## **SEFA ÖZTÜRK 190601073**

## **Yazılım Yaşam Döngüsü Nedir?**

Bilindiği üzere insanların çocukluk,gençlik,yaşlılık gibi evreleri vardır.Bu durum esasında yazılımlarda da geçerlidir.Yazılımlarında kendi içerisinde serpilme,gelişme dönemleri diye nitelendirdiğimiz dönemleri vardır.Birkaç örnek verirsek tasarım aşaması başlı başına bize yeter. Mesela tasarım aşaması olmaz ise bu sefer kesin olmamakla birlikte düzensizlik ortaya çıkar.Tıpkı dereyi görmeden paçayı sıvamak gibi...Siz ne yapacağınızı aklınızda tam oturtmazsanız yazılım sürecinde ve sonrasında sıkıntı çıkabilir.Yazılım süreci sadece bilgisayarın karşısına geçip tüm gün kod yazmak demek değildir.Öncesi ve sonrası da yazılıma dahildir.Bu sebeptendir ki yazılım yaşam döngüsünün her aşaması bizim için altın niteliğindedir.Zaten ilerleyen sayfalarda bunu daha net anlayacağız.

Yazılım yaşam döngüsü 5 aşamadan oluşur.Şimdi ise bunları anlayalım.

**Planlama**: Yazılım projesinin planlanması ve görev dağılımı yapılır.

**Analiz**: Yazılımın ne kadar zaman alacağı ne gibi risklerinin olabileceği belirlenir.

**Tasarım**: Yazılımın nasıl idame ettireceği belirlenir.

**Gerçekleştirme**: Yazılımın kodlara dökülme aşamasıdır.

**Bakım**: Yazılımın kullanıma sunulduktan sonra yeni versiyonlarının çıkmasıdır.Mesela online oyun şirketlerinin “bug”diye nitelendirdiğimiz bir takım hataları bir sonraki patchte o bugu ortadan kaldırmasıdır.

## **Yazılım Yaşam Döngü Modelleri**

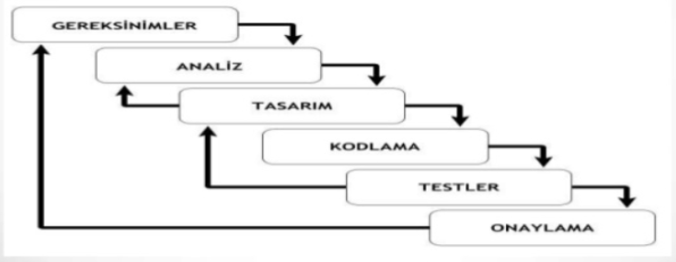
·**Barok ve Gelişigüzel Yaşam Döngü Modelleri**

**Gelişigüzel Model:** Adından da anlaşılacağı üzere herhangi bir süreç yoktur tamamıyla hasbelkader ilerlemektedir.Böyle olması işleri zorlaştırır.Eski modeldir günümüzde kullanılmaz.

**Barok Modeli**:Önceki modeli gibi rastgele ilerlemez.Biraz daha sistematiktir fakat bu eski olduğu gerçeğini değiştirmez.Bu modelde dosyalama o kadar da önemli değildir bununla birlikte süreçler daha ciddi şekilde ele alınmaz.Günümüzde kullanılmaz.

**Şelale (Waterfall) Yaşam Döngü Modeli**

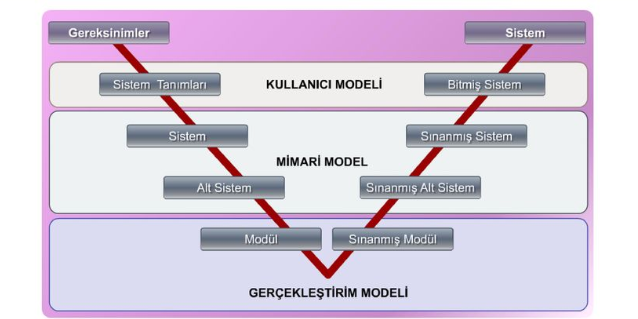
Şelale modeli düzenli ve tertipli yazılım aşamalarının temel taşıdır.Sonraki anlatacağımız projeler ise şelalenin bir nevi türevleridir diyebiliriz.



