**Yazılım Yaşam Döngü Modelleri**

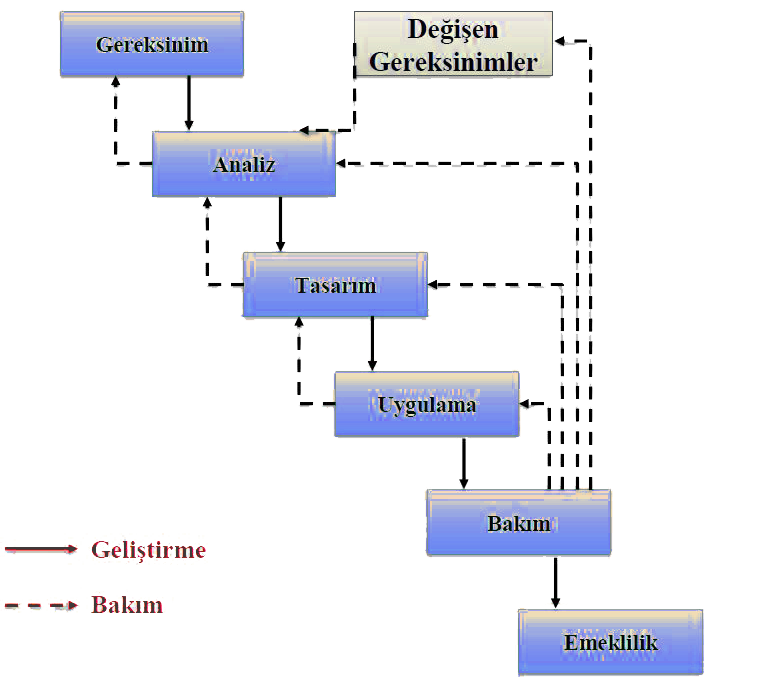
Yazılım yaşam döngü modeli nedir? Bir yazılımı geliştirirken ve bakım sürecini gerçekleştirirken uygulanan adımların tümünün birleşimine yaşam-döngü modeli denir.

Yedi(7) tane farklı yazılım yaşam döngü modeli bulunmaktadır. Bunlar: Gelişigüzel Model, Barok Modeli, Çağlayan Yaşam-Döngü Modeli, V Süreç Modeli, Spiral Model, Artımsal Geliştirme Süreç Modeli ve Kodla ve Düzelt Yaşam-Döngü Modeli’dir. Bu modelleri şu şekilde tanımlayabiliriz:

**Gelişigüzel Model:** Aslında bu model literatürde bir model olarak adlandırılmayan bir modeldir. Çünkü Gelişigüzel Model’de tüm işlemler üretimi yapan kişinin istekleri doğrultusunda ve düzensiz bir şekilde yapılır. Plansız oldukları için bu modelde geliştirilen çalışmaların bakımı, diğer modellerdeki çalışmaların bakımına göre daha zordur. 1960’lı yıllarda kullanılan bu model, özellikle tek kişinin yürüttüğü basit çalışmalarda kullanılmıştır. Şuanda günümüzde kullanılmamaktadır.

**Barok Modeli:** Bu model 1970’li yıllarda ortaya çıkan bir modeldir. Bu modelde gerekli adımlar doğrusal şekilde ele alınır ve geliştirilir. Aşamaları arasında gerekli olan geri dönüşlerin nasıl alınacağına dair bir tanım yoktur. Bu modelde dökümantasyon, günümüzde kullanılan yazılım yaşam döngülerinden farklıdır. Çünkü günümüz yazılım yaşam döngülerine göre ayrı bir süreç olarak ele alınır. Bu modelde, yazılım geliştirilip test edildikten sonra dökümantasyonun yapılması öngörülür. Barok Modeli de Gelişigüzel Model gibi günümüzde kullanılmayan bir yazılım yaşam döngü modelidir.

**Çağlayan Yaşam-Döngü Modeli:** En eski modellerden biri olan Çağlayan Yaşam-Döngü Modeli, geçmişte de en popüler yazılım geliştirme modeli olarak kullanılmıştır. Çağlayan Yaşam-Döngü Modeli’nde yazılım süreci, her bir aşamanın en az bir kez tekrarlanması sonucu oluşur. Tanım olarak üst düzey olan ve üretimi çok fazla zaman gerektirmeyen projeler için uygun bir modeldir. Günümüzdeki kullanım açısından Gelişigüzel Model veya Barok Modeli gibi değildir. Günümüzde hala kullanılsa da kullanımı günden güne azalan bir yazılım yaşam döngü modelidir.



