YAZILIM YAŞAM DÖNGÜSÜ TEMEL AŞAMALARI (SDLC)

Biz canlıların nasıl kendi hayatlarımızda bir döngüsü var ise yazılımın da kendine ait bir yaşam döngüsü vardır. Bir spiral yaklaşımdır. Yedi aşamayı temel alır. Bunlar : Planlama, Tanımlama, Tasarım, Geliştirme, Entegrasyon ve Testler, Uygulama, Bakım ve Değerlendirme.

Okunan ya da araştırılan kaynağa göre bu yedi aşama arttırılıp azaltılabilir. Her sistemi geliştirmede kullanılabilir.

Planlama: Proje yönetim aşamalarının en önemlilerinden biridir. İşin projelendirildiği, fikrin ortaya çıkarıldığı, tartışıldığı aşamadır.

Tanımlama: İstenilen işin ne olduğu temel tanımlar üzerinde konuşulacak kavramların tanımlandığı, dökümante edildiği, aşamadır. Ya da neler istendiğiyle ilgili analizlerin yapılması da bu aşamada gerçekleşir. Bu aşamanın önemi eğer projenin ileriki aşamalarında aynı şeyden bahsedilmiyorsa farklı sonuçlara varılabilir bu da yapılan işin istenilenden farklı sonuçlar doğurmasına yol açar.

Tasarım: Tanımlama aşamasından sonra yazılımın ilk ve temel halinin oluşturulduğu aşamadır. Yazılımın, sistemin tasarımı bu aşamada yapılıyor. Yazılımın projesi çiziliyor.

Geliştirme: Tasarımı bitmiş olan sistemin kodlama aşamasıdır. Sistemin yaşatılmaya başlandığı ilk örneklerinin ortaya çıkmaya başladığı aşamadır.

Entegrasyon ve Testler: Yazılan kod hayata geçirilir. Örnek olarak: Gerekli olan donanımların temin edilmesi ve bağlantılı oldukları birimlerin buna entegre edilmesi, veri tabanı bağlantılarının ayarlanması gibi örnekler bu aşama için verilebilir.

Uygulama: Artık yaşayan bir proje haline gelip, kullanılıyor.

Bakım ve Değerlendirme: Yazılan yazılımda problemler, yeni fikirler, ya da yeni ihtiyaçlara gerek duyuluyor. Gerek duyulan konu bakımından değerlendirmeye alınıyor ve yazılımın bakımı yapılıyor.

Bazı konularda yazılan yazılım hakkında sorunlar ile karşılaşılırsa ya bu döngü döndükten sonra o sorunlarla ilgilenilecek ya da proje tamamen o anda iptal edilecek. Ama döngünün döndükten sonra problemlerle ilgilenmek daha sağlıklıdır.

YAZILIM YAŞAM DÖGÜSÜ

Yazılım Yaşam Döngüsü Nedir? : Herhangi bir yazılımın, üretim aşaması ve kullanım aşaması birlikte olmak üzere geçirdiği tüm aşamalar biçiminde açıklanır.

Yazılım Yaşam Döngüsü Modelleri Neden Önemlidir ?:

Endüstri yazılım ürününden performans ve üretkenlik istemektedir. Yazılım ürünlerinin istenen performans ve üretkenliğe sahip olması için yazılım yaşam döngüsü modelleri sayesinde kontrol ederek daha iyi verim sağlarız ve isteneni oluştururuz.

Yazılım geliştiricilerinin, yazılım geliştirme sürecinin karışıklığı ile baş etmelerine kolaylık sağlar.

Yazılım Yaşam Döngüsü Modelleri Nelerdir ?:

\*Kodla ve Düzelt

\*Şelale Modeli

\* V modeli

\*Evrimsel Geliştirme

\*Prototipleme

\*Spiral Model

\*Formal Sistem Geliştirme

\*Yeniden Kullanıma Yönelik Geliştirme

\*Artımlı Geliştirme

\*Birleşik Süreç

\*Çevik Modeller

Kodla ve Düzelt: Bu model ürünü müşteriye teslim edene kadar kodlama yapılarak devam eder.

