YAZILIM YAŞAM DÖNGÜ MODELLERİ

Yazılım yaşam döngüsünde birden fazla model ele alınır. Bu modellerinin bir değil de daha fazla olmasının nedenleri; yazılım projesinin büyüklüğü, yazılım projesinin kimler için kullanılacağı gibi durumlarla sıralanabilir. Bu yazıda bazı yazılım yaşam döngü modellerinden bahsedilecektir.

· Barok ve Gelişigüzel Yaşam Döngü Modelleri

Gelişigüzel Model: Bir yöntem veya model yoktur. Kişiye özel yapılır ve gözetilmesi oldukça zordur. 1960'larda kullanılmıştır.

Barok Modeli: Yaşam döngü adımları temel olarak doğrusal bir şekilde gözden geçirilir. Döngü yoktur ve belgeleme ayrı bir süreç olarak ele alınır. 1970'lerde kullanılmıştır.

Bu iki model de günümüzde kullanılmamaktadır. Bu yüzden de aslında güncel bir yaşam döngü modeli olarak adlandırılmazlar.

· Şelale (Waterfall) Yaşam Döngü Modeli

Bu model diğer güncel yaşam döngü modellerinin temelidir diyebiliriz. Geleneksel bir modeldir ve geçmişte en popüler döngü modeliydi. Bu modelde her aşama en az bir kez tekrar edilir. Şelale modeli çok iyi tanımlanmış ve kısa sürede bitebilecek projelerde kullanılır. Eğer barok ile kıyaslayacak olursak şelale modelinde belgeleme sürecin içerisindedir. Bir adım tamamlanmadan diğer bir adıma geçilemez. Karşılaştığımız sorunlar ise şunlardır:

Ø Genelde yazılımların hazırlanması ve kullanıma sunulması uzun bir süreçtir ancak şelale modeli bunun için uygun değildir.

Ø Kullanıcı geliştirme sürecinin içinde olmadığı için olası bir sorun çok geç fark edilebilir bu da yazılım maliyetinin artmasına ve sürecin daha uzun sürmesine neden olur.

Ø Genelde yazılımcılar bir an önce programı kodlama ve programı çalıştırma amacında oldukları ve bu modelde kodlama sadece küçük bir kısımda kullanıldığından ekip mutsuzlaşır ayrıca süreç de uzayacağı için yöneticiler bunun bir finansal kayıp olacağını düşünebilirler.

· V Süreç Modeli

V- model şelale modelinin gelişmiş hali olarak düşünülebilir. Bu model belirsizliklerin az olduğu, iş tanımlarının ise belirgin olduğu projelerde kullanılır. Bu tür projelere örnek olarak Bilgi Teknolojileri için geliştirilen projeler verilebilir. Sol taraf üretim, sağ taraf ise sınamadır. Bu döngü esas olarak 3 modelden oluşmaktadır. Kullanıcı Modeli, Mimari Model, Gerçekleştirim Modeli.

Ø Kullanıcı modelinde kullanıcının istekleri ve proje için tanımları kullanılır ve tamamlanmış hali teslim edilir.

Ø Mimari modelde projenin tasarımı ve bunların deneme işlemlerinden oluşur

Ø Gerçekleştirim modeli ise kodlama ve bunların denenmesinden oluşur.

V- modelde karşılaşılan sorunlar ise şunlardır: Aşamalarda tekrar bulunmaz ve risk çözümleme için ayrılan bir yer yoktur

· Helezonik (Spiral) Model

Helezonik modeli diğerlerinden ayıran özellik ise risk analizinin ön planda olması ve prototip oluşturulmasıdır. Risk analizi ön planda olduğu için hataları erken giderme imkânı tanıyabilir. Prototip oluşturma da her aşamada olduğu için kullanıcı da her aşamada yazılım projesinin bir parçasını görme imkanına sahip olur bu da sorunların azalmasını sağlayabilir. 4 temel aşamadan oluşur:

Planlama: Her aşamada olan ara ürün için bir planlama yapılır.

Risk Analizi: Risklerin araştırılması, belirlenmesi ve çözülmesi.

Üretim: Ara ürünün/ürünün üretilmesi.

Kullanıcı Değerlendirme: Oluşturulan ara ürünün sonucunda kullanıcıdan alınan geri dönütlerin değerlendirilerek diğer aşamaya geçilmesi.

Helezonik modelin sorunları; küçük ve düşük riskli projeler için çok pahalı bir sistem olması, karmaşık bir içeriğe sahip olması, uzun sürmesi ve fazla dokümantasyondan oluşması olarak sıralanabilir.

