YAZILIM YAŞAM DÖNGÜ MODELLERİ

Her ürünün olduğu gibi yazılım ürünlerinin de üretim, yaşam, kullanım süreçleri vardır. Yazılım ürünlerinin geçirdiği tüm bu aşamalara yazılım geliştirme yaşam döngüsü (“software development life cycle”, “SDLC”) denir. Eğer kullanışlı, verimli, sorun çözme odaklı bir ürün oluşturmak istiyorsanız bunu planlı, aşamalı ve zamana dayalı olarak yapmalısınız. Böylece ortaya çok daha bilinçli bir şekilde üretilen ürünler çıkıyor. Bu yazılım geliştirme yaşam döngüsünü planlı bir şekilde yapmanın farklı farklı yöntemleri, teknikleri bulunuyor ayrıca günümüz dünyasında bu modeller yerine sürekli güncel modeller çıkabiliyor. Tüm bu yazılım geliştirme modellerine "Yazılım Yaşam Döngü Modelleri" denir.



Bu üretim planmaları bazı temel aşamalardan oluşur. Bu aşamalar: Planlama, Analiz, Tasarım, Gerçekleştirim (Kodlama ve Test).

Planlama: Bu döngünün ilk aşamasıdır, ihtiyaçlar belirlenir, hedefler doğrultusunda bir yol çizilir, proje için fizibilite çalışmaları yapılır, proje planlanır.

Analiz: Bu aşamada ise riskler analiz edilir, sistem analizi, ihtiyaç analizi, projenin ne kadar süreceğinin analizi gibi analizler gerçekleştirilir.

Tasarım: Bu aşamada oluşturulacak ürün tasarlanır, taslak oluşturulur, ürün müşterinin isteğine ve gereksinimlerine göre şekillendirilir. Tasarım aşamasından 2 şekilde bahsedebiliriz, bunlar "Yüksek Düzeyde (Mimari) Tasarım" ve " Detaylı Tasarım" şeklindedir. Mimari tasarım, yazılım modüllerinin genel yapılarıyla ve organizasyonla etkileşimini incelerken, detaylı tasarım da mimari tasarım dökümanları revize edilir.

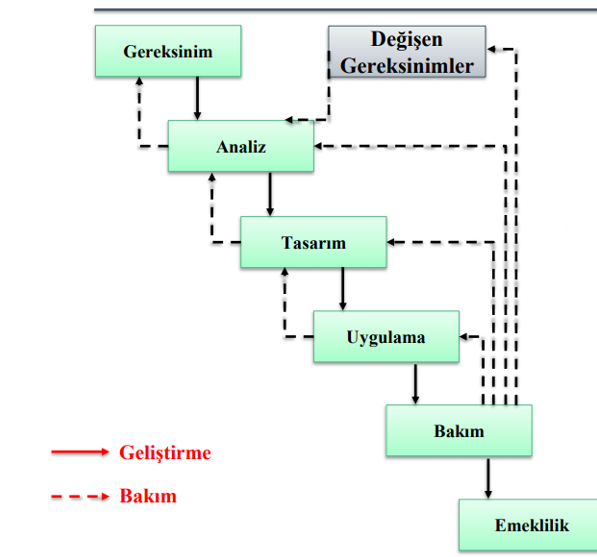
Gerçekleştirim (Kodlama ve Test): Bu aşama yazılım ürününün kodlama ve test kısmıdır. Tasarım aşamasında ilerleme kaydedildikten sonra kodlama aşaması başlar. Kodlama boyunca ve sonrasında ki bir diğer önemli aşama test aşamasıdır. Test aşaması maliyeti, zamanı, hatayı minimuma indirecektir. Böylece ürünü birebir takip edebilecek ve hata oluştuğu anda güncel olarak ona müdahale edebilme şansımız olucak.

