NESNE YÖNELİMLİ PROGRAMLAMA(Object Oriented Programming/OOP)

Öğr. Gör. Celil ÖZTÜRK
Marmara Üniversitesi
Teknik Bilimler Meslek Yüksekokulu

İçerik

- Gereksinim Analizi
- Use Case Diyagramları
- Belirtim Dökümanı
- Yazılım Yaşam Döngüleri
- Çevik(Agile) Yazılım Geliştirme Metodolojisi
- Scrum

Gereksinim Analizi/Gereksinim Mühendisliği

Gereksinim mühendisliği, üzerinde çalışılmaya başlanacak projenin amaçlarını, boyutlarını ve etkilerini belirlemeye yönelik yapılan çalışmalardır.

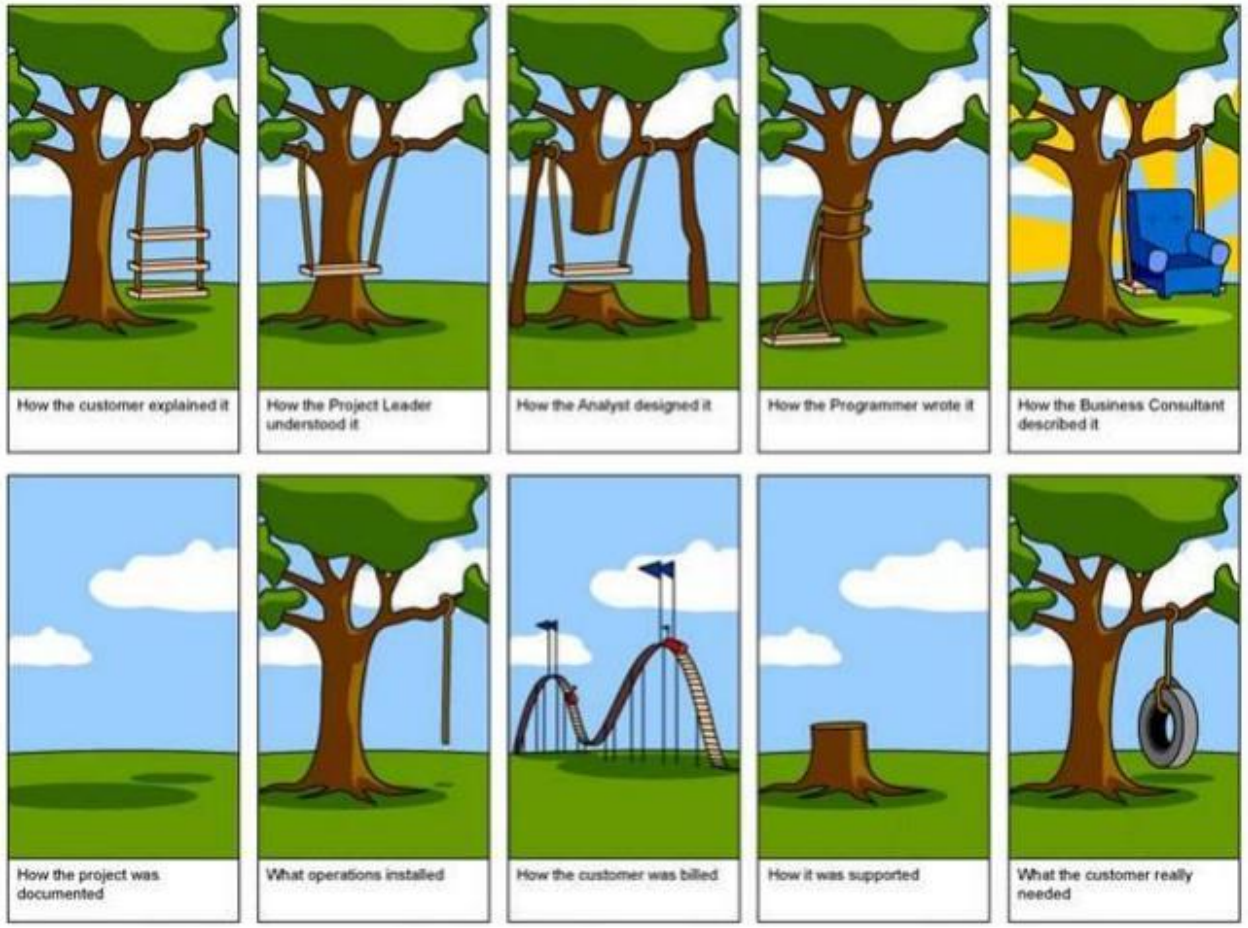
- Müşteri ne istediğini bilmez mi? Gereksinimler zaten belli değil mi?

Gereksinim Analizi/Gereksinim Mühendisliği

Müşteri,

- Çoğunlukla müşterinin kafasında sadece genel bir fikir vardır.
- Yoruma açık ve ayrıntıları kesin çizgilerle belirlenmemiş gereksinimler projenin başarısızlığına davetiye çıkarır.
- Kesin belirlenmiş gereksinimler bile zaman içerisinde değişebilir

Gereksinim Analizi/Gereksinim Mühendisliği



Gereksinim Analizi/Gereksinim Mühendisliği

**Yanlış veya eksik işi yapan mükemmel yazılım değil, doğru işi yapan iyi çözüm gereklidir.

- Tanımlanmış gereksinimlerin tutarsızlıklara karşı sağlaması yapılır.
  - Gereksinimler açıkça ve yoruma yer bırakmayacak şekilde tanımlanır.
  - Birbiri ile çelişen gereksinimler olmamalı
  - Eksik gereksinimler giderilmeli.

Gereksinim Analizi/Gereksinim Mühendisliği

Yazılım geliştirme sürecinde gereksinimlerde değişiklik olabilir !

- Yeni gereksinimler eklenmesi,
- Mevcut gereksinimlerden bazılarının geçerliliğini yitirmesi,
- Gereksinimlerin önem sıralamasının değişmesi,
- Hatalı kestirimlerden dolayı bazı gereksinimlerden vazgeçilmesi.

Use Case

- Yazılım ürünü (product) ile yazılım ürününün kullanıcısı (aktör) arasındaki etkileşimi modelleyen yapıya “use-case”denir.
  - Örneğin; bir banka sistemi; süreç için müşteri ve veznedar olmak üzere iki aktör vardır.

