Yazılım Yaşam Döngü Modelleri

Günümüzde bir yazılım geliştirmek geleneksel bir ürün geliştirmekten daha farklı ve komplikedir. Bu yüzden yazılım geliştirirken yazılım yaşam döngü modelleri kullanılır. Birçok modelin farklı yaklaşımları vardır fakat genellikle şu adımları içerir: Planlama, analiz, tasarım, kodlama, test etme ve bakım. Planlama aşamasında yazılımın gereksinimleri belirlenir ve bu gereksinim doğrultunda hedefler, bütçe ve bunun gibi şeyler belirlenir. Analiz aşamasında yazılımın kullanacağı teknolojiler belirlenir. Örneğin bir web uygulaması için uygulamanın single page (tek sayfa) olması mı yoksa multipage (çok sayfalı) olması mı , kullanılacak kütüphaneler(paketler) veya kullanılacak veri tabanı, nerede host edileceği gibi. Kodlama aşamasında bu gerekesinim ve tasarımlar kullanılarak yazılımın kodu yazılır. Test etme aşamasında yazılım test edilir. Bakım aşamasında ise yeni güncellemeler sayesinde yazılım geliştirilmeye devam edilir. Bu yazımda sizlere Scrum, kodla ve düzelt , çağlayan ve arıtımsal geliştirme modeli ile ilgili bilgi vereceğim ve bu modellerin karşılaştırması ve neden popüler olduğuyla ilgili de açıklamalar yapacağım.

Kodla Ve Düzelt

Aslında isminden de anlaşıldığı gibi basit ve kolay uygulanabilir bir yöntemdir. Fakat bakımı çok zordur. Dokümantasyon yapılmaz. Öncelikle ilk versiyon kodlanır ve müşterinin gereksinimleri karşılanana kadar düzenlenir. Daha sonra teslim edilir ve teslim sonrası oluşan hata veya yeni gereksinimler için bakımı yapılır. Fakat dokümantasyon olmadığı için zordur. Özellikle tecrübesiz şekilde yazılan kod daha sonra birçok sıkıntıya sebep olabilir. Test yoktur. Kodlamaya başlamadan önce tasarım analiz gibi adımları bulunmadığından gelişigüzel bir iş yapılır. Büyük projeler için çok kullanışsız ve yanlış bir yöntemdir. İki aşamadan oluşur. Basitliği nedeniyle küçük deneyimsiz ekipler tarafından kullanılır. Küçük işlevli 100 200 satırdan oluşan hobi gibi projelerde kullanılabilir.

Çağlayan Yazılım Geliştirme Modeli

Çağlayan yazılım geliştirme modeli günümüzde kullanımı azalsa bile daha öncesinde sıkça kullanılmıştır. Geleneksel yazılım geliştirme modeli olarak da bilinmektedir. Emeklilik aşaması da bulunur. Daha çok iyi tanımlanmış gereksinimleri ve müşteri istekleri tam olan ve üretimi az zaman gerektiren projeler için daha uygundur. Aşamalar arası en az 1 tekrar olur. Bu modelde projenin adımları belli olup sırayla tamamlandığı (biri bitmeden bir sonrakine geçilmez) bir yazılım geliştirme modelidir. Her aşamadan sonra dokümantasyon ve test yapılır. Yapısı çağlayana benzediği için de adına çağlayan modeli denmiştir. Çağlayan yazılım geliştirme modeli 5 temel adımdan oluşur. Bu adımlar: İhtiyaç analizi, tasarım, geliştirme, test ve bakım olarak sıralanabilir. Bir adım bitmeden diğer adıma geçilmez ve adımlar arası atlama yapılmaz. Her adım arasında güçlü bir dokümantasyon bulunur ve testler gerçekleştirilir ve eğer dokümantasyon ve test gerçekleştirilmediyse bu adım tamamlanmış sayılmaz. Bu yüzden bakımı daha kolaydır. İhtiyaç analizi çağlayan yazılım geliştirme modelinin en önemli ve en dikkat edilerek yapılması gereken aşamasıdır. Analiz aşamasında müşteri ile beraber uygulamanın ihtiyaçları tanımlanır. Ürün müşterinin istekleri doğrusunda şekillenir. Geliştirme başlamadan önce müşterinin istekleri raporlanır. Tasarım aşamasına geçilmeden önce analiz aşamasının doğru bir şekilde yapılması çok önemlidir. Çünkü yanlış bir tasarım sonucu aşamaların tekrar etmesi gerekebilir. Bu yüzden en önemli aşama analiz aşamasıdır. Tasarım aşamasında kullanılacak teknolojiler projenin işleyişi ile ilgili detaylı bir tasarım oluşturulur ve dokümantasyon ve testler sonrası sonraki aşamaya geçilir. Geliştirme aşamasında isminden de anlaşıldığı gibi tasarımda ortaya çıkan tasarım geliştirilir. Daha sonra uygulama(test) aşamasına geçilir. Bu aşamada ürün testlere tabi tutularak gereksinimleri karşılayıp karşılayamadığı kontrol edilir. Kullanıcı testleri yapılır. Bakım aşaması yazılımın bakımı yeni güncelleştirmelere yapılır. Ortaya çıkan hatalar ve sorunlar düzeltilir, yamanır. Çağlayan modelinin birçok avantajı vardır. Kontrollü bir şekilde ilerlediğinden geliştirme aşamasında bir hata çıktığında veya tasarımsal bir hataya rastlandığında çok daha hızlı bir şekilde yamanabilir. Her adım arası dokümantasyon ve testler yapıldığından çok kontrollü bir yazılım geliştirme yaşam döngüsü olduğu söylenebilir. Projenin adımlarının planlanması daha karmaşık modellere göre çok daha kolaydır. Tüm aşamalar önceden belirlenir. Ve sırasıyla en az bir tekrar olacak şekilde gerçekleştirilir. Bu nedenle planlanması çok daha kolay bir yazılım geliştirme modellidir. Avantajlarına ek olarak birçok dezavantajlıda bulunmaktadır. Analız ve tasarım aşamasında yapılan hatalar çok daha maliyetli sonuçlara yol açar. Müşterinin analiz aşamasında kendini yanlış anlatması sonucu ürünü ancak test aşamasında görebileceği için birçok tekrara ve maliyete yol açabilir. Bu yüzden gereksinimler iyi anlaşılmalıdır. Yazılım ekiplerinde bir an önce kod yazma hevesi olduğu için analız tasarım gibi adımlara daha az önem verilmektedir. Bunun sonucunda maliyetler çok ağır olabilir. Kullanıcı modelin içinde yer almadığından sadece yazılımın sonucunda geri dönüş yapabilir bunun sonucu olarak da birçok aşamada tekrara düşülebilir. Her aşamada dokümantasyon ve test yapıldığından maliyeti yüksektir. Günümüzde de kullanılırlığını yitirmekte olan çağlayan modeli yine de gayet kullanılabilir bir modeldir. Fakat en önemli aşaması olan analiz ve tasarım aşamalarına çok dikkat edilmelidir.

