**Yazılım Yaşam Döngüsü**

Bu yazımda muhtemelen birçok kez duyduğunuz, sıkça karşınıza çıkan yazılım yaşam döngü modellerinden bahsedeceğim. Fakat önce şu soruyu soralım, nedir bu yazılım geliştirme?

Yazılım geliştirme; bir ekip veya bireyin sıfırdan başlayarak bir düzen içerisinde, bir ürün tasarlama ve ortaya koyma sürecidir. Bu süreç sadece kod yazmaktan değil; verimli şekilde plan yapma, istenilen ürünü en uygun şekilde ortaya çıkarmak için tasarım yapma, çıkan ürünü test etme gibi durumları da içerir.

Peki ya yazılım yaşam döngüsü nedir? Yazılım geliştirme sürecinde gereksinim, analiz, tasarım, gerçekleştirme ve bakım adımlarını kullanarak oluşturduğumuz döngüye “yazılım yaşam döngüsü” ya da diğer adıyla “software development cycle” denir. Bu süreç art arda gelen adımların aksine, bir döngü olarak devam etmektedir. Örneğin bir ürünün test sürecinde gereksinimleri yeterli olarak karşılanmadığı durumunda tekrar başa dönülebilir.

**Yazılım Yaşam Döngüsü Adımları**

Teoride yazılım geliştirme adımlarını şu şekilde açıklayabiliriz:

**1.Gereksinim:** Bu adımda projenin temel ihtiyaçları, gereksinimleri belirlenir. Fizibilite çalışmaları yapılır.

**2. Analiz:** Bu adımda gereksinim adımında yapılanın üstüne projenin tüm gereksinim ve işlevleri detaylı olarak incelenir. Gerekli gereksinim ve işlevler ortaya çıktığı için projenin detayları tam olarak hazırlanmış olur.

**3. Tasarım:** Tasarım aşamasında analiz aşamasında ortaya çıkan detaylar aracılığıyla projenin planlaması yapılır. Yazılımın adımları belirlenir ve proje planlaması oluşturulur.

**4. Gerçekleştirme:** İlk üç aşamada detaylı şekilde planlanmış projenin gerçekleştirilme aşamasıdır. Kodlama, test edilme işlemleri de bu aşamada gerçekleştirilir. Sonucunda yazılım ortaya çıkmış olur.

**5. Bakım:** Tamamlanan yazılımdaki oluşabilecek hataları giderme, yeni işlevler ekleme gibi değişimlerin gerçekleştiği aşamadır. Müşteri ve kullanıcılardan gelen geri dönüşlere göre bazı özellikler değiştirilebilir.

Teorik olarak üstteki şekilde olsa da birçok yazılım geliştirme modelleri vardır. Bunlardan bazıları çağlayan modeli, barok modeli, v modeli, spiral modeldir. Peki neden bu kadar çok model var? Oluşturabileceğimiz yazılımlar sınırsız olsa da kaynaklarımız sınırlıdır. Projenin büyüklüğü, projede çalışacak kişi sayısı gibi etmenlerden farklı yöntemler kullanılması kaçınılmazdır. Bir yazılım geliştirmek için birçok yöntem kullanılabilir fakat bunların hepsi en doğru yöntem değildir. Yöntemler arasında “en iyi yöntem” diyebileceğimiz de yoktur çünkü her projede kendine uyumlu bir model vardır. O modele; projenin özelliklerine ve diğer etmenlere göre karar vererek en uygun yöntemi kullanabiliriz.

Peki bu modellerin özellikleri neler? O halde bu modellerden bazılarını ayrıntılı şekilde inceleyelim.

**Yazılım Yaşam Döngü Modelleri**

**Çağlayan Modeli**

“Waterfall” yani şelale yöntemi olarak da bilinen bu model; analiz, tasarım, kodlama, test, sürüm ve bakım aşamalarından oluşur. Geleneksel yazılım metotlarındaki bu safhalar şelale modelinde olduğu gibi doğrusal olarak işlediğinden geleneksel yazılım geliştirme modeli olarak da bilinir.

