YAZILIM YAŞAM DÖNGÜSÜ

GİRİŞ

1.Yazılım Yaşam Döngüsü Nedir, Ne İşe Yarar:

Geliştirilen bir yazılım projesinin planlanmasından teslim aşamasına kadar içinde bulunduğu sürecin tamamından oluşan döngüye, Yazılım Yaşam Döngüsü denir. Yazılım da bir üründür. Ve bu ürünün üretim sürecinde bazı adımları takip eder. Yazılım ürününün gereksinimleri zamanla değişiklik gösterebilir. Bu değişikliklere uyum sağlayabilmek için söz konusu aşamalar, belli bir döngüde tekrarlanırlar. Yazılım yaşam döngüsü tek yönlü veya doğrusal değildir. Döngü içerisinde herhangi bir aşamadan başka bir aşamaya geri dönülebilir veya ileri gidilebilir.

2.Yazılım Yaşam Döngüsü Aşamaları:

2.1. Planlama:

Bu aşama, üretilecek yazılım ile ilgili olarak donanım ve personel temel gereksinimlerinin belirlendiği, fizibilite(maliyetlerin ve sistemin yararlarından faydalanması) çalışmasının yapıldığı ve proje planlamasının oluşturulduğu aşamadır.

2.2. Çözümleme:

Yazılımın işlevleri ile gereksinimlerinin detaylı olarak çıkarıldığı aşamadır. Bu aşamada temel olarak mevcut sistemde var olan işler incelenir. Var olan temel sebepler belirlenerek yazılımın neyi nasıl çözümleyeceği saptanır. Bu aşamadaki asıl amaç, bir yazılım mühendisi gözğyle mevcut yapıdaki işlerin ortaya çıkarılması ve doğru bir şekilde algılanıp belirlenmesidir. Bu aşamada temel UML diyagramlarının çizimine başlanır (Use Case, Activity, Class Diagram vs.). Bu çalışma müşteri ,yazılım mühendisi, sistem analisti, iş analisti, ürün yöneticisi vb. Rollerin bir araya geldiği gruplar tarafından yapılabilir. İhtiyaçların net olmadığı durumlarda yazılım mühendisi ve müşteri arasında iletişim anlamında sorunlar yaşanma ihtimali çok yüksektir.

2.3. Tasarım:

Gereksinimlerin tamamlanmasından sonra tasarım aşamasına geçilir. Tasarım aşaması, çözümleme aşamasından sonra tespit edilen gereksinimleri karşılayacak bir yazılım veya bilgi sisteminin temel yapılarının oluşturulmaya çalışılma aşamasıdır. Yazılım ürün tasarımı, müşterinin isteklerini ve gereksinimlerini karşılamak üzere yazılımın özellikleri, yeteneklerinin belirlenme aşmasıdır. İki farklı türde tasarım aşaması vardır. Birincisi Mantıksal Tasarım'dır. Mantıksal tasarımda, mevcut sistemin değil önerilen sistemin yapısı anlatılır. Gerek duyulabilecek bazı değişiklikler önerilir. İkincisi ise Fiziksel Tasarım'dır. Fiziksel tasarımi yazılımı içeren bileşenler ve bunların ayrıntılarını içerir.

2.4. Gerçekleştirme:

Bu aşamada kendi içinde kodlama, test etme ve kurulum olmak üzere üç aşamaya ayrılır:

Öncelikle, bundan daha önceki aşamalarda yapılan hazırlıklar sonucu kodlama aşamasına geçilir. Müşteriye teslim edilecek ürünü programlama aşamasıdır. Daha sonra yazılan kod test edilir. Hata tespit edilirse bunların üzerine çalışmalar yapılır, Bu hem hata yapma oranını hem de projenin sebep olacağı maliyetleri düşürecektir. Birim testleri, duman testleri, yanlış değer testleri, kabul testleri, kullanım senaryo testleri, yük testleri, kullanıcı kabul testleri, yoldan geçen adam testi, test otomasyonu gibi sürece ve duruma göre uygulanabilecek çok farklı kategorilerde ve özelliklerde test türü de mevcuttur. Ve en son adım olarak ise kurulum gerçekleştirilir.

