YAZILIM YAŞAM DÖNGÜSÜ NEDİR?

- Bir yazılımın geliştirilmesi ve bakımı süresince icra edilen adımlar topluluğuna yazılım yaşam döngüsü denir.
- Yazılımın yaşam döngüsü tek yönlü veya doğrusal değildir ve birkaç temel aşamadan oluşur. Bu aşamalar şunlardır: Planlama, Analiz, Tasarım, Gerçekleştirme ve Bakım.
- Planlama: Yazılım projesinin planlaması ve görev dağılımı yapılır.
- Analiz: Yazılım projesinin ne kadar süreceği ne gibi risklerinin olacağı belirlenir.
- Tasarım: Belirlenen gereksinimlere yanıt verecek yazılım sisteminin temel yapısının oluşturulduğu aşamadır.
- Gerçekleştirme: Yazılım projesinin kodlama ve test kısmıdır.
- Bakım: Yazılım projesinin ürün olarak sunulduktan sonra güncelleme, bakım olaylarının yapılmasıdır.

YAZILIM YAŞAM DÖNGÜ MODELLERİ

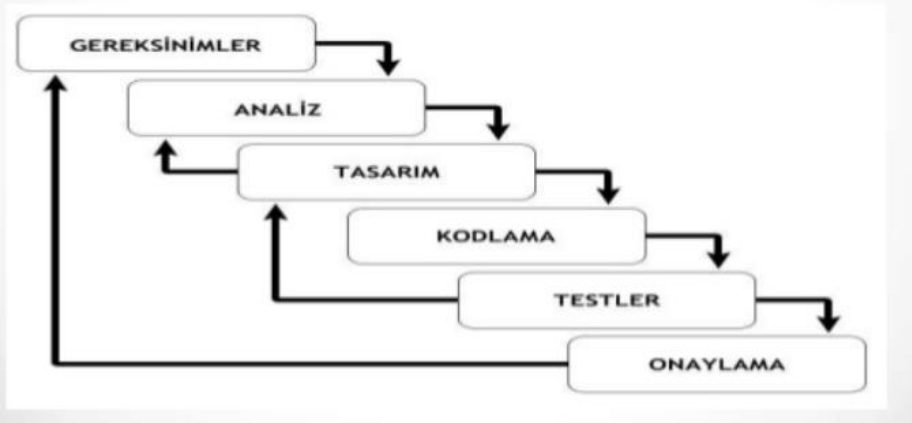
1.GELİŞİGÜZEL MODEL

- Herhangi bir model veya yöntem gözetmeksizin yapılır. Takip edilebilirliği ve bakımı oldukça zordur. Genellikle kişiye bağlı yazılım geliştirme şeklinde yapılır. Basit programlamaya sahiptir ve çoğunlukla tek bir kişinin üretim yaptığı yöntemdir.

2.BAROK MODELİ

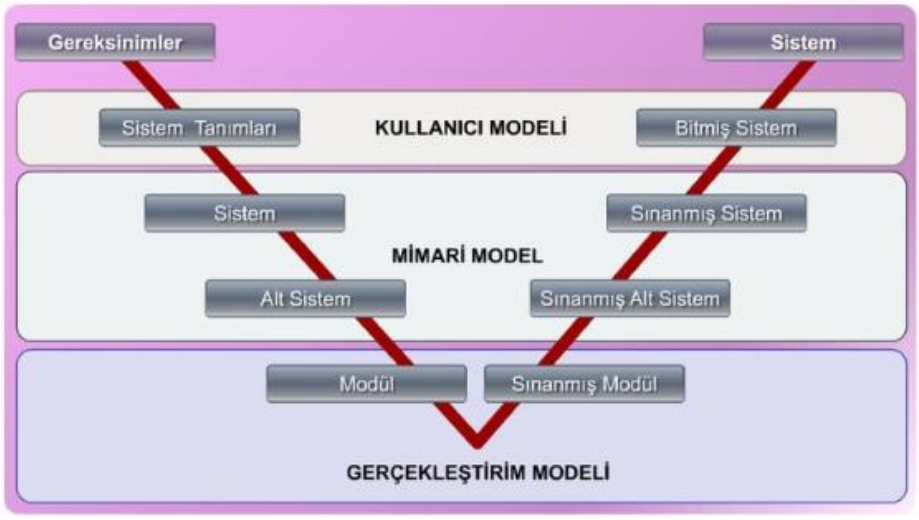
- Barok modelinde, yazılım yaşam döngüsü başlıca adımlarının doğrusal bir biçimde geliştirildiği modeldir.
- Bu modelde, dokümantasyon günümüz modellerinden farklı olarak ayrı bir süreç olarak ele alır. Yazılımın test ve geliştirme faaliyetleri tamamlandıktan sonra yapılmasını öngörür. Adımlar arası geri dönüşlerin ne türlü olacağı tanımlanmamıştır.
- Bu model gerçekleştirim evresine daha çok ağırlık veren bir model olduğundan günümüz yazılım geliştirme projelerinde uygulanan bir model olmaktan çıkmıştır.