Tüm işlemler tek tek, aşama aşama yapılır. Bir aşama tamamlanmadan başka bir aşamayla uğraşılmaz. Her aşamada mutlaka dökümantasyon yazılması gerekmektedir. Her şeyin en az bir dökümanının olması gerekmektedir. Dökümantasyon ve test gerçekleştirilmeyen bir aşama varsa o aşama tamamlanmamış kabul edilir.

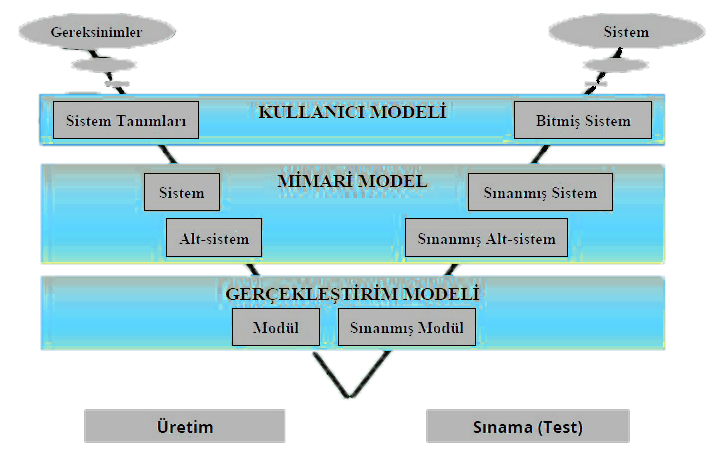
Barok Modeli’nden farklı olarak Çağlayan Yaşam-Döngü Modeli’nde dökümantasyon ayrı bir süreç değil, üretim içerisinde bulunan bir parçadır. Yine Barok Modeli’nden farklı olarak Çağlayan Yaşam-Döngü Modeli’nde aşamalar arasındaki geri dönüşlerin ne şekilde olacağı tanımlanmıştır. Aşamalar arası geri dönüşler yapılabilse de analiz kısmında tüm detayların tasarıma yansıması için müşterinin ve sistemin gereksinimlerinin, ince detaylarına kadar belirlenmesi gerekmektedir.

Tasarım aşamasında da yazılımın karşılaması gereken tüm gereksinimler için kapsamlı bir çalışma gerekmektedir. Çağlayan Yaşam-Döngü Modeli’ni kullanan ekipler en fazla zamanı tasarım ve analiz aşamalarında harcamak durumundadırlar. Ancak geniş zamana yayılan projelerde bile gereksinimlerin değişebilir olması yüksek ihtimalli bir durumdur. Test veya kodlama aşamalarında değişen gereksinimlerin sistem veya yazılıma yansıtma maliyeti çok yüksektir.

Çağlayan Yaşam-Döngü Modeli’nde kullanıcı sürecin içinde yer almamaktadır. Bu sebep ile yazılımın tamamlanmasında sonra kullanıcı fazlaca geri dönüşte bulunabilir. Bu gibi bir durum yazılım geliştirme maliyetini ciddi bir şekilde yükseltmektedir. Ayrıca yazılımın tamamlanıp kullanıcıya iletilme zamanı uzun bir süreçtir. Gereksinim tanımlamaları çoğu zaman net bir şekilde oluşturulmadığı için, düzeltme ve eksiklerin giderilme maliyetleri ciddi miktarlardadır. Genelde yazılım üretme ekipleri bir an önce programı yazma, çalıştırmaya ve sonuç almaya istekli olduğundan Çağlayan Yaşam-Döngü Modeli kullanılarak yapılan projelerde ekibin morali düşebilmektedir.

**V Süreç Modeli:** Özellikle iş tanımlarının daha kesin ve belirsizlik durumlarının daha az olduğu Bilgi Teknolojileri projelerine uygun bir modeldir. Ayrıca V Süreç Modeli işleyişi ile kullanıcının projeye olan katkısını arttırmakta ve projeye dahil olmasını daha kolay bir hale getirmektedir.

