

FAKÜLTE ADI BURAYA YAZILMALIDIR

PROGRAM ADI BURAYA YAZILMALIDIR

TEZSİZ YÜKSEK LİSANS DÖNEM PROJESİ

DÖNEM PROJESİNİN ADI BU BÖLÜME YAZILMALIDIR

HAZIRLAYAN

Hazırlayanın adı soyadı bu bölüme yazılmalıdır

DANIŞMAN ÖĞRETİM ÜYESİ

Danışman adı soyadı bu bölüme yazılmalıdır

2018

# ETİK İLKELERE UYGUNLUK BEYANI

Dönem proje yazma sürecinde bilimsel ve etik ilkelere uyduğumu, yararlandığım tüm kaynakları kaynak gösterme ilkelerine uygun olarak kaynakçada belirttiğimi ve bu bölümler dışındaki tüm ifadelerin şahsıma ait olduğunu beyan ederim.

İmza Adı Soyadı

**DÖNEM PROJESİNİN ADI BU BÖLÜME YAZILMALIDIR**

**Hazırlayanın adı soyadı bu bölüme yazılmalıdır**

**AHMET YESEVİ ÜNİVERSİTESİ**

**PROGRAM ADI BU BÖLÜME YAZILMALIDIR**

**2018**

# ÖZET

Yazılıma altmış yıldan fazla süredir modern toplumun günlük eğlencesinden en karmaşık iş süreçlerine kadar hayatın her alanının bir parçası olmuş durumda. Günümüzde birçok yazılım metodolijis mevcut. Bazı şirketler yazılımlarını geliştirmek için kendi özel metodolijilerini kullanırlar. Ancak söz konusu büyük çoğunluk olduğu zaman iki büyük model karşımıza çıkar: geleneksel model ve çevik model. Bu metodlardan ilki olan geleneksel model, diğer adıyla şelale metodu, yazılım dünyasında uzun yıllar kullanılmış; uzun ve detaylı planlama ve dökümantasyon gerektiren, pahalı bir metodtur.İkinci model, çevik model ise son yıllarda popülerliği artmış ve geleneksel metodun aksine uzun ve ayrıntılı dökümanlar yerine kısa iterativ döngülerle bir ürün ortaya koymayı hedefler.

Bu çalışmada öncelikle yazılım geliştirme metodolijeleriyle ilgili teoril bilgiler verilmiş, ssonrasında şelale modeli ve çevik yöntemler detaylı bir şekilde ele alınmıştır. Ayrıca bu çalışma kapsamında her iki yönetmin kıyaslaması yapılmış, birbirlerine istinaden artıları ve eksileri masaya yatırılmıştır. Detaylı araştırmalar sonucu hazırlanan bu çalışmada sonuç olarak, orta veya küçük ölçekli projelerde çevik yöntemlerin çok güzel sonuçlar verdiği çıkarımı yapılırken büyük ve çok detaylı projelerde keza çevik yöntemlerle beraber şelale yönteminin de başarılı sonuçlar doğurduğu gözlemlenmiştir.

**Anahtar Kelimeler:**

**Danışman:**

**DÖNEM PROJESİNİN İNGİLİZCE ADI BU BÖLÜME YAZILMALIDIR**

**Hazırlayanın adı soyadı bu bölüme yazılmalıdır**

**AHMET YESEVI UNIVERSITY**

**İNGLİZCE PROGRAM ADI BU BÖLÜME YAZILMALIDIR**

**2018**

# ABSTRACT

**Keywords:**

**Advisor:**

# İÇİNDEKİLER

[ETİK İLKELERE UYGUNLUK BEYANI ii](#_Toc511318601)

[ÖZET iii](#_Toc511318602)

[ABSTRACT iv](#_Toc511318603)

[İÇİNDEKİLER v](#_Toc511318604)

[ŞEKİLLER LİSTESİ vi](#_Toc511318605)

[TABLOLAR LİSTESİ vii](#_Toc511318606)

[SİMGELER VE KISALTMALAR viii](#_Toc511318607)

[BÖLÜM I GİRİŞ 1](#_Toc511318608)

[1.1. Problem 1](#_Toc511318609)

[1.2. Araştırmanın Amacı 1](#_Toc511318610)

[1.3. Araştırmanın Önemi 1](#_Toc511318611)

[1.4. Sayıltılar 1](#_Toc511318612)

[1.5. Sınırlılıklar 1](#_Toc511318613)

[1.6. Tanımlar 1](#_Toc511318614)

[BÖLÜM II KAVRAMSAL ÇERÇEVE 2](#_Toc511318615)

[2.1. Başlık 2](#_Toc511318616)

[2.2. Başlık 2](#_Toc511318617)

[2.2.1. Alt başlık 2](#_Toc511318618)

[BÖLÜM III YÖNTEM 3](#_Toc511318619)

[3.1. Araştırmanın Modeli 3](#_Toc511318620)

[3.2. Evren ve Örneklem 3](#_Toc511318621)

[3.3. Veri Toplama Araçları 3](#_Toc511318622)

[3.4. Verilerin Toplanması 3](#_Toc511318623)

[3.5. Verilerin Analizi 3](#_Toc511318624)

[BÖLÜM IV BULGULAR VE YORUM 4](#_Toc511318625)

[4.1. Birinci araştırma sorusuna(alt problem) ilişkin bulgular. 4](#_Toc511318626)

[4.2. İkinci araştırma sorusuna(alt problem) ilişkin bulgular. 4](#_Toc511318627)

[4.3. Üçüncü araştırma sorusuna(alt problem) ilişkin bulgular. 4](#_Toc511318628)

[BÖLÜM V SONUÇ, TARTIŞMA VE ÖNERİLER 5](#_Toc511318629)

[5.1. Sonuç 5](#_Toc511318630)

[5.2. Tartışma 5](#_Toc511318631)

[5.3. Öneriler 5](#_Toc511318632)

[KAYNAKÇA 6](#_Toc511318633)

[EKLER 7](#_Toc511318634)

# ŞEKİLLER LİSTESİ

# TABLOLAR LİSTESİ

# SİMGELER VE KISALTMALAR

# BÖLÜM I **GİRİŞ**

## 1.1. Problem

## 1.2. Araştırmanın Amacı

Günümüzde savnunma sistemlerinden sosyal ağlara, eğlence sitelerinden haber sitelerine kadar yazılım hayatımızın her alanında yerini almış bir kavramdır. Haber, eğlence, sosyalleşme siteleri bireylere hitap ederken, şirketler de iş süreçleri çerçevesinde yazılım. ihtiyaç duymaktadır. Yazılım endütrisi neredeyse en önemlşi endütrilerden biri haline geldi. Bu endüstri özellikle internetin ve yeni teknolijlerin gittikçe dominantlaşmasıyla dinamik, sürekli evrilen bir hal aldı. Büyük değişim, yüksek hız, değişkenlik, karmaşıklık bir çok projenin karekteristiği haline geldi. (Atkinson, Crawford, & Ward, 2006; Winter, Smith, Morris, & Cicmil, 2006). Küresel rekabetin de artmasıyla yazılımın doğru gerçekleştirildiği kadar, müşteri tatminliği, müştertiye sunum hızı da çok önemli bir hal aldı. Yazılımın kullanım amacı, karmaşıklığı, büyüklüğü ne olursa olsun her durumda bir dizi mühendislik ve proje yönetimi yapmak gerekmektedir.

