**T.C**

**İZMİR BAKIRÇAY ÜNİVERSİTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ FAKÜLTESİ**



**YAZILIM MÜHENDİSLİĞİ TEMELLERİ DERSİ**

**YAZILIM YAŞAM DÖNGÜSÜ MODELLERİ MAKALE ÖDEVİ**

**Umut Can GÜR**

**210601706**

**Mühendislik Fakültesi**

**Doç.Dr.Deniz KILINÇ**

**MART, 2021**

**İÇİNDEKİLER**

1- Yazılım yaşam döngüsü ve yazılım yaşam döngüsünün kısımları

2- Yaşam döngüsü modelleri

3-Scrum yapısı ve scrum yapısının çok kullanılmasının nedenleri

**1- Yazılım yaşam döngüsü ve yazılım yaşam döngüsünün kısımları**

Yazılım yaşam döngüsü, (Software Development Life Cycle SDLC) zaman dilimleri belirlenmiş kısımlardan oluşur. Projeleri bir plan ve düzen doğrultusunda ilerletip hataları minimalize etme gayesi vardır. genellikle 5 aşamadan oluşur. Bunlar aşamalar;

Planlama,

Analiz,

Tasarım,

Gerçekleştirim (Kodlama ve test),

Teslim ve bakım

Bu aşamları daha detaylı bir şekilde inceleyelim:

Planlama: Yazılım yaşam döngüsünün ilk kısmı olan planlama Müşterilerden gereksinimlerin alındığı, Bu projede ne yapmak istiyoruz sorusunun cevabını keşfettiğimiz, projenin fizibilitesinin yani gerçekleşme imkanı olup olmadığını buldukları ve projeyi emin adımlarla nasıl başarıya ulaştırırızın cevabını bulduğumuz yerdir.

Analiz: Projenin son teslim tarihinin belirlendiği, projeyi yaparken ne gibi riskler ile karşılaşılabileceği yani risk durumlarının belirlendiği alandır. Aynı zamanda bu aşama projenin işlevlerinin ve kesin gereksinimlerinin analiz edilip dökümante edildiği kısımdır. Bu kısımda genelde temel UML diagramları kullanılmaya başlanır.

Tasarım: Analiz kısmıda tamamlanıktan sonra geçtiğimiz kısımdır. Bu kısımda belirlediğimiz bilgilerden ve dökümantasyonlardan da yararlanarak “Bu projeyi nasıl elde ederiz?” sorusuna cevap buluruz. Projenin gereksinimleri ve işlevlerinin ayrıntılı olarak incelendiği bu kısımda problemleri daha basit bir hale indirgeyerek önemli kısımların yakalanması sağlanır. Bu problemleri daha basit hale indirgeyebilmek içinde Soyutlama(Abstraction) tekniği kullanılır. Bu soyutlama tekniği problemin daha hızlı çözülebilmesi için bazı durumları görmezden gelip yok sayarak problemlerin çözümlerini kolaylaştırır ve anaproblem üzerinde tam yoğunlukta odak sağlamamıza yardımcı olur.

Gerçekleştirim (Kodlama ve test): Tasarım kısmında istenlen düzeyde ilerleme katedildikten sonra gelinen kısımdır. Bu kısımda artık işin kodlama kısmına geçilir. Proje istenilen işlevler ve kesin gereksinimler ışığında kodlanılmaya başlanır. Kod kısmı olabildiğince anlaşılabilir, akıcı ve değiştirilmeye açık olması, yazılan kodun spagetti kod olamaması gibi hususlar oldukça elzemdir. KISS (Keep it simple) prensibine göre kodunuzu bu işten en az anlayan kişinin bile bir şeyler anlayabileceği şekilde yazmak kodun ne kadar iyi yazıldığını gösterir. Kodlama ve test dediğimiz ikilinin ikincisi olan test kısmı kodlama yapılırken ve kodlama bittikten sonra yapılan test kısımlarına denir. Bu testlerde en az kodlama kadar elzemdir. Özellikle büyük çaplı projeler yapılırken teslim zamanında kodda çıkacak bir hata büyük zaman kayıplarına ve maddi zararlara yol açabilir. Bundan dolayı kodlama yapılırken uygulanan testlerde hatanın erken teşhisi ve düzeltilmesi proje çalışanlarını büyük kayıplardan kurtaracaktır.

Teslim ve bakım: Projenin test kısımlarıda bittikten sonra teslimi için uygun bir versiyonu hazırlanır ve teslime sunulur. Fakat proje teslim edilmesi bütün işin bittiğini göstermez hiçbir yazılım kusursuz olmayacağı gibi sununlan projedeki yazılımda tamamen kusursuz değildir. Zamanla kullanıldıkça ortaya aksaklıkların çıkması veya yetersiz yerlerin oluşması muhtemeldir. Bundan dolayı projenin bakım aşamları vardır. Bu bakımlarda hata düzeltme, projeyi iyileştirip yeni özellikler ekleyerek daha işlevsel hale getirme gibi şeyler yapılmaktadır.

