İzmir Bakırçay Üniversitesi, Bilgisayar Mühendisliği, Mehmet Taha Cumurcu, Öğrenci Numarası: 220601027, Yazılım Yaşam Döngüleri.

İçindekiler: Barok Yaşam Modeli, Çağlayan Yaşam Modeli, Evrimsel Yaşam Modeli, ‘‘V’’ Yaşam Modeli, Spiral Yaşam Modeli, Kodla Ve Düzelt Yaşam Modeli, Agile Yazılım Geliştirme, Scrum.

Yazılım Yaşam Modeli: Yazılım yaşam modelleri temel olarak yazılım geliştirmenin temel süreçlerini ele alış biçimi olarak farklı yollar izlenmesiyle ortaya çıkmış süreç yönetim biçimleridir. Temel olarak yazılım yaşam döngülerinin sırasıyla; gereksinim, analiz, tasarım, gerçekleştirme-test, teslim sonrası bakım, emeklilik aşamaları bulunur. Analiz aşaması neye ihtiyacımız olduğuyla, analiz aşaması projenin ne işe yaradığıyla, tasarım aşaması spesifik olarak projenin algoritmik kısımlarıyla, gerçekleştirme aşaması kodlama ve test aşamalarıyla, teslim sonrası bakım aşaması hataların giderilmesi ve projenin geliştirilmesiyle ilgilenir. Emeklilik aşaması ise programın ömrünün dolup kullanımına son verildiği aşamadır. Yazılım dünyasında farklı projeler için bu temel aşamaların farklı yorumlanması birbirinden farklı modeller geliştirilmesine yol açmıştır.

1.Barok Yaşam Modeli: Barok yaşam modeli, yazılım geliştirme modellerinin temel aşamalarını doğrusal bir şekilde izler. Temel aşamaların doğrusal bir şekilde izlenmesi statik bir geliştirme sürecine sebep olur. Yazılım geliştirmenin doğası gereği statik bir geliştirme süreci gerçekçi olmamakla beraber verimli olduğu da tartışılır. Temel aşamaların doğrusal izlenmesi aynı zamanda denetleme mekanizmasının diğer modellere göre zayıf kalmasına, bu sebeple geliştirme sürecinde yaşanan bir hatanın çözümlemesinin diğer modellere göre zor ve maliyetli olmasına sebep olur. Barok yaşam modelinde yapılması gereken geri dönüşlerin nasıl yapılacağı da tanımlı değildir. Geri dönüşlerin tanımlı olmaması, doğrusal bir süreç izlenmesi ve bu sebeple denetleme mekanizmasının zayıf olması Barok modeli günümüz için ideal bir yazılım yaşam modeli olmaktan uzaklaşmasına sebep olmaktadır. Barok yaşam modeli diğer yazılım yaşam modellerinden farklı olarak dokümantasyon olgusu bu modelde ayrıca bir aşama olarak yer alır.

2.Çağlayan Yaşam Modeli: Çağlayan yaşam modeli, Barok yaşam modelinde olduğu gibi yazılım geliştirme modellerinin temel aşamalarını doğrusal bir şekilde işler. Barok yaşam modelinin aksine geri dönüşlerin nasıl yapılacağı tanımlıdır. Temel aşamalardan birinden diğerine geçildiğinde geçilen aşama önceki aşamadan gelen bilgileri kendi perspektifiyle tekrar günceller. Tekrarlama olgusu birden fazla sayıda gerçekleşebilmekle beraber en az bir kere gerçekleşir. Temel aşamalar lineer bir şekilde izlenirken her aşamanın kendine ait bir dokümantasyon ve test olgusu vardır. Lineer bir süreç izlediğinden projenin başında projenin gereksinimleri ve yönetimi projenin başından bellidir. Kümülatif bir süreç izlememesinden ötürü uygulanması ve yönetimi diğer modellere göre daha kolaydır. Bahsedilen sebepler bu modelin küçük çaplı ve net gereksinimleri olan projelerde yol haritası olmasını sağlamıştır. Günümüzde kullanımının gittikçe azalmasının sebepleri olarak; uzun geliştirme süreci , yazılım geliştirme ve gelişen dünyada yazılımın başlangıç aşamasından gereksinimlerinin net bir şekilde belirlenememesi, kullanıcının sürecin içinde bulunmaması ve bu sebeple geri dönüşlerin fazla olması, kullanıcının sürecin sonunda geri dönüş talebiyle geri dönüş maliyetlerinin aşırı fazla olması, kullanıcıyı memnun etmenin zor olması, kağıt üstünde temel

aşamaların ayrı birer basamak olarak gözükmesi fakat realitede aradaki sınırların net bir şekilde belirgin olmaması ve büyük projeler için uygun olmaması gösterilebilir.

