Yazılım Geliştirme Modelleri ve Sistem Yaşam Döngüsü

Yazılım Geliştirme Yaşam Döngüsü, yazılım oluşturmanın temelidir. Çünkü yazılım oluşturmak için tüm süreçlerin adım adım tamamlanmasını sağlar. Mühendisler ve geliştiriciler, maliyet etkinliği ve projenin zamanında tamamlanmasını sağlamak için yazılım oluşturmanın her aşamasında bir planlama kılavuzu olarak SDLC modellerini kullanır. Çünkü yazılım oluştururken hiçbir şey kendiliğinden olmaz, rastgele bir yol izlenemez.

**1. Giriş ve Tanımı**

Yazılım Geliştirme Yaşam Döngüsü (SDLC) modeli, planlamadan bakıma kadar bir yazılım geliştirme projesindeki tüm faaliyetleri açıklayan kavramsal bir çerçevedir. Bu süreç, her biri çeşitli görev ve faaliyetlere sahip farklı modellerle bağlantılıdır. Başka bir ifadeyle SDLC veya Yazılım Geliştirme Yaşam Döngüsü, yüksek kaliteli, etkili ve verimli yazılım geliştirmek ve mümkün olan en kısa sürede uygulamak için çerçeve metodolojisidir. Bu terim aynı zamanda yazılım geliştirme süreci modeli olarak da bilinir. SDLC çerçevesi ise önceden tanımlanmış süreçleri, aşamaları içerir ve SDLC' nin her aşaması, yaşam döngüsünün bir sonraki aşaması için gerekli ürünleri sunar.

**2.Yazılım Geliştirme Yaşam Döngüsü (SDLC) Aşamaları**

Yazılım Geliştirme Yaşam Döngüsü (SDLC) kavramı bir döngü yaklaşımdır Bir sarmal

hareket halinde olan bir yaklaşımdır. SDLC aşamaları aşağıdaki şekilde sıralanabilir.

1. Planlama

2. Tanımlama

3. Tasarım

4. Geliştirme

5. Entegrasyon ve testler

6. Uygulama

7.Bakım

Hayat bir döngü içinde dönmektedir. Hepimiz hayatın içinde belli döngülerde yaşıyoruz. İnsanlar doğuyorlar, yaşıyorlar, ölüyorlar sonra toprak oluyor ve sonra o toprak başka hayatlara kaynak oluyor. Yağmur yağar, sular birikir, sonra buharlaşır, tekrar bulut olur ve tekrar yağmur yağar, Bu döngü her zaman vardır. Dolayısıyla yazılımın da kendi doğal yaşam döngüsü olduğunu bilmeliyiz. Bu döngünün doğru analiz edilmesi durumunda, doğru kontrollerin yapılması durumunda bizi başarıya götüreceğini bilmeliyiz. SDLC kavramı ilk başlarda “Yazılım Yaşam Döngüsü” olarak kabul ediliyordu. Fakat Sistem Geliştirme Yaşam Döngüsü (System Development Life Cycle) ismi daha çok kullanılmaya başlatıldı.

1.1.Planlama

Bu plan aşaması proje yönetimlerinde kullanılan en önemli aşamadır. Dğier adımların da başlangıç aşamasıdır. İşin projelendirildiği, fikrin ortaya çıkarıldığı ve fikrin tartışıldığı aşamadır.

1.2.Tanımlama

Planlamadan sonra planlanan projenin tanımlarının yapıldığı aşamaya geçilir. Genelde istenilen fikrin ne olduğu ve temel tanımlar üzerinde konuşulacak kavramların tanımlandığı aşamadır. Çünkü aynı fikirden bahsedilmiyorsa farklı sonuçlara varılabilir. Bu tanımlar üzrine bir de “tasarım” vardır. Bu analiz aşaması olarak da düşünülebilir. Problemin tanımlandığı veya yaşam döngüsünün tanımlandığı sistemin veyahut yazılımın tanımlandığı aşamadır. Tanımlama aşamasında projede nelerin istenildiği ile ilgili analiz çalışmaları da yapılabilir.

1.3.Tasarım

Yazılımımızın veya sistemimizin tasarımları yapılır. Projeleri çizilir. İnşaat projeleri gibi düşünülebilir.Planlama ve tanımlara göre bir tasarım çizilir. Kararlar verilir, seçimler yapılır, örneğin yazılımın ekranşarı, ekranlarda neler bulunacağı, hangi ekranlara nasıl geçileceği, fonksiyonel olarak hangi adımların oluşturulacağı, hatta yazılımın bileşenleri, modülleri bu aşamada tasarlanır. Bir sonraki adım olan uygulama/geliştirmeye bütün kararlar verilmiş olarak geçilmesi beklenir. Gelişme aşamasında herhangi bir soru veya karar bırakılmaz.

