



# SDLC

## SOFTWARE DEVELOPMENT LIFE CYCLE

(Yazılım Geliştirme Yaşam Döngüsü)

1. Ders



# Bu derste neler öğreneceğiz?

- 1- SDLC NEDİR?**
- 2- SDLC AŞAMALARI NELERDİR?**
- 3- SDLC TAKIM ÜYELERİ VE GÖREVLERİ**



# HARDWARE & SOFTWARE

Hardware (Donanım)



Software (Yazılım)



## DONANIM

MONITOR, KASA,  
KLAVYE, MOUSE, HDD, EKRAN  
KARTI...

Bilgisayari fiziken olusturan ve elle  
dokunabildigimiz tum bilgisayar  
parcalari

## YAZILIM,

değişik ve çeşitli görevler yapma  
amaçlı tasarlanmış  
DONANIMlarin birbirleriyle  
haberleşebilmesini ve uyumunu  
sağlayarak görevlerini ya da  
kullanılabilirliklerini geliştirmeye  
yarayan makine komutlarıdır



# SDLC NEDİR

**SDLC** : Software Development Life Cycle (Yazılım Geliştirme Yasam Dongusu)

**SDLC** : Yuksek Kaliteli (**High-Quality**) ve Kullanıcı Taleplerini (**User Expectation**) Karsilayan Yazılım Ürünlerini geliştirmek için planlanan yazılım geliştirme sürecidir(**Process**).

Yazılım ürünlerinin nasıl geliştirilmesi gerektiğini veya nasıl iyileştirilmesi gerektiğini anlatan ayrıntılı bir plan içerir.



# SDLC NEDİR

## Yuksek Kalite(High-Quality)



**Piyasa degeri 33 Milyar \$**

DİJİTAL 

**Güvenlik açığı ile dünya çapında ses getiren Twitter'ın piyasa değeri düştü**

Twitter'ın hisseleri borsa kapanışının ardından yüzde 4 düşüş gösterdi.

Ekonomi 15.03.2019 Gazete

ABONE OL

Google News

**Boeing 650 milyar dolarla çakıldı! İptaller peş peşe geliyor...**

ABD'nin de katılması ile tüm dünyada Boeing 737 Max'ların uçuşu durduruldu. 4 bin 350 adet daha Max siparişi bulunan Boeing, 650 milyar dolardan olacak.



**BOEING İRTİFA KAYBEDİYOR!**

Davalar ve iptaller peş peşe geldi!  
Zarar dudak uçuklatıyor...



# SDLC NEDİR

## Kullanıcı Talepleri (User Expectation)

2002 Yılı İnternet Tarayıcı (Browser) Kullanım  
Oranları İstatistik Tablosu

2002	IE6	IE5	AOL	Netscape 3	Netscape 5	Netscape 4	IE 4
November	53.5%	29.9%	5.2%	1.1%	4.9%	2.0%	

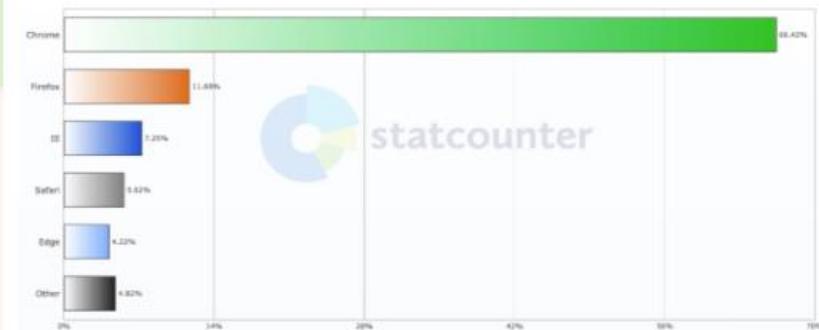
2008 Yılı İnternet Tarayıcı (Browser) Kullanım  
Oranları İstatistik Tablosu

2008	IE7	IE6	IE5	Firefox	Chrome	Safari	Opera
December	26.1%	19.6%	44.4%	3.6%	2.7%	2.4%	



2017 - 2018 Dünya Geneli Masaüstü Bilgisayar  
Internet Tarayıcı İstatistikleri

- Chrome %66.42
- Firefox %11.68
- Internet Explorer %7.25





# NEDEN SDLC?



[Fotoğraf: AA]

ABONE OL

Google News

Takip et: @trhaber

Sosyal medya platformları Whatsapp, Instagram ve Facebook'a dünya genelinde yaşanan erişim sorunu devam ediyor. Facebook'un hisseleri yüzde 5'in üzerinde düştü.

## Yemek Sepeti çöktü: Hem internet sitesine, hem de uygulamaya ulaşılamıyor

18:37 10.01.2021 (güncellendi: 04:48 11.01.2021)



© Fotoğraf

Abone ol

Google News

Hem [yemeksepeti.com](http://yemeksepeti.com) hem Yemek Sepeti iOS ve Android uygulamalarına erişim sağlanamadı. Firmadan henüz bir açıklama yapılmadı.



# NEDEN SDLC?

**Ünlü Alışveriş Sitesi Amazon Çöktü,  
Ürünler 3,6 Kuruştan Satıldı**



Haberler.Com - Haberler | Güncel  
16 Aralık 2014 Salı 13:02

**ABD'li online alışveriş devi Amazon.com'da yaşanan bir hacker skandalı nedeniyle 20 bine yakın ürün, 3,6 kuruştan satıldı.**

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

**Akbank'tan yeni açıklama!  
Akbank sistem arızası düzeldi  
mi, sorun devam ediyor mu?**

Akbank kullanıcıları salı gününden bu yana mobil ve internet bankacılığında sorun yaşıyordu. Öte yandan ATM ve pos cihazlarında da hata olduğu öne sürüldü. 'Oturum kapandı' sorunu ile karşılaşan vatandaşlar Akbank internet bankacılığı çöktü mü, siber saldırıya mı uğradı sorularını gündeme getirdi. Akbank'tan konu hakkında açıklama geldi.

• 09 Temmuz 2021 - 00:42 • Son Güncelleme: 09 Temmuz 2021 - 00:42





# Software Hatalarının Sonuçları

- Therac-25 cihazı, 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 hata, Therac-20'nin kodlarında bulunan ancak fark edilemeyen hatanın, Therac-25'te ortayamasına sebep oldu.
- Bu hata (bug) yüzünden yaklaşık 5 hastanın ağır dozda radyasyon sebebiyle hayatını kaybettiği raporlandı.

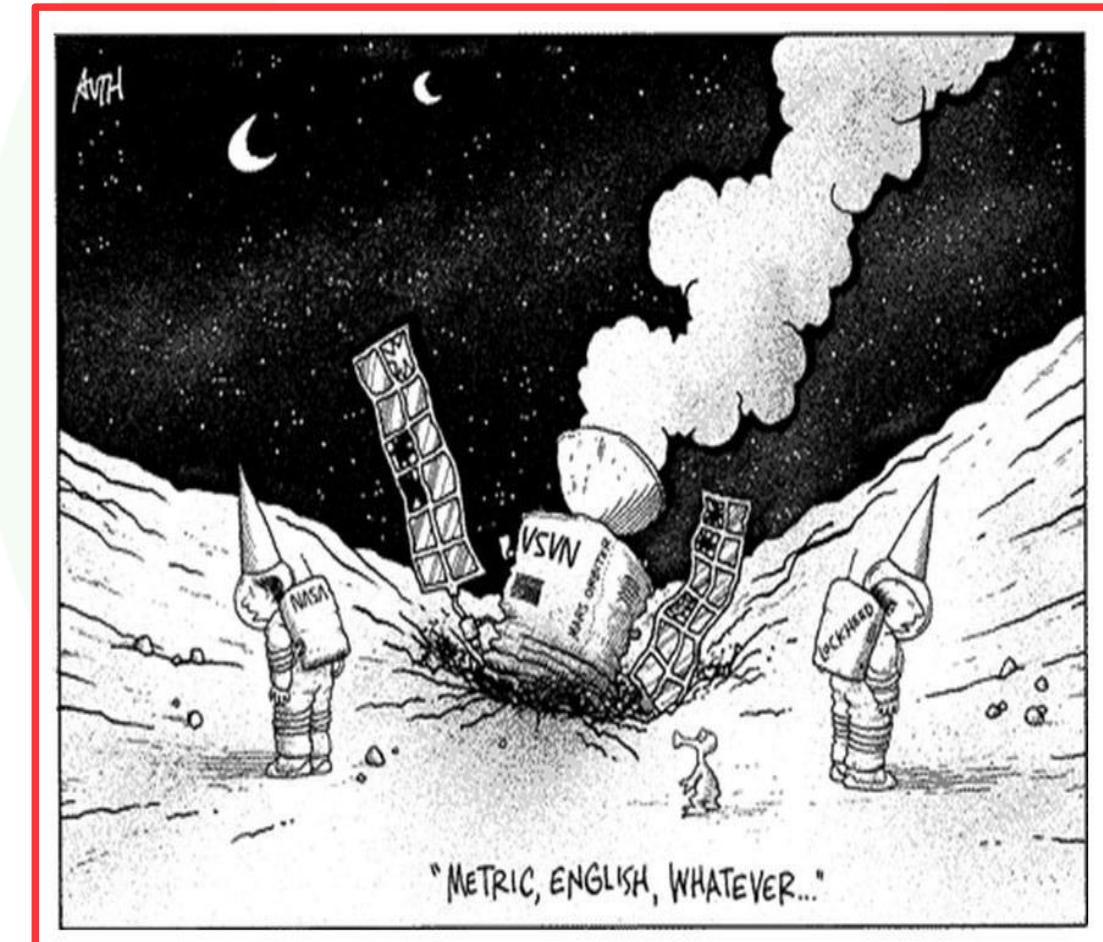




# Software Hatalarının Sonuçları

Mars Climate Orbiter Hatası (23 Eylül 1999):

- Gezegenler arası ilk iklim uydusu olarak 1997'de fırlatıldı.
- Mars Orbiter, 1999'da Mars'ın yörüngeinde kayboldu.
- Kazanın yazılımda kullanılan İngiliz ölçü birimlerinin metrik sisteme yanlış çevrilmesinden kaynaklandığı belirtildi.
- NASA'da bir ekip hesaplarında İngiliz ölçü birimini (inç) kullanırken, projeye katılan diğer ekiple metrik (cm) sistemi kullanmıştı.
- 125 milyon dolarlık uydudan yörüngeye sabitlenmeye çalışırken Mars'a olması gerekenden fazla yaklaşarak imha olduğu düşünülüyor.





# SDLC'nin Faydaları

The screenshot shows a Twitter interface. On the left is a sidebar with navigation links: Home, Explore, Notifications, Messages, Bookmarks, Lists, Profile, and More. The main area displays the 'Home' feed. At the top of the feed is a tweet from 'Engadget UK' (@EngadgetUK) about Netflix editing a scene from '13 Reasons Why'. Below it is a large thumbnail image of a woman's face. Further down the feed is a tweet from 'Android Police' (@AndroidPolice) about the Nokia 2.2 phone. To the right of the feed, there is a 'Trends for you' sidebar listing several trending topics in the United Kingdom, such as Michael Chopra, Gareth Bale, #NewTwitter, #13ReasonsWhy, and Britain First, each with a small thumbnail and the number of tweets.

- 1) Projenin takibini ve kontrolunu saglar
- 2) Tum Planlama ve Process'in yatirimcilar tarafindan gorulebilmesine imkan tanir
- 3) Yapilan planlama ve toplantılarla Projenin Olusturulma ve gelistirme (Development) hizini artirir
- 4) Tum ekibin iletisimini guclendirir
- 5) Projenin risklerini azaltir



# SDLC'nin Faydaları

- Projenin takibini ve kontrolünü sağlar.
- Tüm planlama ve process'in, yatırımcılar ve işveren tarafından görülebilmesine imkan tanır.
- Yapılan planlama ve toplantılar, projenin oluşturulma ve geliştirme hızını artırır.
- Tüm ekibin iletişimini güçlendirir.
- Projenin risklerini azaltır.
- Doğru yapılan SDLC, en yüksek düzeyde yönetim kontrolüne ve dokümantasyona izin verir.
- Çalışanlar neyi, neden yapmaları gerektiğini anlarlar.
- Tüm taraflar hedef üzerinde önceden hemfikirdir ve bu hedefe ulaşmak için net bir plan görür.
- Tüm taraflar, gerekli maliyetleri ve kaynakları görebilirler.





# SOFTWARE DEVELOPMENT PHASES

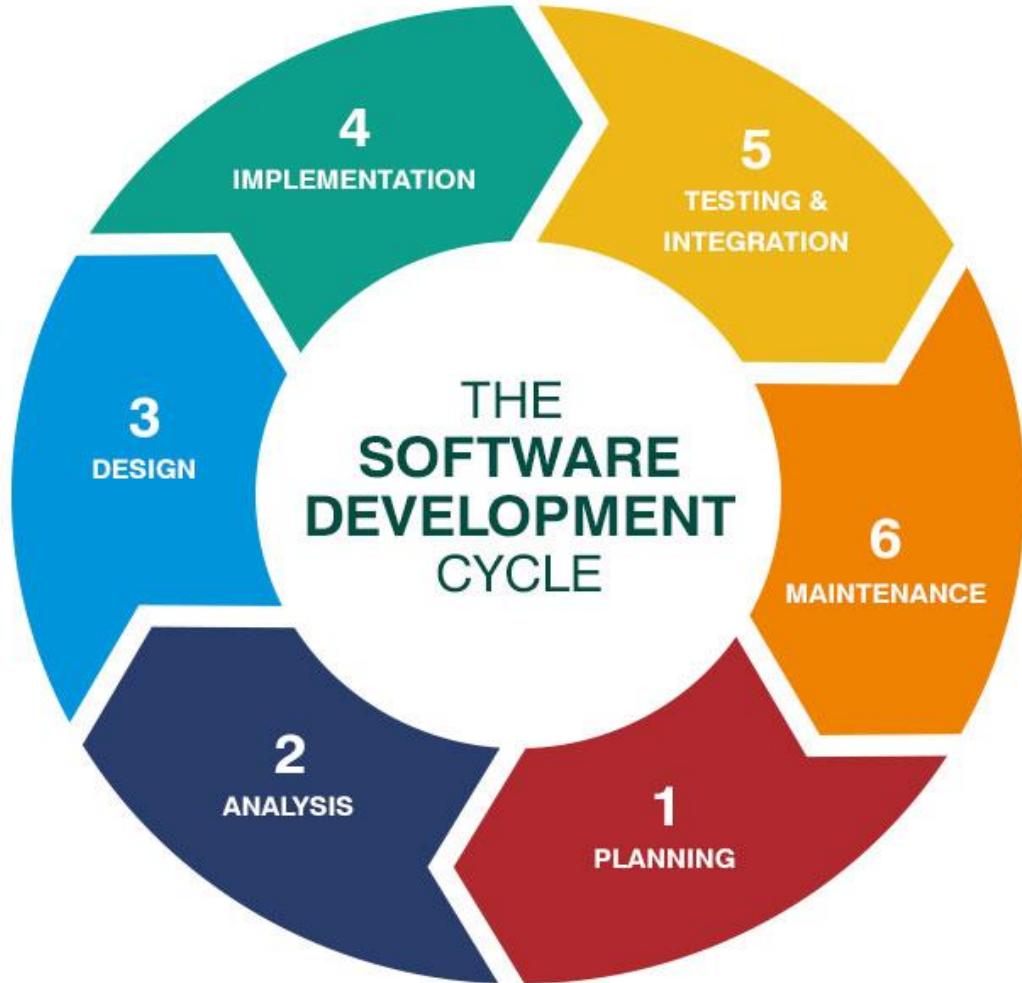
---

# YAZILIM GELISTIRME ASAMALARI

- 1-
- 2-
- 3-
- 4-
- 5-
- 6-



# SDLC AŞAMALARI





# SOFTWARE DEVELOPMENT PHASES (Yazılım Geliştirme Aşamaları)

- 1) Planning and Requirement Analysis  
(Planlama ve İhtiyaç Analizi)
- 2) Defining Requirements  
(Gereksinimleri Tanımlama)
- 3) Designing the product architecture  
(Ürün dizaynını 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)



# SOFTWARE DEVELOPMENT PHASES (Yazılım Geliştirme Aşamaları)

## 1) Planning and Requirement Analysis (Planlama ve İhtiyaç Analizi)

- İhtiyaç analizi SDLC'nin en önemli ve temel aşamasıdır.
- Müşteriden gelen fikirler de göz önünde bulundurularak ekibin kıdemli üyeleri (expert) tarafından gerçekleştirilir.
- 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 yaklaşımlar planlanır.



# SOFTWARE DEVELOPMENT PHASES (Yazılım Geliştirme Aşamaları)

## 2) Defining Requirements (Gereksinimleri Tanımlama)

İhtiyaç analizi yapıldıktan sonraki adım, ürün gereksinimlerini açıkça tanımlamak ve belgelendirmektir (dokumante etmek) Stakeholder / işletmeciden onay alınır.

Bu proje yaşam döngüsü boyunca tasarlanacak ve geliştirilecek tüm ürün gereksinimlerini içeren

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

**FRD (Functional Requirement Document)**

Teknik İşlev İhtiyacları Dokumani hazırlanır

FRD ve BRD en küçük User Case lere kadar hazırlanır



# SOFTWARE DEVELOPMENT PHASES (Yazılım Geliştirme Aşamaları)

## 3) Designing the product architecture    (Ürün dizaynını tasarlama)

BRD (Business Requirement Document) Dizaynırların geliştirilecek ürün için en iyi dizaynla ortaya çıkacakları referanstır.

BRD'de belirtilen gereksinimlere dayanarak, ürün mimarisi için genellikle birden fazla tasarım yaklaşımı taslağı olusturulur.

**DDS(Design Document Specification):** Tasarım spesifikasyonu, bir ürün veya süreçle ilgili noktaların bir listesini sağlayan ayrıntılı bir belgedir.



# SOFTWARE DEVELOPMENT PHASES (Yazılım Geliştirme Aşamaları)

## 4) Building or Developing the Product (Ürünü oluşturma veya geliştirme)

SDLC'nin bu aşamasında gerçek gelişme başlar ve ürün inşa edilir.

- Yazılımcılar (Developers), kuruluşları tarafından tanımlanan kodlama yönergelerine uymak zorundadır.
- Kodlama için FRD baz alınarak
- Developerlar gereken Funcionality'leri oluştururlar.
- Kodlama için C++, Python, Java, .Net vs. gibi farklı üst düzey programlama dilleri kullanılır.



# SOFTWARE DEVELOPMENT PHASES (Yazılım Geliştirme Aşamaları)

## 5) Testing the Product (Ürünü test etme)

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

Ürün iş beklentilerini de karşılamalıdır (requirement specifications)



# SOFTWARE DEVELOPMENT PHASES (Yazılım Geliştirme Aşamaları)

## 6) Deployment in the Market and Maintenance (Ürünü pazarlama ve bakım)

- Ürün test edildikten ve onaylandıktan sonra (hazır olduğunda), resmi olarak uygun görülen şekilde release edilir. (piyasaya sürülür).
- Ürün piyasaya sunulduktan sonra mevcut müşteri tabanı için bakımı yapılır.
- Musteriden (End-User) gelen feedbackler ve Teknolojik Gelişmeler ile ihtiyaçlar yeniden belirlenir ve döngü yeniden başlatılır



# SDLC TEAM

Proje geliştirme sürecinde roller nelerdir?

- 1-
- 2-
- 3-
- 4-
- 5-
- 6-
- 7-



# SDLC TEAM

- 1) Project Manager (PM)  
Proje Yöneticisi
- 2) Business Analyst (BA)  
İş Analisti
- 3) Geliştirme Takımı  
Designer  
Data Analyst  
Developer  
Tester





# PROJECT MANAGER (PM)

Project manager: Proje yöneticisi, takımındaki herkesin rolünü bilmesini ve yerine getirmesini ve bu rollerin gerçekleştirileceği inancına göre hareket etmesini 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 karışımlıkları (conflicts) önler (Kriz yönetiminden sorumludur)
- Görev dağılımını yönetir.



# BUSINESS ANALYST (BA)

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 iletir. İş sorunlarına teknik çözümler geliştirmek için çalışır.

- Business sorunları ve teknoloji çözümleri arasında bir köprü vazifesi görür
- Requirements (Gereksinimler) yönetimini ve iletişimini sağlar.
- Alınan kararların anlaşılır bir dile dökülmesini sağlar.
- **Business Requirement Document (BRD)** oluşturur.

(alınan tüm kararların ve gereksinimlerin dokümü)

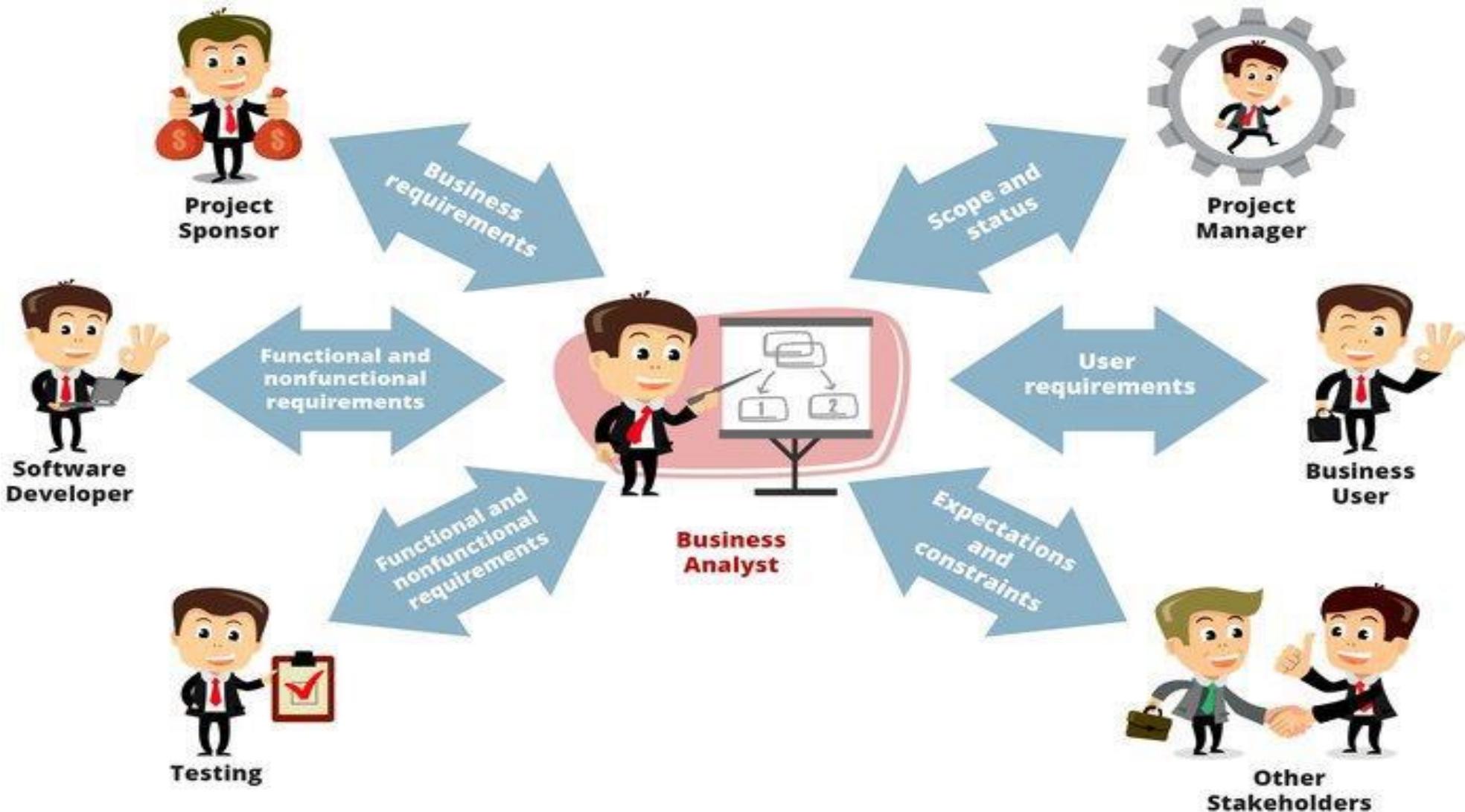
- **Functional Requirement Document (FRD)** oluşturur.