2.5. Bakım:

İşletime alınan yazılımla ilgili olarak hataları giderme ve bakım yapılma aşamasıdır. Tüm test aşamaları tamamlandıktan sonra yazılım ürününün sahaya teslim edilebilir bir versiyonu çıkarılır ve teslim aşaması gerçekleştirilir. Teslim için ürün tek başına yeterli değildir. Kullanıcılar için kullanım kılavuzu ve versiyon fark dökümanı oluşturulmalıdır. Teslim ile birlikte baakım aşaması da başlar. Hata giderici, önleyici, altyapı iyileştirici, ürüne yeni özellikler ekletici gibi farklı bakım çeşitleri bulunmaktadır. Bu aşama yazılımın tüm yaşamı boyunca sürmeye devam eder.

Yazılım Yaşam Döngüsünün temel adımları çekirdek olarak da adlandırılır.

Bu aşamaların gerçekleştirilmesi için yazılım belirtim yöntemleri ve yazılım süreç modelleri kullanılmaktadır.

3. YAZILIM BELİRTİM YÖNTEMLERİ

3.1 Süreç Akış İçin KUllanılan Belirtim Yöntemleri:

Süreci hızlandırmak ve kolaylaştırmak amacıyla kullanılırlar. (veri akış şemaları, yapısal şemalar ve nesne şemaları)

3.2. Süreç Tanımlama Yöntemleri:

Süreçlerin kendi içinde nasıl işlediğini belirten yöntemlerdir. (algoritma, düz metin, anlatım dili,karar tabloları, karar ağaçları).

3.3 Veri Tanımlama Yöntemleri:

Verilerin tanımlanması için süreçler tarafından kullanılan yöntemlerdir. (nesne ilişki modeli, veri tabanı tabloları ve veri sözlüğü).

4.YAZILIM SÜREÇ MODELLERİ (PROCESSES):

Süreç planların ve olguların belli bir taslağa uygun şekilde yapılıp düzenlenmesidir. Bir şeyin yapılışını, üretim şeklini oluşturan zaman dilimidir. Yazılım süreç modelleri, geliştirme

aşamasında planların hangi sırada ve nasıl uygulanacağını belirleyen modellerdir. Yanlış yapılma oranını düşürür. Ekarışıklığı azaltır, daha düzgün bir yapı oluşumunu ve kolaylığı sağlar. Ve daha verimli bir iş çıkarılmasını sağlar. En bilindik yazılım süreç modelleri şu şekildedir:

4.1. DÜZENLEYİCİ SÜREÇ MODELLERİ:

4.1.1ÇAĞLAYAN MODELİ:

Bu model geleneksel geliştirme modeli olarak da bilinir. Çağlayan modelinde yazılım geliştirilirken aşamalar tekrar tekrar geliştirilir. Üretimi çok vakit almayacak projeler Için uygun bir modeldir. Çünkü uzun süreli projelerde gereksinimlerin değişme ihtimali yüksektir. Bu modelin günümüzde kullanımı gittikçe azalmaktadır. Bu modelde uygulamalar sırayla ve aşama aşama yapılır. Başka bir aşamaya geçmek için önce diğeri bitirlmelidir. Her aşamada dökümantasyon yazılmalıdır. Her aşamada dökümantasyon test edilmelidir, test edilmeden geçilmemelidir. Bu modelin belli aşamaları vardır. Ve aşamalar sırasıyla şu şekildedir:

1. Kavram
2. Başlangıç
3. İhtiyaç Toplama ve Analiz
4. Tasarım
5. Uygulama/Kodlama
6. Test Etme
7. Bakım

Çağlayan Modelinin Avantajları:

- Gereksinimleri net olarak belli olan değişmeyen ve iyi anlaşılan projeler için faydalı ve mantıklı bir modeldir.
- Kullanımı kolaydır ve her proje için aynı adımları takip ettiği için anlaşılırdır.
- Projeyi yönetenler için yönetimi ve kontrolü yapma açısından daha kolay bir süreçtir.

Gereksinim aşaması tamamlandıktan sonra proje için sağlam bir temel atılmış olur.

Çağlayan Modelinin Dezavantajları:

- Aşamalar sırayla adım adım gerçekleştirildiği için projenin tasarımında vs herhangi bir değişiklik olması bu model için büyük bir dezavantajdır.
- Değişiklikler maliyetlidir.