V Süreç Modeli

Günümüzde de sıkça kullanılan bir model olan V süreç modeli sürecin V harfine benzemesinden ötürü v süreç modeli olarak adlandırılmıştır. Bu modelde kullanıcının projeye katkısını arttırmak ve yazılım geliştirme sürecinin eksiksiz bir şekilde takip edilerek kontrollü bir şekilde ürün üretmek amaçlanmaktadır. Üç modelde tanımlanır. Bunlar: Kullanıcı modeli, mimari model, gerçekleştirim modelidir. Öncelikle ürün üretilir daha sonra testler yapılır. Bu modelde her adımın sonucu bir sonraki adımda kullanılır. Kullanımı kolay ve kalite kontrolünün iyi sağlandığı bir model olduğu için büyük ve karmaşık projelerde çokça tercih edilmektedir. Sekiz aşamadan ve iki evreden oluşmaktadır. Öncelikle gereksinim analizi yapılır. Bu evrede gereksinimler kullanıcıdan toplanır ve dokümante edilir. Kullanıcı kabul testleri de bu aşamada tasarlanır. Daha sonra sistem tasarımı aşamasında geçilir. Bu aşamada sistem mühendisleri gereksinimlerden yola çıkarak sistem tasarlanır ve dokümantasyon yapılır. Sistem testi için de bu aşamada dokümantasyon hazırlanır. Mimari tasarım aşamasında ise daha çok teknik tasarım yapılır. Hangi veri tabanının kullanılacağı gibi birçok tasarım bu aşamada yapılır. Daha sonra modül tasarımı gelir. Modül tasarımı düşük seviyeli tasarım olarak da adlandırlıabilir. Burda veritabanı tabloları sözde kod ve akış şemaları ile ürün tasarlanır. Daha sonra bu tasarım aşamaları kodlamaya başlanılır. Daha sonra v sürecin diğer tarafı olan sınama tarafına geçilinir. Öncelikle birim testleri(unit testing) yapılır. Daha sonra bir üst olan alt sistem testleri ve daha sonra sistem testleri yapılır. En sonunda ise oluşan sisteme e kullanıcı testleri yapılır. Bu testlerin de sonucunda ürün ortaya çıkar. V süreç modelinin birçok avantajı bulunmaktadır. Kullanıcının projeye katkısı çağlayan modeline göre daha fazladır. Test ve verifikasyonlar erken aşamada tasarlandığından tüm ürünlere uygulanır. V süreç modelinin dezavantajlarına ise fazlar arasında tekrarlama olamaması ve risk çözümleme aktiviteleri olamaması gösterilebilir. Testler sırasında da gereksinimlerin farklık gösterebilir. V süreç modeli çağlayan yazılım geliştirme modeline göre testi çok daha ön planda tutar ve her aşamanın test edilmesi sağlanır. Çağlayan modelinde ise test ayrı bir aşama olarak tutulmaktadır. Çağlayan modeli v modeline göre çok daha doğrusal ilerleyen bir yazılım geliştirme modelidir. İkisinin de de uygulanması kolaydır ve büyük karmaşık projelerde kullanılabilir. Dokümantasyon olduğundan bakımları kodla ve düzelt modeline göre maliyetsizdir.

SCRUM