

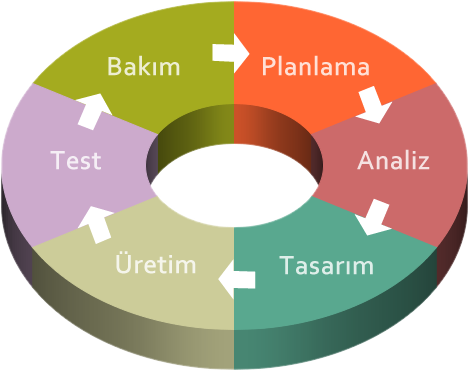
Yazılım yaşam döngüsü

Caner ÇAKMAK 200601070



<https://github.com/CanerCakmak/SDLC>

https://canercakmak.medium.com/yazılım-yaşam-döngüsü-nedir-206fa3db59f7

Yazılım Yaşam Döngüsü Nedir?

Yazılım yaşam döngüsü (SDLC), yazılım ürünleri oluşturmak, geliştirmek, değiştirmek için planlanan sürece verilen isimdir. SDLC’nin amacı yazılım kalitesini arttırmak, genel geliştirme sürecini daha etkili ve daha sağlam temellere oturtmaktır. Yazılım yaşam döngüsü birkaç adet temel aşamadan oluşmaktadır. Bu aşamalar sırasıyla : Planlama, Analiz, Tasarım, Üretim, Test, Teslim ve Bakım.

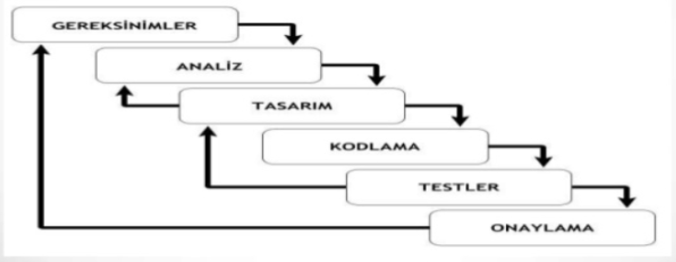
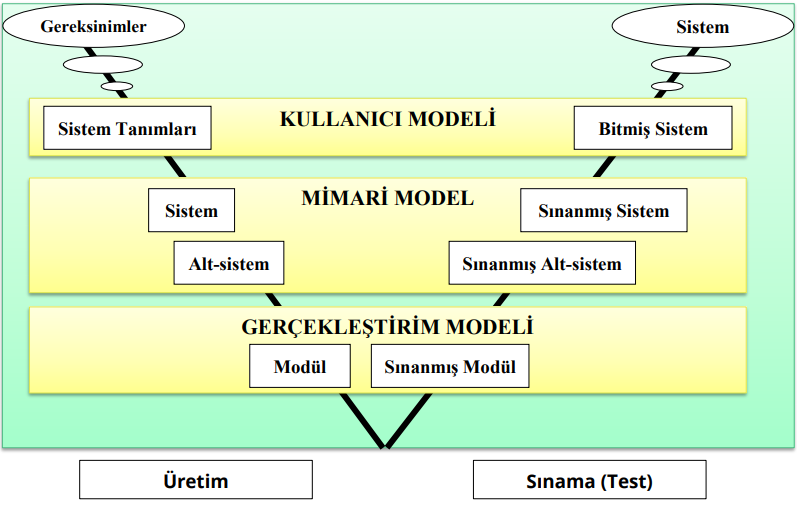
* **Planlama** yazılım yaşam döngüsünün ilk ve en önemli aşamasıdır. Bu aşamada en temel ihtiyaçlar belirlenir. Görevlerin dağılımı yapılır. Projeye en uygun şekilde yaklaşmak için fizibilite çalışması yapılarak projenin maliyeti tanımlanır.
* **Analiz** aşamasında projenin işlevlerini ve olmazsa olmazları tam olarak belirlenir, projenin ne kadar süreceği , ne tür risklerin olacağı belirlenir.
* **Tasarım** aşamasında ise belirlenen gereksinimlere, risklere, maliyete ve müşteri isteğine göre projeye en uygun mimari tasarım gerçekleştirilir. Bu aşamada iki tür tasarımdan bahsedebiliriz: Üst seviye ve mimari tasarım, detaylı tasarım.

