Yazılım Geliştirme Yaşam Döngüsü

Yazılım Geliştirme Yaşam Döngüsü (SDLC) kavramı bir döngü yaklaşımdır. Bir sarmal hareket halinde olan bir yaklaşımdır. SDLC aşamaları aşağıdaki şekilde sıralanabilir:

1. Gereksinim: Müşterinin ihtiyaçlarının alındığı ve yazılımın planlandığı aşamadır.
2. Analiz: Genelde istenilen fikrin ne olduğu ve temel tanımlar üzerinde konuşulacak kavramların tanımlandığı aşamadır. Müşteri ihtiyaçlarının net olmadığı zamanlarda, yazılım mühendisi ile müşterinin yapacağı toplantılar çok daha faydalı olmaktadır.
3. Tasarım: Gereksinim ve analize göre bir tasarım yapılır. Bu aşamada iki çeşit tasarım vardır. Mimari tasarımda yazılımı oluşturacak modüllerin sistem içindeki etkileşimlere ele alınmaktadır. Sonuç olarak mimari tasarım dokümanları oluşturulur. Detaylı tasarım aşamasında ise mimari tasarım dokümanları revize edilmektedir.
4. Gerçekleştirme: Tasarım aşamasında oluşturulan modüllerin kodlanıp birleştirildiği aşamadır. Bu aşamada yazılan kodlar test edilir. Kodların olabildiğince erken test edilmesi ileride karşımıza çıkabilecek olası hataları daha önceden fark ederek zaman ve maliyeti düşürecektir. Yazılımın bağlantılı olduğu birimlerin buna entegre edilmesi, başka yazılımlarla bir entegrasyonu varsa bunun sağlanması ve veri tabanı bağlantıları gibi pek çok adım bu aşamada ele alınır.
5. Bakım: Yazılımda sonradan oluşan hataların giderildiği ve müşteri ihtiyaçları doğrultusunda yeni gereksinimlerin eklendiği aşamadır.

Yazılım Yaşam Döngü Modelleri

Bir yazılım geliştirilirken birçok yazılım yaşam döngüsü modeline göre plan yapılır. Bu yazıda yazılım projelerinde kullanılan yazılım yaşam döngü modellerini açıklayıp, karşılaştırılması yapılacaktır. Hangi projede hangi modeli kullanmamız gerektiğini açıklayacağız. Günümüzde sektörde Scrum modelinin neden popüler olduğuna değineceğiz.

**1. Gelişigüzel Model**

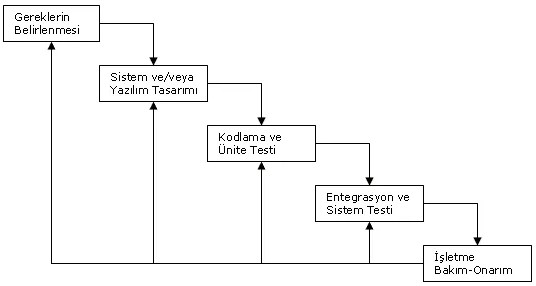
Herhangi bir model veya yöntem gözetmeksizin yapılır. Bu model genellikle kişiye bağlı yazılım geliştirme modelidir. Yazılım oluşturulduktan sonra takip edilebilirliği ve bakımı oldukça zordur. Bu model yazılım dünyası henüz gelişmemişken, 1960’lı yıllarda çoğunlukla tek bir kişinin küçük hacimli programlar yapmak için kullandığı modeldir.

**2. Barok Modeli**

Bu model yaşam döngüsü başlıca adımlarının doğrusal bir biçimde geliştirildiği modeldir. Bu model belgelemeyi ayrı bir süreç olarak ele almaktadır. Ama günümüzde belgeleme yapılan işin doğal bir çıktısı olarak görülmektedir. Adımlar arası geri dönüşlerin ne türlü yapılacağı da tanımlanmamıştır. Bunların yanında en olumsuz özelliği de gerçekleştirme aşamasına daha çok ağırlık veren bir modeldir. Saydığımız bu sebeplerden dolayı 1970’lerde kullanılan bu model günümüzde kullanımı tavsiye edilmemektedir.