Çok küçük projelerde ya da kısa ömürlü yazılımlarda kullanılabilir.

Hata bulmak zordur.

Daha sonra yazılan kodları değiştirmek üzerinde değişiklikler yapmak zordur.

Şelale Modeli: Aşamaları planlama-analiz-tasarım-uygulama-testler olacak şekildedir.

Yeni özelliklerin eklenmediği projelerde kullanılabilir. Bakım aşaması yoktur.

Yeni bir özellik eklemek için planlamadan başlamak gerekir.

Küçük projeler için uygulanabilir. Bilişim projeleri için uygun bir yönetim sistemi değildir.

Anlaşılır adımlardan oluşur. İşin dağılımı yapma açısından kolaydır.

V Modeli: planlama Bakım

İhtiyaç Kabul Testleri

Üst seviye tasarım Entegrasyon Testleri

Detaylı tasarım Birim Testler

Kodlama

Şelale modelinin daha organize edilmiş halidir. Her aşama kendi kontrol aşamasıyla eşleştirilmiştir bu yüzden V şeklini almıştır. Her aşamanın kendi testiyle eşleşmesi gerçekten önemli bir özelliktir.

Yapılanların kontrolü yani test kısmı belirli bir düzen içinde ilerler. Üretim, işletim ve sürdürülebilirlik vardır.

Üst Seviye Tasarım: Genel bakışa göre tasarımdır.

Detaylı Tasarım: Uygulamada girdilerin, çıktıların yazıldığı daha detaylı tasarımdır.

Birim Testler: Her üretilen modülün test edildiği aşamadır.

Entegrasyon Testleri: Her alt modülün birbirleri ile uyumlu çalışıp çalışmama durumlarının kontrol edildiği aşamadır.

Kabul Testleri: Müşteriye uygulamayı teslim ettikten sonra müşterinin uygulamayı test etme sürecidir.

Teslim edilebilir ürünlerde uygulanabilir. Belirsizliklerin az olduğu projelerde de kullanılabilir. Kullanımı kolaydır.

Aşamalar arasında tekrarlamalar yoktur. Risk çözümlerine imkan tanımaz.

Proje yönetimi tarafından kolay takip edilebilir yapıdadır.

Evrimsel Geliştirme: İlk tam ölçekli model olarak bilinir. Müşterinin geri bildirimi sayesinde sistem yavaş yavaş geliştirilebilir. Çok birimli organizasyonlar için önerilebilir bir modeldir.

Genel tanımlama, tanımlama, geliştirme, test etme, ilk sürüm, ara sürümler, son sürüm olarak ayrılır.

Keşifçi geliştirme ve Atılacak prototipleme olmak üzere iki tane evrimsel geliştirme modeli vardır.

Keşifçi geliştirmenin hedefi müşterinin gereksinimlerini incelemek ve müşteri ile çalışıp son sistemi müşteriye teslim etmek. Atılacak prototiplemenin hedefi ise sistem gereksinimlerini anlamaktır.

Değişiklik yapısına sahip değildir. Bakımı zordur. Yazılım gereksinimini yenilemek gerekir.

Kısa süreli kullanılacak sistemlerde kullanılabilir. Ara yüzleri geliştirmek için de kullanılabilir.

Prototipleme: Bir ürünün hizmetin ya da servisin işler taklidi yapan bir versiyonudur. Bir tasarımın ilk gerçekleştiği andır. Girişimin ilk anından itibaren kullanılabilecek yöntemlerden biridir.

En düşük maliyetler ile yeni fikirlerin test edilmek istendiği tüm aşamalarda rahatlıkla kullanılabilir. Ürün hayata geçtikten sonra yeni özellikler ekleneceği zaman da prototipler kullanılabilir. Prototipin kolayca yaratılabildiği alanlar aslında fikirlerin önünü açan platformlar olmasıdır.

Karmaşa ve yanlış anlaşılmaları engeller. Risk kontrolü sağlar. Düzeltme aşaması atlanır ise düşük performansa yol açar.