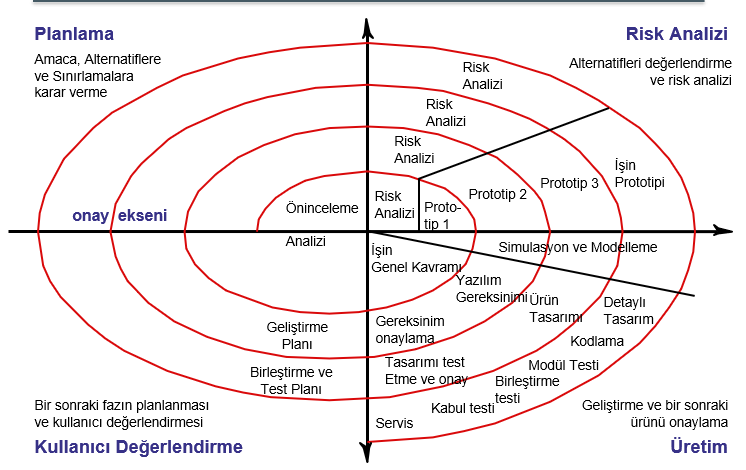
Fakat her ilk süper olmadığı gibi şelale modelinin de hataları vardır.Şelale modelinde her aşama bir kez test edilir.Bu esasında büyük projeler için mantıklı değildir.Mesela devasa şirketler vardır;Microsoft,Google,Yandex,Nvidia...Bu tarz şirketler zaten kolay kolay basit projeler yapmazlar.Büyük projelerinde zahmeti büyük olur,zaman alır.Projeler zaten zaman alıyor birde şelale döngüsünü de işin içine dahil edersen her aşamayı tek tek sınaman gerekir.Dolayısıyla süreç uzadıkça uzar.Nitekim bu model daha çok küçük çaplı yazlımlar için kullanılır.

**V Süreç Modeli:**

Bu süreç şelale modelinden daha mantıklı bir modeldir diyebiliriz zira bu modelde neyin ne olacağı daha çok kesindir.



· **Helezonik (Spiral) Model**



Helezonik model aslında büyük işler için birebirdir lakin dokümantasyon konusunda sıkıntı çıkartabilir zira çok kapsamlı ve bir o kadar da pahalı bir sistemdir.Büyük projelerde kullanılmasının en önemli sebebi ise risk matrisinin ciddi derecede yer edinmiş olmasıdır.Neredeyse her aşamada risk matrisini görmekteyiz fakat küçük projeler için bana sorarsanız gerek yoktur çünkü çok zahmetli bir modeldir.Muvffakiyet oranı yüksektir zira prototip konusunda ve kullanıcıya danışma konuşunda örnek teşkil edebilecek özelliğe sahip bir modeldir.Şimdi ise bu modelin temel taşlarını ele alalım.

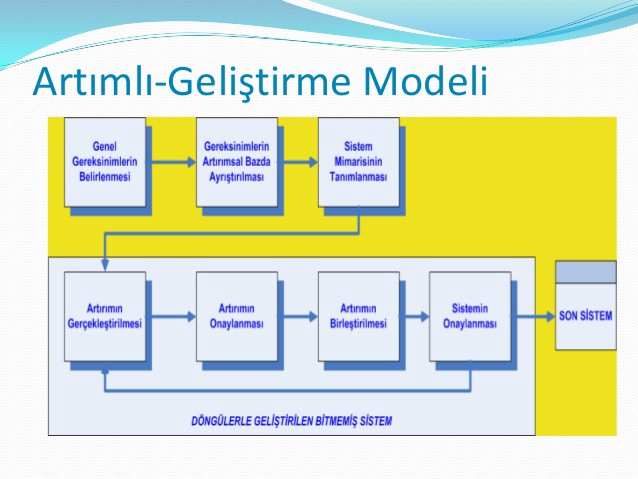
Plan: Plan yapılır

Risk Denetimi: Riskler ölçülür.

İmalat: Süreç boyunca yazılımın her evresinin üretildiği bölümdür.

Müşterinin bakış açısı: Müşterinin görüşü çok önemlidir.Bir insan kendi yaptığı birtakım yazılımlarda hata göremeyebilir çünkü onu en iyi kendisi bilir hatta en iyisi olduğunu bile düşünebilir ama bu yazılımın gelişmesi açısından çok sağlıklı değildir.Farklı fikirlere ve düşüncelere amiyane tabiri ile ezelden beridir ihtiyaç duyulmuştur.Hülasa müşteri yani kullanıcının fikirleri sorulmalıdır.Çünkü bunun neticesinde oluşmuş olan yazılımlar adındanda anlaşılacağı üzere kullanıcılar tarafından kullanmaktadır.

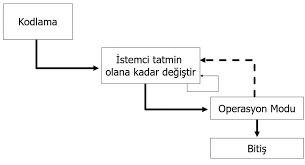
**Artımsal Geliştirme Modeli**



Arttırımlı model demek kabaca tabirle üstüne peyder pey yani zamanla ekleme yapa yapa geliştirmek demektir.Bunu birçok yerde görebiliyoruz.Bu modelde bir yandan üretim olurken diğer yandan da kullanıcılar tarafından kullanılıyor.Hal böyle olunca kullanıcılarla temas olmasından dolayı risk ve başarısızlık şansı azalıyor tabi kesin kalkıyor diye bir yorum yapamayız.Hiçbir modelde kesinlik ifadesi kullanamayız.