(yazılımı yapılacak olan bütün maddelerin dokümü)

- Yeteri kadar Functional Requirement toplandıktan sonra use cases oluşturur
- Akış şemasını oluşturur



# BUSINESS ANALYST (BA)





# DEVELOPER (DEV, Yazılımcı)

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

## Sorumluluklar:

- Kendilerine aktarılan **software requirement** dokümanını toparlar ve gereken application ve programın oluşumunu sağlar.
- Beklentileri ve gereksinimleri (costumer requirement) karşılayacak yüksek kalitede (**High Quality**) code yazarlar.
- Software dokümanını oluşturur ve önceki dökumanları günceller.





# Veri Analisti

Teknolojinin gelişmesiyle şirketlerin kendi bünyelerinde veri tabanı oluşturmaya başlamıştır. Bu veriler kendi başlarına anlam ifade etmezler. Bu verileri isteğiniz doğrultusunda bilgiye çevirmek veri analistinin görevidir. Ham veriyi anlamlı hale getirmek veri analistinin en temel görevidir.





# Veri Analisti

Bu veriler şirketlerin büyüklüğüne göre değişmektedir. Bir şirkette milyonlarca veri olabilirken başka bir şirkette daha az veri olabilir. Bu veriler muhasebe, üretim, satış, pazarlama gibi şirket içindeki tüm hareketleri kapsar. Şirketler bu data içinde bulunan verileri anlamlı ve sade bir şekilde geri dönüştürülmesi için veri analisti ihtiyacına gerek duyarlar. Veri analisti bulunduğu şirket bünyesindeki sorunları bulup eldeki bilgilerle yani verilerle çözüme ulaştırmaya çalışır. Raporları oluştururken yeni veriler, yeni yöntemler keşfetmesi gerekmektedir. Veri analisti; şirketlerin mali kaynaklarını araştırma, şirket ihtiyaçlarını belirleme, şirketler arası iletişimini sağlama görevlerini üstlenmektedir.





# Designer



Designer (Tasarımcı), bir ürünün, sistemin veya arayüzün son kullanıcısı ile buluşmadan önce, kullanıcı tarafından nasıl kullanılacağını çeşitli araştırma yöntemleri kullanarak ön görmeye çalışan ve son kullanıcının ürünü olabilecek en kolay seviyede kullanabilmesini sağlayıp kullanıcı için 'iyi deneyim' yaratmayı amaç edinen bir meslek dalıdır.

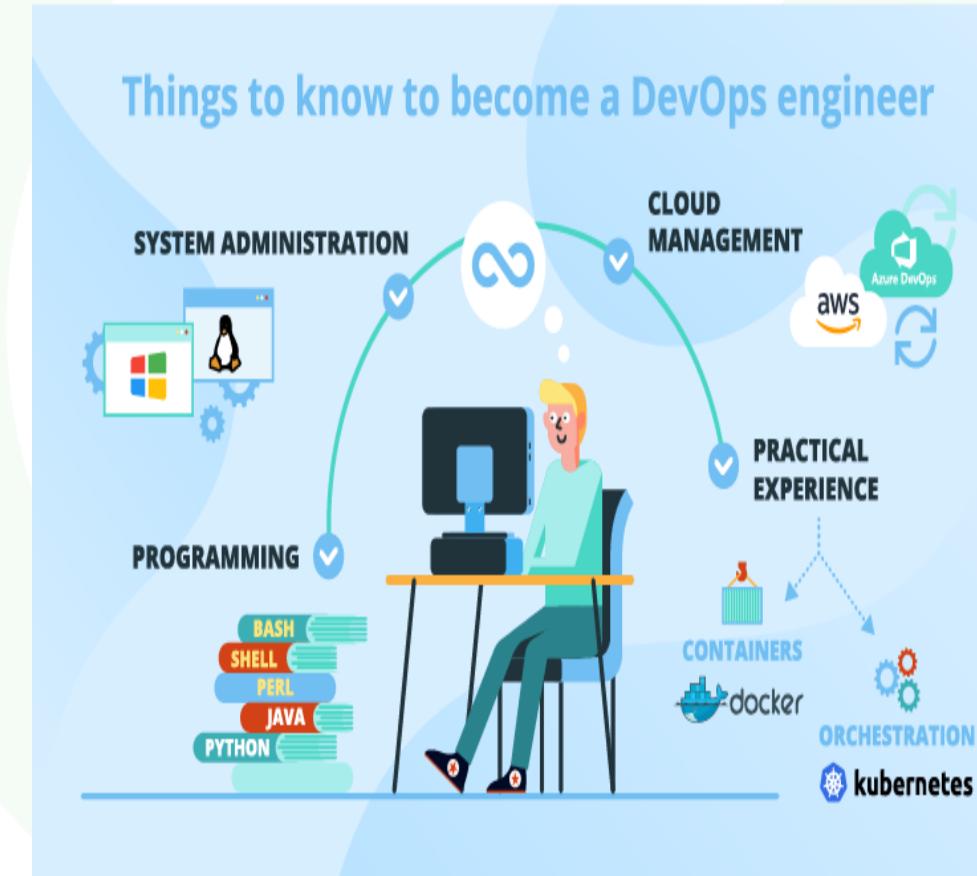
Empati yeteneği güçlü, araştırma yapmayı seven ve ekip çalışmalarına yatkın kişiler için uygun olabilecek bir iş tanımıdır.



# DevOps Engineer

**DevOps mühendisi** geliştiricilerle IT ekibi arasındaki boşluğu doldurmaya yarayan geliştirici (Dev) ve sistem yöneticisi profesyonellerinin ("Op" -erations) bir birleşimidir. DevOps mühendisi çok çeşitli açık kaynaklı teknolojiler ve araçlar etrafında bilgi derinliğini ve yılların uygulamalı deneyimini kullanır.

Ayrıca **DevOps mühendisi** yazılım geliştirme yaşam döngüsünden ve kodlama hatalarını en aza indiren, daha kaliteli ürünlerin hızlı bir şekilde kullanılmasını kolaylaştıran CI / CD veri hattını oluşturmak için kullanılan otomasyon araçlarından anlayan kişidir.





# DevOps Engineer

DevOps mühendisinin temel amacı, yalnızca ürünün kalitesini daha fazla artırmak değil, aynı zamanda Dev ve Ops ekibinin işbirliğini de artırmak ve böylece organizasyon içindeki iş akışının aynı anda daha verimli olmasını sağlamaktır. DevOps mühendisi genel olarak dahili sistemleri entegre etmek için yazılım ve yazılım düzeltmeleri geliştirirler. Kod kalitesini sağlar, kod güncellemelerini test eder, dağıtır ve sunucuların sağlığını, kararlılığını izler.

Bunlarla birlikte bir DevOps mühendisinin üstlenebileceği birçok rol vardır

- ▶ Yükseltmeleri ve düzeltmeleri uygulamak için bulut (AWS, Azure, GCP) bilgi işlem becerilerini uygulamak.
- ▶ Kullanıcı geri bildirimlerine dayanarak yazılım entegrasyonları tasarlamak, geliştirmek ve uygulamak.
- ▶ Üretim sorunlarını gidermek ve kod dağıtımını kolaylaştırmak için geliştirme ekibiyle koordineli çalışmak.
- ▶ Otomasyon araçlarını ve çerçevelerini (CI / CD veri hatları) uygulamak.
- ▶ Uygulamalarda ve projelerin zamanında tamamlanmasında belirgin bir iyileşme sağlamak için kodu analiz etmek ve geliştirme ekiplerine ayrıntılı incelemeler iletmek.
- ▶ Şirketin mühendislik araçlarını, sistemlerini, prosedürlerini ve veri güvenliğini geliştirmek için ekip üyeleriyle işbirliği yapmak.
- ▶ Şirketin bilgi işlem mimarisini optimize etmek.
- ▶ Güvenlik, performans ve kullanılabilirlik için sistem testleri yapmak.
- ▶ Tasarım ve sorun giderme dokümantasyonu geliştirmek, bakımını yapmak.
- ▶ Yeni özelliklerin ve ürünlerin gerektirdiği altyapı değişikliklerinde geliştiricilere sunulan hizmetlerin kullanılabilirliğini sağlamak.



# QUALITY ANALYST (QA, 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.
- Her hangi bir organizasyonun ürünlerini ve hizmetini 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 koyabilsevler (minimum seviyede bug).
- Testing yapılmasıının amacı herhangi bir application'da oluşabilecek hataların ortaya çıkarılmasıdır.





# SOFTWARE DEVELOPMENT PHASES (Yazılım Geliştirme Aşamaları)

## 6 PHASES OF THE SOFTWARE DEVELOPMENT LIFE CYCLE



CTO => Chief Technology Officer

UX => User Experience Designer (Kullanıcı Deneyim Tasarımcısı)

UI => User Interface Designer (Kullanıcı Arayüz Tasarımcısı)



# SDLC

## SOFTWARE DEVELOPMENT LIFE CYCLE

(Yazılım Geliştirme Yaşam Döngüsü)

2. Ders



# YAZILIM GELİŞTİRMEDE KULLANILAN MODELLER

1968 yılından itibaren; yazılım geliştirme, uygun, etkili, güvenilir yazılımları mümkün olduğunda az bir maliyetle elde edebilme yolundaki çalışmaların artmasıyla **YAZILIM MÜHENDİSLİĞİ** olarak adlandırılan disiplin kurulmuştur.



Yazılım Mühendisliği'nin temel hedefi; sistematik ve organize bir yaklaşımı çalışma alanlarına adapte etmek ve bir problemin çözümü için uygun tüm araç ve teknikleri kullanmaktır. Yazılım ürünü ise, bir sistem projesi olarak, bir ekip çalışmasıyla, belli bir sürede hazırlanmaktadır. Yazılım ürünlerinin süresi ve bütçesi; projeye bağlı olarak değişiklik göstermektedir.



# YAZILIM GELİŞTİRMEDE KULLANILAN MODELLER

Yazılım geliştirme modeli; yazılımın gerçekleştirilebilmesi için gerekli stratejiyi ifade eder ki bu strateji bir dizi aktiviteyi ve olayları içermektedir. Ancak seçilen her modelin kendine özgü avantaj ve dezavantajları bulunmaktadır. Dikkat edilmesi gereken nokta çözüm istenen ürüne ve sürece uygun modelin seçilebilmesidir.

En klasik yazılım geliştirme modeli metodolojisi, şelale modelidir. **Şelale modeli dışında, v modeli, prototip modelleme, spiral model, çevik geliştirme** gibi pek çok model bulunmaktadır.



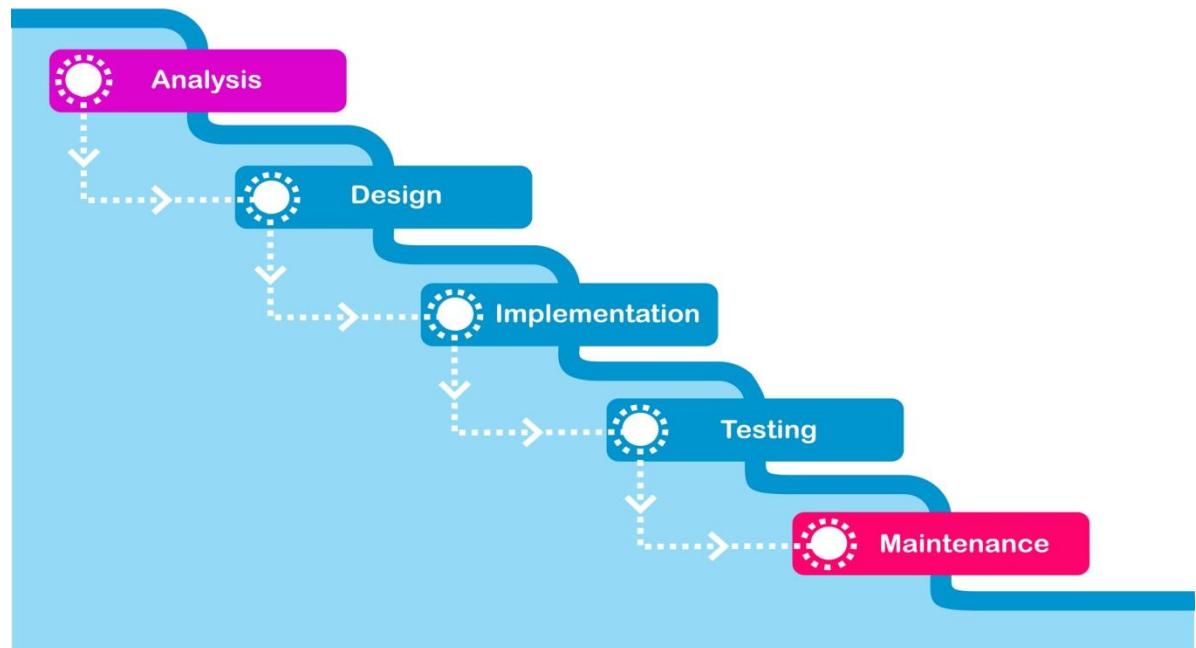
# YAZILIM GELİŞTİRMEDE KULLANILAN MODELLER

Yazılım Geliştirme Metodolojilerinde en yaygın kullanılanlar;

- 1) WATERFALL MODEL(Şelale Modeli)
- 2) V MODEL
- 3) SPIRAL MODEL
- 4) PROTOTİP MODEL
- 5) AGILE METHODOLGY (Çevik Metodoloji)



# WATERFALL MODEL(Şelale Modeli)

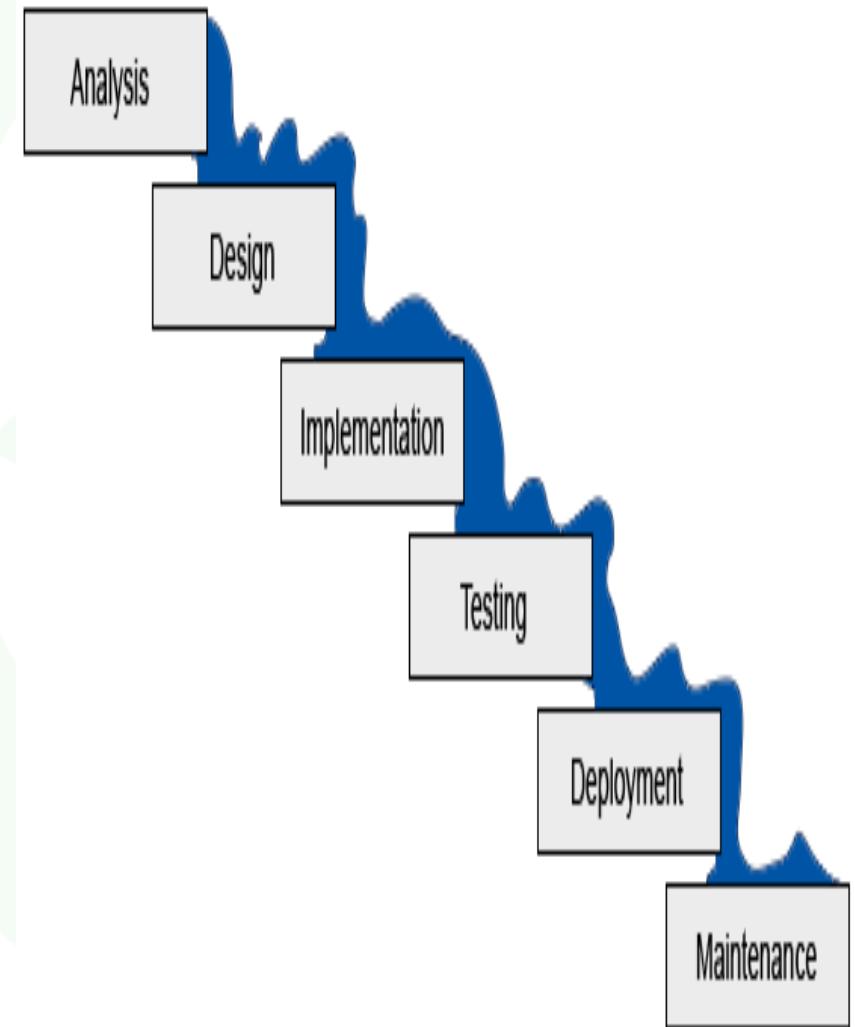


- Şelale modeli (Waterfall) proje yönetim süreci; analiz, tasarım, yazılım, test, yayın gibi fazlardan oluşur.
- Geleneksel bir yöntemdir; süreçler tipki bir şelale gibi yukarıdan aşağıya doğrusal olarak işler.
- Bir faz tamamlanıp yenisine geçildiğinde, bir önceki faza geri dönülmez.
- Proje sahibi, proje tamamlandıktan sonra ürünü görebilir.



# WATERFALL MODEL(Şelale Modeli)

- Şelale modeli, analiz adımı ile başlar. Analiz adımdında, tüm yazılım gereksinimleri net bir şekilde belirlenerek analiz dokümanı üretilir.
- Daha sonra, tasarım adımdında; yazılımın arayüz, veritabanı, sınıf vb. tasarımları yapılarak tasarım dökümanı üretilir.
- Bir sonraki kodlama adımda yazılım; analiz ve tasarım dokümanlarında belirtilen şekilde kodlanır.
- Test adımda; 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 hatası bulunamaz ise, entegrasyon adımına geçilir ve yazılım, canlı ortama entegre edilerek müşterinin kullanımına açılır.





# WATERFALL MODEL(Şelale Modeli)

## AVANTAJLARI

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

## DEZAVANTAJLARI

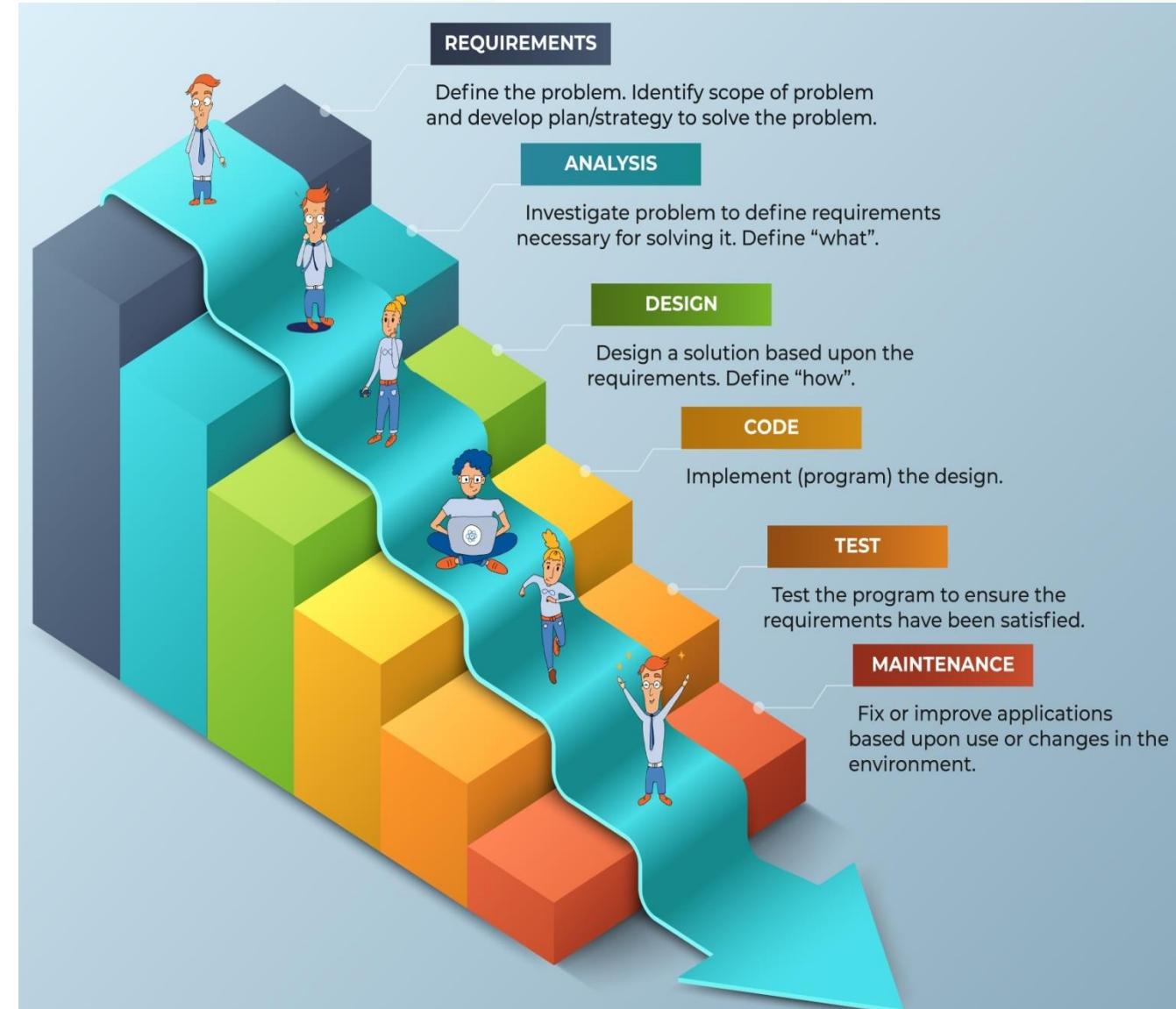
- Değişim ve yenilik zordur.
- Müşteri öngörü ve önerileri önemsenmez.
- Projenin bitimine kadar çalışan ürün yok.
- Beklenmedik riskleri kolayca ele alamıyor.





# WATERFALL MODEL(Şelale Modeli)

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.





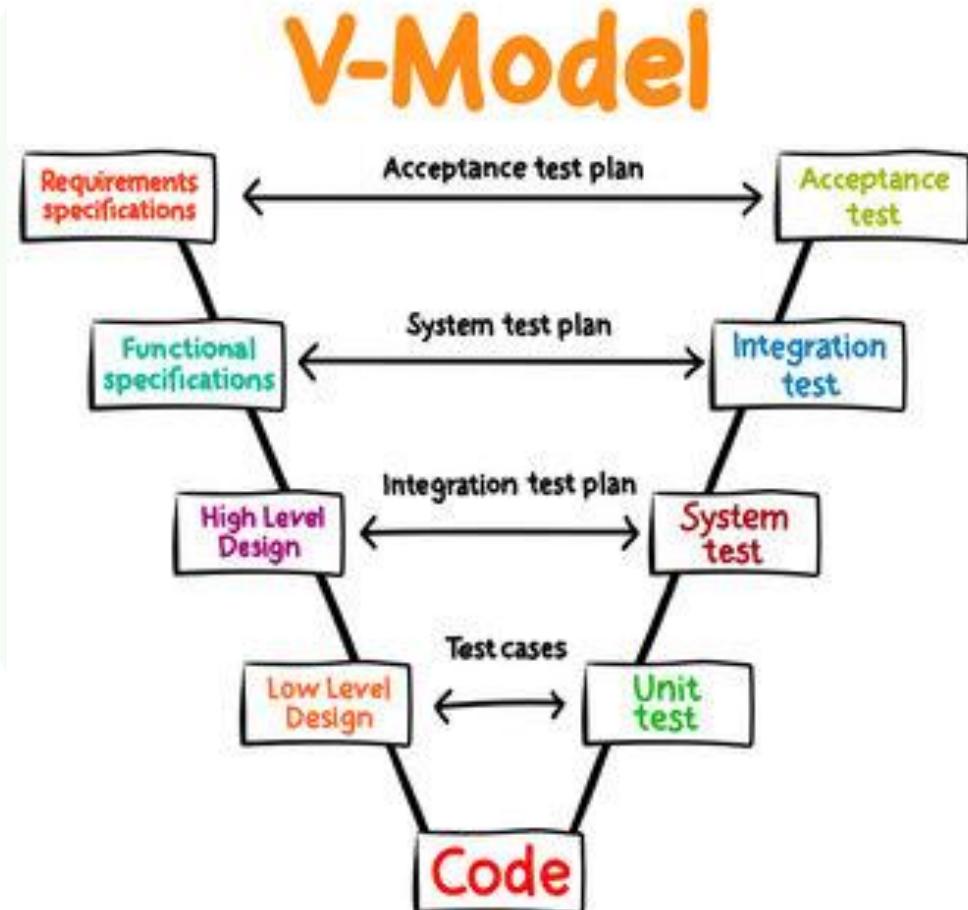
# WATERFALL MODEL(Şelale Modeli)

- 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.
- 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.