1. **Üst Seviye ve Mimari Tasarım:** Önerilen sistemin yapısı (modüller, akış şeması)
2. **Detaylı Tasarım:** Yazılımı içeren bileşenler ve bunların ayrıntıları (veri yapıları, ekran tasarımları)

* **Üretim** aşamasında projenin kodlama bölümü başlar. Kodlama bölümünde ne kadar okunabilir ve anlaşılır bir kod kullanılırsa bakım aşaması da o kadar rahat bir şekilde gerçekleştirirsiniz.
* **Test** aşamasında ise projenin kodları test edilir. Düzeltilmesi gereken hatalar düzeltilir. Test aşaması kod aşamasıyla iç içe olursa zaman kayıpları daha az olur. Çünkü kod yazarken herkes hata yapabilir bunu ne kadar erken görürsek o kadar zamandan tasarruf ederiz.
* **Teslim ve Bakım** aşamasında tüm testler yapıldıktan sonra çıkan ürünün kullanıma uygun bir versiyonu müşteriye teslim edilir. Teslim yapıldıktan sonra bakım süreci başlar. Bakım sürecinde ise alınan geri dönüşlere göre ürün olduğu gibi bırakılabilir, üründe hata saptanırsa hatalar giderilebilir, müşterinin isteğine göre ürüne yeni özellikler eklenebilir.

SDLC kullanımının avantajları için; problemlerin aşamalar arasında geçişlerde fark edilebilmesi ve bir sonraki aşamada düzeltilebilmesi, daha kaliteli bir yazılım ve proje süreci geçirmek, neredeyse her türlü proje için kullanılabilmesi, dinamik ve yineleyici olması gibi şeyler söyleyebiliriz. Dezavantajlarından bahsetmek gerekirse; kullanımı herkes yapamaz projenin içinde en az bir adet uzmana ihtiyaç vardır, bazı basit projelerde uygulanması mantıklı değildir (hesap makinesi kodlama) çünkü 1-2 saat içerisinde yapılabilecek bir projenin sadece süresini uzatır bu yüzden daha büyük ve karmaşık projeler için daha etkilidir.

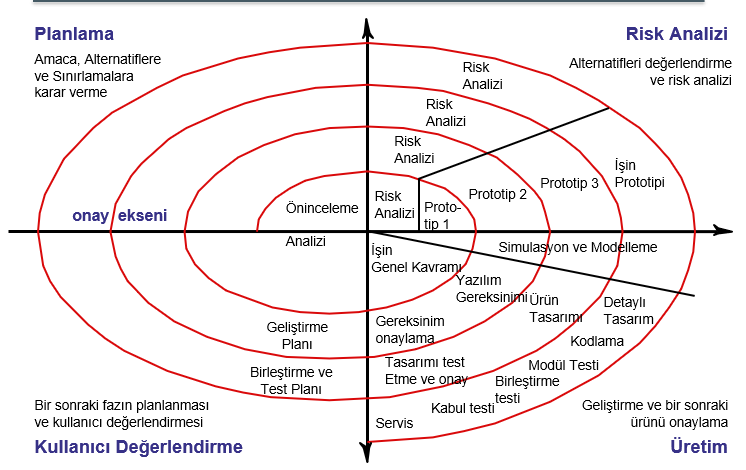
Yazılım yaşam döngüsünün en temel adımları bunlardır. Bunlara ek olarak yazılım yaşam döngüsünün modellerinden de bahsetmek isterim. Bu modellerin kullanımı projenin büyük, küçük, basit veya karmaşık olmasına göre değişebilir. Bu tür projeler geliştiren kurumların ya da toplulukların bu modelleri kullanarak hataların daha erken keşfedilmesi, müşteri memnuniyetinin sağlanması, gereksiz maliyetin azaltılması sağlanabilir . Bu modellerin isimlerini söylemem gerekirse: Gelişigüzel Model, Barok Modeli, Şelale (Waterfall) Yaşam Döngü Modeli, V (V-Shaped ) Modeli, Helezonik (Spiral) Model, Artımsal Geliştirme (Incremental Development) Modeli, Kodla ve Düzelt (Code and Fix) Modeli,Evrimsel Geliştirme(Evolutionary Development)Modeli, Çevik Modeller (XP,SCRUM).

