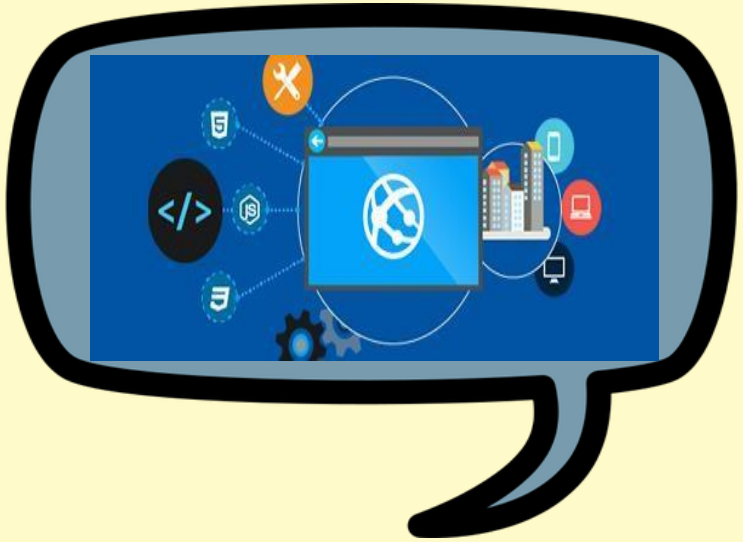
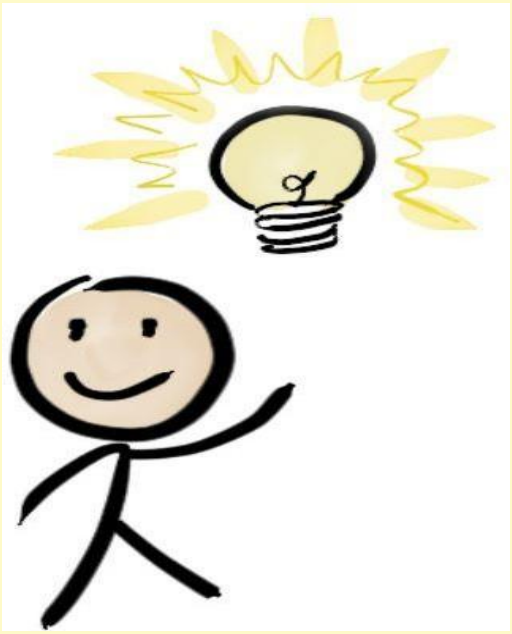
ÇEVİK YAZILIM GELİŞTİRME

Mehmet Salih ÖNDER
18360859028

Günümüzde Yazılım Projelerini Durumu

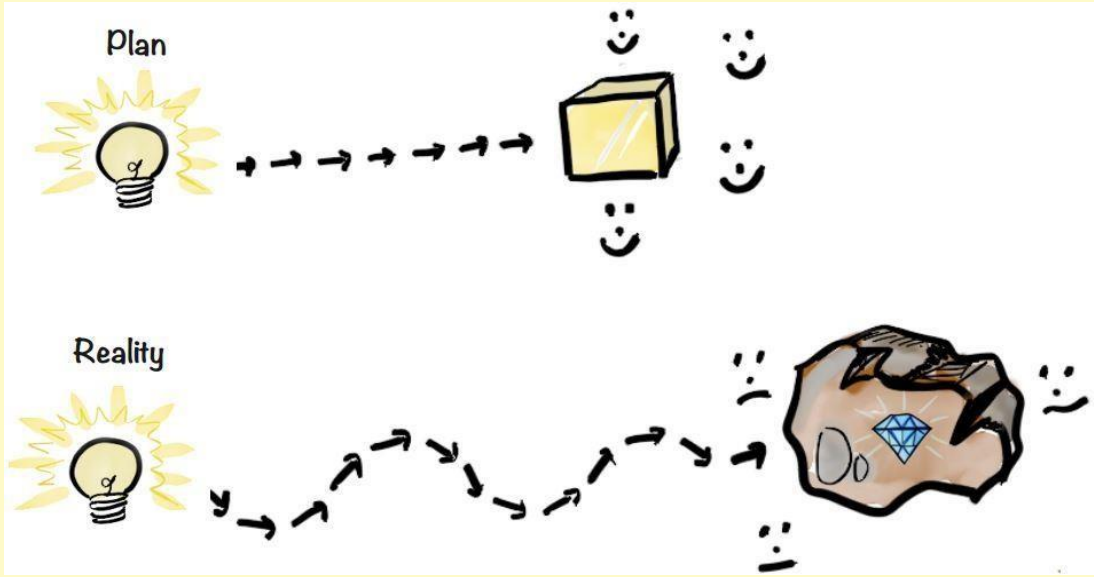
- Birçok harika proje fikirle başlar.