· Artımlı Geliştirme Modeli

Artımlı Geliştirme Modelinde proje parçalara bölünür ve kullanıcının önceliğine göre bu parçalar sıralanır. Sıralanan bu parçalar bittiğinde sırasıyla birer ara ürün geliştirilmiş olur ve bu ara ürünler de kullanıcı tarafından kullanılır. Ara ürünler her seferinde bir öncekinin üstüne bir şeyler katarak çıkartılır. Yani bu modelde bir taraftan üretim kısmı sürerken diğer tarafta ise kullanım kısmı sürer. Artımlı geliştirme modeli uzun zaman alabilecek ve ürünün eksik işlevsellikle çalışabileceği türdeki yazılımlar için uygundur. Bu model ile sistemin başarısız olma olasılığı azalır, ara ürünler yazılımın geliştirilmesinde önemli bir yere sahip olur. Dezavantajları ise her bir parçanın kendi içinde tekrar etmesine izin verilmez bu yüzden de bir ara ürünün bitip diğeri başlayana kadar herhangi bir değişiklik yapılamaz, parçaları oluşturmak için de bu sistemin detaylı bir şekilde tanımlanması lazımdır.

· Kodla ve Düzelt Yaşam Döngü Modeli

Küçük programlar için kullanılabilir, direkt ürün gerçekleştirilir ve emeklilik safhası vardır. Büyük projeler için kullanılamaz, bakım az vardır ama zordur, ürünü hazırlar ve kullanıma sunarsınız.

· Evrimsel Geliştirme Modeli

İlk tam ölçekli modeldir. Büyük alanlara yayılmış, büyük firmalar için önerilir. Her aşamada üretilen ürün tam işlevselliğe sahiptir. Modelin başarısı ilk evrimin başarısına bağlıdır.

Çevik Modeller

Çevik modeller yazılım projelerindeki başarı oranlarını ve ekip içi iletişimi arttırmak amacıyla ortaya çıkmıştır. Bazı çevik yazılım geliştirme modelleri şunlardır:

· Extreme Programming (XP)

Kent Beck ve arkadaşları tarafından 1996 yılında kurulmuştur. 4 temel maddeden oluşur: Basitlik, Cesaret, Geri Dönüş, İletişim.

Basitlik: Yazılan kodun ve yapılan işin sade, anlaşılır ve karmaşık olmadan yapılmasını gerektirir. Uzun uzun dokümantasyondan uzak durulur.

Cesaret: Yapılan işte cesur olunmalıdır, projelerin üstüne korkmadan ilerlenmelidir. Bir kodun gerekirse tamamen silinip yeniden yazılması sağlanmalıdır.

Geri Dönüş: Geri dönüşler ile oluşabilecek hatalar azaltılır/ortadan kaldırılır. Müşteri ile yazılım ekibi birbirleriyle iletişim halindedir.

İletişim: İletişim, projelerde önemli sorunlardan birisidir. XP ise bunu aşmaya çalışmaktadır. Ekip içi iletişime önem verir ve artırılması için çalışır.

· SCRUM

SCRUM büyük projeleri parçalara bölerek her birine “sprint” adını verir. Projeyi böler ve her bir sprinti teker teker geliştirir. SCRUM’da ekip içi iletişim çok önemlidir öyle ki her gün “SCRUM MEETINGS” denen toplantılar yapılır. SCRUM’da üç temel kavram bulunmaktadır, bunlar: Roller, Toplantılar ve Bileşenler/Araçlardır.

Ø **Roller:** Ürün sahibi, Scrum yöneticisi ve Scrum takımından oluşur. Ürün sahibi projenin beynidir denilebilir. Scrum yöneticisi, takımı Scrum etiklerine göre kontrol eder. Scrum takımı ise 5–9 kişiden oluşan birbirleriyle sürekli iletişim halinde olan bir yazılım geliştirme ekibidir.

Ø **Toplantılar:** SCRUM'ın olmazsa olmazı toplantılardır. Her gün SCRUM toplantıları yapılır bu günlük toplantılarda her gün yazılım geliştiricilerinin önceki gün neler yaptıkları, karşılaştıkları sorunlar ve bugün neler yapacakları hakkında bir konuşma olur. Her sprint için de bir gözden geçirme toplantısı yapılır.

Ø **Bileşenler/Araçlar:** Ürün Gereksinim Dokümanı oluşturulur. Bu dokümanda proje boyunca yapılması gerekenler basitçe yazılıdır. Sprint Dokümanı oluşturulur. Sprint dokümanı ürün gereksinim dokümanına takiben oluşturulur ve amacı her sprintin ona uygun ayarlanmasıdır ayrıca bu dokümanı sadece takımdakiler değiştirebilir. Sprint Kalan Zaman Grafiği ise yapılan işin ne seviyede olduğu ve aslında planlanan zaman göre nerede olduğunu belirlemek için hazırlanır.

SCRUM günümüzde en çok kullanılan yazılım geliştirme yöntemidir. Hatta sadece yazılım geliştirmede değil birçok sistemin geliştirilmesinde de kullanılır.

