Yazılım Yaşam Döngüsü.

Bir yazılım projesine başladığınızda bu sadece bilgisayar başında oturup kodlama yapmaktan ibaret değildir. Her şeyde olduğu gibi yazılım projelerinin de belli bir aşamaları vardır. Yazılım projelerinin verimli geçmesi için bazı özelliklere ihtiyacı vardır. Bunlar modellemeler ve planlı bir çalışma sürecinden geçer. Planlı çalışmak için beş temel aşama vardır. Bu aşamalar planlama, analiz, tasarım, gerçekleştirme ve bakım olarak beş ana başlıkta toplanır. Peki bu beş aşama tam olarak nedir şimdi onu inceleyelim.

Yazılım da her şeyde olduğu gibi bir döngünün içerisinde. Bu döngüde aşamaları atlayabilir ve döngüyü tekrardan başlatabilirsiniz. Peki bu döngüler nedir?

Planlama:

Yazılım yaşam döngüsünün ilk aşamasıdır. Projede istenilen unsurlar. Projenin ana gereksinimleri, projede ne yapılacağı ve kimin ne yapacağı, donanım ve yazılım planlaması ve projeyi başarılı kılacak unsurlar belirlenir. Bu kısımda kullanıcı isteklerini iyi bir şekilde anlamalı ve uygun bir biçimde model seçimi yapılmalıdır. Bu aşamada UML diyagramların(Class, object, state, sequence, activity …) çizilmeye başlanıldığı aşamadır. Proje bu aşamada planlanır.

Analiz:

Yazılım yaşam döngüsünün ikinci aşaması olan kısımdır. Projenin ne kadar vakit alacağından ne gibi risklerle karşılaşabilineceğini belirlenir. Projenin net bir şekilde sahip olacağı özellikler ve ihtiyaçlar belirlenir. Mevcutta bir sistem var ise bu sistemin incelenmesi gerçekleştirilir. Belirlenen ihtiyaç ve özellikler belirli bir biçimde belgelenmelidir. Bu belgeleme işlemi müşteri, yazılım mühendisi, sistem analisti, ürün yöneticisi, iş analisti ve benzerlerinin oluşturduğu grup ve ya gruplar tarafından yapılabilir. Müşterinin talepleri net bir şekilde anlaşılmaması gibi durumlarda yazılım mühendisi ve müşterinin daha fazla temasta kalması ve birlikte daha fazla iş yapmaları gerekebilir. Yazılım mühendislerinin işi net ve doğru olarak algıladıklarından emin olunmalıdır.

Tasarım:

Yazılım yaşam döngüsünün üçüncü aşaması olan tasarım aşaması, bu aşamada kodlama kısmına girilmez, kurulacak sistemin gereksinimlerinin bilinmesiyle birlikte sistem tasarımına başlanır. Bu kısımda analizde sorulan isteklerin nasıl hayata geçirebileceği konusunda cevap aranır. Bundan sonra projenin nasıl bir süreç izleyeceği konusunda tasarlama yapılır. Yazılım ürün tasarımı bir binanın temeli gibi düşünülebilir. Temel sağlamsa binayı dayanıklı kılmak daha kolay olur. Yazılım ürününde de tasarım iyi ise üzerine konulan işlevler de daha etkili bir biçimde karşımıza çıkar. İki tür tasarımdan bahsedebiliriz. Mantıksal tasarım ve fiziksel tasarım. Mantıksal Tasarım: Mevcut olan sistemin yapısından ziyade, önerilen sistemin yapısı anlatılır. Olası değişiklikler önerilir. Fiziksel tasarım: Yazılımı içeren bileşenlerin ve bunların detaylarına iner ve bunları içerir. Tasarımda en çok kullanılan yöntemlerden bir tanesi olan soyutlama(Abstraction) yöntemidir. Soyutlanma ile projede ki bazı olaylar görmezden gelinir. Görmezden gelinmesinin sebebi oluşan sıkıntıya odaklanmak ve karmaşadan kurtulmaktır. Bu yöntem sayesinde oluşan problemlerin daha kolay bir biçime sokarak problemlerdeki can alıcı kısımlara dikkat edilmesine olanak sağlar.

Gerçekleştirme:

Yazılım yaşam döngüsünün dördüncüsü olan gerçekleştirme(kodlama) aşaması, tasarım bittikten sonra kodlama aşamasına geçilir. Kodlama kısmı planlama ve tasarımda düşünülen programlama dili, geliştirme ortamları ve de teknolojik ürün yardımıyla yapılır. Kodlamayı yaparken temiz(clean) yazmak avantaj sağlayacaktır. Hem bakım aşamasında hem de test aşamasında önemli olan clean kod iyi yazılmış bir koddur.