Bakım ve Teslim: Bu aşamadaysa, diğer aşamaları başarılı bir şekilde bitirdiğimiz zaman ürünün teslim edilebileceği bir versiyonu oluşturulur ve teslim edilir. Teslim edilen ürünün bakımını yapmaksa yine bize düşer. O zaman ki güncel teknolojiye, ihtiyaca, hatalara göre yazılımın bakımı yapılır, güncellenerek güncel versiyonu tekrar teslim edilir.

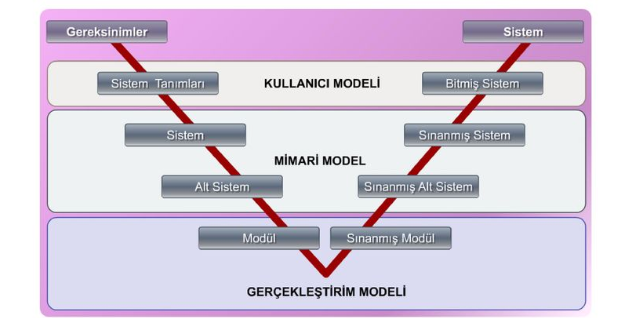
Peki bahsettiğimiz bu yazılım yaşam döngü modelleri nelerdir ? Bu modelleri gelişigüzel model, barok modeli, çağlayan yaşam-döngü modeli, v süreç modeli, helezonik (spiral) modeli, artımsal geliştirme süreç modeli, kodla ve düzelt yaşam-döngü modeli, çevik modeller olarak sıralayabiliriz.

Gelişigüzel Model: Bu yöntemi aslında bir model olarak adlandırmak doğru değildir. Çünkü ortada belirlenmiş bir model veyahut yöntem yoktur. Üretilen yazılımın izlenebilirliği, bakım yapılabilirliği çok zordur. Bu yöntem 1960’lı yıllarda uygulanmakla beraber genellikle basit programlama için kullanılan yöntemdir.

Barok Modeli: Bu yöntem 1970’li yıllarda ortaya çıkmıştır. Barok modelinde, yazılım yaşam-döngüsü temel adımları doğrusal şekilde ele alınır ve geliştirilir. Bu modelde geri dönüşlerin nasıl yapılacağı belli değildir. Ayrıca bu modelde günümüz modellerinden farklı olarak dokümantasyon ayrı bir süreç olarak ele alınır ve en son yapılması öngörülür. Günümüz güncelliğinden uzak kalmıştır ve artık tercih edilmemektedir.

Çağlayan (Şelale) Yaşam-Döngü Modeli: Bu model geçmişte en popüler yazılım geliştirme modeli olarak görülmüştür. En eski, en ünlü ve en temel modeldir. Geleneksel yazılım geliştirme modeli olarak da bilinir. 

Bu modele her aşama en az 1 kez tekrar edilir. Çağlayan modelinin uygulandığı projeler çok iyi tanımlanmıştır ve kısa sürede bitebilecek projelerdir. Eğer barok modeliyle kıyaslarsak çağlayan modelde dokümantasyon sürecin içindedir, her aşamada dokümantasyon yazılır. Adım atlama gibi bir durum yoktur. Popülerliği günümüzde gittikçe azalmaktadır. Eğer bir aşamada dokümantasyon ve test olmadıysa o aşama tamamlanmış sayılmaz. Bu modelde dikkat edilmesi gereken en önemli şey ise, her ne kadar aşamalar arasında geri dönüş mümkün olsa da analiz aşamasında müşteri ve sistem gereksinimlerinin en ince ayrıntısına kadar belirlenmesi gerekir. Böylece tüm detaylar tasarım aşamasına yansıtılır. Bu modelde en fazla zaman alan iki aşama; tasarım ve analizdir. Ayrıca bu modelde kodlama ve test aşamalarında yapılması gerekebilecek değişikliklerin projeye yansıtılma maliyeti çok yüksektir. Bu modelin olumsuz yanlarından bahsedecek olursak; gerçek hayatta ki projeler genelde yineleme gerektirir, genelde ürünün kullanıcıya ulaşma zamanı uzundur, gereksinimler çoğunlukla net tanımlanamadığından hataları ve eksiklikleri düzeltme maliyeti oldukça yüksektir, kullanıcı proje geliştirme sürecinin içerisinde bulunmadığından projeler de hata payı oranı yükselir ve bu da projeye geri dönüşleri arttırır. Bu geri dönüşler de maliyeti oldukça arttırır.