Başarıya ulaşmış projeler başarılı süreçten geçmiş projelerdir. Bşr projeyi başarıya ulaştırmak için doğru yazılım kaynaklarını bulmanın yanında o projenin doğru yönetimi, doğru yazılım geliştirme metodolijisi izlemek de çok önemlidir. Doğru seçilen metodoloji doğru şekilde uygulandığında yazılım geliştirme süreçleri çok daha hızlı ve daha az bütçeyle çok daha efektif sonuçlar vermektedir. Doğru yazılım geliştirme metodolijisi seçmek ihtiyaçların karşılanması, bütçe, ve proje niteliği açısından hayati bir durum arz etmektedir. (Charvat, 2003).

Bu projede yazılım sektöründe neden birden fazla metodoloji olduğunu, bu metodolijerin neler olduğu

**Bu araştırmanın amacı yazılım geliştirme metodoliji kavramını, bu kavram kapsamında en çok kullanılan metodolojiler olan şelale modeli ve çevik modelin karakteristik özelliklerini açıklamaktır. Şu şu şu kaynaklardan yararlanıldı. Kaynaklardan özellikle bu iki modelin limitleri, kuvvetli kısımları ve**

Doğru SDM seçmek proje maliyeti ve başarısı açısından hayati önem taşımaktadır. Yanlış bir SDM seçildiğinde ise proje riski artar, proje çok daha yavaş ilerler. Her proje gereksinimlerine göre farklı bir SDM ye ihtiyaç duyuyor olabilir. Bir projeyi başarıya ulaştırmış bir SDM başka bir projeyi başarısızlığa uğratabilir. Bu “Aynı beden herkese uymaz” prensibine dayanır. (Charvat, 2003; Wysocki, 2009).

Başka bir deyişle farklı projeler farklı yaılım geliştirme metodolojilerine ihtiyaç duyarlar.

Biçok proje planlandığı bütçeyi, zamanı aşabilir veya proje sonunda ortaya çıkan ürün müşterinin ihtiyaçlarıyla örtşmüyor olabilir. Projelerin başarısızlığa uğramasında bir çok sebep vardır. Bunalrdan bir tanesi deyanlış SDm seçimidir. (Charvat, 2003; Sauser et al., 2009).

Tiwana and Keil (2004), 720 yazılım projesini inceledi ve bu projelerin sonuçlarına istinaden yaptıkalrı çıkarımlarda en büyük riskin yanlış SDM seçimi olduğunu söyledi. Yani projenenin amacı, kapsamı, büyüklüğü, karmaşıklığı, personel durumu gibi etmenlerin yanında projeye uygun bir metodolojinin seçilmesi projenenin riskini minimum seviyey indirirken müşteriye doğru ürün ortaya çıkarmakta ve bunu beklenen sürede başarrılı bir şekilde ortaya koymaktaki şansını oldukça yukarı çekmektedir.

## 1.3. Araştırmanın Önemi

Bu araştırma yazılım geliştirme metodolojisi kavramını açıklarken, en çok kullanılan SDM leri de açıklayarak birbirileri ile olan farklarını ortaya koymaktadır. Doğrı SDM seçimi bir proje başarısı için en önemli faktörlerden bir tanesidir. Araştırmada ilgili SDM ler irdelenirken hangi proje tipine hangi sdm nin neden daha uygun olduğu örneklerle açıklanmıştır. G

## 1.4. Sayıltılar

Araştırmada, bazı başlangıç noktalarının, ayrıca kanıtlanmasına gerek görülmeden, “doğru” olarak kabul edilmesi gerekebilir. Bu kabule, varsayım/sayıltı denir. Varsayım, denenmeyen bir yargıdır. Araştırma sonuçlarının geçerliği, bu yargıların doğrultusuna bağlıdır. Araştırmacı, kanıtlanması güç ya da olanaksız görünen, kişisel görüş ve inançlara göre değişebilen bazı konularda, kendi kişisel tercihin ortaya koyarak çalışmasındaki temel dayanakları belirler. Örneğin, araştırmacı “Allah vardır” deyip, tüm çalışmalarını bu temele dayalı olarak yürütebilir.

## 1.5. Sınırlılıklar

Yazılım mühendisliği insanlık tarihinde en çok araç ve metod çeşitliliğine sanip mühendisliklerden bir tanesidir.(KALIn kitap) örneğin, 2016 yılı itibariyle 60 sdm, 50 analiz aracı, 40 yazılım dizayn aracı, 37 değelrlendirme organizasyonu, 20 proje yönetim aracı, 22 test aracı, ve düzinelerce diğer araç çeşitlerine sahiptir.

Yazılım ayrıca 3000 programlama dili, 100den azı sık kullanılıyor olmasına rağmen,. Neredeyse iki haftada bir yen bir programlama dili duyruluyor.Hemen her yıl yeni bir sdm duyruluyor.

Bütün bu çeşitlilik her yazılım için standart bir yönetimin, aracın olmadığının kanıtıdır.

1966 yılından itibaren nereydese 60 tane sdm ortaya çıkmıştır. Ancak bunların arasında bilinenleri şunlardır: [**Waterfall Model:**](https://acodez.in/12-best-software-development-methodologies-pros-cons/#Waterfall_Model)şelale modeli, [**Prototype Methodology:**](https://acodez.in/12-best-software-development-methodologies-pros-cons/#Prototype_Methodology)prototip metodoliji, [**Agile Software Development Methodology:**](https://acodez.in/12-best-software-development-methodologies-pros-cons/#Agile_Software_Development_Methodology)çevik yazılım geliştirme metodolojisi, [**Rapid Application Development:**](https://acodez.in/12-best-software-development-methodologies-pros-cons/#Rapid_Application_Development)(süratli) yazılım geliştirme), [**Dynamic System Development Model Methodology:**](https://acodez.in/12-best-software-development-methodologies-pros-cons/#Dynamic_System_Development_Model_Methodology)(dinamik sistem geliştirme model metodoliji), spiral model(sarmal model), [**Extreme Programing Methodology:**](https://acodez.in/12-best-software-development-methodologies-pros-cons/#Extreme_Programing_Methodology)(uç programlama model), [**Feature Driven Development:**](https://acodez.in/12-best-software-development-methodologies-pros-cons/#Feature_Driven_Development)(özellik güdümlü geliştirme), [**Joint Application Development Methodology:**](https://acodez.in/12-best-software-development-methodologies-pros-cons/#Joint_Application_Development_Methodology)(bireşik uygulama geliştirme metodoloji), [**Lean Development Methodology:**](https://acodez.in/12-best-software-development-methodologies-pros-cons/#Lean_Development_Methodology)(Cılız Geliştirme Metodoloji), [**Rational Unified Process Methodology:**](https://acodez.in/12-best-software-development-methodologies-pros-cons/#Rational_Unified_Process_Methodology)(Rasyonel birleşik proses metodoloji), [**Scrum Development Methodology**](https://acodez.in/12-best-software-development-methodologies-pros-cons/#Scrum_Development_Methodology)(hamle metodoloji).

