Yazılım Yaşam Döngüsü :

Yazılım da aslında hayatımızdaki birçok şey gibi aslında birer üründür. Söz konusu bu ürünlerin de tabii ki birer yaşam süreci vardır. Geliştirilen bir yazılım projesinin de en başta planlamasından başlayarak ilgili kişiler ya da sistemler tarafından kullanılmaya hazır hale gelene kadar geçirdiği tüm aşamalara ve bu aşamaların toplamından meydana gelen döngüye Yazılım Geliştirme Yaşam Döngüsü denir. Oluşturacağımız yazılımın sadece kodlamaktan oluştuğu yanılgısına düşülebilir ama tabii ki de böyle değil. Basitçe ifade etmek istersek çalışılan proje ortaya çıkartılırken planlama, analiz, tasarım, üretim en son da test aşamaları yer almaktadır ve alması bir gerekliliktir. Bu aşamaları bir defa takip ettik sonucunda yazılım ürünümüz ortaya çıktı mı? Hayır. Bu aşamaların geliştirilmeye çalışılan projeye göre farklılık göstermekle birlikte birden fazla sefer tekrar edilmesi gerekir ve hatta ürün ortaya çıktıktan sonra da gelecek yeni istekler, meydana gelen hataların düzeltilmesi, projeye eklenecek yeni modüller gibi konular için proje kullanıldığı sürece döngü süreci de devam eder. Söz konusu proje tamamlandığı daha önce bahsettiğimiz planlama, analiz, tasarım, üretim ve test aşamalarının tamamının tamamlandığını ve proje içerisinde hiçbir hata ya da istek olmadığı senaryoda devreye projenin bakım aşaması girmektedir. Geliştirilen bütün yazılım projelerinin ilerleyen zamanlarda doğabilecek hataları, yeni istekleri ve yeni talepleri için bakım sürecinin de yazılım geliştirme yaşam döngüsünde muhakkak yer alması gerekmekte. Yazılım geliştirme yaşam döngüsünün aşamalarına daha dikkatli bir şekilde bakarsak ilk aşamamız planlama.

1-Planlama : Yazılım yaşam döngüsünün başladığı noktadır. Projenin temel ihtiyaçlarının tespit edildiği, proje için gerekli fizibilite çalışmalarının yapıldığı, projenin planlamasının oluşturulduğu ve görev dağılımı yapılır. Projenin omurgasının oluşturulduğu aşama denebilir.

2-Analiz : Yazılım yaşam döngüsünün en önemli aşamalarından biri olan analiz aşamasında projenin ne kadar zamanda tamamlanacağı ve ne tür riskler barındırdığının belirlendiği aşamadır. Elde edilen bilgilerin özel bir format biçiminde doküman haline getirilmesidir.

3-Tasarım : Bu aşamada “İstediğimizi nasıl elde edeceğiz?” sorusuna yanıt aranır. Analiz aşaması sonrasında ortaya çıkartılan proje detayları göz önüne alınarak proje içerisinde atılacak adımlar teker teker belirlenir ve proje planı ortaya çıkarılır. Proje planının yanında tasarım raporu da oluşturulur. Tasarım dökümanında projeye dair amacı, kapsamı, fikri gibi bilgileri, sistem tasarım bilgileri, tasarımın ufak ayrıntıları, veri modeli olmalıdır. Tasarım raporunun amacı yazılımı geliştirecek olanların yazılımı geliştirirken nelere dikkat edeceği ve ürünün ortaya çıkartılması sürecinde de sonrasında da projeye dahil olacak yazılımcı insanların projeyi daha kolay anlayıp sürece dahil olabileceği bir rapora sahip olmasıdır.

4-Üretim : Planlama, analiz ve tasarım aşamalarından sonra ilerleyişi detaylı şekilde planlanan projenin geliştirme aşamasıdır. Bu aşamadan itibaren hiçbir analiz işlemi yapılmamalı, daha önceden oluşturulan planlara sadık kalınmalı.

5-Test : Üzerine çalışılan yazılım tamamlandıktan sonra müşterilere sunulmadan önce bunun için görevli test ekibi tarafından gerekli testlerin gerçekleştirildiği aşamadır. Bu aşama sırasında hatalar giderilip yazılımın çalıştığından emin olunur.

