T.C.
MANİSA CELAL BAYAR ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ
YAZILIM MÜHENDİSLİĞİ ANABİLİM DALI
YÜKSEK LİSANS PROGRAMI

ÇEVİK YAZILIM GELİŞTİREN ORGANİZASYONLARDA CMMI-DEV V2.0 İLE YAZILIM SÜREÇLERİNİN İYİLEŞTİRİLMESİ

Sedanur AKKOYUN

Danışman
Doç. Dr. Fatih YÜCALAR

MANİSA–2025

SEDANUR
AKKOYUN

ÇEVİK YAZILIM GELİŞTİREN ORGANİZASYONLARDA CMMI-DEV V2.0 İLE YAZILIM SÜREÇLERİNİN İYİLEŞTİRİLMESİ

2025

TAAHHÜTNAME

Bu tezin Manisa Celal Bayar Üniversitesi Lisansüstü Eğitim Enstitüsü Yazılım Mühendisliği Ana Bilim Dalında akademik ve etik kurallara uygun olarak yazıldığını ve kullanılan tüm literatür bilgilerinin referans gösterilerek tezde yer aldığını, tamamen kendi çalışmam olduğunu, her alıntıya kaynak gösterdiğimi, tezin yazımında akademik ve etik kurallara aykırı herhangi bir yapay zekâ ve program kullanmadığımı beyan ederim.

Sedanur AKKOYUN

ÖZET

Yüksek Lisans Tezi

Sedanur AKKOYUN

Manisa Celal Bayar Üniversitesi
Lisansüstü Eğitim Enstitüsü
Yazılım Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Fatih YÜCALAR

Günümüzde yazılım sektöründe müşteri ihtiyaçlarının hızla değişmesi organizasyonların çevik kalmasını zorunlu kılarken, sürdürülebilirlik, kalite ve izlenebilirlik gibi kurumsal gereklilikler de daha yapılandırılmış süreç modellerine olan ihtiyacı artırmaktadır. Bu bağlamda, çevik yazılım geliştirme yaklaşımlarının sağladığı esneklik ile CMMI-DEV v2.0 gibi yetenek olgunluk modellerinin sunduğu yapısal bütünlüğün nasıl bir araya getirilebileceği sorusu bu tez çalışmasının çıkış noktasını oluşturmuştur. Çalışmanın temel amacı, çevik yöntemleri uygulayan yazılım organizasyonlarının kurumsal düzeyde daha tutarlı ve sürdürülebilir süreçler yürütebilmesine katkı sunacak bütünleşik bir yaklaşım geliştirmektir.

Bu doğrultuda, öncelikle çevik yöntemler ile CMMI-DEV v2.0 modeli ayrı ayrı ele alınmış; literatür taraması yoluyla her iki yaklaşımın temel ilkeleri, güçlü yönleri ve sınırlılıkları analiz edilmiştir. Ardından, çevik yöntemlerin esnek yapısını koruyarak CMMI-DEV v2.0'ın yapılandırılmış uygulama alanlarını bu yapıya entegre eden hibrit bir model önerilmiştir. Modelin sektörel uygulanabilirliğini değerlendirmek amacıyla, Türkiye'de çeşitli sektörlerde faaliyet gösteren 12 yazılım organizasyonu ile bir anket çalışması gerçekleştirilmiştir. Bu anket aracılığıyla, organizasyonların yazılım geliştirme sürecine ilişkin yaklaşımları ve çeşitli uygulama alanlarında karşılaştıkları eksikliklere dair algıları analiz edilmiştir.

Elde edilen bulgular, çevik yöntemlerin operasyonel düzeyde etkin olduğunu; ancak karar analizi, kök neden analizi, süreç yönetimi ve yönetim desteği gibi alanlarda yapılandırılmış uygulamaların yetersiz kaldığını ortaya koymuştur. Ayrıca, süreç olgunluk modellerine yönelik farkındalığın sektörde genel olarak düşük seviyede olduğu gözlemlenmiştir. Bu veriler ışığında, önerilen hibrit yaklaşımın yalnızca teorik değil, aynı zamanda pratik düzeyde de önemli bir ihtiyaca yanıt verdiği

sonucuna ulaşılmıştır. Çalışma, Türkçe literatürde bu iki yaklaşımı bir araya getiren nadir örneklerden biri olarak hem akademik hem de sektörel düzeyde katkı sunmayı amaçlamaktadır.

Anahtar Kelimeler: Çevik Yazılım Geliştirme, CMMI-DEV v2.0, Scrum, Süreç İyileştirme, Süreç Olgunluğu

2025, 129 sayfa

ABSTRACT

M.Sc. Thesis

Sedanur AKKOYUN

Manisa Celal Bayar University
Graduate School of Education
Department of Software Engineering

Supervisor: Doç. Dr. Fatih YÜCALAR

In today's software industry, the rapid changes in customer requirements compel organizations to remain agile; meanwhile, corporate necessities such as sustainability, quality, and traceability increase the demand for more structured process models. Within this context, the question of how the flexibility provided by agile software development approaches can be combined with the structural integrity offered by capability maturity models such as CMMI-DEV v2.0 constitutes the starting point of this thesis study. The main objective of the study is to develop an integrated approach that contributes to software organizations applying agile methods to conduct more consistent and sustainable processes at the organizational level.

Accordingly, agile methods and the CMMI-DEV v2.0 model were initially addressed separately; through a literature review, the fundamental principles, strengths, and limitations of both approaches were analyzed. Subsequently, a hybrid model was proposed that integrates the structured process areas of CMMI-DEV v2.0 into the flexible nature of agile methodologies. To evaluate the sectoral applicability of the model, a survey was conducted with 12 software organizations operating in various sectors in Turkey. Through the survey, organizations' approaches to software development process implementations and perceptions regarding deficiencies in various process areas were analyzed.

The findings revealed that although agile methods are effective at the operational level, structured practices remain insufficient in areas such as decision analysis, root cause analysis, process management, and management support. Additionally, awareness of process maturity models was generally observed to be low across the sector. In light of these data, it was concluded that the proposed hybrid

approach responds not only theoretically but also practically to an important need. This study aims to contribute both academically and sectorally as one of the rare examples in the Turkish literature combining these two approaches.

Keywords: Agile Software Development, CMMI-DEV v2.0, Scrum, Process Improvement, Process Maturity

2025, 129 pages

ÖNSÖZ VE TEŞEKKÜR

Günümüzde, özellikle havacılık ve savunma sanayii gibi yüksek kalite ve güvenlik gereksinimleri olan sektörlerde faaliyet gösteren yazılım organizasyonlarında, çevik yazılım geliştirme yaklaşımlarının süreç olgunluk modelleriyle birlikte kullanımı giderek yaygınlaşmaktadır. Hem ürün hem de süreç kalitesine eşzamanlı katkı sağlayan bu bütünleşik yaklaşımın, farklı sektörlerde hizmet veren ve yazılım geliştirme faaliyetleri yürüten organizasyonlar tarafından da uygulanmasının yararlı olacağı kanaatindeyim. Bu doğrultuda hazırlanan bu tez çalışmasının, yazılım geliştirme yöntemi olarak Scrum uygulayan; süreçlerini standardize etmeyi ve iyileştirmeyi amaçlayan organizasyonlara, CMMI-DEV v2.0 modelini benimseme sürecinde yol gösterici olmasını temenni ediyorum.

Tez sürecim boyunca bilgi ve deneyimiyle bana yol gösteren, akademik yönlendirmeleriyle çalışmamın her aşamasında destek olan değerli danışmanım Doç. Dr. Fatih YÜCALAR'a en içten teşekkürlerimi sunuyorum.

Ayrıca, yürütülen anket çalışmasına katkı sağlayarak bu çalışmanın saha verileriyle desteklenmesine olanak tanıyan tüm katılımcı kurumlara ve temsilcilerine de teşekkür ederim.

Son olarak, akademik yaşamım boyunca her daim yanımda olan, maddi manevi desteğini hiçbir zaman esirgemeyen, başta sevgili anne ve babam olmak üzere tüm aileme gönülden teşekkür ederim.

Sedanur AKKOYUN
Manisa, 2025

SİMGELER VE KISALTMALAR DİZİNİ

CAR	Causal Analysis and Resolution (Nedensel Analiz ve Çözüm)
CM	Configuration Management (Konfigürasyon Yönetimi)
CMM	Capability Maturity Model (Yetenek Olgunluk Modeli)
CMMI	Capability Maturity Model Integration (Yetenek Olgunluk Modeli Entegrasyonu)
DAR	Decision Analysis and Resolution (Karar Analizi ve Çözümü)
DP	Defect Prevention (Hata Önleme)
EDP	Engineering and Developing Products (Mühendislik ve Ürün Geliştirme)
ENQ	Ensuring Quality (Kaliteyi Sağlama)
EST	Estimating (Tahmin)
GOV	Governance (Yönetim)
IBM	International Business Machines (Uluslararası İş Makineleri)
IC	Intergroup Coordination (Ekipler Arası Koordinasyon)
II	Implementation Infrastructure (Uygulama Altyapısı)
IM	Integrated Software Management (Entegre Yazılım Yönetimi)
IMP	Improving Performance (Performans İyileştirme)
IPM	Integrated Project Management (Entegre Proje Yönetimi)
IPPD-CMM	Integrated Product and Process Development (Entegre Ürün Geliştirme Yetenek Olgunluk Modeli)
MA	Measurement and Analysis (Ölçüm ve Analiz)
MBR	Managing Business Resilience (İş Sürekliliğini Yönetme)
MC	Monitor and Control (İzleme ve Kontrol)
MPM	Managing Performance and Measurement (Performans Yönetimi ve Ölçümü)
MWF	Managing the Workforce (İşgücünü Yönetme)

OOP Organizational Process Performance (Organizasyonel Süreç Performansı)

OPD Organizational Process Definition (Organizasyonel Süreç Tanımı)

OPF Organizational Process Focus (Organizasyonel Süreç Odağı)

OPM Organizational Performance Management (Organizasyonel Performans Yönetimi)

OT Organizational Training (Organizasyonel Eğitim)

PA Practice Area (Uygulama Alanı)

PAD Process Asset Development (Süreç Varlığı Geliştirme)

PC Process Change Management (Süreç Değişim Yönetimi)

P-CMM People Capability Maturity Model (İnsan Yetenek Olgunluk Modeli)

PCM Process Management (Süreç Yönetimi)

PD Organization Process Definition (Organizasyon Süreç Tanımı)

PE Software Product Engineering (Yazılım Ürün Mühendisliği)

PF Organization Process Focus (Organizasyon Süreç Odaklılığı)

PI Product Integration (Ürün Entegrasyonu)

PLAN Planning (Planlama)

PMC Project Monitoring and Control (Proje İzleme ve Kontrol)

PMW Planning and Managing Work (İş Planlama ve Yönetme)

PP Project Planning (Proje Planlama)

PPQA Process and Product Quality Assurance (Süreç ve Ürün Kalite Güvencesi)

PQA Process Quality Assurance (Süreç Kalite Güvencesi)

PR Peer Reviews (Eşdeğer Gözden Geçirme)

PT Software Project Tracking (Yazılım Projesi Takibi)

QA Software Quality Assurance (Yazılım Kalite Güvencesi)

QM Software Quality Management (Yazılım Kalite Yönetimi)

QP Quantitative Process Management (Nicel Süreç Yönetimi)

QPM Quantitative Project Management (Nicel Proje Yönetimi)

RD Requirements Development (Gereksinim Geliştirme)

RDM Requirements Development and Management (Gereksinim Geliştirme ve Yönetimi)

REQM Requirements Management (Gereksinim Yönetimi)

RM Requirements Management (Gereksinim Yönetimi)

RSK Risk and Opportunity Management (Risk ve Fırsat Yönetimi)

RSKM Risk Management (Risk Yönetimi)

SA-CMM Software Acquisition Capability Maturity Model (Yazılım Tedarik Yetenek Olgunluk Modeli)

SAM Supplier Agreement Management (Tedarikçi Sözleşme Yönetimi)

SE-CMM System Engineering Capability Maturity Model (Sistem Mühendisliği Yetenek Olgunluk Modeli)

SHP Sustaining Habit and Persistence (Alışkanlığı ve Sürekliliği Sürdürme)

SI Supporting Implementation (Uygulama Destekleme)

SM Software Subcontract Management (Yazılım Alt Yüklenici Yönetimi)

SMS Selecting and Managing Suppliers (Tedarikçi Seçme ve Yönetme)

SW-CMM Software Capability Maturity Model (Yazılım Yetenek Olgunluk Modeli)

TM Technology Change Management (Teknoloji Değişim Yönetimi)

TP Training Program (Eğitim Programı)

TS Technical Solution (Teknik Çözüm)

VAL Validation (Geçerleme)

VER Verification (Doğrulama)

VV Verification and Validation (Doğrulama ve Geçerleme)

XP Extreme Programming (Uç Programlama)

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

İÇİNDEKİLER

GİRİŞ

Modern çağın hızla ilerleyen teknolojik gelişmeleri ve yoğun rekabet koşulları, yazılım geliştirme süreçlerinde esneklik ve hız gereksinimlerini ön plana çıkarmıştır. Müşteri taleplerinin ve değişim isteklerinin anında karşılanabilmesi, yazılım projelerinin başarısında kritik bir faktör haline gelmiştir. Bu bağlamda, yazılım geliştirme metodolojilerinin bu hızlı değişimlere uyum sağlayabilmesi ve aynı zamanda maliyet etkinliğini koruyabilmesi büyük önem taşımaktadır.

Bu ihtiyaçlara yanıt vermek üzere ortaya çıkan çevik yazılım geliştirme yaklaşımları, esneklik, müşteri odaklılık ve hızlı teslimat gibi temel prensipler doğrultusunda yazılım süreçlerine yön vermektedir. Özellikle Scrum çerçevesi, küçük ve fonksiyonel ekiplerle çalışarak kısa sürelerde çalışabilir yazılım bileşenleri üretmeye odaklanmakta; müşteri geri bildirimlerinin hızla alınmasını ve yeni gereksinimlerin sürece anında entegre edilmesini mümkün kılmaktadır. Bu sayede yazılım projeleri daha çevik, uyumlu ve verimli bir şekilde ilerleyebilmektedir.

Ancak, çevik yaklaşımlar tek başına süreçlerin dokümantasyonu, süreç olgunluğu ve standartlaşması gibi alanlarda bazı eksikliklere yol açabilmektedir. Bu durum, özellikle büyük ölçekli ve karmaşık projelerde yönetimsel zorlukların artmasına neden olmaktadır. Dolayısıyla, çevik metodolojilerin sağladığı avantajları korurken aynı zamanda süreç olgunluğunu ve kalite standartlarını da gözeten bütüncül yaklaşımlara ihtiyaç duyulmaktadır.

Bu noktada, yazılım süreçlerinin yetkinliğini değerlendirme ve iyileştirme amacıyla geliştirilen Capability Maturity Model Integration for Development (CMMI-DEV) v2.0, organizasyonlara kapsamlı ve sistematik bir yol haritası sunmaktadır. CMMI-DEV v2.0, süreçlerin yapılandırılması, standartlaştırılması ve sürekli iyileştirilmesi doğrultusunda rehberlik ederek, çevik uygulamaların eksik kaldığı alanlarda tamamlayıcı bir rol üstlenmektedir.

Bu çalışmada, çevik metodolojiler ile süreç olgunluk modelleri arasında kurulan entegrasyonun organizasyonel süreç yetkinliğine katkısı ele alınmaktadır. Literatürde, özellikle Türkçe kaynaklarda, Scrum ile CMMI-DEV v2.0

entegrasyonuna ilişkin rehber niteliğindeki çalışmaların sınırlı olduğu görülmüş; bu boşluğu doldurmak amacıyla söz konusu çalışma gerçekleştirilmiştir.

Yalçıner ve arkadaşları (2022) tarafından gerçekleştirilen çalışmada, Scrum ve CMMI-DEV v2.0'ın proje yönetimi ve kalite güvence uygulama alanları arasında kapsamlı bir eşleştirme yapılmış ve bu alanlardaki güçlü yönler ile boşluklar detaylı biçimde ortaya konmuştur. Elde edilen bulgular, proje yönetimi ve kalite güvence alanlarının, Scrum ile tam anlamıyla %63,9 oranında, kısmen ise %30,7 oranında karşılandığını ortaya koymaktadır [1].

Henriquez ve arkadaşları (2022), CMMI-DEV v2.0 ile çevik yaklaşımlar arasındaki uyumu incelemiş, 31 çevik unsurun CMMI-DEV v2.0 pratikleriyle ilişkili olduğunu göstermiştir. Özellikle olgunluk seviyeleri 2 ve 3'teki pratiklerin %41'inin çevik unsurlarla eşleştiği belirtilmiştir. Ancak çalışma, CMMI-DEV v2.0'ın tüm uygukama alanlarını kapsamadığından, Yönetim (GOV), Organizasyonel Eğitim (OT) ve Uygulama Altyapısı (II) gibi kritik alanlarda eksiklikler olduğu vurgulanmıştır. Yazarlar, daha kapsamlı ve uygulamalı araştırmaların bu boşlukları doldurabileceğini ifade etmektedir [2].

De Sousa Morais (2019) tarafından yapılan çalışmada, CMMI-DEV v1.3 süreç alanları ile çevik en iyi uygulamalar arasında bir eşleştirme yapılmış ve bu iki yaklaşımın entegrasyonunda uyumluluk ve uyumsuzlukların anlaşılmasına katkı sağlanmıştır. Çalışma, çevik yöntemlerin olgun organizasyonlara esneklik katarken, CMMI'ın çevik organizasyonların süreç kalitesini artırmasına yardımcı olabileceği sonucuna ulaşarak, bu iki yaklaşımın birlikte kullanılmasının yazılım geliştirmede rekabet gücünü artıracağına işaret etmektedir [3].

Bu çalışmanın temel amacı; yazılım geliştirme yaklaşımı olarak Scrum'ı benimseyen ve süreçlerini standardize etmeyi hedefleyen organizasyonlara, CMMI-DEV v2.0 modelini etkin biçimde entegre edebilmeleri için yol gösterici bir çerçeve sunmaktır. Bu doğrultuda, çalışmada Scrum pratiklerinin CMMI-DEV v2.0 uygukama alanlarıyla nasıl örtüştüğü analiz edilmekte ve bu örtüşmenin yazılım süreçlerini nasıl iyileştirdiği kapsamlı biçimde ele alınmaktadır.

Sonuç olarak, çevik yaklaşımlar ile CMMI-DEV v2.0 birleşiminden doğan hibrit modelin; süreç yönetimi, standardizasyon ve hem süreç hem de ürün kalitesine

olan etkileri değerlendirilerek, yazılım geliştiren organizasyonlar için sürdürülebilir bir iyileştirme yaklaşımı önerilmektedir.

1. ÇEVİK YAZILIM GELİŞTİRME YAKLAŞIMLARI

Bu bölümde, çevik (agile) yazılım geliştirme metodolojisinin tarihçesi ve temel prensipleri ele alınmıştır. İlk olarak, 1990'ların başında geleneksel yazılım geliştirme yöntemlerinin karşılaştığı zorluklar ve bu zorluklara karşılık olarak çevik yaklaşımların nasıl ortaya çıktığı açıklanmıştır. Devamında, çevik yazılım geliştirme sürecinin temel hedeflerinin neler olduğu, 2001 yılında yayımlanan *Çevik Yazılım Geliştirme Manifestosu* ile çevik yaklaşımın dünya çapında nasıl popüler bir yazılım geliştirme standardı haline geldiği anlatılmıştır. Son olarak, çevik yazılım geliştirme prensiplerinin farklı yazılım geliştirme yöntemlerine nasıl yansıdığına değinilmiştir. Scrum, Extreme Programming (XP), Kanban ve Lean gibi çevik yaklaşımların her birinin, çevik manifestonun temel değerleri doğrultusunda farklı stratejiler sunduğu ve hangi tür projelerde avantaj sağladığı açıklanmıştır.

1.1. Çevik Yazılım Geliştirme Yaklaşımlarının Gelişimi

1990'ların başında temeli atılan çevik yazılım geliştirme metodolojisi, genellikle uzun süreli planlama gerektiren, değişen gereksinimlere uyum sağlamada zorlanan geleneksel yazılım geliştirme yaklaşımlarına (Waterfall, V-Model, Spiral) bir alternatif olarak ortaya çıkmıştır. Ayrıca, müşterilerin talep ettiği yazılım ürünlerine ilişkin sürümlerin belirlenen sürede çıkarılamaması, müşterilerden gelen değişiklik taleplerine geri dönüş sağlanamaması, geliştirilen yazılım ürünleri içerisindeki hataların geç fark edilmesi, yazılım geliştirme süreci içerisinde gelen yeni taleplere göre yazılımın kendi yapısını geliştirememesi gibi çeşitli sorunlarla karşılaşılması, çevik yöntemlerin geliştirilmesini tetiklemiştir [4].

2001 yılında, Amerika'nın Utah eyaletinde bir araya gelen 17 yazılım mühendisliği danışmanının iş birliğiyle *"Çevik Yazılım Geliştirme Manifestosu"* yayımlanmıştır. Manifestoda, yazılım geliştirme sürecinde müşteri memnuniyetine odaklanma, değişen gereksinimlere uyum sağlama, geliştiriciler ile iş ekibinin birlikte çalışması ve bireyler arası iletişimin önemi vurgulanmaktadır. Bu manifestoyla birlikte, çevik yazılım geliştirme yaklaşımı daha da popüler hale gelmiş ve dünya çapında tercih edilen temel yazılım geliştirme yöntemlerinden biri olmuştur. Çevik Manifesto, dört temel değer ve on iki ilkeden oluşmaktadır. Bu değerler, yazılım

geliştirme sürecinde önceliklerin belirlenmesinde yol gösterici niteliktedir. Çevik manifestodaki dört ana değere göre [5]:

- Süreçler ve araçlar yerine bireyler ve etkileşimlere öncelik verilmesi,
- Kapsamlı dokümantasyon yerine çalışan yazılımın ön planda tutulması,
- Sözleşme pazarlıkları yerine müşteri ile işbirliği yapılması,
- Bir plana bağlı kalmaktan ziyade değişime uyum sağlanmasının önceliklendirilmesi gerekmektedir.

İlk değer, süreçler ve araçlar yerine bireyler ve etkileşimlere öncelik verilmesini savunur. Bu, yazılım geliştirme sürecinde araçlar ve süreçler kadar, takım içi iletişim ve iş birliğinin de ne kadar kritik olduğunu ifade eder. İkinci değer ise, kapsamlı dokümantasyon yerine çalışan yazılımın ön planda tutulmasını savunur. Bu yaklaşım, yazılımın işlevselliği ve kullanımının öncelikli olduğuna, aşırıya kaçan dokümantasyonun ise zaman kaybına yol açabileceğine dikkat çeker. Üçüncü değer, sözleşme pazarlıkları yerine müşteri ile iş birliği yapılması gerektiğini belirtir. Bu değer, yazılım geliştirme sürecinde müşteri gereksinimlerinin sürekli olarak gözden geçirilmesi ve iş birliği yoluyla şekillendirilmesi gerektiğini vurgular. Son olarak, bir plana bağlı kalmak yerine değişime uyum sağlanması gerektiği ifade edilir. Bu değer, değişen gereksinimlere hızlı bir şekilde adapte olmanın, sabit bir plana sadık kalmaktan daha önemli olduğunu belirtir. Çevik Manifesto'nun bu dört değeri, yazılım geliştirme süreçlerinde esneklik, verimlilik ve müşteri memnuniyetini ön plana çıkaran bir yaklaşımın temellerini atmaktadır.

Çevik yazılım geliştirme ve proje yönetimi süreçlerinin çerçevelerini oluşturan ve dört değeri destekleyen temel prensipler on iki başlık altında toplanmıştır [6], [7]. Bu prensipler aşağıdaki gibi sıralanabilir:

1. Yazılımın erken ve sürekli teslimatı ile müşteri memnuniyeti sağlanır.
2. Geliştirme süreci boyunca değişen ihtiyaçlar karşılanmalıdır.
3. Çalışan yazılımlar sık aralıklar ile teslim edilmelidir.
4. Proje boyunca geliştiriciler ve diğer paydaşlar işbirliği halinde olmalıdır.
5. Proje ekibi motive edilmeli ve ekip içerisinde güven olmalıdır.
6. Etkili iletişim için yüz yüze görüşümeler tercih edilmelidir.
7. Proje sürecinin öncelikli ölçütü çalışan yazılımdır.
8. Çevik süreçler sabit hızlı, sürdürülebilir geliştirmeyi teşvik eder.

9. Teknik detaylara ve tasarıma gösterilen özen çevikliği artırır.
10. Sadelik esastır.
11. En iyi mimariler, gereksinimler ve tasarımlar kendi kendine organize olmuş takımlardan çıkar.
12. Ekip düzenli aralıklarla nasıl daha verimli olacağını konuşmalı ve buna göre aksiyon almalıdır.

Zaman içinde, çevik yazılım geliştirme prensiplerine dayanan çeşitli yöntem ve modeller ortaya çıkmıştır. Bunlar arasında en yaygın olarak benimsenenler; Scrum [8], Extreme Programming (XP) [9], Kanban [10] ve Lean [11] gibi yaklaşımlardır. Her bir metodoloji, çevik ilkeleri esas alarak yazılım geliştirme süreçlerini daha verimli, hızlı ve esnek hale getirmeyi hedeflemektedir. Bu yöntemler, yazılım geliştirme disiplinine daha etkili ve sürdürülebilir çözümler sunmayı amaçlayan, dinamik ve sürekli iyileştirmeyi esas alan yapılardır.

1.2. Scrum

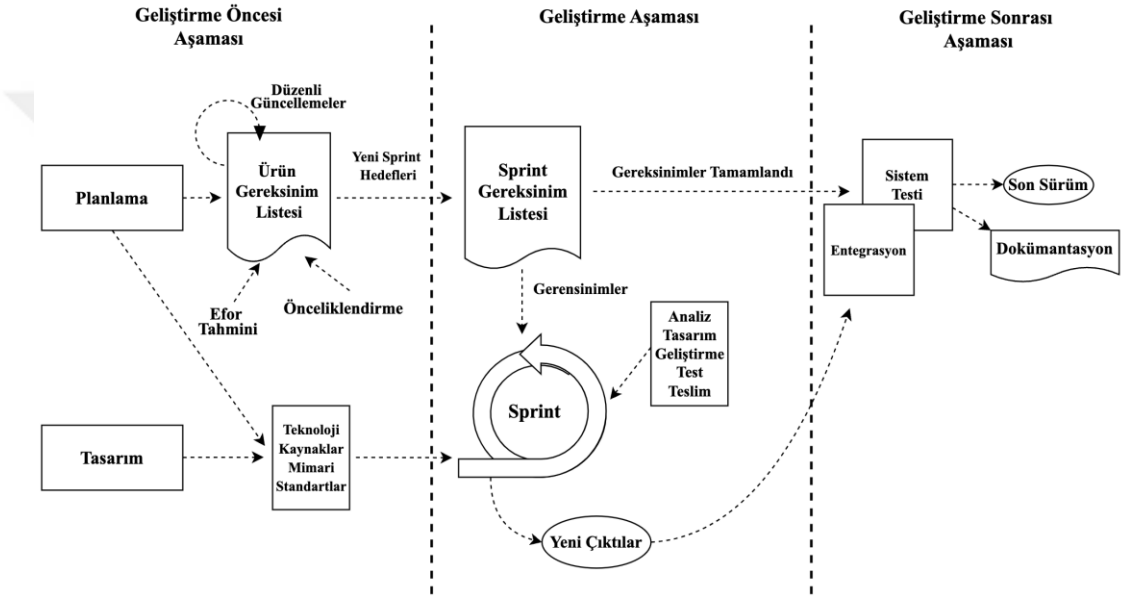
Scrum, 1990'lı yılların başlarında, yazılım geliştirme süreçlerini daha esnek ve hızlı hale getirmek amacıyla Ken Schwaber ve Jeff Sutherland tarafından geliştirilmiştir [12]. Yazılım geliştirme süreçleri boyunca değişmesi muhtemel çeşitli çevresel ve teknik faktörler (örneğin müşteri beklentileri, sistem gereksinimleri, zaman kısıtı, kaynaklar ve teknoloji) projeleri öngörülemez ve karmaşık bir hale getirir. Scrum, sürekli değişen bir ortamda esnek bir şekilde geliştirme yapabilmek için ekip üyelerinin nasıl çalışması gerektiğine odaklanır.

Scrum, yazılım geliştirme projelerinde meydana gelen sık teknolojik ve gereksinim değişikliklerini hızlı bir şekilde uyum sağlama yeteneği ile bilinir ve aynı zamanda projelerde hızlı sonuçlar elde etme olanağı sunar [13]. Esneklik, metodolojinin ana özelliğidir ve aynı zamanda projenin performansını kontrol etme ve iyileştirme mekanizmaları sunar. Metodolojinin yapısı, küçük geliştirme ekipleri için çok uygundur ve yüksek düzeyde erişilebilir harici müşterilere sahip projeler için kullanılması tercih edilir.

Scrum, Microsoft, Google, IBM, Amazon, Yahoo, Siemens, Havelsan, ve Türk Telekom gibi büyük yazılım sağlayıcıları tarafından yaygın olarak kullanılmaktadır, bu da metodolojinin avantajlarının artan önemini ve tanınmasını göstermektedir [14].

1.2.1. Scrum Aşamaları

Scrum yaklaşımı Schwaber tarafından üç ana aşamaya ayrılmıştır. Geliştirme öncesi (pre-game), geliştirme (development) ve geliştirme sonrası (post-game) [15]. Scrum proje geliştirme ve yönetme süreç yapısı Şekil 1'de gösterilmiştir.



Şekil 1. Scrum proje geliştirme ve yönetme süreçleri

Geliştirme öncesi aşaması kendi içerisinde 2 alt aşamaya ayrılır. Planlama (Planning) ve Mimari/Tasarım (Architecture/Design).

- *Planlama:* Planlama evresi geliştirilen sistemin tanımlandığı evredir. Müşteriden, satış ve pazarlama bölümünden ve yazılım geliştiricilerinden toplanan gereksinimler ile bir ürün gereksinimleri listesi (product backlog list) oluşturulur. Bu listedeki maddeler öncelik sırasına göre değerlendirilir ve harcanacak efor için tahmin yapılır. Ürün gereksinim listesi, sürekli olarak yeni öğeler ve öncelik sıralamaları ile güncellenir. Planlamada ayrıca proje ekibinin kullanacağı araçlar, kaynaklar, varsa eğitim ihtiyaçları belirlenir ve risk değerlendirmesi yapılır.

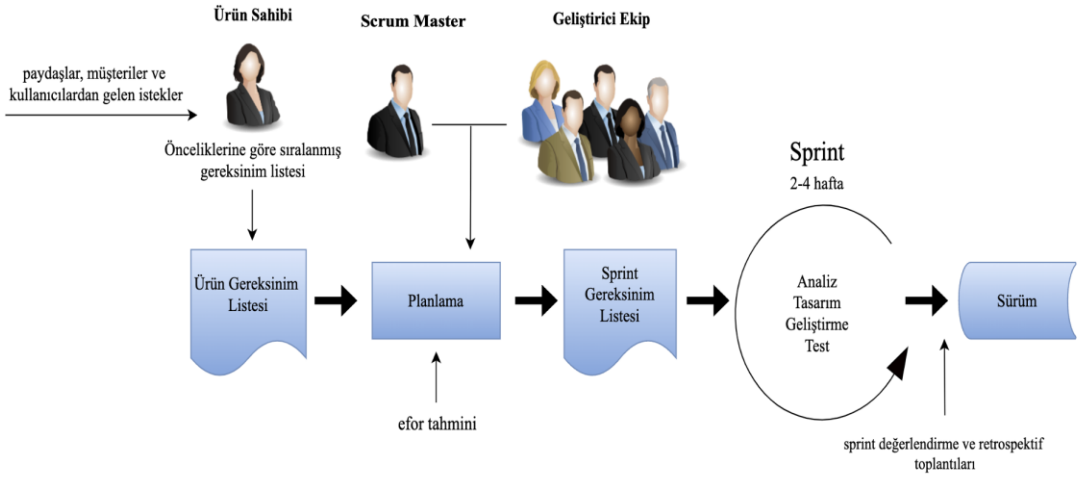
- *Mimari/Tasarım:* Mimari aşamada sistem mimarisi mevcut ürün gereksinimleri listesindeki maddelere dayanarak oluşturulur. Mevcut sistemi geliştirmeye yönelik tasarım evresinde ise yine gereksinimlere göre tasarım şekillendirilir, yapılacak geliştirmelerin sistemde neleri etkileyebileceği belirlenir. Bunun dışında bir tasarım analiz toplantısı gerçekleştirilip geliştirme için sunulan öneriler gözden geçirilerek tasarım konusunda nihai kararlar verilir.

Geliştirme aşaması, Scrum yaklaşımının çevik (Agile) kısmını temsil eder. Geliştirme aşamasında sistem, Sprintler içinde geliştirilir. Sprintler, yeni çıktılar üretmek için işlevsellik geliştirilen yineleyici döngülerdir [16]. Her Sprint, yazılım geliştirme sürecinin geleneksel aşamalarını içerir: gereksinimler, analiz, tasarım, evrim ve teslimat aşamaları. Sprint geliştirme sürecinde sistem mimarisi ve tasarımı evrim geçirir. Bir Sprint, bir haftadan bir aya kadar sürebilir. Bir sistem geliştirme sürecinde, sistem dağıtılmaya hazır hale gelmeden önce üç ila sekiz Sprint olabilir.

Geliştirme sonrası aşaması, bir sürümün kapanışını içerir. Bu aşamaya, gereksinimler gibi çevresel değişkenlerin tamamlandığına dair bir anlaşmaya varıldığında girilir. Bu durumda, artık yeni gereksinimler ve sorunlar bulunamaz. Sistem sürüm için hazır durumdadır ve bu, entegrasyon, sistem testi ve belgeleme gibi görevleri içeren oyun sonrası aşamasında yapılır [17].

1.2.2. Scrum'da Roller ve Sorumluluklar

Scrum, sorumluluk, işbirliği ve yinelemeli bir gelişim yolu ile karmaşık yazılım ürünlerini küçük birimlere (sprint) bölerek geliştirmeyi ve teslimi öngörür. Scrum'ı diğer çevik proje yönetim yaklaşımlarından ayıran nokta belirli rolleri, olayları ve bileşenleri kullanarak nasıl çalıştığıdır [18]. Scrum'da yer alan Scrum Master, Ürün Sahibi (Product Owner), Geliştirici Ekip (Development Team) gibi roller, işlevleri ve diğer rollerle ilişkileri Şekil 2'de gösterilmiştir.



Şekil 2. Scrum'da roller ve sprint içerisindeki sorumluluklar

Scrum master, Scrum tarafından tanıtılan yeni bir yönetim rolüdür. Scrum master, projenin Scrum'ın uygulamaları, değerleri ve kurallarına uygun bir şekilde yürütüldüğünden ve planlandığı gibi ilerlediğinden emin olma sorumluluğuna sahiptir [19]. Scrum master, projenin sürecinde proje ekibiyle birlikte müşteri ve yönetimle etkileşimde bulunur. Ayrıca, takımın mümkün olan en verimli şekilde çalışmasını sağlamak için süreçteki herhangi bir engelin kaldırılmasından ve değiştirilmesinden sorumludur.

Ürün sahibi, resmi olarak projeden sorumlu olan, ürün gereksinim (Product Backlog) listesini yöneten, kontrol eden ve görünür kılan kişidir [20]. Ürün sahibi; Scrum master, müşteri ve yönetim birimi tarafından seçilir. Ürün gereksinim listesine eklenecek görevlerle ilgili nihai kararları alır, listedeki ögeleri için geliştirme eforunu tahmin etmeye katılır ve listedeki sorunları geliştirilecek özelliklere dönüştürür.

Scrum takımı, her Sprint'in hedeflerine ulaşmak için gerekli eylemler konusunda karar alma yetkisine sahip olan proje ekibidir. Scrum takımı, örneğin efor tahmininde, önceliklendirilmiş gereksinim listesinin (sprint backlog) oluşturulmasında, ürün gereksinim listesinin gözden geçirilmesinde ve projeden kaldırılması gereken engelleri önerme konusunda dahil olur.

1.2.3. Scrum Pratikleri

Scrum, çevik yazılım geliştirme metodolojilerinden biri olarak, yazılım projelerinin daha verimli, esnek ve hızlı bir şekilde yönetilmesini sağlamak amacıyla

geliştirilmiştir. Scrum pratikleri, bu metodolojiyi uygularken takip edilen belirli süreçleri, toplantıları, rolleri ve araçları kapsar.

Ürün gereksinim listesi mevcut bilgiye dayalı olarak nihai üründe gereken her şeyi tanımlar. Bu nedenle, ürün gereksinim listesi, projede yapılması gereken işi tanımlar. İnşa edilen veya geliştirilen sistem için iş ve teknik gereksinimlerin öncelikli ve sürekli olarak güncellenen bir listesini içerir [21]. Gereksinim listesi öğeleri, örneğin özellikler, fonksiyonlar, hata düzeltmeleri, kusurlar, talep edilen geliştirmeler ve teknoloji yükseltmeleri içerebilir. Bu gereksinimler genellikle daha küçük ve yönetilebilir parçalara ayrılarak tanımlanır. Büyük kapsamlı işlere Epik (Epic) adı verilirken, bu epiklerin alt bileşenleri olarak daha küçük ve belirli kullanıcı ihtiyaçlarını tanımlayan Kullanıcı Senaryoları (User Stories) oluşturulur. Kullanıcı Senaryoları, bir kullanıcının bakış açısından belirli bir gereksinimi ifade eder. Epikler ise birden fazla kullanıcı senaryosunu kapsayan geniş kapsamlı gereksinimlerdir ve genellikle birden fazla sprint boyunca geliştirilmeye devam eder. Ürün gereksinim listesi öğelerinin oluşturulmasında müşteri, proje ekibi, yönetim, pazarlama ve satış gibi birden fazla aktör yer alabilir. Ürün sahibi, Ürün gereksinim listesinin sürdürülmesinden sorumludur.

Sprint gereksinim listesi, her Sprint için başlangıç noktasıdır. Bu, bir sonraki Sprint'te uygulanacak ürün gereksinim listesi öğelerinin bir listesidir. Öğeler, Sprint Planlama Toplantısı'nda Scrum Takımı, Scrum master ve ürün sahibi tarafından öncelikli öğelere ve Sprint için belirlenen hedeflere dayanarak seçilir. Ürün gereksinim listesinin aksine, Sprint gereksinim listesi, Sprint tamamlanana kadar sabittir. Sprint gereksinim listesindeki tüm öğeler tamamlandığında, sistemin yeni bir iterasyonu teslim edilir.

Sprint Planlama Toplantısı, Scrum master tarafından düzenlenen bir toplantıdır. Müşteriler, kullanıcılar, yönetim, ürün sahibi ve Scrum Takımı, bir sonraki Sprint'in hedefleri ve işlevselliği konusunda karar vermek için toplanır. Ürün gereksinim listesindeki hangi işlerin Sprint'e dahil edileceği ve bu işlerin detayları tartışılır. Sonuç olarak güncellenmiş bir Sprint gereksinim listesi elde edilir.

Tamamlanma Tanımı (Definition of Done – DoD), bir işin tamamlanmış sayılabilmesi için Scrum takım üyelerinin uyması gereken standartlar ve kriterlerden

oluşan kapsamlı bir kontrol listesi oluşturur. Tamamlanma tanımının oluşturulması, Scrum takımının tüm üyelerinin katkısını gerektiren kolektif bir süreçtir. Bu kontrol listesi; otomatik ve manuel testler, dokümantasyon, kullanıcı kabul testleri ve bazen entegrasyon ve güvenlik gereksinimlerini içerebilir [22].

Efor tahmini, belirli bir gereksinim öğesini geliştirmenin tamamlanması için gereken zaman, kaynak ve çabanın tahmin edilmesidir. Bu tahminler, bir ürünün planlanan zaman çerçevesi içindeki iş yükünün anlaşılmasına ve yönetilmesine yardımcı olmak amacıyla yapılır. Tahminler genellikle hikâye puanı (story point) gibi ölçekler kullanılarak yapılır ve işin karmaşıklığı, belirsizliği ve çaba gereksinimi göz önünde bulundurulur. Fibonacci dizisi (1, 2, 3, 5, 8, 13...) gibi ölçekler kullanılarak işlerin büyüklüğü tahmin edilir. Planlama Pokeri (Planning Poker) gibi tekniklerle ekip üyeleri bağımsız tahminler yapar ve ortak bir noktada anlaşır [23]. Doğru bir efor tahmini, sprint planlamasının verimli olmasını sağlar ve ekiplerin kapasitelerini daha iyi yönetmelerine yardımcı olur. Efor tahmini yapma sorumluluğu, ürün sahibinin ve Scrum Takımı'nın üzerindedir [24].

