BURSA TEKNİK ÜNİVERSİTESİ

MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

SEMİNER DERSİ PROJESİ

Çevik(Atik) Yazılım Geliştirme / Agile Software Development

MEHMET SALİH ÖNDER

2021-2022 BAHAR DÖNEMİ

ÖNSÖZ

Büyük ve karmaşık sistemler geliştiren organizasyonların ürün geliştirme süreçlerini tanımlayıp, sürekli iyileştirmesi gerekmektedir. Kalitesiz ürün geliştiren organizasyonlar itibar, müşteri ve gelir kaybı riski ile karşılaşabilir ve hatta organizasyonun geleceğini tehlikeye atabilirler. Ancak bu modeller sayesinde planlı çalışmak ve yazılım projesini en iyi şekilde gerçekleştirmek mümkün olacaktır. Çevik Yazılım Geliştirme ise basit prensiplere dayanan yazılım geliştirme metotları gruplarının genel adıdır. 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. Getirdiği esneklik sayesinde bir çok dev teknoloji firması tarafından da kullanılmaktadır.

Mehmet Salih Önder

Bursa 2022

İÇİNDEKİLER

İçindekiler Tablosu

ÖZET

Bu doküman içerisinde günümüz yazılım dünyasında adını sıkça duyduğumuz ya da duyacağımız Atik Yazılım Geliştirme tanıtılmıştır. Öncelikle geleneksel yöntemler ile kıyaslama yapılıp ardından Atik Yazılım Geliştirme hakkında detaylı bilgi verilmiştir.

Konunun daha iyi kavranması açısından Atik Yazılım Geliştirme metotlarından Scrum ve XP Programming hakkında ayrıntıya girilmiştir Scrum kritik olduğunu düşündüğüm Scrum'da pozisyonlar ve XP Programming için ise kritik olan Refactoring hakkında bilgi verilmiştir.

1.GENEL BİLGİLER

1.1 Yazılım Geliştirme Nedir ?

Yazılım geliştirme, yazılımın ardışık safhalar yoluyla düzenli bir şekilde geliştirilmesi işlemidir. Bu süreç sadece kodun asıl yazarlığını değil, aynı zamanda gereksinimlerin ve hedeflerin hazırlanmasını, neyin kodlanacağının tasarımı ve geliştirilenlerin amaçları sağladığının doğrulanmasını da içerir.

Sistem geliştirme yöntemleri ortaya çıkmadan önce, yeni sistemlerin veya ürünlerin geliştirilmesi genellikle yönetim ve teknik personelin deneyimini ve sezgisini kullanarak gerçekleştirildi. Bununla birlikte, modern sistemlerin ve bilgisayar ürünlerinin karmaşıklığı, uzun zaman önce ihtiyacı bazı tür düzenli gelişme süreçleri için netleştirdi. [1]

<u>Yazılım Projeleri Geliştirirken İzlenmesi Gereken Adımlar</u>

- 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.

Şekil 1: Yazılım Geliştirme Temsili Resim

1.2 Geleneksel Yöntemler Nedir?

Şelale yönteminde yazılım geliştirme süreci analiz, tasarım, kodlama, test, sürüm ve bakım gibi safhalardan oluşur. Geleneksel yazılım metotlarında bu safhalar şelale modelinde olduğu gibi lineer olarak işler. Her safha, başlangıç noktasında bir önceki safhanın ürettiklerini bulur. Kendi bünyesindeki değişikler doğrultusunda teslim aldıklarını bir sonraki safhanın kullanabileceği şekilde değiştirir. Geleneksel Yöntemlerde dokümantasyon ağırlıklı, kolay kontrol edilebilir, yönetilebilir süreç gibi artıları vardır. Büyük karmaşık problemlerde, gerekliliklerin önceden belli olduğu ve değişimin çok az olacağı projelerde kullanılır.

- **Avantajları**

- Kullanımı ve anlaması basittir
- Yönetimi kolaydır
- Projenin safhaları ayrı olduğundan iş bölümü ve iş planı projenin en başında net bir şekilde bellidir. Bu durum projenin yönetimini de oldukça kolay hale getirir.
- Şelale Modeli çok küçük ve gereksinimleri çok iyi anlaşılmış projelerde iyi çalışır.