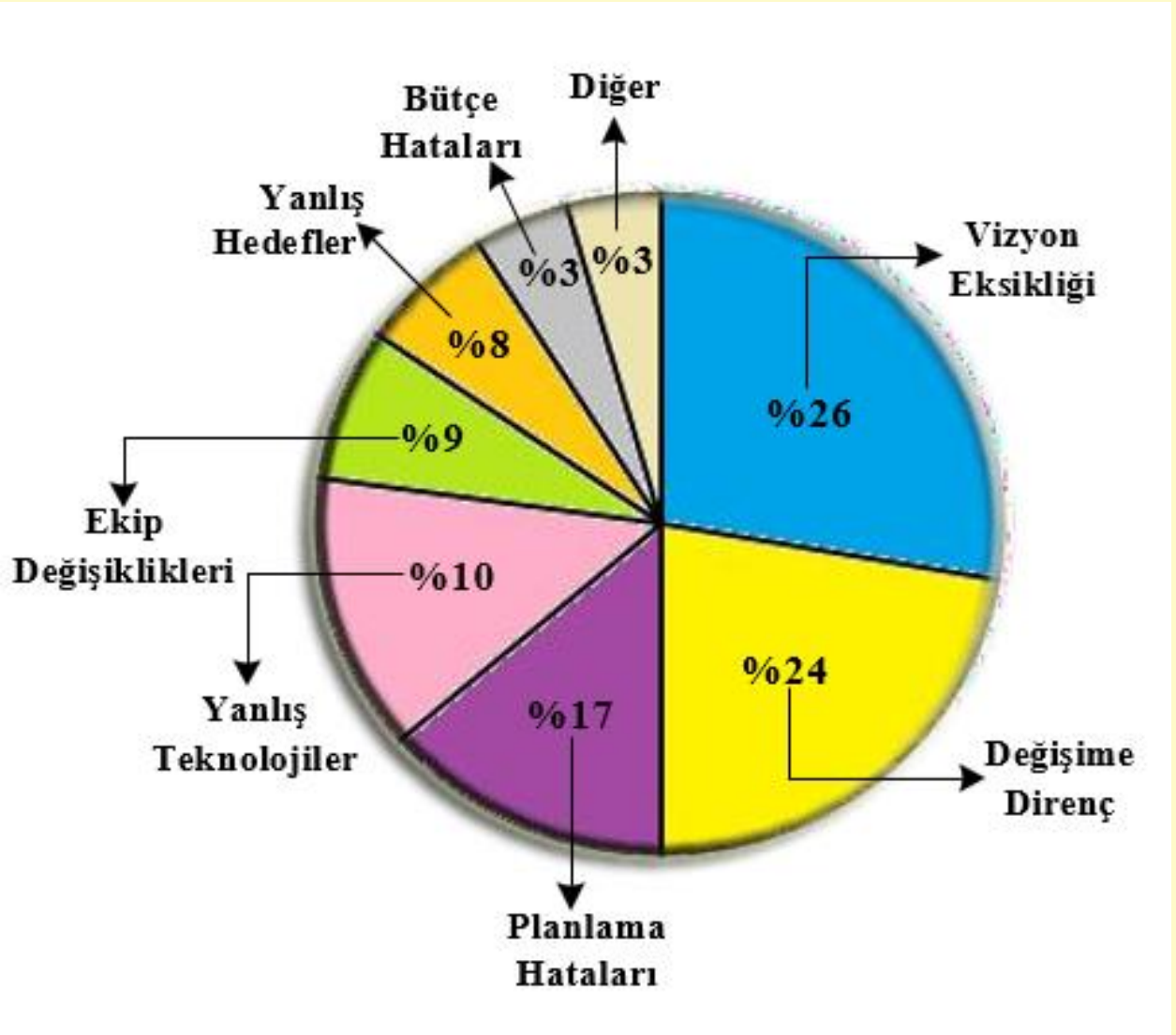
Günümüzde Yazılım Projelerini Durumu

- Ancak maalesef ki bu projelerin büyük bir kısmının başarısız olması muhtemeldir.



