proje yönetim sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde teknik pratiklerin düzenleyici rolü vardır.	Kabul
H_{13c}: Proje yönetim sürecinin çevik yazılım geliştirme proje başarısı üzerindeki etkisinde sosyal ve teknik pratiklerin birlikte düzenleyici rolü vardır.	Kabul

SONUÇ VE TARTIŞMA

Günümüzde çevik yazılım geliştirme yöntemleri, esneklik, iş birliği ve müşteri odaklılığı teşvik eden yapısıyla yazılım geliştirme süreçlerinde yaygın bir şekilde benimsenmektedir. Bununla birlikte, çevik yöntemlerin geniş çapta uygulanmasına rağmen, çevik projelerde elde edilen başarı oranlarının arzu edilen düzeye ulaşmadığı rapor edilmiştir (Mersino, 2020). Bu durum araştırmacıların çevik yazılım projelerinin kritik başarı faktörlerini inceleme gereksinimini ortaya çıkarmıştır. Literatürde, çevik yazılım projelerinin kritik başarı faktörlerini belirlemek, bu faktörlere dayalı kritik başarı faktörleri modeli oluşturmak ve söz konusu faktörlerin proje başarısına etkisini incelemek amacıyla çeşitli araştırmalar gerçekleştirilmiştir. Çevik yazılım geliştirme projelerinde kritik başarı faktörlerinin proje başarısına etkisini inceleyen çalışmalarda, tüm başarı faktörlerini kapsamlı bir şekilde ele alan araştırmaların yanı sıra yalnızca belirli faktörlere odaklanan çalışmalar da bulunmaktadır. Ayrıca, çevik yazılım geliştirme projelerinde kritik başarı faktörlerinin proje başarısına etkisini düzenleyen değişkenleri ele alan araştırmalar da literatürde yer almaktadır (Litchmore, 2016; Bergmann, 2018; Garousi vd., 2019). Bu araştırmalar, çevik yazılım projelerinde kritik başarı faktörlerinin proje başarısı üzerindeki etkilerini farklı perspektiflerden ele alarak alana önemli katkılar sağlamaktadır.

Çevik yazılım geliştirmede insan ve süreç faktörlerinin proje başarısına katkısı sıkça vurgulanmıştır (Chow ve Cao, 2008; Litchmore, 2016; Stanberry, 2018). Bununla birlikte, insan ve süreç faktörlerinin proje başarısına etkisinde çevik pratiklerin düzenleyici rolünü ele alan çalışmaya literatürde rastlanmamıştır. Çevik pratikler, müşteri ihtiyaçlarına hızlı yanıt verebilme, esneklik sunma ve etkili iletişimi teşvik etme gibi avantajlar sağlamanın yanı sıra, sosyal ve teknik unsurları bir araya getirerek yazılım geliştirme projelerinde karşılaşılan zorlukların üstesinden gelinmesinde kritik bir rol oynamaktadır (Jain ve Suman, 2016; Hummel vd., 2015). Bu doğrultuda, çevik yazılım geliştirme sürecinde insan ve süreç faktörlerinin proje başarısı üzerindeki etkisinde çevik pratiklerin düzenleyici rolünü araştıran bu çalışma, elde edilen bulgularla çevik projelerin yönetiminde hem akademik literatüre hem de sektörel uygulamalara katkılar sunmaktadır.

Bu çalışma, çevik yazılım geliştirme projelerinde insan ve süreç faktörlerinin proje başarısına etkisinde çevik pratiklerin düzenleyici rolünü araştırmak amacıyla

gerçekleştirilmiştir. Araştırma kapsamında insan faktörleri müşteri katılımı ve takım yeteneği olarak ele alınırken, süreç faktörleri proje tanımlama süreci ve proje yönetim süreci olarak değerlendirilmiştir. Proje başarısı, literatürde yaygın olarak kabul edilen kalite, kapsam, zaman ve maliyet değişkenleriyle tanımlanmıştır. Çalışmanın düzenleyici değişkenini oluşturan çevik pratikler, sosyal pratikler ve teknik pratikler olmak üzere iki ana grupta sınıflandırılmıştır. Sosyal pratikler, yinelemeli teslimat (sprint/yineleme planlama), ayaküstü toplantılar ve retrospektifler pratiklerinden; teknik pratikler ise kodlama standartları, birim testi, sürekli entegrasyon ve yeniden kod düzenleme pratiklerinden oluşmaktadır.

Nicel araştırma yöntemlerinden ilişkisel tarama deseniyle gerçekleştirilen çalışmada, veriler literatürde yaygın olarak kullanılan ölçeklerin yer aldığı bir anket formu aracılığıyla toplanmıştır. Anket formu, Chow ve Cao (2008) tarafından geliştirilen “Çevik Başarı Faktörleri Ölçeği” ve “Çevik Yazılım Geliştirme Projesinde Algılanan Başarı Ölçeği” ölçekleri ile Tripp vd. (2016) çalışmasında yer alan “Çevik Pratik Kullanımları” ölçeğini içermektedir. Araştırma kapsamında, İstanbul’da bulunan Teknoloji Geliştirme Bölgelerinde (Teknopark) faaliyet gösteren yazılım geliştirme firmalarında çeşitli rollerde (yazılım geliştirici, test sorumlusu, takım lideri, çevik koç) görev yapan 411 katılımcıdan kolayda örnekleme yöntemiyle toplanmıştır.

Araştırma verilerinin analizinde, tanımlayıcı istatistikler IBM SPSS 22 programı kullanılarak hesaplanmış, araştırma hipotezleri ise AMOS 26 programında yapısal eşitlik modeli analizi ile test edilmiştir. Ayrıca düzenleyici etki analizleri, IBM SPSS 22 programında Hayes (2022) tarafından geliştirilen Process Makro (v4.2) kullanılarak gerçekleştirilmiştir. Analizlerin bulgularından edilen sonuçlar literatürle ilişki olarak sunulmuştur.

Müşteri katılımının proje tanımlama süreci üzerinde anlamlı ve olumlu bir etkisi olduğu tespit edilmiştir. Corvello ve Verteramo (2014), müşterinin yazılım geliştirme süreçlerine aktif katılımının proje kapsamını daha net hale getirdiğini belirtmiştir. Perera ve Perera (2019), müşteri iş birliği ve etkili iletişimin proje tanımlama sürecinde müşteri beklentilerinin daha iyi anlaşılmasını sağladığını vurgulamaktadır. Thompson (2019), müşteri geri bildirimlerinin projelere düzenli olarak entegre edilmesinin, yazılım ürünlerinin müşteri ihtiyaçlarına daha uygun hale gelmesine katkı sağladığını ifade

etmiştir. Henriksen ve Pedersen (2017) ise müşteri taleplerinin proje gereksinimlerinin daha iyi anlaşılmasını katkı sağladığını ve dolayısıyla proje başarısını olumlu yönde etkilediğini belirtmiştir. Literatürdeki çalışmalarla uyumlu olarak, elde edilen sonuçlar müşteri katılımının proje tanımlama sürecine önemli katkılar sağladığını ortaya koymaktadır.

Müşteri katılımının proje yönetim süreci üzerinde anlamlı ve olumlu bir etkisi olduğu sonucuna ulaşılmıştır. Çevik yazılım geliştirme yöntemleri, müşteriye geliştirme sürecinin her aşamasına dahil ederek, kullanıcı hikayeleri yazma, ürün özelliklerini tartışma, özellik (iş) listelerini önceliklendirme ve düzenli geri bildirim sağlama süreçlerinde aktif bir rol üstlenmelerini teşvik etmektedir (Hoda vd., 2011; Barros vd., 2024). Bu katılım, projelerde karşılaşılan belirsizliklerin ve sorunların kısa sürede çözümlenmesini sağlayarak projenin etkili ve doğru bir şekilde ilerlemesine katkılar sunmaktadır (Anderson, 2004; Şimşek ve Ateş, 2022; Barros vd., 2024). Beck ve Andres (2004), müşterinin projeye dahil edilmesinin gereksiz çabaların önüne geçeceğini ve sürekli iyileştirme süreçlerini destekleyeceğini ifade etmiştir. Bununla birlikte, müşteri eksikliğinin kullanılmayan özelliklerin geliştirilmesi, kabul kriterlerini yansıtmayan testlerin tanımlanması ve farklı bakış açılarını birleştirme fırsatının kaybedilmesi gibi riskleri artırabileceği vurgulanmıştır. Xu (2009), müşteri katılımının proje yönetiminde kilit bir rol oynadığını, Aldahmash (2018) ise müşteri katılımının özellikle proje süresi boyunca artan bir öneme sahip olduğunu belirtmiştir. Literatürdeki çalışmalarla benzer şekilde elde edilen sonuçlar, müşteri katılımının proje yönetim sürecine önemli katkılar sağladığını ve projelerin daha etkili bir şekilde yönetilmesine imkân sağladığını göstermektedir.

Müşteri katılımının proje başarısı üzerinde anlamlı ve olumlu bir etkisi olduğu tespit edilmiştir. Literatürdeki çalışmalar, müşteri katılımının çevik yazılım geliştirme projelerinin başarısında kritik bir faktör olduğunu geniş bir şekilde vurgulamaktadır (Misra vd., 2006; Misra vd., 2009; Chow ve Cao, 2008; Ahimbisibwe vd., 2015; Silva ve Santos, 2016; Nguyen, 2016; Aldahmash, 2018; Bergmann ve Karwowski, 2019; Kim vd., 2019; Perera ve Perera, 2019; Thompson, 2019; Shakya ve Shakya, 2020; Tam vd., 2020; Tsoy ve Staples, 2020; Azhar ve Abdullah, 2022; Bogopa ve Marnewick, 2022; Ojha, 2023; Barros vd., 2024; Binboğa ve Gümüşsoy, 2024; Palopak ve Huang, 2024; Zhang vd., 2024). Özellikle, müşteri katılımının çevik yazılım projelerindeki başarının en

önemli kritik başarı faktörlerinden biri olduğunu belirten çalışmalar öne çıkmaktadır (Misra vd., 2006; Aldahmash, 2018; Shakya ve Shakya, 2020; Tsoy ve Staples, 2020; Ojha, 2023). Bu bulgular, müşteri katılımının yalnızca proje süreçlerinin etkinliğini artırmakla kalmayıp, aynı zamanda proje başarısındaki rolünü bir kez daha ortaya koymaktadır. Diğer bir ifadeyle, müşteri katılımı çevik yazılım projelerinde kritik başarı faktörü olduğu sonucuna ulaşılmıştır.

Takım yeteneğinin proje tanımlama süreci üzerinde anlamlı ve olumlu bir etkisi olduğu sonucuna ulaşılmıştır. Proje tanımlama süreci, paydaşların iş birliği içinde projenin hedeflerini, kapsamını ve kısıtlarını netleştirmek amacıyla yürüttüğü kritik bir aşama olarak tanımlanmaktadır (PMI, 2021). Takım yeteneği ise yüksek teknik yeterlilik ve uzmanlık, süreç bilgisi, ekip çalışması ve ürün bilgisi gibi unsurları içeren geniş bir kavramdır (Stankovic vd., 2013; Šmite vd., 2017). Çevik yazılım geliştirme yöntemlerinde, kullanıcı hikayelerinin oluşturulması, detaylandırılması ve tahmin edilmesi (örneğin süre ve maliyet), yineleme/sprint planlama gibi aşamalarda takım yeteneği önemli bir rol oynamaktadır (Cole ve Scotcher, 2015; Jain ve Suman, 2016; Schwaber ve Sutherland, 2020). Literatürdeki bulgular da bu durumu desteklemektedir. Sudhakar (2012) ise takım yeteneğinin proje yönetim süreçlerini (örneğin proje planlaması, net proje hedefleri belirleme vb.) doğrudan etkilediğine dikkat çekmiştir. Ahimbisibwe vd. (2015), takım yeteneğinin riskleri algılama ve bu risklere yanıt verme kapasitesini artırdığını vurgulamıştır. Sonuç olarak, yetenekli bir takımın proje gereksinimlerini doğru bir şekilde analiz etme, belirsizlikleri azaltma ve proje hedeflerini net bir şekilde tanımlama konularında kritik bir rol oynadığını açıkça ortaya koymaktadır.

Takım yeteneğinin proje yönetim süreci üzerinde anlamlı ve olumlu bir etkisi olduğu sonucuna ulaşılmıştır. Literatürde, bu etki farklı açılardan ele alınmaktadır. Misra vd. (2009), yetkin bir ekibin müşteri gereksinimlerini hızlı ve etkili bir şekilde karşılayabileceğini ifade etmiştir. Kurapati vd. (2012), takım yeteneğinin yazılım geliştirme süreçlerinde ekiplerin verimliliğini artırarak, proje yönetim süreçlerinin daha etkin bir şekilde optimize edilmesine katkı sağladığını belirtmiştir. Sudhakar (2012), takım yeteneğinin, proje yönetim faktörlerini doğrudan etkileyen bir unsur olduğunu vurgulamaktadır. Aldahmash (2018), ise takım yeteneğinin proje süresi boyunca artan bir öneme sahip olduğunu belirterek, proje yönetim süreçlerini olumlu yönde etkilediğine dikkat çekmiştir. Literatürdeki çalışmalarla benzer şekilde elde edilen sonuçlar, takım

yeteneğinin proje yönetim sürecine önemli katkılar sağladığını ve projelerin daha etkili bir şekilde yönetilmesine imkân sağladığını göstermektedir.

Takım yeteneğinin proje başarısı üzerinde anlamlı ve olumlu bir etkisi olduğu tespit edilmiştir. Literatürdeki çalışmalar, takım yeteneğinin çevik yazılım geliştirme projelerinin başarısında kritik bir faktör olduğunu sıklıkla vurgulamaktadır (Misra vd., 2006; Chow ve Cao, 2008; Vithana vd., 2015; Litchmore, 2016; Aldahmash, 2018; Stanberry, 2018; Garousi vd., 2019; Shakya ve Shakya, 2020; Tam vd., 2020; Tsoy ve Staples, 2020; Qatanani vd., 2021; Azhar ve Abdullah, 2022; Fadilah vd., 2022; Haskin, 2022; Heper, 2023; Barros vd., 2024; Gwangwadza ve Hanslo, 2024; Zhang vd., 2024). Takım yeteneğinin, çevik yazılım projelerinin başarısında en önemli kritik başarı faktörlerinden biri olduğunu belirten çalışmalar yer almaktadır (Misra vd., 2006; Chow ve Cao, 2008; Aldahmash, 2018; Stanberry, 2018; Shakya ve Shakya, 2020; Tam vd., 2020; Shakya ve Shakya, 2020; Haskin, 2022; Gwangwadza ve Hanslo, 2024). Literatürdeki çalışmalarla benzer şekilde elde edilen sonuçlar elde edilen sonuçlar, takım yeteneğinin proje başarısına sağladığını göstermektedir. Diğer bir ifadeyle takım yeteneği, çevik yazılım projelerinde kritik bir başarı faktörü olarak değerlendirilmektedir.

