ŞELALE MODELİ VE ÇEVİK YÖNTEMLERİN YAZILIM PROJE YÖNETİMİ BAKIŞ AÇISIYLA İNCELENMESİ VE KARŞILAŞTIRILMASI

Ali KAYACAN

YÜKSEK LİSANS TEZİ
YÖNETİM BİLİŞİM SİSTEMLERİ ANABİLİM DALI

GAZİ ÜNİVERSİTESİ
BİLİŞİM ENSTİTÜSÜ

EYLÜL 2017

Ali KAYACAN tarafından hazırlanan "ŞELALE MODELİ VE ÇEVİK YÖNTEMLERİN YAZILIM PROJE YÖNETİMİ BAKIŞ AÇISIYLA İNCELENMESİ VE KARŞILAŞTIRILMASI" adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ ile Gazi Üniversitesi Yönetim Bilişim Sistemleri Anabilim Dalında YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Danışman: Prof. Dr. Cevriye GENCER
Endüstri Mühendisliği, Gazi Üniversitesi
Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum ……..

Başkan : Prof. Dr. Ergün ERASLAN
Endüstri Mühendisliği, Yıldırım Beyazıt Üniversitesi
Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum ……

Üye : Doç. Dr. Hacer KARACAN
Bilgisayar Mühendisliği, Gazi Üniversitesi
Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum ……

Tez Savunma Tarihi: 14/09/2017

Jüri tarafından kabul edilen bu tezin Yüksek Lisans Tezi olması için gerekli şartları yerine getirdiğini onaylıyorum.

……
Doç. Dr. Bünyamin CİYLAN
Bilişim Enstitüsü Müdürü

ETİK BEYAN

Gazi Üniversitesi Bilişim Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Ali KAYACAN
14.09.2017

ŞELALE MODELİ VE ÇEVİK YÖNTEMLERİN YAZILIM PROJE YÖNETİMİ BAKIŞ AÇISIYLA İNCELENMESİ VE KARŞILAŞTIRILMASI
(Yüksek Lisans Tezi)

Ali KAYACAN

GAZİ ÜNİVERSİTESİ
BİLİŞİM ENSTİTÜSÜ
Eylül 2017

ÖZET

Günümüzde yazılım, günlük eğlencemizden en karmaşık iş süreçlerine kadar hayatımızın her alanında yerini almış ve etkileri hissedilen bir olgudur. Şirketler, iş süreçlerinin yönetimi için etkin yazılım sistemlerine ihtiyaç duyarlar ve bu sistemleri bilgi sistem departmanları vasıtasıyla üretir ya da yazılım şirketlerinden tedarik ederler. Her iki durumda da yazılımın üretilmesi bir dizi mühendislik ve proje yönetim sürecini gerektirir. Bu süreçlerin metodoloji seçiminde, üretilecek yazılımın büyüklüğü, karmaşıklığı ve kullanım maksadı gibi karakteristik özellikleri, yazılımı üretecek organizasyon ve proje ekibinin yapısı ve kullanılabilecek kaynaklar önem arz eder. Bu çalışmada, yazılım proje başarısına etki ettiği değerlendirilen faktörler temel alınarak bir inceleme ve karşılaştırma ortaya konulmuş; yazılım proje yönetimi metodolojisi seçimi konusunda önerilerde bulunulmuştur. Bu karşılaştırmalarda, 12 durumda çevik yöntemlerin; 9 durumda şelale modelinin avantajlı olduğu değerlendirilmiştir. Elde edilen sonucun, deneysel tecrübe ile karşılaştırılması maksadıyla yazılım sektöründen 97 çalışanın katılımıyla bir anket uygulanmıştır. Anket sonucunda, 1 kriterde şelale modeli tercih edilirken, 48 kriterde çevik yöntemler tercih edilmiştir. Elde edilen cevapların ağırlıkları kullanılarak metodolojiler açısından genel bir karşılama oranı hesaplanmıştır. Metodoloji seçimini etkilediği değerlendirilen kriterlerin karşılama oranları çevik yöntemler için %67,4; şelale modeli için % 51,91 olarak elde edilmiştir.

Bilim Kodu : 114609
Anahtar Kelimeler : Şelale modeli, çevik yöntemler, yazılım proje yönetimi
Sayfa Adedi : 104
Danışman : Prof. Dr. Cevriye GENCER

ANALYSIS AND COMPARISON OF WATERFALL MODEL AND AGILE METHODOLOGIES WITH THE PERSPECTIVE OF SOFTWARE PROJECT MANAGEMENT

(M. Sc. Thesis)

Ali KAYACAN

GAZİ UNIVERSITY
INSTITUTE OF INFORMATICS
September 2017

ABSTRACT

In our time, software is a fact, the influence of which is strongly felt and has attained its place in our lives from daily entertainment to the most complex business processes. Corporations need efficient software systems in order to manage their business processes and they either produce these systems through the agency of their information systems departments or procure from software companies. In both situations, the production of software requires a series of engineering and project management processes. In selection of methodology for these processes, the characteristics of software such as size, complexity and purpose of use, the structure of the project team and organization which are to produce the software and available sources are important. In this study, an analysis and a comparison of waterfall model and agile methodologies based on factors which are evaluated to affect the success of software projects are presented and propsals about selecting a methodology for software project management are submitted. In these comparisons, waterfall model is found to be advantageous in 9 situation and agile methodologies in 12 situations. In order to compare the reults obtained. a survey is organized with the participation of 97 employees in software industry. In survey results, waterfall model is favoured in 1 criteria while agile methodologies in the rest 48. A general coverage ratio is calculated using the weights of the criteria. The coverage ratios of criteria which are evaluated to affect the success of software projects are % 51,91 for waterfall model and %67,4 for agile methodologies.

Science Code : 114609
Key Words : Waterfall model, agile methodologies, software project management
Page Number : 104
Supervisor : Prof. Dr. Cevriye GENCER

TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesinde değerli bilgilerini benimle paylaşan, yönlendirmeleriyle çalışmayı şekillendiren danışman hocam Prof. Dr. Cevriye GENCER'e ve çalışma süresince tüm zorlukları benimle göğüsleyen ve bana destek olan eşim Seda ve kızım Irmak KAYACAN'a teşekkürlerimi sunarım.

İÇİNDEKİLER

Sayfa

ÖZET iv

ABSTRACT v

TEŞEKKÜR vi

İÇİNDEKİLER vii

ÇİZELGELERİN LİSTESİ xi

ŞEKİLLERİN LİSTESİ xii

SİMGELER VE KISALTMALAR xiii

1. GİRİŞ 1

2. PROJE YÖNETİMİ 3

2.1. Proje Nedir 3

2.2. Proje Yönetimi Kavramı ve Gelişimi 3

2.3. Proje Yönetiminin Faydaları 4

2.4. Proje Yönetiminin Dezavantajları 5

2.5. Proje Yönetim Süreçleri 5

2.5.1. Başlangıç süreçleri 6

2.5.2. Planlama süreçleri 6

2.5.3. Yürütme süreçleri 7

2.5.4. İzleme ve kontrol süreçleri 7

2.5.5. Kapanış süreçleri 7

2.6. Proje Yönetimi Bilgi Alanları 8

2.6.1. Proje entegrasyon yönetimi 8

2.6.2. Proje kapsam yönetimi 9

2.6.3. Proje zaman yönetimi 9

2.6.4. Proje maliyet yönetimi 10

2.6.5. Proje kalite yönetimi 10

2.6.6. Proje insan kaynakları yönetimi 10

2.6.7. Proje iletişim yönetimi 11

2.6.8. Proje risk yönetimi 11

2.6.9. Proje tedarik yönetimi 12

Sayfa

2.6.10. Proje paydaş yönetimi 12
2.7. Proje Yönetim Süreçleri ile Proje Yönetim Bilgi Alanlarının İlişkisi 13

3. YAZILIM MÜHENDİSLİĞİ VE YAZILIM PROJE YÖNETİMİ 15
3.1. Yazılım ve Yazılım Mühendisliği 15
3.2. Yazılım Mühendisliği Bilgi Birikimi Kılavuzu 15
3.3. Yazılım Mühendisliği Bilgi Alanları 17
3.3.1. Yazılım gereksinimleri 17
3.3.2. Yazılım tasarımı 18
3.3.3. Yazılımın oluşturulması 18
3.3.4. Yazılım testi 19
3.3.5. Yazılımın bakımı 19
3.3.6. Yazılım konfigürasyon yönetimi 20
3.3.7. Yazılım mühendisliği yönetimi 20
3.3.8. Yazılım mühendisliği süreci 21
3.3.9. Yazılım mühendisliği modelleri ve yöntemleri 21
3.3.10. Yazılım kalitesi 22
3.3.11. Yazılım mühendisliği mesleki uygulaması 22
3.3.12. Yazılım mühendisliği ekonomisi 22
3.3.13. Bilgisayar temelleri 23
3.3.14. Matematik temelleri 23
3.3.15. Mühendislik temelleri 24
3.4. Yazılım Proje Yönetimi 24
3.5. Yazılım Proje Yönetimi Süreçleri 26
3.5.1. Projenin başlatılması ve kapsam tanımlaması 27
3.5.2. Yazılım projesinin planlaması 27
3.5.3. Yazılım projesinin yürütülmesi 29
3.5.4. Gözden geçirme ve değerlendirme 31
3.5.5. Kapanış 31

4. YAZILIM GELİŞTİRME METODOLOJİLERİ 33
4.1. Yazılım Yaşam Çevrimi 33
4.2. Şelale Modeli 33

Sayfa

4.2.1. Tarihçesi 33
4.2.2. İlkeleri ve özellikleri 34
4.2.3. Faydaları 36
4.2.4. Kısıtları 37
4.3. Çevik Yöntemler 38
4.3.1. Tarihçesi 39
4.3.2. İlkeleri ve özellikleri 39
4.3.3. Faydaları 43
4.3.4. Kısıtları 44

5. ŞELALE MODELİ VE ÇEVİK YÖNTEMLERİN KARŞILAŞTIRILMASI 47
5.1. Benzer Çalışmalar 48
5.2. Benzerlikler ve Farklılıklar 50
5.3. Maliyetler 52
5.4. Dış Kaynak Kullanımı 52
5.5. Yazılım Kalitesi 53
5.5.1. Yazılımın genel nitelikleri 53
5.5.2. Kalite güvence ve hata toleransı 54
5.5.3. Değer üretimi 54
5.5.4. Kullanıcı deneyimi 55
5.5.5. Risk yönetimi 55
5.6. Analiz Tasarım Kalitesi 57
5.6.1. Görünürlük 57
5.6.2. Karmaşıklık 58
5.6.3. Proje büyüklüğü 59
5.6.4. Dokümantasyon ve modelleme 61
5.6.5. Yeniden kullanılabilirlik 61
5.6.6. Gereksinimler 62
5.7. Teslimatlar 64
5.8. Zaman Planlaması 65
5.9. Dış Etkenler 65
5.9.1. Müşteriler 65

Sayfa

5.9.2. Kullanım onayı 66
5.10. İç Etkenler 66
5.10.1. Yönetim 66
5.10.2. Organizasyon yapısı 66
5.11. Takım 68
5.11.1. Roller 69
5.11.2. Tecrübe ve yetenek 69
5.11.3. Takım büyüklüğü 69
5.11.4. Alan bilgisi 70
5.11.5. İş birliği 71
5.11.6. Yerleşim 71
5.11.7. Karmaşıklık 71
5.11.8. Ekip çalışması 72
5.12. Karşılaştırma Özeti 73

6. UYGULAMA 75
6.1 Anket Uygulama Sonuçları 78
6.2 Uygulama Sonuçlarının Analizi 80

7. SONUÇ 89

KAYNAKLAR 93

EKLER 99

EK-1. Uygulanan Anket Örneği 100

ÖZGEÇMİŞ 103

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

Çizelge 2.1. Proje yönetim süreçleri ile proje yönetim bilgi alanlarının ilişkisi..... 13

Çizelge 3.1. 2011-2015 CHAOS raporları başarı oranları 25

Çizelge 3.2. CHAOS raporları başarı faktörleri…….. 26

Çizelge 4.1. Başlıca çevik yöntemlerin uygulanma oranları 42

Çizelge 5.1. Uygulanan metodolojiye göre yazılım projeleri başarı oranları..... 47

Çizelge 5.2. Risk türleri ve açıklamaları 56

Çizelge 5.3. Yazılım projelerinin başarı oranlarının proje büyüklüğüne göre dağılımı 60

Çizelge 5.4. Proje büyüklüğüne ve uygulanan metodolojiye göre yazılım projeleri başarı oranları 60

Çizelge 5.5. Karşılaştırma özeti..... 73

Çizelge 6.1. Birinci seviye faktörlerin ağırlık katsayıları..... 75

Çizelge 6.2. İkinci seviye faktörlerin ağırlık katsayıları 76

Çizelge 6.3. Üçüncü seviye faktörlerin ağırlık katsayıları 77

Çizelge 6.4. Diğer seçeneğini işaretleyen katılımcıların görev tanımları 78

Çizelge 6.5. Metodoloji bazında cevap ortalamaları..... 81

Çizelge 6.6. Şelale modeli ikinci seviye kriterlerin karşılanma oranları 83

Çizelge 6.7. Şelale modeli birinci seviye kriterlerin karşılanma oranları..... 84

Çizelge 6.8. Çevik yöntemler ikinci seviye kriterlerin karşılanma oranları 85

Çizelge 6.9. Çevik yöntemler birinci seviye kriterlerin karşılanma oranları..... 86

Çizelge 6.10. İkinci seviye kriterlerin karşılanma oranları karşılaştırması 86

Çizelge 6.11. Birinci seviye kriterlerin karşılanma oranları karşılaştırması..... 87

Çizelge 6.12. Şelale modeli ve çevik yöntemlerin nihai karşılama oranları 87

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil 4.1. Şelale modelinde uygulanan safhalar 34
Şekil 4.2. Organizasyonlarda uygulanan geliştirme yöntemleri 39
Şekil 4.3. Çevik yöntemlerin genel uygulaması 41
Şekil 5.1. Şelale modeli süreç akışı 51
Şekil 5.2. Çevik yöntemler süreç akışı 51
Şekil 5.3. Değer üretimi karşılaştırması 55
Şekil 5.4. Risk karşılaştırması 57
Şekil 5.5. Görünürlük karşılaştırması 58
Şekil 5.6. Karmaşıklık seviyesine göre yazılım sistemleri 58
Şekil 5.7. Proje sürecinde karmaşıklık karşılaştırması 59
Şekil 5.8. İhtiyaca yabancılaşma karşılaştırması 62
Şekil 5.9. Değişiklik maliyeti karşılaştırması 63
Şekil 5.10. Değişikliğe açıklık karşılaştırması 63
Şekil 5.11. Takım büyüklüğü etkisi 70
Şekil 6.1. Katılımcıların cinsiyet dağılımı 78
Şekil 6.2. Katılımcıların görev tanımı dağılımı 79
Şekil 6.3. Katılımcıların eğitim durumu dağılımı 79
Şekil 6.4. Katılımcıların şelale modeli tecrübe durumu dağılımı 80
Şekil 6.5. Katılımcıların çevik yöntemler tecrübe durumu dağılımı 80

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılan kısaltmalar açıklamaları ile birlikte aşağıda sunulmuştur.

Kısaltmalar	Açıklama
AHP	Analitik Hiyerarşi Prosesi
AM	Çevik Modelleme (Agile Modeling)
ASD	Adaptif Yazılım Geliştirme (Adaptive Software Development)
AUP	Çevik Birleştirilmiş Süreç (Agile Unified Process)
DSDM	Dinamik Sistem Geliştirme Metodolojisi (Dynamic System Development Methodology)
ERP	Kurumsal Kaynak Planlama (Enterprise Resource Planning)
FDD	Özellik Güdümlü Geliştirme (Feature Driven Development)
IEEE	Elektrik ve Elektronik Mühendisleri Enstitüsü (Institute of Electrical and Electronics Engineers)
LSD	Yalın Yazılım Geliştirme (Lean Software Development)
PMBOK®	Proje Yönetimi Bilgi Birikimi (Project Management Body of Knowledge)
SDLC	Yazılım Geliştirme Yaşam Çevrimi (Software Development Life Cycle)
SPLC	Yazılım Ürün Yaşam Çevrimi (Software Product Life Cycle)
SWEBOK	Yazılım Mühendisliği Bilgi Birikimi (Software Engineering Body of Knowledge)
XP	Ekstrem Programlama (Extreme Programming)

1. GİRİŞ

Günümüzde yazılım; bulut bilişim, nesnelerin interneti, büyük veri, sosyal ağlar ve dağıtık sistemler gibi gelişerek yaygınlaşan kavramlar ve donanımsal teknolojilerdeki gelişmeler sayesinde gerek gömülü sistemler vasıtasıyla evimizden savunma sistemlerine, gerekse cep telefonu ve bilgisayarlar vasıtasıyla günlük eğlencemizden en karmaşık iş süreçlerine kadar, hayatımızın her alanında yerini almış ve etkileri hissedilen bir olgudur. Tıp, savunma sanayii, kamu yönetimi, ticaret, üretim, bankacılık, eğlence, vb. her türlü endüstri, yapılan işlerin çeşitliliğine rağmen teknoloji ve bilgisayar sistemlerindeki gelişmelerden ve hesaplama kabiliyetinden faydalanarak iş süreçlerini bilgisayar sistemleri üzerinde yürütmeye muhtaçtır ve dolayısıyla yazılıma ihtiyaç duymaktadır. Kurumlar bu yazılımı bilgi sistem departmanları vasıtasıyla üretir ya da yazılım şirketlerinden tedarik ederler. Her iki durumda da yazılımın üretilmesi bir dizi mühendislik ve proje yönetim sürecini gerektirir. Bu süreçlerin metodoloji seçiminde, üretilecek yazılımın büyüklüğü, karmaşıklığı, kullanım maksadı gibi karakteristik özellikleri, yazılımı üretecek organizasyon ve proje ekibinin yapısı ve kullanılabilecek kaynaklar önem arz eder. Bu çalışmanın amacı, yazılım mühendisliği kavramının ortaya çıkmasından beri gelişerek kullanılmakta olan geleneksel plan tabanlı yöntemlerin ve her türlü proje yönetimi metodolojisinin öncüsü konumundaki şelale modeli ile geleneksel plan tabanlı yöntemlerin yetersiz kaldığı durumlara bir alternatif olarak ortaya çıkan ve günümüzde çok önemli bir yer edinmiş olan çevik yöntemler arasında, projelere özgü durumlara uygun metodoloji seçimi konusunda rehberlik yapacak bir karşılaştırma sunmaktır.

Benzer konularda yapılan çalışmalar incelendiğinde yazılım geliştirme metodolojileri ile ilgili çeşitli karşılaştırma çalışmalarına rastlanmaktadır. Bu çalışmalar, çevik yöntemlerin kendi içindeki karşılaştırmaları, çeşitli geleneksel yöntemlerle çevik yöntemlerden bazılarının karşılaştırılması ve metodolojilerin tekil faktörler açısından karşılaştırılması olarak gruplanabilir. Benzer çalışmalarla ilgili daha detaylı bilgi beşinci bölümün birinci kısmında verilmiştir. Bu tezde, benzer çalışmalara nispeten çok daha geniş bir alanda karşılaştırma yapılmış; metodoloji kullanımı ve başarı oranları gibi istatistiki bilgi sunan araştırmaların en güncel verilerinden yararlanılarak genel resmin benzer çalışmalara göre daha güncel bir

şekilde ortaya konması amaçlanmıştır. Ayrıca 97 katılımcının cevapladığı anket ile elde edilen veriler kapsamında yazılım sektörü çalışanlarının şelale modeli ve çevik yöntemler ile ilgili görüşleri alınmış ve iki metodoloji arasında bir karşılaştırma ortaya konmuştur. Bu tez hazırlanırken, karşılaştırma çalışmalarının yanında, çevik yöntemlerin adaptasyonu, geleneksel plan tabanlı yöntemlerden çevik yöntemlere geçiş uygulamalarında elde edilen tecrübeler ve geleneksel plan tabanlı yöntemler ile çevik yöntemlerin birlikte kullanıldığı uygulamalarda elde edilen tecrübeler incelenmiştir.

Çalışmanın ikinci bölümünde proje yönetimi kavramı, üçüncü bölümünde ise yazılım mühendisliği ve yazılım proje yönetimi kavramları, yönetim süreçleri ve bilgi alanlarıyla birlikte özetlenmiştir. Dördüncü bölümde yazılım geliştirme yaşam çevrimi ele alınmış, sırasıyla şelale modeli ve çevik yöntemlerin kısa tarihçeleri, ilkeleri ve özellikleri ile fayda ve kısıtları ortaya konmuştur. Beşinci bölümde bu çalışmanın ana konusunu teşkil eden şelale modeli ve çevik yöntemlerin, yazılım proje yönetimi açısından ortaya çıkan durumlara göre karşılaştırmaları yapılmış, altıncı bölümde ise yapılan anket çalışması ele alınmıştır.

2. PROJE YÖNETİMİ

2.1. Proje Nedir

Proje, benzersiz bir ürün, hizmet ya da sonuç yaratmak için girişilen geçici bir çabadır [1]. Daha detaylı bir tanım yapmak gerekirse, bir probleme çözüm bulma ya da beliren bir fırsatı değerlendirmeye yönelik, bir ekibin, başlangıcı ve bitişi belirli bir süre ve sınırlı bir finansman dahilinde, birtakım kaynaklar kullanarak, müşteri memnuniyetini ve kaliteyi göz önünde bulundururken olası riskleri yönetmek şartıyla, tanımlanmış bir kapsama uygun amaç ve hedefler doğrultusunda özgün bir planı başlatma, yürütme, kontrol etme ve sonuca bağlama sürecidir [2]. Bir projede bulunması gereken özellikler şu şekilde sıralanabilir:

- Projeler bir ihtiyaçtan doğmalı ve bu ihtiyacı karşılamayı hedeflemelidir. Projelerde problemler hedeflere, hedefler faaliyetlere dönüştürülür.
- Projelerde; Projenin içinde bulunduğu çevre, ürünün/hizmetin sunulduğu kitle, kamu otoritesi, finansman kuruluşları ve çevresel koşullar belirtilmelidir.
- Proje mekânı belirli olmalıdır.
- Alışılmış, basit işlerden farklı olmalı ve bir şeyleri değiştirmeyi amaçlamalıdır.
- Kendine özgü olmalıdır.
- Belirli bir başlama ve bitiş süresi olmalıdır.
- Sınırlı kaynaklardan oluşmalıdır.
- Önceden planlanmış olmalıdır.
- Belirli kaynaklara ve bütçeye gereksinim duymalıdır.
- Değerlendirilebilir olmalı ve değerlendirme kriterleri proje başlangıcında belirlenmiş olmalıdır.
- Projeden elde edilecek sonucun bariz bir kullanıcısı veya müşteri olmalıdır.
- İnsan ve işgücü içermelidir. [3]

2.2. Proje Yönetimi Kavramı ve Gelişimi

Proje yönetimi, belirli bir projenin hedef ve amaçlarına ulaşıp bitirilmesi için kaynakların planlanması, organize edilmesi, tedarik edilmesi ve yönetilmesi

disiplinidir. Bir disiplin olarak proje yönetimi, inşaat, mühendislik ve ağır savunma aktivitelerinin uygulamalarından geliştirilmiştir [4]. Proje yönetimine ihtiyaç duyma nedenleri aşağıdaki şekilde sıralanabilir: [5]

- Kısa zamanda gerçekleştirilmesi istenen işler ve dinamik iş ortamları
- Artan koordinasyon gerektiren, çoğalan is gücü ve karmaşıklaşan organizasyon yapıları
- Sınırlı işletme kaynakları
- Yeniliklere adapte olma zorunluluğu
- İletişimin karmaşıklaşması
- İşlenecek bilgi kaynaklarının ve miktarının artması

Modern proje yönetimi teknikleri, 19. Yüzyılın sonlarında karmaşıklaşan iş yaşamı ile birlikte şekillenen ve gelişen yönetim ilkelerinin evrimleşmesi ile elde edilmiştir. Özellikle o yıllarda gerçekleştirilen büyük ölçekli devlet projeleri proje yönetimi tekniklerinin gelişmesinde itici güç olmuştur [6]. Proje yönetiminin en önemli fikir babalarından biri, Gantt çizelgesini tasarlamış olması ile ünlü olan Henry Gantt'tır. Diğer bir önemli fikir babası ise proje yönetiminde 5 yönetim fonksiyonunu kullanan Henri Fayol'dur. 1950'li yıllar, temel mühendislik alanlarının birlikte yürütüldüğü modern proje yönetimi döneminin başlangıcını oluşturmuştur. Proje yönetimi disiplini mühendislik modeli üzerine kurulu yönetim disiplininden ayrı bir disiplin olarak tanınmaya başlanmıştır. Bu dönemde ayrıca iki matematiksel proje zamanlama tekniği geliştirilmiştir. Kritik Yol Yöntemi (CPM), tesis bakımı projelerini yönetmek için DuPont ve Remington Rand Şirketleri tarafından geliştirilmiştir. Program Değerlendirme ve Gözden Geçirme Tekniği (PERT) ise Amerikan Donanması'nın bir deniz altı programı sırasında Booz Allen Hamilton tarafından geliştirilmiştir. Bu matematiksel teknikler hızla özel girişimlere de yayılmıştır. [4].

2.3. Proje Yönetiminin Faydaları

Proje yönetimi ilk bakışta daha iyi kontrol, değerlendirme sağlar ve müşteri ilişkilerinde mükemmelliği öngörür. Daha kısa üretim süreci, azalan maliyetler, yüksek kalite ve güvenilirlik, yüksek kar marjları başarılı proje yönetimlerinin çıktılarıdır. Bununla birlikte organizasyon bölümleri arasında iletişimi sağlayarak koordinasyon çabalarına değer katar. Çalışanların birlikte hareket etmesini

sağlayarak grup değerlerini korur [7]. Proje yönetiminin işletmelere sağladığı faydalar şu şekilde sıralanabilir [8]:

- Daha ekonomik geliştirme süreçleri sağlar.
- Kaynakların daha verimli kullanılmasını ve daha etkin kontrolü sağlar.
- Düşük maliyet ve yüksek kâr yaratır.
- Yüksek kalite ve güvenlik sağlar.
- Etkin koordinasyon ve motivasyon sağlar.
- Müşteri ilişkilerinde iyileştirme sağlar.
- Amaç ve hedeflere ne zaman ve nasıl ulaşılacağını önceden belirler.
- Sürekli raporlama gereksinimini minimum düzeye indirger.
- Proje için gerekli zamanı bastan belirlemeye yardımcı olur.
- Projenin maliyetinin önceden belirlenmesini sağlar.
- Gerekli kaynakların neler olduğunu ortaya koyar.
- Kullanılacak teknolojiyi açıklar.
- Kontrol sisteminin kurulmasını sağlar.
- Tüm görevlerin organizasyon semalarında gösterilmesini sağlar.
- Proje ekip üyelerinin proje geliştirme, uygulama ve tahmin yeteneklerinin geliştirilmesini sağlar.

2.4. Proje Yönetiminin Dezavantajları

Proje yönetiminin yukarıda belirtilen faydalarının yanında uygulanmasının birlikte getirdiği kısıtları da mevcuttur. Bunlar [9] :

- Karmaşık organizasyon yapılarına neden oluşu.
- Kuruluş politikalarının bozulması eğilimi.
- Yönetim güçlüğü.
- Personel kullanımı zorluğudur.

2.5. Proje Yönetim Süreçleri

Projelerin yaşam çevrimi boyunca etkin bir şekilde akışını sağlayan süreçlere proje yönetim süreçleri denir. Proje Yönetimi Bilgi Alanları bölümünde ilgili bilgi alanının altında belirtilen toplam 47 adet proje yönetim süreci bulunmaktadır. Proje Yönetimi Bilgi Birikimi Kılavuzu PMBOK, proje yönetim süreçlerini, birbirleriyle

entegrasyonu, etkileşimi ve hizmet ettikleri amaç doğrultusunda, proje yönetimi süreç grupları adı altında beş grup halinde sınıflandırır [1]:

- Başlangıç Süreçleri
- Planlama Süreçleri
- Yürütme Süreçleri
- İzleme ve Kontrol Süreçleri
- Kapanış Süreçleri

2.5.1. Başlangıç süreçleri

Başlangıç süreçleri, gerekli yetkilerin alınarak yeni bir projenin veya devam eden bir projenin yeni bir safhasının başlatılması faaliyetlerini içerir. Başlangıç süreçleri çerçevesinde başlangıç kapsamı tanımlanır ve başlangıç finans kaynakları edinilir. Projenin sonucunu etkileyecek ve projenin sonucundan etkilenecek iç ve dış paydaşlar belirlenir. Henüz atanmadıysa proje yöneticisi atanır. Proje beratı onaylanır ve proje resmen başlamış olur. Bu süreç grubunun en önemli amacı paydaşların beklentileri ile proje amacını aynı doğrultuya sokmak, projenin kapsamı ve hedefleri konusunda görünürlük sağlamak ve yapacakları katkının beklentilerin karşılanması konusunda nasıl etkili olacağını kendilerine göstermektir. Büyük projeler safhalar bölünmeli ve başlangıç süreçleri bu safhaların başlangıcında proje başlangıcında uygulanan süreçlerin doğrulanması şeklinde tekrar edilmelidir. Bu şekilde projenin ihtiyaçlarına yabancılaşma önlenmiş olur [1].

2.5.2. Planlama süreçleri

Planlama süreçleri, harcanacak emeğin kapsamının ve hedeflerin belirlenmesi ve bu hedeflere ulaşılması için izlenecek yolun çizilmesi faaliyetlerini içerir. Bu süreçler içerisinde projenin yürütülmesini sağlayacak proje yönetim planı ve proje dokümanları geliştirilir. Proje süresince meydana gelen değişiklikler, edinilen ilave bilgiler ve geri beslemeler neticesinde planlama süreçleri ve hatta başlangıç süreçleri tekrar edilebilir. Planlama ve dokümantasyonun sürekli gelişen ve yinelemeli faaliyetleri olduğu manasına gelen proje yönetiminin bu karakteristiğine aşamalı olgunlaşma denir. Bu süreç grubunun en önemli faydası projenin veya safhanın hedeflerine ulaşması için izlenmesi gereken yolun çizilmesi, strateji ve

taktiklerin belirlenmesidir. Planlama süreçlerinin iyi yönetilmesi halinde paydaş katılımı ve desteği kolaylaşır [1].

2.5.3. Yürütme süreçleri

Yürütme süreçleri, projenin hedeflerinin karşılanması için proje yönetim planında belirlenmiş faaliyetlerin icra edildiği süreçleri içerir. Bu süreçler genel olarak insanların ve kaynakların koordine edilmesi, paydaş beklentilerinin yönetilmesi ve proje yönetim planında belirtilen proje faaliyetlerinin icrası ve entegrasyonudur. Proje bütçesinin büyük bir kısmı yürütme süreçlerinin icrası esnasında harcanır [1].

2.5.4. İzleme ve kontrol süreçleri

İzleme ve kontrol süreçleri, projenin ilerleyişi ve performansının izlenmesi, gözden geçirilmesi ve düzenlenmesi, planlarda değişiklik yapılması gereken alanların belirlenmesi, yeni planların uygulanması faaliyetlerini içerir. Bu süreç grubunun en büyük faydası, proje performansının düzenli aralıklarla ölçülmesi, analiz edilmesi ve proje planında meydana gelen sapmaların tespit edilerek gerekli tedbirlerin alınmasının sağlanmasıdır. Bu süreçler içerisinde, değişiklikler takip edilerek olası problemlerin sezilmesi halinde düzeltici veya önleyici tedbirler alınır ve proje yönetim planı ile proje performans ölçümü ana hatları dışına çıkan proje faaliyetleri izlenir [1].