V Süreç Modeli: Bu model ismini “V” şeklinde bir metot izlenmesinden alır. Sol taraf üretimken sağ taraf sınama yani test işlemleridir. V süreç modeli, çağlayan modelinin gelişmiş hali olarak düşünülebilir. Belirsizliklerin az, iş tanımlarının net olduğu BT (Bilgi Teknolojileri) projeleri için uygun bir modeldir. Bu modelde kullanıcı sürecin daha çok içerisindedir. 

V süreç modeli 3 temelden oluşmaktadır:

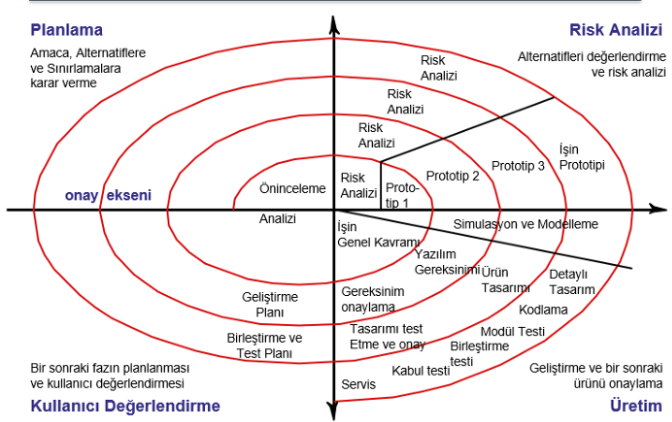
Kullanıcı Modeli: Proje yapım süreci kullanıcı istekleri ve tanımlarına göre şekillendirilir.

Mimari Model: Sistemin tasarımı yapılır ve test işlemleri yapılır.,

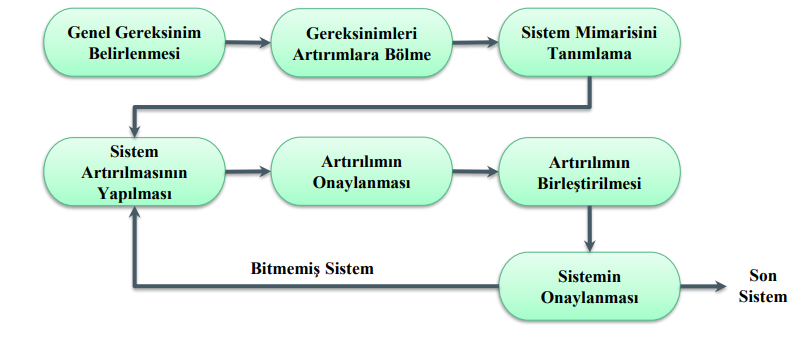
Gerçekleştirim Modeli: Yazılımın kodlanması ve denenmesi gerçekleştirilir.

V süreç modelinin olumsuz yanlarına değinecek olursak; aşamalar tekrar edilmez ve risk çözümleme için ayrı bir bölüm yoktur.