**3. Şelale (Waterfall) Modeli**

1983 yılında literatüre ilk model olarak Benington tarafından kazandırılan şelale modeli aslında fikrin yayılmasından çok daha önce 1956 yılında fikir Amerikan Donanması tarafından düzenlenen bir matematik sempozyumunda yine Benington tarafından sunulmuştur. Şelale modeli tarihsel olarak inşaat ve üretim sektörlerinde kullanılmıştır. Üretim ve inşaat alanında tasarım yapılıp üretim aşamasına geçildikten sonra müşterinin yeni talepleri ya hiç karşılanamamaktadır ya da çok kısıtlı oranda karşılanabildiğinden bu model kullanılır. Proje yönetim modelleri bu sebeple genelde şelale modeli ile başlar. Şelale modeli diğer yazılım geliştirme modellerine göre daha statiktir. Özellikle yazılım gibi esnek bir yapıdaki üretimin çok defa bu şekilde ihtiyaçlarda ve dolayısıyla analiz, tasarım gibi ilerleyen adımlarda değişikliklere ihtiyaç duyulmasından ve bu geri dönüşlerin yazılım maliyetini arttırmasından dolayı bu model yeni modellerin çıkmasına sebep olmuştur.



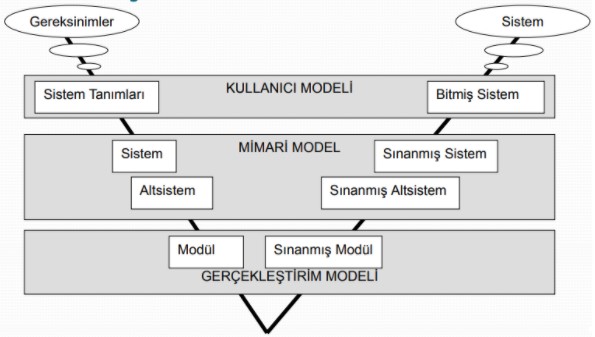
Analiz aşamasında mümkün olduğunca tüm detaylar tasarıma yansıtılabilmesi amacıyla müşteri gereksinimleri en ince ayrıntısına kadar belirlenmelidir. Tasarım aşamasında ise sistem belirlenen tüm gereksinimleri yerine getirecek şekilde tasarlanmalıdır. Bu modelde her ne kadar müşteri gereksinimleri önemli bir yer kaplasa da proje kodlama ve test aşamasında yeni oluşan müşteri gereksinimleri proje maliyetini arttırmaktadır. Bu modelde yazılım ekibinde bir an önce kodları yazıp programı çalıştırma eğilimli olduklarından dolayı ekip mutsuzlaşmakta ve kod yazanlar haricindeki ekip üyelerine değer verilmemektedir.

**4. V Süreç Modeli**

Model ismini adımların V şeklinde oluşmasından alır. Bu süreçte aşağıya doğru akan bir proje yönetimi görülmektedir. Sonrasında yukarı doğru çıkmaya başlar. Şelale modelinden farklı olarak her üretilen modül birleştirilmeden testleri yapılmaktadır. V süreç modelinin temel çıktıları ise 3’e ayrılır.

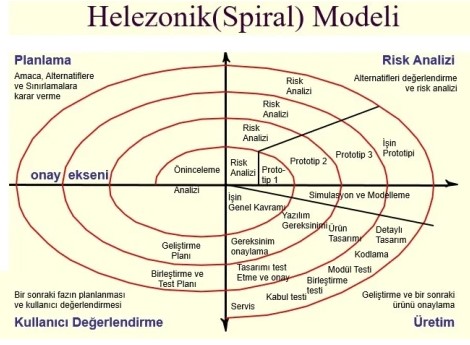
Kullanıcı Modeli: Müşteri gereksinimlerine göre ve kullanıcı istekleriyle beraber sistemin nasıl geliştirilmesi gerektiğine dair planların yapıldığı ön hazırlık aşamasıdır.