4.2.2. KODLA VE DÜZELT:

Direkt olarak yazılım ürünü gerçekleştirilir. Bir yazılım geliştirmenin en kolay yollarından biridir. Daha çok yüzlerce satırdan oluşan programlarda kullanılırlar. Bu modelde dökümantasyon yoktur. Bakım safhası bulunmaktadır fakat çok zordur. Emeklilik safhası da bulunmaktadır.

Kodla ve Düzelt Modelinin Dezavantajları:

- Değişiklik yapmak zor ve maliyetlidir.
- Dökmantasyon yoktur, bakımı zordur.
- Bu modelde yazılım geliştirmek kolay olduğu için tecrübesiz firmaların çoğunda bu model kullanılır.

4.3.3. EVRİMSEL GELİŞTİRME MODELİ:

Coğrafi olarak geniş alanları kapsayan projelerde daha iyi etki gösterir. Tek ölçekli olan ilk modeldir. Her aşamada üretilen ürünler, tam işlevselliği içermektedir.

Evrimsel Geliştirme Modelinin Avantajları:

- Kullanıcılar kendi gereksinimlerini fark edip daha iyi anlarlar.
- Hataları azdır.

Evrimsel Geliştirme Modelinin Dezavantajları:

- Bakımının yapılması zordur.
- Yazılımın gereksinimlerinin yenilenmesi gerekebilir.
- Sürecin görünürlüğü azdır.

4.2.4. SPİRAL MODEL:

Bu modelde her döngü bir fazı temsil etmektedir. Prototip bir bakış açısıyla yaklaşmaktır. Risk analizi, bu modelde ön plandadır. Geliştirmeyi farklı parçalara ayırır ve risk analiziyle en riskli kısımlar daha önce geliştirilir. Risk analizi metodu kullanıldığı için zorlukları engelleme potansiyeli yüksektir. Spiral model, dört aşamadan oluşur, aşamalar şu şekildedir:

1.Plan

2.Risk analizi

3.Üretim

4.Kullanıcı Değerlendirmesi

Spiral Modelin Avantajları:

- Yazılımı gerçekleştirmek ve kontrolün edilmesi daha erken gerçekleştiği için hatalar daha erkenden giderilebilir. İçinde başka yazılım modellerini de bulundurur.

Spiral Modelin Dezavantajları:

- Risk analizi özneldir.
- Diğer modellere göre daha karmaşıktır.
- Küçük çablı projeler için maliyeti yüksek bir yöntemdir.

Yazılım süreç modelleri birbirinden tamamen farklı değildir, hepsinin birbirine benzer ortak yönleri vardır. Her model farklo bir konuda üstündür. Bu yüzden kullanılacak alana göre seçilmelidirler. Maliyet, projenin büyüklüğü, projenin tamamlanması gereken zaman aralığı, projenin karmaşıklığı ve gereksinimleri gibi kriterler değerlendirilip ona göre seçilmelidir. Günümüzde en çok kullanılan model kısa zaman aralığında geliştirilebilir olması ve başarıyla sonuçlanabilir olmasından dolayı Çevik Modeldir.

5.ÇEVİK MODELLER:

SCRUM:

SCRUM, Agile (çevik) modelinin en popüler yöntemidir. Günümüzd bu kadar popüler olmasının bir çok sebebi vardır. Karışık ve zor yazılım süreçlerinin yönetilmesi amacıyla kullanılır. Tekrarlama yönteminden faydalanır. Düzgün bir planlama ve düzenli geri bildirimlerle süreci devam ettirir, müşteri ile sağlam bir iletişimde bulunulur, müşterinin istek ve gereksinimlerine göre bir planlama yapılır. Projelerin büyüklüğü ve karmaşıklığı sonucu gereksinimlerin tam ve doğru belirlenemeyişi gibi pek çok konuda çok büyük kolaylıklar sağlar. Çevik (agile) Yazılım Geliştirme, bireylerin daha etkileşimli olmasına, müşteri ile işbirliği halinde olmaya, değişime uyum sağlamaya dayanmaktadır. Bu şekilde proje sürecinde bile gereksinimler değişirse bu değişimlere göre projenin son aşamasında dahi düzenlemeler yapılıp uyum sağlanabilir. Bu geliştirme modeli sayesinde hata payı azaltıldı, proje/iş sahibi ve çalışanlar kuvvetli bir etkileşim ile birlikte çalıştılar. Ayrıca çalışanlar arası iletişim ççok önemlidir ve kuvvetli olmalıdır. Bu modelin üç temel prensibi bulunmaktadır:

1. Şeffaflık; Proje esnasında gelişmeler ilerlemeler,sonuçlar herkes tarafından açıkça görülebilir ve anlaşılabilir olmalı.
2. Denetleme; Proje süreci devam ederken sürekli ve düzenli olarak kontroller yapılır.
3. Uyarlama; Süreç devam ederken değişiklik olursa proje bu değişimlere uyum sağlayabilmelidir.

