Yazılım Yaşam Döngüsü Nedir?

Yazılım Yaşam Döngüsü adından da anlaşıldığı gibi yazılım sürecinde gerçekleşen, aynı insan yaşamından alıntılanan bir gelişim süreci içerir. Bu süreç içerisinde en başta planlama olmak üzere, analiz, tasarım, gerçekleştirme, bakım ve emeklilik adımları bulunur.

Planlanma-gereksinim aşamasında adından da anlaşıldığı gibi sürecin planlaması yapılır ve süreç boyunca katılım sağlayacak kişilerin görev dağılımı gerçekleştirilir. Müşteri isteklerinin ve yapılan işin ne olduğu anlaşılmaya çalışılır. Müşteri isteklerinin ve ürünün gereksinimlerinin doğru anlaşılması tüm proje için hem hataları hem de oluşacak zarar maliyetlerini düşürecektir.

Analiz aşamasında projenin risk analizi gerçekleştirilir. Karşılaşılabilinecek sorunlar ve sürecin tamamlanabileceği terminler üzerinde durulur. Kapsamlı bir bakışla proje gözden geçirilir. Projenin yönetim planı belirlenir.

Tasarım aşamasında sürecin mimari tasarımı gerçekleştirilir ve sürecin nasıl devam edeceği görüşülür. Oluşturulan ayrıntılı tasarım ve kullanılacak algoritmalar belirlenir.

Test-Gerçekleştirme aşamasında projenin yazılım kısmına ağırlık verilerek kodlama ve test aşamaları uygulanır. Her modül için ayrı ayrı testler ve ayrı kod yazılımları gerçekleştirilir. En sonunda kabul testi uygulanır.

Bakım aşaması ise, ürün müşteriye sunulduktan sonraki süreçte yazılımın bakımı ve güncelleme olaylarının yapılmasını kapsar. Uyarlama, Bakım ve Mükemmelleştirici Bakım olmak üzere ikiye ayrılır. Mükemmelleştirici bakım müşterinin geri dönüşü ve isteği üzerine yapılan geliştirme ve iyileştirme bakımlarını kapsar. Uyarlayıcı bakım ise modülün başka bir yere uyarlanması ve taşınması gerektiği için yapılmaktadır.

Emeklilik aşaması ise yazılım ürününün artık geliştirmelere cevap vermediği ve müşterinin isteklerini karşılayamadığı yerde gelinen noktadır. Yazılım ürününün ömrü dolduğundan yazılım durdurulur.

Yazılım Yaşam Döngüsü Modelleri

Yazılım yaşam döngüsünde birden fazla model bulunmaktadır. Bu modellerin birden fazla ortak noktası bulunmasına rağmen bir çokta farklı noktaları da bulunmaktadır. Kullandıkları yöntemler, projenin kimlere hitap edeceği, proje zorluğu, projede çalışanlar olmak üzere birçok noktada ayrılırlar. Bu modellere Gelişigüzel Model, Barok Model, Çağlayan Yaşam Döngü Modeli, V Süreç Modeli, Kodla ve düzelt Yaşam-Döngü Modeli, Helezonik (Spiral) Modeli, Artımsal Geliştirme Yaşam-Döngü Modeli, Çevik Model ve Scrum gibi örnek verilebilir.

Gelişigüzel Model

Genellikle tercih edilen ve doğru bulunan bir model değildir. Göz önünde bulundurulan bir model bulunmamaktadır. Yazılım ürünü oluşturulurken herhangi bir yöntem kullanılmaz. Kişiye bağlı ve kişiye özel yazılım geliştirme seçenekleri bulunmaktadır. Yazılımın izlenebilirliği ve bakım yapılabilirliği herhangi bir yöntem kullanılmadığından zordur. Geriye dönük hataların telafisi ve tespiti hem maliyetli hem de yorucudur.1960'larda kullanılmaya başlanmıştır. Kullanımı teknoloji ve diğer yazılım yöntemleri geliştikçe giderek azalmıştır.

Örnek olarak Python programında çalışılan küçük öğrenci modelleri verilebilir.

Barok Modeli

Yazılım yaşam döngüsünü doğrusal bir şekilde ele alır. Aşamalar arası gereken geri dönüşlerin nasıl olacağı tanımlı değil, belirsizdir. Dokümantasyon günümüz süreçlerinden farklı olarak ayrı bir süreç gibi ele alınır. İterasyon yer almamaktadır.