Spiral Model: Risk analizi olgusu ön plandadır. Alternatifler değerlendikten sonra riskler belirlenip çözülür. Prototip yaklaşımı vardır.

Dört aşamaya bölünür:

1)Hedeflerin Belirlenmesi

2)Risklerin Belirlenmesi ve Çözümü: Risk analizinden sonra bir prototip çıkarılıyor. Bu prototip müşteriye sunuluyor. Daha sonra projenin durumuna göre birkaç tane daha prototip çıkarılabilir. Uygulamaya geçilmeden son bir prototip daha çıkarılır. Prototipin amacı yazılım ekibinin ve müşterinin projede mutabık olmasını sağlamaktır.

3)Geliştirme ve Testler: Detaylı Tasarım, Kodlama, Entegrasyon, Test, Uygulama aşamaları bu safhada olur.

4) Sonraki Dönüşün Planlanması

Kısaca spiral modelin genel amacı hataları erken gidermeye odaklanmasıdır. Karmaşık ve uzun projeler için kullanılabilir. Küçük projeler için riskli bir modeldir. Çok fazla dokümantasyon gerektirir bu yüzden de karmaşık bir yapısı vardır.

Formal Sistem Geliştirme: Cleanroom yazılım geliştirme yapısına sahiptir. Hata ayıklama işlemini engellemek için kodu yazmaya başlarken doğru yazmak ve test aşamasında doğruluğundan emin olmak gerekir. Doğruluğun ve kesinliğin vurgulanmasını temel alan bir modeldir.

Sistemin bazı taraflarını formal olarak belirtmek zordur. Sistem kullanılmaya başlamadan önce güvenlik ve gizlilik durumlarının kontrolünün sağlanması kritik bir noktadır.

Yeniden Kullanıma Yönelik Geliştirme: Organizasyon tarafından daha önce hazırlanmış veya dışarıdan temin edilmiş yazılımların kullanılması ile geliştirme yapılması son yıllarda popülaritesi artan bir yaklaşımdır.

Maliyet denetimi yapmak mümkündür. Önceden oluşturulmuş sınıflar tekrardan kullanılabilir.

Uygulanması maliyetlidir. Gereksinimleri anlamak güç olduğu için uzmanlık gerektirir.

Artımlı Geliştirme: Çağlayan modeli ve Evrimsel geliştirme arası bir modüldür. Üretilen her yazılım sürümü birbirini kapsayacak ve giderek artan sayıda işlev içerecek şekilde gerçekleştirilir. Eğer müşteri proje de bir değişikliğe ya da bir yeniliğe ihtiyaç duyuyor ise artımlı geliştirme modeli buna ayak uydurur. Yazılımı kısımlara bölerek geliştirip teslim etme amacı vardır. Bu modelde bir taraftan üretim diğer taraftan ise kullanıl yapılır.

Uzun bir süre vakit alacak projeler bu modele uygun olabilir. Gereksinimler müşteriler ile belirlenir. Gereksinimlerin önemine göre teslim edilecek artımlar belirlenir. En önemli sistem özellikleri daha fazla test edilme imkanına sahip olmuş olur. Böl ve yönet yaklaşımına sahiptir.

Gereksinimleri doğru boyuttaki artırımlara atamak zor olabildiği için deneyimli personellere gerek duyulur. Artırımlar kendi içerilerinde tekrarlara izin vermezler.

Birleşik Süreç: Nesneye dayalı yazılım geliştirmek için var olan yöntemlerin deneyimler sonucu kabul gören en iyi özellikleri bir araya getirilerek tümleştirilmiş yazılım geliştirme süreci oluşturulmuştur.

Birleşik süreçte yazılım geliştirme aşamaları dörde ayrılır:

Başlangıç: Vizyon kararı, fizibilite çalışması, tamam ya da devam kararı bu aşamada gerçekleşir.

Ayrıntılandırma: Yüksek riskli kısımların yinemeli olarak oluşturulması gerçekleşir.