Bilgi Teknolojileri projelerinin iki aşamalı olarak ihale edilmesi uygundur. İlk ihalede kullanıcı modeli ve kullanıcı istekleri belirlenerek iş analizi tanımı yapılır. Ve kabul testlerinin tanımları yapılır. İkinci ihaledeyse ilk ihalede elde edilmiş olan kullanıcı modeli tasarlanıp, doğruluğu kabul edilir.



Yukarıdaki model bir V Süreç Modeli’dir. Modelin sağ tarafı üretim işlemlerinin kısmını, sol tarafı ise test işlemlerinin kısmını göstermektedir. Modelin temel süreç çıktıları şu şekildedir:

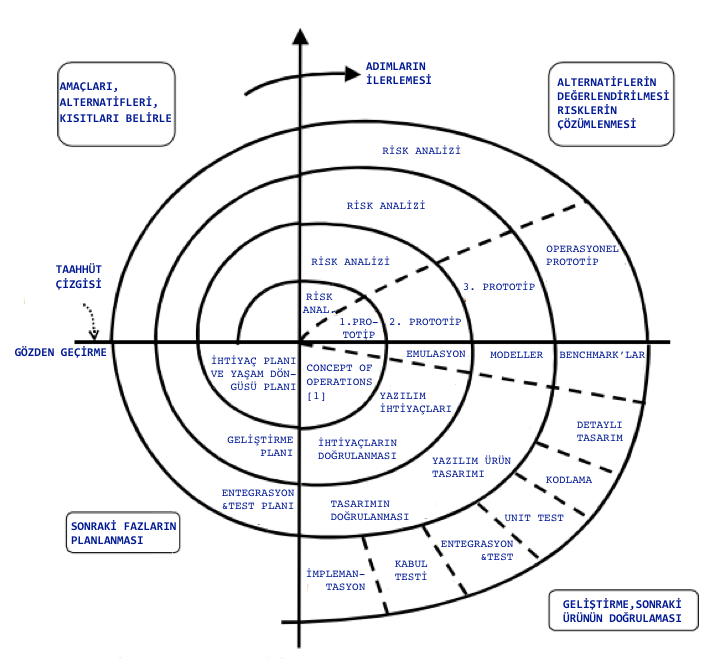
1)Kullanıcı Modeli: Geliştirme süreci ile kullanıcının ilişkilerini tanımlar. Ve sistemin ne şekilde kabul edileceğine dair test açıklamaları ve planları gösterilmektedir.

2)Mimari Model: Sistem tasarımına dair işlevler belirtilir. Oluşacak alt sistem ile tüm sistemin test işlemlerine dair işlevler belirtilir.

3)Gerçekleştirim Modeli: Yazılım parçalarının kodlanması ve test edilmesine dair fonksiyonlar belirtilir.

**Spiral (Helezonik) Model:** Spiral Model 4 farklı kısımdan oluşmaktadır; planlama, risk analizi, kullanıcı değerlendirme ve üretim. Planlama kısmında üretilmesi hedeflenen ara ürün için planlama yapılır. Amaç belirlenir ve bir önceki adımda üretilen ara ürün ile bütünleştirme işlemi gerçekleştirilir. Risk analizi kısmında risk seçenekleri araştırılır ve riskler belirlenir. Üretim aşamasında ise ara ürünün üretimi gerçekleştirilir. Kullanıcı değerlendirmesi aşamasıysa, ara ürün ile ilgili kullanıcı/ kullanıcılar tarafından yapılan test ve değerlendirmelerden oluşur.