Bu modelin her basamağında yer alan aktiviteler eksiksiz olarak gerçekleştirilmek zorundadır. Bu bir sonraki basamağa geçmenin şartı, önceki aşamayı tamamlamaktır. Her safhada dokümantasyon yazılmalıdır, dokümantasyon yazılmadan o safha tamamlanmış sayılmaz. Kullanıcı gereksinimleri başlangıç safhasında tespit edilir ve detaylandırılır. Fakat geliştirme sürecinde kullanıcı yer alamaz, ancak ve ancak yazılım tamamlandıktan sonra geri dönüş verebilir.

Şimdi de yukarıda belirtiğimiz özelliklerden yola çıkarak avantajlarını belirtmek gerekirse şu şekilde sıralayabiliriz: Öncelikle bir faz bitmeden diğerine geçilmemesi, fazların arasındaki sınırı belli eder. Bu da projenin basit ve anlaşılabilir olmasını sağlar. Proje planlaması ve kontrolü kolaydır, bu proje yöneticisi için iyidir ve iş dağılımını da kolaylaştırır. Maliyetinin de diğer birçok modelden düşük olduğu söylenebilir.

Bu kadar avantajı olmasının yanında, tabii ki de dezavantajları olması da kaçınılmazdır. Kullanıcının katılımı sadece tasarım aşamasında mümkün olduğundan dolayı projenin kullanıcıya ulaşması uzun sürer. Müşteri ise bazen tam olarak ne istediğini kendi bile bilmeyebilir. Yani proje öncesi detaylı analizler her zaman müşterinin her gereksimini dile getirdiğinin anlamına gelmez. Kullanıcı sürecin içinde de yer almadığından dolayı geri dönüş verme süresi çok geçtir. Bu da yanlış düzeltilme ve eksik giderme kısmındaki maliyeti arttırabilir.

**V Süreç Modeli**

Çağlayan modelinin gelişmiş hali olarak düşünülebilir. O modele doğrulama yani ”verification” ve onaylama yani “validation” mekanizmalarının eklenmiş hali denebilir.

V süreç modeli; çağlayanın aksine doğrusal bir yönde ilerlemek yerine, süreç adımları kodlama evresinden sonra yukarıya doğru eğim alır ve tipik V şeklini oluşturur. V modeli denme sebebi de budur.

V süreç modeli, birçok evreden oluşur.

İlk evre doğrulama süreci olan gereksinim analizi evresidir. Sistemin gereksinimleri kullanıcının ihtiyaçları analiz edilerek toplanır. Bu evre ideal sistemin gereksinimlerini kurgulamayla ilgilenir. Yazılımın nasıl tasarlanacağını ve inşa edileceğini belirlemez. Kullanıcılar görüşülür ve kullanıcı gereksinimleri dokümanı oluşturulur.

İkinci evre olan sistem tasarımı, evresi sistem mühendislerinin analiz ettiği ve sistemin işleyişini anladıkları evredir. Eğer bir gereksinim uyuşmazlığı varsa, kullanıcıya bununla ilgili olarak bilgi verilir. Çözüm bulunmaya çalışılır ve doküman düzenlenir.

Daha sonra mimarı tasarım ve modül tasarım evreleri vardır. Mimarı tasarıma yüksek seviye tasarım derken modül tasarımını da düşük seviyeli tasarım olarak ifade edebiliriz.

V tasarım evresinde doğrulama ve geçerli kılma olmak üzere iki bölüm vardır. Yukarıdaki dört bölüm, doğrulama kısmını oluşturur. Doğrulama bölümünden her bir safha geçerli kılma bölümünden bir safhaya karşılık gelmektedir.

Birim test etme evresi, modül tasarım evresine karşılık gelir. Birim testi kodun geri kalanından izole edildiğinde düzgünce işleyebilen en küçük varlığı doğrular.