- **Dezavantajları**

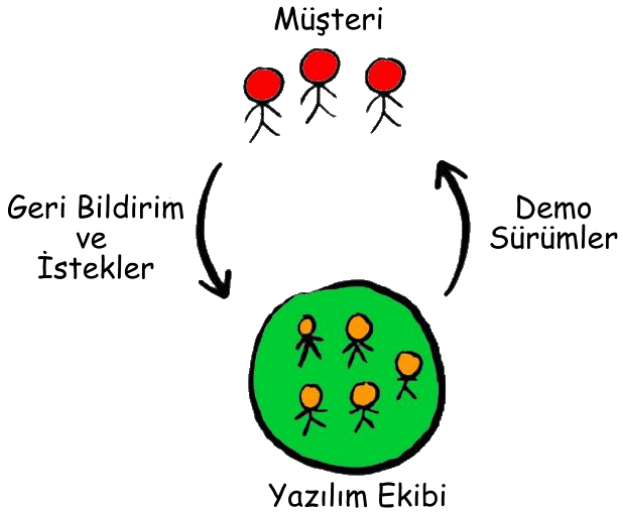
- Karmaşık ve nesne yönelimli projeler için uygun değildir
- Devam eden ve uzun projeler için zayıftır
- Projede oluşabilecek her türlü değişime elverişsiz, katı bir modeldir. Yapılan her değişiklik maliyeti büyük oranda arttırır.
- Müşteri memnuniyetini sağlamak çok zordur çünkü gelişim ve değişime açık bir model değildir.
- Model safhalardan oluştuğu için ürün son safhada tamamlanır, gereksinimlerin iyi tanımlanmadığı müşterinin ne istediğinin anlaşılmadığı bir projede bu durum projenin bittikten sonra iptal edilmesine ve başka gerginliklere sebep olmaktadır.[3]

2.ATİK YAZILIM GELİŞTİRME

2.1 Atik Yazılım Geliştirme 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.

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.



Şekil 2: Atik Yazılım Geliştirme

Ana Maddeler

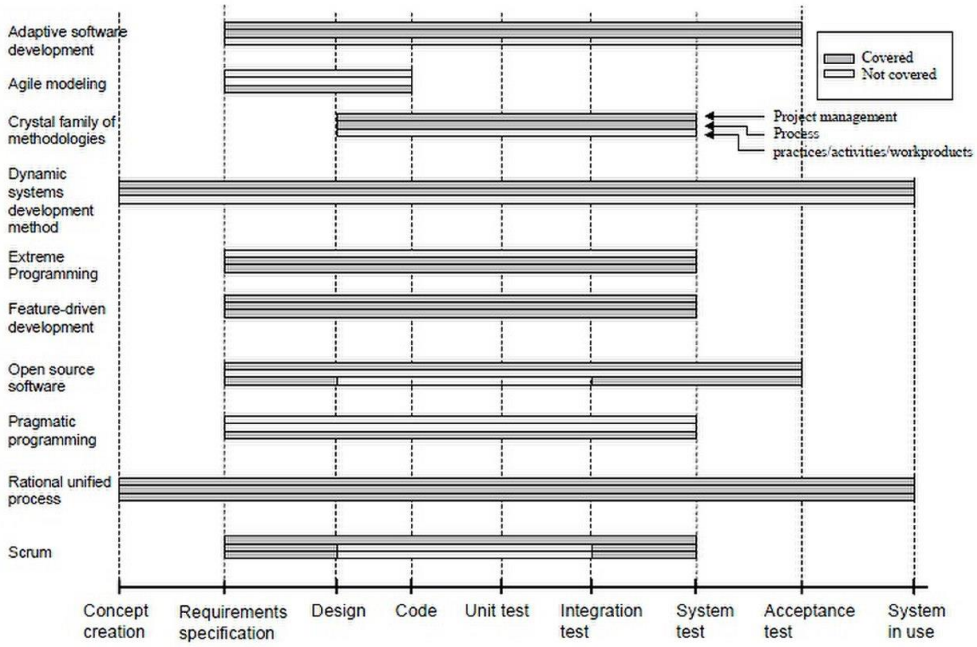
- Bireyler ve etkileşimler, süreçler ve araçlardan;
- çalışan yazılım, kapsamlı dokümantasyondan;
- müşteri ile işbirliği, kontrat görüşmesinden;

- değişikliklere yanıt vermek, bir planı takip etmekten önce gelir.