1.4.Geliştirme

Bu aşama, yazılım projeleri için kodlama olarak düşünülebilir veya sistemin yaşatılmaya başlandığı ilk örneklerinin çıkmaya başlandığı aşama olarak düşünülebilir. Daha önceden tasarım aşamasında karar verilen ortama uygun olarak, yine tasarım aşamasında verilen kararlar doğrultusunda projenin gerçekleştirilmesine başlanır.

1.5.Entegrasyon ve Testler

Sistemin artık gerçek hayata geçtiği aşamadır. Sistem veys yazılım hayat içinde bir yer edinir. Bir şirkette kullanılacak yazılımdan bahsediliyorsa şirketle ilgili birimlerin yazılımla ilgili eğitimlerin alınması, donanımlarının temin edilmesi, bağlantılı olduğu birimlerin buna entegre edilmesi, başka yazılımlarla bir entegrasyonu varsa bunun sağlanması ve veri tabsnı bağlantıları gibi pek çok adım bu aşamada ele alınır.

1.6.Uygulama

Proje artık yaşayan bir yazılım haline gelir. Yazılım kullanılır ve problemler ortaya çıkar. Problemlere karşılık yeni fikirler ortaya çıkar, yeni ihtiyaçlar ortaya çıkar ve bunlar değerlendirilmeye alınır. Bu değerlendirmeler sonrasında bakım aşamasına geçilir.

1.7.Bakım

Projenin bakımı yapılırken farklı aşamalardan geçirilir. Yeni tanımlar ortaya çıkar, yeni tasarımlar yapılır ve bu şekilde devam eden döngü hiçbir zaman bitmez.

**2.SDLC Yaklaşımına Farklı Bakış Açıları**

**Planlama:** Bu aşama, planlamanın bütün gereklerini tamamlar. Zaman kaynak ayrımı, personel, kurumun mevcut bilişim seviyesi ve yeni sistemin kullanılabilirliği, uzun vadede sistemin bakımı ve sürdürülebilirliği gibi çok sayıda parametre bu aşamada incelenir. Bu aşamanın tamamlanmasının ardından bir uygunluk raporu (feasibility report) hazırlanarak kurumdaki irade sahiplerine (yönetim) sunulur ve bilişim sisteminin yapacağı olumlu katkıların maliyet analizine göre katma değeri olduğu kesinleştirilir.

**Sistem Tahlili (Analiz):** Bu aşamada sistem analizi yapılır ve geliştirilmekte olan bilgi sisteminin etkilediği diğer birimler ve mevcut işleyiş tahlil edilir. Örneğin geliştirilmekte olan sistem ödeme emirleri ile ilgili olsun, bu durumda girilen emirlerin satış, defteri kebir, stok kayıtları gibi çok sayıdaki etkilediği diğer kayıt tahlil edilerek bu sistemlerle olan etkileşimi çalışılır.

**Sistem Tasarımı (Design):** Bu aşamada sistemin bütün bileşenleri ve genel yapısı tasarlanır. Çok çeşitli tasarım yöntemleri ve aşamaları olmakla beraber genelde mantıksal bir tasarım aşamasından geçilerek sistemin hedefleri ve etkileşimde olduğu alt bileşenleri arasındaki uyum gözetilir. Ardından bu mantıksal tasarımın fiziksel tasarıma çevrilmesi aşaması gelir. Fiziksel tasarıma çevrilim aşamasında, programlama dilinden kullanılacak olan teknolojilere ve sistemin bütün parçalarına kadar olan tasarımı tamamlanır. (Veri tabanı tasarımı, kullanıcı ekranları, veri akış yolları gibi).

**Uyarlama ve İşletme Aşaması (Implementation and Operation):** Bu aşama, şimdiye kadar yapılan bütün kâğıt üzerindeki işlemlerin gerçeğe dönüştürüldüğü ve yazılım halini aldığı aşamadır. Kodlama, kod kontrolleri, testleri, kurulumu, yönetimi gibi işlemlerin tamamı bu aşamada yapılır.

Yukarıdaki 4 aşama kapsamında SDLC adımlarının çıktıları aşağıdaki şekilde özetlenebilir:

***Tablo 1. SDLC Aşamaları ve Çıktıları***

|  |  |
| --- | --- |
| **Aşama** | **Çıktı** |
| Planlama ve  Seçim safhası | Sistem öncelikleri veri, ağ, donanım ve bilgi sisteminin mimarisi, seçilmiş olan proje için detaylı çalışma planı, sistemin geçerlilik alanı için özellikler, sistemin iş akışı açısından önemi |
| Sistem Tahlili | Mevcut sistemin tanımı, mevcut sistemin düzeltilmesi, iyileştirilmesi, değiştirilmesi için öneriler, muhtelif sistemler için açıklamalar ve alternatiflerin değerlendirilmesi, yeni teknoloji gereksinimlerinin belirlenmesi ve geçiş için öneriler |
| Sistem Tasarımı,  Uyarlama ve  İşletme | Sistemin bütün alt bileşenleri için detaylı birer tasarım  Kod, dokümantasyon, eğitim ve destek kaynakları, yeni sürüm için kod, eğitim ve destek aşamalarındaki değişiklikler |