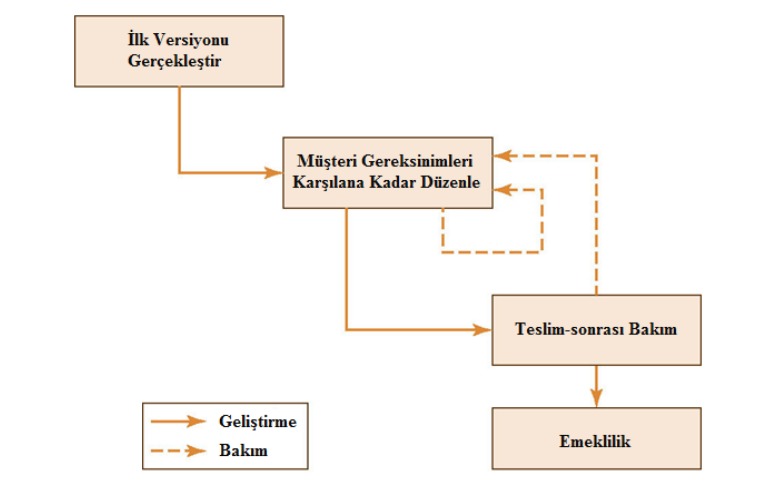
Helezonik (Spiral) Model: Bu modeli diğer modellerden ayıran en emel özellik risk analizinin ön plana çıkması ve prototip hazırlanmasıdır. Risk analizine önem verildiği için hatalar erken fark edilir ve hızlı bir şekilde çözüme ulaştırılabilir. Prototip oluşturulurken her aşamada olduğundan ötürü kullanıcı da her aşamada yazılım projesinin bir kısmını görme imkanına sahip olur bu da sorunların azalmasını sağlayabilir. Helezonik model 4 temel aşamadan oluşur bunlar; planlama, risk analizi, üretim, kullanıcı değerlendirme. Helezonik modelin olumsuz yönlerinden bahsedicek olursak bunlar; küçük ve düşük riskli projeler için çok maliyetli olabilmesi, karmaşık bir model olması, uzun zaman alması ve fazla dokümantasyondan oluşması.



Artımsal Geliştirme Modeli: Bu modelde proje parçalara bölünür ve parçalar kullanıcı ihtiyacına göre sıralanır. Bu modelde sistem tek seferde teslim edilmektense geliştirme ve teslim parçalara bölünür. Projenin parçalarını geliştirmeye başlandığında gereksinimler o an ki haliyle dondurulur ve olası değişiklikler sonraki teslime kalır. Üretilen her sürüm birbirini kapsayan ve daha gelişmiş hali olacak şekilde geliştirilir. Uzun zaman alabilen ve sistemin sorunlu çalışabileceği türdeki projeler için bu model tercih edilmemelidir. Kullanım ve üretim aynı anda yapılır. Artımsal geliştirme modelinde ki temel süreç şu şekildedir:

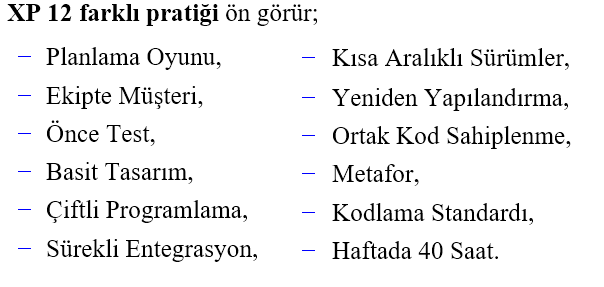


Kodla ve Düzelt Yaşam-Döngü Modeli: Bu model az sayıda koddan oluşan yazılımlar için kullanılabilir. Bu modelde emeklilik aşaması görülür. Büyük yazılım projelerinde kullanılamaz, bakım çok azdır ve zordur. Çünkü sistemin hali hazırda bir dokümantasyonu yoktur. Bu model yazılım geliştirmenin en kolay yoludur ama en maliyetli yöntemdir. Uygulaması kolay olduğu için maalesef küçük ve tecrübesiz firmalar çoğu projesinde bu modeli kullanır.



Çevik Modeller: Bu modeller başarı oranını arttırmayı ve ekip çalışmasını güçlendirmeyi hedefler. Bu modellerin bazıları “Extreme Programming (XP)” ve “SCRUM” şeklindedir.