Kodlama aşamasında bir diğer önemli aşama ise testtir. Eğer erken test aşamasına geçilmesi durumunda hata yapma olasılığımız düşecektir. Bu sayede zaman, para, prestij anlamında negatif durumda olmanızı engeller. Burada testi yaparken belli başlı test teknikleri bulunmaktadır. Bunlardan bazıları: Sınır değer analizi testi, denklik sınıflarına ayırma testleri, durum geçiş testleri, karar tablosu testleri, fonksiyonel kara kutu testleri. Bu testler sayesinde zaman, para, prestij kayıpları minimum seviyede olur.

Bakım:

Yazılım yaşam döngüsünün sonuncusu yani beşincisi olan bakım aşamasında, ürününün testleri bittikten sonra ürün artık kullanıcıya ulaştırılır. Bundan sonra kullanıcıdan dönen hatalara ve istenmeyen özelliklere karşı bakım yapılır. Bu bakım geliştiriciler ve iş analistleri tarafından ayarlanır.

Yazılım yaşam döngüsü içerisinde çok sayıda model mevcuttur. Bunları yaptığımız iş ve şartlara göre bize en uygun modeli seçeriz. Şimdi bu modellerin birkaçını tanıyalım.

Barok Modeli

1970 yıllarında çıkan bu modelde süreç doğrusal ilerler. Modelin en dikkat çeken yanı ise Günümüzde belgeleme neredeyse her süreçte ele alınırken barok modeli belgeleme işini farklı bir bölüm olarak ele almasıdır. Modelin zayıf kalan kısımlarından biri de geri dönüşlerde ki bilgi eksikliğidir.

Şelale ya da Waterfall Modeli

Barokla yakın tarihte çıkan şelale modeli barok modelin aksine belgelemeye her aşamada yer veriyor. Genellikle uzun sürmeyecek projelerde kullanılan bu model her süreci en az bir kere görür. Bir aşama bitmeden diğerine geçilmez. Adım adım gitmesinden dolayı hatalar son aşamalarda gözlemlenir ve buda bize zaman, maliyet olarak geri yansır. Bu sebepten uzun sürecek projelerde şelale modeli etkisiz kalacaktır.

Avantajları: Kullanım ve ekibin anlaması açısından basit bir modeldir. Yönetim olarak bir zorluğu yoktur. Küçük projelerde çok net projenin anlaşılmasını sağlar.

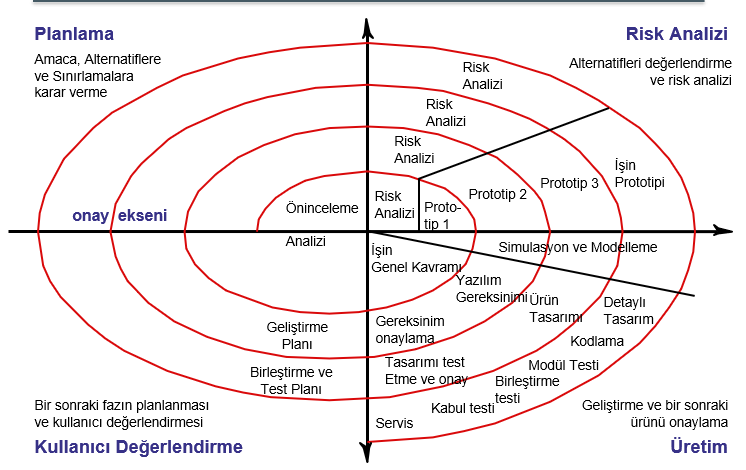
Dezavantajları: Karmaşık projeler için uygun değildir. Devam eden uzun süren projelerde para, prestij, moral, zaman kaybına yol açar. Kullanıcın isteklerinin doğru olup olmadığı sahada anlaşılır bu da projenin heba olmasına yol açabilir. Heba olan proje şirket için ciddi bir prestij, zaman, para kaybına yol açar.

Spiral Model

Üretim, risk analizi, planlama ve kullanıcı yorumlaması olarak 4 ana aşamadan oluşur ve tekrarlanır. Bu işlemlerden her biri faz olarak isimlendirilir. Fazlar sırasında kullanıcıdan feedback için kullanıcıya örnekler sunulur. Sunulan bu örnekler sayesinde geliştirilmiş bir planlama ile yeni faza girilir. Modelin pozitif yanlarından biri kullanıcı sınıfı ile birlikte oluşturulmasıdır. Bu sayede kullanıcının istekleri bilinir ve kullanıcıya istediği uygun bir proje çıkartılır. Proje sona ulaşmadan hatalar fark edilir. Örnek olarak şelale modelinde sonlara yaklaşırken fark ettiğimiz yanlışları, spiral modelde erken fark eder bu durumda bizlere zaman, para maliyetlerinde avantaj sağlar. Adı gibi spiral bir sonsuzluğa gidebilir. Belgeleme oldukça fazla olacağından karmaşıklık artacaktır. Bu yüzden düşük projelerde şelale modeli tarzı modeller daha az maliyetli olacaktır.