Mimari Model: Sistemin tasarımının yapıldığı aşamadır. V süreç modelinde Şelale modelinden farklı olarak bu aşamada Üst seviye tasarımda daha genel bakışa göre tasarım yapılır. Üst seviye tasarımların daha sonra daha detaylı alt seviye tasarımlarının yapıldığı görülmektedir.



**5. Helezonik (Spiral) Model**

Helezonik modeli diğer modellerden ayıran temel fark süreci oluşturan aşamalardan tekrar tekrar geçilmesi ve her geçildiğinde projenin ilerleme kat etmesi amaçlanmaktadır. Bu model literatüre 1980’li yıllarda kazandırılmıştır.



Resimde de görüldüğü üzere dönerek aşamaları ilerleme esastır. Bu modelin diğer modellere göre avantajı dönme sırasında protatipler çıkartılarak müşteriye gösterilmesidir. Müşteri diğer modellerde olduğu gibi soyut bir şey üzerinden fikirlerini belirtmek yerine protatip üstünden isteklerini ve kullanıcı gereksinimlerini daha iyi aktarmaktadır. Üretilecek ara ürün için amaç belirleyerek bir önceki ara ürünle birleştirildiği aşamaya planlanma denmektedir. Risk analizi bu modelde ön plana çıkmaktadır. Ara ürünlerin üretildiği aşama üretim aşamasıdır. Sonrasında üretilen ara ürün kullanıcı tarafından değerlendirilmelidir.

**6. Artımsal Geliştirme Modeli**

Artımsal Geliştirme Modelinde istenilen sistemi müşteriye en son teslim etmek yerine sistem müşterinin öncelik sıralamasına göre parçalara bölünür. Her parça bir ürün olarak müşteriye teslim edilir. Müşteri de sistem kullanıcıları tarafından yapılan geri dönüşlerle yazılım projesi besler ve yeni sistem bu geri dönütlere göre şekillenir. Sistemin oluşturulmasının uzun zaman alacağı ve eksik işlevlerle de çalışabileceği yazılımlarda tercih edilen bir yöntemdir.

**7. Kodla ve Düzelt Yaşam Döngü Modeli**

Küçük yazılım projelerinde kullanılan bir yöntemdir. Diğer modellerden farklı olarak direkt yazılım aşaması gerçekleştirilir. Sistem en son istenen hale gelinceye kadar geliştirilir. Şartname ve tasarım dokümanı olmadığından dolayı yazılımın bakımını yapmak çok zordur. Yazılım geliştirmenin en kolay ve pahalı yoludur. Diğer modellerden farklı olarak emeklilik (retirement) safhası bulunmaktadır.

**EVRİMSEL GELİŞTİRE MODELİ**

Coğrafik olarak geniş alana yayılmış, çok birimli organizasyonlar için kullanılmaktadır. Her aşamada üretilen ürünler üretildikleri alanda tam işlevselliği içermektedirler. Sürekli değerlendirildiği için erken aşamalardaki geliştirme riskleri azalmış olur. Yazılımın daha sonrasında bakımını yapmak zordur.

**Çevik Yazılım Geliştirme Modelleri**

Yazımızda buraya kadar anlattığımız modellerde ortaya çıkan genel sorunları şu şekilde sıralayabiliriz. Yazılım versiyonlarının zamanında ortaya çıkarılmaması, müşterinin değişen ihtiyaçlarına hızlı cevap verememesi, yazılımda oluşan hataların geç fark edilmesi şeklinde sıralanabilir. Bu sebeplerden dolayı Kent Beck ve 16 arkadaşı tarafından çevik manifestosu oluşturulmuştur.

**Extreme Programming(XP)**

Kent Beck tarafından yazılım geliştirme prensibi olarak 1999 yılında ortaya çıkartılmıştır. Bu model 4 temel değerden oluşmaktadır.