Sprint, belirli bir süre içinde bir yazılım ürününün geliştirildiği, test edildiği ve teslim edildiği, sürekli tekrarlanan bir geliştirme döngüsünü ifade eder. Sprint, genellikle 2 ila 4 hafta arasında bir süreyi kapsayan, sabit bir zaman dilimini temsil eder [16]. Scrum Takımı'nın Sprint içerisindeki çalışma araçları, Sprint Planlama Toplantıları, Sprint Gereksinim Listesi ve Günlük Scrum toplantılarıdır. Ekipler, sprint içerisindeki iş süreçlerini daha iyi takip etmek amacıyla çevrimiçi Görev Panosu (Task Board) araçlarını tercih edebilmektedirler. Görev panosu, ekip üyelerinin işlerin ilerleyişini görsel olarak takip etmesini sağlayan bir araç olup, genellikle "Yapılacaklar (To Do)", "Devam Edenler (In Progress)" ve "Tamamlandı (Done)" gibi sütunlardan oluşur.

Günlük Scrum toplantıları (Daily Stand-Up Meeting), Scrum takımının ilerlemesini sürekli olarak takip etmek ve aynı zamanda planlama toplantıları olarak hizmet etmek amacıyla düzenlenir. Son toplantıdan bu yana neler yapıldığı ve bir sonraki toplantıdan önce neler yapılacağı konuşulur. Ayrıca, bu kısa süreli (yaklaşık 15 dakika) günlük toplantıda varsa sorunlar tartışılır ve kontrol edilir. Sistem geliştirme sürecinde veya mühendislik uygulamalarında herhangi bir eksiklik veya engel araştırılır, tanımlanır ve süreci iyileştirmek için kaldırılır. Scrum master, Scrum

toplantılarını yönetir. Toplantı sadece geliştirici ekip tarafından yapılabileceği gibi, yönetim, ürün sahibi gibi iş birimlerinden de katılım sağlanabilir.

Sprint içerisinde geliştirmesi tamamlanan yazılımın kod kalitesini artırmak ve hataları erken tespit etmek amacıyla "Kod İncelemesi (Code Review)" yapılır. Kod incelemesi, ekip üyelerinin birbirlerinin yazdığı kodu gözden geçirerek olası hataları bulmasını, en iyi uygulamaların takip edilmesini ve kodun daha okunabilir hale gelmesini sağlar. Kodun doğruluğunu ve işlevselliğini test etmek için ise Test Planı ve Test Senaryoları. Test planı, hangi testlerin yapılacağını belirlerken; test senaryoları, belirli bir fonksiyonun nasıl test edileceğini tanımlar. Elde edilen test sonuçları (Test Plan, Test Case & Result) ise yazılımın kalitesini ölçmeye yardımcı olur. Test süreçlerinin ardından, kodda gereksiz karmaşıklık veya tekrar eden yapılar tespit edilirse "Kod İyileştirmesi (Code Refactoring)" yapılır. Kod iyileştirmesi, kodun mevcut davranışını değiştirmeden yapısını daha temiz ve anlaşılır hale getirme sürecidir. Böylece kodun bakımı kolaylaşır ve gelecekte yapılacak geliştirmelere daha uygun hale gelir. Tüm bu süreçlerin sonunda ise ürünün kullanıcıya sunulması için "Sürüm Planlaması (Release Planning)" gerçekleştirilir. Sürüm planlama, yeni özelliklerin ne zaman ve hangi sürümde yayınlanacağını belirleyerek yazılımın kontrollü bir şekilde dağıtılmasını sağlar. Bu süreçlerin etkili bir şekilde yürütülmesi, yazılımın daha güvenilir, ölçeklenebilir ve sürdürülebilir olmasını sağlar.

Sprint İnceleme (Review) toplantısı, Sprint'in son gününde yapılır. Scrum takımı ve Scrum master, Sprint'in sonuçlarını yönetim, müşteriler, kullanıcılar ve ürün sahibine bu toplantıda sunar. Katılımcılar, yeni ürün çıktılarını değerlendirir ve takip eden faaliyetler hakkında karar verir. İnceleme toplantısı, yeni gereksinim listesi öğelerini ortaya çıkarabilir ve hatta inşa edilen sistemin yönünü değiştirebilir.

"Sprint Burndown Grafikleri (Sprint Burndown Charts)", bir sprint içindeki işin ilerleyişini takip eden temel görsel araçlardır. Kalan iş miktarını zamana karşı grafiğe dökerek, bu grafikler bir ekibin sprint hedeflerini tamamlama yolunda olup olmadığını anında gösterir. İdeal burndown çizgisinden sapmalar, ekibin geride mi kaldığını yoksa beklenenden daha hızlı mı ilerlediğini hızla ortaya koyarak iş yükü veya kaynaklar konusunda zamanında ayarlamalar yapılmasını sağlar [25].

“Retrospektif (Retrospective)”, Scrum metodolojisinde genellikle Sprint sonunda gerçekleştirilen bir toplantıdır. Bu toplantıda, Scrum ekibi, tamamlanan işleri ve süreci değerlendirir, olumlu ve geliştirilebilecek yönleri belirler ve gelecekteki performanslarını artırmak için eylem planları oluşturur. Retrospektif toplantıları, ekibin birbirleriyle açıkça iletişim kurmaları, sprint içerisinde varsa yaşadıkları sorunları, karşılaştıkları engelleri dile getirmeleri için sağlıklı bir ortam sağlar. Retro çıktıları şu şekilde kategorize edilebilir:

1. Memnun olunan maddeler (Glad): Ekibin neleri doğru yaptığının ve sprint içinde memnun oldukları maddelerin yer aldığı kategoridir.
2. Üzgün olunan maddeler (Sad): Ekibin karşılaştığı zorluklar, hayal kırıklıkları ve olumsuz deneyimlerin yer aldığı kategoridir.
3. Memnun olunmayan olumsuz maddeler (Mad): Ekip üyelerinin birbirlerine veya dış faktörlere karşı hissettiği olumsuz duyguların yer aldığı kategoridir. Bu kategori, iletişim sorunları veya takımdaki potansiyel çatışma noktalarını belirlemek için kullanılır.

Retro çıktı maddeleri Scrum master tarafından sorunların çözümünün bulunması için toplanır. Ekip üyeleri toplantı içinde çözüm üretmeye çalışır. Gerekirse bu “Retrospektif çıktıları” iş birimine de iletilebilir.

“Sprint İş Listesini İyileştirme (Refinement)” toplantısı, bir Scrum ekibinin bir sonraki Sprint’e yönelik hazırlıklarını yapma sürecini ifade eder. Gelecek Sprint’te gerçekleştirilecek işlerin planlama öncesi belirlenmesi ve detaylandırılması için yapılan bir toplantıdır. Genellikle iş analistleri Scrum Takımı’na, planlama toplantısında konuşulacak işleri anlatır, kullanıcı senaryolarının üzerinden geçilir, gerekirse güncellemeler yapılır ve böylelikle ekibin sprint planlamasına hazır olması sağlanır [26]. Bu süreç, ekibin planlamalarını yeniden düşünmeleri ve zamanında teslimat yapabileceklerinden emin olmaları için bir fırsattır [23].

Scrum’ın yapısal doğası, disiplinler arası küçük takımlar ve belirgin roller ile, tüm proje paydaşlarıyla etkili iletişim ve işbirliğini teşvik eder. Bu da yazılım geliştirme sürecinde yüksek verimlilik ve kaliteyi sağlar. Günümüzde, Google, Microsoft, Yahoo, Amazon, Siemens, Havelsan ve Türk Telekom gibi birçok teknoloji şirketi başta olmak üzere, otomotiv, finans, sağlık ve telekomünikasyon gibi sektörlerde de Scrum metodolojisi benimsenmiştir. Türkiye'de Scrum kullanımı,

özellikle finans ve bankacılık sektörlerinde daha fazla yayılmaya başlamıştır. Ancak, Scrum metodolojisinin organizasyonel değişim gerektirmesi ve kültürel engellerin aşılması gibi zorluklar, bazı firmalar için uygulanabilirliği sınırlayabilmektedir.

1.3. Uç Programlama

Uç Programlama (Extreme Programming – XP), 1999 yılında Kent Beck tarafından ortaya atılan bir çevik yazılım geliştirme metodolojisidir. Geleneksel yazılım geliştirme modellerinin uzun geliştirme döngülerinin neden olduğu sorunlara çözüm olarak ortaya çıkmıştır. Hedefi, belirsiz veya hızla değişen gereksinimlere uyum sağlayarak yazılım geliştiren küçük ve orta ölçekli takımlar oluşturmaktır.

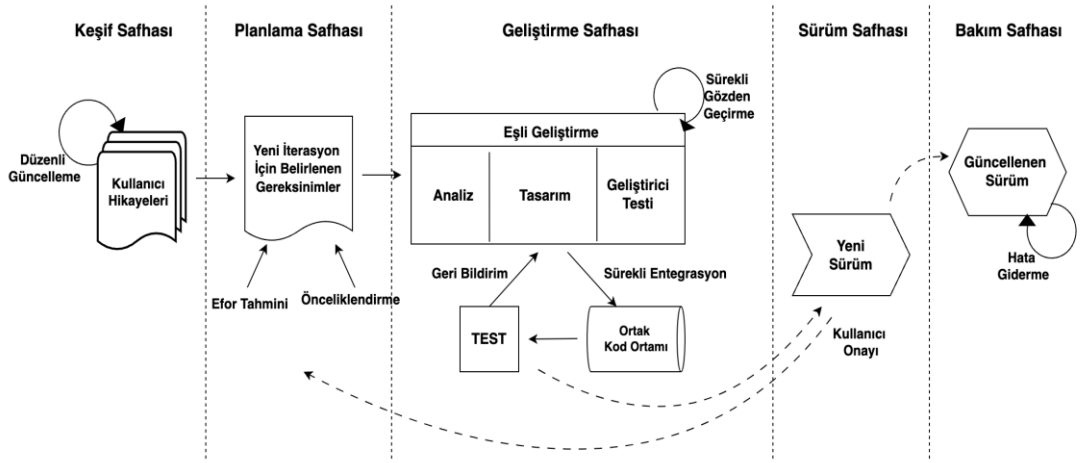
1.3.1. Uç Programlama Değer ve İlkeleri

XP, belirli değerler etrafında şekillenen bir yaklaşımdır ve bu değerler yazılım geliştirme sürecinde takım üyelerinin işbirliğini, verimliliğini ve başarılarını artırmalarına yardımcı olur. XP metodolojisi beş temel değere dayanır:

1. *İletişim:* Yazılım geliştirme, müşterinin ihtiyaçlarını anlamak ve bunları uygulamakla ilgilidir. Takım üyelerinin birbirleriyle iletişim kurması çok önemlidir [27].
2. *Sadelik:* Öncelikle plan yaparak işleri mümkün olduğunca basit hale getirmeye çalışılır. Bu, gereksiz şeylerden kaçınmaya ve geliştiricilerin konsantre olmasına yardımcı olur. Aynı zamanda, mevcut gereksinimler üzerinde çalışmayı teşvik eder. Sistem tasarımı da bakım ve geliştirme kolaylığı için sade tutulmalıdır.
3. *Geri Bildirim:* Önceki çalışmalardan sürekli geri bildirim almak, üründe iyileştirilmesi gereken alanları belirlemeye yardımcı olabilir ve tasarımın daha basit hale gelmesini sağlar.
4. *Cesaret:* Başarısızlığın getirdiği korku ile cesur bir şekilde yüzleşmeli ve böyle durumlarda, önceki ilkeler akılda tutularak sonuçların takıma zarar vermemesi sağlanmalıdır.
5. *Saygı:* Bir projedeki tüm üyeler, yani müşteri ve geliştirici, birbirlerine saygı göstermeli, eleştiri ve geri bildirimi kabul etmelidir. Çünkü bu, projenin başarılı olmasına yardımcı olur [9].

1.3.2. Uç Programlama Aşamaları

XP geliştirme süreci keşif, planlama, geliştirme, sürüm ve bakım aşamalarından oluşur ve bu aşamalar sürekli tekrar eder. Keşif safhasında müşteri gereksinimleri kullanıcı hikayeleriyle belirlenir ve düzenli olarak bu gereksinimler güncellenir. Planlama safhasında koşulacak iterasyon için ilgili gereksinimler öncelik sırasına göre belirlenir ve ekibin ne kadar efor harcayacağının tahmini yapılır. Geliştirme safhasında sadelik esas alınarak tasarım yapılır, çift programlama ve sürekli entegrasyon gibi uygulamalarla kod geliştirilir, düzenli testler yapılır ve müşteri ile sürekli iletişim sağlanır [28]. Geliştirmeler tamamlandıktan sonra Sürüm safhasına geçilir ve kullanıcı onayı alınarak yeni sürüm çıkılır. Son olarak Bakım safhasında sürüm çıkıldıktan sonra gerekli görülen küçük çaplı eklemeler ve gözlemlenen sistem hatalarının giderilmesi için çalışılır. XP proje geliştirme ve yönetme süreç yapısı Şekil 3'te gösterilmiştir.



Şekil 3. XP proje geliştirme ve yönetme süreç yapısı

Böyle bir geliştirme süreci, her bir katılımcının kendi görev ve sorumluluklarına sahip olduğu birçok kişinin iş birliğini gerektirir. XP insanları sistemin merkezine koyar ve iletişim, iş birliği, hızlı tepki verme ve geri bildirim gibi sosyal becerilerin değerini ve önemini vurgular. Bu nedenle, XP genellikle şu rolleri içerir:

- Müşteriler, kullanıcı hikayeleri oluşturarak, sürekli geri bildirim sağlayarak ve projeyle ilgili tüm gerekli iş kararlarını vererek geliştirme sürecine yoğun şekilde katılmaları beklenir [29].

- Programcılar veya geliştiriciler, ürünü fiilen oluşturan ekip üyeleridir. Kullanıcı hikayelerini uygulamaktan ve kullanıcı testlerini yapmaktan sorumludurlar (bazen ayrı bir Testçi rolü de belirlenebilir). XP genellikle çapraz fonksiyonlu ekiplerle ilişkilendirildiğinden, bu üyelerin beceri seti farklı olabilir [30].
- Yöneticiler, müşteriler ve geliştiriciler arasında köprü kurarlar. Bu zorunlu bir rol değildir ve geliştiricilerden biri tarafından gerçekleştirilebilir. Bu kişiler toplantıları organize eder, tartışmaları düzenler ve önemli ilerleme göstergelerini (KPI'lar) takip ederler.
- Koçlar, XP uygulamalarını anlamaya yardımcı olmak için takımlara mentor olarak dahil edilebilir. Genellikle dışarıdan gelen bir yardımcı veya danışmandır ve geliştirme sürecine dahil değildir, ancak daha önce XP'yi kullandığından hatalardan kaçınmaya yardımcı olabilir [31].

1.3.3. Uç Programlama Pratikleri

XP, yazılım projelerinde kolaylığı ve esnekliği sağlamak üzere on iki farklı pratiği öngörmektedir [32]:

- Planlama Oyunu (Planning Game): İş öncelikleri ve teknik tahminleri birleştirerek bir sonraki sürümün kapsamını hızla belirleyin. Müşteri, iş açısından kapsam, öncelik ve tarihlere karar verirken, teknik kişiler ilerlemeyi tahmin eder ve izler.
- Küçük Sürümler (Short Releases): Basit bir sistemi hızla üretime alın. Yeni sürümleri çok kısa bir (iki haftalık) döngüde yayınlayın.
- Metafor (Metaphor): Tüm geliştirmeyi, genel sistemin nasıl çalıştığını anlatan basit, ortak bir hikâye ile yönlendirin.
- Basit Tasarım (Simple Design): Herhangi bir anda mümkün olduğunca basit tasarım yapın.
- Önce Test (Test-First): Geliştiriciler, kusursuz çalışması gereken birim testleri sürekli yazar; müşteriler, fonksiyonların tamamlandığını göstermek için testler yazar. "Önce test, sonra kod" prensibi, başarısız bir test vakasının kod yazmak için bir giriş kriteri olduğu anlamına gelir.
- Yeniden Yapılandırma (Refactoring): Sistemin davranışını değiştirmeden yeniden yapılandırarak, tekrarlamaları kaldırmak, iletişimi iyileştirmek, basitleştirmek veya esneklik eklemek.

- Çift Programlama (Pair Programming): Tüm üretim kodu, bir makinede iki programcı tarafından yazılır.
- Ortak Sorumluluk (Collective Ownership): Herkes, istediği zaman, sistemin herhangi bir yerindeki kodu iyileştirebilir.
- Sürekli Entegrasyon (Continuous Integration): Sistemi günde birçok kez entegre edin ve derleyin (her bir görev tamamlandığında). Sürekli yapılan regresyon testleri, gereksinimler değiştiğinde işlevsellikteki bozulmaları önler.
- Haftada 40 Saat Çalışma (40-Hour Week): Mümkünse haftada 40 saatten fazla çalışmayın; asla iki hafta üst üste fazla mesai yapmayın.
- Ekipte Müşteri (On-site Customer): Soruları yanıtlamak için ekipte tam zamanlı bir kullanıcı bulundurun.
- Kodlama Standartları (Coding Standards): Kod genelinde iletişimi vurgulayan kurallara sahip olun [33].

XP, sürekli test ve yeniden yapılandırma uygulamaları ile hataları en aza indirgemeyi ve istikrarlı sistemler oluşturmayı amaçlar. Sadelik ilkesi, anlaşılır ve kolayca değiştirilebilir kod yazmayı teşvik ederken, iteratif geliştirme hızlı sonuçlar elde edilmesini sağlar [25]. Müşteri katılımı ve sürekli iletişim, projenin şeffaf ve hesap verebilir olmasına katkıda bulunur. Ancak XP'nin bazı dezavantajları da mevcuttur. Müşterinin nihai sonuç hakkında net bir fikre sahip olmaması, proje kapsamı, maliyet ve zaman tahminlerini zorlaştırabilir. Düzenli müşteri toplantıları geliştirme sürecinden zaman alabilir ve yetersiz dokümantasyon, gereksinimlerin açıkça tanımlanmasını engelleyebilir. Ayrıca, çiftli programlama her zaman verimli olmayabilir ve XP'nin yüz yüze görüşme gerektirmesi, uzaktan çalışan ekiplerle uygulanmasını güçleştirebilir. XP, deneyimsiz geliştiricilerle de etkili sonuçlar vermeyebilir [33].

1.4. Lean

Lean, israfı en aza indirerek süreç verimliliğini artırmayı amaçlayan bir yönetim felsefesidir. İlk olarak Toyota üretim sisteminde geliştirilmiş ve otomotiv endüstrisinde başarıyla uygulanmıştır [34]. Daha sonra farklı sektörlere yayılmış ve yazılım geliştirme gibi alanlarda da kullanılmaya başlanmıştır. Lean'in temel amacı, müşteri için değer yaratmayan tüm faaliyetleri ortadan kaldırmak ve süreçleri daha verimli hale getirmektir [35].

1.4.1. Lean Değer ve İlkeleri

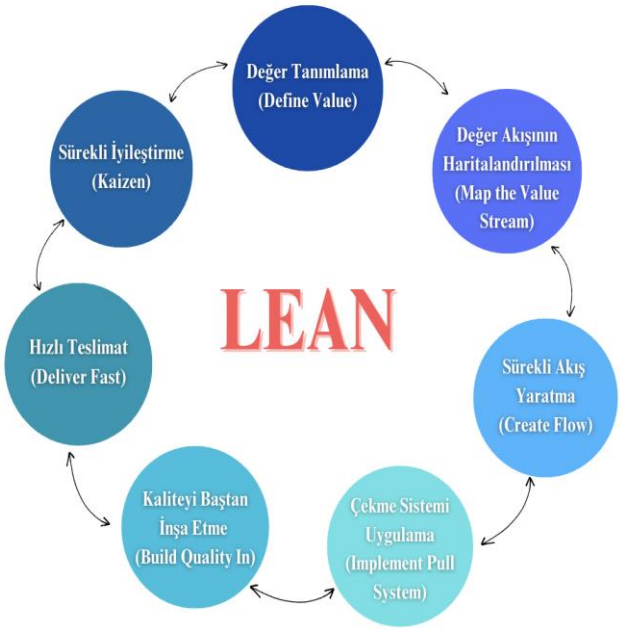
Lean yazılım geliştirme prensipleri, verimliliği artırmak ve israfı azaltmak amacıyla belirlenmiştir [36]. Bu prensipler şu şekilde özetlenebilir [37]:

1. İsrafı Ortadan Kaldır (Eliminate Waste): Müşteri için değer yaratmayan her türlü faaliyeti israf olarak görür ve bunları süreçten çıkarır. Lean felsefesi, yazılım geliştirme alanında yedi temel israf türüne odaklanır: tamamlanmamış iş, gereksiz özellikler, görev değişimi, ekstra süreçler, bekleme, kusurlar ve gereksiz hareketler [38].
2. Kaliteyi Baştan İnşa Et (Build quality in): Kaliteyi baştan itibaren sürecin bir parçası haline getirerek, sonradan hataları düzeltmek yerine, geliştirme sürecinin her aşamasında kaliteyi sağlamayı amaçlar [34].
3. Bilgi Oluştur (Create Knowledge): Sürekli öğrenme ve bilgi paylaşımı üzerine kurulu bir yapı oluşturmayı teşvik eder. Bu, ekip üyelerinin deneyimlerinden öğrenmeyi ve yeni bilgilerle geliştirme sürecini iyileştirmeyi içerir [39].
4. Taahhütleri Ertele (Decide as Late as Possible): Geliştirme sürecinde kritik kararları mümkün olduğunca geç vererek, daha fazla bilgi ve veri toplandıktan sonra doğru kararı verme imkânı sağlar.
5. Hızlı Teslim Et (Deliver Fast): Müşteri taleplerine hızlı yanıt verebilmek için küçük ve sık teslimatlar yapılır.
6. İnsanlara Saygı Göster (Respect people): Ekip üyelerinin, müşterilerin ve tüm paydaşların fikirlerine ve katkılarına değer vermek, motivasyonu ve işbirliğini artırır [39].
7. Bütünü Optimize Et (Optimize the Whole): Sistem ve süreçlerin sadece bir parçasını değil, tamamını optimize etmeyi hedefler [34].

Lean yaklaşımı 3Ms (muda, mura ve muri) olarak da bilinen üç tür israf tanımlar:

- Muda, üretim sistemine değer katmayan ve/veya israfa yol açan belirli görevler ve aktivitelerin ortadan kaldırılmasıdır [40].
- Mura, üretimde miktar veya kalite açısından ortaya çıkan değişkenlik veya düzensizliklerin yarattığı ek yükün ortadan kaldırılmasıdır [40].
- Muri, ise üretim sisteminin, makineler veya işçiler gibi, kapasite veya yeteneklerinin ötesinde aşırı yüklenmesi durumudur. İdeal çalışma kapasitesi %60-70 oranında olmalıdır. Bundan daha fazlası durumunda yürütmeye

çalışılan proje sayısının azaltılması gerekir [18]. Şekil 4'te Lean yazılım geliştirme süreçleri görülmektedir.



Şekil 4. Lean yazılım geliştirme süreçleri

Şekil 4'te de görüldüğü üzere Lean yazılım geliştirme süreci, değer tanımlama (Define Value) ile başlar. Müşteri için değerli olan faaliyetler belirlenir ve değer akışını haritalama (Map the Value Stream) adımıyla israf sayılan aktiviteler tespit edilip ortadan kaldırılır. Kesintisiz akış (Create Flow), analiz, tasarım, kodlama ve test gibi süreçlerin sorunsuz ilerlemesini sağlar. Çekme sistemi (Implement Pull System) ise yalnızca ihtiyaç duyulduğunda iş başlatmayı ifade eder, bu da gereksiz iş yükünü önler. Kaliteyi baştan sağlama (Build Quality In) ile hatalar en baştan önlenir; sürekli entegrasyon, otomatik testler ve kod gözden geçirme bu süreçte kritik rol oynar. Hızlı teslimat (Deliver Fast) ile müşteriye küçük ve sık teslimatlar yapılır, böylece hızlı geri bildirim alınır. Son olarak, sürekli iyileştirme (Continuous Improvement) her iterasyonda süreçleri daha verimli hale getirmeyi amaçlar. Bu adımlar, sadece yazılımı değil, ekip içi işbirliği ve süreç verimliliğini de iyileştirmeyi hedefler.

Lean felsefesi, yazılım geliştirme ekiplerine daha etkin bir süreç yönetimi sağlar ve müşteri memnuniyetini artırmak için esnek, verimli ve sürdürülebilir bir yazılım geliştirme ortamı yaratır.

Lean ile Agile yöntemleri, birbirini tamamlayan yaklaşımlar olarak görülür. Her iki metodoloji de müşteriye hızlı ve sürekli değer sağlamayı amaçlar. Lean, daha çok israfın ortadan kaldırılması ve süreçlerin iyileştirilmesine odaklanırken, Agile esneklik ve hız üzerine yoğunlaşır. Yazılım geliştirmede Kanban gibi araçlar Lean yaklaşımına dayanırken, Scrum gibi yöntemler Agile prensiplerini kullanır [37].

1.4.2. Lean'de Roller ve Sorumluluklar

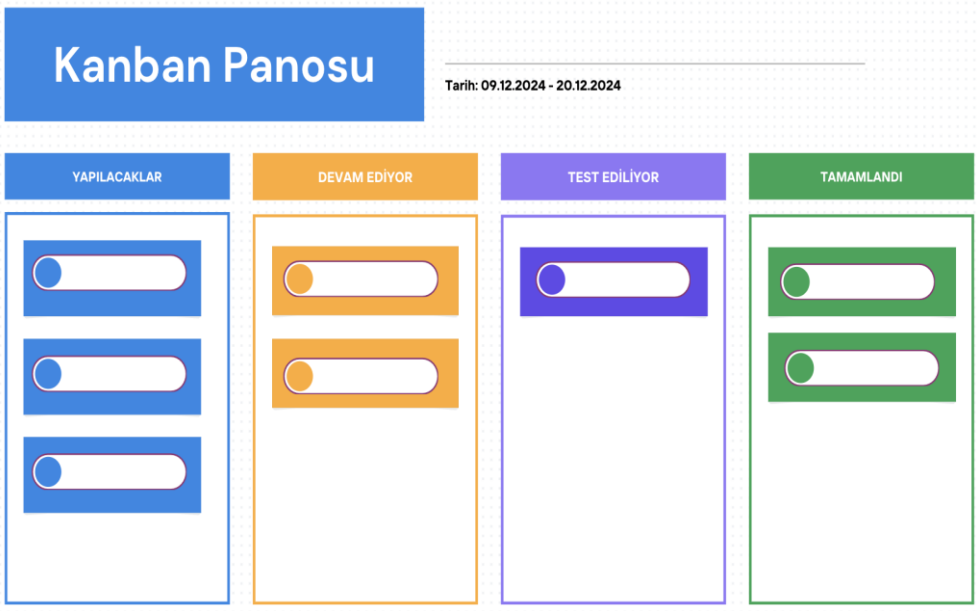
Lean geliştirme sürecinde ürün sahibi veya müşteri rolleri bulunmaz. Lean geliştirme ekiplerine genellikle Toyota'da "baş mühendis", Microsoft'ta "program yöneticisi" ya da 3M'de "ürün şampiyonu" gibi rollere sahip biri liderlik eder. Bu roller, bir girişimi yöneten bir girişimciye benzer şekilde hareket eder. Her ne kadar bu roller, çevik yazılım geliştirmedeki müşteri veya ürün sahibi rollerine benziyor gibi görünse de uygulamada oldukça farklıdır. Bir baş mühendis, bir ürünün genel sorumluluğunu üstlenir, ürünün pazardaki başarısını da kapsar ve çok disiplinli bir ürün ekibindeki herkesi bu başarıya ulaşmak için harekete geçirir. Ayrı bir yazılım geliştirme ekibi için işleri önceliklendiren bir ara rol bulunmaz [34].

1.5. Kanban

Lean ve Kanban yaklaşımları, 1950'lerde Japon üretim sanayisinde tanıtılmıştır. Kanban kullanımının arkasındaki temel fikir, Lean düşüncesini pratiğe dökmektir; ancak Lean, Kanban'dan daha geniş bir kavramdır [41]. Yazılım geliştirmede Kanban yöntemi, ekiplerin iş akışını görselleştirmesini, her iş akışı aşamasındaki işleri sınırlamasını ve bir işi tamamlayana kadar geçirdikleri süreyi ölçmesini sağlar [42].

Şekil 5'te örnek bir Kanban panosu görülmektedir. Kanban panosu, ekip çalışmasını görselleştirmek ve organize etmek için kullanılan temel araçtır. Pano sütunları, belirli aktiviteleri temsil eder ve üzerinde çalışılan özelliklerin kartları bu sütunlara yerleştirilir. Her aktivite için devam eden işlerin (Work In Progress – WIP) sınırı belirlenir. Aktiviteler genellikle "devam ediyor" ve "tamamlandı" şeklinde iki sütunla gösterilir. Ayrıca, bir aktiviteye başlamadan önce bekleyen işlerin tutulduğu başka sütunlar da olabilir [43]. Günümüzde ekipler Kanban panosunu çevrimiçi olarak

yönetmek ve işleri takip edebilmek için çoğunlukla Trello, Jira, Asana, Monday.com gibi dijital araçlardan yararlanmaktadır.



Şekil 5. Kanban panosu

1.5.1. Kanban Değer ve İlkeleri

Kanban prensipleri, yazılım geliştirme süreçlerinde iş akışının verimli bir şekilde yönetilmesine ve optimize edilmesine olanak sağlayan beş temel kuraldan oluşmaktadır [41]:

1. İş Akışını Görselleştirme (Visualize the Workflow): Kanban süreci, iş akışının her aşamasını görselleştirerek, ekibin iş süreçlerini daha şeffaf bir şekilde takip etmesine olanak tanır.
2. Devam Eden İşleri Sınırlama (Limit Work In Progress – WIP): Kanban, eş zamanlı olarak yürütülen işlerin sınırlandırılmasını önerir. Bu sınırlama, ekibin kapasitesine uygun bir iş yükü belirleyerek, aşırı yüklenmeyi önler ve ekip üyelerinin mevcut işlere odaklanarak daha verimli çalışmasını sağlar.
3. Akışı Ölçme ve Yönetme (Measure and Manage Flow): Kanban metodolojisi, işlerin süreç içerisindeki akış hızını ölçmeyi ve yönetmeyi hedefler. Akışın sürekli izlenmesi ve iyileştirilmesi, işlerin zamanında ve planlanan şekilde tamamlanmasına olanak sağlar.
4. Süreç Politikalarını Açık Hale Getirme (Make Process Policies Explicit): Kanban, süreçte izlenen kuralların ve politikaların açık bir şekilde

belirtilmesini gerektirir. Tüm ekip üyeleri, sürecin nasıl işlediğini ve hangi adımların takip edileceğini net bir şekilde bilmelidir.

5. İş Birliği ile Sürekli İyileştirme (Improve Collaboratively): Kanban, ekiplerin süreçleri iyileştirmek amacıyla birlikte çalışmasını teşvik eder. Süreçlerin düzenli olarak gözden geçirilmesi ve iyileştirilmesi, yazılım geliştirme sürecinde verimliliği ve kaliteyi artırır.

Kanban ve Agile yöntemlerinin entegrasyonu genellikle Scrum ile Kanban'ın bir arada kullanılmasıyla gerçekleşir ve bu birleşim Scrumban olarak adlandırılır [44]. Scrumban, Scrum'ın yinelemeli yapısını, Kanban'ın esnek ve sürekli akış sağlayan özellikleriyle harmanlamaktadır. Bu hibrit yaklaşımda, Kanban panoları, Scrum süreçlerinde belirlenen görevlerin aşamalarını görselleştirmek amacıyla kullanılır. Panolar, ekip üyelerinin mevcut iş yükünü ve ilerlemeyi şeffaf bir şekilde takip etmelerine imkân tanır. Scrum'dan farklı olarak, Scrumban'da belirli zaman dilimlerine (sprintler) dayalı çalışma yerine sürekli bir iş akışı benimsenir. Böylelikle ekipler, değişen ihtiyaçlara göre daha küçük ve hızlı teslimatlar gerçekleştirebilir. Duruma ve gereksinimlere göre sprintler sürekli teslim akışıyla değiştirilirken, Scrum'a özgü olan günlük toplantılar (stand-ups) ve periyodik değerlendirmeler (retrospective) gibi temel unsurlar korunur.

Kanban'ın sunduğu görselleştirme ve sınırlı iş yükü prensipleri, finans, sağlık, üretim ve hizmet sektörlerinde de süreç optimizasyonu ve kaynak yönetimi için kullanılmaktadır. Yazılım dışındaki bu alanlarda, projelerin maliyet ve zaman yönetiminde önemli avantajlar sağladığı gözlemlenmiştir. Kanban esnek yapısı, sektörel ihtiyaçlara göre uyarlanabilmesi ve verimliliği artıran yapısıyla günümüzün önemli süreç yönetim araçlarından biri haline gelmiştir.

2. YETENEK OLGUNLUK MODELLERİ

Yazılım geliştirme süreçlerinde kullanılan yetenek değerlendirme modelleri, organizasyonların süreçlerini olgunlaştırmasına ve performansını iyileştirmesine yardımcı olmaktadır. Bu modeller, organizasyonların mevcut yetkinliklerini değerlendirmelerine ve belirlenen hedeflere ulaşmaları için gerekli adımları atmalarına rehberlik etmektedir [45]. Süreçlerin tekrarlanabilir, ölçülebilir ve optimize edilebilir hale getirilmesi, bu modellerin temel amaçlarından biridir. Bu kapsamda Yetenek Olgunluk Modeli (CMM – Capability Maturity Model), Amerika Savunma Bakanlığı'nın 1970'lerde ortaya çıkan yazılım krizinde çözüm bulması amacıyla Carnegie Mellon Üniversitesi'nden yardım istemesi üzerine, üniversite bünyesinde kurulan SEI (Software Engineering Institute) tarafından geliştirilmiştir [46].

2.1. CMM

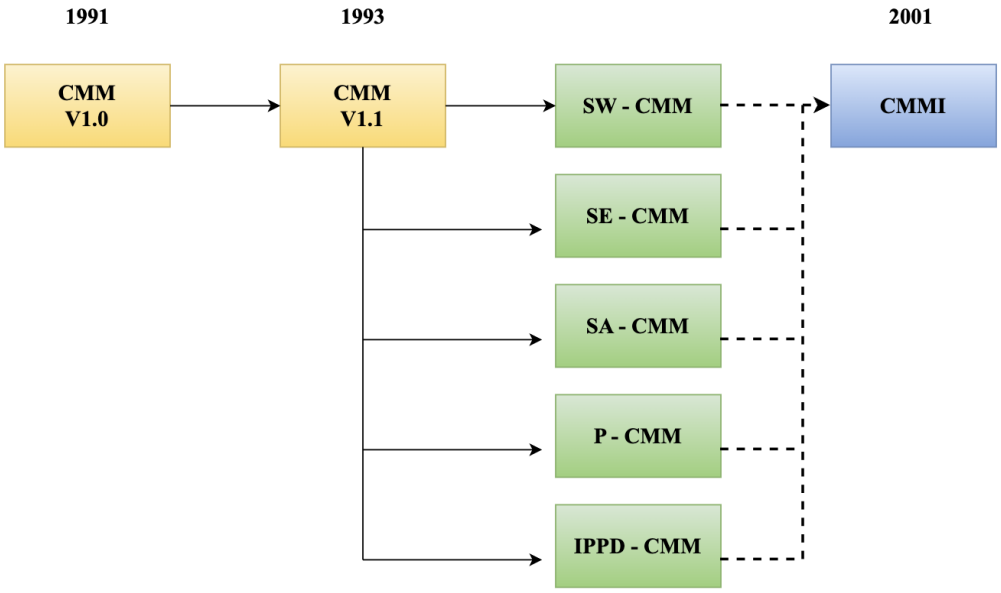
CMM, yazılım süreç olgunluğu ilkeleri ve uygulamalarıyla, yazılım organizasyonlarının süreçlerinin olgunluk seviyelerini iyileştirmelerine yönelik bir rehber sunmayı amaçlamaktadır [47]. Bu model, aynı zamanda organizasyonların stratejik gelişimlerini adım adım planlamalarına olanak tanıyan önemli bir çerçeve sağlar. CMM, organizasyonların insan kaynakları, süreç ve teknolojileri daha verimli bir şekilde yönetmelerine yardımcı olarak, uzun vadede genel performanslarını iyileştirmelerine katkıda bulunur. Zaman içinde, bu modelin kapsamı genişletilmiş ve yazılım, tedarik, sistem mühendisliği ve insan kaynakları gibi farklı alanlara odaklanan çeşitli CMM modelleri eklenmiştir. Tablo 1 farklı CMM türleri ve tanımlarını içermektedir.

Tablo 1. CMM çeşitleri ve tanımları

CMM Modeli	Tanım
Yazılım Yetenek Olgunluk Modeli ***(CMM for Software – SW-CMM)***	Bu model hem yazılım süreç olgunluğunu değerlendirmek için bir referans modeldir hem de yazılım organizasyonlarının geçici, kaotik süreçlerden olgun, disiplinli yazılım süreçlerine doğru evrimsel bir yol izlemelerine yardımcı olur.

Sistem Mühendisliği Yetenek Olgunluk Modeli ***(CMM for System Engineering – SE-CMM)***	Sistem mühendisliği süreçlerini değerlendirmek ve iyileştirmek için oluşturulmuş bir modeldir [48].
Yazılım Tedarik Yetenek Olgunluk Modeli ***(CMM for Software Acquisition – SA-CMM)***	Yazılım edinme süreçlerini iyileştirmek için bir çerçeve sağlar ve üst yönetimin hedef belirlemesini ve potansiyel performansın öngörülmesini destekler [49].
İnsan Yetenek Olgunluk Modeli ***(CMM for People – P-CMM)***	Bir organizasyonun insan kaynakları yönetimi süreçlerini olgunlaştırmak, çalışanların yetkinliklerini geliştirmek ve genel performanslarını iyileştirmek amacıyla tasarlanmış bir modeldir [50].
Entegre Ürün Geliştirme Yetenek Olgunluk Modeli ***(CMM for Integrated Product and Process Development – IPPD-CMM)***	Ürün yaşam döngüsü boyunca ilgili paydaşların zamanında işbirliğini sağlayarak müşteri ihtiyaçlarını daha iyi karşılamak için sistematik bir ürün geliştirme yaklaşımı sunar [51].

Şekil 6'da CMM'nin gelişiminin kronolojik olarak sıralanışını görülmektedir.



Şekil 6. CMM'nin gelişimi

CMM'nin temel amacı, organizasyonların yazılım süreçlerini sistematik bir şekilde olgunlaştırarak daha verimli ve etkili hale getirmektir. Bu hedef doğrultusunda, organizasyonların süreçlerini belirli aşamalarla değerlendirmelerine olanak tanır ve bu aşamalar, yazılım süreçlerinin her seviyede daha yapılandırılmış ve kontrol edilebilir bir hale gelmesini sağlar. CMM'nin Başlangıç, Yönetilen, Tanımlı, Nicel Yönetilen

ve En İyileştiren olmak üzere beş farklı olgunluk seviyesi, her bir seviyede organizasyonların süreçlerini daha iyi yönetmeleri, izlemeleri ve iyileştirmeleri için gerekli olan adımları belirler [52]. Tablo 2'de CMM olgunluk düzeyleri ve tanımları görülmektedir. Her seviye, bir sonraki olgunluk seviyesine ulaşmak için gerekli olan süreç iyileştirme faaliyetlerine rehberlik eder ve böylece organizasyonlar, yazılım geliştirme süreçlerinde sürekli bir gelişim sağlar. Bu bağlamda, her olgunluk seviyesinin kendine özgü anahtar süreç alanları ile tanımlanması, organizasyonların süreç yönetimini daha sistematik ve hedefe yönelik bir şekilde gerçekleştirmelerine olanak tanır.

Tablo 2. CMM olgunluk düzeyleri ve tanımları

CMM Olgunluk Düzeyleri	Tanım
1. Başlangıç Düzeyi (Initial Level)	Süreçlerin kaotik, genellikle bireysel becerilere ve çabanın odaklandığı belirgin bir süreç yönetimi yoktur.
2. Yönetilen Düzey (Managed Level)	Temel süreç yönetimi uygulamalarına geçişin olduğu seviyedir. Süreçler belirli projelerde yönetilir ve proje yönetimi prensipleri uygulanır.
3. Tanımlı Düzey (Defined Level)	Organizasyon genelinde standart bir süreç yönetimi stratejisi benimsenir. Süreçler standartlaştırılır ve organizasyon genelinde uygulanır.
4. Nicel Yönetilen Düzey (Quantitively Managed Level)	Süreç performansı nicel verilere dayalı olarak yönetilir ve kontrol edilir. Süreç iyileştirmesi için ölçümler ve analitik teknikler kullanılır.
5. En İyileştirilen Düzey (Optimizing Level)	Sürekli iyileştirme hedeflenir. Organizasyon, süreçlerini sürekli olarak izler ve iyileştirme fırsatlarını arar.

