# Yazılım Geliştirme Süreci ve Mühendisliği

Yazılım yaşam döngüsü, bir yazılım ürününün geliştirilmesi, test edilmesi, dağıtılması, kullanılması ve bakımı sürecindeki adımların bütünüdür. Bu süreç, yazılım mühendisliğinde önemli bir kavramdır ve yazılım geliştirme ekipleri tarafından kullanılan yöntemler ve teknikler değişebilir. (Yazılım yaşam döngüsü modelleri, bir yazılım ürününün geliştirilmesinde kullanılan farklı stratejileri ve adımları tanımlar. Bu modeller, geliştirme ekibine bir yol haritası sunar ve projenin başarılı bir şekilde tamamlanması için gereken adımları belirler.) Bu makalede, yazılım yaşam döngüsü modellerinin temelleri ve yaygın kullanılan modeller hakkında bilgi vereceğim.

Yazılım yaşam döngüsü modelleri, yazılım geliştirme sürecinde takip edilecek adımları belirleyen çerçevelerdir. Bu modeller, yazılım geliştirme ekiplerinin işbirliği yapmasını ve projelerini başarılı bir şekilde tamamlamasını sağlar.

Yazılım yaşam döngüsü modeli içerisindeki aşama sayıları ve nitelikleri kullanılan modele göre değişir fakat genel olarak aşamalar aşağıdaki gibidir:

Gereksinim: Yazılımın amacını, işlevlerini ve kullanıcı ihtiyaçlarını belirlemek için kullanılır. Bu aşamada müşteri, kullanıcılar ve diğer ilgili taraflarla yapılan görüşmelerle bu gereksinimler belgelenir.

Analiz: Analiz aşaması, gereksinimlerin belirlenmesinden daha detaylı bir şekilde incelenerek, yazılımın hangi işlevleri yerine getireceği, hangi işlemlerin nasıl yapılacağı ve hangi verilerin nasıl işleneceği gibi konulara odaklanır. Bu aşamanın sonucu şartname dokümanı olarak yazılır ve yazılım proje yönetim planı oluşturulur.

Tasarım: Belirlenen gereksinim ve bu gereksinimlere yapılan analizlere göre yazılım ürününün tasarımı yapılır. Bu aşamada iki çeşit yazılım yapılır. Üst seviye ve mimari tasarımla kabataslak planlar ve modüller, ayrıntılı tasarımla da veritabanı tasarımı, kullanılacak algoritmalar ve etkileşim diyagramları tasarlanır. Tasarım aşamasının sonunda bütün kararların alınmış olması beklenir.

Gerçekleştirme ve Test: Tasarım aşamasında oluşturulan her bir modül için ayrı kod yazılır ve her bir modül ayrı bir şekilde test edilir. Ardından modülleri birleştirirken sorun çıkabileceğinden yazılım bütün olarak da test edilir. Ortaya çıkan sisteme gerçek veriyi kullanarak da test yapılır. En sonunda müşteri tarafından kabul testi yapılır.

Teslim Sonrası Bakım: Ürün teslim edildikten sonra yapılan herhangi bir değişikliği kapsar. Düzeltici bakım ve mükemmelleştirici bakımla uyarlanabilir bakımı da içeren özelliklerin arttırılması bu aşamanın iki çeşididir.

Emeklilik: Bu aşamada yazılım vadesini doldurduğunda kullanımı durdurulur.

Yazılım yaşam döngüsü modelleri Çağlayan, V Süreç, Artımsal Gelişme, Spiral, Kodla ve Düzelt Modelleri şeklinde listelenebilir. Bunlara artık günümüzde kullanılmayan Gelişigüzel ve Barok Modelleri de eklenebilir. Ben bu yazımda listede belirttiğim ilk üç modeli aktaracağım.