1. **Gelişigüzel Model:** 1960’lı yıllarda uygulanan bu yöntem aslında bir model değildir. Kişiye özeldir. Takip etmesi ve bakım yapması zor olduğundan tercih edilmemektedir.
2. **Barok Modeli:** 1970’li yıllarda çıkan bu model yazılım yaşam döngüsünün (SDLC) temel adımlarını doğrusal bir şekilde kullanır. Bu modelin günümüzdekilerinde farkı belgeleme işlemi ayrı bir süreç olarak ele alınır.
3. **Şelale (Waterfall) Modeli:** Bu model geçmişte en popüler olan modeldir ve güncel modellerin de temelidir. Geleneksel bir yazılım geliştirme modeli olarak da bilinir. Bu modelde her aşama en az birer kez tekrarlanır. Şelale modeli günümüzde az zaman gerektiren işlerde uygun model olmasına rağmen kullanımı gittikçe azalmaktadır.
4. **V Model:** Bu model şelale modelinin bir üst versiyonu olarak görülebilir. Bu modelde Sol taraf üretim, sağ taraf ise test işlemidir. Bu model kendi içinde üç bölüme ayrılmaktadır: Kullanıcı Modeli, Mimari Model, Gerçekleştirim Modeli.

**Kullanıcı Modelinde** kullanıcının istekleri ve projeye ilişkin geliştirme planları belirlenmektedir.

**Mimari Modelde** projenin tasarımı ve sınama işlemleri yapılır.

**Gerçekleştirim Modelinde** kodlama ve kodların testi yapılır.

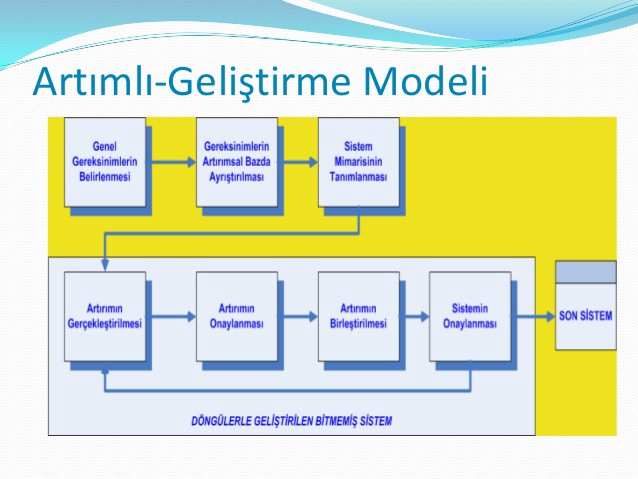
1. **Helezonik (Spiral) Model:** Bu model risk analizinin ön planda ve prototip yaklaşımı olan bir modeldir. Bu model 4 temel aşamada incelenir.

**Planlama:** Her aşamada üretilecek ara ürün için plan yapma. Önceki adımdaki ara ürün ile birleştirme.

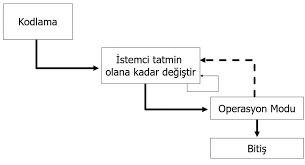
**Risk Analizi:** Risklerin detaylı araştırılması ve belirlenmesi.

**Üretim:** Ara ürünün üretilmesi.

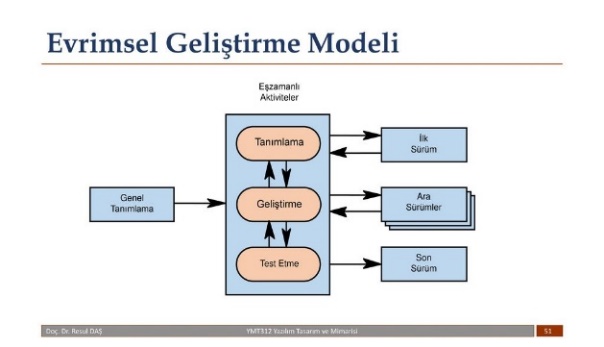
**Kullanıcı Değerlendirmesi:** Oluşturulan ara ürünün kullanıcıdan alınan dönütlere göre değerlendirilmesi.