Bu araştırma kapsamında Bt sektöründe en çok kullanılan iki metodoloji olan

Şelale Modeli ve Çevik Yöntemleri incelenecektir. Çevik Yöntemler grubuna ait olan scrum ve kanbana değinilecektir.

## 1.6. Tanımlar

Araştırma önerisinin bu bölümünde belirsiz ya da açık olmayan ifadeler açıklığa kavuşturulmalıdır. Araştırmada fikirlerin doğru ifade edilmesinde terimlerin anlaşılır olmasının büyük faydası vardır. Bu bölüm sözlük tanımının yanı sıra araştırmacının araştırmaya özgü kullandığı tanımları da içermelidir. Araştırma raporunda sıkça tekrarlanacak bazı ifadeler kısaltılarak kullanılmalıdır. Metin içerisindeki sembol ve kısaltmalar, ayrı sayfalarda olmak şartı ile liste hâlinde ve alfabetik sıra ile verilir. Bu gibi durumlarda kısaltmaların ne anlama geldiği açıkça belirtilmelidir.

# 

# BÖLÜM II KAVRAMSAL ÇERÇEVE

**What is a project?**

**Proposed 18 Stages of Software**

Software development methodologies are created to solve problems, are put into

production, and after a period of usage, will either expand into the mainstream

or drop out of service. A few, such as Agile development, will expand rapidly and

become successful cults. Others, such as computer-aided software engineering

(CASE), will disappear after a few years due to disappointing results. Still others

will continue on with flat use for many years, while others perhaps will have declining

use, such as waterfall. As this methodology use cycle continues, new methodologies

will keep appearing at approximately 7-month intervals, as they have for

over 25 years.

Four interesting

**Methodology Development(kalın kitaptan)**

Stage 1: Identifying problems not solved by existing methodologies

Stage 2: Formulating new or modified methodologies to solve the problems

Stage 3: Trying the new methodologies on one or more projects

Stage 4: Measuring the productivity of the methodology in function points

Stage 5: Measuring the quality of the methodology with DRE

Stage 6: Announcing the new methodology to other organizations

Stage 7: Other organizations using the new methodology on selected projects

Stage 8: Identifying possible harmful side effects and negative results of the

methodology

Stage 9: Identifying possible security weaknesses of the new methodology

Stage 10: Identifying optimal sizes and types of applications where the methodology

is effective

Stage 11: Identifying sizes and types of software where the methodology is

ineffective

Stage 12: Publishing results from initial adopters

Stage 13: Additional users trying the methodology based on favorable results

Stage 14: Finding that the new methodology has problems of its own

Stage 15: Modifying the new methodology to solve observed problems if possible

Stage 16: Seeing whether the new methodology either expands or contracts

based on accumulated results

Stage 17: Publishing statistical ranges of methodology productivity and quality

Stage 18: Publishing warnings and cautions of harmful side effects or unsuitable

applications

----tablo halinde

**Table 2.1 Ten Stages of a Typical Software Methodology Life Cycle**

Stage 1: Identification of a problem not solved

by existing methodologies.

1– 3 months 1–20 people

Stage 2: Formulating a new or modified

methodology to solve the problem.

3– 6 months 1–20 people

Stage 3: Trying the new methodology on one or

more projects.

1– 4 months 5–30 people

Stage 4: Announcing the new methodology to

other organizations.

1– 3 months 1–5 people

Stage 5: Other organizations use the new

methodology on their projects.

6– 12 months 25–250 people

Stage 6: Publication of results from initial

adopters.

3– 6 months 10–30 people

Stage 7: Additional adoption of methodology

based on early adopters.

6– 18 months 500–50,000 people

Stage 8: New methodology found to have

problems of its own.

4– 6 months 10–50 people

Stage 9: New methodology modified to solve

observed problems if possible.

4– 6 months 5–20 people

Stage 10: Methodology use either expands or

contracts based on user perceptions.

12– 36 months − 250 to + 50,000 people

Total methodology life cycle: 36– 100 months (3– 8 years)

-----tablo halinde

Kaos grafikleri (neredeyse 2 sayfa)

## 2.1. y

## azılım geliştirme emtodolijisi

Proje, özgün bir ürün ortaya çıkarmak amacıyla ortaya konan geçici çaba olrak tanımlanabilir.

(PMI, 2008, p. 5). Bu tanıma göre projenin bir amacı vardır ve kendine özgü sonuç hedefini başarma eğilimindedir. (Dalcher & Brodie, 2007). Her bir proje rutin olmaktan ziyade kendine özgüdür. Her proje aslında geçici birer aktivitedir, çünkü başlangıç ve bitiş tarihi vardır ve bu her bir proje için kendine özgü, tekrarlanamayan bir eylemdir. Projeler tamamlanmak için kaynağa, örneğin insan veya bütçeye ihtiyaç duyarlar. Projeler günümüzde bir çok organizasyonun merkezinde yer alan bir aktivitedir, firmalar inovasyon ve değişim için yüklü miktarda yatırımlar yapmaktadırlar. (Sauser et al., 2009; Shenhar & Dvir, 2007).

**Br yazılım projesi** ise iki veya daha fazla kişinin sınırlı bütçe, zaman ve kaynakla var olan iş modeline bir değer katmak amacıyla üstlenilen çaba ve süreçtir. (Wysocki, 2006, p. 5)

Yazılım projeleri genellikle bir organizasyon çatısı altında yapılır ve bir program çıktısı elde etmeyi hedefler. Bu program çıktısının ilgili organizasyonun iş modeline bir değer katması beklenir.

Proje yönetimi, projenin gerekliliklerini yerine getirmek için bilginin, yeteneğin, araçların ve tekniklerin bu projeye uygulanmasıdır. (PMI, 2008, p. 6) İyi bir proje yönetimi projenin bşarısını garanti etmeyeceği gibi bir projenin kötü yönetimi çounlukla projenin başarısızlıkla sonuçlanmasına sebep olur. (Sommerville, 2006 cited in Dalcher & Brodie, 2007). Yazılım proje yönetimi bir çok zorluk barındırır. Diğer iş disiplinlerine nazaran yazılım projeleri çok daha geniş ve kestirmesi güç kapsamlara sahiptir. BT dünyasında teknoloji çok hızlı değiştiği için sürekli güncel kalmak da beraberinde ayrı bir zorluk getirmektedir. Eğer bir proje yöneticisi gücel teknolojilere aşina değilse proje yönetiminde ciddi zorluklarla karşılaşılır.