Çağlayan Modeli

Geleneksel yazılım geliştirme modeli olarak da adlandırılabilir. Aşamaların en az birer kez tekrarlanması ile geliştirilir böylelikle her aşamaya en az birer kez dönüş vardır. Çok iyi tanımlanmış, gereksinimleri net bir şekilde belirlenmiş ve üretimi için az zaman gerektiren projeler için kullanılmaktadır. Bir önceki aşamanın ürettiklerini kullanıp bir sonraki aşamada eksikleri buna göre güncelleyerek devam eder. Şelaleye benzetildiği için Şelale modeli olarak da adlandırılmaktadır. Gelenekselleşmiş bir model olduğundan günümüzde kullanımı azalmıştır. Bir aşama bitmeden diğer aşamaya geçilmediğinden her aşama kendi içinde bir kontrolden geçme durumundadır. Her aşama için dokümantasyon oluşturulur. Dokümantasyonu olmayan aşama tamamlanamaz. Analizdeki tüm detaylar tasarıma aktarılmalıdır. Tasarım detaylı çalışma gerektirir. Uzun zamana yayılan projelerde gereksinim değişir ve böylelikle sonradan geliştirmelere ihtiyaç duyulur. Her sonradan geliştirme de ayrı bir maliyet oluşturmaktadır. Son kullanıcı ve müşteri tarafından anlaşılır olmalıdır. Kullanıcı sürecin içinde yer almaz böylelikle her aşama sonrası geri dönüşler verir. Her geri dönüşün geliştirilmesi daha önce de bahsettiğimiz gibi hem maliyeti hem de hata oranını arttırır. Gereksinim analizi yapılırken çok dikkat edilmelidir. Gerekesinim analizi eksik yapılırsa hata artar ve yapılan geri dönüşlerle son kullanıcıya ulaşma zamanı gidererek artar. Bu durum da müşteri memnuniyetini düşürür.

Evrimsel Model

İşlevsel olarak tüm modeli görüp gerekli güncellemelerle üstüne koyarak ilerler. Banka gibi çok birimli organizasyonlarda kullanışlıdır. Her aşamada üretilen, müşteriye teslime hazır bir ürün vardır. Bakımı ve görünürlüğü zordur. Çünkü her aşama bir önceki aşamanın geliştirilmesiyle oluşturulduğundan hata takibi zordur. Ne kadar çok pilot ve test uygulaması yapılırsa hata oranı bir o kadar düşmektedir. Sürekli yenilik ve değişiklik yapıldığından bu durum ürünün yapısına zarar verebilir.

Evrimsel ve Çağlayan Model arasında en büyük fark evrimsel modelin her aşamasından yeni bir ürün oluşuyor olmasıdır. Oluşan bu ürünler müşteriye teslim hazır kullanılabilir ürünlerdir. Ancak çağlayan modeli kullanılarak oluşturulan yazılım ürünleri, aşamalarla gelişerek en son aşama tamamlandığında tüm geliştirmeler bittiğinde müşteriye teslim haline gelen ürünlerdir.

V Süreç Modeli

İş tanımları belirgin olduğundan belirsizlik azdır. Kullanıcının projeye katkısı vardır. Üretim ve test olmak üzere iki ana dal üzerinde ilerlemektedir. Şelale modelinin gelişmiş bir modeli olarak düşünülebilir. Proje yönetimi her aşamada test olduğundan kolaydır. Aşamalar arası tekrarlama yoktur. Teslim edilebilir tüm ürünlerde ve aşamalarda test ve üretim yapılabildiğinden hata erken fark edilir, geliştirmeler erken uygulanır zaman ve maliyetten

tasarruf edilir. Kullanımı kolaydır. Yazılım diğer sistemler gibi zamanla evrimleştiğinden kaynaktan ve gereksinimlerden uzaklaşılabilme ihtimali vardır.

Kodla ve Düzelt

Bu model en kolay modellerden biridir. Birkaç yüz satırdan oluşan programlar için geçerli olabilir. Yazılım ürünü direkt gerçekleştirilebilir. İlk başta ilk ürün oluşur. Modelin geliştirmeleri kaldıramadığı ve kullanıcı isteklerini karşılayamadığı zaman emeklilik durumu mümkün olmaktadır. Kodlama bitene kadar ürün geliştirmesi devam eder. Resmi olmayan farazi fikirlerle proje çalışılmaya başlanır. Planlı olarak ilerlemez. Küçük ve kısa ömürlü projeler için idealdir. Kontrollü değildir. Planlama olmadığından planlanan bitiş süresi de yoktur. Yapılan kodlamaları sonrandan değiştirmek mümkün olmayabilir.

Spiral Model

Spiral Model de risk analizi ön plan çıkar. Her döngü bir fazdan oluşmaktadır. Dört ana aşaması vardır, bunlar; planlama, risk analiz, üretim, kullanıcı değerlendirmedir. Riskler ayrı ayrı değerlendirilebilmektedir. Ara adım arttıkça oluşturulan dokümantasyon artmaktadır. Prototip yaklaşımına sahiptir. Geliştirmeyi parçalara bölerek tamamlar böylelikle hataları erken görmeye neden olur. Öznel risk değerlendirmeye sahiptir.