CMM, bir organizasyonun beş seviyeyi aşamalı olarak geçiş yaptığı bir olgunluk veya büyüme modeli olarak görülmektedir ve beşinci seviyeye ulaştıktan sonra bile sürekli olarak gelişme ve olgunlaşma sürecindedir [53]. Birinci düzey hariç, her olgunluk seviyesi, bir organizasyonun yazılım sürecini geliştirmek için odaklanması gereken alanları gösteren birkaç anahtar süreç alanına ayrılmıştır. Anahtar süreç alanları, bir olgunluk seviyesine ulaşmak için ele alınması gereken konuları belirler. Her anahtar süreç alanı, birlikte gerçekleştirildiğinde süreç yeteneklerini artırmak için önemli kabul edilen bir dizi hedefe ulaşan ilişkili aktivitelerin bir kümesini tanımlar. Tablo 3'te görüldüğü üzere CMM, ikinci olgunluk düzeyinden itibaren toplam on sekiz anahtar süreç alanına sahiptir [52].

Tablo 3. CMM olgunluk düzeyleri ve anahtar süreç alanları

Olgunluk Düzeyi	Odak Noktası	Anahtar Süreç Alanları
5 En İyileştirilen	Sürekli süreç iyileştirmesi hedefler	Hata Önleme (Defect Prevention - DP)
		Teknoloji Değişim Yönetimi (Technology Change Management – TM)
		Süreç Değişim Yönetimi (Process Change Management – PC)
4 Nicel Yönetilen	Ürün ve süreç kalitesine odaklanır	Nicel Süreç Yönetimi (Quantitative Process Management – QP)
		Yazılım Kalite Yönetimi (Software Quality Management – QM)
3 Tanımlı	Organizasyonun yazılım mühendisliği ve yönetim süreçlerini ele alır	Organizasyon Süreç Odaklılığı (Organization Process Focus – PF)
		Organizasyon Süreç Tanımı (Organization Process Definition – PD)
		Eğitim Programı (Training Program – TP)
		Entegre Yazılım Yönetimi (Integrated Software Management – IM)
		Yazılım Ürün Mühendisliği (Software Product Engineering – PE)
		Ekipler Arası Koordinasyon (Intergroup Coordination – IC)
		Eşdeğer Gözden Geçirme (Peer Reviews – PR)
2 Yönetilen	Proje yönetim süreçlerini ele alır	Gereksinim Yönetimi (Requirements Management – RM)
		Yazılım Proje Planlaması (Software Project Planning – PP)
		Yazılım Projesi Takibi (Software Project Tracking – PT)
		Yazılım Alt Yüklenici Yönetimi (Software Subcontract Management – SM)
		Yazılım Kalite Güvencesi (Software Quality Assurance – QA)
		Yazılım Konfigürasyon Yönetimi (Software Configuration Management – CM)

CMM'nin başarıları ve farklı alanlara uyarlanabilirliği, süreç iyileştirme modellerinin daha kapsamlı ve entegre hale getirilmesi ihtiyacını doğurmuştur. Bu bağlamda, çeşitli CMM modellerinin sağladığı deneyim ve çıktılar, daha geniş çapta bir uygulama alanına sahip olan Yetenek Olgunluk Modeli Entegrasyonu (Capability

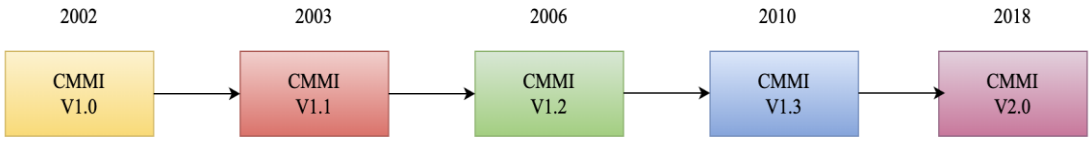
Maturity Model Integration – CMMI) modelinin ortaya çıkmasına zemin hazırlamıştır. CMMI, CMM'nin temel ilkelerini alıp daha geniş bir yelpazede süreç iyileştirme ve yönetim metodolojileri sunarak, organizasyonların yalnızca yazılım değil, sistem mühendisliği, ürün geliştirme, tedarik ve insan kaynakları gibi alanlarda da olgunlaşmasına yardımcı olur. Böylece CMMI, disiplinler arası iş birliğini ve süreç iyileştirmeyi bütüncül bir yaklaşımla ele alan daha gelişmiş bir model olarak geliştirilmiştir.

2.2. CMMI

Günümüz dünyasında, teknolojik gelişmeler hızla ilerlerken, yazılım geliştirme süreçlerinin etkili ve verimli bir şekilde yönetilmesi rekabet avantajı elde etmenin anahtarıdır. Yazılım süreç iyileştirme ve yetenek değerlendirme modelleri, organizasyonların bu hedeflere ulaşmasında kritik bir rol oynamaktadır. Bu modeller, yazılım geliştirme süreçlerinin kalitesini artırmak, bu süreçlerin olgunluk seviyelerini belirlemek ve organizasyonel yetenekleri değerlendirmek için çeşitli yaklaşımlar sunar [45]. Bu bağlamda, Yetenek Olgunluk Modeli Entegrasyonu (Capability Maturity Model Integration – CMMI), yalnızca yazılım mühendisliğini değil, bunun ötesine geçen bir dizi disiplini de kapsayacak şekilde, CMM teknolojisinin kullanılabilirliğini artırmak amacıyla ve CMM modelinin sağladığı başarılardan yola çıkarak geliştirilmiştir.

CMMI, yazılım sektöründeki kuruluşların süreçlerini geliştirmeye yönelik bir modeldir. Bu model organizasyonların, yazılım geliştirme, planlama ve proje yönetimi gibi çeşitli süreçlerinin olgunluk seviyelerini değerlendirmek için kullanılır [54].

Zamanla farklı alanlara odaklanan çeşitli CMM'lerin aynı kuruluş içerisinde bir arada kullanılma isteği, kullanıcılar açısından karışıklıklara yol açmış ve bu modeller arasındaki uyumsuzlukları gün yüzüne çıkarmıştır. Bu durum, kuruluşların süreçlerini daha etkili bir şekilde yönetebilmeleri için yeni ve kapsamlı bir entegre modele duyulan ihtiyacı ortaya çıkarmıştır. Karşılaşılan bu sorunları çözmek amacıyla 2001 yılında CMM Tümleştirme Projesi başlatılmış ve bu projenin sonucunda, 2002'de CMMI v1.0, 2003'te v1.1, 2006'da v1.2, 2010'da v1.3 ve 2018'de v2.0 sürümleri yayınlanmıştır [18]. Şekil 7'de CMMI'ın versiyonları ve yayınlanma tarihleri gösterilmektedir.



Şekil 7. CMMI'ın versiyonları ve yayınlanma tarihleri

CMMI, başlangıçta üç farklı kaynak modelinin entegrasyonunu gerçekleştiren tek bir model olarak tasarlanmıştır: Yazılım Yetenek Olgunluk Modeli (SW-CMM), Sistem Mühendisliği Yetenek Modeli (SE-CMM) ve Entegre Ürün Geliştirme Yetenek Olgunluk Modeli (IPPD-CMM) [55]. Bu üç kaynak model, organizasyonlarda süreç iyileştirmelerine yönelik gösterdiği başarılı uygulama ve vaatler nedeniyle tercih edilmiştir.

CMMI v1.2 ile birlikte organizasyonların farklı alanlarda süreç iyileştirme ihtiyaçları doğrultusunda üç ana alt model tanımlanmıştır.

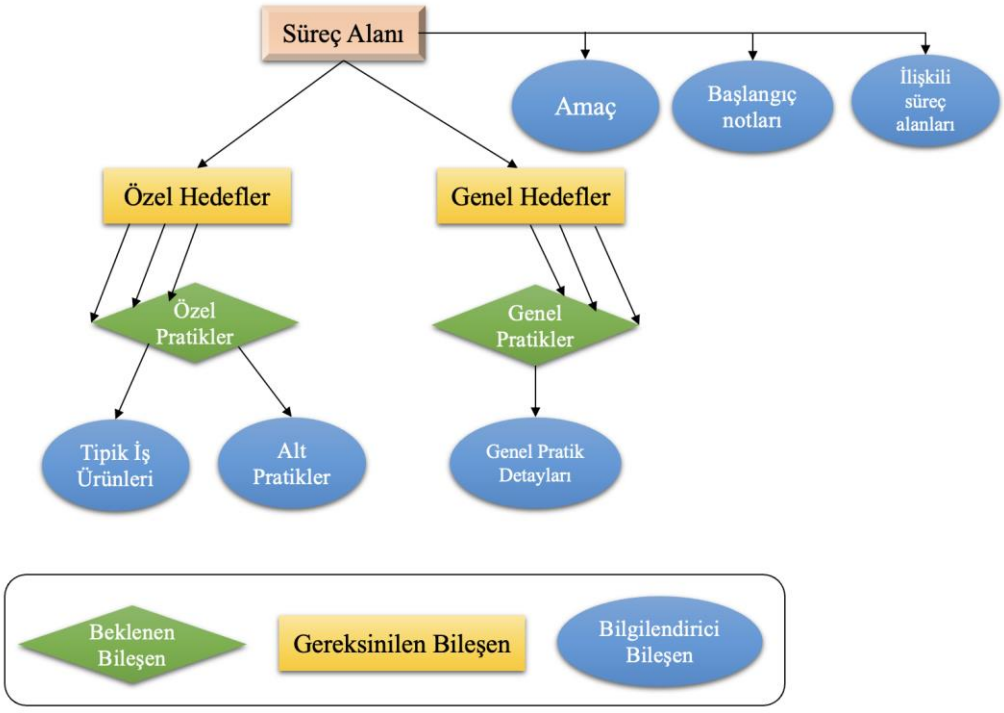
- ***Geliştirme Amaçlı CMMI (CMMI for Development – CMMI-DEV):*** Bu model, yazılım ve sistem geliştirme süreçlerine odaklanmaktadır. Organizasyonların geliştirme süreçlerinin olgunluğunu artırmaya yardımcı olmak, yazılım geliştirme sürecindeki karmaşıklıkları ve gereksinimleri karşılamak için amacıyla tasarlanmıştır [56].
- ***Hizmet Amaçlı CMMI (CMMI for Services – CMMI-SVC):*** Bu model, hizmet sektöründeki organizasyonlar için geliştirilmiştir. Hizmet süreçlerinin yönetimini ve iyileştirilmesini sağlamak, hizmet kalitesini artırmak ve müşteri memnuniyetini sağlamak amacıyla geliştirilmiştir [57]. Savunma, sağlık, bilgi teknolojisi, finans, ulaşım gibi çok çeşitli alanlarda hizmet sunan organizasyonlar tarafından kullanılmaktadır [58].
- ***Satın Alma Amaçlı CMMI (CMMI for Acquisition – CMMI-ACQ):*** Bu model, organizasyonların yazılım ve sistem tedarik süreçlerini iyileştirmeye yönelik olarak geliştirilmiştir [59]. Satın alma süreçlerinde etkinliği artırmayı hedeflerken, ayrıca tedarik zinciri yönetimi gibi konulara da odaklanmaktadır.

Alt modellerin geliştirilmesi, organizasyonların farklı ihtiyaçlarını karşılamak ve çeşitli alanlarda süreç olgunluğunu artırmak amacıyla CMMI'ın esnekliğini ve kapsamını genişletmiştir. Bunun yanında önceki beş olgunluk düzeyi korunmuş ancak daha kapsamlı olacak şekilde yeni süreç alanları eklenmiştir.

2.2.1. CMMI Yapısı

CMMI'ın yapısı, organizasyonların süreçlerini iyileştirmelerini sağlamak için belirli gereksinimleri ve pratikleri tanımlar. Bu yapının nasıl uygulanacağını ve yorumlanacağını belirleyen üç ana bileşen kategorisinden oluşur: gereksinilen, beklenilen ve bilgilendirici bileşenler [60]. Her bir bileşen, organizasyonların süreçlerini farklı seviyelerde nasıl iyileştirmeleri gerektiğine dair rehberlik sağlar ve her seviyedeki hedeflere ulaşabilmek için izlenmesi gereken yolları netleştirir.

Gereksinilen bileşenler (Required Components); organizasyonun süreç alanını icra etmek, belirli bir seviyeye getirmek üzere ne elde etmesi gerektiğini anlatır. CMMI'daki gereksinilen bileşenler, özel ve genel hedeflerdir [60]. Beklenen bileşenler (Expected Components); gereksinilen bileşeni elde etmek üzere ne uygulanacağını anlatır. Beklenilen bileşenler özel ve genel pratikleri kapsar [55]. Bilgilendirici bileşenler (Informative Components); organizasyonun gereksinilen ve beklenilen bileşenlere nasıl yaklaşacağı hakkında yardımcı olacak olan detayları sağlar. Amaç, başlangıç notları, ilişkili süreç alanları, alt pratikler, tipik iş ürünleri ve genel pratik detayları bilgilendirici bileşenlerdir [55]. Şekil 8'de CMMI'ın yapısı görülmektedir [55].



Şekil 8. CMMI'ın yapısı

CMMI'ın temelini süreç alanları oluşturur. Şekil 8'de görüldüğü üzere her süreç alanı; amaç, tanıtıcı başlangıç notları, ilişkili süreç alanları, özel ve genel hedeflere sahiptir [61]. Özel hedefler altında özel pratikler; özel pratikler altında da tipik iş ürünleri ve alt pratikler mevcuttur. Genel hedefler altında ise genel pratikler ve genel pratik detayları bulunur.

Süreç alanı bir organizasyonun ilgili sürecini iyileştirmek amacıyla geliştirilmesi gereken belirli bir odak noktasıdır. Her süreç alanı, bir amaca sahiptir. Başlangıç notları süreç alanı hakkında tanımlayıcı ve önemli bilgileri içerir [62]. Bu notlar sürecin planlanması ve düzgün işleyişi için temel gerekliliklerden biridir İlişkili süreç alanları ise ilgili süreç alanı ile bağlantılı ve tamamlayıcı uygulamalara sahip diğer süreç alanlarının referans listesini oluşturur. Bu referanslar, belirli bir süreç alanının uygulanması sırasında, diğer süreç alanlarında yer alan hedeflerin de dikkate alınması gerektiğini ifade eder.

Her süreç alanı için belirlenen özel hedefler, o alandaki başarıya ulaşmak için organizasyonların karşılaması gereken belirli gereksinimleri tanımlar. Bu hedefler, sürecin başarısını ölçmek için kullanılan spesifik kriterlere odaklanır. Özel pratikler, süreç alanının belirli bir özel hedefine ulaşmak için kritik öneme sahip olarak kabul edilen faaliyet veya işlem olarak tanımlanırlar. Bu pratikler, özel hedeflerinin gerçekleştirilmesi için yapılması gereken gerekli adımları belirler. Alt pratikler, özel pratiklerin nasıl uygulanacağına dair daha detaylı bir açıklama sunar. Bu pratikler, süreçlerin hayata geçirilmesi için gerekli spesifik faaliyetleri tanımlar [63]. Tipik iş ürünleri süreç alanının uygulanması sırasında oluşturulan belge, rapor, analiz veya yazılım çıktısı gibi somut varlıklardır. Bu ürünler, genellikle sürecin doğru bir şekilde gerçekleştirildiğini kanıtlamak için kullanılır ve organizasyonların, sürecin her aşamasını denetleyebilmesini ve iyileştirebilmesini sağlar [64].

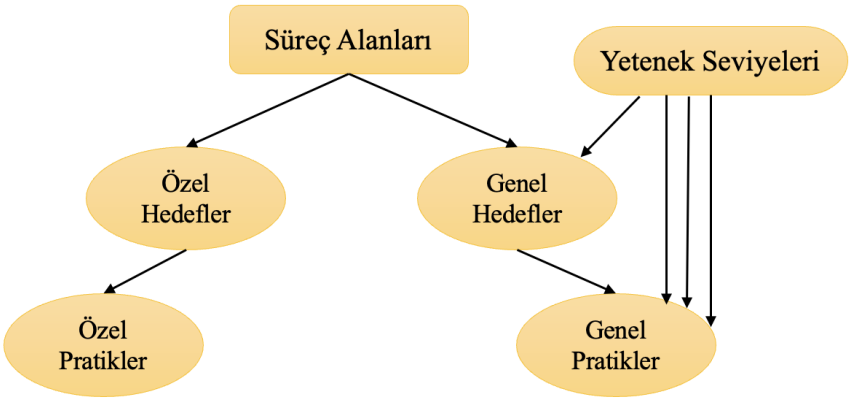
Genel hedefler, organizasyonun süreç olgunluğunu artırmak için uygulanması gereken geniş kapsamlı hedeflerdir [54]. Süreç alanının olgunlaşması için gereksinim duyulan temel unsurları tanımlar. Aynı hedeflerin birden çok süreç alanında ortaya çıkması sebebi ile "genel" olarak adlandırılır. Genel pratikler organizasyon genelinde sürecin etkinliğini ve verimliliğini artırmaya yönelik, süreç alanı için geçerli olan genel hedeflere ulaşmada önemli olarak değerlendirilen aktivitelerdir. Genel pratik detayları, genel pratiklerin nasıl uygulanabileceğine dair rehberlik niteliğinde bilgileri içerir. Bu

yapıların her biri, sürecin etkin bir şekilde uygulanmasını sağlamak için gerekli olan spesifik adımları tanımlar ve organizasyonun hedeflerine ulaşabilmesi için kritik bir yol haritası oluşturur.

2.2.2. CMMI Gösterim Şekilleri

CMMI, organizasyonların süreç iyileştirmeleri ve olgunluk seviyeleri arasındaki geçişleri yönetmelerine yardımcı olmak amacıyla Basamaklı Gösterim (Staged Representation) ve Sürekli Gösterim (Continuous Representation) olmak üzere iki ana gösterim ile sunulur [65].

Sürekli Gösterim Modeli, organizasyonlara süreç alanlarında bağımsız olarak ilerleme ve iyileştirme fırsatı tanır. Bu yaklaşım, süreç alanlarında daha hızlı ilerleme sağlarken, genel olgunluk seviyesinin zamanla artmasını teşvik eder [65]. Her süreç alanında tanımlanan yetenek seviyeleri iyileştirme çalışmaları için sıralı bir yaklaşım sağlar. Sürekli gösterim modelinin yapısı Şekil 9'da görülmektedir [65].

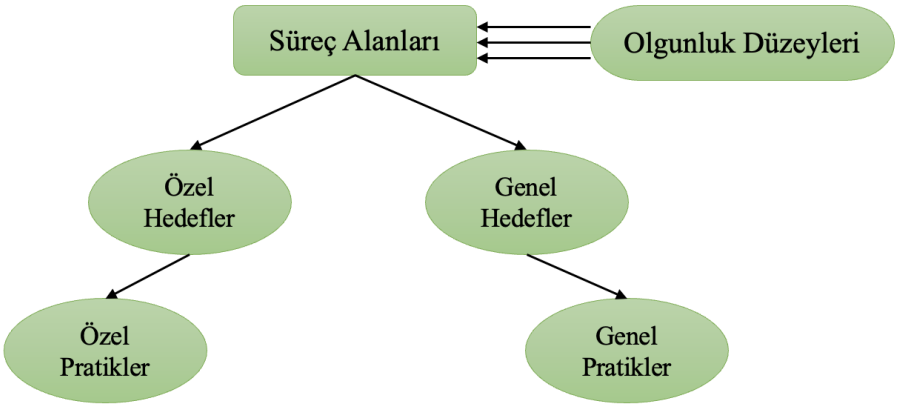


Şekil 9. Sürekli gösterim modeli

Model Eksik, İcra Edilen, Yönetilen, Tanımlı, Nicel Yönetilen ve En İyileşen olmak üzere altı yetenek düzeyine sahiptir, 0 ile 5 arasında numaralandırılmıştır. Yetenek düzeyleri, süreç alanlarında belirli bir seviyeye ulaşmak için gereken adımları, hedefler ve pratikler ise bu adımların nasıl yapılacağını tanımlar [66]. Bu yapı, organizasyonların süreçlerini sürekli olarak iyileştirebilmeleri ve her süreç alanında daha yüksek olgunluk seviyelerine ulaşabilmeleri için gerekli temeli oluşturur [67]. CMMI'ın sürekli gösterim modeli yazılım süreçlerini değerlendirmek ve iyileştirmek için geliştirilmiş uluslararası bir standart olan ISO 1550 (SPICE – Software Process Improvement and Capability Determination) ile de uyumludur [68].

Her iki model de süreçlerin yetenek düzeyleri temelinde değerlendirilmesini ve süreç iyileştirme adımlarının kademeli bir şekilde uygulanmasını öngörmektedir. Bu bağlamda, organizasyonların süreçlerini sürekli olarak daha yüksek olgunluk seviyelerine taşımaları hedeflenmektedir.

Basamaklı Gösterim Modeli, aşama aşama, sistematik ve yapılandırılmış bir yaklaşım sunar. Süreç alanlarını belirli bir sıraya koyarak, organizasyonların her bir seviyeyi tamamlayarak bir üst seviyeye geçmesini öngörür. Bu model, süreç iyileştirmelerini adım adım ve daha düzenli bir şekilde gerçekleştirmenin yolunu sunar [69]. Basamaklı gösterim, organizasyonların hangi süreçlerin öncelikli olarak iyileştirilmesi gerektiğini belirlemek konusunda belirsizlik yaşadığı durumlarda en uygun model olarak kabul edilmektedir. Bu model, her olgunluk seviyesi için belirli süreç alanlarının iyileştirilmesine yönelik öneriler sunarak organizasyonlara yapılandırılmış bir iyileştirme yolu sağlar [70]. Şekil 10'da basamaklı gösterim modelinin yapısı görülmektedir. Model Başlangıç, Yönetilen, Tanımlı, Nicel Yönetilen ve En İyileşen olmak üzere beş olgunluk düzeyine sahiptir ve bu düzeyler 1 ile 5 arasında numaralandırılmıştır [55]. Her olgunluk seviyesinde bir sonraki seviyedeki süreçlerin etkin bir şekilde gerçekleştirilmesi için başarılması gerekli süreç alanları vardır.



Şekil 10. Basamaklı gösterim modeli

Her iki gösterim de farklı ihtiyaçlara ve hedeflere hizmet eder, bu nedenle organizasyonun mevcut süreç yapısına ve iyileştirme hedeflerine uygun olanı seçmek en iyi sonuçları elde etmek için kritik öneme sahiptir. Kullanılacak gösterim modelinin seçimi, organizasyonların iş hedeflerine dayalı olarak yapılmalıdır. Sürekli gösterim, organizasyonlara daha ayrıntılı ve esnek bir iyileştirme stratejisi sunarak, bu stratejinin

organizasyonun genel hedefleriyle uyumlu olmasını sağlar. Bu model, organizasyonların belirli bir süreç alanına odaklanmasına ve o alanın yetenek düzeyini geliştirmesine olanak tanır. Diğer taraftan, basamaklı gösterim, CMM'den CMMI'a geçiş yapmak isteyen organizasyonlar için daha uygun bir seçenektir, çünkü bu geçiş, minimum ek çaba gerektirir. Her iki model de organizasyonların süreçlerini iyileştirmek için yapılandırılmış ve düzenli bir yol haritası sunar. Bu, özellikle organizasyonların sertifikasyon sürecindeki deneyimlerine dayanarak geçişi kolaylaştırır [71].

2.3. CMMI-DEV

2006 yılında yayınlanan CMMI 1.2 versiyonu CMMI v1.1'in devamı olan güncellenmiş bir versiyondur. Bu sürüm CMMI uygulamalarını daha geniş bir yelpazeye yaymak amacıyla geliştirilmiştir [72]. CMMI v1.2 ile "takımyıldızları (constellations)" kavramı tanıtılarak, CMMI modelinin kapsamı genişletilmiş ve spesifik hedeflere yönelik farklı modellerin oluşturulması sağlanmıştır. Her takımyıldızı, belirli bir alandaki ihtiyaçları karşılayacak şekilde, CMMI'ın temel bileşenleri üzerine ilave ek materyaller içerir [73].

Geliştirme amaçlı CMMI (CMMI for Development – CMMI-DEV), ilk takımyıldızıdır. Yazılım ve sistem geliştirme süreçlerinin iyileştirilmesini hedefleyen, sektörde en iyi uygulamalardan derlenen bir modeldir [74]. Ürünlerin ve hizmetlerin tüm yaşam döngüsünü ele alarak geliştirme, teslimat ve bakım süreçlerinde etkinliği artırmayı amaçlar. Model, bir ürünün fikir aşamasından nihai teslimatına kadar geçen süreçlerdeki tüm adımları yapılandırarak, geliştirme faaliyetlerinin daha verimli, sürdürülebilir ve kaliteli bir şekilde yürütülmesini sağlar [55]. CMMI-DEV'in süreçleri, kuruluşların yazılım ve sistem geliştirme süreçlerinde karşılaştığı engelleri aşmalarına ve bu süreçlerdeki olgunluk seviyelerini artırmalarına yardımcı olur.

CMMI-DEV'in tanıtılması sonrasında hükümet ve endüstride tedarik zinciri yönetimi, satın alma ve kaynak kullanım süreçlerini ele alan Tedarik için CMMI (CMMI for Acquisition – CMM-ACQ) ve hizmet sektöründeki organizasyonların hizmetlerini iyileştirmelerini, verimliliklerini artırmalarını ve kaliteli hizmet sunmalarını amaçlayan Hizmet için CMMI (CMMI for Services – CMMI-SVC) takımyıldızları tanıtılmıştır [75].

2010 yılında yayınlanan CMMI v1.3 versiyonu, önceki versiyonlardan alınan geri bildirimlere dayanarak geliştirilmiştir [76]. Öne çıkan ilk değişiklik, yüksek olgunluk seviyelerinin (Olgunluk Düzeyi 4 ve 5) açıklığa kavuşturulmasıdır. CMMI v1.3, olgunluk seviyelerini daha anlaşılır bir şekilde tanımlamış ve organizasyonların süreçlerini geliştirme sürecinde daha belirgin bir yol haritası sunmuştur [77]. Bu sürümde, özellikle iş hedefleri ile süreç iyileştirmeleri arasında güçlü bir bağlantı kuran yeni bir süreç alanı olan Organizasyonel Performans Yönetimi (Organizational Performance Management – OPM) eklenmiştir [78]. OPM, organizasyonların stratejik hedefleri ile süreç iyileştirmelerini birleştirerek, iyileştirme çabalarının doğrudan iş sonuçlarına katkıda bulunmasını sağlar. Bu değişiklik, CMMI'ı yalnızca bir olgunluk modeli olmaktan çıkarıp, organizasyonel hedeflerle uyumlu süreç geliştirme aracına dönüştürmüştür.

CMMI v1.3'teki bir diğer önemli gelişme, basamaklı ve sürekli gösterimler arasındaki farkların daha esnek bir hale getirilmesidir. Önceki sürümlerde bu iki gösterim, farklı süreç yapıları ve ölçütleri içeriyordu, bu da organizasyonları hangi temsili kullanacaklarına dair bir seçim yapmaya zorlayan bir durum oluşturuyordu. Ancak v1.3 ile bu iki temsil arasındaki sınırlar esnetilmiş ve her iki yaklaşımın daha uyumlu bir şekilde uygulanabilmesi sağlanmıştır. Özellikle olgunluk seviyelerinin tanımlanmasında yapılan sadeleştirmeler ve süreç alanlarındaki genel hedeflerin tek bir bölümde toplanması, modelin daha anlaşılır ve kullanıcı dostu olmasına olanak tanımıştır. Modelin içerdiği 22 süreç alanı, organizasyonların süreçlerini planlama, yönetme, izleme ve iyileştirme süreçlerini içermektedir. Süreç alanları, basamaklı gösterimde her biri belirli bir olgunluk seviyesinde geliştirilmesini sağlayacak şekilde yapılandırılır [79]. Bu sayede organizasyonlar, her bir süreç alanını sırayla iyileştirerek daha olgun bir seviyeye ulaşabilirler. Sürekli gösterimde ise süreç alanları Tablo 4'te görüldüğü üzere, kapsam alanlarına göre Mühendislik, Süreç Yönetimi, Proje Yönetimi, Destek olmak üzere dört ana kategoriye ayrılmıştır [64], [73], [80]. Her bir kategori, farklı süreç alanlarını kapsar ve organizasyonlar, kendi ihtiyaçlarına göre bu kategorilerdeki süreç alanlarını seçip iyileştirebilirler.

Tablo 4. CMMI-DEV V1.3 süreç kategorileri ve süreç alanları

Süreç Kategorisi	Süreç Alanları
Mühendislik	Ürün Entegrasyonu (PI)
	Gereksinim Geliştirmesi (RD)
	Teknik Çözüm (TS)
	Doğrulama (VAL)
	Geçerleme (VER)
Süreç Yönetimi	Organizasyonel Süreç Tanımı (OPD)
	Organizasyonel Süreç Odaklılık (OPF)
	Organizasyonel Süreç Performansı (OPP)
	Organizasyonel Performans Yönetimi (OPM)
	Organizasyonel Eğitim (OT)
Proje Yönetimi	Entegre Proje Yönetimi (IPM)
	Proje İzleme ve Kontrol (PMC)
	Proje Planlaması (PP)
	Nicel Proje Yönetimi (QPM)
	Gereksinim Yönetimi (REQM)
	Risk Yönetimi (RSKM)
	Tedarikçi Sözleşme Yönetimi (SAM)
Destek	Nedensel Analiz ve Çözüm (CAR)
	Konfigürasyon Yönetimi (CM)
	Karar Analizi ve Çözümü (DAR)
	Ölçüm ve Analiz (MA)
	Süreç ve Ürün Kalite Güvencesi (PPQA)

2.3.1. CMMI-DEV V2.0

CMMI v2.0 2018 yılının mart ayında yayınlanan yeni bir sürümdür ve daha önceki sürüm olan CMMI V1.3'e göre önemli bir evrim geçirmiştir. Bu yeni sürüm, yalnızca süreçlerin iyileştirilmesiyle değil, aynı zamanda organizasyonların iş performansını artırmaya yönelik somut yollar sunan bir model olarak tanımlanmaktadır [1].

CMMI v2.0, önceki sürümlere göre daha esnek, anlaşılabilir ve kullanıcı dostu bir yapı sunmaktadır. Model, yalnızca belirli bir alanda yetkinlik kazandırmakla kalmayıp, aynı zamanda organizasyonların tüm ilgili yetkinlik alanlarında gelişim göstererek, belirlenen olgunluk seviyelerine ulaşmalarını sağlamaktadır. Bu yaklaşım, organizasyonların süreç iyileştirmeleri için daha kapsamlı ve bütünleşmiş bir yol haritası izlemekte olanak tanır [81].

CMMI v2.0 modeliyle birlikte CMMI Enstitüsü yeni bir terminoloji oluşturmuştur. Yeni versiyonda "Süreç Alanı (Process Area)" kavramının yerini "Uygulama Alanı (Practice Area)" kavramı almıştır. Bu değişiklikle, süreç alanlarının sadece süreçler değil, aynı zamanda organizasyonların değer yaratma çabalarına odaklanan uygulama grupları olduğu vurgulanmıştır. Aynı şekilde CMMI-DEV v1.3'te yer alan süreç alanı kategorileri hem kavramsal hem de içerik olarak değişmiş; "Süreç Alanı Kategorisi" ifadesi yerini Yetenek Alanları (Capability Area)'na bırakmıştır [82]. Eski versiyonda Mühendislik, Süreç Yönetimi, Proje Yönetimi, Destek olarak tanımlı süreç kategorileri kaldırılmış, yeni kategorizasyon yaklaşımıyla organizasyonların kaynaklarını önceliklendirmesi, organize etmesi ve kritik sorunlara odaklanmasına yardımcı olacak Yapım (Doing), Yönetim (Managing), Destekleyici (Enabling) ve İyileştirici (Improving) olmak üzere dört yeni "Kategori Alanı (Category Area)" belirlenmiştir [83].

Kategori alanlarının her biri, bir organizasyonda veya projede performansı iyileştirmeye yönelik uygulamaları gruplandıran tanımlanmış yetenek alanlarını içerir. Her yetenek alanı da organizasyonların belirli yetenekleri nasıl uygulaması gerektiğini daha ayrıntılı bir şekilde değerlendiren uygulama alanlarına sahiptir [84]. CMMI v2.0, bu yapının içinde, toplamda 12 yetenek alanı tanımlar. Ancak, CMMI-DEV v2.0 için yalnızca 9 yetenek alanı geçerlidir [85]. Tablo 5'te CMMI-DEV v2.0'ın sahip olduğu 4 kategori alanı, her kategori alanı altında yer alan yetenek alanları ve her yetenek alanı altında toplanan uygulama alanları görülmektedir.

Tablo 5. CMMI-DEV V2.0 kategori, yetenek ve uygulama alanları tablosu

Kategori Alanı	Yetenek Alanı	Uygulama Alanı
Yapım (Doing)	**Kalite Güvencesi (ENQ)**	Gereksinim Geliştirme ve Bakımı (RDM)
		Süreç Kalite Güvencesi (PQA)
		Eşdeğer Gözden Geçirme (PR)
		Doğrulama ve Geçerleme (VV)
	Mühendislik ve Ürün Geliştirme (EDP)	Teknik Çözüm (TS)
		Ürün Entegrasyonu (PI)
	Tedarikçi Seçme ve Yönetme (SMS)	Tedarikçi Sözleşmesi Yönetimi (SAM)
Yönetim (Managing)	**İş Planlama ve Yönetme (PMW)**	Tahmin (EST)
		Planlama (PLAN)
		İzleme ve Kontrol (MC)
	İş Sürekliliğini Yönetme (MBR)	Risk ve Fırsat Yönetimi (RSK)

	İşgücünü Yönetme (MWF)	Organizasyonel Eğitim (OT)
Destekleyici (Enabling)	**Uygulama Destekleme (SI)**	Neden Analizi ve Çözüm (CAR)
		Karar Analizi ve Çözümü (DAR)
		Konfigürasyon Yönetimi (CM)
İyileştirici (Improving)	**Alışkanlığı ve Sürekliliği Sürdürme (SHP)**	Yönetim (GOV)
		Uygulama Altyapısı (II)
	Performans İyileştirme (IMP)	Süreç Yönetimi (PCM)
		Süreç Varlığı Geliştirme (PAD)
		Performans Yönetimi ve Ölçümü (MPM)

Yapım kategori alanı, temel süreçlerin gerçekleştirilmesine odaklanır ve temel projelerin, faaliyetlerin yürütülmesi, gereksinimlerin karşılanması, ürünlerin geliştirilmesi ve teslim edilmesi süreçlerini içerir [86]. Yapım kategori alanı "Kalite Güvencesi", "Mühendislik ve Ürün Geliştirme" ve "Tedarikçi Seçme ve Yönetme" olmak üzere üç yetenek alanına sahiptir. **Kalite Güvencesi** yetenek alanı, organizasyonun ürün ve hizmetlerini kalite açısından değerlendirmesini ve güvence altına almasını sağlar. Kalite standartlarının belirlenmesi, izlenmesi, denetlenmesi ve iyileştirilmesi bu yetenek alanı altında ele alınır. Hataları önceden önlemeye yönelik önlemler ve sürekli iyileştirme bu yetenek alanının temel konuları arasındadır [82]. **Mühendislik ve Ürün Geliştirme** yetenek alanı, organizasyonun ürün geliştirme faaliyetlerini yönetmesini sağlar. Gereksinim analizi, tasarım, kodlama, test ve ürünün tüm yaşam döngüsü süreçleri bu kategori altında değerlendirilir. Bu yetenek alanı, ürün mühendisliği süreçlerini etkili bir şekilde planlamayı ve uygulamayı hedefler. **Tedarikçi Seçme ve Yönetme** yetenek alanı, organizasyonun dış tedarikçilerle olan ilişkilerini yönetmeyi ve tedarikçi süreçlerini denetlemeyi hedefler. Uygun tedarikçilerin seçilmesi, sözleşmelerin düzenlenmesi, tedarikçi performansının izlenmesi ve tedarik zinciri yönetimi bu yetenek alanı altında değerlendirilir [87].

Yönetim kategori alanı, süreçlerin planlanması, izlenmesi ve kontrol edilmesiyle ilgilidir [86]. Proje yönetimi, kaynak yönetimi, risk yönetimi gibi süreçler, bu kategori altında değerlendirilir. Yönetim kategori alanı "İş Planlama ve Yönetme", "İş Sürekliliğini Yönetme" ve "İşgücünü Yönetme" olmak üzere üç yetenek alanına sahiptir. **İş Planlama ve Yönetme** yetenek alanı, organizasyonun projeleri ve iş faaliyetlerini planlamasını, izlemesini ve yönetmesini hedefler. Proje planlaması, kaynak yönetimi, zaman yönetimi, risk yönetimi ve takım koordinasyonu gibi konular

bu yetenek alanı altında ele alınır [88]. **İş Sürekliliğini Yönetme** yetenek alanı, organizasyonun iş sürekliliğini yönetmesini ve acil durumlarla başa çıkmasını sağlar. İş sürekliliği planlarının oluşturulması, acil durum senaryolarının değerlendirilmesi, bu senaryolara yönelik hazırlıkların yapılması ve organizasyonun hızlı bir şekilde toparlanmasını sağlamak bu yetenek alanı kapsamındadır [89]. **İşgücünü Yönetme** yetenek alanı, organizasyonun insan kaynakları yönetimi ile ilgili konuları ele alır. İşgücü planlaması, personel yönetimi, eğitim, performans değerlendirmesi ve çalışanların motivasyonunu artırmak bu yetenek alanı içinde değerlendirilir. İşgücünün etkili bir şekilde yönetilmesi, organizasyonun hedeflerine ulaşmasına ve süreçlerin sürdürülebilirliğine katkı sağlar.

Destekleyici kategori alanı organizasyonun alt yapı ve destek süreçlerini içerir [86]. İnsan kaynakları yönetimi, eğitim, altyapı yönetimi ve diğer destekleyici süreçler bu kategoriye dahildir. Organizasyonun süreçlerini destekleyen ve etkinleştiren unsurlar burada değerlendirilir. Destekleyici kategori alanı altında olan **Uygulama Destekleme** yetenek alanı, organizasyonun belirli süreçlerin uygulanmasını destekleyen fonksiyonları ele alır. Bu, örneğin, dokümantasyon oluşturma, eğitim sağlama, gereksinimleri belirleme ve süreç uygulamalarının takibini içerir. Uygulama sürecini destekleyen altyapı ve kaynaklar bu yetenek alanının kapsamında değerlendirilir [90].

İyileştirici kategori alanı sürekli iyileştirme ve inovasyon ile ilgilidir. Organizasyonların süreçlerini nasıl optimize edebilecekleri, daha etkili ve verimli hale getirebilecekleri bu kategori altında incelenir. Bu kategori alanı yenilikçilik, sürekli öğrenme ve gelişme konularına odaklanır [84]. Bu kategori alanı altında "Alışkanlığı ve Sürekliliği Sürdürme" ve "Performansı İyileştirme" olmak üzere iki yetenek alanı bulunmaktadır. **Alışkanlığı ve Sürekliliği Sürdürme** yetenek alanı, organizasyonun belirli bir davranış, süreç veya yöntemin benimsenmesini ve sürdürülmesini sağlamayı hedefler. İnsanlar arasında belirli bir alışkanlığın veya sürecin benimsenmesi genellikle zaman alabilir, bu yetenek alanı ise bu benimseme sürecini destekleyerek bir standart oluşturarak sürekliliği sağlamaya odaklanır. Örneğin, yeni bir geliştirme metodolojisinin benimsenmesi veya sürekli iyileştirme alışkanlığının oluşturulması gibi durumları içerebilir [82]. **Performansı İyileştirme** yetenek alanı, organizasyonun sürekli olarak performansını artırmayı hedefler. Sürekli iyileştirme, iş süreçlerinin,

ürünlerin veya hizmetlerin kalitesini ve verimliliğini artırmak amacıyla yapılan sistemli çabaları içerir. Performans iyileştirme süreci, sürekli ölçme, analiz ve uygulama geribildirimini içerir. Organizasyonlar, performanslarını değerlendirir ve sürekli olarak daha etkili ve verimli olmak için çaba gösterir.

2.3.1.1. CMMI-DEV V2.0 Uygulama Alanları

CMMI-DEV V2.0'daki uygulama alanları (Practice Areas, PA'lar), organizasyonların süreçlerini iyileştirmelerine yönelik kapsamlı bir rehber sunar. Her bir uygulama alanı, belirli bir işlevi hedefler ve süreçlerin daha verimli, sürdürülebilir ve yapılandırılmış bir şekilde yönetilmesini sağlar. CMMI-DEV V2.0 içerisinde 20 adet uygulama alanı bulunmaktadır. Uygulama alanları 4 kategori alanının içerdiği 9 yetenek alanı altında gruplanmıştır.

Gereksinim Geliştirme ve Yönetimi (Requirements Development & Management – RDM): Amacı; doğru gereksinimleri belirlemek ve paydaşların beklentilerine uygun olarak gereksinimleri yönetmek ve korumaktır. Eski versiyonlarda bulunan Gereksinim Geliştirme (Requirements Development – RD) ve Gereksinim Yönetimi (Requirements Management – REQM) süreç alanlarının birleşimiyle oluşturulmuş olup, organizasyonların performans iyileştirmelerine odaklanmalarını sağlar [91]. Gereksinim geliştirme ve yönetimi uygulama alanı, yapım kategori alanının içerdiği kalite güvencesi yetenek alanı altında bulunur ve olgunluk düzeyi olarak 1, 2 ve 3. olgunluk düzeylerinde yer almaktadır. Bu uygulama alanı aşağıdaki hususları içerir [81];

- Paydaşların taleplerini, hedeflerini, kısıtlamalarını, bağlantılarını ve arayüzlerini belirlemek,
- Bu unsurları kullanarak önceliklendirilmiş müşteri gereksinimlerini oluşturmak,
- Gereksinimlerin anlamını, gereksinim sahipleriyle görüşerek açıklığa kavuşturmak,
- Sistem gereksinimlerini tanımlamak, analiz etmek ve yönetmek,
- Projeye dahil olan herkese, gereksinimlerin başarıyla yerine getirileceğine dair güvence vermek,
- Gereksinimleri, faaliyetler ve çıktılar ile iki yönlü olarak izlemek,

- Planların, eylemlerin ve çıktıların gereksinimlere uygun olduğundan emin olmak.