Proje tanımlama sürecinin, proje yönetim süreci üzerinde anlamlı ve olumlu bir etkisi olduğu sonucuna ulaşılmıştır. Çevik yazılım geliştirme projelerinde, proje tanımlama süreci müşteri ile yakın iş birliği içinde yürütülmekte ve bu sayede ihtiyaçların doğru bir şekilde tanımlanmasına olanak sağlanmaktadır. Aynı zamanda, süreç boyunca ortaya çıkabilecek değişikliklere hızlı bir şekilde uyum sağlanmasını kolaylaştırmaktadır (Beck ve Andres, 2004; Martin, 2014). Kerzner (2017) ise proje tanımlama sürecinin, sonraki tüm proje faaliyetleri için sağlam bir temel oluşturduğunu ve paydaş beklentilerinin etkin bir şekilde yönetilmesinde önemli bir rol oynadığını vurgulamaktadır. Bu bulgular, proje tanımlama sürecinin proje yönetim süreci üzerindeki etkisini ortaya koymaktadır.

Proje tanımlama sürecinin, proje başarısı üzerinde anlamlı ve olumlu bir etkisi olmadığı sonucuna ulaşılmıştır. Literatürde, proje tanımlama sürecinin çevik yazılım geliştirme projelerinde kritik bir başarı faktörü olduğunu belirten çeşitli çalışmalar bulunmaktadır (Sudhakar vd., 2012; Stankovic vd., 2013; Brown, 2015; Serrador ve Pinto, 2015; Litchmore, 2016; Stanberry, 2018; Haskin, 2022). Araştırma sonucu ilgili

alışmalarla eliřmektedir ve proje tanımlama sürecinin proje başarısı üzerindeki etkisinin sınırlı olabileceğini gstermektedir. Bununla birlikte, alışmamızla benzer sonuçlara ulaşan arařtırmalar da literatrde yer almaktadır (Chow ve Cao, 2008; Tsoy ve Staples, 2020). Bu durum, evik yazılım projelerinde deėiřen gereksinimlerle iliřkili olabilir. evik yaklařımlarda, projenin bařlangı ařamasında tanımlanan gereksinimlerin zamanla yeniden řekillenmesi, proje tanımlama sürecinin proje başarısı üzerindeki etkisini azaltmıř olabilir.

Proje ynetim sürecinin, proje başarısı üzerinde anlamlı ve olumlu bir etkisi olduėu tespit edilmiřtir. Literatrdeki alışmalar, proje ynetim sürecinin evik yazılım geliřtirme projelerinin başarısında kritik bir faktr olduėunu vurgulamakta ve alışmamızın bulgularıyla rtřmektedir (Chow ve Cao, 2008; Sudhakar, 2012; Stankovic vd., 2013; Litchmore, 2016; Nguyen, 2016; Aldahmash, 2018; Stanberry, 2018; Ergder, 2019; Garousi vd., 2019; Kim vd., 2019; Shakya ve Shakya, 2020; Tsoy ve Staples, 2020; Haskin, 2022; Ojha, 2023; Binboėa ve Gmřsoy, 2024). Aldahmash (2018), evik yazılım geliřtirme projelerinde proje ynetim sürecinin kritik bir bařarı faktr olduėunu ifade etmiř, ancak proje ilerledike bu sürecin greceli neminin azaldığını belirtmiřtir. Tsoy ve Staples (2020) ise proje ynetim sürecinin, zellikle bařarılı projelerde daha belirgin bir řekilde etkili olduėunu vurgulamıřtır. Bu bulgular, proje ynetim sürecinin hem bařlangı hem de ilerleyen ařamalardaki rolnn dikkatle ele alınması gerektiğini gstermektedir.

evik pratiklerin insan ve sre faktrlerinin proje başarısına etkisindeki dzenleyici rolne iliřkin arařtırma sonuçları ařaėıda sunulmaktadır. Literatrde evik pratikler genellikle sosyal ve teknik olmak zere iki kategoride ele alınmaktadır (Gupta vd., 2019). Sosyal pratikler, ekip ii iletiřim, iř birliėi ve mřteriyle etkileřim gibi sosyal etkileřimi kolaylařtıran uygulamaları kapsamaktadır (So ve Scholl, 2009; Hummel vd., 2015; Gupta vd., 2019). Teknik pratikler ise kodlama ve test srelerine odaklanan yazılım mhendisliėi uygulamaları olarak tanımlanmaktadır. Yazılımın kalitesini artırmaya, srdrlebilirliğini saėlamaya ve hata oranını azaltmaya ynelik ara ve yntemleri iermektedir (Maruping vd., 2009; Gupta vd., 2019; Klotins vd., 2021). Arařtırmada sosyal pratikler, yinelemeli teslimat (sprint/yineleme planlama), ayakst toplantılar ve retrospektifler pratiklerinden; teknik pratikler ise kodlama standartları, birim testi, srekli entegrasyon ve yeniden kod dzenleme pratiklerinden oluřmaktadır.

Çevik pratiklerin düzenleyici rolüne ilişkin araştırma sonuçları yalnızca sosyal pratiklerin, yalnızca teknik pratiklerin ve sosyal ile teknik pratiklerin birlikte kullanıldığı (“çevik pratikler” olarak ifade edilecektir) durumlar temelinde değerlendirmiştir.

Müşteri katılımının proje başarısı üzerindeki etkisinin sosyal pratiklerin kullanım seviyesiyle anlamlı bir şekilde farklılaştığını sonucuna ulaşılmıştır. Sosyal pratiklerin kullanımının düşük seviyede olduğu durumlarda müşteri katılımının proje başarısına katkısı sınırlı kalmakta, buna karşın sosyal pratiklerin kullanımının yüksek seviyelerde olduğu durumlarda bu etki belirgin bir şekilde artmaktadır. Bu sonuçlar, sosyal pratiklerin müşteri katılımı ile proje başarısı arasındaki ilişkiyi güçlendiren önemli bir düzenleyici değişken olduğunu göstermektedir. Literatürdeki önceki çalışmalar da elde edilen sonuçları desteklemektedir. Misra vd. (2006), çevik yazılım geliştirme projelerinde iletişimin proje başarısı üzerindeki kritik rolünü vurgulamıştır. Siddique ve Hussein (2016), çevik projelerde müşteri ile yakın iş birliğinin proje başarısını etkilediğini belirtmiştir. Aldahmash (2018) ve Syeda (2018) çalışmalarında, iletişim ve iş birliğinin proje başarısında anahtar bir role sahip olduğunu ortaya koymuşlardır. Zhang vd. (2024), aktif iletişimin çevik projelerdeki başarı için vazgeçilmez bir unsur olduğunu ifade etmiştir. Henriksen ve Pedersen (2017), yinelemeli teslimat ve retrospektif gibi çevik pratiklerin müşteri ihtiyaçlarına daha uygun çözümler geliştirilmesine katkı sağladığını ve proje başarısını olumlu yönde etkilediğini belirtmiştir. Pikkarainen vd. (2008), çevik pratiklerin hem ekip içi hem de dış paydaşlarla iletişimi geliştirdiğini; Hummel vd. (2015) ise sosyal pratiklerin doğrudan iletişim üzerindeki olumlu etkilerini vurgulamıştır. Bu bulgular ışığında, müşteri katılımının proje başarısına olan olumlu katkısını üst düzeye çıkarabilmek için sosyal pratiklerin etkin bir şekilde kullanılması ve proje süreçlerinde önceliklendirilmesi önerilmektedir.

Müşteri katılımının proje başarısı üzerindeki etkisinin teknik pratiklerin kullanım düzeylerine göre anlamlı bir farklılık göstermediği sonucuna ulaşılmıştır. Bu bulgu, teknik pratiklerin müşteri katılımı ile proje başarısı arasındaki ilişkide düzenleyici bir rol üstlenmediğini ortaya koymaktadır. Teknik pratikler, kodlama standartları, birim testi ve sürekli entegrasyon gibi yazılım mühendisliği süreçlerine odaklanan uygulamaları içermektedir (Gupta vd., 2019). Bu durum, teknik pratiklerin genellikle yazılım geliştirme ekibi tarafından iç süreçlerde kullanılmasından ve müşteriyle doğrudan etkileşime daha az odaklanmasından kaynaklanıyor olabilir. Teknik pratiklerin müşteri katılımı

üzerindeki sınırlı etkisinin, proje başarısına dolaylı bir etki mekanizması oluşturabileceği değerlendirilmektedir.

Müşteri katılımının proje başarısı üzerindeki etkisinin çevik pratiklerin (sosyal ve teknik birlikte) kullanım seviyesiyle anlamlı bir şekilde farklılaştığını sonucuna ulaşılmıştır. Çevik pratiklerin kullanımının düşük seviyede olduğu durumlarda müşteri katılımının proje başarısına katkısı sınırlı kalmakta, yüksek seviyelerde olduğu durumlarda bu etki belirgin bir şekilde artmaktadır. Bu sonuçlar, çevik pratiklerin kullanımının müşteri katılımı ile proje başarısı arasındaki ilişkiyi güçlendiren önemli bir düzenleyici değişken olduğunu göstermektedir. Ayrıca, sosyal pratiklerin düzenleyici etkisine kıyasla, sosyal ve teknik pratiklerin birlikte kullanımının daha güçlü bir düzenleyici rol oynadığı belirlenmiştir. Çevik pratiklerin etkinliğinin, uygulandığı bağlama ve yazılım geliştirme süreçlerine nasıl entegre edildiğine bağlı olduğu literatürde ifade edilmektedir (Abrantes ve Travassos, 2011). Pratiklerin yalnızca bireysel düzeyde uygulanmasının beklenen faydayı sağlamadığı, bunun yerine çevik pratiklerin birbirleriyle uyumlu bir şekilde entegre edilmesinin daha büyük bir bütün oluşturarak yazılım geliştirme süreçlerinin başarısını artırdığı belirtilmektedir (Beck ve Andres, 2004; Martin, 2014). Literatürle desteklenen bu sonuçlar, müşteri katılımının proje başarısına olan etkisini artırmak için çevik pratiklerin bütüncül ve entegre bir şekilde uygulanmasının önemini vurgulamaktadır.

Takım yeteneğinin proje başarısı üzerindeki etkisinin sosyal pratiklerin kullanım seviyesiyle anlamlı bir şekilde farklılaştığını sonucuna ulaşılmıştır. Sosyal pratiklerin kullanımının düşük seviyede olduğu durumlarda takım yeteneğinin proje başarısına katkısı sınırlı kalmakta, buna karşın sosyal pratiklerin kullanımının yüksek seviyelerde olduğu durumlarda bu etki belirgin bir şekilde artmaktadır. Bu sonuçlar, sosyal pratiklerin takım yeteneği ile proje başarısı arasındaki ilişkiyi güçlendiren önemli bir düzenleyici değişken olduğunu göstermektedir. Literatürdeki önceki çalışmalar da elde edilen sonuçları desteklemektedir. Hummel vd. (2015) sosyal pratiklerin iletişim üzerindeki olumlu etkileri olduğunu belirtmiştir. Ghimire ve Charters (2022) ayakta toplantılar ve retrospektiflerin ekip içi iletişim sorunlarını azalttığını vurgulamıştır. Henriksen ve Pedersen (2017), günlük ayakta toplantıların ekip içi iletişim ve şeffaflığı artırdığını, Sandstø ve Reme-Ness (2021) ise retrospektif ve ayakta toplantıların ekip motivasyonu ve güvenini güçlendirdiğini ifade etmiştir. Qatanani vd. (2021) motivasyonun çevik

projelerdeki başarı üzerinde kritik bir faktör olduğunu vurgulamıştır. Arcos-Medina ve Mauricio (2020), ayakta toplantılar ve retrospektiflerin takım çalışmasını geliştirdiğini, Al-Sakkaf vd. (2017) ise bu pratiklerin takım iletişimini güçlendirdiğini ve yaygın olarak benimsendiğini belirtmiştir. Jain ve Suman (2016), sprint planlama ve ayaküstü toplantılar gibi pratiklerin ekip koordinasyonunu artırdığını ifade etmiştir. Sosyal pratiklerin etkili kullanımı, ekip içi iletişim, motivasyon, şeffaflık ve güveni artırarak takım yeteneğinin proje başarısına olan katkısını güçlendirecektir. Takım yeteneğinin proje başarısına olan olumlu katkısını en üst düzeye çıkarabilmek için sosyal pratiklerin etkin bir şekilde kullanılması ve proje yönetim süreçlerinde önceliklendirilmesi önerilmektedir.

Takım yeteneğinin proje başarısı üzerindeki etkisinin, teknik pratiklerin kullanım seviyesiyle anlamlı bir şekilde farklılaştığını ortaya koymaktadır. Teknik pratiklerin düşük düzeyde uygulandığı durumlarda, takım yeteneğinin proje başarısına olan katkısı sınırlı kalırken, yüksek düzeyde uygulandığında bu etkinin belirgin bir artış gösterdiği tespit edilmiştir. Bu sonuçlar, teknik pratiklerin takım yeteneği ile proje başarısı arasındaki ilişkiyi güçlendiren önemli bir düzenleyici değişken olduğunu göstermektedir. Literatürde yer alan çalışmalar da bu bulguları desteklemektedir. Klotins vd. (2021), yeniden kod düzenleme, birim test ve sürekli entegrasyon gibi teknik pratiklerin kaynak kod ve ürün kalitesi üzerinde doğrudan olumlu etkileri olduğunu, bunun yanı sıra ekip performansını artırarak kaynakların daha verimli kullanılmasını sağladığını belirtmiştir. Maruping vd. (2009), sürekli entegrasyon ve test odaklı geliştirme (birim test) pratiklerinin proje kalitesini artırdığını vurgulamıştır. Pantiuchina vd. (2017), yeniden kod düzenlemeyi kalite odaklı pratikler arasında değerlendirmiştir. Sun ve Schmidt (2018), sürekli entegrasyon, yeniden kod düzenleme, birim test ve kodlama standartlarının yüksek düzeyde uygulanmasının, iş memnuniyetini artırdığı ve iş tükenmişliği gibi olumsuz etkileri azalttığını ifade etmiştir. Sonuçlar göre teknik pratiklerin yüksek düzeyde kullanılması, yalnızca proje başarısını desteklemekle kalmayıp, aynı zamanda ekiplerin performansını ve memnuniyetini artıran bir mekanizma olarak işlev görmektedir. Dolayısıyla takım yeteneğinin proje başarısına olan olumlu katkısını üst düzeye çıkarabilmek için teknik pratiklerin proje yönetim süreçlerine sistematik ve entegre bir şekilde dahil edilerek kullanılması önerilmektedir.

