**YAZILIM YAŞAM DÖNGÜ MODELLERİ**

*Yazılım Yaşam Döngüsü Nedir ve Neden Önemlidir?*

Yazılım yaşam döngüsü genel olarak bir yazılımın sahip olduğu süreçlerin belirli bir plana ve standartlara uygun biçimde geliştirilip bu süreç içerisinde ortaya çıkabilecek hataların minimum düzeye indirilmesi için uygulanan aşamaların bütünüdür. Bu aşamalar yazılım ürününün üretiminin yanında ürünün kullanım sürecini de içermektedir. Bir yazılımın ortaya konma sürecinin yaşam döngüsü şeklinde ele alınması bu yazılımın daha organize bir biçimde geliştirilmesi ve üstüne çalışılan projenin yazılıma uyarlanabilirliğini arttırma açısından çok önemlidir.

Tabii ki birçok yazılım yaşam döngü modeli vardır fakat bir yazılımın yaşam döngüsü belirli temel adımlara dayanmaktadır. *Bu adımlar şu şekildedir;*

*1.* *Gereksinimler (Requirements):* Yazılım başlı başına müşteri için yapılan bir üründür ve bu ürünün müşteri tarafından oluşturulmuş belirli gereksinimleri vardır. Bu aşamada gereksinimler müşteriden alınır.

*2. Analiz (Analysis):* Sistemin işlevi ve gereksinimleri ayrıntılı olarak incelenir. Ortaya çıkabilecek riskler ele alınır. İleriye dönük bir süreç analizi yapılarak zaman planlaması yapılır.

*3. Tasarım (Design):* Yazılımın gereksinimlerini karşılayacak temel yapının tasarlandığı aşamadır.

*a) Üst Seviye ve Mimari Tasarım:* Sistemin genel yapısının, akış şemalarının ve ilgili modüllerin tasarımıdır.

*b) Detaylı Tasarım:* Yazılım içeren ekran, arayüz, veri yapıları ve bunların bileşenlerinin tasarımıdır.

*4. Gerçekleştirme (Implementation):* Yazılım projesinin kodlama ve testinin yapıldığı aşamadır.

*5. Bakım (Maintenance):* Ürünün tesliminden sonrahata ve eksiklerin giderildiği, yeni gereksinimlerin ortaya çıktığı ve bunların uygulandığı aşamadır.

*Yazılım Yaşam Döngü Modelleri Nelerdir?*

Daha önce de bahsettiğimiz gibi birçok yazılım yaşam döngü modeli vardır. Bunun sebebi projelerin her birinin birbirinden farklı özellikleri ve ek gereksinimleri barındırmasıdır. Bu farklılıklar projenin yapılış amacı, boyutu, kullanım yeri ve şekli gibi etkenlere bağlı olabilir. Bu sebeple birçok döngü modeli ortaya çıkmıştır.

Düzenleyici Süreç Modelleri:

*1. Barok Döngü Modeli:*

1970’li yıllarda ortaya çıkmış bir modeldir. Yazılım yaşam döngüsünün temel adımları doğrusal bir şekilde işlenir ve sürdürülür. Günümüzde artık kullanımına devam edilen bir model değildir. Çağlayan modelindeki gibi aşamalar arasında tanımlanmış bir geri dönüş yoktur. Bu modelin en belirgin özelliği günümüzdeki geçerli modellerden dokümantasyonu ayrı bir süreç olarak ele almasıyla ayrılmaktadır. Bu modelde yazılımın geliştirme ve test aşamaları tamamlanıp bir ürün ortaya çıkarıldıktan sonra dokümantasyon işlemi gerçekleştirilmektedir.

*2. Çağlayan-Şelale (Waterfall) Döngü Modeli:*

En eski ve tanınmış modeldir. Aynı zamanda en temel model olarak da bilinmektedir. Geçmişte yazılım dünyasında en popüler yazılım geliştirme modeliydi ve geleneksel model olarak da bilinmektedir. Bu modelin temelinde yazılım her bir aşamanın en az birer kez tekrar edilmesi ile geliştirilmektedir. Bu proje çok iyi bir şekilde işlenmiş ve gerçekleştirme aşaması öncesinde ayrıntılı bir şekilde tanımlanıp üretimi için az zaman gereken projeler için uygundur fakat kullanımı güncel koşullarla birlikte günümüzde oldukça azalmaktadır.

Bu modelde işler aşama aşama yapılmakta ve bir aşama bitmeden diğerine geçilmemektedir. Doküman oluşturmak her safha için ayrı ayrı uygulanmaktadır. Her şeyin dokümanı mutlaka olmalıdır. Bir safhanın bitmesi için gerekli koşullar, o safhanın dokümantasyonunun ve testinin tamamlanmasıdır.

Barok modeline nazaran bu modelde dokümantasyon sürecin doğal bir parçasıdır, ayrı olarak ele alınmaz.