Daha sonraki entegrasyon evresi, test planları mimari tasarım evresi ile geliştirilir. Bu testler hem birimler arasında haberleşebilen hem de aynı zamanda birbirinden bağımsız şekilde test edilen ve oluşturulan birimleri doğrular. Sonuçları müşteri takımlarıyla paylaşılır.

Sistem test etme evresi ise sistem tasarım evresi boyunca geliştirilir. Birim ve entegrasyon test planlarından farklıdır. Sistem testi uygulama geliştirimi beklentilerinin karşılandığından emin olur. Tüm uygulama işlevselliği test edilir. Sistem testi fonksiyonel ve fonksiyonel olmayan gereksinimlerin karşılandığını doğrulayacaktır.

Kabul testi evresindeki planlar gereksinim analiz evresi boyunca geliştirilir. Test planları iş kullanıcıları tarafından birleştirilir. Gerçekçi veriyi kullanarak üretim ortamını benzeterek bir kullanıcı ortamında çalışır. Bu evrede kullanıcının gereksiniminin karşılandığını ve gerçek zamanda kullanım için sistemin hazır olduğunu ve sistemin dağıtıma uygun olup olmadığını kontrol edilir.

Peki özetlemek gerekirse bu modelin avantajları nelerdir? Bu model kullanıcıyla sık sık iletişime geçtiğinden kullanıcının katılımı artar. Bu da gereksinimleri netleştirir ve daha doğru bir sonuç ortaya koyulur. Aşama olduğu için takibi kolaydır ve bu da bu yöntemi proje yöneticisi için tercih edilebilir kılar.

Avantajlarının yanında dezavantajları da kaçınılmazdır. Fazlar arası tekrar bu model için bir seçenek değildir. Tamamlanılıp diğer faza geçilir. Risk çözümleme aktiviteleri de bulunmaz.

**Spiral Model**

Büyük yazılım projelerinde planlama ve gereksinim analizinin zor olması sebebiyle birçok risk karşımıza çıkar. Bu nedenle büyük projelerde tercih edilen spiral modelinde diğer modellerden farklı olarak risk analizi öne çıkmıştır. Küçük ve risk olasılığı düşük yöntemler için tercih edilmez, pahalı bir yöntemdir.

Yinelemeli yaklaşımı vardır, projeyi fazlara böler. Fakat bu fazlar sonsuza gidebilir ve bu dezavantaj oluşturur. Projeleri fazlara bölmekteki asıl amaç hataları, riskleri erken farkına varmaktır. Böylece müşteri ürünü erkenden görür ve hatalar erkenden giderilmiş olur.

Spiral yazılım geliştirme modeli temel olarak dört ana faz içerir. Bunlar; planlama, risk yönetimi, üretim ve kullanıcı değerlendirmeleridir. Planlamada üretilecek ara ürün için planlama, amaç belirleme, bir önceki adımda üretilen ara ürün ile bütünleştirme işlemli gerçekleştirilir. Daha sonra risk seçeneklerinin araştırılması ve risklerin belirlenmesi risk analizi yöntemidir. Sonraki aşamada ürün üretilir ve kullanıcı değerlendirmesine sıra gelir. Burada kullanıcı değerlendirmeler gerçekleştirir.

**Çevik Yazılım**

SCRUM'ın ne olduğunu bilmek için öncelikle çevik yazılımın ne olduğunu bilmeliyiz. Çevik yazılımın kökeni 1957 yıllarındaki IBM’deki yazılım geliştirme çalışmalarına dayanmaktadır. Daha sonra 1990’lı yıllar Şelale modelinin yetersiz bir sistem olduğu düşünülerek onun yerine daha hızlı ve çevik yazılım geliştirme metodolojileri sunmaya çalışılan yıllar olarak geçmiştir. 2001 yılının başlarında yazılım dünyasının önde gelen bazı isimleri Utah’da bir araya gelmiş ve “Çevik Yazılım Geliştirme Manifestosu” ve “Çevik Yazılımın Prensipleri”ni yayınlamışlardır.