## V MODEL

V modeli, **Doğrulama(verification)** ve **Geçerleme (Onaylama-validation)** modeli anlamına gelir. Tıpkı şelale modelinde olduğu gibi yazılım yaşam döngüsü adımları V-şeklinde sıralı bir şekilde uygulanır. Bu modelde de her aşama bir sonraki aşama başlamadan önce tamamlanmalıdır. V-Modelinin en temel özelliği “ürünün test edilmesi kendisine karşılık gelen geliştirme aşamasına paralel olarak planlanmaktadır.”

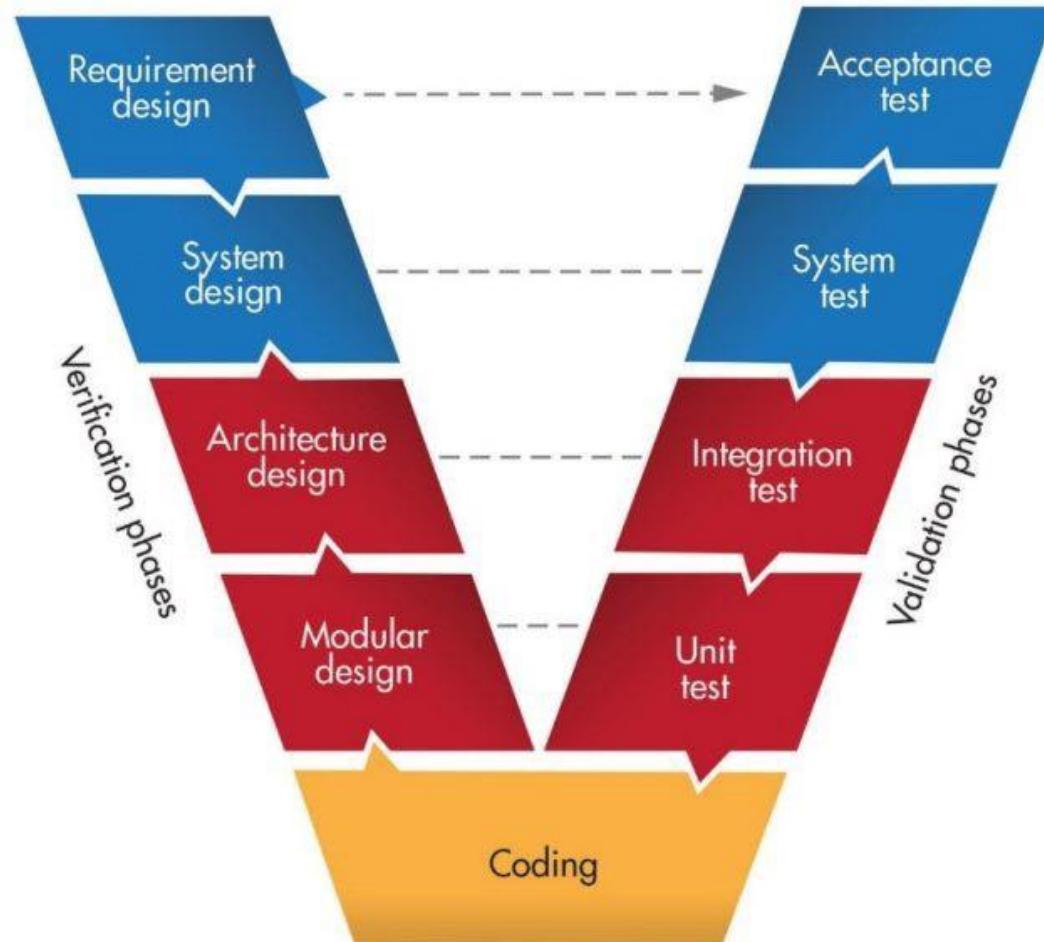




# V MODEL

## V-modelinin avantajları:

- Basit ve kullanımı kolaydır.
- Planlama ve test tasarıımı gibi test faaliyetleri kodlamadan önce gerçekleştirildiği için proje içerisinde çok zaman kazandırır. Bu nedenle şelale modeline göre daha yüksek başarı şansı vardır.
- Hataların bulunması erken aşamada olur.
- Hataların bir sonraki aşamaya geçmesi önlenir.





## V MODEL

### **V modelinin dezavantajları:**

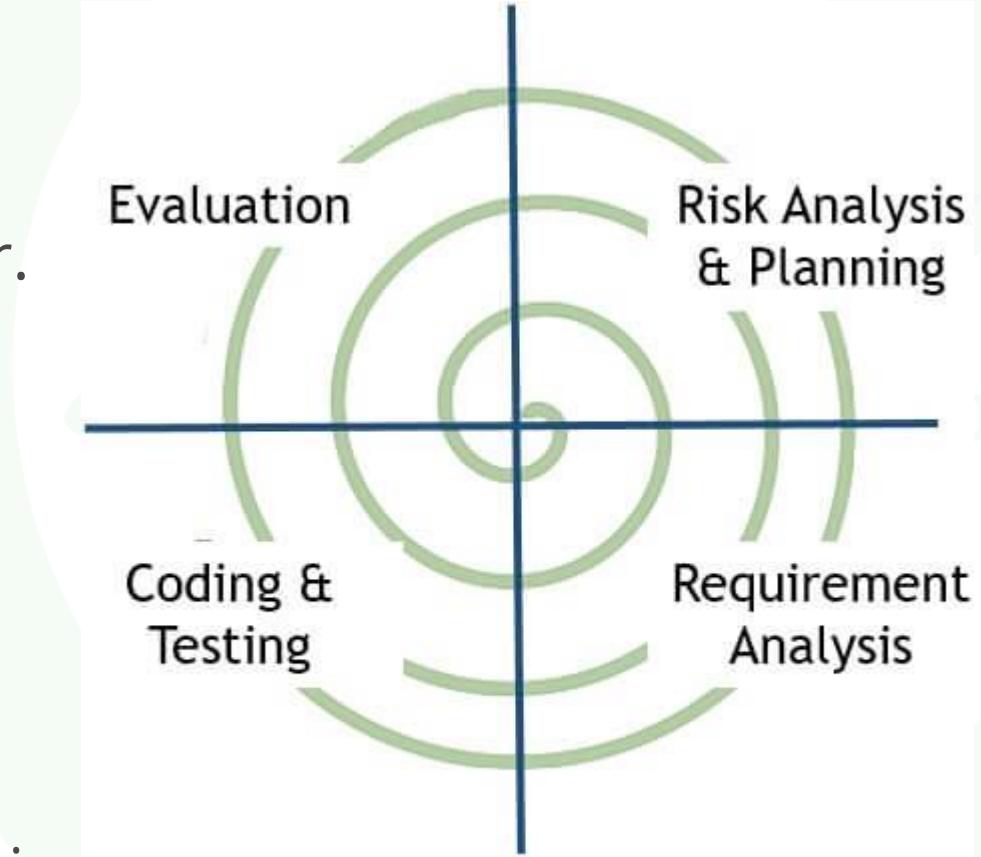
- Uygulama şekli oldukça katı, kesin kurallara bağlıdır.
- Yazılım şelalede olduğu gibi geliştirme aşamasında geliştirilir, bu nedenle yazılımın erken prototipleri üretilmez.
- Herhangi bir aşamada gereksinimler üzerinde değişiklik olursa, test belgelerinin de diğer belgeler ile birlikte güncellenmelidir.

V modeli, gereksinimlerin açıkça tanımlandığı projeler için kullanılabilir.



# SPIRAL MODEL

Tasarımı doğrusal bir süreç olarak gören diğer modellerin aksine, bu model spiral bir süreç olarak görür. Bu, yineleyici tasarım döngülerini genişleyen bir spiral olarak temsil ederek yapılır. Genellikle iç çevrimler, gereksinim tanımının rafine edilmesi için prototipleme ile birlikte ihtiyaç analizinin erken evresini ve dış spiraller yazılım tasarımını aşamalı olarak temsil eder. Her helezonda, tasarım çabalarını ve bu yineleme için ilgili riski değerlendirmek için bir risk değerlendirme aşaması vardır. Her spiralin sonunda, mevcut spiralin gözden geçirilebilmesi ve bir sonraki aşamanın planlanabilmesi için gözden geçirme aşaması vardır.



**Spiral Process Model**



# SPIRAL MODEL

Her tasarım sarmalının altı ana faaliyeti altı temel görevle temsil edilmektedir:

1. Müşteri İletişimi
2. Planlama
3. Risk Analizi
4. Yazılım Tasarımı
5. Üretim-dağıtım
6. Müşteri onayı

## Avantajları

1. Risk analizi yapmaktadır.
2. Bu yazılım tasarım modeli, büyük yazılım projelerini tasarlamak ve yönetmek için daha uygundur.

## Dezavantajları

1. Risk analizi yüksek uzmanlık gerektirir.
2. Kullanması pahalı model
3. Küçük projeler için uygun değildir.

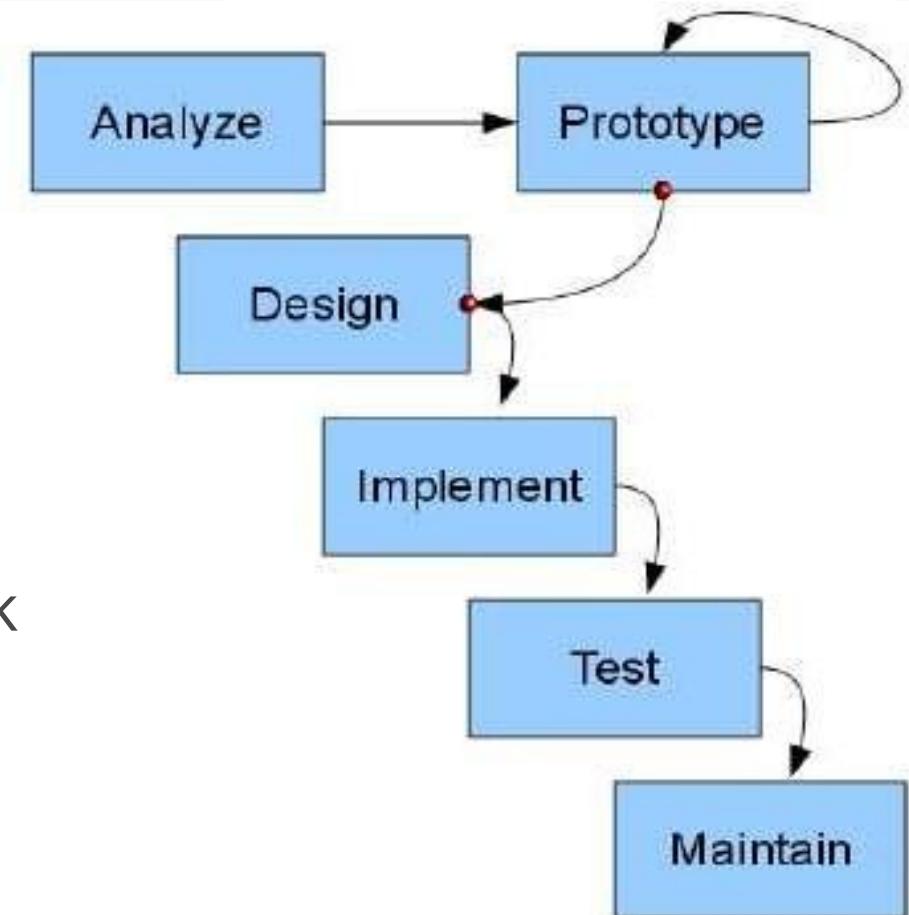


**Spiral Process Model**



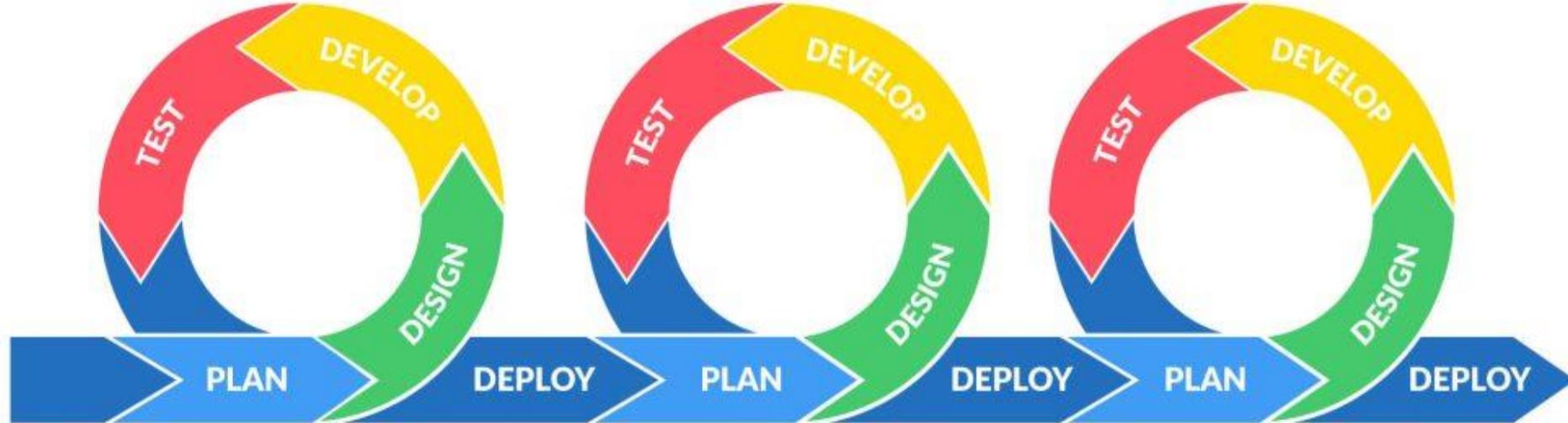
# PROTOTİP MODEL

- Bu modelde çabuk tasarım, prototip geliştirme ve müşteri değerlendirmeinden sonra prototip iyileştirilip referans ürün ortaya konur.
- Müşteriye sunulan ön ürün; ilk ürün olarak kabul edilir, yada iptal edilip en baştan yapılabilir.





# AGILE METHODOLOGY (Çevik Metodoloji)



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



# AGILE METHODOLOGY (Çevik Metodoloji)

Aşırı kuralcı klasik yazılım süreç modellerine tepki olarak ortaya çıkmıştır. Yazılımlar daha yüksek maliyetli ve daha yavaş geliştirilmekteydi. Yazılım geliştirme sürecini hızlandırmak, daha etkin kullanmak ve gereğinde dökümante etmek amacıyla bir çok yaklaşım ortaya çıkmıştır.

- 2001 yılında yazılım dünyasının önde gelen isimlerinden 17 arkadaşı; "Agile (Çevik) Yazılım Geliştirme Manifestosu" ve "Agile (Çevik) Yazılımın Prensipleri" ni yayınlamışlar, bu oluşumu ve gelişimini desteklemek için "Agile Alliance" adıyla, kar amacı gütmeyen bir organizasyon kurmuşlardır.
- Manifesto, nasıl daha iyi bir yazılım geliştirdiklerini ve bunu yapmak isteyenlere yol gösterecek 12 maddeden oluşmaktadır.



# AGILE METHODOLGY (Çevik Metodoloji)

## AGILE MANIFESTO

**1) İlk önceliğimiz kaliteli yazılımı müşteriye teslim edebilmektir.** Bu projenin ilk aşamalarından itibaren sürekli teslimlerle yapılır ve müşterinin yazılımı çok önceden kullanmaya başlayarak değer sağlamasına olanak sağlanır. Günümüzde çevik süreçlere artan ilginin başlıca nedenlerden biri , yapılan yatırımların hızlı geri dönüşünün olmasıdır.

**2) Değişiklikler projenin ileriki aşamalarında dahi olsa kabul edilir.** Amaç müşterinin ihtiyaçlarını karşılayan, onlara yarar sağlayacak, gerçek değer katabilecek yazılım üretmektir ve ihtiyaçlarda meydana gelen değişiklikler projenin sonraki aşamalarında dahi yazılıma aksettirilmelidir. Test, gündemli tasarımlar, kapsamlı otomatik testler, sürekli entegrasyon, basit tasarım gibi pratikler sayesinde değişikliklerin getireceği maliyetler minimuma indirilir ve süreç değişikliklere çabuk adapte hale getirilir.



# AGILE METHODOLGY (Çevik Metodoloji)

## AGILE MANIFESTO

**3) Çok kısa aralıklarla yazılım teslimleri yapılır.** Bu aralıklar tipik olarak 2-4 hafta arasıdır. Bu sayede sürekli geri beslenim (feedback) sağlanır ve müşterinin tam istediği şekilde yazılım geliştirilerek ilerler.

**4) Alan uzmanları , yazılımcılar, testçiler günlük olarak birlikte çalışırlar.** Farklı roller arasında duvarlar örülmez. Rol bazlı ekipler yerine yazılım özelliklerine(features) göre ekipler oluşturulur. Analist (BA), yazılım geliştirici(Dev), tester (QA) vb. aynı ekibin içinde çalışır ve sürekli iletişim halindedir.

**5) Motive olmuş bireyler etrafında projeler oluşturun.** Ekip üyelerine kendileri ile ilgili alacakları kararlar konusunda güvenilir. Ekip kendi kendine organize olacak yetkiye sahiptir. Onlara ihtiyaç duydukları ortamı ve desteği verin ve işi tamamlamaları için onlara güvenin.



# AGILE METHODOLGY (Çevik Metodoloji)

## AGILE MANIFESTO

- 6) Bir geliştirme ekibine ve içinde bilgi aktarmanın en verimli ve etkili yöntemi yüz yüze görüşmedir.**
- 7) Çalışan yazılım, ilerlemenin birincil ölçüsüdür.** Projedeki gelişmenin tek ölçüsü o ana kadar geliştirilmiş özellikler ve çalışan yazılımdır.
- 8) Agile processes (sureçler) sürdürülebilir gelişmeyi teşvik eder.** Planlamaların sağlıklı olması için ekibin iş teslim hızının üzerinde çok oynanaması gereklidir. Örneğin fazla mesailer gibi yöntemlerle ekibin hızını geçici olarak artırmak tercih edilen yöntemlerden değildir.
- 9) Teknik mükemmelliğe ve iyi tasarıma verilen sürekli dikkat, çevikliği artırır.**



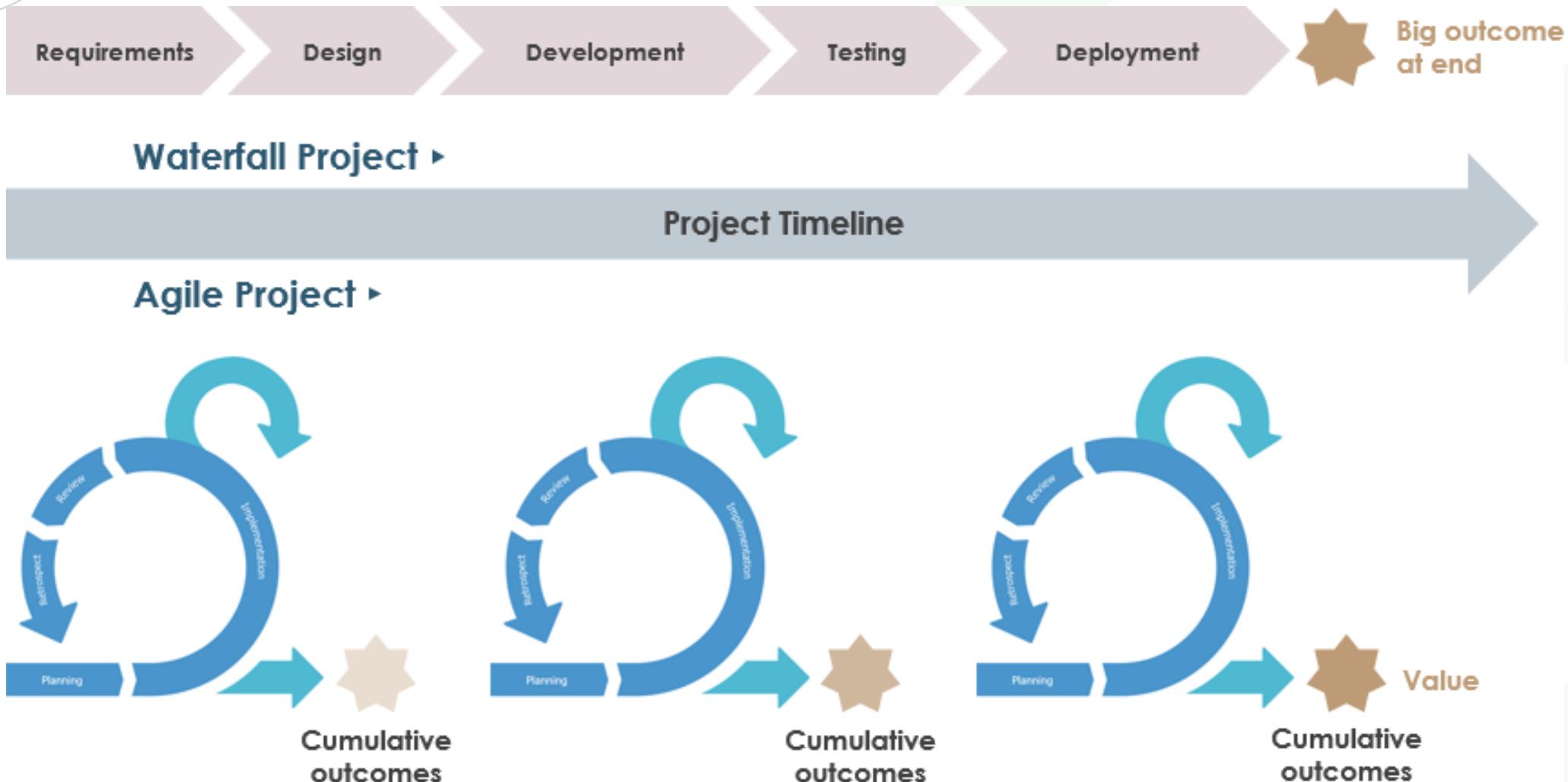
# AGILE METHODOLOGY (Çevik Metodoloji)

## AGILE MANIFESTO

- 10) Sadelik esastır.** Sadelik anlayışı akla gelen baştan savma çözümü uygulamak yerine, anlaşılması ve sonradan değiştirilmesi kolay , maliyeti en düşük ve o an ki gereksinimleri karşılayan çözümü kullanmaktadır.
- 11) En iyi mimariler, gereksinimler ve tasarımlar kendi kendini organize eden ekiplerden ortaya çıkar.** En etkin çalışan ekipler kendilerini organize edebilen , bu konuda yetkin ekiplerdir. Ekip kendi çalışma yöntemlerini sorgulamakta ve gerekli değişiklikleri yapmakta özgürdür.
- 12) Ekip, düzenli aralıklarla nasıl daha etkili olunacağını düşünür, ardından davranışını buna göre ayarlar ve ayarlar.** Ekip kısa sürelerle toplanır, çalışma yöntemlerini gözden geçirir ve daha etkili çalışmak için geriye dönük (retrospective) toplantılar yaparak durumları gözden geçirir.