Günümüzde Yazılım Projelerinin Durumu

- BAŞARISIZLIĞIN ANA SEBEPLERİ
- Müşteri isteklerini doğru analiz
- Proje için uygun ekibin kurulması
- Yanlış teknoloji ve mimari seçimi
- Müşteriyle iletişimden kaçınmak



Yazılım projelerinin temel adımları aşağıdaki gibidir:

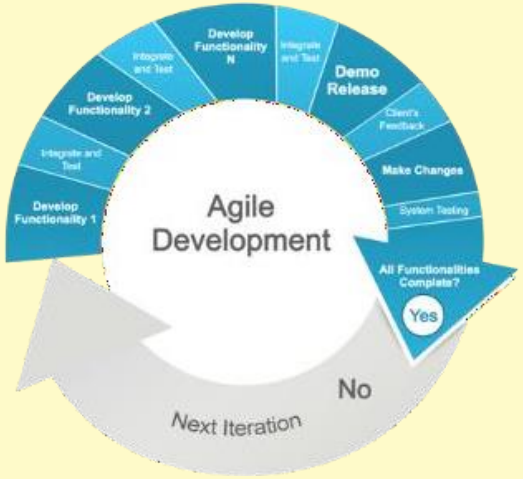
- Planlama: Projeyi sipariş eden veya projenin hedeflediği kullanıcıların istekleri ve ihtiyaçları dikkate alınarak proje ihtiyaçları tespit edilir. Proje boyunca takip edilecek olan yol, projenin maliyeti, iş bölümünün nasıl olacağı ve proje adımlarının zamana nasıl yayılacağı gibi detaylar değerlendirilir.
- Analiz: Geliştirilecek olan üründen tam olarak ne istendiği net bir şekilde kağıda dökülür. Proje somutlaşmaya başlamıştır.
- Tasarım: Yazılım sisteminin temel yapısı bu aşamada oluşturulur. Yazılım ürününün mantıksal ve fiziksel tasarımı üzerine çalışmalar yapılır.
- Geliştirme: Temelleri oluşturulmuş olan ürünün tam olarak kodlanması, kurulması ve test edilmesi bu aşamada gerçekleştirilir.
- Teslim ve Bakım: Test aşamalarının tamamlanması; ürünün hazır olduğu ve müşteriye teslim edilebileceği anlamına gelir. Müşterinin yazılımı aktif olarak kullanması ve kullanım sonucu ihtiyaç duyulan bakımların yapılması bu aşamada gerçekleşir.

Çevik Yazılım Nedir ?

- Atik yazılım geliştirme ya da çevik yazılım geliştirme, basit prensiplere dayanan yazılım geliştirme metotları gruplarının genel adıdır.
- Tekarlamalı ve artırımsal bir ürün geliştirme yöntemidir.
- 1974 yılında “Uyumlu Sistemler İçin Yazılım Geliştirme” başlıklı bir makale ile tanıtılmıştır. 1990’lı yıllar ise Şelale modelinin hantal bir sistem olduğu iddia edilerek onun yerine daha hızlı ve çevik yazılım geliştirme metodolojileri sunmaya çalışılan yıllar olarak geçmiştir. 2001 yılının Şubat ayında yazılım dünyasının önde gelen 17 ismi Utah’da bir araya gelerek “Çevik Yazılım Geliştirme Manifestosu” ve “Çevik Yazılımın Prensipleri”ni yayınlamışlardır.

Manifesto

- Manifesto, daha iyi bir yazılım geliştirmenin yöntemlerini açıklayan 4 ana madde ve 12 temel ilkeden oluşmaktadır.
- Ana Maddeler
- Bireyler ve etkileşimler, süreçler ve araçlardan;
- çalışan yazılım, kapsamlı dokümantasyondan;
- müşteri ile işbirliği, kontrat görüşmesinden;
- değişikliklere yanıt vermek, bir planı takip etmekten önce gelir.



Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

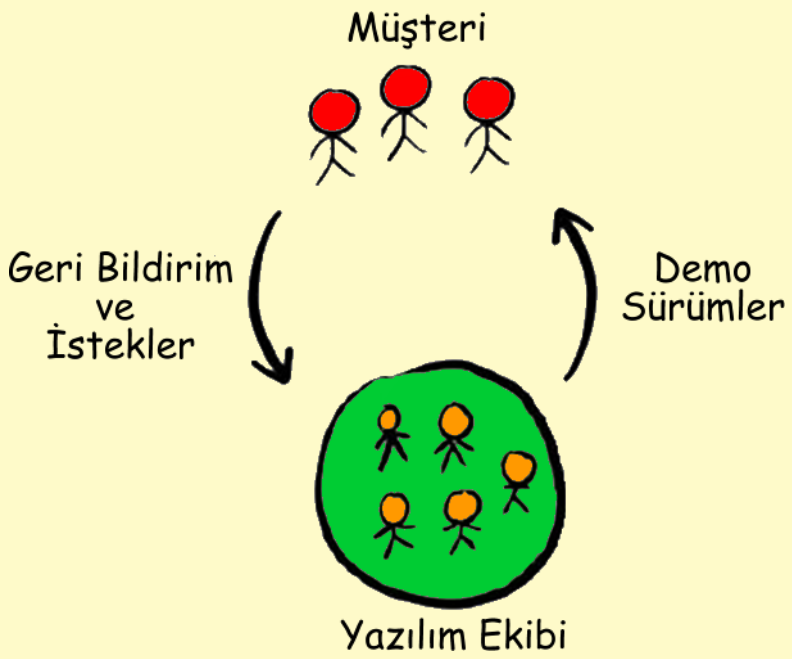
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

https://agilemanifesto.org/

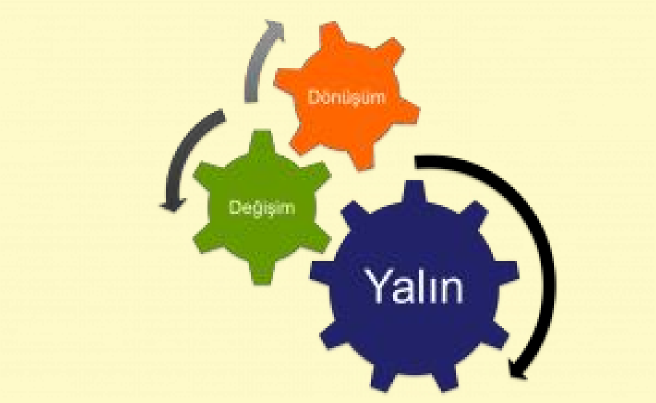
Çevik Yazılım Yöntemi



- Çevik yazılım metodu, kısa vadeli planlar ve küçük parçalar halinde yazılımın geliştirilmesini ön görür. Yazılımın geliştirilmesindeki geri dönüş (feedback) ve değişikliklere uyum sağlamak son derece önemlidir. Her yapılan yineleme yazılımı hedeflenen adıma bir adım daha yakınlaştırır. İstenilen sonuca ulaşmak adına birden çok yineleme gereklidir.

Çevik Yazılım Yöntemi

- Temel prensipler:
- Müşteriyi memnun etmek
- Değişen ihtiyaçları karşılamak
- Sık aralıklarla ürün teslimi yapmak
- Yüz yüze iletişime önem vermek
- Sürdürülebilir gelişmeyi desteklemek
- Teknik mükemmeliyete, iyi dizayna ve sadeliğe odaklanmak