Çevik Yazılım Geliştirme Temel İlkeleri

- Değerli yazılımların erken ve sürekli teslimi ile müşteri memnuniyeti.
- Geç geliştirme aşamasında bile değişen gereksinimleri memnuniyetle karşılayın.
- Çalışan yazılımı sık sık teslim edin (aylar yerine haftalar)
- İş adamları ve geliştiriciler arasında yakın, günlük işbirliği
- Projeler, güvenilmesi gereken motive olmuş bireyler etrafında inşa edilir.
- Yüz yüze görüşme en iyi iletişim şeklidir (birlikte konum)
- Çalışan yazılım, ilerlemenin birincil ölçüsüdür
- Sürdürülebilir kalkınma, sabit bir tempoyu koruyabilme
- Teknik mükemmelliğe ve iyi tasarıma sürekli dikkat
- Sadelik – yapılmayan iş miktarını en üst düzeye çıkarma sanatı – esastır
- En iyi mimariler, gereksinimler ve tasarımlar kendi kendini organize eden ekiplerden ortaya çıkar
- Ekip düzenli olarak nasıl daha etkili olunacağını düşünür ve buna göre kendini ayarlar.

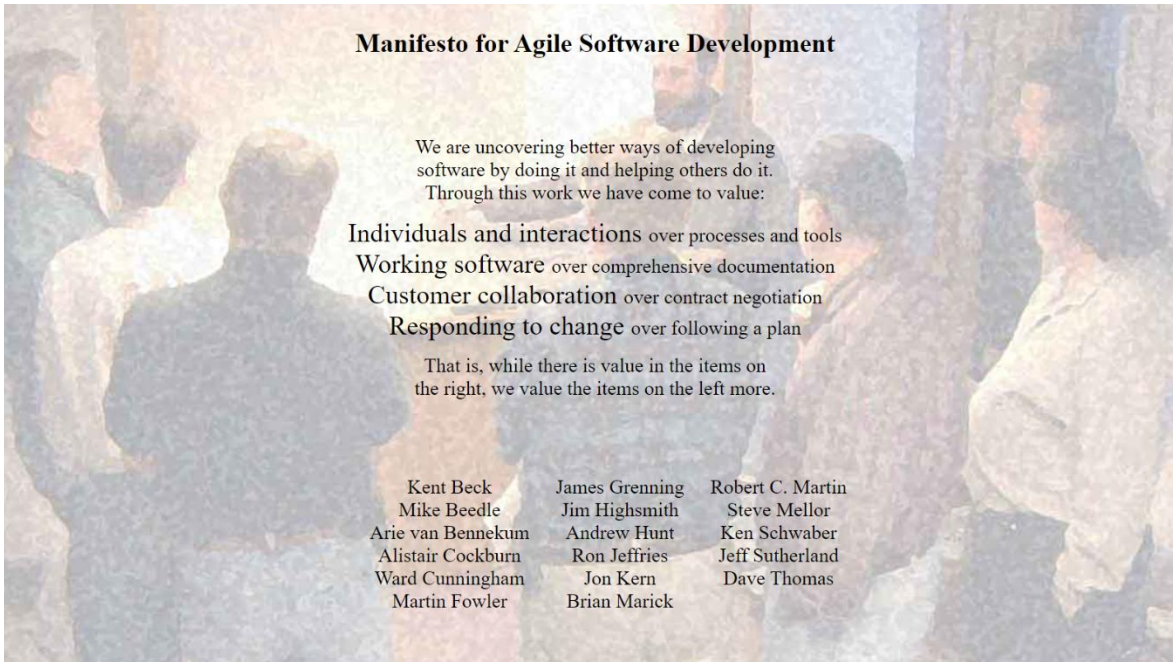
Çevik yazılım geliştirme yöntemleri, yazılım geliştirme yaşam döngüsünün geniş bir yelpazesini destekler. Bazı yöntemler uygulamalara odaklanırken (örneğin, XP, pragmatik programlama, çevik modelleme), bazıları ise iş akışını yönetmeye odaklanır (örneğin, Scrum, Kanban). Gereksinimlerin belirlenmesi ve geliştirilmesi için bazı destek faaliyetleri (örneğin, FDD), bazıları ise tam geliştirme yaşam döngüsünü kapsamaya çalışır (örneğin, DSDM, RUP).



Sekil 3: Yazılım Geliştirme Yaşam Döngüsü Desteği

2.2 Çevik Yazılım Geliştirmenin Tarihçesi

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, daha iyi bir yazılım geliştirmenin yöntemlerini açıklayan 4 ana madde ve 12 temel ilkeden oluşmaktadır.



Şekil 4: Manifestonun orijinal sitesi ekran görüntüsü

İlgili sitenin linki : https://agilemanifesto.org/iso/tr/manifesto.html

Sitede birden fazla dil desteği mevcuttur.

