## Yazılım yaşam döngü modelleri nedir?

En basit şekli ile yazılım yaşam döngü modelleri belli bir yazılımın geliştirilmesinin planlanmasıdır. Bu ‘planlamalar’ yazılımın geliştirilirken takip edilecek safhalardan ibarettir. Bahsi geçen safhalar kullanılan yazılım yaşam döngü modeline göre değişebilir. Yazılım yaşam döngü modelleri Elliott & Strachan & Radford’a (2004) göre 1960’larda büyük firmalar için büyük çaplı fonksiyonel iş sistemleri yapmak için kullanılmaya başlandı.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

## Neden yazılım yaşam döngü modelleri kullanılır?

Yazılım yaşam döngü modellerinin 1960’lardan beri kullanılmasının sebebi yazılım mühendislerine nelerin ne kadar sürede yapılacağını ve müşteriye nelerin ne kadara, ne zamana yapılacağını açıkça anlatması ana sebeplerden biridir. Başka önemli bir sebep ise yazılım yaşam döngü modellerinin yazılım mühendislerine projeler ile ilgili şablon oluşturmasıdır, bu şablonlar sayesinde yazılımlar daha organize bir şekilde tamamlanır.



## Yazılım geliştirme yaşam döngüsünün (SDLC) temel aşamaları nelerdir?

Farklı yazılım yaşam döngüsü modellerinde kullanılan adımlar birbirinden farklı olabilir ama çoğunda ortak bulunan ve endüstri tarafından kabul edilen bazı adımlar vardır; bu adımlar sırasıyla gereksinim, analiz, tasarım, gerçekleştirme, teslim-sonrası bakım, emeklilik aşamasıdır.

Gereksinim adımında müşterinin ne istediğini, neye ihtiyaç duyduğunu belirlemek hedeflenir.

Analiz adımında toplanmış gereksinimler analiz edilir, şartname dokümanı yazılır, yazılım proje yönetim planı da bu adımda hazırlanır.

Tasarım adımında 2 farklı tasarım yapılır: üst seviye ve mimari tasarım ile kaba taslak plan ve modüller; ayrıntılı tasarım ile veri tabanı, algoritmalar, sınıflar, diyagramlar vs. tasarlanır.

Gerçekleştirme (Implementation) adımında her bir modül birbirinden ayrı kodlanır ve test edilir, bu test edilen modüller birleştirilir ve ortaya çıkan hatanlar düzeltilir, gerçek veriler ile yazılım test edilir.

Teslim-sonrası bakımı düzeltici ve özellik arttırıcı bakım olmak üzere ikiye ayrılır. Düzeltici bakımda yazılımdaki hatalar düzeltilir, Özellik arttırıcı bakımda şartname dokümanına veya müşterinin isteğine göre yazılıma yeni özellikler eklenir.

## Bazı yazılım yaşam döngü modelleri nelerdir?

1960’larda yazılıma artan talep sonucu yazılım firmaları, karmaşık yazılımları daha verimli bir şekilde üretmek için zaman içinde farklı yazılım yaşam döngü modelleri oluşturup kullanmaya başladılar.

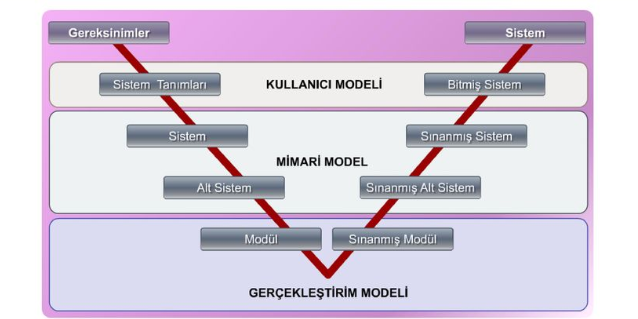
Gelişigüzel model: Teknik olarak model değildir. Kullanılan model veya yöntem yoktur. 1960’larda genellikle bir kişi tarafından basit bir yazılım üretirken kullanılırdı.

Barok modeli: Bu model 1970’lerde ortaya çıkmıştır, yazılım geliştirme yaşam döngüsünün (SDLC) temel adımları doğrusal bir şekilde yapılır, gerekli geri dönüşlerin nasıl yapılacağı tanımlı değildir, dokümantasyon gerçekleştirme aşamasından sonra yazılır. Günümüzde kullanılmamaktadır.