Çağlayan Modeli: Gereksinimi iyi anlaşılmış, üretimi az zaman gerektiren projeler için uygundur. Karmaşık ve nesne yönelimli projeler için uygun değildir. Kullanımı ve yönetimi kolaydır. Müşteriler ve son kullanıcılar tarafından da anlaşılması kolaydır. İşler aşama aşama gerçekleştirilir, bir aşama bitmeden diğerine geçilmez ve her aşama birer kez tekrarlanır. Tekrarlamalar önceki ve sonraki adımlarla gerçekleşir. Her aşamanın, her detayın dokümantasyonunun olması gerekir. Herhangi bir aşamada dokümantasyon ve test olmamışsa o aşama tamamlanmamış sayılır. Tasarım aşaması detaylı bir çalışma gerektirir. Analizdeki tüm detaylar tasarıma ayrıntılı bir şekilde yansıtılamazsa düzeltilme maliyeti çok yüksek olur. Kullanıcı sürecin içerisinde yer almaz ve genelde yazılımın son kullanıcıya ulaşması zordur.Günümüzde kullanımı gittikçe azalmaktadır.

V Süreç Modeli: Belirsizliklerin az, iş tanımlarının belirgin olduğu projelere uygundur. Model, adını kodlama aşamasından tasarım aşamasına ve sonra da geliştirme aşamasına yukarıya doğru eğimli bir şekilde çıkmasından almıştır. Sol taraf üretim sağ taraf sınama bölümüdür. Çıktıları en altta gerçekleştirim modelinde modüllerin kodlanması ve sınanması, bir üstündeki mimari modelde sistem ve alt sistem tanımı ve tüm sistemin sınanması, en üstte geliştirme sürecinin kullanıcıyla ilişkisinin tanımlanması ve sisteme ilişkin sınama belirtimleri ve planları şeklindedir. Modelin kullanımı kolaydır, proje yönetimi takibi de kolay olur. Kullanıcının projeye dahil olmasını sağlar. Doğrulama ve onaylama planları erken aşamalarda vurgulanır ve sadece son üründe değil tüm teslim edilebilir ürünlerde uygulanır fakat bu modelin risk çözümleme aktiviteleri yoktur. Aşamalar arasında da tekrarlamalar yoktur.
Artımsal Geliştirme Modeli: Uzun zaman alabilecek ve sistemin eksik işlevsellikle çalışabileceği türdeki projeler için uygundur. Yazılımın küçük parçalara bölünerek, her bir parçanın ayrı ayrı geliştirilmesi ve test edilmesi ile ilerleyen bir böl ve yönet yaklaşımı modelidir. Bu yaklaşım sayesinde en önemli sistem özellikleri daha fazla sınanma imkanı bulmuş olur ve tüm projenin başarısız olma riskini azaltır. Sistemi tek seferde teslim etmek yerine kullanıcı gereksinimlerinin önemine göre sistemi, geliştirme ve teslim parçalarına böler. Öncelikle en önemli gereksinimler için çekirdek bir sistem geliştirilir. Her teslim artırımı beklenen işlevselliğin bir parçasını karşılar ve her teslimle birlikte müşteriye sunulabilecek bir değer olduğundan, sistemin işlevselliği erken aşamalarda ortaya çıkar. Artımsal geliştirme modelinde bir taraftan üretim bir taraftan da kullanım yapılır. Bir parçanın geliştirilmesi başladığında gereksinimleri durdurulur ve olası değişiklikler sonraki teslimlerde ele alınır. Erken artırımlar prototip gibi davranarak gereksinimlerin daha iyi anlaşılmasını sağlar ama artırımları tanımlamak için tüm sistem tanımlanmalıdır. Ayrıca gereksinimleri doğru artırımlara atamak gerekir, bu da bazen zor olabilir. Artımların kendi içlerinde tekrarlama özelliği yoktur. Bu model için deneyimli personel gereklidir.
Çağlayan, V Modeli ve Artımsal Gelişme Modelleri, yazılım yaşam döngüsü modellerinin açıklamalarını yaptım. Şimdi bu üç modeli detaylıca karşılaştıralım:
Öncelikle gereksinim belirleme zamanlarının Çağlayan ve V Modellerinde başlangıçta, Artımsal Modelde belirli sıklıklarda olduğunu görüyoruz. Bununla beraber Çağlayan ve V Modellerinde aşamaların iç içe olmadığını, Artımsal Modelde ise iç içe ve kapsayıcıdır. Esneklik konusu, Çağlayanda neredeyse yok denecek kadar azken, V Modelinde düşük ve Artımsalda çok yüksek seviyededir. Bu karşılaştırmalar sonucunda da Çağlayan ve V Modellerinde, Artımsala göre çok daha zor değişiklik yapılabildiğini ve beraberinde Çağlayan ve V'nin, Artımsala göre daha maliyetli olduğunu gösteriyor. Başarı garantisi de bu sebeplerden en yüksek Artımsal Modelde, onu V Modeli takip ediyor ve en düşük başarı garantisi Çağlayan Modelinde oluyor. Proje süresi üçünde de çok uzun ve proje uygulaması üçünde de kolay seviyede. Genel olarak, Çağlayan Modeli ve V Modeli daha geleneksel bir yaklaşım sergilerken, Artımsal Gelişme Modelleri daha modern bir yaklaşım olarak öne çıkıyor. Çağlayan Modeli ve V Modeli daha fazla planlama ve öngörülebilirlik sağlarken, Artımsal Gelişme Modelleri daha fazla esneklik ve uyum sağlar. Tercih edilecek model, projenin gereksinimlerine ve özelliklerine bağlıdır.
Yazılım yaşam döngüsü modelleri, yazılım geliştirme sürecinde kullanılan tekniklerin ve metodolojilerin bir bütünüdür ancak yazılım süreç yönetim modelleri, bir proje yöneticisi veya ekip lideri tarafından uygulanan yönetim tekniklerini kapsar.
Yazılım süreç yönetim modelleri, yazılım endüstrisindeki büyük projelerin daha iyi yönetilmesi ve kalitesinin artırılması ihtiyacından ortaya çıkmıştır. Bir yazılım projesinin başarılı bir şekilde yönetilmesi yani proje yönetimi, kalite kontrol, risk yönetimi ve diğer proje