2.3 Çevik Yazılım Geliştirmenin Felsefesi

Geleneksel yazılım mühendisliği ile karşılaştırıldığında, çevik yazılım geliştirme temel olarak dinamik, deterministik olmayan ve doğrusal olmayan özelliklere sahip karmaşık sistemleri ve ürün geliştirmeyi hedefler. Doğru tahminler, istikrarlı planlar ve tahminler genellikle erken aşamalarda elde etmek zordur ve bunlara olan güvenin düşük olması muhtemeldir. Çevik uygulayıcılar, herhangi bir değer kanıtı elde edilmeden önce ihtiyaç duyulan inanç sıçramasını azaltmaya çalışacaklardır. Gereksinimler ve tasarım ortaya çıkacak şekilde tutulur. Büyük ön spesifikasyonlar, bu gibi durumlarda muhtemelen çok fazla israfa neden olur, yani ekonomik olarak sağlam değildir.

Şekil 5: Çevik Brezilya 2014 Konferansı

2.4 Cevik Yazılım Geliştirmenin Avantaj ve Dezavatantajları

Agile metodu, önde gelen yazılım uzmanlarının gerçek yaşam projelerindeki deneyimlerinden ortaya çıkmıştır. Bu nedenle, geleneksel kalkınmanın zorlukları ve sınırlamaları ortadan kaldırılmıştır. Daha sonra agile metodu, endüstri tarafından proje geliştirmeye daha iyi bir çözüm olarak kabul edilmiştir. Hemen hemen her yazılım geliştirici, agile metodunu bir biçimde kullanmıştır. Bu yöntem yardımcı ekipler için hafif bir çerçeve sunar. Hızlı teslimat üzerinde çalışmalarına ve odaklanmalarına yardımcı olur. Bu odaklanma yetenekli kuruluşlara yazılım geliştirme ile ilgili genel riskleri azaltma konusunda yardım sağlar.

Agile metodu, bu değerin geliştirme süreci boyunca optimize edilmesini sağlar. Yinelemeli planlama ve geri bildirim kullanımı, bir müşterinin istenen ihtiyaçlarını yansıtan bir ürünü sürekli olarak düzenleyebilen ekiplerde sonuçlanır. Bir projenin durumunu ölçerek ve değerlendirerek süreç boyunca değişen gereksinimlere kolayca adapte olur. Ölçme ve değerlendirme, her projenin gelişimine doğru ve erken görünürlük sağlar.

Agile metodunun şirketlerin doğru ürünü oluşturmalarına yardımcı olduğu söylenebilir. Yazılı olmadan önce yazılımı pazarlamaya çalışmak yerine, agile metodu, geliştirme sırasında sürümleri optimize etmek için ekipleri yetkilendirir. Bu, ürünün pazarda mümkün olduğunca rekabetçi olmasını sağlar. Kritik pazarın düzeyini korur ve bir ekibin çalışmasının geride kalmamasını sağlar. Bu nedenle agile metodu hem kullanıcılar hem de geliştiriciler için cazip bir seçenektir. Agile metodunun birçok eleştirisi vardır; ancak, bu yöntem müşterilerin memnun olacakları sonuçlar verir.

Agile metodu ile ilgili eleştiriler:

• Kullanıcı merkezli yerine geliştirici merkezlidir.

• Agile, gereksinim alma ve kod geliştirme süreçlerine odaklanır ve ürün tasarımına odaklanmaz.

• Agile metodolojiler büyük kuruluşlarda ve belirli türdeki projelerde yetersiz olabilir.n uyarlanabilir, yinelemeli ve evrimsel gelişim lehine şekillenmesine yardımcı oldu.[5]

2.5 Çevik ve Geleneksel Yöntemler Arasındaki Temel Farklar

Temel Varsayımlar:

Geleneksel: Sistemler tamamen tanımlanabilir, tahmin edilebilir, titiz ve kapsamlı bir planlama ile inşa edilebilir.

Agile (Çevik): Hızlı geri bildirim sağlanarak değişime dayalı sürekli tasarım geliştirilir. Test etme ilkeleri kullanılarak küçük ekipler tarafından yüksek kaliteli, uyarlanabilir yazılım geliştirilebilir.

Kontrol:

Geleneksel: Süreç merkezli

Agile: İnsan merkezli

Yönetim stili:

Geleneksel: Komuta Kontrol

Agile: Liderlik ve İşbirliği

Bilgi yönetimi:

Geleneksel:Açık

Agile: Üstü kapalı

Rol ataması:

Geleneksel: Birey – uzmanlaşmayı tercih eder.

