

SDLC NEDİR?

Yazılım Geliştirme Yaşam Döngüsü'nü (**Software Development Life Cycle ,SDLC**), yazılımları tasarlamak, geliştirmek ve test etmek amaçlı kullanılan bir süreç olarak ifade edebiliriz. Yazılımin nasıl geliştirileceği, sürdürüleceği ve daha iyi hale nasıl getirileceğini açıklayan bir plandan oluşmaktadır. Buradan da yazılımin aslında bir ürün olduğu ve o ürünün de bir yaşam süreci olduğunu gözlemlemiştir oluyoruz.

SDLC, kullanıcı taleplerini (**user expectation**) karşılayacak şekilde zaman ve maliyet tahminleri dahilinde tamamlanması beklenen, yüksek kaliteli (**high quality**) yazılım üretmeyi hedefler.

hardware (donanım)



Donanımlar bilgisayarların elle tutulabilir, gözle görülebilir somut fiziki parçalarıdır.

Örneğin; Ekran Kartı, Klavye, Monitör, Anakart, Fare, Yazıcı, Harddisk gibi bileşenler.

Donanımlar kasanın içerisinde yer alabileceği gibi kasanın dışında da yer alabilir. Donanım iki kısma ayrılır. **İç Donanım** (Dahili Donanım) veya **Dış Donanım** (Harici Donanım).

software (yazılım)



Bilgisayar donanımlarının çalışmasını ve kendi aralarında haberleşmesini olanaklı hale getiren programların tamamına yazılım adı verilmektedir. Yazılım olmadan donanım olmaz. Yazılım çalıştırır, donanım ise yazılım sayesinde çalışır.

Yazılımlar **sistem yazılımları** ve **uygulama yazılımları** olarak ikiye ayrılır.

NEDEN SDLC ?



Twitter'ın güvenlik kalkanlarını aşmayı başaran siber korsanlar, platformdaki birçok tanınmış hesabı (Elon Musk, Bill Gates, Jeff Bezos, Uber, Apple ve Kanye West gibi) ele geçirerek kullanıcılarından bitcoin istedi.

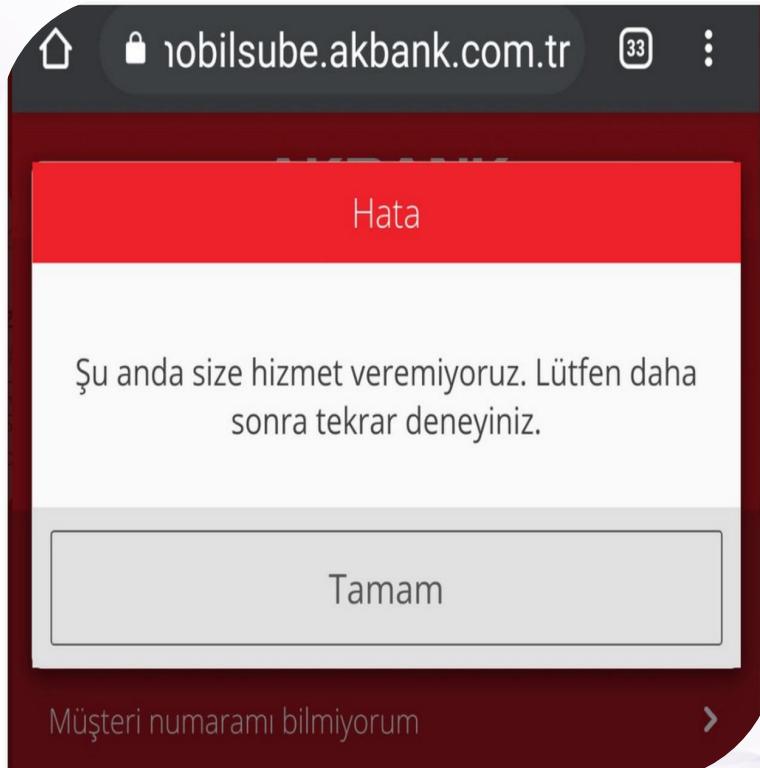
NEDEN SDLC ?

Endonezya ve Etiyopya'daki uçak kazalarının ardından Boeing model 737-8 Max ve 737-9 Max uçaklarının, tüm uçuşlarının birçok ülke ve hava yolu şirketi tarafından durdurulması, şirketin hisselerinin adeta çakılmasına sebep oldu. Gelen son verilere göre Boeing'in piyasa değeri yaklaşık 22 milyar dolar eridi.



22 Milyar Dolar Kaybetti
422 dolardan el değiştiren Boeing hisseleri 377 dolara indi. Kazadan önce Boeing'in piyasa değeri 238 milyar doların üzerindeyken bugün itibarıyla 213 milyar dolara geriledi ve 22 milyar doların üzerinde değer kaybetti.

NEDEN SDLC ?



Akbank müşterileri, bankanın ana sisteminin çökmesi sebebiyle günlerce işlem gerçekleştiremedi. Banka, web sitesinde yaptığı açıklamada sıkıntının merkez binalarındaki ana bilgisayarda meydana gelen teknik arızadan kaynaklandığını bildirdi.

NEDEN SDLC ?

Amazon çöktü 70 bin TL'lik ürün 3 kuruşa satıldı

16/12/2014 09:52

A⁺ A⁻

ABD'li online alışveriş devi Amazon.com'un İngiltere operasyonunda yaşanan bir hacker skandalı, 20 bine yakın ürünün 1 penny'lik (yaklaşık 3.6 kuruş) fiyattan satılmasına yol açtı.



Amazon.co.uk sitesi üzerinden yapılan satışlarla ilgili bir açıklama yapan şirket, olaya bir yazılım hatasının yol açtığını bildirdi.

Software Hatalarının Sonuçları

NASA, 4 Haziran 1996'da fırlatılması planlanan Ariane 5 uzay aracını kodlarken, Ariane 4 roketinin kodlarını kopyalayarak, bir hata yaptığıının farkında değildi. O gün fırlatma için geri sayım yapıldı ve roketin motorları ateşlenerek kalkış başladı.

Hızlanarak yoluna 37 saniye boyunca devam eden Ariane 5 roketi; o saniyeden sonra yanlış yöne doğru 90 derece dönmeye başladı. Bu durum, roketin kendini imha etme mekanizmasını tetikledi. Bu kaza, NASA'ya 370 milyon dolara mal oldu.

Software Hatalarının Sonuçları

Therac-25 makinesi, kanser hastalarının tedavisinde kullanılmak için tasarlanmıştı. Yapılan değişiklik Therac-20'de güvenlik önlemi olarak bulunan elektromekanik güvenlik kilitlerinin yazılımsal güvenlik önlemleriyle değiştirilmesiydi. Ne yazık ki gelişim olarak görülen bu güncelleme, Therac-20'nin kodlarında bulunan ancak fark edilemeyen hatanın, Therac-25'te ortayamasına sebep oldu. Bu bug yüzünden yaklaşık 5 hastanın ağır dozda radyasyon sebebiyle hayatını kaybettiği raporlandı.

Software Hatalarının Sonuçları

Mars Climate Orbiter Hatası (23 Eylül 1999): Gezegenler arası ilk iklim uydusu olarak 1997'de fırlatıldı. Mars Orbiter, 1999'da Mars'ın yörüngesinde kayboldu. Kazanın yazılımda kullanılan İngiliz ölçü birimlerinin metrik sisteme yanlış çevrilmesinden kaynaklandığı belirtildi. NASA'da bir ekip hesaplarında İngiliz ölçü birimini (inci) kullanırken, projeye katılan diğer ekip ise metrik (cm) sistemi kullanmıştı. 125 milyon dolarlık uydunun yörüngeye sabitlenmeye çalışırken Mars'a olması gerekenden daha fazla yaklaşarak imha olduğu düşünülüyor.

Software Hatalarının Sonuçları

Dakikalar içinde kaybedilen 460 milyon dolar. New York borsasında piyasaları yönlendiren en büyük şirketlerden biri olan Knight Capital, 1 Ağustos 2012'de yeni bir yazılım güncellemesi yapma kararı aldı. Saat 09.00 sularında New York Borsası işlemler için açıldı ve Knight Capital'ın yatırımcıları, varlıklarını satmak veya almak için talimat verdi. Yalnızca 45 dakika sonra Knight Capital'ın sunucuları 4 milyon işlem gerçekleştirdi ve şirkete 460 milyon dolar kaybettirerek iflasın eşiğine getirdi.

Olay, bir teknisyenin yeni Perakende Likidite Programı (RLP) kodunu, Knight'ın hisse senedi siparişleri için otomatik yönlendirme sistemi olan sekiz adet SMARS bilgisayar sunucusundan birine kopyalamayı unutması sonrasında meydana geldi.

Software Hatalarının Sonuçları

1991 yılının şubat ayında gerçekleşen Körfez Savaşı sırasında ABD'nin Suudi Arabistan'ın Zahran şehrindeki üssünde bir patlama yaşandı. Patlamanın sebebi ise üste bulunan anti balistik füze sisteminin doğru çalışmamasıydı. Yapılan sorgulamaların ve araştırmaların sonucunda patlama sebebinin üste bulunan anti balistik füze sisteminin bir yazılım hatası yüzünden ateşlenmemesi olduğu anlaşıldı.

Bir insan için inanılmaz küçük olan 0,33 saniye, Al Hussein füzesini takip etmek için yapılan bir sistem için inanılmaz büyük bir hataydı. MIM-104 Patriot, havada bir cisim olduğunu algılamayı başardı ancak bug yüzünden cismi takip edemedi ve bunun bir füze olduğunu anlayamadı. Engellenemeyen füze yüzünden üste bulunan 28 asker hayatını kaybetti.

NEDEN SDLC ?

1996 - 2022 yılları arası dünya genelindeki en popüler web tarayıcıları değişimleri

NEDEN SDLC ?

The 100 Billion Dollar Club

Billionaires with more than \$100 billion in net worth
(as of Oct 15, 2021)