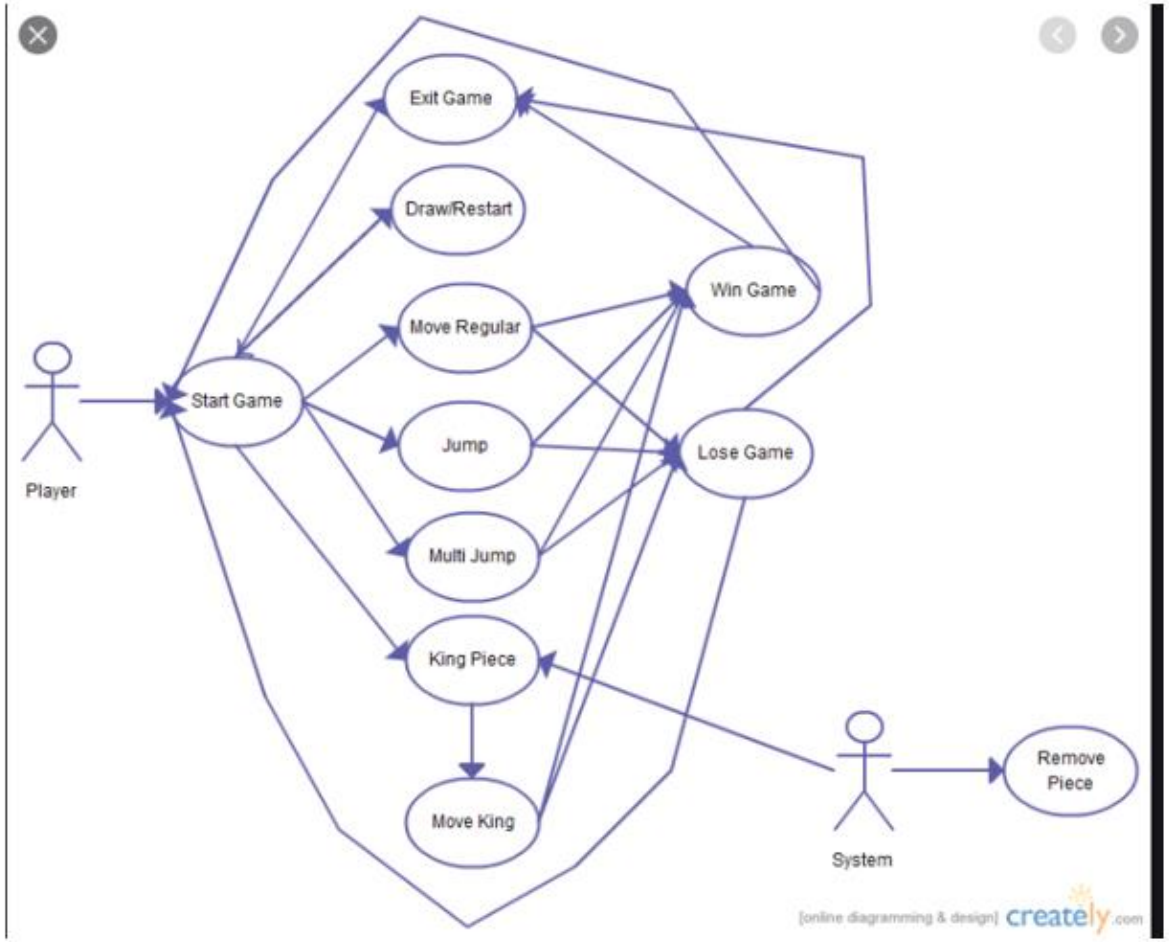
Use Case

- Aktör, yazılım ürünü dışındaki dünyanın bir üyesidir.
- Genellikle bir aktörü belirlemek kolaydır.
  - Bir aktör çoğunlukla yazılım ürününün kullanıcısıdır.

Genelde, aktör yazılım ürününde rol alır. Bu roller,

- Kullanıcı (user),
- Başlatıcı (initiator), ön ayak olan kimse,
- Use case'in kritik bir kısmında rol oynayan kimse.

Use Case

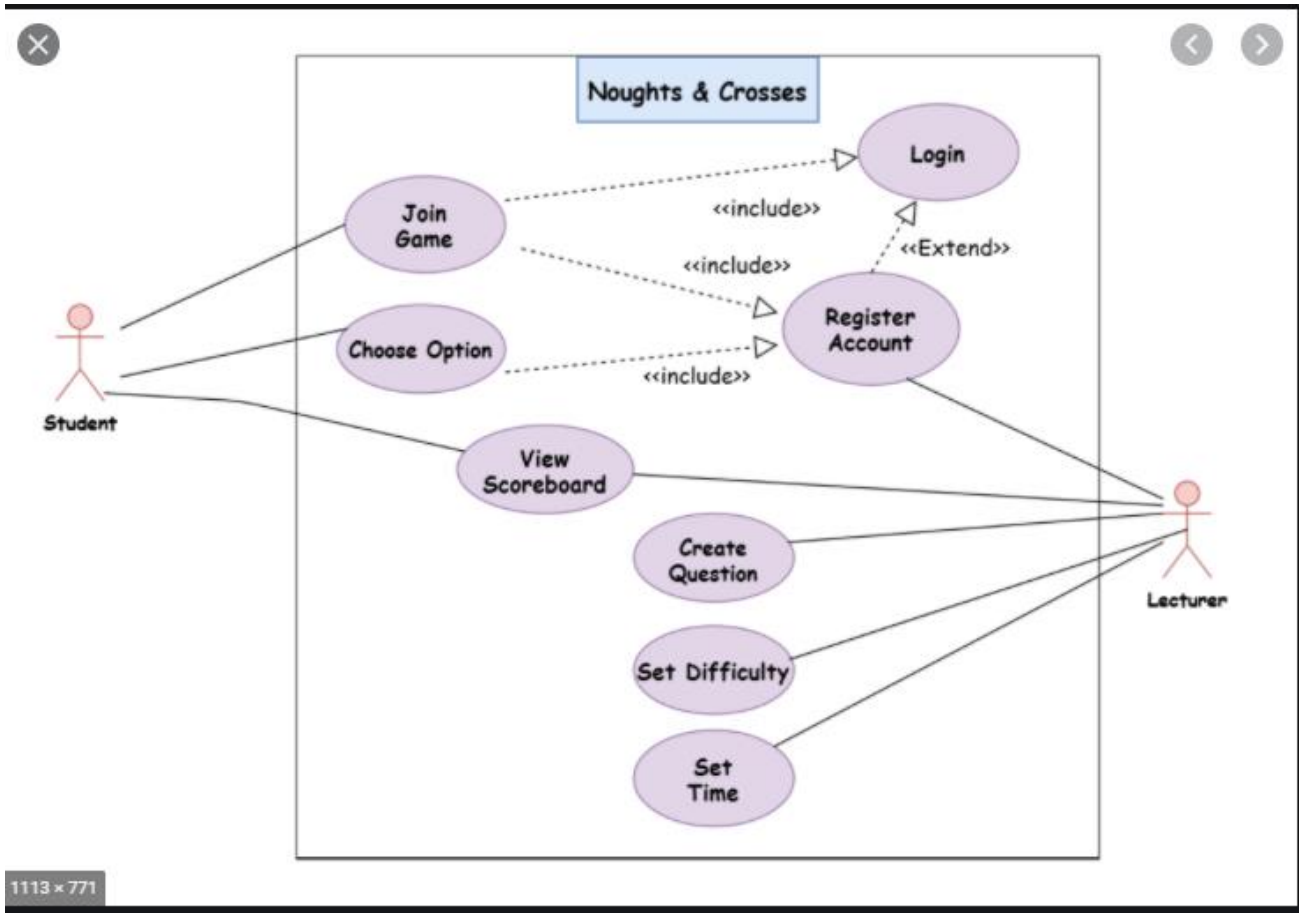


Use Case 1: Start Game use-case'i,
Kısa açıklama: Oyunu başlatmayı sağlar.
Adım adım açıklama: Oyunu başlat.