**What is a project management methodology?**

## Proje yönetimi enstitüsü (PMI, 2008, p. 438) belirli bir disiplinde kullanılan bir takım teknik, prosedür ve kurallar felsefesi olarak açıklar. Türk Dil kurumu ise metodoloji(yöntembilm) Özellikle felsefe ve bilim alanında yöntem araştırmak ve yeni yöntemler yaratmak için ilkeler geliştiren bilim, metodoloji olarak tanımlar.Metod ve metodoloji kavaramı birbiri ile karıştırılmamalıdır. Metodlar teknik ve prosedürlelerle ilgiliyken metodoloji ise aktivitelerin koordinasyonlarını adresler. Koordinasyon, işbirliği ve iletişim olmadan herhangibir proje grubu istediği kadar yetenekli olsun,ortaya güzel sonuçlar çıkaramayacaktır.

**What is a software development methodology (SDM)?**

**Yazılım geliştirme metodolojisi ise bir yazılım projesininde koordinasyon, işbirliği ve iletişim süreçlerinyle beraber yazılımın geliştirilmesi, planlanması ve kontrolünü sağlayan bir yapıdır. Her projenin karakteristiğine göre ideal farklı bir metodolojisi olabilir. Bir projede başarılı olmuş bir metodoloji bir diğer projede başarı getirecek dye bir şey söz konusu değildir. Bir yazılım geliştirme metodolojisi genelde yazılımgeliştirme felsefesi ve yazılım geliştirme sürecinde rol alacak yöntemler, araçlar ve modellerden meydana gelir.(**

**Elliot, G.** (2004). *Global Business Information Technology: An integrated systems approach.* Pearson Education. p.87.

)

## Why do we need methodologies?

Pazarın dinamik ve çok hızlı değişen bir yapıda olduğunu düşünürsek bu hızla değişen ihtiyaçlara cevap verebilmek için metodoloji çok önem arz etmektedir. Projeninbüyüklüğünden bağımsı olarak metodoloji bir projenin hayat döngüsü boyunca performansını tutarlı bir şekilde optimize etmeye çalışır(Charvat, 2003). SDm nin kullanımının yazılım projelerinde üretkenlii-ği ciddi derecede artırdığı gözlemlenmiştir. (Dybå, Moe, & Arisholm, 2005). Metodoloji ayrıca, bir yazılım projesinin yönetiminde ortak bir lügat, format ve strateji rolünü üstlenir. Bir yazılım projesinde ekip üyeleri değişebilir ancak tutarlı bir metodoloji olduğu sürece projede ris minimum seviyede olacaktır. Metodoloj kullanılmayan projelerde ise risk yüksektir.

## Tarihçe ?

Yazılım geliştirme metodolojilerinin tarihi 1960lı yıllara dayanmaktadır.

Bilgi sistemleri üretimi için en eski metolodoji yalaşımı olarak karşımıza Yazılım geliştirme yaşam döngüsü çıkar. YGYD‟nin amacı yazılım geliştirme sürecinin planlanmış, yapısal ve metotlara uygun felesefe olmasını sağlamaktır.

Mikroişlemci teknolojilerinin gelişemsiyle beraber yazılım üretkelnliği de lineer olarak artmıştır. Çok daha komplek yazılımlar üretilmeye başlanmıştır. Bu yıllardada yüryazılımların kompleksleşmesi ve projelerin büyümeiyle beraber yazılım sistemleri geliştirmesi çok daha uzun ve masraflı bir hal almaya başlamıştır. Bu problemleri çözmek için yazılımda kullanılan programlama dilleri evrimleşirken, örneğin object oriented programming ortaya çıkmış, yazılım metodolojisi kavramı da paralelde gelişmeye ve çeşitlenmeye başlamıştır.

1970 yılında Dr. Winston Royce tarafından yayınlanan bir bildiride Geleneksel Yazılım geliştirme emtodolojisi olan Şelale Modeli tanıtıldı([11] W.W.Royce, “Managing The Development of Large Software Systems”, **The Proceedings of the WESCON**, San Francisco, A.B.D, 328-339, 1970.) şelale yöntemi günümüzde kullanılan bir çok yazılım geliştirme metodolojisinin temelini oluşturur.

1970li yıllardan itibaren ortaya çıkan yazılım geliştirmeleri özetle:

1970‟lerde

• Yapısal programlama

1980‟lerde

• Yapısal sistem analizi ve tasarımı metodolojisi

Bilgi gereksinimi analizi sistemler metodolojisi

1990‟larda

• Süratli Uygulama Geliştirme

Dinamik sistemler geliştirme metodu

• Scrum

• Rasyonel Bütünleştirme Süreci

• Takım Yazılım Süreci

Uç programlama

2000‟lerde

• Uç Programlama

• Çevik bütünleşik Süreç

• disiplşne çevik servisi

**2010s**

* [Scaled Agile Framework](https://en.wikipedia.org/wiki/Scaled_Agile_Framework) (ölçekli çevik çatısı
* [Large-Scale Scrum](https://en.wikipedia.org/wiki/Large-Scale_Scrum) (büyük ölçekli scrum)

Şeklindedir.

[1] **Elliot, G.** (2004). *Global Business Information Technology: An integrated systems approach.* Pearson Education. p.87.

Büyüyen ve karmaşıklaşan büyük projeleri idame ettirebilmek için firmalar çeşitli arayışlara girmeye başlamıştır. 90lı yıllarda hafifsıklet(lightweight) yazılım geliştirme modelleri ortaya çıkmıştır. Örneğin xp, scrum ve feature-driven development. Bunlar daha sonra çevik metodlar olarak anılacaktır. The term “agile” orijini 2001 yılına dayanmaktadır. Bu tarihlerden itibaren Çevik metodlar pazarda ciddi anlamda tartışılmaya başlanmış, bir takım faydaları, proje süresinin kısalması, daha az kaynakla daha sağlıklı ürünler elde edilebilmesi, nitelikte artış gibi sebeplerle bu yöntem kendine büyük bir yer bulmuştur. Değişen market koşullarına uyum sağlama konusunda da çevik metodlar olumlu tepkiler vermiştir. (Lycett,

Macredie, Patel, and Paul 2003).

## 2.2. Yazılım geliştirme emtodoliji çeşitleri.

Genel olarak günümüz yazılım dünyasında en sık kullanılan bazı yazılım geliştirme metodolijleri açıklanmıştır.