6. HANGİ PROJELER HANGİ ALANDA DAHA İYİ?

Günümüzde Gelişigüzel ve Barok modellerinin kullanımı azalmıştır. Çünkü dökümantasyon içerirler ve tekrarlı değillerdir. Çağlayan Modeli planlama yapma, takım çalışması, iş bölümü gibi konularda oldukça kullanışlıdır. Kullanımı ve uygulaması kolaydır. Fakat büyük ve karışık projelerde yetersiz kalması ve kullanıcı ile arasındaki letişimin yetersiz olması ve proje sürecinde oluşabilecek değişimlere uyum sağlayamamasından dolayı günümüzde sık kullanılmaz. V Modeli, Çağlayan Modelinin gelişmiş versiyonudur. Takip kontrolü yapılması ve kullanımı kolaydır. Kontrol ve onaylama işlemleri erken adımlarda yapılabilmesi avantajlıdır. Fakat risk çözümleme basamakları bulunmaması ve fazlar arası yineleme imkanının olmayışı dezavantajdır. Spiral Model daha şeffaftır. Kullanıcılar süreci, ara ürünleri görüp net bir şekilde takip edebilirler. Bu takipler sonucu hatalar erkenden tespit edilip giderilebilir. Spiral Modelde risk analizi çok önemli bir yerdedir. Büyük projeler için kullanımı avantajlıdır. Kodla ve Düzelt

Modelinde ise daha çok küçük projelerde kullanılır. Emeklilik safhası bulunur. Dökümantasyon gibi olaylar yoktur. Bakım safhası zordur. Maliyetlidir. Program en son halini alana kadar kodlama yapmaya devam edilir. Ve sonra teslim edilir. Evrimsel Geliştirme Metodunda ise gereksinimler daha iyi anlaşılır. Fakat sürecin görünürlüğü azdır ve bu modelin bakımı zordur. Çevik Modellerde kullanıcı ile program geliştiriciler arasındaki iletişim oldukça kuvvetlidir. Bu yüzden tartışılarak hatalar olabildiğince azaltılır ve daha gelişmiş bir proje elde edilir. Değişime uyum sağlayabilen bir modeldir, alışılması kolaydır. Takım çalışmasının önemli bulunması, bu modelde baskın bir özelliktir. Günümüzde çok sık kullanılan bir modeldir.

HANGİ PROJEDE HANGİ MODEL KULLANILMALI?

Büyük bir kitleye ulaşacak bir proje olması hedefleniyor ise Evrimsel Geliştirme Modeli tercih edilebilir. Büyük, maliyetli ve uzun süreli projelerde ise Spiral Model veya Artırımsal Geliştirme Modeli kullanılabilir. Kişiye özel küçük programlarda Kodla ve Düzelt Modeli uygundur. Küçük, gereksinim ve özellikleri açık ve net olarak belli olan projelerde Çağlayan Modeli kullanılabilir. Belirsizlik içermeyen, tanımların belirgin olduğu projelerde V Modeli kullanılabilir. Çok büyük olmayan, uzun bir süreci kapsamayan projelerde ise Çevik Modeller kullanılabilir.

KAYNAKLAR:

https://akademiksunum.com/index.jsp?modul=document&folder=a93e3a2fccf8eb56a557c55c5f0d5cf10789abe2

https://www.tutorialspoint.com/software_engineering/software_development_life_cycle.htm

https://tr.linkedin.com/pulse/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-nedir-veysel-ugur-kizmaz#:~:text=Yaz%C4%B1l%C4%B1m%20Ya%C5%9Fam%20D%C3%B6ng%C3%BCs%C3%BC%20Nedir%3F%20(Software,Yaz%C4%B1l%C4%B1m%20Geli%C5%9Ftirme%20Ya%C5%9Fam%20D%C3%B6ng%C3%BCs%C3%BC%20denir.

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

NİLAY YAMAN

220601078