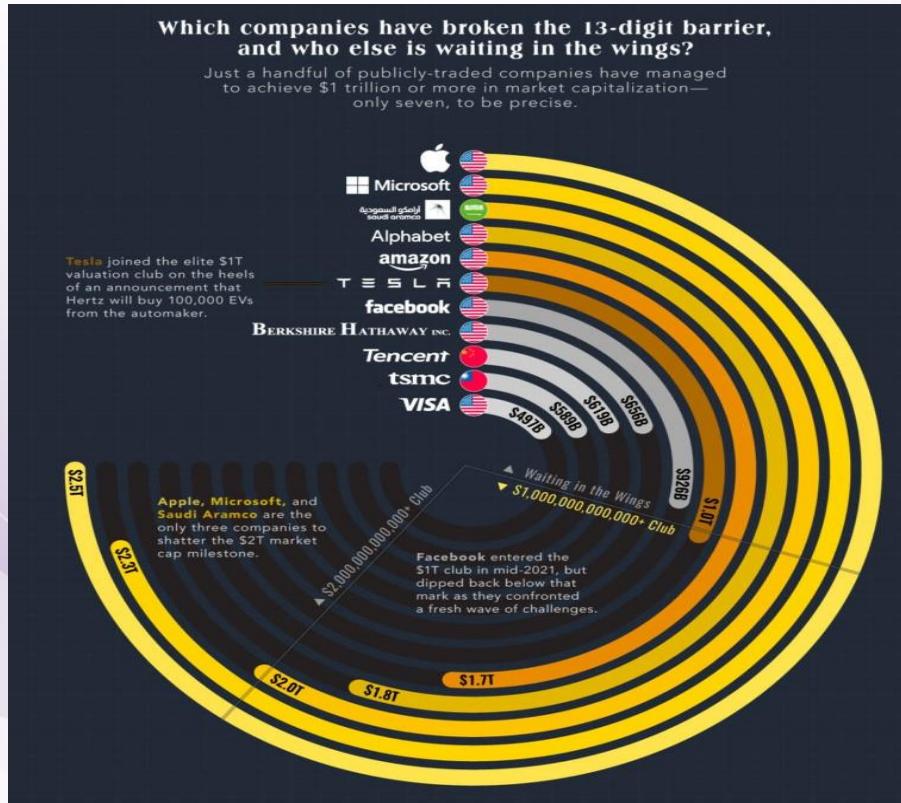
Source: Bloomberg Billionaires Index



statista 

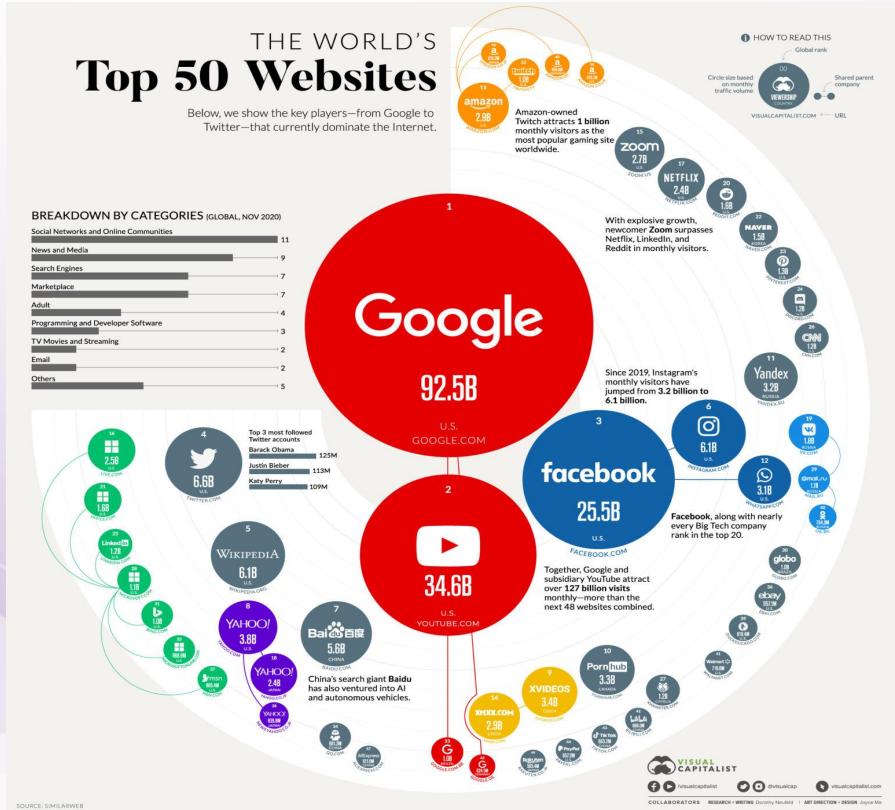
Serveti 100 Milyar Dolari Aşan Zenginler Kulübü’nde yer alan isimlerden 7 tanesinin yatırımları teknoloji sektöründedir.

NEDEN SDLC ?



Bir otomobil markası olan **TESLA** teknoloji ve yazılıma yaptığı yatırımlarla sektöründeki rakiplerinden ayrılmış ve **Piyasa Değeri Trilyon Doları Geçen Şirketler Kulübü**'ne girmeyi başaran tek otomobil şirketi olmuştur.

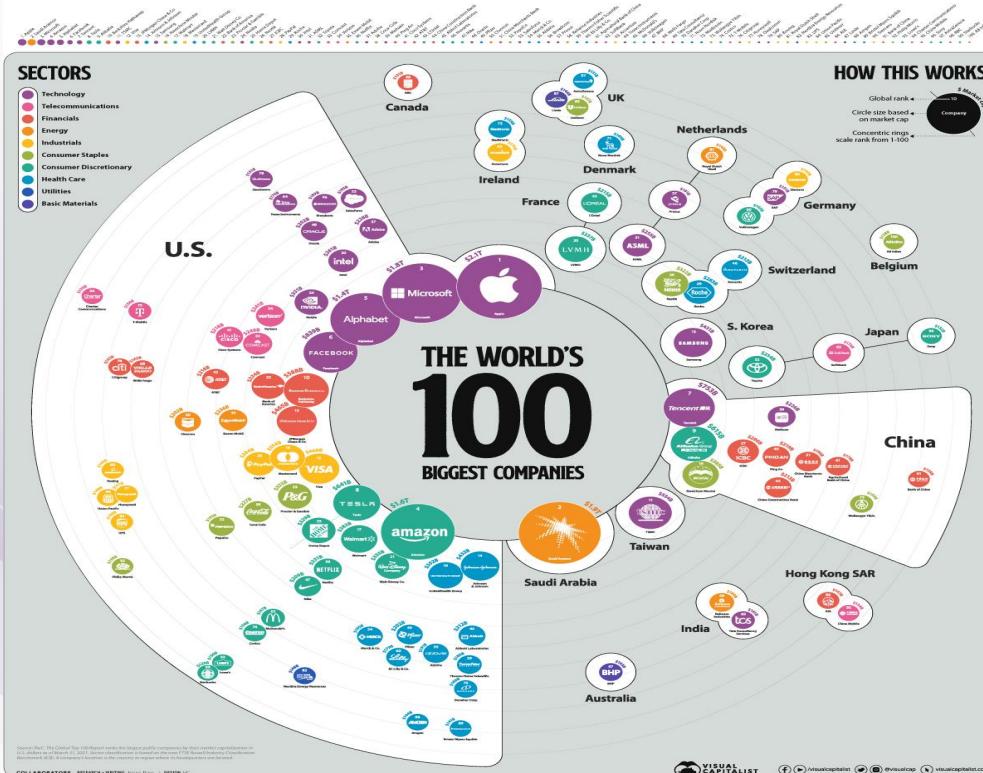
NEDEN SDLC ?



Dünyanın En Çok Ziyaretçi Alan Web Siteleri

NEDEN SDLC ?

GLOBAL MARKET CAP RANKINGS March 31, 2021



Dünyanın En Büyük Şirketleri

SOFTWARE DEVELOPMENT PHASES

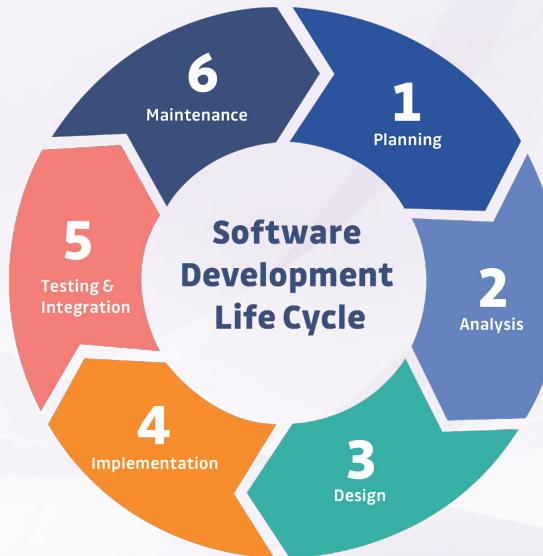
(yazılım geliştirme aşamaları)

- PLANLAMA
- ANALIZ
- TASARIM
- GELISTIRME - KODLAMA
- TEST
- BAKIM - GELISTIRME

SOFTWARE DEVELOPMENT PHASES

(yazılım geliştirme aşamaları)

SDLC mümkün olan en kısa sürede, en yüksek kalitede ve en düşük maliyetle yazılım üretmeyi hedefleyen bir süreçtir. SDLC, bir organizasyonun iyi test edilmiş ve üretim aşamasında, kullanıma hazır yüksek kaliteli yazılımı hızla üretmesine yardımcı olan iyi yapılandırılmış bir aşama akışı sağlar.



SOFTWARE DEVELOPMENT PHASES

(yazılım geliştirme aşamaları)

1	Planning	Planlama
2	Analysis	Analiz
3	Design	Tasarım
4	Implementation	Kodlama
5	Testing	Test
6	Maintenance	Teslim ve Bakım

SOFTWARE DEVELOPMENT PHASES

(yazılım geliştirme aşamaları)



Planlama (Planning)

“Ne istiyoruz?” SDLC’ nin bu aşamasında ekip, analiz edilen gereksinimlerin uygulanması için gereken maliyeti ve kaynakları belirler. Ayrıca ilgili riskleri detaylandırır ve bu riskleri azaltmak için alt planlar sunar.

Başka bir deyişle, ekip projenin fizibilitesini ve projeyi en düşük riski göz önünde bulundurarak nasıl başarılı bir şekilde uygulayabileceklerini belirlemelidir.

SOFTWARE DEVELOPMENT PHASES

(yazılım geliştirme aşamaları)

Analiz (Analysis)

Buradaki temel amaç yazılım mühendisi açısından mevcut plandaki çözümün incelenip müşteri taleplerinin doğru bir şekilde anlaşılır anlaşılmadığının ortaya çıkarılmasıdır. Bu aşamada gereksinimler detaylı bir şekilde incelenir ve projede nelerin istenildiği ile ilgili analiz çalışmaları yapılır. Daha ayrıntılı bir şekilde yapılan problem tanımlama aşamasıdır da diyebiliriz.

Her şey iki tarafça açıkça anlaşılır kesinleştirikten sonra dokümantasyon yapılır.



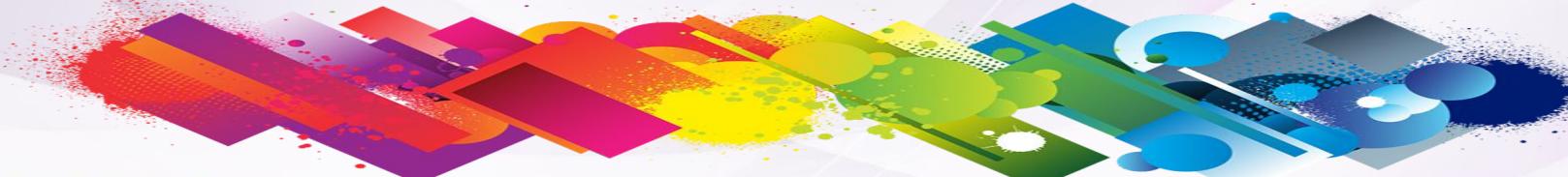
SOFTWARE DEVELOPMENT PHASES

(yazılım geliştirme aşamaları)

Tasarım (Design)

Bu aşamada gereksinimlerin analiz edilmesiyle yazılım sistemi tasarlanır. Tasarım aşamasında kodlama yapmak söz konusu değildir. Analiz aşamasında problemin ne olduğu belirlenirken tasarım aşamasında problemin nasıl çözüleceği belirlenir. Yazılım ürününün gereksinimleri karşılayan özellikleri, faydaları ve yetenekleri belirlenir.

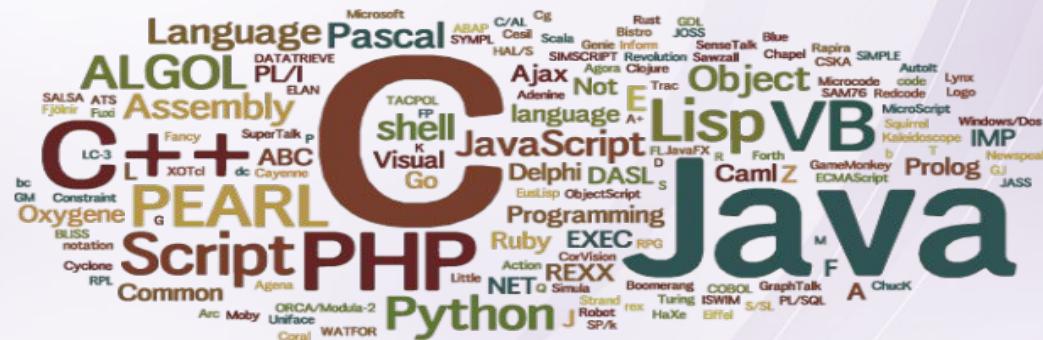
Mimari tasarımda yazılım ürününün genel bir planı yapılır ve modüller belirlenir. Ayrıntılı tasarımda yazılımda kullanılacak algoritmalar, programlama dilleri ,veritabanları ve bunun gibi detaylar belirlenir.



SOFTWARE DEVELOPMENT PHASES

(yazılım geliştirme aşamaları)

Kodlama (Implementation)



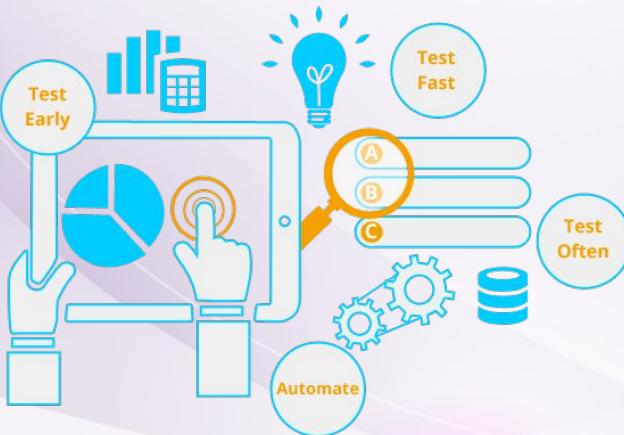
Kodlama sürecinin başladığı aşamadır. Bu aşamada doğru kodlama yapılması önemlidir. Doğru kodlama şekli bir başkasının da rahatça okuyabileceği ve bakım yapabileceği kodlamadır. Bu kodlama biçimini temiz kodlama (clean code) olarak adlandırılır.

