Yazılım Yaşam Döngüsü nedir

Bir yazılımın planlanmasından itibaren kullanım ve bakımına kadar geçirdiği tüm aşamalar ve bu aşamalardan oluşan döngüdür. Doğada her zaman bir döngü vardır. Canlılar doğar gelişir yaşar ve ölürler. Yazılımın da tıpkı canlılar gibi bir doğal yaşam döngüsü vardır. Bu döngünün doğru kontrol edilmesi ve analiz edilmesi durumunda yazılım başarılı bir şekilde gelişir.

Yazılım Yaşam Döngü Modelleri, projelerin kalitesine etki ettiği gibi, üretim sürecindeki karışıklıkları önleyerek uygun düzeni sağlar. Özellikle birçok çalışanın içinde bulunduğu büyük projelerde, büyük şirketlerde yazılım yaşam döngü modelleri yazılımın olmazsa olmazıdır. Analiz aşamasından başlayıp bakım aşamasıyla son bulan yazılım yaşam döngü süreci, içerisinde birkaç temel aşama bulundurur. Bunlar;

1. Planlama: Yazılım yaşam döngüsünün ilk adımı olan bu aşamada müşteriyle birebir iletişim halinda olunmalıdır. Çünkü müşteri, gereksinim analizin için en faydalı olabilecek kaynaktır. Müşterinin istekleri net ve eksiksiz bir biçimde dinlenip dikkate alınmalıdır. Ayrıca bu aşamada projedeki temel ihtiyaçlar, maddi ve teknik kaynaklar belirlenmelidir ve görev dağılımı yapılmalıdır.
2. Analiz: Yazılım yaşam döngüsünün en önemli aşamalarından biri olan analiz aşamasına ‘’Çözümleme’’ aşaması da denilir. Bu aşamada projenin tüm işlevleri ayrıntılı olarak belirlenir. Buna bağlı olarak sistem gereksinimleri netleşir, uzmanlar tarafından proje ile ilgili tüm raporlar yazılır ve tüm UML diyagramları çizilir. Ayrıca projenin ne kadar süreceği, ne gibi riskler taşıdığı, avantajlı ve dezavantajlı yönleri detaylı bir şekilde ele alınıp rapora dökülür.
3. Tasarım: Bu aşamada analiz aşamasıyla birlikte ortaya çıkan proje detayları ve gereksinimleri ele

alınarak uygun arayüz, yazılım veya bilgi sisteminin temel yapısı oluşturulur, ancak kodlama henüz söz konusu değildir. Bu aşama temel olarak 2 gruba ayrılır:

-Mantıksal Tarasım: Önerilen, düşünülen sistemin yapısı ele alınır

-Fiziksel Tasarım: Yazılım içeren bileşenler ve bunların detayları incelenir.

1. Gerçekleştirme: Bu aşama, tasarım aşaması yeterli olgunluğa geldikten sonra kodlama ile başlar.

Sistemin yaşatılmaya başlandığı aşama olarak düşünülebilir. Test etme ve kurulum safhaları ile devam

eder.

1. Bakım: Teste etme ve kurulum aşamasından sonra gerçek ortama kurulan yazılım ürününün; hata giderme, güncelleme gibi işlemlerinin yapıldığı aşamadır. Bu aşama yazılım yaşam boyunca süren tek aşamadır.

Yazılım yaşam döngüsünde birden fazla model vardır. Bu modellerin çok sayıda olmasının; yazılım projesinin büyüklüğü, projede çalışacak kadronun büyüklüğü ve çeşitliliği, yazılımı kimlerin kullanacağı gibi sebepleri vardır. Yazılım yaşam döngü modellerinin birebir karşılaştırılması doğru değildir, çünkü her yazılım yaşam döngü modelinin avantajlı ve dezavantajlı olduğu noktalar vardır. Yani x modeli y modeline göre daha doğrudur denilemez, mevcut projeye göre hangi model daha uygunsa o seçilmelidir.

Bu yazıda ele alacağım yazılım yaşam döngü modelleri şunlar:

1)Gelişigüzel Model

2)Barok Modeli

3) Çağlayan Modeli

4) V Modeli

5) Kodla ve Düzelt Modeli

6) Artımlı Geliştirme Modeli

7) Spiral Model

1) Gelişigüzel Model

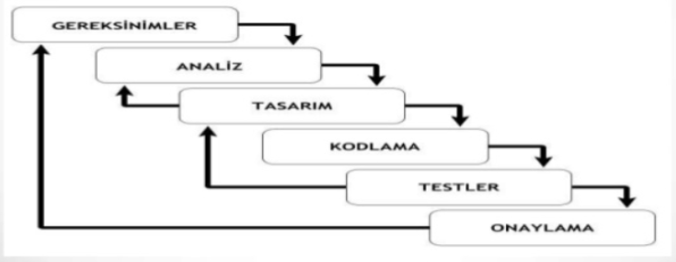
1960’larda kullanılan bu modelin aslında bir model olarak adlandırılması doğru değildir. Genelde amatör yazılımcılar tarafından kişiye özel yapılır ve takip edilmesi çok zordur. Günümüzde artık kullanılmamaktadır

2) Barok Modeli

1970’lerde kullanılan bu model yaşam döngü aşamalarını temel olarak doğrusal bir şekilde ele alır. Gerçekleştirim aşamasına daha fazla ağırlık verir. Günümüzde kullanımı görülmemektedir.