Tanım odaklı konuşmak gerekirse atik yazılım geliştirme ya da diğer adıyla çevik yazılım geliştirme, basit prensiplere dayanan yazılım geliştirme metotların genel adıdır. Bu metotların asıl amacı en hızlı şekilde, değişen isteklere en hızlı yanıtlarla, müşteriye en ideal zamanda ürün sunmaktır.

Çevik yazılım metotları yüksek verimli, ucuz, düşük hata oranlı ve hızlı çözümler sağlayarak yazılımın oluşmasını sağlar. Extreme Programming (XP), SCRUM, Agile Unified Process gibi birçok çevik yazılım metotları bulunmaktadır.

**Scrum**

Scrum, çevik yazılım geliştirme yöntemi olarak karmaşık bir ortamda ürünleri geliştirmek, sunmak ve sürdürmek için bir türdür. Bu türün temel özelliği gözlemci, geliştirmeci ve tekrara dayalı olmasıdır. Scrum sadece yazılım projelerinde değil, birçok projede kullanılabilir. Çoğu modern yazılım projesinin karmaşık olduğu ve en baştan tümünü planlamanın zor olacağı şeklinde varsayım üzerine bir metodolojidir. Bu karmaşıklığı üç ilke ile azaltmaya çalışır; şeffaflık, denetleme ve uyarlama.

Şeffaflık ile projedeki ilerlemeler ve sorunlar günlük olarak tutulur ve herkes tarafından izlenebilir olması sağlanır. Gözlem ile ürünün parçaları ya da fonksiyonları düzenli aralıklarla teslim edilir ve değerlendirilir. Uyumlanma sayesinde ise ürün için gereksinimler en baştan bir defalığına belirlenmez, her teslimat tekrar tekrar değerlendirilir ve duruma göre değişimler yapılır.

Yani şeffaflığı, sürekli geri bildirimi, hızlı sorun çözmesi, etkili ve güvenli olması, verimliliği gibi sebeplerden Scrum günümüzde sıkça tercih edilen popüler yöntemlerdendir.

Scrum'da ürün sahibi (product owner), takım (team), scrum ustası (scrum master) olmak üzere üç proje rolü vardır. Özellikle "geleneksel" "proje yöneticisi" rolü bulunmaz. Ürün Sahibi stratejik ürün geliştirmeden sorumlu kişidir. Yazılım ekibinin görevi ürün sahibinin taleplerine ve sıralamasına uygun ürünün gereksinimlerini sağlayarak ürünü teslim etmektir. Scrum Ustasının görevi takımı korumak ve yardımcı olmak ve ürün sahibine yardımcı olmaktır.

Scrum’daki bir diğer önemli kavram toplantılardır.

Sprint planlama toplantısında gereksinimler ve takımlar belirlenir. Araçlar, altyapı, maliyet gibi konulara değinilir.

Her iş günü başlamadan evvel 15 dakikalık bilgi paylaşımı için günlük Scrum toplantısına günlük scrum yani daily scrum denir. Bu görüşmede herhangi bir problem değerlendirilmez, yalnızca "dün ne yaptım, bugün ne yapacağım, beni ne engelliyor" konularına değinilir. Belli durumlarda görevler parçalanıp takıma dağıtılabilir.

Değerlendirme Sprint'in sonunda takım tarafından yapılır ve başlangıçta belirlenen hedefler konusunda yeterli olup olmadığı ürün sahibi tarafından değerlendirilir. Bu toplantı sprint değerlendirmesi yani sprint reviewdır. Eğer teslim edilen projede eksiklik var ise o kullanıcı hikâyesi tekrardan gönderilir ve öncelik sırası verilir.

***Berkay Uzun***

*Kaynakça*

"tr.wikipedia.org", “fikirjeneratoru.com” ve Bakırçay Üniversitesi Yazılım Mühendisliği

Temelleri ders notları kaynaklardan yararlanılmıştır.