2.5.5. Kapanış süreçleri

Kapanış süreçleri, projenin, safhanın ya da sözleşmelerden doğan zorunlulukların resmi olarak tamamlanması için gereken bütün proje yönetim süreç grubu faaliyetlerini içerir. Bu süreçler aynı zamanda ertelenen veya başarısız olan projelerin kapatılış faaliyetlerini de içerir. Proje veya safha kapanışında aşağıdaki hususlar gerçekleştirilir [1]:

- Proje veya safhanın kapanışı için müşteri veya sponsorun kabulü alınır.
- Proje sonu veya safha bitişi gözden geçirme işlemleri yapılır.
- Süreç uyarlamalarının etkileri kaydedilir.
- Alınan dersler belgelenir.
- Kurumsal süreçlere gerekli güncellemeler uygulanır.
- Geçmişe dönük veri oluşturması maksadıyla her türlü proje dokümanı proje

yönetimi bilgi sistemine kaydedilir.

- Bütün tedarik işlemleri durdurulur ve ilgili sözleşmeler feshedilir.
- Proje elemanlarının değerlendirmeleri alınır ve proje kaynakları serbest bırakılır.

2.6. Proje Yönetimi Bilgi Alanları

Bilgi alanı terimi; bir mesleki alan, proje yönetim alanı veya uzmanlık alanını meydana getiren bütün kavramlar, faaliyetler ve terimleri temsil eder. Proje Yönetimi Bilgi Birikimi Kılavuzu PMBOK®, proje yönetimi ile ilgili on bilgi alanı sunmaktadır [1]:

- Proje Entegrasyon Yönetimi
- Proje Kapsam Yönetimi
- Proje Zaman Yönetimi
- Proje Maliyet Yönetimi
- Proje Kalite Yönetimi
- Proje İnsan Kaynakları Yönetimi
- Proje İletişim Yönetimi
- Proje Risk Yönetimi
- Proje Tedarik Yönetimi
- Proje Paydaş Yönetimi

2.6.1. Proje entegrasyon yönetimi

Proje entegrasyon yönetimi, proje yönetim süreç gruplarındaki çeşitli süreç ve proje yönetim faaliyetlerinin, tanımlanması, belirlenmesi, birleştirilmesi, bütünleştirilmesi ve koordine edilmesi faaliyetlerini içerir [1]. Proje entegrasyon yönetimi, projenin başarılı bir bicimde tamamlanabilmesi için projenin bütün unsurlarının doğru zamanda bir araya gelebilmesini garanti etmektedir. Proje yönetiminin omurgasını oluşturması sebebiyle proje yönetimi içerisinde yer alan on bilgi alanının birincisi olarak incelenmektedir [10]. Proje entegrasyon süreci boyunca proje yöneticisi tarafından proje elemanlarının, planların ve yerine getirilmesi gereken görevlerin koordinasyonu sağlanır, üst düzey yöneticiler ile iletişim kurulur [6]. Proje entegrasyon yönetimi süreçleri aşağıda sunulmuştur [1]:

- Proje Beratının Hazırlanması
- Proje Yönetim Planının Hazırlanması
- Proje İşlerinin Yönetilmesi ve Yönlendirilmesi
- Proje İşlerinin İzlenmesi ve Kontrolü
- Bütünleşik Değişiklik Kontrolünün İcrası
- Proje veya Safhanın Kapatılması

2.6.2. Proje kapsam yönetimi

Proje kapsam yönetimi, projenin başarıyla sonuçlanması için gereken bütün işleri ve sadece gereken işleri içerdiğinden emin olunmasını sağlayan süreçleri içerir [1]. Proje kapsam yönetimi, proje kapsamı içinde olan ve olmayan noktaların tanımlanması ve kontrol edilmesi süreçlerini içerir. Kapsamın belirlenmesinde projenin içereceği kavramların netleştirilmesi ve hem proje ekibi hem de müşteri tarafından aynı şekilde anlaşılması kadar projenin kapsamına dahil edilmeyen noktaların da belirlenmesi projenin başarısı acısından önemlidir [10]. Proje kapsam yönetimi süreçleri aşağıda sunulmuştur [1]:

- Kapsam Yönetiminin Planlanması
- Gereksinimlerin Toplanması
- Kapsamın Tanımlanması
- İş Dökümü Yapısının Yaratılması
- Kapsamın Doğrulanması
- Kapsamın Kontrolü

2.6.3. Proje zaman yönetimi

Proje zaman yönetimi, projenin zamanında tamamlanması için gereken faaliyetleri içerir [1]. Öncelik proje takviminin oluşturulmasıdır. Proje takvimi, proje faaliyet planının tarifeye dönüştürülmüş halidir. Projeyi izlemek ve proje faaliyetlerini kontrol etmek için temel oluşturur [10]. Proje zaman yönetimi süreçleri aşağıda sunulmuştur [1]:

- Takvim Yönetiminin Planlanması
- Faaliyetlerin Tanımlanması
- Faaliyetlerin Sıralanması

- Faaliyet Kaynaklarının Tahmini
- Faaliyet Sürelerinin Tahmini
- Takvimin Oluşturulması
- Takvimin Takip ve Kontrolü

2.6.4. Proje maliyet yönetimi

Proje maliyet yönetimi, projenin belirlenen bütçe içerisinde tamamlanabilmesi maksadıyla maliyetlerin planlanması, tahmini, bütçelenmesi, fonlanması, yönetilmesi ve kontrolü faaliyetlerini içerir. Proje maliyet yönetimi süreçleri aşağıda sunulmuştur [1]:

- Maliyet Yönetim Planlaması
- Maliyet Tahmini
- Bütçenin Belirlenmesi
- Maliyet Kontrolü

2.6.5. Proje kalite yönetimi

Proje kalite yönetimi, hedef ve sorumluluklar ile kalite politikalarını belirleyen icracı kurumun, projenin gereksinimlerinin karşılanabilmesi amacıyla yürüttüğü bütün süreç ve faaliyetleri içerir. Proje kalite yönetimi süreçleri aşağıda sunulmuştur [1]:

- Kalite Yönetim Planlaması
- Kalite Güvence Uygulamaları
- Kalite Kontrol Uygulamaları

2.6.6. Proje insan kaynakları yönetimi

Proje insan kaynakları yönetimi, proje elemanlarının organize edilmesi, yönetilmesi ye yönlendirilmesi faaliyetlerini içerir [1]. Projeler ve organizasyonların başarılarının çalışanlar tarafından belirlenmesi, proje insan kaynakları yönetimini proje yönetiminin, özellikle de yetişmiş is gücünün temini ve muhafazasının çok zor olduğu bilgi teknolojileri proje yönetiminin, en can alıcı noktalarından biri haline getirmiştir oluşturur [10]. Proje insan kaynakları yönetimi süreçleri aşağıda sunulmuştur [1]:

- İnsan Kaynakları Yönetimi Planlaması

- Proje Elemanlarının Temini
- Proje Takımının Oluşturulması
- Proje Takımının Yönetilmesi

2.6.7. Proje iletişim yönetimi

Proje iletişim yönetimi, her türlü proje bilgisinin uygun ve zamanında planlanma, toplanma, dağıtılma, yönetim, kontrol, izlenme ve son olarak elden çıkarma faaliyetlerini içerir [1]. bilgi teknolojilerinin sürekli değişiminin ve gelişiminin yeni teknik jargonlar getirmesi, her gecen gün daha fazla kişinin bilgisayar kullanmasına rağmen, kullanıcılarla yazılım geliştiriciler arasındaki iletişimin güçleşmesine sebep olmaktadır [10]. Proje iletişim yönetimi, projede kullanmak üzere toplanan, yayılan ve depolanan bilginin doğruluğunu, projeye uygun olmasını ve projeye zamanında ulaşmasını sağlamaktır. Projenin başarılı olması için gereken bilgi, fikir ve insanlar arasındaki kritik bağlantıları sağlar [6]. Proje iletişim yönetimi süreçleri aşağıda sunulmuştur [1]:

- İletişim Yönetiminin Planlanması
- İletişimin Yönetilmesi
- İletişimin Kontrolü

2.6.8. Proje risk yönetimi

Proje risk yönetimi, proje süresince olumlu olayların gerçekleşip olumsuz olayların önlenmesi için gereken faaliyetleri içerir [1]. Risk, projenin başarısını tehdit edebilecek ve henüz ortaya çıkmamış problemler olarak tanımlanabilir. Risk yönetimi sureci, proje kapsamındaki tüm risklerin belirlenmesini, analiz edilmesini, en aza indirgenmesini ve risklerin proje üzerindeki etkisini kontrol etmeyi amaçlar [11]. Proje risk yönetimi süreçleri aşağıda sunulmuştur [1]:

- Risk Yönetiminin Planlanması
- Risklerin Tanımlanması
- Niteliksel Risk Analizinin Yapılması
- Niceliksel Risk Analizinin Yapılması
- Risk Yanıtlarının Planlanması
- Risklerin İzlenmesi ve Kontrolü

2.6.9. Proje tedarik yönetimi

Proje tedarik yönetimi, ihtiyaç duyulan ürün ve hizmetlerin proje ekibi dışından satın alınması veya temin edilmesi faaliyetlerini içerir [1]. Projelerde dış kaynak kullanımının çeşitli nedenleri vardır:

- Maliyetleri azaltmak.
- Kendi faaliyet alanına girmeyen işlerle zaman kaybetmeden kendi faaliyet alanlarına yoğunlaşmak.
- Organizasyonun öz yeteneklerini daha etkin kullanabilecekleri yeni teknolojilere ve bilgilere ulaşmak.
- Olaylara ve durumlara hızlı uyum sağlamak ve esneklik.
- Sorumlulukları sözleşmelerle kayıt altına almak.

Dış kaynak kullanımına karar verilirken, yukarıda belirtilen faydalarının yanı sıra sorunların da olabileceği bilinmeli ve dıs kaynak kullanım kararları bu doğrultuda alınmalıdır. Karşılaşılabilecek sorunların bazıları, aşırı bağlılık olarak adlandırılan dış kaynak kullanan firmanın tedarikçi işletmeye bağımlı hale gelmesi, esnekliğin, kontrolün ve rekabet avantajının kaybedilmesi, işletmenin içinin boşaltılmasıdır [12]. Proje tedarik yönetimi süreçleri aşağıda sunulmuştur [1]:

- Tedarik Yönetiminin Planlanması
- Tedarik Faaliyetlerinin İcra Edilmesi
- Tedarik Faaliyetlerinin İzlenmesi ve Kontrolü
- Tedarik Faaliyetlerinin Tamamlanması

2.6.10. Proje paydaş yönetimi

Proje paydaş yönetimi; projeyi etkileyebilecek ve projeden etkilenebilecek insanların, grupların ve organizasyonların tanımlanmasını, paydaşların beklentilerinin ve projeye etkilerinin analiz edilmesini ve proje kararları ile proje yürütülmesi faaliyetlerine paydaşların katılımı için uygun yönetim stratejilerinin geliştirilmesini sağlayacak süreçleri içerir. Proje paydaş yönetimi süreçleri aşağıda sunulmuştur [1]:

- Paydaşların Belirlenmesi
- Paydaş Yönetiminin Planlanması

- Paydaş Katılımının Yönetimi
- Paydaş Katılımının Kontrolü

2.7. Proje Yönetim Süreçleri ile Proje Yönetim Bilgi Alanlarının İlişkisi

Bölüm 2.6'da incelenen on proje yönetim bilgi alanında yer alan kırk yedi proje yönetim sürecinin, bölüm 2.5'de incelenen proje yönetim süreç gruplarına göre dağılımı Çizelge 2.1'de sunulmuştur.

Çizelge 2.1. Proje yönetim süreçleri ile proje yönetim bilgi alanlarının ilişkisi [1]

		PROJE YÖNETİM SÜREÇ GRUPLARI				
		Başlangıç Süreçleri	Planlama Süreçleri	Yürütme Süreçleri	İzleme ve Kontrol Süreçleri	Kapanış Süreçleri
BİLGİ ALANLARI	Proje Entegrasyon Yönetimi	Proje Beratının Hazırlanması	Proje Yönetim Planının Hazırlanması	Proje İşlerinin Yönetilmesi ve Yönlendirilmesi	Proje İşlerinin İzlenmesi ve Kontrolü Bütünleşik Değişiklik Kontrolünün İcrası	Proje veya Safhanın Kapatılması
	Proje Kapsam Yönetimi		Kapsam Yönetiminin Planlanması Gereksinimlerin Toplanması Kapsamın Tanımlanması İş Dökümü Yapısının Yaratılması		Kapsamın Doğrulanması Kapsamın Kontrolü	
	Proje Zaman Yönetimi		Takvim Yönetiminin Planlanması Faaliyetlerin Tanımlanması Faaliyetlerin Sıralanması Faaliyet Kaynaklarının Tahmini Faaliyet Sürelerinin Tahmini Takvimin Oluşturulması		Takvimin Takip ve Kontrolü	
	Proje Maliyet Yönetimi		Maliyet Yönetim Planlaması Maliyet Tahmini Bütçenin Belirlenmesi		Maliyet Kontrolü	
	Proje Kalite Yönetimi		Kalite Yönetim Planlaması	Kalite Güvence Uygulamaları	Kalite Kontrol Uygulamaları	

Çizelge 2.1. (devam) Proje yönetim süreçleri ile proje yönetim bilgi alanlarının ilişkisi [1]

		PROJE YÖNETİM SÜREÇ GRUPLARI				
		Başlangıç Süreçleri	Planlama Süreçleri	Yürütme Süreçleri	İzleme ve Kontrol Süreçleri	Kapanış Süreçleri
BİLGİ ALANLARI	Proje İnsan Kaynakları Yönetimi		İnsan Kaynakları Yönetimi Planlaması	Proje Elemanlarının Temini Proje Takımının Oluşturulması Proje Takımının Yönetilmesi		
	Proje İletişim Yönetimi		İletişim Yönetiminin Planlanması	İletişimin Yönetilmesi	İletişimin Kontrolü	
	Proje Risk Yönetimi		Risk Yönetiminin Planlanması Risklerin Tanımlanması Niteliksel Risk Analizinin Yapılması Niceliksel Risk Analizinin Yapılması Risk Yanıtlarının Planlanması		Risklerin İzlenmesi ve Kontrolü	
	Proje Tedarik Yönetimi		Tedarik Yönetiminin Planlanması	Tedarik Faaliyetlerinin İcra Edilmesi	Tedarik Faaliyetlerinin İzlenmesi ve Kontrolü	Tedarik Faaliyetlerinin Tamamlanması
	Proje Paydaş Yönetimi	Paydaşların Belirlenmesi	Paydaş Yönetiminin Planlanması	Paydaş Katılımının Yönetimi	Paydaş Katılımının Kontrolü	

3. YAZILIM MÜHENDİSLİĞİ VE YAZILIM PROJE YÖNETİMİ

3.1. Yazılım ve Yazılım Mühendisliği

Yazılım kelimesinin Türk Dil Kurumu Güncel Türkçe Sözlüğündeki karşılığı: “Bir bilgisayarda donanıma hayat veren ve bilgi işlemde kullanılan programlar, yordamlar, programlama dilleri ve belgelemelerin tümü” olarak verilmiştir [13]. Yazılım sadece bir programlama dilinde yazılmış kod satırlarını değil, somut dokümantasyonu da içerir. Yazılım mühendisliği ise sistem tanımlamasının ilk safhalarından, yazılımın kullanım ve bakım safhalarına kadar yazılım üretiminin her aşamasında icra edilen faaliyetleri içeren mühendislik disiplinidir. Yazılım mühendisliği, yazılım geliştirme sürecinin sadece teknik bölümünü içermez, yazılım proje yönetimi ve yazılım üretiminin destekleyici unsurları olan metodoloji, kullanılan araçlar ve kavramların geliştirilmesi de yazılım mühendisliğinin kapsama alanına girer [14].

Yazılım mühendisliğinde, kullanılan metodoloji ne olursa olsun, dört temel faaliyetten meydana gelen sistematik bir yaklaşım uygulanır. Bu faaliyetler;

- Müşteriler ve geliştiriciler tarafından, üretilecek yazılımın gereksinim ve kısıtlarının belirlendiği *Sistem Tanımlaması.*
- Yazılımın tasarlandığı ve kodlamanın yapıldığı *Yazılım Geliştirme.*
- Yazılımın, müşterinin talep ettiği ürün olduğunun onaylandığı *Yazılım Doğrulama.*
- Yazılımın, değişen müşteri ve pazar ihtiyaçları doğrultusunda güncellendiği *Yazılım Evrimi* [14].

3.2. Yazılım Mühendisliği Bilgi Birikimi Kılavuzu

IEEE tarafından 1998 yılından beri yürütülen ve en son 2014 yılında 3. sürümü yayınlanan Yazılım Mühendisliği Bilgi Birikimi Kılavuzu (SWEBOK), yazılım mühendisliği alanında genel kabul edilen bilgilerin tanımlandığı, yazılım mühendislerinin temel dokümandır. SWEBOK aynı zamanda *19759 ISO Teknik Raporu* olarak da bilinir. Bu dokümanın oluşturulmasının beş hedefi vardır. Bunlar;

- Yazılım Mühendisliğine dünya çapında tutarlı bir bakış açısı getirilmesine önayak olmak,
- Yazılım Mühendisliğinin kapsamını belirlemek ve Bilgisayar Bilimi, Proje Yönetimi, Bilgisayar Mühendisliği ve Matematik gibi diğer disiplinler arasındaki yerini ortaya koymak,
- Yazılım Mühendisliği disiplininin içeriğini tanımlamak,
- Yazılım Mühendisliği Bilgi Birikimine konu bazında erişim sağlamak,
- Öğretim programları, sertifikasyon ve lisanslama işlemleri için temel oluşturmak.

Birinci hedefin gerçekleşmesi için SWEBOK dünya çapında 33 ülkeden 150 katılımcının katkıları ile hazırlanmıştır. İkinci hedef doğrultusunda yazılım mühendisliğinin kapsamını oluşturan 15 bilgi alanı tanımlanmıştır:

- Yazılım Gereksinimleri
- Yazılım Tasarımı
- Yazılımın Oluşturulması
- Yazılım Testi
- Yazılımın Bakımı
- Yazılım Konfigürasyon Yönetimi
- Yazılım Mühendisliği Yönetimi
- Yazılım Mühendisliği Süreci
- Yazılım Mühendisliği Modelleri ve Yöntemleri
- Yazılım Kalitesi
- Yazılım Mühendisliği Mesleki Uygulaması
- Yazılım Mühendisliği Ekonomisi
- Bilgisayarın Temelleri
- Matematik Temelleri
- Mühendislik Temelleri

Yukarıda belirtilen bilgi alanları ile ilgili temel hususlar üçüncü kısımda belirtilecektir. Kapsamın tanımlanması hususunda yazılım mühendisliğinin kesiştiği disiplinlerin de tanımlanması önem arz etmektedir. Bu kapsamda SWEBOK'a göre yazılım mühendisliği 7 ana disiplin ile etkileşim halindedir. Bu disiplinler:

- Bilgisayar Mühendisliği
- Bilgisayar Bilimi
- Genel Yönetim
- Matematik
- Proje Yönetimi
- Kalite Yönetimi
- Sistem Mühendisliği

Yazılım mühendisliği bilgi alanları bazında oluşturulan bölümler ve bilgi alanlarının alt alanları ile yazılım mühendisliği disiplininin içeriği tanımlanmış ve yazılım mühendisliği bilgi birikimine konu bazında erişim sağlanmıştır. Beşinci ve son hedefin tahakkuku kapsamında SWEBOK'ta ileri araştırma ve uzmanlık bilgilerine yer verilmemiş, çoğu zaman, çoğu projeye uygulanabilen ve değeri ile faydası hakkında fikir birliği olan bilgi ve uygulamalara değinilmiştir. SWEBOK içerisinde alt alanlarla ilgili ayrıca detaylı referans kaynaklar bulunmaktadır [15].

3.3. Yazılım Mühendisliği Bilgi Alanları

Yazılım mühendisliği bilgi alanları, yazılım mühendisliği disiplininin kapsamını oluşturan ana başlıklardır. Yazılım geliştirme faaliyetlerinde uygulanan metodoloji ne olursa olsun bilgi alanlarında belirtilen hususlar geçerliliğini korur. Metodoloji sadece bu hususların uygulanma şeklini ve miktarını belirler. Bilgi alanlarının çok büyük bir kapsama sahip olduğu göz önünde bulundurularak bu bölümde her alan ile ilgili kısa bilgi verilecek ve alt alanları belirtilecektir.

3.3.1. Yazılım gereksinimleri

Yazılım gereksinimleri bilgi alanı, gereksinimlerin edinilmesi, analizi, tanımlanması, doğrulanması ve yazılımın yaşam çevrimi boyunca gereksinimlerin yönetilmesi konularıyla ilgilidir. Yazılım gereksinimleri ile ilgili faaliyetlerin yetersiz icra edildiği yazılım projeleri her zaman tehlikeye açık projelerdir. Yazılım Gereksinimleri Bilgi Alanının alt konu başlıkları aşağıdaki gibidir [15]:

- Yazılım Gereksinimlerinin Esasları
- Gereksinim Süreci
- Gereksinimlerin Temini

- Gereksinimlerin Analizi
- Gereksinimlerin Belirtimi
- Gereksinimlerin Doğrulanması
- Uygulamaya Yönelik Dikkate Alınması Gereken Hususlar
- Yazılım Gereksinim Araçları

3.3.2. Yazılım tasarımı

Tasarım kelimesi; hem bir sistem veya bileşenin mimarisi, alt bileşenleri, ara yüzleri ve diğer niteliklerinin tanımlanması sürecini, hem de bu süreç sonucunda elde edilen sonucu tanımlar [16]. Süreç olarak ele alındığında tasarım, yazılımın oluşturulmasına temel arz edecek yazılım iç yapısının tanımlanabilmesi maksadıyla yazılım gereksinimlerinin analiz edilmesi faaliyetlerini kapsar. Bu süreç sonunda elde edilen tasarım ise yazılımı oluşturan alt bileşenler ve bu alt bileşenler arasındaki ara yüzleri oluşturan mimariyi, yazılımın oluşturulması aşamasında faydalanılacak bir ayrıntı seviyesinde tanımlamalıdır. Yazılım Tasarımı bilgi alanının alt konu başlıkları aşağıdaki gibidir [15]:

- Yazılım Tasarımının Esasları
- Yazılım Tasarımındaki Temel Sorunlar
- Yazılımın Yapısı ve Mimarisi
- Kullanıcı Ara Yüzü Tasarımı
- Yazılım Tasarımı Kalitesinin Analizi ve Değerlendirilmesi
- Yazılım Tasarımı Gösterimi
- Yazılım Tasarımı Strateji ve Yöntemleri
- Yazılım Tasarım Araçları

3.3.3. Yazılımın oluşturulması

Yazılımın oluşturulması, kodlama, doğrulama, birim testleri, entegrasyon testleri ve hata ayıklama sonucunda çalışan yazılımın yaratılması faaliyetidir. Bu bilgi alanı diğer bütün bilgi alanları ile bağlantılı olsa da tasarımı girdi olarak kullandığı ve test faaliyetlerinde kullanılacak çıktıyı ortaya koyduğu için en çok Yazılım Tasarımı ve Yazılım Testi Bilgi Alanları ile ilişkilidir. Bu ilişkinin sırası sayısı ve yoğunluğu, kullanılan yazılım geliştirme metodolojisine göre değişiklik gösterir. Yazılımın Oluşturulması bilgi alanının alt konu başlıkları aşağıdaki gibidir [15]:

- Yazılımın Oluşturulmasının Esasları
- Yazılımın Oluşturulmasının Yönetimi
- Uygulamaya Yönelik Dikkate Alınması Gereken Hususlar
- Yazılım Oluşturma Teknolojileri
- Yazılım Oluşturma Araçları

3.3.4. Yazılım testi

Yazılım Testi, bir programın çalışma senaryolarından seçilen durumlarda programın belirli girdilerle çalıştırılarak kendisinden beklenen davranışları sergileyip sergilemediğinin denenmesidir. Test faaliyetlerinin icrası için kodlamanın bitmesi gerekmez. Test faaliyetleri bütün geliştirme ve bakım safhalarına yayılmalıdır. Yazılımın Oluşturulması bilgi alanının alt konu başlıkları aşağıdaki gibidir [15]:

- Yazılım Testinin Esasları
- Test Seviyeleri
- Test Teknikleri
- Testlerle İlgili Ölçümler
- Test Süreci
- Yazılım Test Araçları

3.3.5. Yazılımın bakımı

Yazılım projeleri sonucunda müşterinin gereksinimlerini karşılayan yazılım ürünü teslim edilir. Fakat bu yazılım kullanıldıkça hatalar tespit edilir, çalışma ortamı değişir, yeni kullanıcı gereksinimleri ortaya çıkar. Bakım safhasının, teslimat sonrası bir garanti süreci ile başlamasına rağmen bakım faaliyetleri çok daha erken icra edilmeye başlanır. Yazılımın bakımı, yazılıma maliyet etkin bir destek sağlayabilmek için gereken bütün faaliyetler olarak tanımlanabilir. Bu faaliyetler hem teslimat öncesi hem de teslimat sonrası icra edilebilir. Teslimat öncesi faaliyetlere örnek olarak, teslimat sonrası işlerin planlaması, bakım kabiliyetinin ve geçiş faaliyetlerine lojistik desteğin belirlenmesi verilebilir. Teslimat sonrası faaliyetlere örnek olarak ise, yazılımın tadilatı, eğitim, yardım masası işletilmesi veya desteği verilebilir. Yazılımın Oluşturulması bilgi alanının alt konu başlıkları aşağıdaki gibidir [15]:

- Yazılım Bakımının Esasları
- Yazılım Bakımının Temel Sorunları
- Bakım Süreci
- Bakım Teknikleri
- Yazılım Bakımı Araçları

3.3.6. Yazılım konfigürasyon yönetimi

Sistem, bir veya daha fazla amacı gerçekleştirmek için oluşturulmuş birbiri ile etkileşim halinde olan bileşenler bütünüdür. Konfigürasyon ise elde edilen üründe bulunan ya da teknik dokümantasyonda belirtilen donanım ve yazılımın işlevsel ve fiziksel nitelikleridir. [16]. Bu kapsamda, konfigürasyon yönetimi, bir sistemin yaşam çevrimi müddetince konfigürasyonunun bütünlüğünün ve izlenebilirliğinin sağlanması ve konfigürasyonunda meydana gelen değişikliklerin sistemli bir şekilde yönetilebilmesi maksadıyla herhangi bir zamanda o anlık konfigürasyonunun belirlenmesi disiplinidir. Yazılımın Oluşturulması bilgi alanının alt konu başlıkları aşağıdaki gibidir [15]:

- Yazılım Konfigürasyon Yönetimi Sürecinin İdaresi
- Yazılım Konfigürasyonunun Tanımlanması
- Yazılım Konfigürasyonunun Kontrolü
- Yazılım Konfigürasyonu Durum Muhasebesi
- Yazılım Konfigürasyon Denetimi
- Yazılım Sürüm Yönetimi ve Teslimat
- Yazılım Konfigürasyon Yönetimi Araçları

3.3.7. Yazılım mühendisliği yönetimi

Yazılım mühendisliği yönetimi, yazılım ürünleri ve yazılım mühendisliği hizmetlerinin, etkin, verimli ve paydaşların fayda sağlayacağı şekilde ortaya konmasını sağlamak maksadıyla, planlama, koordinasyon, ölçme, izleme, kontrol ve raporlama gibi faaliyetlerin yönetimi olarak tanımlamaktadır. Yazılım mühendisliği yönetimi bilgi alanı yazılım proje yönetimi faaliyetlerini kapsamaktadır. Bu nedenle ilgili alt başlıklar 3.5.Yazılım Proje Yönetimi Süreçleri kısmında daha detaylı incelenecektir. Yazılımın Mühendisliği Yönetimi bilgi alanının alt konu başlıkları aşağıdaki gibidir [15]:

- Başlatma ve Kapsam Tanımı
- Yazılım Proje Planlaması
- Yazılım Proje İcrası
- Gözden Geçirme ve Değerlendirme
- Kapanış
- Yazılım Mühendisliği Ölçümlemesi
- Yazılım Mühendisliği Yönetim Araçları

3.3.8. Yazılım mühendisliği süreci

Yazılım Mühendisliği Süreci Bilgi Alanı, gereksinimler, tasarım, oluşturma, test gibi yazılımın geliştirilmesi, işletilmesi ve bakımı amacıyla yazılım mühendisleri tarafından icra edilen faaliyetlerle ilgilidir. Yazılımın Mühendisliği Süreci bilgi alanının alt konu başlıkları aşağıdaki gibidir [15]:

- Yazılım Mühendisliği Süreçlerinin Tanımı
- Yazılım Yaşam Çevrimi
- Yazılım Mühendisliği Süreçlerinin Değerlendirilmesi ve Gelişimi
- Yazılım Ölçümü
- Yazılım Mühendisliği Süreç Araçları

3.3.9. Yazılım mühendisliği modelleri ve yöntemleri

Yazılım mühendisliği modelleri ve yöntemleri, yazılım mühendisliği faaliyetlerine bir yapısallık empoze ederek, ilgili faaliyetin sistematik, tekrarlanabilir ve başarı odaklı olmasını amaçlar. Model kullanımı, problem çözümü konusuna bir yaklaşım sunar ve gösterim kolaylığı sağlar, gelecek model oluşturma ve analiz işlemlerinde yol gösterici olur. Yöntemler ise, son kullanıcıya teslim edilen yazılım elde edilene kadar icra edilen faaliyetlere sistematik ele alış biçimi getirir. Yazılımın Mühendisliği Modelleri ve Yöntemleri bilgi alanının alt konu başlıkları aşağıdaki gibidir [15]:

- Modelleme
- Model Çeşitleri
- Modellerin Analizi
- Yazılım Mühendisliği Yöntemleri

3.3.10. Yazılım kalitesi

Yazılım kalitesinin tanımı ile ilgili birçok görüş olmakla birlikte, en kısa ve öz olan tanımlardan birine göre yazılım kalitesi, yazılım ürününün belirli şartlar altında ifade edilen ihtiyaçları karşılama kapasitesidir [17]. Yazlımın kalitesi doğrudan gereksinimlerle ilgilidir. Yazılımın Mühendisliği Modelleri ve Yöntemleri bilgi alanının alt konu başlıkları aşağıdaki gibidir [15]:

- Yazılım Kalite Esasları
- Yazılım Kalite Yönetim Süreçleri
- Uygulamaya Yönelik Dikkate Alınması Gereken Hususlar
- Yazılım Kalite Araçları

3.3.11. Yazılım mühendisliği mesleki uygulaması

Yazılım Mühendisliği Mesleki Uygulaması Bilgi Alanı, yazılım mühendislerinin yazılım mühendisliği mesleğini sorumlu, meslek etiği çerçevesinde ve profesyonelce icra edebilmeleri için sahip olmaları gereken bilgi, beceri ve davranışlarla ilgilidir. Yazılımın Mühendisliği Mesleki Uygulaması bilgi alanının alt konu başlıkları aşağıdaki gibidir [15]:

- Profesyonellik
- Grup Dinamikleri ve Psikoloji
- İletişim Becerileri

3.3.12. Yazılım mühendisliği ekonomisi

Bir yazılım ürünü, hizmeti veya çözümünün başarısı iyi bir ticari yönetime bağlıdır. Bu nedenle bu bilgi alanı yazılım mühendisliği ile ilgili faaliyetlerde alınan kararların ticari bakış açısıyla alınması konusu ile ilgilidir. Yazılım Mühendisliği Ekonomisi bilgi alanının alt konu başlıkları aşağıdaki gibidir [15]:

- Yazılım Mühendisliği Ekonomisinin Esasları
- Yaşam Çevrimi Ekonomisi
- Risk ve Belirsizlik
- İktisadi Analiz Yöntemleri
- Uygulamaya Yönelik Dikkate Alınması Gereken Hususlar

3.3.13. Bilgisayar temelleri

Bilgisayarın Temelleri Bilgi Alanının kapsamını, yazılımın oluştuğu ve işletildiği ortam olan bilgisayarlar meydana getirir. Bilgisayar ve onu oluşturan yazılım ve donanım hakkında bilgi sahibi olmak yazılım mühendisliğinin en temel gereklerindendir. Bu nedenle her yazılım mühendisi bu bilgi alanına hâkim olmak durumundadır. Bu bilgi alanının alt konu başlıkları aşağıdaki gibidir [15]:

- Problem Çözme Teknikleri
- Programlama Dillerinin Temelleri
- Algoritmalar ve Karmaşıklık
- İşletim Sitemlerinin Temelleri
- Ağ İletişiminin Temelleri
- Temel Geliştirici İnsan Unsuru
- Soyutlama
- Hata Ayıklama Araçları ve Teknikleri
- Bir Sistemin Temel Konsepti
- Derleyici Temelleri
- Paralel ve Dağıtık Hesaplama
- Güvenli Yazılım Geliştirme ve Bakım
- Programlamanın Esasları
- Veri Yapısı ve Sunumu
- Bilgisayar Organizasyonu
- Veri Tabanı Temelleri ve Veri Yönetimi
- Temel Kullanıcı İnsan Unsuru

3.3.14. Matematik temelleri

Yazılım profesyonelleri programlarla yaşar ve bunun için anlaşılmış bir mantığı koda çevirebilme becerisi gerekir. Bu nedenle Matematiğin Temelleri Bilgi Alanında ele alınan matematik, geleneksel, sayıların birbiri ile etkileşim halinde olduğu matematikten çok mantıksal işlemlerle ve muhakeme ile ilgilidir. Matematik Temelleri bilgi alanının alt konu başlıkları aşağıdaki gibidir [15]:

- Kümeler, İlişkiler ve Fonksiyonlar
- Kanıt Teknikleri

- Grafikler ve Ağaçlar
- Sonlu Makineler
- Sayısal Doğruluk, Kesinlik ve Hata
- Cebirsel Yapılar
- Temel Mantık
- Saymanın Temelleri
- Ayrık Olasılık
- Gramer
- Sayı Teorisi

3.3.15. Mühendislik temelleri

IEEE mühendisliği, sistematik, disiplinli ve ölçülebilir bir yaklaşımın, yapılara, makinelere, ürünlere, sistemlere veya süreçlere uygulanması olarak tanımlar [16]. Bu bilgi alanı yazılım mühendislerinde bulunmasının faydalı olduğu değerlendirilen bazı mühendislik temellerini kapsamaktadır. Mühendislik Temelleri bilgi alanının alt konu başlıkları aşağıdaki gibidir [15]:

- Ampirik Yöntemler ve Deneysel Teknikler
- İstatistiksel Analiz
- Ölçüm
- Mühendislik Tasarımı
- Modelleme, Simülasyon ve Prototip Çalışması
- Standartlar
- Kök Neden Analizi

3.4. Yazılım Proje Yönetimi

Yazılım projeleri özelleşmiş projelerdir ve proje yönetimi ile ilgili temel özellikleri taşırken, içeriği gereği özel yönetim teknikleri gerektirir. Yazılım projelerini diğer projelerden ayıran önemli farklar mevcuttur. Diğer mühendislik alanlarında meydana getirilen son ürünün fiziksel bir varlığı mevcutken, yazılım projelerinde ortaya çıkan ürün düşünsel süreçlerin bir tezahürü olduğu için soyuttur. Bu soyutluk hem proje yönetim süreçlerinin izlenebilirliğini sekteye uğratmakta, hem de müşterinin tam olarak ne istediğini anlatabilmesini zorlaştırdığı için proje başlangıcında tanımlanması gereken isterlerin doğru tanımlanmasını olumsuz

etkilemekte ve projenin belirsizliğini arttırmaktadır. Aynı zamanda bu soyutluk gerekli geri dönüşleri mümkün kılmaktadır. Örneğin %75'i tamamlanmış bir köprü projesinde imkânsız olan geriye dönük değişiklikler, %75'i tamamlanmış bir yazılım projesinde mümkündür. Genel proje yönetimine göre belirsizlikler daha fazla olduğu için yazılım projelerinde sarf edilecek iş gücünü ve maliyeti hesaplamak daha zordur ve başarısızlık ihtimali daha fazladır. Diğer mühendislik alanlarında icra edilen projeler zaman ve mekâna göre çok farklılık göstermediği için proje yönetimi konusunda tecrübeler edinilmekte ve yeni projelerde bu tecrübelerden yararlanılmaktadır. Yazılım projelerinde ise bilgi teknolojilerinin hızlı gelişimi ve müşteri isterlerindeki değişimler, mevcut proje içerisinde dahi değişikliğe sebep olabileceği gibi önceki projelerde kazanılmış deneyimi neredeyse geçersiz kılmaktadır. Bu nedenlerden dolayı yazılım projelerinde başarı oranları düşüktür. Projelerin başarılı olup olmadığını belirleyen ölçütler, proje yönetimi üçgeni olarak da bilinen zaman-kaynak-kapsam öğeleridir. Standish Group International tarafından 1994 yılından beri her yıl yazılım projelerinin başarı oranları konusunda yayınlanan CHAOS raporlarına göre, 2011-2015 yılları arasında yazılım projelerinde elde edilen başarı oranları Çizelge 3.1'de sunulmuştur.

Çizelge 3.1. 2011-2015 CHAOS Raporları başarı oranları [18]

	2011	**2012**	**2013**	**2014**	**2015**
Başarılı Projeler	% 29	% 27	% 31	% 28	% 29
Sorunlu Projeler	% 49	% 56	% 50	% 55	% 52
Başarısız Projeler	% 22	% 17	% 19	% 17	% 19

CHAOS raporlarında belirtilen başarılı projeler, planlanan zaman içerisinde, planlanan bütçe aşılmadan hedefe ulaşılan ve tatmin edici bir sonuç yaratan projelerdir. Sorunlu projeler, tamamlanan fakat zaman ve bütçe konusunda planlananın dışına çıkmış, ya da gereksinimler ve fonksiyonlar açısından hedeflerine ulaşamamış projelerdir. Başarısız projeler ise iptal edilmiş ya da tamamlanamamış projelerdir. CHAOS raporlarında önemle üzerinde durulması gereken bir diğer konu ise proje başarısına etki eden faktörlerdir. 2015 yılı için belirlenen faktörler ve proje başarısına etki puanları Çizelge 3.2'de sunulmuştur.

Çizelge 3.2. CHAOS Raporları Başarı faktörleri [18]

Başarı Faktörü	Proje Başarısına Etkisi
Üst Yönetim Desteği	% 15
Proje Çalışanlarının Duygusal Olgunluk Seviyesi	% 15
Müşteri/Kullanıcı Katılımı	% 15
Proje Optimizasyonu	% 15
Proje Çalışanlarının Yetenek Seviyesi	% 10
Projelerde uygulanan Yöntemlerin Standartlığı	% 8
Çevik Süreçlere Hakimiyet	% 7
Sade Yönetim	% 6
Proje Yönetimi Uzmanlığı	% 5
Açık ve Anlaşılır Hedefler	% 4

Çizelge 3.2'de belirtilen faktörlerin hemen hepsi proje yönetimi ve uygulanan metodoloji ile doğrudan ilgilidir. Yazılım projelerinde karşılaşılan düşük başarı yüzdeleri ve başarı faktörlerinin metodolojiyle olan bağıntıları ise uygulanan metodoloji konusuna verilmesi gereken önemi vurgulamaktadır.

3.5. Yazılım Proje Yönetimi Süreçleri

SWEBOK, yazılım mühendisliği yönetimini, "Yazılım ürünleri ve yazılım mühendisliği hizmetlerinin, etkin, verimli ve paydaşların fayda sağlayacağı şekilde ortaya konmasını sağlamak maksadıyla, planlama, koordinasyon, ölçme, izleme, kontrol ve raporlama gibi faaliyetlerin yönetimi" olarak tanımlamaktadır [15]. Yazılım mühendisliği yönetimi; yazılım proje yönetimi, yazılım mühendisliği ölçümlemesi ve yazılım mühendisliği araçları ile ilgili süreçlerden meydana gelmektedir. Yazılım proje yönetimi süreçleri, ikinci bölümde ele alınan proje yönetim süreç gruplarının yazılım projelerindeki karşılıklarıdır:

- Projenin Başlatılması ve Kapsam Tanımı,
- Yazılım Projesinin Planlaması,
- Yazılım Projesinin Yürütülmesi,

- Gözden Geçirme ve Değerlendirme,
- Kapanış.

Yazılım proje yönetimi süreçleri bölümü SWEBOK'tan yararlanılarak yazılmıştır.

3.5.1. Projenin başlatılması ve kapsam tanımlaması

Projenin başlatılması ve kapsam tanımlaması, yazılım projesine başlanması kararının verilmesi ile ilgili süreçleri kapsar. Bu süreçlerin odak noktası çeşitli yöntemler kullanılarak gereksinimlerin etkin bir şekilde elde edilmesi ve proje fizibilite çalışmasının analizidir. İcra edilen faaliyetler, gereksinimlerin görüşülmesi ve kararlaştırılması, fizibilite analizi ve gereksinimlerin gözden geçirilmesi ve gerekiyorsa düzeltilmesidir.

Gereksinimlerin görüşülmesi ve kararlaştırılması

Gereksinimlerin görüşülmesi ve kararlaştırılması, yerine getirilecek görevlerin belirlenmesi ve sınırların çizilmesini sağlar. Gereksinimler çeşitli tekniklerle müşteriden temin edilir, analiz edilir, tanımlamalar yapılır ve son olarak doğrulanır. Bu sayede kısıtlar ve hedefler belirlenerek projenin kapsamı ortaya çıkarılır.

Fizibilite analizi

Fizibilite analizinin amacı, mevcut teknoloji, kaynaklar ve sosyo-kültürel etkenler çerçevesinde, uygulanan çözümün en iyi çözüm olup olmadığının, kararlaştırılabilmesi için proje hedeflerinin açık bir şekilde tanımlanarak alternatif yaklaşımların değerlendirilmesidir.

Gereksinimlerin gözden geçirilmesi ve yenilenmesi

Proje ilerledikçe meydana gelen gelişmeler çerçevesinde gereksinimler özden geçirilmeli ve gerekiyorsa yenilenmelidir. Değişiklikler kabul edilecek ise değişimin etkilerinin ne şekilde olacağını belirlemek maksadıyla izlenebilirlik ve risk analizleri yapılmalıdır.

3.5.2. Yazılım projesinin planlaması

Proje planlaması sürecinde yapılması gereken ilk faaliyet, uygun bir yazılım geliştirme metodolojisinin seçimi ve gerekiyorsa metodolojinin, proje kapsamı,

yazılım gereksinimleri, risk değerlendirmesi, yazılımın kullanım alanı, fonksiyonel ve teknik karmaşıklık seviyesi ve yazılım kalite gereksinimleri çerçevesinde uyarlanması olmalıdır. Bütün paydaşların katıldığı bir risk değerlendirmesi yapılmalı ve yine bütün paydaşların mutabık olduğu bir risk profili çıkarılmalıdır. Planlama sürecinin bir parçası olarak yazılım kalite yönetimi süreçleri belirlenmelidir. Planlama sürecinde icra edilen faaliyetler, sürecin planlanması, teslim edilecek çıktıların kararlaştırılması, gayret-takvim-maliyet tahminleri, kaynak tahsisi, risk yönetimi, kalite yönetimi ve plan yönetimidir.

Sürecin planlanması

Süreç planlaması, uygulanan yazılım geliştirme metodolojisine göre farklılık göstermektedir. Şelale modeli gibi kestirimsel metotlar plan tabanlı yöntemler oldukları için proje başlangıcında yoğun planlama gerektirirler. Bu metotlarda yazılım proje faaliyetleri doğrusal bir sıralama halinde uygulanır. Çevik yöntemler gibi uyarlamalı metotlarda ise planlamaya çok zaman harcanmaz ve faaliyetler yinelemeli bir sırayla uygulanır.

Çıktıların kararlaştırılması

Yazılım mimari tasarım dokümanı, denetim raporları, testi bitirilmiş yazılım gibi bütün proje faaliyetleri sonunda elde edilecek ürünler tanımlanmalıdır. Önceki projelerde üretilmiş ürünlerin yeniden kullanımı ya da rafta hazır bekleyen ürünlerin değerlendirilmesi kararlaştırılmalıdır. İhtiyaç duyulması halinde hangi yazılımların tedarik edileceği, alt yüklenici kullanımı gerekiyorsa kullanım şartları belirlenmelidir.

Tahminler

Bir projede ya da projenin belli bir bölümünde ne kadar emek sarf edileceği, mevcut veri varsa eski projelerdeki tecrübelerle mukayese edilerek ya da uzman bilgisine başvurularak tahmin edilebilir. Kestirimsel metotlarda takvimin oluşturulması, görevlerin başlangıç ve bitiş zamanları ile süreleri, planlama safhasında belirlenir. Uyarlamalı metotlarda ise başlangıç için elde edilen bilgiler ışığında hesaplamalar yapılır ve uygulanacak tahmini yineleme sayısı ve sarf edilecek emek kapasitesi belirlenir fakat proje boyunca değişiklikler mümkündür.

Kaynak tahsisi

Tanımlanmış belli görevler için ekipman, personel ve tesis tahsisi yapılmalı, gerek projenin tamamı gerekse belli bölümleri için sorumluluklar dağıtılmalıdır. Kaynak tahsisi, özellikle insanların yeteneklerine göre uygun işlerde kullanımı ve kaynakların optimizasyonu için önemli bir konu başlığıdır.

Risk yönetimi

Risk ve belirsizlik ilgili fakat farklı kavramlardır. Belirsizlik bilgi eksikliğinden kaynaklanır. Risk ise proje üzerinde olumsuz bir etki yaratacak bir olayın meydana gelme ihtimalidir. Risk yönetimi, risk faktörlerinin tanımlanması, her risk faktörünün meydana gelme ihtimalinin ve potansiyel etkilerinin analizi, risk faktörlerinin önceliklendirilmesi, olasılıkların ve olumsuz etkilerin minimize edilmesi için risk azaltma stratejilerinin geliştirilmesi faaliyetlerini kapsar.

Kalite yönetimi

Bir yazılım projesi için yazılım kalite gereksinimleri hem nitel hem de nicel olarak tanımlanmalıdır. Yazılım kalite güvence ve kalite geliştirme süreçleri ile teslim edilecek ürünlerin doğrulama ve geçerleme süreçleri de kalite planlama sürecinde tanımlanmalıdır.

Plan yönetimi

Değişimin muhtemel olduğu yazılım projelerinde planlar yönetilmelidir. Yazılım geliştirme için seçilen plan ve süreçler devamlı izlenmeli, gözden geçirilmeli, raporlanmalı ve gerektiğinde yenilenmelidir.

3.5.3. Yazılım projesinin yürütülmesi

Yazılım projesinin yürütülmesi safhasında planlar uygulanır ve planlarda oluşturulan süreçler gerçekleştirilir. Seçilen yazılım geliştirme metodolojisinin süreçlerine uyulmalıdır. Yazılım projesinin yürütülmesi sürecinde icra edilen faaliyetler, planların uygulanması, yazılım tedariği ve tedarikçi sözleşme yönetimi, ölçüm süreçlerinin uygulanması, izleme süreci, kontrol süreci ve raporlamadır.

Planların uygulanması

Proje faaliyetleri proje planlarına göre yürütülmeli, kaynak kullanımı ve çıktı üretimi planlamalar doğrultusunda gerçekleştirilmelidir.

Yazılım tedariği ve tedarikçi sözleşme yönetimi

Yazılım tedariği ve tedarikçi sözleşme yönetimi, yazılım geliştirme organizasyonunun, üretilen yazılım ürünlerini satın alan müşterileri ve yazılım ürün ve hizmetlerinin sağlayıcısı olan alt yüklenicileri ile yaptığı sözleşmelerin yönetimini kapsar. Bu kapsamda yapılan sözleşmenin tipi; örneğin sabit fiyat, sabit maliyetler artı sabit fiyat ya da sabit maliyetler artı teşvik; belirlenir. Sözleşmede, hangi ürünlerin üretileceği, ne zaman teslimat yapılacağı, geç teslimat veya teslimat olmaması durumunda uygulanacak yaptırımlar belirtilir. Sözleşme uygulamaya konulduktan sonra projenin yönetilmesinin sözleşme şartları ile paralel yürütülmesi titizlikle takip edilmelidir.

Ölçüm süreçlerinin uygulanması

Projenin yürütülmesi sürecinde, ölçümü yapılacak ilgili veriler toplanmalıdır.

İzleme süreci

Proje süresince planlara bağlılık, alt görevlere ait çıktılar ve tamamlanma kriterleri devamlı izlenmelidir. Güncel efor sarfiyatı, takvime uyum ve maliyetler analiz edilmeli, kaynak kullanımı incelenmelidir. Proje risk profili göz önünde bulundurulmalı ve yazılım kalite gereksinimleri değerlendirilmelidir. Mevcut durumun beklenen durumdan farkını ortaya koyan sapma analizleri yapılmalıdır.

Kontrol süreci

Proje izleme süreçlerinin çıktıları, verilecek kararlar için temel teşkil eder. Risk faktörlerinin olasılıkları ve muhtemel etkileri gerektirdiğinde, projede değişikliklere gidilebilir. Bu değişiklikler düzeltici faaliyetler olabileceği gibi ek önlemler şeklinde de uygulanabilir. İstenmeyen olayları ve etkilerini bertaraf etmek için proje planı ve diğer proje dokümanları revize edilebilir. Hatta bazı durumlarda kontrol süreci projenin sonlandırılması ile de sonuçlanabilir.

Raporlama

Belirlenmiş zaman aralıklarında, projenin o ana kadarki ilerleyişi ile ilgili gerek kurum içi gerek kurum dışı paydaşlara rapor sunulmalıdır. Bu raporlar takım içi raporlar gibi detaylı olmak yerine hedef kitlenin bilgi ihtiyaçlarına hitap etmelidir.

3.5.4. Gözden geçirme ve değerlendirme

Önceden belirlenmiş zamanlarda ve ihtiyaç duyulduğunda, proje hedeflerine erişilme seviyesi ve paydaş gereksinimlerinin karşılanma durumu değerlendirilmelidir. Gözden geçirme ve değerlendirme sürecinde icra edilen faaliyetler, gereksinimlerin karşılanma seviyesinin belirlenmesi ve performansın gözden geçirilmesi ve değerlendirilmesidir.

Gereksinimlerin karşılanması

Paydaş memnuniyetinin sağlanması yazılım proje yöneticisinin öncelikli hedeflerinden biri olduğu için bu hedefe ne kadar ulaşıldığı periyodik olarak değerlendirilmelidir. Bu değerlendirme projenin önemli safhalarında ya da yineleme sonlarında elde edilen ürünler üzerinden yapılmalıdır. Gereksinimlerin karşılanması konusunda sapmalar yaşanıyorsa, bir önceki bölümde kontrol sürecinde sözü edilen tedbirler alınmalıdır. Alınan kararlar belgelenmeli ve ilgili paydaşlara sunulmalıdır.

Performansın gözden geçirilmesi ve değerlendirilmesi

Proje elemanlarının periyodik performans değerlendirmeleri, plan ve süreçlere bağlılık durumu ile takım içi çatışma gibi olası problem alanları hakkında fikir verir.

3.5.5. Kapanış

Bir proje, büyük bir safha ya da bir yineleme, bütün plan ve süreçleri tamamlandığında kapanış aşamasına gelmiş olur. Bu aşamada başarı kriterleri değerlendirilmelidir. Kapanış gerçekleştirilirken arşivleme, retrospektif inceleme ve süreç geliştirme faaliyetleri icra edilmelidir.

Kapanış kararının verilmesi

Belirlenen görevler tamamlandığında ve başarı kriterleri yerine getirildiğinde kapanış gerçekleşir. Yazılım gereksinimleri “karşılandı” veya “karşılanmadı” olarak onaylanır ve hedeflere ulaşılma derecesi belirlenir. Kapanış süreçleri paydaşların katılımı ile gerçekleştirilmeli ve sonuçlar paydaşların onayı ile belgelenmelidir.

Kapanış faaliyetleri

Kapanış kararı verildikten sonra paydaşların onayladığı yöntem, yer ve zamanda proje materyallerinin arşivlemesi, varsa hassas bilgi, yazılım ve cihazların imhası gerçekleştirilmelidir. Karşılaşılan sorunların, risklerin ve fırsatların ele alınacağı bir retrospektif analizi yapılmalıdır. Projeden alınan dersler çıkarılmalı ve kurumsal öğrenme ve gelişim bölümlerine aktarılmalıdır.

4. YAZILIM GELİŞTİRME METODOLOJİLERİ

4.1. Yazılım Yaşam Çevrimi

Yazılım geliştirme yaşam çevrimi (SDLC), yazılım gereksinimlerini, yazılım ürününe dönüştüren yazılım süreçlerini kapsar. Yazılım ürün yaşam çevrimi (SPLC) ise bir yazılım geliştirme yaşam çevrimine ek olarak yazılım konfigürasyon yönetimi ve yazılım kalite güvence süreçleri ile birlikte yazılımın kullanıma alınması, bakımı, desteği, değerlendirilmesi ve kullanımdan çıkarılması gibi "baştan-sona" süreçleri kapsar. Tekil olarak yazılım geliştirme süreçleri arasında zamansal bir sıralama yoktur. Bu süreçler arasındaki sıralamaları yazılım yaşam çevrimi modelleri (SDLC ya da SPLC) yaratır. Yaşam çevrimi modelleri kendi felsefelerine göre önemli süreçleri tanımlarlar ve bu süreçler arasındaki sıralama ve etkileşimleri belirlerler [15]. Yaşam çevrimi modelleri yelpazesi kesin ayrım çizgileri olan doğrusal bir yapı değildir. Bu yelpazenin üzerinde birbirine çok benzer ve birbirinden çok farklı metotlar yer alır ve bir ucunda plan tabanlı şelale modeli bulunurken diğer ucunda uyarlamalı metotlardan çevik yöntemler yer almaktadır. Bu bölümde, çalışmanın asıl konusunu teşkil eden karşılaştırmanın yapılacağı şelale modeli ve çevik yöntemler ile ilgili bilgi sunulacaktır.

4.2. Şelale Modeli

Şelale modeli, yazılım projelerinde uygulanan faaliyetlerin ardışık safhalar halinde icra edildiği, yazılım mühendisliğinin en eski ve temel modelidir. Günümüzde kamu kuruluşları ve büyük şirketler tarafından her türlü projenin yönetim standardı olarak kabul görmektedir [19].

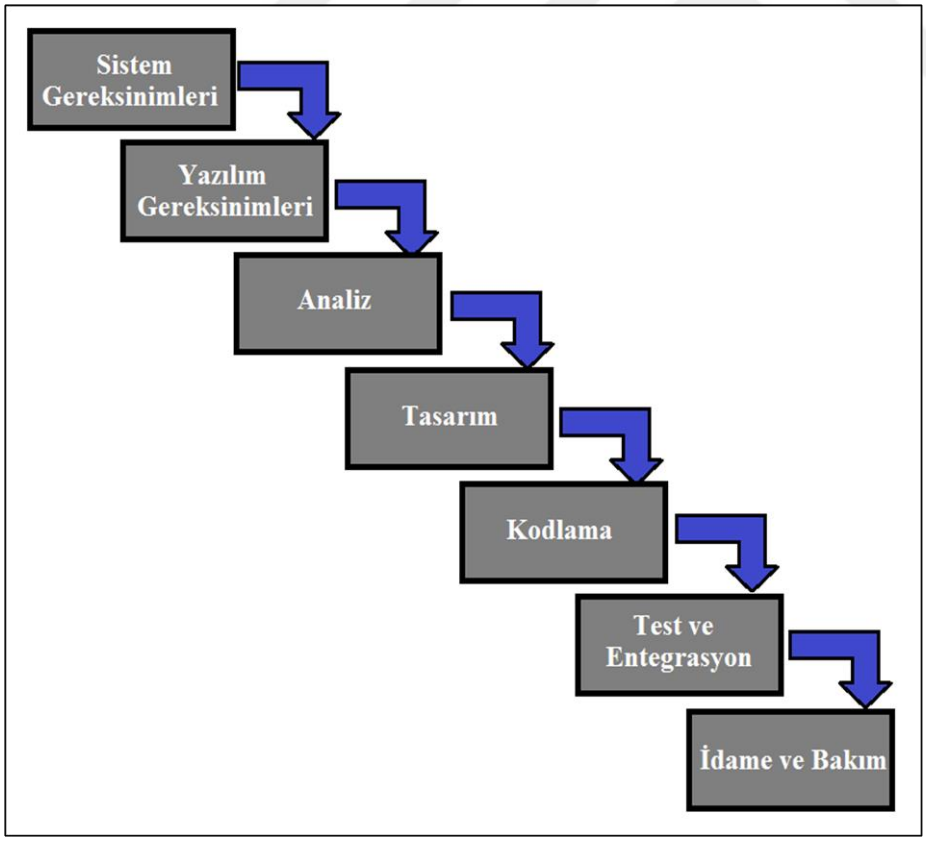
4.2.1. Tarihçesi

1950'li yıllarda uygulanmakta olan ilk yazılım geliştirme yöntemi "kodla ve düzelt" daha çok bilinen ismiyle "ad hoc ya da kovboy programlama", herhangi bir planlama veya tasarım safhası olmadan, doğrudan gereksinimlerin kodlanması ve süreç içi ya da ürün teslimi sonrası karşılaşılan hataların düzeltilmesi prensibine dayanmaktaydı. Zaman içinde yazılım projelerinin büyüyerek bir proje yönetim metodolojisi gerektirmesi, kritik sistemlerde hatalara tolerans gösterilemeyecek olması, teknolojinin gelişerek bilgisayar kapasitelerinin hızla artması ve kaliteli

yazılım üretilememesi gibi nedenlerden dolayı 1960'lı yıllarda yaşanan yazılım krizi sonucu NATO Bilim Komitesi tarafından 1968 yılında Almanya'da Yazılım Mühendisliği Konferansı düzenlendi [20]. Bu konferansta üzerinde durulan temel konular yazılım mühendisliği kavramının tanımlanması ve yazılım geliştirme süreçlerinde uygulanması gereken standartların tespit edilmesidir [20]. Yakın bir zaman sonra 1970 yılında Dr. Winston Royce tarafından yayınlanan bir bildiride Şelale Modeli tanımlandı [21]. Her ne kadar Dr. Royce tarafından önerilen metodoloji evrimsel ve yinelemeli süreçler ile geliştirme süresince müşterinin katılımı gibi yaklaşık 30 yıl sonra "çevik" olarak adlandırılacak uygulamalara vurgu yapsa da [22] Şelale yöntemi zaman içinde bugün bildiğimiz şeklini aldı ve yazılım geliştirme metodolojilerinin temeli haline geldi.

4.2.2. İlkeleri ve özellikleri

Şelale modelinde uygulanan safhalar Şekil 4.1'de gösterilmiştir.



Şekil 4.1. Şelale modelinde uygulanan safhalar

Her safhanın sonunda genellikle dokümantasyondan oluşan bir çıktı elde edilir ve her çıktı kendini takip eden safhanın girdisini oluşturur [23]. Çıktı-girdi ilişkisi

nedeniyle yaşanacak olası değişimlerin projeye maliyeti yüksek olacağı için safhalar arasında atlama veya büyük çapta geri dönüşlere izin verilmez. Şelale modelinin uygulandığı projelerde safha bitişleri net bir şekilde tanımlanmıştır ve her safha değişiminde katı incelemeler, yoğun dokümantasyon ve yönetim onayı gibi kontrollerle proje yoğun bir denetim altında icra edilir. Bu nedenle uygulanması katı bir disiplin gerektirir. Bu model, “önden büyük tasarım” felsefesini benimser ve kodlama safhasından önce büyük bir analiz ve tasarım yapılır [23]. Sistem ve yazılım gereksinimleri, gerekli her şeyin baştan bilindiği ve değişiklik olmayacağı faraziyeleri kabul edilerek en başta belirlenir [22]. Bu nedenle belirsizlik içermeyen tam anlaşılmış ve değişme ihtimali olmayan projeler için daha uygundur [14] çünkü şelale modelinin en büyük zayıflığını bu faraziyeler oluşturmaktadır. Müşteri, gereksinimler tanımlanırken projenin içindedir, analiz, tasarım ve kodlama safhalarında görünmez, test ve entegrasyon safhasında yeniden ortaya çıkar [22]. Şelale modelinde en çok planlama, zaman çizelgeleri, hedef tarihler, bütçe ve bütün sistemin bir defada kullanıma alınması konuları üzerinde durulur.

Bu yaklaşımın temel amacı proje süresince değişikliklere izin vermeyerek, kapsam, zaman ve kaynaklar gibi faktörleri baştan sabitlemek ve büyük bir risk faktörü olan değişimi etkisiz kılmaktır [24]. Şelale modelinin ortaya çıktığı zaman ve şartlar değerlendirildiğinde, iş süreçleri bilgi sistemlerine günümüzde olduğu kadar bağımlı değildi ve projelerin aceleyle bitirilmesi gerekmiyordu. Teknolojik gelişme ve iş yaşamındaki değişim günümüzde olduğu kadar hızlı değildi bu yüzden projeler günümüzde olduğu kadar hızlı bir tempoyla ilerlemiyordu. Teknoloji ve kaynak kısıtlarından dolayı bilgi sistemleri donanım merkezliydi ve yazılımın ana maksadı, depolama kapasitesi ve işlemci gücü gibi kısıtlı donanımsal kaynakları en optimum şekilde kullanabilmekti [25]. Bu nedenlerle ortaya çıktığı zamanlarda şelale modeli daha çok bir risk yönetim metodu olarak görülmekteydi ve bu riskin yönetilebilmesi için gerekli görülen ve daha sonra şelale modeli ile birlikte geleneksel proje yönetim dünyasının temeli olan hususlar şunlardı:

- Planlama ve bütçelemenin doğru yapılabilmesi için bütün gereksinimler erkenden tanımlanmalıydı.
- Gereksinimleri erkenden tanımlamak ve değişiklik olmayacak şekilde

sabitlemek kapsam aşımını önleyecekti.