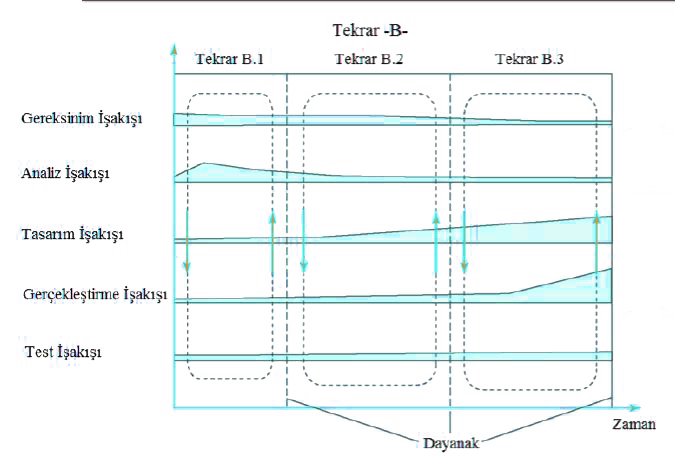
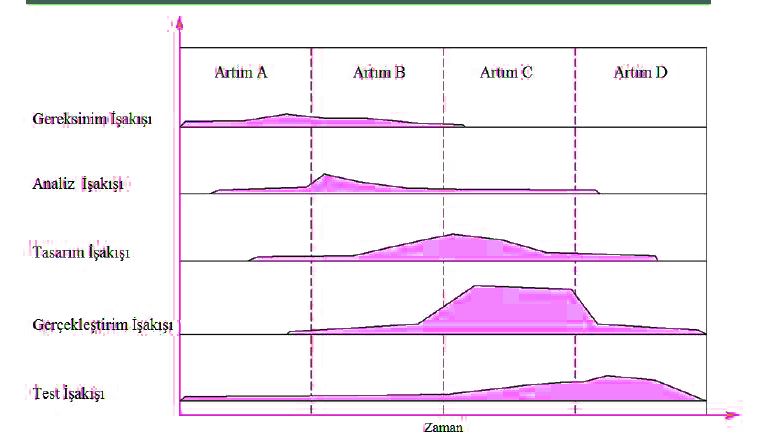
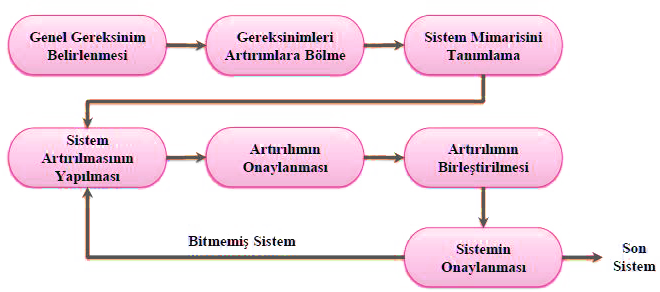
Spiral (Helezonik) Model’de risk analizi kavramı ön plana çıkmaktadır. Her döngü bir fazı temsil eder. Yinelemeli artımsal ve prototip ağırlıklı bir yaklaşıma sahiptir.



Spiral (Helezonik) Model’in kullanıcı, yönetici ve mühendis açısından avantajları bulunmaktadır. Kullanıcıyı sürece erken bir sürede dahil ettiği için proje ilerledikten sonra ortaya çıkacak bir takım sorunlar engellenmiş olur. Üretim süresince ara ürün üretilmesi ve sistematik olarak bu ara ürünün kullanıcı tarafından test edilmesi mantığına dayanır. Yöneticiler, oluşturulan yazılımları komtrol edebilir. Bunun için daha kolay izleme ve hak etmeye dair planlamalar yapılır. Yazılım geliştirici (Mühendis) açısından bakacak olursak Spiral (Helezonik) Modelle ilerleyen bir projede yazılımın kodlama kısmı ve test edilme aşaması daha erken başlar.

**Artımsal Geliştirme Süreç Modeli:** Artımsal Geliştirme Süreç Modeli’nde hazırlanan sistem bir kerede teslim edilmez. Geliştirme ve teslim olmak üzere ikiye ayrılır. Her teslimde beklenen işlevlerin bir parçası karşılanır. Kullanıcının beklediği gereksinimler önceliklidir. Öncelikli gereksinimler erken teslimlerde teslim edilmektedir. Gereksinimler, gereksinimlerin birbirine bağımlılıkları göz önünde bulundurularak önem derecelerine göre sıralanır. Ve her yinelemede bu gereksinimlerin bir kısmı tamamlanır. Eğer bir parçanın geliştirme sürecine başlandıysa gereksinimleri dondurulur. Olası değişiklikler daha sonraki teslim süreçlerinde ele alınır.