Avantajları: Kullanıcı sistemi erken tanır, riskli kısımlara öncelik verilir, birden fazla model içerisinde bulunur, riski azaltan bir davranış biçimi vardır, hatalar konusunda erken davranır, yazılım ve donanım için genel bir çerçeve sağlar.

Dezavantajları: Hata ve hatadan kaynaklı oluşan sıkıntısı küçük olan ya da hata payı az olan projeler için fazla maliyete yol açabilir, karmaşık bir yapıya sahiptir bu da bir önceki maddede belirttiğimiz maliyete sebep olacaktır, sonsuz bir döngüye girebilir, içten dışa doğru bir gelişim söz konusu olduğunu kabul eder.



Artırımlı Geliştirme Süreç Modeli

Tek versiyon yerine birden fazla versiyonla geliştirilen projelerde, her aşamada yeni işlevler eklenerek geliştirilir. Ürün kullanılırken geliştirme aşaması devam eder ve versiyon versiyon ürün geliştirilir. Bitmesi uzun süren ve eksiklerine rağmen çalışabilecek projeler için idealdir. Ürün geliştirme aşamasındayken bir kullanıcı tarafından kullanıldığından riski düşüren ve projeye olan isteği ürün tamamlanmadan görülebilen bir geliştirme modelidir. Böylece zaman, para gibi unsurlardan minimum zarar edilebilir.

Avantaları: Projeye ihtiyaç duyduğumuz özellikleri kullanıcı belirler bu da bizim kullanıcının istediği bir projeyi önüne sunmamızı sağlar. Projenin hepsinin başarısız olmasının önüne geçer. En önemli özellikler belirlenir belirlenen bu özellikler testlerden geçer, bu da bizim en ihtiyaç duyduğumuz özelliklerin hatalarını görmemizi sağlar. Böl ve yönet yaklaşımıdır. İhtiyaç önemine göre bir çekirdek sistem geliştirilir.

Dezavantajları: Versiyon tanımlaması için tüm sistemin tanımlanması gerekmektedir. Alanında uzman personel gerekmektedir bu da personel açığına sebep olabilir. Kendi içinde tekrarlamalara izin vermez.

Evrimsel Geliştirme Süreç Modeli

Kullanıcının isteklerinin kavranamadığı zaman tercih edilen bir modeldir. Kullanıcı ile birlikte bir taslak oluşturulur. Kavranan öğelerden başlanır ve kullanıcıya göre yeni entegreler uygulanır. Bu model sayesinde kullanıcının isteklerinin ne olduğunu anlamamız konusunda yardımcı olur. Sürekli değişime uğrama ihtimalinden kaynaklı yapılan işteki yapı hasar alabilir bu da proje bakımına güçlük katar.

Avantajları: Kullanıcın istekleri konusunda net bir bilgi sağlar. Erken aşamalarda hataları anlayarak yukarda bahsettiğim şelale metodunda oluşacak projenin heba olması olayının önüne geçer. Hataları minimuma indirir.

Dezavantajları: Süreç görünürlüğü azdır. Sürekli değişiklikten kaynaklı sistem yapısı zarar görür. Bakımı zordur. Yazılım için ihtiyaç duyulan donanım, yazılım gibi faktörlerin değişmesiyle ihtiyaç değişikliğine gidilebilir.

**Scrum Nedir?**

Scrum, yinelemeli ve artımlı süreçlere dayalı Yazılım geliştirmede kullanılan çevik bir geliştirme metodolojisidir. Scrum, projenin gelişimi boyunca müşteriye değer sağlamak için tasarlanmış, uyarlanabilir, hızlı, esnek ve etkili çevik bir çerçevedir. Scrum'ın birincil amacı, iletişimde şeffaflık, kolektif sorumluluk ve sürekli ilerleme ortamı aracılığıyla müşterinin ihtiyacını karşılamaktır. Geliştirme, neyin inşa edilmesi gerektiğine dair genel bir fikirle başlar ve ürün sahibinin elde etmek istediği önceliğe (ürün biriktirme listesi) göre sıralanmış bir özellik listesi hazırlar. Kısaca Scrum, çalışanların karmaşık sorunları doğru biçimde ele alırken, bir yandan da yüksek değere sahip üretim yapmaları hedefleyen bir çerçevedir. Scrum, bu verimli üretimin yaratıcı ve sorun çözümünün adaptasyona açık olmasını da teşvik eder.