SOFTWARE DEVELOPMENT PHASES

(yazılım geliştirme aşamaları)

Test (Testing)

Kodlama yapılrken ve kodlama sonrasında birçok test yapılır. Bu testlerden bazıları ; birim testleri, zorlanım-performans testi , yanlış değer testleri, tümleyim testi, kullanım senaryo testleri, yük testleri, kullanıcı kabul testi, test otomasyonu gibi testlerdir. Ayrıca analiz aşamasından itibaren testlerin yapılması yazılım ürününde hata oranını azaltacağı için kaliteyi arttırmır, maliyetleri(para, zaman vb) azaltır. Bu yaklaşım erken test yaklaşımı (early testing) olarak adlandırılır.



SDLC TEAM

SDLC TEAM Kimlerden Oluşur?



SDLC TEAM



(PM)

Project Manager

Proje Yöneticisi



(Dev)

Developer

Yazılım Geliştirici



SDLC TEAM

Project Manager

Proje yöneticisi, takımdaki herkesin rolünü bilmesini ve yerine getirmesini sağlar.

- Proje planının geliştirilmesinden sorumludur.
- Proje sahipleri (Stakeholder) ile yakın ilişki kurar.
- Takım içerisindeki iletişimini sağlar.
- Proje riskini yönetir.
- Proje çizelgesini hazırlar.
- Proje bütçesini yönetir.
- Projede çıkabilecek karışımları (conflicts) önler (Kriz yönetiminden sorumludur).
- Görev dağılımını yönetir.

SDLC TEAM

Business Analyst

Şirketlerin, iş süreçlerini değerlendirme, gereklilikleri öngörme, iyileştirme alanlarını açığa çıkarma ve çözümler üretme faaliyetlerini yürütür. Bir proje veya programın ihtiyaçlarını belirleyerek, bunları yönetici ve ortaklara ileter. İş sorunlarına teknik çözümler geliştirmek için çalışır.

- Business sorunları ve teknoloji çözümleri arasında bir köprü vazifesi görür.
- Gereksinimler (requirements) yönetimini ve iletişimini sağlar.
- Alınan kararların anlaşılır bir dile dökülmesini sağlar.
- **Business Requirement Document (BRD)** oluşturur.
(alınan tüm kararların ve gereksinimlerin dokümü)
- **Functional Requirement Document (FRD)** oluşturur.
(yazılımı yapılacak olan bütün maddelerin dokümü)
- Yeteri kadar Functional Requirement toplandıktan sonra user case oluşturur
- Akış şemasını oluşturur.

SDLC TEAM

Developer

Developer yazılım programlarının arkasındaki yaratıcı beyindir ve bu programları oluşturmak veya bir ekip tarafından oluşturulmalarını denetlemek için teknik becerilere sahip olan kişidir.

- Kendilerine aktarılan software requirement dökümanını toparlar ve gereken application ve programın oluşumunu sağlar.
- Beklentileri ve gereksinimleri (Customer Requirement) karşılayacak yüksek kalitede (High Quality) code yazarlar.
- Software dökümanını oluşturur ve önceki dökümanları günceller.

SDLC TEAM

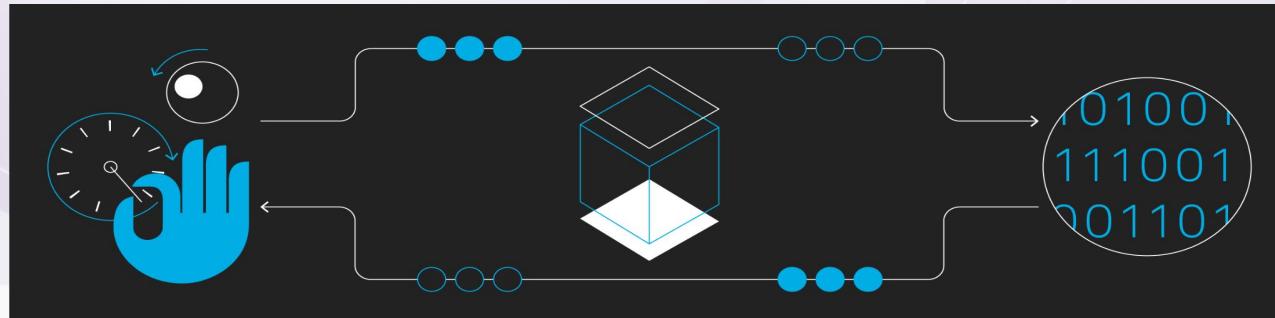
Quality Analyst (Tester)

Müşteri ve kullanıcı memnuniyetini göz önünde bulundurarak analiz ve test aşamalarında gerekli düzenlemeleri yapan kişidir.

- Son kullanıcıya bırakılmadan önce analiz ve testlerdeki tüm hataların düzeltilmesini sağlayarak hatasız ürünler sunmak.
- Herhangi bir organizasyonun ürünlerinin ve hizmetinin beklenen kalite standartlarını karşılayacak şekilde oluşturulmasını sağlar.
- Oluşturulan application'ın istenilen plan çerçevesinde yapılmasını sağlar.
- Application'daki hatalar Quality Analyst tarafından bulunmalıdır ki Developer'lar bulunan hataların üzerinde çalışıp sorun teşkil etmeyecek ürün ortaya koyabilsinler.
- Testing yapılmasının amacı herhangi bir application'da oluşabilecek hataların ortaya çıkarılmasıdır.

Bugün Neler Öğrendik?

Software Development Life Cycle (**SDLC**) : Yüksek kaliteli (**high quality**) ve kullanıcı bekłentilerini (**user expectation**) karşılayan yazılımları tasarlamak, geliştirmek ve test etmek için yazılım endüstrisi tarafından kullanılan bir süreçtir.



Bugün Neler Öğrendik?

SOFTWARE DEVELOPMENT PHASES (Yazılım Geliştirme Aşamaları)

- Planlama (Planning)
- Analiz (Analysis)
- Tasarım (Design)
- Kodlama (Implementation)
- Test (Testing)
- Teslim ve Bakım (Maintenance)



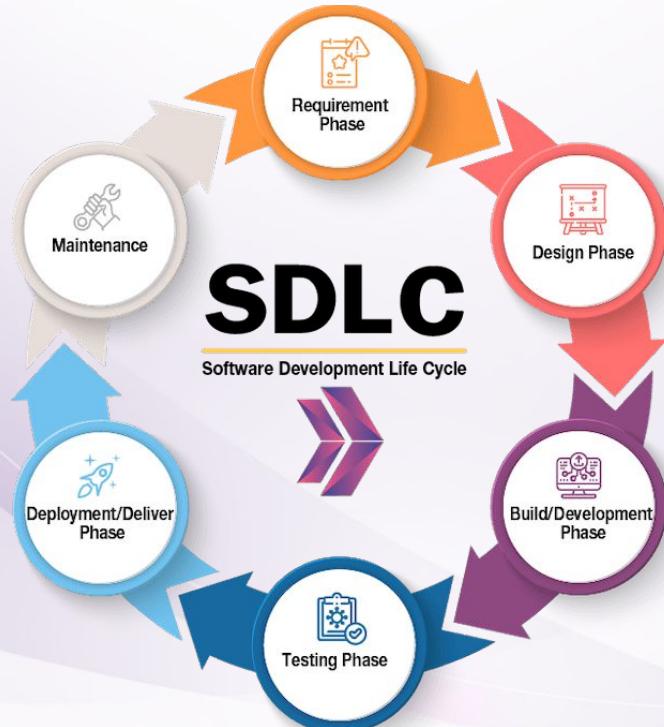
Bugün Neler Öğrendik?

SDLC TEAM

- 1) **Project Manager (PM)** Proje Yöneticisi
- 2) **Business Analyst (BA)** İş Analisti
- 3) **Developer (Dev)** Yazılım Geliştirici
- 4) **Quality Analyst (QA)** Kalite Analisti (Tester)

- Business Requirement Document (**BRD**): Alınan tüm kararların ve gereksinimlerin dökümü
- Functional Requirement Document (**FRD**): Yazılımı yapılacak olan bütün maddelerin dökümü

Geçen Ders Neler Öğrendik?



- Requirement (Planlama ve Analiz)
- Design (Tasarım)
- Development (Kodlama)
- Testing (Test)
- Deployment (Dağıtım)
- Maintenance (Bakım)

Geçen Ders Neler Öğrendik?

**PM****Project Manager**

Proje Yöneticisi

BA**Business Analyst**

İş Analisti

DEV**Developer**

Yazılım Geliştiricisi

QA**Quality Assurance**Kalite Kontrol
Analisti**SDLC Team**

Phases of the Software Development Life Cycle

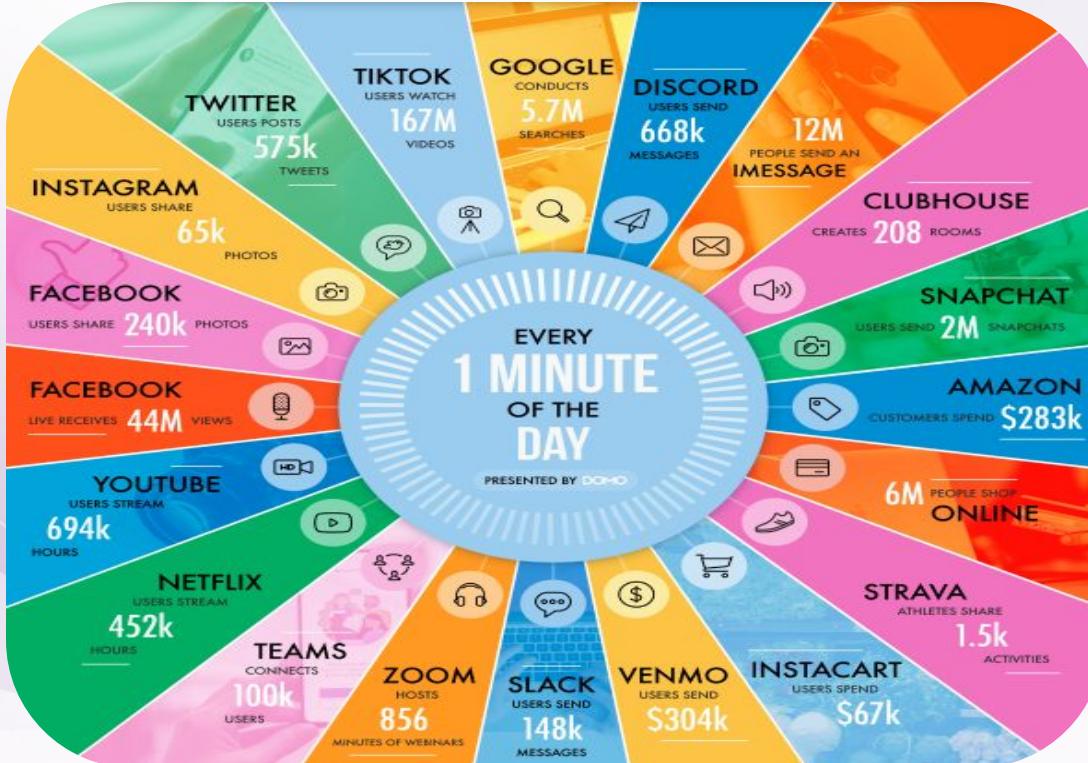


CTO => Chief Technology Officer (Teknolojiden Sorumlu Yönetici)

UX => User Experience Designer (Kullanıcı Deneyim Tasarımcısı)

UI => User Interface Designer (Kullanıcı Arayüz Tasarımcısı)

Software Development Life Cycle



Dünya
Genelinde
Internet
Üzerinden
1 Dakikada
Yapılan
İşlemler
(2021)

