YAZILIM YAŞAM DÖNGÜSÜ

Yapılmak istenilen bir projeyi geliştirmek sadece kodlamadan ibaret değildir. Yazılım geliştirme bazı

aşamalardan oluşur ve bu aşamaların bütününe yazılım yaşam döngüsü denir. Yazılım yaşam döngüsü modelleri

ihtiyaçlara göre değişiklik göstermekle beraber bir döngü halindedir doğrusal değildir. Yazılım yaşam döngüsü

temel olarak beş adımdan oluşur. Bu adımlar gereksinim, analiz, tasarım, gerçekleştirme ve testtir. Gereksinim

safhası bu döngünün ilk adımıdır. Bu adımda mevcut problemler ve müşterinin gereksinimleri doğrultusunda

projenin planlaması yapılır. Analiz aşamasında gereksinim aşamasında tespit ettiğimiz problemleri detaylıca

inceleyerek temel sorunlar ortaya çıkarılır ama bu aşamada müşterinin isteklerine göre değil ihtiyaçlarına göre

hareket etmek gerekir. Tasarım aşamasında, analiz aşamasındaki sonuçlara göre yapılacak işlemler adım adım

belirlenir yani plan oluşturulur. Tasarım aşaması mimari tasarım ve detaylı tasarım olarak iki aşamadan oluşur.

Gerçekleştirme adımında elde edilen sonuçlara göre kodlama yapılır ve test edilir testte çıkan hatalar düzeltilir.

Bu ve bundan sonraki aşamalarda analiz yapılmaması gerekir. Son adım yani bakım aşaması projenin

yayınlandıktan veya teslim edildikten sonraki oluşabilecek hataların giderilmesi ve projenin geliştirilmesi gibi

adımları içerir. Bu adımda kullanıcılardan veya müşterilerden gelen geri dönüşlere dikkat edilir. Yaygın olarak

kullanılan 9 tane yazılım geliştirme modeli vardır. Bunlar Büyük Patlama (Big-Bang) Modeli, Kodlama ve

Düzeltme (Code and Fix) Modeli, Çağlayan (Waterfall) Modeli, V Modeli, Artırımsal Geliştirme (Incremental

Development) Modeli, Helezonik (Spiral) Model, Barok Modeli, Gelişigüzel Model, Evrimsel Geliştirme

(Evolutionary Development) Modelidir.

1)Büyük patlama (Big-Bang) Modeli: Hiç planlamanın olmadığı veya çok az olduğu basit bir modeldir.

Uygulaması kolay ve adımlar içermez bundan dolayı proje için bitirme tarihi vermek çok zordur. Genellikle

küçük geliştirme ekipleri ve akademik yazılım projeleri için kullanılır.

2)Kodlama ve Düzeltme (Code and Fix) Modeli: Genellikle kısa projeler için tercih edilir. Tamamlanması

gereken ürüne odaklanılır. Direkt kodlamaya başlanır ve kodlama bitene kadar kodlamaya devam edilir.

Kodlama sürerken tespit edilen hatalar düzeltilir. Planlama olmadığı için bu modelde de proje bitiş zamanı

söylemek zordur. Bakım safhası mevcuttur fakat dokümantasyon olmadığı için bakım yapmak oldukça zordur ve

bir değişiklik yapılmak istenirse bu oldukça maliyetli olur. Bir yazılım ürünü geliştirmek için uygulanabilecek en

kolay modeldir çünkü dokümantasyon yoktur.

3)Şelale (Waterfall) Modeli: Klasik yaşam döngüsü olarak da adlandırılır. Kullanılması ve yönetimi kolaydır.

Her adım en az bir kere tekrarlanır. Her safhada dokümantasyon yazılır. Dokümantasyon süreçle beraber devam

eder ve .Süreç uzun süreceğinden dolayı gereksinimler değişebilir. Bu değişimler maliyeti arttırır Tüm ihtiyaçlar

Proje başında bilinir. Şelale modelinde geri dönüşler ve değişiklik yapmak zordur. Kullanıcı sürecin içinde yer

almaz. Bundan dolayı bakım aşamasında istekler artar bu da maliyeti arttırır.