Use Case 2: Jump use-case'i , karakterin x birim kuvvetle zıplamasını sağlar.

Use Case 3: ….
Game Case :…..

Use Case


Noughts & Crosses
Login
Join Game
<<include>>
<<Extend>>
<<include>>
Register Account
Choose Option
<<include>>
Student
View Scoreboard
Create Question
Lecturer
Set Difficulty
Set Time
1113 × 771

Belirtim Dökümanı 1/2

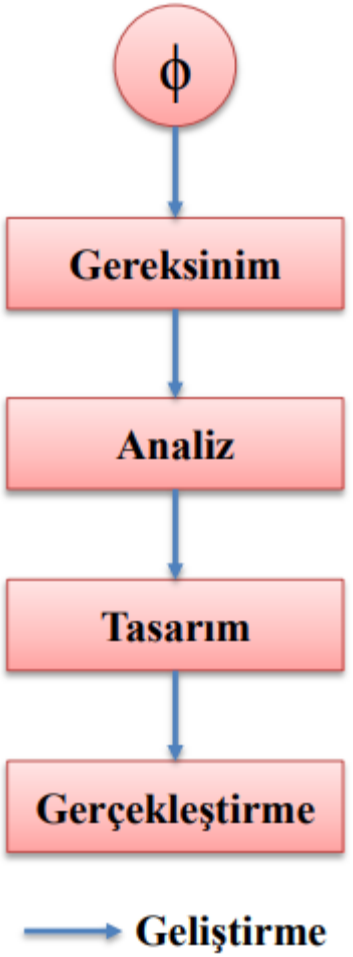
- Belirtim dokümanı (specification document) müşteri ile yazılımcı arasındaki bir sözleşmedir. Resmi sözleşmenin parçasıdır.
- Belirtim dokümanı, müşterinin rahat anlayacağı açıklık ve netlikte olmalı, özel dil kullanımından kaçınılmalı.
  - Müşteri bir bilgisayar/yazılım mühendisi varsayılmamalı.
- Bu doküman tasarımcılar için hiçbir yanlış anlamaya yer vermeyecek kadar formal (resmi) olmalıdır.

Belirtim Dökümanı 2/2

- Belirtim dokümanı (specification document) yazılımın geliştirilmesinde ilk önemli adımdır. Buradaki yanlışlıkların ilerde düzeltilmesi maliyetli olur.
- Bu doküman mümkün olduğunca, kolay anlaşılır, net, kesin, ve eksiksiz olmalıdır.
- Belirtim dokümanının önemli bileşenlerinden biri kabul kriterlerine ait şartlardır.
  - Sistem hangi şartlar altında müşteri tarafından kabul edilebilir?
- Yazılım ürünü birçok önemli testten geçer. Eğer yazılım ürünü bu testlerden başarılı bir şekilde geçerse, belirtim dokümanı yerine getirmiş sayılır.

Yazılım Geliştirme

- Teorikte yazılım geliştirme, bir işe sıfırdan başlayarak, gereksinimler, analiz, tasarım ve gerçekleştirme adımlarının doğrusal olarak tanımlanmasıyla gerçekleşmektedir.
- Pratikte yazılım geliştirme iki nedenden dolayı oldukça farklıdır.
  - Yazılım geliştiriciler hata yapabilirler.
  - Yazılım ürünü geliştirilirken, müşterinin gereksinimleri değişebilir.



Yazılım Yaşam Döngüsü

- Bir yazılımın geliştirilmesi ve bakımı süresince icra edilen adımlar topluluğuna **yaşam–döngü modeli** denir.
- Yazılım sektöründe yaygın olarak kullanılan yazılım yaşam döngü modelleri:
- Çağlayan Yaşam-Döngü Modeli
- V Süreç Modeli
- Helezonik (Spiral) Model

Yazılım Yaşam Döngüsü

- Yazılım yaşam-döngüsü herhangi bir yazılımın üretim ve kullanım aşaması birlikte olmak üzere geçirdiği tüm aşamalar olarak tanımlanır.
- Yazılım yaşam-döngüsü, yazılım işlevleri ve ihtiyaçları sürekli değiştiği ve geliştiği için mutlaka bir döngü biçiminde düşünülmelidir. Tek yönlü yani doğrusal olarak kesinlikle düşünülmemelidir.

Yazılım Yaşam Döngüsü/Adımlar 1/2

- Gereksinim(Requirements):
  - Müşteri gereksinimlerinin elde edildiği ve fizibilite çalışmasının yapıldığı aşamadır.
- Analiz: Sistem gereksinimlerinin ve işlevlerinin ayrıntılı olarak incelendiği aşamadır. Var olan işler incelenir, temel sorunlar ortaya çıkarılır
- Tasarım(Design):Belirlenen gereksinimlere yanıt verecek yazılım sisteminin temel yapısının oluşturulduğu aşamadır.
  - Mantıksal tasarım; önerilen sistemin yapısı (modüller, akış şemaları…)
  - Fiziksel tasarım; yazılımı içeren bileşenler ve bunların ayrıntıları (veri yapıları, ekran tasarımları…)