Takım yeteneğinin proje başarısı üzerindeki etkisinin çevik pratiklerin (sosyal ve teknik birlikte) kullanım seviyesiyle anlamlı bir şekilde farklılaştığını sonucuna ulaşılmıştır. Çevik pratiklerin kullanımının düşük seviyede olduğu durumlarda takım yeteneğinin proje başarısına katkısı sınırlı kalmakta, yüksek seviyelerde olduğu durumlarda bu etki belirgin bir şekilde artmaktadır. Bu sonuçlar, çevik pratiklerin kullanımının takım yeteneği ile proje başarısı arasındaki ilişkiyi güçlendiren önemli bir düzenleyici değişken olduğunu göstermektedir. Ayrıca, sosyal pratiklerin düzenleyici etkisine kıyasla, sosyal ve teknik pratiklerin birlikte kullanımının daha güçlü bir düzenleyici rol oynadığı belirlenmiştir. Söz konusu durum teknik pratikler içinde geçerlidir. Literatürde yer alan çalışmalar da elde edilen sonucu desteklemektedir. Çevik pratiklerin etkinliğinin, uygulandığı duruma ve yazılım geliştirme süreçlerine nasıl entegre edildiğine bağlı olduğu literatürde ifade edilmektedir (Abrantes ve Travassos, 2011). Pratiklerin yalnızca bireysel düzeyde uygulanmasının beklenen faydayı sağlamadığı, bunun yerine çevik pratiklerin birbirleriyle uyumlu bir şekilde entegre edilmesinin daha büyük bir bütün oluşturarak yazılım geliştirme süreçlerinin başarısını artırdığı belirtilmektedir (Beck ve Andres, 2004; Martin, 2014). Tripp vd. (2016), çevik pratiklerin ekipler arasında iş tatminini artırdığını belirtmiştir. Leal vd. (2019) ise çevik proje başarısını etkileyen önemli bir faktör olarak doğru araç ve pratiklerin seçimine dikkat çekmiştir. Literatürle desteklenen bu sonuçlar, takım yeteneğinin proje başarısına olan etkisini güçlendirmek için çevik pratiklerin bütüncül ve entegre bir şekilde uygulanmasının önemini vurgulamaktadır.

Proje tanımlama sürecinin proje başarısı üzerindeki etkisinin sosyal pratiklerin kullanım seviyesiyle anlamlı bir şekilde farklılaştığını sonucuna ulaşılmıştır. Sosyal pratiklerin kullanımının düşük seviyede olduğu durumlarda proje tanımlama sürecinin proje başarısına katkısı sınırlı kalmakta, buna karşın sosyal pratiklerin kullanımının yüksek seviyelerde olduğu durumlarda bu etki belirgin bir şekilde artmaktadır. Bu sonuçlar, sosyal pratiklerin proje tanımlama süreci ile proje başarısı arasındaki ilişkiyi güçlendiren önemli bir düzenleyici değişken olduğunu göstermektedir. Proje tanımlama süreci, paydaşlar arasında ortak bir anlayış oluşturmayı ve projenin hedeflerini, kapsamını, risklerini ve kısıtlarını netleştirmeyi amaçlamaktadır (PMI, 2021; Kerzner, 2017). Literatürdeki çalışmalar da bu sürecin sosyal pratiklerle desteklenmesinin önemini vurgulamaktadır. Ghimire ve Charters (2022), çevik pratiklerin müşteri gereksinimlerinin

yanı sıra proje gereksinimleri ve önceliklerinin de daha net tanımlanmasını sağladığını tespit etmiştir. Pikkarainen vd. (2008), çevik pratiklerin hem ekip içi iletişimi hem de yazılım ekibi ile diğer paydaşlar arasındaki iletişimi geliştirdiğini ifade etmiştir. Sosyal pratiklerin sistematik ve bütüncül bir yaklaşımla kullanılması, proje gereksinimlerinin daha iyi anlaşılmasını, paydaşlar arasındaki iletişimin güçlendirilmesini ve proje hedeflerinin daha net bir şekilde tanımlanmasını sağlayarak proje başarısına katkıda bulunacaktır. Sonuç olarak, proje tanımlama sürecinin proje başarısına olan olumlu katkısını en üst düzeye çıkarabilmek için sosyal pratiklerin etkin bir şekilde uygulanması önerilmektedir.

Proje tanımlama sürecinin proje başarısı üzerindeki etkisinin teknik pratiklerin kullanım düzeylerine göre anlamlı bir farklılık göstermediği sonucuna ulaşılmıştır. Diğer bir ifadeyle, teknik pratiklerin proje tanımlama süreci ile proje başarısı arasındaki ilişkide düzenleyici bir rol üstlenmediğini ortaya koymaktadır. Bu durum, teknik pratiklerin genellikle yazılım geliştirme ekibi tarafından iç süreçlerde kullanılmasından ve müşteriyle doğrudan etkileşime daha az odaklanmasından kaynaklanıyor olabilir. Teknik pratiklerin müşteri katılımı üzerindeki sınırlı etkisinin, proje başarısına dolaylı bir etki mekanizması oluşturabileceği değerlendirilmektedir. Teknik pratiklerin proje tanımlama sürecinden ziyade proje yürütme ve teknik gereksinimlerin yerine getirilmesinde daha etkili olduğunu göstermektedir. Bu durum, teknik pratiklerin daha çok proje sürecinin ilerleyen aşamalarında kaliteyi artırma ve teknik riskleri yönetme amacıyla kullanıldığına işaret eden literatürü desteklemektedir (Maruping vd., 2009; Abrantes ve Travassos, 2011; Gupta vd., 2019; Klotins vd., 2021).

Proje tanımlama sürecinin proje başarısı üzerindeki etkisinin çevik pratiklerin kullanım seviyesiyle anlamlı bir şekilde farklılaştığını sonucuna ulaşılmıştır. Çevik pratiklerin kullanımının düşük seviyede olduğu durumlarda proje tanımlama sürecinin proje başarısına katkısı sınırlı kalmakta, yüksek seviyelerde olduğu durumlarda bu etki belirgin bir şekilde artmaktadır. Bu sonuçlar, çevik pratiklerin kullanımının proje tanımlama süreci ile proje başarısı arasındaki ilişkiyi güçlendiren düzenleyici değişken olduğunu göstermektedir. Ayrıca sosyal pratiklerin düzenleyici etkisine kıyasla, sosyal ve teknik pratiklerin birlikte kullanımının daha güçlü bir düzenleyici rol oynadığı belirlenmiştir. Literatürdeki mevcut çalışmalar da bu bulguları desteklemektedir. Ghimire ve Charters (2022), proje gereksinimleri ve öncelikleri üzerinde daha fazla etki

yaratabilecek yaygın çevik uygulamalar arasında ayaküstü toplantılar, birim testi, retrospektifler ve yinelemeli teslimat gibi pratikleri vurgulamaktadır. Kurapati vd. (2012), çevik pratiklerin bağlama özgü bir şekilde yeniden düzenlenebileceğini ve projenin gereksinimlerine uygun hale getirilebileceğini ifade etmektedir. Literatürle desteklenen bu sonuçlar, proje tanımlama sürecinin proje başarısına olan etkisini güçlendirmek için çevik pratiklerin bütüncül olarak kullanılmasının önemini vurgulamaktadır.

Proje yönetim sürecinin proje başarısı üzerindeki etkisinin sosyal pratiklerin kullanım seviyesiyle anlamlı bir şekilde farklılaştığını sonucuna ulaşılmıştır. Sosyal pratiklerin kullanımının düşük seviyede olduğu durumlarda proje yönetim sürecinin proje başarısına katkısı sınırlı kalmakta, buna karşın sosyal pratiklerin kullanımının yüksek seviyelerde olduğu durumlarda bu etki belirgin bir şekilde artmaktadır. Bu sonuçlar, sosyal pratiklerin proje yönetim süreci ile proje başarısı arasındaki ilişkiyi güçlendiren düzenleyici değişken olduğunu göstermektedir. Literatürdeki çalışmalar da bu bulguları desteklemektedir. Aldahmash (2018) ve Syeda (2018), iletişim ve iş birliğinin proje başarısındaki kritik önemine dikkat çekmiştir. Benzer şekilde Pikkarainen vd. (2008) ise çevik pratiklerin hem ekip içi hem de dış paydaşlarla iletişimi geliştirdiğini dile getirmiştir. Henriksen ve Pedersen (2017), yinelemeli teslimat (sprint planlama) ve retrospektif gibi sosyal pratiklerin proje kapsamını yönetmeye ve müşteri ihtiyaçlarına daha uygun çözümler üretmeye yardımcı olduğunu belirtmiştir. Recker vd. (2017), ayaküstü toplantıların müşteri taleplerine yanıt verme verimliliğini artırdığını ifade etmiştir. Jain ve Suman (2016), sprint planlama ve ayaküstü toplantıların, proje ilerlemesini izlemek ve sorunların erken çözülmesini sağlamak açısından etkili olduğunu vurgulamıştır. Bu bulgular, proje yönetim sürecinin proje başarısına olan olumlu etkisini en üst düzeye çıkarabilmek için sosyal pratiklerin etkin bir şekilde kullanılmasının önemini ortaya koymaktadır.

Proje yönetim sürecinin proje başarısı üzerindeki etkisinin, teknik pratiklerin kullanım seviyesiyle anlamlı bir şekilde farklılaştığını ortaya koymaktadır. Teknik pratiklerin düşük düzeyde uygulandığı durumlarda, proje yönetiminin proje başarısına olan katkısı sınırlı kalırken, yüksek düzeyde uygulandığında bu etkinin belirgin bir artış gösterdiği tespit edilmiştir. Bu sonuçlar, teknik pratiklerin proje yönetim süreci ile proje başarısı arasındaki ilişkiyi güçlendiren düzenleyici değişken olduğunu göstermektedir. Literatürde yer alan çalışmalar da bu bulguları desteklemektedir. Misra vd. (2006), teknik

süreçlerin çevik yazılım geliştirme projelerindeki başarıda önemli bir rol oynadığını belirtmiştir. Gupta vd. (2019), teknik pratiklerin yazılım geliştirme sürecinin teknik yönlerini optimize ederek proje yönetim süreçlerine dolaylı katkı sağladığını ifade etmiştir. Abrantes ve Travassos (2011) ise sürekli entegrasyon, ortak kod sahipliği, yeniden kod düzenleme, basit tasarım ve kodlama standartları gibi teknik pratiklerin yazılım süreçlerinde yaygın olarak uygulandığını ve bu pratiklerin yazılım geliştirme süreçlerinin iyileştirilmesinde önemli bir rol oynadığını vurgulamıştır. Sonuç olarak, teknik pratiklerin yüksek düzeyde uygulanması hem proje yönetim sürecinin etkinliğini artırmakta hem de proje başarı oranlarını yükseltmektedir. Dolayısıyla proje yönetim süreçlerinin etkinliğini artırmak için teknik pratiklerin sistematik bir şekilde uygulanması önem göstermektedir.

Proje yönetim sürecinin proje başarısı üzerindeki etkisinin çevik pratiklerin kullanım seviyesiyle anlamlı bir şekilde farklılaştığını sonucuna ulaşılmıştır. Çevik pratiklerin kullanımının düşük seviyede olduğu durumlarda proje yönetim sürecinin proje başarısına katkısı sınırlı kalmakta, yüksek seviyelerde olduğu durumlarda bu etki belirgin bir şekilde artmaktadır. Bu sonuçlar, çevik pratiklerin kullanımının proje yönetim süreci ile proje başarısı arasındaki ilişkiyi güçlendiren düzenleyici değişken olduğunu göstermektedir. Ayrıca sosyal pratiklerin düzenleyici etkisine kıyasla, sosyal ve teknik pratiklerin birlikte kullanımının daha güçlü bir düzenleyici rol oynadığı belirlenmiştir. Benzer sonuca yalnızca teknik pratiklerin kullanılması durumunda da ulaşılmıştır. Literatürde yer alan çalışmalar bu bulguları desteklemektedir. Licorish vd. (2016), kodlama standartları, yeniden kod düzenleme, sürekli entegrasyon ve yineleme planlaması gibi çevik pratiklerin yaygın olarak benimsendiğini ve bu pratiklerin proje yönetim süreçlerini desteklediğini ifade etmiştir. Kurapati vd. (2012), ayaküstü toplantılar, sprint/yineleme, sürekli entegrasyon, testler, sprint planlama, ortak kod sahipliği ve retrospektif gibi pratiklerin proje yönetim süreçlerini optimize etme amacıyla yaygın olarak kullanıldığını belirtmiştir. Tripp vd. (2016) ise çevik pratiklerin geri bildirim süreçlerini iyileştirdiğini vurgulamıştır. Literatürle desteklenen bu sonuçlar, proje yönetim sürecinin proje başarısına olan etkisini güçlendirmek için çevik pratiklerin bütüncül olarak kullanılmasının önemini vurgulamaktadır.

Sonuç olarak, bu çalışma, çevik yazılım projelerinin başarısını artırmada insan ve süreç faktörleri ile çevik pratiklerin bütünsel bir yaklaşımla ele alınmasının önemini

vurgulamaktadır. Elde edilen sonuçlar hem literatüre hem de yazılım geliştirme sektörüne rehberlik edebilecek değerli çıkarımlar sunarak, çevik projelerin etkinliğini artırmaya yönelik stratejik bir yaklaşım sağlamaktadır.

ÖNERİLER

Çevik yazılım geliştirmede insan ve süreç faktörlerinin proje başarısına etkisinde çevik pratiklerin düzenleyici rolüne araştırma amacıyla gerçekleştirilen çalışmanın sonuçlarına göre akademik çalışmalara ve uygulamaya yönelik önerilere yer verilmiştir.

Akademik Çalışmalara Yönelik Öneriler

1. *Araştırma Kapsamının Genişletilmesi:* Çalışma, İstanbul’da faaliyet gösteren teknoparklardaki yazılım geliştirme firmalarını kapsamaktadır. Gelecek araştırmalarda farklı bölgelerdeki teknoparklar ya da ticaret odalarına kayıtlı yazılım firmaları üzerinde benzer çalışmalar gerçekleştirilebilir.
2. *Sektörel Odağın Derinleştirilmesi:* Katılımcılar, çeşitli sektörlerde faaliyet gösteren yazılım geliştirme firmalarından seçilmiştir. Belirli bir sektöre odaklanan çalışmalar yapılarak sektöre özgü dinamiklerin proje başarısına etkileri daha ayrıntılı olarak incelenebilir.
3. *Yazılım Proje Yönetim Yaklaşımlarının Karşılaştırılması:* Çalışmada çevik yazılım geliştirme yöntemleri ele alınmıştır. Gelecekteki araştırmalarda çevik ve geleneksel proje yönetim yaklaşımlarının insan ve süreç faktörleri üzerindeki etkileri karşılaştırılabilir. Ayrıca, hibrit proje yönetimi uygulamalarının proje başarısı üzerindeki etkisinin araştırılması yararlı olabilir.
4. *Çevik Yöntem Temelli İncelemeler:* Katılımcıların çoğunun Scrum metodolojisini kullandığı gözlemlenmiştir. Gelecek araştırmalarda belirli çevik yazılım geliştirme yöntemlerine odaklanarak daha derinlemesine incelemeler yapılabilir.
5. *Çevik Başarı Faktörlerinin Genişletilmesi:* Çalışmada insan ve süreç faktörleri ele alınmıştır. Gelecekteki araştırmalar, proje başarısına etki eden diğer faktörleri (örneğin, organizasyon, proje, teknik) içerecek şekilde kapsamını genişletebilir.
6. *Çeşitli Düzenleyici Değişkenlerin İncelenmesi:* Çalışmada çevik pratiklerin düzenleyici rolü araştırılmıştır. Gelecek çalışmalar, farklı çevik pratiklerin veya

proje karmaşıklığı, proje büyüklüğü, takım ortalama yaşı gibi değişkenlerin düzenleyici etkilerini inceleyebilir.