# AGILE METHODOLOGY VS WATERFALL



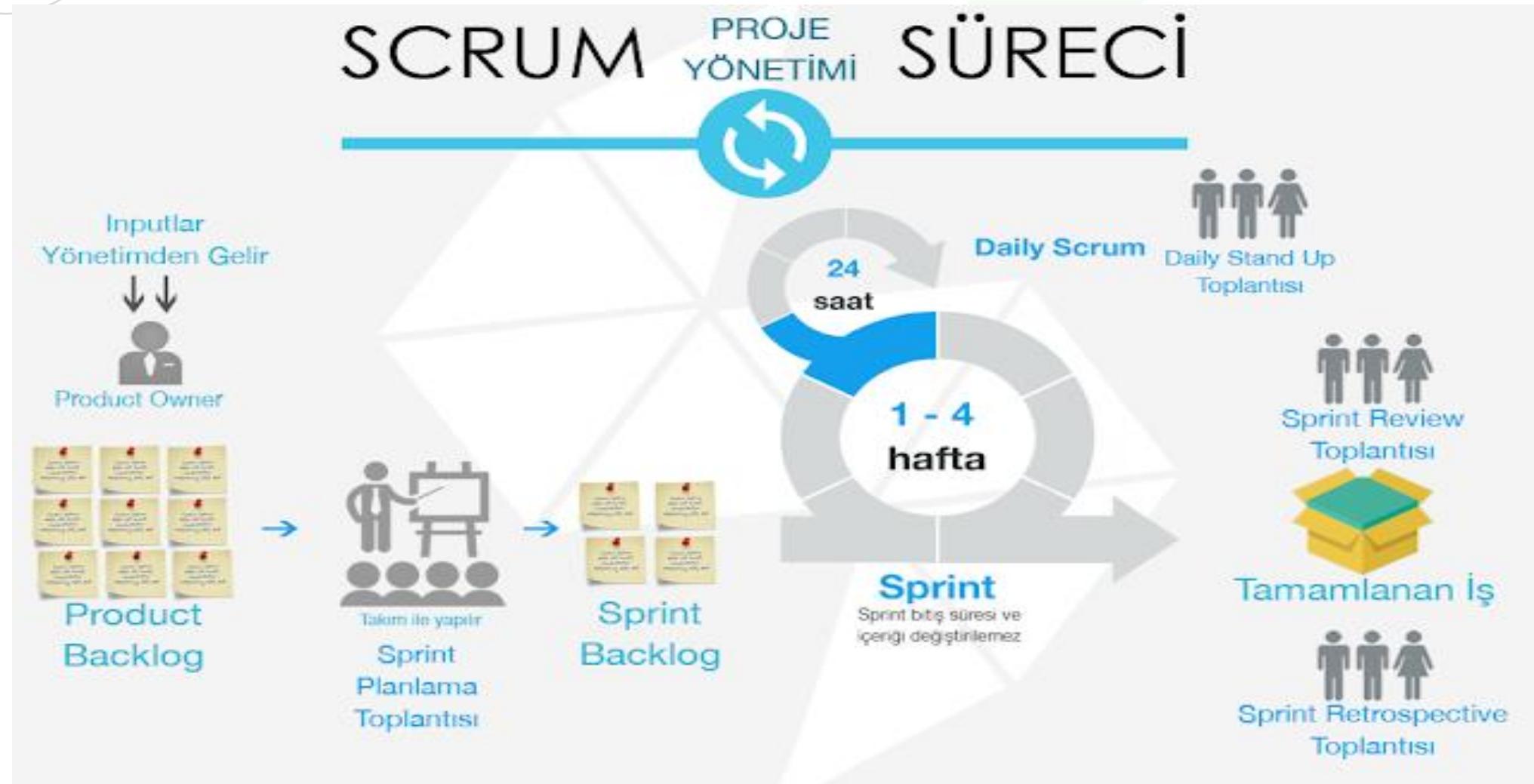


# AGILE METHODOLOGY VS WATERFALL

Method	Successful	Challenged	Failed
Agile	42%	50%	8%
Waterfall	26%	53%	21%



# SCRUM





# SCRUM

Zaman içerisinde projelerin daha büyük ve karmaşık bir hal alması, bununla beraber müşterinin büyük resmi göremeyip gereksinimlerini tam olarak ortaya koyamaması, teknolojinin çok hızlı değişmesi ile beraber gereksinimlerin çabuk değişmesi ve bunu projemize entegre edemeyişimiz gibi problemlerden dolayı çoğu proje başarısızlık ile sonuçlanmaya başladı. Böylece proje sürecinin yönetilmesi konusu önemli bir konu oldu ve “Çevik (Agile) Yazılım Geliştirme Manifestosu” ortaya çıktı.

**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 ihtiyaçına göre şekillendiği için müşterinin geri bildirimine göre yapılanmayı sağlar. İletişim ve takım çalışması çok önemlidir.*

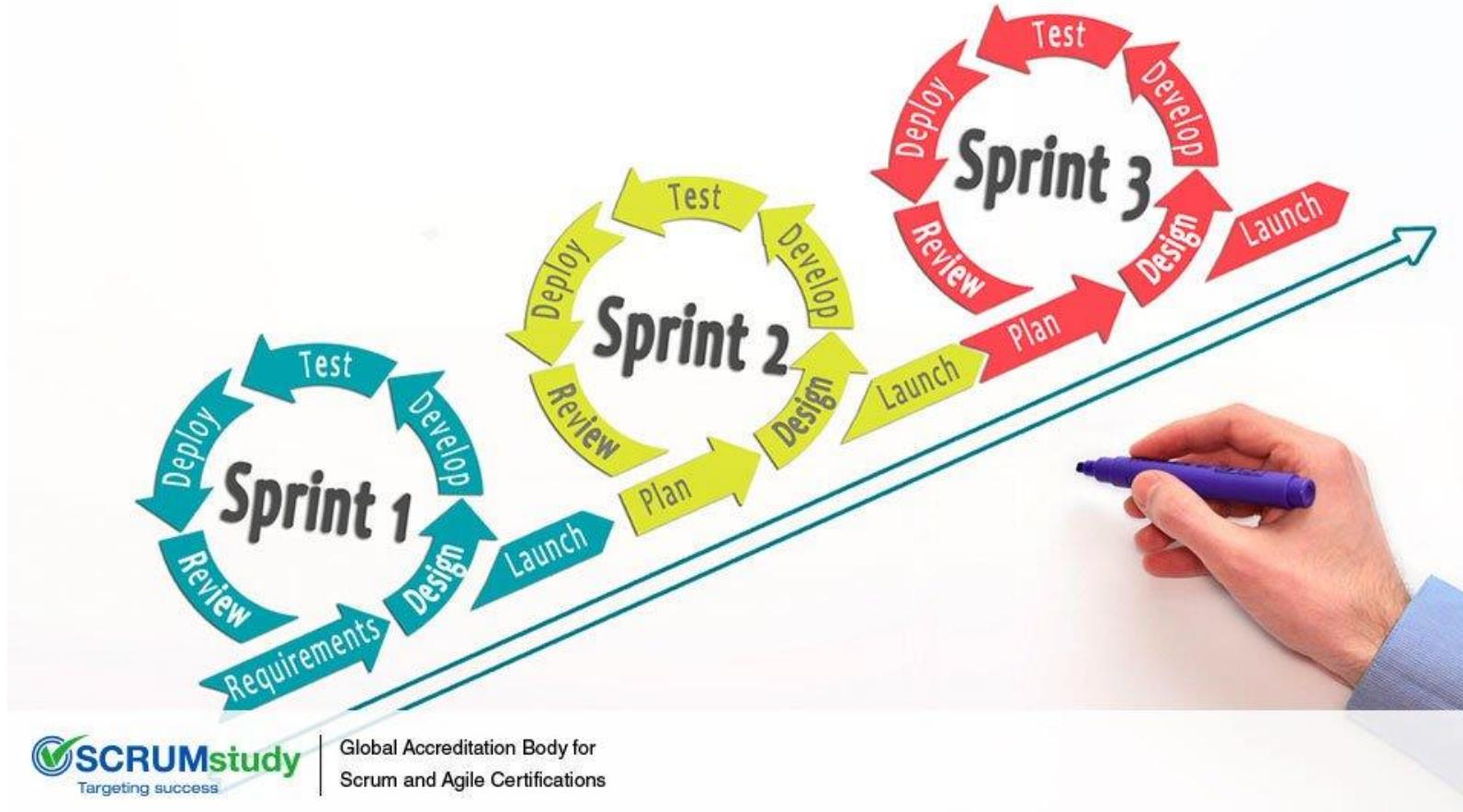


## Scrum teorisi 3 temel değere dayanır;

- **Şeffaflık** : Ekip, scrum boyunca ugruna çalıştığı ürünün mevcut durumunu görebilmeli.
- **Denetleme** : Herkesi aynı noktada tutabilmenin tek yolu sık sık kontrol noktalarında soluklanmaktır.
- **Adaptasyon** : Ekipten biri dahi sprint planından saptığında planda en hızlı şekilde yeniden düzenlemeler yapılmalıdır.



# SCRUM

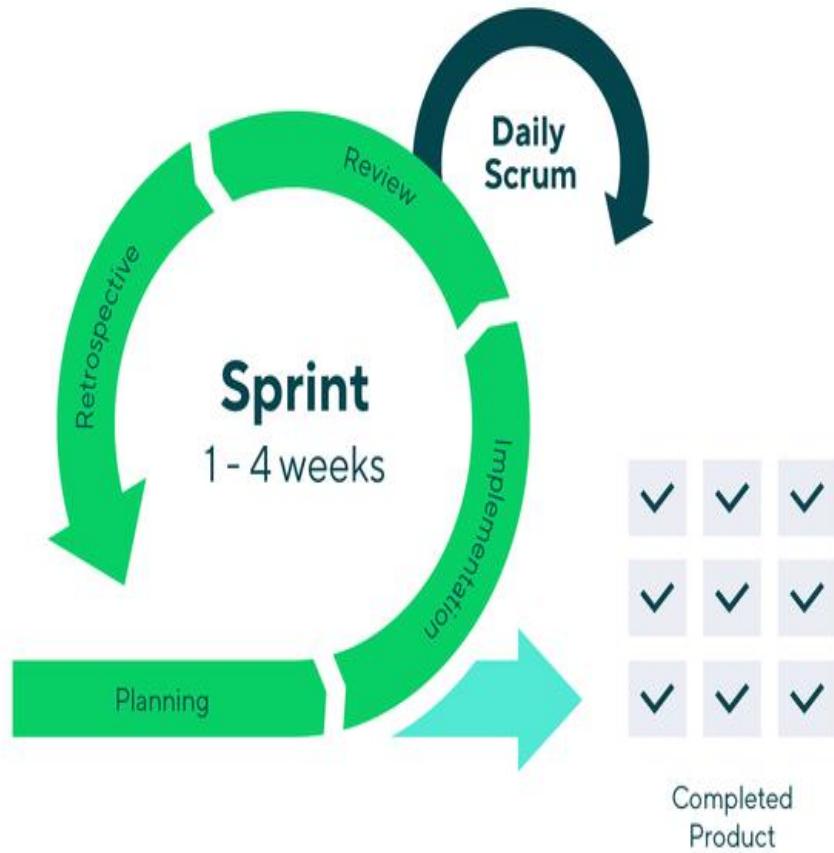


Global Accreditation Body for  
Scrum and Agile Certifications

“Rugby” yaklaşımı : “takımın, mesafenin tümünü hep beraber, bir birim halinde topu ileri geri atarak kat etmesidir.”



# SCRUM

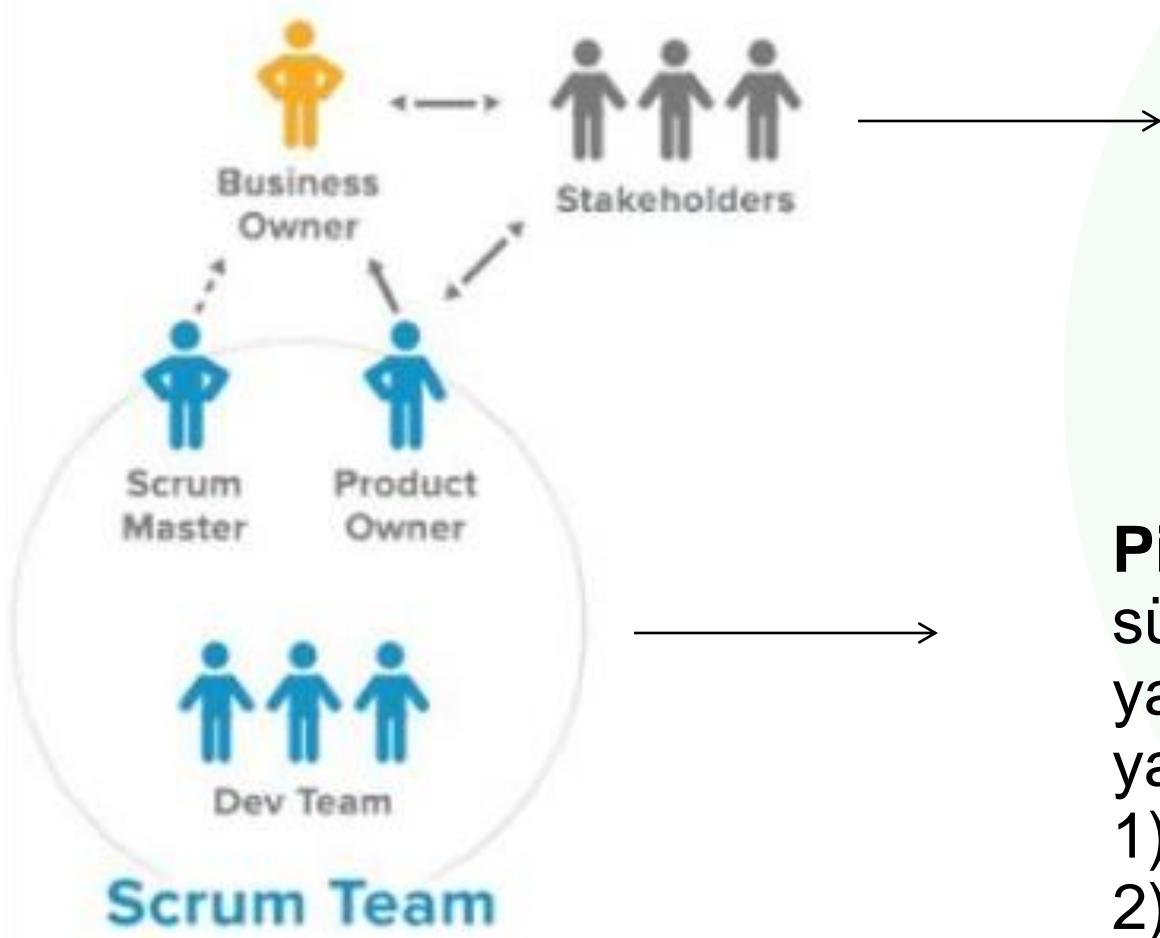


## SPRINT

- 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 tasarlanır, kodlanır ve test edilir.
- Yapılacak tüm işler, Product Backlog da biriktirilir, Product Owner (PO) nun belirlediği önceliğe göre Sprint Backloguna alınır ve bir sprintte bitirilerek ürünün demosuna eklenir.



# SCRUM TEAM



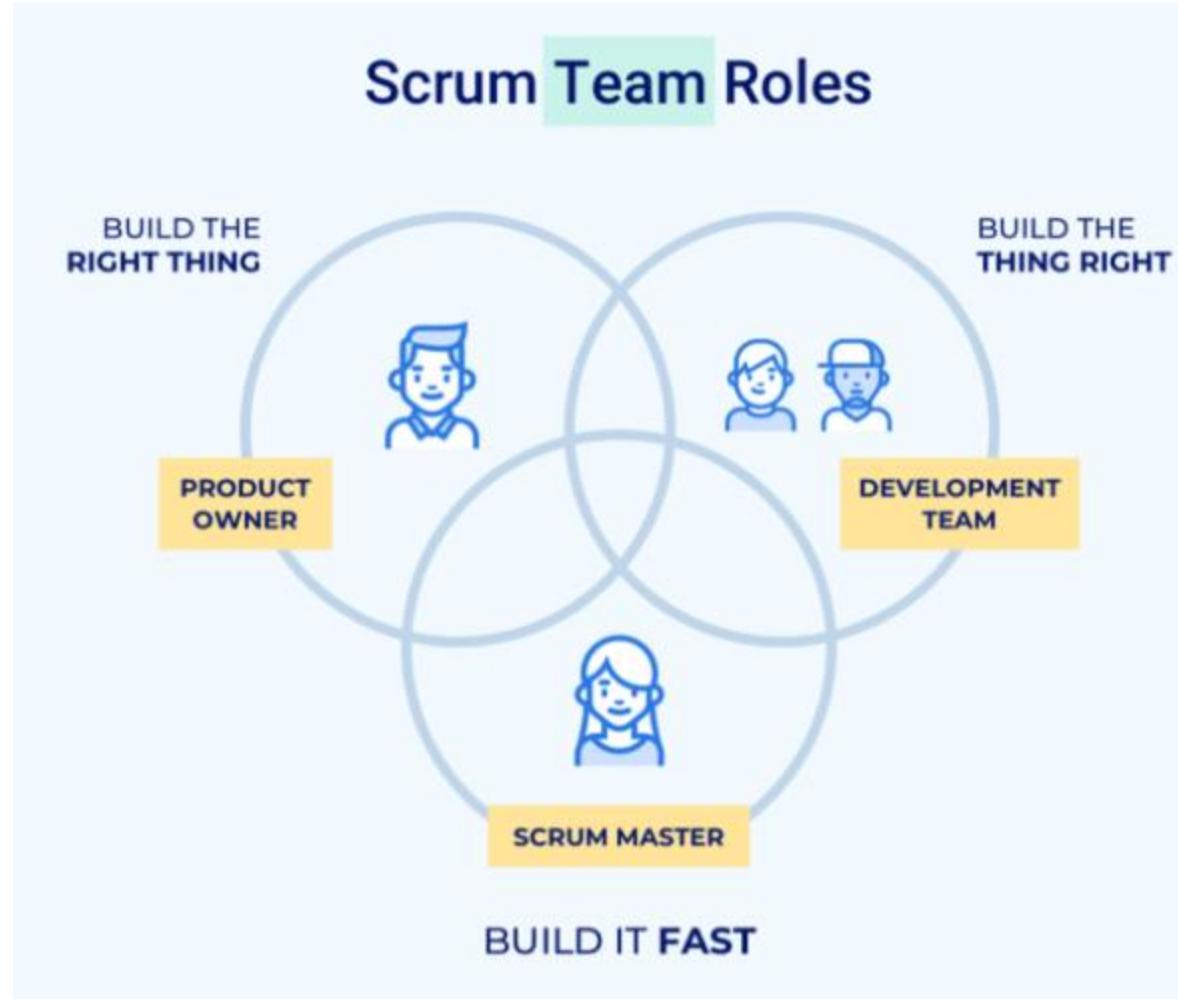
**Chicken Roller:** Scrum’ın işleyişinde aktif olarak yer almayan kişilerdir. Müşteriler, satıcılar gibi.

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

- 1) Product Owner (PO)
- 2) Scrum Master
- 3) Geliştirme Takımı



# SCRUM TEAM ROLES



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



# SCRUM TEAM ROLES

## Product Owner:

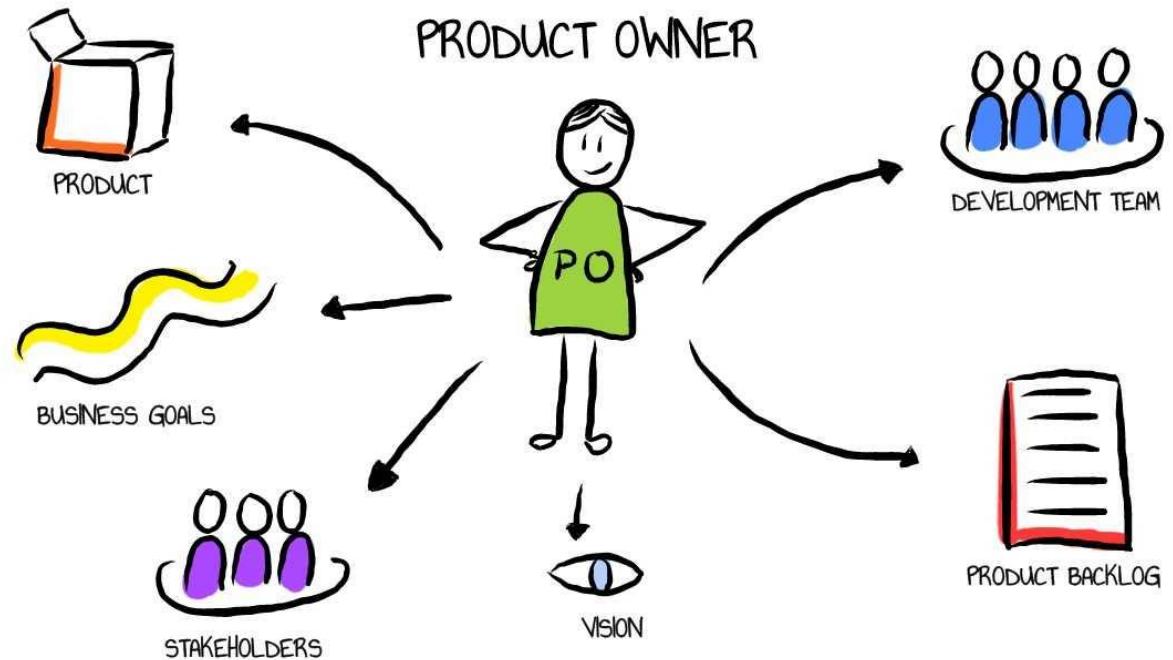
**Geliştirme takımı ve müşteri** arasındaki iletişimini sağlar. Projenin özelliklerini tanımlar. Projenin önceliklerine göre Product Backlog (iş listesi) oluşturur. Ekipte Stakeholders'ı temsil eder.

- İş Listesini (Product Backlog) yönetir. Birinci önceliği; iş listesi'ni yönetmektir. Product Backlog'i yönetmek demek, ürünü yönetmek demektir.
- Ürünle ilgili yapılacak bütün geliştirme Product Backlog'da bulunur.
- Product Backlog'un sahibi PO'dur.
- 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 organizasyonunu yapar.
- Sprint değerlendirme toplantıları'nın sahibidir.



# SCRUM TEAM ROLES

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



R.B.



# SCRUM TEAM ROLES

## Product Owner Kim Değildir :

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



# SCRUM TEAM ROLES



## 2) Scrum Master:

Scrum kurallarını, teorilerini ve pratiklerini iyi bilir ve takımın bu kurallarını uygulamasından sorumlu kişidir. Takımın yönetici değilidir. Takımı rahatsız eden, verimli çalışmalarını engelleyen durumları ortadan kaldırır.