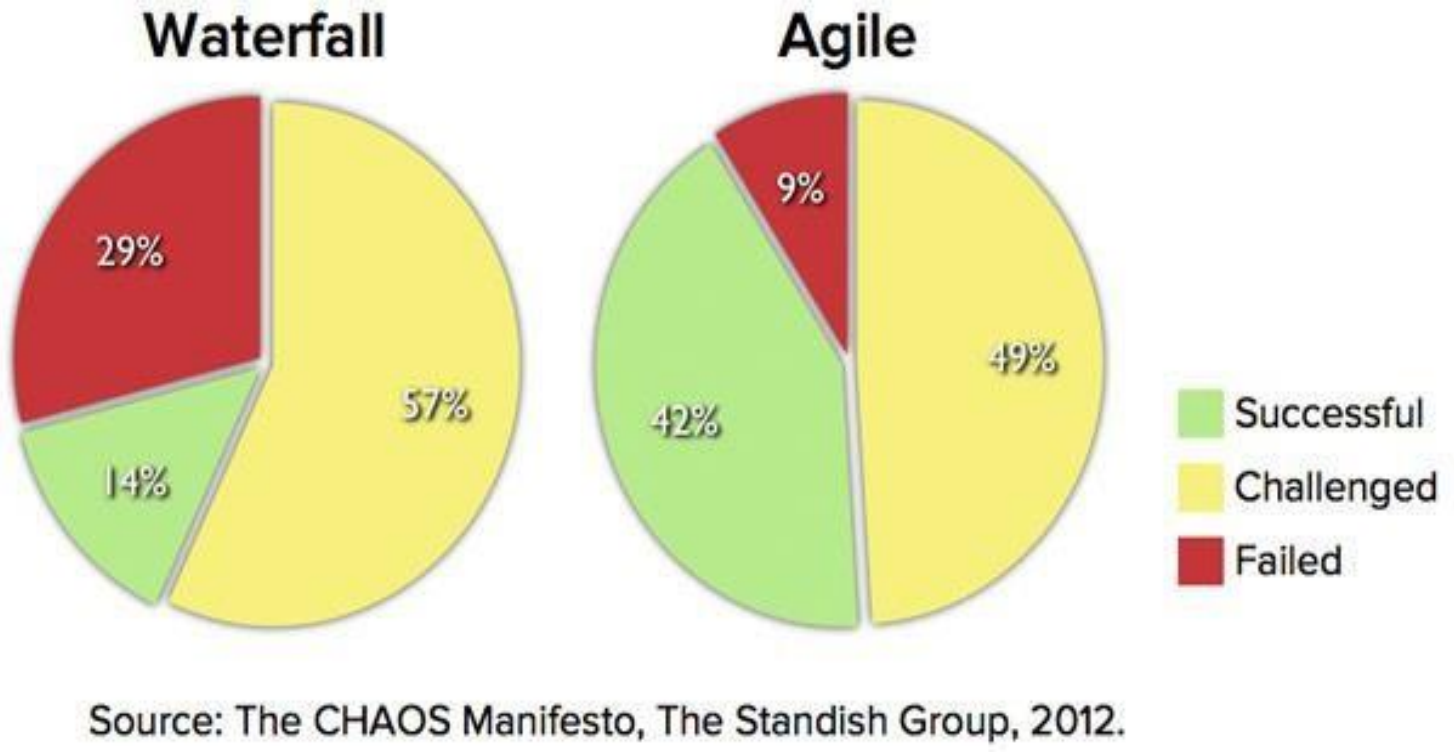
Geleneksel Yöntemler vs. Agile

- =Geleneksel Yöntemler
- Müşteriler ne istediğini bilir.
- Geliştiriciler neyi , ne şekilde üreteceklerini iyi bilir.
- Plan , yol boyunca değişmeyecektir.

Geleneksel Model vs. Agile

- =Çevik Yöntem
- Müşteriler ne istediğini keşfeder.
- Geliştiriciler neyi nasıl üreteceğini keşfeder.
- Bu yol boyunca bir çok değişiklik yapılabilir.

