**YAZILIM GELİŞTİRME YAŞAM DÖNGÜ MODELLERİ**

Yazılım yaşam döngü modelleri hakkında konuşmadan önce yazılım geliştirme yaşam döngüsü (SDLC) nedir ve temel aşamaları nelerdir açıklayalım.

**SDLC nedir?**

Yazılım Geliştirme Yaşam Döngüsü (SDLC), yazılım endüstrisi tarafından yazılımları tasarlamak, geliştirmek ve test etmek için kullanılan bir süreçtir. SDLC belirli bir yazılımın nasıl geliştirileceğini, korunacağını, değiştirileceğini veya iyileştirileceğini açıklayan ayrıntılı bir plandan oluşur. SDLC bir döngü yaklaşımıdır. SDLC aşamaları aşağıdaki şekilde sıralanabilir.

1. Planlama ve İhtiyaç Analizi

2. Tanımlama

3. Tasarım ve Analiz

4. Geliştirme

5. Testler

6. Uygulama

7. Bakım

Yukarıdaki aşama listesi temel olarak her kaynakta aynıdır fakat bazı kaynaklarda bazı adımların ismi değiştirilmiş ya da farklı isimlerle kullanılmış olabilir. Bu durumun bir önemi yoktur temelde anlatılmak istenenler aynıdır.

**Planlama ve İhtiyaç Analizi**

İlk olarak ihtiyaç analizinin yapılmasıyla başlar. İhtiyaç analizi müşterinin istekleri doğrultusunda ortaya çıkacak olan yazılım hakkında ilk bilgilerin alındığı adımdır. Bu adımda müşteri iyi bir şekilde dinlenmeli ve istenilenler tam ve net olarak belirlenmelidir

Planlama kısmında ise SDLC'nin bu aşamasında ekip, analiz edilen ihtiyaçları karşılamak için gereken maliyeti ve kaynakları belirler(fizibilite çalışması). Ayrıca, ilgili riskleri detaylandırır ve bu riskleri yumuşatmak için alt planlar sağlar. Diğer bir deyişle, ekip projenin en düşük risk göz önünde bulundurularak projeyi nasıl başarılı bir şekilde uygulayabileceğini belirlemelidir.

**Tanımlama**

Planlama ve ihtiyaç analizi yaptıktan sonraki aşama yazılım ürününün ihtiyaçlarını açık ve herkesin anlayacağı şekilde tanımlamak ve belgelemek, bu belgeyi müşteriye ve analistlere onaylatmaktır. Bu belge SRS(Software Requirement Specification)dir.

**Tasarım ve Analiz**

Bu aşamada planlama ve tanımlaya göre bir tasarım çizilir. Kararlar verilir, seçimler yapılır, örneğin yazılımın ekranları, ekranlarda neler bulunacağı, hangi ekranlara nasıl geçileceği vb. şeyler bu aşamada tasarlanır. SRV belgesine göre sistem ve tasarım dokümantasyonu (DDS) yapılır. Bir sonraki adım olan uygulama geliştirmeye bütün kararlar verilmiş olarak geçilmesi beklenir. Geliştirme aşamasında herhangi bir soru veya karar bırakılmaz.

**4.Geliştirme**

Bu aşama daha önce belirlenmiş yönergelere bağlı kalınarak kodlama kısmının yapıldığı kısımdır. Daha önce tasarım aşamasında verilen kararlar doğrultusunda ve daha önceki aşamalarda belirlenen kararlar doğrultusunda yazılım ürünün kodlama kısmına başlanır. Bu aşamada dikkat edilmesi gereken kodlamanın sade basit temiz okunabilir kod yazılmasıdır. Çünkü ileriki zamanlarda kodların iyileştirilmesi müşteri ihtiyacına göre değiştirilmesi gerekebilir.

**5.Testler**

Aslında test aşması tüm aşmaların bir alt basamağı olarak düşünülebilir. Fakat buradaki test aşaması yazılım ürününün beklentileri karşılayıp karşılamadığı, bunların takip edilmesi, rapor edilmesi, düzeltilmesi ve yeniden test edilerek istenen ürünün hazır hale getirilmesinin sağlandığı aşamadır.

**6.Uygulama**

Bu aşamada artık elimizde bir yazılım ürünü vardır. Yazılım ürünü kullanılır problemler çıkar ve problemlere çözümler üretilir tartışılır. Daha sonra bakım aşmasına geçilir.

**7.Bakım**

Bu aşamadaki amaç yazılım ürününün ilk aşamalarda belirlenen kriterlere göre çalışmasını sağlamaktır. Farklı senaryolarda ürünün verdiği hatalar rapor edilir. Bazı yeni özellikler eklenir. Yazılım yeni sürümlerine yükseltilir.