Üretilen her yazılım sürümü birbirini kapsar. Ve yazılım sürümü yükseldikçe işlev sayısı da artacak şekilde geliştirmeler yapılır. Bir taraftan test edilme/kullanım yapılırken diğer taraftan da üretim kısmına devam edilir. Yapım aşaması uzun süren ve belli işlevler olmadan da çalışabilecek projeler için Artımsal Geliştirme Süreç Modeli’nin kullanımı uygundur.



Artımsal Geliştirme Süreç Modeli’nde yazılım geliştirme süreci yinelemeli bir süreçtir. Yeni bir sürece geçiş yapıldıkça sonuca daha da yaklaşılmaktadır. Tekrarlama/yineleme olmadan yazılım mühendisliği çalışmaz. Her teslimde müşteri, gözle görülebilir bir değer gördüğü için erken aşamalarda sistemin işlevselliğinin farkına varılabilir. Erken teslimler, sonraki teslimlerin gereksinimlerini belirlemek için bir prototip görevi görebilir. Artımsal Geliştirme Süreç Modeli yapısı itibarı ile projenin tümden batma riskini azaltır. Ve öncelikli gereksinimleri karşılayan sistem işlevleri daha çok sınamadan geçer.

**Kodla ve Düzelt Yaşam-Döngü Modeli:** Kodla ve Düzelt Yaşam-Döngü Modeli birkaç yüz satırdan oluşan programlar için kullanılmaya elverişlidir. İlk adımda yazılım ürününün ilk sürümü geliştirilir. Ancak Kodla ve Düzelt Yaşam-Döngü Modeli’nde sisteme ait bir dökümantasyon bulunmamaktadır. Ayrıca Kodla ve Düzelt Yaşam-Döngü Modeli’nde emeklilik adımı vardır. Teslim sonrası bakımdan sonra emeklilik safhasına geçilir. Ancak teslim sonrası bakımı süreci dökümantasyon olmadığı için Kodla ve Düzelt Yaşam-Döngü Modeli ile yapılan projelerde oldukça zorlayıcı olabilir. Yazılım geliştirme için en kolay yol olarak gözükse de yazılım geliştime için en maliyetli modeldir. Kodlamadan sonra yazılımda değişiklikler sıkça olabildiğinden maliyeti oldukça yüksektir. Ayrıca şartname veya tasarım dokümanı olmadığı için bakım yapma süreci oldukça zorlayıcıdır. Tecrübesiz veya küçük bir ekip tarafından geliştirilen yazılımlarda Kodla ve Düzelt Yaşam-Döngü Modeli ile sıkça kullanılır.

**Hangi Model En İyisidir?:** Bu tamamen üretici ve tasarlayıcı ekiplere kalmış bir durumdur. Ancak bazı durumlar (maliyet, bakım süreci vb.) kullanılacak modeli değiştirebilmektedir. Veya yeterli olmayan, tek düze oluşum içeren bazı modeller (Gelişigüzel Model veya Barok Modeli) günümüzde kullanılmamaktadır. Günümüzde yazılım tasarlama ve oluşum süreçleri artık daha organize yapıldığı için Spiral(Helezonik) Model, Artımsal Geliştirme Süreç Modeli ve özellikle Scrum’ın kullanılması daha olasıdır. Bu üç modelde de sürece kullanıcı dahil edilir. Ama BT (Bilgi Teknolojileri) projeleri için V Süreç Modeli’nin kullanılması daha uygundur. Bu modelde de kullanıcı süreç içerisine dahil edilir. Ancak Kodla ve Düzelt Yaşam-Döngü Modeli’ne baktığımızda, kullanıcı sürece son safhalarda dahil olmaktadır. Yazılım ekibi yazılımı bitirip kullanıcıya sunana kadar kullanıcı proje içerisinde aktif rol oynamamaktadır. Bu sebepten dolayı maliyeti diğer modellerle üretilen projelere göre daha fazladır.