Agile: Kendini organize eden ekipler – rol değişimini teşvik eder

İletişim:

Geleneksel: Resmi

Agile: Resmi değil

Müşterinin rolü:

Geleneksel: Önemli

Agile: Kritik

Proje döngüsü:

Geleneksel: Görevler veya faaliyetler rehberliğindedir.

Agile: Ürün özelliklerine göre yönlendirilir.

Geliştirme modeli:

Geleneksel: Yaşam döngüsü modeli (Şelale, Spiral veya bazı varyasyonlar)

Agile: Evrimsel teslim modeli

İstenilen organizasyonel form / yapı:

Geleneksel: Mekanik

Agile: Organik

Teknoloji:

Geleneksel: Kısıtlama yok

Agile: Nesneye yönelik teknolojiyi destekler.

Özetle;

Şekil 6 : Güdümlü füze ve Çevik Yöntem benzetmesi

=>Çevik Yöntem

Yanda Çevik Yazılım Geliştirmenin sembolik fotoğrafi mevcuttur.

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

Şekil 7 : Planlı top atışı ve Geleneksel Yöntem benzetmesi

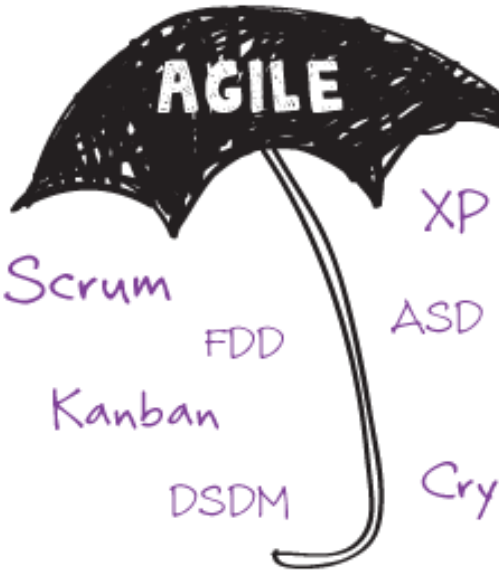
=> Geleneksel Yöntemler

Bu fotoğrafta ise Geleneksel Yöntemin temsili bir fotoğrafını görmekteyiz.

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

2.6 Çevik Yazılım Geliştirme Türleri

Çevik Yazılım Geliştirme bünyesinde bir çok farklı metot barındırmaktadır. En çok bilinen ve kullanılan metot olanlardan Scrum ve Xp Programlama' yı açıklamış olacağım. Aşadaki listede daha adı olmayan yöntemlerde mevcuttur.



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

Şekil 10 : Çevik Yöntem alt modelleri

2.6.1 Scrum Nedir ?

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 Scrum , Microsoft, Google, Facebook gibi şirketler tarafından kullanılmaktadır.

Scrum 'da geliştiricilerin baskı altında olmadığından bahsetmiştim. Bu durum farklı pozisyonlar sayesinde gerçeklişiyor.

Scrum' da olan bazı pozisyonlar;

Product Owner -> Bu pozisyonda olan kimse müşteriyi temsil eder. Teknik altyapıya sahip olabilir.

Dev Team -> Developer team ise söz konusu uygulamayı geliştircek olan yazılım ekibidir.

Scrum Master -> Scrum Master Product Owner ile Dev Team arasındaki bir köprü görevi görmektedir. Bu pozisyon için eğitim görüp sertifika alınabilmektedir. Product Owner ve Dev Team ile sürekli irtibat halindedir. Dev Team üzerine oluşabilecek baskıyı kırabilir.

Şekil 11: Scrum Master Temsili Resmi

2.6.2 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 Prensipleri

- Rapid Feedback (Hızlı geri bildirim)
  Sık ve hızlı geri bildirim edinmek, projenin gidişatını olumlu etkiler. Geri bildirim sayesinde yanlış anlaşılmalar ve hatalar ortadan kaldırılır.

- Assume Simplicity (Basitliği tercih etmek)
  Basit çözümler kolay implemente edilir ve kısa zamanda oluşturulur. Bu geri bildirimin de hızlı bir şekilde gerçekleşmesini sağlar. Basit çözümlerin kavranması ve anlatılması daha kolaydır.