7. *Zamansal Analizler:* Çalışma, kesitsel bir araştırma yöntemiyle gerçekleştirilmiştir. Gelecekte farklı zaman dilimlerinde veri toplanarak boylamsal analizler yapılabilir ve proje dinamiklerindeki değişimler daha kapsamlı şekilde değerlendirilebilir.
8. *Araştırma Yöntemlerinin Çeşitlendirilmesi:* Araştırma nicel yöntemlerle gerçekleştirilmiştir. Gelecekteki çalışmalar, nitel veya karma yöntemler kullanılarak daha kapsamlı analizler sunabilir ve derinlemesine sonuçlar sağlayabilir.

Uygulamaya Yönelik Öneriler

1. *Takım Yeteneğinin Güçlendirilmesi:* Araştırma bulguları, çevik yazılım geliştirme projelerinde takım yeteneğinin proje tanımlama, proje yönetimi ve proje başarısı üzerindeki kritik rolünü vurgulamaktadır. Bu doğrultuda, çevik projelerde yalnızca yetenekli ekipler oluşturmakla kalınmamalı, aynı zamanda çevik yöntemler, pratikler ve teknik araçlar konusunda düzenli eğitim programları uygulanmalıdır.
2. *Müşteri Katılımının Desteklenmesi:* Araştırmada müşteri katılımının çevik projelerdeki önemi açıkça ortaya konulmuştur. Bu bağlamda, müşteriyle iş birliği ve iletişim düzenli olarak sürdürülmeli ve müşteri katılımını kolaylaştıracak iş birliği araçları etkin bir şekilde kullanılmalıdır.
3. *Proje Tanımlama Sürecinin İyileştirilmesi:* Proje tanımlama sürecinin proje yönetimi sürecini doğrudan etkilediği bulgusu doğrultusunda, proje tanımlama süreci müşteri katılımı ve çevik pratiklerle desteklenerek daha etkin hale getirilmelidir. Bu süreç, projenin temel hedeflerinin netleştirilmesini sağlayacak şekilde yapılandırılmalıdır.
4. *Proje Yönetim Sürecinin Güçlendirilmesi:* Çevik yazılım geliştirme projelerinde proje yönetimi sürecinin proje başarısı üzerindeki etkisi dikkate alınarak, proje yönetimini destekleyen bilgilendirici ve iş birliğini artırıcı çalışma alanları oluşturulmalıdır. Proje yönetimi süreci, müşteriyle sürekli etkileşim içinde yürütülmelidir.

5. *Sosyal Pratiklerin Güçlendirilmesi:* Sosyal pratiklerin insan ve süreç faktörleri üzerindeki düzenleyici etkisi göz önüne alındığında, ekip içi iletişim, iş birliği ve geri bildirim mekanizmalarına öncelik verilmelidir. Planlama toplantıları, ayaküstü toplantılar, retrospektifler ve müşteri geri bildirim süreçleri düzenli ve yapılandırılmış bir şekilde yürütülmelidir.
6. *Teknik Pratiklerin Uygulanması:* Teknik pratiklerin takım yeteneği ve proje yönetimi süreci üzerindeki düzenleyici etkisi dikkate alınarak, yazılım ve test pratikleri süreçlere entegre edilmelidir. Yeniden kod düzenleme, sürekli entegrasyon, birim test ve kodlama standartları gibi teknik pratikler düzenli olarak uygulanmalı ve yazılımın kalite ve sürdürülebilirliği sağlanmalıdır.
7. *Pratiklerin Entegrasyonu:* Teknik ve sosyal pratiklerin birlikte kullanılması, tanımlayıcı etkiler yaratarak süreçlerin etkinliğini artırabilir. Bu nedenle uygulayıcılar, bu iki pratik grubunu entegre ederek süreçlerini optimize etmeli ve organizasyonel verimliliği artırmalıdır.

KAYNAKÇA

- Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). *Agile software development methods: Review and analysis* (VTT Publications). VTT Technical Research Centre of Finland.
- Abrantes, J. F., & Travassos, G. H. (2011). Common agile practices in software processes: A systematic review. *Proceedings of the 2011 International Symposium on Empirical Software Engineering and Measurement* (pp. 355-358). <https://doi.org/10.1109/ESEM.2011.47>
- Aggarwal, K. K., & Singh, Y. (2005). *Software Engineering: Programs, Documentation, Operating Procedures* (Revised second edition). New Age International Publishers.
- Agile Alliance. (2001a). *History: The Agile manifesto*. Agile Manifesto. <https://agilemanifesto.org/history.html>
- Agile Alliance. (2001b). *Manifesto for Agile Software Development*. Agile Manifesto. <https://agilemanifesto.org/>
- Agile Alliance. (2001c). *Principles behind the Agile manifesto*. Agile Manifesto. <https://agilemanifesto.org/principles.html>
- Aini, Q., Budiarto, M., Putra, P. O. H., & Santoso, N. P. L. (2021). Gamification-based the Kampus Merdeka learning in 4.0 era. *Indonesian Journal of Computing and Cybernetics Systems*, 15(1), 31–42. <https://doi.org/10.22146/ijccs.59023>
- Aitken, A., & Ilango, V. (2013, January). A comparative analysis of traditional software engineering and agile software development. *46th Hawaii International Conference on System Sciences*, 4749-4760. <https://doi.org/10.1109/HICSS.2013.31>
- Akalın, M. (2018). *Örnek açıklamalarıyla sosyal bilimlerde araştırma tekniği: Anket* (2. bs.). Seçkin Yayıncılık.
- Aksu, G., Eser, M. T., & Güzeller, C. O. (2017). *Açımlayıcı ve doğrulayıcı faktör analizi ile yapısal eşitlik modeli uygulamaları*. Detay Yayıncılık.

- Alaidaros, H., Omar, M., & Romli, R. (2021). The state of the art of Agile Kanban method: Challenges and opportunities. *Independent Journal of Management & Production*, 12(8), 2535-2550. <https://doi.org/10.14807/ijmp.v12i8.1482>
- Alazzawi, A., Yas, Q. M., & Rahmatullah, B. (2023). *A comprehensive review of software development life cycle methodologies: Pros, cons, and future directions*. *Iraqi Journal for Computer Science and Mathematics*, 4(4), 901-173. <https://doi.org/10.52866/ijcsm.2023.04.04.041>
- Aldahmash, A. M. (2018). *A review on the critical success factors of agile software development: An empirical study* (Doctoral dissertation, University of Southampton). ProQuest Dissertations & Theses Global. (Publication No. 27776028).
- Aldahmash, A., Gravell, A. M., & Howard, Y. (2017a). A review on the critical success factors of agile software development. *EuroSPI 2017: Systems, Software and Services Process Improvement*, 748, 1-9. https://doi.org/10.1007/978-3-319-64218-5_41
- Aldahmash, A., Gravell, A. M., & Howard, Y. (2017b). Using factor analysis to study the critical success factors of agile software development. *Journal of Software*, 12(12), 957-963. <https://doi.org/10.17706/jsw.12.12.957-963>
- Al-Sakkaf, A. M., Hashim, N. L., & Omar, M. (2017). Using hierarchical cluster analysis to generate clusters of agile practices. *Journal of Telecommunication, Electronic and Computer Engineering*, 9(1-2), 53-56.
- Alshamrani, A., & Bahattab, A. (2015). A comparison between three SDLC models: Waterfall model, spiral model, and incremental/iterative model. *International Journal of Computer Science Issues*, 12(1), 106-111.
- Anderson, D. J. (2004). *Agile management for software engineering*. Upper Saddle River, NJ: Prentice Hall.
- Anderson, D. J. (2010). *Kanban: Successful Evolutionary Change for Your Technology Business*, Blue Hole Press, Ed.1.

- Anjani, R. F., Raharjo, T., Hardian, B., & Suhanto, A. (2021). Measuring the relationship between organizational culture and project success: A survey of Agile software development teams. *International Journal of Research in Business and Social Science* (2147-4478), 10(2), 76-85. <https://doi.org/10.20525/ijrbs.v10i2.1044>
- Anwer, F., Aftab, S., Waheed, U., & Muhammad, S. S. (2017). Agile software development models TDD, FDD, DSDM, and Crystal methods: A survey. *International Journal of Multidisciplinary Sciences and Engineering*, 8(2), 1-10.
- Arcos-Medina, G., & Mauricio, D. (2020). Identifying factors influencing on agile practices for software development. *Journal of Information and Organizational Sciences*, 44(1), 1-31. <https://doi.org/10.31341/jios.44.1.1>
- Arora, M., Chopra, S., Rakhra, M., Minhas, V., Walia, R., Aggarwal, R., Kumar, M., & Shabaz, M. (2021). *Agile umbrella methodologies and its global impact*. Annals of R.S.C.B., 25(4), 2990–3003. <http://annalsofrscb.ro>
- Atkinson, R. (1999). Project management: Cost, time and quality, two best guesses and a phenomenon, it's time to accept other success criteria. *International Journal of Project Management*, 17(6), 337–342. [https://doi.org/10.1016/S0263-7863\(98\)00069-6](https://doi.org/10.1016/S0263-7863(98)00069-6)
- Awad, M. A. (2005). A Comparison Between Agile And Traditional Software Development Methodologies. University of Western Australia
- Azhar, N. A., & Abdullah, N. A. S. (2022). Mapping study of critical success factors for agile software projects. *Proceedings of the IEEE International Conference on Computing (ICOCO)*, 108-113. <https://doi.org/10.1109/ICOCO56118.2022.10031946>
- Babar, M. A., Brown, A. W., & Mistrick, I. (2013). *Agile software architecture: Aligning agile processes and software architectures* (1st ed.). Morgan Kaufmann.
- Baron, R. M., & Kenny, D. A. (1986). The moderator–mediator variable distinction in social psychological research: Conceptual, strategic, and statistical considerations. *Journal of personality and social psychology*, 51(6), 1173-1182.

- Barros, L., Tam, C., & Varajao, J. (2024). Agile software development projects–Unveiling the human-related critical success factors. *Information and Software Technology*, 170, 107432. <https://doi.org/10.1016/j.infsof.2024.107432>
- Başar, A., Özkaya, A., & Kesgin, F. (2015). Yazılım geliştirme süreçlerinde şelale yönteminden çevik yaklaşıma geçiş: Bir teknoloji şirketinde uygulama. 5. *Ulusal Yazılım Mühendisliği Sempozyumu (UYMS'15)*, 26-28 Eylül, Ankara, Türkiye.
- Bayram, N. (2016). *Yapısal eşitlik modellemesine giriş: AMOS uygulamaları* (3. bs.). Ezgi Kitabevi.
- Baytam, V. ve Kapısız, O. (2011). Scrum yazılım geliştirme modeli yönetim aracı: ScrumMApp. 5. *Ulusal Yazılım Mühendisliği Sempozyumu (UYMS'11)*, 26-28 Eylül, Ankara, Türkiye.
- Beck, K. (1999). Embracing change with extreme programming. *Computer*, 32(10), 70-77. <https://doi.org/10.1109/2.796139>
- Beck, K. (2000). *Extreme programming: Die revolutionäre Methode für Softwareentwicklung in kleinen Teams*. Addison-Wesley.
- Beck, K. (2002). *Test-driven development: By example*. Addison-Wesley.
- Beck, K., & Andres, C. (2004). *Extreme programming explained: Embrace change* (2nd ed.). Addison-Wesley.
- Bergmann, T. (2018). *The relationship between agile project management and project success outcomes* (Doctoral dissertation, University of Central Florida). STARS Electronic Theses and Dissertations. <https://stars.library.ucf.edu/etd/6210>
- Bergmann, T., & Karwowski, W. (2019). Agile project management and project success: A literature review. In J. I. Kantola, S. Nazir, & T. Barath (Eds.), *Advances in human factors, business management and society* (pp. 405-414). Springer. https://doi.org/10.1007/978-3-319-94709-9_39
- Beynon-Davies, P., Carne, C., Mackay, H., & Tudhope, D. (1999). Rapid application development (RAD): An empirical review. *European Journal of Information Systems*, 8(3), 211-223. <https://doi.org/10.1057/palgrave.ejis.3000325>

- Binboğa, B., & Gümüşsoy, C. A. (2024). Factors affecting agile software project success. *IEEE Access*. <https://doi.org/10.1109/ACCESS.2024.3384410>
- Boehm, B. W. (1979). Guidelines for verifying and validating software requirements and design specifications. In *Proceedings of Euro IFIP 79* (Vol. 1, pp. 711–719).
- Boehm, B. W. (1988). A spiral model of software development and enhancement. *IEEE Computer*, 21(5), 61–72. <https://doi.org/10.1109/2.59>
- Bogopa, M. E., & Marnewick, C. (2022). Critical success factors in software development projects. *South African Journal of Business Management*, 53(1), 1-11. <https://doi.org/10.4102/sajbm.v53i1.2987>
- Boström, G., Wäyrynen, J., Bodén, M., Beznosov, K., & Kruchten, P. (2006). Extending XP practices to support security requirements engineering. *Proceedings of the 2006 International Workshop on Software Engineering for Secure Systems (SESS '06)*, 11–18. <https://doi.org/10.1145/1137627.1137631>
- Bougroun, Z., Zeaaraoui, A., & Bouchentouf, T. (2014, October). The projection of the specific practices of the third level of CMMI model in agile methods: Scrum, XP, and Kanban. *IEEE Xplore Digital Library*. <https://doi.org/10.1109/XYZ>
- Brezočnik, L., & Majer, Č. (2016). Comparison of agile methods: Scrum, Kanban, and Scrumban. In *Proceedings of the 19th International Multiconference on Information Society* (Vol. 100, No. July, ss. 1-5).
- Brown, G. A. (2015). *An examination of critical success factors of an agile project* (Doctoral dissertation, Capella University). ProQuest Dissertations & Theses Global. (Publication No. 3684950).
- Brown, T. A. (2015). *Confirmatory factor analysis for applied research* (2nd ed.). The Guilford Press.
- Bullen, C. V., & Rockart, J. F. (1981). *A primer on critical success factors* (CISR No. 69). Center for Information Systems Research, Sloan School of Management, Massachusetts Institute of Technology.