Extreme Programming (XP): 1999 yılında Kent Beck tarafından yazılım geliştirme disiplini olarak ortaya çıkarılmıştır. XP kolay, ekip çalışmasını önemseyen, geri dönüş kolaylığı sağlayan bir yöntemdir. Bu işlevselliği ve kolaylığı 12 farklı pratiğe borçludur. XP dört temel değere sahiptir. Bunlar; iletişim, basitlik, geri bildirim (feedback), cesarettir. Piyasada ki bir çok projeyi ve yapım aşamalarını incelediğimizde en önemli problemin insanların tam olarak birbirleriyle anlaşamaması olarak gözlemleriz. XP yöntemindeyse bu sorunun üstüne gidilmiş ve bunu çözmek için oluşturulmuştur. XP yönteminde iletişim yüz yüzedir. Geliştiriciler ile kullanıcılar arasında sürekli ve güçlü bir ilişki vardır. Böylece sorunlar erkenden fark edilip hızlıca çözülür. XP’nin 12 pratiği şu şekildedir:



SCRUM: Bu modelin adı Rugby sporunda ki bir hücum taktiğinden gelmektedir. Bu taktikte top, tüm oyuncularla birlikte karşı takım sahasına taşınarak atak yapılır. SCRUM Jeff Sutjerland ve Ken Schawaber tarafından 1990’ların ortalarında geliştirildi. Sadece yazılım projelerine özgü değil, uygun olan her projeye uygulanabilen bir yöntemdir. Karmaşık büyük işleri, daha basit küçük birimlere bölerek geliştirmek üzerine kuruludur. Bu metot karmaşık ortamlarda adım adım yazılım geliştiren ekipler için uygundur. SCRUM’da ekip çalışması çok önemlidir, bundan dolayı her gün “SCRUM Mettings” isimli toplantılar yapılır.



SCRUM’da üç temel kavram vardır. Bunlar; Roller, Toplantılar ve Bileşenlerdir.

Roller: Ürün sahibi, SCRUM Yöneticisi, SCRUM Takımından oluşur. Ürün sahibi projenin ana beynidir. Yöneticiyse, takımı SCRUM prensiplerine göre yönetir. Takım ise birbiriyle sürekli iletişim halinde olan ekiplerdir.

Toplantılar: Sprint Planlama, Sprint Gözden Geçirme, Günlük SCRUM Toplantısından oluşur. SCRUM’da toplantılara büyük önem verilir. Bu toplantılarda o ana kadar neler yapıldığı, hatalar ve bugün neler yapılacağı konuşulur.

Bileşenler: Ürün Gereksinim Dokümanı, Sprint Dokümanı, Sprint Kalan Zaman Grafiğinden oluşur.

SCRUM günümüzün en çok tercih edilen ürün geliştirme modelidir. Sadece yazılım geliştirmede değil, birçok alanda proje geliştirirken tercih edilen yöntemdir. Bunun sebeplerinden bahsedecek olursak:

# Zaman ve para açısından oldukça verimlidir.

# Günümüz teknolojileri ve gelecek teknolojilere kolayca uyum sağlayabilmesi

# Kullanıcıların daha çok dinlenmesi ve hataların daha az olması

# Ekip çalışması ve iletişimi ön planda olması

# Karmaşık yapıların, daha küçük yapılara bölünerek basite indirgenebilmesi

KAYNAKÇA

# <https://medium.com/@omerharuncetin/yazılım-yaşam-döngü-modelleri-543c7879a742#:~:text=Yazılımın%20yaşam%20döngüsü%20tek%20yönlü,boyunca%20süren%20bir%20döngüdür%20diyebiliriz>.

# [**Doç. Dr. Deniz KILINÇ**](https://akademik.yok.gov.tr/AkademikArama/AkademisyenGorevOgrenimBilgileri?islem=direct&authorId=AFFDF876E26B94C2)**(Bölüm Başkanı),** Bakırçay Üniversitesi Yazılım Mühendisliğine Giriş Dersi 2. Ve 3. Hafta sunumları