Süreç Kalite Güvencesi (Process Quality Assurance – PQA): Bu uygulama alanı, süreçlerin belirli kalite standartlarına uygunluğunu sağlamak için gerekli olan kalite güvence (Quality Assurance) faaliyetlerini kapsar [92]. Amacı; süreçlerin sürekli olarak izlenmesi ve denetlenmesi yoluyla kaliteyi, iş verimliliğini ve müşteri memnuniyetini artırmaktır [93]. Bu yaklaşım, hataların erken tespit edilmesini sağlar ve süreçlerin belirlenen standartlara uygunluğunu garanti eder. Süreç kalite güvencesi uygulama alanı, yapım kategori alanının içerdiği kalite güvencesi yetenek alanı altında bulunur ve 1, 2 ve 3. olgunluk düzeylerinde yer almaktadır. Bu uygulama alanı aşağıdaki hususları içerir [81], [94];

- Geçmiş kalite verilerine dayalı olarak bir kalite güvence yaklaşımı ve planı geliştirmek,
- Proje boyunca, gerçekleştirilen süreçleri ve ürün çıktılarını, geçerli standartlarla objektif bir şekilde değerlendirmek,
- Kalite ve uyumsuzluk sorunlarını iletmek ve bu sorunların çözülmesini sağlamak.

Eşdeğer Gözden Geçirme (Peer Reviews – PR): Bu uygulama alanı, süreçlerin ya da ürünlerin, benzer yetkinlikteki kişiler tarafından gözden geçirilmesini içerir. Amacı; ekipler arasındaki iş birliğini artırarak, süreç kalitesini ve sonuçların doğruluğunu artırmaktır [91]. Karşılıklı değerlendirmelerle hataların erken tespiti ve çözümü sağlanır. Bu sayede maliyetler ve yeniden çalışma azalır. Eşdeğer gözden geçirme uygulama alanı, yapım kategori alanının içerdiği kalite güvencesi yetenek alanı altında bulunur ve 1, 2 ve 3. olgunluk düzeylerinde yer almaktadır. Uygulama alanı aşağıdaki maddeleri içermektedir [82];

- Ürünleri incelemek ve sorunları raporlamak,
- Eşdeğer gözden geçirme (meslektaş incelemesi) yapmak için kullanılan prosedürleri ve destekleyici materyalleri geliştirmek ve güncel tutmak,
- Seçilen ürünler üzerinde, belirlenmiş prosedürlere uygun olarak inceleme yapmak,
- İncelemelerde tespit edilen sorunları çözmek,

- İnceleme sonuçlarını ve verilerini analiz etmek.

Doğrulama ve Geçerleme (Verification & Validation – VV): Bu uygulama alanı ile birlikte CMMI V1.3'te iki ayrı süreç alanı olan doğrulama (Verification – VER) ve geçerleme (Validation – VAL) tek bir süreç altında toplanmıştır [91]. Doğrulama, geliştirilen ürünün ya da sistemin belirlenen gereksinimlere uygun olup olmadığını kontrol eder. Geçerleme ise ürün ya da sistemin gerçek dünya koşullarında belirlenen işlevleri yerine getirip getirmediğini değerlendirir. Uygulama alanının amacı, ürünlerin kalite ve performans açısından beklentileri karşılamasını sağlamaktır [88]93. Doğrulama ve geçerleme, yapım kategori alanının içerdiği kalite güvencesi yetenek alanı altında bulunur ve olgunluk düzeyi olarak 1, 2 ve 3. olgunluk düzeylerinde yer almaktadır. Bu uygulama alanı aşağıdaki hususları içerir [82];

- Gereksinimlerin uygulandığından emin olmak için doğrulama yapmak ve sonuçları raporlamak,
- Çözümün hedef ortamda planlandığı gibi çalışacağını doğrulamak için geçerleme yapmak ve sonuçları raporlamak,
- Doğrulama ve geçerleme için bileşenleri ve yöntemleri seçmek,
- Doğrulama ve geçerlemeyi desteklemek için gereken ortamı geliştirmek, güncel tutmak ve kullanmak,
- Doğrulama ve geçerleme için prosedürler geliştirmek ve güncel tutmak,
- Doğrulama ve geçerleme için kriterler geliştirmek, güncel tutmak ve kullanmak,
- Sonuçları analiz etmek ve raporlamak.

Teknik Çözüm (Technical Solution – TS): Bu uygulama alanı, teknik gereksinimlerin karşılanması için gerekli olan çözümlerin geliştirilmesini içerir [95]. Ürün veya hizmetin mühendislik aşamasında kullanılan tekniklerin ve yaklaşımların tanımlandığı bu alan, inovatif ve sürdürülebilir çözümlerin bulunmasına olanak tanır. Teknik çözüm, mevcut ve yeni teknolojiler, ticari ürünler, araçlar gibi alternatiflerle çözüm geliştirmeyi içerir. Ayrıca, en iyi çözümü seçmek için belirli kriterler (maliyet, zaman, ürün performansı vb.) geliştirilir ve bu kriterlere dayalı olarak en uygun çözüm seçilir [64]. Teknik çözüm, yapım kategori alanının içerdiği mühendislik ve ürün

geliştirme yetenek alanı altında bulunur ve 1, 2 ve 3. olgunluk düzeylerinde yer almaktadır. Bu uygulama alanı aşağıdaki hususları içerir [62], [96];

- Alternatif çözümler ve seçim kriterleri geliştirmek,
- Ürün bileşeni çözümünü seçmek,
- Ürün bileşenini tasarlamak,
- Teknik veri paketini oluşturmak,
- Kriterleri kullanarak arayüz tasarlamak,
- Üretim, satın alım veya yeniden kullanım analizlerini yapmak,
- Tasarımı uygulamak,
- Ürün destek dokümantasyonu geliştirmek.

Ürün Entegrasyonu (Product Integration – PI): Uygulama alanı, ürün bileşenlerinin entegre edilmesi ve bu bileşenlerin gereksinimlere uygun işleyişi sağlanarak nihai ürünün müşteriye teslim edilmesi sürecini kapsar [77]. Entegrasyon işlemi iki farklı şekilde yapılabilmektedir. İlki, tüm ürün bileşenlerinin birleştirilip test edilmesi ve nihai ürün olarak teslim edilmesidir. İkincisi ise bileşenlerin aşamalı olarak entegre edilmesidir [64]. Ürün entegrasyonu uygulama alanı, yapım kategori alanının içerdiği mühendislik ve ürün geliştirme yetenek alanı altında bulunur ve 1, 2 ve 3. olgunluk düzeylerinde yer almaktadır. Bu uygulama alanı aşağıdaki hususları içerir [82];

- Entegrasyon stratejisi geliştirmek,
- Entegrasyon ortamını geliştirmek,
- Çözümleri ve bileşenleri entegre etmek için prosedürler ve kriterler geliştirmek,
- Entegrasyon öncesinde, her bileşenin doğru bir şekilde tanımlandığından ve gereksinimlerine ve tasarımına uygun çalıştığından emin olmak,
- Çözümleri ve bileşenleri entegrasyon stratejisine göre entegre etmek,
- Entegre edilen bileşenleri değerlendirerek, çözümün gereksinimlerine ve tasarımına uyum sağladığından emin olmak,
- Arayüz veya bağlantı açıklamalarını gözden geçirmek ve çözümün ömrü boyunca kapsayıcılık, eksiksizlik ve tutarlılık açısından güncel tutmak,
- Entegrasyon öncesinde, bileşen arayüzlerinin veya bağlantılarının arayüz veya bağlantı açıklamalarıyla uyumlu olduğundan emin olmak,

- Entegre edilen bileşenleri, arayüz veya bağlantı uyumluluğu açısından değerlendirmek.

Tedarikçi Sözleşmesi Yönetimi (Supplier Agreement Management – SAM): Bu uygulama alanı, seçilen tedarikçilerle bir anlaşma kurmayı, anlaşma süresince hem tedarikçinin hem de alıcının anlaşma şartlarına uygun şekilde performans göstermesini sağlamayı ve tedarikçinin teslimatlarını değerlendirmeyi içerir [64]. Sözleşmelerin doğru bir şekilde yönetilmesi, proje boyunca tedarikçilerin sağladığı hizmetlerin belirlenen standartlara uygun olmasını sağlar. Bu süreç, alıcı ve tedarikçi arasında, tedarikçi teslimatlarının başarıyla teslim edilmesi için açık bir anlayış oluşturarak, her iki tarafın da anlaşma koşullarına uygun bir şekilde çalışmasını garanti eder. Bu uygulama alanında, "tedarikçi teslimatı (supplier deliverable)" terimi, bir anlaşma kapsamında alıcıya veya başka bir alıcıya sağlanacak öğeyi ifade eder. Bu öğe bir belge, donanım veya yazılım ürünü, bir hizmet ya da herhangi bir çözüm veya iş ürünü olabilir [93]. Tedarikçi sözleşmesi yönetimi, yapım kategori alanının içerdiği tedarikçi seçme ve yönetme yetenek alanı altında bulunur ve 1, 2, 3 ve 4. olgunluk düzeylerinde yer almaktadır. Tedarikçi sözleşmesi yönetimi uygulama alanı aşağıdaki konuları içerir [73], [81];

- Ürün ya da bileşen için edinim türünün belirlenmesi,
- Tedarikçilerin seçilmesi ve değerlendirilmesi,
- Tedarikçi anlaşmalarının kurulması ve sürdürülmesi,
- Tedarikçi anlaşmalarının uygulanması,
- Alınan tedarikçi teslimatını kabul etmeden önce, tedarikçi anlaşmasının şartlarının yerine getirildiğinden emin olunması,
- Edinilen ürünlerin teslim alınması ve kabul edilmesi,
- Tedarikçi faturalarını, tedarikçi anlaşmalarının şartlarına uygun olarak yönetilmesi ve işlenmesi.

Tahmin (Estimating – EST): Bu uygulama alanı, proje maliyetleri, süreler ve gerekli kaynaklarla ilgili tahminlerin yapılmasını içerir. Bu tahminler, projenin bütçelenmesi ve planlanması açısından kritik önem taşır, planlama yapmak ve belirsizliği azaltmak için bir temel sağlar [91]. Tahmin uygulama alanı, yönetim

kategori alanının içerdiği iş planlama ve yönetme yetenek alanı altında bulunur ve 1, 2 ve 3. olgunluk düzeylerinde yer alır. Uygulama alanı aşağıdaki süreçleri içerir [81];

- Tahminlerin dayandırılacağı bir temel oluşturmak,
- Proje süresi, maliyet, efor ve kaynaklar gibi unsurlar için tahminler geliştirmek,
- Geliştirilen tahminleri paydaşlarla paylaşmak,
- Tahminleri proje boyunca takip etmek ve gerekirse güncellemek.

Planlama (Planning – PLAN): Bu uygulama alanı, proje faaliyetlerinin başarılı bir şekilde uygulanması için detaylı planlar hazırlamayı kapsar. Planlama, tahminlere dayalı olarak bütçeler ve takvimler geliştirmeyi, uygun paydaş ve görev setini belirlemeyi, riskleri yönetmeyi, gerekli kaynakları tespit etmeyi ve projeyi nasıl yürütüleceğini yansıtacak şekilde proje planlarını geliştirip güncellemeyi içerir [93]. Ayrıca, herhangi bir planı uygulamak için gereken kaynakları belirlemeye ve organizasyonun standartları ve kısıtlamaları içinde işi gerçekleştirmek için nelerin gerektiğini anlamaya yardımcı olur. Planlama uygulama alanı, yönetim kategori alanının içerdiği iş planlama ve yönetme yetenek alanı altında bulunur ve 1, 2, 3 ve 4. olgunluk düzeylerinde yer alır [82]. Uygulama alanı aşağıdaki hususları içerir [77];

- İşi tamamlamak için güncel bir plan geliştirmek,
- İşi yapabilmek için gereken bilgi ve becerilere yönelik bir plan oluşturmak,
- Yapılan tahminlere dayalı olarak bir bütçe ve zaman çizelgesi oluşturmak,
- Paydaşların katılımı için uygun bir plan hazırlamak,
- Tüm bileşenlerin tutarlı olduğundan emin olarak güncellenmiş bir proje planı oluşturmak,
- Planları değerlendirmek ve paydaşlardan taahhütler almak.

İzleme ve Kontrol (Monitor & Control – MC): Bu uygulama alanı, proje ilerlemesini görmeyi ve anlamayı sağlar. İzleme ve kontrol ile performansta planlardan sapmalar olursa hızlı bir şekilde düzeltici önlemler alarak hedeflere ulaşma olasılığı artırılır [93]. Projenin, belirlenen hedefler doğrultusunda ilerleyip ilerlemediği bu uygulama alanı ile kontrol edilir. Uygulama alanı, yönetim kategori alanının içerdiği iş planlama ve yönetme yetenek alanı altında bulunur ve 1, 2 ve 3. olgunluk düzeylerinde yer alır [82]. İzleme ve kontrol, aşağıdaki pratikleri içerir [81];

- Proje boyunca elde edilen sonuçları, öncesinde tahmin edilen boyut, çaba, zaman, kaynaklar, beceriler ve bütçe ile karşılaştırarak takip etmek,
- İlgili paydaşların taahhütlerini ve katılımlarını takip etmek,
- Elde edilen sonuçlar, tahmin edilen sonuçlardan önemli ölçüde saparsa düzeltici önlemler almak.

Risk ve Fırsat Yönetimi (Risk & Opportunity Management – RSK): Risk ve fırsat yönetimi uygulama alanı, potansiyel risklerin tanımlanması, bu risklerin meydana gelme olasılıklarının değerlendirilmesi ve bu risklerin yönetilmesi süreçlerini kapsar [97]. Bu uygulama alanında "risk" terimi, hedeflere ulaşmayı olumsuz etkileyebilecek belirsizlikleri ifade eder [93]. Aynı zamanda, projedeki fırsatların belirlenmesi ve bu fırsatların etkin bir şekilde değerlendirilmesi de bu uygulama alanı ile sağlanır. Amacı; özellikle projenin başarısını artırmak ve belirsizlikleri yönetmektir. Risk ve fırsat yönetimi, yönetim kategori alanının içerdiği iş sürekliliğini yönetme yetenek alanı altında bulunur ve 1, 2 ve 3. olgunluk düzeylerinde yer alır. İçerdiği pratikleri aşağıdaki gibidir [98];

- Potansiyel risk ve fırsatları tanımlamak,
- Riskleri ve fırsatları analiz etmek,
- Riskleri ve fırsatları önceliklendirmek,
- Risklere karşı önlemler almak,
- Fırsatları değerlendirmek ve bunlardan yararlanmak,
- Risk yönetim süreçlerini sürekli izlemek,
- Fırsatları sürekli gözden geçirmek.

Organizasyonel Eğitim (Organizational Training – OT): Organizasyonel eğitim uygulama alanı, organizasyonun stratejik eğitim ihtiyaçlarını ve projeler ile destek gruplarında yaygın olan taktiksel eğitim ihtiyaçlarını belirler [77]. Özellikle organizasyonun standart süreçlerini yerine getirmek için gereken becerileri personele kazandırmak amacıyla eğitimler temin ederek personelin beceri, bilgi ve rollerini verimli bir şekilde yerine getirmelerine yardımcı olmayı amaçlar [73]. Eğitimin temel bileşenleri, yönetilen bir eğitim geliştirme programı, belgelenmiş planlar, uygun bilgiye sahip personeller ve eğitim programının etkinliğini ölçmek için kullanılan araçları içerir. Organizasyonel eğitim uygulama alanı, yönetim kategori alanının

içerdiği iş gücünü yönetme yetenek alanı altında bulunur ve 1, 2 ve 3. olgunluk düzeylerinde yer alır. Uygulama alanı aşağıdaki hususları içerir [54], [99];

- Organizasyonun eğitim ihtiyaçlarını belirlemek,
- Personeli eğitmek ve kayıtları tutmak,
- Organizasyonun stratejik ve kısa vadeli eğitim ihtiyaçlarını geliştirmek ve güncel tutmak,
- Projeler ve organizasyon arasında eğitim ihtiyaçlarını ve teslimatı koordine etmek,
- Organizasyonun stratejik ve kısa vadeli eğitim planlarını geliştirmek, güncel tutmak ve takip etmek,
- Organizasyonel eğitim ihtiyaçlarını karşılamak için bir eğitim yeteneği geliştirmek, güncel tutmak ve kullanmak,
- Organizasyonun eğitim programının verimliliğini değerlendirmek,
- Organizasyonel eğitim kayıtlarını tutmak, güncel tutmak ve kullanmak.

Neden Analizi ve Çözümü (Causal Analysis & Resolution – CAR): Kök neden analizi ve çözümü uygulama alanı, olumsuz sonuçların tekrarını önlerken, olumlu sonuçların devam etmesini sağlamak için gereken eylemleri belirler [77]. Temelde, kök neden analizi ve çözümü, tekrar eden hataların önüne geçerek kaliteyi artırmak ve verimliliği yükseltmek için önemli bir araçtır. Bu uygulama alanı, sorunun kaynağını bulmak ve kök nedenlerini ortadan kaldırmak için bir yaklaşım sunar, böylece yeniden çalışma gereksinimini azaltır ve genel olarak kaliteyi iyileştirir [93]. Neden analizi ve çözümü, destekleyici kategori alanının içerdiği uygulama destekleme yetenek alanı altında bulunur ve 1, 2, 3, 4 ve 5. olgunluk düzeylerinde yer alır. Uygulama alanı aşağıdaki pratikleri içerir [73];

- Analiz edilecek olumlu veya olumsuz sonuçların seçilmesi,
- Nedenlerin analiz edilmesi,
- Çözüm önerilerinin uygulanması,
- Uygulanan eylemlerin ne kadar etkili olduğunun değerlendirilmesi,
- Kök neden analizi verilerinin kaydedilmesi.

Karar Analizi ve Çözümü (Decision Analysis & Resolution – DAR): Karar analizi ve çözümü uygulama alanının amacı, karşılaşılan karar seçeneklerini belirli

kriterlere göre değerlendirmektir. Bu uygulama alanı, en iyi çözümü bulmak için alternatifleri sistematik bir şekilde analiz etmek ve karşılaştırmak için resmi bir değerlendirme yöntemi kullanır [64]. Bu süreç, organizasyonel hedeflere en uygun çözümleri sunar. Karar analizi ve çözümü, destekleyici kategori alanının içerdiği uygulama destekleme yetenek alanı altında bulunur ve 1, 2 ve 3. olgunluk düzeylerinde yer alır. Uygulama alanı aşağıdaki adımları içerir [77];

- Karar analizi için yönergelerin oluşturulması adımında hangi kararların resmi değerlendirme tekniği ile değerlendirilmesi gerektiğine karar verilir. Yönergelerde şu noktalar belirtilebilir:
  - Orta ila yüksek etki riski oluşturabilecek sorunlar
  - Konfigürasyon yönetimi altındaki iş ürünlerinde değişiklik yapılması gereken kararlar
  - Belirli bir süreyi aşan gecikmelere yol açabilecek kararlar
  - Projenin hedeflerine ulaşmayı etkileyebilecek kararlar
- Değerlendirme kriterlerinin belirlenmesi,
- Alternatif çözümlerin tanımlanması,
- Değerlendirme yöntemlerinin seçilmesi,
- Alternatif çözümleri değerlendirilmesi,
- Çözümlerin seçilmesi.

Konfigürasyon Yönetimi (Configuration Management – CM): Konfigürasyon yönetimi, bir organizasyonun ürünlerinin veya sistemlerinin bütünlüğünü sağlamak ve sürdürülebilirliğini güvence altına almak için kullanılan önemli bir uygulama alanıdır. Bu uygulama alanı, projelerin her aşamasında yapılan değişikliklerin düzgün bir şekilde izlenmesi, kontrol edilmesi ve belgelenmesini içerir. Uygulama alanının temel amacı, iş ürünlerinin doğru sürümlerinin her zaman mevcut olmasını ve ilgili taraflar tarafından erişilebilir olmasını sağlamaktır [100]. Konfigürasyon yönetimi, destekleyici kategori alanının içerdiği uygulama destekleme yetenek alanı altında bulunur ve basamaklı gösterimde olgunluk düzeyi olarak 1 ve 2. olgunluk düzeylerinde yer alır [101]. Uygulama alanı aşağıdaki hususları içerir [77];

- Proje kapsamında takip edilmesi gereken, yapılandırılabilir öğeleri belirlemek,
- Projede yapılan değişiklikleri kontrol etmek,

- Proje öğelerinin doğru şekilde yapılandırılabilmesi için gerekli spesifikasyonları (talimatlar, teknik detaylar vb.) sağlamak,
- Projenin ilk halini gösteren ve referans alınacak temel belgelerin korunması,
- Tüm paydaşlara, ürünün güncel durumunu paylaşıp bilgi sağlamak.

Yönetim (Governance – GOV): Yönetim uygulama alanı, üst düzey yöneticilere iş süreçlerinin organizasyon için önemli ve uygun şekilde yürütülmesini sağlama konusunda rehberlik eder. Bu uygulama alanı, süreç uygulama maliyetlerini düşürmeyi, hedeflere ulaşma olasılığını artırmayı ve süreçlerin iş başarısına katkı sağlamasını garantilemeyi amaçlar. Organizasyonel hedeflerin gerçekleştirilmesini sağlamak için gerekli liderlik ve yönlendirme süreçlerini kapsar [102]. Yönetim, iyileştirici kategori alanının içerdiği alışkanlığı ve sürekliliği sürdürme yetenek alanı altında bulunur ve 1, 2, 3 ve 4. olgunluk düzeylerinde bulunur. Uygulama alanı aşağıdaki pratikleri içerir [81], [103];

- Üst düzey yönetim tarafından, süreçlerin uygulanması ve iyileştirilmesi için gerekli yönergelerin belirlenmesi ve iletilmesi,
- Süreçlerin etkin bir şekilde uygulanabilmesi için gerekli kaynakların ve eğitimin sağlanması,
- İhtiyaç duyulan bilgilerin belirlenip, veriler kullanılarak süreçlerin iyileştirilmesinin denetlenmesi,
- Bireylerin, organizasyonel talimatlara uymak ve süreçlerin benimsenmesi ile iyileştirilmesi hedeflerinin başarılması konusunda sorumlu tutulması,
- Üst düzey yönetim tarafından, organizasyon genelindeki hedefleri destekleyen ölçümlerin toplanması, analiz edilmesi ve kullanılmasının sağlanması,
- Üst düzey yönetim tarafından, yetkinliklerin ve süreçlerin organizasyonun hedefleriyle uyumlu olmasının temin edilmesi,
- Üst düzey yönetim tarafından, seçilmiş kararların performans ve kalite ile süreç performansı hedeflerine ilişkin istatistiksel ve nicel analizler doğrultusunda alınmasının sağlanması.

Uygulama Altyapısı (Implementation Infrastructure – II): Uygulama altyapısı, organizasyonel süreçlerin oluşturulması, uygulanması, sürdürülmesi ve iyileştirilmesi için gerekli altyapıyı sağlar. Bu altyapı; süreç tanımları, kaynak erişimi,

finansman, eğitim ve nesnel süreç değerlendirmesi gibi unsurları içerir [93]. Organizasyonun, hedef ve amaçlarına verimli ve etkili bir şekilde ulaşma yeteneğini sürekli olarak korumasına yardımcı olur [91]. Uygulama altyapısı uygulama alanı iyileştirici kategori alanının içerdiği alışkanlığı ve sürekliliği sürdürme yetenek alanı altında bulunur ve 1, 2 ve 3. olgunluk düzeylerinde bulunur. Uygulama altyapısı aşağıdaki hususları içerir [81];

- Süreçlerin etkin bir şekilde uygulanmasını ve sürekli iyileştirilmesini destekleyen kaynakların (insan, araçlar, eğitim, finansman vb.) tahsis edilmesi,
- Süreçleri geliştirilmesi, güncel tutulması ve takip edildiklerinin doğrulanması,
- Süreçlerin güncellenerek mevcut durumlarının değerlendirilmesi ve doğru şekilde uygulandıklarından emin olunması,
- Organizasyonel süreçler ve süreç varlıkları kullanılarak işin planlanması, yönetilmesi ve uygulanması,
- Organizasyonel süreçlere uyumun ve etkinliğin değerlendirilmesi,
- Organizasyona süreçle ilgili bilgi veya süreç varlıklarıyla katkıda bulunulması.

Süreç Yönetimi (Process Management – PCM): Süreç yönetimi uygulama alanı, organizasyonun mevcut süreçlerinin güçlü ve zayıf yönlerini analiz ederek bu bilgilere dayalı iyileştirmeleri planlama, uygulama ve yaygınlaştırma yoluyla performansın artırılmasını sağlar. Süreçlerin ve altyapının sürekli iyileştirilmesini yöneterek iş hedeflerine ulaşmayı destekler, en faydalı süreç iyileştirmelerini belirleyip uygular ve bu iyileştirmelerin sürdürülebilir olmasını sağlar [93]. Böylece, süreçler ve altyapı, organizasyonun genel başarısına doğrudan katkıda bulunur. Süreç yönetimi, iyileştirici kategori alanının içerdiği performans iyileştirme yetenek alanı altında bulunur ve 1, 2, 3 ve 4. olgunluk düzeylerinde bulunur. Uygulama alanı aşağıdaki hususları içerir [98];

- Süreçlerin ve süreç varlıklarının iyileştirme fırsatlarını belirlemek,
- Seçilen süreç iyileştirmelerinin uygulaması için planlar geliştirilmek, güncel tutulmak ve bu planları takip etmek,
- İş hedefleriyle izlenebilir süreç iyileştirme hedefleri geliştirmek, güncel tutmak ve kullanmak,
- İş hedeflerine ulaşmada en büyük katkıyı sağlayan süreçleri belirlemek,

- İyileştirme fırsatlarını belirlemek için potansiyel yeni süreçler, teknikler, yöntemler ve araçlar keşfetmek ve değerlendirmek,
- Süreç iyileştirmelerini uygulamak, yaymak ve sürdürülebilir kılmak için destek sağlamak,
- Organizasyonel standart süreçleri ve süreç varlıklarını uygulamak,
- Uygulanan iyileştirmelerin süreç iyileştirme hedeflerine ulaşmadaki etkinliğini değerlendirmek ve raporlamak.

Süreç Varlığı Geliştirme (Process Asset Development – PAD): Süreç varlıkları, süreçleri tanımlamak, uygulamak ve iyileştirmek için temel unsurlar olarak tanımlanan soyut varlıklardır [104]. Süreç varlıklarını edinmenin veya geliştirmenin nihai beklentisi, bir şirketin iş hedeflerine ulaşmasına yardımcı olmaktır [77]. Süreç varlığı geliştirme uygulama alanı, süreçlerin başarılı bir şekilde yürütülmesi için gerekli süreç varlıklarını, dokümantasyonu ve süreç rehberlerini oluşturmayı ve geliştirmeyi içerir. Temel amacı, başarılı sonuçları anlayabilme ve tekrarlayabilme yeteneğini sağlamaktır [93]. Bu uygulama alanı iyileştirici kategori alanının içerdiği performans iyileştirme yetenek alanı altında bulunur ve 1, 2 ve 3. olgunluk düzeylerinde bulunur. Uygulama alanı aşağıdaki hususları içerir [98];

- Çalışmaları yürütmek için gerekli süreç varlıklarının belirlenmesi,
- Süreç varlıklarını geliştirilmesi, satın alınması veya yeniden kullanılması,
- Süreçlerin ve varlıkların erişilebilir hale getirilmesi,
- Süreç varlıklarını oluşturma ve güncelleme stratejisinin geliştirilmesi, güncel tutulması ve uygulanması,
- Organizasyonun süreçlerinin ve süreç varlıklarının yapısını tanımlayan bir süreç mimarisi geliştirilmesi, kaydedilmesi ve güncel tutulması,
- Organizasyonel ölçüm ve analiz standartlarının geliştirilmesi, güncel tutulması ve kullanıma sunulması.

Performans Yönetimi ve Ölçümü (Managing Performance & Measurement – MPM): Performans ve ölçüm yönetimi, iş performansını ölçme ve yönetme süreçlerini kapsayan bir uygulama alanıdır [91]. İş gereksinimleri ve hedefleri doğrultusunda ölçüm ve performans hedeflerini belirlemeyi, bu hedeflere uygun veri toplamayı ve analiz etmeyi içerir. Ölçüm ve performans hedefleri, istatistiksel ve nicel

analiz gerektirmeyen nitel veya nicel hedefleri ifade eder. Bu uygulama alanının temel amacı, iş performansını ölçüm ve analiz yoluyla etkin bir şekilde yönetmek ve iş hedeflerine ulaşılmasını sağlamaktır. Ayrıca, maliyet, zaman planlaması ve kalite performansına odaklanarak yönetim ve iyileştirme çabalarının iş getirilerini en üst düzeye çıkarmayı amaçlar [93]. Performans yönetimi ve ölçümü uygulama alanı iyileştirici kategori alanının içerdiği performans iyileştirme yetenek alanı altında bulunur ve 1, 2, 3, 4 ve 5. olgunluk düzeylerinde bulunur. Uygulama alanı aşağıdaki hususları içerir [81], [98];

- İş gereksinimlerini ve hedeflerini kullanarak ölçüm ve performans hedeflerini belirlemek, belgelemek ve güncel tutmak,
- Organizasyonun ölçümleri için operasyonel tanımları oluşturmak ve güncellemek,
- Operasyonel tanımlara uygun şekilde gerekli ölçüm verilerini toplamak,
- Performans ve ölçüm verilerini operasyonel tanımlara dayalı olarak analiz etmek,
- Ölçümlerle ilgili veri, spesifikasyonlar ve analiz sonuçlarını içeren bir depolama dosyası oluşturmak,
- Hedeflere ulaşmada yaşanan sorunları tespit edip gerekli aksiyonları almak,
- Performans sonuçlarını ve sağlanan iş faydalarını tüm paydaşlar için net şekilde görünür hale getirmek.

3. CMMI-DEV V2.0 KAPSAMINDA SCRUM UYGULAMALARININ DEĞERLENDİRİLMESİ VE EŞLEŞTİRİLMESİ

2023 yılında yayınlanan 17. Çeviklik Durum Raporu'na göre, ankete katılan organizasyonların %71'i yazılım geliştirme yaşam döngülerinde çevik metodolojileri kullanmayı tercih etmektedir. Bu organizasyonların %63'ü Scrum veya Scrum tabanlı yaklaşımları çevik yazılım geliştirme metodolojisi olarak kullanmaktadır [105]. Öte yandan CMMI, dünya çapında en çok tercih edilen performans iyileştirme modellerinden biridir ve son yıllarda kullanımı sürekli olarak artmıştır [2]. Özellikle yazılım geliştirme için en iyi uygulamaları sağlayan CMMI-DEV, organizasyonlar tarafından yaygın bir şekilde benimsenen bir modeldir.

CMMI'ın benimsenmesi, organizasyonlara verimli zaman yönetimi, zamanında teslimat ve hata azaltımı gibi fırsatlar sunsa da çeviklik bağlamlarında bu modelin benimsenmesi zorlayıcı olabilmektedir [106]. Bu da organizasyonların, bu hibrit süreçleri etkili bir şekilde uygulayabilmesi için rehber niteliğinde yönergelere ihtiyaç doğurmuştur. Bu çalışmanın amacı da Scrum'ın pratikleri ile CMMI-DEV V2.0 uygulama alanlarını doğru bir şekilde ilişkilendirmek ve bu sayede ortaya çıkan hibrit yaklaşım ile proje yönetimi ve yazılım geliştirme süreçlerinin iyileştirilmesine katkı sağlamaktır.

CMMI-DEV V2.0 uygulama alanları ile ilişkilendirmek üzere ele alınan Scrum uygulamaları ve kısa tanımları Tablo 6'da görülmektedir [19], [107].

Tablo 6. Scrum uygulamaları ve tanımları

Scrum Uygulamaları	Tanım
Ürün Gereksinim Listesi (Product Backlog)	Ürün Sahibi tarafından yönetilen önceliklendirilmiş gereksinimler listesi
Sprint Gereksinim Listesi (Sprint Backlog)	Sprint için seçilen Ürün Gereksinim maddelerinin alt kümesi
Sprint Planlama (Sprint Planning)	Sprint boyunca tamamlanacak işleri belirlemek ve planlamak için yapılan toplantı
Efor Tahmini (Effort Estimation)	Gereksinim maddeleri için gerekli iş gücünü tahmin etme süreci
Sprint Koşumu (Sprint Execution)	Sprint'in çalışma aşaması, görevlerin tamamlandığı süreç
Günlük Toplantılar (Daily Meetings)	İlerleme ve sorunları güncellemek için yapılan günlük toplantılar
Görev Panosu (Task Board)	Görevlerin ilerlemesini takip etmek için kullanılan görsel araç
Kullanıcı Senaryoları (User Stories)	Kullanıcının bakış açısından gereksinim tanımları

Epikler (Epics)	Daha küçük görevlere bölünen büyük kullanıcı senaryoları
Kod İncelemesi (Code Review)	Kod kalitesini kontrol etme ve hataları bulma süreci
Kod İyileştirmesi (Code Refactoring)	Kodun davranışını değiştirmeden yapısını iyileştirme süreci
Test Planı & Senaryosu & Sonucu (Test Plan & Case & Result)	Testleri planlama, yürütme ve belgelenme süreci
Sürüm Planlama (Release Planning)	Ürün sürümlerini hazırlama ve zamanlama süreci
Sprint Değerlendirmesi (Sprint Review)	Tamamlanan işin gösterildiği ve geri bildirim alındığı toplantı
Retrospektif (Retrospective)	Gelecek Sprint için iyileştirme fikirlerinin tartışıldığı toplantı
Sprint Burndown Grafikleri (Burndown Sprint Charts)	Tamamlanan ve kalan işleri gösteren grafikler
Tamamlanma Tanımı (Definition of Done)	Bir görevin tamamlanmış sayılması için gereken kriterler

3.1. Eşleştirme Yöntemi

Scrum uygulamaları ile CMMI-DEV V2.0 uygulama alanlarının ilişkilendirilmesinde yöntem olarak aşağıdaki adımlar izlenmiştir:

- 1. Adım: Scrum uygulamaları içerik ve kapsam olarak detaylı bir şekilde incelenmiştir [24].
- 2. Adım: CMMI-DEV V2.0 uygulama alanları içerik ve kapsam olarak detaylı bir şekilde incelenmiştir [93].
- 3. Adım: Literatürde CMMI-DEV V2.0 uygulama alanları ile Scrum pratikleri arasındaki ilişkileri belirleme ve ölçme üzerine var olan çalışmalar analiz edilmiştir [1], [2], [108], [109].
- 4. Adım: Her bir Scrum pratiği ve CMMI-DEV V2.0 uygulama alanı içerik ve kapsam olarak örtüşecek şekilde eşleştirilmiştir. Bunun için 1, 2 ve 3. adımlardan elde edilen verilerden faydalanılmıştır.

3.2. Eşleştirme Sonuçları

Yapılan eşleştirmeler sonucu Scrum uygulamaları ile CMMI-DEV V2.0 uygulama alanları arasında oluşan eşleşmeler Tablo 7'de gösterilmektedir.

Tablo 7. CMMI-DEV V2.0 uygulama alanları ile Scrum uygulamaları eşleştirilmesi

Scrum Uygulamaları ve CMMI-DEV V2.0 Uygulama Alanları	Ürün Gereksinim Listesi	Sprint Gereksinim Listesi	Sprint Planlama	Efor Tahmini	Sprint Koşumu	Günlük Toplantılar	Görev Panosu	Kullanıcı Senaryoları	Epikler	Kod İncelemesi	Test Planı, Senaryosu, Sonucu	Kod İyileştirme	Sürüm Planlama	Sprint Değerlendirme	Retrospektif	Sprint Burndown Grafikleri	Tamamlanma Tanımı
RDM	■	■	■	■	■			■	■						■		■
PQA			■	■	■	■				■	■			■	■	■	
PR					■					■		■		■			
VV	■	■			■						■						■
TS			■		■										■		
PI	■	■	■	■					■				■				
EST	■	■	■	■		■	■	■	■		■			■			
PLAN	■	■	■	■			■										■
MC	■	■	■	■	■	■	■							■	■	■	■
RSK			■			■							■			■	
OT						■								■	■		
CAR					■							■			■	■	
DAR	■	■	■										■				■
CM			■			■	■				■		■	■	■	■	■
PCM	■	■			■										■	■	■
PAD			■		■		■					■	■		■	■	
MPM			■		■	■						■			■	■	
SAM																	
II																	
GOV																	

Tablo 7'de görüldüğü üzere, SAM, II ve GOV uygulama alanları haricinde CMMI-DEV V2.0 uygulama alanları Scrum pratikleriyle ilişkilendirilebilmektedir [110]. Bu üç uygulama alanının Scrum ile eşleşmemesinin temel nedeni, Scrum içerisinde doğrudan karşılık gelen bir süreç veya pratiğin bulunmamasıdır. Bununla birlikte, iki modelin birleşimiyle oluşturulan hibrit yaklaşım, mevcut süreçlerin yanı

sıra yönetilmeyen veya eksik kalan bu alanların da CMMI çerçevesinde ele alınmasını, yönetilebilir ve sürdürülebilir hale gelmesini sağlamaktadır.

3.3. İyileştirme Fırsatlarının Belirlenmesi

Yapılan eşleştirmeler sonucu ortaya çıkan hibrit yaklaşımın proje yönetimi ve yazılım geliştirme süreçlerine sunduğu iyileştirme fırsatları aşağıda ayrıntılı olarak incelenmektedir.

Scrum'da gereksinimler Ürün Gereksinim Listesi, Epikler veya Kullanıcı Senaryoları formatında ele alınırken detaylı analiz, izlenebilirlik ve paydaşlarla sürekli iletişim gibi kritik unsurlar için belirgin bir süreç tanımlanmamıştır. Bu noktada **Gereksinim Geliştirme ve Yönetimi (RDM)** uygulama alanı Scrum'da eksik kalan bu unsurların tamamlanmasına katkı sağlayabilir. RDM, paydaşların taleplerini, hedeflerini ve kısıtlarını kapsamlı bir şekilde belirleyerek gereksinimlerin daha eksiksiz ve tutarlı olmasını sağlar [111]. Ayrıca, Scrum'da gereksinimlerin önceliklendirilmesi Ürün Sahibi tarafından iş değeri ve müşteri beklentileri gibi genel faktörlere dayanarak yapılırken, RDM uygulama alanı iş hedefleri, teknik gereksinimler ve riskler gibi daha geniş kriterleri dikkate alarak önceliklendirme sürecine standartlaştırılmış bir yaklaşım getirir.

Süreç Kalite Güvencesi (PQA) uygulama alanı, Scrum'da eksik olan süreç izleme ve denetleme mekanizmalarının tamamlanmasında önemli bir rol oynar. Scrum, ekiplerin kendi süreçlerini iyileştirmelerine odaklanırken, süreçlerin belirli kalite standartlarına uygunluğunu sağlamak için yapılandırılmış bir kalite güvence mekanizması sunmaz. Sprint Değerlendirme ve Retrospektif gibi toplantılar süreç değerlendirmesi için fırsatlar sunsa da, bu değerlendirmeler genellikle ekip içi geri bildirimlere dayanır ve sistematik kalite güvence kontrollerini içermez. PQA, süreçlerin ve çıktıların belirlenen kalite standartlarına uygun olup olmadığını objektif olarak değerlendirme, kalite sorunlarını tespit etme ve çözüm süreçlerini belirleme konularında Scrum ekiplerine rehberlik edebilir.

Scrum ekipleri genellikle kod inceleme süreçlerini Pull Request (PR) gözden geçirme, çiftler halinde kod yazma (pair programming) gibi mühendislik uygulamalarıyla yürütür ve bu süreç Kod İncelemesi olarak tanımlar. CMMI-DEV V2.0'daki **Eşdeğer Gözden Geçirme (PR)** uygulama alanı ise yalnızca kod

incelemeleriyle sınırlı kalmayıp, tasarım dokümanları, teknik spesifikasyonlar ve test senaryoları gibi proje çıktılarının düzenli bir şekilde değerlendirilmesini kapsar. Bu yapı, süreçlerin ve ürünlerin belirlenen standartlara uygunluğunu sağlamakla birlikte, ekip içi iş birliğini güçlendirerek erken hata tespiti ve kalite iyileştirmesine katkıda bulunur. Dolayısıyla, Scrum içinde zaten yaygın olarak kullanılan kod inceleme süreçleriyle birlikte, PR uygulama alanı daha geniş bir gözden geçirme pratiği sunarak Scrum ekiplerinin kalite güvence süreçlerini destekleyici bir rol oynayabilir.