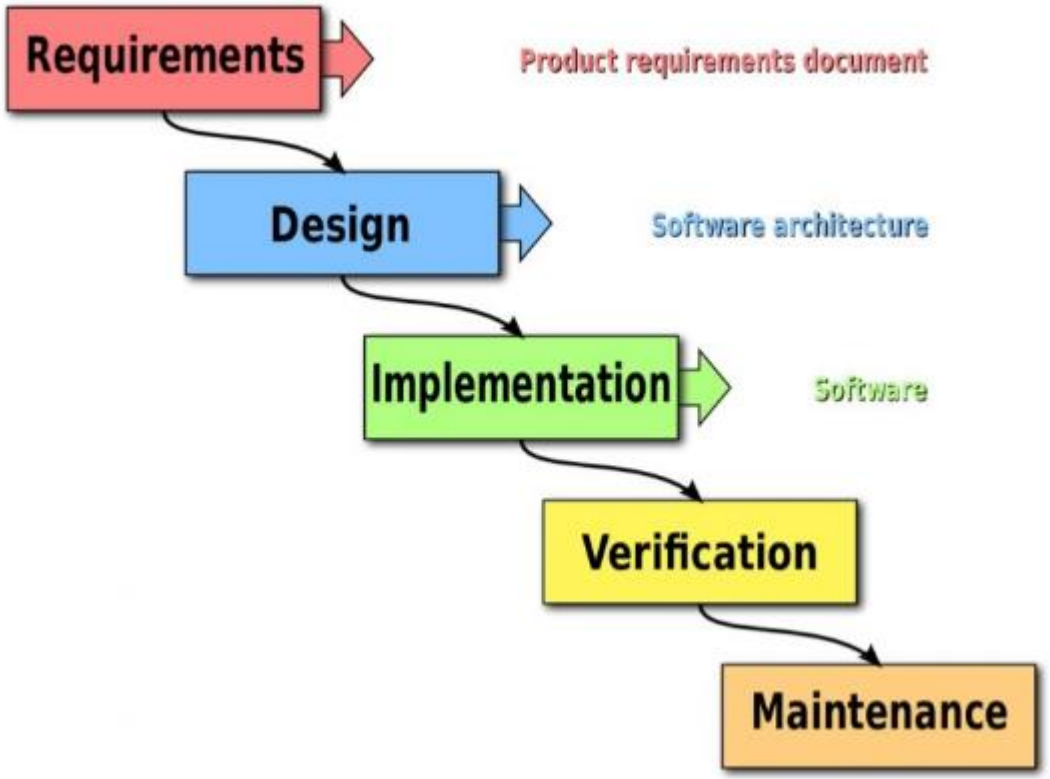
Yazılım Yaşam Döngüsü/Adımlar 2/2

- Gerçekleştirme(Implementation): Modüllerin kodlandığı, birleştirildiği, test edildiği ve kurulum çalışmalarının yapıldığı aşamadır.
- Bakım (Maintenance):Hata giderme ve yeni eklentilerin eklendiği aşamadır (teslimden sonra).

Çağlayan Yaşam Döngüsü(Waterfall)

- Çağlayan modeli yakın zamanlara kadar en popüler yazılım geliştirme modeli olarak görülmüştür.
- Geleneksel yazılım geliştirme modeli olarak da bilinir.
- Çağlayan modelinde yazılım, aşamaların en az birer kez tekrarlanması ile geliştirilir.
- Çok iyi tanımlanmış ve üretimi az zaman gerektiren projeler için uygun bir model olmakla birlikte günümüzde kullanımı gittikçe azalmaktadır.

Çağlayan Yaşam Döngüsü(Waterfall)

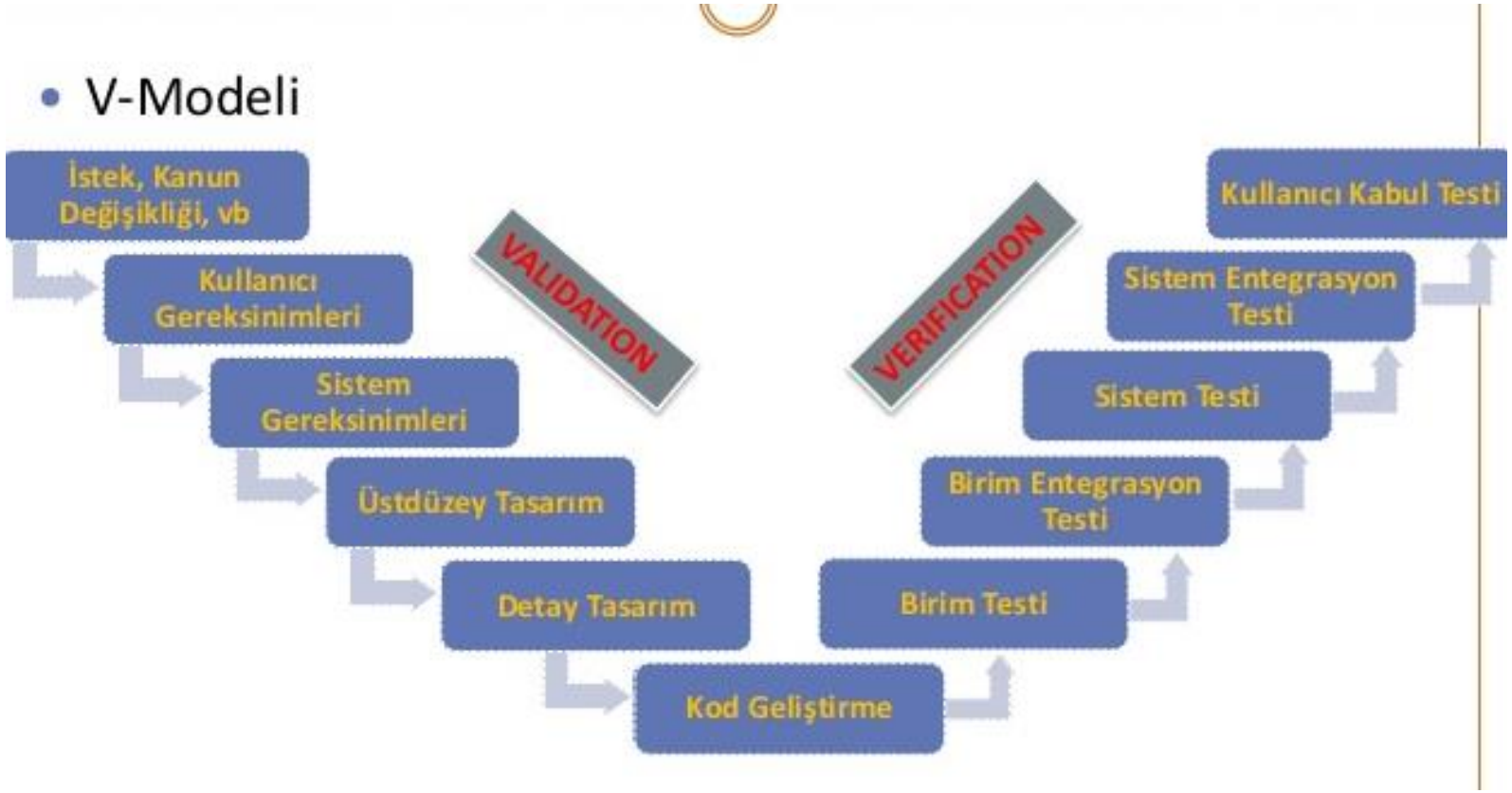


Waterfall Model Nedir? Özellikleri Nelerdir? » TechWorm (tech-worm.com)

