Yazılım Yaşam Döngü Modelleri

Yazılım da canlılar gibi yaşayan bir varlıktır ve yaşam döngüsüne sahiptir. Yazılım yaşam döngüsü, bir yazılımı oluşturmak ve kalitesini kontrol etmek için uygulanan süreçtir. Yazılım geliştirirken uygulanan ilk döngü elemanı planlamadır. Döngünün en önemli aşamalarından biridir. Bu aşamada projenin görev dağılımı ve genel planlaması yapılır, yazılım geliştirilmeye başlanmadan önce ne istenildiği belirlenir. Sonraki aşama tanımlamada ise temel kavramlar açıklanır, tanımlamaları yapılır ve planlama evresine geçilir. Planlama evresine alınan kararların doğru bir şekilde tanımlanması ve açıklanması projenin sağlıklı bir şekilde ilerlemesi ve devamlılığı için çok önemlidir. Planlamadan sonra tasarlama evresine geçilir. Geliştirilen yazılımın projeleri çizilir. Geliştirme aşamasında projesi çizilen ve az çok şekillenen yazılımın artık kodları yazılmaya başlanır. Kodlar yazıldıktan sonra test evresine geçilir. Geliştirilen yazılıma testler yapılır ve uygulama evresine geçilir. Uygulama evresinde geliştirilen yazılım artık kullanıma hazır hale gelmiştir, yazılım kullanılır ve varsa hatalar tespit edilip ekleme çıkarma işlemleri yapılır. Son aşama olan bakım evresinde geliştirilen yazılımın güncellemeleri ve genel bakımı yapılır.

Yazılım yaşam döngüsünün bazı avantaj ve dezavantajlarını sıralayacak olursak:

\*Yazılım geliştirilirken elde edilen bilgiler sistemsel olarak analiz edilebilir. Bir problem oluşması durumunda problemin döngünün hangi adımında oluştuğu ve nasıl çözüleceği gibi konularda kolaylık sağlar.

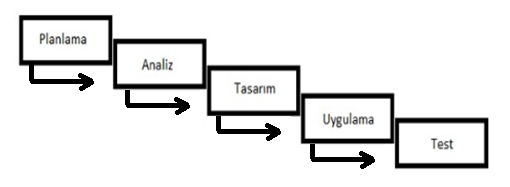
\*Dinamiktir yani uzun yıllar boyunca kullanılabilirliğe sahip.

\*Yazılım yaşam döngüsü büyük projeler için kolaylık sağlasada küçük projeler için bütçe ve zaman açısından verimli olmayabilir.

\*Yazılım yaşam döngüsü sürecini yönetmek tecrübe ve uzmanlık ister.

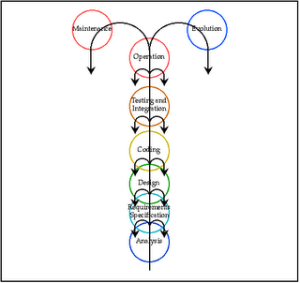
Yazılım yaşam döngüsü, geliştirilen yazılımın kimin tarafından kullanılacağına veya projenin büyüklüğü gibi faktörlere göre birden fazla modellere ayrılır:

1.Şelale Modeli



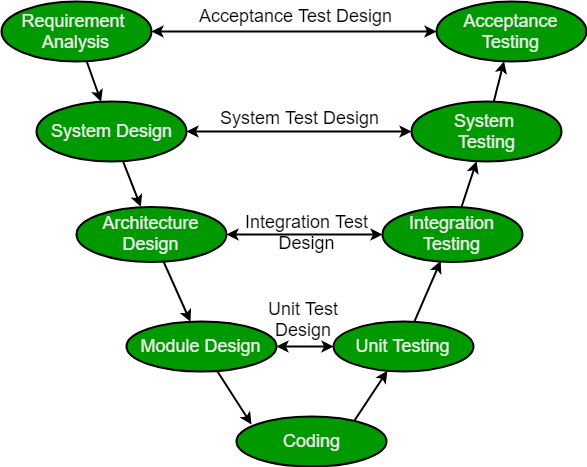
Bu modelde bir adımdan sonraki adıma geçerken, önceki adım veya adımların eksiksiz tamamlanması gerekmektedir. Bu durum projenin ilerleyen aşamalarda farklı kısımların ele alınması gerektiğinde zorluk gösterir. Müşteri ile iletişim azdır. Daha çok statik projeler için uygundur. Yazılım projeleri için uygulanabilecek en kötü modeldir. Yazılım projelerinde çok sayıda ve sıklıkla bakım ve güncellemeler gerekli olduğu için bu projelerde kullanılması tavsiye edilmez.