Software Development Life Cycle

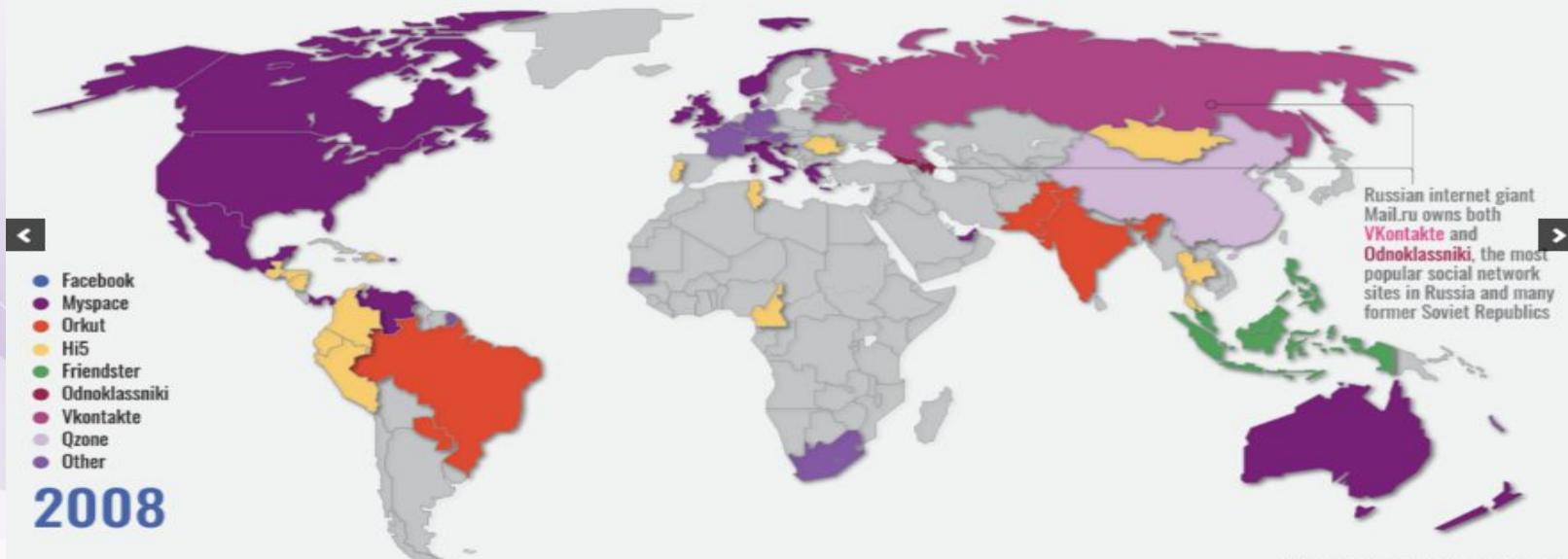


Facebook's Path to Domination 2008



Mapping Each Country's Most Popular Social Network

(according to Alexa & SimilarWeb traffic data)



Source: Vincos.it, Alexa, SimilarWeb

Software Development Life Cycle



Software Development Life Cycle



Software Development Life Cycle



SDLC PHASES

PLANNING AND REQUIREMENT ANALYSIS

İhtiyaç analizi SDLC'nin en önemli ve temel aşamasıdır.

- İhtiyaç analizi, müşteriden gelen fikirler de göz önünde bulundurularak ekibin kıdemli üyeleri (expert) tarafından yapılır.
- Bu bilgiler daha sonra temel proje yaklaşımını planlamak için kullanılır.
- Kalite güvence gerekliliklerinin planlanması ve projeye ilişkili risklerin belirlenmesi de planlama aşamasında yapılır.
- Minimum risklerle projeyi başarıyla uygulamak için izlenebilecek teknik yaklaşımalar planlanır.



SDLC PHASES

ANALYSIS, DEFINING REQUIREMENTS

İhtiyaç analizi yapıldıktan sonraki adım, ürün gereksinimlerini açıkça tanımlamak ve belgelendirmektir (dökümanete etmek).

Stakeholder / işletmeciden onay alınır.

Proje yaşam döngüsü boyunca tasarlanacak ve geliştirilecek tüm ürün gereksinimlerini içeren FRD ve BRD en küçük User Case'lere kadar hazırlanmalıdır.

BRD (Business Requirement Document) İş Gereksinimleri Dökümanı iş ihtiyaçlarını ve paydaş gereksinimleri açıklayan bir gereklilik paketidir.

FRD (Functional Requirement Document) Teknik İşlev İhtiyaçları Dökümanı hazırlanır.

SDLC PHASES

DESIGNING THE PRODUCT ARCHITECTURE

BRD (Business Requirement Document)

Designer'ların geliştirilecek ürün için en iyi dizaynla ortaya çıkacakları referanstır.

BRD'de belirtilen gereksinimlere dayanarak ürün mimarisi için genellikle birden fazla tasarım taslağı oluşturulur.

DDS(Design Document Specification)

Tasarım spesifikasyonu, bir ürün veya süreçle ilgili noktaları listeleyen ayrıntılı bir belgedir.



SDLC PHASES

BUILDING OR DEVELOPING THE PRODUCT

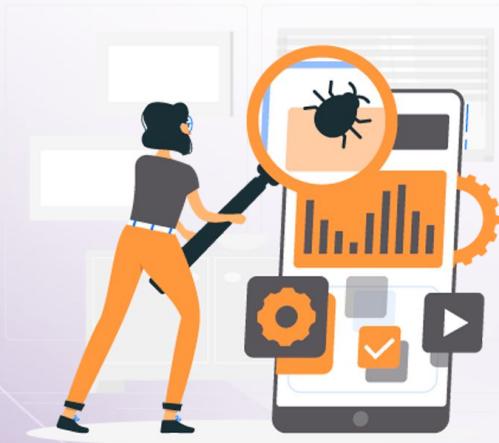
SDLC'nin bu aşamasında yazılım geliştirme başlar ve ürün inşa edilir.

- Yazılımcılar (Developers), planlama safhasında hazırlanan ve dökümanlarda tanımlanan kodlama yönergelerine uymak zorundadır.
- Developerlar gereken Functionality'leri oluştururlar.
- Kodlama için FRD baz alırlar.
- Kodlama için C ++, Java, .Net vs. gibi farklı üst düzey programlama dilleri kullanılır.



SDLC PHASES

TESTING THE PRODUCT



Bu aşama, ürünün BRD'de tanımlanan kalite standartlarına ulaşıcaya kadar, ürün kusurlarının (bug) rapor edildiği, izlendiği, düzeltildiği ve tekrar test edildiği aşamadır.

Ürün iş bekleyenlerini de karşılamalıdır.
(requirement specifications)

STLC => Software Testing Life Cycle

Test > Takip > Bulunan Hatanın Dev. Gönderilmesi > Raporlama -
Düzelte > Yeniden Test Etme > Onaylama > Raporlama

ÖRNEK TEST

1. “<https://www.saucedemo.com/>” adresine gidin.
2. Username kutusuna “standard_user” yazın.
3. Password kutusuna “secret_sauce” yazın.
4. Login butonuna basın.
5. İlk ürünün ismini kaydedin ve bu ürünün sayfasına gidin.
6. Add to Cart butonuna basın.
7. Alışveriş sepetine tıklayın.
8. Seçtiğiniz ürünün başarılı olarak sepete eklendiğini kontrol edin.
9. Sayfayı kapatın.



SDLC PHASES

DEPLOYMENT AND MAINTENANCE

Ürün test edildikten ve onaylandıktan sonra (hazır olduğunda), resmi olarak uygun görülen şekilde release edilir. (piyasaya sürüülür).

- Ürün piyasaya sunulduktan sonra mevcut müşteri tabanı için bakımı yapılır.
- Müşteriden (End-User) gelen feedbackler ve teknolojik gelişmeler neticesinde ihtiyaçlar yeniden belirlenir.
- Bu ihtiyaçlar ışığında SDLC döngüsü yeniden başlatılır.



SDLC MODELLERİ

SDLC Yazılım organizasyonu içinde bir yazılım projesi için izlenen süreçtir. Yazılımların nasıl geliştirileceği, korunacağı, değiştirileceği veya sürdürüleceğine dair ayrıntılı plan içerir. Yaşam döngüsü, yazılım kalitesini ve genel geliştirme sürecini iyileştirme yöntemlerini anlatır.

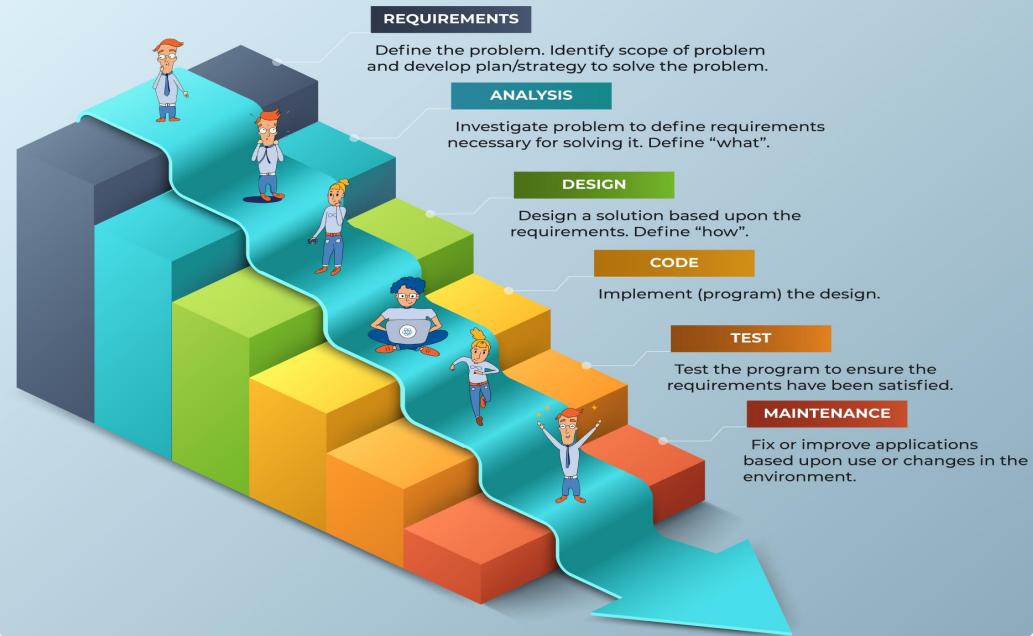
SDLC Modelleri

- Waterfall Model
- Iterative Model
- Spiral Model
- V Model
- Big Bang Model
- Agile Model
- RAD (Rapid Application Development) Model
- Prototyping Model

WATERFALL METODOLOGY (şelale)



WATERFALL METODOLOGY (şelale)



Selale modelinin (Waterfall) proje yönetim süreci; analiz, tasarım, yazılım, test, yayın ve bakım gibi fazlardan oluşur.

Geleneksel bir yöntemdir; süreçler tipki bir şelale gibi yukarıdan aşağıya doğrusal olarak işler.

WATERFALL METODOLOGY (şelale)



Bir faz tamamlanıp
yenisine geçildiğinde,
bir önceki faz'a geri
dönülmez.

Proje sahibi, proje
tamamlandıktan sonra
ürünü görebilir.

WATERFALL METODOLOGY (şelale)

- Şelale modeli, analiz adımı ile başlar. Analiz adımında, tüm yazılım gereksinimleri net bir şekilde belirlenerek analiz dokümanı üretilir.
- Daha sonra, tasarım adımında; yazılımın arayüz, veritabanı, sınıf vb. tasarımları yapılarak tasarım dökümanı üretilir.
- Bir sonraki kodlama adımında yazılım; analiz ve tasarım dokümanlarında belirtilen şekilde kodlanır.
- Test adımında; analiz ve tasarım dokümanlarındaki tüm fonksiyonel ve fonksiyonel olmayan gereksinimler ve tasarımlar için test senaryoları yazılır ve bu test senaryoları icra edilerek yazılımın testleri yapılır.
- Test adımı sonunda, yazılımda herhangi bir hata bulunmazsa, entegrasyon adımına geçilir ve yazılım, canlı ortama entegre edilerek müşterinin kullanımına açılır.

WATERFALL METODOLOGY (şelale)

ÖZET

Şelale modelinde analiz ve tasarım aşamaları oldukça detaylı yapıldığından, bu adımlar uzun sürmektedir. Ancak, analiz ve tasarım aşamalarında gereksinimlerin ve tasarımın net bir şekilde ortaya konulmasından dolayı, kodlama ve test aşamaları çok kısa sürmektedir. Test aşamasında çıkan hata sayısı azdır.