**Kodla ve Düzelt Yaşam Döngü Modeli**

Sanıyorum bu modeli açıklmaya gerek yok çünkü ismi her şeyi belli ediyor.İnsanların “işte aradığımda buydu.”demelerine kadar kodlamayı bırakmazsın.Zaten geniş kapsamlı projelerde bu tarz basit modeller kullanılmaz.Daha çok basit projelerde uygulanır.



**Evrimsel Geliştirme Modeli**

Arttırımlı-Geliştirme modeli ile kardeş denilebilir.Farkları ise arttırımlı da kullanıcılar bir yandan da ara ürünü kullanabilirken evrimselde böyle bir durum yoktur.Ara ürünlerin tam teşekkürlü olması gerekir.Aynı zamanda tam ölçekli modellerin babasıdır.

**Çevik Modeller**

Çevik modeller yazılım projelerindeki başarı oranlarını ve ekip içi iletişimi arttırmak amacıyla ortaya çıkmıştır. Bazı çevik yazılım geliştirme modelleri şunlardır:

· **Extreme Programming (XP)**

4 genç tarafından bir nevi bulundukları durumu düzeltme ve böyle olmaması gerektiği savunan bir modeldir.Ve bunun 4 kuralı vardır.Basit olma,Cesaret duygusu,Geri dönüş,Diyalog.

Gelin bunlar üzerinde düşünüp konuşalım.

Basitlik: Kodların ve işlerin fazla sıkıntıya girmeden,çile çekmeden anlaşılır,naif ve basit olan kuraldır.

Cesaret: Yapılan işte cesur olunmalıdır, projelerin üstüne korkmadan ilerlenmelidir. Bir kodun gerekirse tamamen silinip yeniden yazılması sağlanmalıdır.

Geri Dönüş:Ben paramı verdim ne yaparsanız yapın bana istediğimi verin düşüncesinden daha çok kullanıcının yazılımcı ile etkileşim halinde bulunmasıdır.

İletişim: Her ne kadar mühendiste olsalar sonuçta onlar da bir insandır robot değildir.Onların da duygu ve düşünceleri vardır.Ekip içi diyalog hem özgüven açısından hem motivasyon açısından her zaman önemlidir fakat bu her zaman böyle olmayabilir.İnsanına göre huy değişir.Kimisi çok düşüncesizdir lafını düşünmeden söyler,kimisi çok gamsızdır,kimisi çok alıngandır.Bu durum insanları üzebilir işi aksatabilir dolayısı ile insanların bir arada bulunması muhabbet etmesi her zaman iyidir diyemeyiz.Hülasa(sonuç olarak) burada anlatılmak istenen,yazılımcıların birbirlerine karşı kırıcı olmadan konuşmasıdır,birbirlerini desteklemesidir.Kusuruyla hatasıyla görmezden gelip adeta kardeşmiş gibi çalışıp yardımcı olunmasıdır.

· **SCRUM**

Scrum dediğimiz şey esasında takım oyunudur.Bana sorarsanız futbol ile aynıdır.Örneğin futbolda 11 kişi vardır.Her bir futbolcunun bir görevi vardır.Bu takım başı boş değildir.Takımı yöneten,ne yapması gerektiğini öğreten,söyleyen teknik direktörü vardır.Ve bu takım tarafta hizmet eder.Emin olun scrumın bundan hiçbir farkı yoktur.Başta bir yönetici vardır bunun yanında ekip vardır.Ekibin sayısı net değildir.Şirketine göre değişir.Burada daha çok takım oyunu önceliklidir.Ve birde her zaman olduğu gibi klasik müşteri vardır.Zaten müşteri olmazsa eğer hiçbir şey olmaz.Meslekler çoğu zaman ihtiyaçların getirmiş olduğu gerekliliktir**.**



Scrumlarda takım oyunu dediğimiz bir yapı olduğu için iletişi kaçınılmazdır.Hal böyle olunca kimin ne iş yaptığı kimin nerede kaldığı çok önemlidir.Ve haftanın belirli günlerinde toplanıp toplantı adı verilen konuşmalar olur.Yalnız sırf toplanmak için toplanılırsa bu sefer işler aksar.Amaç toplanmaktan ziyade fikir alışverişidir.

Araçlar: Gerekli olan bir takım ihtiyaçların karşılanması ciddi ölcüde önemlidir.Bunlar birtakım maddeler olabileceği gibi fikirler ya da düşünceler de olabilir.