3.ÇAĞLAYAN YAŞAM-DÖNGÜ MODELİ



- Çağlayan modeli geçmişte en popüler yazılım geliştirme modeli olarak görülmüştür. En eski, en tanınmış ve en temel modeldir.
- Yaşam döngüsü temel adımları baştan sona en az birer kez tekrarlanması ile gerçekleştirir.
- Bir sonraki aşama, bir önceki aşama tamamlanmadan başlayamaz. Her safhada dokümantasyon yazılmalıdır. Eğer bir safhada dokümantasyon ve test olmamışsa, o safhanın tamamlandığı kabul edilemez.
- Yazılım üretim ekipleri bir an öce program yazma, çalıştırma ve sonucu görme eğiliminde olduklarından, bu model ile yapılan üretimlerde ekip mutsuzlaşmaktadır.
- Kullanıcı sürecin içinde yer almaz ve bu durum yazılım tamamlandıktan sonra geri dönüşleri arttırabilir. Bu geri dönüşler, yazılım geliştirme maliyetini büyük oranda yükselten bir durumdur.
- Çağlayan yaşam-döngü modeli, Barok modelinden farklı olarak proje içerisindeki dokümantasyonu ayrı bir süreç olarak değil, üretimin doğal bir parçası olarak ele alır.

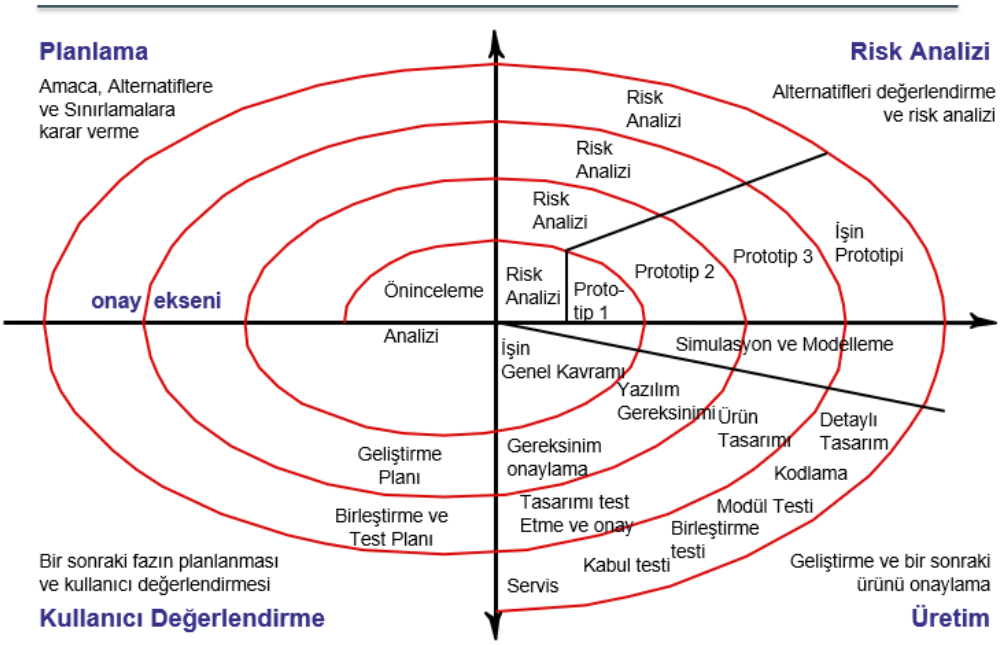
4. V SÜREÇ MODELİ



- Sol taraf üretim, sağ taraf ise sınama bölümüdür. Şelale modelinin biraz daha gelişmiş halidir. Bu model kullanıcının projeye katkısını arttırır. Bu döngü esas olarak 3 modelden oluşmaktadır: Kullanıcı Modeli, Mimari Model ve Gerçekleştirim Modeli.