[**Waterfall Model:**](https://acodez.in/12-best-software-development-methodologies-pros-cons/#Waterfall_Model)şelale modeli,: Şelale modeli geleneksel model olarak da bilinir. Yazılım projelerinde yer alan faaliyetlerin kesin çizgilerle ayrdılığı ve birbiri ardına geldiği bir modeldir. Her faaliyetin başlaması kendisinden önceki faaliyetin bitmesine ve dökümante edilemsine bağlıdır. Günümüzde büyük ölçekli projelerde yaygın olarak kullanılmaktadır.([9] N.M.A.Munassar, A.Govardhan, “A Comparison Between Five Models Of Software Engineering”, International Journal of Computer Science Issues - IJCSI, Vol 7, Issue 5, 94-101,2010.)

RESİM EKLENEBİLİR



(kaynak(!!AA-örnek yabanci)

*Prototyping Model: prototip modeli hedeflenen bir sistemin taklidi bir sisem ortaya koyma işlemidir. Bu aşamada şelale yönteminden de yararlanılabilir. Amaç kompleks bir projenin tamamlanması öncesinde çalışan sade bir örnek üretmektir. Yeteri kadar kaynak ayırmadan önce problem maliyetini daha sağlıklı oranlamaya yardımcı olur.*

[**Agile Software Development Methodology:**](https://acodez.in/12-best-software-development-methodologies-pros-cons/#Agile_Software_Development_Methodology)çevik yazılım geliştirme metodolojisi,

AGILE AÇIKLARKEN BURAYA DA YAZ.

[**Rapid Application Development:**](https://acodez.in/12-best-software-development-methodologies-pros-cons/#Rapid_Application_Development)(süratli) yazılım geliştirme),: geleneksel metodlarla çok hılzı ve aynı zamanda yüksek nitelikli sonuçlar elde etme amaçlı ortaya çıkmış bir metodolojidir. Güçlü bir geliştirme için maksimum avantaj sağlayacak şekilde tasarlanmıştırç Sürstli uygulama geliştirme metodolojisi görevlerin planlanmasına daha az vurgu yaparken ürün geliştirmeye daha şok vurgu yapmaktadır. PROJE geliştirilirken aynı anda birden fazla YAZILIM görevİ paralelde yerine getirilebilir. Bu merodloji birden fazla modüle ayrılabilen küçük orta veya büyük ölçekli projeler için uygundur. Ürün sahibi her modül tamamlandıktan sonra gerbildirimini yapar.( [15] J. Martin, *Rapid application*

*development*, Publisher: Macmillan

Publishing, 1991, pg. 788, ISBN:0-02-

376775-8)

**spiral model(sarmal model),:**

İteratif bir yöntem izler. Yani geleneksel metodolojinin tüm projeyi bir kere analiz tasarım geliştirme gibi aşamalarından geçirmesindense uygulamayı yine aynı aşamaları(analiz, tasarım,geliştirme) içerecek şekilde tekrarlı bir biçimde ele alır. Her iterasyon bir tasarım hedefiyle başlar, ve iterason bittiğinde müşterinin fikri alınır. Bir sonraki iterasyonda bu geribildirimler de göz önünde bulundurularak gelişen bir yapıda proje devam eder.(

[2] **Rossberg, J.** (2008). *Pro Visual Studio Team System Application Lifecycle Management.* Apress.

)

[**Extreme Programing Methodology:**](https://acodez.in/12-best-software-development-methodologies-pros-cons/#Extreme_Programing_Methodology)(uç programlama model),: Yazılım projesini yönetilebilir olacak şekilde küçük parçalara ayırır. Projeyi bütünüyle plan, analiz ve tasarım aşamalarından geçirmektense ayırdığı küçük parçalarda bu aşamaları aynı anda ele alır. Yeri geldiğinde iki yazılım geliştirici bir bilgisayara oturur ve biri kod yazarken diğeri ona görevi anlatabilir. Bir diğer parçada iki yazılım geliştiricisi arasında roller değişebilir. Yazılım geliştiriler genelde yazılımdan önce birim testleri hazırlarlar. Proje yöneticisi projedeki görevlerin önceliğinin kararını veren tek yetkili kişidir. Uç programlamada proje sahibi yazılım geliştrme ekibiyle daima beraberdir ve bu sayede direkt geribildirim yapabilir. Bu metodoloji küçük orta veya büyük ölçekli projelerde kullanılabilir.

[5] K. Beck, “Embracing change with

extreme programming”, *Computer*, vol. 32

, no.10, pg. 70 – 77, 1999, doi:

10.1109/2.796139

[**Lean Development Methodology:**](https://acodez.in/12-best-software-development-methodologies-pros-cons/#Lean_Development_Methodology)(Cılız Geliştirme Metodoloji),

Kolayca değişebilen dinamik yazılımlarda tercih edilen bir metodolojidir. Asıl amaç çok kısıtlı bütçe ve kaynakla akışı devam ettirmektir. İteratif yöntemi kullanır, ürüne faydası olmayan herhangi bir bileşeni projeye dahil etmez, müşteri odaklı yaklaşım sergiler. Devamlı geliştirme ve takım çalışması odaklıdır.

# BÖLÜM III **YÖNTEM**

## 3.1. waterfall tarihçe

1960’lı yıllarda yazılım projelerinde kullanılan yöntem kodla ve hatayı düzelt yöntemiydi. Yani kapsamlı bir sistematik yoktu. Projeler ilgili yazılım dilleriyle kodlanıyor, hata çıkması durumunda bu hata çözülüyordu. Christophe Thibuat de bahsettiği gibi “bir yıl koda gömül, diğer yıl hata ayıkla” şeklinde bir prensip hakimdi. Kodla ve hatayı düzelt yaklaşımının zorlukları nedeniyle Winston Royce 1970 yılında şelale metodolijisini ortaya çıkardı.[KAYNAK: !!aA10.1.1.464.6090] Dr. Royce tarafından sunulan şelale metodolojisi ilk önerildiğinde yenilemeli ve evrimsel süreçler ve geliştirme sırasında müşterinin katılımı gibi günümüzde çevik metodolojiyle özdeşleşmiş uygulamalara dem vurduysa da zamanla günümüzde kullanılan şeklini almıştır.( M.S.Palmquist, M.A.Lapham, S.Miller, T.Chick, I.Ozkaya, **Parallel Worlds: Agile and Waterfall Differences And Similarities**, Software Engineering Institute, Carneige Mellon University, A.B.D, 2013.)