Scrum, esnek ve iteratif bir yapıya sahip olup müşteri geri bildirimine hızlı yanıt verme avantajı sunsa da doğrulama ve geçerleme süreçleri açısından belirgin eksiklikler barındırmaktadır. Scrum'da kalite güvencesi, Spring Retrospektif, Sprint Değerlendirme ve Test planı, senaryosu ve sonuçları ile ele alınır. Ancak, sistematik doğrulama ve geçerleme faaliyetleri, özellikle tanımlı prosedürler, test kriterleri ve hedef ortamda çalışabilirliğin kontrol edilmesi gibi unsurlar açısından yetersizdir. CMMI-DEV 2.0'nin **Doğrulama ve Geçerleme (VV)** uygulama alanı, Scrum'ın bu eksik yönlerini tamamlayarak süreçlerin daha güvenilir ve kontrollü bir yapıya kavuşmasını sağlayabilir. Öncelikle, VV uygulama alanı, gereksinimlerin tam anlamıyla karşılandığını değerlendirmek için yapılandırılmış bir doğrulama süreci sunarak, Scrum'ın Tamamlanma Tanımı mekanizmasına daha kesin kriterler eklenmesini mümkün kılar. Ayrıca, Scrum'da iteratif geliştirme süreci prototipler ve kullanıcı geri bildirimleri ile desteklenmesine rağmen, üretilen çözümün hedef ortamda beklendiği gibi çalıştığını gösteren kapsamlı geçerleme faaliyetleri eksiktir. VV uygulama alanı, bu süreci daha yapılandırılmış hale getirerek, doğrulama ve geçerleme için kriterlerin belirlenmesi, uygulanması ve sonuçların sistematik olarak raporlanmasını sağlar.

Teknik Çözüm (TS) uygulama alanı, Scrum'da eksik olan yapılandırılmış teknik karar alma ve çözüm geliştirme süreçlerinin tamamlanmasına katkı sağlar. Scrum, ürün geliştirme sürecinde esneklik sağlarken, teknik gereksinimlerin belirlenmesi, alternatif çözümlerin değerlendirilmesi ve mühendislik kararlarının planlı ve yöntemsel bir şekilde alınması için belirli bir çerçeve sunmaz. Teknik Çözüm uygulama alanı, ekiplerin teknik gereksinimlere uygun çözümler geliştirmesine, mevcut ve yeni teknolojiler arasından en uygun olanı seçmesine, bileşen tasarım süreçlerini yönetmesine ve teknik dokümantasyon oluşturmasına rehberlik eder. Bu

sayede, Scrum ekipleri yalnızca iteratif geliştirme döngülerine odaklanmak yerine, teknik çözümlerin kalite, maliyet, sürdürülebilirlik ve performans gibi faktörlere göre optimize edilmesini sağlayabilir. Böylece, ürün geliştirme süreci daha yapılandırılmış hale gelir ve teknik kararların rastlantısallığı azaltılarak uzun vadeli sürdürülebilir çözümler elde edilebilir.

Ürün Entegrasyonu (PI), Scrum çerçevesinde açık bir şekilde tanımlanmayan, ancak yazılım geliştirme süreçlerinde kritik bir rol oynayan entegrasyon faaliyetlerinin sistematik bir şekilde yürütülmesini sağlamaktadır. Scrum, iteratif ve artımlı bir geliştirme yaklaşımı benimsediğinden, entegrasyon süreci genellikle sprintler boyunca kademeli olarak gerçekleştirilir. Ancak, entegrasyon stratejisinin belirlenmesi, entegrasyon ortamının hazırlanması, bileşenler arası uyumluluğun sağlanması ve entegrasyon sonrası doğrulama süreçleri gibi hususlar Scrum içinde doğrudan ele alınmamaktadır. PI uygulama alanı, Scrum süreçlerini tamamlayıcı nitelikte olup, Sürekli Entegrasyon (Continuous Integration – CI) ve Sürekli Teslimat (Continuous Deployment – CD) gibi mühendislik uygulamalarının etkin şekilde yönetilmesini desteklemektedir. Ayrıca, entegrasyon sırasında ortaya çıkabilecek arayüz ve bileşen bağlantı uyumsuzluklarının erken aşamada tespit edilmesini sağlayarak, teknik borcun birikmesini önlemekte ve ürün kalitesini artırmaktadır. PI'nin Scrum ile bütünleştirilmesi, entegrasyon faaliyetlerinin daha yapılandırılmış hale gelmesini sağlayarak, geliştirme sürecinin daha öngörülebilir ve sürdürülebilir olmasına katkıda bulunmaktadır.

Scrum'da tahminleme genellikle story point**,** velocity (sprint içerisinde tamamlanan ortalama iş veya story point sayısı [112]) ve burndown grafikleri gibi tekniklere dayansa da, bu yaklaşımlar çoğu zaman standartlaşmış bir tahmin yöntemi ve dokümantasyon eksikliği nedeniyle organizasyonel düzeyde tutarsızlıklara yol açabilir. **Tahmin (EST)** uygulama alanı, Scrum süreçlerinde bütçe, zaman ve kaynak tahminlerinin sistematik hale getirilmesini sağlayarak belirsizliği azaltır ve proje yönetim süreçlerini güçlendirir [113]. EST uygulama alanı, proje boyutlandırma, maliyet analizi ve efor tahminleme süreçlerine yönelik daha yapılandırılmış bir çerçeve sunarak, tahminlerin doğruluğunu artırır ve güncellenebilir bir tahmin yönetim modeli oluşturulmasını destekler [114]. Böylece, Scrum süreçlerinde tahminleme

sürecinin tekrarlanabilir, ölçülebilir ve izlenebilir hale getirilmesine katkı sağlayarak planlama süreçlerinin daha güvenilir ve öngörülebilir olmasını mümkün kılar [91].

Scrum, iş planlamasında kısa vadeli ve takım odaklı bir yaklaşım benimserken, bütçe, zaman çizelgesi ve kaynak planlaması gibi uzun vadeli stratejik yönetim konularında doğrudan bir rehberlik sunmaz. **Planlama (PLAN)** uygulama alanı, proje faaliyetlerini başarıyla gerçekleştirmek için detaylı bir bütçe, zaman çizelgesi, kaynak planlaması ve risk yönetimi sunar.

Scrum, sprint bazında iş yükünü ve kaynakları belirlerken ve Sprint Gereksinim Listesi'ni oluştururken, genellikle yalnızca işin yapılabilirliğine dayalı tahminler yapar ve Hikâye Puanı kullanarak iş yükünü belirlerken, bu uygulama alanı, riskleri önceden tanımlama ve yönetme, projeye dahil olan tüm paydaşları belirleme ve uzun vadeli planlar oluşturma süreçlerini güçlendirir [115]. Ayrıca, PLAN uygulama alanı, sadece insan kaynağı değil, proje için gerekli diğer kaynakları – altyapı, ekipman ve eğitim gibi – da dikkate alarak kapsamlı bir kaynak planlaması sağlar. Ayrıca, Scrum'daki Sprint Planlama ve Günlük Toplantılar gibi kısa vadeli süreçlere ek olarak, proje planlarının tutarlılığını sağlamak için tüm paydaşlarla etkileşimi yönetir. Böylece, PLAN uygulama alanı, Scrum'un çevik yapısını bozmadan proje yönetimini daha kapsamlı bir şekilde ele alır.

Scrum, süreç ve performans takibini Günlük Toplantılar, Görev Panoları ve her sprint sonunda gerçekleştirilen Sprint Değerlendirme ve Retrospektif toplantıları ile sağlar. Bu mekanizmalar, ekibin ilerlemeyi düzenli olarak değerlendirmesine ve gerekli iyileştirmeleri yapmasına olanak tanır. **İzleme ve Kontrol (MC)** uygulama alanı ise proje ilerleyişini sürekli takip etmeyi, belirlenen hedeflerden sapmaların tespit edilmesini ve gerektiğinde düzeltici önlemler alınmasını içerir. Scrum'daki burndown chart'lar, kalan iş yükünü ve ilerlemeyi görselleştirerek ekibin zaman ve kaynak yönetimini daha etkin yapmasını destekler. Bununla birlikte, MC uygulama alanı, proje planının yalnızca anlık değil, uzun vadeli hedeflere uygunluğunu da değerlendirme konusunda ek katkı sağlayabilir. Böylece, kapsam sapmalarını, kaynak kullanımını ve zaman aşımını daha etkin yönetmek mümkün hale gelir.

Scrum, risk yönetimi konusunda doğrudan tanımlanmış bir süreç sunmamakla birlikte, sprint planlamaları, günlük Scrum toplantıları ve retrospektifler aracılığıyla

risklerin ele alınmasına imkân tanır. Ancak bu yaklaşımlar genellikle mevcut sorunlara odaklanır ve risklerin sistemli bir şekilde öngörülmesini ve yönetilmesini her zaman garanti etmez. **Risk ve Fırsat Yönetimi (RSK)** uygulama alanı, Scrum'daki bu eksikliği gidererek risklerin önceden belirlenmesini, analiz edilmesini ve önceliklendirilmesini sağlar. Özellikle sprint öncesinde risklerin tanımlanması ve olası etkilerinin değerlendirilmesi, ekiplerin karşılaşabileceği belirsizliklere yönelik daha bilinçli kararlar almasına katkıda bulunur. Ayrıca, fırsatların belirlenmesi ve bunlardan en iyi şekilde yararlanılması, yalnızca riskleri azaltmakla kalmayıp projenin değer üretme potansiyelini de artırabilir.

Organizasyonel Eğitim (OT) uygulama alanı, Scrum'un organizasyon genelindeki eğitim süreçlerini kapsamadığı noktada önemli bir tamamlayıcı işlev görmektedir. Scrum, bireysel ve takım seviyesinde öğrenmeyi teşvik etse de organizasyon çapında düzenli bir eğitim yönetim süreci sunmaz. Çevik yaklaşımlarda eğitim süreçlerinin çoğunlukla proje bazlı ve kısa vadelidir, ancak organizasyonel yetkinliklerin sürdürülebilir bir şekilde geliştirilmesi için daha geniş kapsamlı eğitim stratejilerine ihtiyaç duyulmaktadır [99]. Bu bağlamda, OT uygulama alanı, Scrum'da eksik kalan organizasyonel eğitim planlaması, yetkinlik yönetimi ve süreç iyileştirmeye yönelik bilgi aktarımını destekler. Özellikle organizasyonun eğitim ihtiyaçlarını belirleme, eğitim programlarını yönetme ve çalışanların uzun vadeli yetkinlik gelişimini sağlama konularında önemli bir rol oynar. Ayrıca, eğitim süreçlerinin etkililiğini değerlendirme ve güncelleme mekanizmaları sağlayarak, Scrum ekiplerinin bilgi ve becerilerinin sürekli gelişimine katkıda bulunur.

Neden Analizi ve Çözümü (CAR) uygulama alanı, Scrum'daki Retrospektif pratiğini daha derinlemesine bir analiz çerçevesiyle destekleyerek, tekrarlayan hataların önlenmesine ve süreç iyileştirmelerinin daha veriye dayalı hale gelmesine katkı sağlar [116]. Scrum'da retrospektifler genellikle ekip üyelerinin deneyimlerine ve öznel gözlemlerine dayanırken, CAR süreci hataların, gecikmelerin ve beklenmeyen sonuçların kök nedenlerini belirlemek için nicel ve nitel analiz tekniklerini kullanır. Örneğin, Sprint'lerde sürekli olarak tahminlerin tutmaması veya belirli bir tür hatanın tekrar etmesi durumunda, CAR uygulamaları istatistiksel analiz, hata kümeleme ve neden-sonuç diyagramları gibi tekniklerle bu sapmaların kaynağını tespit etmeye yardımcı olabilir. Bunun yanı sıra, uygulanan iyileştirme aksiyonlarının

etkinliği retrospektiflerde genellikle subjektif değerlendirmelere dayanırken, CAR, belirlenen önlemlerin gerçekten sorunu çözüp çözmediğini ölçmek için nicel metrikler ve takip mekanizmaları önerir.

Scrum, hızlı ve esnek bir çalışma modeli sunmasına rağmen, karar alma süreçlerinde belirli değerlendirme yöntemlerinin eksikliği hissedilebilir. **Karar Analizi ve Çözümü (DAR)**, kararların algılara ve kişisel yargılara değil, tanımlanmış kriterlere dayalı olarak objektif bir şekilde alınmasını sağlar [116]. Teknik seçimler, backlog önceliklendirme, sürekli teslimat stratejileri ve risk yönetimi gibi konularda daha bilinçli kararlar alınmasına yardımcı olur. Alternatif çözümleri belirli kriterlere göre değerlendirerek sezgisel yaklaşımların ötesine geçmeyi sağlayan DAR, özellikle mimari kararlar, bağımlılık yönetimi ve entegrasyon süreçlerinde etkili olabilir. Ayrıca, Sprint Retrospective toplantılarında süreç iyileştirmeleri geçmiş veriler üzerinden değerlendirildiğinde daha verimli sonuçlara ulaşılabilir. DAR uygulamalarının entegrasyonu, Scrum'ın karar alma mekanizmasını güçlendirerek proje başarısını artırabilir.

Konfigürasyon Yönetimi (CM), Scrum'ın sunduğu esneklik ve çevikliğin yanında, değişiklik yönetimi, sürüm kontrolü ve dokümantasyon gibi eksik kalan yönleri tamamlayarak sürecin daha sürdürülebilir hale gelmesini sağlar. Scrum, iteratif ve sürekli gelişime dayalı bir model sunduğu için proje öğelerinin versiyonlanması, değişikliklerin izlenmesi ve yapılandırılabilir öğelerin kontrolü konusunda belirli bir standart sunmaz. CM, Backlog yönetiminde yapılan değişikliklerin izlenmesini sağlarken, kod ve ürün sürümlerinin kontrol altına alınmasını, yapılan değişikliklerin etkilerinin değerlendirilmesini ve tüm paydaşlara şeffaf bir şekilde iletilmesini mümkün kılar. Ayrıca, test süreçlerini destekleyerek, her iterasyonun çıktılarını belirli bir yapı içinde tutarak kalite güvencesini artırır. Bunun yanı sıra, Scrum ekiplerinde bazen geri planda kalan dokümantasyon sürecini sistematik hale getirerek organizasyonel hafızayı güçlendirir. Sonuç olarak, CM'nin Scrum ile entegrasyonu, teknik borcun azaltılmasına, ürün güvenilirliğinin artırılmasına ve proje süreçlerinin daha izlenebilir ve güvenli hale gelmesine önemli katkılar sağlar.

Süreç Yönetimi (PCM) uygulama alanı, Scrum çerçevesinde mevcut süreçlerin sistematik bir şekilde analiz edilerek iyileştirilmesine ve süreç yönetimi açısından eksik kalan yönlerin tamamlanmasına katkı sağlayabilir. Scrum, iteratif ve

artımlı geliştirme modeliyle sürekli iyileştirmeyi teşvik etse de, süreçlerin organizasyonel düzeyde kapsamlı bir şekilde yönetilmesine yönelik doğrudan bir yapı sunmamaktadır. PCM, süreçlerin güçlü ve zayıf yönlerini değerlendirerek iyileştirme stratejilerinin belirlenmesini, uygulanmasını ve sürdürülebilir kılınmasını sağlar. Bu bağlamda, Scrum'ın sunduğu Sprint Retrospektif ve Sprint Değerlendirme gibi süreç iyileştirme mekanizmaları, PCM ile daha sistematik ve uzun vadeli bir iyileştirme döngüsüne entegre edilebilir. Ayrıca, süreçlerin organizasyonel hedeflerle uyumlu hale getirilmesi, Gereksinim Listesi İyileştirme ve Tamamlanma Tanımı gibi pratiklerin etkinliğinin artırılması açısından PCM önemli bir tamamlayıcı unsur olarak değerlendirilebilir. Böylece, süreç yönetiminin sürekli izlenmesi ve optimize edilmesi sağlanarak, Scrum'ın operasyonel etkinliği ve organizasyon genelindeki uygulanabilirliği artırılabilir.

Scrum, ekip düzeyinde çevik uygulamaları etkin bir şekilde desteklese de, süreç varlıklarının standardizasyonu, organizasyonel düzeyde erişilebilirliği ve sürekli iyileştirilmesi konularında sınırlıdır. **Süreç Varlığı Geliştirme (PAD)** uygulama alanı, süreçlerin yürütülmesini destekleyen dokümantasyon, rehberler ve süreç mimarisini oluşturmayı, sürdürmeyi ve güncellemeyi hedefler. Bu kapsamda, organizasyon genelinde Scrum uygulamalarının tutarlılığını artırmak, süreçlerin tekrar edilebilirliğini sağlamak ve ekipler arası bilgi paylaşımını güçlendirmek amacıyla süreç varlıklarının merkezi bir yapı altında yönetilmesini sağlar. Böylece, Scrum ekipleri arasında bilgi paylaşımı ve süreç tutarlılığı artırılarak kurumsal çeviklik güçlendirilebilir. Ayrıca, organizasyonel ölçüm ve analiz standartlarını belirleyerek, Scrum çerçevesinde genellikle eksik kalan süreç metriklerinin tanımlanmasını ve performans değerlendirmelerinin sistematik hale getirilmesini mümkün kılar.

Scrum, performans ölçümüne yönelik Retrospektif çıktıları, Burndown Grafikleri ve Sprint Değerlendirmeleri gibi mekanizmalara sahip olsa da bu ölçümler genellikle kısa vadeli değerlendirmelere odaklanır ve proje hedefleri ile stratejik performans göstergeleri açısından sınırlı kalabilir. **Performans Yönetimi ve Ölçümü (MPM)** uygulama alanı, iş gereksinimleri doğrultusunda ölçüm hedeflerinin belirlenmesini, veri toplama süreçlerinin standartlaştırılmasını ve analiz sonuçlarının iş hedefleriyle ilişkilendirilmesini sağlar. Bu bağlamda, Scrum'da eksik olan uzun vadeli performans izleme, süreç iyileştirme kararlarının veri odaklı alınması ve

organizasyonel hedeflerle uyumlu performans değerlendirme süreçlerinin geliştirilmesi açısından kritik bir rol oynar. Ayrıca, MPM'nin sağladığı operasyonel tanımlar ve istatistiksel analiz yöntemleri, Scrum ekiplerinin yalnızca iterasyon bazında değil, proje geneli ve organizasyon seviyesinde performans eğilimlerini değerlendirmesine olanak tanır.

Tedarikçi Sözleşmesi Yönetimi (SAM) uygulama alanı, Scrum'da doğrudan ele alınmayan tedarikçi yönetimi süreçlerini tamamlayarak, dış kaynak kullanımına dayalı geliştirme faaliyetlerinin sistematik bir şekilde yürütülmesini sağlar. Scrum, çevik metodolojilere odaklandığından, tedarikçilerin seçimi, sözleşmelerin yönetimi ve dış kaynaklı bileşenlerin entegrasyonu gibi konulara yönelik resmi bir pratik sunmaz [115]. SAM, bu eksikliği gidererek edinim türünün belirlenmesi, tedarikçi seçim kriterlerinin oluşturulması, tedarik süreçlerinin izlenmesi, teslim edilen ürünlerin kabulü ve entegrasyonu gibi kritik süreçleri tanımlar. Bu bağlamda, SAM'ın uygulanması, dış tedarikçilerin ürün ve hizmetlerinin Sprint süreçlerine entegrasyonunu kolaylaştırırken, kabul kriterlerinin netleştirilmesi, kalite güvence mekanizmalarının güçlendirilmesi ve sözleşmeye dayalı performans takibinin yapılmasını mümkün kılar. Böylece, dış tedarik süreçlerinin çevik geliştirme döngüleriyle uyumlu hale getirilmesi sağlanarak, proje yönetimi ve ürün teslim süreçlerinde bütüncül bir yapı oluşturulur [117].

Scrum, çevik ve iteratif bir yazılım geliştirme yaklaşımı olarak süreçlerin dinamik bir şekilde yönetilmesini teşvik etmekle birlikte, organizasyonel altyapının oluşturulması, sürdürülmesi ve iyileştirilmesi konusunda kapsamlı bir çerçeve sunmamaktadır. Bu sebeple **Uygulama Altyapısı (II)** uygulama alanı ile ilişkilendirilebilecek herhangi bir tanımlı sürece sahip değildir. Uygulama Altyapısı, organizasyonel süreçlerin etkin bir şekilde uygulanmasını ve sürekliliğini destekleyen kaynakların (insan gücü, teknolojik araçlar, eğitim, finansman vb.) tahsis edilmesini sağlayarak bu eksikliği gidermektedir. Bu uygulama alanı, sistem bakım süreçleri, teknik destek mekanizmaları ve operasyonel iyileştirme faaliyetleri gibi konuları içermekte olup, Scrum ekiplerinin uzun vadeli sürdürülebilirlik ve verimlilik açısından ihtiyaç duyduğu altyapıyı sağlamaktadır. Ayrıca, süreçlerin sürekli olarak değerlendirilmesi, güncellenmesi ve geliştirilmesine yönelik mekanizmalar sunarak

Scrum çerçevesinin ölçeklenebilirliğini artırmakta ve ekiplerin operasyonel süreçlerini daha sistematik bir şekilde yönetmelerine katkı sağlamaktadır**.**

Scrum'da, yönetim süreçleri doğrudan ele alınmadığından, organizasyonel hedeflerin belirlenmesi, süreçlerin standardizasyonu ve üst yönetim desteği gibi kritik unsurlar eksik kalmaktadır. Bu durum, özellikle büyük ölçekli organizasyonlarda kurumsal tutarsızlıklara ve verimsizliklere yol açabilir. **Yönetim (GOV)** uygulama alanı, üst yönetimin süreçlerin etkinliğini artırmak için yönlendirici kararlar almasını, gerekli kaynakların ve eğitimlerin sağlanmasını ve süreçlerin sürekli iyileştirilmesini zorunlu kılar. Bununla birlikte, Scrum'daki tahminleme ve süreç takibi teknikleri (örneğin, velocity, burndown grafikleri) ekip içi düzeyde işlevsel olsa da GOV uygulama alanının sunduğu veri odaklı karar alma mekanizmaları ve istatistiksel analiz yöntemleri, organizasyonel seviyede daha tutarlı ve güvenilir süreç yönetimi sağlar [103]. Ayrıca, kurumsal tutarsızlık riskini azaltarak hem çevik ekiplerin hem de üst yönetimin ortak bir yönetişim anlayışı çerçevesinde çalışmasını mümkün kılar.

3.4. Anket Uygulaması

Bu bölümde, çalışmanın saha araştırması kapsamında gerçekleştirilen anket uygulamasına ait bulgulara yer verilmektedir. Anket, yazılım geliştirme süreçlerinde Scrum metodolojisini etkin bir biçimde uygulayan kuruluşların süreç yönetimi, ölçümleme, iyileştirme ve standardizasyon konularındaki durumlarını ortaya koymak ve CMMI-DEV V2.0 modeline yönelik farkındalık düzeylerini ve bu modele duyulan gereksinimi değerlendirmek amacıyla tasarlanmıştır.

Toplam 24 sorudan oluşan ankette, ilk 20 soru doğrudan CMMI-DEV V2.0'ın uygulama alanlarıyla ilişkilendirilmiştir. Bu sorular aracılığıyla katılımcı kuruluşların süreç olgunluğu, performans yönetimi, dokümantasyon düzeyi ve kurumsal stratejileri analiz edilmiştir. Kalan dört soru ise yazılım geliştirme süreçlerinin yazılı olarak tanımlanma ve standardizasyon düzeyine, organizasyonel ölçekte yürütülen süreç iyileştirme faaliyetlerinin varlığına, Scrum'ın süreç olgunluğu üzerindeki yeterliliğine ilişkin algıya ve CMMI-DEV V2.0 modeline dair bilgi seviyelerine odaklanmaktadır.

Anketin hedef kitlesini, organizasyonlarında Scrum uygulamalarına doğrudan dâhil olan ve süreç olgunluğunu değerlendirme yetkinliğine sahip profesyoneller oluşturmaktadır. Bu kapsamda; yazılım geliştiriciler, takım liderleri, proje yöneticileri

ve scrum master gibi rollerde görev alan katılımcılara ulaşılmıştır. Araştırma, 12 farklı yazılım geliştirme organizasyonundan elde edilen veriler doğrultusunda gerçekleştirilmiştir. Katılımcı kuruluşlar, sektörel çeşitlilik ve organizasyonel ölçek açısından farklılık göstermekte olup; bu çeşitlilik yoluyla ulaşılan bulguların daha genellenebilir ve kapsamlı bir görünüm sunması amaçlanmıştır.

Veri toplama süreci, çevrim içi anket yöntemi ile yürütülmüş ve Microsoft Forms aracı kullanılmıştır. Anket soruları kapalı uçlu olarak hazırlanmış ve katılımcıların mevcut uygulamalara ilişkin algı ve deneyimlerini yansıtacak biçimde dereceli seçenekler sunulmuştur. Cevap seçenekleri, genellikle "tam olarak uygulanıyor" ifadesinden başlayarak "hiç uygulanmıyor" veya "bilgi sahibi değilim" düzeyine kadar uzanan ölçekler halinde düzenlenmiştir. Bu yapı, uygulamaların kurumsal düzeydeki yerleşikliğini ve katılımcıların bilinç düzeyini ölçmeye olanak sağlamıştır. Ayrıca, bazı sorularda yer alan "emin değilim" gibi nötr ifadeler sayesinde, süreçlerin organizasyon içindeki görünürlüğü ve farkındalık düzeyi hakkında daha derinlemesine analizler yapılabilmiştir.

Ankete katılan 12 kuruluş, Türkiye'de faaliyet gösteren ve yazılım geliştirme süreçlerinde etkin rol oynayan firmalar ile kamu kurumlarından oluşmaktadır. Bu kuruluşlar; perakende, finans, bankacılık, yazılım geliştirme, savunma sanayi, eğitim teknolojileri ve kamu Ar-Ge gibi geniş bir sektörel yelpazeyi temsil etmektedir. Kuruluş yılları 1938 ile 2022 arasında değişen bu şirketler hakkında temel bilgiler Tablo 8'de sunulmuştur.

Katılımcı seçiminde temel ölçütler; yazılım geliştirme süreçlerinin kurumsal düzeyde yürütülüyor olması ve Scrum uygulamalarının organizasyon genelinde benimsenmiş olmasıdır. Araştırmada anonimlik ilkesine bağlı kalınarak, her bir katılımcı kuruluşa alfabetik harf kodları (A–L) atanmış ve analizler bu kodlar üzerinden gerçekleştirilmiştir. Bu yaklaşım hem katılımcıların kimliklerinin gizliliğini korumakta hem de değerlendirmelerin nesnelliğini desteklemektedir.

Tablo 8. Katılımcı kurumlara ait temel bilgiler

Kurum Kodu	Kuruluş Yılı	Faaliyet Sektörü	Çalışan Sayısı	Ölçek
A	1938	Bankacılık	10.000 +	Büyük
B	1954	Perakende ve Bilgi Teknolojileri	10.000 +	Büyük
C	1968	Kamu Araştırma ve Geliştirme (Ar-Ge)	2.000 +	Büyük
D	1982	Savunma Sanayi ve Bilişim	1000 – 5000	Büyük
E	1984	Kurumsal Yazılım Geliştirme	1000 – 5000	Büyük
F	1997	Bankacılık	10.000 +	Büyük
G	1997	Yazılım Eğitimi ve BT Danışmanlığı	1000 – 5000	Büyük
H	2005	Finansal Teknolojiler ve Bankacılık Yazılımları	600 – 700	Orta
I	2017	Büyük Veri, Analitik ve Yazılım Danışmanlığı	1000 – 5000	Büyük
J	2021	Havacılık ve Bilişim Teknolojileri	1000 – 5000	Büyük
K	2021	Elektronik Para ve Ödeme Sistemleri	500 – 1000	Orta
L	2022	Dijital Platform ve Finansal Teknolojiler	50 – 200	Küçük

Anket soruları Tablo 9'da gösterilmektedir. Her sorunun cevap seçenekleri ile beraber verilen yanıtlarla oluşturulmuş grafikleri ve bu grafiklerin analizi bir sonraki alt başlık altında detaylı bir şekilde ele alınmıştır.

Tablo 9. Anket soruları

Soru Numarası	Soru
1	Ürün gereksinimlerinin izlenebilirliği organizasyonunuzda nasıl sağlanıyor?
2	Organizasyonunuzda kalite güvence faaliyetleri hangi düzeyde yürütülmektedir?
3	Ekip içinde yapılan ürün veya çıktı gözden geçirme (peer review) faaliyetleri ne düzeyde yürütülmektedir?
4	Ürünlerin doğrulama (verification) ve geçerleme (validation) süreçleri organizasyonunuzda nasıl uygulanmaktadır?
5	Teknik gereksinimlerin karşılanması için gerekli olan çözümlerin geliştirilmesi süreci organizasyonunuzda nasıl yürütülmektedir?
6	Scrum süreçlerinizde ürün bileşenlerinin entegrasyonu nasıl gerçekleştirilmektedir?

7	Scrum ekipleriniz dış tedarikçilerle çalıştığında, bu süreçler nasıl yönetilmektedir?
8	Scrum ekiplerinizde, proje süresi, maliyet, efor ve kaynaklar gibi unsurlar için tahmin süreci nasıl yürütülmektedir?
9	Scrum ekiplerinizde, proje faaliyetlerinin bütçesi, takvimi, görev dağılımı, paydaş katılımı ve gerekli becerilere göre nasıl bir planlama süreci yürütülmektedir?
10	Projenizde ilerleme, tahminlerle karşılaştırılarak nasıl izleniyor ve sapmalar tespit edildiğinde hangi önlemler alınmaktadır?
11	Projenizde potansiyel riskler ve fırsatlar nasıl yönetiliyor? Bu süreçte hangi uygulamaları dikkate alıyorsunuz?
12	Organizasyonunuzda eğitim ihtiyaçları nasıl belirleniyor ve bu eğitimlerin etkinliği nasıl değerlendirilip izleniyor?
13	Organizasyonunuzda kök neden analizi ve çözümü süreçleri nasıl yürütülmektedir?
14	Karar analizi ve çözümü süreçleri organizasyonunuzda nasıl uygulanmaktadır?
15	Projelerde hangi dosya, belge veya sistem bileşeninin hangi sürümde olduğunu bilmek, değişiklikleri takip edebilmek ve güncel bilgileri tüm ekip üyeleriyle paylaşmak adına bir takip ve kontrol sistemi uygulanıyor mu?
16	Üst düzey yöneticiler (örneğin genel müdür, direktör vb.), projelerin doğru yürütülmesi ve süreçlerin iyileştirilmesi için ekiplere net yönlendirme ve destek sağlıyor mu?
17	İş süreçlerini uygularken veya iyileştirirken ihtiyaç duyduğunuz destek altyapısına (örneğin rehber dokümanlar, eğitim materyalleri, yazılım araçları, uzman desteği, zaman, finansman vb.) ne ölçüde erişebiliyor ve bu altyapının işinizi ne kadar kolaylaştırdığını düşünüyorsunuz?
18	Çalıştığınız projelerde ya da birimde, süreçlerin nasıl daha iyi hale getirileceğine dair fikirlerin toplandığını, uygulandığını ve sonuçlarının değerlendirildiğini düşünüyor musunuz?
19	Çalışmalarınızı yürütürken size yol gösteren standart belgeler (örneğin rehberler, kontrol listeleri, şablonlar, süreç açıklamaları vb.) yeterli düzeyde mevcut ve erişilebilir mi?
20	Yaptığınız çalışmaların performansı (örneğin kalite, zamanlama, maliyet gibi) düzenli olarak ölçülüyor, bu ölçümler analiz edilerek sizinle ve/veya ekiple paylaşılıyor mu?
21	Organizasyonunuzda yazılım geliştirme süreçleri (analiz, tasarım, kodlama, test, teslimat vb.) ne ölçüde standartlaştırılmış ve yazılı olarak belgelenmiştir?
22	Scrum uygulamalarının dışında, organizasyonel düzeyde süreç iyileştirme çalışmaları yürütülüyor mu?
23	Sizce Scrum uygulamaları, organizasyonel süreçlerin yeterince olgunlaşması ve standartlaştırılması için tek başına yeterli midir?
24	CMMI-DEV V2.0 (Capability Maturity Model Integration for Development) hakkında bilginiz var mı?

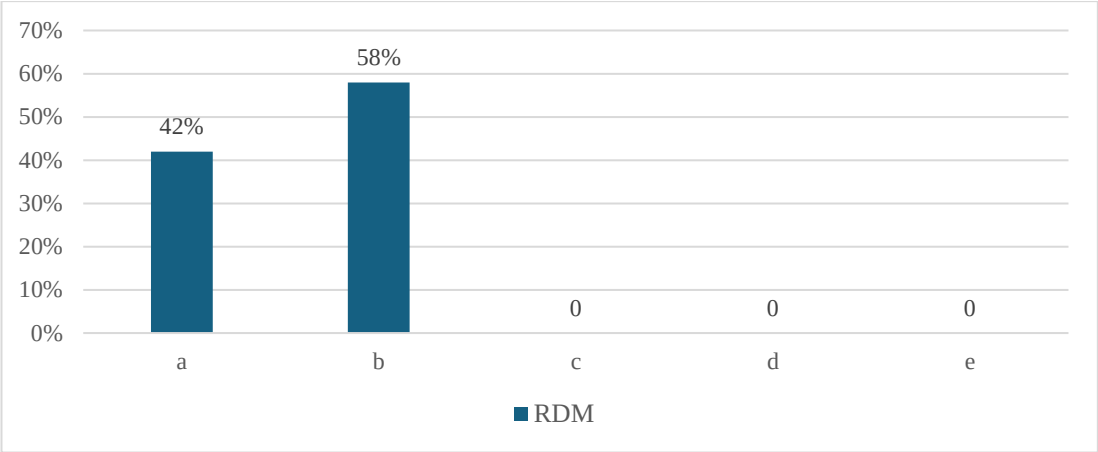
3.4.1. Anket Çalışmasının Sonuçlarının Değerlendirilmesi

Bu başlık altında, anket kapsamında yer alan 24 soruya verilen yanıtlar, her bir soruya özel olarak hazırlanmış grafiklerle birlikte analiz edilmektedir. Her bir soru, ilgili olduğu uygulama alanı ve değerlendirme boyutu çerçevesinde ele alınmış; katılımcıların yanıtları üzerinden mevcut durumun güçlü yönleri ile iyileştirme alanları belirlenmiştir. Grafiksel gösterimler, yanıt dağılımlarını daha anlaşılır ve görsel olarak takip edilebilir hale getirirken, yapılan yorumlar bu dağılımlar doğrultusunda şekillendirilmiştir. Aşağıda, her bir soruya ilişkin bulgular ve değerlendirmeler sırasıyla sunulmaktadır.

"1. Soru: Ürün gereksinimlerinin izlenebilirliği organizasyonunuzda nasıl sağlanıyor?

a) Kullanıcı hikayeleri, sistematik olarak teknik gereksinimlere ve çıktılara iki yönlü olarak izlenebilir şekilde bağlanmakta; gereksinimler açık, analiz edilmiş ve yönetilmektedir.
b) Kullanıcı hikayeleri, teknik görevlerle kısmen ilişkilendiriliyor ve gereksinim sahipleriyle görüşülerek açıklığa kavuşturuluyor ancak sistematik izlenebilirlik eksik.
c) Sadece kullanıcı hikayeleri tanımlanıyor, teknik izlenebilirlik veya sistem gereksinimi tanımı yapılmıyor.
d) Gereksinim takibi yapılmıyor, süreçler spontane şekilde ilerliyor.
e) Bu konuda emin değilim veya bilgim yok.**"**

Birinci soru, organizasyonlarda ***Gereksinim Geliştirme ve Yönetimi (RDM)*** uygulama alanının pratiklerinin hangi ölçüde uygulandığını değerlendirmeyi amaçlamaktadır. Şekil 11'de görüldüğü üzere ankete katılan 12 firmadan 5'i (%42) a şıkkını, 7'si (%58) ise b şıkkını işaretlemiştir. Elde edilen veriler, organizasyonların çoğunda kullanıcı hikayelerinin tanımlandığını ve teknik görevlerle kısmen ilişkilendirildiğini, ancak gereksinimlerin sistematik ve iki yönlü izlenebilirliğinin tam anlamıyla sağlanmadığını göstermektedir. Bu durum, Scrum uygulamalarının gereksinim yönetimi açısından belirli bir temel sunduğunu ancak izlenebilirlik gibi kritik unsurlarda yapısal eksikliklerin bulunduğunu ortaya koymaktadır.



Şekil 11. Soru 1 – Ürün gereksinimlerinin izlenebilirliği

Bu bağlamda, CMMI-DEV V2.0 kapsamında yer alan RDM uygulama alanının, Scrum kullanılan yapılarda bu eksikliği gidermek üzere tamamlayıcı bir yapı sağlayabileceği değerlendirilmektedir. Özellikle gereksinimlerin açıkça tanımlanması, faaliyet ve çıktılarla iki yönlü olarak ilişkilendirilmesi gibi yetkinliklerin kurumsallaşması açısından RDM uygulama alanının entegrasyonu, süreçlerin olgunluk düzeyini artırmadan ziyade, çevik yöntemlerin altyapısını güçlendirmeye katkı sunmaktadır.

"2. Soru: Organizasyonunuzda kalite güvence faaliyetleri hangi düzeyde yürütülmektedir?

a) Geçmiş kalite verilerine dayanarak oluşturulmuş bir kalite güvence planı ile süreç ve çıktıların objektif, belgeli denetimleri düzenli olarak yapılıyor; kalite sorunları sistematik olarak ele alınıyor.

b) Proje süreçleri ve çıktı ürünleri belirli standartlara göre periyodik olarak gözden geçiriliyor ancak bu faaliyetler belirlenmiş bir plana bağlı değil.

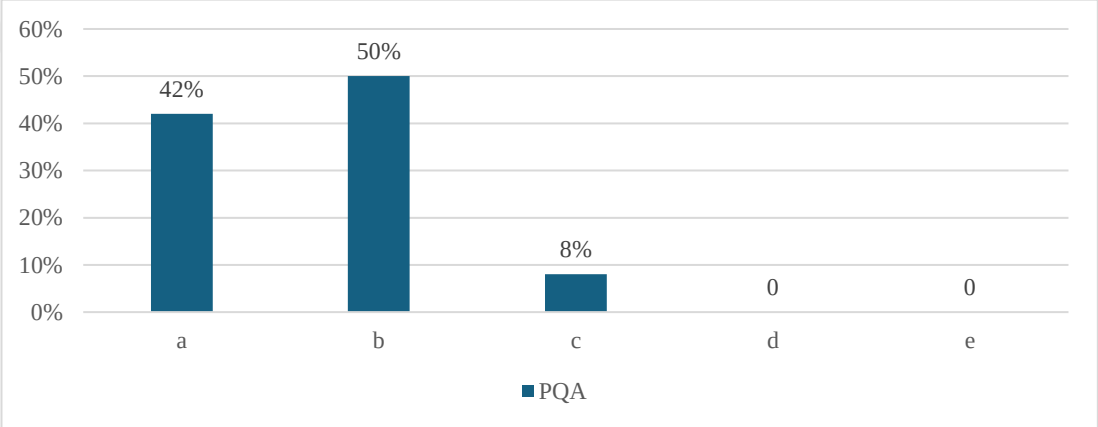
c) Süreç kalitesi genellikle takım içi gözlemler ve toplantılarla kontrol ediliyor; resmi kalite güvence planı veya değerlendirmesi yapılmıyor.

d) Kalite güvence faaliyetleri yapılmıyor ya da yalnızca hatalar ortaya çıktığında müdahale ediliyor.

e) Bu konuda emin değilim veya bilgim yok.**"**

İkinci soru, organizasyonlarda ***Süreç Kalite Güvencesi (PQA)*** uygulama alanının pratiklerinin hangi ölçüde uygulandığını değerlendirmeyi amaçlamaktadır.

Şekil 12'de görüldüğü üzere ankete katılan 12 firmadan 5'i (%42) a şıkkını, 6'sı (%50) b şıkkını ve diğer 1'i (%8) c şıkkını işaretlemiştir. Katılımcı organizasyonların önemli bir kısmı kalite güvence süreçlerini ya planlı ve sistematik biçimde ya da belirli standartlara dayalı fakat plansız olarak yürüttüğünü belirtmiştir. Bu durum, çevik yöntemleri benimseyen birçok organizasyonda kalite güvence anlayışının belirli ölçüde kurumsallaştığını, ancak tümüyle yapılandırılmış bir sistemin her zaman uygulanmadığını göstermektedir. Diğer yandan, yalnızca bir organizasyon kalite güvenceyi gayrı resmi yöntemlerle yürüttüğünü ifade etmiştir; bu da Scrum çerçevesinde kalite güvenliğinin bazı durumlarda yalnızca takım içi gözleme dayalı sınırlı bir uygulama olarak kaldığını göstermektedir.