Bu modelde aşamalar arası geri dönüşler bulunmaktadır ve bu geri dönüşler döngü sürecinde tanımlanmıştır. Fakat bu modelde aşamalar arası geri dönüş mümkün olsa dahi analiz aşamasında olabilecek en detaylı incelemenin yapılması ve gereksinimlerin tüm ayrıntılarıyla belirlenip tasarıma aktarılması gerekmektedir. Bu yüzden sürecin büyük bir kısmını bu aşamalar oluşturmaktadır. Bu yönüyle çağlayan modeli yazılımın ilerleyen aşamalarında ortaya çıkabilecek problemlerin ve yeni gereksinimlerin projeye dinamik olarak dahil edilmesi için uygun bir model değildir. Çünkü gereksinimlerin değişmesi kaçınılmazdır ve buna hazırlıklı olmak gerekir. Kullanıcı sürece yeteri kadar dahil olmadığından oluşacak sorunlar, yeni veya eksik gereksinimler süreci zaman ve maddi kaynaklar açısından sıkıntıya sokmaktadır. Her ne kadar sürecin sonunda müşteri geri bildirimde bulunabilse de bunların projeye uygulanması maliyeti büyük oranda arttırır.

Genel olarak projelerde yineleme gereklidir ve yazılımın kullanıcıya ulaşması uzun bir zaman alır. Bu modelde sürecin büyük bir bölümü analiz ve tasarıma ayrıldığı için kodlama ve test aşamalarına ulaşmak uzun bir zaman alır. Bu sebeple yazılımın üretiminde görev alan ekipler ortaya konulmuş bir ürün göremeyip mutsuz olurlar. Böylece motivasyon düşer, verim azalır ve projenin maliyeti artar. Tüm bunlar istenmeyen birçok durumu da beraberinde getirir. Tüm bu sebepler bu modelin günümüzde popülerliğini kaybetmesine neden olmaktadır.

*3. V Süreç (V-shaped) Modeli:*

V süreç modeli kendi içerisinde 3 aşamadan ve bu aşamaların farklı çıktılarından oluşmaktadır. Ayrıca bu model 2 ana yapıya (üretim ve sınama) dayanmaktadır. *V süreç modelinin aşamaları şu şekildedir;*

*a) Kullanıcı Modeli:* Geliştirme sürecine kullanıcının da dahil olması sağlanır. Sistem sınanır ve planlar ortaya çıkarılarak bu sistemin kullanıcı tarafından geri dönütü alınır. Proje tanımlarının son hali oluşturulur ve kullanıcı bilgilendirilir.

*b) Mimari Model:* Arayüz, modül, mimari sistem tasarımları ve bunlarla birlikte oluşan alt sistemlerin de tasarımı yapılarak bu sistemlerin uyum ve işlevlerine yönelik sınamalar yapılır.

*c)Gerçekleştirim Modeli:* Yazılım modüllere ayrılarak kodlanır ve bu modüller fonksiyonlarıyla birlikte sınanırlar.

Belirsizliklerin az, yürütülen işin tanımlarının belirgin bir şekilde ortaya konulduğu projeler için uygundur. Bilgi teknolojileri alanındaki projeler buna örnek olarak verilebilir. Bu modelde kullanıcının projeye katkısı fazladır. BT projelerinin iki aşamada kullanıcıya sunulması için uygun bir modeldir. İlk aşamada kullanıcı modeli ve projenin analiz ve kabul sınamaları tanımlanır. İkinci aşamada ise ilk aşamada oluşturulmuş kullanıcı modeli ve temel yapının tasarımı ortaya konularak gerçekleştirimi yapılmaktadır.

*4. Gelişigüzel Model:*

1960’lı yıllarda uygulanmaya başlanmıştır ve tek bir kişiye dayalı basit programlama içermektedir. Bu geliştirme şeklinde bir yöntem ya da bir model yapısı bulunmamaktadır. Bu sebeple bu yöntemi bir model olarak tanımlamak da pek doğru sayılmaz. Bu model genel olarak kişiye özgü bir geliştirme şekline dayanmaktadır. Bu sebeple yazılımın sistemli bir biçimde izlenebilirliği ve bakım yapılabilirliği pek de mümkün değildir.

*5. Helezonik (Spiral) Model:*

Bu model için prototip oluşturmak kritik bir önem taşımaktadır. Ayrıca bu modelde risk analizi de önem gösterilen bir diğer unsurdur. Bu iki unsurun her bir tekrarda uygulanması, oluşacak hataların erken farkına varılmasını ve düşük maliyetli olarak atlatılmasını sağlar. Aynı zamanda bu prototipler kullanıcı ile etkileşimi arttırdığı ve kullanıcıya somut bir örnek ortaya koyduğu için kullanıcının geri dönütleriyle birlikte projenin verimliliğini büyük ölçüde arttırır. Bu model kendi içinde 4 temel yapıdan oluşmaktadır. *Bu aşamalar şunlardır;*

*a) Planlama:* Üretilecek prototip için plan yapılır, amaçlar belirlenir ve bundan önceki adımda üretilen ara ürün ile ilişkilendirilerek bir bütün haline getirilir.