**2.Yaşam döngüsü modelleri:**

Yazılım yaşam döngüsünün temel gayeleri ve yaşam döngüsündeki gerçekleştiridiğimiz adımların genel olarak aynı kalması şartı ile kullanım alanlarına göre daha işlevsel olabilecek farklı yaşam döngüsü modelleri vardır. Yaşam döngüsü modellerinin projelere göre değişkenlik gösteren bu işlevselliğinin kullanılması firmaları büyük yüklerden kurtarabilir. Zaman ilerledikçe ihtiyaçlar ve bu ihtiyaçlara oranla çözümlerde değişecektir. Bu çözümler değiştikçe bunları geliştirme şekillerimizinde değişikliklere uğrayabilmesi normaldir. Doğru model seçimi hem ekonomik hem de zaman açısından büyük avantajlar sağlayabilir. Belli başlı yaşam döngüsü modelleri:

Gelişigüzel Model

Barok Modeli

Çağlayan Modeli

V Süreç Modeli

Helezonik Sprial Model

Artırımsal Geliştirme Süreç Modeli

Kodla ve Düzetl Modeli

**Gelişigüzel Model**

Bu yöntemi bir model olarak değerlendirmek pek doğru sayılmaz. 1960 yıllarında kullanılan bu model tek kişilik üretim ortamlarında görülen ve basit programlar yazmak için kullanılan bir modeldir. Kişiye bağlı yazılım geliştirme şeklinde yapıldığı için izlenebilirliliği de bakım yapılabilirliği de oldukça zordur.

**Barok Modeli:**

1970 li yılların ortalarında ortaya çıkan bu model yazılım yaşam döngüsünün temel adımlarını doğrusal bir biçimde izleyerek ilerlemeyi öngörür. Barok modeli belgelendirmeyi sadece projenin en sonunda yapılmasını ele alır. Bu da barok modelinin zayıf yönlerinden bir tanesidir. Belgelendirmeler sadece projenin en sonunda yer alıdığı için eksiklikler ve gözden kaçan kısımlar oluşmaktadır. O zamanın projeleri için uygun olsada şuanda yapılan büyük projeler için pek doğru bir seçim olmayacaktır.

**Çağlayan Yaşam-Döngü Modeli (Waterfall-Şelale):**

1970 lerin sonlarında ortaya çıkan bir modeldir. Waterfall modeli barok modelinden farklı olarak dökümantasyonları ayrı bir süreç olarak değil üretimin doğal bir parçası olarak ele almaktadır. Fakat waterfall modelininde ayrı hususları bulunmaktadır. Waterfall modeli kullanılırken analiz aşamasındaki tüm detayların tasarıma aktarılabilmesi için müşterilerin ve sistemin en ince ayrıntılarına kadar belirlenmesi gerekir. Waterfull modeli ne kadar kusursuz iş yapmaya odaklı olsada genelde yazılımın kullanıcıya ulaşma süresi çok uzun olduğu için sıkıntılar çıkabilmektedir.

**V Süreç Modeli:**

V nin sol tarafı üretim sağ tarafı sınama test işlemleri olmak üzere yukarıdan aşağı doğru gelen bir “V” yi andırdığı için ismi V süreç modelidir. Her bir sol üretim tarfının sağında kontrol tarafı olduğu için hatalara dönüş daha kolaydır ve hatalar minimalize edilir. Waterfall modelinin bir başka uyarlaması olarak düşünülebilir. Belirsizliklerin az olduğu projelerde waterfall modelide V süreç modelide kullanılabilir.

**Helezonik Model:**

Risk analizleri üzerinde çok duran bir modeldir. Prototipler üretilerek yinemeli bir yaklaşımla ilerlenir. 4 ana kısımdan oluşur bunlar; planlama, risk analizi, kullanıcı değerlendirme ve üretimdir. Bu 4 ana kısım üreticinin isteği kadar tekrarlanır. Hazırlanan prototipler kullanıcıya sunulur ve değerlendirmeler alınır bu değerlendirmelerin sonucunda proje yeniden şekillendirilir. Modelin en güzel özelliklerinden birisi kullanıcıyı da yapım aşaması içine alarak projeyi ilerletmektir. Bu sayede proje içinde kullanıcının isteyeceği ve istemeyeceği durumlar daha yapım aşamasındayken saptanmış olur.

**Arttırımsal Geliştirme Süreç Modeli:**

Arttırımsal geliştirme modelinde projenin tek seferde tesliminden ziyade projenin çalışan bir hali kullanıcılara sunulur. Proje halihazırda kullanılırken yeni versiyonları çıkartılarak projenin üstünü her seferinde eklemeler yapılır ve hatalar minimalize edlimeye çalışılır.Projenin işlevselliği daha çok başlardan anlaşıldığı için projenin tümden batma riski azalır. Bu model üretimi uzun süren fakat eksik gereksinimleri olmasına rağmen kullanılabilir projeler üretmek için ideal bir yöntemdir.