Şekil 12. Soru 2 – Kalite güvence faaliyetlerinin yürütülme düzeyi

Bu çerçevede, CMMI-DEV V2.0 kapsamındaki PQA uygulama alanı Scrum gibi çevik yöntemleri uygulayan organizasyonlarda kalite süreçlerinin daha yapılandırılmış ve sürdürülebilir hale gelmesine katkı sağlayabilir. Özellikle kalite verilerine dayalı planlama, objektif değerlendirme ve sistematik geri bildirim döngüsü, süreçlerin izlenebilirliğini ve iyileştirilmesini kolaylaştırarak Scrum uygulamalarını destekleyici bir çerçeve sunabilir.

"3. Soru: Ekip içinde yapılan ürün veya çıktı gözden geçirme (peer review) faaliyetleri ne düzeyde yürütülmektedir?

a) Gözden geçirmeler, tanımlı prosedürlere uygun olarak gerçekleştirilmekte; sorunlar belgelenip çözümlenmekte ve sonuçlara ilişkin veriler analiz edilmektedir.

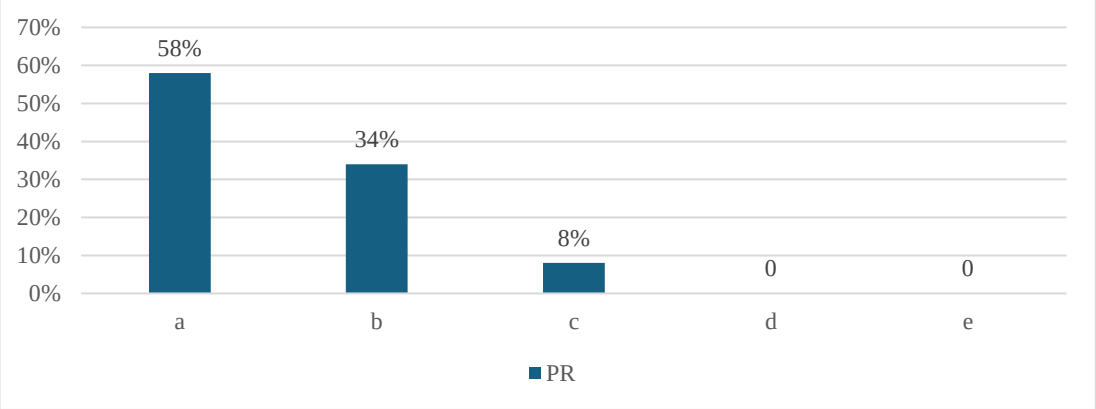
b) Ekip içinde düzenli olarak eşdeğer gözden geçirme yapılmakta ve tespit edilen hatalar üzerinde birlikte çalışılmaktadır ancak bu süreç standart bir prosedüre bağlı değildir.

c) Zaman zaman kod veya ürün gözden geçirmeleri yapılmakta, ancak çıktılar belgelenmemekte ve sistematik izleme yapılmamaktadır.

d) Herhangi bir gözden geçirme süreci uygulanmamaktadır veya yalnızca bireysel sorumluluklarla sınırlı kalmaktadır.

e) Bu konuda emin değilim veya bilgim yok."

Üçüncü soru, organizasyonlarda ***Eşdeğer Gözden Geçirme (PR)*** uygulama alanının pratiklerinin hangi ölçüde uygulandığını değerlendirmeyi amaçlamaktadır. Yanıtlar, çoğu şirketin ekip içinde ürün ve çıktı gözden geçirme faaliyetlerini belirli bir yapıya oturtarak gerçekleştirdiğini göstermektedir. Şekil 13'de gösterildiği gibi 7 şirketin (%58), gözden geçirmeleri tanımlı yöntemlere göre yürüttüğü ve elde edilen bulgular üzerinden çözüm ve analiz çalışmaları yaptığı belirlenmiştir. Bu durum, kaliteyi artırmaya yönelik uygulamaların kurumsal düzeyde benimsendiğine işaret etmektedir. 4 şirketin (%34) ise bu faaliyetleri düzenli yürütmesine rağmen standart bir yöntem benimsememesi, gözden geçirme uygulamalarının olgunluk seviyesi açısından farklılık gösterdiğini ortaya koymaktadır. Tek bir şirketin (%8) gözden geçirme faaliyetlerini düzensiz yürütmesi ise bu alanda gelişime açık noktaların varlığını göstermektedir.



Şekil 13. Soru 3 – Ekip içi eşdeğer gözden geçirme uygulamaları

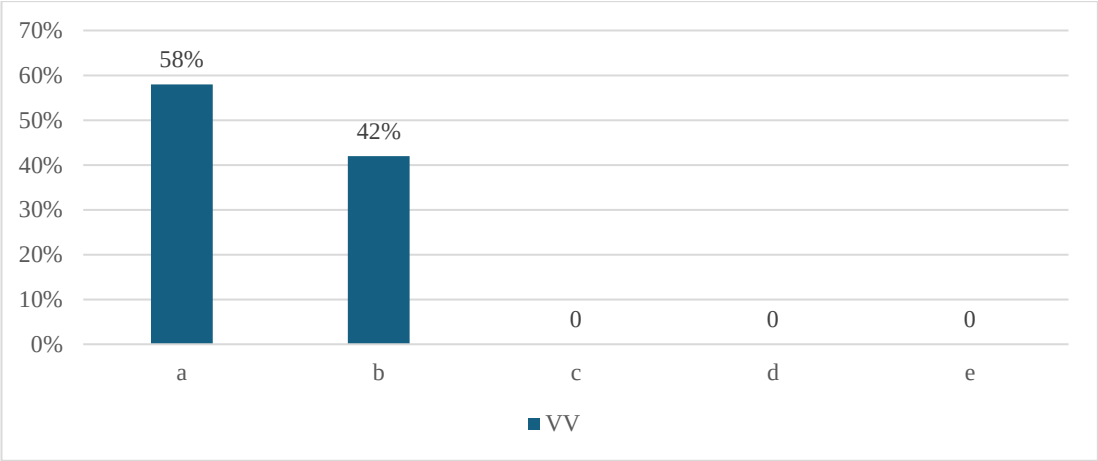
Bu bulgular ışığında, CMMI-DEV V2.0'ın PR uygulama alanının, çevik yöntemlerin uygulandığı ortamlarda çıktıların gözden geçirilmesine yönelik

uygulamalara yapı kazandırabileceği değerlendirilmektedir. Tanımlı prosedürlerin benimsenmesi, kalite kontrolünü daha güvenilir kılarken; erken hata tespiti ve ekip içi öğrenme kültürünü destekleyerek yazılım süreçlerinin verimliliğini artırabilir.

"4. Soru: Ürünlerin doğrulama (verification) ve geçerleme (validation) süreçleri organizasyonunuzda nasıl uygulanmaktadır?

- **a)** Doğrulama ve geçerleme faaliyetleri sistematik olarak gerçekleştirilmekte, tanımlı prosedürler ve kriterlere göre yürütülmekte, uygun ortamlar sağlanmakta ve sonuçlar analiz edilerek raporlanmaktadır.
- **b)** Test faaliyetleri düzenli olarak yapılmakta, temel doğrulama ve geçerleme süreçleri uygulanmakta ancak detaylı prosedürler ve kriterler tanımlanmamıştır.
- **c)** Fonksiyonel testler yapılmakta ancak geçerleme (ürünün hedef ortamda işlevselliği) ya hiç yapılmamakta ya da sınırlı kalmaktadır.
- **d)** Test ve onay süreçleri düzensiz yürütülmekte ya da kişisel sorumluluklara bırakılmaktadır.
- **e)** Bu konuda emin değilim veya bilgim yok."

Dördüncü soru, organizasyonlarda ***Doğrulama ve Geçerleme (VV)*** uygulama alanının pratiklerinin hangi ölçüde uygulandığını değerlendirmeyi amaçlamaktadır. Cevaplar, katılımcı şirketlerin önemli bir kısmında doğrulama ve geçerleme süreçlerinin tanımlı kriterlere dayalı biçimde yürütüldüğünü göstermektedir. Şekil 14'te şirketlerin 7 şirketin (%58) bu faaliyetleri yapılandırılmış prosedürlerle gerçekleştirdiği, 5 şirketin (%42) ise bu süreçleri temel düzeyde yürüttüğü ancak ayrıntılı tanım ve kriterlerin eksik olduğu görülmektedir. Bu tablo, pek çok organizasyonda test faaliyetlerinin yalnızca hata tespitiyle sınırlı kalmayıp, ürünün hem teknik doğruluğunu hem de kullanıcı ihtiyaçlarına uygunluğunu değerlendirmeye yönelik çabalar içerdiğine işaret etmektedir. Ancak detaylı prosedür eksikliği, süreçlerin tutarlılığı ve tekrarlanabilirliği açısından gelişim alanlarına da dikkat çekmektedir.



Şekil 14. Soru 4 – Doğrulama ve geçerleme süreçlerinin uygulanma düzeyi

Bu bağlamda, CMMI-DEV V2.0 kapsamındaki VV uygulama alanı, Scrum gibi çevik yöntemlerin kullanıldığı yapılarda doğruluk ve geçerlilik kontrollerinin daha disiplinli ve izlenebilir biçimde gerçekleştirilmesine katkı sağlayabilir.

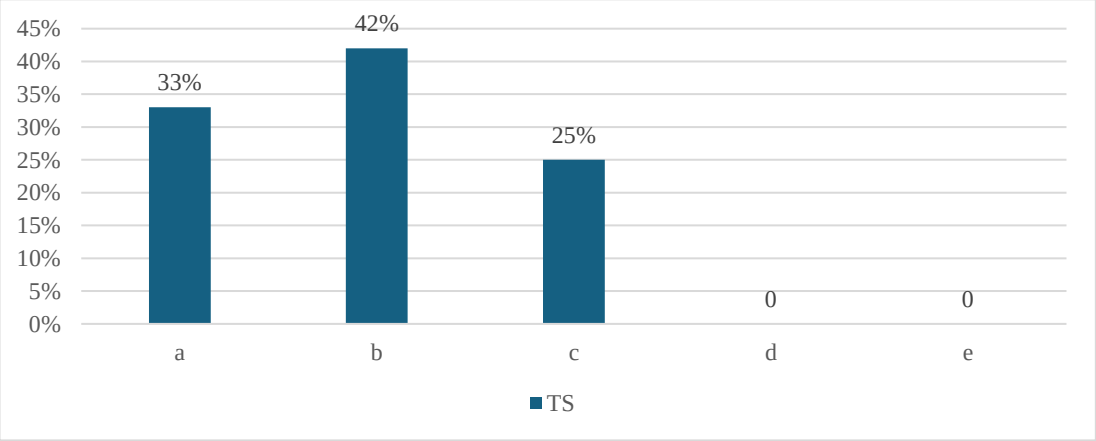
"5. Soru: Teknik gereksinimlerin karşılanması için gerekli olan çözümlerin geliştirilmesi süreci organizasyonunuzda nasıl yürütülmektedir?

- **a)** Çözüm geliştirme sürecinde alternatifler belirlenmekte, seçim kriterleri tanımlanmakta ve bu kriterlere göre en uygun teknik çözüm seçilerek tasarlanmakta, uygulanmakta ve destek dokümanlarıyla tamamlanmaktadır.
- **b)** Belirli teknik çözümler ekip içinde tartışılarak seçilmekte, tasarım ve uygulama aşamaları yürütülmekte, ancak sistematik kriterler ve alternatif analizleri yapılmamaktadır.
- **c)** Ürün bileşenleri doğrudan uygulanmakta, teknik analiz veya tasarım dokümanları genellikle oluşturulmamaktadır.
- **d)** Teknik çözümler daha çok bireysel tecrübeye dayalı olarak belirlenmekte, ekip içinde yapılandırılmış bir çözüm süreci izlenmemektedir.
- **e)** Bu konuda emin değilim veya bilgim yok."

Beşinci soru, organizasyonlarda ***Teknik Çözüm (TS)*** uygulama alanının pratiklerinin hangi ölçüde uygulandığını değerlendirmeyi amaçlamaktadır. Şekil 15'te görüldüğü üzere soruya verilen yanıtlarda 12 şirketten 4'ü çözüm geliştirme sürecinde alternatifleri değerlendirerek, seçim kriterlerine göre en uygun çözümü seçip bunu dokümantasyonla desteklediklerini belirtmiştir. Bu şirketler, CMMI-DEV v2.0

kapsamındaki TS uygulama alanının öngördüğü iyi uygulamaları büyük ölçüde gerçekleştirmektedir. Ancak 5 şirket (%42) teknik çözüm seçimlerinin ekip içinde tartışılarak yapıldığını, fakat alternatif analizleri ve seçim kriterlerinin net olarak tanımlanmadığını ifade etmiştir. Bu durum, çözümlerin yalnızca deneyime ya da alışkanlıklara dayanarak belirlendiğini ve kurumsal bir teknik karar süreci eksikliği olduğunu göstermektedir. 3 şirket (%25) ise teknik çözümlerin doğrudan uygulandığını ve genellikle teknik analiz veya tasarım dokümanlarının oluşturulmadığını belirtmiştir.

Bu çerçevede, CMMI-DEV V2.0 kapsamındaki TS uygulama alanı, pratiklerinin çevik organizasyonlarda çözüm geliştirme süreçlerinde kullanılmasıyla çözümlerin daha tutarlı, ihtiyaçlara uygun**,** uygulanabilir ve sürdürülebilir hale gelmesini sağlayarak ürün kalitesi ve müşteri memnuniyetini artırabilir.



Şekil 15. Soru 5 – Teknik çözüm geliştirme sürecinin uygulanma düzeyi

“6. Soru: Scrum süreçlerinizde ürün bileşenlerinin entegrasyonu nasıl gerçekleştirilmektedir?

a) Ürün bileşenleri entegrasyon stratejisine göre aşamalı ya da toplu olarak entegre edilmekte; her bileşenin doğruluğu, uyumluluğu ve arayüz tutarlılığı entegrasyon öncesinde ve sonrasında sistematik olarak değerlendirilmektedir.

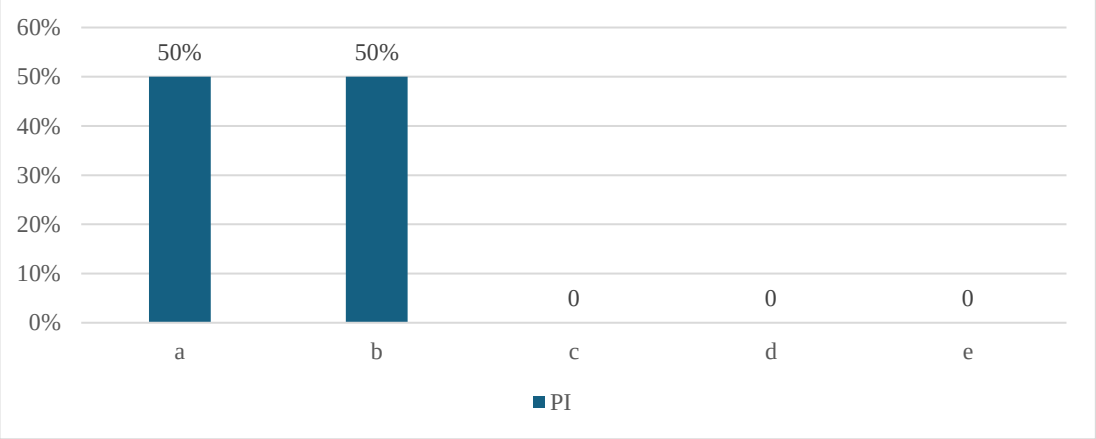
b) Bileşenler genellikle planlı bir entegrasyon ortamında test edilerek birleştirilmekte, ancak entegrasyon stratejisi ve prosedürleri dokümante edilmemekte ya da sistematik olarak uygulanmamaktadır.

c) Entegrasyon genellikle sprint sonlarında birleştirme şeklinde yapılmakta, ancak bileşen uyumu, gereksinim karşılaması veya arayüz tutarlılığı gibi kriterler genellikle göz ardı edilmektedir.

d) Bileşenler doğrudan birleştirilmekte, arayüz ya da sistem uyumu gibi entegrasyon kriterleri değerlendirilmemektedir.

e) Bu konuda emin değilim veya bilgim yok."

Altıncı soru, organizasyonlarda ***Ürün Entegrasyonu (PI)*** uygulama alanının ne ölçüde uygulandığını değerlendirmeyi amaçlamaktadır. Scrum uygulayan kuruluşlardan alınan yanıtlar, Şekil 16'da görüldüğü üzere, çoğu organizasyonun entegrasyon faaliyetlerini belirli bir stratejiye dayandırarak sistematik şekilde yürüttüğünü göstermektedir. Katılımcıların yarısı, ürün bileşenlerini aşamalı ya da toplu olarak entegre ettiklerini ve entegrasyon öncesi-sonrası doğruluk, uyumluluk ve arayüz tutarlılığı gibi kriterleri dikkate aldıklarını belirtmiştir. Bu durum, CMMI-DEV V2.0 kapsamındaki PI uygulama alanı prensiplerinin uygulandığını göstermektedir. Diğer katılımcılar ise entegrasyonun planlı ortamda yürütüldüğünü ancak strateji ve prosedürlerin yeterince dokümante edilmediğini ifade etmiştir.



Şekil 16. Soru 6 – Ürün entegrasyon süreçlerinin uygulanma düzeyi

Bulgular, entegrasyon süreçlerinin var olduğunu ancak süreç olgunluğu ve kurumsallaşma düzeyinde iyileştirme gereksiniminin bulunduğunu ortaya koymaktadır. Sonuç olarak, süreçlerin daha tutarlı ve sürdürülebilir hale gelmesi için CMMI-DEV rehberliğinde ek yapılandırmalara ihtiyaç duyulmaktadır.

“7. Soru: Scrum ekipleriniz dış tedarikçilerle çalıştığında, bu süreçler nasıl yönetilmektedir?

a) Tedarikçiler, belirlenmiş kriterlere göre seçilmekte; sözleşmeler açık bir şekilde tanımlanmakta ve anlaşma süresince teslimatlar ile faturalar sistematik olarak değerlendirilerek, anlaşma şartlarına uygunluk doğrulanmaktadır.

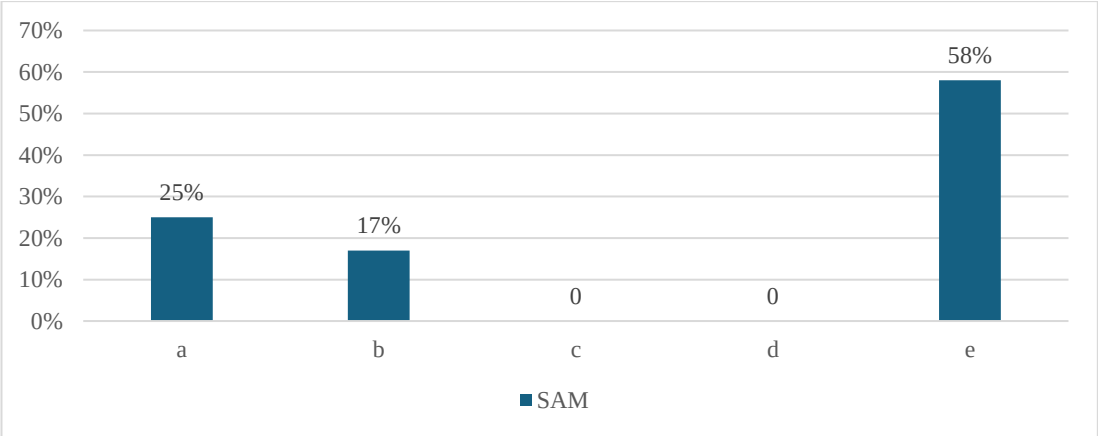
b) Tedarikçiler belirli bir değerlendirme süreciyle seçilmekte ve sözleşmeler oluşturulmaktadır; ancak teslimat ve fatura takibi süreçleri sistematik olarak yürütülmemektedir.

c) Tedarikçi seçimi çoğunlukla deneyime dayalı yapılmakta, sözleşmeler genel geçer hazırlanmakta ve takip süreçleri rastlantısaldır.

d) Dış kaynak kullanımında sözleşme ve tedarikçi değerlendirme süreçleri resmi olarak yapılmamaktadır.

e) Bu konuda emin değilim veya bilgim yok.”

Yedinci soru, organizasyonların dış tedarikçilerle olan ilişkilerinde ***Tedarik Yönetimi (SAM)*** uygulama alanı pratiklerinin ne ölçüde uygulandığını değerlendirmeye yöneliktir. Şekil 17’de gösterilen yanıtlara göre, katılımcı kuruluşların yalnızca 5’i (%42) tedarik süreçlerini açık sözleşmeler ve sistematik teslimat-fatura değerlendirmeleriyle yönettiklerini belirtmiştir. Buna karşılık, diğer 7 kurum (%58), bu konuda bilgi sahibi olmadığını veya emin olmadığını ifade etmiştir. Bu bulgular, dış kaynak kullanımında süreç farkındalığının düşük olduğunu ve iyileştirme gereksiniminin yüksek olduğunu ortaya koymaktadır. SAM uygulama alanının benimsenmesiyle birlikte, tedarikçilerin seçiminden teslimatların değerlendirilmesine kadar tüm süreçlerin belirli kriterlere göre yürütülmesi sağlanabilir. Bu durum hem hizmet kalitesinin artırılmasına hem de proje hedeflerinin zamanında ve bütçeye uygun şekilde gerçekleştirilmesine katkı sağlayacaktır. Ayrıca, sözleşme yönetimi faaliyetlerinin kurumsal düzeyde sistematikleştirilmesi, dış kaynaklı risklerin azaltılmasına ve sürdürülebilir iş ortaklıklarının kurulmasına olanak tanır.



Şekil 17. Soru 7 – Tedarikçi yönetimi süreçlerinin olgunluk düzeyi

Sonuç olarak, SAM uygulama alanı, dış tedarikçi ilişkilerinde süreç olgunluğunu artırmak isteyen Scrum ekipleri için önemli bir iyileştirme fırsatı sunmaktadır.

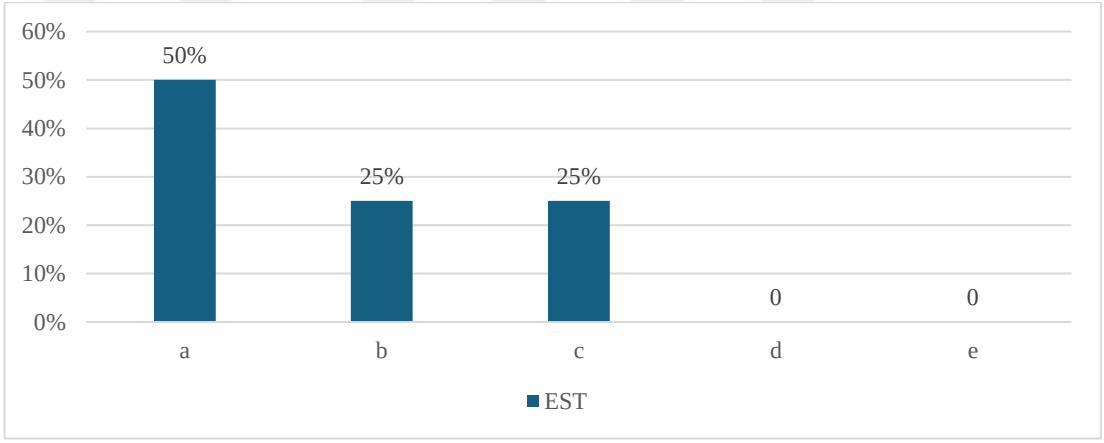
"8. Soru: Scrum ekiplerinizde, proje süresi, maliyet, efor ve kaynaklar gibi unsurlar için tahmin süreci nasıl yürütülmektedir?

- **a)** Tahminler, geçmiş veriler ve belirlenmiş bir temele dayanarak geliştirilmekte; proje boyunca paydaşlarla paylaşılmakta ve gerektiğinde güncellenmektedir.
- **b)** Tahminler yapılmakta ve proje başlangıcında paydaşlarla paylaşılmaktadır; ancak tahminler genellikle proje boyunca güncellenmemektedir.
- **c)** Tahminler ekip içi deneyime dayalı yapılmakta, resmi bir temel veya güncelleme süreci bulunmamaktadır.
- **d)** Projelerde tahmin yapılmamakta ya da yalnızca çok yüzeysel tahminler kullanılmaktadır.
- **e)** Bu konuda emin değilim veya bilgim yok."

Sekizinci soruya verilen yanıtlar, ***Tahmin (EST)*** uygulama alanı çerçevesinde değerlendirildiğinde, kuruluşların tahmin süreçlerinde farklı olgunluk seviyelerinde konumlandığını ortaya koymaktadır. Şekil 18'de gösterildiği gibi katılımcıların yarısı (%50), tahminlerin geçmiş verilere ve belirlenmiş bir temele dayandırıldığını, paydaşlarla paylaşıldığını ve proje boyunca güncellendiğini ifade etmiştir. Bu yaklaşım, EST uygulama alanında tanımlanan tüm temel süreçlerin uygulandığını göstermektedir. Katılımcıların 3'ü (%25) ise tahminlerin yalnızca proje başında

yapıldığını ve süreç boyunca güncellenmediğini belirtmiş; bu durum, sürecin kısmen yapılandırıldığını ancak sürdürülebilirliğinin sınırlı olduğunu göstermektedir. Geriye kalan 3 kurumun (%25), tahmin sürecini ekip deneyimine dayalı olarak ve resmi bir temel ya da güncelleme süreci olmaksızın yürüttüğü görülmektedir. Bu bulgu, bazı organizasyonlarda tahmin uygulamalarının belirsizlik taşıdığını ve süreç olgunluğunun düşük olduğunu göstermektedir.

Sonuç olarak, EST uygulama alanının, organizasyonlarda süreç tutarlılığını ve planlama doğruluğunu artırmak için önemli bir iyileştirme alanı sunduğu söylenebilir. CMMI-DEV V2.0 rehberliği doğrultusunda tahmin süreçlerinin sistematik hale getirilmesi hem proje performansını hem de paydaş güvenini artıracaktır.



Şekil 18. Soru 8 – Tahmin süreçlerinin uygulanma düzeyi

“**9. Soru: Scrum ekiplerinizde, proje faaliyetlerinin bütçesi, takvimi, görev dağılımı, paydaş katılımı ve gerekli becerilere göre nasıl bir planlama süreci yürütülmektedir?**

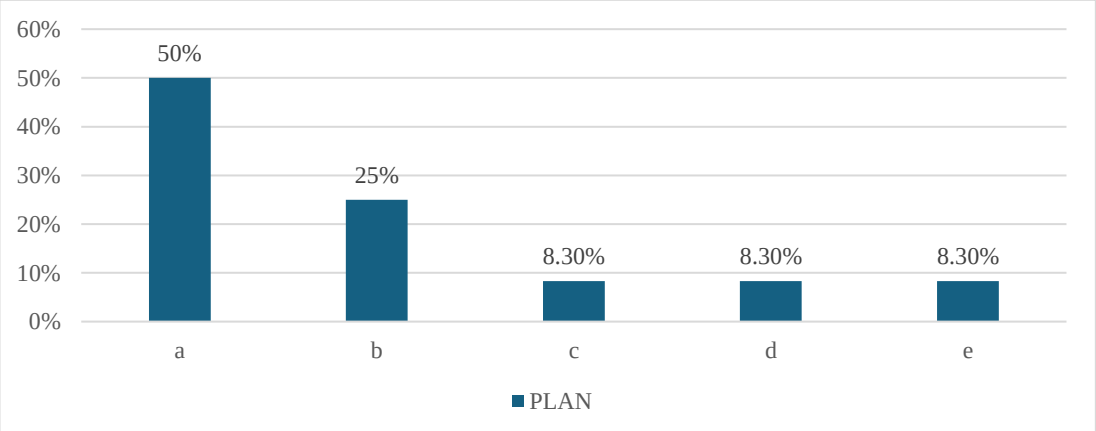
- **a)** Tahminlere dayalı olarak güncel planlar geliştirilmekte, gerekli kaynak ve beceriler belirlenmekte, tüm paydaşlar sürece dahil edilmekte ve planlar proje boyunca gözden geçirilerek güncellenmektedir.
- **b)** Projeye başlangıçta bir plan yapılmakta ve gerekli kaynaklar belirlenmekte, ancak planlar nadiren güncellenmektedir.
- **c)** Planlama daha çok sprint düzeyinde yapılmakta, uzun vadeli planlama ve paydaş katılımı sınırlı kalmaktadır.

d) Planlama süreci oldukça yüzeysel yürütülmekte, genellikle bireysel görev bazında ilerlenmektedir.

e) Bu konuda emin değilim veya bilgim yok.”

Dokuzuncu soruya verilen yanıtlar, ***Planlama (PLAN)*** uygulama alanı ile ilişkilendirildiğinde, kuruluşların bu alandaki pratikleri ne ölçüde uyguladığı görülmektedir. Şekil 19’da görüldüğü üzere katılımcıların yarısı (%50), tahminlere dayalı olarak kapsamlı planlar oluşturduklarını, gerekli kaynak ve becerileri belirlediklerini, paydaşları sürece dahil ettiklerini ve planları proje boyunca güncel tuttuklarını belirtmiştir. Bu yaklaşım, PLAN uygulama alanının tüm temel gereklerini karşıladığını ve kuruluşların bu konuda yüksek bir süreç olgunluğuna sahip olduğunu göstermektedir. Katılımcıların 3’ü (%25) başlangıçta planlama yaptıklarını ancak bu planların nadiren güncellendiğini belirtmiştir. Bu durum, planlamanın yapılandırılmış biçimde başladığını ancak sürdürülebilirliğinde eksiklikler olduğunu ve değişen koşullara adaptasyonun sınırlı kaldığını göstermektedir. Diğer 3 kurumun (%25) yanıtları, planlamanın sprint düzeyine sınırlı kalması, yüzeysel yürütülmesi ya da bilgi eksikliği gibi nedenlerle, organizasyonların bu uygulama alanında henüz yeterli bir kurumsal olgunluğa ulaşamadığını ortaya koymaktadır.

Sonuçlar, PLAN uygulama alanının, özellikle planların güncellenmesi, uzun vadeli düşünme ve paydaş katılımının artırılması yönleriyle bazı kuruluşlar için önemli bir iyileştirme fırsatı sunduğunu göstermektedir. CMMI-DEV V2.0 rehberliği ile planlama süreçlerinin sistematik ve sürdürülebilir hale getirilmesi, ekiplerin stratejik hedeflerle daha uyumlu çalışmasına katkı sağlayacaktır.



Şekil 19. Soru 9 – Planlama süreçlerinin uygulanma düzeyi

“10. Soru: **Projenizde ilerleme, tahminlerle karşılaştırılarak nasıl izleniyor ve sapmalar tespit edildiğinde hangi önlemler alınmaktadır?**

a) Proje ilerlemesi, tahminlerle düzenli olarak karşılaştırılır; sapmalar tespit edildiğinde düzeltici önlemler alınır ve paydaş katılımı sürekli izlenir.

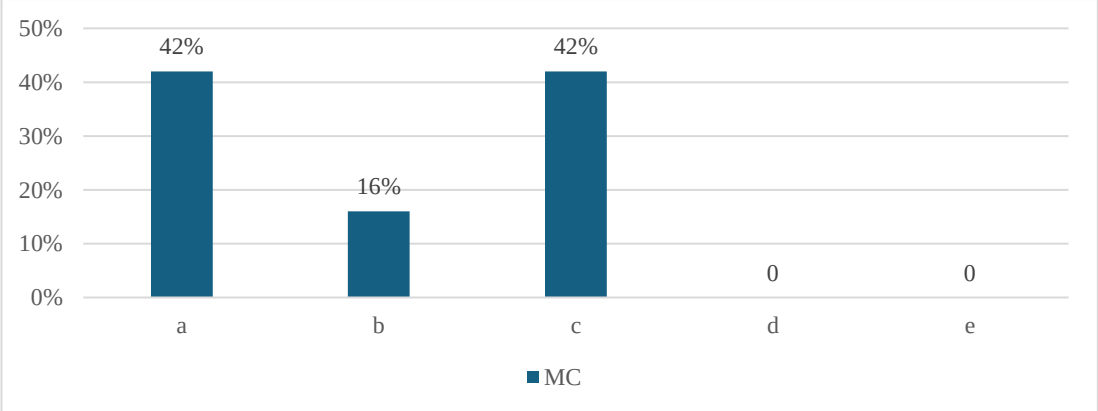
b) Proje ilerlemesi, sadece belirli zamanlarda gözden geçirilir ve sapmalar tespit edilse de genellikle düzeltici önlemler alınmaz.

c) İlerleme çoğunlukla tahminlere dayalı olarak takip edilmez, ancak bazı kritik kilometre taşlarında değerlendirmeler yapılır.

d) Proje izleme süreci genellikle eksik ve katılımcıların taahhütleri izlenmeden ilerlenir.

e) Bu konuda emin değilim veya bilgim yok.”

Onuncu soruya verilen yanıtlar, CMMI-DEV V2.0 kapsamındaki ***İzleme ve Kontrol (MC)*** uygulama alanında tanımlanan süreçlerin organizasyonlar tarafından hangi düzeyde uygulandığına dair önemli veriler sunmaktadır. Şekil 20’den anlaşılacağı üzere kuruluşların bir kısmı, proje boyunca elde edilen çıktıları önceden yapılan tahminlerle düzenli olarak karşılaştırarak süreci etkin bir şekilde izlemekte ve sapma durumlarında gerekli önlemleri alma eğilimi göstermektedir. Ancak bazı katılımcıların yalnızca sınırlı dönemlerde değerlendirme yaptığı veya süreci yalnızca kritik aşamalarda gözden geçirdiği anlaşılmaktadır. Bu durum, proje hedeflerine ulaşmada izleme ve müdahale uygulamalarının her zaman yeterince etkin biçimde yürütülmediğini göstermektedir. Özellikle paydaş taahhütlerinin takibi ve sapma durumlarına hızlı tepki verilmesi gibi pratiklerin her kuruluşta eşit düzeyde uygulanmadığı görülmektedir.



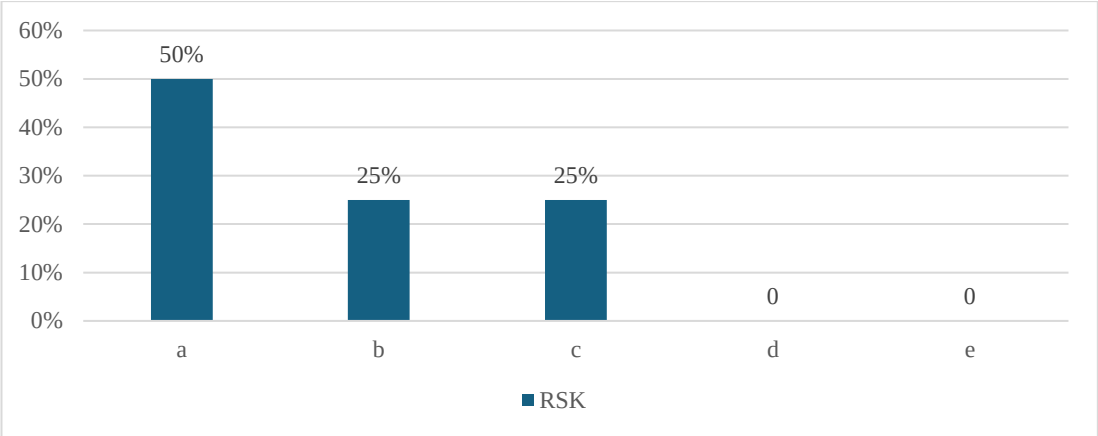
Şekil 20. Soru 10 – İzleme ve kontrol süreçlerinin etkinliği

Bu çerçevede, MC uygulama alanının, özellikle proje ilerlemesinin düzenli olarak izlenmesi, sapmaların hızlı bir şekilde düzeltilmesi ve paydaş katılımının sağlanması açısından bazı kuruluşlar için önemli bir iyileştirme fırsatı sunduğunu göstermektedir. CMMI-DEV V2.0 rehberliği ile izleme ve kontrol süreçlerinin etkili bir şekilde yapılandırılması, projelerin daha kontrollü bir şekilde yönetilmesine ve hedeflere ulaşmada daha yüksek başarı oranlarına katkı sağlayacaktır.

"11. Soru: Projenizde potansiyel riskler ve fırsatlar nasıl yönetiliyor? Bu süreçte hangi uygulamaları dikkate alıyorsunuz?

a) Riskler ve fırsatlar belirlenip analiz edildikten sonra önceliklendirilir, riskler için önlemler alınır ve fırsatlar değerlendirilir. Sürekli izleme ve gözden geçirme yapılır.
b) Riskler tanımlandıktan sonra fırsatlar göz ardı edilir, sadece risklere karşı önlemler alınır.
c) Riskler ve fırsatlar belirlenir, ancak analiz ve önceliklendirme süreçlerine yeterince önem verilmez.
d) Risk ve fırsat yönetimi süreci genellikle yapılmaz veya sadece belirli dönemlerde gözden geçirilir.
e) Bu konuda emin değilim veya bilgim yok."

On birinci soruya verilen yanıtlar, CMMI-DEV V2.0 kapsamındaki ***Risk ve Fırsat Yönetimi (RSK)*** uygulama alanında tanımlanan süreçlerin organizasyonlar tarafından hangi düzeyde uygulandığına dair önemli veriler sunmaktadır. Şekil 21'de kurumların verdiği yanıtların dağılımı gösterilmektedir. Yanıtlar, katılımcı kuruluşların çoğunun (%50) riskleri ve fırsatları tanımlama konusunda belirli bir farkındalığa sahip olduğunu, ancak analiz, önceliklendirme ve sürekli izleme gibi adımların bazı kuruluşlar (%50) tarafından sınırlı ölçüde uygulandığını göstermektedir. Özellikle yalnızca riskleri yöneten ancak fırsatları göz ardı eden veya her iki unsuru tanımlasa dahi derinlemesine analiz etmeyen kuruluşlar, belirsizlikleri yönetme ve projeyi stratejik fırsatlar doğrultusunda şekillendirme potansiyelinden uzak kalmaktadır.



Şekil 21. Soru 11 – Risk ve fırsat yönetimi süreçlerinin olgunluğu

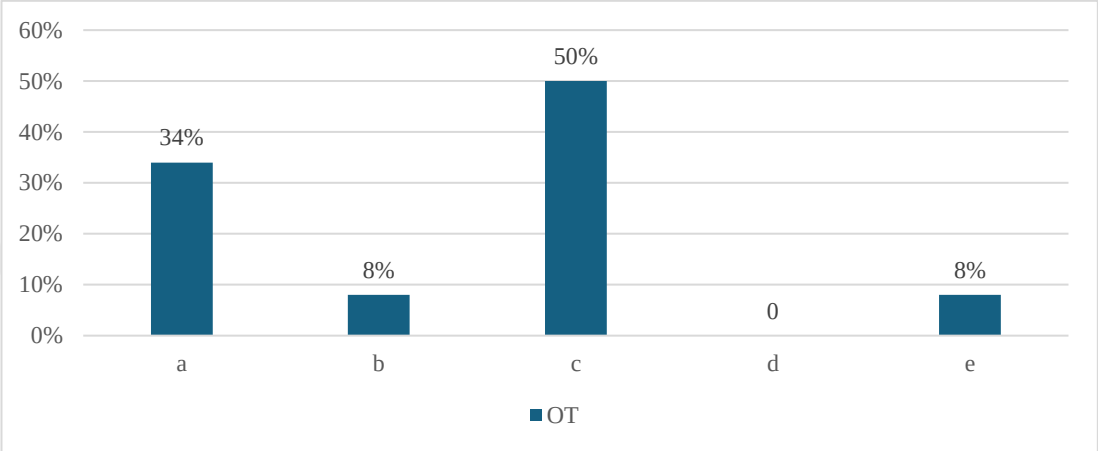
Bu bağlamda, RSK uygulama alanının sistemli şekilde benimsenmesi, kuruluşların projelerdeki belirsizlikleri daha etkin biçimde yönetmelerine, riskleri minimize ederken ortaya çıkan fırsatları değerlendirme kapasitesini artırmalarına olanak sağlayacaktır. Böylece hem proje başarısı üzerindeki olumsuz etkilerin azaltılması hem de değer yaratma potansiyeli taşıyan fırsatların sürdürülebilir katma değere dönüştürülmesi mümkün olabilecektir.

"12. Soru: Organizasyonunuzda eğitim ihtiyaçları nasıl belirleniyor ve bu eğitimlerin etkinliği nasıl değerlendirilip izleniyor?

a) Eğitim ihtiyaçları belirlenip organizasyonun stratejik hedeflerine uygun eğitim programları geliştirilir ve etkinlikleri sürekli izlenir.
b) Eğitim ihtiyaçları yalnızca projeler için belirlenir, organizasyonel ihtiyaçlar dikkate alınmaz.
c) Eğitim planları oluşturulur, ancak etkinlikleri değerlendirmek için yeterli araç ve takip yapılmaz.
d) Eğitim ihtiyaçları belirlenmez ve bu alanda herhangi bir eğitim süreci yoktur.
e) Bu konuda bilgi sahibi değilim veya süreçlerin nasıl yürütüldüğünden emin değilim."