Şelale modelinde, üst adımlarda yapılan hataların yarattığı zaman kaybı oldukça fazladır. Örneğin test aşamasında karşılaşılan bir hatanın analizden kaynaklandığı tespit edilirse; analiz, tasarım ve test dokümanlarının güncellenmesi ve kodun düzeltilmesi gereklidir, ki bu oldukça ciddi zaman ve dolayısıyla para kaybına yol açar.

WATERFALL METODOLOGY (şelale)

ÖZET

Şelale modelinin dezavantajlarından bir tanesi de, ürünün ortaya çıkması için tüm aşamaların tamamlanmasını beklemek zorunda kalmaktır. Örneğin; proje 4 sene sürecek ise, müşterinin ilk prototipi görmesi için, analiz, tasarım ve kodlama aşamalarının bitmesini (yani en az 2-3 sene) beklemesi gerekecektir. Bu durum; bazı sabırsız müşteriler için sorun teşkil edebilir. Bu gibi durumlarda **Agile** yöntemler daha faydalı olabilir.

WATERFALL METODOLOGY (şelale)

AVANTAJLARI

- Kullanımı ve yönetimi kolaydır.
- Gereksinimler iyi anlaşılır.
- Proje bilgisini aktarmak daha kolaydır.
- Küçük projeler için daha iyidir.
- Görevler mümkün olduğunca sabit kalır.
- Kapsamlı dokümanlar oluşturulur.

WATERFALL METODOLOGY (şelale)

DEZAVANTAJLARI

- Değişim ve yenilik zordur.
- Müşteri öngörü ve önerileri önemsenmez.
- Projenin bitimine kadar çalışan ürün yoktur.
- Beklenmedik riskler kolayca ele alınamaz.

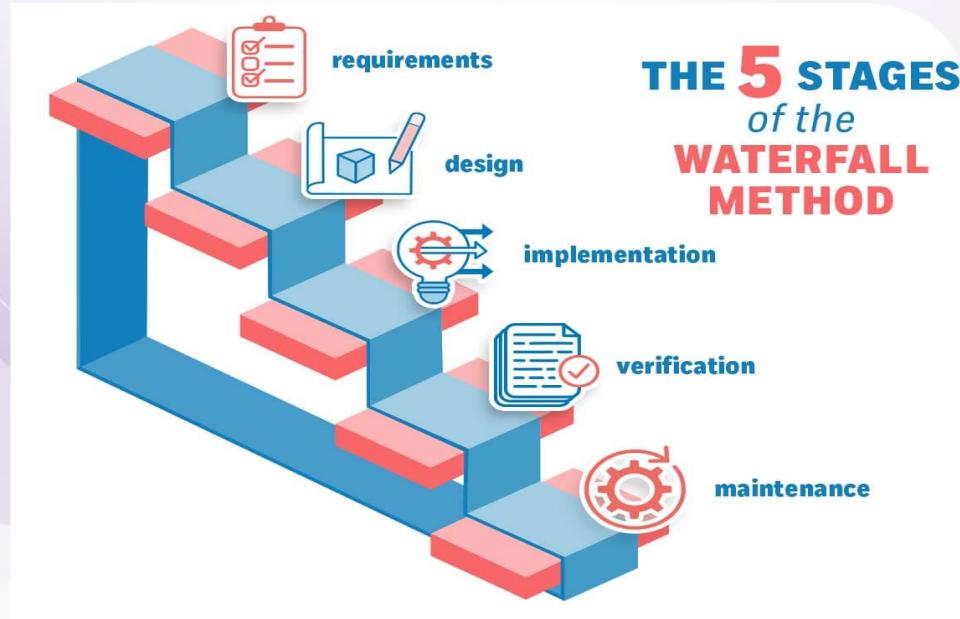
Bugün Neler Öğrendik?

SDLC PHASES

- 1) Planning and Requirement Analysis (Planlama ve İhtiyaç Analizi)
- 2) Defining Requirements (Gereksinimleri Tanımlama)
- 3) Designing The Product Architecture (Ürün mimarisini tasarlama)
- 4) Building or Developing The Product (Ürünü oluşturma veya geliştirme)
- 5) Testing The Product (Ürünü test etme)
- 6) Deployment in The Market and Maintenance (Ürünü pazarlama ve bakım)

Bugün Neler Öğrendik?

WATERFALL MODEL (ŞELALE MODELİ)



Yazılım geliştirmeyi önceden tanımlanmış aşamalara bölen sıralı bir modeldir. Her aşama, aşamalar arasında çakışma olmadan bir sonraki aşamanın başlayabilmesi için tamamlanmalıdır. Her aşama, SDLC aşaması sırasında belirli aktiviteyi gerçekleştirmek için tasarlanmıştır. 1970 yılında Winston Royce tarafından tanıtıldı.

Geçen Ders Neler Öğrendik?

SDLC Modelleri

- Waterfall Model
- Iterative Model
- Spiral Model
- V Model
- Big Bang Model
- Agile Model
- RAD (Rapid Application Development) Model
- Prototyping Model

Geçen Ders Neler Öğrendik?

Waterfall Methology

- Waterfall yöntemi, yazılım geliştirmede kullanılan erken SDLC yaklaşımıdır.
- Eski firmalarda ve devlet projelerinde kullanılır.
- Planlamanın önemli olduğu askeri projelerde tercih edilir.
- Waterfall'da her aşama bir önceki aşama tamamen bittikten sonra başlıyor.
- Waterfall modelinde fazlar çakışmaz.
- Tüm bu aşamalar, ilerlemenin aşamalar boyunca sürekli olarak aşağı doğru (şelale gibi) aktığı şekilde kademelendirilir.
- Analiz ve Tasarım sürecinin detaylı ve sorunsuz yapılması önemlidir.
- Küçük projeler için daha iyidir.
- Uzun süre devam eden projeler için zayıf modeldir.

Geçen Ders Neler Öğrendik?



Waterfall Methodology (Şelale Modeli)

AGILE PROCESS

Konu Akışı

- 1) KISA TARİHÇE
- 2) TAKIM ÜYELERİ VE GÖREVLERİ
- 3) BÜTÜNÜ OLUŞTURAN PARÇALAR
- 4) TOPLANTILAR



AGILE METHODOLOGY (çevik)

KISA TARİHÇESİ

Scrum'ın Ortaya Çıkışı :

90'lı yıllarda Scrum'in kurucularından olan Jeff Sutherland zor bir görev üzerinde çalışmaya başladı: Bir yazılım şirketi olan Easel Corporation, altı aydan daha kısa bir sürede eski ürünlerini tamamen yenilemek istiyordu.

Sutherland'ın temel yaklaşımı, **kısa günlük ekip toplantılarının grup verimliliğini** önemli ölçüde artırdığı şeklindeydi.

Sutherland, yazılım geliştirmenin yeni bir yolunu keşfetti ; Rugby oyunundan esinlenerek adını “**scrum**” (hamle, saldırısı) koydu.

Scrum yöntemleri, görünüşte karmaşık ve bilinmezliği olan projelerin zamanında, az bütçeyle ve daha önceki herhangi bir sürümden daha az hatayla tamamlamasını sağladı.

AGILE METHODOLOGY (çevik)



Agile Metodoloji (Çevik Metodoloji) yazılım sistemlerini etkili ve verimli bir şekilde modellemeye ve dokümantasyonunu yapmaya yönelik, pratiğe dayalı bir yöntemdir.

AGILE METHODOLOGY (çevik)

Diğer sektörlerdeki yaklaşımların bir devamı olarak 1990'lı yıllarda itibaren yazılım sektöründe uygulanmaya başlanmıştır.

Aşırı kuralcı klasik yazılım süreç modellerine tepki olarak ortaya çıkan Agile Manifestosu öncesinde yazılımlar daha yüksek maliyetli ve daha yavaş geliştirilmekteydi.

Yazılım geliştirme sürecini hızlandırmak, daha etkin kullanmak ve gerektiğinde dokümantete etmek amacıyla ortaya çıkan, dünyada birçok yazılım firmasının farklı projelerinde benimsediği bu metodun kullanımı 2000'lerde hız kazanmıştır.

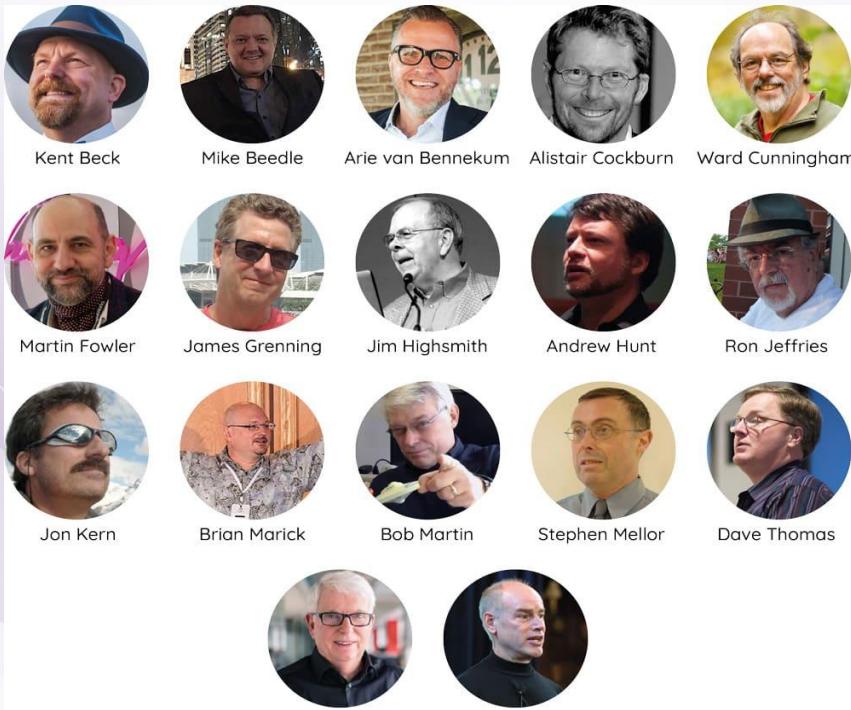
AGILE METHODOLOGY (çevik)

2001 yılında yazılım dünyasının önde gelen isimlerinden 17 arkadaşı; “Agile (Çevik) Yazılım Geliştirme Manifestosu” ve “Agile (Çevik) Yazılımın Prensipleri” ni yayımlamışlar, bu oluşumu ve gelişimini desteklemek için “Agile Alliance” adıyla, kar amacı gütmeyen bir organizasyon kurmuşlardır.

Manifesto, nasıl daha iyi bir yazılım geliştirilebileceğini gösteren 12 maddeden oluşmaktadır.

<https://www.agilealliance.org/>

AGILE METHODOLOGY (çevik)



Düşünce liderlerinden oluşan 17 kişilik bir grup önce, 2000 yılında Oregon'da bir otelde daha sonra da 2001 yılında Utah'da bir kayak merkezinde buluşarak yazılım geliştirme sürecinde daha üretken ve verimli işler sunabilmek adına beyin fırtınası yaptılar.

Toplantılarının sonucunda fikir birliğine vararak Agile Manifesto'yu ve Çevik Yazılım'ın 12 Prensibini yazıya geçirdiler.

www.wisequarter.com

AGILE METHODOLOGY (çevik)

Agile Manifesto

“Bizler uygulayarak ve başkalarının da uygulamalarına yardım ederek daha iyi yazılım geliştirme yollarını 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.”

AGILE METHODOLOGY (çevik)

1. Süreçler ve araçlardan ziyade **bireyler ve etkileşimlere**

Agile Manifesto'nun bu değeri müşterilerle olan iletişime daha fazla önem vermenin üzerinde duruyor. Müşterilerin sormak isteyebileceği birçok şey vardır ve müşterilerden gelen tüm soruların ve önerilerin hızlı bir şekilde ele alınmasını sağlamak ekip üyelerinin sorumluluğundadır.



AGILE METHODOLOGY (çevik)

2. Kapsamlı dokümantasyondan ziyade çalışan yazılıma



Geçmişte, projenin her aşamasının doğru bir şekilde belgelendirilmesine projenin kendisinden daha fazla odaklanılıyordu. Hatta, uygun bir dokümantasyonun nihai ürün pahasına yapıldığı da çok kez görülmüştür. Çevik değerler, proje ekibinin ilk ve en önemli görevinin müşteriler tarafından belirlenen nihai teslimatları tamamlamak olduğunu belirtir.