Popüler SDLC modelleri

**1.Şelale Modeli:** SDLC’de kullanılan ilk modeldir. Doğrusal sıralı model olarak da bilinir. Bu modelde, bir aşamanın sonucu bir sonraki aşamanın girdisidir. Bir sonraki aşmaya geçebilmek için önceki aşamanın tamamlanması gerekir. Her aşama tamamlandıktan sonra o aşamaya ait dokümantasyon oluşturulur. Şelale modeli statiktir. Statik projelerde kullanılabilir. Fakat yazılım sektörü dinamik bir sektör olduğu ve projelerde gereksinimler her an değişebildiği için iyi bir model olduğu söylenemez hatta en kötü modellerden birisidir de denebilir. Yine de ilk model olması bir örnek teşkil etmesi açısından önemlidir.

**2.V Modeli:** V modeli, aşamaların V şeklinde önce aşağı doğru projenin temel aşamaları sırasıyla yapılır, sonra yukarı doğru paralel olacak şekilde temel aşmaların test kısımları yapılır. V modelinde her aşama bir önceki aşamaya bağlıdır. V modeli oldukça disiplinli bir modeldir. Sade ve basit bir model olmasından dolayı yönetilmesi de kolaydır. V modeli değişikliklere esnek olmaması nedeniyle günümüz dünyasının dinamik yapısı içinde sıklıkla olan ihtiyaç değişiklikleri karşısında kullanılması çok pahalı ve zor olan bir modeldir.

**3.Big Bang Modeli:** Bu modelde belirli bir plan veya süreç yoktur. Big Bang Modeli, planlama olmadan ya da çok az planlama ile yazılım projesi içindeki her şeye odaklanmayı içerir. Gereksinimler ortaya çıktıkça anlaşılır ve çözümler üretilip uygulanır. Bu model bir veya iki kişiden oluşan küçük gruplar için ideal bir modeldir. Esnek basit ve yönetmesi kolaydır. Büyük projeler karmaşık işler için uygun değildir.

**4.Fıskiye Modeli:** Bu model şelale modelinden ilham alınarak oluşturulmuş bir modeldir. Şelale modelinde farkı döngülerin olmasıdır. Bu döngüler her aşamada vardır.

**5.Kodla ve Düzelt Modeli:** Yazılım yaşam döngü aşamalarından olan analiz aşamasından sonra direk yazılım kodlanır ve ortaya çıkarılır. Daha sonra proje üzerinde düzeltmeler yapılır. Kolay bir modeldir. Dokümantasyon yoktur.

**6.Prototip Modeli:** Prototip modeli, prototipin gerçek yazılımdan önce geliştirildiği bir modeldir. Prototip modelleri, gerçek yazılımla karşılaştırıldığında sınırlı işlevsel yeteneklere ve verimsiz performansa sahiptir. Prototipler oluşturmak için kukla işlevler kullanılır. Bu, müşterilerin ihtiyaçlarını anlamak için değerli bir mekanizmadır. Müşteriden değerli geri bildirim almak için gerçek yazılımdan önce yazılım prototipleri oluşturulur. Geri bildirimler uygulanır ve prototip, herhangi bir değişiklik için müşteri tarafından yeniden incelenir. Bu süreç, model müşteri tarafından kabul edilinceye kadar devam eder.

**7.Spiral Model:** Spiral model, belli aşamalara geri dönebilecek şekilde tasarlanmıştır.Spiral modelde süreçlere tekrar tekrar geri dönüş yapılır. Bundan dolayı bu modele Spiral Model ismi verilmiştir. Her çevrimde ürünün yeni bir sürümü ortaya çıkmaktadır. Her çevrimde her aşama değerlendirilir.

**8.Evrimsel Model:** müşterinin ne istediğini bilmediği durumlarda kullanılması uygun bir model olarak nitelendirilebilir. Diğer modellere göre yavaş bir modeldir. Gereksinimleri anlamayı kolaylaştırır. Sürekli olarak yazılım ürününde bir değişim olur.

**9.Artımsal Geliştirme Modeli:**Bu modelde gereksinimler projenin başlangıcında gruplara ayrılır.Her grup bir SDLC modeli takip eder.Her grup gereksinimler karşılanıncaya kadar SDLC işlemini tekrarlar.Böylece her döngüde bakım da yapılmış olur.

**Çevik Yazılım**

Yazılım ve bilgisayar mühendislerinin yazılım geliştirirken en çok kullanması gereken modeldir. Yazılımların ömrünü uzatan, daha etkin bir performans göstermesini sağlayan modeldir. Çevik yazılım metodolojisi, büyük bir projeyi küçük parçalara ayırarak yazılımın geliştirilmesini ön görür. Yazılımın geliştirilmesi aşamasında müşteri ile iç içe uyumlu bir şekilde çalışılır ve müşteriden alınan geri dönüşlere göre yazılım yeni şeklini alır.