- Assume Simplicity (Basitliği tercih etmek)
  Basit çözümler kolay implemente edilir ve kısa zamanda oluşturulur. Bu geri bildirimin de hızlı bir şekilde gerçekleşmesini sağlar. Basit çözümlerin kavranması ve anlatılması daha kolaydır.

- Play to win (Kazanmak için oyna)

XP takımları kazanmak için oynar. Her zaman gözlerinin önünde nihai sonuç vardır: programı tamamlamak ve müşteriye teslim etmek.

- Quality Work (Kaliteli iş)
- XP projelerinde kaliteli işin ortaya konabileceği bir ortamın oluşturulması gerekmektedir. Hiçbir programcı hatalı program yazmak istemez. Çalışma ortamınında etkisiyle yüksek kalitede yazılım yapmak hem programcının öz güvenini artırır hem de müşteriyi tatmin edici ürünlerin ortaya konmasını sağlar.

Uç Programlama 'nın özünde aşağıdaki uygulamalar yer alır:

- Müşteriyle yakın iletişim
- Planlama
- Sık ve küçük sürümler
- Basit tasarım:
- Önce test
- Refactor etme

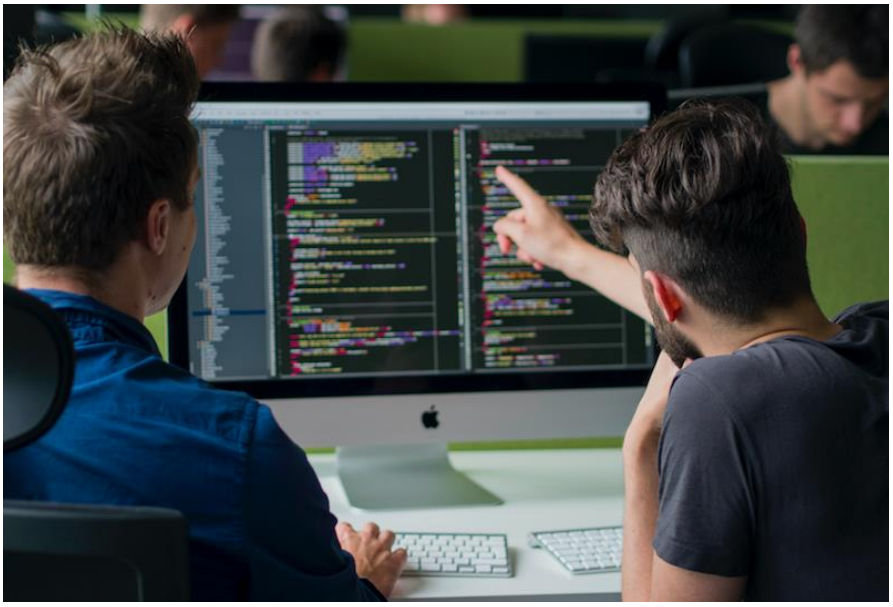
Refactor Nedir ?

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.

2.6.3 Pair Programming (Çift Programlama)



Şekil 12: Çift olarak programlama yapan ekip

Kelime anlamı olarak eşli kodlama anlamına gelmektedir. Türkiye de pekte uygulanmasa bile gayet başarılı bir geliştirme imkanı sunan bir çalışma şeklidir. Bir kişi bilgisayar başında kod yazarken diğer kişi onu izleyerek gerekli yerlerde yönlendirir. Buna şu şekilde bakmak gerekiyor. 2 pilotun bir uçağı kullanması gibi düşünülebilir. Birisi daldığı ya da hata yaptığı anda diğer pilot duruma müdahile ederek durumu kurtarır.

Avantajları Nelerdir?

1-Bir geliştirici hep aynı mantıkla düşünür fakat ikinci geliştirici olaya farklı bir açıdan bakarak diğer geliştiricinin gözünden kaçan bir yolu görmesini sağlar.

2-CodeReview, 2 geliştirici aynı kod üzerine odaklandığı zaman otomatik olarak birbirlerini düzelterek ilerlerler ve ortaya daha temiz bir kod çıkar.

3-Bir geliştirici bazen çok basit bir iş için bile saatlerde düşünmek zorunda kalır ve artık bir noktada üretkenlik durur ve ikinci göze ihtiyaç duyar bu noktada 2 kişi çalışmak faydalıdır.

4-Geliştiriciler birbirlerine yeni özellikler kazandırır.

5-Ekibe yeni katılan birisinin yapıya daha hızlı adapte olmasını sağlanır.

6-Oluşan hatayı daha hızlı bulunmasını sağlar.

Dezavantajları Nelerdir ?