AGILE METHODOLOGY (çevik)

3. Sözleşme pazarlıklarından ziyade *müşteri ile işbirliğine*

Çevik ilkeler, müşterilerin projenin her aşamasına dahil olmalarını gerektirir. Waterfall yaklaşımı ya da geleneksel metodolojiler müşterilerin sadece projeden önce ve sonra pazarlık yapmasına imkân tanır. Bu hem zaman ve hem de kaynak israfına neden oluyordu. Müşteriler geliştirme sürecinin tamamında döngü içinde tutulabilirse, takım üyeleri nihai ürünün müşterinin bütün gereksinimlerini karşılamasını sağlayabilir.



AGILE METHODOLOGY (çevik)

4. Bir plana bağlı kalmaktan ziyade **değişime karşılık vermeye**



Geçmişteki yönetim metodolojilerinin tersine, çevik değerler projeye başlamadan önce oluşturulan ayrıntılı planlara ve ne olursa olsun plana sadık kalmaya karşıdır. Koşullar değişebilir ve bazen müşteriler nihai ürün için projenin kapsamını değiştirebilecek yeni özellikler talep edebilir. Böyle durumlarda, proje takımları ve yöneticileri yüksek kalitede bir ürün teslim edebilmek ve %100 müşteri memnuniyetini sağlamak adına hızlı bir şekilde değişen yeni şartlara uyum sağlamalıdır.

AGILE METHODOLOGY (çevik)

Agile Prensipleri

1) Değerli yazılımın erken ve sürekli teslimatı ile müşteri memnuniyeti sağlanır:

Değerli yazılım kavramı müşteri tarafından önceliklendirilen ve kullanılabilir, kaliteli yazılım olarak düşünülebilir. Yazılımın tamamlanan kısımlarının erken ve sürekli müşteriye aktarımı, görüşlerinin alınması memnuniyeti arttırmır.

Scrum yöntemi kısa döngüler ile çıktı üretmek ve bu süreci yönetmek için etkili bir yöntemdir.



AGILE METHODOLOGY (çevik)

Agile Prensipleri

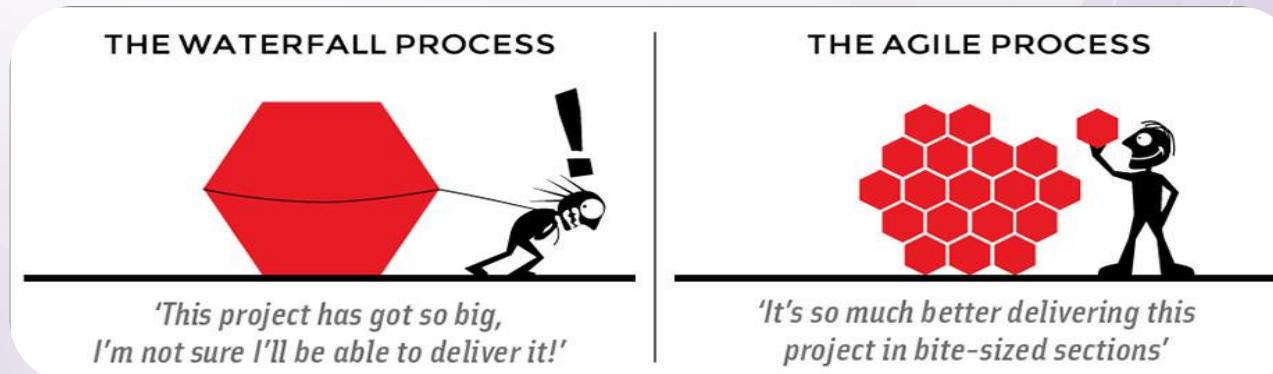
2) Değişiklikler projenin son aşamalarında dahi olsa kabul edilir:

Yazılım projelerinde sıkılıkla rastlanan değişiklik talepleri rekabet avantajının korunabilmesi için çevik yöntemlerle yönetilmelidir, ancak teslim süresi ve kaynaklar taleplere göre etkin ve doğru planlanmalıdır.

AGILE METHODOLOGY (çevik)

Agile Prensipleri

3) Çalışan yazılımlar sık aralıklar ile teslim edilir:



Tamamlanan iş paketleri aylık yerine haftalık şekilde müşteriye canlı ortamda teslim edilmelidir. İdeal olarak kabul edilen 2-4 haftalık “Sprint” adı verilen süreçlerdir.

AGILE METHODOLOGY (çevik)

Agile Prensipleri

4) Müşteri ve programcılar proje boyunca ekip halinde çalışır:

Taleplerin sahibi olan iş birimleri/müşteri ile farklı rollere sahip analist, yazılım geliştirici vb. taleplerin doğru anlaşılması ve hızlı karşılanması adına aynı ekibin içinde çalışmalı ve sürekli iletişim halinde olmalıdır.



AGILE METHODOLOGY (çevik)

Agile Prensipleri

5) Projeler motive edilmiş bireyler ile güven esasına dayalı kurulur:

Proje ekibinde yer alan her üye ayrı yetkinlik ve karakteristik özellikleri ile takımın bir parçası olarak kabul edilir. Özellikle öz motivasyonu yüksek ekip bireyleri ile çalışmak ekip performansı açısından olumlu etki yaratmaktadır. Bunun yanında ekip üyelerinin kendilerine atanan işleri zamanında bitireceğine ve yetkinliklerine dair güven esası da motivasyonu arttırmada önemli bir faktördür.

AGILE METHODOLOGY (çevik)

Agile Prensipleri

6) En etkili iletişim yolu yüz yüze görüşmektir:

Proje ekibinin aynı lokasyonda bulunması ve mümkün olduğunda yüz yüze iletişim kurması tercih edilir. Proje ekibine ait masa çalışma düzeni U şeklinde düzenlenebilir, bu sayede yanlış anlaşılmalar ortadan kalkarak, sorunlar hemen açıklığa kavuşturulabilir. Coğrafi olarak farklı lokasyonlarda olan ekip üyeleri için ise telekonferans gibi hızlı sonuç alınabilecek iletişim yöntemleri tercih edilmelidir.

AGILE METHODOLOGY (çevik)

Agile Prensipleri

7) Sürecin öncelikli ölçüdü çalışan yazılımdır:

Müşteriye kısa aralıklarla çalışır durumda olan yazılımı sunmak gereksinimlerin takibi ve süreç ilerleyişinin anlaşılması konusunda önemlidir.

AGILE METHODOLOGY (çevik)

Agile Prensipleri

8) Çevik süreçler sabit hızlı, sürdürülebilir gelişitmeyi teşvik eder:

Sabit bir çalışma temposunun oluşturulması ve eşit görev dağılımı sürdürülebilirlik açısından önem taşır.

AGILE METHODOLOGY (çevik)

Agile Prensipleri

9) Teknik mükemmelliğe ve iyi tasarıma dikkat etmek çevikliği ileri taşırlı:

Kalite beklentisi ve müşteri memnuniyeti sağlanması için en iyi araçlar ve yetkinlikte ekip üyeleri ile çalışılmalıdır.

AGILE METHODOLOGY (çevik)

Agile Prensipleri

10) Sadelik esastır:

Müşteriyi daha memnun edeceği düşüncesi ile yapılan ve proje yönetim terminolojisinde “Altın Kaplama” adı verilen ilaveler ile kapsam dışına çıkmamalı, müşterinin ana gereksinimlerine uygun ve değişikliği kolay çıktılar üretilmelidir.

AGILE METHODOLOGY (çevik)

Agile Prensipleri

11) En iyi mimariler, gereksinimler ve tasarımlar kendi kendine organize olmuş takımlardan çıkar:

Çevik ekipler kendi başlarına organize olarak görev paylaşımında bulunma yetkinliğine sahip bireylerden oluşturulmalıdır. Böylelikle, ekip kendi çalışma yöntemlerini sorgulamakta ve gerekli değişiklikleri yapmakta özgür olarak gelişime ve öğrenmeye açık olarak ilerleme kaydeder.

AGILE METHODOLOGY (çevik)

Agile Prensipleri

12) Takım düzenli aralıklarla nasıl daha verimli olacağını konuşur ve buna göre davranışlarını ayarlar:

Çevik projelerde değişen koşullara uyum sağlamak önemlidir. Scrum metodolojisine göre ekip üyeleri günlük olarak en fazla 15 dakika süren ve ayakta yapılan Stand-up adı verilen toplantılar ile bir araya gelir, günlük iş planlarını tartışır. Sprint adı verilen 2-3 haftalık iş planlamalarının süresi tamamlandığında ise gelecekte karşılaşılabilen değişiklere karşı daha iyi adaptasyon sağlayabilmek adına geçmişe dönük değerlendirme yapmak ve öğrenilmiş dersleri çıkarmak üzere Retrospective adı verilen toplantılar yapılır.

AGILE METHODOLOGY (çevik)



AGILE METHODOLOGY (çevik)

SCRUM

Scrum: Agile proje yönetim metodolojilerinden biridir. Kompleks yazılım süreçlerinin yönetilmesi için kullanılır. Bunu yaparken bütünü parçalayan; tekrara dayalı bir yöntem izler. Düzenli geri bildirim ve planlamalarla hedefe ulaşmayı sağlar. Bu anlamda ihtiyaca yönelik ve esnek bir yapısı vardır.

Müşteri ihtiyacına göre şekillendiği için müşterinin geri bildirimine göre yapılanmayı sağlar. İletişim ve takım çalışması çok önemlidir.

AGILE METHODOLOGY (çevik)

SCRUM

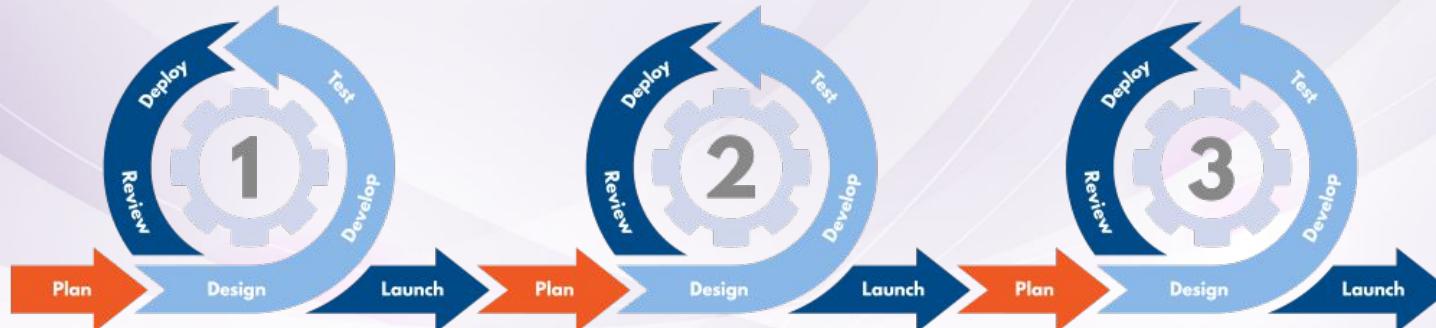


“Rugby Yaklaşımı” olarak adlandırılan temel felsefe , “takımın, mesafenin tümünü hep beraber, bir birim halinde topu ileri geri atarak kat etmesidir.”

AGILE METHODOLOGY (çevik)

SCRUM

Her sprint genellikle 2 ila 4 hafta veya en fazla bir takvim ayı sürer. Ürünleri her seferinde küçük bir parça oluşturmak, üretkenliği teşvik eder ve ekiplerin geri bildirime ve değişime yanıt vermesini ve tam olarak gerekli olanı oluşturmasını sağlar. Scrum'da ürün sprint'te tasarılanır, kodlanır ve test edilir.



AGILE METHODOLOGY (çevik)

SCRUM TEAM

Pig Roller: Scrum sürecine dahil olanlar yani projede asıl işi yapan kişilerdir.

- 1) **Product Owner (PO)**
(Ürün Sahibi)
- 2) **Scrum Master**
(Çoban Köpeği)
- 3) **Development Team**
(Geliştirme Takımı)



AGILE METHODOLOGY (çevik)

SCRUM TEAM

Chicken Roller: Scrum'ın işleyişinde aktif olarak yer almayan kişilerdir.