**Çevik Yazılımın Temel İlkeleri**

* Müşteriyi her aşamada memnun etmek
* Müşterinin değişen ihtiyaçlarını karşılamak
* Değişen gereksinimler yazılım sürecinin son aşamasında bile kabul edilmelidir.
* Çalışan yazılım bir kaç haftada ya da bir kaç ayda bir düzenli olarak müşteriye sunulmalıdır
* Yüz yüze iletişime ön planda tutmak
* Kendi kendine organize olan takımlar kurmak. Takım, düzenli aralıklarla nasıl daha etkili ve verimli olabileceğinin üzerinde düşünür ve davranışlarını buna göre ayarlar ve düzenler.
* Çalışan yazılım gelişimin en önemli ölçüsüdür.
* Çalışan yazılım bir kaç haftada ya da bir kaç ayda bir düzenli olarak müşteriye sunulmalıdır.
* Basitlik çok önemlidir.
* Yazılım geliştiricileri ile müşteri arasındaki iletişim çok sık olmalıdır.

Çevik yazılım en temelde bu şekilde işler ama kendisi de alt kategorilere sahiptir.Yazılım projelerinde yaygın kullanılan çevik metodolojiler

Extreme Programing(XP)

Rational Unified Process

Feature-Driven Development(FDD)

Test Driven Development (TDD)

LEAN Development

Dynamic System Development Methodology (DSDM),

Microsoft Solution Framework (MSF),

SCRUM

örnek verilebilir. Biz bu makalede scrumdan bahsedip scrumın günümüzde neden bu kadar popüler olduğunu konuşacağız.

**SCRUM**

Çevik yazılım yöntemi kendi içerisinde özü aynı, fakat süreçlerinde farklılaşan alt kollara ayrılmaktadırBu metodojiler içerisinde popüler olanlarından SCRUM’ın adı Rugby sporundaki bir hücum taktiğinden gelmektedir. SCRUM, gereksinimlerin tam olarak belli olmadığı, her an değişebilecek, karmaşık yazılım projelerin yönetimi için uygulanması en ideal yöntemdir.Esnek bri geliştirme yöntemidir.Düzenli geri bildiirim verilir fakat müşteriyle doğrudan bir iletişim olmadığından yazılım ekibi bir baskıda hissetmez ve işine yoğunlaşır.Büyük şirketlerin kullandığı(Microsoft,Google,Facebook) popüler bir metodolojidir.

**SCRUM GÜNÜMÜZDE NEDEN POPÜLER?**

Scrumın bu kadar popüler olmasının birçok sebebi vardır.. Birincisi, Scrum elbette olgun bir geliştirme ve proje yönetimi çerçevesidir. Sık yinelemeler ve sürekli geri bildirimlerle, teslim edilen son ürünün müşteri ihtiyaçlarına uygun olduğundan emin olmanıza yardımcı olur. Gereksinimleri anında değiştirme yeteneği, projenin raydan çıkmamasını sağlar. Scrum, iyi süreçlere ek olarak psikolojik faydalarıyla da birlikte gelir. Takımların performanslarını geliştirmelerine yardımcı olmak için psikolojik mekanizmalar ve akıllı motivasyon üzerine inşa edilir.Ancak sadece bu da değil: Yüz yüze toplantıların (diğer geliştiriciler, müşteriler ve hatta son kullanıcılarla) yoğun kullanımı sayesinde, geliştiriciler işlerinin ne kadar önemli olduğuna dair gerçek bir fikir edinir. Ekip üyeleri, ilerlemelerini rapor etmeli, yapılacakları açıklamalı ve günlük standuplar sırasında karşılaştıkları sorunları açıklamalıdır. Geriye dönük toplantılar, yapısal sorunların çözülmesi için süreçlerin sürekli iyileştirilmesine yardımcı olur. Sorumluluk duygusu, Scrum'ın yönetim görevlerini takıma devretme uygulamasıyla daha da derinleşir. Sonuç olarak, Scrum basitliği ve yüksek performansı nedeniyle en popüler Çevik proje yönetimi metodolojisidir.

KAYNAKLAR:

<https://stackify.com/what-is-sdlc/>

<https://www.tutorialspoint.com/sdlc/sdlc_overview.htm>

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

<http://ybsansiklopedi.com/wp-content/uploads/2015/08/Yaz%C4%B1l%C4%B1m-Geli%C5%9Ftirme-Modelleri-Yaz%C4%B1l%C4%B1m-Ya%C5%9Fam-D%C3%B6ng%C3%BCs%C3%BCSDLCYBS.pdf>

ÜMİT YILMAZ

200601039