İZMİR BAKIRÇAY ÜNİVERSİTESİ

MÜHENDİSLİK VE MİMARLIK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

YAZILIM MÜHENDİSLİĞİ TEMELLERİ

ÖDEV SUNUMU

BAHAR DÖNEMİ

HAZIRLAYAN

HASAN ÖZER

220601064

Danışman

DR. ÖĞR. ÜYESİ Zekeriya Anıl GÜVEN

İzmir

2023

YAZILIM YAŞAM DÖNGÜ MODELLERİ VE SCRUMUN HAYATIMIZDAKİ YERİ

Yazılım yaşam döngüsü modeli, yazılım geliştirme aşamalarını ve bunların birbirleriyle ilişkisini tanımlayan bir çerçevedir. Seçilebilecek birçok farklı model vardır ve her birinin kendi güçlü ve zayıf yönleri vardır. Yazılım yaşam döngüsü modelleri ve Scrum, yazılım geliştirme sürecinin yönetiminde farklı seçenekler sunar. Her bir modelin kendine özgü avantajları ve dezavantajları vardır ve bir organizasyonun ihtiyaçlarına ve geliştirme ekibinin çalışma tarzına bağlı olarak tercih edilirler. Yazılım da aslında bir üründür ve ürünlerin bir yaşam süreci vardır. Yazılımın yaşam döngüsü tek yönlü veya doğrusal değildir ve birkaç temel aşamadan oluşur. Bu aşamalar şunlardır: Planlama, Analiz, Tasarım, Gerçekleştirme ve Bakım. Yani yazılım projelerinin analiz ve planlanmasından başlayarak kullanımı boyunca süren bir döngüdür diyebiliriz.

İlk önce yazılımın aşamalarından bir bir bahsedelim;

1-Gereksinim (Toplama) Aşaması: Bu aşamada müşterinin istekleri doğrultusunda ortaya çıkarılacak ürünün yapıtaşlarını oluşturacak ayrıntılar ve spesifikasyonlar üzerinde durulur. İstenilen ürünü ortaya çıkarmak için ihtiyaç duyduğumuz bütün araçların bir araya getirilmesi aşaması diyebiliriz. Bir ürünü oluşturmadan önce, ürünle ilgili temel bir anlayış veya bilgi çok önemlidir.

2-Analiz (Analysis) Aşaması: Gereksinim aşamasında elde ettiğimiz verilerin derinlemesine irdelenmesi ve süreç devamında “Ne gibi durumlarla karşılaşabiliriz?”, “Hata ve eksikliklerin giderilmesi konusunda ne gibi yol izleyeceğiz?” “Risk durumlarına yönelik çözüm yönelimlerimiz neler olmalı?” gibi soruların cevaplandığı aşamadır.

3-Tasarım (Design) Aşaması: Bu aşamada gereksinim ve analiz aşamasında toplanan veriler üzerinden girdi sağlanarak sistemi oluşturacak olan yazılım mimarisinin ortaya çıkarıldığı, türetildiği aşamadır.

4-Gerçekleştirim (Kodlama) Aşaması: Tasarım aşamasında ortaya çıkarılan mimari program dilleri kullanılarak makine koduna dönüştürülerek yazılım ürününün tüm bileşenleri bu aşamada uygulanır.

5-Test Etme Aşaması: Bütün aşamalar sonucunda ortaya çıkardığımız ürünün eksiklikleri ve hataları var mı kontrolü sağlanır, “Ürün istenilen sonucu veriyor mu”, “Sistem sunulmaya hazır mı?” gibi soruların işlenmesi sonucu tüm sistem kontrolü ve test aşaması bittikten sonra doğrulama aşaması ile ürün kullanıcıya hazır hale getirilir.

6-Bakım Aşaması: Analiz ve Test etme aşamalarının sonucunda ortaya çıkabilecek potansiyel sorunların çözümünün sağlandığı hata giderme ve geliştirme işlerinin sağlandığı aşamadır.