yönetimi faaliyetlerinin gerçekleştirilmesi için kullanılan bir dizi teknik, strateji ve yöntemlerdir. Yazılım süreç yönetimi modeline Extreme Programming, Scrum, Agile Unified Process, Feature-Driven Development, Test-Driven Development, Lean, Microsoft Solution Framework, Capability Maturity Model Integration, ISO 12207 modelleri örnek verilebilir. Ben günümüzde en popüler modellerden olan Scrum hakkında bilgi vereceğim ve popülerliğinin sebebini anlatacağım.

Scrum, özellikle yazılım geliştirme süreçlerinde kullanılan bir proje yönetim metodolojisidir. Bu metodoloji, takım çalışmasını vurgulayan bir çerçevedir ve sadece yazılım geliştirme için değil herhangi bir proje için uygulanabilir. Bu yöneyim yaklaşımı karmaşık ortamlarda adım adım yazılım geliştiren ekiplere ve kolaylıkla tamamlanamayan gereksinimlere sahip kaotik durum projelerine uygundur. Bu karmaşıklığı şeffaflık, denetleme ve uyarlama ilkeleriyle azaltmaya çalışır.

Scrum, belirli bir süre boyunca (genellikle iki hafta) yapılan Sprint adı verilen periyodik zaman dilimleriyle çalışır. Her Sprint başlangıcında, takımın bir hedef ve bu hedefe ulaşmak için yapılacak işleri belirlemesi gerekir. Bu işler, bir Product Backlog adı verilen bir liste üzerinden takım tarafından belirlenir ve öncelik sırasına göre düzenlenir. Düzenlenen bu işler Sprint Backlog denen 15-30 günlük proje zaman diliminde tamamlanır. Sprint boyunca, takım, günlük toplantılar ve iş birliği ile hedefe ulaşmak için çalışır. Sprint sonunda, takımın başarıları değerlendirilir ve yeni bir Sprint başlatılır. Bu süreç, takımın sürekli gelişmesine ve ürünün daha iyi hale gelmesine yardımcı olur.

Scrum modelinde üç çeşit temel kavram vardır: roller, toplantılar ve bileşenler.