Hangi Yazılım Geliştirme Modeli Daha İyi

Gelişigüzel ve barok modelleri artık kullanımını yitirmiştir bunun nedenleri yinelemeli olmamaları, dokümantasyonu işin içinde barındırmamaları gibi sıralanabilir.

Şelale modelinin kullanılması ve uygulanması basit, iş bölümü ve iş planlaması detaylıdır. Küçük ve özellikleri iyi belirlenmiş projelerde kullanılır. Eskiden yaygın olarak kullanılıyordu ancak şimdi kullanıcı ile iletişimin az olması, büyük projelerde yetersiz kalması, değişimlere kapalı olması gibi sorunlar yüzünden kullanımı çok düşüktür.

V Modeli şelale modelinin gelişmiş versiyonudur. Kullanımı ve takibi kolay olması, onaylama ve doğrulama işlemlerinin planın erken aşamasında uygulanabilir olması avantajlıdır ancak fazlar arası tekrarlamalara imkân tanımaması, risk çözümleme basamaklarının bulunmaması gibi sorunlar bulundurur.

Spiral modelde kullanıcılar ara ürünleri gördükleri için sistemin içinde sayılırlar, risk analizi çok önemli bir yerdedir, büyük projeler için idealdir, hatalar erken giderilir, planlama büyük bir parçayı oluşturur ancak çok karmaşık olması, çok uzun sürmesi, dokümantasyonun çok fazla tutulması gibi nedenlerle büyük projelerde kullanılır.

Artımsal Geliştirme modelinde her teslimde müşterinin geri dönütü alındığı için işlevsellik erken ortaya çıkar, proje daha güvenli bir hal alır, gereksinimler müşteri ile belirlenir, özellikler daha fazla test edilmiş olur ancak her artımı tanımlayabilmek için sistemin detaylı bir şekilde tanımlanması lazımdır, deneyim gerektirir ve artımlar kendi içlerinde tekrarlanamaz.

Kodla ve düzelt modeli küçük programlarda kullanılır, bakım safhası oldukça zordur, emeklilik safhası bulunur; büyük projeler için kullanılamaz çok maliyetlidir, genelde tek bir kişinin isteğine göre program en son halini alana kadar sadece kodlama yapılır ve teslim edilir, dokümantasyon gibi olaylar yoktur o yüzden kullanımı sadece kişiye özel ile sınırlıdır.

Çevik modellerde müşteri ile iletişim ve ekip içi iletişim yüksek olduğu için hatalar minimuma indirilir, alışılması basittir, değişime açık ve esnektir, takım önemli bir konumdadır ancak dokümantasyon çok detaylı olmadığı için alışılagelmişin dışındadır, ihtiyaçlar sürekli değiştiğinden çalışma saatleri fazla olabilir. Günümüzde çokça kullanılan bir modeldir.

Evrimsel Geliştirme Metodunda kullanıcılar gereksinimleri daha iyi anlarlar, hatalar azalır ancak sürecin görünürlüğü azdır ve bakımı zordur.

Hangi Projede Hangi Modeli Kullanmalıyız?

- Belirsizliklerin az olduğu, iş tanımlarının belirgin olduğu bilgi teknolojileri projeleri için V Modeli uygun bir modeldir.
- Büyük kitlelere ulaşacak projelerde evrimsel geliştirme metodu kullanılabilir.
- Büyük, maliyetli ve uzun süren projelerde spiral model veya artımsal geliştirme modeli uygundur.
- Orta ve küçük büyüklükte, uzun sürmeyen projelerde çevik modeller uygundur.
- Kişiye özel, zaman sorunu olmayan, küçük programlarda kodla ve düzelt kullanılabilir.
- Küçük ve özellikleri iyi tanımlanmış projelerde şelale modeli kullanılabilir.

SCRUM Günümüzde Neden Popüler?

- Zamandan ve paradan büyük ölçekte tasarruf edilmesi,
- Yüksek teknolojiler ve son gelişmelere kolaylıkla uyum sağlayabilmesi,
- Karmaşık görülen ve gereksinimleri tam belirlenmemiş projeler için ideal olması,
- Ekip içi iletişimin yüksek tutulması ve bununla beraber hataların erken fark edilip düzeltilmesi,
- Kullanıcıdan sürekli geri bildirim gerektirmesi ve bununla beraber sorunların azalması,
- Diğer yazılım geliştirme metodolojileri gibi yinelemeli olması,
- Değişen gereksinimlere hızlı bir şekilde tepki vermesi,
- Böl ve fethet yaklaşımı içerisinde olmasıdır.

Kaynaklar

https://www.quora.com/Why-is-the-Scrum-process-so-popular-in-the-software-industry

https://www.techwell.com/2013/02/why-scrum-so-popular

Hesap Adresleri

https://www.linkedin.com/in/faruk-bozkaya-254125194/

https://medium.com/@farukbey

https://github.com/wfarukbzk/

Mehmet Faruk BOZKAYA

190601061