2.Fıskiye Modeli



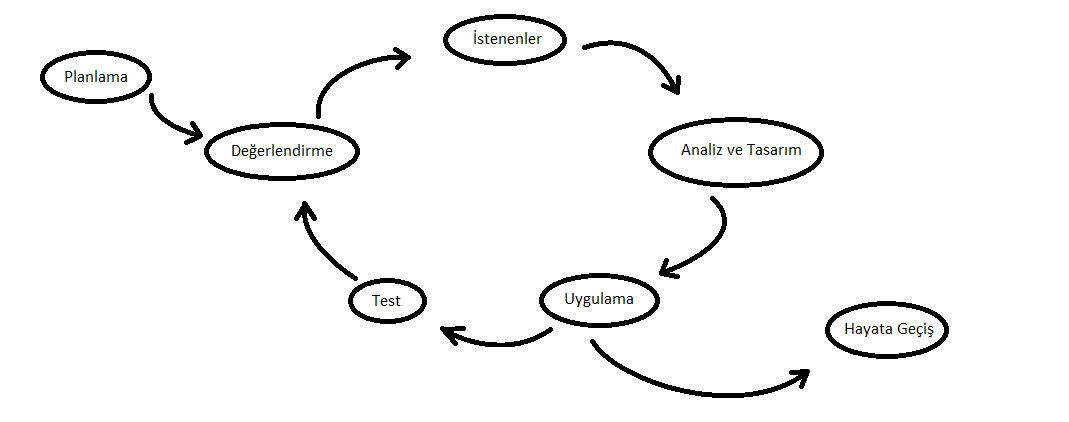
Brian Henderson-Sellers ve JM Edwards tarafından geliştirilen bu model şelale modelinden ilham alarak oluşturulmuştur. Her bir adım kendi döngüsüne sahip olduğundan ileri bir aşamadan geri dönülmesi diğer modellere göre daha kolaydır. İki ekip ile uygulanır. Ekiplerden biri bakım işlemlerini yürütürken diğer ekip eklenen yenilikleri kontrol eder.

3.V Modeli



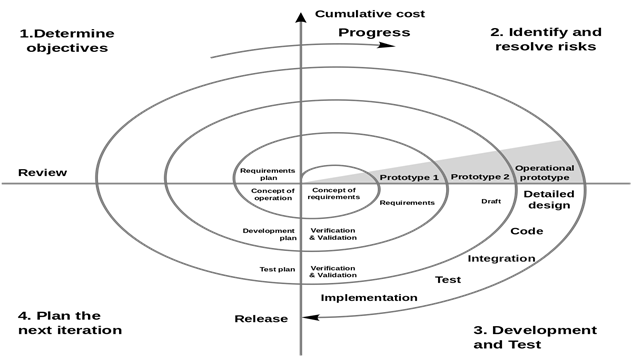
Şelale modeline benzer bir modeldir. Fark olarak tasarım daha detaylı olarak ele alınır. Aynı seviyede olan kutular birbiri ile ilişkilidir. Yapılan işlerin kontrolleri kısa sürede sağlanır. İhtiyaçlar ile bitmiş sistem karşılaştırılır, nelerin yapılabildiği veya yapılamadığı kontrol edilir. Risk çözümlemede yetersizdir.

4.Artırımlı Model



Artırımlı modelde uygulama aşamasından sonra proje hayata geçer fakat aynı zamanda testler de devam eder. Gereksinimler müşterinin görüşleriyle birlikte belirlenir ve değerlendirilir. Hayata geçen projeden gelen bilgiler ve testler değerlendirildikten sonra döngü tekrarlanır, döngü sürekli beslenerek devam eder.

5.Spiral Model



Spiral modeli diğer modellerden ayıran en önemli özelliği risk analizinin diğer modellere göre daha ön planda olmasıdır. Risk analizi odaklı olduğu için hataların daha kolay giderilmesine imkan tanır. Büyük projeler için kullanılması uygundur. Bu avantajlarına karşın düşük bütçeli projeler için fazla maliyetli ve karmaşıktır.

6.Kodla ve Düzelt Modeli

Sonuca hızlı ulaşmaya odaklı bir modeldir. Genellikle düşük bütçeli ve küçük projeler için kullanılır. Hızlıca bilgi toplanarak yazılım geliştirmeye başlanır. Hızlı ve basit olmasına karşın büyük projelerde kullanılmaya uygun değildir. Bakım aşaması zordur.