1- Kaynakların yetersiz kullanılması olarak görülebilir.

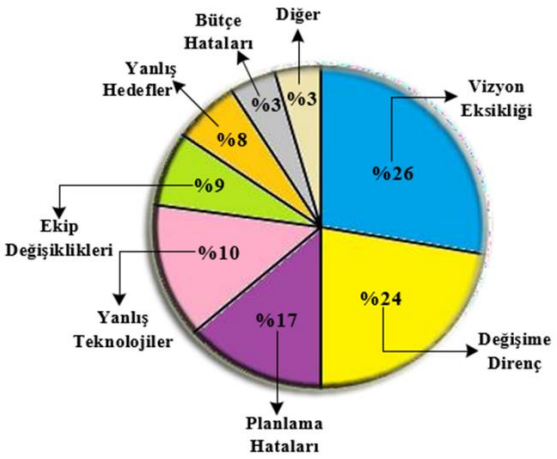
2- Maliyetli bir yöntemdir.

3. BAŞARI ORANLARI

3.1 Çevik Yazılım Geliştirme Başarı Oranı

Genel anlamda tüm projeleri incelersek başarısızlığın sebeplerini söyle sıralayabiliriz.

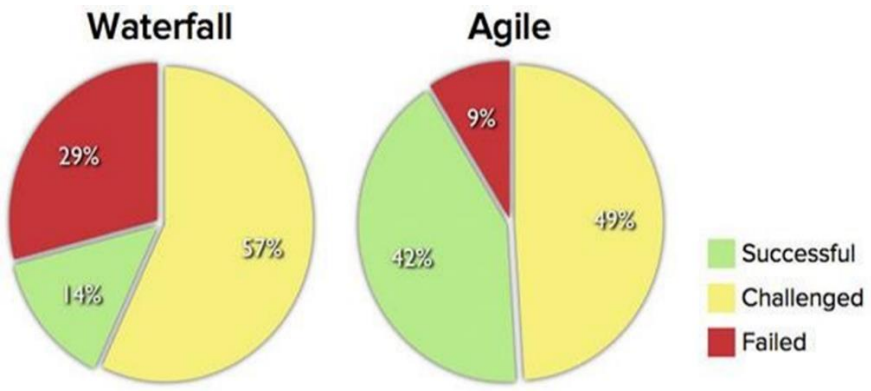
- Müşteri isteklerini doğru analiz
- Proje için uygun ekibin kurulması
- Yanlış teknoloji ve mimari seçimi
- Müşteriyle iletişimden kaçınmak



Şekil 8 : Başarısızlığın sebepleri

Şekil 5'de görebileceğimiz üzere genel anlamda projelerde %24'lük dilimde Değişime Direnç problemi var. Çevik Yazılım Geliştirmenin bize sunduğu esneklik sayesinde projeler başarısızlıkla sonuçlanmıyor.

Elimizde bu konuda yapılan 2 araştırma mevcuttur. İlki 2012 yılında yapılan bir araştırma diğeri ise 2015 yılında yapılan bir araştırmadır. 2012 yılındaki araştırmada proje büyüklüğüne bakılmaksızın Şelale ve Çevik Yazılım Geliştirme Metodları karşılaştırılmış. Şekilden de anlayabileceğimiz ölçüde Çevik Yazılım Geliştirme proje başarı oranı olarak ciddi oranda bir üstünlük kurmuştur. Bu araştırma sonrası da Çevik Yöntem daha çok tercih edilmiştir.



Source: The CHAOS Manifesto, The Standish Group, 2012.

Şekil 9: 2012 yılında yapılan bir araştırma

2015 yılında yine aynı kurum tarafından yapılan araştırma proje büyüklüğü ve proje başarı oranları olarak elimizde somut veri sunmaktadır. Bu veri eşliğinde büyük projelerde Çevik Yazılımın proje başarı oranı küçük ölçekli projelere göre büyük düşüş yaşamıştır.

CHAOS RESOLUTION BY AGILE VERSUS WATERFALL

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	29%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

The resolution of all software projects from FY2011–2015 within the new CHAOS database, segmented by the agile process and waterfall method. The total number of software projects is over 10,000.

UYGULAMA

Özel sektörde bulunan orta ölçekli firmalardan birinin üretim bandını takip eden yazılım ihtiyaçları vardır. Bu sebepten ötürü bir yazılım firmasıyla anlaşılmıştır. Yazılım firmasının ilgili ekibi müşteri ile iletişime geçmiştir. Bu ekip bir kaç randevudan sonra müşteri firmanın düşündüğü yazılımı saptamıştır. Gereksinimler belirlenmiştir. Henüz yazılımın projesinin başında yazılım ekibibir kaç yerde çelişki farketmiştir.