1) **Business Owner (BO)**

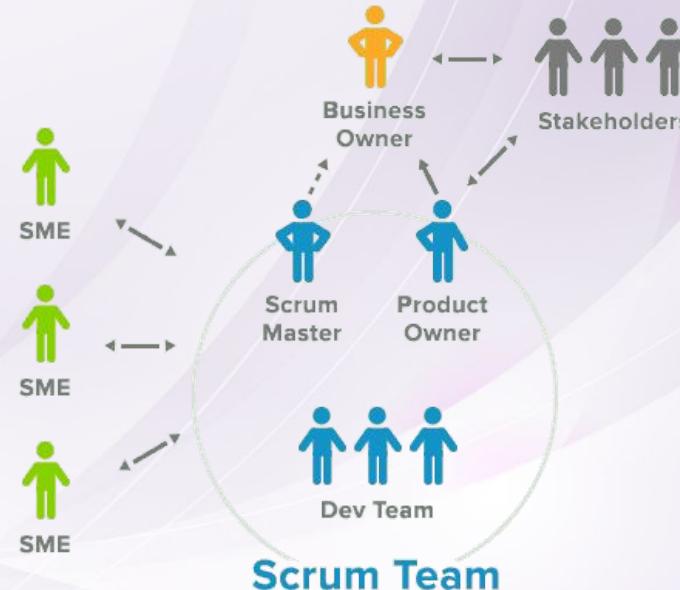
(İş Sahibi)

2) **Stakeholders**

(Paydaşlar)

3) **Subject Matter Expert (SME)**

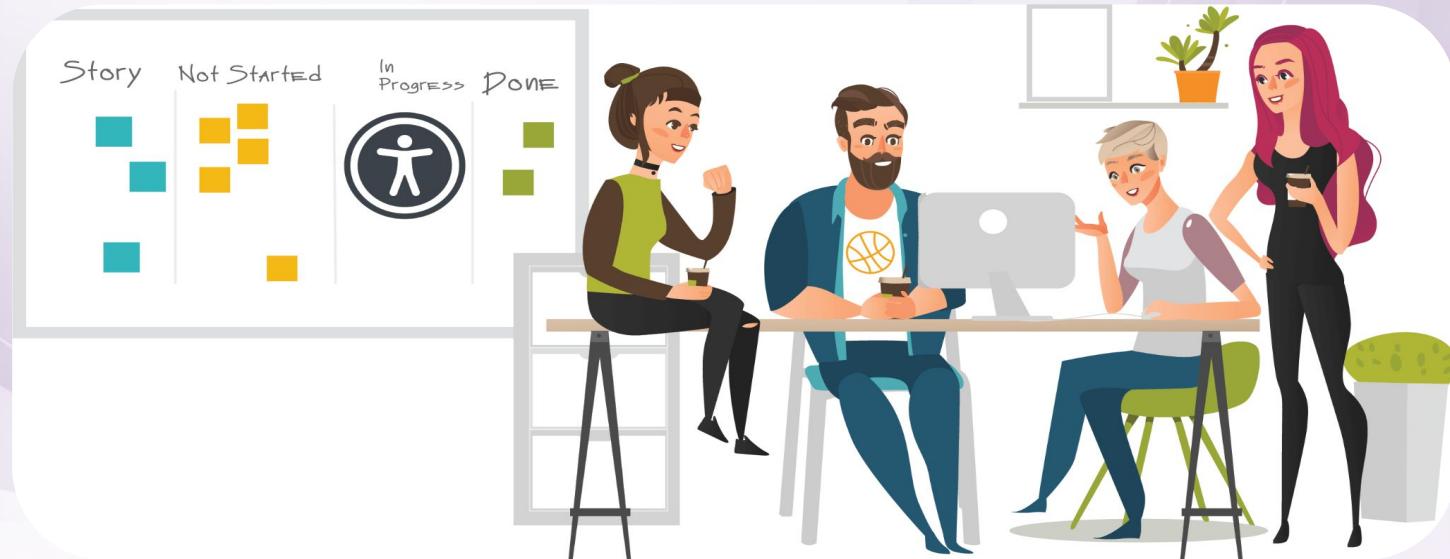
(Konu Uzmanı)



AGILE METHODOLOGY (çevik)

SCRUM TEAM

Takım kendi kendini örgütler. (Self Organized) Böylece kendi içerisinde uyum içinde olan takımlar daha başarılı sonuçlar alırlar.



AGILE METHODOLOGY (çevik)

SCRUM TEAM

1) Product Owner: Geliştirme takımı ve müşteri arasındaki iletişimini sağlar.

- Projenin özelliklerini tanımlar. Projenin önceliklerine göre **Product Backlog** (iş listesi) oluşturur.

- Ekipde Stakeholders'ı temsil eder.

- Product Backlog'un sahibi PO'dur.

- Birinci önceliği; İş Listesini (Product Backlog) yönetmektir.

Product Backlog'i yönetmek demek, ürünü yönetmek demektir.

- Ürünle ilgili yapılacak bütün geliştirmeler Product Backlog'da bulunur.

- Product Backlog herkes tarafından erişilebilir ve anlaşılabilir olmalıdır.

- Product Owner, Geliştirme Takımı ve Stakeholders arasındaki iletişimini gerçekleştirir.

- Her Sprint'te hangi user story'lerin sprinte dahil edileceğine karar verir.

- Sprint değerlendirme toplantılarının (**Sprint Review**) sahibidir. Bu toplantıların organizasyonunu yapar.

AGILE METHODOLOGY (çevik)

SCRUM TEAM

Product Owner Kimdir?

- Scrum takımının üyesidir.
- İletişimi kuvvetlidir.
- Müşteri ve Development Team arasında iletişim kurar.
- Sorumluluğu elinde tutar.
- Belirleyicidir.
- Kararlı ve ulaşılabilir.
- Ürün sorumluluğunu kabul eder.
- Karar verme yetkisine sahiptir.
- İş ve etki alanı yeterliliğine sahiptir.



AGILE METHODOLOGY (çevik)

SCRUM TEAM

Product Owner Kim Değildir?

- Development Team'in ve Scrum'in yönetici değilidir.
- Geliştirme Takımı teknik konularda kendi kendini yönetir.
- PO, teknik bir geçmişe sahip olabilir.
- Hangi işi kimin yapacağına karışamaz.
- PO, iş ve görev ataması yapamaz.
- İşin nasıl yapılacağına karışamaz!



AGILE METHODOLOGY (çevik)

SCRUM TEAM

2) Scrum Master: Scrum kurallarını, teorilerini ve pratiklerini iyi bilir ve takımın bu kuralları uygulamasından sorumlu kişidir. Takımın yöneticiisi değildir. Takımı rahatsız eden, verimli çalışmalarını engelleyen durumları ortadan kaldırır.



“Scrum Master, takımın Scrum değerlerine, pratiklerine ve kurallarına bağlı kalmasını garanti altına almakla sorumludur. Scrum Master, takımı ve organizasyonu Scrum'a adapte eder.”

AGILE METHODOLOGY (çevik)

SCRUM TEAM

Scrum Master Kimdir?

- Scrum takımının üyesidir.
- İş birlikçidir.
- Koruyucudur.
- Yardımcıdır.
- Problem çözücüdür.
- Kararlı ve ulaşılabiliridir.
- Bilgilidir.



AGILE METHODOLOGY (çevik)

SCRUM TEAM

- Scrum Master, takıma rehberlik ve koçluk eder, karşılaşılan engelleri ortadan kaldırırmalarına yardımcı olur.
- Takım içi harmoniyi, ekip elemanları arasındaki uyumu, iletişimini artırmak için çabalar.
- Sprint Planlama, Sprint Retrospektif Günlük Scrum ve Sprint Review gibi Scrum ritüellerini ve toplantılarını kolaylaştırır.
- Scrum Master takımın güvenli ve sorunsuz bir ortamda çalışabildiğinden emin olmalı, gerektiğinde takım elemanlarına bireysel koçluk da dahil olmak üzere birçok hizmeti sunmalıdır.



AGILE METHODOLOGY (çevik)

SCRUM TEAM

- *Product Owner ile ilişkileri yönetir.*

Scrum Master, ürün sahibi tarafından belirlenmiş işlerin takımındaki herkes tarafından anlaşıldığından emin olur; Product Owner iş listesini etkili bir şekilde organize edebilmesi için teknikler bularak ona yardımcı olur.

- *Değişime liderlik eder.* Tüm bu görevlerin yanı sıra Scrum Master'lar değişime liderlik ederler.

- Scrum Master, Scrum'ın ve organizasyondaki çevik değişimin vücut bulmuş halidir.



SCRUM MASTER ?



AGILE METHODOLOGY (çevik)

SCRUM TEAM

3) Development Team: Geliştirme Ekibi, Front-End Developer, Back-End Developer, Dev-Ops, QA (Tester), İş Analisti (BA), Data Base Analisti (DBA) vb. gibi özel becerilere sahip kişilerden oluşabilir.

- Product Owner'ın yapılacak işler listesi olan Product Backlog'tan, belli bir sürede (Sprint) yapabileceği kadar işi, Product Owner'ın belirlediği önceliğe göre yapan geliştirme takımıdır.
- Teknik konularda sorumluluk Development Team'e aittir.
- PO ve SM Development Team'in yapacağı işlerin öncelliğini belirleyebilir ama neyi nasıl yapacaklarına karışmazlar.
- Takım içerisindeki kişilerin rolleri ve yetenekleri ne olursa olsun, dışarıya karşı bir işin tamamlanmasından tüm takım sorumludur.

AGILE METHODOLOGY (çevik)

SCRUM TEAM

- Bir Sprint'e alınan bütün işleri tamamlayacak özelliklere sahip kişilerdir. Sprint Backlog'u oluştururlar. Kendi kendini yönetir. İşin verilmesini beklemezler, işi kendileri alır ve geliştirirler.
- Development Team, Product Backlog'tan çektiği bir Product Backlog Item'ı, Product Owner'ın önüne, çalışan bir kod parçası olarak koymakla yükümlüdür.
- “Self Organize” olmalıdır. Development Team, sorumluluğunu aldığı işlerin yapılması için bir iş tanımlamasına veya bir iş takipçisine ihtiyaç duymaz.



AGILE METHODOLOGY (çevik)

SCRUM TEAM

Development Team Kimdir?

- Scrum takımının üyesidir.
- Dışarıdan herhangi bir yardıma ihtiyaç duymadan çalışmalarını tamamlamak için gerekli tüm beceri setlerine sahiptir..
- Kendi kendine yeterlidir.
- Kendi kendine organizedir.
- Yeteneklidir.
- Kararlı ve ulaşılabilirdir.
- Sorumludur.

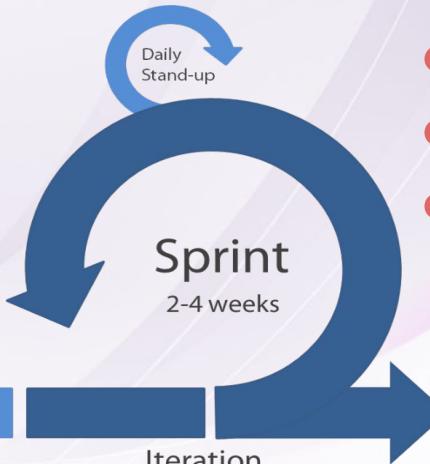


SCRUM PROJE YÖNETİM SÜRECI

Bütünü Oluşturan Parçalar

Süreci Oluşturan Parçalar

- Scrum
- Sprint
- Toplantılar



Iteration



Potentially shippable
product increment

Ürünü Oluşturan Parçalar

- Product Backlog
- Sprint Backlog
- Feature
- Epic
- User Story

SCRUM PROJE YÖNETİM SÜRECI

Süreci Oluşturan Parçalar

SCRUM

- Projenin açık ve net olması hem zaman kazandırır hem de projenin başarılı sonuçlanması sağlar.
- Projenin anlaşılabilir ve yönetilebilir parçalara ayrılması olası sorunları hızlı bir şekilde tespit etmekte ve düzeltmekte zaman kazandırır.
- Bütün ekibin, projenin tüm akışından haberdar olması takım içindeki iletişimini artırır.
- Müşteri ile geliştiriciler arasında güven oluşur ve böylelikle projenin başarılı sonuçlanması beklenir.
- Scrum mantığının doğru ve anlaşılır bir şekilde uygulanmasını sağlayan kişi **Scrum Master**'dir.