Çağlayan Yaşam Döngüsü(Waterfall)

- Mümkün olan tüm detayın tasarıma yansıtılabilmesi için müşteri ve sistem gereksinimlerinin en ince ayrıntısına kadar belirlenmesi gerekir
- Tasarım aşaması da, yazılımın tüm gereksinimlerini karşılayacak şekilde detaylı bir çalışma gerektirmektedir.
- Dolayısı ile, çağlayan yaşam-döngü modelini kullanan proje ekipleri en fazla zamanı bu iki aşamada(gereksinim analizi, tasarım) harcamak zorundadırlar.
- Tüm bu efor ve detaylı çalışmalara rağmen özellikle uzun zamana yayılan projelerde gereksinimlerin değişecek olması kaçınılmazdır.
- Kodlama veya test aşamalarında olabilecek bu değişikliklerin sisteme/yazılıma yansıtılması maliyeti ise çok yüksektir.

V Süreç Modeli

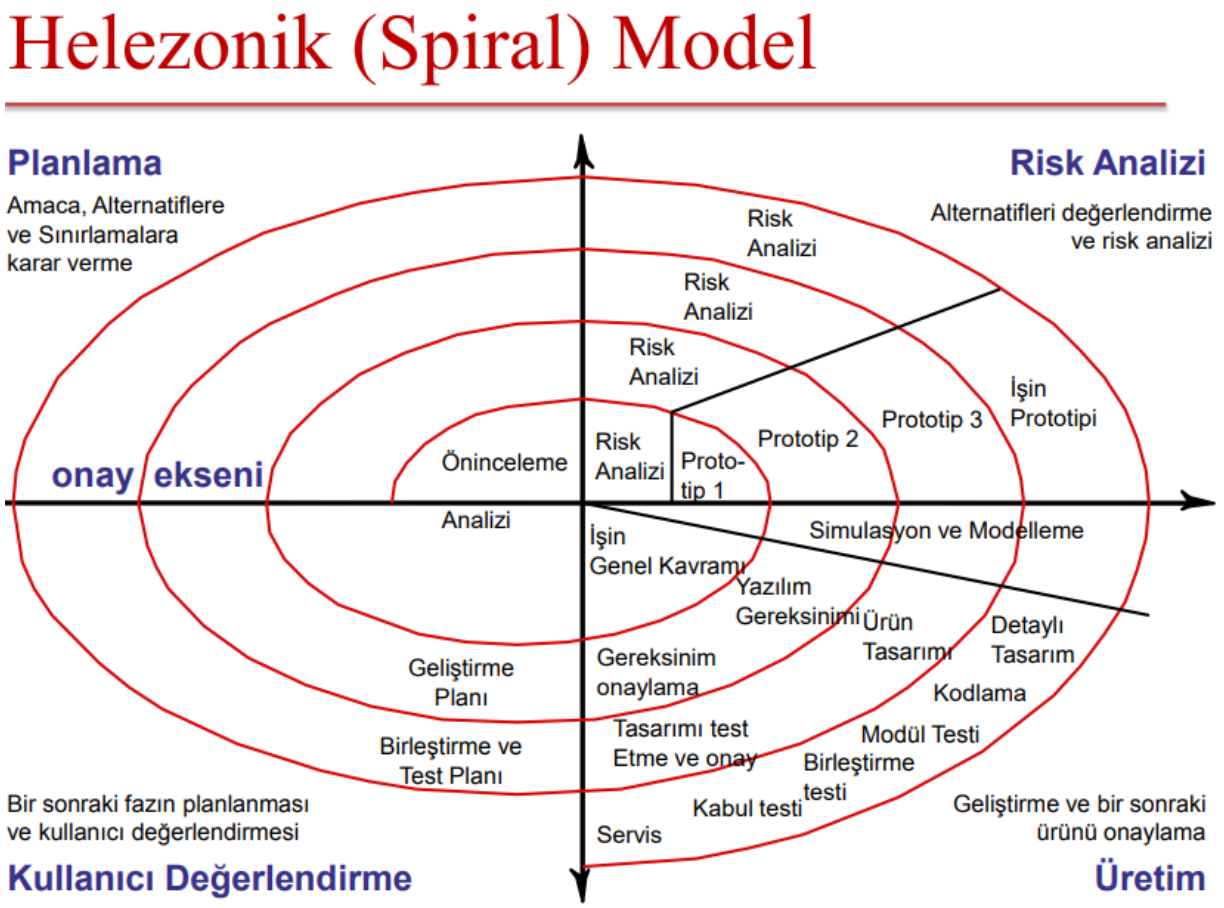


Yazılım Süreç Yönetim Modelleri ve Karşılaştırılması – Fikir Jeneratoru

V Süreç Modeli

- Sol taraf üretim, sağ taraf sınama işlemleridir.
- Belirsizliklerin az, iş tanımlarının belirgin olduğu BT projeleri için uygun bir modeldir.
- Model, kullanıcının projeye katkısını arttırmaktadır.

Helezonik(Spiral) Model



Helezonik(Spiral) Model

- Risk analizi olgusu ön plana çıkmıştır.
- Her döngü bir fazı ifade eder. Doğrudan gereksinim, analiz, tasarım,... vs. gibi bir faz yoktur.
- Yinelemeli artımsal bir yaklaşım vardır.
- Prototip yaklaşımı vardır.
- Süreç arka arkaya devam eden sıralı aktiviteler şeklinde gösterilmek yerine spiral şekilde gösterilir.

Helezonik(Spiral) Model Avantajları

- Kullanıcılar sistemi erken görebilirler.
- Geliştirmeyi küçük parçalara böler . En riskli kısımlar önce gerçekleştirilir.
- Pek çok yazılım modelini içinde bulundurur.
- Riske duyarlı yaklaşımı potansiyel zorlukları engeller.
- Seçeneklere erken dikkate odaklanır.
- Hataları erken gidermeye odaklanır.

Helezonik(Spiral) Model Dezavantajları

- Komplekstir (karmaşık).
- Spiral sonsuza gidebilir.
- Ara adımların fazlalığı nedeniyle çok fazla dokümantasyon gerektirir.
- Büyük ölçekte projelerde kullanılır.
- Küçük ve düşük riskli projeler için pahalı bir yöntemdir.