- Büyüköztürk, Ş. (2016). Sosyal bilimler için veri analizi el kitabı: İstatistik, araştırma deseni *SPSS uygulamaları ve yorum*. (22. Bs.) Pegem Akademi.
- Cao, D. B. (2006). *An empirical investigation of critical success factors in agile software development projects* (Doctoral dissertation, Capella University). ProQuest Dissertations & Theses Global. (Publication No. 3238566).
- Chiyangwa, T. B., & Mnkandla, E. (2017). Modelling the critical success factors of agile software development projects in South Africa. *South African Journal of Information Management*, 19(1), a838. <https://doi.org/10.4102/sajim.v19i1.838>
- Chow, T., & Cao, D. B. (2008). A survey study of critical success factors in agile software projects. *Journal of Systems and Software*, 81(6), 961-971. <https://doi.org/10.1016/j.jss.2007.08.020>
- Chowdhury, A. F., & Huda, M. N. (2011). Comparison between Adaptive Software Development and Feature Driven Development. In *Proceedings of 2011 International Conference on Computer Science and Network Technology* (pp. 363-367). IEEE. <https://doi.org/10.1109/ICCSNT.2011.6181977>
- Cohen, D., Lindvall, M., & Costa, P. (2004). An introduction to agile methods. In *Advances in Computers* (Vol. 62, pp. 1-66). Elsevier. [https://doi.org/10.1016/S0065-2458\(03\)62001-2](https://doi.org/10.1016/S0065-2458(03)62001-2)
- Cole, R., & Scotcher, E. (2015). *Brilliant agile project management: A practical guide to using Agile, Scrum, and Kanban*. Pearson Education Limited.
- Collier, J. (2020). *Applied structural equation modeling using AMOS: Basic to advanced techniques*. Routledge.
- Conboy, K., & Fitzgerald, B. (2010). Method and developer characteristics for effective agile method tailoring: A study of XP expert opinion. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 20(1), Article 2, 1–30. <https://doi.org/10.1145/1767751.1767753>
- Corvello, V., & Verteramo, S. (2014). The role of the customer in the adoption of agile software development methodologies. *Proceedings of the 8th Mediterranean*

Conference on Information Systems (MCIS), Verona, Italy.
<https://aisel.aisnet.org/mcis2014/3>

- Coşkun, R., Altunışık, R., Bayraktaroğlu, S. ve Yıldırım, E. (2015). Sosyal Bilimlerde Araştırma Yöntemleri SPSS Uygulamalı, Sakarya Yayıncılık, 8. Baskı, Sakarya.
- Çokluk, Ö., Şekercioğlu, G., & Büyüköztürk, Ş. (2021). *Sosyal bilimler için çok değişkenli istatistik: SPSS ve LISREL uygulamaları* (6. bs.). Pegem Akademi.
- Dada, O. A., & Sanusi, I. T. (2022). The adoption of software engineering practices in a Scrum environment. *African Journal of Science, Technology, Innovation and Development*, 14(6), 1429–1446.
<https://doi.org/10.1080/20421338.2021.1955431>
- Darwish, N. R., & Rizk, N. M. (2015). Multi-dimensional success factors of agile software development projects. *International Journal of Computer Applications*, 118(15), 23-30. <https://doi.org/10.5120/20823-3453>
- Daud, N. M. N., Bakar, N. A. A. A., & Rusli, H. M. (2010). Implementing rapid application development (RAD) methodology in developing practical training application system. In *2010 International Symposium on Information Technology* (pp. 1664-1667). IEEE. <https://doi.org/10.1109/ITSIM.2010.5561634>
- Deek, F. P., McHugh, J. A., & Eljabiri, O. M. (2005). *Strategic software engineering: An interdisciplinary approach*. Auerbach Publications.
- Deemer, P., Benefield, G., Larman, C., & Vodde, B. (2010). *The Scrum primer: An in-depth introduction to the theory and practice of Scrum* (Version 1.2). Scrum Training Institute.
- Demirel, Ş. (2016). *Agile software development project evaluation by using the partial least squares-structural equation modeling (PLS-SEM) approach in the view of critical success indicators' success research* (Tez No. 449229) [Yüksek Lisans Tezi, Bahçeşehir Üniversitesi]. YÖK Ulusal Tez Merkezi. <https://tez.yok.gov.tr>
- Dennehy, D., & Conboy, K. (2017). Going with the flow: An activity theory analysis of flow techniques in software development. *The Journal of Systems and Software*, 133, 160-173. <https://doi.org/10.1016/j.jss.2016.10.003>

- Despa, M. L. (2014). Comparative study on software development methodologies. *Database Systems Journal*, 5(3), 37-56.
- Diel, E., Bergmann, M., Marczak, S., & Luciano, E. (2015, October). What is agile, which practices are used, and which skills are necessary according to Brazilian professionals: Findings of an initial survey. *6th Brazilian Workshop on Agile Methods (WBMA)*. <https://doi.org/10.1109/WBMA.2015.10>
- Drury-Grogan, M. L. (2014). Performance on agile teams: Relating iteration objectives and critical decisions to project management success factors. *Information and Software Technology*, 56(5), 506–515. <https://doi.org/10.1016/j.infsof.2013.11.003>
- Eisinga, R., Grotenhuis, M., & Pelzer, B. (2013). The reliability of a two-item scale: Pearson, Cronbach, or Spearman-Brown?. *International Journal of Public Health*, 58(4), 637-642. <https://doi.org/10.1007/s00038-012-0416-3>
- Ergüder, H. (2019). *Agile versus traditional approaches to information technology project management: A comparison of critical success factors* (Tez No. 556711) [Yüksek Lisans Tezi, Boğaziçi Üniversitesi]. YÖK Ulusal Tez Merkezi. <https://tez.yok.gov.tr>
- Estrada-Esponda, R. D., López-Benítez, M., Matturro, G., & Osorio-Gómez, J. C. (2024). Selection of software agile practices using Analytic hierarchy process. *Heliyon*, 10(e22948). <https://doi.org/10.1016/j.heliyon.2023.e22948>
- Fadilah, R., Raharjo, T., Hardian, B., Suhanto, A., & Anjani, R. F. (2022). People factors influencing project success in software development: A survey of agile development teams in Indonesia. *Proceedings of the 1st International Conference on Science and Technology Innovation (ICoSTEC)*, Yogyakarta, Indonesia, February 26, 2022, 1-8.
- Farlik, J. T. (2016). *Project success in agile development software projects* (Order No. 10108921). Available from ProQuest Central; ProQuest Dissertations & Theses Global; Publicly Available Content Database. (1793941023). <https://www.proquest.com/dissertations-theses/project-success-agile-development-software/docview/1793941023/se-2>

- Flora, H. K., & Chande, S. V. (2014). A systematic study on agile software development methodologies and practices. *International Journal of Computer Science and Information Technologies*, 5(3), 3626-3637.
- Fornell, C., & Larcker, D. F. (1981). Evaluating structural equation models with unobservable variables and measurement error. *Journal of Marketing Research*, 18(1), 39-50. <https://doi.org/10.2307/3151312>
- Fry, C. J. (2024). *A Correlational Study of Technical and Social Factors That Affect Agile Software Project Success* (Order No. 31297438). Available from ProQuest Dissertations & Theses Global. (3053876742). <https://www.proquest.com/dissertations-theses/correlational-study-technical-social-factors-that/docview/3053876742/se-2>
- Garousi, V., Tarhan, A., Pfahl, D., Coşkunçay, A., & Demirörs, O. (2019). Correlation of critical success factors with success of software projects: An empirical investigation. *Journal of Systems and Software*, 144, 277-294. <https://doi.org/10.1016/j.jss.2018.07.011>
- Gencer, C., & Kayacan, A. (2017). Yazılım proje yönetimi: Şelale modeli ve çevik yöntemlerin karşılaştırılması. *Bilişim Teknolojileri Dergisi*, 10(3), 335-347. <https://doi.org/10.17671/gazibtd.331054>
- George, D., & Mallery, M. (2010). *SPSS for Windows step by step: A simple guide and reference, 17.0 update* (10a ed.). Pearson.
- Ghimire, D., & Charters, S. (2022). The impact of agile development practices on project outcomes. *Software*, 1(3), 265–275. <https://doi.org/10.3390/software1030012>
- Greene, L. E. (2022). *Organizational, Technical, and Process Factor Relationship on Agile IT Project Success* (Order No. 29253160). Available from ProQuest Central; ProQuest Dissertations & Theses Global. (2684958301). <https://www.proquest.com/dissertations-theses/organizational-technical-process-factor/docview/2684958301/se-2>
- Gupta, M., George, J. F., & Xia, W. (2019). Relationships between IT department culture and agile software development practices: An empirical investigation.

International Journal of Information Management, 44, 13–24.
<https://doi.org/10.1016/j.ijinfomgt.2018.09.006>

Gürbüz, S. (2021a). AMOS ile Yapısal Eşitlik Modellemesi Temel İlkeler ve Uygulamalı Analizler. Seçkin Yayınevi (2. Baskı)

Gürbüz, S. (2021b). *Sosyal bilimlerde aracı, düzenleyici ve durumsal etki analizleri* (2. bs.). Seçkin Yayıncılık.

Gürbüz, S., & Şahin, F. (2018). *Sosyal bilimlerde araştırma yöntemleri: Felsefe - Yöntem - Analiz* (5. bs.). Seçkin Yayıncılık.

Güriş, S., & Astar, M. (2015). Bilimsel araştırmalarda SPSS ile İstatistik. (2. Bs.) Der Yayınları

Gwangwadza, A., & Hanslo, R. (2024). Identification and prioritization of success factors in agile software development in the South African software development industry. *IEEE Engineering Management Review*.
<https://doi.org/10.1109/EMR.2024.3449445>

Hair, J. F., Black, W. C., Babin, B. J., Anderson, R. E., & Tatham, R. L. (2019). *Multivariate Data Analysis* (8th ed.). Pearson Education Limited.

Hamid, M., Zeshan, F., Ahmad, A., Malik, S., Saleem, M., Tabassum, N., & Qasim, M. (2022). Analysis of software success through structural equation modeling. *Intelligent Automation & Soft Computing*, 31(3), 1689-1702.
<https://doi.org/10.32604/iasc.2022.020898>

Haskin, J. (2022). *Agile Software Development and Successful Project Completion: A Quantitative Study* (Order No. 29391257). Available from ProQuest Central; ProQuest Dissertations & Theses Global; Publicly Available Content Database. (2723528094). <https://www.proquest.com/dissertations-theses/agile-software-development-successful-project/docview/2723528094/se-2>

Hayes, A. F. (2013). *Introduction to mediation, moderation, and conditional process analysis: A Regression-Based Approach*. Guilford Press.

- Hayes, A. F. (2022). *Introduction to mediation, moderation, and conditional process analysis: A Regression-Based Approach* (3th ed.). Guilford Press
- Henriksen, A., & Pedersen, S. A. R. (2017). A qualitative case study on agile practices and project success in agile software projects. *The Journal of Modern Project Management*, 5(1). <https://doi.org/10.19255/JMPM01306>
- Henseler, J., Ringle, C. M., & Sarstedt, M. (2015). A new criterion for assessing discriminant validity in variance-based structural equation modeling. *Journal of the Academy of Marketing Science*, 43(1), 115-135. <https://doi.org/10.1007/s11747-014-0403-8>
- Heper, S. (2023). *Çevik proje yönetiminde kritik başarı faktörleri* (Tez No. 838616) [Doktora Tezi, Ankara Üniversitesi]. YÖK Ulusal Tez Merkezi. <https://tez.yok.gov.tr>
- Highsmith, J. (2002). *Agile software development ecosystems*. Addison Wesley.
- Highsmith, J. A. (2000). *Adaptive software development: A collaborative approach to managing complex systems*. Dorset House Publishing.
- Hoda, R., Noble, J., & Marshall, S. (2011). The impact of inadequate customer collaboration on self-organizing Agile teams. *Information and Software Technology*, 53(5), 521–534. <https://doi.org/10.1016/j.infsof.2010.10.009>
- Hu, L. T., & Bentler, P. M. (1999). Cutoff criteria for fit indexes in covariance structure analysis: Conventional criteria versus new alternatives. *Structural Equation Modeling: A Multidisciplinary Journal*, 6(1), 1-55. <https://doi.org/10.1080/10705519909540118>
- Hughes, B., Cotterell, M., & Mall, R. (2011). *Software project management* (5th ed.). Tata McGraw-Hill Education.
- Hummel, M., Rosenkranz, C., & Holten, R. (2015). The role of social agile practices for direct and indirect communication in information systems development teams. *Communications of the Association for Information Systems*, 36(15), 273–300. <https://doi.org/10.17705/1CAIS.03615>

- IEEE Computer Society. (2014). *Guide to the software engineering body of knowledge (SWEBOOK), version 3.0*. IEEE. <https://www.computer.org/education/bodies-of-knowledge/software-engineering>
- IEEE. (1990). *IEEE Standard Glossary of Software Engineering Terminology* (IEEE Std 610.12-1990). Institute of Electrical and Electronics Engineers.
- Ika, L. A. (2009). Project success as a topic in project management journals. *Project Management Journal*, 40(4), 6–19. <https://doi.org/10.1002/pmj.20137>
- Jain, R., & Suman, U. (2016). Effectiveness of agile practices in global software development. *International Journal of Grid and Distributed Computing*, 9(10), 231-248. <https://doi.org/10.14257/ijgdc.2016.9.10.21>
- Jalali, S., & Wohlin, C. (2011). Global software engineering and agile practices: A systematic review. *Journal of Software: Evolution and Process*, 24(6), 643–659. <https://doi.org/10.1002/smr.561>
- Javanmard, M., & Alian, M. (2015). Comparison between Agile and Traditional software development methodologies. *Cumhuriyet Üniversitesi Fen-Edebiyat Fakültesi Fen Bilimleri Dergisi*, 36(3), 1386-1394.
- Junior, A. C. P., da Silva, V. R., & Aquino, P. T. (2024). Critical success factors for agile software development. *IEEE Transactions on Engineering Management*, 71, 14807-14820. <https://doi.org/10.1109/TEM.2024.3441829>
- Jyothi, V. E., & Rao, K. N. (2012). Effective implementation of agile practices in coordination with lean Kanban. *International Journal on Computer Science and Engineering (IJCSE)*, 4(1), 87-91.
- Kalaycı, Ş. (2014). *SPSS uygulamalı çok değişkenli istatistik teknikleri* (6. bs.). Asil Yayın Dağıtım.
- Karaca, S. (2021). *Bilimsel araştırma yöntemleri*. Gazi Kitabevi.
- Karagöz, Y. (2019). *SPSS, AMOS, META uygulamalı istatistiksel analizler* (2. bs.). Nobel Yayıncılık.