On ikinci soruya verilen yanıtlar, CMMI-DEV V2.0 kapsamındaki ***Organizasyonel Eğitim (OT)*** uygulama alanında tanımlanan süreçlerin organizasyonlar tarafından hangi düzeyde uygulandığına dair önemli veriler sunmaktadır. Şekil 22'de kurumların verdiği yanıtların dağılımı gösterilmektedir. Yanıtlar, katılımcı kuruluşların büyük bir kısmında eğitim ihtiyaçlarının tespitine

yönelik yapılandırılmış bir yaklaşımın eksik olduğunu ve uygulanan eğitimlerin etkinliğinin izlenmesi konusunda belirgin boşluklar bulunduğunu göstermektedir. Özellikle katılımcıların yarısından fazlası, eğitim etkinliğinin değerlendirilmesine yönelik takip mekanizmalarının yetersizliğine işaret etmektedir. Bu durum, çalışanların bilgi ve becerilerinin organizasyonel hedeflerle ne ölçüde örtüştüğünü belirlemeyi ve yetkinlik gelişimini izlemeyi zorlaştırmaktadır.



Şekil 22. Soru 12 – Eğitim süreçlerinin yönetimi ve izlenebilirliği

CMMI-DEV V2.0 kapsamındaki Organizasyonel Eğitim (OT) uygulama alanı, yalnızca bireysel değil, kurumsal düzeyde de öğrenme ve gelişimi desteklemeyi hedeflemektedir. Bu uygulama alanının benimsenmesiyle, kuruluşlar hem stratejik hem de operasyonel düzeyde eğitim ihtiyaçlarını daha sağlıklı biçimde belirleyebilir, öğrenme çıktılarının ölçülmesini sağlayabilir ve bilgi paylaşımını kurumsal bir kültüre dönüştürerek insan kaynağının niteliğini artırabilir. Böylece hem çalışanların performansında hem de süreçlerin verimliliğinde sürdürülebilir bir iyileşme sağlanabilir.

“13. Soru: Organizasyonunuzda kök neden analizi ve çözümü süreçleri nasıl yürütülmektedir?

a) Olumsuz sonuçlar ve başarılar analiz edilip kök nedenler belirlenir, çözüm önerileri uygulanır ve etkiler değerlendirilir.

b) Sadece olumsuz sonuçlar analiz edilir ve çözüm önerileri uygulanır, ancak etkiler yeterince izlenmez.

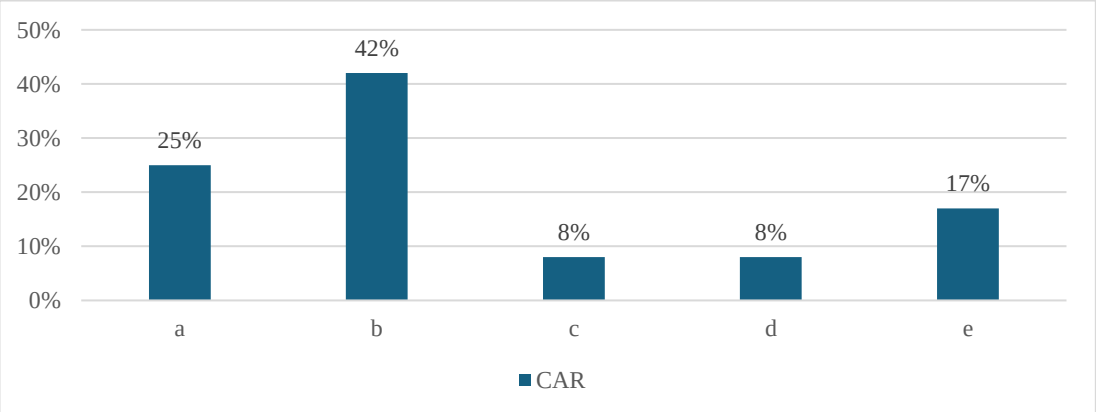
c) Kök neden analizi yapılır, ancak çözüm önerileri uygulanmaz ve etkileri değerlendirilmez.

d) Kök neden analizi ve çözümü süreçleri hiç yapılmaz.

e) Bu konuda bilgi sahibi değilim veya süreçlerin nasıl yürütüldüğünden emin değilim."

On üçüncü soruya verilen yanıtlar, CMMI-DEV V2.0 kapsamındaki ***Neden Analizi ve Çözümü (CAR)*** uygulama alanında tanımlanan süreçlerin organizasyonlar tarafından hangi düzeyde uygulandığına dair önemli veriler sunmaktadır. Şekil 23'teki grafikte yanıtlar, kuruluşların kök neden analizi ve çözümü süreçlerine dair belirli bir farkındalığa sahip olmakla birlikte, bu sürecin uygulama bütünlüğünde ve sürekliliğinde eksiklikler yaşandığını göstermektedir. Yanıtların çoğunluğu (%42), yalnızca olumsuz sonuçlara odaklanıldığını, olumlu çıktılardan öğrenme fırsatlarının ise yeterince değerlendirilmediğini ortaya koymaktadır. Ayrıca çözüm önerilerinin uygulanması ve sonuçların izlenmesi konularında çeşitli zayıflıklar bulunduğu gözlemlenmektedir.

CAR uygulama alanı, tekrar eden hataların önüne geçerek kaliteyi artırma, verimliliği yükseltme ve yeniden iş yapma ihtiyacını azaltma potansiyeli sunar. Bu uygulama alanının benimsenmesi, kuruluşların sadece sorunları gidermekle kalmayıp, aynı zamanda güçlü yönlerini sürdürülebilir kılmasını mümkün kılar; böylece sürekli iyileştirme kültürünün gelişimine katkı sağlar.

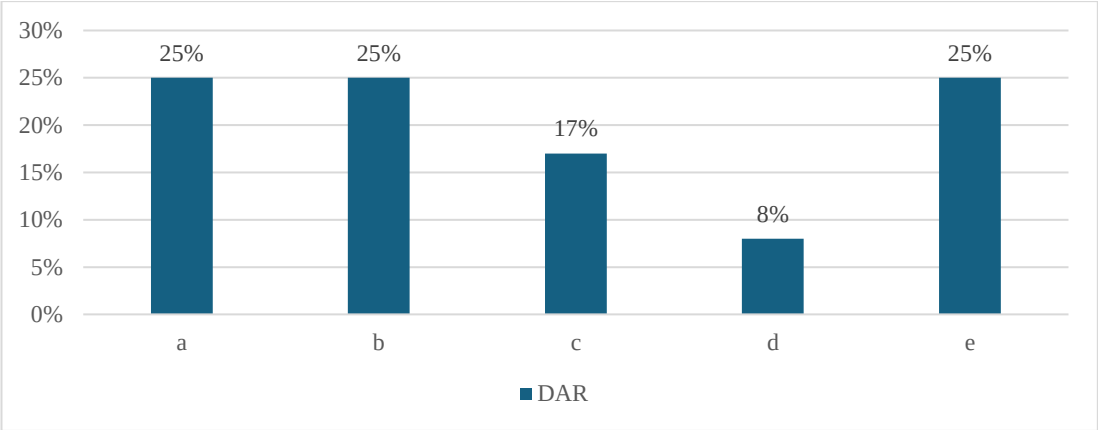


Şekil 23. Soru 13 – Kök neden analizi ve çözüm süreçlerinin uygulanma düzeyi

“**14. Soru: Karar analizi ve çözümü süreçleri organizasyonunuzda nasıl uygulanmaktadır?**

a) Karar analizi için resmi değerlendirme yöntemleri kullanılır, alternatifler sistematik şekilde analiz edilir ve değerlendirilir.
b) Yalnızca bazı kararlar için resmi değerlendirme teknikleri kullanılır, ancak alternatiflerin değerlendirilmesi sınırlıdır.
c) Karar analizi yapılır, ancak değerlendirme kriterleri ve alternatiflerin analizi genellikle eksiktir.
d) Karar analizi ve çözümü süreçleri hiç yapılmaz.
e) Bu konuda bilgi sahibi değilim veya süreçlerin nasıl yürütüldüğünden emin değilim.”

On dördüncü soruya verilen yanıtlar, CMMI-DEV V2.0 kapsamındaki ***Karar Analizi ve Çözümü (DAR)*** uygulama alanında tanımlanan süreçlerin organizasyonlar tarafından hangi düzeyde uygulandığına dair önemli veriler sunmaktadır. Şekil 24’de görülen yanıtlar, kuruluşların karar analizi ve çözümü süreçlerine yaklaşımında farklılıklar bulunduğunu göstermektedir. Yanıtlara göre yalnızca 3 kuruluş (%25), karar süreçlerinde resmi değerlendirme yöntemlerini kullanarak farklı seçenekleri karşılaştırma esasına dayalı karar verme yaklaşımlarını benimsemektedir. 3 kuruluş (%25) ise bu tür yöntemleri sadece belirli kararlar için uygulamakta, bu da tutarlı bir yaklaşımın yerleşmediğini göstermektedir. Diğer yanıtlar ise değerlendirme kriterlerinin yeterince açık tanımlanmadığı, seçeneklerin yüzeysel ele alındığı ya da karar süreçlerinin neredeyse hiç yürütülmediği durumları yansıtmaktadır. Bu tablo, karar alma sürecinde netlik, tutarlılık ve bilinçli tercih yapma kapasitesinin pek çok kuruluşta sınırlı kaldığını göstermektedir.



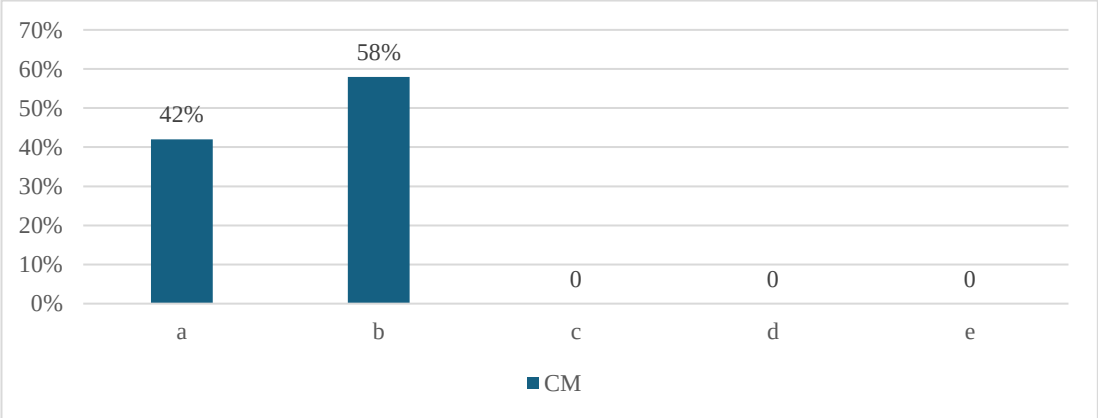
Şekil 24. Soru 14 – Karar analizi ve çözümü süreçlerinin uygulanma düzeyi

DAR uygulama alanı, karar süreçlerinin daha etkili yürütülmesi için rehberlik sağlar. Bu uygulama alanı, özellikle etkisi yüksek olan karar noktalarında hangi kriterlere göre hareket edileceğinin netleştirilmesini ve seçeneklerin bu kriterler doğrultusunda karşılaştırılmasını teşvik eder. Ayrıca, proje hedeflerini etkileyebilecek olasılıkları önceden görerek daha isabetli çözümlerin belirlenmesine katkı sunar. Anket sonuçları doğrultusunda, bu uygulama alanının daha geniş kapsamda benimsenmesi; kuruluşların karar alma süreçlerinde daha güçlü analiz becerileri geliştirmesine, stratejik hedeflerle uyumlu seçimler yapmasına ve değişen koşullara karşı daha hazırlıklı olmalarına olanak tanıyacaktır.

"15. Soru: Projelerde hangi dosya, belge veya sistem bileşeninin hangi sürümde olduğunu bilmek, değişiklikleri takip edebilmek ve güncel bilgileri tüm ekip üyeleriyle paylaşmak adına bir takip ve kontrol sistemi uygulanıyor mu?

- **a)** Evet, hangi dosya ve sistem bileşeninin hangi sürümde olduğu açıkça takip ediliyor; yapılan değişiklikler kontrol ediliyor ve güncel bilgiler tüm ekip üyeleriyle paylaşılıyor.
- **b)** Kısmen uygulanıyor; bazı doküman ve bileşenler takip ediliyor ama her zaman düzenli değil.
- **c)** Sadece temel belgeler veya bazı kritik dosyalar takip ediliyor, genel bir sistem yok.
- **d)** Hayır, böyle bir takip ve kontrol süreci uygulanmıyor.
- **e)** Bu konuda bilgi sahibi değilim."

On beşinci soruya verilen yanıtlar, CMMI-DEV V2.0 kapsamındaki ***Konfigürasyon Yönetimi (CM)*** uygulama alanında tanımlanan süreçlerin organizasyonlar tarafından hangi düzeyde uygulandığına dair veriler sunmaktadır. Projelerde konfigürasyon yönetimi uygulamalarının olgunluk seviyesi, yapılan anketin sonuçlarına göre farklılık göstermektedir. Şekil 25'teki grafikte yanıtlar incelendiğinde, 5 kuruluşun (%42) proje belgeleri, dosyalar ve sistem bileşenleri üzerinde etkin bir takip ve kontrol süreci uyguladığı, yani değişikliklerin izlendiği, sürümlerin kaydedildiği ve güncel bilgilerin ekip içinde paylaşıldığı anlaşılmaktadır. Kalan 7 kuruluş (%58) ise bu süreci kısmen uyguladığını belirtmiş, yani belge ve bileşen takibinin yapıldığı ancak bunun her zaman düzenli işlemediğini ifade etmiştir. Bu durum, sürüm kontrolü uygulamalarının tüm organizasyon genelinde yaygın ve tutarlı şekilde yürütülmediğine işaret etmektedir. CM uygulama alanı, projelerde kullanılan iş ürünlerinin (belgeler, kodlar, tasarımlar, bileşenler vb.) tutarlılığını ve izlenebilirliğini sağlamak amacıyla kapsamlı bir çerçeve sunar.



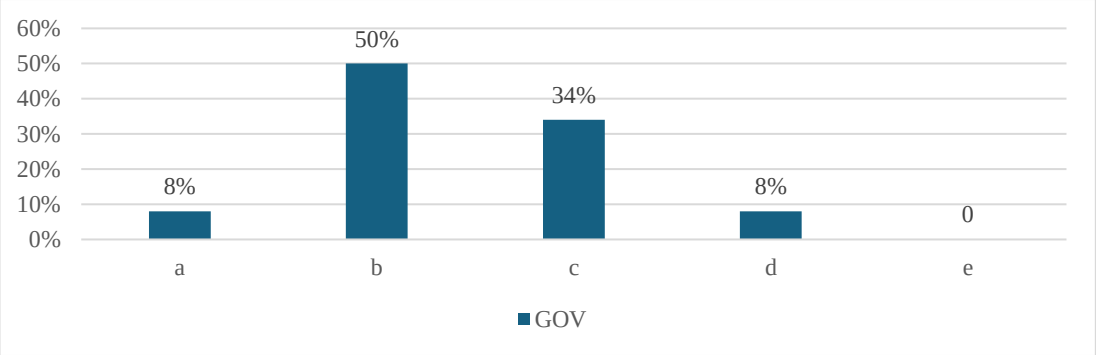
Şekil 25. Soru 15 – Konfigürasyon yönetimi süreçlerinin uygulanma düzeyi

Anket sonuçları dikkate alındığında, sürüm ve değişiklik takibinde eksiklik yaşayan kuruluşların bu uygulama alanını daha yapısal biçimde benimsemeleri; özellikle paralel çalışan ekiplerde entegrasyon sorunlarının önüne geçilmesini, hata tekrarlarının azaltılmasını ve proje çıktılarına güvenin artmasını sağlayacaktır.

“16. Soru: Üst düzey yöneticiler (örneğin genel müdür, direktör vb.), projelerin doğru yürütülmesi ve süreçlerin iyileştirilmesi için ekiplere net yönlendirme ve destek sağlıyor mu?

a) Evet, üst yönetim süreçleri yakından takip eder, gerekli yönlendirmeleri yapar ve ekipleri destekler.
b) Kısmen; bazı durumlarda yönlendirme yapılır ama sürekliliği yoktur.
c) Genellikle süreçlere müdahale etmezler; sadece büyük sorunlarda devreye girerler.
d) Hayır, üst yönetim bu konularla ilgilenmez.
e) Bu konuda bir fikrim yok.”

On altıncı soruya verilen yanıtlar, CMMI-DEV V2.0 kapsamındaki ***Yönetim (GOV)*** uygulama alanında tanımlanan pratiklerle örtüşen süreçlerin organizasyonlarda ne ölçüde mevcut olduğunu değerlendirmek amacıyla analiz edilmiştir. Şekil 26’da gösterildiği gibi verilen yanıtlara göre yalnızca bir kuruluşta (%8) üst yönetimin süreçlere aktif biçimde yönlendirme sağladığı, süreçlerin yürütülmesini doğrudan desteklediği ve ekiplere gerekli desteği sunduğu görülmektedir. 6 kuruluşta (%50) ise bu desteğin sınırlı olduğu, yalnızca belirli zamanlarda veya ihtiyaç duyulan durumlarda yönlendirme yapıldığı ifade edilmiştir. Geriye kalan kuruluşlar ise üst yönetimin süreçlere doğrudan müdahale etmediğini ve yalnızca önemli sorunlar ortaya çıktığında sürece dahil olduğunu belirtmiştir. Bu durum, Scrum kullanan birçok kuruluşta süreçlerin uygulama ve iyileştirme sorumluluğunun ekip düzeyinde kaldığını, üst yönetimin ise yönlendirici ve destekleyici bir rol üstlenme konusunda yeterince etkin olmadığını göstermektedir.



Şekil 26. Soru 16 – Üst yönetimin süreçlere katkı düzeyi

GOV uygulama alanı, süreçlerin hedeflerle uyumlu şekilde yürütülmesi, gerekli kaynakların sağlanması, performans verilerinin kullanılması ve karar alma süreçlerine yön verilmesi gibi önemli pratikleri içermektedir. Anket bulguları, söz konusu pratiklerin bu kuruluşlarda henüz bütüncül olarak uygulanmadığını ancak kısmen karşılandığını göstermektedir. Bu uygulama alanının kuruma kazandırılması, süreçlerin sadece ekip inisiyatifiyle değil, aynı zamanda yönetim desteğiyle daha güçlü ve kararlı şekilde yürütülmesini; böylece organizasyonel hedeflere ulaşmada daha etkili sonuçlar elde edilmesini mümkün kılacaktır.

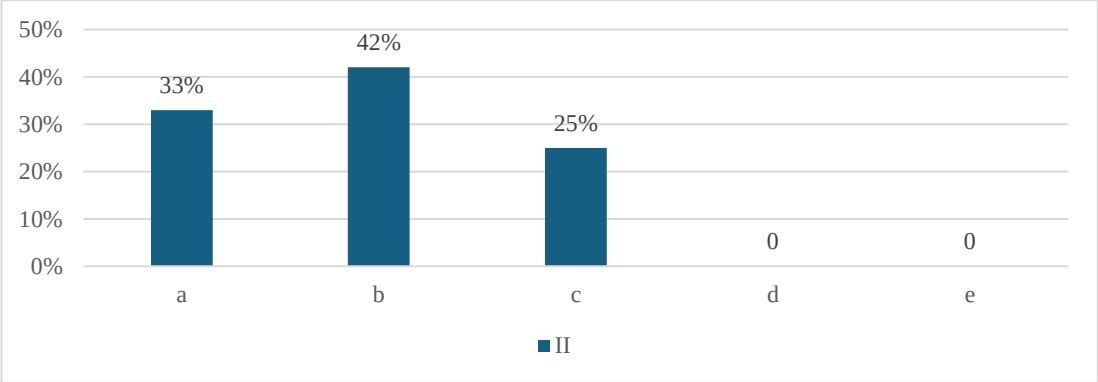
"17. Soru: İş süreçlerini uygularken veya iyileştirirken ihtiyaç duyduğunuz destek altyapısına (örneğin rehber dokümanlar, eğitim materyalleri, yazılım araçları, uzman desteği, zaman, finansman vb.) ne ölçüde erişebiliyor ve bu altyapının işinizi ne kadar kolaylaştırdığını düşünüyorsunuz?

- **a)** Tüm kaynaklara kolayca erişebiliyorum ve bu destek altyapısı işimi büyük ölçüde kolaylaştırıyor.
- **b)** Genellikle ihtiyaç duyduğum kaynaklara ulaşabiliyorum, altyapı işimi çoğunlukla destekliyor.
- **c)** Bazı kaynaklara erişimde zaman zaman zorluk yaşıyorum, altyapı kısmen yeterli.
- **d)** Kaynaklara erişim zor ve altyapı işimi verimli yapmamı önemli ölçüde engelliyor.
- **e)** Bu konuda bir değerlendirme yapamıyorum / fikrim yok."

On yedinci soruya verilen yanıtlar, CMMI-DEV V2.0 çerçevesinde tanımlanan ***Uygulama Altyapısı (II)*** uygulama alanının kapsamına giren destek unsurlarının organizasyonel düzeyde ne ölçüde karşılandığını ortaya koymaktadır. Elde edilen bulgular, incelenen kuruluşların süreçlerin yürütülmesi ve iyileştirilmesine yönelik altyapı kaynaklarına erişim düzeylerinde belirgin farklılıklar olduğunu göstermektedir. Şekil 27'deki grafikte görüldüğü gibi katılımcıların 4 tanesi (%33), süreç uygulamalarını destekleyen tüm kaynaklara kolaylıkla erişebildiklerini ve bu altyapının operasyonel verimliliğe yüksek düzeyde katkı sunduğunu belirtmiştir. Bu durum, II uygulama alanı prensiplerinin uygulandığını göstermektedir. 5 kuruluş (%42) ise bu kaynaklara çoğunlukla erişebildiğini; ancak zaman zaman yetersizlikler

yaşandığını ifade etmiştir. Kalan 3 kuruluş (%25) ise destek altyapısının kısıtlı ve erişimin zor olduğunu; bunun da süreç performansı üzerinde olumsuz etkiler yarattığını dile getirmiştir.

II uygulama alanı, süreçlerin bütünsel olarak tasarlanmasını, yürütülmesini ve sürdürülebilirliğinin sağlanmasını destekleyen kaynak planlaması, süreç varlıklarının güncellenmesi, kaynak tahsisi, değerlendirme ve iyileştirme mekanizmaları gibi unsurları içermektedir. Anket sonuçları, destek altyapısında eksiklik yaşayan kuruluşların bu uygulama alanının sunduğu yapıyı benimsemeleri halinde; süreçlerin izlenebilirliğini artırabilecekleri, insan ve teknik kaynakların daha etkin tahsis edilebileceği, süreç çıktılarına yönelik kalite ve tutarlılık düzeyinin iyileştirilebileceği sonucunu ortaya koymaktadır.

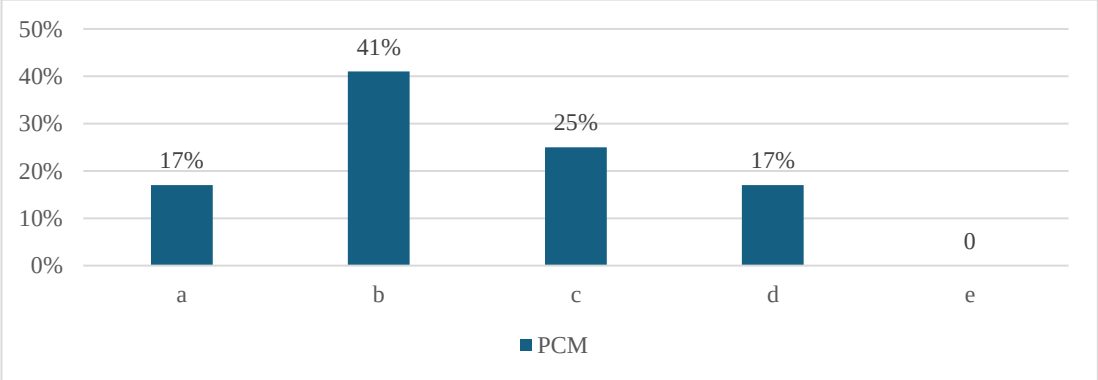


Şekil 27. Soru 17 – Uygulama altyapısına erişim ve katkı düzeyi

"18. Soru: Çalıştığınız projelerde ya da birimde, süreçlerin nasıl daha iyi hale getirileceğine dair fikirlerin toplandığını, uygulandığını ve sonuçlarının değerlendirildiğini düşünüyor musunuz?

a) Evet, süreç iyileştirme çalışmaları düzenli yapılıyor, fikirlerimiz alınıyor ve sonuçları değerlendiriliyor.
b) Zaman zaman iyileştirme çalışmaları yapılıyor, ancak tüm süreçleri kapsadığı söylenemez.
c) İyileştirme fikirleri alınıyor ama uygulanması ya da sonuçlarının izlenmesi konusunda eksiklikler var.
d) İyileştirmelere dair sistematik bir yaklaşım olduğunu düşünmüyorum.
e) Fikrim yok / bu konuda bilgi sahibi değilim."

On sekizinci soruya verilen yanıtlar, organizasyonların süreç iyileştirme faaliyetlerini ne ölçüde planlı, yaygın ve sürdürülebilir bir biçimde yürüttüğünü ortaya koymakta ve CMMI-DEV V2.0 kapsamında yer alan ***Süreç Yönetimi (PCM)*** uygulama alanı ile ilişkilendirilmektedir. Şekil 28'de gösterildiği üzere katılımcı kuruluşların yalnızca 2'si (%17) süreç iyileştirme faaliyetlerinin düzenli olarak gerçekleştirildiğini, çalışan görüşlerinin alındığını ve bu faaliyetlerin çıktılarının değerlendirildiğini belirtmiştir. Bu durum, PCM uygulama alanı prensiplerinin uygulandığını göstermektedir. Beş kuruluş, iyileştirme girişimlerinin kısıtlı olduğunu, yalnızca belirli süreçlerle sınırlı kaldığını ve kurum genelinde yaygın bir yapı oluşturulamadığını ifade etmiştir. Diğer kuruluşlar ise, iyileştirme fikirlerinin toplandığını ancak bunların uygulamaya alınması, sonuçlarının izlenmesi ve kurumsal düzeyde bir iyileştirme yaklaşımının oluşturulması konularında belirgin eksiklikler yaşandığını ifade etmiştir. Bu bulgular, birçok organizasyonda süreç iyileştirme mekanizmalarının kurumsallaşmamış, bütünsel bir stratejiye oturtulmamış olduğunu göstermektedir.



Şekil 28. Soru 18 – Süreç iyileştirme faaliyetlerinin yaygınlığı

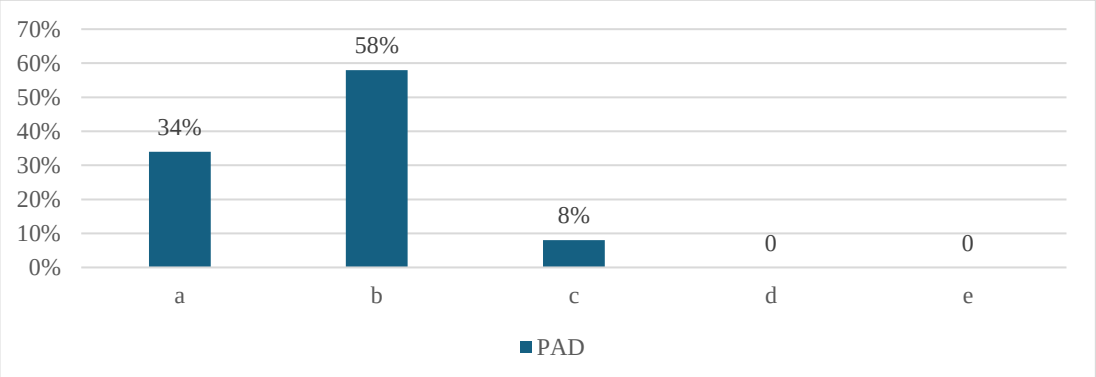
Anket verileri, iyileştirme faaliyetlerinin çoğunlukla reaktif ve sınırlı kaldığını, bu nedenle süreçlerin verimlilik, kalite ve izlenebilirlik açısından sürekli gelişimine olanak tanımadığını ortaya koymaktadır. Bu bağlamda, organizasyonların PCM uygulama alanında tanımlanan pratikleri benimseyerek; süreç iyileştirme faaliyetlerine stratejik bir yön kazandırmaları, iyileştirme hedefleriyle iş hedefleri arasında izlenebilir bir ilişki kurmaları ve uygulanan iyileştirmelerin etkinliğini periyodik olarak ölçmeleri önerilmektedir. Bu yaklaşım, organizasyonel öğrenmenin

desteklenmesini, operasyonel mükemmelliğin artırılmasını ve süreç olgunluğunun yükseltilmesini sağlayacaktır.

"19. Soru: Çalışmalarınızı yürütürken size yol gösteren standart belgeler (örneğin rehberler, kontrol listeleri, şablonlar, süreç açıklamaları vb.) yeterli düzeyde mevcut ve erişilebilir mi?

a) Evet, gerekli belgeler mevcut, kolayca erişebiliyorum ve işimi yaparken aktif olarak kullanıyorum.
b) Bazı belgeler mevcut ama eksikler var veya güncel değiller.
c) Belgeler var ama erişimi zor veya pratikte kullanılmıyor.
d) Belgelerin varlığından haberdar değilim ya da hiç görmedim.
e) İşimi yaparken böyle belgelere ihtiyaç duymuyorum."

Bu soruya verilen yanıtlar, CMMI-DEV V2.0 kapsamında yer alan ***Süreç Varlığı Geliştirme (PAD)*** uygulama alanında tanımlanan süreçlerin organizasyonlar tarafından hangi düzeyde uygulandığına dair veriler sunmaktadır. Şekil 29'da verilen grafikten anlaşıldığı gibi kuruluşların %34'ü gerekli süreç varlıklarına kolayca erişebildiklerini ve işlerini yürütürken bu varlıkları aktif olarak kullandıklarını belirtmiştir. Bununla birlikte, diğer kuruluşlar süreç varlıklarının mevcut olduğunu ancak bunların eksik veya güncel olmadığını, bazılarının ise erişiminin zor olduğunu ifade etmiştir. Bu bulgular, birçok organizasyonda süreç varlığı geliştirme faaliyetlerinin belirli bir düzen içinde yürütülmediğini ve mevcut belgelerin süreçleri destekleme konusunda yeterli olmadığını göstermektedir.



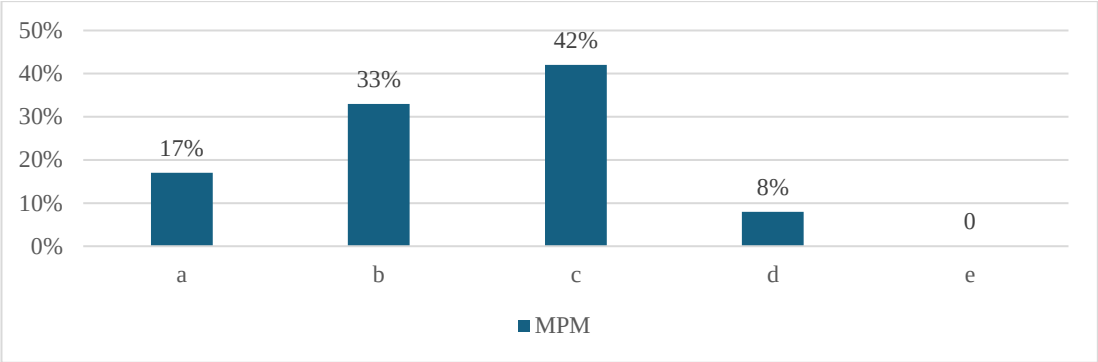
Şekil 29. Soru 19 – Süreç belgelerine erişim ve kullanım durumu

Yanıtlar, süreç rehberleri, şablonlar ve benzeri belgelerin ya eksik ya da güncel olmadığını; bazı durumlarda ise erişimin sınırlı kaldığını ortaya koymaktadır. Bu eksiklikler, çalışanların süreçleri tekrarlanabilir ve güvenilir şekilde uygulamasını zorlaştırmaktadır. Bu doğrultuda, organizasyonların PAD uygulama alanında tanımlanan pratikleri dikkate alarak; süreç belgelerini oluşturmaları, bu belgeleri güncel tutmaları ve erişilebilirliğini artırmaları önerilmektedir. Ayrıca, bu belgelerin pratikte ne ölçüde kullanıldığının ve iş süreçlerine nasıl katkı sağladığının düzenli aralıklarla değerlendirilmesi, süreç kalitesinin artırılmasına ve kurumsal öğrenmenin desteklenmesine katkı sağlayacaktır.

"20. Soru: Yaptığınız çalışmaların performansı (örneğin kalite, zamanlama, maliyet gibi) düzenli olarak ölçülüyor, bu ölçümler analiz edilerek sizinle ve/veya ekiple paylaşılıyor mu?

- **a)** Evet, performansımıza dair veriler düzenli olarak toplanıyor, analiz ediliyor ve sonuçlar bizimle paylaşılıyor. Bu veriler, iyileştirme amaçlı kullanılıyor.
- **b)** Bazı ölçümler yapılıyor ama sonuçlar nadiren paylaşılıyor ya da iyileştirme sürecine katkısı net değil.
- **c)** Ölçüm yapılıyor ama neye göre yapıldığını ya da sonuçların nasıl yorumlandığını bilmiyorum.
- **d)** Herhangi bir ölçüm yapılmıyor ya da bizimle paylaşılmıyor.
- **e)** Bilmiyorum / Hiç karşılaşmadım."

Yirminci soruya verilen yanıtlar, CMMI-DEV V2.0 kapsamındaki ***Performans Yönetimi ve Ölçümü (MPM)*** uygulama alanında tanımlanan pratiklerin organizasyonlarda ne ölçüde uygulandığını ortaya koymaktadır. Şekil 30'da gösterildiği üzere katılımcı kuruluşların %17'sinin performans verilerini düzenli olarak topladığını, analiz ettiğini ve ekiplerle paylaşarak iyileştirme amaçlı kullandığı belirtilmiştir. %33'ü ölçümlerin yapıldığını ancak sonuçların nadiren paylaşıldığını ya da paylaşılsa bile iyileştirmeye katkısının belirsiz olduğunu ifade etmiştir. %42'si ise ölçüm süreçlerinin var olduğunu ancak bunların hangi kriterlere göre gerçekleştirildiğini veya elde edilen verilerin nasıl yorumlandığını bilmediklerini; diğer bir kuruluş ise performans ölçümü yapılmadığını veya sonuçların kendileriyle paylaşılmadığını bildirmiştir.



Şekil 30. Soru 20 – Performans ölçümü ve paylaşımı durumu

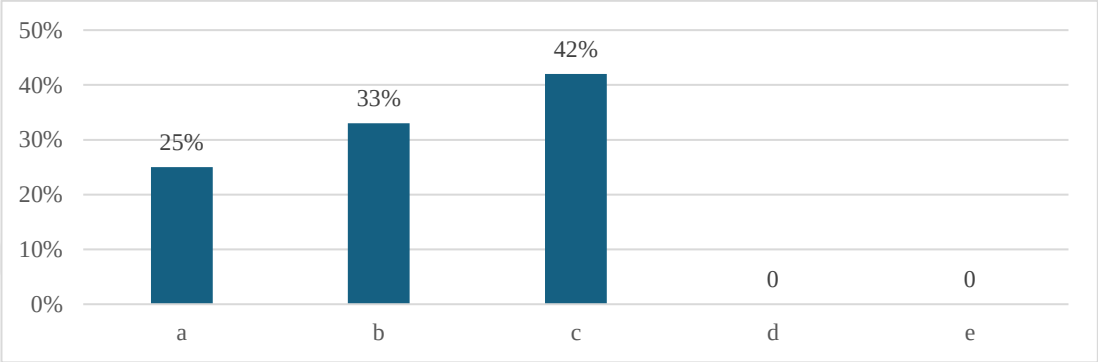
MPM uygulama alanı, iş gereksinimleri ve hedefleri doğrultusunda ölçüm hedeflerinin belirlenmesi, operasyonel tanımların oluşturulması, veri toplama ve analiz süreçlerinin işletilmesi ile performans sonuçlarının paydaşlar için görünür kılınmasını içerir. Yanıtlar, performans ve ölçüm yönetiminde eksiklik yaşayan kuruluşların bu pratikleri hayata geçirmeleri durumunda; kalite, maliyet ve zaman performansının daha etkin izlenebileceğini, analiz sonuçlarına dayalı aksiyon planlarının geliştirilebileceğini ve elde edilen verilerin ekipler arasında bilgi paylaşımını güçlendireceğini göstermektedir.

“**21. Soru: Organizasyonunuzda yazılım geliştirme süreçleri (analiz, tasarım, kodlama, test, teslimat vb.) ne ölçüde standartlaştırılmış ve yazılı olarak belgelenmiştir?**

- **a)** Tüm yazılım geliştirme süreçleri yazılı olarak açıkça tanımlanmış, standartlaştırılmış ve ekipler tarafından düzenli olarak uygulanmaktadır.
- **b)** Çoğu süreç yazılı olarak tanımlanmıştır, ekipler genellikle bu standartlara uymaktadır.
- **c)** Bazı süreçler tanımlı olsa da yazılı standartlar eksiktir ve uygulamalar ekipten ekibe değişiklik göstermektedir.
- **d)** Süreçler büyük ölçüde kişisel tercihlere dayalı olarak yürütülmektedir, yazılı bir standart bulunmamaktadır.
- **e)** Emin değilim / Bu konuda bilgi sahibi değilim.”

Yirmi birinci soruya verilen yanıtlar, yazılım geliştirme süreçlerinin organizasyon genelinde ne ölçüde tanımlandığını, yazılı hale getirildiğini ve uygulamada ne derece tutarlılık gösterdiğini değerlendirmeye yöneliktir. Şekil 31’deki

grafikte gösterilen yanıtlara göre, yalnızca 3 kuruluş (%25) yazılım yaşam döngüsünün tüm aşamalarının açıkça tanımlandığını, yazılı olarak standartlaştırıldığını ve ekiplerce düzenli biçimde uygulandığını belirtmiştir. 4 kuruluş (%33) çoğu sürecin yazılı olarak tanımlandığını ve genel olarak bu standartlara uyulduğunu ifade ederken, 5 kuruluş (%42) ise süreçlerin yalnızca kısmen tanımlı olduğunu, yazılı standartların eksik olduğunu ve uygulamaların ekipler arasında farklılık gösterdiğini beyan etmiştir.



Şekil 31. Soru 21 – Yazılım geliştirme süreçlerinin standardizasyon durumu

Bu bulgular, birçok organizasyonda yazılım geliştirme süreçlerinin kurumsal düzeyde tanımlanmasında ve standartlaştırılmasında önemli eksiklikler bulunduğunu göstermektedir. Süreçlerin açıkça tanımlanmaması ve yazılı dokümantasyonun yetersizliği, uygulamada tutarsızlıklara yol açmakta; bu durum da kalite, zamanlama ve maliyet gibi proje performansı bileşenlerini olumsuz etkilemektedir. Ayrıca, süreçlerin kişisel tercihlere veya ekip bazlı yaklaşımlara dayalı olarak yürütülmesi, kurumsal öğrenmenin ve bilgi birikiminin kurallar çerçevesinde aktarımını zorlaştırmaktadır.

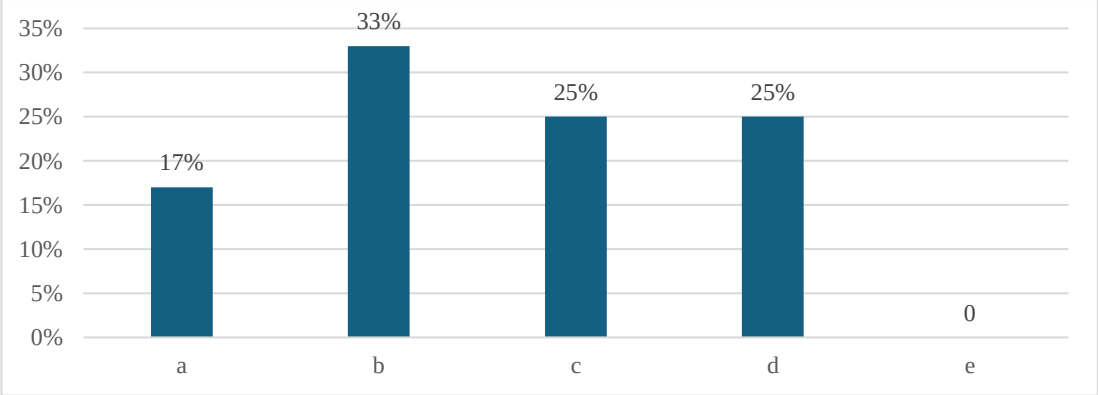
Bu çerçevede, CMMI-DEV V2.0 modelinin uygulanması, süreçlerin tanımlanması, belgelenmesi ve sürekliliğinin sağlanması bakımından kuruma önemli kazanımlar sunmaktadır. Model kapsamında yer alan uygulama alanları, yazılım süreçlerinin açık bir şekilde dokümante edilmesini, süreçler arası bütünlük ve rol tanımlarının netleştirilmesini teşvik etmektedir. Ayrıca, süreçlerin ölçülebilir ve yönetilebilir hale getirilmesiyle birlikte, kalite güvencesi, proje yönetimi ve risk kontrolü gibi alanlarda daha etkin uygulamalar gerçekleştirilebilmektedir. Bu bağlamda, CMMI-DEV V2.0'ın benimsenmesi; süreç olgunluğunun artırılması,

organizasyonel verimliliğin iyileştirilmesi ve yazılım geliştirme faaliyetlerinde sürdürülebilir başarı elde edilmesi açısından stratejik bir yaklaşım sunmaktadır.