Çevik(Agile) Yazılım Geliştirme

- Yazılım geliştirme süreci sıkıntılı ve uzun süren bir dönemdir.
- Yazılım sektöründe, yazılım sürümlerinin zamanında ortaya çıkarılamaması, değişiklik isteklerine çabuk cevap verilememesi, yazılım hatalarının geç fark edilmesi ve zaman içerisinde gelen isteklere göre sistemin kendi yapısını geliştirememesi gibi çeşitli sorunlar ortaya çıkmıştır.
- Bu sorunların aşılmasına yönelik yapılan çalışmalar sonucu, 1990'lı yılların sonlarına doğru "çevik (agile)" olarak isimlendirilen metotlar geliştirilmiştir.

Çevik(Agile) Yazılım Geliştirme

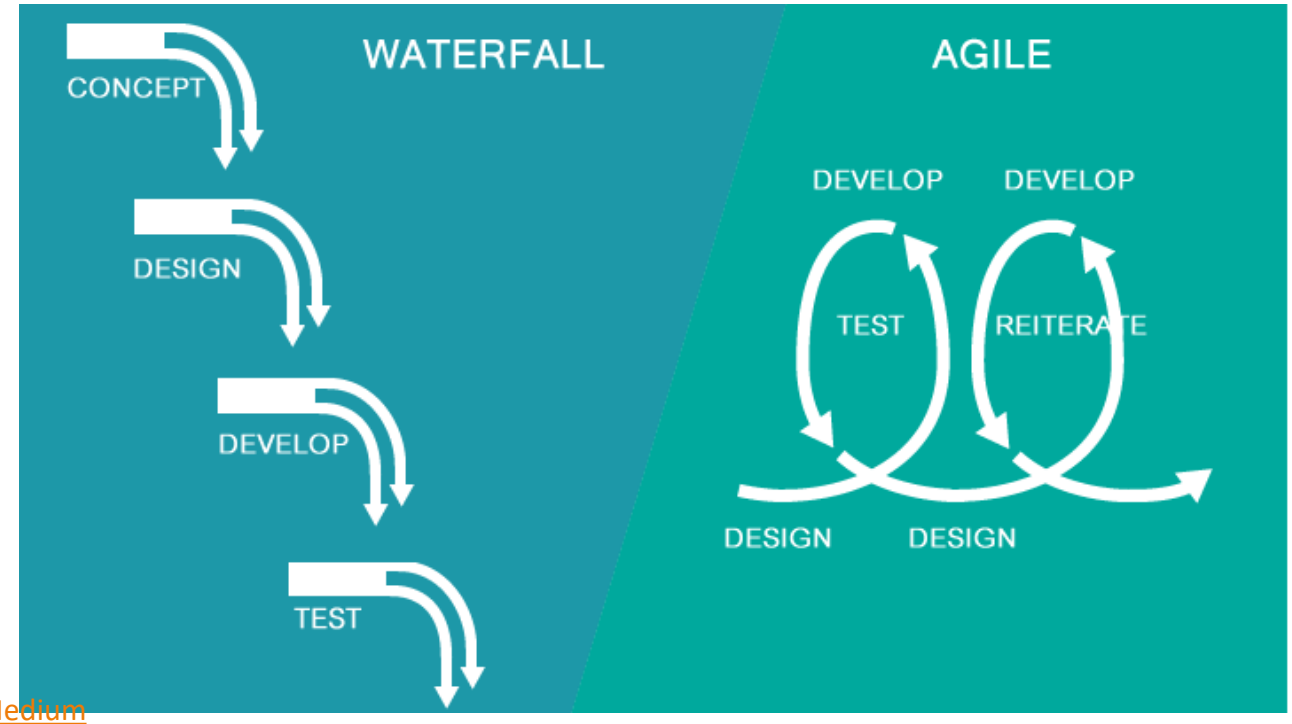
- Çevik yazılım geliştirme metotları,
  - piyasaya çok çabuk ürün çıkarabilme,
  - değişen isteklere hızla yanıt verme ve
  - en kısa sürede bir yazılım ürününü müşteri hizmetine sunmayı amaçlamaktadırlar.
- Çevik yazılım geliştirme metotları, verimliliği yüksek, esnek, hata oranı düşük, hızlı ve ucuz çözümler sağlamaktadırlar.

Çevik(Agile) Yazılım Geliştirme

- Bu metodolojide projenin ölçeği ne olursa olsun, proje küçük yinelemelere ayrılır ve her yineleme başlı başına bir proje gibi ele alınarak geliştirilir.
- Her yinelemenin (iteration) sonunda da proje ekibi tarafından müşteriye, projenin ne kadarının gerçekleştirildiğine dair bilgi verilir.
- “Çevik” ile her bir yinelemenin 2-4 hafta kadar sürmesi planlanmaktadır.
  - Her yinelemenin kendi içerisinde çalışan bir sistem olması sonucu müşteriye sürekli çalışan bir yazılım teslim edilerek, müşteri memnuniyetinin artması sağlanmaktadır.

Çevik(Agile) Yazılım Geliştirme

"Çevik" in hızı proje ekibinde çalışan tüm ekip üyelerinin sürekli iletişim halinde olmasından kaynaklanmaktadır.



Manajemen Proyek: Waterfall atau Agile? Mana lebih baik? | by Sigit | SkyshiDigital | Medium

Agile Temel Prensipler 1/2

Çevik manifestonun altında yatan temel prensipler şunlardır;

- Birinci öncelik, müşteriye hızlı ve sürekli olarak kullanılabilir yazılım teslimatı yapmaktır.
- Kodlamanın ilerleyen safhalarında bile gereksinim değişiklikleri kabul edilir, esneklik vardır.
- Mümkün olduğunca kısa zaman aralıklarında çalışan, kaliteli yazılım teslimatı yapılır.
- Analistler, uzmanlar, yazılımcılar, testçiler vs. tüm ekip elemanları birebir iletişim halinde, birlikte çalışırlar.

Agile Temel Prensipler 2/2

Çevik manifestonun altında yatan temel prensipler şunlardır;

- Ekip içerisinde kaliteli bilgi akışı için yüz yüze iletişim önemlidir.
- İlerlemenin birincil ölçüsü çalışan bir yazılımdır.
- Çevik süreçler, sürdürülebilir geliştirmeyi destekler.
- Sağlam teknik alt yapı ve tasarım, çevikliği arttırır.
- Basitlik önemlidir.
- Düzenli aralıklarla ekip kendi yöntemlerini gözden geçirir ve verimliliği arttırmak için gerekli iyileştirmeleri yapar.

Çevik Yazılım Geliştirme Avantaları

- Yazılım ekibinin motivasyonu sürekli yüksek seviyede olur.
- Kısa sürede müşteri memnuniyeti sağlanır.
- Üretkenlikler artar.
- Yazılım kalitesi artar.
- Maliyetler düşer.
- Yazılım projelerinin başarısı %55'e kadar artış gösterebilir.