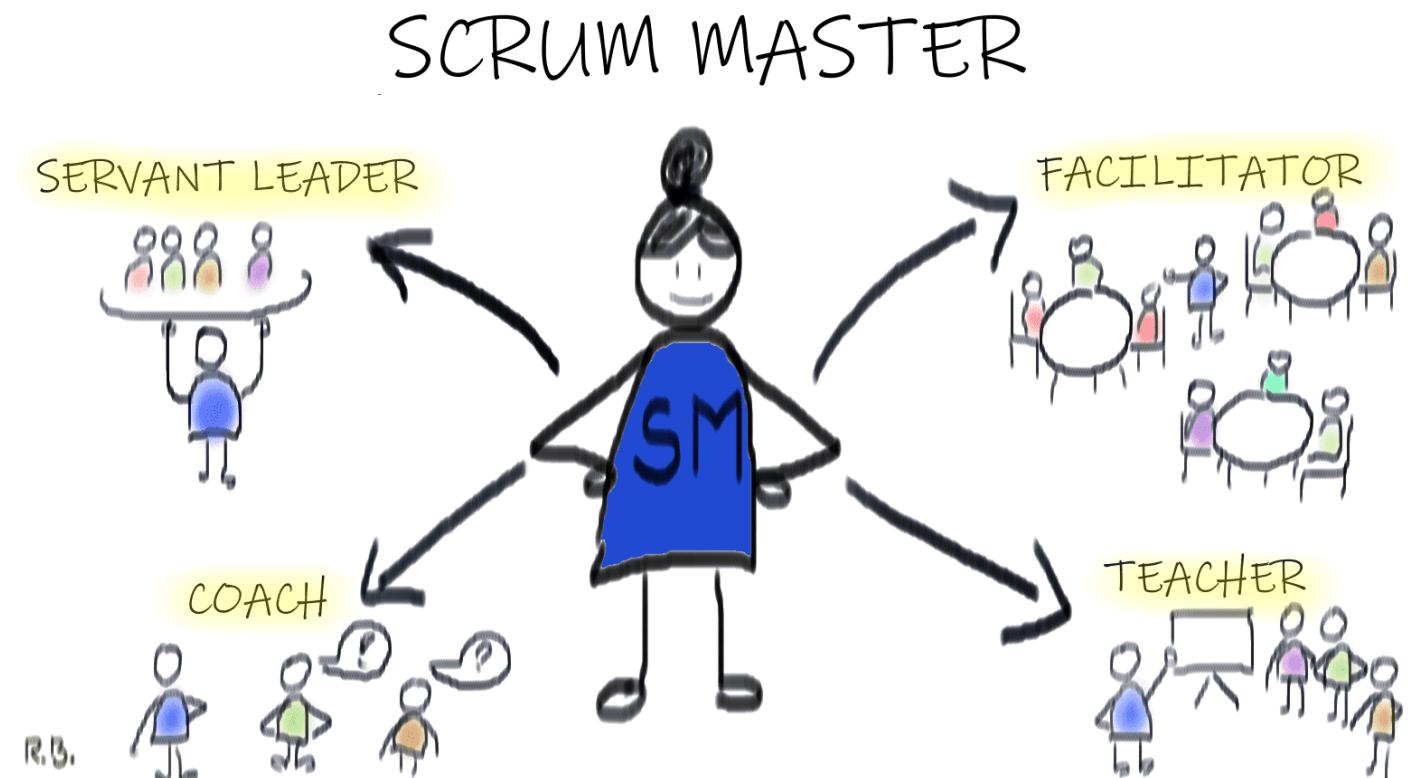
“Scrum Master, takımın Scrum değerlerine, pratiklerine ve kurallarına bağlı kalmasını garanti altına almakla sorumludur. Scrum Master, takımı ve organizasyonu Scrum'a adapte eder.”



# SCRUM TEAM ROLES

## 2) Scrum Master:

- Scrum takımının üyesidir
- İş birlikçidir
- Koruyucudur
- Yardımcıdır
- Problem çözücüdür
- Kararlı ve ulaşılabilir
- Bilgiliidir





# SCRUM TEAM ROLES

## 2) Scrum Master:

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



# SCRUM TEAM ROLES

## 2) Scrum Master:

- Product Owner ile ilişkileri yönetir. Scrum Master, ürün sahibi tarafından belirlenmiş işlerin takımdaki herkes tarafından anlaşıldığından emin olur; ürün sahibinin 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 TEAM ROLES



## 3)Development Team:

Geliştirme Ekibi, Front-End Developer, Back-End Developer, Dev-Ops, QA (Tester), İş Analisti (BA), UI-UX designer 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 önceliğini belirleyebilir ama neyi nasıl yapacaklarına karışmazlar.



# SCRUM TEAM ROLES

## 3)Development Team:

- Takım içerisindeki kişilerin rolleri ve yetenekleri ne olursa olsun, dışarıya karşı bir işin tamamlanmasından tüm takım sorumludur.
- Bir Sprint'e alınan bütün işleri tamamlayacak özelliklere sahip kişilerdir. sprint backlogu 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 parçacığı olarak koymakla yükümlüdür.
- “Self Organize” olmalıdır. Development Team, sorumluluğunu aldığı işlerin yapılması için bir iş tanımlamasına veya bir iş takipçisine ihtiyaç duymaz.



# SCRUM TEAM ROLES

## 3) Developmet Team:

- Scrum takımının üyesidir
- Çapraz Fonksiyonel: Dışarıdan herhangi bir yardıma ihtiyaç duymadan çalışmalarını tamamlamak için gerekli tüm beceri setlerine sahiptir.
- Kendi kendine yeterli
- Kendi kendine organize
- Yetenekli
- Kararlı ve ulaşılabilir
- Sorumlu



# SCRUM SÜRECi

PROJE  
YÖNETİMİ



Inputlar  
Yönetimden Gelir



Product Owner



Product  
Backlog

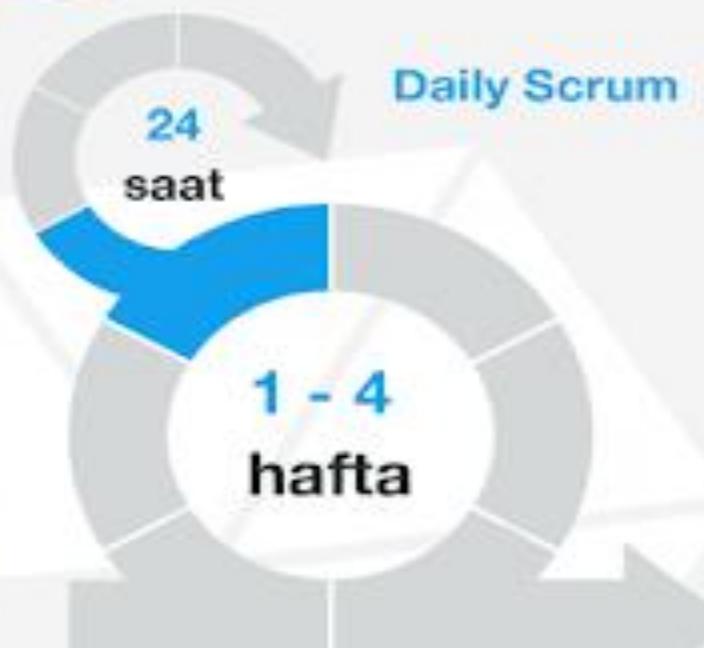


Takım İle yapır

Sprint  
Planlama  
Toplantısı



Sprint  
Backlog



Sprint

Sprint bitiş süresi ve  
İçeriği değiştirilemez



Daily Scrum

Daily Stand Up  
Toplantısı



Sprint Review  
Toplantısı



Tamamlanan İş



Sprint Retrospective  
Toplantısı



# SDLC

## SOFTWARE DEVELOPMENT LIFE CYCLE

(Yazılım Geliştirme Yaşam Döngüsü)

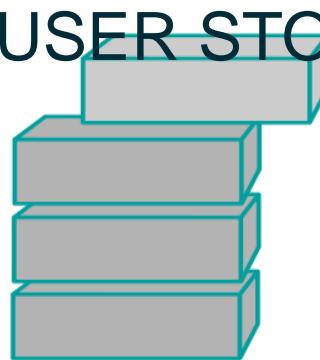
3. Ders



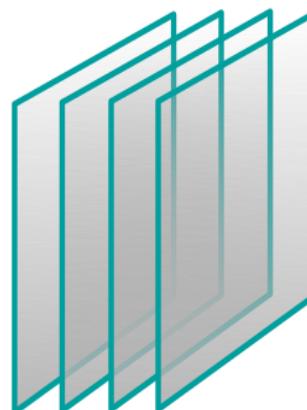
# SCRUM SÜRECİ

## ÜRÜNÜ OLUŞTURAN PARÇALAR

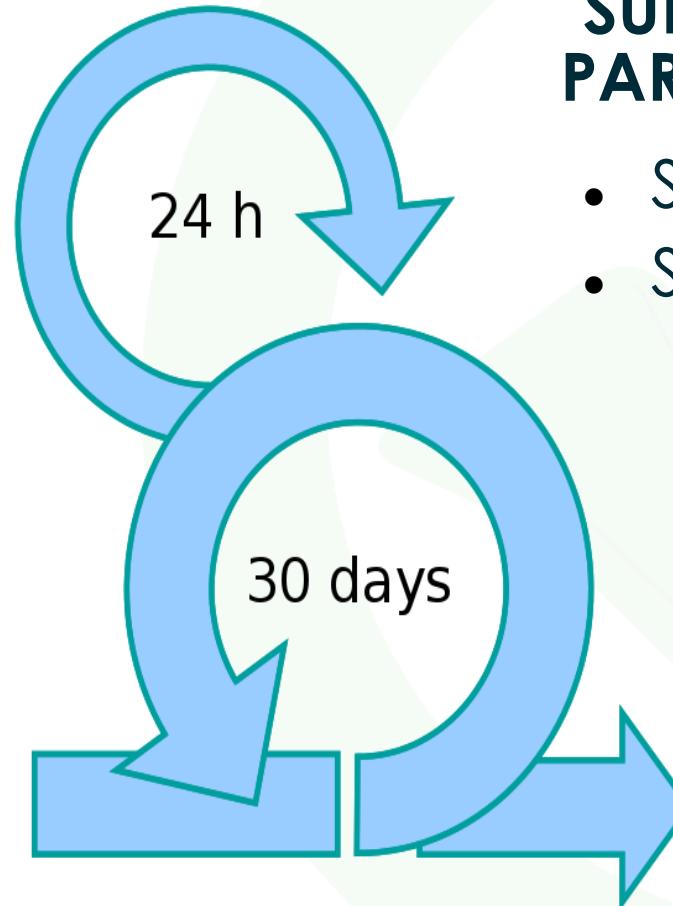
- PRODUCT BACKLOG
- SPRINT BACKLOG
- FEATURE
- EPIC
- USER STORY



Product Backlog



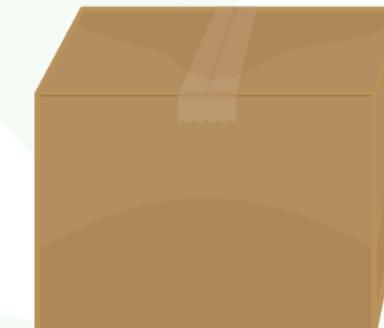
Sprint Backlog



Sprint

## SÜRECİ OLUŞTURAN PARÇALAR

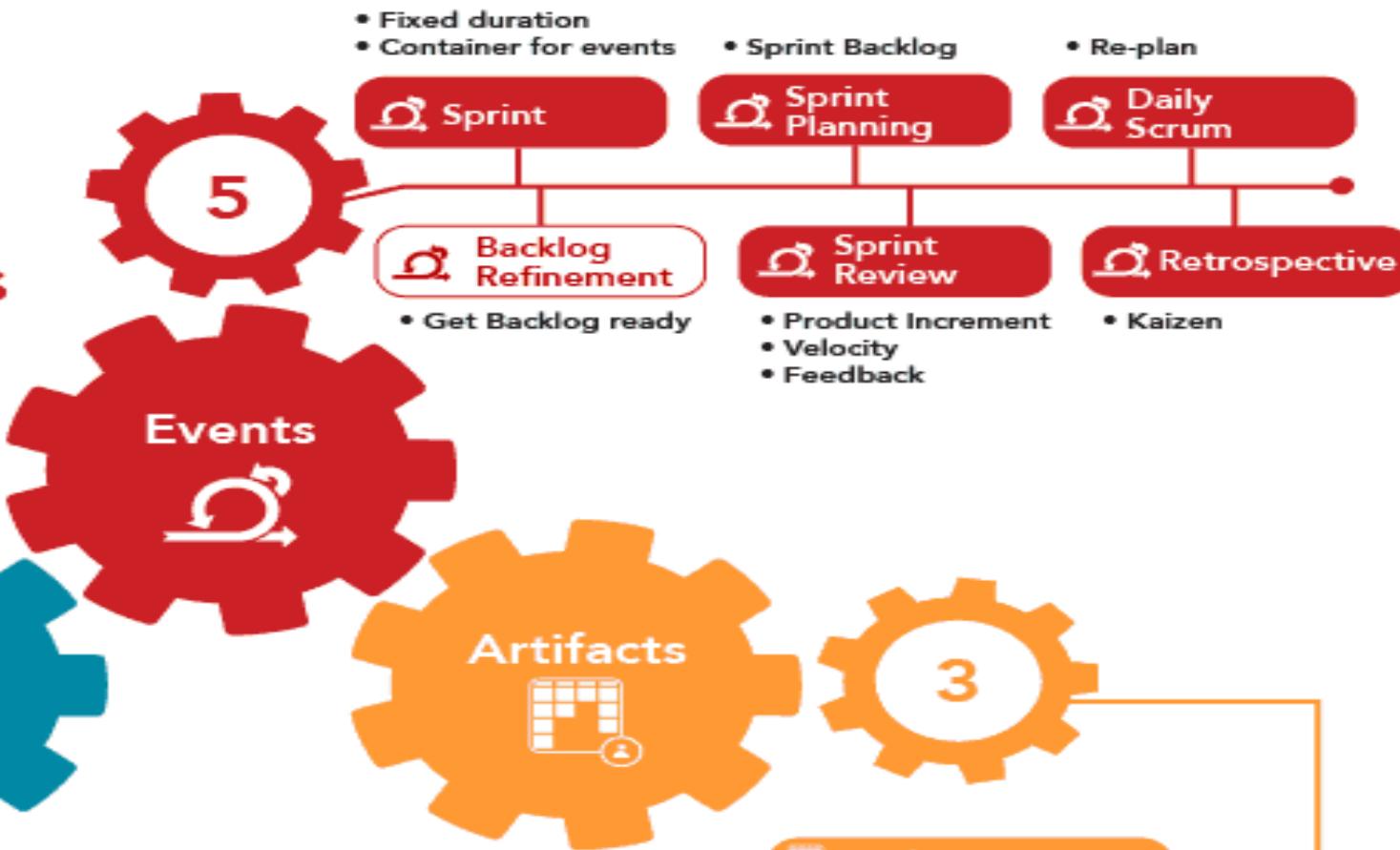
- SCRUM
- SPRINT



Working increment  
of the software

# Scrum's Simple Rules

3 Roles • 5 Events • 3 Artifacts



**Without the 3-5-3 you  
are not doing Scrum.**

**scruminc.**



# 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.



# SPRINT

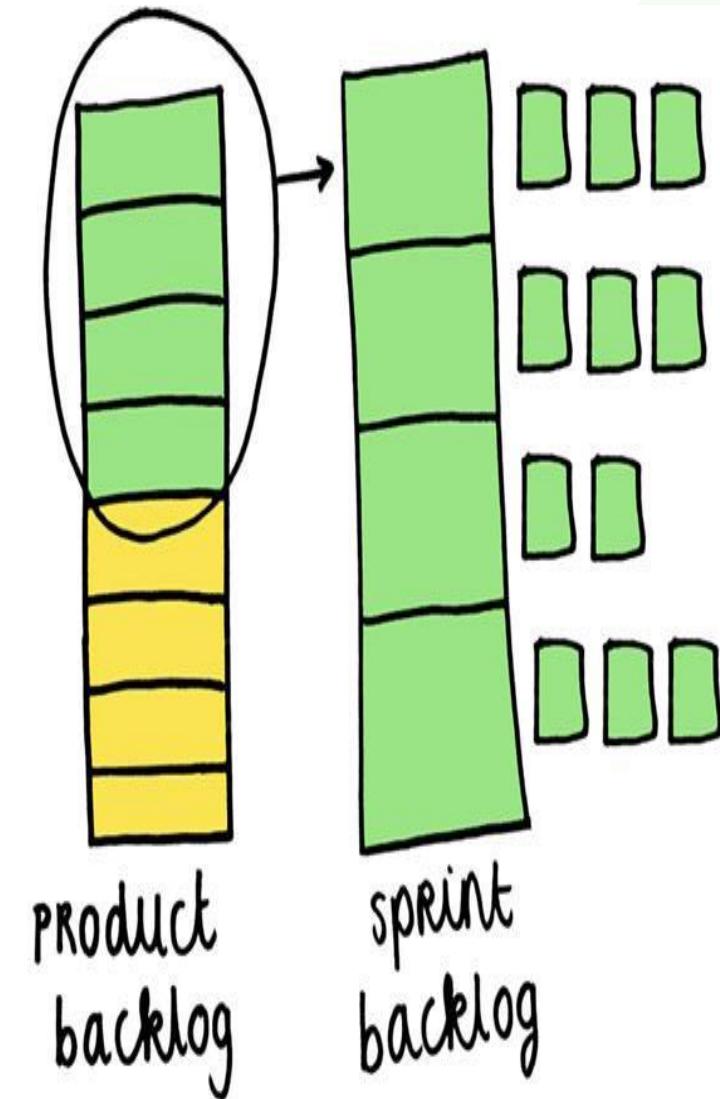
- Scrum süreci 2-4 hafta uzunlukta, ara vermeksizin birbirini takip eden Sprint'lerden (koşu) oluşmaktadır.
- Sprint Scrum takımının ritmi, yani kalp atışıdır. Scrum içerisindeki tüm aktiviteler Sprint içerisinde gerçekleşir.
- Bu kısa süreler, takımlara esneklik ve çeviklik avantajı sağlamaktadır. Scrum 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.



# BACKLOG

**Product Backlog:** Proje için gerekli olan gereksinimler listesidir. Proje sonunda “Ne üretilmek isteniyor?” sorusuna cevap aranır. Product owner tarafından müşteriden gereksinimler alınır, öncelik sırasına göre sıralanır. Product owner, değişen ihtiyaçlara göre product backlog'a ekleme veya çıkarma yapabilir. Böylece değişim, projenin her aşamasında projeye kolayca entegre edilebilir olur.

**Product Backlog Item:** Product backlog içindeki her bir gereksinime verilen isimdir.





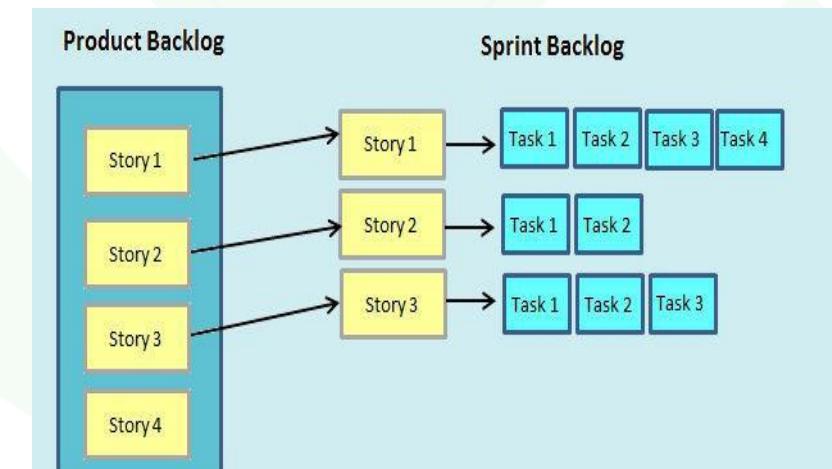
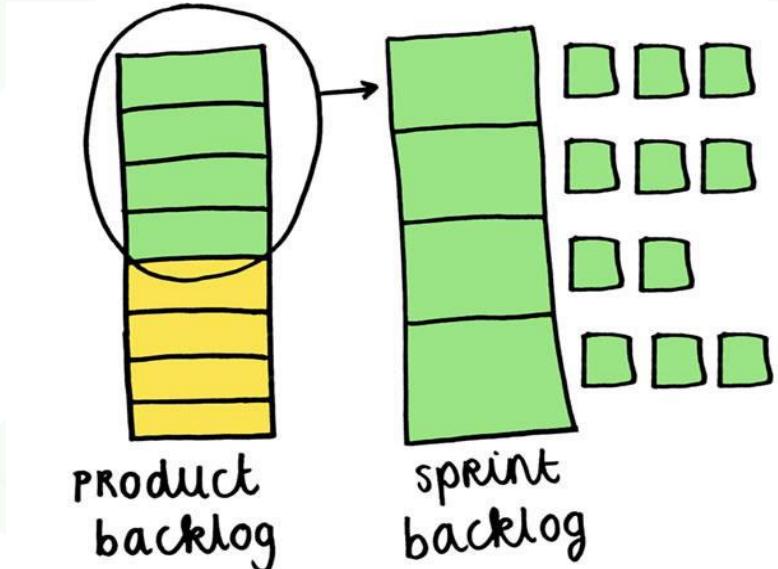
# BACKLOG

- Product Backlog yaşayan bir dokümandır. Çevresel koşullar, müşteri talepleri, teknolojik gelişmeler ve geliştirme yetkinliğinde meydana gelebilecek değişimlerden en yüksek faydayı sağlayacak şekilde sürekli değişir.
- İyi bir Product Backlog takımıın önündeki 2-3 sprint için net anlaşılmış ve yeterince küçük PBI'ları barındırmalıdır.
- Product Owner (Ürün sahibi), Product Backlog'u yöneten yegane kişidir. Bu işi yaparken paydaşlardan, geliştirme takımından ve Scrum Master'dan destek ve yardım alabilir.



# SPRINT BACKLOG

- Sprint Backlog, takımın Sprint boyunca yapacağı işlerin adımlandırılmış, detaylandırılmış, saatlendirilmiş içeriğini teşkil etmektedir.
- Sprint Backlog Development Team'in oluşturduğu, Development Team'in Sprint'te almış olduğu işleri ve bu işleri nasıl tamamlayacaklarına ilişkin stratejilerini içeren bir çıktıdır.
- Sprint Backlog'a alınacak user story'ler belirlenirken de product backlog'daki öncelik sırasını gözetilir. Düşük öncelikli olanları ilk sprint'lere koymazlar.
- Sprint Backlog'a alınacak user story miktarı takimin verdiği point'lere göre değişir.