İletişim: Yazılım ekiplerindeki temel sorunlarında birisi de kendi aralarındaki anlaşmazlıklardır. Diğer modellerde gördüğümüzden farklı olarak müşteri ile yazılım ekibi arasında sağlam bir iletişim olmalıdır. Bir konuda müşterinin geri dönütüne ihtiyaç duyulduğunda hızlı bir şekilde bu sağlanarak vakit kaybedilmeden projeye devam edilmesi sağlanmalıdır.

Basitlik: Basitlik oluşturulması kolay gibi gözüken bir konu olasına karşın aslında basitliğin yazılımda kurulması zordur. XP bu basitliği en kolay şekilde sağlamak için günün ihtiyaçlarını karşılayarak, esnek ve basit bir sistem oluşturmayı hedeflemektedir.

Geri Bildirim: Geri bildirim sayesinde projenin oluşturulma aşamalarında oluşan hatalar hızlı bir şekilde giderilerek müşteri memnuniyetinin artmasına sebep olmaktadır.

Cesaret: Yazılım geliştirme sürecinde başarısızlıktan korkmamalıyız onun yerine başarısızlıklarımızı en kısa zamanda telafi etmeliyiz.

**XP Pratikleri**

Planlama Oyunu: Planlama oyunu yazılım ekibiyle müşterinin yer aldığı bir toplantıda yazılımın o kısmının ne kadar sürede yapılacağının planlandığı aşamadır.

Ekipte Müşteri: Yazılım ekibinin yazılımı geliştirirken ihtiyaç duyduğu bilgilere kısa sürede erişebilmesi için ve müşterinin ihtiyaçlarında anlaşılmayan kısımları açıklamak için yazılım ekibiyle beraber ortamda müşteri temsilcisi de bulunmalıdır.

Önce Test: Kod yazılmadan önce test programı yazılarak olası hataların daha önceden ortaya çıkması sağlanmalıdır.

Basit Tasarım: Günümüzde geliştirilen programlardaki fonksiyonların yaklaşık %64’ü hiç kullanılmamaktadır. Bu yüzden müşterinin ihtiyaçlarını karşılayacak en basit şekilde program yazılmalıdır.

Çiftli Programlama: Yazılımcıların yetenek setleri birbirinden farklıdır. Bu yöntemle beraber yazılımcılar sorunları daha hızlı çözer. Bu yöntem yeni başlayan yazılımcılar için önemlidir.

Sürekli Entegrasyon: Yazılım oluşturulurken yapılan tüm değişikliklerin sisteme entegre edilmesi faydalıdır. Böylelikle sistem üzerindeki olası hataların erken tespit edilebilmesini sağlar.

Kısa Aralıklı Sürümler: Yazılım küçük parçalara bölünerek haftalık halde ilerlemektedir.

Yeniden Yapılandırma(Refactoring): Kod ve proje tasarımı sürekli gözden geçirilerek müşteri memnuniyetinin artması hedeflenmektedir.

Ortak Kod Sahiplenme: Yazılan kod parçaları kişiye aittir şeklinde bir yaklaşım yanlıştır. Yazılımı oluşturulan kodlar herkese aittir.

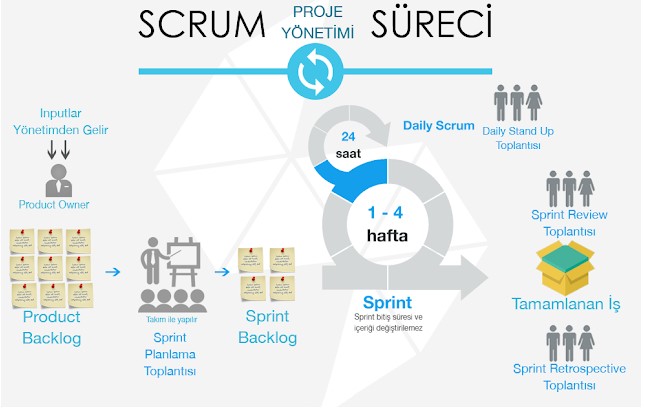
Benzetim(Metafor): Yazılımı gerçekleştiren parçalar bir başka sistemle benzeştirilerek yazılım geliştirmeye çalışılır.