SCRUM PROJE YÖNETİM SÜRECI

Süreci Oluşturan Parçalar

SPRINT

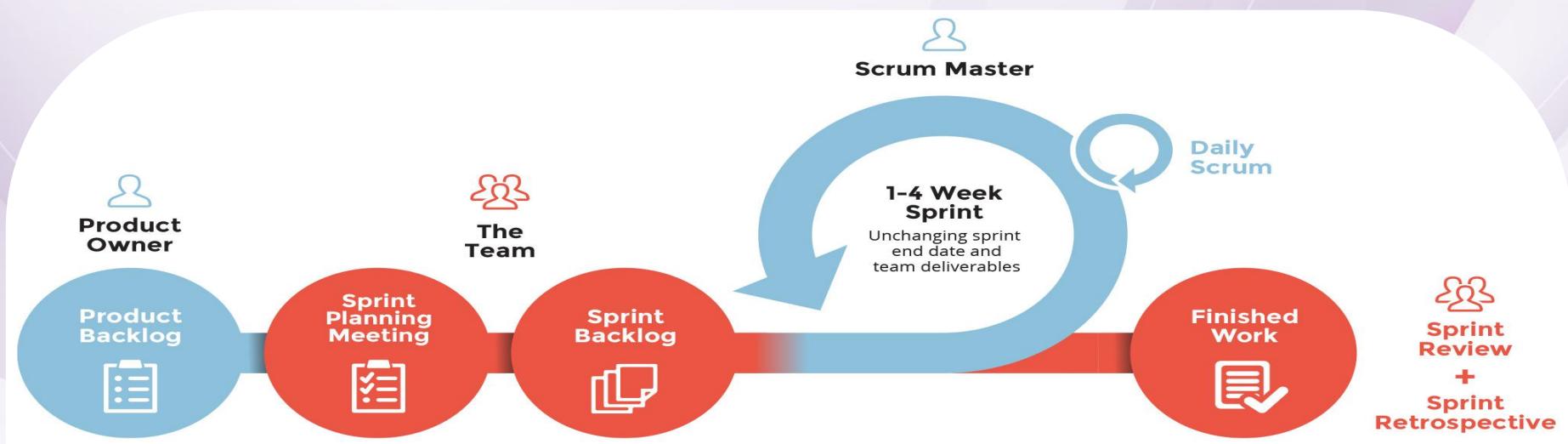
- Scrum süreci 2-4 hafta uzunlukta, ara vermeksizin birbirini takip eden Sprint'lerden (koşulardan) oluşmaktadır.
- Sprint Scrum takımının ritmi, yani kalp atışıdır. Scrum içerisindeki tüm aktiviteler Sprint içerisinde gerçekleşir.
- Bu kısa süreler, takımlara esneklik ve çeviklik avantajı sağlamaktadır. Scrum takımları ürün geliştirmeye başlarken Sprint sürelerini belirler, başlangıç ve bitiş gün ve saatlerine karar verirler, sonrasında da alınan bu karar doğrultusunda ara vermeksizin Sprint'leri koşmaya başlarlar.



AGILE METHODOLOGY (çevik)

SCRUM

Toplantılar



AGILE METHODOLOGY (çevik)

SCRUM

Toplantılar



- Daily Scrum (Daily Stand Up)
(Günlük Scrum Toplantısı)
- Sprint Planning
(Sprint Planlama)
- Sprint Review
(Sprint İnceleme)
- Sprint Retrospective
(Geriye Dönük Sprint Değerlendirme)

AGILE METHODOLOGY (çevik) SCRUM

Toplantılar / Sprint Planning

Her Sprint'i başlatan toplantıdır.

- Product Owner ve Development ekibinin Sprint'e hangi Ürün İş Listesi Öğelerinin (PBI'ler) dahil edileceğini tartıştığı yerdir.
- Product Owner Sprint'e potansiyel olarak dahil edilmek üzere her PBI'ya öncelik verme hakkına sahip olsa da, Development Team yanıt vermeye, sorunları gündeme getirmeye ve gerekiğinde geri itmeye teşvik edilir.



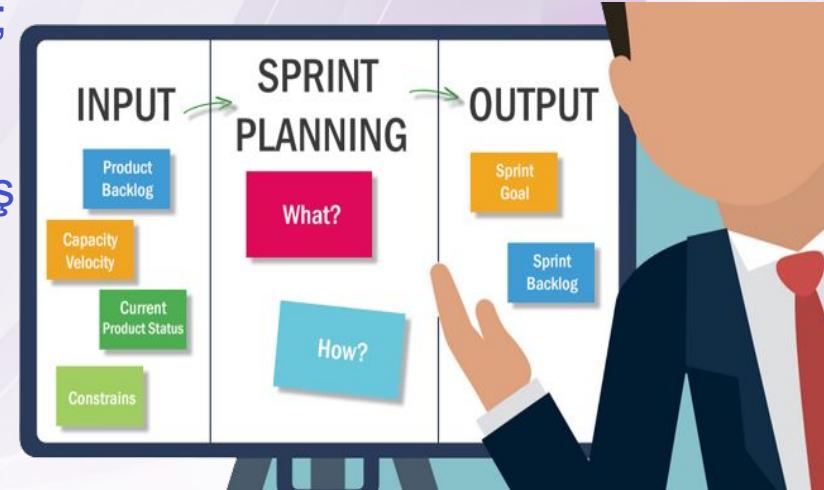
AGILE METHODOLOGY (çevik) SCRUM

Toplantılar / Sprint Planning

Sprint Planlama Toplantisının amacı, herkesin kabul ettiği gerçekçi ve ulaşılabilir bir Sprint Hedefi ve Sprint İş Listesi elde etmektir.

Sprint Planlamada şu sorulara cevap verilir;

- 1) Başlayan Sprint'te task olarak ne teslim edilebilir? (**Ne Yapacağız?**)
- 2) Uygulamayı teslim etmek için gerekli olan iş nasıl başarılıacak? (**Nasıl Yapacağız?**)
(2. madde için gerekirse Sprint süresi içerisinde eski adıyla Grooming yeni adıyla Refinement (detaylandırma) aktivitesi gerçekleştirilebilir.)



AGILE METHODOLOGY (çevik)

SCRUM

Toplantılar / Daily Scrum

Daily Scrum (Daily Stand Up)

Scrum zamanınızı ve kaynaklarınızı verimli bir şekilde kullanmayı amaçlar.

- Yaklaşık 15 dakika kadar sürer. Ayağa kalkmak zorunlu değildir. Ancak, birçok takım bu toplantıyı kısa ve öz tutmak için yararlı bir teknik olarak bulmaktadır.
- Geliştirme Ekibinin oto-kontrol yapması, Sprint hedefine ulaşma yolundaki ilerlemeyi değerlendirmesi ve önümüzdeki 24 saat boyunca faaliyetlerini gözden geçirmesi ve planlaması için bir fırsatır.



AGILE METHODOLOGY (çevik)

SCRUM

Toplantılar / Daily Scrum

- Genelde 'Dün neler yaptık?,
Bugün ne üzerinde çalışacağım?,
Herhangi bir sıkıntı veya engel ile
karşılaştım mı?' soruları cevaplanır.
- Sorulara verilecek cevaplar açık ve
net olmalıdır. 7 kişiden oluşan bir
Development Team'i düşünürsek bir
üyenin konuşabileceği 128 saniyesi
olacaktır. Her soruya cevap verilecek
sure ise 43 saniyedir.



AGILE METHODOLOGY (çevik)

SCRUM

Toplantılar / Daily Scrum

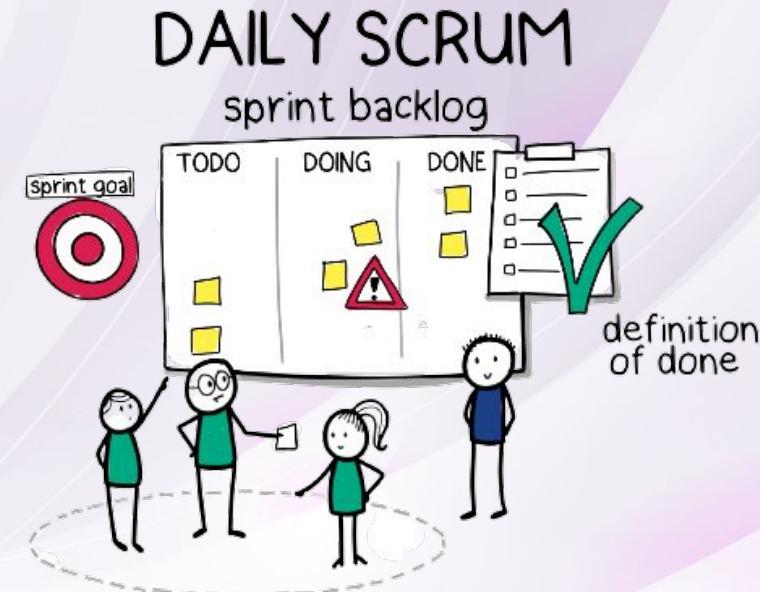
- **İyi Örnek :**

Dün Müşteri Tanımlama Ekran testini yaptım.
Herhangi bir sorunla karşılaşmadım.

Bugün ise Müşteri Tanımlama ekranının,
bireysel kredi planı çıkarma ekranı
arkasındaki bağlantıyı test edeceğim.
Eğer vaktim kalırsa vadeli müşteri hesap
ekranını test edeceğim.

Sonra da Zeynep ile code review yapmayı
planlıyorum.

Önümde DB bağlantı hatası var, sizlerden de
karşılaşan varsa beraber bakabilir miyiz?

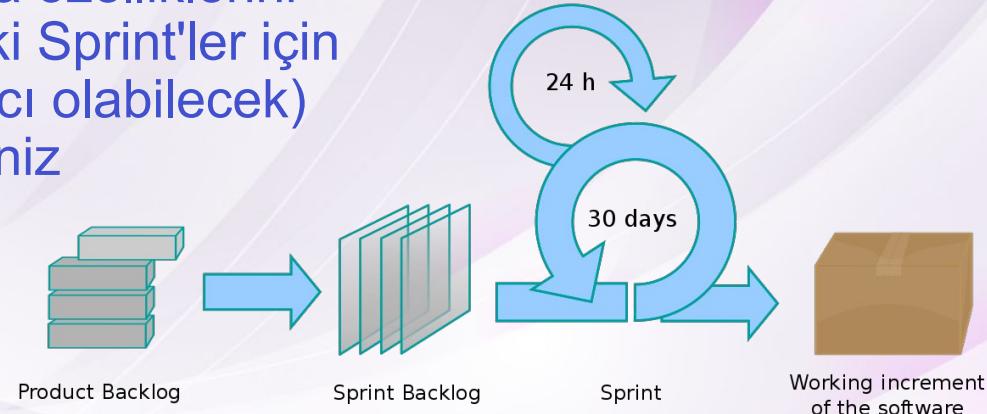


AGILE METHODOLOGY (çevik)

SCRUM

Toplantılar / Sprint Review

- Genellikle Sprint'in son gününde yapılır ve Stakeholder'lara (müşteriler, yönetim ve ilgilenen diğer herkes) "done" (tamamlanmış) yapılan işi gösterme fırsatı verir.
- Sprint sırasında üretilen çalışma özelliklerini göstermenin yanı sıra, gelecekteki Sprint'ler için (çalışmayı yönlendirmeye yardımcı olabilecek) Product Backlog'a ekleyebileceğiniz yararlı geri bildirimler de alıyorsunuz.
- Sprint Review Toplantısı'nın sahibi Product Owner'dır.



AGILE METHODOLOGY (çevik)

SCRUM

Toplantılar / Sprint Review

- Burada yapılması gereken şey Stakeholders'a “ürünün onların ürünü” olduğunu vurgulamaktır. Onlara anlatacaklarınızla, göstereceklerinizle sizden istediklerinin örtüşüp örtüşmediğinin bu toplantıda anlaşılacağını anlatmaktadır.
- Stakeholders çalışan ürünü görerek bu ürüne eklenmesi istedikleri yeni özellikler ya da değiştirilmesini istedikleri özelliklerin hangi niteliğe, görselliğe, yetkinliğe sahip olması gerektiğini yüz yüze, birinci ağızdan söyleyebilirler.



AGILE METHODOLOGY (çevik)

SCRUM

Toplantılar / Sprint Retrospective

- Retrospective toplantıları bir sonraki Sprint'te iş süreçlerini iyileştirmek için geçmiş Sprint'in incelendiği ve “**Nasıl daha iyi performans gösterebiliriz?**” sorusuna cevap aranan toplantılardır. Bu toplantıya **Development Team, Scrum Master ve Product Owner** katılır.
- Sprint'teki son toplantıdır ve “Sprint Review” toplantısının hemen ardından yapılır.
- Scrum ekibi gelecekteki Sprint'ler için nelerin geliştirilebileceğini ve nasıl yapmaları gerektiğini gözden geçirir.