3.Evrimsel Yaşam Modeli: Evrimsel yaşam modeli, Çağlayan ve Barok yaşam modellerinden farklı olarak kümülatif bir yapıdadır. İsmini programın ihtiyaçlara göre gelişmesinden yani kısmi bir evrim geçirmesinden almıştır. Aynı zamanda ilk tam ölçekli model olarak da değerlendirilir. Evrimsel yaşam modeli; büyük ölçekli, genel olarak geniş bir alana yayılmış, birden fazla yapılaşması olan oluşumlar tarafından kullanılması önerilen bir yaşam modelidir. Bunun sebebi olarak evrimsel yaşam modelinin sürekli olarak gelişen bir yapıda olması ve ihtiyaçlara göre ekstra özellikler eklenmesi, bu özelliklerin analizinin iyi yapılabilmesiyle sağlıklı bir proje geliştirme sürecinin mümkün olmasını beraberinde getirmektedir. Bu analizlerin iyi yapılmasının da en kolay yollarından biri olarak geniş bir kitleye geniş bir coğrafyada hitap etmek olması sebebiyle büyük ve birden fazla yapılaşması olan oluşumlar tarafından kullanılması önerilir. Tüm aşamalarda üretilen ürünler, üretildikleri aşama için tam fonksiyonel olarak geliştirilmelidir. Bu sürekli gelişen yapı beraberinde kullanıcıların gereken işlevlere ulaşmasını kolaylaştırır. Proje bir evrim süreci içinde olduğundan ötürü değerlendirme ve risk yönetimi yapmak kolaylaşır. Sürekli bir evrim süreci bahsedilen avantajların yanında görünürlüğü az olan bir yapı ortaya çıkarır. Bu yaşam modelini izleyen projelerde güncelleme yapmak çok zor ve risklidir.

V Yaşam Modeli: V yaşam modeli, Çağlayan yaşam modeline benzemekle beraber, Çağlayan yaşam modelindeki lineer yapıyı bulundurmaz. İsmini çift taraflı görsel şemasından alır. Şemanın sol tarafı geliştirme aşamasına ait iken sağ tarafı test aşamasını belirtir. Yazılım geliştirmenin temel aşamalarının her birinin kendine ait test aşaması, kodlama aşamasından sonra gerçekleşir. Evrimsel yaşam modelinde olduğu gibi bu model de kullanıcının geliştirme sürecinde etkin bir rol almasını sağlar. Kullanıcının etkin bir rol alabilmesini her aşamanın kendine ait test aşaması olmasına borçludur. Her aşamanın kendine ait bir test aşamasıyla beraber yer alması projenin yönetimi ve takibini kolaylaştırır. Projenin takibinin ve yönetiminin kolaylaşmasının yanı sıra aşamalar arasında dönüş mekanizması ve risk yönetimi planlamasına sahip değildir.

Spiral Yaşam Modeli: İsmini birbirini tekrar eden dört aşamadan alır. Planlama, risk analizi, üretim ve kullanıcı değerlendirmesi aşamalarından oluşur. Risk analizi aşamasının öne çıkmasının yanı sıra her aşama yinelemeli artırımlı bir yapıya sahiptir. Projenin ilk olarak bir prototipi, sonrasındaysa versiyonları oluşur. Her bir versiyondan temel 4 aşama tekrar edilir. Bu artırımlı yapı kullanıcının proje hakkında erken bir fikir sahibi olmasını sağlar. Dönüşümlü yapısı sayesinde geliştirme süreci parçalara bölünür ve en riskli kısımlar baştan yapılır. Pek çok yazılım yaşam döngüsünün olumlu özelliklerini içinde barındırır. Risk yönetiminin öne çıkması geliştirme sürecinin potansiyel risklere hazırlıklı olmasını sağlar. Döngüsel yapısı projenin her aşamada tekrar tekrar sınanmasını sağladığından hataların maliyeti düşüktür. Bu avantajların yanı sıra döngüsel yapının sayısının tahmin edilememesi projenin geliştirme aşamasını çok karmaşık bir hale getirebilir. Yeni versiyon geliştirme sürecinde her aşamanın baştan uygulanması dokümantasyon olgusunun sayıca çok fazla olmasına ve karmaşık bir hale bürünmesine sebebiyet verir.

Kodla Ve Düzelt Yaşam Modeli: Kodla Ve Düzelt yaşam modeli, yazılım geliştirmenin temel aşamalarını izlemez. Basit projeler için uygulanabilir. Tek bir aşamadan oluşur ve bu projeyi gerçekleştirmektir. Bakım safhası, Kodla Ve Düzelt yaşam modelinin yazılım geliştirmenin temel aşamalarını izlememesinden ötürü çok maliyetli ve zordur. Dokümantasyon yoktur. Yazılım geliştirmenin en kolay yollarından biri olmasından ötürü tecrübesiz veya küçük firmalarda izlenen yollardan bir tanesidir.