**Kodla ve Düzelt Yaşam Döngü Modeli:**

Bu model genelde küçük çaplı yazılımlar ve programlar için kullanılır. Projeyi direkt olarak gerçekleştirme odaklıdır ve belgelendirme yapılmaz. Bu belge eksikliği yüzünden Kodla ve düzelt ile yapılmış büyük bir yazılımın bakımının yapılması çok zorlaşır.

**3-Scrum Yapısı ve Scrum yapısının çok kullanılmasının nedenleri**

SCRUM agile yaşam döngüsünün alt basamağıdır. Günümüzde en çok kullanılan yazılım yaşam döngüsü olan SCRUM geliştirme için iyi bir takım çalışması ve koordinasyonu sağlar. SCRUM da proje parçalara bölünür parçaların her birine sprint denir ve geliştirme grubunun her üyesi kendi uzmanı olduğu konuya göre sprintleri alır. SCRUM diğer geliştirme modellerine kıyasla en uzun süren olmasa da hayli uzun sürer bunun sebebi ise her basamakta geri dönüşlerin yapılması ve sprintlerin incelenmesidir. SCRUM da 3 temel aşama bulunur. Bunlar roller toplantılar ve bileşenlerdir. SCRUM un en ayırt edici özelliği 2. Aşama olan toplantılar aşamasıdır. Günlük olarak SCRUM meetings yani SCRUM toplantıları olarak adlandırabileceğimiz toplantılar yapılır.Temel aşamlara gelicek olursak ilk temel aşama rollerdir ilk rol ürün sahibi ürünün geliştirilmesini isteyen kişi ile birebir temas halinde bulunan kişidir. Müşterinin ihtiyaçlarını veya değiştirmek istediği şeyleri geliştirme takımına bildiren kişidir. Diğer rol ise SCRUM master yani SCRUM yöneticisidir SCRUM un her aşamasına hakim ve kurallarını iyi bilen birisi olması gerekmektedir.Görevlerine gelecek olursak günlük olarak yapılan toplantılara liderlik eder. İkinci temel aşama ise toplantılardır 3 çeşit toplantı bulunur sprint planlama sprint gözden geçirme ve günlük SCRUM toplantısı. Sprint planlamada projenin gereksinimleri ve sprintlere ayrılması tartışılır. Sprint gözden geçirme ise her bir sprint sonunda yapılır. Sprint istenilen amaca hizmet ediyor mu ? Ya da gelecekte değiştirilebilir mi ? Gibi bir çok sorunun sorulduğu ve tartışıldığı toplantılardır. Günlük sprint toplantıları ise her günün sonunda bugün kim ne yaptı veya yarın ne yapılacak gibi konuların konuşulduğu geliştirme takımı üyelerinin karşılaştığı sorunları takım olarak çözümlemeye çalıştıkları toplantılardır genelde 10 15 dakika arası her gün aynı saat ve yerde yapılır. Üçüncü temel aşama ise bileşenlerdir. Ürün gereksinim dokümanı sprint dokümanı ve sprint kalan zaman grafiği örnek oluşturur. Ürün gereksinim dokümanına gelecek olursak belirli bir kalıbı olmadan yeni gelen veya eskiyen ya da değiştirilen isteklerin bulunduğu dokümandır her zaman ekleme ve çıkarma yapılabilir. Sprint dokümanı ise geliştirilen sprintlerin sprint backloglarının toplanıldığı dokümandır. Son olarak ise sprint zaman grafiği gelmektedir. Sprint zaman grafiğinde kaç ve hangi sprintler ne kadar zamanda yapılmıştır gibi verilerin toplanıldığı projenin hangi aşamaya ne kadar zamanda geldiğinin net olarak görüldüğü grafiktir. Bu sistematiklik ve doğruluk payı sayesinde SCRUM günümüzde en çok kullanılan yazılım yaşam döngüsü olmuştur.

<https://iskulubu.com/yazilim/yazilim-gelistirme-yasam-dongusu/>

https://medium.com/@tunaytoksoz/yazilim-yaşam-döngüsü-sdlc-ve-modelleri̇-c3fe40f6e4e8

Doç. Dr. Deniz KILINÇ Yazılım Mühendisliği Temelleri ders notları

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

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

<https://medium.com/@denizkilinc/yazılım-yaşam-döngüsü-temel-aşamaları-software-development-life-cycle-core-processes-197a4b503696>

https://medium.com/@umutcangurr/1-yazılım-yaşam-döngüsü-ve-yazılım-yaşam-döngüsünün-kısımları-c62abae1c79f