Çevik Metodolojiler

- Extreme Programming (XP),
- SCRUM,
- Agile Unified Process,
- Feature-Driven Development (FDD),
- Test-Driven Development (TDD)
- LEAN Development,

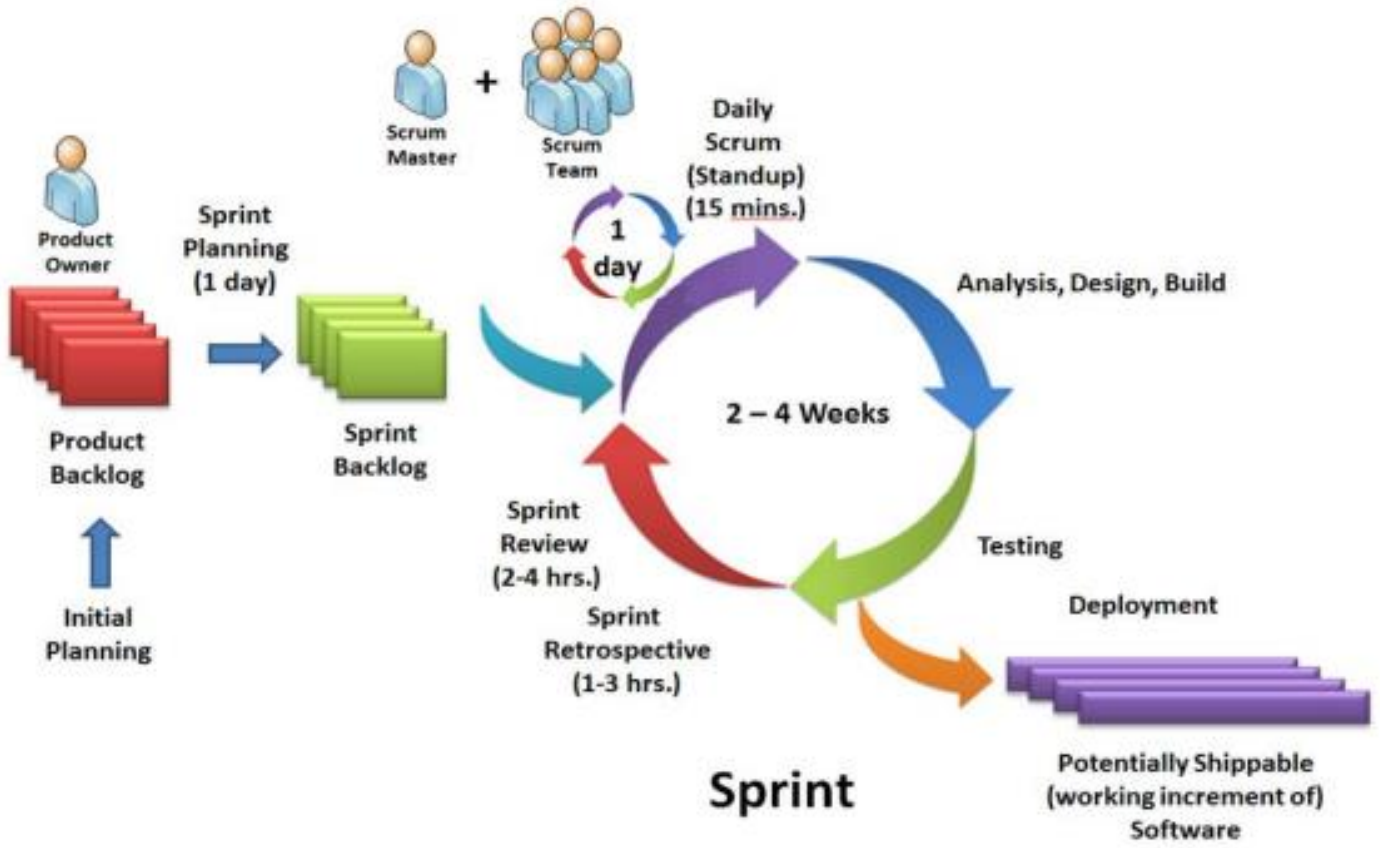
SCRUM

- 1990’ların ortalarında geliştirlen çevik yazılım geliştirme motodolojileriyle uygulanabilecek bir yaklaşımdır.
- Karmaşık yazılım işlerini küçük birimlere (sprint) bölerek geliştirmeyi öngörür.
- Bu metodoloji, karmaşık ortamlarda adım adım yazılım geliştiren küçük ekipler için uygundur.
- Gereksinimlerin kolaylıkla tanımlanamadığı ve kaotik durumların beklendiği projeler için en uygun metodolojidir.
- Bu metodolojide bir yinelemenin tamamlanması 30 günden fazla sürmemekte ve günlük 15 dakikalık toplantılarla sürekli iş takibi yapılmaktadır.

SCRUM


Günlük toplantı
DAILY SCRUM MEETING
24 saat
24 HOURS
Çalışan bir sürüm
Yapılacak işler
PRODUCT BACKLOG
SPRINT BACKLOG
POTENTIALLY SHIPPABLE PRODUCT INCREMENT
2 - 4 Hafta
2-4 WEEKS
Aktif yapılacaklar

SCRUM


Scrum Master
+
Scrum Team
Daily Scrum (Standup) (15 mins.)
1 day
Product Owner
Sprint Planning (1 day)
Analysis, Design, Build
Product Backlog
Sprint Backlog
2 – 4 Weeks
Sprint Review (2-4 hrs.)
Testing
Initial Planning
Sprint Retrospective (1-3 hrs.)
Deployment
Sprint
Potentially Shippable (working increment of) Software

Scrum

- Scrum Yöneticisi (Scrum Master)
  - Takımın Scrum'ın temel değerlerine, pratiklerine ve kurallarına bağlı kalmasını garanti altına alır.
  - Takımı ve organizasyonu Scrum'a adapte eder.
- Scrum Takımı (Scrum Team)
  - Scrum Takımı, devamlı iletişim halinde olan ve tek bir hedefe ulaşmak için mücadele eden kişilerden oluşur

Scrum

- Toplantılar(Meetings)
  - Sprint (Koşu) Planlama (Sprint Planning)
    - Geniş kapsamlı gereksinim listesinin çıkarılması.
    - Başarılı geliştirme için uygun dağıtım gereksinimlerinin belirlenmesi
    - Dağıtımlar için gereksinimlerin eşleştirilmenin yapılması
    - Dağıtımlar için takımların belirlenmesi – Risk değerlendirmesi ve risk kontrollerinin belirlenmesi
    - Gözden geçirmeler ve olası gereksinim değişikliklerinin belirlenmesi
    - Geliştirme araçları ve altyapısının onaylanması
    - Dağıtım, geliştirme ve pazarlama maliyetlerinin hesaplanması
    - Yönetimi ve destekleri gözden geçirme ve onaylama