Yazılımın geliştirilmesi ve onarılması sürecinde birtakım adımlar sonucunda ortaya bazı modeller çıkmıştır. Bu modellerin oluşturduğu topluluğa yaşam döngü modelleri denir. Temel olarak yazılım sektöründe yaygın olarak kullanılan modeller; Yaşam-Döngü Modeli, Kodla ve Düzelt Yaşam-Döngü Modeli, V Süreç Modeli, Helezonik (Spiral) Model, Artımsal Geliştirme Süreç Modeli'dir. Bu yazımda ise bu 4 modeli ele alaraktan bu modellerin yazılım sektörü ve hayatımızdaki yerinden söz edeceğim.

İlk olarak Yaşam-Döngü modelini ele alırsak, geleneksel yazılım geliştirme modeli olarak da bilinir ve bu süreçte gereksinim, analiz, tasarım, test, sürüm ve bakım safhaları yer alır. Bu modelde her safha en az bir tekrar şeklinde geliştirilir. Genelde bu model iyi tanımlanmış ve az zaman zarfında üretimi sağlanan projeler için kullanılır. Her safhada belgelendirme yöntemine başvurulur. Eğer bir safhada belgelendirme ve test aşamaları yapılmamışsa o safha tamamlanmış sayılmaz. Kendi içindeki değişiklikler doğrultusunda teslim aldıklarını bir sonraki safhanın kullanabileceği şekilde değiştirir. Avantajlarından bahsedecek olursak; kullanımı ve anlaması basittir, yönetimi basittir, gereksinimleri iyi tanımlanmış projelerde yapısı daha elverişlidir, tekrarlamalar sonraki ve önceki safhalarla gerçekleşir. Bunun yanında dezavantajlı olduğu durumlar ise; yazılımın son kullanıcıya ulaşması uzun süreç gerektirir, karmaşık ve uzun süreçli projeler için elverişsizdir, yanlışların ve eksiklerin düzeltilmesi için gereken maliyet çok fazladır, gelişim ve değişime açık olmadığından müşteri memnuniyeti sağlanamaz, kullanıcıyı süreç içerisine dahil etmez.

Kodla ve Düzelt Yaşam-Döngü Modelini açıklamak istersek; yazılım geliştirme sürecinde kullanılan birkaç yüz satır kod ile direkt olarak yazılım ürününün geliştirdiği modeldir ve diğer modellere kıyasla yazılım geliştirmenin en kolay ama en pahallı olduğu, tecrübe gerektirmeyen, küçük firmaların projelerinde kullanılır. Belgelendirme yok, bakım zor, bunun dışında emeklilik safhasını (retirement) içerir. Avantajları; herhangi bir planlamaya ihtiyaç duyulmaz, safhalar üzerinde fazla zaman harcanmaz, 7'den 70'e herkes bu modeli uygulayabilir. Dezavantajları; tekrarlamalar yoktur, yönetim projenin bitiş zamanı belli değildir, hatalar ve eksiklikleri düzeltmek çok zor(unflexible) ve maliyetlidir.

Bir sonraki modelimiz ise V Süreç Modeli; Bir bakıma Modelinin geliştirilmiş halidir, fakat safhalar doğrusal şekilde değil "V" şeklinde ilerler. V harfinin sol eğimli kısmında üretim(production) sağ eğimli kısmında ise sınama (test) aşamaları yer alır. Yine modelde uygulandığı gibi iyi tanımlanmış projeler için elverişlidir. Temelde 3 adım uygulanır Kullanıcı Modeli, Mimari Model, Gerçekleştirim Modeli ile istenilen sonuca ulaşılır. Kullanıcı Modelinde; Kullanıcının istekleri doğrultusunda sistem hazırlanır, sınama belirtimleri ve projede uygulanacak programlar bu adımda belirlenir. Mimari Modelde ise sistem tasarımı, alt sistem ve tüm sistemde sınama işlemleri yapılır. Son olarak gerçekleştirim modelinde