*b) Risk Analizi:* Ortaya çıkabilecek sorunların ve risklerin projeye verebileceği zararlar araştırılarak ayrıntılı olarak incelenir.

*c) Üretim:* Prototiplerin ya da ürünün üretilmesidir.

*d) Kullanıcı Değerlendirmesi:* Prototiplerden yola çıkılarak kullanıcıların yaptığı değerlendirme ve geri dönütlerdir.

Tüm bu aşamalar projenin verimliliğini arttırıcı yönde rol oynarlar.

Genel anlamıyla bu modeli değerlendirecek olursak. Risk analizi bu model için önemli bir basamaktır.

Her bir tekrar bir fazı ifade eder ve bu fazlar kendi içinde belirli aşamalardan oluşur. Doğrudan gereksinim, analiz, tasarım, gerçekleştirme… vs. birer fazı ifade etmezler. Yinelemeli olarak ilerler ve artımsal bir yaklaşım izlenir. Prototipler projede önemli bir faktördür.

Bu model birçok açıdan sürece katkı sağlar.

*Bunlar şu şekildedir;*

*-* Kullanıcılar süreç boyunca prototiplere ulaşım sağlayarak projenin güncel durumunu görürler. Böylece yazılımın eksik ve hatalı bir yeri varsa kullanıcıdan alınan geri dönütlerle birlikte değerlendirilip erkenden müdahale edilerek düzeltilir.

*-* Gerçekleştirme aşaması proje geneline prototiplerle yayıldığı için proje sahibi ve yazılım ekibi proje üzerinde daha kolay bir izlenim oluşturur. Bunula birlikte süreç daha iyi bir şekilde yönetilerek maliyet ve zaman açısından minimum kayıp sağlanmış olur.

*-* Yazılım ekibi sürecin gerçekleştirme aşamasına daha erken dahil olduğu için ortaya çıkan ürünlerle motive olur ve verimlilik artar.

Bu proje her ne kadar büyük ölçekli projeler için avantajlı bir metot olsa da düşük bütçeli ve küçük ölçekli projeler için pek de öyle değildir. Çünkü çok fazla çıktıyla birlikte fazla sayıda doküman hazırlamak gereklidir. Bunlar da beraberinde süreyi ve maliyeti arttırmaktadır.

*6. Artımsal Geliştirme (Incremental Development) Süreç Modeli:*

Bu modelde yazılım ürünü müşteriye tek seferde teslim edilmez. Ortaya eksik ama çalışabilir bir ürün koyulur. Bu ürün müşteriye sunulur ve ürünün kullanımı ürünün geliştirmesiyle paralel olarak ilerler. Bu sebeple ürünün her bir sürümü kendi başına bir üründür. Bu sürümler ürünün ileride gereksinimlerinin belirlenmesinde kritik rol oynarlar. Ürünün aynı zamanda kullanılıyor olması ve kullanıcının bu ürünü deneyimleyip geri dönüşlerle yazılım ekibini bilgilendirmesi zamana yayılmış ve sistemli bir geliştirme ortamı oluşturur. Her bir yeni sürüm bir önceki teslim edilen sürümün hatalarının ve eksiklerinin düzeltilmesi, yeni ortaya çıkan gereksinimlerin eklenmesiyle daha iyi hale gelir ve zamanla ürünün son hali ortaya çıkmaya başlar. Öncelikli gereksinimleri karşılayan işlevler daha çok test edilerek gittikçe daha az hataya yer verilir. Bu yöntemle proje zaten çalışır bir halde kullanıcıya ulaştığı için kullanıcı sürece büyük oranda dahil olur. Böylece projenin başarısız olma riski azalır ve kullanıcının gereksinimlerini karşılayan, müşteri odaklı bir ürün ortaya çıkarılmış olur. Her ne kadar birçok avantajı da olsa ürünün gelişim evreleri önceki sürümün üstüne koyarak ilerlemeye dayalı olduğu için ürünün çok iyi bir şekilde tanımlanması gerekir ve bu modelde her bir sürüm kendi içinde tekrara yer vermez bu sebeple değişiklikler yalnızca bir sonraki sürümle birlikte kullanıcıya ulaşabilmektedir.