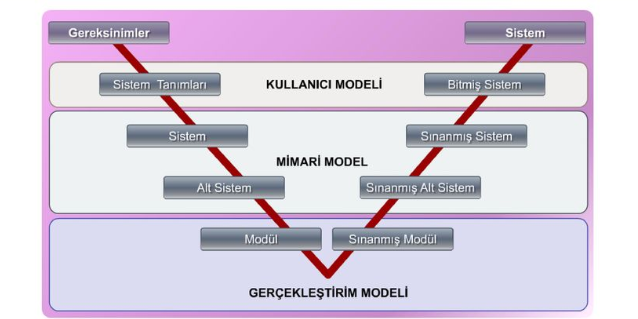
3) Çağlayan Yaşam Döngü Modeli

Bu modelde yazılım yaşam döngü aşamaları baştan sona en az 1 kez tekrar edilir. Kendi içerisinde oldukça tutarlı, kullanımı ve anlaşılması basit ve yönetimi kolay bir model olmasına karşın günümüzde kullanımız giderek azalmaktadır. Bunun başlıca sebepleri ise şunlar olarak düşünülebilir; Projede yapılan her değişiklik maliyeti büyük oranda arttırır, diğer modellere göre uzun sürebilir, müşteri memnuniyetini sağlamak zordur.



4) V Modeli

Süreç adımları kodlama evresinden sonra yukarıya doğru eğim alıp V şeklini oluşturan bu model, Çağlayan yaşam döngü sürecinin gelişmiş hali olarak düşünülebilir.



Görüldüğü üzere bu model aslında 3 temel modelden oluşmaktadır:

-Kullanıcı Modeli

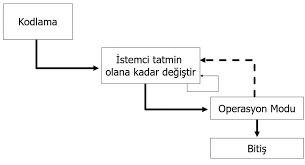
-Mimari Model

-Gerçekleştirim Modeli

Proje yönetimi tarafından takibi kolay ve kullanımı kolay olan bu modelin, aşamalarda tekrar bulunamama ve risk çözümleme için ayrı bir yer olmaması gibi sıkıntıları da vardır.

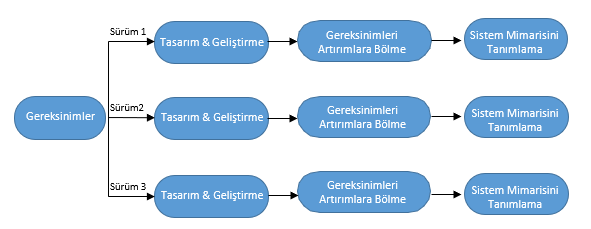
5)Kodla ve Düzelt Modeli

Genelde küçük programlar için kullanılan bu basit model, direkt ürün gerçekşetirir. Büyük projeler için kullanılmamasının sebebi ise kontolsüz, düzeltilmesi güç ve bakımın hayli zor olması denilebilir.



6)Artımlı Geliştirme Modeli

Bu model genelde ürünlerinde değişikliğe ihtiyacı olan müşteriler için kullanılır. Yazılımı bir takvime bağlı olarak, bölüm bölüm geliştirip teslim etmeye dayanır. Her yeni bölümde bir öncekinin üstüne bazı ek işlevlerin eklendiği görülür. Az sayıda personel ile bu model uygulanabilir ancak personelin tecrübeli olması gerekir. Bir başka dezavantajı ise artımlı parçaları hatasız oluşturmak için sistemin detaylı bir şekilde tanımlanması gerekir ki bu da çok kolay değildir.

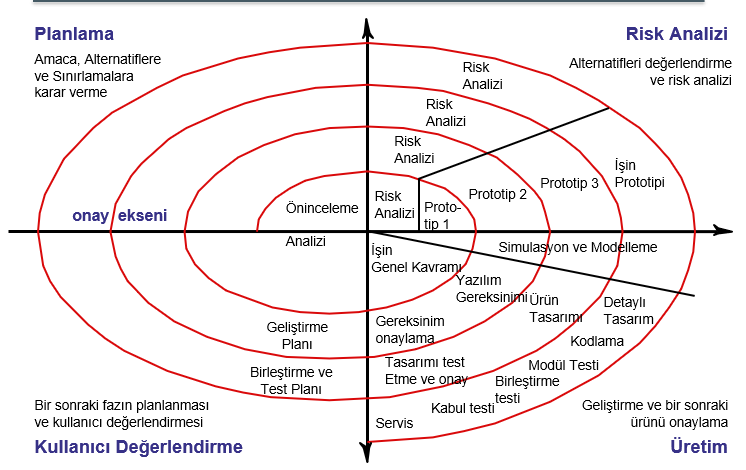


7)Spiral (Helezonik) Model

Risk analizinin ön planda tutulduğu bu modelde süreçler, her bir halkanın bir fazı gösterdiği belirttiği spiral şeklinde gösterilir. Süreci küçük parçalara bölerek riskleri erkenden tespit edip düzeltilmesini amaçlar. 4 temel aşamadan oluşur:

-Planlama -Risk Analizi -Üretim -Kullanıcı Değerlendirme

Dezavantajlarını ise şöyle sıralayabiliriz: Ara adımların çokluğu nedeniyle fazla dökümantasyon gerekebilir, karmaşıktır, küçük projeler için pahalıdır.



Genel olarak bu yazımda bu yazılım yaşam döngü modellerini ele aldım, fakat bunların sayısı artırılabilir bu kadarla sınırlı değil. Başta da değindiğim gibi her modelin artı ve eksi yönleri var ve her yazılım projesine uygun ve uygun olmayan modeller mevcut. Peki hangi projelerde hangi modeli kullanmalıyız ?

* Belirsizliğin az olduğu, iş tanımlarının net olduğu projelerde V Modelini
* Büyük, maliyetli ve uzun sürecek projelerde Spiral veya Artımsal Geliştirme Modelini
* Küçük, bireysel projelerde Kodla ve Düzelt Modelini
* Küçük ve analizi iyi yapılmış projelerde Çağlayan Modelini

kullanmak uygundur.

Volkan DUMAN

190601019