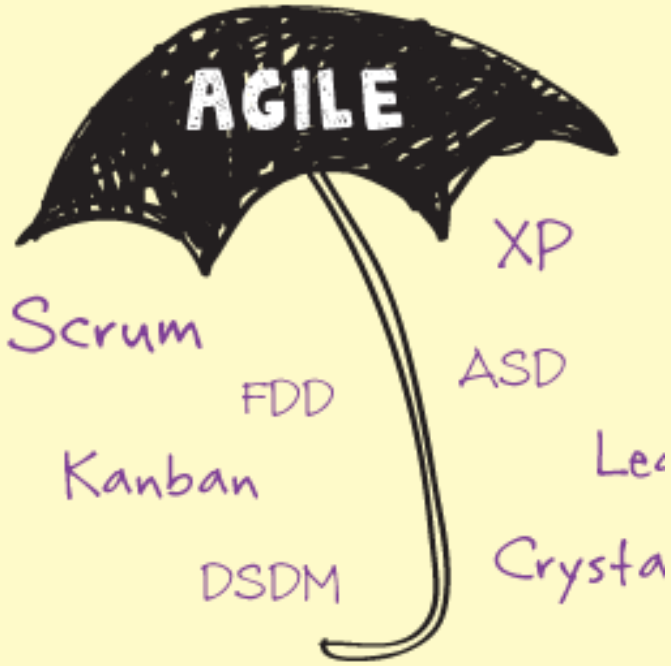
Geleneksel Model vs. Agile



Dezavantajlar

- Çevik süreçlerin dezavantajları
- Uygun olmayan ekiple (eğitimsiz, ekip çalışmasına uygun olmayan, işini kişiselleştiren, kişilik problemleri olanlar) çevik süreç yönetimi çalışamaz.
- Kalabalık ekip veya büyük ölçekli projeler için uygun görülmemektedir. Ekip ve projeler parçalara ayrılırsa, iyi bir yönetim ile başarmak mümkündür.
- Bir dış denetleyicinin dahil olduğu ve ayrıntılı kuralların gerektiği denetlemelerin zorunlu olduğu projelerde yetersiz kalmaktadır.
- Çevik çalışmak disiplinsizlik olarak yorumlanmamalıdır.