- Gelecekte referans oluşturması için bütün tasarım ve karar verme süreçleri belgelenmeliydi.
- Safhalar arası yoğun incelemeler bütün paydaşların süreçten haberdar olmasının yanında doğabilecek problemlerin erken tespit edilmesini sağlayacaktı.
- Mimarinin uygun olduğundan emin olunabilmesi için, tasarım faaliyetleri başlamadan önce bütün gereksinimler belgelenmeliydi.
- Tüm sistemin kabiliyetlerinin değerlendirildiğinden emin olunabilmesi için test işlemlerinden önce bütün kodlama işlemleri tamamlanmalıydı.
- Sistem uygulamaya koyulmadan önce bütün gereksinimler karşılanmalıydı [22].

4.2.3. Faydaları

- Projenin başlangıcında gereksinimlerin açık, anlaşılır ve tam olarak tanımlanması, daha doğru zaman ve maliyet tahmini yapılmasını sağlayarak projedeki belirsizlikleri azaltır.
- Çok kullanılan bir metot olduğu için genelde teoride iyi bilinir.
- Doğrusal bir süreç olduğu için uygulanması kolaydır.
- Kimin ne zaman ne yapması gerektiği katı bir şekilde tanımlandığı ve takip edildiği için kendi başına iş yapma yeterliliği ve tecrübesi olmayan proje elemanları, proje yöneticileri ve performansı iniş çıkışlı proje ekipleri için idealdir.
- İyi bir zaman, maliyet ve gayret tahmini yapıldığı takdirde tahsis edilmiş kaynakların aşımı ihtimali minimaldir.
- Bir safhaya başlanmadan önce bir önceki safha tamamlandığı için proje ilerleyişinin takibi kolaydır. Ayrıca eş zamanlı ilerleyen safha olmaması kaynak takibini de kolaylaştırır.
- Dokümantasyonun devamlı bir faaliyet olması proje sonunda dokümantasyon maksadıyla geçmişe dönme emeğini engellediği gibi projenin herhangi bir zamanında dokümantasyonun mevcut duruma göre tam olmasını sağlar ve bu dokümantasyon üzerinden projenin mevcut halinin değerlendirilebilmesine imkân verir.

- Şelale modeli disiplinli ve istikrarlı bir modeldir ve proje elemanlarına projenin iyi bir şekilde yönetildiği ve her şeyin kontrol altında olduğu hissiyatını verir.
- Müşterinin projeye ne zaman ve ne kadar katkı yapacağı bellidir ve proje boyunca zaman ayırması beklenmez.
- Fonksiyonel organizasyonlarda birden fazla projede görev alan proje elemanlarının işlerinin üst üste binmesini engeller. Örneğin analiz çalışması tamamlandıktan sonra başka bir projede görev almaya devam eden bir elemanın geri dönüşü gerekmez.
- Hazırlanan detaylı ve kapsamlı dokümantasyon bir iletişim aracı olarak kullanılabilir ve özellikle büyük projelerde ve bütün elemanların aynı ortamda bulunmadığı projelerde yüz yüze iletişim ihtiyacını azaltır. [19, 23, 26]

4.2.4. Kısıtları

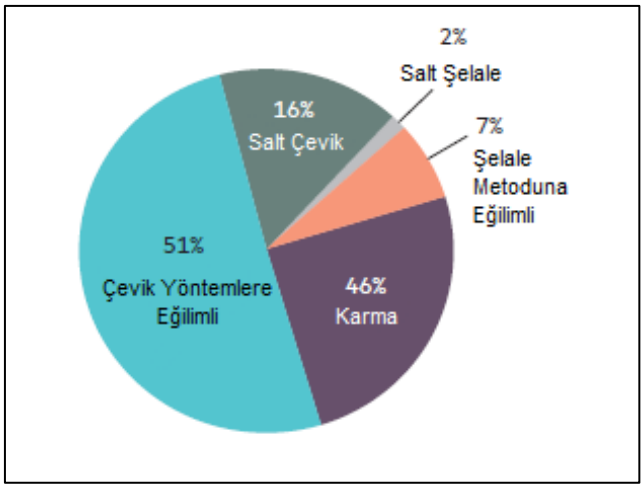
- Değişiklik, geliştirme veya düzeltme maksatlı geriye dönüşler şelalede akıntıya karşı yüzmeye benzetilir, zor ve maliyetlidir. Bu yüzden değişikliklere cevap veremez ve süreç içerisinde gelen istekler çoğunlukla kabul edilmez.
- Herhangi bir safhada tespit edilmemiş bir olumsuzluk sonraki bütün safhaları etkiler.
- Yapısal özelliği ve sıkı kontrolleri nedeniyle esnek olmayan, yavaş ve hantal bir doğası vardır.
- Gereksinimlerin proje başında tam ve anlaşılır bir şekilde tanımlanmasının beklenmesi çok gerçekçi olmayabilir ve müşterilerin ihtiyaçlarını tam ve doğru ifade edememesi proje sonunda eksik veya yanlış bir ürün elde edilmesine neden olabilir.
- Geliştiricilerin gereksinimleri müşteri veya kullanıcılarla yüz yüze görüşerek anlamaları yerine analiz ve tasarım safhalarında hazırlanan dokümanlardan okumaları, gereksinimlerin anlaşılmasını zorlaştırır, geliştiricilerin girdi yapmasına imkân vermez.
- Gereksinim tutarsızlıkları, eksik sistem bileşenleri ve beklenmeyen geliştirme ihtiyaçları, ancak tasarım veya kodlama aşamasında

belirlenebilir.

- Sistem performansı tüm kodlama tamamlanana kadar ölçülemez ve tüm sistem testine kadar hatalar tespit edilemez.
- Yoğun ve güncel dokümantasyon zaman ve emek gerektiren bir faaliyettir ve genel olarak geliştiriciler tarafından gereksiz bir yük olarak görülür.
- Yazılı sistem tanımlamalarının kullanıcı veya müşteriler tarafından okunarak tam olarak anlaşılması ve uygulanması zordur.
- Bir safhanın tamamlanmadan sonraki safhanın başlamaması nedeniyle, tüm proje elemanlarını ilgilendirmeyen gecikmeler dahi bütün proje elemanları için zaman israfına neden olabilir.
- Yüksek yönetim giderleri küçük projeler ve küçük takımlar için gereksiz maliyetlerdir.
- Yazılım geliştirme yaratıcılık gerektirir ve şelale modelinin yüksek disiplinli yaklaşımı yaratıcılığı olumsuz etkiler. [19, 23, 27, 28]

4.3. Çevik Yöntemler

Çevik yazılım geliştirme, kendi kendini organize eden takımlar tarafından, son derece işbirlikçi bir anlayışla, “yeteri kadar” resmiyet içeren etkili bir yönetişim çerçevesinde, zamanında, uygun maliyetli ve paydaşların değişen ihtiyaçlarını karşılayan yüksek kalitede çözümler üreten yinelemeli ve artımlı (evrimsel) bir yaklaşımdır [29]. Çevik yöntemler, geleneksel yöntemlerin yetersiz kaldığı değerlendirilen konular için alternatif çözümler olarak ortaya çıkmıştır. HP‘nin 2017 yılında yaptığı bir araştırmaya göre organizasyonlar yazılım proje yönetiminde daha çok çevik yöntemleri tercih etmektedir [30]. İlgili araştırma sonucu elde edilen veriler Şekil 4.2’de sunulmuştur. AgileTurkey tarafından yayınlanan Türkiye Çeviklik Raporuna göre de incelenen organizasyonların %72’si çevik yöntemleri kullanmaktadır [31].



Şekil 4.2. Organizasyonlarda uygulanan geliştirme yöntemleri [30]

4.3.1. Tarihçesi

XP [32], Scrum [33], eXtreme Test [34], Crystal Metodoloji Ailesi [35], DSDM [36], ASD [37] ve FDD [38] gibi çevik yöntemleri meydana getiren başlıca metodolojiler, 1990'ların ikinci yarısından itibaren ortaya çıkmaya başladılar [24]. 2001 yılında, çevik metotların yaratıcısı ve önde gelen uygulayıcılarından on yedi yazılım mühendisi, dokümantasyona dayalı geleneksel yazılım geliştirme süreçlerine bir alternatif bulabilmek ve uyguladıkları metodolojilere dayalı ortak bir zeminde bulaşabilmek maksadıyla bir araya geldiler [39]. Kendilerine çevik ittifak adını veren bu uzmanlar, çalışmanın sonucunda çevik yöntemlerin temel ilke ve değerlerini kararlaştırarak Çevik Bildiriyi yayınladılar [40]. Yapılan araştırmalara göre çevik yöntemlerin benimsenmesi dünyada ve Türkiye'de 2010 yılından itibaren büyük bir yükselişe geçmiştir [30, 31].

4.3.2. İlkeleri ve özellikleri

2001 yılında Çevik İttifak tarafından yayınlanan Çevik Bildiri felsefesine göre:

- Süreçler ve araçlardan ziyade bireyler ve etkileşimlere,
- Kapsamlı dokümantasyondan ziyade çalışan yazılıma,
- Sözleşme pazarlıklarından ziyade müşteri ile iş birliğine,
- Bir plana bağlı kalmaktan ziyade değişime karşılık vermeye değer verilir [40].

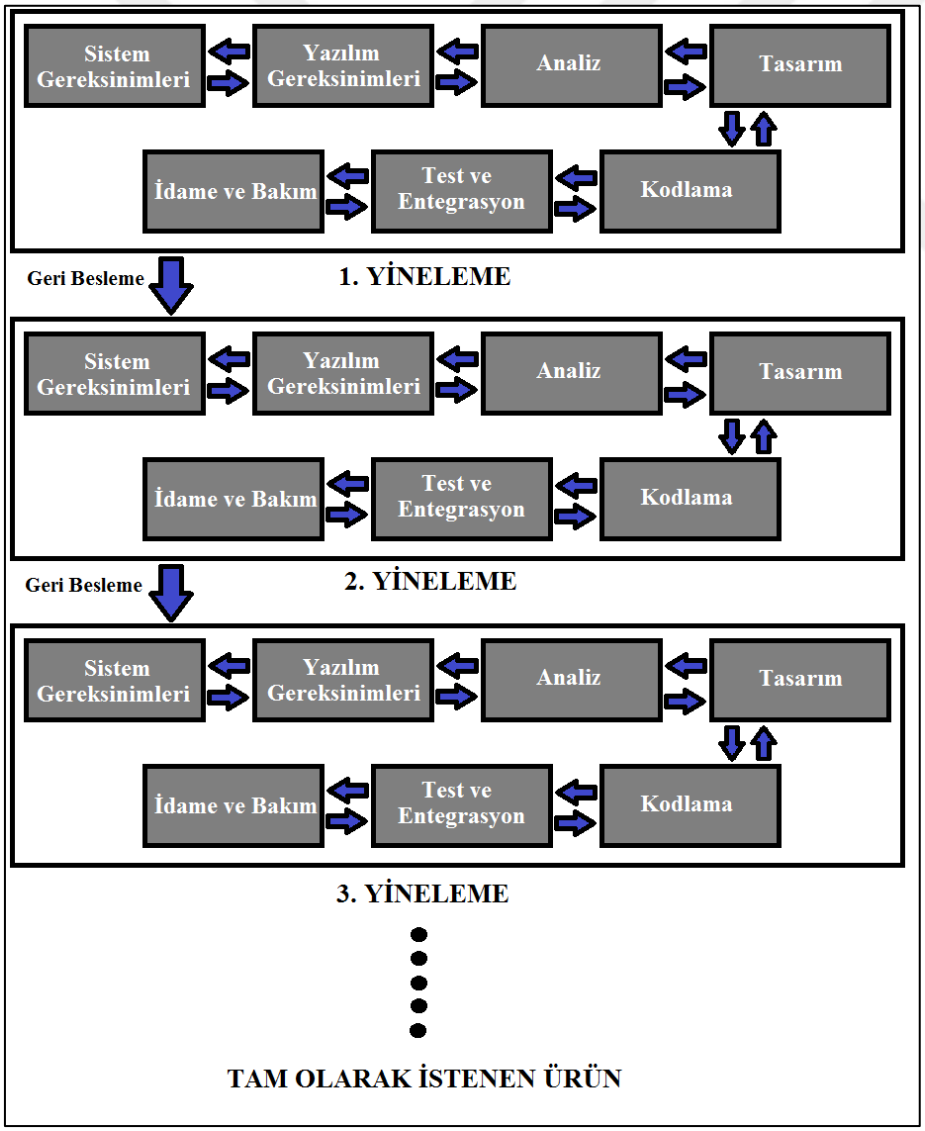
Yukarıda temel değerleri belirtilen çevik yöntemlerin, bu felsefeden doğan on iki ilkesi vardır. Bu ilkeler:

- En önemli öncelik değerli yazılımın erken ve devamlı teslimini sağlayarak müşterileri memnun etmektir.
- Değişen gereksinimler yazılım sürecinin son aşamalarında bile kabul edilmelidir. Çevik süreçler değişimi müşterinin rekabet avantajı için kullanır.
- Çalışan yazılım, tercihen kısa zaman aralıkları belirlenerek birkaç haftada ya da birkaç ayda bir düzenli olarak müşteriye sunulmalıdır.
- İş süreçlerinin sahipleri ve yazılımcılar proje boyunca her gün birlikte çalışmalıdırlar.
- Projelerin temelinde motive olmuş bireyler yer almalıdır. Onlara ihtiyaçları olan ortam ve destek sağlanmalı, işi başaracakları konusunda güven duyulmalıdır.
- Bir yazılım takımında bilgi alışverişinin en verimli ve etkin yöntemi yüz yüze iletişimdir.
- Çalışan yazılım ilerlemenin birincil ölçüsüdür.
- Çevik süreçler sürdürülebilir geliştirmeyi teşvik etmektedir. Sponsorlar, yazılımcılar ve kullanıcılar sabit tempoyu sürekli devam ettirebilmelidir.
- Teknik mükemmeliyet ve iyi tasarım konusundaki sürekli özen çevikliği artırır.
- Sadelik, yapılmasına gerek olmayan işlerin mümkün olduğunca arttırılması sanatı, olmazsa olmazlardandır.
- En iyi mimariler, gereksinimler ve tasarımlar kendi kendini örgütleyen takımlardan ortaya çıkar.
- Takım, düzenli aralıklarla nasıl daha etkili ve verimli olabileceğinin üzerinde düşünür ve davranışlarını buna göre ayarlar ve düzenler [40].

Çevik yöntemlerde uygulanan yinelemeler, önceki yinelemelerde elde edilen tecrübeler ve tespit edilen aksaklıklar doğrultusunda şekillenir. Yapılması gereken görevler, taşıdıkları iş değerine göre önceliklendirilir. Yapılacak işin en iyi nasıl yapılacağını, mevcut kaynaklar ve kısıtlar çerçevesinde proje takımı kendi belirler. Takım belli görevleri belli sürelerde (yineleme süresi içinde) tamamlamalıdır. Yineleme sonunda teslim edilecek ürünü meydana getirmekten sorumlu olan güçlü

ve zayıf yanlarıyla takımdır. Bu nedenle takım içi iş birliği önem arz eder. Bu kapsamda çevik yöntemlerin dayandığı temel esaslar, deneysellik, önceliklendirme, kendi kendini örgütleme, zaman çerçevesi, iş birliği şeklinde sıralanabilir [41].

Çevik yöntemlerde yazılım geliştirme faaliyetleri yinelemeli safhalar halinde uygulanır. Bu yinelemeler sonucunda kullanılabilir ürün ortaya çıkarılır ve müşteri veya kullanıcıların geri beslemeleri ışığında ve varsa değişen gereksinimler doğrultusunda geliştirme süreçleri tekrarlanır. Bu yinelemeli süreçler müşterinin tam olarak istediği ürün ortaya çıkana kadar sürdürülür. Çevik yöntemlerde uygulanan safhalar Şekil 4.3'te gösterilmiştir.



Şekil 4.3. Çevik yöntemlerin genel uygulaması

Başlıca çevik metodolojilere ait kullanım oranları Çizelge 4.1'de sunulmuştur. Gerek dünya genelinde gerekse Türkiye'de en popüler metodoloji Scrum olmakla birlikte XP ve Kanban diğer metodolojilere göre öne çıkmaktadır. Scrum yönteminin Kanban ve XP ile beraber kullanıldığı ve organizasyonların kendi ihtiyaçlarına göre uyarladığı karma metodolojiler de çevik yöntemlerin başlıca kullanım şekillerindendir. VersionOne [42] araştırmasında katılımcılardan tek metodoloji belirtmeleri istenirken, Forrester [43] ve AgileTurkey [31] araştırmasında çoklu seçeneğe izin verilmiştir. Bu nedenle Forrester araştırmasında elde edilen Scrum, XP ve Kanban verilerine karma uygulamalar da dahildir.

Çizelge 4.1. Başlıca çevik yöntemlerin uygulanma oranları

	VersionOne	**Forrester**	**AgileTurkey**
Scrum	% 58	% 86	% 65
XP	% 1	% 29	% 7
Kanban	% 5	% 57	% 32
Scrum/XP Karma	% 10	-	% 8
Scrum/Kanban Karma	% 8	-	% 7
Çoklu Karma Uyarlamalar	% 8	-	% 8

Çevik yöntemlerde tatbik edilen temel ortak uygulamalar bulunmaktadır. Bu uygulamaların başlıcaları şu şekildedir:

- Günlük Ayaküstü Toplantılar
- Önceliklendirilmiş İş Listeleri
- Kısa Süreli Yinelemeler (2-4 hafta)
- Yineleme Sonu Retrospektif Değerlendirme Toplantıları
- Yineleme Planlamaları
- Hikâye Tabanlı Gereksinimler
- Erken ve Sık Geri Besleme
- Devamlı Entegrasyon

- Devamlı Ürün Teslimi
- Eşli Programlama

4.3.3. Faydaları

- Çevik yöntemlerin felsefesi gereği proje süresince değişikliklerin hoş karşılanması ve hatta teşvik edilmesi nedeniyle ve yinelemeli süreç yapısı sayesinde esneklik yüksektir ve değişen şartların yönetilmesi kolaydır.
- Ayaküstü kısa toplantılar ve yüz yüze iletişim felsefesi ile çevik yöntemlerde uygulanan insan ve iletişim merkezli süreçler sayesinde proje elemanlarının moral ve motivasyonu artar, değerli bilgi ve görüşlerin paylaşımı ile üretkenlik seviyeleri yükselir.
- Kısa süreli yinelemeler ve yineleme sonucunda teslim edilen çalışır ürün sayesinde projenin ilerlemesi daha sağlıklı bir şekilde gözlemlenir ve proje riski azalır, muhtemel hatalar daha erken tespit edilir, teslimat öngörülebilirliği artar ve ürün pazara daha çabuk ulaşır.
- Müşteri veya kullanıcıların proje sürecinde proje elemanları ile yüz yüze iletişim kurabilmeleri sayesinde iş süreçleri ile yazılım geliştirme süreçlerinin entegrasyon seviyesi yükselir, istenen özelliklerin tam olarak ifade edilebilmesiyle yazılımın fonksiyonel kalitesi artar ve müşteri memnuniyeti sağlanır.
- Yineleme sonunda teslim edilen çalışır üründen elde edilen geri beslemeler vasıtasıyla tespit edilen aksaklıklar bir sonraki yinelemelerde ele alınarak yazılımın bakım ve idame süreci kolaylaşır ve bakım-idame maliyetleri azalır.
- Çevik yöntemler, uygulanan safhalar arası geçişlere ve geri dönüşlere müsait olduğu için süreç devam ederken tespit edilen geriye dönük aksaklıklar kolaylıkla düzeltilebilir.
- Genel olarak üretilen yazılımın sadece ürün özellikleri dokümantasyona çevrilir ve diğer süreçler hariç tutulur. Hafif dokümantasyon iş yükünü azaltır.
- Entegrasyon ve test faaliyetlerinin bütün yinelemelerde icra edilmesi, bu konuda çalışan elemanların işlerinin yığılmasını engelleyerek daha istikrarlı bir emek sarf edilmesini sağlar. [42-45]

4.3.4. Kısıtları

- Çevik yöntemlerde süreçler iletişim merkezli olduğu için proje elemanlarının birbirleriyle veya müşteriyle aynı ortamda bulunmaması, proje gelişimini olumsuz etkiler. Video konferans gibi uzaktan iletişim kurulabilecek teknolojiler kullanılabilir fakat bu da gerekli altyapı ve teknoloji nedeniyle maliyetleri arttırır.
- Alt yüklenicinin teklif verebilmesi ve bir kontrat yapılabilmesi için istenenlerin kesin olması ve değişmemesi gerekir. Çevik yöntemlerin belirsiz ve değişken ortamı alt yüklenici uygulamalarını zorlaştırır.
- Kalabalık projelerde iletişim ve etkileşim daha az olacağından çevik yöntemlerin etkinliği azalır.
- Fonksiyonel olarak bütünlük gerektiren büyük ve karmaşık sistemler yinelemeler için uygun parçalara bölünemeyebilir.
- Zaman ayıramama veya isteksiz olma gibi nedenlerle müşteri veya son kullanıcıların proje içerisinde olma oranı düşükse gereksinimler doğru biçimde tanımlanamayabilir.
- Değişime açık olması nedeniyle projenin kaynak ve kapsam planlaması zordur.
- Dokümantasyon hazırlama veya güncelleme süreci, değişen ortam ve gereksinimler nedeniyle karmaşıktır.
- ” Gerektiği kadar dokümantasyon” anlayışı ile hazırlanan dokümantasyonun yetersiz olması durumunda projeye sonradan katılan geliştiriciler, yasal düzenleme ile sertifikasyon süreçleri ve bakım-idame safhaları olumsuz etkilenir.
- Takımlar kendi kendini organize ettiği için bireylerin yetenekli, tecrübeli ve çevik süreçlere hâkim olması gerekir. Yeteneksiz, tecrübesiz elemanlar takım performansını düşürerek proje gelişim sürecini olumsuz etkiler.
- Önceliklerini tam olarak belirleyemeyen ve yapılacaklar listesindeki öncelikleri sık sık değiştirmek isteyen müşteriler planlamaları olumsuz etkileyebilir.
- Çevik yöntemlerle, spesifik problemler veya fonksiyonlara göre tanımlanan gereksinimleri karşılayacak yeterlikte çözüm geliştirildiği için, ürünlerin diğer projelerde yeniden kullanılabilirliği olmaz.

- Yazılım hatasının insan hayatını tehlikeye attığı veya büyük maddi zararlar doğurduğu hata toleransı olmayan sistemler için çevik yöntemlerin test ve kalite kontrol yöntemleri yeterli olmayabilir.
- Çevik yöntemlerde müşteri faydası ön planda tutulur fakat müşteri faydası genel olarak müşterinin istediği fonksiyonların karşılanmasıdır. Yalın geliştirmenin bir kalıtımı olarak basitlik felsefesi uygulandığı için kullanıcı ara yüzü ve kullanıcı deneyimi gibi kullanıcı tatmini ile ilgili konular göz ardı edilebilir. [27, 44, 46, 47]

5. ŞELALE MODELİ VE ÇEVİK YÖNTEMLERİN KARŞILAŞTIRILMASI

Bu tezde, yazılım proje yönetimi metodolojileri yelpazesinin iki ucunda yer alan şelale modeli ile çevik yöntemler, proje başarısına etki ettiği değerlendirilen çeşitli faktörler açısından karşılaştırılmıştır. Şelale modeli, günümüzde devlet kurumları ve büyük şirketler tarafından her türlü projenin yönetim standardı olarak kabul görmektedir [19]. Çevik yöntemler ise, geleneksel yöntemlerin yetersiz kaldığı değerlendirilen konular için alternatif çözümler olarak ortaya çıkmıştır ve HP'nin 2017 yılında yaptığı bir araştırmaya göre yazılım proje yönetiminde en çok tercih edilen metodoloji çevik yöntemlerdir [30].

Projelerin başarı oranları uygulanan metodolojiye göre farklılık göstermektedir. Standish Group International tarafından yayınlanan 2015 CHAOS raporunda verilen, 2011-2015 yılları arasında 10000'in üzerinde yazılım projesinde uygulanan metodolojiye göre elde edilen başarı oranları Çizelge 5.1'de sunulmuştur.

Çizelge 5.1. Uygulanan metodolojiye göre yazılım projeleri başarı oranları [18]

	Şelale Modeli	Çevik Yöntemler
Başarılı Projeler	% 11	% 39
Sorunlu Projeler	% 60	% 52
Başarısız Projeler	% 29	% 9

Çizelge 5.1 incelendiğinde, elde edilen veriler ışığında çevik yöntemlerle yönetilen projelerin başarı oranlarının şelale modeli ile yönetilen projelerin başarı oranlarının yaklaşık 4 katı, başarısızlık oranlarının ise 3'te 1'i olduğu görülmektedir. Buna rağmen çevik yöntemlerin başarı ihtimalinin şelale modeline göre her zaman daha fazla olacağı söylenemez. Proje başarısı için proje şartlarına uygun metodoloji seçilmelidir.

5.1. Benzer Çalışmalar

Literatür incelendiğinde yazılım geliştirme metodolojileri ile ilgili çeşitli karşılaştırma çalışmalarına rastlanmaktadır. Bu çalışmalar, çevik yöntemlerin kendi içindeki karşılaştırmaları, çeşitli geleneksel yöntemlerle çevik yöntemlerden bazılarının karşılaştırılması ve metodolojilerin tekil faktörler açısından karşılaştırılması olarak gruplanabilir.

Çevik yöntemlerin kendi içinde karşılaştırıldığı çalışmalar

Çevik yöntemlerin kendi içindeki karşılaştırma çalışmalarının başlıcalarından Hiwarkar ve diğerleri [48] beş çevik yöntemi (Scrum, XP, DSDM, ASD, FDD), dokümantasyon, müşteri etkileşimi, toplantılar ve proje büyüklüğü ile karmaşıklığı konularında karşılaştırmış; Flora ve Chande [44], çevik yöntemlerle ilgili sistematik bir çalışma ortaya koyarak, on çevik yöntemin (XP, Scrum, LSD, Kanban, ASD, FDD, DSDM, AM, Crystal, AUP) ilkelerini ve özelliklerini inceleyerek çevik yöntemlerin avantaj ve dezavantajlarını ortaya koymuş; Qumer ve Henderson-Sellers [49], yaptıkları çeviklik tanımı ve geliştirdikleri analitik yöntem ile altı çevik yöntemi (XP, Scrum, FDD, ASD, DSDM, Crystal) çeviklikleri açısından karşılaştırmışlardır.

Geleneksel yöntemler ve çevik yöntemlerden bazılarının karşılaştırıldığı çalışmalar

Çeşitli geleneksel yöntemlerle çevik yöntemlerden bazılarının karşılaştırılmasına örnek teşkil edecek başlıca çalışmalardan Munassar ve Govardhan [19], beş yazılım geliştirme metodolojisini (Şelale, Yinelemeli Geliştirme, V-Model, Spiral Model, XP) özellikleri, avantajları ve dezavantajları açısından incelemiş; Vohra ve Singh [24], yazılım geliştirme süreçlerindeki ilerlemeyi ve değişimi anlatarak plan tabanlı, değişim tabanlı ve bileşen tabanlı yaklaşımları genel hatlarıyla tarihçeleri, avantaj ve dezavantajları ile birlikte ele almış; Balaji ve Murugaiyan [26], şelale modeli, V model ve çevik yöntemleri özellikleri, fayda ve kısıtları şeklinde incelemiş; Mahapatra ve Goswami [50], altı yazılım geliştirme metodolojisini (şelale modeli, prototip modeli, yinelemeli geliştirme, spiral geliştirme, RAD, XP) belirli proje karakteristikleri açısından karşılaştırmış; Simao [51] ise beş metodolojiyi (şelale modeli, spiral model, RAD, Scrum ve XP) temel özelliklerinin

SWEBOK bilgi alanları tanımlarını karşılaması üzerinden incelemiştir. Geleneksel yöntemlerin temeli olarak görülen şelale modelinin, çevik yöntemlerin en çok kullanılan metodolojileri Scrum ve XP'den biriyle karşılaştırılması da literatürde karşımıza çıkan çalışma konularıdır. Mahalakshimi ve Sundararajan [52], Scrum metodolojisinin temellerini ve şelale modeli ile farklarını incelemiş; Macit ve Tüzün [53], şelale modeli ile scrum modelini proje yönetimi bakışı ile karşılaştırmış; Dorette [54], iki yeni start-up firmasında XP ve şelale modeli ile yazılım geliştirme süreçlerini karşılaştırmıştır.

Yazılım geliştirme metodolojilerinin belirli faktörlere göre karşılaştırıldığı çalışmalar

Bazı önemli çalışmalarda metodolojiler belirli faktörler açısından ele alınmıştır. Farrell [23], çalışmasında şelale modeli ve çevik yöntemlerin de içinde bulunduğu on bir metodolojiyi incelemiş ve organizasyonel karakteristiklere göre metodoloji seçimi ile ilgili önerilerde bulunmuştur. Palmquist ve diğerleri [22], çevik yöntemler ve şelale modelinin benzerlik ve farklılıklarını ele almış, çalışmalarını Amerikan Savunma Bakanlığı projelerinin genel isterleri çerçevesinde şekillendirmişlerdir. Komai ve diğerleri [55], çevik yöntemler ve şelale modelini projede harcanan adam-iş saatine göre; Fair [41] ise çevik yöntemler ve şelale modelini ERP projeleri açısından karşılaştırmıştır.

Bu tezde ise şelale modeli ile bir bütün olarak çevik yöntemler ele alınmış, yazılım proje yönetimi açısından başarıya etki ettiği değerlendirilen aşağıdaki başlıklar altında karşılaştırmalar yapılmıştır:

- Benzerlikler ve Farklılıklar
- Maliyetler
- Dış Kaynak Kullanımı
- Yazılımın Genel Nitelikleri
- Kalite Güvence ve Hata Toleransı
- Değer Üretimi
- Kullanıcı Deneyimi
- Risk Yönetimi
- Görünürlük
- Proje Karmaşıklığı

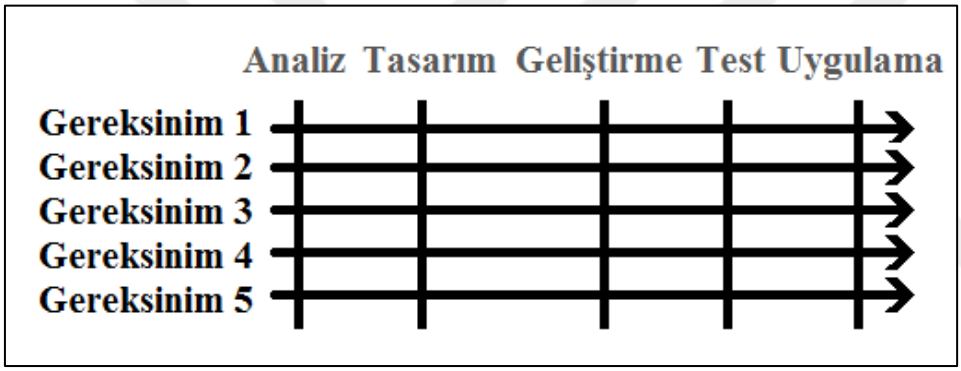
- Proje Büyüklüğü
- Dokümantasyon ve Modelleme
- Yeniden Kullanılabilirlik
- Gereksinimler
- Teslimatlar
- Zaman Planlaması
- Müşteriler
- Kullanım Onayı
- Yönetim
- Organizasyon Yapısı
- Roller
- Tecrübe ve Yetenek
- Takım büyüklüğü
- Alan Bilgisi
- İş birliği
- Yerleşim
- Proje Elemanları Açısından Karmaşıklık
- Ekip Çalışması

Bu çalışmada karşılaştırma yapılan faktörler benzer çalışmalara göre oldukça geniş bir alanda ele alınmıştır. Ayrıca yazılım projeleri başarı ve metodoloji kullanım oranları açısından sektörün önde gelen araştırmalarının en güncel verileri kullanılmıştır.