**Scrum Nedir?:**Scrun özellikle son zamanlarda adını sıkça duyduğumuz “Çevik(Agile) Yazılım Geliştirme” adı altında ki kavram sonucunda ortaya çıkan, en popüler yazılım geliştirme modelidir. Yakın geçmişte projelerin daha kapsamlı ve detaylı olması, kullanıcının isteklerini tam olarak tasarım ve yazılım ekibine iletememesi gibi sebeplerden ötürü “Çevik Yazılım Geliştirme Manifestosu” karşımıza çıkmıştır. Çevik Yazılım Geliştirme Manifestosu’na göre süreçten daha çok bireylerin etkileşim halinde olması gerektiğini, kapsamı yüksek olan dokümantasyondan çok çalışan bir yazılıma, sabit bir plana bağlı kalmaktansa değişime karşılık vermeye dayanmalıdır. Bu mantıkla tasarlanıp üretilen projelerde son aşamada bile gereksinimlerin değiştirilmek istenmesi daha kabul edilebilir oldu, bildirimlerin kısa süreli olmasından dolayı hata payı azaltıldı ve çalışan yazılımlar müşterilere sunuldu. İş sahibi ve yazılımcılar süreç içerisinde yüz yüze iletişime dayalı bir bilgi alışverişinde bulundular. Bu gibi olumlu durumlar “Çevik(Agile) Yazılım Geliştirme” mantığı doğrultusunda daha fazla projenin üretilmesine bir ivme kazandırdı.

Scrum ise Çevik(Agile) proje yönetim metodolojilerinden birisidir. Daha komplike olan yazılım süreçlerini yönetmek için kullanılır. Bütünü parçalara ayıran ve tekrar içeren bir yöntem izler. Planlama ve düzenli geri bildirimler(feedback) ile hedefe ulaşmayı amaçlar. İhtiyaca yönelik esnek bir yapıya sahiptir. Müşterinin ihtiyaçlarına göre şekillenmesi sebebiyle müşterinin geri bildirimine göre yapılanma sağlar. İletişimin ve takım çalışmasının çok önemli unsurlar olduğu Scrum’da 3 temel prensip vardır:

1. **Şeffaflık:** Projeye ait tüm gelişmeler, problemler veya projenin bulunduğu durum herkes tarafından görülebilir olmalıdır.
2. **Denetleme:** Projenin ilerleyişi düzenli bir şekilde kontrole tabi tutulmalıdır.
3. **Uyarlama:** Proje, sonradan yapılabilecek değişikliklere uyum sağlayabilmelidir.

Günlük toplantılar ile ekip yapılan işlerden ve takım arkadaşlarından sürekli haberdar olurlar. İletişim unsuru arttığı için de proje çalışanları içerisindeki kopukluk durumu en az seviyeye inmiş olur. Ekip 3 (üç) soruya başlıca cevaplar verir:

1. Dün ne yaptım?
2. Bugün ne yapacağım?
3. Beni engelleyen sorunlar var mı?

Aslında bu temel 3 soru çalışmanın ve çalışan kişilerin verimini arttırma, sorun ve hedeflerin daha iyi belirlenmesi için kişinin kendisine sorduğu sorulardır.

**Scrum Günümüzde Neden Popüler?:** Genel olarak baktığımızda Scrum’ın yaygın bir kullanımı olmasının sebepleri şunlardır: Komplike projeleri daha iyi bir şekilde ele almak, çalışan ekip arasındaki iletişimi kolaylaştırmak, daha seri bir şekilde geri dönüşler almak, her ne zaman olursa olsun projede değişikliğe hazır olmak, kullanıcı ve üretici ekip arasında yüz yüze bir iletişim ile projeyi iki tarafın da içine sinecek halde tasarlayıp oluşturmak ve çok komplike projeler olduğu için müşterinin en başta projenin bütününü görememesi ihtimaline karşılık projeyi her daim esnek bir yapıda tutmaktır. Bu gibi sebepler, günümüzde Scrum’ın popüler olmasına sonuç sağlamaktadır.

**Kaynakça:**

<https://dhalsim.github.io/software/2015/11/14/software-development-processes>

<https://medium.com/@secilcor/scrum-nedi%CC%87r-6a4326951dd8>

<https://www.startupnedir.com/scrum-nedir-2/>

Doç. Dr. Deniz Kılınç, Bakırçay Üniversitesi Yazılım Mühendisliğine Giriş Dersi 2 ve 3. Hafta Sunumları

**Hazırlayan:** Ahmet Altunsu 200601076

https://ahmettaltunsu.medium.com/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BC-modelleri-a147dc2ca873