- Kerzner, H. (2017). *Project management: A systems approach to planning, scheduling, and controlling* (12th ed.). Wiley.
- Keskinkılıç, M., & Kahveci, F. (2019). Yazılım mühendisliğinde çevik yöntemler üzerine kavramsal bir inceleme ve sınıflandırma. *Atatürk Üniversitesi Sosyal Bilimler Enstitüsü Dergisi*, 23(3), 1067-1091.
- Kim, S., Cho, K., & Jung, H. (2019). The role of clients in software projects with agile methodology: Client involvement, requirements changes, and expectations. *Proceedings of the 2019 International Conference on Software Engineering and Knowledge Engineering*, 345-350. <https://doi.org/10.1109/SEKE.2019.00050>
- Kimberlin, C. L., & Winterstein, A. G. (2008). Validity and reliability of measurement instruments used in research. *American Journal of Health-System Pharmacy*, 65(23), 2276-2284. <https://doi.org/10.2146/ajhp070364>
- Kirpitsas, I. K., & Pachidis, T. P. (2022). Evolution towards hybrid software development methods and information systems audit challenges. *Software*, 1(3), 316–363. <https://doi.org/10.3390/software1030015>
- Kline, R. B. (2015). *Principles and practice of structural equation modeling* (4th ed.). Guilford Press.
- Klotins, E., Unterkalmsteiner, M., Chatzipetrou, P., Gorschek, T., Prikladnicki, R., Tripathi, N., & Pompermaier, L. B. (2021). Use of agile practices in start-up companies. *e-Informatica Software Engineering Journal*, 15(1), 47–64. <https://doi.org/10.37190/e-Inf210103>
- Kniberg, H., & Skarin, M. (2010). *Kanban and Scrum: Making the most of both*. C4Media Inc.
- Kukreja, V., Singh, A., & Ahuja, S. (2021). *Identification, assessment, and ranking agile software development critical success factors*. *Journal of Software Engineering*, 12(4), 37-48.
- Kurapati, N., Manyam, V. S. C., & Petersen, K. (2012). Agile software development practice adoption survey. In C. Wohlin (Ed.), *XP 2012: Agile processes in software*

engineering and extreme programming (pp. 16-30). Springer.
https://doi.org/10.1007/978-3-642-30350-0_2

Kute, S. S., & Thorat, S. D. (2014). A review on various software development life cycle (SDLC) models. *International Journal of Research in Computer and Communication Technology*, 3(7), 776-781.

Lalsing, V., Kishnah, S., & Pudaruth, S. (2012). People factors in Agile software development and the impact on team performance. *International Journal of Software Engineering & Applications (IJSEA)*, 3(1), 117-137.
<https://doi.org/10.5121/ijsea.2012.3109>

Larman, C. (2003). *Agile & iterative development: A manager's guide*. Addison-Wesley.

Leach, R. J. (2016). *Introduction to software engineering* (2nd ed.). CRC Press.

Leal, G. C. L., Prikladnicki, R., Ebert, C., Balancieri, R., & Pompermaier, L. B. (2019). Practices and tools for software start-ups. *IEEE Software*, 37(1), 72-77.
<https://doi.org/10.1109/MS.2019.2946783>

Lei, H., Ganjeizadeh, F., Jayachandran, P. K., & Ozcan, P. (2017). A statistical analysis of the effects of Scrum and Kanban on software development projects. *Robotics and Computer-Integrated Manufacturing*, 43, 59-67.
<https://doi.org/10.1016/j.rcim.2015.12.001>

Licorish, S. A., Spínola, R. O., Mendes, T. S., Holvitie, J., Hyrynsalmi, S., Leppänen, V., MacDonell, S. G., & Buchan, J. (2016). Adoption and suitability of software development methods and practices. *23rd Asia-Pacific Software Engineering Conference (APSEC)*, 369-372. <https://doi.org/10.1109/APSEC.2016.47>

Litchmore, K. A. H. (2016). *A comparative study of agile methods, people factors, and process factors in relation to project success* (Order No. 10142854). Available from ProQuest Central; ProQuest Dissertations & Theses Global.

Livermore, J. A. (2008). Factors that significantly impact the implementation of an agile software development methodology. *Journal of Software*, 3(4), 31-36.
<https://doi.org/10.4304/jsw.3.4.31-36>

- Lorcu, F. (2015). *Örneklerle veri analizi: SPSS uygulamalı* (1. baskı). Detay Yayıncılık.
- Mahnič, V. (2013). Applying Kanban principles to software development. In *Proceedings of the IT for Practice 2013, 16th International Conference on Information Technology for Practice* (pp. 89-97). Ostrava, Czech Republic: VŠB - Technical University of Ostrava.
- Mahnič, V. (2014). Improving software development through combination of Scrum and Kanban. In *Recent Advances in Computer Engineering, Communications and Information Technology* (pp. 281-288).
- Malik, M., Sarwar, S., & Orr, S. (2021). Agile practices and performance: Examining the role of psychological empowerment. *International Journal of Project Management*, 39(1), 10–20. <https://doi.org/10.1016/j.ijproman.2020.09.002>
- Malik, S. (2017). Software testing: Essential phase of SDLC and a comparative study of software testing techniques. *International Journal of System and Software Engineering*, 5(2), 38-45.
- Martin, R. C. (2014). *Agile software development: Principles, patterns, and practices* (Pearson New International Edition). Pearson Education Limited.
- Maruping, L. M., Venkatesh, V., & Agarwal, R. (2009). A control theory perspective on agile methodology use and changing user requirements. *Information Systems Research*, 20(3), 377–399. <https://doi.org/10.1287/isre.1090.0238>
- Matharu, G. S., Mishra, A., Singh, H., & Upadhyay, P. (2015). Empirical study of agile software development methodologies: A comparative analysis. *ACM SIGSOFT Software Engineering Notes*, 40(1), 1-6. <https://doi.org/10.1145/2693208.2693233>
- Maximilien, E. M., & Williams, L. (2003, May). *Assessing test-driven development at IBM*. In *Proceedings of the 25th International Conference on Software Engineering* (pp. 564-569). IEEE. <https://doi.org/10.1109/ICSE.2003.1201238>
- Mersino, A. (2020, Mayıs 25). *Why agile is better than waterfall (Based on Standish Group Chaos Report 2020)*. Medium. <https://medium.com/leadership-and->

agility/agile-project-success-rates-are-2x-higher-than-traditional-projects-376a05e590d4

- Meydan, C. H., & Şeşen, H. (2015). *Yapısal eşitlik modellemesi AMOS uygulamaları*. (2. Bs.) Detay Yayıncılık.
- Misra, S. C., Kumar, V., & Kumar, U. (2006). Success factors of agile software development. *Software Engineering Research and Practice*, 1, 233-239
- Misra, S. C., Kumar, V., & Kumar, U. (2009). Identifying some important success factors in adopting agile software development practices. *The Journal of Systems and Software*, 82(12), 1869-1890. <https://doi.org/10.1016/j.jss.2009.05.052>
- Modi, H. S., Singh, N. K., & Chauhan, H. P. (2017). Comprehensive analysis of software development life cycle models. *International Research Journal of Engineering and Technology (IRJET)*, 4(6), 117-122.
- Moustafa, A. R., & Abdulrasoul, M. G. (2024). Factors affecting the success of agile software teams. *International Journal of Computers and Informatics*, 3(2), 1-11. <https://doi.org/10.59992/IJCI.2024.v3n2p1>
- Mulugeta, M. (2021). *Agile Practice and Project Success in Software Development Projects: The Case of Cyber Soft Company* (Doctoral dissertation, St. Mary's University).
- Munassar, N. M. A., & Govardhan, A. (2010). A comparison between five models of software engineering. *International Journal of Computer Science Issues*, 7(5), 94-101.
- Nakazawa, S., & Tanaka, T. (2016). Development and application of Kanban tool visualizing the work in progress. *Proceedings of the 2016 5th IIAI International Congress on Advanced Applied Informatics*, 908-913. <https://doi.org/10.1109/IIAI-AAI.2016.156>
- Nasir, M. H. N., & Sahibuddin, S. (2011). Critical success factors for software projects: A comparative study. *Scientific Research and Essays*, 6(10), 2174-2186. <https://doi.org/10.5897/SRE10.1171>

- Naslund, D., & Kale, R. (2020). Is agile the latest management fad? A review of success factors of agile transformations. *International Journal of Quality and Service Sciences*, 12(4), 489–504. <https://doi.org/10.1108/IJQSS-12-2019-0142>
- Neumann, M. (2022, January). *The integrated list of agile practices: A tertiary study*. In *International Conference on Lean and Agile Software Development*. Hochschule Hannover.
- Newkirk, J. (2002). Introduction to agile processes and extreme programming. *Proceedings of the 24th International Conference on Software Engineering*, 695–696. <https://doi.org/10.1145/581339.581340>
- Nguyen, D. S. (2016). Success factors that influence agile software development project success. *American Scientific Research Journal for Engineering, Technology, and Sciences*, 17(1), 172–222.
- Nizam, A. (2015). *Yazılım proje yönetimi* (2. baskı). Papatya.
- Nordberg, M. E. (2003). Managing code ownership. *IEEE Software*, 20(2), 26–33. <https://doi.org/10.1109/MS.2003.1184163>
- O'Regan, G. (2021). *Concise guide to software engineering: From fundamentals to application methods* (2nd ed.). Springer.
- Ojha, T. R. (2023). Critical success factors of agile software development: A systematic literature review. *SCITECH Nepal: Journal of Scientific and Technical Studies*, 17(1), 49–57. <https://doi.org/10.3126/scitech.v17i1.60467>
- Paez, N., Fontdevila, D., & Oliveros, A. (2020, December). On the influence of Agile in the usage of software development practices. *2020 IEEE Congreso Bienal de Argentina (ARGENCON)*, 1–7. <https://doi.org/10.1109/ARGENCON49523.2020.9505407>
- Paez, N., Fontdevila, D., Gainey, F., & Oliveros, A. (2018, May). Technical and organizational agile practices: A Latin-American survey. In J. Garbajosa, X. Wang, & A. Aguiar (Eds.), *Agile Processes in Software Engineering and Extreme Programming* (pp. 146–159). Springer. https://doi.org/10.1007/978-3-319-91602-6_10

- Palmer, S. R., & Felsing, J. M. (2002). *A practical guide to feature-driven development*. Prentice Hall PTR.
- Palopak, Y., & Huang, S. J. (2024). Perceived impact of agile principles: Insights from a survey-based study on agile software development project success. *Information and Software Technology*, 176, 107552. <https://doi.org/10.1016/j.infsof.2024.107552>
- Pantiuchina, J., Mondini, M., Khanna, D., Wang, X., & Abrahamsson, P. (2017, May). Are software startups applying agile practices? The state of the practice from a large survey. In H. Baumeister, H. Lichter, & M. Riebisch (Eds.), *Agile processes in software engineering and extreme programming* (pp. 167-183). Springer. https://doi.org/10.1007/978-3-319-57633-6_11
- Perera, C., & Perera, I. (2019). The impact of client involvement towards agile project success in Sri Lankan software industry. *2019 Moratuwa Engineering Research Conference (MERCon)*, 279-284. <https://doi.org/10.1109/MERCon.2019.8818910>
- Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., & Still, J. (2008). The impact of agile practices on communication in software development. *Empirical Software Engineering*, 13(3), 303-337. <https://doi.org/10.1007/s10664-008-9065-9>
- Pinto, J. K. (2020). *Project management: Achieving competitive advantage* (5th ed.). Pearson.
- Pollack, J., Helm, J., & Adler, D. (2018). What is the Iron Triangle, and how has it changed? *International Journal of Managing Projects in Business*, 11(2), 527-547. <https://doi.org/10.1108/IJMPB-09-2017-0107>
- Poppendieck, M. (2007, May). *Lean software development*. In *29th International Conference on Software Engineering (ICSE'07 Companion)* (pp. 165-166). IEEE. <https://doi.org/10.1109/ICSECOMPANION.2007.46>
- Poppendieck, M., & Poppendieck, T. (2003). *Lean software development: An agile toolkit*. Addison Wesley.

- Portman, H. (2021, Ocak 6). *Review Standish Group – CHAOS 2020: Beyond Infinity*. Henny Portman's Blog. <https://hennyportman.wordpress.com>
- Pressman, R.S. (2001). *A Practitioner's Approach*, Fifth Edition, Published by McGraw-Hill.
- Pressman, R. S. (2001). *Software engineering: A practitioner's approach* (5th ed.). McGraw-Hill.
- Pressman, R. S. (2010). *Software engineering: A practitioner's approach* (7th ed.). McGraw-Hill Education.
- Pressman, R. S., & Maxim, B. R. (2015). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill Education.
- Project Management Institute, & Agile Alliance. (2017). *Agile practice guide*. Project Management Institute.
- Project Management Institute (PMI). (2017). *A guide to the project management body of knowledge (PMBOK guide)* (6th ed.). Project Management Institute.
- Project Management Institute (PMI). (2021). *A guide to the project management body of knowledge (PMBOK guide)* (7th ed.). Project Management Institute.
- Qatanani, D., Al-Tawara, F., & Qusef, A. (2021). Success factors of agile projects: Case study for projects in Jordan during COVID-19 pandemic. *2021 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, 95-102. <https://doi.org/10.1109/JEEIT53412.2021.9634094>
- Rastogi, V. (2015). Software development life cycle models: Comparison, consequences. *International Journal of Computer Science and Information Technologies (IJCSIT)*, 6(1), 168-172.
- Rather, M. A., & Bhatnagar, M. V. (2015). A comparative study of software development life cycle models. *International Journal of Application or Innovation in Engineering & Management (IJAIEEM)*, 4(10), 23-29.
- Recker, J., Holten, R., Hummel, M., & Rosenkranz, C. (2017). How agile practices impact customer responsiveness and development success: A field study. *Project*

- Management Journal*, 48(2), 99-121.
<https://doi.org/10.1177/875697281704800208>
- Reel, J. S. (1999). Critical success factors in software projects. *IEEE Software*, 16(3), 18-23. <https://doi.org/10.1109/52.765782>
- Royce, W. W. (1970). Managing The Development of Large Software Systems. The Proceedings of the WESCON. (*Tekrar Baskı, Royce, W. W. (1987, March). Managing the development of large software systems: concepts and techniques. In Proceedings of the 9th international conference on Software Engineering (pp. 328-338). IEEE Computer Society Press.)*
- Sabale, R. G., & Dani, A. R. (2012). Comparative study of prototype model for software engineering with system development life cycle. *IOSR Journal of Engineering (IOSRJEN)*, 2(7), 21-24.
- Salman, S., Nisar, B., & Ashraf, H. (2018). *Comparative analysis of agile practices: XP vs SCRUM*. Journal of Software Engineering & Intelligent Systems, 3(2), 192-199.
- Sandstø, R., & Reme-Ness, C. (2021). Agile practices and impacts on project success. *Journal of Engineering, Project, and Production Management*, 11(3), 255-262. <https://doi.org/10.2478/jeppm-2021-0024>
- Sarker, I. H., Faruque, F., Hossen, U., & Rahman, A. (2015). A survey of software development process models in software engineering. *International Journal of Software Engineering and Its Applications*, 9(11), 55-70. <https://doi.org/10.14257/ijseia.2015.9.11.05>
- Schwaber, K. (1997). Scrum development process. In *Business Object Design and Implementation: OOPSLA '95 Workshop Proceedings 16 October 1995, Austin, Texas* (ss. 117-134). Springer London.
- Schwaber, K. (2004). *Agile project management with Scrum*. Microsoft Press.
- Schwaber, K., & Sutherland, J. (2017). *The Scrum Guide: The definitive guide to Scrum: The rules of the game*. Scrum.org. <https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf>