- Kullanıcı modelinde kullanıcının istekleri ve proje için tanımları kullanılır ve tamamlanmış hali teslim edilir.
- Mimari modelde sistem tasarımı ve oluşacak alt sistem ile tüm sistemin sınama işlemlerine ilişkin işlevlerden oluşur.
- Gerçekleştirim modeli ise kodlama ve bunların denenmesinden oluşur.

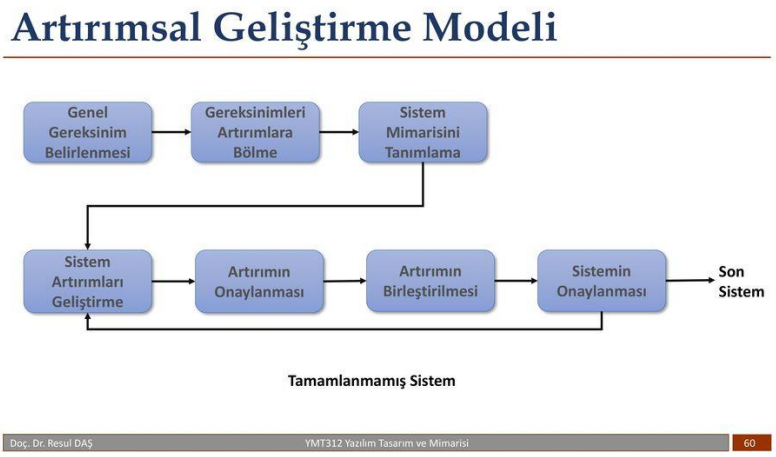
5.HELEZONİK(SPİRAL)MODEL



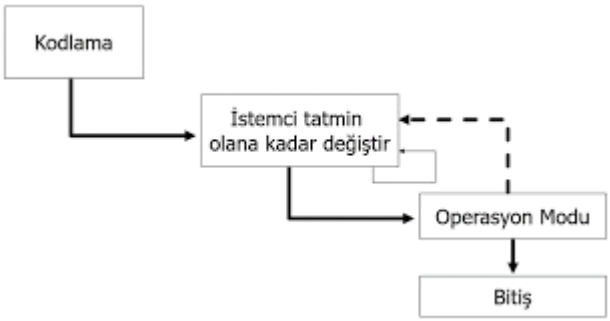
- Helezonik modelde risk analizi ön planda olduğu için hataları erken giderme imkânı tanıyabilir. Prototip oluşturma da her aşamada olduğu için kullanıcı da her aşamada yazılım projesinin bir parçasını görme imkanına sahip olur bu da sorunların azalmasını sağlayabilir. Helezonik modelin sorunları; küçük ve düşük riskli projeler için çok pahalı bir sistem olması, karmaşık bir içeriğe sahip olması, uzun sürmesi ve fazla dokümantasyondan oluşması olarak sıralanabilir. 4 temel aşamadan oluşur:
- Planlama: Her aşamada olan ara ürün için bir planlama yapılır.
- Risk Analizi: Risklerin araştırılması, belirlenmesi ve çözülmesi.
- Üretim: Ara ürünün üretilmesi.
- Kullanıcı Değerlendirmesi: Oluşturulan ara ürün sonucunda kullanıcıdan alınan geri dönütlerin değerlendirilerek diğer aşamaya geçilmesi.