Scrum'ın tarihi, Hirotaka Takeuchi ve Ikujiro Nonaka'nın “ yeni ürün geliştirme” başlıklı Harvard Business Review (HBR) makalesinde 1986 yılına kadar izlenebilir . Bu makale Honda, Canon ve Fuji-Xerox gibi şirketlerin, ürün geliştirmeye yönelik ölçeklenebilir ve ekip tabanlı bir yaklaşım kullanarak dünya çapında nasıl yeni ürünler ürettiğini açıklamaktadır. Bu yaklaşım, kendi kendini organize eden ekipleri güçlendirmenin önemini vurgular. Şimdi Scrum dediğimiz şeyi doğuran birçok kavramın geliştirilmesinde etkili oldu. Scrum, bir faulden sonra veya top oyundan çıktığında oyunun nasıl yeniden başlatıldığını ifade eden Rugby'den alınan bir terimdir.1993 yılında [Jeff Sutherland](https://en.wikipedia.org/wiki/Jeff_Sutherland" \t "_blank)ve Easel Corporation'daki ekibi, 1986 makalesindeki kavramları nesne yönelimli geliştirme, ampirik süreç kontrolü, yinelemeli geliştirme ve artımlı, yazılım süreçleri kavramlarıyla birleştirerek yazılım geliştirme süreçlerinde kullanılmak üzere Scrum sürecini yarattı. ve üretkenlik artışının yanı sıra karmaşık ve dinamik sistemlerin geliştirilmesi. Scrum, tam olarak Çevik Yönetimin bir evrimidir. Scrum metodolojisi, yazılım geliştirme sürecinde dahil edilmesi gereken çok tanımlanmış uygulamalara ve rollere dayanmaktadır. Ürünün tüm ekip üyeleri tarafından kabul edilen bir bağlamda 12 çevik  uygulanmasını ödüllendiren esnek bir metodolojidir . Scrum, geri bildirim ve yansıtma terimi olan, genellikle 2 ila 4 hafta arasında değişen, Sprint adı verilen kısa ve periyodik geçici bloklarda yürütülür. Her Sprint kendi içinde bir varlıktır, yani tam bir sonuç, talep edildiğinde müşteriye mümkün olan en az çabayla teslim edilmesi gereken nihai ürünün bir varyasyonunu sağlar. Sürecin başlangıç ​​noktası olarak proje planını oluşturan bir hedefler/gereksinimler listesi vardır. Değer ve maliyet dengesini göz önünde bulundurarak bu hedefleri önceliklendiren projenin müşterisidir, yinelemeler ve sonuç teslimatları bu şekilde belirlenir. Bir yandan pazar, bir şirketin ürün geliştirmede çok çevik ve esnek olması gereken kaliteyi, daha düşük maliyetlerle hızlı teslimatı, sonucun kalitesini düşürmeden müşterilerin talebini karşılayabilecek kısa geliştirme döngüleri elde etmeyi talep ediyor. . Uygulanması çok kolay bir metodolojidir ve aldığı hızlı sonuçlar için çok popülerdir. Scrum metodolojisi esas olarak yazılım geliştirme için kullanılır, ancak [diğer sektörler](https://www.benlinders.com/2016/agile-scrum-beyond-sw-development/) de bu metodolojiyi satış, pazarlama ekipleri gibi organizasyonel modellerinde uygulayarak faydalarından yararlanmaktadır.

Referanslar:

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

<https://medium.com/@tunaytoksoz/yazilim-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-sdlc-ve-modelleri%CC%87-c3fe40f6e4e8#:~:text=Yaz%C4%B1l%C4%B1m%20ya%C5%9Fam%20d%C3%B6ng%C3%BCs%C3%BC%2C%20(Software%20Development,a%C5%9Famalar%C4%B1n%20t%C3%BCm%C3%BCne%20verilen%20isimdir%20diyebiliriz>.

<https://medium.com/@denizkilinc/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-temel-a%C5%9Famalar%C4%B1-software-development-life-cycle-core-processes-197a4b503696>

<https://iskulubu.com/yazilim/yazilim-gelistirme-yasam-dongusu/>

<https://github-wiki-see.page/m/muratcem35/makalelerim/wiki/YAZILIM-YA%C5%9EAM-D%C3%96NG%C3%9CS%C3%9C-VE-MODELLER%C4%B0>

<https://tr.wikipedia.org/wiki/Scrum>

<https://www.argenova.com.tr/scrum-nedir>

<https://miesofficial.com/blog/scrum-framework-nedir/>