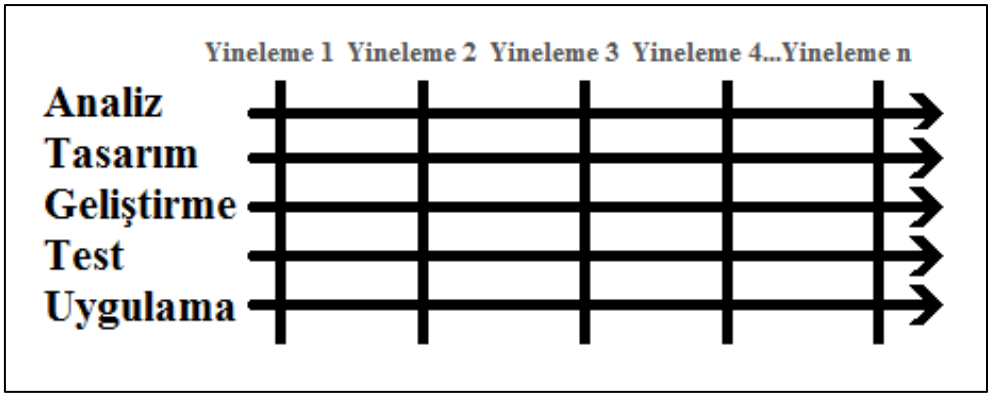
5.2. Benzerlikler ve Farklılıklar

Şelale modeli ile çevik yöntemler aynı amaç doğrultusunda ortaya çıkmış yaklaşımlardır. Bu nedenle hedefleri aynıdır. Her iki metodoloji de öngörülebilir, etkin ve duyarlı bir anlayışla kaliteli ürün ortaya koymayı amaç edinir. Her iki yaklaşımda da planlama önemli bir yer tutar. Şelale modelinde bütün süreci kapsayan uzun vadeli ve detaylı bir planlama yapılırken, çevik yöntemlerde değişen şartlara cevap verebilmek için uzun vadeli planlamadan ziyade kısa vadeli ve daha detaylı planlamalar yapılır. Bütün süreç sürüm planlarına, sürüm planları yineleme planlarına, yineleme planları ise günlük planlara bölünür. Böylece daha

detaylı hedefler belirlenebilir. Her iki yaklaşımda da süreç içinde detaylı incelemelere önem verilir. Şelale modelinde bu incelemeler safhalar arası geçişte yapılan detaylı kontroller ve dokümantasyon vasıtasıyla yapılırken, çevik yöntemlerde yinelemeler sonucu üretilen çalışan ürünler ve müşterinin bütün sürecin içinde bulunması ile detaylı bir inceleme sağlanmış olur. İki metodolojide de yapılan faaliyetler –analiz, tasarım, geliştirme, test, uygulama- ortaktır. Bu faaliyetlerin icra şekilleri farklıdır. Şelale modelinde bu faaliyetler her gereksinime uygulanan safhalar olarak karşımıza çıkar ve ana sürecin alt bileşenleridir. Çevik yöntemlerde ise her yinelemenin uygulandığı faaliyetler olarak karşımıza çıkarlar ve ana sürecin alt bileşeni yinelemeleri oluşturan faaliyetlerdir. Metodolojilere ait temel süreç akışları Şekil 5.1 ve Şekil 5.2'de gösterilmiştir.



Şekil 5.1. Şelale modeli süreç akışı [22]



Şekil 5.2 Çevik yöntemler süreç akışı [22]

Şelale modelinde gereksinimlerden oluşan paket proje sürecinde safhalar üzerinde ilerlerken çevik yöntemlerde safhalardan oluşan paket yinelemeler üzerinde ilerler. Projenin başında tanımlanan gereksinimler ve önceliklere göre teslimat yapılması nedeniyle şelale modelinde devamlı geriye doğru bir bakış vardır. Çevik yöntemlerde ise evrilen gereksinim ve önceliklere göre teslimat yapılması

nedeniyle devamlı ileriye doğru bir bakış vardır. Yapısal süreçlere dayalı olması nedeniyle şelale modeli daha çok bir mühendislik yaklaşımı sunarken kişilere ve iletişime dayalı olması nedeniyle çevik yöntemlerde yaratıcılık ön plandadır. [22, 41, 56]

Çalışmanın bundan sonraki bölümünde; Çizelge 3.2'de belirtilen CHAOS raporları başarı faktörleri, literatür araştırmaları ve yazılım projelerinde edinilen kişisel tecrübeler ışığında proje başarısına etki ettiği değerlendirilen konularda, şelale modeli ve çevik yöntemler için karşılaştırma ve değerlendirmeler sunulacaktır.

5.3. Maliyetler

Şelale modelinde çoğu zaman tercih sebebi olan daha detaylı analiz, dokümantasyon ve testlerin yapılabilmesi için daha çok bütçe, zaman ve işgücü gerekir [57]. Çevik yöntemler analiz, dokümantasyon ve test faaliyetlerinin daha az olması nedeniyle maliyet konusunda şelale modeline göre avantaj sağlar.

Yazılım geliştirme sürecinde yapılan faaliyetlerin aynı olduğu, bu faaliyetlerin sıralamasında ve takibinde farklılıklar olduğu göz önünde bulundurularak, şelale modeli ile çevik yöntemlerin, adam-saat işgücü açısından karşılaştırılması için belirleyici iki faktör mevcuttur. Bunlar şelale modelinde detaylı bir şekilde yapılan analiz ve gereksinim tanımlamaları ile çevik yöntemlerde yapılmış olan işlerde ortaya çıkan değişiklikler yüzünden tekrar yapılan işlerdir. Çevik yöntemlerde gereksinimlerin tanımlanması konusunda tasarruf edilen zaman, değişiklikler yüzünden yeniden yapılan işlere harcanan zamandan fazla olduğu takdirde, adam-saat karşılaştırılması açısından çevik yöntemler daha avantajlıdır. Değişiklik nedeniyle yapılmış olan işlere tekrar harcanan işgücü gereksinim tanımlamasında tasarruf edilen işgücünden fazla ise şelale modeli daha avantajlıdır [55].

5.4. Dış Kaynak Kullanımı

Dış kaynak kullanılacak durumlarda, yapılması istenen ürün veya ürün parçasının detaylı tanımlanması şarttır. Alt yüklenici kendisinden istenen ürüne göre bir maliyet, zaman ve efor tahmini hazırlayarak bir teklif verir ve sözleşme yapılır [46]. Maliyetleri arttıracağı için ihtiyaçlarda doğabilecek değişiklikler kabul edilmez. Bir tedbir alınmadığı müddetçe ihtiyaç sahibi müşteri ve proje takımı ile alt yüklenici

arasında herhangi bir iletişim olmaz. İletişim olmadan müşteri etkileşimi ve proje takımı ile dayanışmadan söz edilemez. Bu nedenlerle çevik yöntemler, altyüklenici kullanımına uygun değildir. Tanımlanmış isterler ve zaman kısıtları nedeniyle şelale modeli alt yüklenici kullanımına daha müsaittir.

5.5. Yazılım Kalitesi

Yazılımın genel nitelikleri, kalite güvence ve hata toleransı, değer üretimi, kullanıcı deneyimi ve risk yönetimi ile ilgili konularda yapılan karşılaştırmalar, yazılım kalitesi başlığı altında ele alınmıştır.

5.5.1. Yazılımın genel nitelikleri

Yazılımdan beklenen nitelikler, yazılımın uygulama alanına göre değişiklik gösterir. Örneğin bir banka sisteminden güvenli olması, iletişim ağlarının devamlılığının olması, interaktif bir oyunun hassas bir tepkisellik sunması beklenir. Ürün nitelikleri aşağıdaki şekilde genellenebilir:

- Sürdürülebilirlik: Yazılım müşterilerin değişen ihtiyaçlarına göre evrilebilecek şekilde üretilmelidir.
- Güvenilirlik ve emniyet: Güvenilir yazılım, bir hata halinde fiziksel veya maddi hasar oluşturmamalıdır. Kötü niyetli kullanıcılar sisteme erişip zarar verememelidir.
- Verimlilik: Yazılım, işlemci ve RAM gibi sistem kaynaklarını boşa harcamamalıdır.
- Kabul edilirlik: Yazılım, hitaben üretildiği kullanıcı tarafından kabul edilmelidir. Anlaşılır, kullanılabilir ve diğer sistemlerle uyumlu olmalıdır. [14]

Bu niteliklere metodolojiler açısından bakılacak olursa, sürdürülebilirlik ve kabul edilirlik nitelikleri açısından çevik yöntemler şelale modeline göre daha uygundur. Şelale modeli değişimi kabul etmezken çevik yöntemlerde değişim hoş karşılanır ve teşvik edilir. Çevik yöntemlerde yinelemeler sonunda yapılan ürün teslimleri ve alınan geri beslemeler ile müşterinin proje sürecinin tamamına katılımı, teslim safhasında kabul edilirlik açısından şelale modeline göre avantaj sağlar. Güvenilirlik ve emniyet ile verimlilik nitelikleri açısından ise şelale modeli ön plana çıkmaktadır. Şelale modelinde uygulanan katı süreçler ile test yöntemleri, çevik

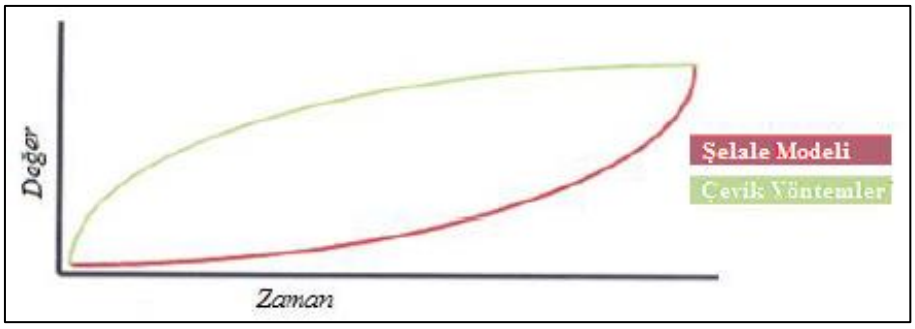
yöntemlerin test ve kalite yöntemlerine göre daha güvenilirdir. Ayrıca ürünün sık ve çabuk teslim edilmesi ve müşteri gereksinimlerinin işlevselliğine odaklanılması, çevik yöntemlerde güvenlik, verimlilik gibi konuların göz ardı edilmesine sebep olabilir.

5.5.2. Kalite güvence ve hata toleransı

Çevik yöntemlerde eşli programlama, önce test yaklaşımı, gözden geçirme toplantıları, ürün değerlendirmeleri gibi tekniklerle sürekli inceleme ve ürünün geliştirme boyunca iyileştirilmesi sağlanmaya çalışılsa da, bu önlemlerin hata toleransı olmayan kritik sistemlerde istenen kalite için yeterli olmayacağı değerlendirilmektedir [46]. Bu sistemlerde arzu edilen kalitenin sağlanabilmesi için icra edilen analiz, geliştirme ve test faaliyetleri daha detaylı, resmi ve katı uygulamalar içermelidir [58]. Çevik yöntemlerin başlıca avantajları olan erken değer üretimi, sürekli geri besleme ve değişen şartlara adapte olabilme yeteneği, ilk ve en büyük önceliği hataya karşı sıfır tolerans olan bu sistemler için bir önem ifade etmez. Bu nedenle kalite güvence anlayışı açısından şelale modeli, çevik yöntemlere göre daha uygun bir yöntemdir.

5.5.3. Değer üretimi

Gerek şelale modeli gerekse çevik yöntemler yazılımın üreteceği değere önem verir. Kazanılmış değer analizi, yazılım projelerinin önemli bir yönetim alanıdır. Çevik yöntemlerde müşterinin istediği yazılım değeri proje süreci boyunca öğrenilir ve üretilir. Şelale modelinde ise proje başlangıcında anlaşılması ve proje sonunda üretilmesi öngörülür [59]. Şelale modelinde proje sonunda ürün teslimi yapıldığı için değer üretimi proje sonunda gerçekleşir. Çevik yöntemlerde ise yinelemeler sonunda teslim edilen her çalışan ürün müşteri için bir fayda sağlar. Yazılımdan beklenen değerin iyi anlaşılması için, müşterinin isteklerini iyi ifade edebilmesi gerekir. Şelale modelinden çevik yöntemlere geçen bir şirketin bir proje yöneticisi, gereksinimlerin son kullanıcı tarafından değil de kullanıcı olmayan bir müdür tarafından tanımlanması nedeniyle yaptıkları işlerin % 80'inin kullanıcı için bir değer üretmediğini gördüklerini belirtmiştir [56]. Şelale modeli ve çevik yöntemlerin değer üretimi grafiği Şekil 5.3'te sunulmuştur.



Şekil 5.3. Değer üretimi karşılaştırması [53]

5.5.4. Kullanıcı deneyimi

Şelale modelinde kullanıcı deneyimi ve etkileşimi için detaylı bir tasarım ve geniş bir zaman ayrılır. Çevik yöntemlerde ise uzun vadeli plan ve detaylı tasarım yapılmaz ve kısa zamanlı yinelemeler, bütünsel bir yaklaşım gerektiren kullanıcı deneyimi ve etkileşimi için yapılacak çalışmalar açısından uygun değildir [60]. Buna rağmen teslim edilen ürünler neticesinde alınan geri beslemelerle kullanıcı deneyiminin iyileştirilmesi çevik yöntemlerde mümkün olmakla birlikte şelale modelinde böyle bir imkân yoktur.

5.5.5. Risk yönetimi

Risk yönetimi, risk faktörlerinin tanımlanması ve her risk faktörünün gerçekleşme ihtimali ile potansiyel etkisinin analizi, risk faktörlerinin önceliklendirilmesi, gerçekleşme ihtimallerin düşürülebilmesi için risk azaltma stratejilerinin belirlenmesi ve probleme dönüşen risk faktörlerinin olumsuz etkilerinin en aza indirgenmesi faaliyetlerini kapsar [15]. Çizelge 5.2'de genel olarak risk türleri ve kısaca açıklamaları verilmiştir.

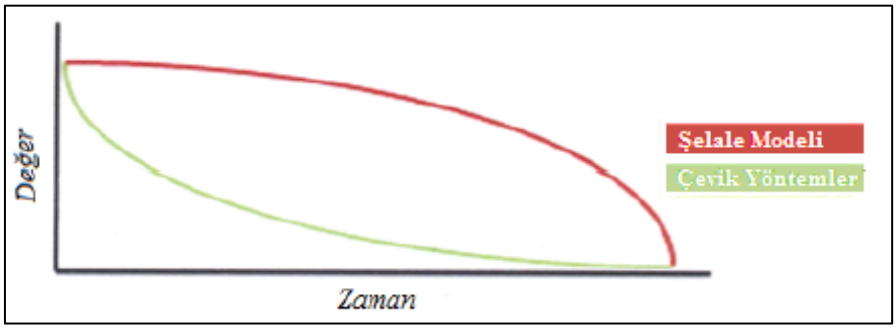
Söz konusu risk türleri, şelale modeli ve çevik yöntemler açısından değerlendirilecek olursa; zaman riskleri ve bütçe riskleri açısından şelale modelinde proje başlangıcında yapılan planlamaların detaylı ve gerçekçi bir şekilde yapılması önem arz etmektedir. Çevik yöntemlerde ise yinelemeli geliştirme anlayışı ve her yineleme sonunda yapılan teslimatlar bu risklerin izlenmesi ve yönetilmesini kolaylaştırmaktadır. Bunun yanında çevik yöntemlerin değişime açık yapısı, zaman ve bütçe risklerinin yönetimi açısından dezavantaj oluşturmaktadır. Şelale modeli ile çevik yöntemlerin yönetim anlayışındaki farklar

nedeniyle yönetim risklerinin ele alınması da farklı olmalıdır. Şelale modelinde kontrol, otorite ve yönetim konuları, çevik yöntemlerde ise iletişim, organizasyon ve iş birliği konuları üzerinde durulmalıdır. Teknik riskler ve program risklerinin yönetimi açısından çevik yöntemlerin yineleme ve teslimatları izlenebilirliği kolaylaştırıp avantaj sağlarken, şelale modelinde kapsam konusunda değişikliğe kapalı olunması olası problemleri indirgemektedir. Çevik yöntemlerin personel bilgi ve tecrübesine dayalı yapısı nedeniyle personel risklerinin yönetimi konusu, şelale modeline göre daha fazla önem arz etmektedir.

Çizelge 5.2. Risk Türleri ve Açıklamaları [74]

Risk Türü	**Açıklama**
Zaman Riskleri	Projenin, yanlış görev ve malzeme paylaşımından dolayı beklenen süre içerisinde gerçekleşmesine veya tamamlanmasına engel olan risk çeşididir.
Bütçe (Maliyet) Riskleri	Gerçekçi olmayan bütçe tahminleri sonucu finansal sorunlara yol açan risklerdir. Bu risklerin gerçekleşmesi durumunda Tablolar değişmekte, maliyetler artmaktadır.
Yönetim Riskleri	Amaçların net olmayışı, planlama eksikliği, yönetim tecrübesi ve eğitim eksikliği, iletişim sorunları, örgütsel sorunlar, otorite eksikliği ve kontrol problemlerini kapsamaktadır.
Teknik Riskler	Genelde fonksiyonların yanlış olmasından kaynaklanır. Müşteri taleplerinin sürekli değişmesi, gelişmiş tekniklerin kullanılmaması ve geliştirilecek olan projenin zor faaliyetler içermesi gibi sebeplerden kaynaklanmaktadır.
Program Riskleri	Proje kapsamının dışına çıkan, kontrol dışı durumlardan veya önceliklerin sürekli değişmesinden doğan risklerdir.
Sözleşme ve Yasal Riskler	Değişen ihtiyaçları, pazar odaklı programları, sağlık ve güvenlik sorunlarını, hükümet düzenlemeleri ve ürün garantisi konularını içerir.
Personel Riskleri	Personel duraklamaları, deneyim ve eğitim sorunları, etik ve ahlak konularını, personel çatışmalarını ve verimlilik sorunlarını içermektedir.
Diğer Kaynaklı Riskler	Mevcut olmayan veya geç teslim edilen donanım ve sarf malzemeleri, yetersiz tesisleri, dağıtılan bölgeleri, bilgisayar kaynaklarının olmayışı ve yavaş tepki sürelerini kapsamaktadır.

Şelale modeli ve çevik yöntemlerin zaman içinde taşıdıkları risk grafiği Şekil 5.4'te sunulmuştur.



Şekil 5.4. Risk karşılaştırması [53]

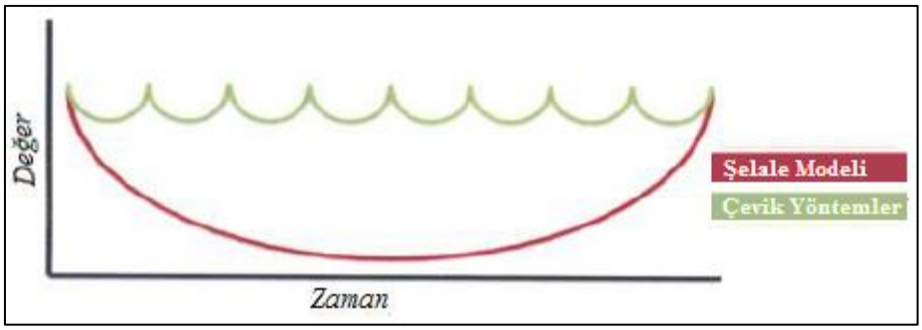
Şelale modeli, proje başlangıcından itibaren hem üretim hem de kabul risklerini barındırmaktadır. Süreç içerisinde uygulanan önlemler ile risk bir miktar azalsa da kabul aşamasına kadar ana hatlarıyla devam etmektedir. Çevik yöntemlerde ise tamamlanan her yinelemede üretim riskleri, her teslimde ise müşteri katılımı sağlandığı için [24] kabul riskleri azalmaktadır. [53]

5.6. Analiz Tasarım Kalitesi

Görünürlük, karmaşıklık, proje büyüklüğü, dokümantasyon ve modelleme, yeniden kullanılabilirlik ve gereksinimler ile ilgili konularda yapılan karşılaştırmalar, analiz tasarım kalitesi başlığı altında ele alınmıştır.

5.6.1. Görünürlük

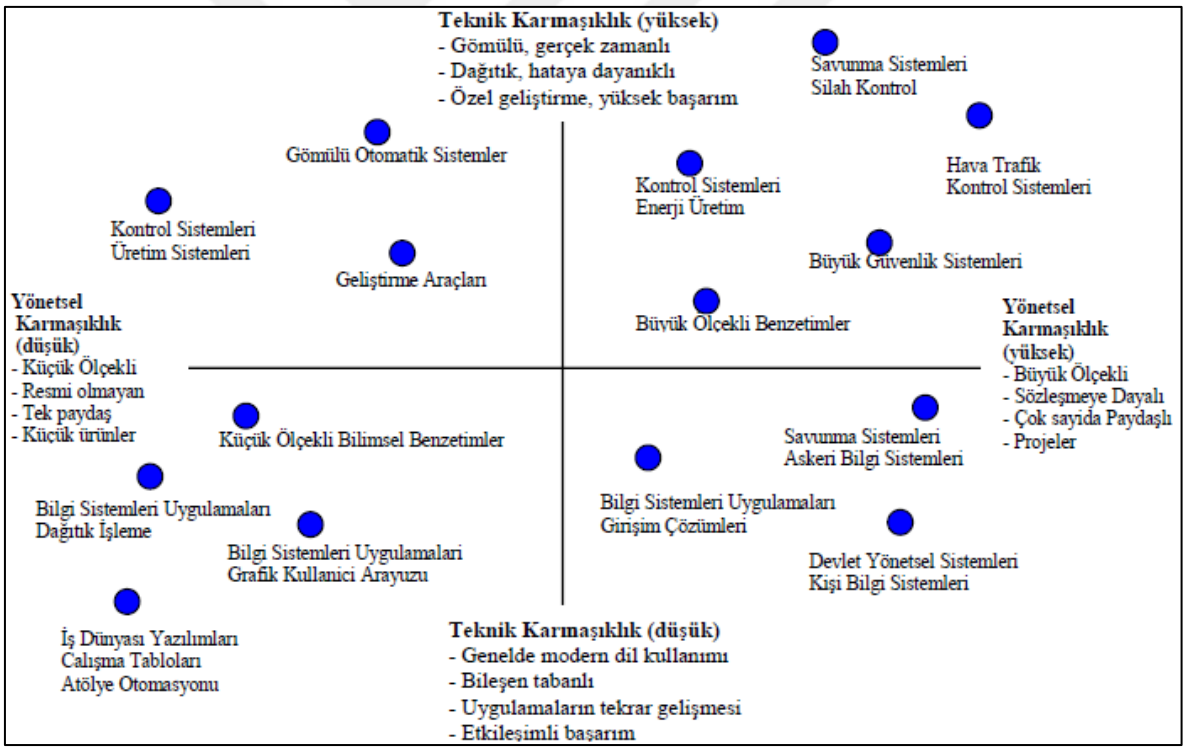
Şelale modelinde proje görünürlüğü uygulanan safhalar arasında yapılan kontroller ve dokümantasyonla sağlanırken, çevik yöntemlerde yinelemeler sonunda ortaya çıkarılan çalışır ürünler ve ürün iş listeleri üzerinden sağlanır. Çevik yöntemler bu kapsamda müşteriden doğrudan çalışan ürünün onayını alarak proje ilerlemesini gerçekleştirdiği için şelale modeline göre avantaj sağlar. Şelale modeli ve çevik yöntemlerin görünürlük grafiği Şekil 5.5'te sunulmuştur. Çevik yöntemlerin görünürlük olarak avantajlı olduğu projeler, çalışan ürünün kullanıcı ara yüzlü ürün olduğu projelerdir. Kullanıcı tarafından değerlendirilemeyen ya da fonksiyonel olarak küçük parçalara ayrılamayan projelerde dokümantasyon daha iyi bir görünürlük sağlayacağı için şelale modeli daha uygundur [46].



Şekil 5.5. Görünürlük karşılaştırması [53]

5.6.2. Karmaşıklık

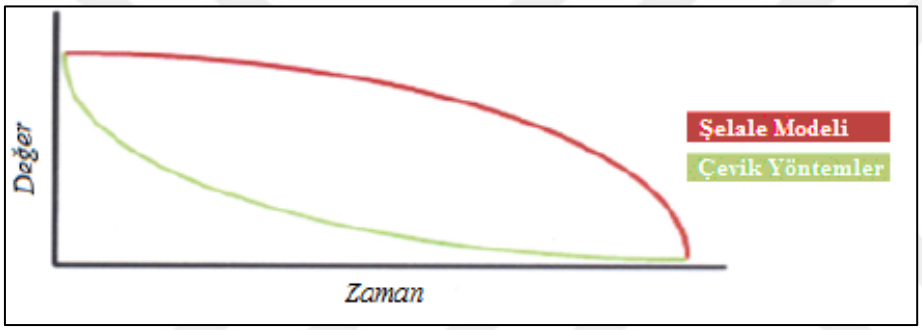
Günümüzde kullanılan yazılım sistemlerinin karmaşıklık dereceleri teknik ve yönetsel olarak gruplandırıldığında, hangi alanda ne kadar karmaşıklık derecesi ile karşılaşılabileceği Şekil 5.6'da görülmektedir [61].



Şekil 5.6. Karmaşıklık seviyesine göre yazılım sistemleri [62]

Üretilecek yazılımın karmaşıklık seviyesi arttıkça ihtiyaç duyulan analiz ve yönetim disiplini artar. Örneğin teknik ve yönetsel karmaşıklık açısından zirvede bulunan, karmaşık ve hassas zamanlama gereksinimi duyan gerçek zamanlı savunma sistemleri [61] için kodlama aşamasına kadar detaylı analizler yapılmış olmalıdır. Bu şartlarda şelale modeli, plan tabanlı bir yaklaşım olduğu için en iyi seçenektir.

Çevik yöntemler en erken zamanda müşteri için değer üretmeyi amaçlar ve eldeki mevcut girdilerle kodlamaya geçilerek yinelemeler içerisinde durumun olgunlaşması amaçlanır. Bu nedenle çevik yöntemler daha az analiz gerektiren nispeten daha az karmaşık sistemler için daha uygundur. Detaylı analiz ve planlama şelale modelinin çevik yöntemlere göre avantajı iken, yineleme sonlarında yapılan çalışır ürün teslimleri vasıtasıyla yapılabilen gözden geçirme, alınan geri besleme ve değişim yapabilme imkânı, proje sürecinde karmaşıklık ile başa çıkabilme açısından çevik yöntemlerin şelale modeline göre avantajıdır. Şelale modeli ve çevik yöntemlerin proje sürecinde karmaşıklık grafiği Şekil 5.7'de sunulmuştur.



Şekil 5.7. Proje sürecinde karmaşıklık karşılaştırması [53]

5.6.3 Proje Büyüklüğü

Proje büyüklüğü ile ilgili genel kanı, büyük projelerin detaylı dokümantasyon ile daha iyi yönetilebileceği, küçük projelerin ise katılımcılar arasındaki doğrudan iletişim sayesinde daha etkin sürdürülebileceği, bu nedenle büyük projelerin şelale, küçük projelerin ise çevik yöntemler için daha uygun olduğu yönündedir [58]. Proje boyutunun temel unsurları bütçe, süre ve proje takımının teşkilatlanmasıdır. Proje büyüdükçe çalışan sayısı ve bütçe artar. Bu da daha çok koordinasyon gerektirir. Şelale modeli, kapsamlı planlama, dokümantasyon ve büyük takımlar için daha iyi iletişim ve koordinasyon süreçleri sunarak bu ortamı destekler [27].

Proje büyüdükçe çevik yöntemlerin maruz kaldığı dezavantajlara karşı tedbirler alınmakta ve büyük ölçekli projelerde kullanılmak üzere özelleşmiş geniş kapsamlı çevik metodolojiler ortaya çıkmaktadır. Geniş kapsamlı çevik yöntemlerin en önde gelenleri Disciplined Agile Delivery (DAD), Large-Scale Scrum (LeSS), ve Scaled Agile Framework (SAFe) metodolojileridir [73].

2011-2015 yılları arasında yazılım projelerinde elde edilen başarı durumlarının proje büyüklüğüne göre dağılımı Çizelge 5.3'te, proje büyüklüğü ve uygulanan metodolojiye göre başarı durumları Çizelge 5.4'te sunulmuştur.

Çizelge 5.3. Yazılım projelerinin başarı oranlarının proje büyüklüğüne göre dağılımı [18]

Proje Büyüklüğü	**Başarılı**	**Sorunlu**	**Başarısız**
Büyük	% 8	% 25	% 41
Orta	% 30	% 59	% 48
Küçük	% 62	% 16	% 11
Toplam	% 100	% 100	% 100

Çizelge 5.4. Proje büyüklüğüne ve uygulanan metodolojiye göre yazılım projeleri başarı oranları [18]

Proje Büyüklüğü	**Metodoloji**	**Başarılı**	**Sorunlu**	**Başarısız**	**Toplam**
Büyük	Çevik	% 18	% 59	% 23	% 100
	Şelale	% 3	% 55	% 42	% 100
Orta	Çevik	% 27	% 62	% 11	% 100
	Şelale	% 7	% 68	% 25	% 100
Küçük	Çevik	% 58	% 38	% 4	% 100
	Şelale	% 44	% 45	% 11	% 100

Çizelge 5.3'te görüleceği üzere başarılı yazılım projelerinin %92 gibi çok büyük bir oranını küçük ve orta ölçekli projeler oluştururken sorunlu projelerin %84, başarısız projelerin ise %89 unu orta ve büyük ölçekli projeler oluşturmaktadır. Proje boyutu büyüdükçe başarı şansı büyük oranda azalmaktadır. Çevik yöntemlerin uygulandığı projelerdeki başarı oranının, şelale modelinin uygulandığı projelerdeki başarı oranından daha yüksek olduğu Çizelge 5.1'de görülmekteydi. Çevik yöntemler her ne kadar küçük, aynı yerde konuşlanmış iletişimi iyi takımlarda daha etkin olsa da karşılaştırmanın içine proje boyutu girdiğinde de

sonucun değişmediği Çizelge 5.4'te görülmektedir. Genel kanının [58] aksine elde edilen sonuçlar çevik yöntemlerin her türlü proje büyüklüğünde daha başarılı olduğunu göstermektedir.

5.6.4. Dokümantasyon ve modelleme

Üretilen yazılımın kullanılması öngörülen sürenin uzun ve kullanıcıların değişken olduğu durumlarda dokümantasyon bir başvuru kaynağı olarak önem arz edecektir. Geliştirme ekibinin dağınık olduğu veya bir altyüklenici kullanılan durumlarda da dokümantasyon önemli bir iletişim kanalı olarak kullanılabilir [14]. Dokümantasyon ve modelleme, müşteri için bir değer üretmediği durumlarda bile geliştiriciler için faydalı olabilir. Evrilen büyük sistemlerin geliştirilmesinde başvurulacak mevcut modelleme yoksa, geliştiriciler sistemi anlamak ve yapılacak değişimin etkisini belirleyebilmek için çalışan kodu analiz etmek zorunda kalırlar [46]. Dokümantasyon açısından uyguladıkları felsefe nedeniyle böyle durumlarda çevik yöntemlerden ziyade şelale modeli ile yazılım projesinin yönetilmesi daha faydalı olacaktır.

