Yazılım yaşam döngüsü , bir yazılımın oluşturulurken geçirdiği tüm aşamalarının bir bütün altında adlandırılmasıdır.

Bu aşamalar şunlardır;

Planlama: İlk aşamadır. Nasıl bir bina yapılmadan önce planı çıkartılıyorsa yazılımda da aynı şekilde yapılacaklar belirlenir, görev dağılımı yapılır ve ihtiyaçlar belirlenir.

Analiz: Yapılan görev dağılımlarından sonra yapılacak yazılımın gereksinimleri ve yapıcağı işlevler hakkında ayrıntılı olarak analiz edilir. Önceden yapılmış olan işler incelenir ve oluşabilecek sorunlar gözden geçirilir. Analiz sonrasında yapılacaklar dokümante edilerek muğlaklık aza indirilir.

Tasarım: Yapılan analizler sonucu oluşan gereksinimlerin tasarım aşamasında bir zemine oturtama çalışmaları başlar. Yazılımın tasarım aşamasında müşterinin ihtiyaçları ve istekleri üzerine yapılacak işin özellikleri, yapıcakları ve görünümü belirlenir. İki çeşit tasarım söz konusudur. Mimari tasarım, yazılımın genel yapısı ve etkileşimlerini ilgilendirir ve tasarımla ilgili dokümanlar oluşturulur. Detaylı tasarımda mimari tasarım aşamasında yaptığımız dokümanlar revize edilirler. Bu aşamada daha çok ne-nasıl soruları soruları ile ilgilenir.

Gerçekleştirme: Gerçekleştirme aşaması kodlama ve test aşamasıdır. Sonunda yazılımcıların olmak istediği aşamaya gelmiş ve müşterinin istediği yazılımın kodlanması başlar. Kodlamayla birlikte tested önemli bir unsurdur çünkü yazılımcılar insandır ve insanlar hata yapar bu yüzden hataların en aza indirgenmesi için kodlamanın yanında test işlemleride başlar.

Teslim ve Bakım: En son aşamalardır. Bu aşamada müşteriye hazırlamış olduğumuz yazılımın ürününü teslim eder ve kullanımdan sonra oluşabilicek hatalar için bakım sürecinide ekleriz.

Yazılımlarda bir gereksinim sonucu ortaya çıkmakla beraber pek çok ürünün olduğu gibi bir hayatı vardır. Bu yaşam döngülerininde farklı çeşitleri vardır, bunlar;

* Gelişigüzel Model
* Barok Model
* Çağlayan (Şelale-Waterfall) Modeli
* V Modeli
* Helezonik (spiral) Model
* Artımlı Geliştirme Modeli
* Kodla ve Düzelt
* Çevik Modelleridir (XP, Scrum).

Gelişigüzel Model:

Adında model diye geçmesine rağmen bir model ve ya yöntem fark etmeksizin yazılan yazılımların model şeklidir. Hata düzeltmesi ve anlaşılması oldukça zordur. Genellikle tek kişilik yazılım çalışmalarında görülür ve 1960 yıllardan kalmaktadır.

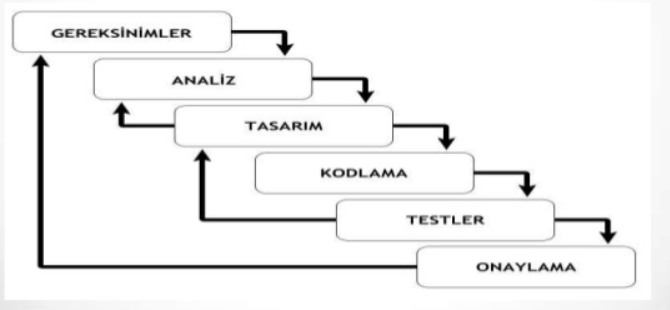
Barok Model:

Yazılım yaşam döngüsü adamlarını direk düz bir şekilde hareket eden bir modeldir. Geri dönüş yoktur ve dokümantasyon ayrı süreç olarak gösterilir. Yazılım bittikten sonra yazılımla ilgili dokümantasyon başlar. 1970 yıllardan kalmaktadır ve gelişigüzel model gibi artık günümüzde kullanılmamaktadır.