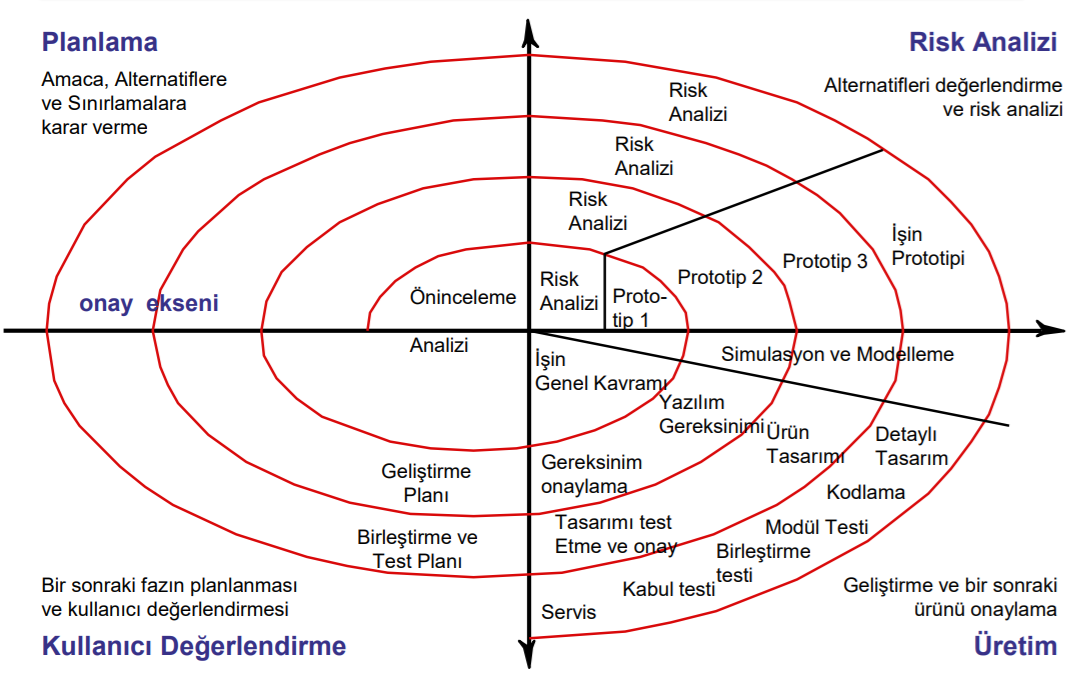
metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

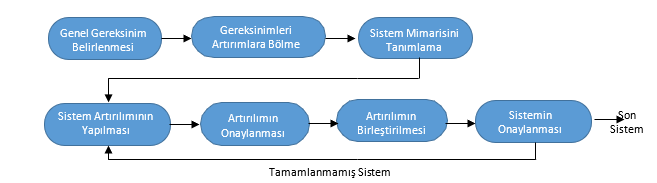
Çağlayan yaşam-döngü modeli: Geleneksel yazılım geliştirme modeli olarak da bilinir. Yazılım adım adım yapılır, bir adım bitmeden diğerine geçilmez, her adımda dokümantasyon yazılır ve yazılım test edilir, her adım en az bir kez tekrarlanır. En fazla zaman tasarım ve analiz aşamalarında harcanır. Uzun zamana yayılan projelerde adımları tekrar etmek önemli olduğu için bu model tercih edilebilir. Bu modelin kullanımı günümüzde azalmaktadır.



V süreç modeli: ‘V’ harfinin sol tarafı üretim, sağ tarafı test tarafıdır. Proje gereksinimlerin alınmasıyla başlanır, üretim bittikten sonra testler başlanır, son üretilen ilk test edilir. Belirsizliklerin az, kesinliklerin çok olduğu projelerde kullanılır. Kullanıcı projeye katıldığı için teslim-sonrası hata düzeltme güncellemeleri daha az olur. Bu yaklaşım iki aşamalı ihale edilen BT projeleri için oldukça uygundur.



Helezonik (spiral) model: Helezonik modelin müşteriden gelen geri bildirime göre yinelemeli artımsal bir yaklaşımı vardır. Her döngü bir fazı temsil eder, risk analizi vardır. İkinci ve sonrasındaki fazlar kendilerinden önceki faz(lar)ın geri bildirimlerine göre tekrardan planlanır, analiz edilir, kodlanır. Helezonik yaklaşımda müşteri katkısı fazladır bu yüzden son üründeki müşteri memnuniyeti fazla, teslim-sonrası hata düzeltme güncellemeleri daha azdır; SDLC’nin en fazla zaman tüketen gereksinim ve analiz adımları helezonik modelde her fazda tekrar edildikleri için normalden az vakit tüketirler. Bu yaklaşım kullanıcı geri bildirimlerini kolay alması ve az süren gereksinim, analiz adımları sayesinde mobil uygulamaları geliştirmek için idealdir.