SCRUM günümüzde en çok kullanılan yazılım geliştirme yöntemidir. Hatta sadece yazılım geliştirmede değil birçok sistemin geliştirilmesinde de kullanılır. Bunun nedenleri ise

Düzenli bir sistem olmasından ötürü para ve zamandan tasarruf

Bir elin nesi var iki elin sesi var mantığı ile birçok zorluğun üstesinden gelinmesi,

Kullanıcı ile bağın kesilmemesi,

Diğer yazılım geliştirme metodolojileri gibi yinelemeli olması,

Değişime ayak uydurması,

Küçük parçalara ayıklayıp sorunların çözülmesi,

## **Hangi Yazılım Geliştirme Modeli Daha İyi**

Bilindiği gibi milenyum çağında doğru ilerlemekteyiz ve bununla birlikte teknoloji de giderek ciddileşmektedir.Dolayısı ile **barok** ve **gelişigüzel** modeller çağımıza ayak uyduramıyor.Zaten kullanan sayısıda neredeyse kalmadı.Eski olmasından ve dokümantasyon içermemesinden hülasa bir düzeni olamamasından dolayı günümüzde bu iki modelin bir getirisi yoktur.

Şelale modeli daha çok küçük ve ne olacağı az çok kestirilen işler için kullanılır.Barok ve gelişigüzel gibi eskiden sıklıkla kullanılan modeldi yalnız zamanla oda etkisini yitirenler listesine girdi.Esnek değildir.Dolayısıyla günümüzde büyük şirketler tarafından sıcak karşılanmaz. Basit projelerin öncüsüdür.

V modelini başta yazdığım gibi esasında şelale modelinin bir üst versiyonudur denilebilir fakat bana kalırsa v modeli şelaleye nazaran daha mantıklı bir modeldir zira kullanıcılar ile bir temas vardır.

Fakat helozonik model gibi bir risk matrisi yoktur.Hal böyle olunca da büyük projeler için kesin kullanılmaz demiyorum lakin kullanılmadan önce 1 kez daha sağlam kafayla düşünülmesi gerekilir.

Spiral model büyük şirketler için bire birdir.Emsal olarak ciddi bir işe ciddi bir bütçe ile girilir ve bu bütçeyi de yazık etmeden,risk almadan ve yahut alınacaksa düşük ihtimalli risklere kafa tutarak alınması gerekir.Kullanıcılar ara ürünlerin prototipinden faydalandığından ötürü hata payı ciddi derece düşer ve bununla birlikte gelişimde katkı sağlanır.

Artımsal Geliştirme modelinde adından anlaşılacağı üzere artımsal yani ekleme yapmak bir başka deyişle geliştirmek ve dahası.

Büyük projeler için bir şey söylemem fakat onun dışında her proje için kullanılmasında bence sakınca yoktur.

Kodla ve Düzelt modeli kişinin isteğine göre kodlanmış programlara göre oluşturulan modeldir.Zahmetlidir zira kimin kaç kez isteyeceği belli değildir.Kimi ilkinde tatmin olur,kimi kırk kez yapsan tatmin olmaz.Dokümantasyon işleri ile pek uğraşılmaz.

Çevik modeller tıpkı scrumlar gibi takım içi etkileşimli ismi gibi hızlı,esnek modeldir.Orta dereceli programlar için kullanılabilir.Sosyal olmak gerekir ve doğru iletişim olmazsa olmazdır.

Evrimsel Geliştirme Metodunda kullanıcılar gereksinimleri daha iyi anlarlar, hata ihtimali düşer ve bakımı zahmetlidir.Çünkü evrimselde ara ürününde ciddi bir ürün olması gereklidir.

## **Hangi Projede Hangi Modeli Kullanalım?**

**Bence;**

Belirsizliklerin az olduğu, iş tanımlarının belirgin olduğu bilgi teknolojileri projeleri için V Modeli uygun bir modeldir.

Büyük kitlelere ulaşacak projelerde evrimsel geliştirme metodu ve spiral model de kullanılabilir.

Büyük bütçe ile hazırlanan projelerin zarar etmemesi için helozonik model kullanması gerekir zira risk matrisi bu modelde çok önemli yer teşkil eder dolayısıyla yüksek zararlara girilmemesi içindir ki bu model en uygun olur.

Orta ve küçük büyüklükte, uzun sürmeyen projelerde çevik model,

Kişinin kendi şahsına münhasır olacaksa şayet kodla ve düzelt,

Küçük ve özellikleri iyi tanımlanmış projelerde şelale modeli ve bir ihtimal V modeli kullanılabilir.