Scrum

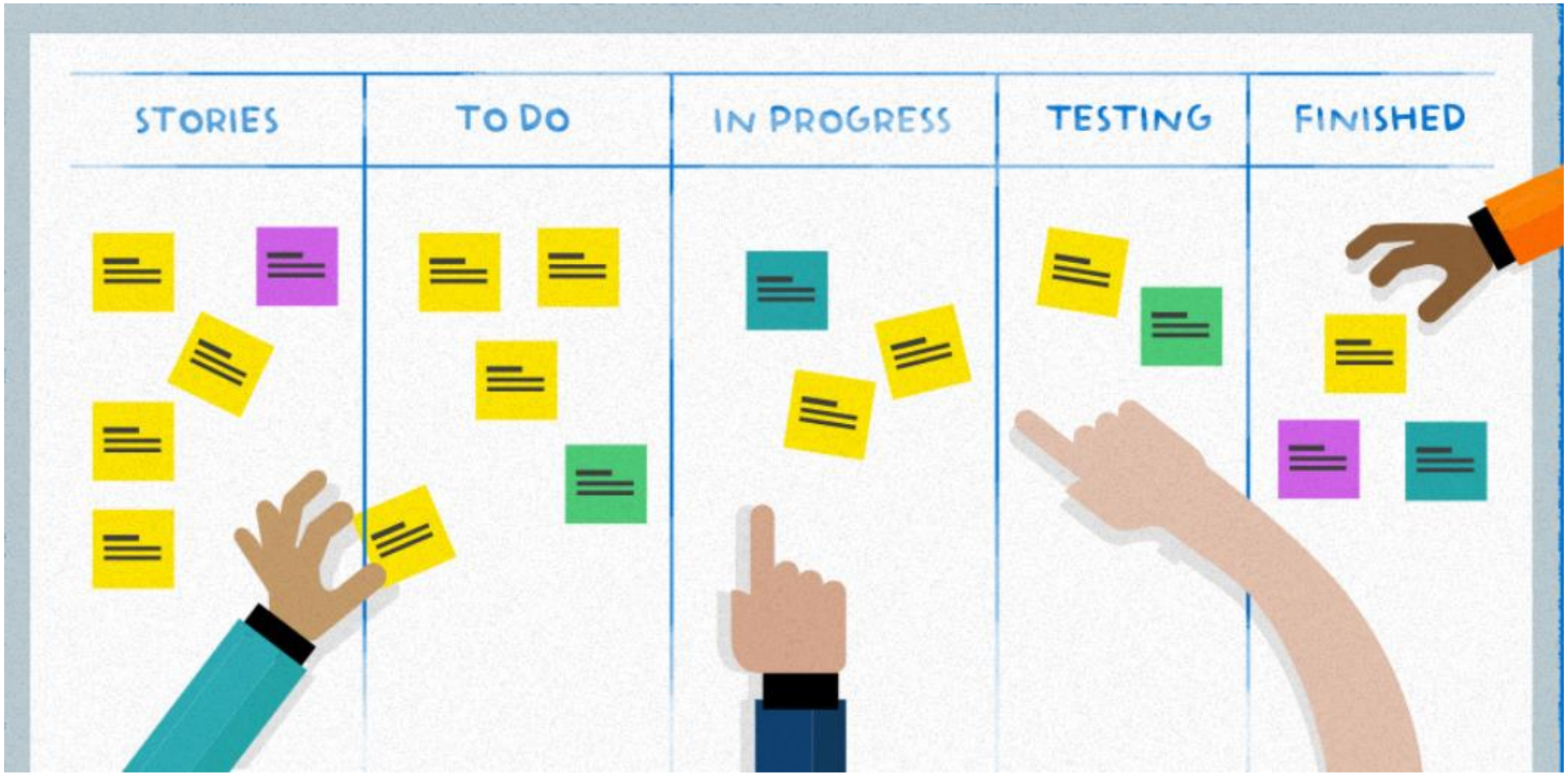
- Toplantılar(Meetings)
  - Günlük Scrum Toplantısı (Daily Scrum Meeting)
    - Koşu başladıktan sonra takım sürecin başka bir anahtar aktivitesi olan Günlük Scrum Toplantılarını gerçekleştirir.
    - Bu kısa toplantı (15 dk.) her iş gününde belirlenen saatte gerçekleştirilir ve tüm takım katılır (genelde sabahları).
    - Takımın ilerleyişini ve karşılaştıkları engelleri görmek için önemli bir fırsattır.
    - Teker teker tüm ekip üyeleri, şu soruların cevaplarını verir;
      - Dün ne yaptım?
      - Bugun ne yapacağım?
      - Karşılaştığım/Karşılaşacağım sorunlar ve engeller neler ?

Scrum

What is a Daily Scrum?

Zoom vs Google Meet: Picking the Best Platform for Your Daily Scrum Meeting - Droplr

Scrum



How to Get the Most Out of Scrum Meetings – GovWebworks

Scrum Kullanan Şirketler

[Is Your Company Agile? - Scrum Inc]

Kaynaklar

- Dr. Öğr. Gör. Fatih YÜCALAR –Celal Bayar Üniversitesi Yazılım Mühendisliğine Giriş Dersi Sunumları
- Dr. Öğr. Gör. Fatih YÜCALAR –Celal Bayar Üniversitesi Yazılım Yazılım Gereksinim Analizi Dersi Sunumları
- Dr. Öğr. Gör. Emin BORANDAĞ –Celal Bayar Üniversitesi Yazılım Yapımı Dersi Sunumları
- Java ve Java Teknolojileri, *Tevfik KIZILÖREN* – Kodlab Yayınları
- Waterfall Model Nedir? Özellikleri Nelerdir? » TechWorm (tech-worm.com)
- Yazılım Yaşam Döngü Modelleri. Yazılım Yaşam Döngüsü Nedir? | by Ömer Harun Çetin | Medium
- Yazılımcılar Dünyası (yazilimcilardunyasi.com)
- Waterfall Model Nedir? Özellikleri Nelerdir? » TechWorm (tech-worm.com)
- YMT 412-Yazılım Kalite Ve Güvencesi Test Stratejileri (firat.edu.tr)

Kaynaklar

- Highsmith, J., (2002), “Agile Software Development Ecosystems”, Addison Wesley.
- Schach, S. R., (2011), "Object-Oriented and Classical Software Engineering", 8 th Ed., McGraw-Hill.
- Agile Manifesto, http://agilemanifesto.org
- http://agilemanifesto.org/principles.html
- http://www.agilealliance.org/
- http://www.acm-software.com/
- Beck, K., (2002), “Extreme Programming”, Addison Wesley.
- Schwaber, K., (2004), "Agile Project Management with Scrum", Microsoft Press