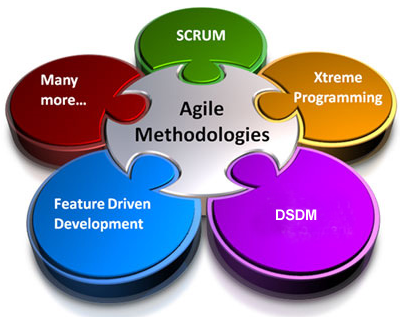
1. **Artımsal Geliştirme Modeli:** Bu modelde proje parçalara ayrılır. Ve bu parçalar kullanıcının gereksinim önceliğine göre erken teslimlere dahil edilir. Bir parçanın geliştirilmesine başlandığında o parçanın gereksinimleri sabitlenir eğer bir değişiklik olursa sonraki teslimde ele alınır. Bu model kullanılarak sistemin başarı seviyesi yükselir fakat zaman açısından biraz maliyetlidir.



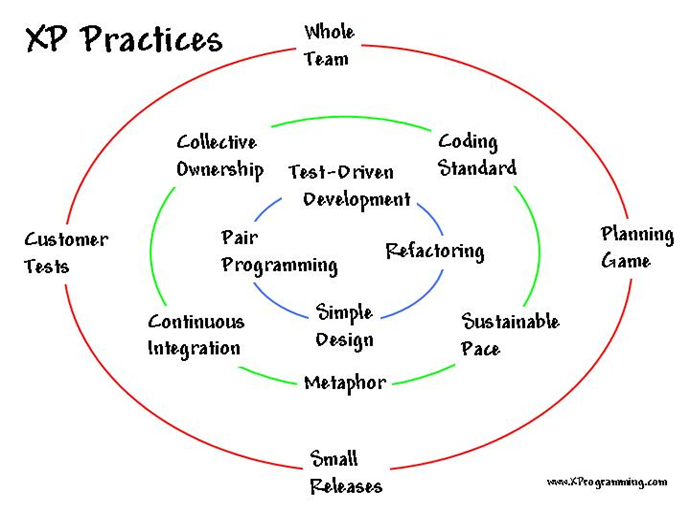
1. **Kodla ve Düzelt Modeli:** Bu model hakkında pek söylenebilir bir şey yok. En basit yaşam döngü modelidir fakat en pahalısıdır. Bakım aşaması da zordur. Bu model kolay olduğundan dolayı küçük ve tecrübesiz firmalar tarafından kullanılır.



1. **Evrimsel Geliştirme Modeli:** İlk tam ölçekli geliştirme modelidir. Bütün projenin başarısı ilk evrimin başarısına bağımlıdır. Tüm aşamalarda üretilen ürünler tam etkinlik içermektedir. Dünya üzerinde uluslararası adı olan (banka uygulamaları) kuruluşlar için etkilidir.

****

1. **Çevik Modeller:** Bu modeller projelerdeki başarı oranlarını ve iletişimi üst düzeyde tutmak için oluşturulmuş modellerdir. Ayrıca bu metotlar piyasaya çabuk ürün çıkarma, değişken piyasa ve isteklere daha hızlı yanıt verme amacı da taşırlar. Çevik modeller kendi içinde pratik farklılığından dolayı çeşitli metotlara ayrılmaktadır ama ben bu yazımda sadece XP ve SCRUM metotlarına değineceğim.



* **Extreme Programming (XP)** : 1996 yılında Kent Beck ve arkadaşları tarafından geliştirilmiştir. XP grup içi iletişimine ve geri dönütlere önem veren bir yapısı vardır. XP yazılım geliştirmede kolaylık ve esneklik açısından 12 adet pratiği bize sunar. Ayrıca bunların yanında XP dört temel maddeden oluşur:

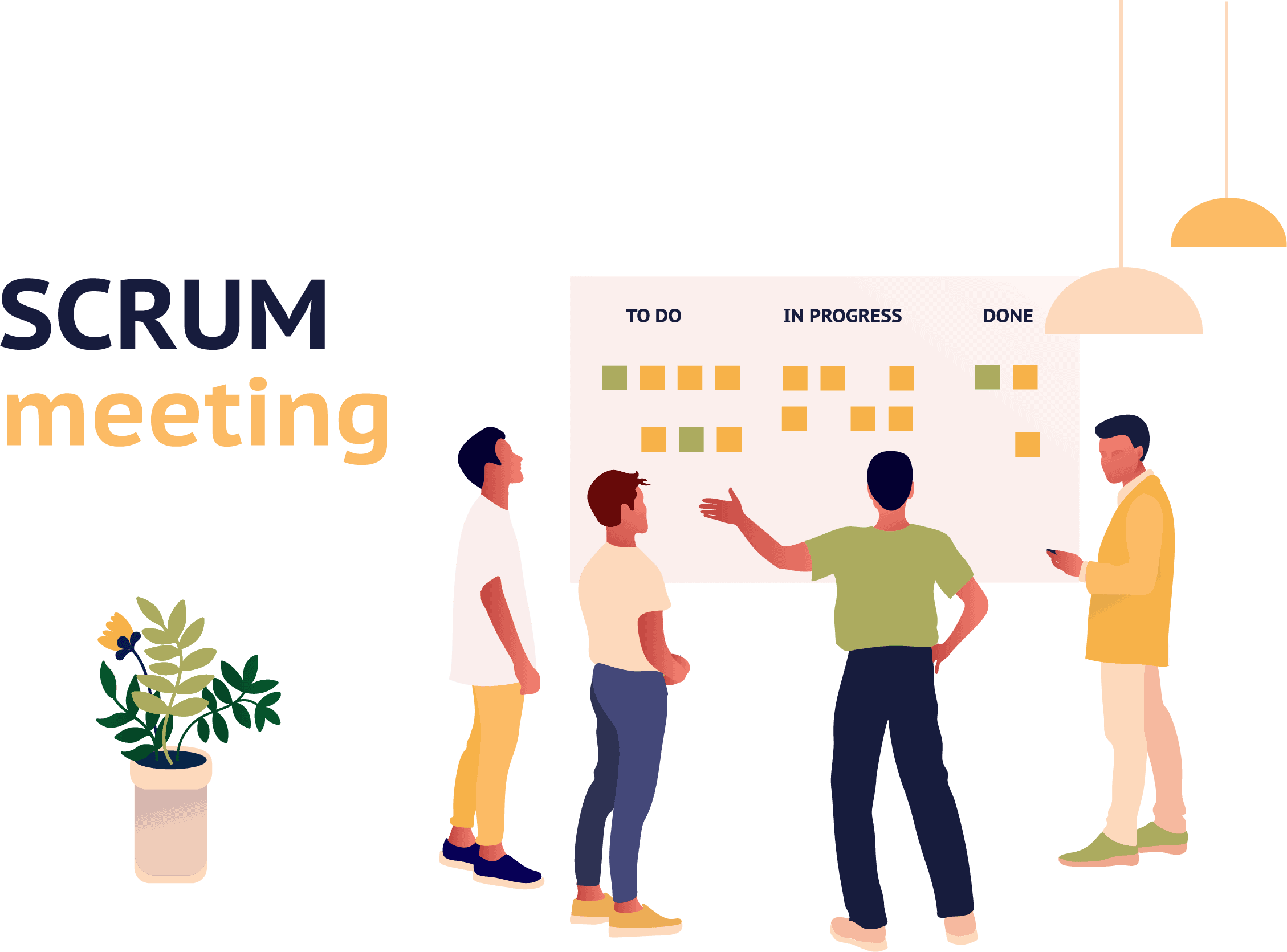
**Cesaret:** Bu madde yapılan işte cesur olunmanın, projelerden korkmadan pes etmeden üstüne gitmenin hatta gerektiğinde kodu silip en baştan başlamanın gerektiğini söyler.

**Basitlik:** Bir kodda basitliği sağlamak cidden zordur ama doğru olandır. Yazılan kodun basit, anlaşılır ve okunur olması önemlidir. Proje için gereksinimleri karşılayan en basit çözüm kullanılmalıdır.