4) V modeli: V harfine benzediğinden dolayı V modeli olarak adlandırılmıştır. V modelinin sol tarafı üretim,

Sağ tarafı test aşamalarıdır. Kullanıcıyla iletişim vardır. Kodlamayla eş zamanlı olarak test de gerçekleştirilir

Her adımda test yapılır. Bundan dolayı kodlamada oluşacak hatalar erken tespit edilir ama sürekli test yapıldığı

için projeyi tamamlamak fazla zaman alır.

5)Artırımsal Geliştirme (Incremental Development) Modeli: Şelale modelindeki adımları yinelemeli olarak

Kullanır. Divide an conquer (Böl ve Yönet) yaklaşımını kullanır. Yani döngü vardır. Tek seferde teslim edilmez

parçalara bölünür ve parçalar eklenerek teslim edilir. Ürün kullanılırken aynı zamanda üretimi devam eder. Ürün

kullanılırken üretim devam ettiğinden dolayı müşterinin ihtiyaçları gözden kaçmaz ve istenildiği gibi bir ürün

çıkar ve projenin batma ihtimali düşüktür. Proje parça parça teslim edildiğinden dolayı projenin tamamının

bitmesi uzun sürer.

6)Helezonik (Spiral) Model: Yineleme ve şelale modellerini kombinasyonudur. Risk analizi aşamasının önemli

olduğu bir modeldir. 4 ana aşaması vardır ve genel Olarak bu aşamaların yinelenmesi şeklinde devam eder. Bu

aşamalar Planlama, Risk Analizi, Üretim ve Kullanıcı değerlendirmesidir. Prototiplerden dolayı müşteriyi de

kapsayan bir modeldir. Bundan dolayı oluşabilecek istenmeyen durumların önüne geçilmiş olur. Spiral sürekli

tekrarlandığından dolayı sonsuza kadar sürebilir bu da zamanı ve maliyeti olumsuz bir şekilde etkiler. Genellikle

küçük projeler için uygun değildir çünkü pahalıdır.

7)Barok Modeli: Yaşam döngüsünün temel adımlarını doğrusal bir bicinde uygulamayı amaçlayan bir modeldir.

Bir aşamadan geriye dönmek oldukça zordur. Barok modelinde belgeleme önemlidir ve ayrı bir süreç olarak ele

alınır. Barok modeli günümüzde fazla kullanılan bir model değildir eski bir modeldir 1970’li yıllarda ortaya

çıkmıştır. Gerçekleştirim evresine çok önem verilir.

8)Gelişigüzel Model: Model veya yöntem gözetmez. Herhangi bir yöntem ve aşamaları yoktur. Projeyi

geliştirecek kişiye bağlıdır. Genellikle tek kişinin geliştireceği projelerde kullanılır. Çok basit programlamaya

sahiptir ve anlaşılması zorudur.

9)Evrimsel Geliştirme (Evolutionary Development) Modeli: Evrimsel geliştirme modeli ilk tam ölçekli

Modeldir. Projenin uygulanma sahası olarak fazla olan projeler için önerilir. Kullanıcıların gereksinimlerini

Daha iyi anlamalarını sağlar. Sürekli olarak değerlendirilir bu da hataların minimalize edilmesine olanak sağlar.

Modelinin bakımı zordur ve değişiklilerin fazla olması yazılımın olumsuz etkilenmesini sağlar.

Genel Olarak Yazılım Yaşam Döngülerinin Avantajları ve Dezavantajları:

1)Büyük patlama (Big-Bang) Modeli:

Avantajları: Basit ve yönetimi kolay bir modeldir.

Dezavantajları: Karmaşık ve uzun vadeli projeler için risklidir.

2)Kodlama ve Düzeltme (Code and Fix) Modeli:

Avantajları: Plana ihtiyaç duyulmadan direkt başlanır. Aşamaları hızlı geçilir. Herkes tarafından kullanılabilir.

Dezavantajları: Kontrolsüzdür. Koddaki hataların bulunması, değiştirilmesi ve düzeltilmesi zordur aynı zamanda

maliyetlidir. Bitiş süresi belli değildir.