Tamamlama: Daha az riskli ve düşük öncelikli kısımların yinemeli olarak gerçekleştirilmesi gerçekleşir.

Yayım: Beta testleri, piyasaya sürme aşamaları gerçekleşir.

Değişen isteklere uygundur. Büyük sistemlerde uygulanabilirlik vardır. Erken ürün elde edilmesi ve risklerin erken giderilmesi sayesinde takıma moral olur.

Dokümantasyon yükü ağırdır bu yüzden maddi açıdan fazladır. Karmaşık bir yapıya sahiptir.

Çevik Modeller: Yazılım sistemlerini etkili ve verimli modellemeye ve dokümantasyonunu yapmaya yönelik pratiğe dayalı yöntemlere verilen addır. Çevik modellemenin başlıca özelliği veri modelleri ve ara yüz modelleri gibi bu tekniklerin nasıl uygulanması gerektiğini söyler. Müşterilerin ihtiyaçlarının karşılanmasında ve aynı zamanda da her türlü değişikliğe kolayca adapte olabilmesinde geliştiricilere yol gösterir.

Değişime açıklık ve esneklik üst düzeydedir. Proje planlama ve yürütme bir arada bulunur. Öğrenim gerektirmez uyum çok hızlı bir şekilde gerçekleşir. Müşteri ihtiyaçlarına odaklanır.

Sürekli değişen ihtiyaçlar yüzünden aşırı çalışma gerektirir. Kurumsal bir yapıda uygulaması zordur.

SCRUM Nedir ?:

Scrum, yazılım mühendisliğinde bir uygulama geliştirme yöntemidir. Çevik proje yönetim metodolojilerinden biridir. Günümüzde Scrum’a projelerin başarıyla sonuçlanmasını sağlamak ve risk faktörlerini ortadan kaldırmak için yoğun bir ilgi bulunmaktadır. Scrum aslında mümkün olan en yüksek değere sahip ürünleri üretken ve yaratıcı bir şekilde üretirken, geliştirirken karmaşık problemlerinde adreslenebildiği bir çerçevedir.

Basittir. Anlaşılması kolaydır. Ancak üzerinde ustaca yönetilebilmesi ve ustalaşılabilmesi zordur.

Bir ürün geliştirme tekniği ya da yöntemi değildir. Daha çok farklı tekniklerin ve yöntemlerin içerisinde kullanılabildiği bir çerçevedir.

Ürünlerde iyileştirme için fırsat yaratır.

SCRUM da hedeflenenler:

Bir ürünü daha az maliyet ile üretmek. Daha az insan ile projeyi bitirmek. Daha az zamanda yapılması gerekenleri bitirmek. Daha kaliteli ürünler koymak. Scrum’ın amaçladıkları bunlardır.

SCRUM bize ne sağlar ?

Scrum’ın getirdiği en büyük yeniliklerden birisi, proje süresince gereksinimlerde ihtiyaç duyulan değişikliklerin kolayca yapılabilmesidir. Bu nedenle Scrum, ürün geliştirme süreçlerimizi yönetmek, müşterilerimizi iş ortağımız gibi görerek isteklerine daha hızlı cevap verebilmek, esnek, kalitesi yüksek yazılımlar üretebilmek için popüler bir yazılım geliştirme yönetimsel altyapısıdır.

REFERANSLAR

https://e-bergi.com/y/cevik-modelleme-ve-cevik-yazilim-gelistirme/

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

https://www.haberler.com/scrum-ile-cevik-proje-yonetimi-9577509-haberi/

http://web.firat.edu.tr/mbaykara/ytm2.pdf

https://www.youtube.com/watch?v=u2rU8Wss4bw

https://www.youtube.com/watch?v=6Ao75g13wjY&t=264s

https://www.youtube.com/watch?v=p-YfduOnPIE&t=184s

https://www.youtube.com/watch?v=9o0c34VL5Nw

https://www.youtube.com/watch?v=2f5d0TdSmeM

https://www.youtube.com/watch?v=qJH6Gou31-U

<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>

Esra ÇAĞLAK 190601072