Kodlama Standardı: Her şirketin kendine oluşturduğu bir kodlama standardı vardır ve yazılım ekibine yeni dahil olanlar bu standartlara göre kodlarını yazmalıdır.

Haftada 40 Saat: Ekip üyeleri haftada 40 saati her gün 8’er saat çalışarak doldurmalıdır. Aksi durumda fazladan mesaiye kalındığında takımın verimliliği düşeceğinden ve çalışanlar mutsuz olacağından dolayı süreci olumsuz etkilemektedir.

**SCRUM**

Scrum adını Rugby sporundaki bir hücum taktiğinden almaktadır. 1990’lı yıllarda Jeff Sutjerland ve Ken Schawaber tarafından geliştirilen bir yöntemdir. Scrum sadece yazılım projeleri için değil diğer tüm projeleri yönetmek için kullanılan bir yaklaşımıdır. Bu metodoloji, yazılımı küçük küçük sprint olarak adlandırılan parçalara bölerek geliştirmeyi amaçlar. Bu yöntemde her gün ekip 15 dakikalık toplantılarla iş takibi yapmaktadırlar.



Scrum metodunda 3 temel kavram vardır:

1. Roller(Roles)

Scrum takımı devamlı birbirleriyle etkileşim içinde projeyi sürdürürler ve çalışma sonuçlarını belli aralıklar ile ürün sahibine gösterirler. Scrum yöneticisi takımı ve organizasyonu scruma adapte etmekle sorumludur. Ürün sahibi müşteri tarafından görevlendirilerek geri dönüşler vermektedir.

2. Toplantılar(Meetings)

Proje gereksinimlerinin geliştirilerek oluşturulan takımlara dağıtımlarının yapıldığı aşamadır.

3. Bileşenler

Proje gereksinimlerinin takibinin yapıldığı basit listeye ürün gereksinim dokümanı denmektedir. Sprint Dokümanı mevcut sprint içinde yapılması gereken iş ve görevleri kapsar. Sprint kalan zaman grafiği ise planlanan sprint süresinde ne kadar iş yapılması gerektiği ile yapılan işleri karşılaştırmak için kullanılır.

Günümüzde en çok kullanılan model çevik modeldir. Kısa sürede, esnek bir şekilde geliştirilerek, yüksek başarımla sonuçlanan bir model olması günümüzde popülerliğini arttırmıştır. Google, Microsoft, IBM gibi birçok şirket Scrum modelini kullanmaktadır.

KAYNAKÇA

1. Doç. Dr. Deniz Kılınç, Bakırçay Üniversitesi Yazılım Mühendisliğine Giriş Dersi 2. ve 3. Hafta Sunumları

2. <https://fikirjeneratoru.com/yazilim-proje-yonetimi-yontemleri/>

3. <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>

4. <https://medium.com/@denizkilinc/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-temel-a%C5%9Famalar%C4%B1-software-development-life-cycle-core-processes-197a4b503696>

5. <https://erenozdemir.net/waterfall-yazilim-gelistirme/>

6. <https://systemanalize.wordpress.com/gelistirme-metotolari/>

7. <https://anilsarier.com/2019/10/28/yazilim-muhendisligi/>

**Medium Linki:** [Yazılım Geliştirme Yaşam Döngüsü. Yazılım Geliştirme Yaşam Döngüsü (SDLC)… | by Ali Bayram | Mar, 2021 | Medium](https://alibayram-02.medium.com/yaz%C4%B1l%C4%B1m-geli%C5%9Ftirme-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-652fcbcddce8)

Gihub Linki: <https://github.com/aliibyrm/Yazilim-Yasam-Dong--Modelleri>