- Schwaber, K., & Sutherland, J. (2020). *The Scrum guide: The definitive guide to Scrum: The rules of the game*. Scrum.org.
- Seçer, İ. (2015). *Psikolojik test geliştirme ve uyarlama süreci: SPSS ve LISREL uygulamaları*. Anı Yayıncılık.
- Sekaran, U. (2003). *Research methods for business: A skill-building approach* (4th ed.). John Wiley & Sons Inc.
- 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. <https://doi.org/10.1016/j.ijproman.2015.01.006>
- Shah, M. D., & Dhiman, C. (2016). Software process models outline. *IRACST - International Journal of Computer Science and Information Technology & Security (IJCSITS)*, 6(2)
- Shakya, P., & Shakya, S. (2020). Critical success factor of agile methodology in software industry of Nepal. *Journal of Information Technology*, 2(03), 135–143. <https://doi.org/10.36548/jitdw.2020.3.001>
- Shaw, M. (1990). Prospects for an engineering discipline of software. *IEEE Software*, 7(6), 15-24. <https://doi.org/10.1109/52.62930>.
- Sheffield, J., & Lemétayer, J. (2013). Factors associated with the software development agility of successful projects. *International Journal of Project Management*, 31(3), 459-472. <https://doi.org/10.1016/j.ijproman.2012.09.011>
- Shylesh, S. (2017, April). A study of software development life cycle process models. In *National Conference on Reinventing Opportunities in Management, IT, and Social Sciences* (pp. 534-541).
- Siddique, L., & Hussein, B. A. (2016). A qualitative study of success criteria in Norwegian agile software projects from suppliers' perspective. *International Journal of Information Systems and Project Management*, 4(2), 63-79. <https://doi.org/10.12821/ijispm040204>

- Sienkiewicz, Ł. (2012). Scrumban – The Kanban as an addition to Scrum software development method in a network organization. *Informatyka Ekonomiczna*, 2(24), 73-81.
- Silberschatz, A., Galvin, P. B., & Gagne, G. (2014). *Operating System Concepts* (9th ed.). Wiley.
- Silva, K. M. B., & Santos, S. (2016). Critical factors in agile software projects according to people, process, and technology perspective. *Proceedings of the 2015 6th Brazilian Workshop on Agile Methods*. <https://doi.org/10.1109/WBMA.2015.19>
- Šmite, D., Moe, N. B., Šāblis, A., & Wohlin, C. (2017). Software teams and their knowledge networks in large-scale software development. *Information and Software Technology*, 86, 71–86. <https://doi.org/10.1016/j.infsof.2017.01.003>
- Smith, G., & Sidky, A. (2009). *Becoming agile in an imperfect world*. Manning Publications.
- So, C., & Scholl, W. (2009). *Perceptive agile measurement: New instruments for quantitative studies in the pursuit of the social-psychological effect of agile practices*. In P. Abrahamsson, M. Marchesi, & F. Maurer (Eds.), *XP 2009: Proceedings of the 10th International Conference on Agile Processes in Software Engineering and Extreme Programming* (pp. 83–93). Springer. https://doi.org/10.1007/978-3-642-01853-4_9
- Sommerville, I. (2015). *Software engineering* (10th ed.). Pearson.
- Sommerville, I. (2021). *Engineering software products: An introduction to modern software engineering* (Global ed.). Pearson.
- Stanberry, L. (2018). *Critical Success Factors for Large and Distributed Agile Software Development Projects Using Scrum in U.S.-Based Global Companies* (Order No. 10748199). Available from ProQuest Central; ProQuest Dissertations & Theses Global; Publicly Available Content Database. (2027207191).
- Stankovic, D., Nikolic, V., Djordjevic, M., & Cao, D. B. (2013). A survey study of critical success factors in agile software projects in former Yugoslavia IT companies.

- Journal of Systems and Software*, 86(6), 1663–1678.
<https://doi.org/10.1016/j.jss.2013.02.027>
- Stapleton, J. (1997). *DSDM, Dynamic Systems Development Method: The Method in Practice*. Cambridge University Press.
- Stapleton, J. (1999). DSDM: Dynamic Systems Development Method. In *Proceedings Technology of Object-Oriented Languages and Systems. TOOLS 29* (p. 406). IEEE. <https://doi.org/10.1109/TOOLS.1999.779095>
- State of Agile. (2020). *14th annual state of agile report 2019*.
<https://stateofagile.com/#ufh-i-615706098-14th-annual-state-of-agile-report/7027494>
- Sudhakar, G. P. (2012). A model of critical success factors for software projects. *Journal of Enterprise Information Management*, 25(6), 537-558.
<https://doi.org/10.1108/17410391211272829>
- Sun, W., & Schmidt, C. (2018). Practitioners' agile-methodology use and job perceptions. *IEEE Software*, 35(2), 52-61. <https://doi.org/10.1109/MS.2018.2141026>
- Sutherland, J., & Schwaber, K. (2011). *The Scrum papers: Nuts, bolts, and origins of an agile framework*. Scrum Inc.
- Suvacı, E. (2019). Proje ve proje yönetimine giriş. H. Durucasu (Ed.), *Proje yönetimi içinde* (s. 2-23). Anadolu Üniversitesi Yayını.
- Sürücü, L., & Maslakçı, A. (2020). Validity and Reliability in Quantitative Research. *Business & Management Studies: An International Journal*, 8(3), 2694-2726.
<http://dx.doi.org/10.15295/bmij.v8i3.1540>
- Sürücü, L., Şeşen, H., Maslakçı, A., & Yeloğlu, H. O. (2023). *SPSS, AMOS ve PROCESS Macro ile ilişkisel, aracı/düzenleyici ve yapısal eşitlik modellemesi* (2. bs.). Detay Yayıncılık.
- Sverrisdottir, H. S., Ingason, H. T., & Jonasson, H. I. (2014). The role of the product owner in Scrum: Comparison between theory and practices. *Procedia - Social and Behavioral Sciences*, 119, 257-267. <https://doi.org/10.1016/j.sbspro.2014.03.030>

- Syeda, S. H. (2018). *An Exploratory Study to Identify Critical Success Factors of Agile Systems Engineering* (Order No. 10743062). Available from ProQuest Dissertations & Theses Global. (2026669496). <https://www.proquest.com/dissertations-theses/exploratory-study-identify-critical-success/docview/2026669496/se-2>
- Şimşek, V., & Ateş, V. (2022). Bilişim sistemi geliştirme sürecindeki insan faktörü üzerine sistematik literatür taraması. *Organizasyon ve Yönetim Bilimleri Dergisi*, 14(1), 85-98.
- T.C. Sanayi ve Teknoloji Bakanlığı. (2021). *Teknoloji geliştirme bölgeleri istatistikleri – Kasım 2021*. <https://www.sanayi.gov.tr/istatistikler/istatistiki-bilgiler/mi0203011501>
- Tabachnick, B. G., & Fidell, L. S. (2013). *Using multivariate statistics* (6th ed., International ed.). Pearson.
- Takeuchi, H., & Nonaka, I. (1986). The new new product development game. *Harvard Business Review*, 64(1), 137-146.
- 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. <https://doi.org/10.1016/j.ijproman.2020.02.001>
- Thakur, A., Singh, D., & Chaudhary, V. (2015). Comparing various SDLC models on the basis of available methodology. *International Journal of Modern Engineering Research (IJMER)*, 5(3), 34-39.
- Thakur, S., & Singh, H. (2014, May). *FDRD: Feature Driven Reuse Development Process Model*. In *2014 IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)* (pp. 1593–1598). IEEE. <https://doi.org/10.1109/ICACCCT.2014.7019376>
- Thompson, C. R. (2019). The impact of customer collaboration on agile product development success in technology startups within the Pacific Northwest

- (Doctoral dissertation, Liberty University). Liberty University Digital Commons. <https://digitalcommons.liberty.edu/doctoral/2201>
- Tripp, J. F., Riemenschneider, C., & Thatcher, J. B. (2016). Job satisfaction in agile development teams: Agile development as work redesign. *Journal of the Association for Information Systems*, 17(4), 267-307. <https://doi.org/10.17705/1jais.00426>
- Tsoy, M., & Staples, D. S. (2020, January). Exploring critical success factors in agile analytics projects. *Proceedings of the 53rd Hawaii International Conference on System Sciences*. <https://hdl.handle.net/10125/63861>
- Turkey Agility Report. (2019). *Türkiye 8. Çeviklik Raporu*. Agile Turkey. <http://www.agileturkey.org/>
- Tutar, H., & Erdem, A. T. (2022). *Örnekleriyle bilimsel araştırma yöntemleri ve SPSS uygulamaları* (2. bs.). Seçkin Yayıncılık.
- Türk Dil Kurumu. (t.y.). *Yazılım. İçinde Güncel Türkçe Sözlük*. <https://sozluk.gov.tr/> (Erişim Tarihi: 3 Eylül 2024)
- Vallon, R., Estácio, B. J. da S., Prikladnicki, R., & Grechenig, T. (2018). Systematic literature review on agile practices in global software development. *Information and Software Technology*, 96, 161–180. <https://doi.org/10.1016/j.infsof.2017.12.004>
- VersionOne. (2008). *3rd Annual "State of Agile Development" Survey*.
- Vithana, V. N., Fernando, S. G. S., & Kapurubandara, M. (2015). Success factors for agile software development: A case study from Sri Lanka. *International Journal of Computer Applications*, 113(17), 10-17. <https://doi.org/10.5120/19981-1925>
- Webb, A. (2015). *Agile project management methodology for beginners: Scrum project management for beginners*. Smashwords Edition.
- Whittaker, J. A. (2000). What is software testing? And why is it so hard? *IEEE Software*, 17(1), 70-79. <https://doi.org/10.1109/52.819971>

- Williams, L. (2010). Agile software development methodologies and practices. In M. V. Zelkowitz (Ed.), *Advances in computers* (Vol. 80, pp. 1–44). Elsevier. [https://doi.org/10.1016/S0065-2458\(10\)80001-4](https://doi.org/10.1016/S0065-2458(10)80001-4)
- Williams, L., & Cockburn, A. (2003). Agile software development: It's about feedback and change. *Computer*, 36(6), 39-43. <https://doi.org/10.1109/MC.2003.1204373>
- Wysocki, R. K. (2006). *Effective software project management*. Wiley Publishing.
- Xu, B. (2009). Towards high quality software development with Extreme Programming methodology: Practices from real software projects. *2009 International Conference on Management and Service Science*, Beijing, China, 1–4. <https://doi.org/10.1109/ICMSS.2009.5302042>
- Yaşlıoğlu, M. M. (2017). Sosyal bilimlerde faktör analizi ve geçerlilik: Keşfedici ve doğrulayıcı faktör analizlerinin kullanılması. *İstanbul Üniversitesi İşletme Fakültesi Dergisi*, 46, 74-85.
- Yeşilyurt, K. (2022). *Çevik proje yönetim yaklaşımlarının proje başarısı üzerindeki etkilerinin incelenmesi* (Tez No. 735189) [Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi]. YÖK Ulusal Tez Merkezi. <https://tez.yok.gov.tr>
- Yitmen, M. ve Gürbüz, E. (2018). *SCRUM: Usta sorulara uzman cevaplar* (1. baskı). Seçkin Yayıncılık.
- Zayat, W., & Senvar, O. (2020). Framework study for agile software development via Scrum and Kanban. *International Journal of Innovation and Technology Management*, 17(4), 2030002. <https://doi.org/10.1142/S0219877020300025>
- Zhang, F., Abdullah, N. A. S., & Rosli, M. M. (2024). Critical success factors of agile software projects: A review. *Engineering, Technology & Applied Science Research*, 14(5), 16866-16873. <https://doi.org/10.48084/etasr.8358>

EKLER

EK 1 Anket Formu

ÇEVİK YAZILIM GELİŞTİRMEDE İNSAN VE SÜREÇ FAKTÖRLERİNİN PROJE BAŞARISINA ETKİSİNDE ÇEVİK PRATİKLERİN DÜZENLEYİCİ ROLÜNÜN ARAŞTIRILMASI ANKETİ

Değerli Katılımcı,

Bu anket, Atatürk Üniversitesi Sosyal Bilimler Enstitüsü Yönetim Bilişim Sistemleri Anabilim dalında yürütülen doktora tezi kapsamında hazırlanmıştır. Vereceğiniz yanıtlar kesinlikle gizli tutulacak ve sadece etik kurallar çerçevesinde bu tez kapsamında bilimsel amaçlar için kullanılacaktır. Sorularını yanıtlamanız yaklaşık 15 dakikanızı alacaktır. Değerli vaktinizi ayırıp, araştırmaya katıldığınız için çok teşekkür ederiz.

Öğr. Gör. Abdullah BALCI
Doktora Öğrencisi

Dr. Öğr. Üyesi Mustafa KESKİNKILIÇ
Tez Danışmanı

Bölüm 1 Kişi ve Firma Bilgileri

Anketin bu bölümünde anketi cevaplayan kişinin demografik özelliklerini ve çalıştığı firmayla ilgili bilgileri belirlemeye yönelik sorular yer almaktadır.

Lütfen size uygun seçeneği X işareti ile belirtiniz.

1. Cinsiyetiniz:

Erkek () Kadın ()

2. Yaşınız:(Lütfen Belirtiniz)

3. Eğitim Düzeyiniz:

Lise () Önlisans () Lisans () Yüksek lisans () Doktora ()

4. Çevik yazılım geliştirmedeki toplam iş tecrübeniz (deneyiminiz):

1 yıl ve az () 4-6 yıl () 10 yıl ve üzeri ()
1-3 yıl () 7-9 yıl ()

5. Çalıştığınız firma hangi sektörde faaliyet göstermektedir?