Arttırımsal Geliştirme Modeli

Sprial model de olduğu gibi geliştirme parçalara bölünür anacak diğerlerinden farklı olarak teslim de parçalara bölünür. Her aşamada farklı bir bölüm tamamlanmaktadır. Olası değişiklerin geliştirmesi bir sonraki teslimde ele alınır. İşlevsellik her aşamada mevcuttur. Gereksinimlerin önem sırasına göre ilerlenir. Böl ve yönet mantığı ile çalışılmaktadır. Her yazılım sürümü birbirini kapsamaktadır. Gereksinimlerin önceliklendirilmesi kullanıcı tarafından yapılmaktadır. Sistem fazla sınandığından başarısızlık riski azdır.

Çevik Yazılım Geliştirme Modelleri

Adından da anlaşıldığı gibi modelin ana teması çevikliktir. Gelen ve değişen isteklere hızla yanıt verebilir, ürünü en hızlı şekilde çıkarabilir. Verimliliği fazla esnek ve hata oranı düşüktür. Hızlı olmasına artı olarak ucuz bir model olduğundan tercih edilir. Küçük iterasyon ve projelerle ilerlenir. Müşteriyle iletişim çok önemlidir her proje sonucundan müşteriye bilgi verilir. Sürekli müşteriye bilgi verildiğinden ve hızlı aksiyonlar alındığından geriye dönük hataların düzeltilmesi kolay ve diğer modellere göre incelendiğinde daha maliyetsizdir. Takım ile ilerlenir ve sürekli iyileştirmeyi hedef alır. Öğrenim gerektiren bir durum bulundurmadığından adaptasyonu kolaydır. En popüler ve tercih edilen modelleri ise XP ve Scrum dur.

XP (Expand Programming)

Grup içi iletişim önemlidir. Geri dönüşün daha fazla olmasına imkân sağlar. 12 farklı pratikte ilerler. Müşteriyle kurulan sürekli iletişim ve geri dönüşler hatayı azaltma yönündedir.

Scrum

Her projeye uygulanabilen çevik süreç ile ilerlemeyi kolaylaştıran bir modeldir. Her iterasyonda geri dönüş mevcuttur. Karmaşık yazılımları parçalara bölerek karmaşıklığı azaltır, böldüğü her

parça “sprint” olarak adlandırılır. Dokümantasyon sadece gerekli durumlarda kullanılır. Günlük ve kısa vadeli toplantılar sürekli iş takibi sağlar. Her sprint sonunda yazılımın fonksiyonel bir parçası biter ve müşteriye hazır olur. Model üç temel kavram üzerinden ilerlemektedir. Bunlar; roller, toplantılar ver bileşenlerdir. Roller kendi içinde üçe ayrılır; ürün sahibi, scrıum yöneticisi ve scrum takımı. Proje yöneticisi olmadığından projeden herkes sorumludur ve takım özerkliğini bozmamaktadır. Ürün sahibi projenin beyni olarak görev almaktadır. Toplantılar da daha önce de bahsettiğimiz gibi üzere sprint ve günlük scrum olmak üzere ikiye ayrılır. Sprintler gözden geçirme adı altında yapılır. Bileşenler ise dokümanları ve gereksinimleri oluşturur. Dokümanlar, ürün gereksinim dokümanı ve sprint dokümanı olmak üzere ikiye ayrılır. Dokümanlar canlıdır ve bu sebepten bakım gerektirir.

Scrum günümüzde en popüler yazılım ve aynı zamanda sistem geliştirme modellerinden biridir. Sahip olduğu popülerlik her geçen gün artmaktadır.

Peki Scrum Neden Bu Kadar Popüler?

Scrum her çeşitli teknolojiye gelişen yeni sistemlere, projelere uyumlu bir modeldir. Aynı zamanda karmaşık yapılarda, küçük birimlere bölüp ilerleme katettiğinden kaotik süreçler için birebirdir. Ekip içi iletişim ve sürekli geri bildirim sayesinde hatalar erken fark edilir ve sorunlar minimuma indirilir. Böylelikle hem zamandan hem de hatalardan oluşacak maliyetler düşürülür. Değişen gereksinimlere yapılan toplantılarla hızla ayak uydurabilir.

Hangi model bizim için daha avantajlıdır?

Bahsettiğimiz bir sürü yazılım geliştirme modeli sunduk ancak tercihlerimizi neye göre belirlemeliyiz?