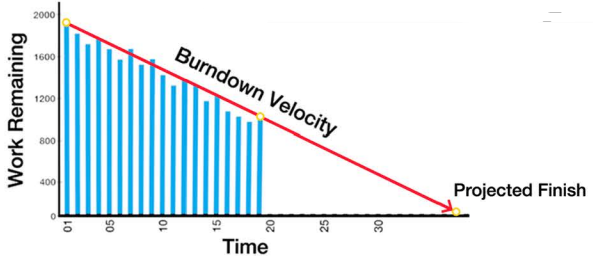
**İletişim:** Genel olarak projelerde önemli problemlerin en başında iletişimin geldiği görülür. Bir projede hatta hayatın herhangi bir yerinde başarılı olabilmek için iyi bir iletişim şarttır. XP iletişim eksikliğini ortadan kaldırmayı amaçlar. Eğer ekip arasında iyi bir iletişim olursa sorunlar daha erken fark edilir bu da zamandan kaybı önler.

**Geri Dönüt:** Geri dönütler projeler için etkili bir hedef göstericidir. Geri dönütler sayesinde ortaya çıkan, çıkabilecek hatalar kaldırılır. Müşteri ile yazılım ekibi arasında olan görüşmeler ise hem müşteri memnuniyeti için iyi hem de sonrasında oluşabilecek anlaşmazlıkların önüne geçebilmek için bir fırsattır.

* **SCRUM:** SCRUM Jeff Sutjerland ve Ken Schawaber tarafından 1990’da geliştirilen bir modeldir.SCRUM benim görüşüme göre en etkili modeldir. SCRUM büyük projeleri “sprint” adı verilen parçalara ayırır. Bölünmüş projedeki her sprint tek tek geliştirilir. SCRUM’da olan ve benim en beğendim özellik iletişime önem vermesi ve hatta her gün 15 ila 30 dakika arasında olacak şekilde “SCRUM MEETINGS” denilen toplantılar yapılmasıdır. Bu toplantılarda genelde “Dün ne yaptın?”, ”Bugün ne yapacaksın?” gibi kısa sorular sorulur. SCRUM’da üç temel kavram vardır : Roller, Toplantılar, Bileşenler/Araçlar.

**Roller:** Ürün Sahibi, Scrum yöneticisi ve Scrum takımından oluşur. Ürün sahibi projenin beynidir. Scrum yöneticisi Scrum etiğine uygun bir şekilde Scrum takımını kontrol eder. Scrum takımı ise 5 ila 9 kişiden oluşan aralarında sürekli iletişimin bulunduğu bir takımdır.

**Toplantılar:** Toplantılar Scrum için en önemli parçalardan biridir. Her gün bu toplantılar yapılır ve kısa bir şekilde herkesin ne yaptığı, ne yapacağı konuşulur. Ayrıca her yeni bir sprint başlangıcında takım bir planlama toplantısı yapar. Toplantının ilk kısmı ürün gereksinimlerinin belirlenmesi gibi temel konular olur.

**Bileşenler/Araçlar:** Ürün gereksinim dokümanı oluşturulur ve bu doküman proje sürecinde yapılması gerekenlerin yazıldığı bir liste içerir. Sprint dokümanı oluşturulur ve bu dokümanın amacı oluşturulan ürün gereksinim dokümanına uygun şekilde sprint ayarlamaktır. Sprint kalan zaman grafiği oluşturulur. Bu grafik planlanan zamana göre işin ne kadarının yapıldığını gösterir.

Sprint Kalan Zaman Grafiği Örnek

Modellerin Avantajları Dezavantajları Nelerdir?

**Gelişigüzel** ve **Barok** modeli artık neredeyse bitmek üzeredir bunun sebebi ise yinelemeli olmadıklarındandır.

**Şelale modelinin** eski olmasına rağmen uygulamasının etkili ve kullanışlı olması, kullanımının kolay olması ayrıca detaylı olması küçük projelerde kullanımı çok uygunken günümüzdeki büyük projelerde pek etkin olmaması, geri dönüşe izin vermeyen yapısı olması, grup içinde veya müşteri ile iletişimin az olması ve bu yüzden müşterinin değişen isteklerine ayak uyduramayıp proje bittikten sonra anlaşmazlığa gidilip projenin iptaline kadar götürülebilmesi gibi sebepler yüzünden artık etkisini yitirmiştir.

**V modeli** kullanımı kolay, proje takibi kolay bir model olmasına rağmen kendi içindeki bölümleri yinelememesi bu yüzden değişen gereksinimlere ayak uyduramaması ve işin risk kısmını ele alan bir bölüm olmamasından ötürü günümüzde pek etkili değildir.