5.6.5. Yeniden kullanılabilirlik

Çevik yöntemlerde basitlik prensibiyle hareket edildiği için, temel ihtiyacı karşılamak için çaba harcanır. Ürünün hiç gerçekleşmeyebilecek durumlar için genelleştirilmesine ve kapsamın genişletilmesine izin verilmez. Yeniden kullanılabilir bir ürün için bu şartların tanımlandığı dokümantasyon da gerekecektir ve bu da iş yükünü gereksiz yere arttıracaktır [46]. Bu nedenle üretilen ürünler daha spesifiktir. Dokümantasyon ve modelleme ihtiyaç duyulmadığı sürece detaylı olarak üretilmediği için ürünlerin yeniden kullanılabilirliği çok zayıftır. Genel maksatlı, kullanımı genişletilebilecek ya da farklı projelerde yararlanılabilecek yazılım için, dokümantasyon ve modelleme imkânı nedeniyle şelale modeli daha uygundur.

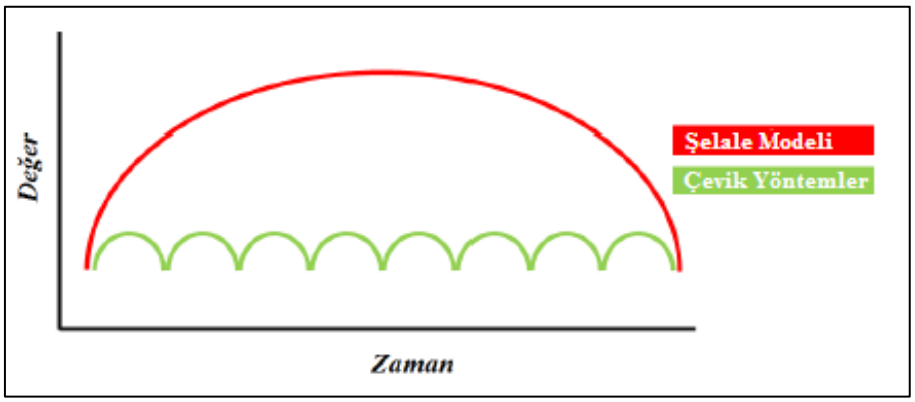
5.6.6. Gereksinimler

Belirsizlik

Şelale modeli, plan tabanlı ve önceden tanımlanan süreçler olduğu için belirsizlik toleransı yoktur. Çevik yöntemler ise planların değişeceği varsayımıyla önceden detaylı plan yapılmaması ve planlamadan ziyade ürün ortaya koyularak zamanın harcanması felsefesini benimsediği için belirsiz ortamlar açısından şelale modeline göre daha uygundur [58].

İhtiyaca yabancılaşma

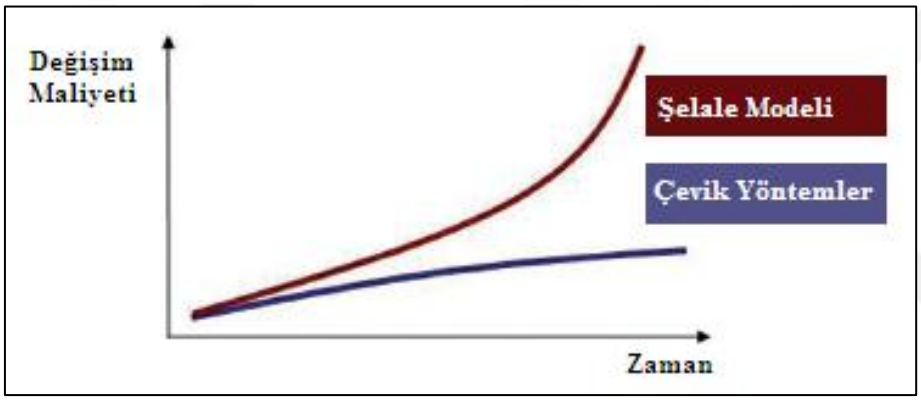
Bir projedeki kullanıcı tarafından belirtilen ihtiyaçlar sırasıyla gereksinimlere, tasarım modeline, yazılım koduna ve son olarak ihtiyacı karşıladığı değerlendirilen ürüne dönüşmektedir. Şelale modelinde projenin başındaki gereksinim analizi asıl ihtiyaç ile etkileşim halinde iken müteakip aşamalarda bir önceki aşamanın çıktısı ile işlem yapılır. Projenin tamamlanma süreci göz önüne alındığında gereksinim analizinden kabul dönemine kadar ihtiyaca yabancılaşma söz konusu olur. Çevik yöntemlerde ise her yineleme başlangıcında gereksinimler gözden geçirilir. Müşteri ise her yineleme sonucu teslim edilen ürünün ihtiyacını karşılayıp karşılamadığını görür. Bu nedenle ihtiyaca yabancılaşma etkisi bir yineleme süresi ve boyutu ile sınırlı kalmaktadır. [53] Şelale modeli ve çevik yöntemlerin ihtiyaca yabancılaşma grafiği Şekil 5.8'de sunulmuştur.



Şekil 5.8. İhtiyaca yabancılaşma karşılaştırması [53]

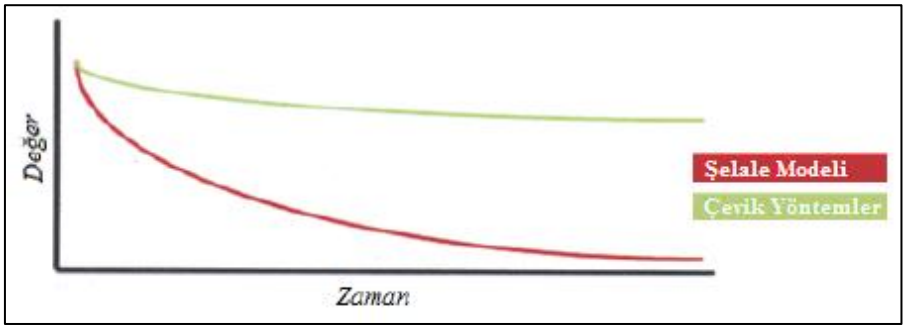
Değişikliğe açıklık

Çevik yöntemlerin ortaya çıkmasının en büyük sebeplerinden ve çevik yöntemlerin en büyük avantajlarından biri gereksinimlerdeki değişikliklerin proje boyunca kabul edilmesidir. Şelale modelinde gereksinim analizi safhası tamamlandıktan sonra projenin ilerleyen safhalarında gereksinimlerde yapılacak değişiklikler hem maliyeti arttıracağı hem de önceki safhalara dönülmesini gerektireceği için kabul edilmez. Çevik yöntemlerde değişim maliyetleri şelale modeline göre nispeten daha düşüktür. Şelale modeli ve çevik yöntemlerin değişim maliyeti karşılaştırması Şekil 5.9'da gösterilmiştir.



Şekil 5.9. Değişiklik maliyeti karşılaştırması [63]

Bu kapsamda, uzun şartname hazırlıkları, bütçe-ödenek ve ihale süreçleri ile proje süreleri göz önüne alındığında bazen şelale modeli ile üretilen yazılımlar yıllar önceki ihtiyaçlara cevap vermek durumunda kalmaktadır [22]. Şelale modeli ve çevik yöntemlerin değişikliğe açıklık grafiği Şekil 5.10'da sunulmuştur.



Şekil 5.10. Değişikliğe açıklık karşılaştırması [53]

Anlaşılırlık, uygulanabilirlik, güvenilirlik

Geliştirme faaliyetine başlanmadan önce detaylı bir gereksinim tanımlama ve tasarım ihtiyacı var ise ayrılan zaman ve süreçler nedeniyle şelale modeli daha uygun bir yaklaşımdır [14]. Fakat bu safhaların başarıyla aşılabilmesi için proje başlangıcında bütün ihtiyaçların biliniyor olması ve bu ihtiyaçların müşteri tarafından doğru ve uygulanabilir bir şekilde ifade edilebilmesi gerekir. Şelale modelinde, gereksinimler tam anlaşılmadığında müşteriyi temsilen kimse bulunmadığı için geliştiriciler iş süreçleri ve gereksinimler konusunda varsayımlarda bulunmak zorunda kalırlar [57]. Geri dönüşler de kabul edilmediği için bu konulardaki aksaklıkların yaratacağı sıkıntılar proje boyunca hissedilecektir. Gereksinimlerin anlaşılırlığı, uygulanabilirliği ve güvenilirliği ile ilgili bir belirsizlik veya tereddüt varsa esnek yapısı nedeniyle çevik yöntemler daha uygun bir yaklaşım sunabilir.

Gereksinim yönetimi

Şelale modelinde detaylı hazırlanan gereksinimlerin tasarım ile birleşerek kodlanacak görevlerin oluşturulması sonucu, gerek proje yöneticileri gerekse dokümantasyon vasıtasıyla iyi bir gereksinim yönetimi sağlanmış olur. Çevik yöntemlerde ise gereksinimlerin müşteriler ile birlikte belirlenip önceliklendirilmesi ve geliştirme takımının bu sürece dahil olması küçük projelerde bir sorun yaratmasa da gereksinimlerin birden fazla grup tarafından tanımlandığı ve geliştirme takımlarının hepsiyle irtibatının olmadığı büyük projelerde gereksinim yönetimi eksikliği yaşanabilir [60].

5.7. Teslimatlar

Jenerik yazılım üreterek pazara sunan firmalar, sürümden belli bir süre önce ürünün pazarlaması için çalışmalara başlarlar. Şelale modelinde proje sonunda tek bir teslimat yapılması, bu felsefeye uygundur fakat çevik yöntemlerdeki kısa süreli yinelemeler sonucu verilen ürün versiyonları ve gereksinim önceliklerinin belirlenmesindeki esnek yaklaşım, pazarlama departmanlarına hangi özellikteki ürünü ne zaman pazarlayacakları konusunda net bir görüş sağlayamaz [60]. Bu nedenle jenerik yazılım üreten firmalar için uzun vadeli planlamaları gereği şelale

modeli daha uygun bir yaklaşımdır. Bunun yanında, pazar şartlarının ve iş süreçlerinin hızlı değiştiği, değişikliklere çabuk reaksiyon gösterilmesi gereken ortamlar ve sürekli değer üretimi beklenen projeler için, yinelemeler sonunda ürün tesliminin yapılması, çevik yöntemleri avantajlı kılar. Teslim edilen ürünlerden alınacak somut geri beslemeler sayesinde hatalar ayıklanır, düzeltmeler yapılır ve proje süreci daha sağlıklı ilerler.

5.8. Zaman Planlaması

Şelale modelinde safha bitimi için bütün işlerin bitmesi gerekir ve henüz bitirilmemiş işler yüzünden safha ilerlemediğinde işini bitiren personel zaman kaybeder. Proje elemanlarının hep beraber takım olarak çalışması kaynakların daha etkin kullanımını sağlar [64]. Bu nedenle şelale modelinde zaman israfı yaşanması mümkün iken çevik yöntemlerde kısa süreli yinelemeler sayesinde zaman en etkin şekilde kullanılır.

5.9. Dış Etkenler

Müşteriler ve yazılımın kullanım onayı ile ilgili konularda yapılan karşılaştırmalar, dış etkenler başlığı altında ele alınmıştır.

5.9.1. Müşteriler

Müşteri katılımının istenen seviyede sağlanabildiği durumlarda çevik yöntemler proje sonucunda elde edilecek yazılımın nitelikleri açısından şelale modeline göre daha avantajlıdır. Gereksinimlerin ve iş süreçlerinin sağlıklı bir şekilde tanımlanması, ara teslimatlar ve geri beslemeler sayesinde elde edilen ürünün tam olarak istenen ürün olması sağlanabilir. Bunun yanında proje sürecine müşteri katılımının sağlanabilmesi için müşteri açısından istek, zaman ve bilgi gerekir. Proje elemanlarıyla birlikte proje süresince çalışacak eleman bulundurmak müşteri için külfetli olabilir ve istenmeyebilir. Geleneksel proje yönetimine alışkın müşteriler proje başında ihtiyaçlarını tanımlayıp bir daha zaman ayırmak istemeyebilir [57] çünkü müşterilerin kendi günlük işleri her zaman daha önemlidir [75]. Aynı yerde ve zaman diliminde olmayan müşteriler ile yüz yüze iletişim kurulamaması faaliyetleri aksatabilir. Müşteri adına projeye katılan personel, iş süreçlerine hâkim olmayabilir. Böyle durumlarda gereksinimlerin sağlıklı tanımlanamaması ve teslim

edilen ürünlerden geri besleme alınamaması muhtemeldir. Bu gibi nedenlerle müşteri katılımının sağlıklı olarak sağlanamayacağı ortamlar açısından şelale modeli daha uygundur çünkü çevik yöntemlerin en önemli dayanaklarından biri müşteri katılımı iken şelale modelinde proje başlangıcından sonra teslime kadar bir müşteri katılımı beklenmez.

5.9.2. Kullanım onayı

Üretilen yazılımın, kullanılacağı alan gereği bir otorite ya da düzenleyici kuruluş tarafından onaylanması gerekiyorsa (örneğin havacılık düzenlemeleri, adli düzenlemeler,) bu onaylama sırasında incelenecek dokümantasyonun detaylı olması gerekecektir ve söz konusu onay son ürüne verileceği için ara ürün teslimatı yapılmasının hiçbir önemi yoktur [14]. Bu tarz yazılımlar için şelale modeli daha uygun bir yaklaşımdır.

5.10. İç etkenler

Projenin genel olarak yönetimi ve projenin yürütüldüğü organizasyon yapısı ile ilgili konularda yapılan karşılaştırmalar, iç etkenler başlığı altında ele alınmıştır.

5.10.1. Yönetim

Merkezi ve hiyerarşik komuta-kontrol yapılı yönetim sistemleri, şelale modeli ile uyumludur. Liderlik, dayanışma ve çalışanların işlerini kolaylaştırıcı anlayışta yönetim tarzı ise çevik yöntemler için uygun ortamlardır [58]. Çevik yöntemlerde takımı motive eden, engelleri kaldıran, süreçleri kolaylaştıran bir koçluk ortaya koyulurken, şelale modelinde klasik komuta-kontrol yönetim tarzı uygulanır [65]. Kendi kendini organize bağlamında proje yönetimi faaliyetleri liderler, müşteriler ve takım elemanları arasında paylaşılır [66].

5.10.2. Organizasyon yapısı

Organizasyonlar için, derecesine göre sınıflandırma yapılan en temel karakteristikler, organizasyondaki bürokrasi, organizasyon yapısı, üretilen ürünün karmaşıklığı ve gerektirdiği çaba, icra edilen görevler ve organizasyondaki değişim ortamıdır [23]. Organizasyon teorisyenleri, bu ve bunun gibi karakteristikleri göz önüne alarak organizasyon çeşitlerini tanımlamışlardır. Henry Mintzberg

oranizasyon çeşitlerini, Girişimci Organizasyon, Makine Organizasyon, Profesyonel Organizasyon, Farklılaşmış Organizasyon, Yenilikçi Organizasyon, Görev Organizasyonu ve Politik Organizasyon olarak yedi ana grup altında toplamıştır [67]. Yazılım geliştirme organizasyonları genel olarak girişimci, farklılaşmış ve yenilikçi organizasyonlardır fakat makine ve profesyonel organizasyonlarda da yazılım projeleri yürütülebilir [23]. Görev organizasyonu ve politik organizasyonlar yazılım proje yönetimi açısından karşılaşılacak organizasyon tipleri değildir Bu nedenle ilk beş organizasyon ele alınacaktır.

Girişimci organizasyon

Bu tip organizasyonlar esnektir ve düşük hiyerarşiye sahiptir. Basit ve dinamik bir ortamları vardır. Genellikle kuruluş aşamasındadırlar. Çoğu girişimci organizasyon yüksek derecede tepkiseldir ve görev anlayışları yüksek seviyededir. Çoğu kez organizasyonun başı çalışanlarla birebir iletişim halindedir. En büyük olumsuzlukları dengesiz ve sağlamlaşmamış iş ortamlarıdır [23, 68]. İletişim ortamının müsait ve yapılarının esnek olması nedeniyle ve iş süreçlerinin değişen şartlara uyum sağlaması gerektiğinden bu tip organizasyonlarda çevik yöntemlerin yinelemeli ve değişime açık yaklaşımlarının uygulanması şelale modeline göre daha uygundur fakat genellikle yeni oluşumlar oldukları için yetenekli ve tecrübeli çalışanları istihdam etme konusunda sıkıntı yaşayabilirler.

Makine organizasyon

Bu tip organizasyonlar fabrikalar gibi hiyerarşik ve olgunlaşmış yapılardır ve değişime direnirler. Durağan ve stabil bir ortamları vardır. Merkezi bir bürokrasi ve formel süreçler işler. Çalışanlar fonksiyonel gruplara ayrılmıştır ve genelde hep aynı işi yaparlar. En büyük olumsuzlukları kontrol saplantısı nedeniyle çalışanlarla ilgili problemlere açık olmaları ve değişen şartlara uyum sağlayamamalarıdır [23, 68]. Bu tip organizasyonlar her yönüyle şelale modeli için biçilmiş kaftandır ve çevik yöntemlerin uygulanması neredeyse imkansızdır.

Profesyonel organizasyon

Bu tip organizasyonlar bürokratik fakat merkezi olmayan, hastane, üniversite benzeri organizasyonlardır. Otonom olarak çalışan profesyonel personel

gruplarından oluşurlar. Genel stratejileri stabildir fakat gerekli spesifik detaylar sık sık değişebilir. Çalışanlar işlerini iyi şekilde bilirler ve yönetim kademesi iş detaylarına karışmaz. Profesyonel organizasyonlarda koordinasyon konusunda problemler yaşanabilir [23, 68]. Şelale modelinin hiyerarşik komuta-kontrol yapısı bu tip organizasyonlar için uygun değildir. Çalışanların profesyonel olması ve organizasyonun kişisel otonomiye önem vermesinden dolayı çevik yöntemler bu tip organizasyonlar için daha uygun yaklaşımlardır.

Farklılaşmış organizasyon

Farklılaşmış organizasyonlar, bünyelerinde yapılan işin farklılığına göre oluşturulmuş şirket benzeri birimler bulunduran, bu birimlerin insan kaynakları, finans, hukuk gibi ortak fonksiyonlarını da kendi çatısı altında tutan holding benzeri yapılardır [68]. Bu tip organizasyonlar için hiyerarşik yapı ve uzmanlık alanlarındaki farklılıklar nedeniyle şelale modeli daha uygundur. Her birim kendisiyle ilgili faaliyeti icra edecek ve biten işi diğer birime iletecektir. Çevik yöntemlerde gerekli iletişim ve koordinasyon ortamının bu tip dağıtık organizasyonlarda sağlanması zordur. Uzmanlık alanlarına göre personel ayrılarak matriks yapı benzeri proje grupları oluşturulması halinde çevik yöntemler uygulanabilir.

Yenilikçi organizasyon

Bu tip organizasyonlar merkezi olmayan ve hiyerarşi seviyesi düşük organizasyonlardır. Fonksiyonel uzmanlardan oluşur ve karşılıklı koordinasyona dayanır. Kimin ne görev yapacağı çok net belirlenmemiştir. Bir masanın etrafında bir problemi çözmek için toplanmış uzmanlar topluluğuna benzetilebilir [23, 68]. Tanımından da anlaşılacağı üzere uzmanlık, koordinasyon ve otonom yapısı nedeniyle çevik yöntemlerin felsefesi ile birebir örtüşen bir organizasyon tipidir. Komuta kontrol yönetim sistemine uymaması nedeniyle şelale modeli için uygun değildir.

5.11. Takım

Yazılım proje elemanlarının rolleri, tecrübe ve yetenekleri, takımın büyüklüğü, proje elemanlarının alan bilgisi, iş birliği, proje elemanlarının yerleşimi, projenin

proje elemanları açısından karmaşıklığı ve ekip çalışması ile ilgili konularda yapılan karşılaştırmalar, takım başlığı altında ele alınmıştır.

5.11.1. Roller

Yazılım projelerinde şelale modeli ile çevik yöntemlerdeki roller başta yönetici rolleri olmak üzere farklıdır. Şelale modelinde proje elemanları, temel yazılım geliştirme ve mühendislik faaliyetlerini geliştirici ve test mühendisi gibi belirlenmiş uzmanlık alan adları altında gerçekleştirirler. Gereksinimlerin toplanması, netleştirilmesi, efor hesaplamaları gibi yönetim faaliyetleri ise proje yöneticileri veya iş analistleri tarafından yapılır. İşler, proje yöneticileri tarafından proje elemanlarına paylaştırılır. Çevik yöntemlerde ise yönetim faaliyetleri lider rollerdeki personelle birlikte kendi kendini organize eden takımların da katılımı ile icra edilir. Otonom takımlar, yazılım geliştirmenin geleneksel sorumluluklarının yanında hesaplamalar, planlama, görev dağılımı, projenin izlenmesi gibi proje yönetim faaliyetlerini de yürütürler. Takımların bu serbestliğinin yanında proje yönetim faaliyetleri ve karar verme süreçlerinin içine girişi, karar verme otoritesini operasyonel seviyeye indirerek süreçlerde hızlanma ve problem çözümlerinde isabet kazandırır [65].

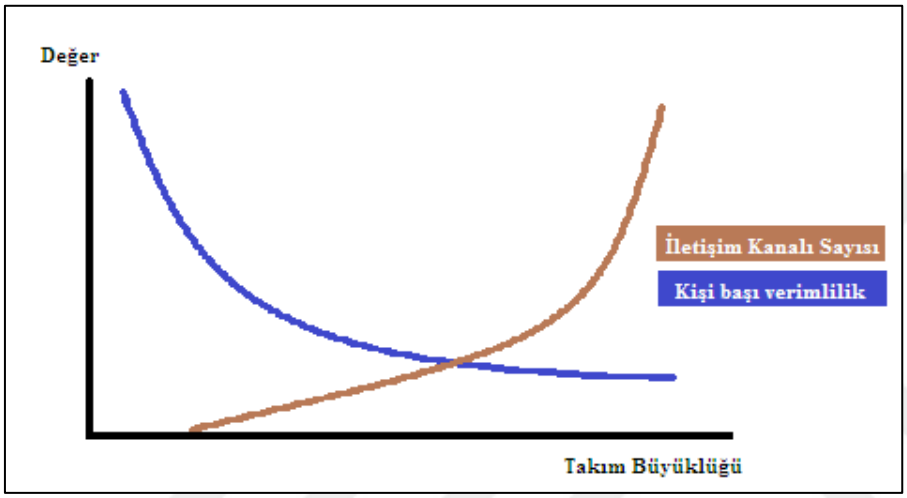
5.11.2. Tecrübe ve yetenek

Çevik yöntemler, kendi kendini organize eden takımlara dayalı olması ve nispeten daha az detaylı tasarımı koda çevirmesi nedeniyle tecrübeli ve yetenekli yazılımcılara ihtiyaç duyar. Takımın kendi kendini ve projenin verimliliğini değerlendirebilmesi gerekir. Bu nedenle takım elemanları bir karşılaştırma yapabilmek için geçmiş projelerde başarı ve başarısızlığı tecrübe etmiş olmalıdır [46]. Proje elemanları tecrübesiz ise otonom bir ortamda kontrolün kaybedilmemesi için plan tabanlı bir yaklaşımla detaylı bir şekilde hazırlanan tasarımın kodlanması daha uygun bir hal tarzıdır.

5.11.3. Takım büyüklüğü

Takımlar büyüdükçe elemanların iletişim etkinliği azalmaktadır. N elemanlı bir grup içindeki tek yönlü iletişim kanalı sayısı n*(n-1) dir. 8 kişilik bir takımda 56 muhtemel iletişim kanalı bulunur [14]. Ayrıca büyük takımlarda, kendi kendini yönetme

becerisi zayıflarken grubun yönetim ihtiyacı doğar. Bu nedenle grup büyüdükçe çevik yöntemler gibi iletişim ve koordinasyona dayalı metodolojilerin verimliliği düşecektir. Şelale modelinde yerleşmiş bir yönetim anlayışı vardır ve yapılan katı iş tanımları nedeniyle iletişim daha az öneme sahiptir. Ayrıca süreç gereği üretilen detaylı dokümantasyon iyi bir iletişim aracıdır. Bu nedenlerle büyük takımlar için şelale modeli daha uygun bir yaklaşımdır. Takım büyüklüğü arttıkça iletişim kanalı sayısı artarken kişi başı verimlilik düşmektedir.



Şekil 5.11. Takım büyüklüğü etkisi [69]

5.11.4. Alan bilgisi

Şelale modelinde proje başlangıcında detaylı gereksinim analizi ve tasarım çalışması yapıldığı için yazılım geliştirme ekibinde çalışan personelin yazılımın kullanılacağı iş alanı ile ilgili bilgisi olmasına gerek yoktur. Geliştirme safhasında kullanılacak girdiler önceki safhaların sonucunda elde edilen çıktılar olduğu için yapılmış analiz ve tasarıma göre kod yazılır. Çevik yöntemlerde ise müşterinin ihtiyacı olan iş süreçlerinin takım tarafından bilinmesine ihtiyaç vardır. Gerek detaylı analiz ve tasarım eksikliği gerekse değişimlere cevap verme zorunluluğu nedeniyle alan bilgisi eksikliği çevik yöntemlerde problemlere neden olabilir. Müşteri katılımı az olduğunda alan bilgisi olmayan bir takımda kullanıcı hikayelerine değer atamak ve öncelik belirlemek büyük zorluk oluşturur [57].

5.11.5. İş birliği

Şelale modelinde görev tanımları nettir ve görevler proje yöneticileri tarafından verilir Bu nedenle proje elemanlarının kendi işlerini yapmaları yeterlidir. İşini iyi yapan personel de bundan dolayı ödüllendirilebilir. Çevik yöntemlerde ise esas olan takımdır ve görev tanımları net değildir. Bu yüzden takımın görevini yerine getirebilmesi için takım elemanlarını çapraz fonksiyonlu olmaları gerekir. Başarılı ya da başarısız olan takımdır bu nedenle bireysel ödül veya ceza olmaz. Şelale modelinde proje süreçlerinde icra edilen görevler proje elemanlarına paylaştırılır ve safha sonunda yapılan işler toparlanarak tamamlanır. Çevik yöntemlerde ise icra edilen faaliyetler takım bütünlüğü içinde icra edilir ve proje küçük parçalara ayrılarak bütün halinde çalışılır. Bu nedenle çevik yöntemlerde görev alan proje elemanlarının projenin bütünü ve genel hedefleri konusunda farkındalık seviyesi daha yüksektir [64].

5.11.6. Yerleşim

Büyük şirketlerin dünyaya yayılmış olmalarının yanında farklı ülkelerdeki işgücünün daha ucuz olması sebebiyle proje bazlı işe alımlar da yapılmaktadır. Proje elemanlarının coğrafi olarak geniş bir alana dağıldığı durumlarda çevik yöntemlerin temel dayanaklarından iletişim zarar görür. Proje elemanlarının çalışma saatleri, büyük zaman dilimi farklılıkları nedeniyle eşzamanlı olamayabileceği için önemli toplantılara katılım konusunda problemler yaşanır, koordinasyonda güçlük çekilir. Telekomünikasyon vasıtalarıyla sağlanan iletişimde yakınlık hissi yüz yüze iletişimdeki kadar yoğun olmaz ve dayanışma azalır. Şelale modelinde yerleşim çevik yöntemler kadar önemli değildir çünkü projenin belli parçaları belli ekiplere bölünebilir fakat çevik yöntemler bu tarz bölünmelere izin vermez [60].

5.11.7. Karmaşıklık

Çevik yöntemlerin, şelale modeline göre, proje elemanları için projenin karmaşıklığı ile ilgili sıkıntıları aşabilmeleri açısından karmaşıklığı azaltan veya karmaşıklığı daha yönetilebilir hale getiren avantajları mevcuttur. Bu avantajlar [64]:

- Takım içinde koordinasyon yapıldığı için herkesin kimin ne yaptığını bilmesi,
- İşin küçük parçalara bölünerek tamamlanması sonrasında müşteriye teslim edilip geri besleme alınması,
- Takım elemanlarının planlama süreçlerinin içinde bulunması
- Takım elemanlarının yapacakları işi kendilerinin seçmesi,
- Proje başında büyük bir planlama yapılmayarak aylık/haftalık planlamaların yapılması.

5.11.8. Ekip çalışması

Ekip çalışmasının kalitesini belirleyen alt başlıklar, iletişim, koordinasyon, üye katkısı dengesi, karşılıklı destek, çaba ve uyumdur [70]. Ekip çalışmasının şelale modeli ve çevik yöntemlerdeki kalitesinin karşılaştırılması için bu alt maddelerin karşılaştırılması gerekir. İletişim, şelale modelinde formeldir ve daha çok proje yöneticisine yazılı durum raporları şeklindedir. Çevik yöntemlerde ise enformel bir iletişim mevcuttur ve ayaküstü toplantılar, kısa sohbetler, ekran karşısında anlatım şeklinde gerçekleşir. Koordinasyon şelale modelinde proje yöneticisinin kontrolü altındadır ve karar verme, tahminlerin yapılması, önceliklerin belirlenmesi ve görev dağılımı proje yöneticisi tarafından icra edilir. Çevik yöntemlerde bu uygulamaların hepsi otonom takımlar tarafından gerçekleştirilir. Şelale modelinde üye katkısının dengeli olması söz konusu değildir. Proje yöneticisi kimin ne iş yapacağına karar verir ve proje elemanları kendilerine verilen görevleri yapmakla mükelleftirler. Çevik yöntemlerde çapraz fonksiyonlu takımlar içinde bütün takım elemanlarının katkısı beklenir ve bu husus günlük toplantılarda koordine edilir. Karşılıklı destek açısından bakıldığında şelale modelinin hiyerarşik yapısı takım üyelerinin arasında karşılıklı bir desteği zorlaştırırken çevik yöntemlerde kodun bütün takıma ait olması, günlük toplantılar, retrospektif incelemeler karşılıklı desteği ve dayanışmayı canlandırır. Şelale modelinde çaba, bireylerin kendilerine verilen görevleri tamamlamaları için harcanır. Çevik yöntemlerde ise çaba takıma aittir ve bireysel iş yoktur. Şelale modelindeki hiyerarşik yapı ve formel iletişim, takım içi uyumu olumsuz etkilerken çevik yöntemlerde genellikle aynı fiziksel alanda bulunan takım elemanları arasındaki etkileşim ile daha yüksek bir uyum yakalanır [70]. Bu sebeplerle ekip çalışması bakımından çevik yöntemler şelale modeline göre daha kaliteli bir yaklaşım ortaya koyar ve ekip çalışmasının önemli olduğu

projeler için çevik yöntemler daha uygundur.

5.12. Karşılaştırma Özeti

Şelale modeli ve çevik yöntemler için yukarıda detaylı karşılaştırması yapılan faktörlerin avantajlı durum özeti Çizelge 5.5'te belirtilmiştir.