Yukarıdaki 4 adımlı yaklaşımın yanında Kendall and Kendall tarafından sunulan 7 adımlı yaklaşımdan söz etmek de mümkündür. Bu yaklaşım aşağıdaki adımlardan oluşur (Kendall, 1992):

1. Problem, fırsatlar ve hedeflerin belirlenmesi
2. İnsan seviyesi enformasyon ihtiyaçlarının belirlenmesi
3. Sistem ihtiyaçlarının (isterlerinin) belirlenmesi
4. Şimdiye kadar olan adımlar ışında elde edilen isteklere cevap veren bir sistemin tasarlanması
5. Yazılımın geliştirilmesi ve dokümantasyonu
6. Sistemin testleri ve bakım adımlarının belirlenmesi
7. Sistemin gerçeğe alınması ve değerlendirilmesi

Yine Kendall ve Kendall kitaplarında bu geliştirme sürecindeki kaynak harcama miktarlarını aşağıdaki şekilde modellenmektedir;

**Şekil 1 SDLC Proje Maliyet Grafiği**

Şekilden de anlaşılacağı üzere, sistemin geliştirilme aşamasında sürekli artan (üstel olarak artan) proje maliyeti varken, projenin canlıya geçmesinin ardından bu maliyet düşmeye başlar. Zamanla sistemdeki istek ve ihtiyaçların ölçeğinin artması ile proje maliyetleri de artmaya başlar. Hatta sistemin geliştirme aşamasındaki maliyetlerin bile üzerine çıkabilir. Bu yüzden projenin tasarımının ve analizinin ileride çıkabilecek ihtiyaçları en verimli şekilde karşılayacak yeterlilikte olması çok önemlidir.

**3. SDLC Yaklaşımının Avantajları**

Yazılım geliştirme yaşam döngüsü, geliştiricilere ve test uzmanlarına yazılım uygulamaları geliştirmek için kolay bir yaklaşım sunar. SDLC’nin geliştiriciler ve test uzmanları arasında oldukça popüler olmasını sağlayan bazı temel avantajlar aşağıdakileri içerir:

* Yazılım geliştirme sürecine dahil olan adımlara ve faaliyetlere yapı sağlar.
* Tüm SRS ile uyumlu olmasını sağlayarak Yazılım Geliştirme Yaşam Döngüsünün her aşamasında yazılım geliştirme faaliyetlerinin kontrolünü sağlar.
* Yazılım uygulamalarını yönetmek için açık ve özlü bir çerçeve sunarak planlanan bütçe, program ve kapsam dahilinde tamamlanmalarını sağlar.
* Geliştirilen yazılım uygulamasının son kullanıcının hedefleri ve amaçları ile uyumlu olmasını sağlamaya yardımcı olur, böylece onlara maksimum değer verir.
* Yüksek özellikli uygulamaların geliştirilmesindeki tüm süreçlerin şeffaflığını ve görünürlüğünü sağlar.
* Yazılım Geliştirme Yaşam Döngüsünü takip ederek, yüksek kaliteli yazılım uygulamalarını hızlı bir şekilde elde etmeye daha fazla yönelirsiniz. SDLC size temel kavramları hakkında net bir fikir verecektir.

**4. SDLC Yaklaşımının Dezavantajları**

SDLC’nin bazı dezavantajları da vardır, bunlardan bazıları şunlardır:

**1. Esnek Olmayabilir:** SDLC, önceden belirlenmiş bir çerçevedir ve bu nedenle esnek olmayabilir. Değişen gereksinimlere ve koşullara uyum sağlamakta zorlanabilirsiniz.

**2. Bürokratik Olabilir:** SDLC, her aşamada belgelendirme ve onaylar gibi bürokratik işlemler gerektirebilir. Bu sayede, projenin gecikmesine neden olabilir.

**3. Tüm Projeler için Uygun Olmayabilir:** SDLC, küçük ve basit projeler için uygun olmayabilir. Bu tür projeler için daha esnek bir yaklaşım daha uygun olabilir.

**4. Karmaşık Olabilir:** SDLC, özellikle büyük ve karmaşık projeler için karmaşık olabilir. Bu tür projeler için SDLC’nin uyarlanması gerekebilir.

SDLC, yazılım projelerinin başarılı bir şekilde tamamlanması için önemli bir araçtır. Doğru SDLC modelinin seçilmesi ve projenin bu modele uygun şekilde yönetilmesi, yüksek kalitede ve hatasız bir yazılımın zamanında ve bütçe dahilinde geliştirilmesini sağlayabilir.