Üç çeşit ana rol vardır, ilki ürün sahibi. Ürün sahibi, işin işvereni veya müşterisi olan kişidir ve ürünün geliştirilmesindeki gereksinimleri ve hedefleri belirler. Ürün sahibi, ürün gereksinimlerini mümkün olan en iyi şekilde belirlemeli ve geliştirme ekibi ile sürekli olarak iletişim halinde olmalıdır. Ayrıca, ürün sahibi, sprint sonunda teslim edilen ürünün kalitesini onaylar. Diğeri Scrum yöneticisi. Scrum yöneticisi, Scrum sürecinin etkin bir şekilde uygulanmasını sağlamak için sorumludur. Geliştirme ekibine destek olur ve sorunları çözmelerine yardımcı olur. Ayrıca, sprint toplantılarının düzenlenmesi, sprint hedeflerinin belirlenmesi ve sprint takibinin yapılması gibi süreçlerin düzenlenmesinden de sorumludur. Sonuncu olarak da Scrum takımı. Scrum takımı, ürünün geliştirilmesi ve teslim edilmesi için sorumludur. Takım; programlama, test etme ve kullanıcı arayüzü tasarımı gibi tüm işlevleri gerçekleştirir. Geliştirme ekibi, sprint boyunca birlikte çalışarak ürün sahibinin belirlediği gereksinimleri karşılamak için en iyi çözümleri bulmak için birbirleriyle işbirliği yaparlar.

Diğer bir kavram toplantılardır. Üç çeşit ana toplantı vardır. İlki Sprint planlama toplantısıdır. Her Sprint başlamadan önce gerçekleştirilir. Bu toplantıda başarılı geliştirme için geniş kapsamlı gereksinim listesi çıkarılır. Ve takımlar belirlenir. Takımlar kendi aralarında bölüşür ve hedefleri belirler. Projenin risk ve kontrolleri değerlendirilir ve olası gereksinim değişkenleri belirlenir. Toplantıda, geliştirme araçları ve altyapısı onaylanır. Aynı zamanda dağıtım, geliştirme ve pazarlama maliyetleri hesaplanır. Yönetim ve destekler irdelenir ve onaylanır. Diğer bir toplantı çeşidi Sprint gözden geçirme toplantısıdır. Bu toplantıda takım üyeleri Ürün Gereksinim Listesinin en üstünden başlayarak Sprint Gereksinim Listesi oluştururlar. Bu Scrum'ın anahtar uygulamasıdır. Takım, ürün sahibinin belirlediği öncelikli gereksinimlerden yapılabilecekleri belirler. Son toplantı çeşidi ise günlük Scrum toplantısıdır. Bu toplantı, sprint boyunca her gün gerçekleştirilir. Geliştirme ekibi, sprint hedefleri doğrultusunda yapılan işleri, sorunları ve ilerlemeleri tartışır. Toplantı, genellikle 15 dakika sürer ve tüm ekibin katılımı zorunludur. Son kavram olan bileşenler ya da araçlar kavramıdır. Bu kavramın da üç başlığı mevcuttur. İlki Product Backlog olarak geçen Ürün Gereksinim

Dokümanıdır. Bu dokümanda proje iş elemanlarının basit bir listesi oluşturulur. Geçerli ve kullanışlı olması gerektiğinden sürekli bir güncelleme gereklidir. Bu doküman Scrum günlük toplantılarında takip edilir. Genellikle kullanıcı hikayelerinden oluşur bu durumda kullanıcı bakış açısından bakılmasını sağlar. Diğer başlık Sprint Backlog yani Sprint dokümanıdır. Mevcut Sprint için Product Backlog'dan elde edilmiş görevleri içerir. Dokümanın amacı Sprint sonunda son ürünün işlevsel bir parçasını elde etmektir. Üçüncü ve sonuncu başlık Burndown Chart adıyla Sprint Kalan Zaman Grafiğidir. Bu grafik, Sprint boyunca kalan süreyi takip etmek için kullanılır. Bu grafik, sprint boyunca tamamlanması gereken görevlerin bir listesini ve her görevin tahmini süresini gösterir. Grafik, günlük olarak güncellenir ve sprintin sonuna kadar ne kadar zaman kaldığını gösterir.