kodlama ve bunların sınanmasına ilişkin işlemler yapılır. Avantajları; verification ve validation planları aşamalarda ele alınır, verification ve validation planları son ürün odaklı değil tüm ürün çıktılarında uygulanır, proje yönetim ve takibi kolaydır, kullanımı elverişlidir, kullanıcının projeye olan katkısını arttırır. Dezavantajları; safhalar arasında yinelemeler kullanılmamaktadır, riske dayalı çözüm aktivitelerini içermez, gereksinimler süreç içerisinde ve devamında değişkenlik gösterebilir, aynı anda gerçekleştirilecek olan olaylara kolay imkân vermez.

Yazımda bahsedeceğim son döngü modeli ise Artımsal Geliştirme Süreç Modeli. Bu model yazımızda bahsettiğimiz modellere göre daha kapsamlı bir model olmakla beraber diğer modellerden farklı bir yaklaşım türü olan Divide and Conquer (Böl ve Yönet) metodunu kullanmaktadır. Bu modelin temel fikri, yazılımın birkaç yineleme veya döngü ile geliştirilmesidir. Her yinelemede, bir önceki yinelemede oluşturulan parçaların üzerine eklemeler yapılır ve yeni bir artımlı versiyon oluşturulur. Bu artımlar, yazılımın tamamlanmasına kadar sürekli olarak geliştirilir ve düzenlenir.

Modelin temel esası yazılım ürününü tek seferde teslim etmektense geliştirme ve teslim parçalara bölünmüş vaziyettedir. Yani kullanıcı ürünün daha yapılış aşamasındayken bile parça parça teslim edildiğinden kullanıcının geri dönütüyle beraber işlevsellik erken aşamalarda saptanabilir.

- Her bir teslim beklenen işlevselliği az miktarda olsa içerir.
- Kullanıcı *gereksinimleri* ön planda tutulur.
- İlk parçadan itibaren geliştirilmeye başlanan bütün ürünlerin gereksinimleri *dondurulur*.
- Değişimin istendiği durumlar sonraki teslimlerde ele alınır.
- Geliştirilen yazılım parçalarının her sürümü bir sonraki parçayı kapsayacak şekilde ve gelişmeye devam edecek şekilde işlevselleştirilir.
- Bir yandan kullanım, öteki yandan üretim işlemleri gerçekleşir.
- Uzun süreçli projeler ve parça parçada olsa işlevsellikle çalışan projelerde bu model kullanışlıdır.

Modelin en büyük avantajı, yazılımın geliştirilmesinin sürekli olarak müşteri geri bildirimine dayalı olarak yapılabilmesidir. Bu nedenle, müşterilerin ihtiyaçlarına daha uygun yazılım geliştirilmesi mümkündür. Ayrıca Artımlı Model, geliştirme maliyetlerini azaltır ve hataların erken tespit edilmesine yardımcı olur. Artımlı Model, yazılım geliştirme sürecinde daha fazla esneklik sağlar ve müşteri ihtiyaçlarına daha uygun yazılımın geliştirilmesine yardımcı olur. Ancak, bu modelin dezavantajları da vardır ve projenin başarısı, her artımın başarısıyla doğrudan ilişkilidir.

Avantajları; Sistem parça parça oluşturulduğundan bütün sistemin başarısız olma riski azdır, sürekli teslim, kullanım ve üretim gerçekleştiğinden daha fazla sınanma imkânı vardır, prototip gibi davranarak gereksinimlerin iyi anlaşılmasını sağlar.

Dezavantajları; Gereksinimler doğru parçalara artımlanmayabilir, kendi içerisinde artımlar tekrarlanmaz, her bir artımlar tanımlanırken tüm sistem tanımlanmalıdır.