Çizelge 5.5. Karşılaştırma özeti

Faktörler	Şelale Modeli	Çevik Yöntemler
Maliyetler	✗	✓
Dış Kaynak Kullanımı	✓	✗
Yazılımın Genel Nitelikleri	✓	✓
Kalite Güvence ve Hata Toleransı	✓	✗
Değer Üretimi	✗	✓
Kullanıcı Deneyimi	✓	✓
Risk Yönetimi	✗	✓
Görünürlük	✓	✓
Proje Karmaşıklığı	✓	✓
Proje Büyüklüğü	✗	✓
Dokümantasyon ve Modelleme	✓	✗
Yeniden Kullanılabilirlik	✓	✗
Gereksinimler	✗	✓
Teslimatlar	✗	✓
Zaman Planlaması	✗	✓
Müşteriler	✗	✓
Kullanım Onayı	✓	✗
Yönetim	✓	✓
Organizasyon Yapısı	✓	✓
Roller	✗	✓
Tecrübe ve Yetenek	✓	✗
Takım büyüklüğü	✓	✗
Alan Bilgisi	✓	✗
İş birliği	✗	✓
Yerleşim	✓	✗
Proje Elemanları Açısından Karmaşıklık	✗	✓
Ekip Çalışması	✗	✓

Çizelge 5.5 incelendiğinde 12 maddede çevik yöntemlerin, 9 maddede ise şelale modelinin daha fazla avantaja sahip olduğu görülmektedir. 6 maddede ise her iki metodolojinin de avantajlı olduğu durumlar mevcuttur.

6. UYGULAMA

Bölüm 5'te, şelale modeli ve çevik yöntemler, literatür araştırması ve kendi değerlendirmemizle belirlenen faktörlere göre karşılaştırılmış ve avantajlı durum özeti Çizelge 5.5'te verilmiştir. İlaveten çalışmada elde edilen bulgunun desteklenmesi için yazılım sektöründe çalışan insanların görüşlerinin alınması ve bu metodolojiler için deneyime dayalı bir karşılaştırma ortaya konması maksadıyla anket uygulaması yapılmıştır.

Yanlış veya yetersiz veri girişini önlemek maksadıyla ankette sorulan sorulara cevap verilmesi zorunlu tutulmamıştır. Katılımcılar istedikleri soruları boş bırakabilmiştir. Uygulanan anket örneği Ek-1'de sunulmuştur.

Literatürde Anaral [71] yazılım geliştirme metodoloji seçiminde kullanılacak faktörleri ve ağırlıklarını çok kriterli karar verme yöntemlerinden AHP ile belirlemiştir. İlgili çalışmada hesaplanan ağırlıklar 1.seviye, 2.seviye ve 3.seviye halinde sınıflandırılarak Çizelge 6.1, 6.2 ve 6.3'te sunulmuştur.

Çizelge 6.1. Birinci seviye faktörlerin ağırlık katsayıları [71]

Faktör	Ağırlık Derecesi
Maliyet	0,33329
Kalite	0,30645
Zaman	0,23490
Kapsam	0,12535

Çizelge 6.2. İkinci seviye faktörlerin ağırlık katsayıları [71]

Maliyet Alt Faktörleri	
Faktör	**Ağırlık Derecesi**
Süreç Maliyetleri	0,75002
Dış Kaynak Maliyetleri	0,24998
Kalite Alt Faktörleri	
Faktör	**Ağırlık Derecesi**
Analiz Tasarım Kalitesi	0,66667
Yazılım Kalitesi	0,33333
Zaman Alt Faktörleri	
Faktör	**Ağırlık Derecesi**
Teslim Süreleri	0,75000
Zaman Planlaması	0,25000
Kapsam Alt Faktörleri	
Faktör	**Ağırlık Derecesi**
İç Etkenler	0,658657
Dış Etkenler	0,185164
Takımın Durumu	0,156180

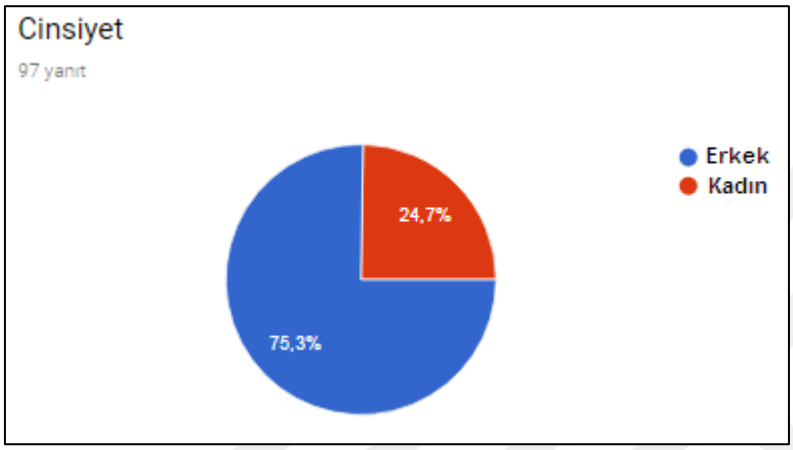
Çizelge 6.3. Üçüncü seviye faktörlerin ağırlık katsayıları [71]

Süreç Maliyetleri Alt Faktörleri	
Faktör	**Ağırlık Derecesi**
Test Maliyeti	0,441308
Yazılım Maliyeti	0,214852
Analiz Maliyeti	0,135988
Tasarım Maliyeti	0,103982
Destek Maliyeti	0,061143
Yaygınlaştırma Maliyeti	0,042727
Dış Kaynak Maliyetleri Alt Faktörleri	
Faktör	**Ağırlık Derecesi**
Dış Kaynak Yazılım Maliyeti	0,625026
Dış Kaynak Destek Maliyeti	0,238476
Dış Kaynak Analiz Maliyeti	0,136498
Yazılım Kalitesi Alt Faktörleri	
Faktör	**Ağırlık Derecesi**
Test Kolaylığı	0,230363
Çalışabilirlik	0,223727
Kod Kalitesi	0,215941
Kullanılabilirlik	0,132776
Güvenilirlik	0,103793
Esneklik	0,056037
Genişletilebilirlik	0,037363
Analiz Tasarım Kalitesi Alt Faktörleri	
Faktör	**Ağırlık Derecesi**
Kullanım Kolaylığı	0,481438
Doküman Kapasitesi	0,295404
Doğruluk	0,130901
Bütünlük	0,092257
Teslim Süreleri Alt Faktörleri	
Faktör	**Ağırlık Derecesi**
Test Süresi	0,302366
Destek Süresi	0,292905
Yazılım Süresi	0,187721
Analiz Süresi	0,120782
Tasarım Süresi	0,058002
Yaygınlaştırma Süresi	0,038224
Zaman Planlaması Alt Faktörleri	
Faktör	**Ağırlık Derecesi**
Öncelikli İşlerin Teslim Süreleri	0,750019
Fazla Mesai Süreleri	0,249981
Dış Etkenler Alt Faktörleri	
Faktör	**Ağırlık Derecesi**
Proje İhtiyaçlarındaki Değişimin Sıklığı	0,508573
Müşteri ile Uyum	0,248966
Teknolojik Gelişmeler	0,176565
Müşteri ile İletişim	0,065896
İç Etkenler Alt Faktörleri	
Faktör	**Ağırlık Derecesi**
Müşteri Talebinin Doğru Şekilde Karşılanması	0,240513
Müşteri Memnuniyeti	0,161326
Müşterinin Ek Taleplerinin Karşılanması	0,151843
Planlama	0,151843
Karmaşıklık	0,090316
Müşterinin Süreç İçerisinde Yer Alması	0,087399
Organizasyon Alışkanlıkları	0,062510
Doküman Sayısı	0,027407
Doküman İhtiyacı	0,026842
Takımın Durumu Alt Faktörleri	
Faktör	**Ağırlık Derecesi**
Takım İçi Organize Yapı	0,301183
Takım Toplantılarının Verimliliği	0,251162
Takım Üyelerinin Motivasyonu	0,14395
Takım Üyelerinin Konsantrasyonu	0,140474
Takımın Tecrübesi	0,06377
Takımın Uyumu	0,060293
Takımdaki Üye Sayısı	0,039168

Çalışmada uygulanan anketin etkinliğini arttırmak için alınan cevaplar Anaral [71] tarafından belirlenen ağırlıklar ile hesaplanarak metodolojiler için karşılama yüzdeleri ortaya konmuştur.

6.1. Anket Uygulama Sonuçları

Ankete katılan 97 kişinin 73'ü erkek, 24'ü kadındır. Katılımcı cinsiyet durumunun toplam katılımcı sayısına oranı Şekil 6.1'de görülmektedir.

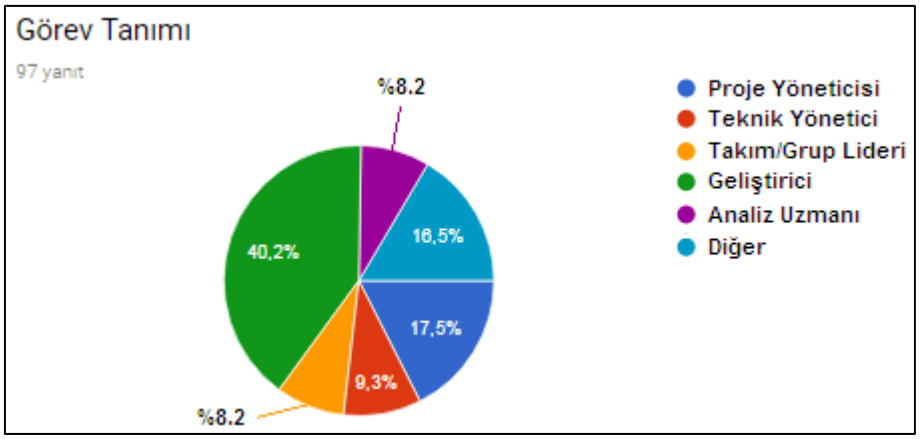


Şekil 6.1. Katılımcıların cinsiyet dağılımı

Katılımcıların 17'si proje yöneticisi, 9'u teknik yönetici, 8'i takım veya grup lideri, 39'u geliştirici, 8'i analiz uzmanıdır. 16 katılımcı diğer seçeneğini işaretlemiştir. Diğer seçeneğini işaretleyen katılımcıların görev tanımları Çizelge 6.4'te sunulmuştur. Katılımcıların görev tanımı dağılımları Şekil 6.2'de görülmektedir.

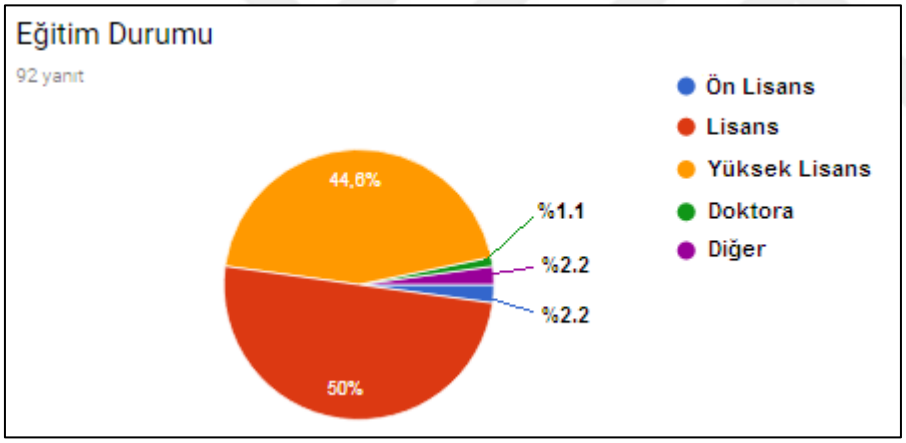
Çizelge 6.4. Diğer seçeneğini işaretleyen katılımcıların görev tanımları

Yapılması Gereken Her Şeyi Yaparım
Çevik Koçu (2)
Çevik Danışmanı
Scrum Master (2)
Yalın Danışmanı
İnsan Kaynakları
Baş Test Mühendisi
Program Yöneticisi
Scrum Master
Kalite Güvence Mühendisi
CIO
Ürün Müdürü
Product Owner
Yazılım Tasarım Uzmanı



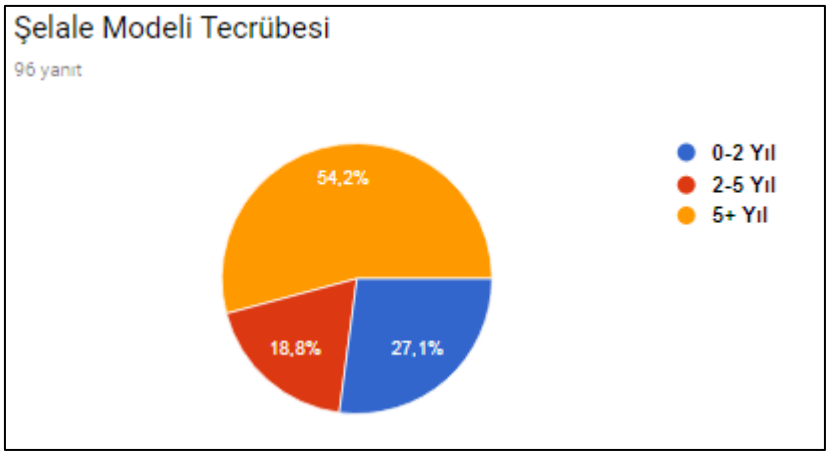
Şekil 6.2. Katılımcıların görev tanımı dağılımı

Katılımcıların 2'si ön lisans, 46'sı lisans, 41'i yüksek lisans, 1'i doktora mezunudur. 5 katılımcı cevap vermemeyi tercih etmiş, 2 katılımcı ise diğer seçeneğini işaretlemiştir. Katılımcıların eğitim durumu dağılımları Şekil 6.3'de görülmektedir.

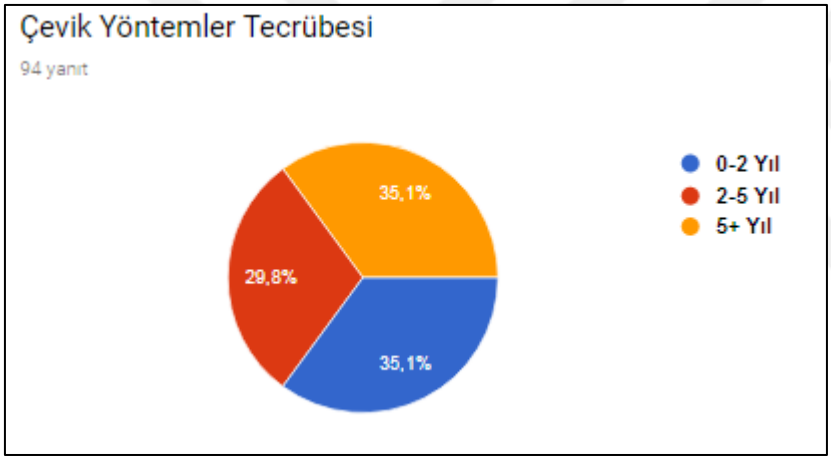


Şekil 6.3. Katılımcıların eğitim durumu dağılımı

Katılımcıların şelale modeli tecrübesi ile ilgili soruya 26 kişi 0-2 yıl, 18 kişi 2-5 yıl, 52 kişi ise 5 yıl ve üzeri seçeneğini işaretlemiş, 1 kişi cevap vermemiştir. Çevik yöntemler tecrübesi ile ilgili soruya ise 33 kişi 0-2 yıl, 28 kişi 2-5 yıl, 33 kişi ise 5 yıl ve üzeri seçeneğini işaretlemiş, 3 kişi cevap vermemiştir. Katılımcıların şelale modeli ve çevik yöntemler ile ilgili tecrübe durumları dağılımları Şekil 6.4 ve 6.5'te görülmektedir.



Şekil 6.4. Katılımcıların şelale modeli tecrübe durumu dağılımı



Şekil 6.5. Katılımcıların çevik yöntemler tecrübe durumu dağılımı

6.2. Uygulama Sonuçlarının Analizi

Anket formunda belirtilen sorulara verilen cevaplar, sorunun içeriğine göre aşağıdaki şekilde birden beşe kadar karşılığı olan seçenekler halinde alınmıştır.

- Çok Yüksek – Yüksek – Orta – Düşük – Çok Düşük
- Çok Kötü – Kötü – Orta – İyi – Çok İyi
- Çok Uzun – Uzun – Orta – Kısa – Çok Kısa
- Çok Fazla – Fazla – Orta – Az – Çok Az

Sorulara verilen cevapların ortalamaları ve ortalamaya göre yüksek metodoloji Çizelge 6.5'te sunulmuştur.

Çizelge 6.5. Metodoloji bazında cevap ortalamaları

Kriter	Şelale Modeli Ortalama	Çevik Yöntemler Ortalama	Sonuç
Analiz Maliyeti	1,82	3,03	Çevik Yöntemler
Destek Maliyeti	2,34	3,05	Çevik Yöntemler
Tasarım Maliyeti	2,14	2,94	Çevik Yöntemler
Test Maliyeti	2,58	2,89	Çevik Yöntemler
Yaygınlaştırma Maliyeti	2,23	3,00	Çevik Yöntemler
Yazılım Maliyeti	2,34	2,90	Çevik Yöntemler
Dış Kaynak Analiz Maliyeti	2,20	2,59	Çevik Yöntemler
Dış Kaynak Destek Maliyeti	2,41	2,73	Çevik Yöntemler
Dış Kaynak Yazılım Maliyeti	2,37	2,70	Çevik Yöntemler
Esneklik	2,27	4,11	Çevik Yöntemler
Genişletilebilirlik	2,62	4,04	Çevik Yöntemler
Güvenilirlik	3,19	3,69	Çevik Yöntemler
Kod Kalitesi	3,26	3,60	Çevik Yöntemler
Kullanılabilirlik	3,02	3,96	Çevik Yöntemler
Test Kolaylığı	2,93	3,75	Çevik Yöntemler
Çalışabilirlik	3,07	3,74	Çevik Yöntemler
Bütünlük	3,24	3,48	Çevik Yöntemler
Doküman Kapasitesi	3,47	3,13	Şelale Modeli
Doğruluk	3,05	3,80	Çevik Yöntemler
Kullanım Kolaylığı	2,98	3,78	Çevik Yöntemler
Analiz Süresi	1,81	3,54	Çevik Yöntemler
Destek Süresi	2,24	3,30	Çevik Yöntemler
Tasarım Süresi	2,13	3,39	Çevik Yöntemler
Test Süresi	2,35	3,46	Çevik Yöntemler
Yaygınlaştırma Süresi	2,40	3,53	Çevik Yöntemler
Yazılım Süresi	2,31	3,35	Çevik Yöntemler
Fazla Mesai Süreleri	2,29	3,25	Çevik Yöntemler
Öncelikli İşlerin Teslim Süreleri	2,12	3,67	Çevik Yöntemler
Müşteri ile İletişim	2,70	4,13	Çevik Yöntemler
Müşteri ile Uyum	2,61	4,04	Çevik Yöntemler
Proje İhtiyaçlarındaki Değişime Uyum	2,19	4,24	Çevik Yöntemler
Teknolojik Gelişmelere Uyum	2,33	4,11	Çevik Yöntemler
Doküman İhtiyacı	2,25	3,54	Çevik Yöntemler
Doküman Sayısı	2,28	3,38	Çevik Yöntemler
Karmaşıklık	2,58	3,40	Çevik Yöntemler
Müşteri Memnuniyeti	2,82	3,92	Çevik Yöntemler
Müşteri Talebinin Doğru Şekilde Karşılanması	2,76	3,99	Çevik Yöntemler
Müşterinin Ek Taleplerinin Karşılanması	2,25	4,05	Çevik Yöntemler
Müşterinin Süreç İçerisinde Yer Alması	2,58	4,19	Çevik Yöntemler
Organizasyon Alışkanlıklarına Uyum	2,90	3,60	Çevik Yöntemler
Planlama	3,098	3,54	Çevik Yöntemler
Takım İçi Organize Yapı	3,05	3,89	Çevik Yöntemler
Takım Toplantılarının Verimliliği	2,84	4,03	Çevik Yöntemler
Takım Üyelerinin Konsantrasyonu	2,84	3,93	Çevik Yöntemler
Takım Üyelerinin Motivasyonu	2,79	4,04	Çevik Yöntemler
Takım için Gereken Üye Sayısı	2,68	3,61	Çevik Yöntemler
Takımın Tecrübesi	2,88	3,80	Çevik Yöntemler
Takımın Uyumu	2,89	3,99	Çevik Yöntemler

Çizelge 6.5 incelendiğinde kriter değerlerine göre “Doküman Kapasitesi” haricinde bütün kriterlerde çevik yöntemlerin üstün geldiği görülmektedir.

Kriter değerleri 97 katılımcının cevaplarının aritmetik ortalaması alınarak hesaplanmıştır. Ancak bütün kriterler aynı önem derecesine sahip değildir. Bu nedenle anketin analizinde sonuçların sadece aritmetik ortalamaya göre değerlendirilmesinin doğru olmayacağı düşünülerek, Anaral [71] tarafından belirlenen kriter ağırlıkları kullanılmıştır. Elde edilen cevapların ağırlıkları ile maliyet, kalite, zaman ve kapsam başlıkları ve bu başlıkların ağırlıkları ile metodolojiler açısından genel bir karşılama oranı hesaplanmıştır.

Şelale modeli için üçüncü seviye kriterlerin ağırlık katsayıları ve karşılama oranları dikkate alınarak ikinci seviye kriterleri karşılama oranları hesaplanmış ve Çizelge 6.6’da sunulmuştur.

Çizelge 6.6. Şelale modeli ikinci seviye kriterlerin karşılanma oranları

2. Seviye Kriter	3. Seviye Kriter	Karşılama Oranı	Ağırlık	Toplam
Süreç Maliyetleri	Analiz Maliyeti	1,82	0,14	%46,5
	Destek Maliyeti	2,34	0,06	
	Tasarım Maliyeti	2,14	0,10	
	Test Maliyeti	2,58	0,44	
	Yaygınlaştırma Maliyeti	2,23	0,04	
	Yazılım Maliyeti	2,34	0,21	
Dış Kaynak Maliyetleri	Dış Kaynak Analiz Maliyeti	2,20	0,14	%47,1
	Dış Kaynak Destek Maliyeti	2,41	0,24	
	Dış Kaynak Yazılım Maliyeti	2,37	0,62	
Yazılım Kalitesi	Esneklik	2,27	0,06	%60,4
	Genişletilebilirlik	2,62	0,04	
	Güvenilirlik	3,19	0,10	
	Kod Kalitesi	3,26	0,22	
	Kullanılabilirlik	3,02	0,13	
	Test Kolaylığı	2,94	0,23	
	Çalışabilirlik	3,07	0,22	
Analiz Tasarım Kalitesi	Bütünlük	3,24	0,09	%63,2
	Doküman Kapasitesi	3,47	0,30	
	Doğruluk	3,05	0,13	
	Kullanım Kolaylığı	2,98	0,48	
Teslim Süreleri	Analiz Süresi	1,81	0,12	%44,7
	Destek Süresi	2,24	0,29	
	Tasarım Süresi	2,13	0,06	
	Test Süresi	2,35	0,30	
	Yaygınlaştırma Süresi	2,40	0,04	
	Yazılım Süresi	2,31	0,19	
Zaman Planlaması	Fazla Mesai Süreleri	2,29	0,25	%43,3
	Öncelikli İşlerin Teslim Süreleri	2,12	0,75	
Dış Etkenler	Müşteri ile İletişim	2,70	0,07	%47,6
	Müşteri ile Uyum	2,61	0,25	
	Proje İhtiyaçlarındaki Değişime Uyum	2,19	0,51	
	Teknolojik Gelişmelere Uyum	2,33	0,18	
İç Etkenler	Doküman İhtiyacı	2,25	0,03	%53,8
	Doküman Sayısı	2,28	0,03	
	Karmaşıklık	2,58	0,09	
	Müşteri Memnuniyeti	2,82	0,16	
	Müşteri Talebinin Doğru Şekilde Karşılanması	2,76	0,24	
	Müşterinin Ek Taleplerinin Karşılanması	2,25	0,15	
	Müşterinin Süreç İçerisinde Yer Alması	2,58	0,09	
	Organizasyon Alışkanlıklarına Uyum	2,90	0,06	
	Planlama	3,10	0,15	
Takımın Durumu	Takım İçi Organize Yapı	3,05	0,30	%57,3
	Takım Toplantılarının Verimliliği	2,84	0,25	
	Takım Üyelerinin Konsantrasyonu	2,84	0,14	
	Takım Üyelerinin Motivasyonu	2,79	0,14	
	Takım için Gereken Üye Sayısı	2,68	0,04	
	Takımın Tecrübesi	2,88	0,06	
	Takımın Uyumu	2,89	0,06	

Şelale modeli için benzer şekilde ikinci seviye kriterlerin ağırlık katsayıları ve karşılama oranları dikkate alınarak birinci seviye kriterler karşılama oranları hesaplanmış ve Çizelge 6.7'de sunulmuştur.

Çizelge 6.7. Şelale modeli birinci seviye kriterlerin karşılanma oranları

1. Seviye Kriter	2. Seviye Kriter	Karşılama Oranı	Ağırlık	Toplam
Maliyet	Süreç Maliyetleri	%46,5	0,75	%46,7
	Dış Kaynak Maliyetleri	%47,1	0,25	
Kalite	Yazılım Kalitesi	%60,4	0,33	%62,3
	Analiz Tasarım Kalitesi	%63,2	0,67	
Zaman	Teslim Süreleri	%44,7	0,75	%44,4
	Zaman Planlaması	%43,3	0,25	
Kapsam	Dış Etkenler	%47,6	0,19	%53,7
	İç Etkenler	%53,8	0,66	
	Takımın Durumu	%57,3	0,16	

Çevik yöntemler için üçüncü seviye kriterlerin ağırlık katsayıları ve karşılama oranları dikkate alınarak ikinci seviye kriterleri karşılama oranları hesaplanmış ve Çizelge 6.8'de sunulmuştur.

Çizelge 6.8. Çevik yöntemler ikinci seviye kriterlerin karşılanma oranları

2. Seviye Kriter	3. Seviye Kriter	Karşılama Oranı	Ağırlık	Toplam
Süreç Maliyetleri	Analiz Maliyeti	3,03	0,14	%58
	Destek Maliyeti	3,05	0,06	
	Tasarım Maliyeti	2,94	0,10	
	Test Maliyeti	2,89	0,44	
	Yaygınlaştırma Maliyeti	3	0,04	
	Yazılım Maliyeti	2,90	0,21	
Dış Kaynak Maliyetleri	Dış Kaynak Analiz Maliyeti	2,59	0,14	%53,8
	Dış Kaynak Destek Maliyeti	2,73	0,24	
	Dış Kaynak Yazılım Maliyeti	2,70	0,62	
Yazılım Kalitesi	Esneklik	4,11	0,06	%75,4
	Genişletilebilirlik	4,04	0,04	
	Güvenilirlik	3,69	0,10	
	Kod Kalitesi	3,60	0,22	
	Kullanılabilirlik	3,96	0,13	
	Test Kolaylığı	3,75	0,23	
	Çalışabilirlik	3,74	0,22	
Analiz Tasarım Kalitesi	Bütünlük	3,48	0,09	%71,2
	Doküman Kapasitesi	3,13	0,30	
	Doğruluk	3,80	0,13	
	Kullanım Kolaylığı	3,78	0,48	
Teslim Süreleri	Analiz Süresi	3,54	0,12	%68
	Destek Süresi	3,30	0,29	
	Tasarım Süresi	3,39	0,06	
	Test Süresi	3,46	0,30	
	Yaygınlaştırma Süresi	3,53	0,04	
	Yazılım Süresi	3,35	0,19	
Zaman Planlaması	Fazla Mesai Süreleri	3,25	0,25	%71,3
	Öncelikli İşlerin Teslim Süreleri	3,67	0,75	
Dış Etkenler	Müşteri ile İletişim	4,13	0,07	%84
	Müşteri ile Uyum	4,04	0,25	
	Proje İhtiyaçlarındaki Değişime Uyum	4,24	0,51	
	Teknolojik Gelişmelere Uyum	4,11	0,18	
İç Etkenler	Doküman İhtiyacı	3,54	0,03	%76,6
	Doküman Sayısı	3,38	0,03	
	Karmaşıklık	3,40	0,09	
	Müşteri Memnuniyeti	3,92	0,16	
	Müşteri Talebinin Doğru Şekilde Karşılanması	3,99	0,24	
	Müşterinin Ek Taleplerinin Karşılanması	4,05	0,15	
	Müşterinin Süreç İçerisinde Yer Alması	4,19	0,09	
	Organizasyon Alışkanlıklarına Uyum	3,6	0,06	
	Planlama	3,54	0,15	
Takımın Durumu	Takım İçi Organize Yapı	3,89	0,30	%78
	Takım Toplantılarının Verimliliği	4,03	0,25	
	Takım Üyelerinin Konsantrasyonu	3,93	0,14	
	Takım Üyelerinin Motivasyonu	4,04	0,14	
	Takım için Gereken Üye Sayısı	3,61	0,04	
	Takımın Tecrübesi	3,80	0,06	
	Takımın Uyumu	3,99	0,06	

Çevik yöntemler için benzer şekilde ikinci seviye kriterlerin ağırlık katsayıları ve karşılama oranları dikkate alınarak birinci seviye kriterler karşılama oranları hesaplanmış ve Çizelge 6.9'da sunulmuştur.

Çizelge 6.9. Çevik yöntemler birinci seviye kriterlerin karşılanma oranları

1. Seviye Kriter	2. Seviye Kriter	Karşılama Oranı	Ağırlık	Toplam
Maliyet	Süreç Maliyetleri	%58	0,75	%57
	Dış Kaynak Maliyetleri	%53,8	0,25	
Kalite	Yazılım Kalitesi	%75,4	0,33	%72,6
	Analiz Tasarım Kalitesi	%71,2	0,67	
Zaman	Teslim Süreleri	%68	0,75	%68,8
	Zaman Planlaması	%71,3	0,25	
Kapsam	Dış Etkenler	%84	0,19	%79
	İç Etkenler	%76,6	0,66	
	Takımın Durumu	%78	0,16	

Şelale modeli ve Çevik yöntemlerin karşılaştırılmasının yapılması için ikinci seviye kriterlere ait sonuçları Çizelge 6.10'da, birinci seviye kriterlere ait sonuçları Çizelge 6.11'de sunulmuştur.

Çizelge 6.10. İkinci seviye kriterlerin karşılanma oranları karşılaştırması

Kriter	Karşılama Oranı		Sonuç
	Şelale Modeli	Çevik Yöntemler	
Süreç Maliyetleri	%46,5	%58	Çevik Yöntemler
Dış Kaynak Maliyetleri	%47,1	%53,8	Çevik Yöntemler
Yazılım Kalitesi	%60,4	%75,4	Çevik Yöntemler
Analiz Tasarım Kalitesi	%63,2	%71,2	Çevik Yöntemler
Teslim Süreleri	%44,7	%68	Çevik Yöntemler
Zaman Planlaması	%43,3	%71,3	Çevik Yöntemler
Dış Etkenler	%47,6	%84	Çevik Yöntemler
İç Etkenler	%53,8	%76,6	Çevik Yöntemler
Takımın Durumu	%57,3	%78	Çevik Yöntemler

Çizelge 6.11. Birinci seviye kriterlerin karşılanma oranları karşılaştırması

Kriter	Karşılama Oranı		Sonuç
	Şelale Modeli	Çevik Yöntemler	
Maliyet	%46,7	%57	Çevik Yöntemler
Kalite	%62,3	%72,6	Çevik Yöntemler
Zaman	%44,4	%68,8	Çevik Yöntemler
Kapsam	%53,7	%79	Çevik Yöntemler

Çizelge 6.10 ve Çizelge 6.11 incelendiğinde katılımcıların verdikleri cevaplar kapsamında gerek ikinci seviye kriterler gerekse birinci seviye kriterler açısından yapılacak karşılaştırmalarda çevik yöntemlerin, şelale modeline göre üstünlük sağladığı görülmektedir.