6-Bakım : Yazılım ürünü, döngünün bütün aşamaları tamamlanıp sahaya çıkartıldıktan sonra başlar. Proje yayınlandıktan sonra ortaya çıkabilecek hataların giderilmesi, yazılımın daha da geliştirilmesi ve yeni becerilerin eklenmesi sürecidir. Kullanıcılardan gelen geri dönüşler yardımı ile hatalar ve iyileştirmeler tespit edilir. Yazılım yaşam döngüsünde çok sayıda model bulunur. Neden sadece bir değil de daha fazla model var? Yazılım projeleri oldukça büyük olabilir ya da bu projeyi kullanacak kişilere göre farklılıklar gösterebilir.

Yazılım Yaşam Döngü Modelleri :

1-Gelişigüzel Model : Hiçbir biçimde bir kurala ya da formata sahip değildir. Tamamen yazılımın geliştiricisine bağımlı bir modeldir. Hatta düzensizliği ve kuralsızlığı dolayısıyla kendi geliştiricisi tarafından bile anlaşılamayıp gelişimi sekteye uğrayabilir. İzlenebilirliği ve bakımı son derece zordur. Daha çok 60'lı yıllarda tek kişilik üretimlerde kullanılan basit bir geliştirme modelidir. Günümüzde daha çok öğrenci projelerinde rastlanır.

2-Barok Modeli : Geliştirim aşamasına daha fazla odaklanan bir modeldir. Yaşam döngüsü adımlarını doğrusal bir biçimde gerçekleştirir. Belgelemeyi günümüzden farklı olarak ayrı bir süreç olarak uygular ve testlerden sonra yapılmasını uygun görür. Günümüzde ise belgeleme işlemi yapılan işin bir parçasıdır. Aşamalar arasında geriye dönüşler yapmak gerektiğinde bunun nasıl gerçekleştirileceğine dair bir rotası yoktur. Daha çok 70'li yıllarda kullanılmıştır. Gelişigüzel model gibi barok modeli de güncel bir yaşam döngü modeli olarak görülmezler.

3-Çağlayan (Şelale) Modeli : Bu model diğer güncel yaşam döngü modellerinin temeli denilebilir. 70'li yılların ortalarında yapısal programlama ile kullanılmaya başlanan geleneksel bir modeldir ve geçmişti oldukça popüler olduğunu söyleyebiliriz ancak şu anda kullanımı yok denecek kadar azdır. Çağlayan modelinde yazılım yaşam döngüsü aşamalarının her biri en az bir kere izlenir. Daha çok iyi tanımlanmış ve kısa sürede tamamlanabilecek projelerde faydalanılır. Günümüzde genellikle yazılımların hazırlanması ve kullanıma geçmesi için uzun süreler gerektirmektedir. Çağlayan modeli bu uzun süreli hazırlık açısından uygun değildir. Kullanıcılar yazılımın geliştirildiği aşamanın içinde değildir bundan dolayı oluşabilecek problemler çok geç fark edilebilir, bu hem yazılım maliyetini oldukça artırırken hem de sorunların çözümlerini zorlaştırır. Kodlamanın modelin küçük bir kısmında yer alması yazılımcıları huzursuz edebilir çünkü yaptıkları iş miktarı ile projenin toplam süresi orantısız durumdadır ayrıca uzun süren proje, projeyi finanse eden yöneticileri de rahatsız eder. Yöneticiler tamamlanmayacak bir iş için boşa servet harcadıklarını düşünebilir. Günümzde kullanımı gittikçe azalmaktadır.

4-V Süreç Modeli : V modeli çağlayan yönteminin geliştirilmiş hali denilebilir. Bu modelden de çağlayan modeli gibi işin tanımının yüksek seviyede yapıldığı, belirsizliklerin az olduğu projelerde faydalanılır. Bilgi teknolojileri alanında kullanılmak üzere geliştirilen projeler örnek verilebilir. Model v harfine benzetildiğinden v harfinin sol tarafı üretim safhalarını, sağ tarafı ise test safhalarını içerir. V süreç modeli esas olarak 3 modelden oluşmaktadır. Bunlar Kullanıcı Modeli, Mimari Model ve Gerçekleştirim Modelidir.

a)Kullanıcı Modeli : Ürünün ortaya çıkması aşamasında kullanıcı ile olan ilişkinin tanımını yapar. Sistemin nasıl kabul edileceğine dair test belirtimleri ve planları belirlenmektedir.

b)Mimari Model : Sistemin tasarımı ve meydana gelecek alt sistem ile tüm sistemin test işlemlerine ilişkin işlevler.

c)Gerçekleştirim Modeli : Söz konusu yazılımın kodlamasının yapılması ve test edilmesi üzerine fonksiyonlar içerir.