Müşteri firma ile görüşen ekiple irtibata geçerek bu duruma netlik kazanmak istemişlerdir. İlgili ekip müşteri ile tekrar irtibata geçerek bu durumu çözmeye çalışmışlardır. Gün sonunda müşteri firmanın nasıl bir yazılım istediği henüz kendileri dahi farkında değildir.

Yazılım firması bu gelişme sonucu Geleneksel yöntemlerden Şelale metodu yerine Agile yöntemlerden biri olan Scrum kulanılmaya karar verilmiştir. Scrum'da olması gereken pozisyonlara çalışanlar atanıştır. Bu pozisyonlar Scrum Master(Product Owner ile Yazılım ekibi arasında köprü görevi görür.) ve Product Owner(Müşteri firmayı temsil eden ve kısmı de olsa teknik bilgiye sahip birisi olmalıdır.)'dır. Ayrıca bu adımdan sonra yazılım ekibi sürümler halinde istenen yazılımı ulaşmak istemektedir. Yapılan ilk sürüm müşteri firmaya ulaştırılmış ve sürüm hakkında geri dönüt alınmıştır. Bu dönütler sonucu yeni versiyon hazırlıkları tamamlanmıştır. Yazılan yeni versiyon tekrar müşteriye ulaştırılmıştır. Müşteri firma, ihtiyaçları olan yazılıma çok yakın bir proje olduğunu farkeder. Ufak değişikliler yapılması amaçlı tekrar Product Owner üzerinden Scrum ekibiyle iletişime geçer.

Yazılım ekibi son gereksinimleri göz önüne alarak son sürümü çıkartır. Bu sürüm müşteri firmaya iletilir. Müşteri firma aldıkları hizmet karşısında memnun kalmıştır. Daha sonra ortaya çıkmıştır ki müşteri firmanın bu yazılımı elde etmek için daha önce 3-4 yazılım firmasıyla görüşülmüş ve firmaların geleneksel yöntemler kullandıkları için değişimlere ayak uydurulamayacağı düşünülerek bu yazılım firmalarıyla çalışılmasına sıcak bakılmamıştır.

SONUÇLAR

Geleneksel Yazılım Geliştirme metodu daha çok savunma sanayisinde kullanılmaktadır. Dokümantasyonun çok çok kritik olduğu bazı sektörlerde de kullanılmaya devam etmektedir.

Çevik Yazılım Geliştirme metodu ise bir çok sektörde kullanılmaktadır. Yazılım ekibi , ürün ve müşteri üçleminde müşteri ihtiyacına dinamik şekilde karşılık vermeyi hedefleyen Çevik Yazılım Geliştirme metodu müşterinin tam olarak ne istediğini bilmediği projelerde yardımımıza koşuyor.

Bu raporda da belirttiğim üzere genel anlamda Çevik Yazılım Geliştirme metodu başarı oranı daha yüksek bir yöntemdir. Bu sebeple Geleneksel Yazılım Geliştirme metoduna göre daha çok kullanılmaktadır.

ÖNERİLER

Bu yazımı 30 Mayıs 2022 ile itibari ile yazıyorum. Çevik Yazılım Geliştirme metodunu olabilecek en güzel anlattım. Çevik Yazılım Geliştirmenin ve Geleneksel Yazılım Geliştirmenin değerini daha iyi anlayabilmek için bu yöntemler için tabii ki deneyimleyebilmemiz gerekmektedir. İki farklı yönteminde kendi içinde avantaj ve dezavantaj barındırdığını unutmamalıyız. Çalıştığımız sektöre göre bu seçim yapılabilir.

KAYNAKÇA

1: https://kurtuluskara.com/yazilim-gelistirme-nedir/

2: https://www.catsbilisim.net/yazilim-gelistirme-surecleri/

3: https://fikirjeneratoru.com/yazilim-proje-yonetimi-yontemleri/

4: https://stringfixer.com/tr/Agile_development

5: https://toptalent.co/agile-nedir-agile-metodu-nasil-uygulanir

6: https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

8: https://commons.wikimedia.org/wiki/File:Pair_programming_1.jpg

9: https://bilgisayarkavramlari.com/2012/09/09/ikili-programlama-pair-programming/

10: https://mertmekatronik.com/agile-nedir-agile-metodlari