Şelale modeli ve çevik yöntemlerin, metodoloji seçimine etki ettiği değerlendirilen kriterleri nihai karşılama oranları Çizelge 6.12'te sunulmuştur.

Çizelge 6.12. Şelale modeli ve çevik yöntemlerin nihai karşılama oranları

Metodoloji	1. Seviye Kriter	Karşılama Oranı	Ağırlık	Toplam
Şelale Modeli	Maliyet	%46,7	0,33	% 51,91
	Kalite	%62,3	0,31	
	Zaman	%44,4	0,23	
	Kapsam	%53,7	0,13	
Çevik Yöntemler	Maliyet	%57	0,33	% 67,4
	Kalite	%72,6	0,31	
	Zaman	%68,8	0,23	
	Kapsam	%79	0,13	

Çizelge 6.12 incelendiğinde, uygulanan anket sonucu metodoloji seçimine etki ettiği değerlendirilen faktörlerin karşılanmasında, çevik yöntemler için %67,4; şelale modeli için ise % 51,91 karşılama oranı elde edildiği görülmektedir.

7. SONUÇ

Yazılım projeleri özelleşmiş projelerdir ve proje yönetiminin temel özelliklerini taşırlar. Fakat yazılım projelerini diğer projelerden ayıran önemli farklar bulunmaktadır. En önemli fark, diğer mühendislik süreçlerinde üretilen ürünün fiziksel olarak somut olması, yazılımın ise düşünsel süreçlerin bir tezahürü olarak soyut olmasıdır. Bu soyutluk nedeniyle proje yönetim süreçlerinin izlenebilirliği sekteye uğramakta ve projenin belirsizliği artmaktadır. Bu nedenle yazılım projeleri içerikleri gereği özel yönetim teknikleri gerektirirler.

Şelale modeli yazılım mühendisliğinin en eski ve temel modelidir. Günümüzde kamu kuruluşları ve büyük şirketler tarafından her türlü projenin yönetim standardı olarak kabul görmektedir [19].

Çevik yöntemler, geleneksel yöntemlerin yetersiz kaldığı değerlendirilen konular için alternatif çözümler olarak ortaya çıkmıştır. Günümüzde gerek Türkiye gerekse dünya çapında yazılım proje yönetiminde daha çok çevik yöntemler tercih edilmektedir [30,31].

Çalışma kapsamında, çevik yöntemler ile şelale modelinin proje başarısına etki ettiği değerlendirilen faktörler açısından durumsal incelemeleri ve karşılaştırmaları yapılmıştır. Bu karşılaştırmalarda, 12 durumda çevik yöntemlerin; 9 durumda şelale modelinin avantajlı olduğu değerlendirilmiştir. Yapılan genel değerlendirmede çevik yöntemlerin şelale modeline göre daha avantajlı olduğu sonucuna varılmıştır.

Elde edilen sonucun, deneysel tecrübe ile karşılaştırılması maksadıyla yazılım sektöründen 97 çalışanın katılımıyla bir anket uygulanmıştır. Anket sonucunda, 1 kriterde şelale modeli tercih edilirken, 48 kriterde çevik yöntemler tercih edilmiştir.

Ankette yer alan bütün kriterlerin aynı önem derecesine sahip olmadığı ve anketin analizinde alınan sonuçların salt aritmetik ortalamasının kullanılmasının doğru olmayacağı değerlendirilmiş, bu nedenle Anaral [71] tarafından belirlenen kriterler ve ağırlıklar kullanılmıştır. Elde edilen cevapların ağırlıkları ile maliyet, kalite, zaman ve kapsam başlıkları ve bu başlıkların ağırlıkları ile metodolojiler açısından genel bir karşılama oranı hesaplanmıştır. Metodoloji seçimini etkilediği

değerlendirilen kriterlerin karşılama oranları çevik yöntemler için %67,4; şelale modeli için %51,91 olarak elde edilmiştir. Çalışma kapsamında çevik yöntemler bir bütün olarak ele alınmıştır. Bu nedenle anket katılımcılarının, deneyimli oldukları çevik yöntemleri oluşturan metodolojilerle ilgili tecrübe ettikleri olumlu ve olumsuz durumların bir bütün olarak çevik yöntemlerle ilgili görüş oluşturduğu dikkate alınmalıdır.

Yapılan durumsal karşılaştırmalarda, uygulanan ankete verilen cevaplarda ve son olarak anket sonuçlarının kriter ağırlıklarına göre değerlendirilmesi sonucunda çevik yöntemlerin şelale modeline göre daha avantajlı olduğu görülmüştür. Buna rağmen şelale modeli ve çevik yöntemler güçlü ve zayıf yönleriyle uygunluk açısından proje bazında değerlendirilmesi gereken metodolojilerdir.

Şelale modeli projenin geçmiş safhalarına bağımlı olarak yürütüldüğü için anlayış olarak geriye dönüktür ve bütün proje elemanlarının kendisi için tanımlanan işi yapmasının beklendiği bir mühendislik düzenidir. Bu düzenden faydalanan şelale modeli daha çok, gereksinimlerin ve hedeflerin açık ve net olduğu, çözümlerin ve yapılacak işin bilindiği, gereksinimlerin değişmeyeceği, üretilecek yazılımın hata toleransının olmadığı ve fonksiyonel bütünlüğe sahip projeler için uygundur. En iyi kullanım alanı olarak savunma sistemleri ve gömülü sistemler verilebilir.

Çevik yöntemler ise değişimi ve geri beslemeyi teşvik eden yapısı nedeniyle ileri dönük bir anlayışa sahiptir. Otonom takımların her işi kendi kendine yapması ve bu süreçte çapraz fonksiyonellikten ve deneycilikten faydalanmaları beklenir. Bu da düzenden çok bir yaratıcılık yaklaşımı gerektirir. Bu nedenle çevik yöntemler gereksinimlerin ve hedeflerin belirsiz ve bilinmez olduğu, değişikliğin sık, değer üretiminin kısa vadede arzu edildiği projeler için daha uygundur. Çevik yöntemler için en iyi kullanım alanı web tabanlı iş akış yönetimi sistemleridir.

İki yöntemin karışımı olarak ortaya çıkan metodolojiler de kullanılmakta, yazılım üreticileri ihtiyaçlarına göre metodolojilerin arzu edilen tekniklerini birleştirilerek karma metodolojiler uygulamaktadırlar. Örneğin birçok organizasyon, yazılım projesinin kodlama ve test safhalarının scrum metodu ile, planlama, tasarım ve teslimat gibi geri kalan safhaların da şelale modelinin prensipleri ile gerçekleştirildiği Water-Scrum-Fall metodolojisini uygulamaktadır [72].

Günümüzde teknolojideki hızlı gelişim, bütün iş süreçlerini, pazar dinamiklerini ve bu doğrultuda yazılıma ihtiyaç duyan müşterilerin gereksinimlerini etkilemekte ve geçmişe nazaran daha değişken bir ortam yaratmaktadır. Bu değişken ortamda çevik yöntemler şelale modeline göre avantajlı olsa da bazı şartlarda şelale modelinin sağladığı güven ve düzeni sağlayamamaktadır. Bu nedenle başarılı bir yazılım proje yöneticisi hem şelale modeli hem de çevik yöntemlerin anlayışına, avantaj ve dezavantajlarına hâkim olmalı ve projenin ihtiyaçlarına göre uygun bir metodoloji benimsemelidir. Bu çalışma yazılım proje yönetiminde karşılaşılabilecek durumlar için şelale modeli ve çevik yöntemlerin avantaj ve dezavantajları açısından bir öneri niteliği taşımaktadır.

Daha önce de belirtildiği üzere şelale modelinin en büyük avantajlarından biri kamu kuruluşları ve büyük şirketler tarafından proje yönetim standardı olarak kabul görmesidir. Bundan sonra yapılacak çalışmalarda, özellikle Millî Savunma Bakanlığı yazılım projelerinde, çevik yöntemlerin avantajlarından yararlanılması ve şelale modeli yerine çevik yöntemlerin yaygınlaştırılmasının ele alınabileceği değerlendirilmektedir.

KAYNAKLAR

1. Project Management Institute. (2013). *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, A.B.D. Pennsylvania: Project Management Institute.

2. İnternet:URL: http://www.webcitation.org/query?url=https%3A%2F%2Ftr.0wikipedia.org%2Findex.php%3Fq%3DaHR0cHM6Ly90ci53aWtpcGVkaWEub3JnL3dpa2kvUHJvamU&date=2017-09-28, Son Erişim Tarihi: 29.09.2017.

3. Heyworth, F., & European Centre for Modern Languages. (2002). *A guide to project management*. Strasbourg: Council of Europe.

4. İnternet:URL: http://www.webcitation.org/query?url=https%3A%2F%2Ftr.0wikipedia.org%2Findex.php%3Fq%3DaHR0cHM6Ly90ci53aWtpcGVkaWEub3JnL3dpa2kvUHJvamVfecO2bmV0aW1p&date=2017-09-28, Son Erişim Tarihi: 29.09.2017.

5. Burke, R. (2003). *Project management: Planning and control techniques (Fourth Editiion)*. Chichester: Wiley,10-11.

6. Demirel, K. (2014). *Proje Yönetimi El Kitabı,* Yayımlanmamış Yüksek Lisans Tezi, Kocaeli Üniversitesi Sosyal Bilimler Enstitüsü, Kocaeli.

7. Kır, E. (2007). *Yazılım Sektöründe Proje Yönetimi*, Yayımlanmamış Yüksek Lisans Tezi, Kadir Has Üniversitesi Sosyal Bilimler Enstitüsü, İstanbul.

8. Albayrak, B. (2005). *Proje yönetimi*. İstanbul: Nobel Yayın Dağıtım.

9. Butler, A. G. (1973). Project Management: A Study in Organizational Conflict. *The Academy of Management Journal, 16,* 1, 84-101.

10. Esatoğlu, N. (2010). *Bilgi Teknolojileri Proje Yönetimi ve Başarı Koşulları*, Yayımlanmamış Yüksek Lisans Tezi, Ankara Üniversitesi Sosyal Bilimler Enstitüsü, Ankara.

11. Türkiye Bilişim Derneği 2. Çalışma Grubu. (2006), Bilişim Teknolojilerinde Risk Yönetimi; Türkiye Bilişim Derneği, *Antalya.*

12. Koçel, T. (2005). *İşletme yöneticiliği*. İstanbul: Arıkan Yayınevi, s.390-392.

13. İnternet:URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.tdk.gov.tr%2Findex.php%3Foption%3Dcom_gts%26arama%3Dgts%26guid%3DTDK.GTS.59cd94ca817c86.31051137&date=2017-09-28, Son Erişim Tarihi: 29.09.2017.

14. Sommerville, I. (2011). *Software engineering*. Boston: Addison-Wesley.

15. IEEE Computer Society (2014). *SWEBOK: Guide to the software engineering body of knowledge*. A.B.D.: IEEE.

16. ISO/IEC/IEEE (2010). *ISO/IEC/IEEE 24765:2010 Systems and Software Engineering—Vocabulary*. A.B.D.: ISO/IEC/IEEE.

17. ISO/IEC (2011). *ISO/IEC 25010:2011 Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE)—Systems and Software Quality Models*, A.B.D.: ISO/IEC.

18. Standish Group International. (2015). CHAOS Report 2015; Standish Group International. *U.S.A.*

19. Munassar, N.M.A., Govardhan, A. (September 01, 2010). A Comparison Between Five Models Of Software Engineering. *International Journal of Computer Science Issues, 7,* 5, 94-101.

20. Naur, P., Randell, B., & NATO Science Committee. (1968). *Software engineering: Report on a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th Oct. 1968*. Brussels: NATO Scientific Affairs Division.

21. Royce, W. W. (1987, March). Managing the development of large software systems: concepts and techniques. In *Proceedings of the 9th international conference on Software Engineering*. 328-338.

22. Palmquist, M. S., Lapham, M. A., Miller, S., Chick, T., & Ozkaya, I. (2013). *Parallel worlds: Agile and waterfall differences and similarities* (No. CMU/SEI-2013-TN-021). Carnegie-Mellon University Pıttsburgh Pa Software Engıneerıng Inst.

23. Farrell, A. (2007). Selecting a Software Development Methodology based on Organizational Characteristics. *An Essay Submitted in Partial Fulfillment of the Requirements for the Degree of "Master of Science in Information Systems", Athabasca University, Athabasca*.

24. Vohra, P., Singh, A. (2013). A Contrast and Comparison of Modern Software Process Models. *International Conference on Advances in Management and Technology (iCAMT-2013)*. 23-27.

25. Endres, A. (1996). A synopsis of software engineering history: the industrial perspective. In *History of Software Engineering, Position Papers for Dagstuhl Seminar*. Vol. 9635. 20-24.

26. Balaji, S., Murugaiyan, M. S. (2012). Waterfall vs. V-Model vs. Agile: A comparative study on SDLC. *International Journal of Information Technology and Business Management*, *2*(1), 26-30.

27. Awad, M. A. (2005). *A comparison between agile and traditional software development methodologies*. University of Western Australia.

28. İnternet: The Standard Waterfall Model for Systems Development. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fweb.archive.org%2Fweb%2F20040403211247%2Fhttp%3A%2F%2Fasd-www.larc.nasa.gov%2Fbarkstrom%2Fpublic%2FThe_Standard_Waterfall_Model_For_Systems_Development.htm&date=2017-09-28, Son Erişim Tarihi: 29.09.2017.

29. İnternet: Ambler, S. W., Disciplined Agile Software Development: Definition. URL:http://www.webcitation.org/query?url=http%3A%2F%2Fwww.agilemodeling.com%2Fessays%2FagileSoftwareDevelopment.htm+&date=2017-09-28, Son Erişim Tarihi: 29.09.2017.

30. Hewlett-Packard Enterprise. (2017). Agile is the new normal; Hewlett-Packard Enterprise, *U.S.A*.

31. AgileTurkey. (2016). 5th Annual Agility Report; AgileTurkey, *Türkiye.*

32. Beck, K. (1999). Embracing change with extreme programming. *Computer*, *32*(10), 70-77.

33. Schwaber, K. (1997). Scrum development process. In *Business Object Design and Implementation* 117-134. Springer London.

34. Jeffries, R.E. (1999). Extreme Testing – Why Aggressive Software Development Calls for Radical Testing Efforts, *Software Testing & Quality Engineering*, Vol.March/April, 23-26.

35. Cockburn, A. (1998). *Surviving object-oriented projects: A manager's guide*. Reading, Mass: Addison-Wesley.

36. Stapleton, J., & DSDM Consortium. (2003). *DSDM: Business focused development*. London: Addison-Wesley.

37. Highsmith, J. A. (2000). *Adaptive software development: a collaborative approach to managing complex systems*. New York, N.Y: Dorset House.

38. Coad, P., Lefebvre, E., DeLuca. J. (2000). *Java modeling in color with UML: Enterprise components and process*. Upper Saddle River NJ: Prentice Hall PTR.

39. İnternet: Highsmith, J. (2001). History: The Agile Manifesto. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fagilemanifesto.org%2Fhistory.html&date=2017-09-28, Son Erişim Tarihi: 29.09.2017.

40. İnternet: Cunningham, W. (2001). Çevik Yazılım Geliştirme Manifestosu. URL:http://www.webcitation.org/query?url=http%3A%2F%2Fagilemanifesto.org%2Fiso%2Ftr%2Fmanifesto.html&date=2017-09-28, Son Erişim Tarihi: 29.09.2017.

41. Fair, J., Project Management Institute., & PMI Global Congress--EMEA. (2012). *Agile versus Waterfall: Approach is right for my ERP project?*. Newtown Square, Pa: Project Management Institute.

42. VersionOne Inc. (2017). 11th Annual State of Agile Report; VersionOne Inc., *U.S.A.*

43. Forrester Research Inc. (2015). The 2015 State of Agile Development; Forrester Research Inc., *U.S.A.*

44. Flora, H. K., Chande, S. V. (2014). A systematic study on agile software development methodologies and practices. *International Journal of Computer Science and Information Technologies*, *5*(3), 3626-3637.

45. Lapham, M. A., Williams, R., Hammons, C., Burton, D., Schenker, A. (2010). *Considerations for using agile in DoD acquisition* (No. CMU/SEI-2010-TN-002). Pittsburgh, Pa: Carnegie Mellon University, Software Engineering Institute

46. Turk, D., France, R. D., & Rumpe, B. (2005). Assumptions Underlying Agile Software-Development Processes. *Journal of Database Management, 16,* 4, 62-87.

47. Turk, D., France, R., and Rumpe, B., (2002), Limitations of Agile Software Processes, *Proceedings of the Third International Conference on eXtreme Programming and Agile Processes in Software Engineering*, 43-46.

48. Hiwarkar, K., Doshi, A., Chinta, R., Manjula, R. (2016). Comparative Analysis of Agile Software Development Methodologies-A Review. *International Journal of Engineering Research and Applications*, 6(3), 80-85.

49. Qumer, A., & Henderson-Sellers, B. (2008). An evaluation of the degree of agility in six agile methods and its applicability for method engineering. *Information and software technology*, *50*(4), 280-295.

50. Mahapatra, H.B., Goswami, B. (2013). Selection of Software Development Methodology (SDM): A Comparative Approach, *International Journal of Advanced Research in Computer Science and Software Engineering*, 5(3), 58-61.

51. Simão, E. M. (2011). *Comparison of software development methodologies based on the SWEBOK* (Doctoral dissertation), Universidade do Minho, Escola de Engenharia.

52. Mahalakshmi, M., Sundararajan, M. (2013). Traditional SDLC vs Scrum methodology–A comparative study. *International Journal of Emerging Technology and Advanced Engineering*, *3*(6), 192-196.

53. Macit, Y., Tüzün, E. (2015). Uygulama Yaşam Döngüsü Yönetimi Karşılaştırmalı Süreç İncelemesi. *9'uncu Ulusal Yazılım Mühendisliği Sempozyumu-UYMS 2015*, 122-133.

54. J Dorette Jacob, J. (2011). *Comparing agile xp and waterfall software development processes in two start-up companies*. Chalmers University of Technology, Department of Computer Science and Engineering.

55. Komai, S., Saidi, H., Nakanishi, H. (2016). Man-Hour Comparison Between Two Methods of Agile and Waterfall in IT System Development, *Proceedings of the 13th International Conference on Innovation and Management*, 829-836.

56. Wells, H., Dalcher, D., Smyth, H. (2015, January). The adoption of agile management practices in a traditional project environment: An IT/IS Case Study. In *2015 48th Hawaii International Conference on System Sciences (HICSS)-IEEE,* 4446-4453.

57. Van Waardenburg, G., Van Vliet, H. (2013). When agile meets the enterprise. *Information and software technology*, *55*(12), 2154-2171.

58. Edberg, D., Ivanova, P., Kuechler, W. (2012). Methodology mashups: An exploration of processes used to maintain software. *Journal of Management Information Systems*, *28*(4), 271-304.

59. Dingsøyr, T., Lassenius, C. (2016). Emerging themes in agile software development: Introduction to the special section on continuous value delivery. *Information and Software Technology*, *77*, 56-60.

60. Dikert, K., Paasivaara, M., Lassenius, C. (2016). Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software*, *119*, 87-108.

61. Gümrükçü, G. (2010). *Savunma Sistemlerinde Yazılım Proje Yönetimi*, Yayımlanmamış Yüksek Lisans Tezi, Ankara Üniversitesi Fen Bilimleri Enstitüsü, Ankara.

62. Sarıdoğan, M. E. (2004). *Yazılım mühendisliği*. İstanbul: Papatya.

63. Canditt, S., Rauh, D., Wittmann, M. (2010). Brückenschlag: Das V-Modell XT mit Scrum inside. *Objekt Spektrum*, (5), 54.

64. Sohi, A. J., Hertogh, M., Bosch-Rekveldt, M., Blom, R. (2016). Does Lean & Agile Project Management Help Coping with Project Complexity?. *Procedia-Social and Behavioral Sciences*, *226*, 252-259.

65. Hoda, R., Murugesan, L. K. (2016). Multi-level agile project management challenges: A self-organizing team perspective. *Journal of Systems and Software*, *117*, 245-257.

66. Hoda, R., Noble, J., Marshall, S. (2011). The impact of inadequate customer collaboration on self-organizing Agile teams. *Information and Software Technology*, *53*(5), 521-534.

67. Mintzberg, H. (1989). *Mintzberg on Management*. New York, NY: The Free Press.

68. İnternet: Bulu, M. Nasil Bir Organizasyon Yapisina Ihtiyaciniz Var?. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.competitiveturkey.org%2Ftrial%2Fmelih_bulu_yazi%2FNasilBirOrganizasyonYapisinaihtiyacinizVar.pdf&date=2017-09-28, Son Erişim Tarihi: 29.09.2017.

69. Cockburn, A. (1999). Methodology per project, *Human and Technology Technical Report*, 04 Dec 1999.

70. Lindsjørn, Y., Sjøberg, D. I., Dingsøyr, T., Bergersen, G. R., Dybå, T. (2016). Teamwork quality and project success in software development: A survey of agile development teams. *Journal of Systems and Software*, *122*, 274-286.

71. Anaral, F. (2012). *Çok Kriterli Karar Verme Yöntemi ile Yazılım Geliştirme Metodolojisi Seçimi*, Yayımlanmamış Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

72. West, D., Gilpin, M., Grant, T., Anderson, A. (2011*).* Water-Scrum-Fall is The Reality Of Agile For Most Organizations Today; Forrester Research Inc., *U.S.A.*

73. Vaidya, A. (2014). Does DAD Know Best, Is it Better to do LeSS or Just be SAFe? Adapting Scaling Agile Practices into the Enterprise, *PNSQC 2014 Proceedings*, 21-38.

74. Calp, M. H., Akcayol, M. A. (2015). Yazılım Projelerinde Karşılaşılan Risk Faktörleri ve Risk Yönetim Süreci, *Marmara Fen Bilimleri Dergisi*,*1*, 1-13.

75. Katz, K. (2009). In Defense of Waterfall: Deconstructing the Agile Manifesto, *Better Software Confrence & Expo, Las* Vegas*, A.B.D.*

EKLER

EK-1. Uygulanan Anket Örneği

ŞELALE MODELİ İLE ÇEVİK YÖNTEMLERİN KARŞILAŞTIRILMASI

Değerli katılımcı,
Bu çalışmanın amacı, günümüzün en popüler yazılım proje yönetimi metodolojileri olan şelale modeli ile çevik yöntemlerin, proje başarısına etki eden faktörler açısından karşılaştırılmasıdır. Ankette bulunan sorulara vereceğiniz cevaplar tarafımızca saklı tutulacak ve sadece bilimsel amaçlı olarak kullanılacaktır. Anket sonuçlarının sağlıklı olabilmesi için soruları samimi ve doğru bir şekilde yanıtlamanız büyük önem taşımaktadır. İlgi ve yardımlarınız için teşekkür ederiz.

Katılımcı Profili:

Cinsiyet : Erkek ☐
Kadın ☐

Görev Tanımı : Proje Yöneticisi ☐
Teknik Yönetici ☐
Takım / Grup Lideri ☐
Geliştirici ☐
Analiz Uzmanı ☐
Diğer: .....

Şelale Modeli Tecrübesi:
0-2 Yıl ☐
2-5 Yıl ☐
5 Yıl ve Üzeri ☐

Eğitim Durumu : Ön Lisans ☐
Lisans ☐
Yüksek Lisans ☐
Doktora ☐

Çevik Yöntemler Tecrübesi:
0-2 Yıl ☐
2-5 Yıl ☐
5 Yıl ve Üzeri ☐

EK-1. (devam) Uygulanan Anket Örneği

MALIYET		*Şelale Modeli*					*Çevik Yöntemler*				
SÜREÇ MALIYETLERI	Analiz Maliyeti	Çok Yüksek	Yüksek	Orta	Düşük	Çok Düşük	Çok Yüksek	Yüksek	Orta	Düşük	Çok Düşük
	Destek Maliyeti	Çok Yüksek	Yüksek	Orta	Düşük	Çok Düşük	Çok Yüksek	Yüksek	Orta	Düşük	Çok Düşük
	Tasarım Maliyeti	Çok Yüksek	Yüksek	Orta	Düşük	Çok Düşük	Çok Yüksek	Yüksek	Orta	Düşük	Çok Düşük
	Test Maliyeti	Çok Yüksek	Yüksek	Orta	Düşük	Çok Düşük	Çok Yüksek	Yüksek	Orta	Düşük	Çok Düşük
	Yaygınlaştırma Maliyeti	Çok Yüksek	Yüksek	Orta	Düşük	Çok Düşük	Çok Yüksek	Yüksek	Orta	Düşük	Çok Düşük
	Yazılım Maliyeti	Çok Yüksek	Yüksek	Orta	Düşük	Çok Düşük	Çok Yüksek	Yüksek	Orta	Düşük	Çok Düşük
DIŞ KAYNAK MALIYETLERI	Dış Kaynak Analiz Maliyeti	Çok Yüksek	Yüksek	Orta	Düşük	Çok Düşük	Çok Yüksek	Yüksek	Orta	Düşük	Çok Düşük
	Dış Kaynak Destek Maliyeti	Çok Yüksek	Yüksek	Orta	Düşük	Çok Düşük	Çok Yüksek	Yüksek	Orta	Düşük	Çok Düşük
	Dış Kaynak Yazılım Maliyeti	Çok Yüksek	Yüksek	Orta	Düşük	Çok Düşük	Çok Yüksek	Yüksek	Orta	Düşük	Çok Düşük

KALİTE		*Şelale Modeli*					*Çevik Yöntemler*				
YAZILIM KALİTESİ	Esneklik	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
	Genişletilebilirlik	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
	Güvenilirlik	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
	Kod Kalitesi	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
	Kullanılabilirlik	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
	Test Kolaylığı	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
	Çalışabilirlik	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
ANALİZ TASARIM KALİTESİ	Bütünlük	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
	Doküman Kapasitesi	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
	Doğruluk	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
	Kullanım Kolaylığı	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi

ZAMAN		*Şelale Modeli*					*Çevik Yöntemler*				
TESLİM SÜRELERİ	Analiz Süresi	Çok Uzun	Uzun	Orta	Kısa	Çok Kısa	Çok Uzun	Uzun	Orta	Kısa	Çok Kısa
	Destek Süresi	Çok Uzun	Uzun	Orta	Kısa	Çok Kısa	Çok Uzun	Uzun	Orta	Kısa	Çok Kısa
	Tasarım Süresi	Çok Uzun	Uzun	Orta	Kısa	Çok Kısa	Çok Uzun	Uzun	Orta	Kısa	Çok Kısa
	Test Süresi	Çok Uzun	Uzun	Orta	Kısa	Çok Kısa	Çok Uzun	Uzun	Orta	Kısa	Çok Kısa
	Yaygınlaştırma Süresi	Çok Uzun	Uzun	Orta	Kısa	Çok Kısa	Çok Uzun	Uzun	Orta	Kısa	Çok Kısa
	Yazılım Süresi	Çok Uzun	Uzun	Orta	Kısa	Çok Kısa	Çok Uzun	Uzun	Orta	Kısa	Çok Kısa
ZAMAN PLANLAMASI	Fala Mesai	Çok Fazla	Fazla	Orta	Az	Çok Az	Çok Fazla	Fazla	Orta	Az	Çok Az
	Öncelikli İşlerin Teslim Süreleri	Çok Uzun	Uzun	Orta	Kısa	Çok Kısa	Çok Uzun	Uzun	Orta	Kısa	Çok Kısa

EK-1. (devam) Uygulanan Anket Örneği

KAPSAM		*Şelale Modeli*					*Çevik Yöntemler*				
DIŞ ETKENLER	Müşteri ile İletişim	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
	Müşteri ile Uyum	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
	Proje İhtiyaçlarındaki Değişime Uyum	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
	Teknolojik Gelişmelere Uyum	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
İÇ ETKENLER	Doküman İhtiyacı	Çok Fazla	Fazla	Orta	Az	Çok Az	Çok Fazla	Fazla	Orta	Az	Çok Az
	Doküman Sayısı	Çok Fazla	Fazla	Orta	Az	Çok Az	Çok Fazla	Fazla	Orta	Az	Çok Az
	Karmaşıklık	Çok Karmaşık	Karmaşık	Orta	Anlaşılır	Çok Anlaşılır	Çok Fazla	Fazla	Orta	Az	Çok Az
	Müşteri Memnuniyeti	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
	Müşteri Talebinin Doğru Şekilde Karşılanması	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
	Müşterini Ek Taleplerinin Karşılanması	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
	Müşterinin Süreç İçerisinde Yer Alması	Hiç Yer Almaz	Çok az Yer Alır	Orta	Yeteri Kadar Yer Alır	Çok Sık Yer Alır	Hiç Yer Almaz	Çok az Yer Alır	Orta	Yeteri Kadar Yer Alır	Çok Sık Yer Alır
	Organizasyon Alışkanlıklarına Uyum	Hiç Uygun Değil	Uygun Değil	Orta	Uygun	Çok Uygun	Hiç Uygun Değil	Uygun Değil	Orta	Uygun	Çok Uygun
	Planlama	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
TAKIMIN DURUMU	Takım İçi Organize Yapı	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
	Takım Toplantılarının Verimliliği	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
	Takım Üyelerinin Konsantrasyonu	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
	Takım Üyelerinin Motivasyonu	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
	Takım için gereken üye sayısı	Çok Fazla	Fazla	Orta	Az	Çok Az	Çok Fazla	Fazla	Orta	Az	Çok Az
	Takımın Tecrübesi	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi
	Takımın Uyumu	Çok Kötü	Kötü	Orta	İyi	Çok İyi	Çok Kötü	Kötü	Orta	İyi	Çok İyi

ÖZGEÇMİŞ



Kişisel Bilgiler

Soyadı, adı	: KAYACAN, Ali
Uyruğu	: TC
Doğum tarihi ve yeri	: 13.06.1983 - Balıkesir
Medeni hali	: Evli
Telefon	: 0 (266) 239 45 78
e-mail	: ali.kayacan83@gmail.com

Eğitim Derecesi	Okul/Program	Mezuniyet Yılı
Yüksek Lisans	Gazi Üniversitesi Bilişim Enstitüsü Yönetim Bilişim Sistemleri	2017
Lisans	Kara Harp Okulu	2005
Lise	Maltepe Askeri Lisesi	2001

İş Deneyimi, Yıl	Çalıştığı Yer	Görev
2005-2010	Kara Kuvvetleri Komutanlığı	Takım Komutanı
2010-2011	Kara Harp Okulu	Bil.Sis. Subay Temel Kursu (Kursiyer)
2011-Devam Ediyor	Kara Kuvvetleri Komutanlığı	Bil.Sis. Subayı

Yabancı Dil

İngilizce, Almanca

Hobiler

Futbol, kayak, tenis, bilgi teknolojileri, seyahat.



GAZİLİ OLMAK AYRICALIKTIR.