Agile (Çevik) Yazılım Geliştirme: Yazılım geliştirmenin doğası gereği dinamik ve uzun süren bir süreçtir. Yazılım projeleri yönetimin zorluğu, proje gereksinimlerinin karmaşıklığı vb. sebeplerle kısmen memnuniyetle sonuçlanmaktadır. Bu problemin ışığında yapılan çalışmalar 1990'lı yılların sonunda Agile (Çevik) Yazılım Geliştirme modelleri ortaya çıkmıştır.

Bu modellerin amacı değişken proje isteklerine cevap verebilmek, projeleri zamanında ve ucuz bir şekilde üretebilmektir. Metodolojinin temel ilkesi, proje büyüklüğü fark etmeksizin projeyi küçük iterasyonlara ayırmakta yatar. İterasyonların sonunda geliştirici ekip ile müşteri arasında bir komünikasyon kurularak projenin işleyişi tartışılır. Çevik yazılım geliştirmenin bir iterasyonu 2 ila 4 hafta arasında sürerken bu durumun amacı müşteriyle geliştirici ekip arasındaki komünikasyonun güçlü kalmasını sağlamaktır. Proje geliştirme sürecinin temel ilkesi güçlü komünikasyon olmasından ötürü proje geliştirme aşamasında çevik manevralar yapılması kolaydır. Çevik yazılım geliştirmenin dayandığı ilkelerden ötürü müşteri memnuniyeti yüksek, geliştirme ekibinin motivasyonu yüksek, kaliteli bir geliştirme süreci vardır. Bunların yanı sıra çevik manevralar artan mesai saatleri anlamına gelebilir, çevik bir geliştirme sürecini verimsiz bir şekilde tamamlamak takım üzerinde olumsuz bir imaj bırakabilir.

SCRUM: Çevik Yazılım Geliştirme modellerinden biri olan SCRUM aynı zamanda sadece yazılım geliştirme alanında değil herhangi bir alandaki herhangi bir projeye uygulanabilir. SCRUM bir projeyi parça parça geliştirmeyi (sprintlere bölerek) amaçlar. Bir projeyi parça parça geliştirmek karmaşaya yol açabileceğinden 3 olguyla bu durumun önüne geçilmeye çalışılmıştır: Şeffaflık, denetleme, uyarlama. Şeffaflık, yazılım geliştirme süreci boyunca geliştirme süresince oluşan ürünün herkes tarafından izlenebilir olmasını sağlamak ve bu sayede oluşabilecek bir probleme kolay bir şekilde müdahale edilebilmesini sağlar. Denetleme, üretim sürecindeki projenin düzenli bir şekilde kontrol edilmesini ve bir aksaklık çıkmamasını sağlar. Uyarlama ise sprintler sonucu elimizdeki ürünün bilgisayar sistemlerinde nasıl çalışacağını yani sprintler sonunda elinizdeki projeyi görmenizi sağlayan aşamadır. Bu olguların yanı sıra SCRUM 4 yapıtaşı daha içerir: Product Backlog, Sprint Backlog, Daily Meetings, Members. Product Backlog (Ürün Dökümanı), projenin genel gereksinim listesidir. Tüm sprintleri kapsar. Sprint Backlog (Depar Dökümanı), 2-4 hafta arası olan bir sprint'in gereksinim listesidir.Daily Meetings (Günlük Toplantı), gün içinde sprint hakkındaki fikir alışverişlerinin yapıldığı toplantıdır. Members (Üyeler), 3 temel öğeden oluşur: Müşteri, yönetici, geliştirici takımı. Müşteri yapılan toplantılarda projeye yön verir. Yönetici tecrübeleriyle geliştirici takımını yönlendirerek sprintin sağlıklı bir şekilde gerçekleşmesini sağlar. Bu ilkeler ışığında SCRUM bir projenin parça parça tamamlanmasını hedefler. Parça parça geliştirme süreci geri dönüşlerle çevik bir yapı kazandırır.

Yararlanılan Kaynaklar:

https://medium.com/@secilcor/scrum-nedİr-6a4326951dd8

https://bilginc.com/tr/blog/scrum-nedir-184/

http://agilemanifesto.org

Schach, S.R., (2011),"Object-Orientedand ClassicalSoftwareEngineering",8thEd ., McGraw-Hill.

Highsmith,J. , (2002),“Agile Software Development Ecosystems”, Addison Wesley.

Beck, K., (2002), “Extreme Programming”, Addison Wesley.

Schwaber, K., (2004), "Agile Project Management with Scrum", MicrosoftPress

Yücalar,F., (2013-2018),"Yazılım Mühendisliğine Giriş Ders Notları", Celal Bayar Üniversitesi

Medium: https://medium.com/@m.tahacumurcu

GitHub: https://github.com/mehmettahacumurcu?tab=repositories