Avantajları : Model kullanıcıların birçok geri dönüt vermelerinden dolayı projeye katkılarını artırır. Proje yönetimi ve takibi kolaydır. BT projeleri için oldukça uygun bir modeldir.

Dezavantajları : Bu modelin aşamaları arasında tekrarlar bulunmaz ve risk çözümleme için ayrılan bir yer yoktur. İşin ve ürünlerin ihtiyaçları değişkenlik gösterebilir.

5-Helezonik Spiral Model : Helezonik modeli diğer modellerden ayıran ise risk analizinin ön planda olması ve prototip oluşturmasıdır. Risk analizi ön planda olduğu için oluşacak problemleri erken safhada çözüme kavuşturmak mümkün olabilir. Prototip oluşturmanın da her aşamada olması kullanıcının da her aşamada yazılımı yapılan projenin bir parçasını görmesini sağlar bu da oluşacak hataların miktarını önemli ölçüde düşürür. Projenin maksadı, hedefi, alternatifleri ve ne gibi kısıtlamalarının olacağı belirlenir. Yinelemeli artımsal bir yapı benimsenmiştir. 4 temel aşamadan oluşur, bunlar:

a)Planlama : Geliştirilecek ara ürün için planlama, amaç belirleme, önceki adımda geliştirilen ara ürün ile birleştirme.

b)Risk Analizi : Risklerin araştırılması, tespit edilmesi ve çözüme kavuşturulması.

c)Üretim : Ara ürünlerin üretilmesi

d)Kullanıcı Değerlendirme : Oluşan ara ürünlerin kullanıcılara denetilerek kullanıcılardan alınacak geri beslemelerin değerlendirilerek diğer aşamaya geçilmesi.

Helezonik modelin dezavantajı küçük, düşük riskli, düşük maliyetli projeler için çok pahalı bir sistem oluşturur. Karmaşık bir içeriği vardır. Projenin tamamlanması uzun sürebilir ve çok fazla dokümantasyon ortaya çıkabilir. Kontrat tabanlı yazılıma uymaz. Kişilerin kendi deneyimlerine bağımlı bir risk denetimi vardır.

6-Evrimsel Model : İlk tam ölçekli modeldir. Genelde coğrafi açıdan çok geniş coğrafyalara yayılmış (bankalar vs.) organizasyonlarda faydalanılır. Her aşamada ortaya çıkan ürünler üretildikleri alan için tam işlevselliği içermektedir. Pilot uygulama durumunda sistemi kullan, sına, güncelle ve diğer birimlere taşı prensibi geçerlidir. Modelin başarısı için ilk evrimin de başarılı olması gerekir.

Avantajları : Kullanıcılar bu model sayesinde kendi ihtiyaçlarını daha iyi algılar. Devamlı olarak gerçekleşen değerlendirme erken safhalarda geliştirme risklerini ve problemleri azaltır.

Dezavantajları : Sürecin görünürlük oranı düşüktür. Bakım yapması son derece zordur.

7-Scrum : Ekiplerin kendi kendilerini organize edebilmek amacı ile faydalandığı, ortak bir hedefe yönelik çalışabilmesi için kullandığı bir yönetim çerçevesidir. Maliyet, emek ve zaman açısından verimli bir proje teslimi için gereken toplantı, araç ve rolü tanımlar. Scrum uygulaması yazılımcıların kişisel deneyimlerinden öğrenmesine, kendi içlerinde kendilerini yönetmelerine ve oluşacak değişimlere adapte olmalarını kolaylaştırır. Günümüzde yazılım ekipleri çoğunlukla scrum kullanır. Scrum metodolojisi bir takım ilkeler içerir. Bunlar :

a)Şeffaflık : Yazılımcılar birbirlerinin karşılaşabileceği zorluk ve problemlerden haberdar olabileceği bir ortamda çalışır. Farklı alan ve görevlerde çalışan kişilerin düzenli iletişimi yanlış anlaşılmaları önler.