Çağlayan (Şelale-Waterfall) Modeli:

Çağlayan modeli günümüzün temelli olarak adlandırılan bir modeldir ve diğer modellerin atası olarak nitelendirilir. Geleneksel bir model olmakla birlikte eski zamanlarda en çok kullanılan modeldir. Bu modelde her aşama gereksinimi kadar en az 1 kere tekrarlanır ve gerekiyorsa geri dönüşler yapılabilir. Bir aşama bitmeden diğer aşamaya geçiş yapılmaz ve her aşamada dokümantasyon çok önemlidir, dokümantasyon yazılım yaşam döngüsünün bir parçası halindedir. Bu yüzden aşamalar arasında en çok zaman harcanan aşamalar analiz ve gereksinim aşamalarıdır. Fazla dokümantasyon muğlaklığı nerdeyse 0’a indirir ama müşterinin sonradan gelen isteklerini engelleyeceğinden müşteri yazılım geliştirilmesindeyken içinde olamaz. Eğer müşteri bir isteğini yaptırmak isterse bu hem zaman hem de para kaybına yol açar. Öte yandan en çok zaman aşamalar gereksinim ve analiz aşamaları olduğundan yazılımcıları mutsuzlaştırır ve üründe hata çıkma olasılığını artırır.

Bu model daha çok iyi tanımlanmış ve hemen bitmesi istenen işlerde kullanılır. Barok modeliyle kıyaslarsak en büyük fark şelale modelinde dokümantasyon direk yaşam döngüsünün bir parçası iken barok modelde ayrı ele alınmaktadır. 1980 yıllarında ortaya çıkmıştır ve hala daha en çok bilinen döngü modelidir.

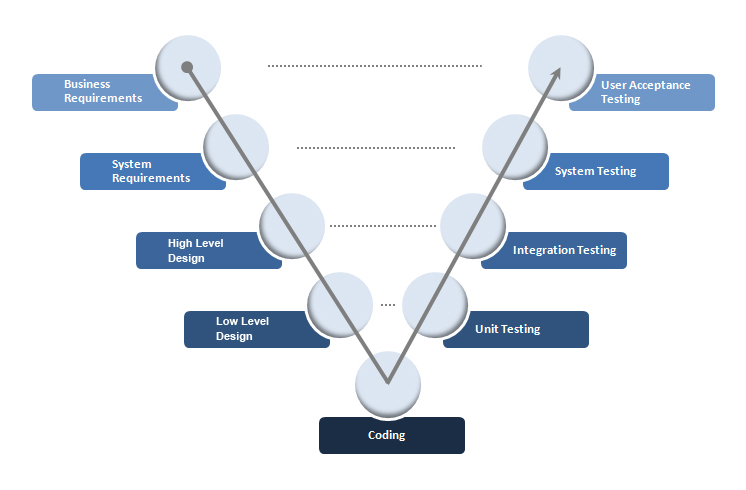


V Modeli:

V Modeli adını ilerleyiş biçiminden almaktadır. Gelişmiş bir şelale modeli de denebilir. Bu model muğlaklıkların çok az olduğu ve tanımlamalarının düzgün olduğu işlerde kullanılır. İlk başlangıçta üretim başlar ve üretim işi bittikten sonra test aşaması başlar ve tüm aşamalar sınanarak hata payı en aza indirilir. Döngü 3 tane modelden meydana gelmektedir. Bu modeller kullanıcı modeli, mimari model ve gerçekleştirme modelidir.

Kullanıcı modelinde müşterinin ihtiyaçları ve yapılmasını istedikleri işe tanımlanır ve bitirilmiş hali müşterinin kullanımına sunulur. Mimari modelde en başta bahsettiğimiz gibi tasarımı ve bunun sınanmasından oluşur. Gerçekleştirme modelinde klasik olarak kodlama ve bu kodlarının testi yer alır.

V Modellinin en büyük sıkıntısı çağlayandaki gibi aşamalara geri dönüş yoktur ve risk çözümlemede yer almaz.



Helezonik (Spiral) Model:

Helezonik modelde ön plana çıkan özellik risk analizidir. 4 temele ayrılan bir yapıdadır ve spiral şeklinde dönerek ilerleyen de bir yapısı vardır. Bu modelde amaç risk analizinin çokluğundan oluşan hataların erken fark edilip hata payını çok az yapmaktır. Bu modelle birlikte prototip kavramı da baya kullanılmaya başlar. Prototip oluşturma her aşamanın sonunda müşteriye somut bir veri sunar ve müşterinin döngünün içine dahil olmasını kolaylaştırır ve bu da risk analizlerinde müşterinin isteklerini karşılayıp karşılamadığını iyi görebilmemizi sağlar. 4 temele ayrılan bu yapı şu şekilde adlandırılırlar;