3)Şelale (Waterfall) Modeli:

Avantajları: Kullanımı, yönetimi ve anlaşılması kolay bir modeldir. Aşamalar ayrı olduğundan iş bölümü ve

Planlanması kolay bir modeldir. Küçük projeler için idealdir.

Dezavantajları: Karmaşık ve uzun projeler için uygun değildir. Değişim yapmak zor ve maliyetlidir bundan

dolayı müşterinin ihtiyaçları değişirse sıkıntı yaratır.

4) V modeli:

Avantajları: Kullanımı ve yönetimi kolay bir modeldir. Testle içi içe bir model olduğu için hata oranı düşüktür.

Dezavantajları: Projenin tamamlanması uzun sürer. Gerçekleştirilebilecek olaylara kolay imkan tanımaz.

5)Artırımsal Geliştirme (Incremental Development) Modeli:

Avantajları: Prototip kullanıldığı için müşterinin istekleri yerine getirmek kolaydır. Projenin başarısız olma

İhtimali azdır. Gereksinimlerin önemine göre belirlenir.

Dezavantajları: Deneyimli personel gerektirir. Projenin tamamlanması uzun sürer.

6)Helezonik (Spiral) Model:

Avantajları: Riskli kısımlardan başlanır. Hataları erken gidermeye odaklanılır.

Dezavantajları: Küçük ve düşük riskli projeler için pahalıdır. Karmaşıktır ve çok uzun sürebilir.

7)Barok Modeli:

Avantajları: Belgelendirme önemlidir.

Dezavantajı: Geri dönüşler hakkında bilgi vermez. Geriye dönmek zordur.

8)Gelişigüzel Model:

Avantajları: Kişisel bir modeldir. Tek kişiliktir.

Dezavantajları: Bakım yapmak imkansızdır. Plansız olduğundan anlaşılması güçtür.

9)Evrimsel Geliştirme (Evolutionary Development) Modeli:

Avantajları: Hata azdır. Kullanıcıların gereksinimlerini Daha iyi anlamalarını sağlar

Dezavantajları: Sürecin görünürlüğü azdır. Bakımı zordur. Değişiklik yapmak projeye zarar verir.

Çevik Yazılım Geliştirme: Bu modeli uygulamak süreci daha esnek ve güçlü yapar. Aynı zamanda yazılımın

tamamlanma sürecini de azaltır. Bu modelde katı planlama ve geliştirme süreçleri yoktur. Proje küçük parçalara

ayrılır ve her bölüm kendi başına bir projeymiş gibi değerlendirilir ve ona göre geliştirilir. Tamamlanan küçük

parçalar hakkında müşteriye bilgi verilir. Küçük parçalardan oluşması hataların tespiti ve düzeltilmesine olanak

sağlar. Çevik Yazılım verimliliği yüksek, hızlı, ucuz ve düşük hatalıdır. Temel olarak prensipleri müşteriyle olan

iletişimi arttırarak müşteriye sürekli ve hızlı teslimatlar yapmaktır. Projede bir değişiklik yapmak oldukça

kolaydır. Kısa zamanda kaliteli proje teslimi yapmayı amaçlar. Verimliliği arttırmak için projedeki insanlara

güvenmek ve onlara destek sağlamak önemlidir. Yaygın olarak kullanılan 2 tane çevik yazılım yöntemi vardır.

Bunlar Extreme Programming (XP) ve Scrum’dur.

1) Extreme Programming (XP): XP 1999 yılında Kent Beck tarafından ortaya çıkarılmıştır. Takımdaki iletişime

Önem veren, kolay ve geri dönüşlerin artmasını sağlayan bir Çevik yazılım geliştirme modelidir. 4 temel değer

üzerine kurulmuştur. Bunlar iletişim, basitlik, geri bildirim ve cesarettir. Xp yazılım geliştirme disiplini ayrıca

esnekliği ve kolaylığı arttırmak için 12 farklı pratik yönteme de sahiptir. Bunlar ise planlama oyunu, ekipte

müşteri, önce test, basit tasarım, çiftli programlama, sürekli entegrasyon, kısa aralıklı süremler, yeniden

yapılandırma, ortak kod sahiplenme, metafor, kodlama standartı, ve haftada 40 saat’tir.