b)Gözlem : Ekip üyelerinin devamlı kendi ilerlemelerini göz önünde bulundurabilmeleri için çerçeveye ufak aralıklarla değerlendirme noktaları yerleştirilmiştir. Proje yöneticileri bu değerlendirmelerde gelecek planlarını hazırlar ve öngörüler edinir. Proje daha düzenli ve düşük maliyetli olarak ilerleyebilir.

c)Adaptasyon : Yazılım ekip üyeleri müşteri ihtiyaçlarına bağlı olarak h.angi iş ve görevlerin önce halledileceğine, incelenileceğine karar verebilir.

Scrum ekipleri 5 temel değeri izler. Bunlar :

a)Bağlılık : Ekipler her daim zamana dayalı görev ve hedeflere bağlıdır . En doğru çözümü bulabilmek için devamlı gelişmeye odaklanır.

b)Cesaret : Scrum ekipleri net, zorlayıcı sorular sorarak cesaret örneği gösterir. En iyi çözüme ulaşmak amacı ile dürüst ve şeffaf tartışmalar yapar.

c)Odaklanma : Ekip üyeleri sınırlı zamanda özellikle belirledikleri özel görevlere odaklanırlar.

d)Açıklık : Scrum ekip üyeleri kişisel gelişimleri ve ekip gelişimi açısından proje kalitesini destekleyen yeni fikir ve fırsatlara açıktır.

e)Saygı : Ekip üyeleri birbirlerine, yöneticilere ve scrum sürecine net bir saygı duyarlar. Bu saygının sonucu olarakta yardımlaşmanın bol olduğu çalışma ruhu yüksek bir ortam elde edilir.

Scrum nasıl çalışır ?

Scrum öğrenmesi basit ama uzman haline gelmesi zorlu bir süreçtir. Scrum temel kavramları Scrum kılavuzunda açıklanmıştır. Bu kılavuz scrum sürecine dair ayrıntılı bir bakış sunar.

Scrum Yapıtları Nelerdir :

a)Ürün iş listesi : Ürün iş listesi, projenin başarılı olabilmesi adına gereken özellik, ihtiyaç, iyileştirme ve düzeltmelerden oluşan değişken bir listedir. Temelde sürekli olarak

incelenen ve pazarda yaşanacak değişikliklere göre uyacak şekilde tekrar önceliklendirilen bir yapılacaklar listesidir.

b)Sprint iş listesi : Sprint iş listesi, mevcut Sprint döngüsünde yazılım ekibinin tamamlaması gereken öğelerin listesidir. Her sprint'ten önce yazılım ekipi, ürün iş listesine bakar sonrasında üzerine uğraşacağı kısımları seçer. Sprint iş listesi esnektir, değişikliklere uğrayabilir ve sprint sırasında gelişebilir.

c) Ürün parçası : Ürün parçası, bir hedef veya vizyon doğrultusunda bir adımdır. Bir sprint'in kullanılabilir son ürünü budur. Ekipler, sprint hedeflerini tanımlamak ve bulmak için çeşitli yöntemlere başvurabilir. Esnekliğe rağmen, ekibin halihazırda var olan sprinte göre ulaşmak istediği asıl sprint hedefinden ödün verilemez.

Scrum rolleri :

Bir Scrum ekibi üç role ihtiyaç duyar. Bunlar :

Ürün Sahibi : Ürün sahibi, üretim ekibinin işletmeye en iyi değerleri hazırladığından emin olmaya çalışır. Kullanıcıların ve müşterilerin değişen gereksinimlerini anlar ve öncelik sırasını ona göre değiştirir. Etkili ürün sahipleri şunlara önem verir :

a)Geliştirme ekibine sonraki aşamada hangi özelliklerin teslim edileceğiyle ilgili açık rehberlik sunar.

b)İşletmenin istekleri ile gelişim ekibinin anladıkları arasında oluşabilecek farklılıkları giderir.

c)Sürümlerin zaman ve sıklığına dair karar verir.