Artımsal geliştirme süreç modeli: Artımsal geliştirme süreç modelinde sistem tek seferde teslim edilmek geliştirme ve teslim küçük paketler bölünür. Müşterinin gereksinimleri öncelikli olduğundan dolayı ilk paketler müşterinin gereksinimlerinin en temel halidir. Bir paket geliştirilmeye başlandığında o pakette hiçbir değişiklik yapılmaz, değişiklikler sadece paket tesliminden sonra ele alınır. İkinci ve sonraki paketler sisteme büyük yamalar olarak gelir. Müşterinin gereksinimleri tamamlandıktan sonra teslim-sonrası hata düzeltme güncellemeleri yapılabilir.

## Modellerin karşılaştırılması

Yukarıda 6 farklı modeli açıkladım. Bu modellerin yazılım geliştirmeye birbirinden farklı yaklaşımları, güçlü ve güçsüz yanları, hedef kitle/platformları bulunmakta.

Gelişigüzel model: Teknik olarak model değildir. Günümüzdeki tek kişi tarafından yapılan küçük hobi projeleri bu modeli kullanır, plan-program ve dokümantasyon yok. Teslim-sonrası bakımı zor.

Barok modeli: Barok modelinin olumlu yanları gelişigüzel modele göre daha organize, SDLC adımları takip ediliyor olması, ekip olarak proje üzerinde daha rahat çalışılır; olumsuz yanları gerekli geriye dönüşlerin nasıl yapılacağı tanımlı değil, dokümantasyon yazmak gerçekleştirme adımından sonra ayrı bir evre olarak ele alınır.

Çağlayan yaşam-döngü modeli: En basit ve en temel model olarak bilinir. Çağlayan yaşam-döngü modelinin olumlu yanları yazılımın adım adım yapılması, her adımda dokümantasyon yazılması ve yazılımın test edilmesi, her adımın en az bir kez teslim edilmesidir; olumsuz yanları müşterilerin yazılımın geliştirilme sürecinde yazılım mühendisleri ile temaslı olamaması, yüksek risk olması, karışık ve nesneye yönelik yazılımlar için iyi bir model olmamasıdır.

V süreç modeli: V süreç modelinin olumlu yanları kullanması basit, test aşamaları diğer modellere göre daha verimli olmasıdır; olumsuz yanları esnek olmaması, yazılımın prototipi olmamasıdır. V süreç modeli küçük-orta boyutlu gerekliliklerin kesin olduğu projelerde kullanılır.

Helezonik (spiral) model: Helezonik modelin olumlu yanları güçlü dokümantasyonlar olması, büyük ve önemli projeler yapılabilmesi, fonksiyonelliğinin sonraki bir tarihte arttırılabilmesi, prototipin erken bir tarihte ortaya çıkmasıdır; olumsuz yanları diğer modellere göre kullanması pahalı olması, alanda uzman biri tarafından yapılmazsa risk analizi yanlış bir sonuç verebilmesi, küçük projelerle pek uyumlu değildir. Helezonik model genelde yüksek riskli projelerde kullanılır.

Artımsal geliştirme süreç modeli: Artımsal gelişim süreç modelinin olumlu yanları projenin başında bir prototip üretebilmesi, diğer modellere göre daha esnek olması, test etmenin daha kolay olmasıdır; olumsuz yanları ise düzgün bir olana ve dizayna ihtiyaç duyması, sistemi minik paketlere ayırabilmek için gerekliliklerin açık ve tam bir tanıma ihtiyaç duyması, toplam maliyetin çağlayan yaşam-döngü modeline göre daha fazla olmasıdır. Artımsal geliştirme süreç modeli bir yazılımı markete erken çıkarma gereği duyuluyorsa, yeni bir teknoloji kullanılıyorsa kullanılır.

KAYNAKLAR: Ahmet Ertuğrul KAYA 200601049

<https://en.wikipedia.org/wiki/Systems_development_life_cycle#History_and_details>

BİL102 3. hafta ders sunumu

<https://tr.wikipedia.org/wiki/Waterfall_model#/media/Dosya:Waterfall-tr.png>

<https://en.wikipedia.org/wiki/Systems_development_life_cycle#/media/File:SDLC-Maintenance-Highlighted.png>

<https://medium.com/@omerharuncetin/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BC-modelleri-543c7879a742>

<http://www.vgizy.com/yazilim-muhendisligi-ve-yazilim-gelistirme/>

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

<http://tryqa.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/>

<http://tryqa.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/>

<http://tryqa.com/what-is-spiral-model-advantages-disadvantages-and-when-to-use-it/>

<http://tryqa.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/>