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 [12] Şelale yöntemi zaman içinde bugün bildiğimiz şeklini aldı ve yazılım geliştirme metodolojilerinin temeli haline geldi

şelale yöntemi, iyi tanımlanmış proje safhaları arasında yapısal bir ilerlemeye vurgu yapar. Her safha kesin ve net bir şekilde tanımlanmıştır ve bu safha tamamlanmadan bir sonraki safhaya geçilemez. Safhaların isimleri frklılık gösterebilir ancak temel felsefe şudur: ilk safha sistemin ne yapacağıyla ilgilenir, bu sistemin ihtiyaçlarını belirler. İkinci safha ise bu sistemin nasıl tasarlanacağıyla ilgilenir. Üçüncü safha yazılım geliştiricilerin yazılımı gerçekleştirdiği safhadır. Dördüncü safha yazılımın testinin yapıldığı safhadır ve son safha youn şekilde sistemin dökümante edilmesi ve eğitim dosyalarının hazırlanmasıdır.

## 3.2. waterfall teknik

Geleneksel yaklaşım gereksinimlerin iyi tanımlandığı, bu gereksinimlerin proje süresince değişmediği veya çok az değiştiği, proje kapsamında yerine getirilecek görevlerin süre tahminlerinin iyi yapıldığı durumlarda oldukça başarılı sonuçlar vermektedir. Geleneksel yaklaşımda amaç proje tanımını açık bir dille yapıp, proje safhaları arsında net olmayan yerlerin bırakılmaması olmalıdır. Şelale modelinin bir takım dezavanları da mevcuttur. Bunlar:

Proje sırasında müşteri isteklerinin ani değişebilmesi, proje sonlanana kadar bir çıktı elde edilememesi, acil gereksinimler, plan güdümlü yaklaşımından noksan takım çalışanları. (Boehm & Turner, 2003)

Şelale modelini anlamak ve yazılım projelerine uygulamak oldukça basittir. Şelale modelinde her safha kendisinden sonra gelen safha başlamadan önce mutlaka itmiş olmalı ve safhalar arasında çatışma olmamalıdır. Şelale modeli yazılım geliştirme prosesinde lineer ardışık bir akış sergiler. Bunun anlamı br prosesin başlaması mutlaka bir önceki prosesin sonlanmış olmasına bağlıdır.

Şelale yaklaşımında proje brbirinden bağımsız safhalara bölünür. Her bir safhanın çıktısı bir sonraki safhanın girdisi olacak şekilde bir davranış sergiler. Bu ardışık şekilde safhalar eşliğinde ilerler.

Şelale modelinde kullanılan ardışık safhalar şunlardır:

* Gereksinimlerin elde edilmesi ve analizi: Geliştirmesi yapılacak sistemin tüm ihtimal dahilindeki gereksinimleri elde edilir ve gereksinim dökümanı olarak dökümante edilir.
* Sistem tasarımı: İlk safhada elde edilen gereksinmler incelenir ve sistemin tasarımı hazırlanır. Sistem tasarımı ilgili sistemin gereksinimleri ışığında bu sistemin tamamının mimarisi hakkında bilgi sağlar.
* Geliştirme.: Sistem tasarımında elde edilen bilgiler ışığında sistem öncelikle küçük program parçacıkları şeklinde geliştirilir. Sonrasında bu küçük program parçacıklarının her birinin birim testi(unit testing) yapılır.
* Entegrasyon ve test : Geliştirme aşamasında birim testi yapılmış küçük program parçacıkları entegre edilir ve sonunda oluşan tek sistemin testi yapılır.
* Sistemin intikali: Geliştirmesi ve testleri başarıyla tamamlanmış ürün müşterinin hizmetine sunulmak üzere canlı ortama alınır.
* Bakım: Ürün canlı ortama alındıktan yani müşterinin kullanımına sunulduktan sonra bir takım sorunlarla, hatalarla karşılaşılabilir. Bu problemlerin üstesinden geçmek için ilgili problem çözümleri de adım adım canlı ortama alınarak sistemin daha sağlıklı çalışır durumda yaşamına devam etmesi sağlanır.

## 3.3. agile tarihçe

Çevik modelinin ortaya çıkışı 1990ların ikinci yarısına dayanmaktadır. Geleneksel yöntemi

Bu metodoloji iteratif ve çoğalan bir öze sahip olmakla birlikte proje boyunca esnek, minimum ön çalşma gereksinimi, insan ilişkileri odaklı, müşteriyle interaktif çalışma, ve düzenli ürün ortaya çıkarmak gibi karakteristiğie sahiptir.2001 yılında 17 uzman Snowbird, Utah ‘ta bu prensiplere uygun ortak çevik metodolojilerini ele almak üzere bir araya geldiler. (Cockburn, 2007), ve çevik yazılım geliştirme konusyla ilgili bir strateji belirlediler. Ortak amaç projede öncelikli hedefin müşterinin taleplerinin kısa zamanda ve değişime esnek olacakbir yapıda yerine getirilmesiydi. Proje geliştirme esnasında değişimlere açık olan yapısıyla çevik model geleneksel yaklaşımla karşılaştırıldığında birçok yenilik ve farklılıklar getirmiştir.

## 3.4. agile teknik

Çevik ekip tarafından 2001yılında yayınlanan bildirgede Çevik Bildiri felsefesinin ilkeleri ve *Özellikleri :*

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

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

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

Bahsi geçen çevik Bildiri felsefesi ve ilkelerinden doğan 12 ilke vardır.

En önemli önceliğimiz  
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 yüzyüze iletişimdir.

Çalışan yazılım ilerlemenin birincil öçüsüdür.

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.

[30] Çevik Yazılım Geliştirme Manifestosu, http://agilemanifesto.org/iso/tr/manifesto.html, 15.10.2018

Çevik yazılım geliştirme hayat döngüsü 6 temel safha içermektedir. Bu safhalar:

Proje öncesi planlama: Projenin yapı taşları ve pazarözellikleri bu safhada tanımlanır.

Başlangıç safhası:

Sistemin gereksinimleri berlirlenir ve bu gereksinimlere öncelikler verilir.

Yapım: Bu safhada yazılım geliştirme yapılır. Kullanıcından gelen isteklere esnek bir yaklaşım sergilenir. İteraif bir yöntem izlenir. Yani kullanıcıdan gelen bilgilerle bir ürün oluşturulur sonra kullanıcdan gelen geribildirimlerle aynı işlemler tekrarlanarak nihai ürün elde edilmiş olur.

*Üretim: Tüm işlemler ve verilen destek safhasına verilen simdir.*

*Seçim: Yapının amacı*

*5) Seçilim: Ürün kullanıcıya sunulduktan ve canlıya alındıktan sonra da sistemin aktif ve kullanılır olmasının sağlandığı safhadır.*

*6) ayrılış:* sistemin deaktive edilmesi anlamına gelir.



Fig 2: Process of Agile Software Development

KAYNAK[!!AA,10.1.1.657]

Çevik yöntem çeşitlerine örnek olarak: Uç programlama, Scrum, kristal metodoloji, Dinamik sstem geliştirme metodu, uyarlanır yazılım geliştirme gibi örnekler verilebilir.