İletişim: Projelerde genellikle oluşan büyük problemlerin başında ekipteki insanların birbirleriyle etkili bir

şekilde anlaşamamasından dolayı oluşmaktadır ama projenin başarıya ulaşması için ilk başta ekipteki insanların

sağlıklı bir şekilde iletişim kurmaları gerekmektedir. XP bu iletişimsizliği ortadan kaldırmayı ve verimi

arttırmayı hedefler. Sağlıklı iletişim yüz yüze ve anlaşılır olmalıdır. Özellikle proje ekibi ve müşteri arasındaki

iletişim daha önemlidir. Bu sağlıklı iletişim yanlış anlaşılmanın önüne geçer ve projede oluşabilecek hataları

engellemiş olur. Tam anlaşılmayan bir kısım olduğunda kısa bir sürede müşteriyle iletişime geçerek projenin

geliştirmesinin hız kesmeden devam etmesini sağlar.

Basitlik: Yapılan projedeki işlerin karmaşık gözükmemesi gerekmektedir. Karmaşıklık Xp’ye aykırıdır ama

basitliği sağlamak güçtür. Xp basitliği en iyi şekilde sağlanmasını amaçlar.

Geri Bildirim: Geri bildirimler oldukça önemlidir. Geri bildirimler gelecekte ortaya çıkabilecek problemlerin

Ortadan kaldırılmasını sağlar. Geri bildirimler sadece müşteri tarafından yapılmaz aynı zamanda proje yöneticisi

ve ekip çalışanları tarafından da yapılabilir. Kısa süreli aralıklarla müşteri ve ekip arasında yapılacak görüşmeler

ileriki dönemlerde oluşabilecek büyük hataların ve anlaşmazlıkların önüne geçer.

Cesaret: Bu 4 temel değerden gerçekleştirmesi en zor olan cesarettir. Projeye korkusuzca yaklaşmak, projenin

geliştirilmesine çok büyük katkı sağlar çünkü korkarsak yapmamız gereken veya yapmamamız gereken şeyleri

gözümüzden kaçırmış oluruz ve bu projenin geliştirilmesini olumsuz etkiler. Elinizi korkak alıştırmadan, hata

yapmaktan korkmadan yazılım geliştirilmesi gerekir. Xp hatalarımızdan korkmayı değil hatalarımızı hızlıca

telefi etmemizi önerir. Başarısızlıktan ve hatalardan korkmak projenin daha hızlı geliştirilmesinin önüne geçer ve

yavaşlamamıza neden olur.

Scrum: Scrum çevik yazılım geliştirme yöntemlerinin başında gelir. Adını rugby sporundaki bir hücum

Taktiğinden alır. Bu taktik tüm oyuncularla birlikte hücum yapmayı amaçlar. Scrum 1990’ların ortalarında

Jeff Sutjerland ve Ken Schawaber tarafından geliştirilen çevik yazılım geliştirme yöntemlerinden biridir ve en

popüleridir. Modern yazılım geliştirmenin daha da karmaşıklaşmasından dolayı bu yöntem ortaya çıkmıştır.

Karmaşık kısımları küçük parçalara bölmeyi ve adım adım çözmeyi amaçlar. Karmaşık ve anlaşılması güç

projeler için en uygun yöntemdir. 15-30 günlük sürelerde günde 15 dakikalık toplantılarla gelişmeler

değerlendirilir. Scrum’un 3 temel prensibi vardır. Bunlar şeffaflık, Gözlem ve adaptasyondur.

Şeffaflık: projenin gidişatı, karşılaşılan sorunlar ve gelişmeler herkese iletirlir.

Gözlem: Projenin ilerleyişi ve gelişimi ortak olarak kontrol edilmedir.

Adaptasyon: projedeki sorunlar çözülemiyor ve başta belirlenen planın dışına çıkılıyorsa uygulanan adımlar ve

Ürün değiştirilebilir.