Scrum'ın temelinde daha önce de belirttiğim gibi üç temel ilke vardır: şeffaflık, denetleme ve uyarlama. Bu ilkeler projelerin daha verimli olmasını sağladığından Scrum için üst düzey öneme sahiplerdir. Bu yüzden bu ilkeleri biraz daha açmak isterim. Şeffaflık, tüm projelerin tüm paydaşları arasında tamamen şeffaf ve anlaşılır olması gerektiğini vurgular. Bu, proje ilerlemesi, iş yükü ve riskler hakkında açık ve net bir iletişim sağlamayı içerir. Denetleme ise takımı, Sprint boyunca gerçekleşen ilerlemeyi ve sonuçları düzenli olarak denetlemeler. Bu sayede, hedeflenen sonuçların gerçekleştirildiğinden ve sürecin en verimli şekilde işlediğinden emin olunur. Uyarlama, proje planlarını ve iş yükünü sürekli olarak gözden geçirerek ve gerektiğinde planları değiştirerek yapılır. Ekibin hedeflenen sonuçlara ulaşmak için gereken uyumu sağlaması ve uyarlanabilir olması önemlidir.

Bu üç Scrum ilkesi, Scrum'ın günümüzdeki popülerliğini anlamamızda yardımcı olsa da eklemek istediğim başka ilkeler de var. Öncelikle Scrum, Sprintlerle yönetilir. Bu projenin küçük parçalara ayrılıp her bir projenin belirli bir süre içinde tamamlanmasıdır. Bu durum projenin hızlı bitmesine yardımcı olur. Diğer bir ilke ekip çalışmasını teşvik ettiğinden Scrum takımındaki tüm üyerelr işbirliği içinde ve yüksek motivasyonla çalışır. Aynı zamanda sık sık geri bildirim alarak sürekli iyileştirme yapabilme ilkesine de sahiptir bu da başarının sürekli artırılması anlamına gelir. Bu nedenlerden dolayı, Scrum yöntemi, özellikle yazılım geliştirme endüstrisi için popüler bir seçenek haline gelmiştir. Ancak Scrum'un avantajları, sadece yazılım geliştirme için değil, diğer endüstriler için de geçerlidir ve bu nedenle Scrum, işletmelerin birçok farklı bölümünde uygulanmaktadır. Bazı farklı bölümlere örnek vermem gerekirse; Satış ekipleri, müşteri beklentilerini daha iyi anlamak ve satış hedeflerini gerçekleştirmek için, Pazarlama ekipleri, kampanyalarını yönetmek, marka bilinirliğini artırmak ve ürünlerini tanıtmak için, İnsan kaynakları ekipleri, işe alım süreci, performans değerlendirmeleri ve eğitim programları gibi işlevleri yönetmek için, Üretim ekipleri, üretim süreçlerini optimize etmek ve müşteri taleplerine daha hızlı yanıt vermek için Scrum'u kullanabilirler.

Özetle, bu yazımda yazılım yaşam döngüsünden va aşamalarından bahsettim. Bu döngü içindeki modellerden Çağlayan Modeli, V Modeli ve Artımsal Gelişme Modelini anlattıım ve karşılaştırmalarını yaptım. Sonrasında, yazılım süreç modellerinden bahsettim ve örneklerini verdim. Yazılım süreç modellerinden günümüzde en popüler olan Scrum'ı detaylıca anlattım ve bu popüleritenin sebeplerini sıraladım. Son olarak Scrum'ın ilkelerinden kaynaklı her türlü işletmede ve bu işletmelerin farklı bölümlerinde kullanılabileceğini tekrar vurgulayıp bunlara örnek verdim.

Perin Tanç
220601125

## Kaynakça

Aktan, C. C. (2011). Organizasyonlarda Değişim Yönetimi: Değişim Mühendisliği. Organizasyon ve Yönetim Bilimleri Dergisi, 3(1), 1-16. ISSN: 1309-8039 (Online)

Seker, S. E. (2015). Yazılım Geliştirme Modelleri ve Sistem/Yazılım Yaşam Döngüsü. YBS Ansiklopedi, 2(3), 1877-0428.

Scrum.org. (2021). The Scrum Guide. Retrieved from https://www.scrum.org/resources/scrum-guide

Fikir Jeneratoru. (2021). Yazılım Proje Yönetimi Yöntemleri. Retrieved from https://fikirjeneratoru.com/yazilim-proje-yonetimi-yontemleri/

Bakırçay Üniversitesi Yazılım Mühendisliği Temelleri ders slaytları