*7. Kodla ve Düzelt (Code and Fix) Döngü Modeli:*

Küçük ölçekli (birkaç yüz satırlık) programlar için uygundur. En başta yazılım ürününün ilk sürümü gerçekleştirilir. Geliştirme aşaması sürekli olarak, ürün en son halini alıncaya kadar devam eder. Ürünün aşaması vardır fakat zorlu bir süreçtir çünkü ürünün geliştirme aşamasında sistemli bir dokümantasyona yer verilmez bu da ürünü tanımlamayı zorlaştırır ve bakım yapmak zor bir durum haline gelir. Ürünün bir emeklilik (retirement) aşaması vardır. Tüm bu sebepler yüzünden büyük projelere uygulanabilirliği çok azdır. Çünkü dokümantasyonun olmaması, bakımın az ve zor olması gibi durumlar projenin kaynağını ve sürdürülebilirliğini tüketici birer unsur haline dönüşür. Yazılım geliştirmenin en kolay fakat en pahalı yoludur. Çünkü sürecin belirli bir aşamasından sonra (kodlamadan sonra) ürün üzerinde yapılacak her bir değişikliğin maliyeti katlanarak artar. Bu da bize çok yüksek maliyetler doğurur. Dokümantasyonun da olmaması bu bakım sürecini ve yapılacak değişiklikleri çok daha komplike bir hale getirerek her türlü açıdan maliyeti arttırır. Bu model genel olarak çok tecrübeli ve profesyonel firmalarca tercih edilmese de yazılım geliştirmenin kolay olması sebebiyle küçük ve nispeten daha az tecrübeye sahip firmalarca uygulanmaktadır.

*8. Prototipleme (Prototyping) Modeli:*

Modelin temeli bir prototip üretip tekrar bu prototipi istekler ve gereksinimler doğrultusunda geliştirmeye dayanmaktadır. Bu sebeple gereksinimlerin iyi analiz edilip yazılım ekibince anlaşılması önemlidir. Sonrasında hızlı bir şekilde planlama, tasarım ve kodlama işlemleri yapılır ve müşteriye kullanılabilir bir prototip sunulur. Müşteri bu prototipi değerlendirir ve ürün tekrardan istekler ve gereksinimler doğrultusunda şekillenir. Bu şekilde ürün prototipi müşterinin dönütleriyle son haline ulaşır. Elde edilen son prototipin devamında ürünün son halinde neler olması gerektiği üzerine müşteriyle konuşulur ve prototip üzerinden belirli miktarda tekrar sonucunda bir ürün meydana getirilir. Bu model doğrusal modellemenin döngüye çevrilmiş halidir.

Modelin içerisine kullanıcı da dahil olduğu için süreç boyunca sistemin gereksinimleri yazılım ekibinin yanında kullanıcı tarafından da görülebilir. Böylece eksikler, hatalar ve yeni gereksinimler erken fark edilip hızlı bir biçimde ürüne uygulanır. Bu da projenin riskini ve maliyetini büyük oranda azaltır.

Bu modelin birçok avantajı olduğu gibi dezavantajları da vardır. Sürekli ve hızlı bir biçimde işleyen bir süreç olduğu için içerisinde dokümantasyon aşaması bulunmaz. Bu da belirli bir süreden sonra üründe yapılacak değişikliklerin ve bakımın zor ve maliyetli bir durum haline geleceğini gösterir. Prototiplerin geliştirilmesi ve düzeltilmesine büyük önem gösterilmesi gerekmektedir. Aksi taktirde kullanıcı oluşturulan prototipleri ürünün kendisi ile karıştırabilir. Burada şunu unutmamak gerekir. Prototip kendi başına ürünün kendisi değildir. Prototipin son hali dahi daha bitmemiş bir üründür. Prototipler ürünün şekillenebilmesi için ortaya koyulan birer taslak niteliği taşımaktadır.

*9. Evrimsel Geliştirme (Evolutionary Development) Modeli:*

Büyük firmalar tarafından tercih edilmektedir ve ilk tam ölçekli modeldir. Ürünler aşama aşama güncellenerek üretilir ve her aşamada üretilen ürün tam işlevselliğe sahiptir. Modelin başarısı ilk evrimin başarısına bağlıdır. Genel olarak bu modelin işleyişi şu şekildedir. Pilot uygulama kullan - Test et - Güncelle - Diğer birimlere taşı.

Evrimsel geliştirmenin iki ayrı çeşidi vardır. *Bunlar şu şekildedir;*

*a) Keşifçi Geliştirme (Exploratory Development):* Odağımız müşteridir ve müşterinin gereksinimleri doğrultusunda ürün geliştirilir. Gereksinimlerin en başından iyi belirlenmesi önemlidir.

*b) Atılacak Prototipleme (Throw-Away Prototyping):* Odağımız sistemdir. Sistemin gereksinimleri doğrultusunda ürün geliştirilir. Gereksinimler tam anlaşılamaz. Çünkü gereksinimlerde müşteri bakış açısı yoktur.

Evrimsel geliştirme kullanıcının gereksinimleri üzerine yoğunlaşıldığı için kullanıcının gereksinimleri daha iyi anlaşılır. Kullanıcın dönütleri önemlidir. Böylece değerlendirmelerle risk ve son üründeki hatalar azalır.