"22. Soru: Scrum uygulamalarının dışında, organizasyonel düzeyde süreç iyileştirme çalışmaları yürütülüyor mu?

- **a)** Evet, organizasyon genelinde yapılandırılmış süreç iyileştirme stratejileri uygulanıyor ve düzenli olarak gözden geçiriliyor.
- **b)** Evet, bazı bölümler veya takımlar düzeyinde süreç iyileştirme çalışmaları var ama kurumsal bir strateji veya standardizasyon bulunmuyor.
- **c)** İyileştirme çalışmaları tamamen bireysel takımlara bırakılmış durumda, kurumsal düzeyde bir yönlendirme yok.
- **d)** Hayır, süreç iyileştirme çalışmaları yapılmıyor.
- **e)** Emin değilim / Bu konuda bilgim yok."

Yirmi ikinci soruya verilen yanıtlar, organizasyonel düzeyde süreç iyileştirme faaliyetlerinin kurumsal bir stratejiye dayanıp dayanmadığını ve ne ölçüde yapılandırıldığını göstermektedir. Şekil 32'de gösterilen yanıtlara göre, yalnızca 2 kuruluş (%17), organizasyon genelinde yapılandırılmış süreç iyileştirme stratejilerinin uygulandığını ve düzenli olarak gözden geçirildiğini belirtmiştir. 4 kuruluş (%33) ise, bu çalışmaların sadece bazı bölümler veya takımlar düzeyinde gerçekleştirildiğini, ancak kurumsal bir standardizasyonun bulunmadığını ifade etmiştir. 3 kuruluş (%25), süreç iyileştirme faaliyetlerinin yalnızca takımlara bırakıldığını ve organizasyonel düzeyde herhangi bir yönlendirmenin olmadığını vurgulamıştır. 3 kuruluş (%25) ise, bu tür çalışmaların yapılmadığını belirtmiştir.



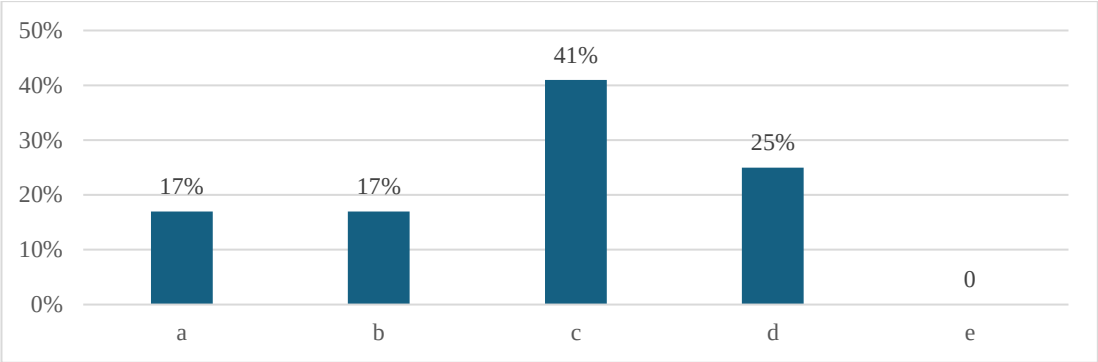
Şekil 32. Soru 22 – Organizasyonel düzeyde süreç iyileştirme faaliyetleri

Sonuçlar, birçok organizasyonda süreç iyileştirme çalışmalarının geniş bir stratejiye oturtulmadığını ve kurumsal düzeyde yapılandırılmadığını ortaya koymaktadır. Bu durum, süreçlerin verimliliğini ve etkinliğini olumsuz yönde etkileyebilir. Ayrıca, yalnızca belirli takımların sorumlu olduğu iyileştirme faaliyetleri, organizasyonel öğrenmenin bütünsel bir şekilde yayılmasını engellemektedir. Bu bağlamda, CMMI-DEV V2.0'ın benimsenmesi, süreç iyileştirme çalışmalarını kurumsal düzeyde daha etkili bir şekilde yapılandırmaya yardımcı olabilir. Bu model, süreçlerin sürekli iyileştirilmesini ve izlenebilirliğini sağlayarak organizasyonun tüm faaliyetlerinde tutarlılığı artırır. CMMI-DEV V2.0, süreçlerin organizasyonel hedeflerle uyumlu bir şekilde iyileştirilmesi için gerekli yönlendirmeleri sunmakta, bu sayede daha sürdürülebilir sonuçlar elde edilmesini mümkün kılmaktadır.

"23. Soru: Sizce Scrum uygulamaları, organizasyonel süreçlerin yeterince olgunlaşması ve standartlaştırılması için tek başına yeterli midir?

a) Evet, Scrum süreçleri yeterlidir ve ek bir sürece ihtiyaç duyulmaz.
b) Genel olarak yeterlidir ama bazı alanlarda ek rehberlik veya yapı gerekebilir.
c) Hayır, organizasyonel olgunluk ve standardizasyon için mutlaka ek süreç yönetimi/iyileştirme modelleri de gereklidir (örneğin: CMMI, ISO 15504 vb.).
d) Bu, organizasyonun ihtiyaçlarına ve büyüklüğüne göre değişir.
e) Kararsızım / Bu konuda yeterli bilgiye sahip değilim."

Yirmi üçüncü soruya verilen yanıtlar, Scrum uygulamalarının organizasyonel süreçlerin olgunlaşması ve standartlaştırılmasındaki yeterliliği konusunda farklı görüşleri yansıtmaktadır. Şekil 33'te gösterilen yanıtlara göre, 2 kuruluş (%17) Scrum süreçlerinin yeterli olduğunu ve ek bir sürece ihtiyaç duymadığını belirtmiştir. 2 kuruluş (%17) ise, genel olarak Scrum'ın yeterli olduğunu ancak bazı alanlarda ek rehberlik veya yapı gerektiğini ifade etmiştir. 5 kuruluş (%41), organizasyonel olgunluk ve standardizasyon için yalnızca Scrum'ın yetersiz olduğunu ve ek süreç yönetimi/iyileştirme modellerinin de gerektiğini belirtmiştir. 3 kuruluş (%25) ise, bu gerekliliğin organizasyonun ihtiyaçlarına ve büyüklüğüne göre değişebileceğini ifade etmiştir.



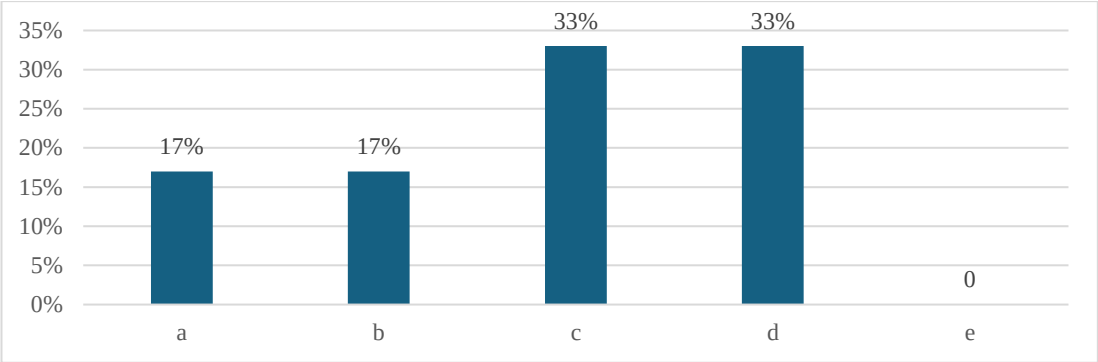
Şekil 33. Soru 23 – Scrum uygulamalarının süreç olgunluğu konusunda yeterlilik değerlendirmesi

Sonuçlar, Scrum'ın organizasyonel süreçleri olgunlaştırmak ve standartlaştırmak için genellikle yeterli görülmediğini göstermektedir. Özellikle süreçlerin organizasyonel düzeyde tutarlılığın sağlanabilmesi için, Scrum uygulamalarına ek olarak süreç yönetimi ve iyileştirme modellerine olan ihtiyaç belirginleşmektedir. CMMI-DEV V2.0 gibi yetenek olgunluk modelleri, organizasyonların süreçlerini daha etkili bir şekilde yönetmelerine yardımcı olabilir ve süreçlerin sürekli iyileştirilmesini sağlar. Bu tür modeller, Scrum'ın pratiklerine rehberlik ederken, organizasyonel olgunluğu yükseltmek ve daha geniş bir yapı altında süreçlerin standardizasyonunu sağlamak için önemli bir araç sunmaktadır.

"24. Soru: CMMI-DEV V2.0 (Capability Maturity Model Integration for Development) hakkında bilginiz var mı?

a) Evet, detaylı bilgim var ve uygulamalarda yer aldım.
b) Evet, teorik düzeyde bilgi sahibiyim.
c) Adını duydum ama detaylarını bilmiyorum.
d) Hayır, hiç duymadım.
e) Emin değilim."

Yirmi dördüncü soruya verilen yanıtlar, CMMI-DEV V2.0 modeline ilişkin farkındalık düzeyinin katılımcı kuruluşlar arasında oldukça sınırlı olduğunu göstermektedir. Şekil 34'te görülen sonuçlarına göre yalnızca 2 kuruluş (%17) CMMI-DEV V2.0 hakkında detaylı bilgiye sahip olduğunu ve uygulamalarda yer aldığını belirtmiştir. Yine 2 kuruluş (%17) teorik düzeyde bilgi sahibi olduğunu ifade ederken, 4 kuruluş (%33) yalnızca adını duyduğunu ancak detaylarını bilmediğini; diğer 4 kuruluş (%33) ise bu modeli hiç duymadığını beyan etmiştir.



Şekil 34. Soru 24 – CMMI-DEV V2.0 modeline ilişkin farkındalık düzeyi

Bu bulgular, yazılım geliştirme süreçlerinde kalite, süreklilik ve olgunluk seviyesinin artırılması açısından önemli fırsatlar barındıran CMMI-DEV V2.0 modelinin, sektörde yeterince tanınmadığını ortaya koymaktadır. Modelin sahip olduğu kapsamlı süreç iyileştirme çerçevesi ve olgunluk düzeyleri, organizasyonların süreçlerini yapılandırarak sürdürülebilir başarı elde etmelerine katkı sunabilir. Ancak bu potansiyelin etkin biçimde değerlendirilebilmesi için, öncelikle CMMI-DEV V2.0'a yönelik farkındalığın artırılması ve modelin içerdiği uygulama alanlarının organizasyonlara tanıtılması gerekmektedir. Bu bağlamda, sektörel bilgilendirme çalışmaları ve modelin pratik uygulama örneklerinin yaygınlaştırılması, organizasyonların süreç olgunluğunu artırmada önemli bir adım olacaktır.

4. SONUÇLAR VE ÖNERİLER

4.1. Sonuçlar

Modern yazılım projelerinde yöntemsel çeşitlilik giderek artmakta, farklı ihtiyaçlara yanıt verebilecek esnek ve aynı zamanda denetlenebilir yaklaşımlara duyulan gereksinim belirginleşmektedir. Uygulamada karşılaşılan karmaşık problem alanları, yalnızca çeviklik ilkeleriyle değil, aynı zamanda yapısal çerçevelerle de desteklenen çözümler geliştirilmesini gerekli kılmaktadır. Bu bağlamda, çalışma kapsamında ele alınan yöntemsel birliktelik, mevcut uygulamaların değerlendirilmesi ve yeni bir yaklaşım önerisiyle bütünleştirilerek, yazılım geliştirme süreçlerinin farklı boyutlarıyla ele alınmasına olanak sağlamıştır.

Bu çalışma kapsamında, çevik yazılım geliştirme metodolojileri ile yetenek olgunluk modeli olan CMMI-DEV V2.0'ın entegrasyonu ele alınmıştır. Çalışmanın temel amacı, çevikliğin sağladığı esneklik ve adaptasyon yeteneklerini koruyarak; karar alma, risk yönetimi, süreç ölçümü ve sürekli iyileştirme gibi kurumsal düzeyde stratejik öneme sahip süreçleri güçlendiren bütüncül bir yaklaşım geliştirmektir. Bu doğrultuda öncelikle literatür taraması gerçekleştirilmiş, çevik yöntemlerin temel özellikleri ve sınırlılıkları değerlendirilmiş; ardından CMMI-DEV V2.0'ın uygulama alanları bu bağlamda analiz edilmiştir. Her iki yaklaşımın tamamlayıcı yönleri bir araya getirilerek, yapılandırılmış ve ölçülebilir süreçleri çevik metodolojilerin iteratif ve esnek yapısıyla uyumlu hale getiren hibrit bir model önerilmiştir. Bu model ile hem müşteri odaklılık, ekip içi etkileşim ve hızlı adaptasyon gibi çevik kazanımlar korunmakta; hem de süreçlerin daha tutarlı, izlenebilir ve sürekli gelişime açık bir yapıya kavuşturulması hedeflenmektedir.

Modelin sektörel uygulanabilirliğini ve geçerliliğini değerlendirmek amacıyla, Türkiye'de çeşitli sektörlerde faaliyet gösteren 12 yazılım organizasyonunun katılımıyla bir anket çalışması yürütülmüştür. Elde edilen veriler, organizasyonların özellikle karar alma, izlenebilirlik ve süreç yönetimi gibi alanlarda çeşitli zorluklarla karşılaştığını ortaya koymuştur. Bu durum, çevik yaklaşımların operasyonel çevikliği sağlamakta başarılı olsa da kurumsal düzeyde sürdürülebilir bir süreç olgunluğu için tek başına yeterli olmadığını ve CMMI-DEV'in rehberliğine duyulan ihtiyacı açık biçimde göstermektedir.

Anket bulguları, CMMI-DEV V2.0 kapsamındaki uygulama alanlarının birçok kuruluşta kısmen karşılandığını, ancak bütüncül ve kurumsal düzeyde uygulanma oranlarının oldukça sınırlı kaldığını ortaya koymuştur. Özellikle karar analizi (DAR), kök neden analizi (CAR), organizasyonel eğitim (OT), performans yönetimi ve ölçümü (MPM), süreç yönetimi (PCM) ve yönetim desteği (GOV) gibi kritik alanlarda yapılandırılmış süreçlerin eksikliği dikkat çekmektedir. Detaylı inceleme sonucunda; karar alma ve problem çözme süreçlerinde sistematik yaklaşımların yeterince benimsenmediği, eğitim ihtiyaçlarının kurumsal hedeflerle yeterince ilişkilendirilmediği, performans göstergelerinin düzenli biçimde izlenmediği ve analiz edilmediği, süreçlerin tanımlanması ve yönetiminde tutarlılığın sağlanamadığı ve üst yönetimin süreç iyileştirme faaliyetlerinde yeterince etkin rol almadığı gözlemlenmiştir. Bu eksiklikler, çevik uygulamaların kurumsal sürdürülebilirliğe dönüşebilmesi açısından CMMI-DEV V2.0 rehberliğinde yapılandırılmış bir yaklaşıma duyulan ihtiyacı daha da belirgin hale getirmektedir.

Ayrıca, süreç olgunluğu ve kalite yönetimi açısından kritik öneme sahip olan CMMI-DEV V2.0 gibi modeller konusunda sektörel farkındalık düzeyinin düşük olması, yapılandırılmış süreçlerin benimsenmesini ve yaygınlaştırılmasını zorlaştırmakta; bu da süreç iyileştirme çabalarının kurumsal düzeye taşınmasını engellemektedir. Bulgular, önerilen hibrit modelin yalnızca teorik değil, aynı zamanda pratik düzeyde de önemli bir boşluğu doldurma potansiyeline sahip olduğunu ortaya koymaktadır.

Bu bağlamda, önerilen hibrit model yalnızca teorik bir çerçeve sunmakla kalmamakta, aynı zamanda pratikte gözlemlenen boşlukları doldurma potansiyeli taşıyan uygulanabilir bir çözüm önerisi olarak değerlendirilmektedir. Elde edilen bulgular, çevik yöntemleri tek başına uygulayan organizasyonlar için CMMI-DEV V2.0 ile entegrasyonun bir tercih değil, sürdürülebilir ve olgun yazılım süreçleri oluşturabilmek adına bir gereklilik olduğunu ortaya koymuştur. Ayrıca, Türkçe literatürde bu iki yaklaşımın entegrasyonuna yönelik çalışmaların sınırlı olması, bu tezin hem akademik hem de sektörel düzeyde önemli bir boşluğu doldurduğunu göstermektedir.

Sonuç olarak, bu çalışma ile önerilen hibrit yapı, yazılım geliştirme süreçlerinin şeffaflığını artırma, performans takibini kolaylaştırma ve standartlaşmayı

sağlama açısından önemli katkılar sunmakta; özellikle Türkiye'de faaliyet gösteren yazılım organizasyonları için hem uygulanabilir hem de değerli bir çerçeve oluşturmaktadır. Model, çevik yaklaşımların esnekliğini, yapılandırılmış süreç olgunluğu ve sürekli iyileştirme pratikleriyle dengelerken, organizasyonların süreç kalitesini sistematik biçimde yükseltmelerine olanak tanımaktadır.

4.2. Öneriler

Gelecekte yapılacak çalışmalar, önerilen yaklaşımın farklı sektörlerde, kurumsal olgunluk seviyelerinde ve proje büyüklüklerinde uygulanabilirliğini ve etkinliğini daha kapsamlı şekilde irdeleyebilir. Ayrıca, vaka analizleri ve saha uygulamaları ile desteklenen ampirik veriler, modelin gerçek dünya süreçlerine entegrasyonunda karşılaşılan zorluklar ve başarı faktörleri hakkında somut bilgiler sağlayarak, uygulamadaki yol haritasını daha net ortaya koyacaktır. Böylelikle, hibrit modelin pratikteki faydaları ve sektörel etkileri daha derinlemesine kavranabilecektir.

KAYNAKLAR

[1] Yalçıner, B., Chouseinoglou, O., & Efe, Ö. M. (2022). Mapping CMMI-Dev v2. 0 with Scrum: A Project Management and Quality Assurance Perspective. *Available at SSRN 4310530*.

[2] Henriquez, V., Calvo-Manzano, J. A., Moreno, A. M., & San Feliu, T. (2022). Agile-CMMI V2. 0 alignment: Bringing to light the agile artifacts pointed out by CMMI. *Computer Standards & Interfaces*, *82*, 103610.

[3] de Sousa Morais, A. M. (2019). *Mapping CMMI Process Areas to Agile Best Practices* (Master's thesis, Universidade NOVA de Lisboa (Portugal)).

[4] Akbayır, D. (2010). Bir çevik yazılım geliştirme sürecinin uyarlanması ve uygulanması. *Yüksek Lisans Tezi*, Maltepe Üniversitesi, Fen Bilimleri Enstitüsü İstanbul, Türkiye.

[5] Agile Alliance. (2001). *Manifesto for Agile Software Development*. Retrieved February 2, 2020, from http://agilemanifesto.org

[6] Agile Alliance. (2001). *Principles behind the Agile Manifesto*. Retrieved February 2, 2020, from http://agilemanifesto.org/principles.html

[7] Kakar, A. K. (2023). A Rhetorical Analysis of the Agile manifesto on its 20th Anniversary. *The Journal of the Southern Association for Information Systems*, *10*(1), 20-29.

[8] Schwaber, K. (2004). Agile project management with Scrum. *Microsoft Press*.

[9] Beck, K. (2000). Extreme programming explained: embrace change. *Addison-Wesley Professional*.

[10] Stellman, A., & Greene, J. (2014). Learning agile: Understanding Scrum, XP, Lean, and Kanban. *O'Reilly Media, Inc.*

[11] Miletić, M., & Miletić, I. (2017). Lean methodology and its derivates usage for production systems in modern industry. *Appl. Eng. Lett*, *2*(4), 144-148.

[12] Sassa, A. C., de Almeida, I. A., Pereira, T. N. F., & de Oliveira, M. S. (2023). Scrum: A systematic literature review. *International Journal of Advanced Computer Science and Applications*, *14*(4).

[13] Ameta, U., Patel, M., & Sharma, A. K. (2021). Scrum Framework Based on Agile Methodology in Software Development and Management. In *Emerging Trends in Data Driven Computing and Communications: Proceedings of DDCIoT 2021* (pp. 321-332). Springer Singapore.

[14] Jaskyte, K., Hunter, A., & Mell, A. C. (2024). Predictors of interdisciplinary team innovation in higher education institutions. *Innovative Higher Education*, *49*(1), 113-132.

[15] Schwaber, K. (1997). *SCRUM Development Process* (pp. 117–134). Springer, London.

[16] Hron, M., & Obwegeser, N. (2018). Scrum in practice: an overview of Scrum adaptations. In *Hawaii International Conference on System Sciences* (pp. 4496-4505). Curran Associates, Inc.

[17] Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2017). Agile software development methods: Review and analysis. *arXiv preprint arXiv:1709.08439*.

[18] Yücalar, F. (2020). Kurumsal Bilgi Sistemi Uygulamaları İçin Popüler Proje Yönetim Yaklaşımları. *Geleceğin Dünyasında Bilimsel ve Mesleki Çalışmalar*, *Ekin Basım Yayın Dağıtım*, Bursa, ss.51-68.

[19] Sachdeva, S. (2016). Scrum methodology. *Int. J. Eng. Comput. Sci*, *5*(16792), 16792-16800.

[20] Srivastava, A., Bhardwaj, S., & Saraswat, S. (2017, May). SCRUM model for agile methodology. In *2017 International Conference on Computing, Communication and Automation (ICCCA)* (pp. 864-869). IEEE.

[21] Sedano, T., Ralph, P., & Péraire, C. (2019, May). The product backlog. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)* (pp. 200-211). IEEE.

[22] Kopczyńska, S., Ochodek, M., Piechowiak, J., & Nawrocki, J. (2022). On the benefits and problems related to using Definition of Done—A survey study. *Journal of Systems and Software*, *193*, 111479.

[23] Sauer, S., Nicklich, M., & Pfeiffer, S. (2021). The Agile Imperative: A Multi-Level Perspective on Agility as a New Principle of Organizing Work. *The Agile Imperative: Teams, Organizations and Society under Reconstruction?*, 1-15.

[24] Alostad, J. M., Abdullah, L. R., & Aali, L. S. (2017). A fuzzy based model for effort estimation in scrum projects. *International Journal of Advanced Computer Science and Applications*, *8*(9).

[25] Pendyala, V. (2024). Leveraging information & data visualization in agile software development. *International Journal of Computer Engineering and Technology (IJCET), 15*(5), 863–873.

[26] Jégou, O., & Souayah, F. (2021). What Do Workers Get Out of Agility? Examining Workers' Capability for Democratic Self-Government of Work. *The Agile Imperative: Teams, Organizations and Society under Reconstruction?*, 203-224.

[27] Van Deursen, A. (2001, October). Program comprehension risks and opportunities in extreme programming. In *Proceedings Eighth Working Conference on Reverse Engineering* (pp. 176-185). IEEE.

[28] Cohn, T., & Paul, R. (2001). A comparison of requirements engineering in extreme programming (XP) and conventional software development methodologies. *AMCIS 2001 Proceedings*, 256.

[29] Bahli, B., & Zeid, E. S. A. (2005, December). The role of knowledge creation in adopting extreme programming model: an empirical study. In *2005 International Conference on Information and Communication Technology* (pp. 75-87). IEEE.

[30] Wood, W. A., & Kleb, W. L. (2002, August). Extreme programming in a research environment. In *Conference on Extreme Programming and Agile Methods* (pp. 89-99). Berlin, Heidelberg: Springer Berlin Heidelberg.

[31] Akhtar, A., Bakhtawar, B., & Akhtar, S. (2022). Extreme programming vs scrum: A comparison of agile models. *International Journal of Technology Innovation and Management (IJTIM)*, *2*(2), 80-96.

[32] Borandağ, E., Yücalar, F., & Eren, Ş., (2010). An Incomplete Software Project s Accomplishment by the Aid of Extreme Programming Methodology . The 1st International Symposium on Computing in Science Engineering (ISCSE 2010) (pp.396-400). Aydın, Turkey

[33] Fojtik, R. (2011). Extreme Programming in development of specific software. *Procedia Computer Science*, *3*, 1464-1468.

[34] Poppendieck, M., & Cusumano, M. A. (2012). Lean software development: A tutorial. *IEEE software*, *29*(5), 26-32.

[35] Hibbs, C., Jewett, S., & Sullivan, M. (2009). *The art of lean software development: a practical and incremental approach*. " O'Reilly Media, Inc.".

[36] Kumar, D. R. (2005). Lean software development. *The project perfect white paper collection*.

[37] Rodríguez, P., Mäntylä, M., Oivo, M., Lwakatare, L. E., Seppänen, P., & Kuvaja, P. (2019). Advances in using agile and lean processes for software development. In *Advances in computers,* vol. 113, pp. 135-224). Elsevier.

[38] Poppendieck, M., & Poppendieck, T. (2003). Lean software development: An agile toolkit: An agile toolkit. *Addison-Wesley.*

[39] Poppendieck, M., & Poppendieck, T. (2009). Leading lean software development: Results are not the point. *Pearson Education.*

[40] Radin Umar, R. Z., Tiong, J. Y., Ahmad, N., & Dahalan, J. (2024). Development of framework integrating ergonomics in Lean's Muda, Muri, and Mura concepts. *Production Planning & Control*, *35*(12), 1466-1474.

[41] Ahmad, M. O., Markkula, J., & Oivo, M. (2013, September). Kanban in software development: A systematic literature review. In *2013 39th Euromicro*

conference on software engineering and advanced applications (pp. 9-16). IEEE.

[42] Kniberg, H., & Skarin, M. (2010). Kanban and Scrum-making the most of both. *Lulu.com.*

[43] Corona, E., & Pani, F. E. (2013). A review of lean-kanban approaches in the software development. *WSEAS transactions on information science and applications*, *10*(1), 1-13.

[44] Ladas, C. (2009). Scrumban-essays on kanban systems for lean software development. *Lulu.com.*

[45] Vasylieva, K., Kuhrmann, M., Xavier, M. K., & Klünder, J. (2023, September). How agile are you? discussing maturity levels of agile maturity models. In *2023 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (pp. 270-277). IEEE.

[46] Yeh, K. B., Adams, M. L., Marshall, E. S., Dasgupta, D., Zhunushov, A., Richards, A. L., & Hay, J. (2017). Applying a Capability Maturity Model (CMM) to evaluate global health security-related research programmes in under-resourced areas. *Global Security: Health, Science and Policy*, *2*(1), 1-9.

[47] Paulk, M. C. (2009). A history of the capability maturity model for software. *ASQ Software Quality Professional*, *12*(1), 5-19.

[48] Chang, G. S., Perng, H. L., & Juang, J. N. (2008). A review of systems engineering standards and processes. *Journal of Biomechatronics Engineering*, *1*(1), 71-85.

[49] Cooper, J., & Fisher, M. (2002). Software Acquisition Capability Maturity Model (SA-CMM) Version 1.03. (Technical Report CMU/SEI-2002-TR-010). Retrieved June 18, 2025.

[50] Curtis, B., Hefley, B., & Miller, S. (2009). People capability maturity model (P-CMM) version 2.0. *Software Engineering Institute*, *18*, 1-533.

[51] Bate, R., Gibson, D., & Richter, K. (2001). CMMI and Integrated Product and Process Development (IPPD).

[52] Paulk, M. C. (1994). *A Comparison of ISO 9001 and the Capability Maturity Model for Software*. Pittsburgh, PA, USA: Carnegie Mellon University, Software Engineering Institute.

[53] Guerrero, F., & Eterovic, Y. (2004). Adopting the SW-CMM in a Small IT Organization. *IEEE software*, *21*(4), 29-35.

[54] Khraiwesh, M. (2020). Measures of Organizational Training in the Capability Maturity Model Integration (CMMI®). *International Journal of Advanced Computer Science and Applications*, *11*(2), 584-592.

[55] Chrissis, M. B., Konrad, M., & Shrum, S. (2011). *CMMI for development: guidelines for process integration and product improvement*. Pearson Education.

[56] Ayyagari, M. R., & Atoum, I. (2019). CMMI-DEV implementation simplified. *International Journal of Advanced Computer Science and Applications*, *10*(4).

[57] Kundu, G. K., & Murali Manohar, B. (2012). A unified model for implementing lean and CMMI for Services (CMMI-SVC v1. 3) best practices. *Asian Journal on Quality*, *13*(2), 138-162.

[58] Forrester, E., Buteau, B., & Shrum, S. (2011). *CMMI for services: guidelines for superior service*. Pearson Education.

[59] Gallagher, B., Phillips, M., Richter, K., & Shrum, S. (2008). CMMI-ACQ: guidelines for improving the acquisition of products and services. *Pearson Education.*

[60] Yücalar, F., Şahinaslan, E., Borandağ, E., & Şahinaslan, Ö. (2010). Yazılım Yöneticileri için Tümleşik Yetenek Olgunluk Modeli: Genel Bir Bakış. *Akademik Bilişim'10*, 306.

[61] Keshta, I. (2022). A model for defining project lifecycle phases: Implementation of CMMI level 2 specific practice. *Journal of King Saud University-Computer and Information Sciences*, *34*(2), 398-407.

[62] Alparslan, S. A. (2017). CMMI ile yazılım süreçlerinin iyileştirilmesi ve yazılım şirketlerinin CMMI 3 seviyesine göre değerlendirilmesi, *Yüksek Lisans Tezi*, Fen Bilimleri Enstitüsü, Akdeniz Üniversitesi, Antalya.

[63] Rout, T. P., & Tuffley, A. (2007). Harmonizing iso/iec 15504 and cmmi. *Software Process: Improvement and Practice*, *12*(4), 361-371.

[64] Chaudhary, M., & Chopra, A. (2016). CMMI for development: Implementation guide. *APress*.

[65] Ojha, T. R. Role of Capability Maturity Model Integration (CMMI) in Software Process Improvement. *Journal of Advancement in Software Engineering and Testing*, *5*(2).

[66] Miyashiro, M. A. S., & Ferreira, M. G. (2016, July). Factors to be considered for the adaptation of "cyclic process" (CMMI-DEV level 2) – In the development of embedded components. In *2016 SAI Computing Conference (SAI)* (pp. 1178-1185). IEEE.

[67] Hadyan, N. N., Budiardjo, E. K., Alqadri, Y., & Ferdinansyah, A. (2019, January). Evaluation of capability level and improvements prioritization on device accreditation services based on CMMI-SVC framework continuous representation. In *Proceedings of the 2019 Asia Pacific Information Technology Conference* (pp. 84-90).

[68] Kneuper, R. (2018). Software process assessment and improvement. In *Software Processes and Life Cycle Models: An Introduction to Modelling, Using and Managing Agile, Plan-Driven and Hybrid Processes* (pp. 211-260). Cham: Springer International Publishing.

[69] Triana, Y. S. (2021). Process improvement software through assessment using CMMI framework.

[70] Yücalar, F., & Erdoğan, Ş. Z., (2008). CMMI Basamaklı Modeli ile Yazılım Firmalarının Değerlendirilmesi için Bir Yöntem ve Uygulama. Yazılım Kalitesi ve Yazılım Geliştirme Araçları Sempozyumu (YKGS'2008) (pp.53-57). İstanbul, Turkey

[71] Van Sickle, M. (2006). *Transitioning from the Software Capability Maturity Model to the Capability Maturity Model Integrated* (Doctoral dissertation).

[72] Kitson, D. H., Vickroy, R., Walz, J., & Wynn, D. (2009). An initial comparative analysis of the CMMI version 1.2 development constellation and the ISO 9000 family.

[73] Team, CMMI Product (2018). CMMI for Development, Version 1.3. Carnegie Mellon University. Report. https://doi.org/10.1184/R1/6572342.v1

[74] Al-Tarawneh, M. Y. (2013). *Harmonizing CMMI-DEV 1.2 and XP Method to Improve The Software Development Processes in Small Software Development Firms* (Doctoral dissertation, Universiti Utara Malaysia).

[75] Mosquera Díaz, J., Ocampo Cortés, O., & García Monsalve, L. S. (2014). Functional Prototype of a Web Information System to Assist Planning Software Projects, Based on CMMI-DEV 1.2. *Tecciencia*, *9*(17), 67-77.

[76] Phillips, M., & Shrum, S. (2010). Process improvement for all: what to expect from CMMI Version 1.3. *Crosstalk–The Journal of Defense Software Engineering*.

[77] Team, C. P. (2018). CMMI for Development, Version 1.3 (Version 1). Carnegie Mellon University.

[78] Software Engineering Institute. (2010). CMMI V1.3: Planned improvements. Carnegie Mellon University.

[79] Proença, D., & Borbinha, J. (2018). Formalizing ISO/IEC 15504-5 and SEI CMMI v1. 3–Enabling automatic inference of maturity and capability levels. *Computer Standards & Interfaces*, *60*, 13-25.

[80] Valverde, L., Mira da Silva, M., & Gonçalves, M. R. (2018, October). CMMI-DEV v1. 3 reference model in ArchiMate. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* (pp. 191-208). Cham: Springer International Publishing.

[81] Husni, M. (2024). CMMI V2. 0 Maturity Level 2 and Scrum Applicability in Jordanian Agile Companies Based on Expert Review.

[82] Balla, K., Tang, M., Mowat, P., Rasking, M., Chaobo, S., van Veenendaal, E., & Hongbao, Z. (2020). Changes in CMMI 2.0 and how they can affect TMMi. *TMMi Foundation, Bulverde, TX, USA, Tech. Rep*.

[83] Lazzaris, J., Silva, M., Pereira, T. F., Faria, V., Compadrinho, P., & Machado, R. J. (2024). Visualization of Information Through Complex Networks–An Applied Case of CMMI and OpenUp Alignment. *Procedia Computer Science*, *239*, 743-750.

[84] Ariffin, K. A. Z., & Ahmad, F. H. (2021). Indicators for maturity and readiness for digital forensic investigation in era of industrial revolution 4.0. *Computers & Security*, *105*, 102237.

[85] Marappa, K. (2018). CMMI Development V2.0 – for Improving Business Performance.(https://drive.google.com/file/d/13k6uvcAxCPkrj5NeudR4OmPkwVcUAuXW/view)

[86] Al-Matouq, H., Mahmood, S., Alshayeb, M., & Niazi, M. (2020). A maturity model for secure software design: a multivocal study. *IEEE Access*, *8*, 215758-215776.

[87] Omar, M., Helmy, Y., & Farid, A. B. (2020). Supplier Qualification Model (SQM): A Quantitative Model for Supplier Agreements Evaluation. *International Journal of Advanced Computer Science and Applications*, *11*(11).

[88] da Cunha, J. Y. C., & Oliveira, S. R. B. (2024, January). A guidance for implementing the Planning and Managing Work capability area of CMMI 2.0.

In 20th CONTECSI-International Conference on Information Systems and Technology Management Virtual.

[89] Russo, N., & Reis, L. (2020). Updated analysis of business continuity issues underlying the certification of invoicing software, considering a pandemic scenario. *Advances in Science, Technology and Engineering Systems Journal*, *5*(6), 845-852.

[90] ISACA. (2023). February *2020* Quality Tip – Appraising the Supporting Implementation Capability Area. *Online:* https://cmmiinstitute.com/resource-files/public/quality/quality-corner/february-2020-quality-tip-appraising-the-supportin

[91] Degerli, M. (2020, September). Practical Suggestions to Successfully Adopt the CMMI V2. 0 Development for Better Process, Performance, and Products. In *2020 5th International Conference on Computer Science and Engineering (UBMK)* (pp. 126-129). IEEE.

[92] Mondragon, O. A., & Jasso, R. (2019, July). INCOSE SE Handbook V4 to CMMI V2. 0 Development Mapping. In *INCOSE International Symposium* (Vol. 29, No. 1, pp. 723-737).

[93] https://www.cmmiinstitute.com/

[94] Oyshi, F., Bonik, A., Shin, M. O. H., Rashid, J., Akter, M., Hasan, M., & Sadia, F. (2023). A Novel model to adapt CMMI Level 2 by Assessing the Local SMEs of Bangladesh. *Procedia Computer Science*, *219*, 2043-2050.

[95] Gonçalves, T. G., Oliveira, K. M., & Kolski, C. (2018). Identifying HCI approaches to support CMMI-DEV for interactive system development. *Computer Standards & Interfaces*, *58*, 53-86.

[96] Jansson, A. S. (2007). Software maintenance and process improvement by CMMI. *UPTEC STS07037 November Examensarbete*, *20*.

[97] Fariz, A. A., Raharjo, T., & Genia, V. (2023). ISO 9001: 2015 and Capability Maturity Model Integration 3.0 in Software Development Project. *The Indonesian Journal of Computer Science*, *12*(6).

[98] Broadsword Solutions Corporation. (2019). *CMMI-DEV V1.3® to CMMI Development V2.0® crosswalk*. https://broadswordsolutions.com/wp-content/uploads/2019/06/CMMI-V2-Crosswalk-v.4.pdf

[99] Henriquez, V., Moreno, A. M., & Gutiérrez, S. (2023). Organizational training in agile settings under the lens of CMMI V2. 0. *Journal of Software: Evolution and Process*, *35*(9), e2502.

[100] Kumar, U. (2020). A Journey of CMMI Level-3 Model Implementation in Product-Centric Agile organization.

[101] Waina, R. (2019). The new CMMI® V2.0 enhances capability improvement. Multi-Dimensional Maturity. Copyright 2018, CMMI Institute. https://static.spacecrafted.com/eff8f1444ff547dc97bb98fe24e32d2d/r/bcadbce7d8524899a4eeeba71308c14c/1/CMMI%20V2.0%20Overview.pdf

[102] Degerli, M. (2020, October). Crafting a CMMI V2 Compliant Process for Governance Practice Area: An Experiential Proposal. In *2020 Turkish National Software Engineering Symposium (UYMS)* (pp. 1-3). IEEE.

[103] Henriquez, V., Calvo-Manzano, J. A., Moreno, A. M., & San Feliu, T. (2025). Agile governance practices by aligning CMMI V2. 0 with portfolio SAFe 5.0. *Computer Standards & Interfaces*, *91*, 103881.

[104] Sanchez-Segura, M. I., Ruiz-Robles, A., Medina-Dominguez, F., & Dugarte-Peña, G. L. (2017). Strategic characterization of process assets based on asset quality and business impact. *Industrial Management & Data Systems*, *117*(8), 1720-1737.

[105] Digital.ai. (2023). *17th State of Agile Report* (pp. 1–26). https://stateofagile.com/#ufh-i-661275008-17th-state-of-agile-report/7027494

[106] Esher, J. J. (2019). *Agile Methodology Tailoring Under CMMI®*. Wilmington University (Delaware).

[107] Garcia, L. A., OliveiraJr, E., Leal, G. C. L., & Morandini, M. (2020, November). A unified feature model for Scrum artifacts from a literature and practice perspective. In *Escola Regional de Engenharia de Software (ERES)* (pp. 296-305). SBC.

[108] Dalton, J., Timmerman, R., Adkins, L., Botula, K., Potter, N., Torrens, D., & Glover, M. T. (2016). A guide to Scrum and CMMI: Improving agile performance with CMMI. *CMMI Institute*, 1-130.

[109] Carlson, D., & Soukup, E. (2017). Combining Agile Practices with CMMI Process Areas. *CrossTalk*, 17.

[110] Henriquez, V., Moreno, A. M., Calvo-Manzano, J. A., & San Feliu, T. (2021). Agile–CMMI alignment: Contributions and to-dos for organizations. *Computer*, *54*(12), 38-49.

[111] Joembunthanaphong, P., & Sriharee, G. (2022, December). The improvement process for the software development and requirements management to achieve capability level 3 of CMMI. In *2022 26th International Computer Science and Engineering Conference (ICSEC)* (pp. 296-301). IEEE.

[112] Park, D. S., & Noh, J. Y. (2023, October). The Effect of Sprint Duration to the Velocity in a Large-Scale Embedded Software Project. In *2023 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)* (pp. 1-5). IEEE.

[113] Alqadri, Y., Budiardjo, E. K., Ferdinansyah, A., & Rokhman, M. F. (2020, January). The CMMI-dev implementation factors for software quality improvement: a case of XYZ corporation. In *Proceedings of the 2020 2nd Asia Pacific Information Technology Conference* (pp. 34-40).

[114] Albuquerque, R., Santos, G., Malucelli, A., & Reinehr, S. (2020, December). Abandonment of a Software Process Improvement Program: Insights from

Case Studies. In *Proceedings of the XIX Brazilian Symposium on Software Quality* (pp. 1-10).

[115] Foegen, M., & Croome, D. (2011). How Scrum helps with CMMI. *wibas GmbH.*

[116] Sreenivasan, S., & Kothandaraman, K. (2019). Improving processes by aligning capability maturity model integration and the scaled agile framework®. *Global Business and Organizational Excellence*, *38*(6), 42-51.

[117] Amer, S. K., Badr, N., & Hamad, A. (2019, March). Combining CMMI specific practices with Scrum model to address shortcomings in process maturity. In *International Conference on Advanced Machine Learning Technologies and Applications* (pp. 898-907). Cham: Springer International Publishing.