# SCRUM CEREMONIES

DAILY SCRUM MEETING



SPRINT PLANNING MEETING



SCRUM  
CEREMONIES

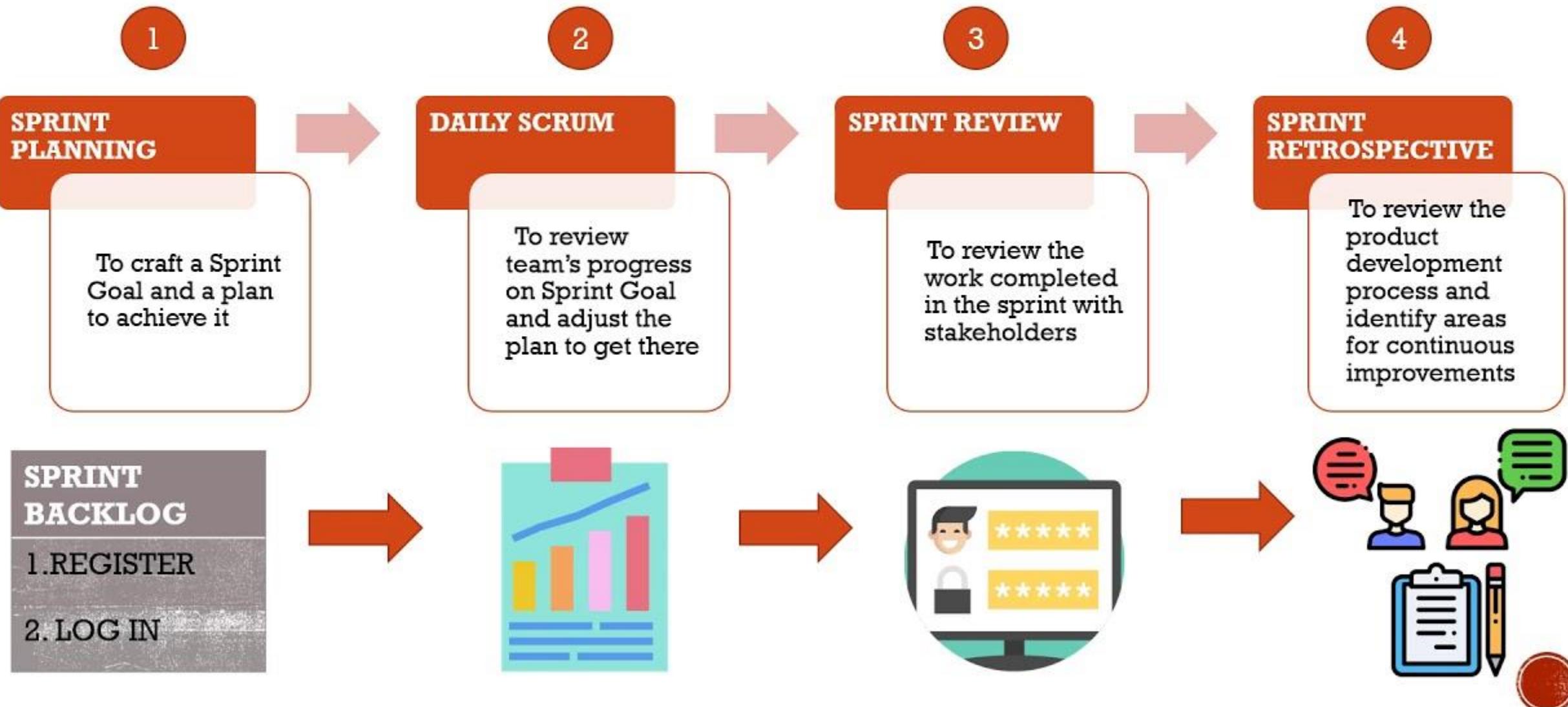
SPRINT REVIEW MEETING



SPRINT RETROSPECTIVE MEETING



# EVENTS/CEREMONIES IN AGILE METHODOLOGY (SCRUM)



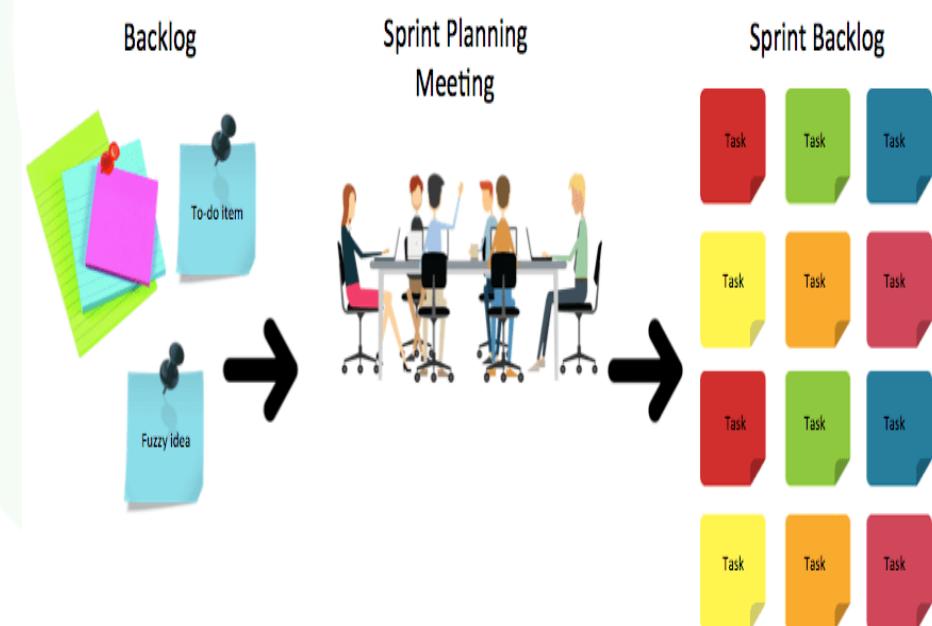


# SPRINT PLANNING

Bu, her Sprint'i başlatan etkinliktir

- 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, Geliştirme ekibi yanıt vermeye, sorunları gündeme getirmeye ve gerektiğinde geri itmeye teşvik edilir.
- Development team daha sonra hız, kaynaklar ve mevcut zaman ve kaynakları etkileyebilecek faktörler hakkında bilgi sahibi olduklarıda Sprint'te kaç PBI sunabileceklerini tahmin eder.
- **Sprint Planlama Toplantısının sonucu, herkesin kabul ettiği gerçekçi ve ulaşılabilir bir Sprint Hedefi ve Sprint İş Listesi elde etmektir.**

## Sprint Planning Meeting





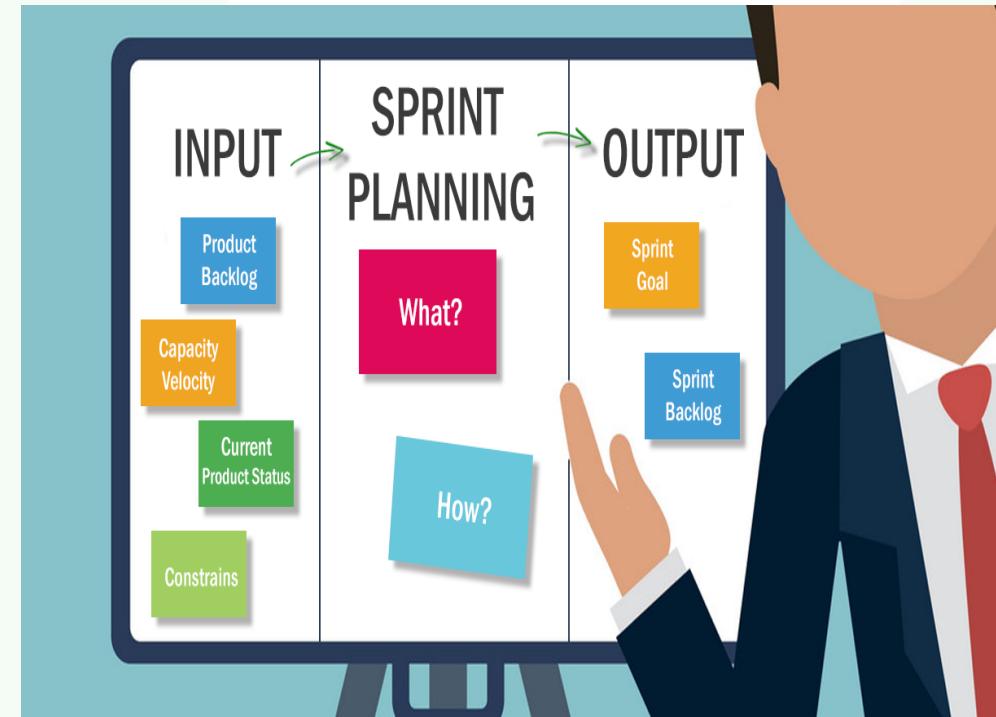
# SPRINT PLANNING

Sprint Planlamada şu sorulara cevap verir;

- 1) Başlayan Sprint'te task olarak ne teslim edilebilir? (Ne Yapacağınız)
- 2) Uygulamayı teslim etmek için gerekli olan iş nasıl başarılacak? (Nasıl Yapacağınız)

2. madde için gerekirse Sprint süresi içerisinde eski adıyla **grooming** yeni adıyla refinement (detaylandırma) aktivitesi gerçekleştirilebilir.

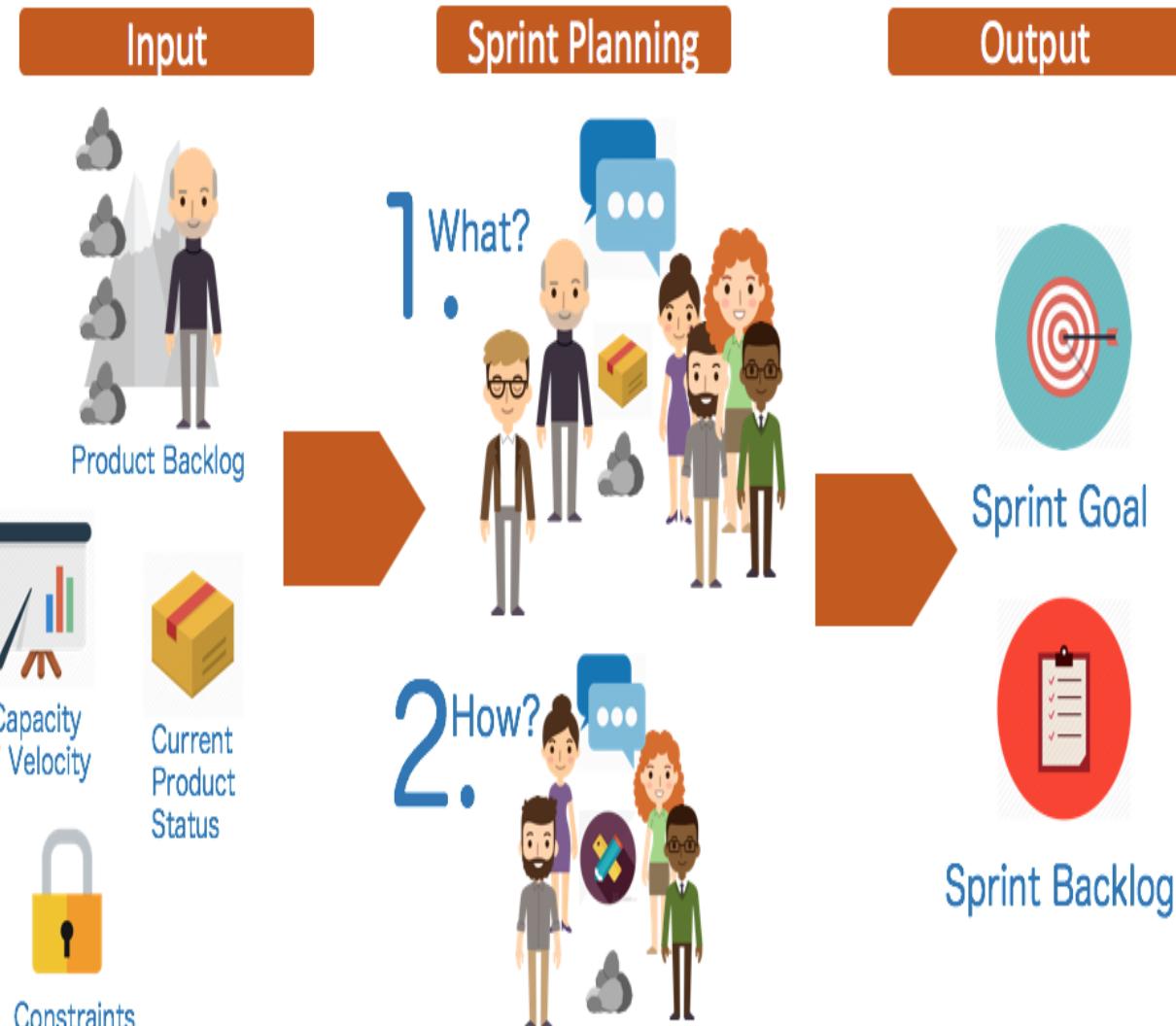
Scrum Master, toplantıının yapılmasını ve Development Team'in etkinliğin amacını anlamasını sağlar.





# SPRINT PLANNING

- Bir önceki Sprint Retrospektif Toplantısı'nda alınan kararlar içinde Sprint Planlama Toplantısını etkileyen aksiyonlar varsa bunları yerine getirildiği kontrol edilir.
- Herşeyin başlangıcı iyi bir planlamadır. Sprint'e iyi başlarsanız başarılı bir şekilde bitirme şansınız yüksek olur. İyi Scrum Master, Scrum Takımı'nın iyi planlama yapmasını sağlar.
- İş Listesi Maddeleri'nde kabul kriterleri belirtilmiş mi?

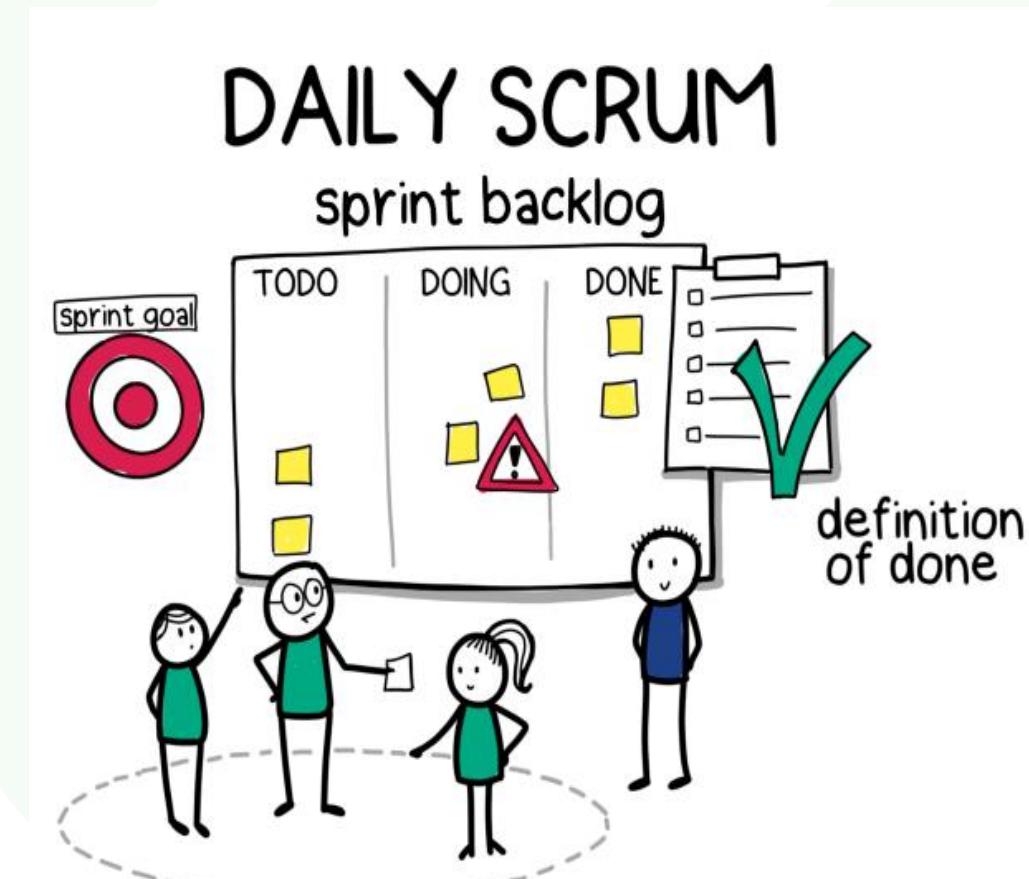




# DAILY SCRUM / DAILY STANDUP

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 Hedefi'ne ulaşma yolundaki ilerlemeyi değerlendirmesi ve önümüzdeki 24 saat boyunca faaliyetlerini gözden geçirmesi ve planlaması için bir fırsatır.
- Genelde dün neler yaptı? Bugün ne üzerinde çalışacağız? Herhangi bir sıkıntı veya engel ile karşılaştık mı?





# DAILY SCRUM / DAILY STANDUP

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 süre ise 43 saniyedir.

**İyi örnek:**

Dün müşteri tanımlama ekranını test ettim, herhangi bir sorunla karşılaşmadım.

Bugünse müşteri tanımlama ekranının, bireysel kredi planı çıkarma ekranı arkasındaki bağlantıyı test edeceğim eğer vaktim kalırsa da vadeli müşteri hesap ekranını test edeceğim. Sonra da Zeynep ile code review yapmayı planlıyorum.

Önümde veritabanı bağlantı hatası var. Sürekli aynı hatayı alıyorum. Sizlerden biri karşılaştıysa beraber bakabilir miyiz?

## DAILY STAND-UP MEETING



Time Box (15 min)



Same place



Same time



Facilitated by Scrum Master



Full team presence



Focus upon 3 questions

### 3 Main Questions

1. What I did yesterday?
2. What I'll do today?
3. What's in my way?

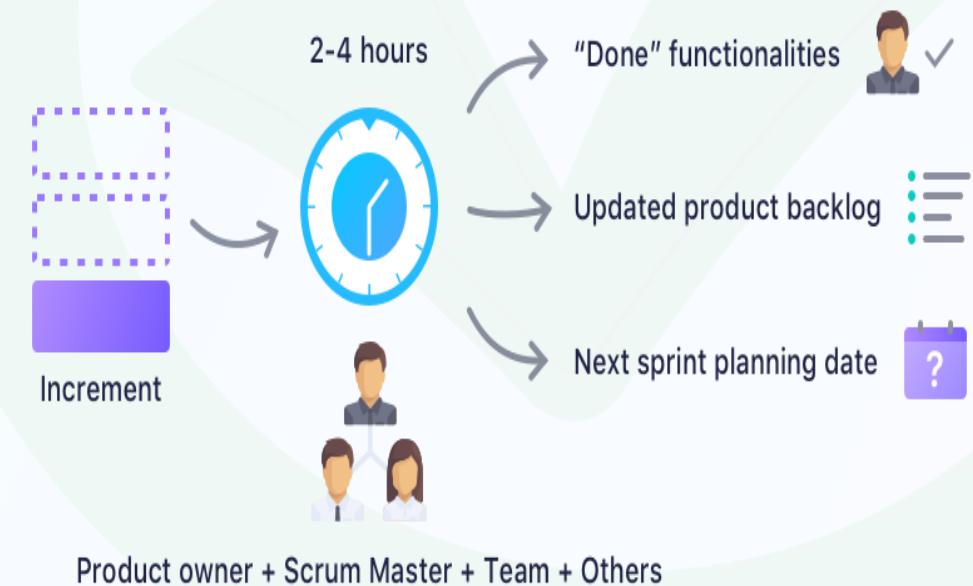


# SPRINT REVIEW – SPRINT DEMO

- Genellikle Sprint'in son gününde yapılır ve Stakeholder'lara(müşteriler, yönetim ve ilgili ve ilgilenilen 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 Ürün İş Listesi'ni ekleyebileceğiniz yararlı geri bildirimler de alıyorsunuz.
- SprintReview Toplantısı'nın sahibi Product Owner'dır.

## Sprint Review

Meeting at the end of the sprint to check the increment





# SPRINT REVIEW – SPRINT DEMO

- 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.



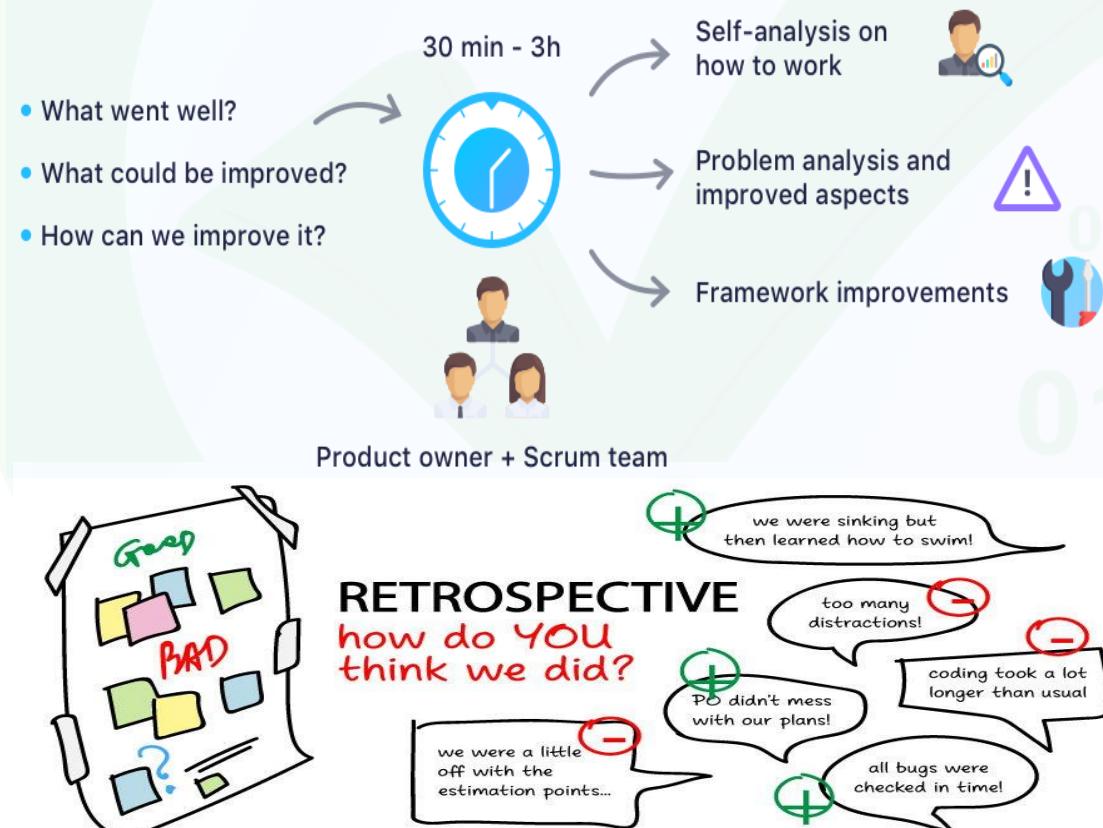


# SPRINT RETROSPECTIVE – SPRINT RETRO

- 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 geliştirme ekibi, 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.
- Ne tür engellerle karşılaşlıklar ve hangi fikirlerin ve güncellemelerin daha fazla gelişim sağlamalarına katkıda bulunduğuunu değerlendirdirler.

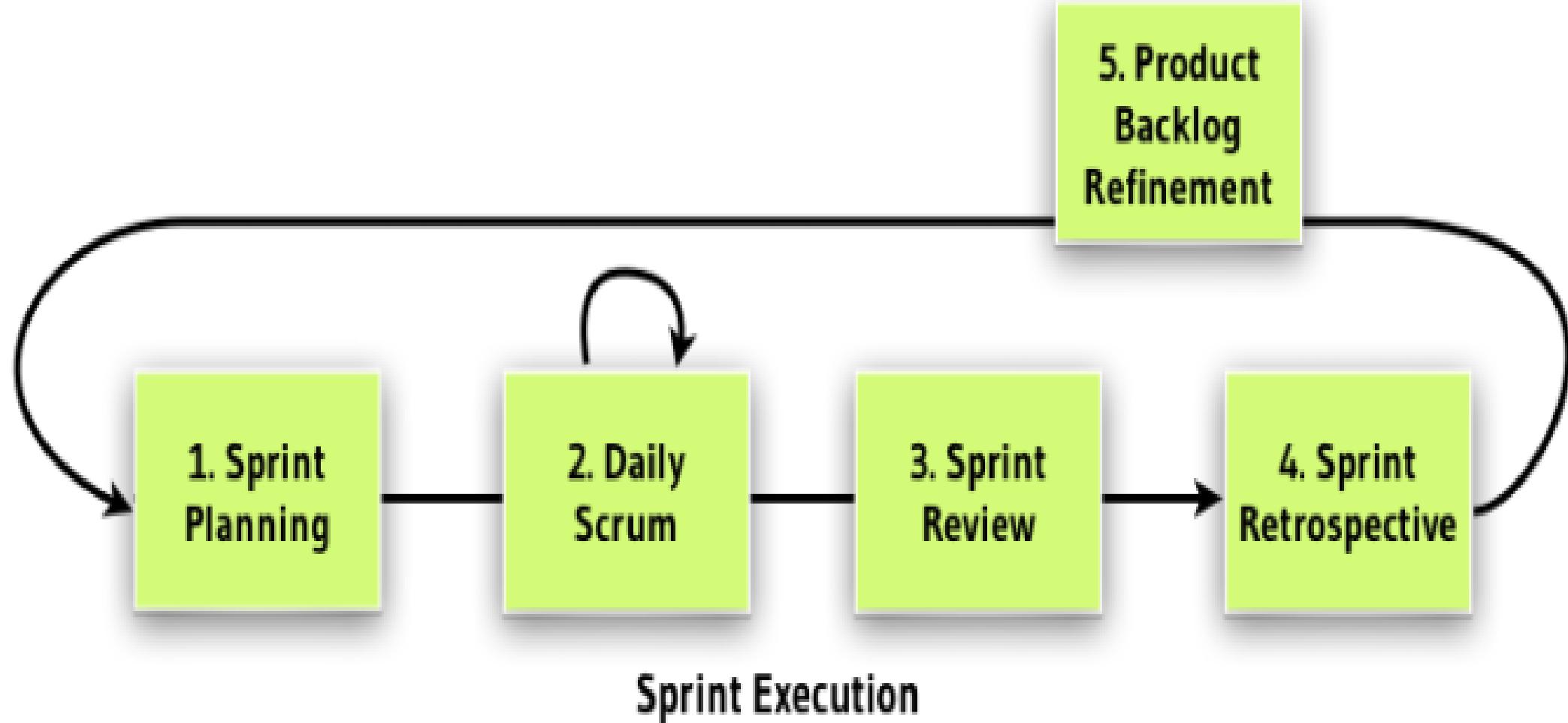
## Sprint Retrospective

Meeting after Sprint Review to review processes



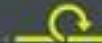


# SPRINT EXECUTION





# SCRUM TASK BOARD



scrum master



sprint backlog



scrum team



STORY	TO DO	IN PROGRESS	TO VERIFY	DONE



# KANBAN

Kanban uzakdoğu kökenli bir kelimedir. Japonca'da "sinyal tahtası", Çince de ise "büyük görsel tahta" anlamına gelmektedir.