Bir modelin tercih edilmesi için maliyetinin düşük, iletişimin yüksek, geri dönüşlerin kolay alınabildiği, hızlı sonuç alınabilen ve gereksiz iş gücünü ortadan kaldıran bir model olmalıdır.

Bu tercih sebeplerini yerine getirmeyen modeller zamanla işlevselliğini yitirmiştir. Bu modeller arasında Gelişigüzel ve Barok modeli bulunmaktadır. Yinelemeli iterasyonlar içermediğinden geri dönüşleri ve hata tespitlerini zorlaştırdığından tercih edilmemeye başlanmıştır. Aynı zamanda dokümantasyon sürecini iş akışlarına çok fazla dahil ettiklerinden dolayı iş gücünü arttırmaktadırlar.

Şelale yani Çağlayan Modeli ise eskiden çok kullanılmasına rağmen günümüzde kullanımı giderek azalmıştır. Çünkü küçük projeler için yeterli olduğundan günümüzde artan talep ve projelerin büyümesi sebebiyle ve iletişimin az olmasından dolayı yetersiz kalmaya başlamıştır.

V modeli çağlayan modelinden biraz daha gelişmiş olsa dahi tekrarlamaları içermemesi, risk analiz aşamalarının bulunmamasına rağmen kolay kullanımı ve erken aşamalarda gerçekleştirilebilmesinden dolayı tercih edilmektedir.

Spiral model ise istediğimiz çoğu geliştirmeyi ve olanağı taşımasına rağmen karmaşık yapısı, sonsuzluğu, dokümantasyon sürecinin fazla olması gibi nedenlerle tercih edilmemektedir. Ancak risk analizin ön plana çıkması yinelemeli olması büyük bir avantajdır.

Gereksinimlerin evrimsel süreçte çok iyi gözükmesine rağmen görünürlüğü ve bakımı zor olduğundan giderek önemini yitirmektedir.

Kodla düzelt modeli ise kolay olmasına rağmen bir süre sonra model geliştirmeleri kabul edemeyip emeklilik içermesinden ve bakımının çok zor ve değişkenliğin fazla olmasından dolayı popülerliğini yitirmiştir.

Çevik modeller ise Scrum kadar öne çıkmasa daha günümüzde çok kullanılmaktadır. İletişim hem ekip içi hem de müşteri ile süreklilik taşıdığından erken hataların fark edilmesi ve karmaşıklığın az olmasından dolayı tercih edilmektedir. Ancak ekip ile ilerlendiğinden kariyer riski bulundurmakta ve kurumsal yapılarda ilerlemesi zordur.

Hangi projelerde hangi modelleri kullanmak doğru olur?

Coğrafi olarak geniş, büyük kitlelerle çalışılacak ise evrimsel model tercih edilebilir.

Kişiye özel programlarda ise kodla düzelt modeli kullanılabilmektedir. Çünkü kişiye özel geliştirmeler ile ilerlediğinden dolayı bireysel ihtiyaca yönelik ürünler oluşmaktadır.

Gereksinimleri iyi tanımlanmış projelerde ise V tipi Modeli kullanmak bize kolaylık sağlamaktadır.

Ne çok büyük ne de çok küçük kitleli projelerde orta derece bir zaman sürecinde çevik modeli kullanmak idealdir.

Maliyet açısından değerlendirecek olursak en düşük maliyetli modeller Artırımlı, Kodla düzelt ve evrimsel modellerdir. En yüksek maliyetli model ise çevik modeldir.

Aynı zaman da çevik modellerde uzman gerekliliği çok yüksektir.

Oluşturulan fazların örtüşmesi de sadece artırımlı ve çevik modellerde mevcuttur.

Risk duyarlılığı ise evrimsel, çağlayan ve kodla düzelt modellerinde çok yüksektir. Böylelikle yapılan risk analizleri ve yoğun önlemlerle hata riski ve maliyet düşürülmesi hedeflenir.

Diğer modellere göre de çağlayan ve kodla düzelt modelinin esnekliği neredeyse hiç yok denebilecek kadar azdır ve bakımları zordur.

Sprial modeli uygulamasında diğer modellere kıyasladığımızda çok zor kalmaktadır.

Yukarıda kıyaslama yaptığımız birçok kriter model seçimimiz için çok önemli rol oynamaktadır.

Sonuç

Büyük resme baktığımızda aslında her model birbirinden ayrı bir parça halindedir. İstediğiniz proje çeşitlerine uygun birden fazla model bulunmaktadır. Her modelde projenize uygun bir parça seçebilirsiniz. Önemli olan projenizi iyi tanımak ve projenizdeki kriterleri belirlemektir. Müşterinin istek ve gereksinimlerini iyi tanımanız, kurumun yapısını iyi anlamanız çok önemlidir.