Süreç boyunca her aşamada üretilen ürün tam işlevselliğe sahip olduğu ve geri bildirimlerle yürütüldüğü için süreç uzun ve görünürlüğü azdır. Düzenli bir ürün oluşumu yoktur. Sürekli değişiklik yapmak yazılım için zararlı bir durumdur. Değişiklik denetimi yoktur. Bakımı zor ve maliyeti yüksektir.

*10. Araştırma Tabanlı (Resource Based) Model:*

Yap-at prototipi olarak bilinir. Yapılan işlerden elde edilecek sonuçlar belirsizdir. Araştırma ortamları tamamen belirsizlik üzerine çalışan ortamlardır.

*11. Formal Sistem Geliştirme (Formal System Development) Modeli:*

Matematiksel bir teknik olarak karşımıza çıkmaktadır. Bu model temel olarak karmaşık sistemleri ve programları geliştirmeyi destekler niteliktedir. Bu model daha kullanıcının ilk kullanım anından itibaren hataların minimum düzeyde olmasını öngörür ve bunun için de belirtim, tasarım ve geçerleme kullanarak yazılımın doğruluğunun geliştirilmesi üzerinde durur. Yazılım sürekli üstüne koyularak artımlar ile geliştirilmektedir. Önceki ürünlerle tümleştirme söz konusudur ve bu tümleştirme sürekli olarak devam eder. Bununla birlikte yazılımın eski sürümünün üzerine yeni bir versiyonu da eklenerek aradaki yazılımsal hatalar minimuma indirgenerek fonksiyonellik arttırılmış olur. Bu modelde kodu ilk başta yazarken tamamen doğru yazmak önemli bir noktadır ve doğru yazılan bu kodun test aşamasında bir kez daha doğruluğu test edilerek ortaya ürünün son hali çıkartılmış olur.

Bu model yazılımın sürekli kendi içinde karşılaştırılmasından dolayı belirsizlikleri, eksikleri ve uyumsuzlukları kolayca görmemizi sağlar. Bu şekilde hatasız bir yazılım geliştirme imkânımız olur. Her bir iterasyondan sonra artarak devam eden efektif çözümler sunulur. Karmaşıklığa yer verilmez.

Bunların yanında belirli dezavantajları da vardır. Kodun bütünüyle kusursuz olması ve sürekli birbiriyle karşılaştırılması uzun zaman alan ve pahalı bir yöntemdir. Bu işlemler çok fazla teknik bilgi gerektirir. Bu yüzden teknik olmayan personelle iletişimi zordur. Küçük bir ekip dahi bu model üzerinde çalışacaksa her bir üyenin geniş bir bilgi yelpazesine sahip olması gerekmektedir.

*12. Bileşen Tabanlı Geliştirme (Component Based Development) Modeli:*

Bu model geleneksel modellerden farklı olarak sistemi fonksiyonel olarak değil yapı olarak ayrıştırmaktadır. Bu ayrıştırma sistemin yapı taşlarını ortaya koymak için sistemin yukarıdan aşağıya (genelden özele) doğru bir ayrıştırımı şeklindedir. Bu model dört ana aşamadan meydana gelmektedir. Sistem spesifikasyonu, sistem ayrıştırımı, bileşen spesifikasyonu ve entegrasyon. Bu aşamaların sonucunda bir de test aşaması bulunmaktadır.

*13. Yeniden Kullanıma Yönelik Geliştirme (Re-use Based Development) Modeli:*

Günümüzde gittikçe daha fazla kullanılmaya başlanmaktadır*.* Geliştirilecek olan yazılımın daha önce kullanılmış, hazırlanmış ya da dışarıdan temin edilmiş yazılımlar çerçevesinde oluşturulması temeline dayanır. Bu model genel olarak organizasyonu ve altyapısı belirli bir seviyeye ulaşmış kurumlarca tercih edilmektedir. Çünkü güvenilir bir kaynaktan işlevi ve verimliliği yüksek yazılım veya yazılım parçaları bulmayı ve organize bir biçimde bunları işleyip yeni bir ürün ortaya koymayı bilmek gerekmektedir. Bu modelde kaynak kontrolü mümkündür. Maliyet kontrolü halihazırda yazılmış yazılımlar kullanıldığı için daha olasıdır. Geliştirilecek yazılıma uygun yazılımlar kullanıldığı için basit ve anlaşılırdır. Önceden ortaya konulan sınıflar tekrar kullanılabilir ve yazılımın geliştirme süreci kısa tutulmuş olur. Fakat kodları birbirine entegre etmek uzmanlık gerektirir. Ürünün bir bütün halinde kusursuz çalışması gerekmektedir ve bu zorlu bir süreçtir. Bu yüzden projenin başarıya ulaşacağı kesin değildir.

*14. Big-Bang Modeli:*

Bu model az bir planlama ile daha çok kodlama üzerine yoğunlaşılarak yazılım geliştirmeye dayanır. Kodlama açısından her türlü kaynağı kullanmaya odaklanılır. Gereksinimler önceden net bir biçimde belirlenmez, anlık olarak ihtiyaçlar ortaya çıkar ve bu ihtiyaçlar giderilir. Genellikle küçük ölçekli proje ekiplerince yine küçük projelerde tercih edilmektedir. Belirli bir süreç ve bu sürece ait önceden belirlenmiş aşamalar yoktur.