Agile yaklaşımı içinde de en çok tercih edilen metodlardan birisidir.

Temelini de 1950li yıllarda kullanılmaya başlayan Toyota Üretim Sistemine dayanmaktadır.

2000li yılların başında ise David Anderson tarafından da bir değişim yönetim metodu olarak geliştirilmiştir.

Kanban metodu evrimsel bir metot olduğundan mevcuttaki çalışma metodunuza uyarlanabilmektedir. Waterfall sürecinde de kanban prensipleri uygulayabildiğiniz gibi Scrum çerçevesinde de uygulayabilirsiniz.

Hatta son zamanlarda Scrum ve Kanban pratiklerinin beraber kullanıldığı "Scrumban" metotuda sıkça kullanılmaktadır.





# KANBAN

Kanban, ekip yapınızda herhangi bir değişiklik yapmadan süreçleri ve iş akışı verimliliğini iyileştirmek için kullanılabilir. **Kanban Metodu**'nu uygulamadan önce, temel ilkelerini anlamak ve benimsemek önemlidir:

- **Ne yapıyorsanız oradan başlayın** - Kanban, belirli bir kurulum gerektirmez ve doğrudan mevcut iş akışınıza uygulanabilir. Bu, mevcut süreçlerinizi değiştirmenize gerek olmadığı için uygulamayı kolaylaştırır.
- **Değişimi kabul etmelisiniz** - Kapsamlı değişiklikler ekipleri rahatsız edebilir, akışı bozabilir ve performansa zarar verebilir. Kanban, sürekli, artan ve evrimsel değişiklikleri teşvik ederek minimum dirençle karşılaşacak şekilde tasarlanmıştır.
- **Mevcut süreçce, rollere ve sorumluluklara saygı gösterin** - Başlangıçta herhangi bir organizasyon değişikliği olmamalıdır. Kanban, mevcut süreçlerin, rollerin ve sorumlulukların değerini olabileceğini ve korunmaya değer olduğunu kabul etmelisiniz.
- **Her düzeyde liderliği destekleyin** - Kanban, tüm üyeler arasında liderliği ve karar almayı destekler. En alt sıradaki ekip üyesinin parlak bir fikri varsa, bu kabul edilmeli ve benimsenmelidir.



# KANBAN PRATİKLERİ NELERDİR?

1

GÖRSELLEŞTİR



2

DEVAM EDEN İŞLERİ  
LİMİTLE



3

AKIŞI YÖNET



4

POLİTİKALARINI  
AÇIK HALE GETİR



5

GERİ BİLDİRİM  
DÖNGÜLERİ KUR



6

İŞBİRLİĞİ İLE İYİLEŞ,  
DENYESELLİKLE EVRİMLES  
(BİLİMSEL YÖNTEM VE MODELLERİ KULLAN)





# KANBAN

Başarılı Kanban uygulaması için gözlemlemeniz gereken altı temel uygulama vardır.

**İş akışını görselleştirin** - İlk ve en önemli görev, mevcut iş akışını anlamaktır - bir öğeyi talepten teslim edilebilir bir ürüne taşımak için uygulanacak adımların sırası nedir? Bu, kartları ve sütunları olan bir Kanban panosu kullanılarak yapılır: her sütun iş akışınızdaki bir adımı temsil eder ve her kart bir iş öğesini temsil eder.

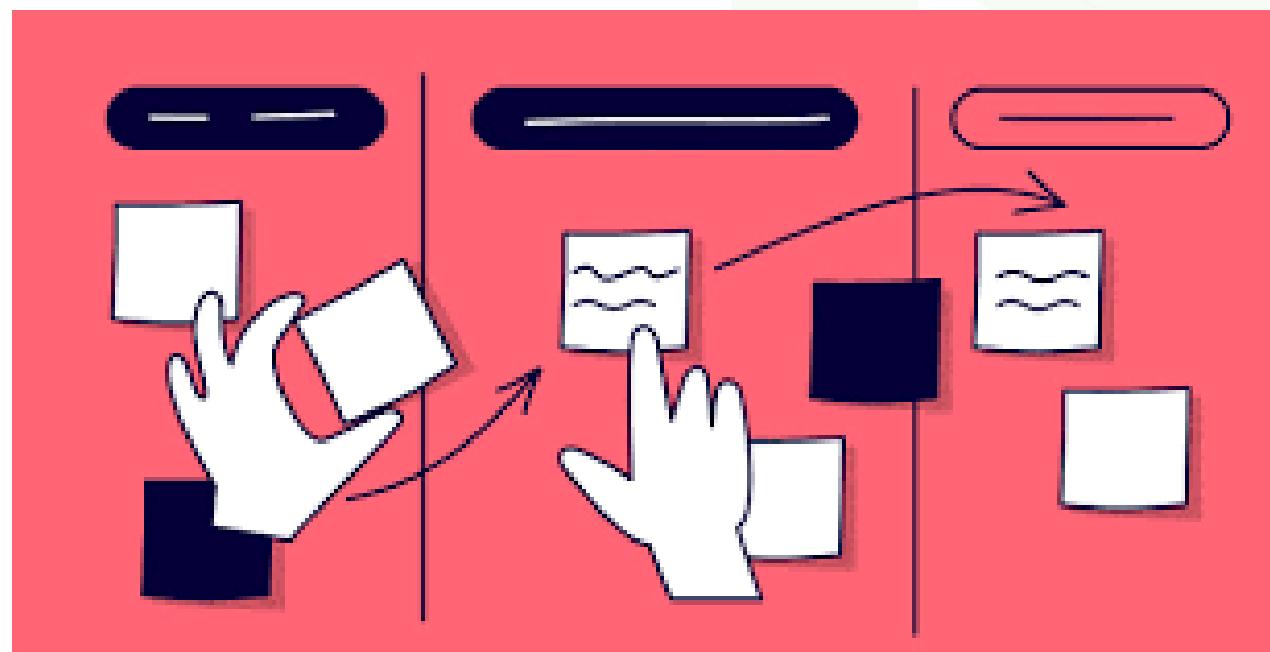




# KANBAN

## Devam Eden Çalışmayı Sınırla

Odaklanma kaybı, ekibinizin performansına ciddi şekilde zarar verebilir, bu nedenle bu uygulama, devam eden işe sınırlar koyarak kesintileri ortadan kaldırılmaya odaklanır. Devam Eden Çalışma için sınırlar uygulayan ekipler, yeni işe başlamadan önce olağanüstü işleri bitirmeye odaklanır. Devam Eden Çalışma'nın sınırlandırılması, Kanban'ın başarılı bir şekilde uygulanması için kritiktir.





# KANBAN

**Akışı yönetin - Akış** verimliliğini gözlemleyerek ve analiz ederek, sorunlu alanları tanımlayabilirsiniz. Kanban'ı uygulamanın temel amacı, teslim sürelerini iyileştirmek ve gecikmeleri önleyerek sorunsuz bir iş akışı oluşturmaktır.

**Süreç politikalarını açık hale getirin -** Süreç, ekipteki herkes için açıkça tanımlanmalı, yayınlanmalı ve onaylanmalıdır: İnsanlar, yararlı olacağını düşünmedikçe bir şeyin parçası olmak için motive olmayacaklar. Herkes açık politikalardan haberdar olduğunda, her kişi performansınızı artıracak iyileştirmeler önerебilir.

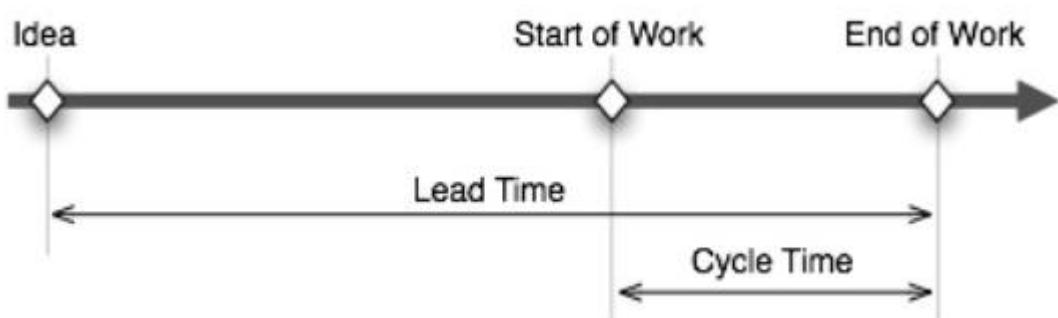




# KANBAN

**Geri bildirim döngülerini kullanın** - Olumlu değişimin gerçekleşmesi için, tüm ekibe önemli geri bildirimler sağlamak için düzenli toplantılar gereklidir. Bu toplantıların sıklığı değişir, ancak fikir, düzenli olarak, belirli bir zamanda olmalarıdır.

**İşbirliği yaparak geliştirin** - Kanban, sürekli değerlendirme, analiz ve iyileştirme gerektirir. Takımlar süreç hakkında ortak bir anlayışa sahip olduklarında, herhangi bir sorun çıkması durumunda bir fikir birliğine varma olasılıkları daha yüksektir.

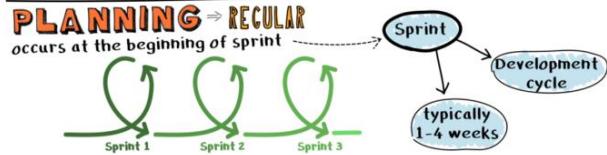




# SCRUM

VS

# KANBAN



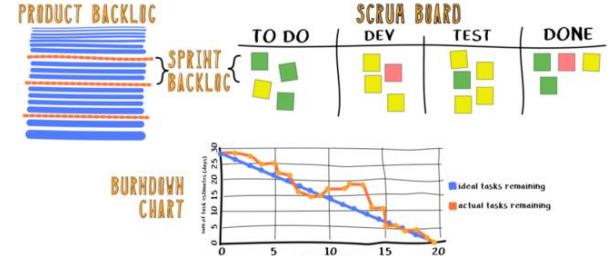
**ESTIMATIONS of TIME** ⏳  
before start of sprint  
! REQUIRED  
items should be small to finish within sprint  
if not, split them to smaller pieces

**CHANGES TO WORK SCOPE**  
should wait for next sprint



**OWNERSHIP** → Product Owner  
**WHEN TO USE**  
small items - small value  
adding increments possible  
requirements in a good shape  
roadmap is clear  
more cross-dependent teams

## BOARDS / ARTIFACTS



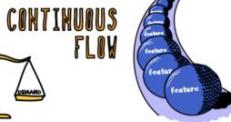
## PLANNING

NOT PRECISE planning routine

PLAN WHEN they

FINISH items

demand planning



## ESTIMATIONS of TIME

optional

when items are completed

teams simply

PULL

next item from backlog and implement it

LIMIT

how many items can be in working columns at the same time

## CHANGES TO WORK SCOPE

**ROLES AS NEEDED**  
added AS NEEDED

**MEETINGS** NONE REQUIRED

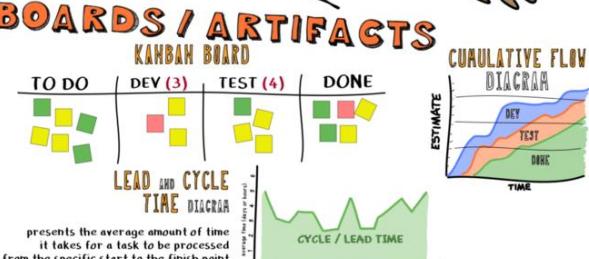
**OWNERSHIP** DEPENDS on defined roles and necessities  
who me?

## WHEN TO USE

Changes are too fast

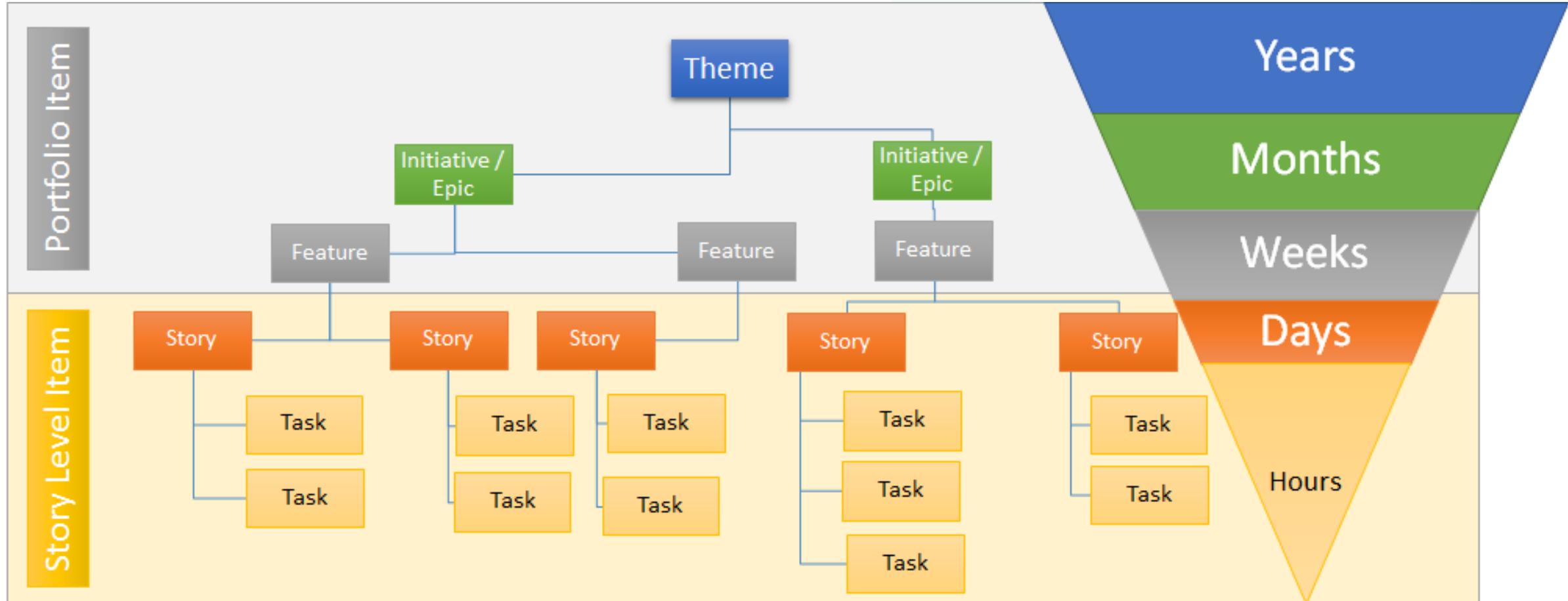


bugfix



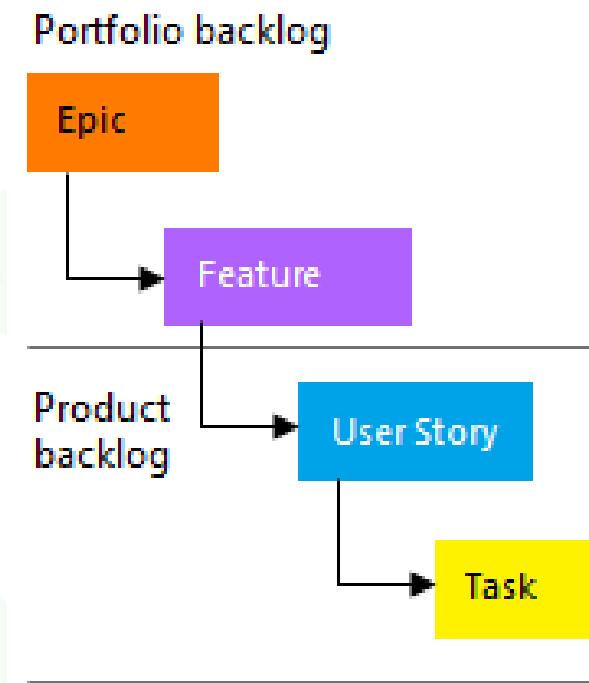
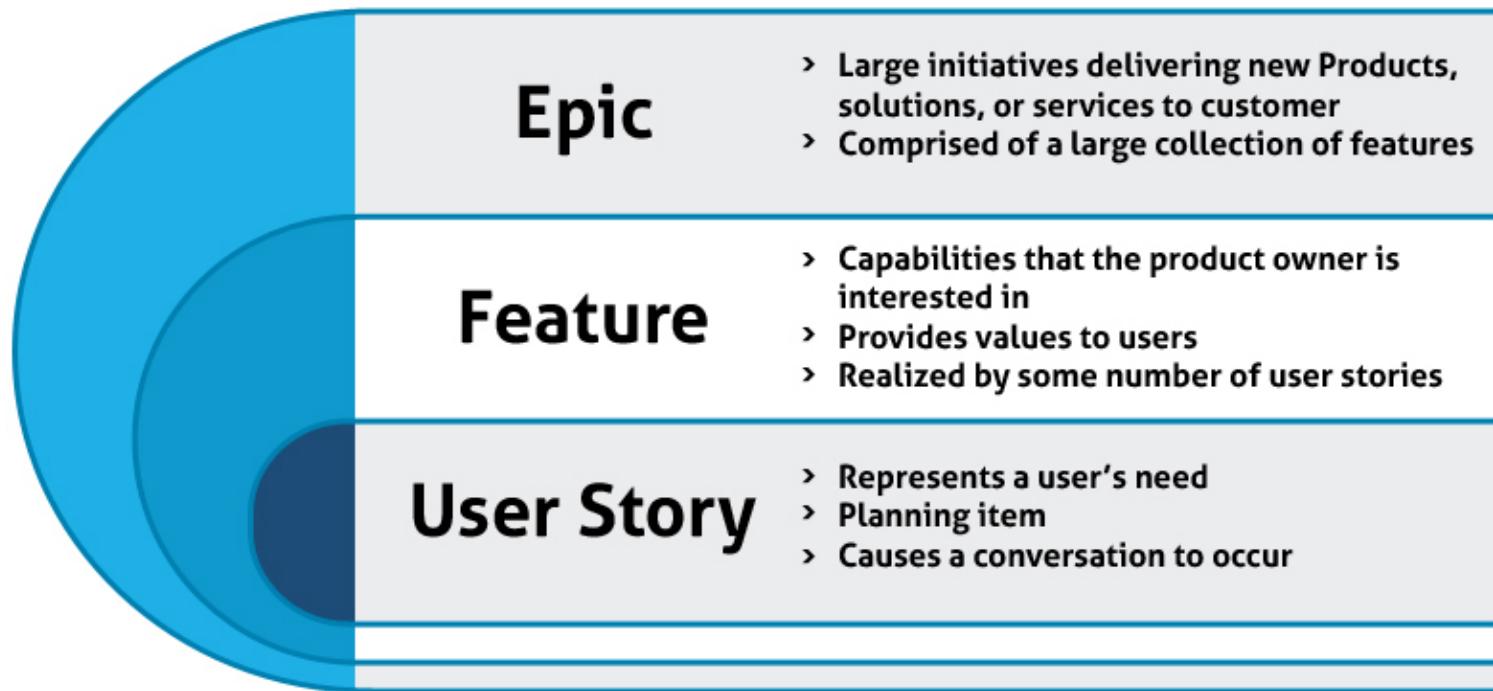


# EPICS , FEATURE , USER STORY , TASK





# EPICS , FEATURE , USER STORY , TASK





# EPIC , FEATURE NEDİR?

**EPIC** : Belirli bir vizyona ulaşabilmek için şirketlerin metrikler ile tanımlı, ulaşması gereken hedefleri vardır. Ekipler ise bu hedeflere ulaşabilmek için yeni girişimler/özellikler ile gelir ve sahip oldukları ürünü/ürünlerde yeni güncellemeler yayınlarlar. Epics, yapılması planlanan bir veya birden fazla yeni fonksiyonların yahut özelliklerin gruplanmasıdır.

Product ekibinde yapılan her şey son kullanıcıya sunulmaz veya yeni bir özellik olarak sonuçlanmaz. Bu yüzden özellikler(features) olarak değil epics olarak adlandırılırlar. Ayriyeten bir sprint'ten daha uzun süren çalışmalar da “epics” olarak adlandırılmaktadır.

**FEATURE** : Epicler içerisinde PO tarafından seçilerek Product Backlog'a eklenen, birden fazla user story içeren, birden fazla haftada tamamlanacak modüllerdir. Son kullanıcıya tamamlanmış ürün olarak ulaşır.

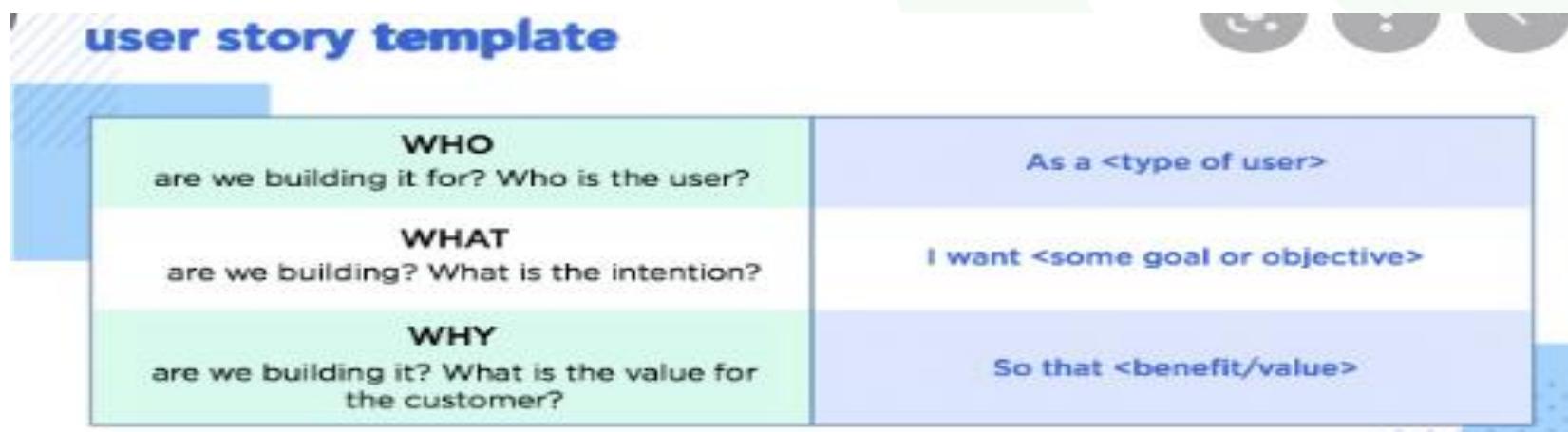


# USER STORY NEDİR?

User storyler 3 bölümden oluşur. Bunlar;

- Kullanıcının hikaye tanımı (Value statement)
- Kabul kriterleri (Acceptance criterias)
- Bitti tanımı (Definition of done)

User Stories, Agile metodolojisinin bir parçasıdır. Son kullanıcıya sunulacak herhangi bir özelliği veya ürünü açıklama yollarından biridir.





## USER STORY #1

### Hikâye Tanımı (Value Statement)

Ürün ekibi yetkilisi olarak, paket yenilemelerini arttırmabilmek için; üyelerimize paketlerinin bitmesine 1 ay kala “yenileme yapmak ister misiniz” mail gönderimi yapılmasını, mail şablonu içerisinde X, Y, Z paket detayları ve ilgili müşteri temsilcilerinin ad, soyad, iletişim bilgilerinin yer olmasını istiyorum.