Scrum lideri : Scrum ekibinin çalışma verimliliğini kontrol etmek, Scrum liderlerinin görevidir. Scrum süreçlerini optimize etmek ve teslimatı iyileştirmek için ekiplere, ürün sahiplerine ve işletmeye destek sağlarlar. Scrum liderleri, şunları yapmaktan da sorumludur:

a)söz konusu Sprintler için gereken kaynakların planını yapma.

b)Diğer Sprint uygulamalarını ve ekip toplantılarını organize etme.

c)Ekip dahilinde dijital evrim ve adaptasyona yön verme.

d)Yeni teknolojileri bünyesine katarken ekibin kendini geliştirmesine imkan sağlama.

e)Geliştirme ekibin tamamının karşılaşabileceği zorlukları çözmek için dış gruplarla iletişim kurma.

Scrum geliştirme ekibi : Scrum ekibi sınayıcılar, tasarımcılar, kullanıcı tecrübesi yetkilileri, operasyon mühendisleri ve ürün geliştiricilerinden oluşur. Ekipin her bir parçası farklı

yetenek ve deneyimlere sahiptir ve birbirlerine yardımcı olurlar. Dolayısıyla, iş tamamlanıp teslim aşamasına geldiğinde herhangi birisi sorun yaşamaz.

Scrum geliştirme ekipleri şunları yapar :

a)Sprint'in hatasız bir şekilde hazır hale gelmesi için iş birliği ve yardımlaşma içirişinde çalışma.

b)Sürdürülebilir geliştirme uygulamalarına destek çıkmak.

c)Kendi içersinde kendisini organize etme.

d)Her bir Sprint için ne kadar iş hazırlanabileceği konusunda organizasyon yapma.

Scrum Yazılım Alanında Niçin Değerli ve Günümüzde Neden Popüler ?

Her türden ekip, scrum'ı etkili şekilde kullanabilir. Ancak daha çok yazılım geliştiren ekipler ve mühendislik ekipleri arasında yaygındır. Ekiplerin, maliyet ve bütçenin kontrol altında tutularak değişen ihtiyaçlara daha çabuk şekilde karşılık vermesine olanak tanır. Kalite güvencesi testleri, scrum içerisinde temel olarak bulunur. Ekipler, ihtiyaçlarını tüm Sprintlerin başında tanımlarını yapar. Ekipler, "tamamlanmış süreç" kavramına ilişkin ekip vizyonu oluştururken yazılım geliştirme yaşam döngüsünü de bütün olarak değerlendirir. Bu da ihtiyaçların kısa bir süre boyunca alakalı ve ulaşılabilir olması anlamına gelir. Ürün sahibinin düzenli geri dönütleri ve sprint yorumları, ekibin proje sürecinde devamlı olarak gelişmesine ve değişmesine yardımcı olur. scrum ekipleri, ihtiyaçları müşteri değerine ve risk analizine göre değişen bir sıralama yapar. Odaklandığı nokta, erken müşteri geri dönütünü alabilmek için piyasaya gönderilebilecek çalışan bir birincil ürün ortaya çıkartmaktır. Scrum geliştirme düşük maliyetli, geliştirme ekibi verimliliği ve finansal tasarruflar ile çabuk problem tespit eden yaklaşımıyla bilinir. Kendi içerisinde kendini yöneten ve organize eden ekipler, üyelerin daha yaratıcı ve yenilikçi fikirlere sahip olmasına müsaittir. Ekip üyeleri işlerini çalışma tarzlarına, karakterlerine ve hayattaki hedeflerine göre şekillendirme esnekliğine sahiptir. Farklı fonksiyonlarla birlikte çalışmak, ekip üyelerinin yeni yetenek ve beceriler edinmesine ve birbirine destek olmasını sağlar.

İSİM : Ahmet Ayık No : 220601020

Medya Bağlantıları :

LinkedIn : https://www.linkedin.com/in/ahmet-ayık-3baa70245/

Github : https://github.com/ahmetayikk?tab=repositories

Medium : https://medium.com/@ahmetayikk03/yazılım-yaşam-döngüsü-f321ff26362d