Scrum’da 3 temel kavram vardır. Bunlar Roller (Roles), Toplantılar (Meetings) ve Bileşenler (Artifacts)dır.

1) Roller (Roles):

Ürün Sahibi (Product Owner): Projedeki geri dönüşlerden sorumlu olan müşteridir.

Scrum Yöneticisi (Scrum Master): Takımın kurallara ve temel değerlere bağlı kalmasını sağlayan yöneticidir.

Scrum Takımı (Scrum Team): Genellikle 5-9 kişiden oluşan, birbirleriyle devamlı iletişim halinde olan ve

hedefe ulaşmak için mücadele eden takımdır.

Chicken Roller: Projede ve işleyişte doğrudan yer almayan kişilerdir.

2)Toplantılar (Meetings):

Koşu Planlama (Sprint Planning): Gereksinim listesi çıkartılır ve dağıtım gereksinimleri belirlenir. Dağıtım için

takımlar belirlenir ve takımlara uygun biçimde dağıtım yapılır. Risk değerlendirilmesi ve kontrolleri belirlenir.

Geliştirme araçları ve altyapısı onaylanır.

Koşu Gözden Geçirme (Sprint Review): Yapılan sprint ve ortaya çıkan ürün değerlendirilir. Ürünün müşterinin

veya kullanıcının gereksinimlerine uygun olup olmadığına bakılır ve hata tespit edilirse hata düzeltilir.

Günlük Scrum Toplantısı (Daily Scrum Meetings): Her gün Düzenli olarak yapılır. Takımdaki üyelerin dün

ne yaptıklarını, gelecek toplantıya kadar ne yapacaklarını belirttikleri kısa süreli toplantılarıdır ayrıca herhangi

bir problemleri varsa toplantıdan sonra scrum master veya toplantıdaki üyelerden yardım isteyebilir. Çok aykırı

bir durum olmadıkça her gün bu toplantı gerçekleştirilir. Takımdaki birkaç kişinin toplantıya gelememesi

toplantıyı etkilemez.

3) Bileşenler (Artifacts):

Ürün Gereksinim Dokümanı (Product Backlog): Proje boyunca yapılması gereken iş elemanlarının listesidir.

Bu doküman canlı bir dokümandır ve devamlı bakım gerektirmektedir. Listedeki elemanlar değişiklik

gösterebilir.

Koşu Dokümanı (Sprint backlog): Mevcut sprintdeki iş ve görevleri kapsar. Sadece takım tarafından değiştirilir.

Sprint Kalan Zaman Grafiği: İşlerin ne kadar yapıldığı, ne kadar yapılması gerektiğini gösteren ve zamanı

planlamamızı sağlayan grafiklerdir.

Scrum Neden Popülerdir:

Müşterinin isteklerini ve ihtiyaçlarını açıklaması scrum’un doğrudan müşteriyi projeye dahil etmesinden dolayı

çok daha kolay oluyor. Ekip çalışması ön planda ve ekip içi iletişim kuvvetli olduğundan dolayı projeyi

geliştirmek çok daha kolay oluyor. Karmaşık yapıları küçük parçalara ayırıp basitleştiriyor ve ona göre

bu küçük parçaları ekibe dağıtıyor .Her gün gerçekleştirilen kısa toplantılar projenin hangi aşamaya geldiğini ve

daha neler yapılması gerektiğini görmeyi kolaylaştırıyor. Müşteri ve proje geliştirme ekibi arasında bir güven

oluşturuyor bu da projenin daha başarılı olmasına olanak sağlıyor. Bütün bu olumlu yönleri scrum’u günümüzde

en kullanışlı ve popüler yazılım geliştirme çerçevesidir.

Kaynakça:

1. <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>
2. Doç. Dr. Deniz KILINÇ Yazılım Mühendisliği Temelleri ders notları
3. <https://enprobilisim.com/yazilim-gelistirme-sureci-modelleri-sdmp/>
4. https://medium.com/@secilcor/scrum-nedi̇r-6a4326951dd8

210601016 Mehmet Eren Erdemir