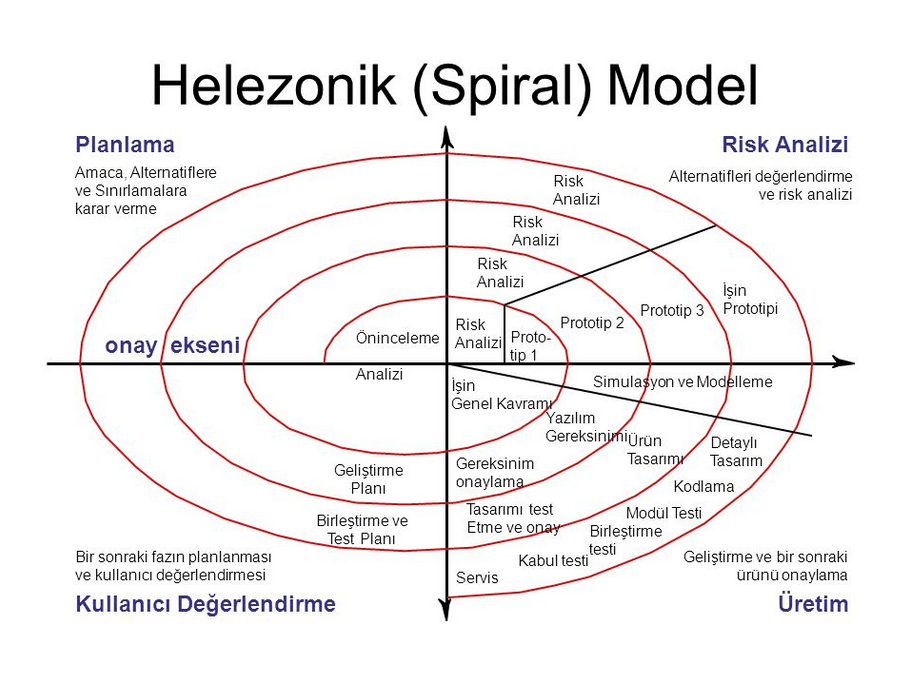
Planlama: Aşamaların sonunda oluşacak prototip için araştırma ve gereksinimler ele alınarak planlaması yapılır.

Risk Analizi: Diğer modellerden en çok ayıran yapıdır. Riskleri analiz edip hataların tespitini ve doğru haline getirildiği aşamadır.

Üretim: Prototip ve ana ürün bu aşamada oluşturulur. Prototipler gittikçe daha çok istenene yaklaştırılır ve en sonunda ana yazılım ortaya çıkarılır.

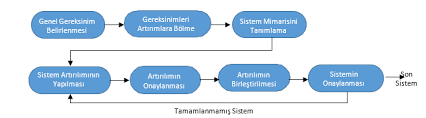
Kullanıcı Değerlendirme: Oluşturulan Prototipler müşteri tarafından denenerek oluşan sıkıntılar ve ya başka istekler değerlendirilerek kullanıcının tam istediği bir yazılım elde edilmeye çalışılır.

Helezonik modelin günümüzde fazla kullanılmamasının nedenleri yapılan işe oranla risk oranının çok düşük olmasından çok pahalıya mal olması, kolay anlaşılamayan bir yapıda olması, zaman kaybının çok olması ve fazla yazılı işlem gerektirdiğinden olabilir.



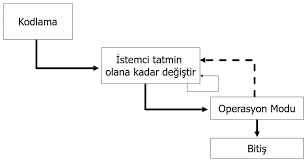
Artımlı Geliştirme Modeli:

Artımlı Geliştirme modeli günümüzde kullanılan çevik modellere benzeyip projeyi parçalara ayrılan bir modeldir. Bu modelde amaç gereksinimleri önceliklerine ve birbirlerine bağlılıklarına göre parçalarına ayırarak müşterinin ürüne en hızlı elden ulaşmasını sağlamaktır. İlk başta en öncelikli gereksinimler karşılığında üretim başlar ve bitene kadar müşteriyle temasa geçilmez. Ürün ortaya çıktıktan sonra müşterinin fikirleri alınır ve ona göre ilk çıkanın üstüne yeni bir versiyon çıkartılır. Bu olay istenilen yazılım ürünü elde edilene kadar devam eder. Model de göze çarpan özellik bir yandan üretim devam ederken bir yandan önceki sürümler kullanılmaktadır. Artımlı Geliştirme modeli daha çok işlevleri birbirinden ayrı çalışabilecek ve çıkması uzun sürebilecek olan işlerde kullanılır. Yazılımın başarısız olma olasılığı azdır ve ara ürünler olarak bir ilerleyiş söz konusudur. Sıkıntıları ise söylediğimiz gibi başka bir sürüm çıkana kadar olan sürümde bir değişiklik yapılamaz ve parçaları birleştirebilmek için bu yapıyı iyi bir şekilde tanımlanması gereklidir.