6.ARTIMSAL GELİŞTİRME SÜREÇ MODELİ

- Artımlı Geliştirme Modelinde proje parçalara bölünür ve kullanıcının önceliğine göre bu parçalar sıralanır. Sıralanan bu parçalar bittiğinde sırasıyla birer ara ürün geliştirilmiş olur ve bu ara ürünler de kullanıcı tarafından kullanılır. Ara ürünler her seferinde bir öncekinin üstüne bir şeyler katarak çıkartılır. Yani bu modelde bir taraftan üretim kısmı sürerken diğer tarafta ise kullanım kısmı sürer.
- Artımlı geliştirme modeli uzun zaman alabilecek ve ürünün eksik işlevsellikle çalışabileceği türdeki yazılımlar için uygundur. Bu model ile sistemin başarısız olma olasılığı azalır, ara ürünler yazılımın geliştirilmesinde önemli bir yere sahip olur.
- Bu modelde bakım safhası vardır fakat çok zordur çünkü dokümantasyon yoktur.



7.KODLA VE DÜZELT YAŞAM-DÖNGÜ MODELİ



- Küçük programlar için kullanılabilir, direkt ürün gerçekleştirilir ve emeklilik safhası vardır.
- Bakım safhası vardır fakat çok zordur. Çünkü sisteme ait dokümantasyon yoktur.
- Yazılım geliştirmenin en kolay yoludur. Ancak en pahalısıdır.

HANGİ PROJEDE HANGİ MODELİ KULLANILMALIYIZ?

- Belirsizliklerin az olduğu, iş tanımlarının belirgin olduğu bilgi teknolojileri projeleri için V Süreç Modeli uygun bir modeldir.
- Büyük, maliyetli ve uzun süren projelerde spiral model veya Artımsal Geliştirme Modeli uygundur.
- Orta ve küçük büyüklükte, uzun sürmeyen projelerde çevik modeller uygundur.
- Kişiye özel, zaman sorunu olmayan, küçük programlarda Kodla ve Düzelt kullanılabilir.
- Küçük ve özellikleri iyi tanımlanmış projelerde Çağlayan Yaşam Döngü Modeli kullanılabilir.

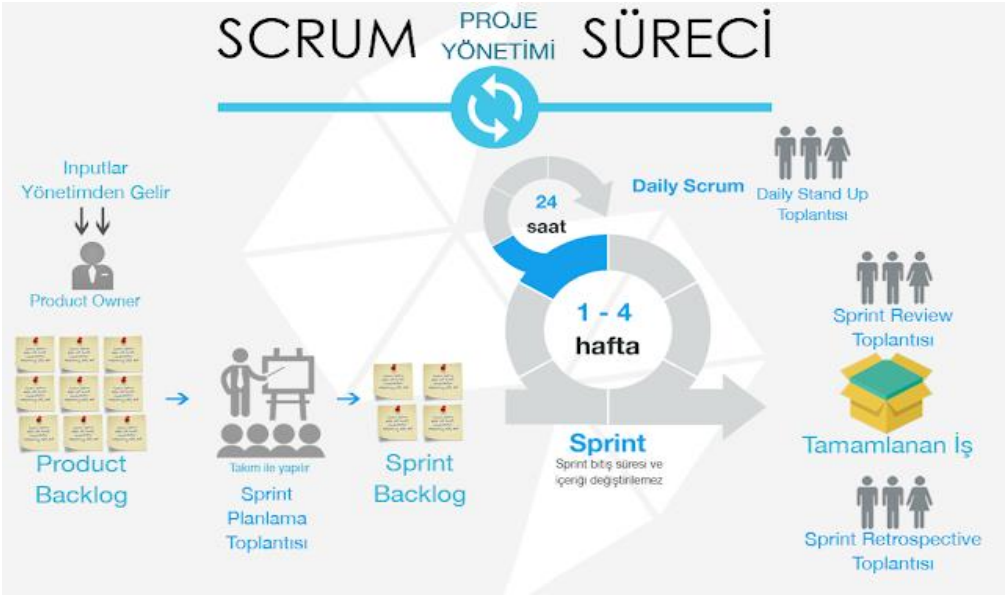
ÇEVİK MODELLER

- Çevik yazılım geliştirme metotları, verimliliği yüksek, esnek, hata oranı düşük, hızlı ve ucuz çözümler sağlamaktadır.
- Bu metodolojide projenin ölçeği ne olursa olsun, proje küçük yinelemelere ayrılır ve her yineleme başlı başına bir proje gibi ele alınarak geliştirilir.
- Ayrıca projenin küçük parçalardan oluşması da geriye dönük hataların düzeltilmesini kolaylaştırır.
- Bazı çevik yazılım geliştirme metotları şunlardır:

EXTREME PROGRAMMİNG (XP)

- XP kolay, grup içi iletişime önem veren, geri dönüşlerin daha fazla olmasına imkan sağlayan bir yazılım geliştirme yöntemidir.
- Extreme programming'in dört temel değeri vardır: İletişim, Basitlik, Geri Bildirim, Cesaret.
- İletişim: İletişim, projelerde önemli sorunlardan birisidir. XP ise bunu aşmaya çalışmaktadır. Ekip içi iletişime önem verir ve artırılması için çalışır.
- Basitlik: Yazılan kodun ve yapılan işin sade, anlaşılır ve karmaşık olmadan yapılmasını gerektirir. Uzun uzun dokümantasyondan uzak durulur.
- Geri Bildirim: Geri bildirimler ile oluşabilecek hatalar azaltılır/ortadan kaldırılır. Müşteri ile yazılım ekibi birbirleriyle iletişim halindedir.
- Cesaret: Yapılan işte cesur olunmalıdır, projelerin üstüne korkmadan ilerlenmelidir. Bir kodun gerekirse tamamen silinip yeniden yazılması sağlanmalıdır.

SCRUM



- Scrum, karmaşık yazılım işlerini küçük birimlere(sprint)bölerek geliştirmeyi öngörür.
- Bu metodoloji, karmaşık ortamlarda adım adım yazılım geliştiren ekipler için uygundur.
- Scrum'da üç temel kavram vardır. Roller, Toplantılar ve Bileşenler/Araçlar.
- Roller: Ürün sahibi, scrum yöneticisi ve scrum takımından oluşur. Ürün sahibi projenin iş değeri açısından geri dönüşü ile sorumludur. Scrum yöneticisi, takımı scrum etiklerine göre kontrol eder. Scrum takımı ise 5–9 kişiden oluşan birbirleriyle sürekli iletişim halinde olan bir yazılım geliştirme ekibidir.
- Toplantılar: Scrum'ın olmazsa olmazı toplantılardır. Her gün scrum toplantıları yapılır. Bu günlük toplantılarda her gün yazılım geliştiricilerinin önceki gün neler yaptıkları, karşılaştıkları sorunlar ve bugün neler yapacakları hakkında bir konuşma olur. Her sprint için de bir gözden geçirme toplantısı yapılır.
- Bileşenler/Araçlar: Ürün gereksinim dokümanı oluşturulur. Bu dokümanda proje boyunca yapılması gerekenler basitçe yazılıdır. Sprint dokümanı oluşturulur. Sprint dokümanı ürün gereksinim dokümanına takiben oluşturulur ve amacı her sprintin ona uygun ayarlanmasıdır ayrıca bu dokümanı sadece takımdakiler değiştirebilir. Sprint kalan zaman grafiği ise yapılan işin ne seviyede olduğu ve aslında planlanan zaman göre nerede olduğunu belirlemek için hazırlanır.
- Scrum günümüzde en çok kullanılan yazılım geliştirme yöntemidir. Hatta sadece yazılım geliştirmede değil birçok sistemin geliştirilmesinde de kullanılır. Bunun nedenleri ise; zamandan ve paradan büyük

ölçüde tasarruf edilmesi, yüksek teknolojiler ve son gelişmelere kolaylıkla uyum sağlayabilmesi, karmaşık görülen ve gereksinimleri tam belirlenmemiş projeler için ideal olması, ekip içi iletişimin yüksek tutulması ve bununla beraber hataların erken fark edilip düzeltilmesi, kullanıcıdan sürekli geri bildirim gerektirmesi ve bununla beraber sorunların azalması, diğer yazılım geliştirme metodolojileri gibi yinelemeli olması, değişen gereksinimlere hızlı bir şekilde tepki vermesi gibi nedenler örnek olarak verilebilir.

KAYNAKLAR

https://slideplayer.biz.tr/slide/12386328/

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

https://medium.com/@HayriRizaCimen/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-ve-s%C3%BCre%C3%A7-modelleri-70fdfb2f8f77

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

Sena KOÇAK

190601045

İzmir Bakırçay Üniversitesi

Bilgisayar Mühendisliği