**Spiral modelde** ise kullanıcılar ara ürünleri de gördüklerinden ötürü projeye dahil olurlar ve spiral model için risk çok önemli bir yerdedir ayrıca ara ürünler sayesinde hataların erken giderilmesi avantajlarındayken, dezavantajları için karmaşık yapılı olması çok geniş bir zaman ve doküman tutumu istemesi gibi nedenler sunabiliriz.

**Artımsal modelde** ise her yapılan teslimde müşteriden alınan geri dönüt ile müşteri memnuniyeti ve proje içinde herhangi bir sorun hemen tespit edilebilir. Her teslim yeni bir test demek olacağından dolayı projenin başarılı olma olasılığı daha da artar fakat bu modelin en büyük dezavantajı zaman maliyetinin yüksek olması.

**Kodla ve Düzelt** modelinin avantajından çok dezavantajı vardır. Bakımı zahmetli, yeni teknolojiye hızlı ayak uyduramaz, maliyeti yüksek, büyük projelerde kullanılamaz, zaman açısından bitiş süresi belli değildir , doküman tutma yok ve tek kişi ile sınırlı bir modeldir. Avantajları ise planlamaya gerek duyulmaz, herkes bu modeli kendi başına kullanabilir, küçük ve tek kişilik projeler için gayet uygundur.

**Evrimsel Geliştirme** modelinde düzenli teslim edilen ürün olmadığından müşteri memnuniyeti düşebilir, bakım aşaması zahmetlidir. Avantajları ise sürekli olarak ortada bir program olduğundan geri dönüşler etkilidir bu sayede projelerdeki hatalar kolay fark edilebilir.

**Çevik modellerde** her gün toplantılar, müşteri ile iletişim gibi yüksek iletişim olduğundan hata payı düşük ve takım oyunu gibidir. Bu model değişime açıktır , çalışma saatleri bakımından esnektir. Günümüzde kullanımı da en yaygın olanıdır.

Hangi Projede Hangi Modeli Kullanmalıyız?

Küçük, tek kişilik, zaman sıkıntısı olmayan programlara kodla ve düzelt modeli uygundur.

Küçük ve detayları güzelce belirlenmiş projelere şelale modeli uygundur.

Orta ve küçük büyüklükte, ekip işi gerektiren, çok vakit almayan projelere çevik modeller uygundur.

Büyük ve zaman açısından maliyetli projelere Spiral veya Artımsal model uygundur.

Çok büyük alanda, çok fazla insana hitap edecek proje için Evrimsel Geliştirme modeli uygundur.

SCRUM Günümüzde Neden Popüler ?

Scrum günümüzde en çok kullanılan modeldir. Bunun başlıca nedeni ortadaki iletişim sorununu günlük ve her sprint için yapılan toplantılarla kaldırmaktadır. Bunun yanında scrum:

* Yeni teknolojilere uyum sağlayabilmesi
* Kendi içinde yinelemeli olması
* Değişen gereksinimlere çabuk adapte olması
* Büyük projeyi küçük parçalara ayırarak daha detaylı çalışması
* Ekonomik ve zaman açısından tasarruflu olması
* İletişimin yüksek olması dolayısıyla hataların erken fark edilip düzeltilmesi gibi sebeplerden dolayı Scrum günümüzde en popüler ve etkili modellerden biridir.

Yararlanılan kaynaklar:

Doç. Dr. Deniz KILINÇ,İzmir Bakırçay Üniversitesi Yazılım Mühendisliğine Giriş Dersi 3. Hafta Sunumu

<https://medium.com/@denizkilinc/yazılım-yaşam-döngüsü-temel-aşamaları-software-development-life-cycle-core-processes-197a4b503696>

<https://zeynepaygun.wordpress.com/2017/05/29/what-is-sdlc-sdlc-nedir/>

<https://www.tutorialspoint.com/sdlc/sdlc_overview.htm>

<https://fikirjeneratoru.com/yazilim-proje-yonetimi-yontemleri/>

<https://www.orhanyener.net/yazilim-testi-2/sdlc-software-development-lifecycle/>

https://medium.com/@secilcor/scrum-nedi̇r-6a4326951dd8