### Kabul kriterleri (Acceptance criterias)

- Paket bitimine 1 ay kalan kullanıcılaraya mail gönderimlerinin başarıyla yapılması
- Üye – müşteri temsilcisi eşleşmesinin doğru yapılmış olması
- Gönderimlerin sadece mail gönderimine izin verilen kullanıcılaraya yapılması
- Mail opt-out kurgularının sağlıklı bir şekilde çalışması

### Bitti Tanımı (Definition of Done)

- Kod kontrol edildi (Code review)
- Birim testleri yapıldı (Unit tests)
- Kabul kriterleri karşılandı (Acceptance criterias)
- Product Owner(PO) story kabulü (PO accepts user story)



User Story ID	Description	Acceptance Criteria
US 0001	ATM kullanıcısı olarak, nakit çekebilmem için PIN kodumu girmek istiyorum	PIN dört basamaklı olmalıdır PIN özel karakterlere (sembollere) izin vermemelidir PIN'in 30 saniye içinde girilmesi gereklidir, aksi takdirde işlem iptal edilir
US 0002	Facebook login (Giriş sayfası) geçersiz kimlik bilgileriyle erişilmemelidir	Gecersiz kullanıcı adı ile erişim sağlanamaz Geçersiz şifre ile erişim sağlanamaz Geçersiz kullanıcı adı ve şifre ile erişim sağlanamaz

User Stories üç ortak özelliği barındırır:

- Kısa ve açıklayıcı olma:** Ürün ya da özelliğin kullanarak ne yapılmak istendiğini kısaca anlatabilmenizi sağlar.
- Kullanıcı tipini belirleme:** Kullanıcı hakkında detaylı bilgi verilmese de, kullanıcının karakteristik özelliklerinin ürün ya da yazılımla ilişkisi vurgulanır.
- Problemi tanımlama:** Kullanıcı hikayeleri, ürünün ya da yazılımın çözmesi gereken problemleri ortaya çıkarır.



WHO

WHAT

WHY

As a **role**, I want **action** (so that **benefit**)

KİM?

NE?

NEDEN?

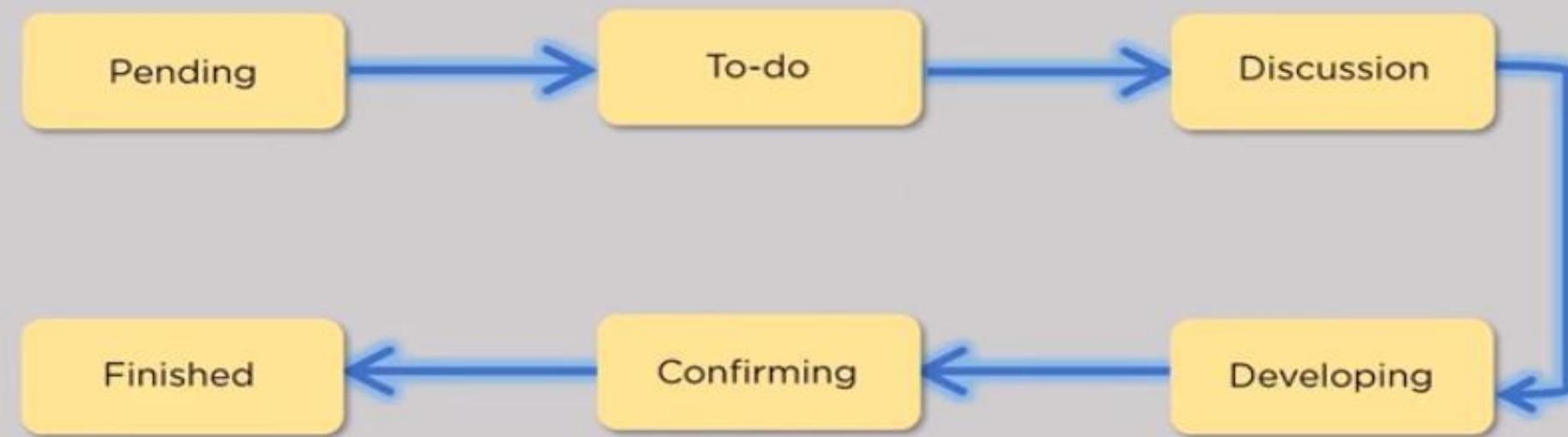
Rol olarak, **aksiyon** yapmak istiyorum, çünkü **yarar**.

- Bir kullanıcı olarak şifre oluştururken, güvenlik amacıyla, güçlü bir şifre oluşturmak istiyorum.
- Bir yönetici olarak, çalışanlarının bilgilerini, filtreleyerek daha çabuk ulaşmak istiyorum.
- Bir kullanıcı olarak birden fazla kişiye, daha hızlı olacağı için aynı anda mesaj göndermek istiyorum
- “Sıkça sinemaya giden bir izleyici olarak, yakınımdaki sinema salonlarında vizyona yeni girecek filmler hakkında bütün bilgilere telefonumdan ulaşabilmek, böylece her filmin web sitesini ayrı ayrı kontrol etmeye ihtiyaç duymamak istiyorum.”

	A	B	C	D	E
1	User_Story_ID	Description	Acceptance Criteria	Priority	Validation
2	US_001	System should allow any user to register with valid credentials validating the success message	There should be a valid SSN respecting the "-" where necessary, it should be 9 digits long	High	UI and Backend Testing
3			There should be a valid name that contains chars and cannot be blank		
4			There should be a valid lastname that contains chars and it is a required field		
5			You can provide chars and digits to describe a valid address along with zip code		
6			User should provide 10 digit-long mobilephone number as a required field respecting the "-"		
7			User cannot use just digits, but can use any chars and digits along with them and of any length		
8			You should provide a valid email format that contains "@"sign and "." extensions		
9					
10	US_002	System should not allow anyone to register with invalid credentials seeing the message "If you want to sign in, you can try the default accounts:- Administrator (login="admin" and password="admin") - User (login="user" and password="user").	Any field on the registration page should not be left blank	High	UI and Backend Testing
11			SSN number cannot be of any chars nor spec chars except "-"		
12			Mobilephone number cannot be of any chars nor spec chars except "-"		
13			email id cannot be created without "@" sign and "." extensions		
14					
15	US_003	Registration page should restrict password usage to a secure and high level passcode	There should be at least 1 lowercase char for stronger password and see the level chart change accordingly	High	UI and Backend Testing
16			There should be at least 1 uppercase char and see the level chart change accordingly		
17			There should be at least 1 digit and see the level chart change accordingly		
18			There should be at least 1 special char and see the level bar change accordingly		
19			There should be at least 7 chars for a stronger password		
20					
21	US_004	Login page should accessible with valid credentials	There should be a valid username and password validating the success message to login	High	UI and Backend Testing
22			There should be an option to cancel login		
23					
24	US_005	Login page should not be accessible with invalid credentials	User cannot login with invalid username validating the error message	High	UI and Backend Testing
25			User cannot login with invalid password validating the error message		
26			User cannot login with invalid username and password validating the error message		
27			User with invalid credentials should be given an option to reset their password		
28			User should be given the option to navigate to registration page if they did not register yet		
29					



## LIFECYCLE OF A USER STORY





# DEFINITION OF DONE (DoD) TANIMI

Development Team, kendilerinden tamamlanması beklenen işleri Product Owner'a sunmadan önce bir kalite-kontrol süreci gibi kontrol listesi hazırlar. Bu liste tüm User Story'ler için ortaktır ve biten tüm işler bu prosedürden geçmesi gereklidir. Buna kısaca "Done" veya "DoD — Definition of Done (Bitti Tanımı)" denir. Bu liste genellikle şu kontrolleri içerir:

- Unit testleri
- QA testleri
- User Story'deki tüm dökümanların güncellenmesi
- Kodun review edilmesi
- User Story kabul kriterlerinin karşılanması
- Product Owner onayı

DoD checklisti, Sprint veya Release sonunda belirli tamamlama kriterlerinin karşılanıp karşılanmadığını kontrol etmek için de kullanılmak üzere hazırlanabilir.



# DEFINITION OF DONE (DoD) TANIMI

## Definition of Done

2 / 5

Add a new item

- =  ~~Functionality described in the story is implemented and tested by the QA. Technical tasks are accepted by the Component lead.~~
- =  Change has been deployed to the test server
- =  Unit tests are written for any new code  
- =  The documentation is updated Checked by Claribel Spring on Mon Mar 25 2019 14:40:03 GMT+0100
- =  CHANGLOG entry is added for all services impacted during work on ticket.



# TAKIM KAPASİTESİ VE PUANLAMA

User storyler sprint backlog'a alınırken takim tarafından puanlanır (**POINT**)

- Sprint baslangic toplantisi veya refinement toplantisinda User Story okunur ve development team'in tum uyeleri user story'e point verir
- Verilen pointler kartlara写ilarak gösterilir
- Pointler Fibonacci Serisindeki sayilara gore verilir.

1 1 2 3 5 8 13 21 34 .....

- Her üye niçin bu point'i verdigini açıklar ve team'in ortak karariyla user story'nin point'i belirlenir
- Team'deki developer ve QA sayisina gore takım (team) kapasitesi (capacity) belirlenir.

1 point = 1 günlük çalışma yani 8 saat





# TAKIM KAPASİTESİ VE PUANLAMA

**Örnek:** 5 developer'in takım kapasitesi nedir?

$5 \times 1 = 5$  (5 developer'in günlük kapasitesi 5 point)

1 haftalık kapasitesi  $\Rightarrow 5 \times 5 = 25$  point

2 haftalık kapasitesi = 50 point

**Örnek:** 2 QA'in takım kapasitesi nedir?

$2 \times 1 = 2$  (2 developer'in günlük kapasitesi 2 point)

1 haftalık kapasitesi  $\Rightarrow 5 \times 2 = 10$  point

2 haftalık kapasitesi = 20 point

Development team'de 1 kişinin 2 haftalık sprintti yapacağı iş miktarı  $5+5=10$  point'tır.





01  
01  
01



# PROJE GELİŞTİRME SÜRECİ





# Proje Fikrinin Ortaya Çıkması

- Business ihtiyaçları
- Girişimler

Örnek senaryo: Bir bankaya ait proje





# Proje Detaylarının Belirlenmesi



**Örnek senaryo:** Bir bankaya ait proje



# Anlaşma sağlanması

- Müşteri gereksinimleri ve bekleyenler
- Zamanlama
- Bütçe
- Riskler



Örnek senaryo: Bir bankaya ait proje



# Planlama

- Proje fazlarının belirlenmesi
- Takımda çalışacak kişilerin belirlenmesi
- Riskler ve eylem planları
- Geliştirme modelinin belirlenmesi
- Eğitimlerin planlanması



Yönetici ve expert kişiler tarafından gerçekleştirilir



# Analiz

Detaylar belirlenir.....

**Talep : Yönetici olarak tüm müşteri hareketlerini ve çalışan hareketleri görebileceğim bir uygulama istiyorum**

Epic ve Featurelar ortaya konur. Planlaması yapılır

**Faz 1 : Yönetici rolü(Epic) 6ay**

- Müşteri hareketleri ekranı( Feature)
  - Para transfer işlemleri --US( User story)
    - EFT (Task)
    - Havale (Task)
    - Nakit çekme (Task)
  - Kredi talepleri
- Çalışan hareketleri ekranı (Feature)
  - Müşteri talepleri
  - Kredi onayı

**Faz 2 : Çalışan Rolü(Epic) 6 ay**

**Faz 3 : Kullanıcı Rölü(Epic)**



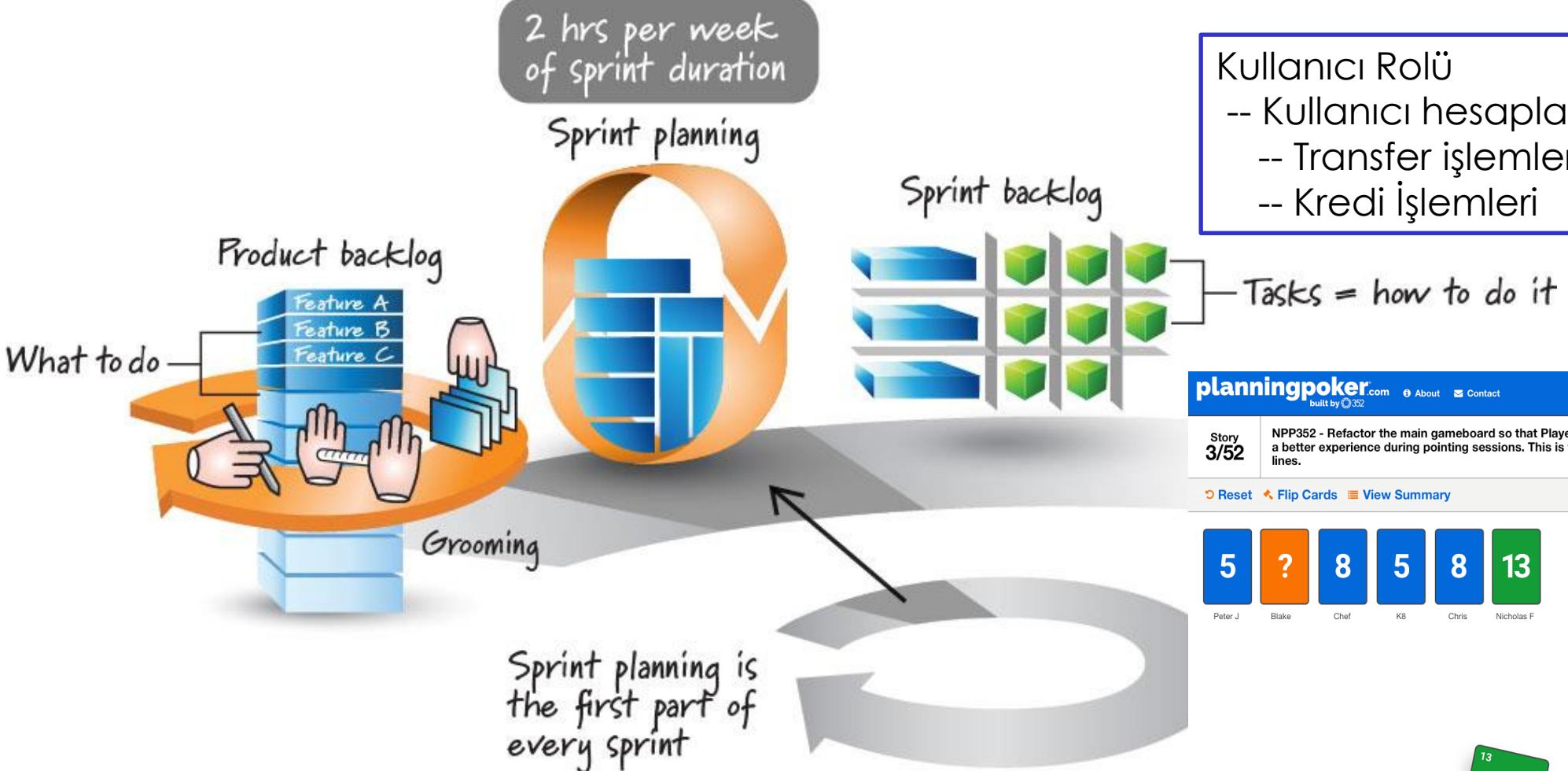
# Tasarım



- Yetkilendirilmenin yapılması
- Kullanılacak veri kaynaklarının ve bu kaynağa nasıl ulaşılacağının belirlenmesi
- Kullanıcı dostu arayüzlerin tasarlanması
- Sistem mimarisinin belirlenmesi
- Çalışılacak ortamların ayağa kaldırılması



# Kodlama ve Test



## Kullanıcı Rolü

- Kullanıcı hesapları ile işlem yapabilmeli
- Transfer işlemleri
- Kredi işlemleri

Tasks = how to do it

planningpoker.com built by 352 [Dashboard](#) [Log Out](#)

Story 3/52 3:52 [+1min](#)

NPP352 - Refactor the main gameboard so that Players and Organizers can have a better experience during pointing sessions. This is to show a story on three lines.

[Reset](#) [Flip Cards](#) [View Summary](#) [Prev](#) [Skip](#)

Players	Stories
Peter J	5
Blake	?
Chef	8
K8	5
Chris	8
Nicholas F	13

Velocity 138 / 352

13 Edit Score

Move	#	Story	Value
↓	1	Story Title Here	8
↑	2	Story Title Here	8
↑	3	Story Title Here	8
↑	4	Story Title Here	8
↑	5	Story Title Here	8
↑	6	Story Title Here	8
↑	7	Story Title Here	8
↑	8	Story Title Here	8
↑	9	Story Title Here	8

13 20 40 100 ? Pass ☕ ☕

352 Need UX, design or product development support? Check us out!

[Invite Players](#) [Add Story](#)

Privacy Policy | Terms of Use



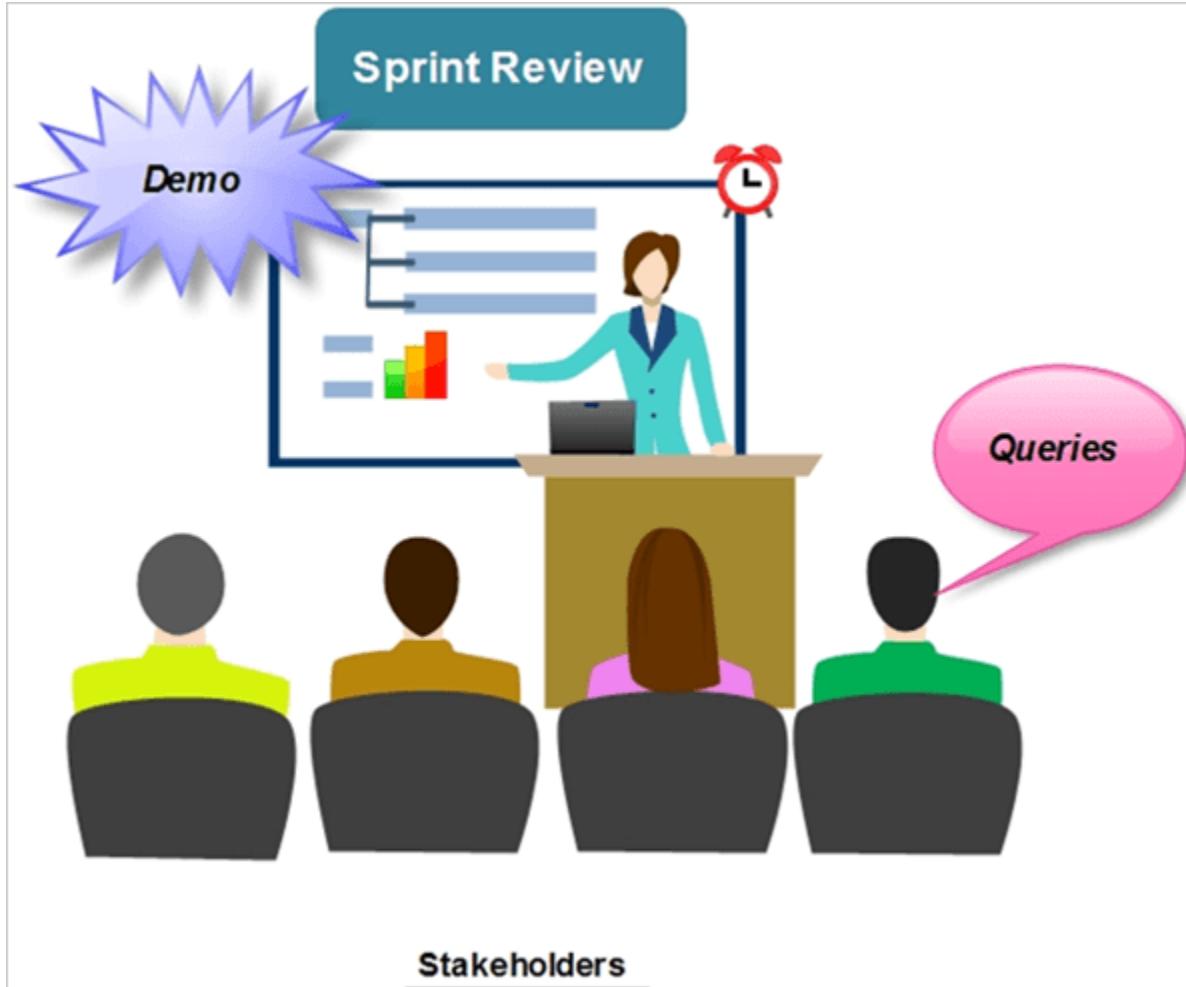
# Kodlama ve Test



Sprint boyunca hergün Daily ler yapılarak yukarıdaki sorular yanıtlanır.  
Amaç oluşabilecek olumsuz durumları hemen ortadan kaldırılmaktır.



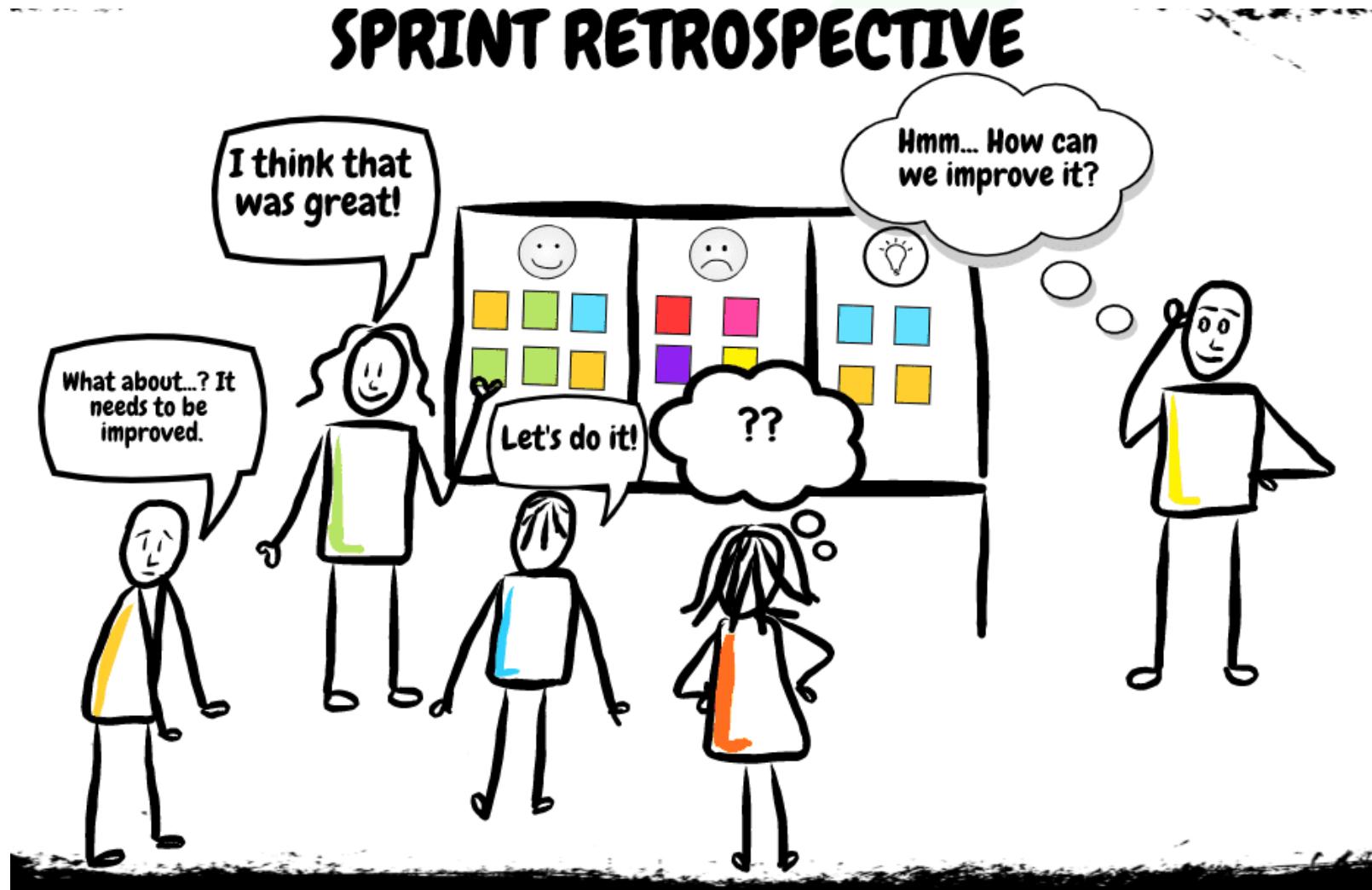
# Kodlama ve Test



- Geliştirmesi ve testi tamamlanmış PBI ların müşteriye sunumu
- Müşteriden feedback alınması



# Kodlama ve Test





# DEPLOYMENT