Tablo 3. Başlıca Çevik Yöntemlerin Uygulanma Oranları

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| (Implementation Rates of Primary Agile Methodologies) **VersionOne**  **[32]** | | **Forrester**  **[33]** | | **AgileTurkey**  **[21]** | |
| **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 |

current work method (55%).

Regarding the reasons of failure in adoption of

agile projects, the next two positions are taken

by the lack of experience in using agile

methods (41%) and lack of managerial

support (38%).

The main barrier blocking the adoption of

Agile is the ability to change the

organizational culture, followed by the

organization resistance to change (42%) and

existence of a more rigid ([14] *The 2015 State of Scrum Report*,

https://www.scrumalliance.org/socialnetworks,

2015)

**Scrum:**

Bu çevik yöntem yaklaşımıy 1986 yılında Takeuchi and Nonaka tarafından “yeni yeni ürün geliştirme oyunu” adıyla ilk defa ortaya çıkmıştır. Scrum’da iterasyonlara sprint adı verilir. Bu sprintlerin onaylanınca ilgili iterasyonlara bir ekleme yapılamaz. (Dybå & Dingsøyr, 2008). Scrum takım çalışmasına dayanan efektif bir çevik yazılım geliştirme metodolojisidir. Scrum scrum takımından meydana gelir. Proje süresince kesin bir görev ayrımı yoktur. Biresysellikten ziyade takım ön plandadır. Bir görevin ancak bir takım tarafından yerine getirlir, başarı veya başarısızlık bireylere değil takıma bağlanır. Scrum ın merkezinde sprint kavramı yatar. Bir sprint iki veya dört hafta arasında bir süreye tekabül eder ve sonunda mutlaka bir ürün beklenen bir süreçtir. Sprint sprint planlama, günlük scrum, geliştirme takımı, sprint gözden geçirmesi ve sprint geçmişinden oluşur.

Sprint planlamada o sprintte yapılacak işlerin planı yapılır. Günlük scrum toplantıları 15 dakikayla sınırlı bir aktivite olup ekibin günlük durum değerlendirmesini içeren bir toplantıdır. Sprint gözden geçirmesi sprint sonunda o sprintte planlanan işlerin nihai sonucunun ele alınmasıdır. Sprint geçmişi ise scrum ekibinin o sprintte nereleri iyi yaptıkları, neleri kötü aptıkalrı ve neleri daha iyi yapabileceklerini sadece ekip olarak ele aldıkları bir toplantıdır.

Scrum’da scrum metodolojisinin doğu işleyebilmesi için scrum master görevini üstlenen biri mevcuttur. Bu kişinin görevi günlük scrum toplantılarını yapmak, ekibin bu metodolojiye olan uyumunu sağlamaktır.

Ürün sahibi, sprint sonlarında lrtaya çıkan ürünlerle ilgilenen ve sprintlerde yapılacak işleri bildiren kısacası ürünün sahibi kişidir.

Scrumt takımı, ilgili ürünü geliştirmek için değişik alanlarda uzman kişilerden oluşan ekiptir. Ekipte analist, yazılımcı, testçi gibi keskin görev sahipleri yoktur. Ekip fazla kalabalık olmaz. Takım birbiriyle yakın çalışır ve sorunları beraber irdeler.

Scrum da katı bir şekilde yönetici kavramı yoktur. Ekip bir bütündür ve yapılacak işlerin eforlarını belirleyen, hangi işlerin alınacağını da belirleyen yine bu ekiptir. Başarıyı ve başarısızlığı ekip olarak üste alınır. Genel olarak Scrum çevik yazılım geliştirme yaklaşımı aşağıdaki şemaya özetlenebilir.

## 3.4.1.1 kanban

*Kanban sinyal veren alet anlamına gelmektedir. Yani ben daha fazla işe hazırım mesajı taşımaktadır.*

kalın kitap, sayfa 244) Kanban ın temel konsepti tüm işi tanımlanmış segment ve durumlara ayırıp duvarda her birini bir kolon olarak göstermektir. Her eleman bir karta yazılır ve bu kart ilgili sehment kolonunun altına konularak genel iş akışı takip edişmiş olur. Her bir sütunda aynı anda kaç işin yapılıyor olduğu takip edilir ve belli bir limiti geçmesi istenmez. Her bir elemanın süresi optimum tutulmaya çalışılır.

Kanban ın faydaları:

Uygulama geliştirme esnasında dar boğazlar açıkça görünür olur, bu sebeple ekibin ayrı ayrı işler yapmasındansa bu darboğaza odaklanıp projenin önünün açılması sağlanabilir.

Projede çok çeşitli faktör ve kesin olmayan durum çokluğu durumlarında efektif bir çalışma şekli sağlar

Organizasyondaki envanter sayısını çok fazla artırmaz, dökümantasyon bolluğuna sebep olmaz.

SEGMENTLER, DURUMLAR GÖRSEL OLARAK organize edildiği için gerekli gereçler, bekleme zamanlarında düzelme, ürün geliştirme hızında artış ve devam eden görevlerin iş akışları efektif bir şekilde gözlemlenebilir.

Envanter ve doküman bolluğuna sebebiyet vermediği için kaynak ve aynı zamanda zaman tasarrufu sağlar.

KANBAN BOARD RESMİ konulabilir.

# BÖLÜM IV BULGULAR VE YORUM

## 4.1. waterfall +ları -leri

Şelale modelinin temeli bütün projeyi bölümlere ayırarak kontrol etmektir. Bu bölümlerin tanımları iyi yapılmış olduğundan adım adım ilerleyerek proje stabil bir şekilde ilerler. Geliştirme sırasıyla konsept belirleme, analiz,tasarım, geliştirme, test ve bakım safhalarından geçer ve bu safhaların sırası ve tanımı nettir.

Şelale modeli anlaması ve uygulaması basit bir metodolojidir.

Her safha kendine özgü belirgin prensiplere sahip olduğundan safhaların birbirine karışması söz konusu değildir.

Safhalar sadece bir kere işleme alınır.

Gereksinimleri net olan küçük orta veya büyük projelerde oldukça başarılı sonuçlar elde edilir.

Projenin kilometre taşlarını anlaşılabilir kılar.

Projede yapılacak görevleri düzenlemek kolaydır.

Kimin hangi görevi ne kadar sürede yapacağüı net olduğu için, performansı inişli çıkışlı olan ekipler için kestirilebilir sonuçlar alınmasını sağlar

Şelale modeli istikrarlı ve disiplinli bir yöntem olduğundan projede her şeyin kontrol altında olduğu güveni ve hissiyatını yaşatır.

Takım içerisinde görev paylaşımı net olduğu için proje elemenalarının işlerinin çakışması gerçekleşmez.

Kapsamlı doküman çıktıları olduğu için bütün ekip elemanlarının aynı ortamda bulunmadığı projelerde yüzyüze iletişim ihtiyacını azaltır.