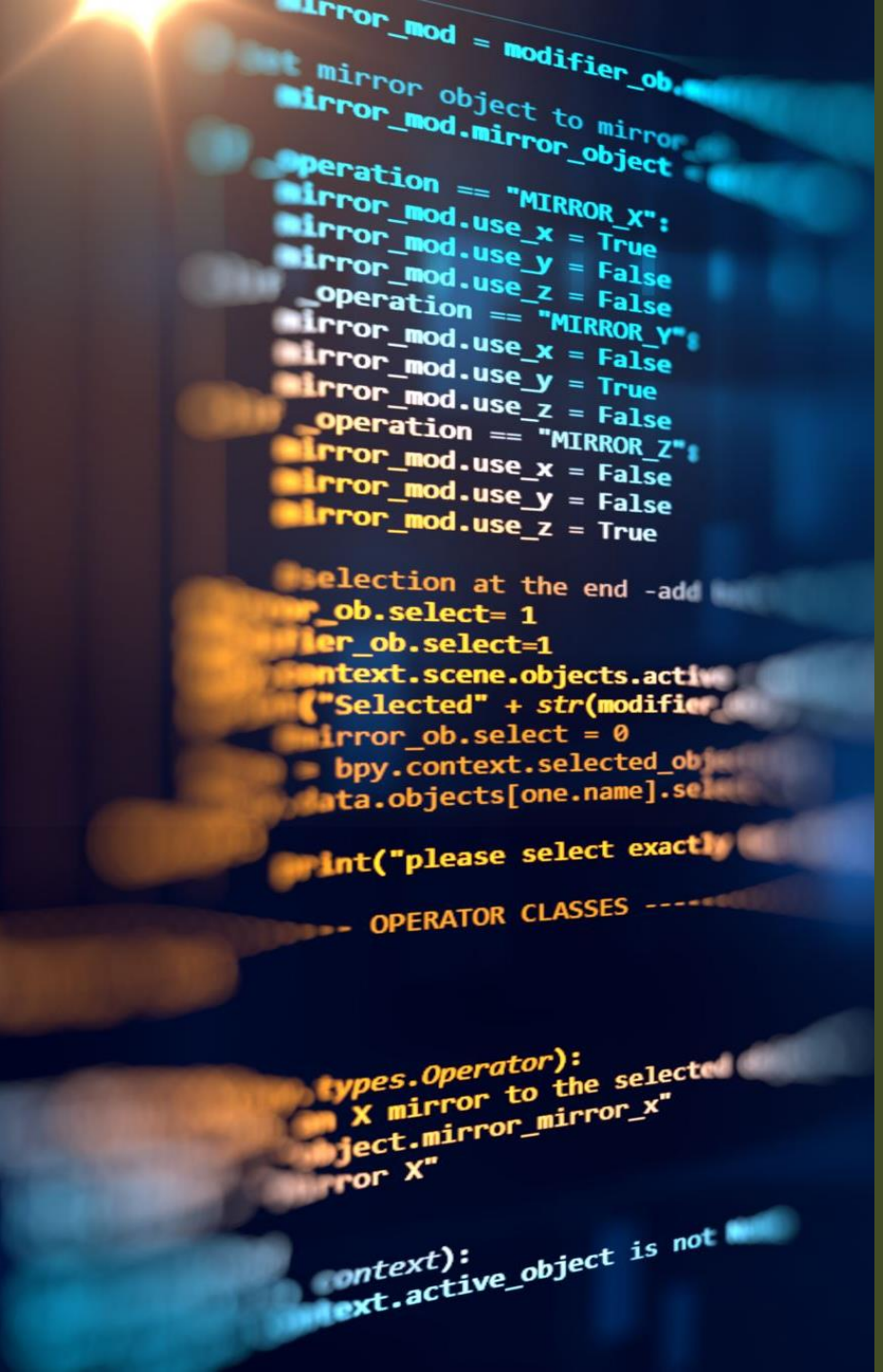
Çevik Yazılım Türleri



- XP (EXTREME PROGRAMMİNG)
- SCRUM
- FEATURE-DRİVE DEVELOPMENT(FDD)
- DYNAMİC SYSTEM DEVELOPMENT(DSDM)
- ADAPTİVE SOFTWARE DEVELOPMENT(ASD)
- ....

Scrum

- Çevik yazılım yöntemi kendi içerisinde özü aynı, fakat süreçlerinde farklılaşan alt kollara ayrılmaktadır Bu popüler kollardan bir tanesi de SCRUM’ dır. SCRUM, kelime olarak Rugby oyununda oluşturulan küçük ekiplere verilen isimdir
- SCRUM, gereksinimleri açıkça belli olmayan, değişime açık, karmaşık yazılım projelerin yönetimi için uygulanması en ideal yöntemdir. İhtiyaca yönelik, esnek, bir geliştirme sürecidir. Düzenli geri bildirim ve kısa dönemli planlamalarla hedefe ulaşmayı sağlıyor. En önemli özelliği ise müşteri ve geliştirici doğrudan temas halinde olmadıkları için yazılım geliştirici baskı altında olmaz, yapması gereken işe yoğunlaşır. Başarısı kanıtlanmış bir yazılım geliştirme yöntemi olan Microsoft, Google, Facebook gibi şirketler tarafından kullanılmaktadır.



Uç Programlama (Extreme Programming XP) Nedir?

- **Uç Programlama (XP)**, yazılım geliştirme süreci boyunca son derece *kaliteli olmak koşuluyla* **çalıştırılabilir kod** üretmeye odaklanmış bir yazılım geliştirme metodolojisidir. Yazılım geliştirme sürecinin en temel, en önemli ve final çıktısı ya da ürünü çalıştırılabilir kod olduğundan, XP metodolojisi sürecin en başından itibaren çalıştırılabilir kodu sürecin merkezinde tutmaktadır.
- **Uç Programlama'nın özünde aşağıdaki uygulamalar yer alır:**
- Müşteriyle yakın iletişim
- Planlama
- Sık ve küçük sürümler
- Basit tasarım:
- Önce test
- Refactor etme

Refactor Nedir ?

- Refactoring, kodun işlevselliğini değiştirmeden, kodun kalitesinin artırılması sürecidir. Evet refactoring bir süreçtir. Her ne kadar amaç/niyet her zaman en baştan temiz kod yazmak olsa da, (her ne kadar yazarken en temiz şekilde yazılsa da) değişen ve gelişen kodların zaman içerisinde sürekli optimize edilmesi gerekecektir.

Refactoring Nasıl Yapılır ?

Refactoring yaparken, yapılacak olanların bir sıralamaya göre yapılması önemlidir. Ayrıca diğer bir önemli konu ise bu işi belirli bir ritimle yapmak önemlidir. Yani sırası ile yapılacak olan her küçük değişiklikten sonra test yapmak gerekir. Küçük değişikler, test, küçük değişiklikler, test, ... gibi bir ritimle bu işi yapmak en basit ve garantili yoldur. Refactoring'in tanımında da bahsedildiği gibi, amacımız kodun işlevselliğinde hiçbir değişiklik yapmadan, kodların kalitesini en optimum seviyeye getirmek.Kodun işlevselliğinin değişmediğini garanti etmenin en iyi yolu da elbette test yazmaktır.

Kaynaklar

- https://akademiksunum.com/index.jsp?modul=document&folder=18cae50892480f745e89a5be097ad74b5cf05d84
- https://tr.wikipedia.org/wiki/Atik_yaz%C4%B1l%C4%B1m_geli%C5%9Ftirme

DİNLEDİĞİNİZ İÇİN TEŞEKKÜRLER...