7.Çevik Model

Projeler ilerledikçe yaşanan zorluklar artmaktadır. Bu zorlukları azaltmada çevik model etkin rol oynar. Çevik model müşteri ile olan etkileşime, çalışanlar arasındaki iletişime odaklanır. İletişimden doğacak olan sorunlar minimuma indirildiği için müşteriden gelen bildiri ve geri dönüşler daha iyi değerlendirilir, takım içi çalışma daha verimli olur. Çevik model kendi içinde farklı modellere ayrılır:

\*Extreme Programming

Dört temel maddeden oluşur:

1-İletişim: Hızlı geri bildirim odaklıdır. Alınan bildirimler hızlı ve yeterli sayıda olduğundan hayta yapma olasılığı ve yanlış anlaşılma durumları minimuma indirilir.

2-Basitlik: Geri bildirimlerin en hızlı şekilde işlenebilmesi için basit olması gerekmektedir. Ayrıca basitlik, çözümlerde kavramayı kolaylaştırdığı için avantaj sağlar.

3-Cesaret: Sorumluluk almaktan korkulmamalıdır.

4-Geri Bildirim: Müşteriyle mutlaka iletişim içinde olunmalı ve geri bildirim alınmalıdır. Alınan geri bildirimler projenin ilerlemesinde büyük önem taşır.

\*Scrum

Scrum, karmaşık ve büyük projelerde kullanılır. Görev dağılımı yapılır, proje parçalara ayrılır ve her parça ayrı ayrı geliştirilir. Parçalara ayrılan projenin her parçası tamamlandığında müşteri ile bir araya gelinip değerlendirmesi yapılır, geri dönüşler alınır. Proje parçalara ayrıldığı için ekip içi iletişim ve geri bildirimler çok önemli rol oynar. Görev dağılımını yapan bir kişiye ihtiyaç duyulur. Bu kişi her gün kısa süreli toplantılar, görüşmeler düzenleyerek ekip üyelerinin çalışmalarına engel olan bir durumun olup olmadığını, önceki gün neler yaptıklarını, bugün neleri yapacaklarını konuşur. Bu toplantılar projede oluşabilecek problemlerin erkenden tespit edilip gerekli müdahalelerin yapılabilmesine olanak sağlar. Scrum günümüzün en popüler sistemlerinden birisidir. Bunun nedenleri:

\*Müşteriyle sürekli etkileşim halinde olma ve geri bildirim alma,

\*Ekip içi iletişime önem verilmesi,

\*Projede değiştirilmesi gereken bir durum olduğunda erken fark edilebilmesi ve müdahale edilebilmesi,

\*Proje parçalara ayrıldığı ve her parça ayrı ayrı ele alındığı için yüksek verim alınabilmesi,

\*Modern teknolojilere uyum sağlayabilmesidir.

Hangi Projelerde Hangi Modeller Tercih Edilmeli

\*Şelale Modeli: Yeniliklere açık olmadığı için inşaat gibi sabit projelerde kullanılması daha uygun olur.

\*V Modeli: Büyük projeler için kullanılası daha uygundur.

\*Artırımlı Model: Genellikle e-ticaret sitelerinde, mobil uygulamaların geliştirilmesinde kullanılır.

\*Spiral Model: Büyük projelerde, maliyet değerlendirmesinin yüksek önem taşıdığı, yüksek riskli projelerde kullanılır.

\*Kodla ve Düzelt Modeli: Küçük çaplı ve kısa ömürlü projelerde kullanılır.

\*Çevik Model: Büyük ve karmaşık projelerde kullanılır.

Kaynakça

\* <http://ybsansiklopedi.com/wp-content/uploads/2015/08/Yaz%C4%B1l%C4%B1m-Geli%C5%9Ftirme-Modelleri-Yaz%C4%B1l%C4%B1m-Ya%C5%9Fam-D%C3%B6ng%C3%BCs%C3%BCSDLCYBS.pdf>

\* <http://bilgisayarkavramlari.com/2013/01/10/sdlc-yazilim-gelistirme-yasam-dairesi/?highlight=yaz%C4%B1l%C4%B1m%20ya%C5%9Fam>

\* <https://eternalsunshineoftheismind.wordpress.com/>

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

\* Doç. Dr. Deniz Kılınç, Bakırçay Üniversitesi, Yazılım Mühendisliği Temelleri 3. ve 4. Hafta Sunumları

Alperen BAYRAKDAR Bilgisayar Mühendisliği 200601079