Kodla ve Düzelt:

Genelde bu model daha yetişme çağındaki öğrencilerin kullandığı direk olarak ürünün çıktığı bakımının zor ama az olan bir modeldir. Büyük çaplı işlerde kullanılmaz. Ürün yapılır ve kullanılır, isteneni gerçekleştirdikten sonra emeklilik safhası başlar.



XP (Extreme Programming):

1996 yılında Kent BECK ve arkadaşları tarafından yaratılmıştır. Ana 4 maddeden oluşmaktadır, bunlar; Basitlik, Cesaret, Dönüş, İletişim

Basitlik: Yapılan yazılımın karmaşık ve kompleks yapıda olmasındansa daha anlaşılabilir, sade ve kısa şekilde yazılmasını temel almayı anlatır. Uzun satırlar boyunca dokümantasyondan uzak durularak verimliliğin artırılması planlanmıştır.

Cesaret: Yapılacak işten korkmadan hata yapılabileceğini aklında tutup ilerlenmesini anlatır. Gerekli durumlarda en baştan başlamaktan bile korkulmamalıdır.

Dönüş: Müşteriye olan geri dönüşlerle hatalar en aza indirgenerek en iyi kodu çıkarmaya çalışılmıştır. Müşterinin memnuniyeti sağlanmaya çalışılmıştır.

İletişim: Pek çok modelde sıkıntı çıkaran konu iletişim bu modelde ana yapılardan biridir ve ekip içi iletişimin önem verilir. Bu şekilde birlikte çalışma daha verimli ve hızlı olur.

Scrum:

Scrum, Artımlı Geliştirme modelindeki gibi tüm planı parçalara ayırıp bu parçaları ani koşmalar gibi bir şekilde bitirilmeye çalışılır. Ekip içi iletişimlerin çok önemli olduğu bu modelde günlük yapılan toplantılar olur. Scrum 3 ana temelden kavramlar vardır; Roller, Toplantılar ve Bileşenler.

Roller: Scrum yöneticisi, Ürün Sahibi ve Scrum Takımı olarak 3 rol vardır. Ürün Sahibi projenin iş değeri açısından geri dönüşü ile sorumludur. Scrum yöneticisi günlük toplantıların başı olup o parçada neler yapılacağını anlatıp takımı birleştirmekle görevlidir. Yöneticinin diğer kimseden bir üstünlüğü yoktur, amacı takımdaki herkesi olaya dahil edip hataların çözülmesine yollar bulmaktır. Scrum takımı ise 8-9 kişiden oluşan sürekli bir iletişim halinde olan ve tek bir amaçları doğrultusunda ilerleyen gruptur.

Toplantılar: Günlük yapılan toplantılarda ilk başta önceki günün hakkında konuşulur ve hatalar, veriler ve çıktılar üzerinde tartışılır. Eğer önceki günden kalma bir problem varsa onun çözümü için birlikte hareket edilir. Bu yüzden takım çalışmasını güçlendirmek için günlük olarak 10-15 dakika arasında süren scrum yöneticisinin başta olduğu konuşmalar önemlidir. Bu sayede verimlilik artar ve hatalar en aza indirgenir. Önceki gün hakkında konuşmalar bittikten sonra o gün ne yapacakları hakkında konuşarak iş dağılımı yapılır. Scrum diğer modellerden en büyük özellik toplantılarıdır.

Bileşenler: Yazılım gereksinim dokümanı yapılır ve bu doküman projede yapılacakları basit bir şekilde gösterir. Sprint Dokümanı yapılır ve sprint dokümanı projede yapılanların sırayla tutar. Amacı Her sprintin ona uygun hazırlanmasıdır. Sprint zaman grafiği ise işin ne kadar sürede nasıl geliştiğini ve ne kadar süre kaldığını yazar. Bu sayede takımdakiler kendilerini ona göre planlayıp düzenli bir şekilde devam edebilirler.

Scrum günümüzün en popüler yazılım yaşam döngüsü modelidir. Bunun sebebi olabildiğince hızlı ve pahalı olmaması, Gündemi takibinin kolaylığı, Zor ve içinden çıkılmaz işleri basitleştirebilmesi, takım çalışmasının çok önemli olması yüzünden verimliliğin artması ve hata payının azalması, müşteriyle devamlı bir iletişim halinde olunduğundan, Değişen istek ve ihtiyaçlara çabuk adapte olabilmesinden ve böl ve fethet yaklaşımını içermesidir.

Kaynakça:

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

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

<http://furkanalniak.com/yazilim-muhendisligi-yazilim-surec-modelleri/>

<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://www.codex.com.tr/yazilim-gelistirme-modelleri>

Bakırçay Üniversitesi BİL102 hafta 2 ve 3 Sunumları

Kayra AKBAYRAMLAR

190601010

Bakırçay üniversitesi

Bilgisayar Mühendisliği