*15. Birleşik Süreç (Unified Process):*

Nesneye dayalı yazılım geliştirme süreçlerinin en iyi özelliklerinin bir araya getirilip bütünleştirilmesi ile oluşturulmuş yazılım geliştirme sürecidir. Yinelemeli, arttırmalı, evrimsel ve risk güdümlü bir yazılım geliştirme sürecidir.

*16.Çevik Yazılım Geliştirme Süreci (Agile Programming):*

Gereksinimler ve çözümler kendinden örgütlü (Self-organized) ekiplerin birlikte çalışması ile oluşturulur. Bu yazılım geliştirme sürecinin içerisinde sık sık denetimler gerçekleştirilir. Ekiplerin hem bireysel hem de topluluk olarak adaptasyonuna ve motivasyonuna önem gösterilir. Bu süreç izlenebilirliği yüksek, liderlik ve ekip bilinci oluşmuş, hızlı ama kaliteli yazılımların ortaya çıkarıldığı, müşteri ihtiyaçlarının en iyi şekilde giderildiği, planlı, müşteri ihtiyaçları ve firma amaçlarını bir araya getiren bir yaklaşıma sahip olmalıdır. Çevik yaklaşımlar temelde doğrudan yazılım geliştirme yaklaşımı olarak ele alınmamalıdır. Çünkü çevik yaklaşımlar yazılım geliştirme yaklaşımı değildir. Geliştirme süreçlerinin bir bütünüdür.

*Bu sürecin genel özellikleri şu şekildedir;*

-Kapsamlı bir dokümantasyona sahiptir.

-Projede müşteri ile etkileşimli olarak çalışılır.

-Bir plan vardır ve ortaya çıkan değişiklikler bu plan üzerinden yürütülür.

-Ürünü hızlı teslim etme isteği vardır.

-Sürecin asıl değerlendirilme ölçütü ürünün çalışan son halidir.

-Günlük ve düzenli olarak iletişim kurulur.

-Komplike olmayan basit sistemler tasarlanmaya çalışılır.

-Koşulların değişmesi halinde koşullara adapte olunmaya çalışılır.

-Proje motivasyonu ve güvenilirliği yüksek kişiler üzerinden yürütülür.

-Yüz yüze iletişime önem verilir.

Bu süreç genel olarak birçok avantaja sahiptir. İnsanın istekli bir biçimde çalışmasını sağlar. Sürekli çıktı üretilir, müşteri projeye dahil olur ve motivasyon kaybı önlenir. Döngüler kısadır ve verimlilik yüksektir. Sürdürülebilirliği vardır. Planlı ve yürütülebilir bir proje ortaya konur. Takım olarak organize olunur.

Fakat az da olsa dezavantajları da vardır. Kurumsal olarak uygulanması zordur. İhtiyaçlar değiştikçe sık çalışma gerektirir. Takım olarak proje baskısı oluşur.

Çevik modellerden birçoğu günümüzde popülerliğini yitirmiştir.

*a) Uç Programlama (XP-Extreme Programming):* Belirlenen tüm gereksinimler küçük modüllere ayrılır ve gruplara dağıtılır. Grup içi iletişime ve geri dönütlere çok önem verilir. İletişim yüz yüze olmalıdır. İletişim yalnızca ekiplerin içerisinde yoktur. Ekipler arasında ve müşteri ile kipler arasında da olmak üzere herkes tüm sürece dahil olur. Böylece hatalar erken fark edilir. Bilgiler kısa sürede elde edilir ve yazılım geliştirme süreci durmadan devam eder. Yazılımın karmaşık olması istenmeyen bir durumdur, basitlik ön planda tutulur. Sürekli ve çok sayıda çıktı alınarak projenin o anki durumu değerlendirilir. Daha sonra doğabilecek sorunlar da bu aşamada çözüme kavuşturulur. Projenin her türlü aşamasında cesaret çok önemli bir faktördür. Her türlü olumsuzlukta dahi yılmadan proje işletilmeye devam edilmelidir. Başarısız da olunsa yeniden başlayabilmek gerekmektedir. Başarısızlıktan hiçbir yazılımcı korkmamalıdır ve motivasyonunu kaybetmemelidir. XP içerisinde dört değer vardır.

*Bu dört temel şunlardır;* İletişim, Basitlik, Geri bildirim, Cesaret*.*

*Bunun yanında XP 12 farklı pratiği ön görür;* Planlama oyunu, Ekipte müşteri, Önce test, Basit tasarım, Çiftli programlama, Sürekli entegrasyon, Kısa aralıklı sürümler, Yeniden yapılandırma, Ortak kod sahiplenme, Metafor, Kodlama standardı, Haftada 40 saat.