16] S.Balaji, M.S.Murugaiyan, “Waterfall vs. V-Model vs Agile: A Comparative Study on SDLC”, International Journal of Information Technology and Business Management, Vol.2, No.1, 26-30, 2012

[13] A.Farrell, **Selecting a Software Development Methodology Based On Organizational Characteristics**, Yüksek Lisans Tezi, Athabasca University, School of Computing and Information Systems, 2007.

[9] N.M.A.Munassar, A.Govardhan, “A Comparison Between Five Models Of Software Engineering”, International Journal of Computer Science Issues - IJCSI, Vol 7, Issue 5, 94-101,2010.

Dökümantasyona oldukça önem verdiği için sisteme dışardan yeni adapte olacak kişilerin adaptasyonu hızlı olabilir.

## Waterfall Model - Disadvantages

Şelale yönteminin en bariz dezavantajı proje safhalarının keskin bir şekilde birbirinden ayrılmış olması sebebiyle bir hata durumunda bir önceki safhaya dönüp müdahale etme imkanının çok zor olmasıdır.

The major disadvantages of the Waterfall Model are as follows −

* Proje sonlanıncaya kadar elde herhangi bir ürün olmaz.
* Net olmayan, nesneye yönelik ve kompleks projelerde ciddi risk söz konusudur
* Uzun süredir devam eden projeler için zengin bir model değildir.
* Proje geliştirmesi esnasında ihtiyaçları sürekli değişen bir yapışa asla sağlıklı değildir.
* Değişen ihtiyaçlara adapte olamaz
* Proje devame derken bir safhada yapılan bir değişiklik tüm projeyi olumsuz etkileyip projeyi başarısızlığa uğratabilir.
* Dökümantasyona fazlasıyla önem verdiği için bu durum bazen zaman sıkıntısı ortaya çıkarabilmektedir.

(9,13,17 M.A.Awad, **A Comparison Between Agile and Traditional Software Development Methodologies**, The University of Western Australia, Avustralya, 2005.

## 4.2. agile +ları

KÜçük, orta veya büyük ölçekli projelerde kullanılabilir.

Yazılım ihtiyaçlarının doğası gereği bir değişim olduğunda bu değişime ayak uydurabilir.

Takım çalışmasına önem verir, ekibin her üyesinden maksimum verim alabilir.

Çalışan bir yazılım için projenenin bitmesi beklenmez, çok daha erken safhalarda çalışan bir yazılım üretilmiş olur

Dökümantasyonu dikte etmez, onun yerine yüzyüze iletişimle daha hızlı aksiyon alınmasını teşvik eder. Dökümantasyonla kaybedilen zamandan tasarruf sağlar

Planlamayı istenerek yapılan bir süreç haline getirir ve süreci basit ve esnek kılar.

Kullanıcı ile yazılım geliştiriciler sürekli iletişim halinde olduklarından müşterinin ihtiyaçları daha net anlaşılabilir ve eğri bildirimlere alınan aksiyon süresi kısalır. Neticede müşterinin istediği ürün elde edilmiş olur.

**7.** Best for projects that deal with services oriented and non physical deliverables like code, copywriting and design projects.

[KAYNAK:IJSETR…

,agile –ler

Kapsamlı bir başlangıç planlaması olmadığından tüm proje boyunca topyekün bir planlama olmamasının eksikliğini yaşatır.

Efektif bir şekilde uygulanabilmesi için iyi bir çevik lider, proje yönetim tecrübesi ve iyi bir yönetim sistemi gerekir.

Kısa sürelerde çıktı üretme alışkanlığı edinene kadar zorlukalr yaşanır, katı zaman diliminde üretilen çıktılar karmaşıklığa yol açabilir.

Müşteri katılımı şarttır. Müşteri ve yazılım geliştirici işbirliği sağlanamazsa iş süreçleri başarızlığa uğrayabilir.

Dökümantasyona önem vermediği için takıma yeni biri katıldığında projeye adaptasyonu zor olabilir.

.

## 4.3. waterfall ve agile farklar

|  |  |  |  |
| --- | --- | --- | --- |
| heavyweight methodologies is shown in the table below. **Agile Methods** | | **Heavy Methods** | |
| **Approach** | Adaptive | | Predictive |
| **Success Measurement** | Business Value | | Conformation to plan |
| **Project size** | Small | | Large |
| **Management Style** | Decentralized | | Autocratic |
| **Perspective to Change** | Change Adaptability | | Change Sustainability |
| **Culture** | Leadership-Collaboration | | Command-Control |
| **Documentation** | Low | | Heavy |
| **Emphasis** | People-Oriented | | Process-Oriented |
| **Cycles** | Numerous | | Limited |
| **Domain** | Unpredictable/Exploratory | | Predictable |
| **Upfront Planning** | Minimal | | Comprehensive |
| **Return on Investment** | Early in Project | | End of Project |
| **Team Size** | Small/Creative | | Large |

(!!AA10.1.1.464.6090)



**Figure 2-6: Balancing optimizing and adapting dimensions (Adapted from Highsmith, 2002, p. 360)**

. Tablo 4. Uygulanan Metodolojiye Göre Yazılım Projeleri Başarı Oranları [8]

|  |  |  |  |
| --- | --- | --- | --- |
| (Software Projects Success Rates According to the Methodology Implemented) **Şelale Modeli** | | **Çevik Yöntemler** | |
| **Başarılı Projeler** | % 11 | | % 39 |
| **Sorunlu Projeler** | % 60 | | % 52 |
| **Başarısız Projeler** | % 29 | | % 9 |

Tablo 5. Yazılım Projeleri Başarı Oranlarının Proje Büyüklüğüne Göre Dağılımı [8]

|  |  |  |  |
| --- | --- | --- | --- |
| (The Resolution of Software Projects Success Rates by Project Size) **Proje Büyüklüğü** | **Başarılı** | **Sorunlu** | **Başarısız** |
| **Büyük** | % 8 | % 25 | % 41 |
| **Orta** | % 30 | % 59 | % 48 |
| **Küçük** | % 62 | % 16 | % 11 |
| **Toplam** | % 100 | % 100 | % 100 |

Tablo 6. Proje Büyüklüğüne ve Uygulanan Metodolojiye Göre Yazılım Projeleri Başarı Oranları [8]

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (Software Projects Success Rates according to Methodology and Project Size) **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 | |

[8] Standish Group International, **CHAOS Report**, A.B.D, 2015.

.

.

.

.

# BÖLÜM V SONUÇ, TARTIŞMA VE ÖNERİLER

## 5.1. Sonuç

Agile orta küçük ölçekli projeler. Waterfall büyük projeler.

5.2. Türkiye de durum

## 5.3. Öneriler

Şu şu kaynakalrın irdelenip şu şu eylemlerin yapılıp bilerek anyarak uygula.

# KAYNAKÇA

# EKLER