Ayrıca Çağlayan, V-Model ve Spiral modelleri "geleneksel" yazılım geliştirme yaşam döngü modelleri olarak adlandırılırken, Agile(Çevik) yöntemler gibi ayrıca yazımın önemli konularından biri olan Scrum daha modern ve esnek olarak kabul edilir. Geleneksel modeller ile Agile yöntemler arasındaki temel fark, gereksinimler ve değişikliklere yaklaşımlarıdır. Geleneksel modeller, önceden planlama ve belgeleme üzerinde dururken, Agile yöntemler uyum sağlama ve geliştirme sürecindeki değişikliklere yanıt verme önceliği verir.

Çağlayan ve V-Model'de her aşama tamamlanmadan bir sonraki aşamaya geçilemez. Bu, sonraki aşamalarda yapılan herhangi bir değişikliğin uygulanması zaman alıcı ve maliyetli olabileceği anlamına gelir. Spiral modeli, her döngüde risk analizi ve değişikliklere izin vererek bunu hafifletmeye çalışır, ancak Agile yöntemlerden daha yavaş ve daha fazla kaynak gerektirebilir.

Bu yazımda ele alacağımız bir başka konu ise günümüzde popüler olan Scrum`dır.
Scrum yaklaşımı en popüler Agile yöntemidir.

Scrum Agile Yöntemi, yazılım geliştirme sürecinde sıklıkla kullanılan bir yöntemdir. Bu yöntem, yazılım geliştirme sürecini küçük, bağımsız ekiplerle çalışarak daha verimli hale getirmeyi hedefler.

Scrum, tipik olarak "Sprint" adı verilen 2-4 haftalık zaman dilimleri içinde çalışır. Her Sprint'te, ekip belirli bir hedefe ulaşmak için çalışır. Sprint sonunda, ekip üyeleri arasında bir gözden geçirme yapılır ve bir sonraki Sprint için plan yapılır.

Scrum Agile Yöntemi, diğer Agile yöntemlerinden farklı olarak, belirli rol ve süreçler kullanır. Bu süreçler arasında, Product Backlog, Sprint Backlog ve Daily Scrum yer alır. Bu süreçlerin amaçları, ekibin birlikte çalışmasını ve hedeflere ulaşmasını kolaylaştırmaktır.

Scrum Agile Yöntemi'nin en büyük avantajı, yazılım geliştirme sürecini daha esnek ve etkili hale getirmesidir. Bu yöntem, projenin herhangi bir aşamasında değişiklik yapılabilmesine olanak tanır ve müşterilerin ihtiyaçlarına daha uygun yazılımın geliştirilmesine yardımcı olur.

Ayrıca, Scrum Agile Yöntemi, yazılım geliştirme sürecini daha öngörülebilir hale getirir ve hedeflere ulaşmak için daha etkili bir planlama sağlar. Bu nedenle, birçok şirket ve organizasyon-Yaklaşık %84-Scrum Agile Yöntemi'ni tercih etmektedir.

Ancak, Scrum Agile Yöntemi de bazı dezavantajlara sahiptir. Bu yöntem, belirli bir ekip ve süreçler gerektirdiği için, daha küçük ölçekli projeler için uygun değildir. Ayrıca, sürekli bir iletişim gerektirdiği için, bazı ekipler için zaman ve kaynak açısından zorlayıcı olabilir.

Sonuç olarak, Scrum Agile Yöntemi, yazılım geliştirme sürecinde daha fazla esneklik ve etkinliğe olanak tanır. Bu nedenle, birçok şirket ve organizasyon bu yöntemi tercih etmektedir. Ancak, bu yöntem de bazı dezavantajlara sahiptir ve her proje için uygun olmayabilir.

Kaynakça:

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

https://medium.com/@ardaodabasi/yazilim-yaşam-döngü-modelleri-ve-scrum-1267fd869641

https://phoenixnap.com/blog/software-development-life-cycle

https://tr.myservername.com/sdlc-phases

https://medium.com/@alminaecemeskicindil/yazılım-yaşam-döngü-modelleri-c0e5b5404ce3

https://www.linkedin.com/pulse/yazılımın-yaşam-döngüsü-beyza-nur-karakoç?trk=pulse-article&originalSubdomain=tr