Bankacılık/Finans ()	Eğitim ()	E- Ticaret ()
Bilgi Teknolojileri ()	Enerji ()	İmalat ()
Danışmanlık ()	Telekomünikasyon ()	İnşaat ()
Sağlık/Tıp ()	Savunma Sanayi ()	Diğer (Belirtiniz)

6. Çalıştığınız firmadaki mevcut pozisyonunuz (göreviniz)

Proje Yöneticisi ()	Müşteri/Ürün Sahibi ()
Takım Lideri ()	Çevik Koç ()
Scrum Master ()	Takım Üyesi ()
Yazılım Geliştirici ()	Diğer () Belirtiniz

7. Çalıştığınız firmada hangi çevik yöntem/yöntemler kullanılmaktadır? (Birden fazla işaretlenebilir)

Scrum ()	XP (Uç Programlama) ()
Kanban ()	Scrum/XP Karması ()
Scrumban ()	Diğer () Belirtiniz

Bölüm 2 İnsan ve Süreç Faktörlerine İlişkin İfadeler

Bu bölüm, çevik yöntemlerin kullanıldığı yazılım geliştirme projelerinin insan ve süreç başarı faktörlerini içeren ifadeler yer almaktadır.

İfadelerin her birine verilecek yanıtlar

(1) Kesinlikle katılmıyorum, (2) Katılmıyorum, (3) Biraz katılmıyorum, (4) Kararsızım, (5) Biraz katılıyorum, (6) Katılıyorum, (7) Kesinlikle katılıyorum olarak değerlendirilmelidir.

Lütfen soruları geliştirdiğiniz son çevik yazılım geliştirme projesini göz önüne alarak cevaplandırınız.

	İnsan ve Süreç Faktörlerine İlişkin İfadeler	KATILMA DÜZEYİ						
		1	2	3	4	5	6	7
1	Seçilen proje ekibi üyeleri yüksek teknik yeterliliğe ve uzmanlığa (problem çözme, programlama, alan bilgisi) sahipti.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
2	Proje ekip üyeleri yüksek motivasyona sahipti ve kendilerini proje başarısına adanmışlardı.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
3	Proje yönetimi, çevik ilkeler ve süreçler konusunda bilgiliydi.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
4	Proje yönetimi, esnek ve/veya uyarlanabilir yönetim tarzına sahipti. Örneğin yaratıcı, esnek çalışma ortamını teşvik ederken, projenin çeşitli bölümleri arasındaki karşılıklı etkileşimlerden faydalanarak çalışanları sürekli öğrenme ve adaptasyona yönlendirirdi.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
5	Proje, konu ve çevik süreçler hakkında eğitim de dâhil olmak üzere ekibe uygun teknik eğitim sağladı.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
6	Proje yönetiminin müşteriyle iyi bir ilişkisi vardı.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
7	Proje güçlü bir müşteri bağlılığına ve varlığına sahipti. Çalışma ortamında proje ekibinin bir üyesi olarak sıkı ve tam zamanlı çalışan en az bir müşteri temsilcisi vardı.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
8	Projedeki müşteri temsilcisi, proje gereksinimlerini ve değişikliklerini onaylama, onaylamama ve önceliklendirme gibi kararlar alma konusunda tam yetki ve bilgiye sahipti.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
9	Projede çevik odaklı gereksinim süreci takip edildi. Örneğin başlangıç gereksinimlerinin çok yüksek düzeyde belirlenirken, proje ilerledikçe yorum ve uyarılama için çok fazla alan bırakıldı.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
10	Projede çevik proje yönetimi takip edildi. Örneğin planlar genellikle ayrıntılı olarak belgelenmemekte; değişiklikler kolayca kabul edilmekte ve proje planına dâhil edildi.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
11	Projede çevik odaklı konfigürasyon yönetimi süreci takip edildi. Örneğin, yeniden düzenleme çabalarına ve sık yapılanmaya uyum sağlamak için iyi sürüm kontrolü veya kaynak kod yönetimi yapıldı.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
12	Proje yöneticisi, çevik dostu bir ilerleme izleme mekanizması takip etti. Örneğin, belgeleme aşamaları veya iş dökmü yapısı yerine esnek zaman sınırlaması veya hızlı ilerleme ölçüm teknikleri kullandı.	(1)	(2)	(3)	(4)	(5)	(6)	(7)

13	Proje güçlü bir iletişim odağına ve titiz bir iletişim programına sahipti. Yani yüz yüze ve anlık iletişim kanalları (ekip üyeleri arasında, ekip ile yönetim arasında ve ekip ile müşteriler arasında), günlük ayaküstü toplantıları, dönem toplantılar vardı.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
14	Proje, düzenli bir çalışma programı sağladı. Örneğin haftalık 40 saat çalışma vardı ve fazla mesai yapılmadı.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
15	Proje kapsamı ve hedefleri iyi tanımlanmıştı.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
16	Projede önceden ayrıntılı maliyet değerlendirmesi yapıldı ve onaylandı.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
17	Projede çevik yöntem kullanımı için önceden risk analizi yapıldı ve değerlendirildi.	(1)	(2)	(3)	(4)	(5)	(6)	(7)

Bölüm 3 Çevik Pratiklerin Kullanımına İlişkin İfadeler

Bu bölüm, çevik pratiklerin kullanımına ilişkin ifadeler yer almaktadır. İfadelerin her birine verilecek yanıtlar

(1) Kesinlikle katılmıyorum, (2) Katılmıyorum, (3) Biraz katılmıyorum, (4) Kararsızım, (5) Biraz katılıyorum, (6) Katılıyorum, (7) Kesinlikle katılıyorum olarak değerlendirilmelidir.

Lütfen soruları geliştirdiğiniz son çevik yazılım geliştirme projesini göz önüne alarak cevaplandırınız.

	Çevik Pratiklerin Kullanımına İlişkin İfadeler	KATILMA DÜZEYİ						
		1	2	3	4	5	6	7
1	Her çalışma döngüsünün başında ekip ve iş sahipleri, çalışma döngüsü boyunca nelerin teslim edileceği konusunda hemfikirlerdir.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
2	Ekip, bir çalışma döngüsünde işin ne kadarının tamamlanabileceğini belirtir.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
3	Ekip, her bir özelliğin tamamlanması için gereken iş miktarını tahmin eder.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
4	Ekibimiz hem müşterilerin sürümlerle ilgili iş kararları almasına hem de teknik kişilerin sürümlerle ilgili teknik kararlar almasına olanak tanır.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
5	Ekip, her gün projede neler olup bittiğini tartışmak için kısa bir toplantı yapar.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
6	Tüm ekip üyeleri üzerinde çalıştıklarını ekiple her gün paylaşır.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
7	Ekip, sorunları günlük olarak birlikte tartışır.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
8	Ekip düzenli olarak önceki çalışmalar üzerinde düşünür ve ekip performansını iyileştirmenin yollarını arar.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
9	Her çalışma döngüsünün sonunda ekip, son çalışma döngüsü sırasında kendisine "neyin iyi gittiğini" sorar.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
10	Her çalışma döngüsünün sonunda, ekip bir sonraki döngüde "neyi iyileştirilebileceğini" sorar.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
11	"Gerçek" kodu test etmek için özel olarak yazılmış ayrı bir "test" kodu setimiz vardır.	(1)	(2)	(3)	(4)	(5)	(6)	(7)

12	Her programcı, yazdığı kod için birim testleri yazmaktan sorumludur.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
13	Programcılar, değişiklikleri kayıt etmeden önce başarılı bir şekilde çalışana kadar bir dizi birim testini kişisel olarak yürütmekten sorumludur.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
14	Ekip üyeleri, kod değişikliklerini mümkün olan en kısa sürede entegre eder.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
15	Ekip, yazılımı günde birkaç kez otomatik olarak yeniden oluşturan (rebuild) bir sürece sahiptir.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
16	Ekip, kodun otomatik olarak derlenmesi, dağıtılması veya test edilmesiyle ilgili herhangi bir sorun hakkında otomatik olarak bilgilendirilir.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
17	Kodumuzun adlandırılması ve yapısı tutarlıdır.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
18	Ekibimiz tutarlı kod biçimlendirme için standartlar kullanır.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
19	Tüm kodların tutarlı bir şekilde biçimlendirilmesi takımın başarısı için önemlidir.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
20	İhtiyaç gördüğümüzde, daha önce yazdığımız kodun tasarımını iyileştiririz.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
21	Ekibin her üyesi, değişiklik yaparken kodun yapısını iyileştirmeye çalışır.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
22	Kullanılmayan bir kod bulursak, kodu kaldırırız.	(1)	(2)	(3)	(4)	(5)	(6)	(7)

Bölüm 4 Çevik Proje Başarı Algısına İlişkin İfadeler

Bu bölüm, elinizdeki çevik yazılım geliştirme projesinin algılanan başarı düzeyinizin özelliklerini içerir. İfadelerin her birine verilecek yanıtlar

(1) Kesinlikle katılmıyorum, (2) Katılmıyorum, (3) Biraz katılmıyorum, (4) Kararsızım, (5) Biraz katılıyorum, (6) Katılıyorum, (7) Kesinlikle katılıyorum olarak değerlendirilmelidir.

Lütfen soruları geliştirdiğiniz son çevik yazılım geliştirme projesini göz önüne alarak cevaplandırınız.

	Çevik Proje Başarı Algısına İlişkin İfadeler	KATILMA DÜZEYİ						
		1	2	3	4	5	6	7
1	Proje, proje çıktısının veya ortaya çıkan yazılım ürününün kalitesi açısından başarılı oldu.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
2	Proje, kapsam ve projenin gereksinimlerini karşılama açısından başarılı oldu.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
3	Proje, projenin tamamlanma takvimi açısından başarılı oldu.	(1)	(2)	(3)	(4)	(5)	(6)	(7)
4	Proje, maliyetler ve çabalar açısından bütçenin altında veya tahminler dâhilinde başarılı oldu.	(1)	(2)	(3)	(4)	(5)	(6)	(7)

EK 2 Ölçeklerin Kullanım İzni

“Çevik Başarı Faktörleri” ve “Çevik Yazılım Geliştirme Projesinde Algılanan Başarı” ölçeklerinin kullanımına yönelik izin görüntüsüdür.


Abdullah Balci
3 days ago

Dear Dac- Buu Cao

I am working as a lecturer at Tokat Gaziosmanpaşa University in Turkey. I am also continuing my doctorate education in management information systems. With your permission, I would like to use the scale in your article in my doctoral thesis.

Best regards.

Abdullah Balci
abdullah.balci@gop.edu.tr
Tokat Gaziosmanpaşa University


Dac-Buu Cao to you
 3 days ago

Dear Mr. Balci,

You have my permission as requested.
Best wishes on your research.

Regards,
Dac-Buu Cao

“Çevik Pratiklerin Kullanımı Ölçeği”nin kullanımına yönelik izin görüntüsüdür.

AB

abdullah balci

Kime: JFTRIPP@clemson.edu

20.06.2022 Pzt 14:04

Dear Profoser John F. Tripp

I am working as a lecturer at Tokat Gaziosmanpaşa University in Turkey. I am also continuing my doctorate education in management information systems. With your permission, I would like to use the scale in your article in my doctoral thesis. Can I get information from you so that it will be more understandable when I translate the scale into Turkish?

Does the word "business owners" in the "ID1" item in the "iterative delivery" dimension refer to the company that developed the software or the company that requested the software?

Similarly, I request your help in the words "business people" and "technical people" in the "ID4" article.

Best regards.

The article I want to use with your permission:

Tripp, J. F., Riemenschneider, C., & Thatcher, J. B. (2016). Job satisfaction in agile development teams: Agile development as work redesign. *Journal of the Association for Information Systems*, 17(4), 1.

Abdullah Balci

abdullah.balci@gop.edu.tr

JT

John Frederick Tripp <JFTRIPP@clemson.edu>

Kime: Siz

21.06.2022 Sal 02:33

That's fine.

"Business owner" is the term that agile teams use to refer to the person for whom the system is being built. The one who directs the requirements and funding - not necessarily the person who owns a business.

Dr. John F. Tripp

Assistant Professor
Department of Management
409-E Wilbur O. and Ann Powers College of Business
Clemson University
Clemson SC 29634-1305
jftripp@clemson.edu
clemson.zoom.com/jv/jftripp

EK 3 Etik Kurul Onay Belgesi



T.C.
ATATÜRK ÜNİVERSİTESİ
Sosyal ve Beşeri Bilimler Etik Kurul Başkanlığı

Sayı : E.88656144-000- 2200383449

ETİK KURUL ONAY BELGESİ

KARAR BİLGİLERİ	Oturum Sayısı : 21 Karar No : 320	Toplantı Tarihi: 21.11.2022
	Aşağıda bilgileri verilen proje ile ilgili çalışmanın, etik ilkeler açısından değerlendirilmesi isteği ile ilgili husus görüşüldü. Yapılan görüşmelerden sonra; söz konusu projeye alakalı yapılacak çalışma için, araştırmanın gerekçe, amaç, yaklaşım ve yöntemleri dikkate alınarak konuyla ilgili çalışmanın gerçekleştirilmesinde <i>bilim etiği yönünden sakınca bulunmadığına</i>, Etik Kurulu oy birliği ile karar vermiştir.	
PROJE - TEZ BİLGİLERİ	Proje- Tez Danışmanı : <i>Dr.Öğr.Üyesi Mustafa KESKİNKILIÇ</i> Proje- Tez Yürütücüsü : <i>Doktora Öğrencisi Abdullah BALCI</i> Proje -Tez Konusu: "ÇEVİK YAZILIM GELİŞTİRMEDE İNSAN VE SÜREÇ FAKTÖRLERİNİN PROJE BAŞARISINA ETKİSİNDE ÇEVİK PRATİKLERİN DÜZENLEYİCİ ROLÜNÜN ARAŞTIRILMASI"	
	SOSYAL VE BEŞERİ BİLİMLER ETİK KURULU	
	Unvanı/Adı Soyadı	Görevi
	<i>Prof.Dr. Vedat KAYA</i>	<i>Etik Kurul Başkanı.</i>
	<i>Prof.Dr. Cavit YEŞİLYURT</i>	<i>Etik Kurul Üyesi</i>
	<i>Prof.Dr. Dilaver DÜZGÜN</i>	<i>Etik Kurul Üyesi</i>
	<i>Prof.Dr. Gökalp Nuri SELÇUK</i>	<i>Etik Kurul Üyesi</i>
	<i>Prof.Dr.Ufuk ŞİMŞEK</i>	<i>Etik Kurul Üyesi</i>
	<i>Prof.Dr. Musa BİLGİZ</i>	<i>Etik Kurul Üyesi</i>
	<i>Prof.Dr. Adem YILMAZ</i>	<i>Etik Kurul Üyesi</i>



ÖZ GEÇMİŞ

Kişisel Bilgiler	
Adı Soyadı	Abdullah BALCI
Doğum Yeri ve Tarihi	
Eğitim Durumu	
Lisans Öğrenimi	Gebze Yüksek Teknoloji Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği
Y. Lisans Öğrenimi	Afyon Kocatepe Üniversitesi Fen Bilimleri Enstitüsü İnternet ve Bilişim Teknolojileri Yönetimi
Bildiği Yabancı Diller	İngilizce
Bilimsel Faaliyetleri	
İş Deneyimi	
Stajlar	
Projeler	
Çalıştığı Kurumlar	
İletişim	
E-Posta Adresi	
Tarih	20.12.2024