*b) SCRUM:* Günümüzde oldukça fazla tercih edilen bir metottur. Karmaşık yazılımları küçük parçalara bölerek geliştirmeye dayanır. Bu küçük parçalara ‘sprint’ ismi verilir. Bu metot karmaşık ortamlarda adım adım yazılım geliştiren ekiplere oldukça uygundur. Gereksinimler kolay bir şekilde tanımlanamıyorsa ve projenin ilerleyen safhalarında ne gibi durumlarla karşılaşılacağı konusunda net bir fikir yoksa bu tarz durumlar için SCRUM metodu en uygun yaklaşımdır. İletişime ve periyodik olarak çalışmaya önem verilmektedir. Her bir yineleme en fazla 30 gün sürmekte ve günlük olarak 15 dakikalık toplantılarla projenin güncel durumu takip edilmektedir. Bu toplantıların temel soruları vardır. Dün ne yaptın? Bugün ne yapacaksın? Seni engelleyen neydi? Gibi sorular toplantının temelini oluşturur. Bu metotta müşteri ile ilişki çok önemlidir ve iletişim sürekli olarak sağlanır. Projenin durumu, gereksinimleri gibi konular üzerine müşteri ile sürekli iletişim halinde olmak gereklidir. Böylece riskler hızlıca tespit edilir, hatalar erken safhada düzeltilir ve maliyet azaltılmış olur. Scrum metodunda organizasyon ve planlama da büyük bir önem taşımaktadır. Her sprint öncesinde takımların belirleneceği, gereksinimlerin oluşturulacağı, risk analizlerinin yapılıp maliyetlerin gözden geçirileceği planlama toplantılar yapılmaktadır. Gereksinimler önceliklendirilir ve bu önceliklere göre projenin hangi aşamasında hangi gereksinimler üzerinde çalışılacağı ürün sahibine taahhüt edilir. Gerekli durumlarda da dokümantasyon yapılır. Fakat bu dokümantasyonun en az insan çabası ile en verimli şekilde üretilmesi amaçlanır. Tüm bu organizasyondan ve projenin değerlendirilip yürütülebilir bir duruma getirilmesinden sonra daha önce de bahsettiğimiz 15 dakikalık günlük scrum toplantıları gerçekleştirilir ve dinamik bir biçimde süreç takip edilir.

*Product Backlog***:** Müşteri ile iletişime geçilerek belirlenen, yüksek öncelikli proje gereksinimleridir.

*Sprint Backlog***:** Projelerin kendi içinde ayrıldığı zaman dilimleridir. Sprintlerin her biri 15- 30 gün civarındadır ve bir proje birden fazla sprintten oluşabilmektedir.

*Scrum Daily Meeting:*Her gün Scrum takımı ile gerçekleştirilen maksimum 15 dakika sürmesi ön görülen ve genel durum hakkında bilginin alındığı toplantılardır. 3 temel soru üzerinde durulur. Dün ne yaptın? Bugün ne yapacaksın? Seni engelleyen neydi?

Scrum’un tüm bu yapısından da anlaşılacağı gibi scrum üç temel unsurdan meydana gelmektedir. *Bu unsurlar şu şekildedir;*

*1. Roller (Roles):*

*Ürün Sahibi (Product Owner):* Müşteri ile iletişimi sağlayan kişidir. Müşterinin kendisi olabileceği gibi, farklı birisi ya da scrum ekibinden birisi de olabilir.

*Scrum Yöneticisi (Scrum Master):* Scrum ekibini organize eden ve genel sorunlara çözüm bulan deneyimli scrum ekip üyesidir.

*Scrum Takımı (Scrum Team):* Birbiriyle iletişim halinde olan ve aynı hedef için çabalayan kişilerin oluşturduğu takımdır. 5- 9 kişiden oluşur.

*2. Toplantılar (Meetings):*

*Sprint (Koşu) Planlama (Sprint Planning):* Gereksinimler belirlenir, takımlar oluşturulur, risk ve maliyet değerlendirilmesi yapılır, takımlara gereksinimler dağıtılır.

*Sprint (Koşu) Gözden Geçirme (Sprint Review):* Sprint başlangıcında planlama toplantısı yapılır. Toplantı iki aşamadan oluşur. İlk aşamada Ürün sahibi ve Takım bir araya gelerek gereksinimleri gözden geçirip gereksinim elemanlarının hedeflerini ve içeriklerini belirler. İkinci aşamada ise takım üyeleri gereksinimleri önceliklerine göre sıralarlar. Bu uygulama Scrum’ın anahtar uygulamasıdır. En sonunda da ürün sahibine önceliklendirilmiş gereksinimlerden ne kadarının yapılacağı taahhüt edilir.

*Günlük Scrum Toplantısı (Daily Scrum):* Günlük olarak sprint’in genel durumunun ele alındığı 15 dakikalık toplantılardır. Takımın ilerleyişi ve karşılaşılan engeller görülmüş olur.

*3. Bileşenler/Araçlar (Artifacts):*

*Ürün Gereksinim Dokümanı (Product Backlog):* Dinamik olarak değişen bir dokümandır. İçerisinde projenin gereksinimleri ve yapılması gerekenlerin bir listesi bulunur.

*Sprint (Koşu) Dokümanı (Sprint Backlog): Mevcut sprint’in iş ve görevlerini içerir. Ana amaç, son ürünün çalışabilir ve işlevselliği olan bir parçasını elde etmektir.*

*Sprint Kalan Zaman Grafiği (Burndown Chart):* Spint boyunca işlerin ne kadar yapıldığını ve normal koşullarda ne kadarının yapılmış olması gerektiğini gösteren grafiktir. Bu grafikten mevcut projenin zamanında bitip bitmeyeceğini ve ne tür bir strateji izlenmesi gerektiği konusunda bilgi edinilebilir.

Scrum Günümüzde Neden Popüler?

Scrum’ın bu kadar popüler olmasında birçok etken bulunmaktadır. Ama yalnızca yazılım alanında değil her türlü alanda uygulanabilir bir model olduğunu da unutmamak gerekir. Scrum’ın oldukça fazla bir kitle tarafından kullanılmasının başlıca nedenleri şu şekildedir;

-Günlük olarak projenin girdisi çıktısı takip edilir ve süreç değerlendirmesi yapılır.

-Kullanıcı ile etkileşim yüksektir.

-Sık sık, işlevli ve sürekli gelişen bir ürün teslim etmeyi hedefler.

-Yazılım ekibinin sosyalliği gibi bireylerin ihtiyacı olabilecek her türlü gereksinim de önemsenmektedir.

-Organizasyon ve takımlar arası iletişim önemlidir.

-Geri bildirimlere değer verilir ve bu geri bildirimler ışığında ürün geliştirilir.

-Basitlik ön plandadır.

-Yazılım ekibinin hatalardan korkmaması ve yapılan yanlışlar en kısa zamanda geri dönebilme isteğini kendinde bulması önemlidir.

-Birçok sisteme uyarlanabilir bir modeldir.

-Planlama yapılmaktadır.

-Gereğinden fazla herhangi bir aşamada zaman kaybedilmez. Örneğin dokümantasyon işlemi yalnızca gerekli oldukça, net bir biçimde ve en az miktarda insan kaynağı kullanılarak yapılır.

-Çalışan bir ürün ortaya koymak temel amaçtır.

İşte scrum’ın bu derece fazla avantajının bulunması ve birçok alanda farklı farklı sistemlere uyarlanabilir olması. Scrum’ın günümüzde gittikçe daha da fazla popülerleşmesini sağlamıştır.

Hangi Projede Hangi Modeli Kullanmalıyız?

Aslında bir proje için doğrudan bu model kullanılmalıdır demek pek de mümkün değildir. Her türlü projeye her türlü model uygulanabilir. Fakat projenin yapısına uygun bir model seçilmediği zaman projenin maliyeti, kaynak tüketimi, verimliliği ve sürecin uzunluğu gittikçe artar. Hatta proje tamamen iptal edilme aşamasına dahi gelebilir. Bu sebeple projeye ve sürece uygun bir model seçmek her iş için kritik bir önem taşır.

Gelelim hangi projede hangi modeli kullanacağımıza. Modeller net bir skalaya koyulamasa da belirli başlı özellikleri ve içerdikleri avantaj ve dezavantajlarla hangi projelerde uygulanacakları kısmen belirlenmiştir.

1. Küçük ve orta ölçekli, kısa zamanda bitirilmesi hedeflenen projeler için çevik yöntemler uygundur.

2. Birkaç yüz satırdan oluşan kişisel yazılımların geliştirilmesi için kodla ve düzelt yöntemi uygundur.

3. Gereksinimleri çok iyi belirlenmiş, süreci iyi tanımlanmış ve az zaman gerektiren projelerde çok tekrara dayanarak hataları kolayca görmemizi sağladığı için cağlayan modeli uygundur.

4. Büyük ölçekli firmalarca yürütülecek olan ve coğrafi açıdan geniş sahalarda yürütülecek projeler için evrimsel geliştirme modeli uygundur.

5. Bilgi teknolojileri alanındaki iyi tanımlanmış projelerde v süreç modeli uygundur.

6. Uzun zaman alabilecek ve eksik işlevsellikle çalışabilecek projeler için artımsal geliştirme modeli uygundur.

7. Maliyeti yüksek ve uzun zaman alabilecek projeler için spiral model uygundur.

Utku Aydın

200601019

Bilgisayar Müh.

https://utkuaydinn.medium.com/yazilim-yaşam-döngü-modelleri̇-c8af07b496e0

<https://github.com/utkuaydinn/Yazilim-Yasam-Dongu-Modelleri>