# YAZILIM GELİŞTİRME VE SÜREÇ MODELLERİ

## Yazılım Geliştirme Yaşam Döngüsü

Yazılımla ilgili herhangi bir projeyi geliştirme sadece kodlamadan oluşmamaktadır. Öncelikle bir yazılım geliştirme yaşam döngüsü bulunmaktadır. Yazılımın hem üretim hem de kullanım süresi boyunca geçirdiği tüm aşamalar yazılım geliştirme yaşam döngüsü olarak tanımlanır.

Başarılı bit yazılım geliştirme süreci için her aşamanın ayrıntılı bir şekilde planlanması, süreçlerin sağlıklı bir şekilde yönetilmesi sağlanmalıdır. Yazılım projelerin temel adımları aşağıdaki gibidir:

* Planlama
* Analiz
* Tasarım
* Gerçekleştirme
* Teslim ve Bakım

### 1.Planlama

Yazılım yaşam döngüsünün ilk aşamasıdır. Projeye başlanmadan önce kullanıcı istekleri dikkate alınarak proje ihtiyaçları belirlenir. Proje süreci boyunca izlenecek yol ve yöntemler ile, donanım maliyeti, personel iş bölümü, proje yöneticisinin ön görü ve seçimleri, proje adımlarının sürece dağıtılması gibi unsurlar değerlendirilir.

### 2.Analiz

Sistem ihtiyaç ve işlevlerinin ayrıntılı bir şekilde çıkarıldığı ve belirli kurallar çerçevesinde kağıda döküldüğü aşamasıdır. Geliştirilecek olan yazılım ürününden tam olarak ne istendiği net bir şekilde belirlenir.

### 3.Tasarım

Belirlenen istekler çerçevesinde yazılım sisteminin temel yapısının oluşturulduğu aşamadır. Tasarım safhası boyunca 2 tür tasarımdan bahsedilebilir:

a)Mantıksal Tasarım

Önerilen sistemin yapısı anlatılır.

b)Fiziksel Tasarım

Yazılımı içeren bileşenler ve bunların ayrıntıları anlatılır.

### 4.Gerçekleştirme

Kodlama, test etme ve kurulumun yapıldığı aşamadır:

a)Kodlama

Oluşturulan yazılım ürünün programlama aşamasıdır. Basit ,okunabilir ve bakımı kolay olan kodlama tekniğiyle programlama yapılmalıdır.

b)Test Etme

Önceden belirlenen ihtiyaçları karşılayıp karşılamadığı ,doğru çıktı üretip üretmediği testleri yapılarak sistemdeki açıklıklar kapatılmalıdır.

### 5.Teslim ve Bakım

Tüm test aşamaları tamamlandıktan sonra ürün müşteriye teslim edilir. Yazılım ürünü kullanılmaya başlandıktan sonra bakım aşaması da başlar. Ürün kullanıldıktan sonra yazılımın desteklenip desteklenmemesine göre hata giderilir ya da yeni eklentiler yapılır.

## YAZILIM YAŞAM DÖNGÜSÜNÜN TEMEL ADIMLARI

Yazılım yaşam döngüsün temel adımları çekirdek süreçler olarak adlandırılır. Bu süreçlerin gerçekleştirilmesi amacıyla Yazılım Belirtim Yöntemleri ve Yazılım Süreç Modelleri kullanılır.

### Yazılım Belirtim Yöntemleri

Bir çekirdek sürece ilişkin fonksiyonları yerine getirmek amacıyla kullanılan yöntemlerdir.

#### 1.Süreç Akışı İçin Kullanılan Belirtim Yöntemleri

Süreçler arası ilişkilerin ve iletişimin gösterildiği yöntemlerdir.

(Veri Akış Şeması,Yapısal Şemalar,Nesne-Sınıf Şemaları)

#### 2.Süreç Tanımlama Yöntemleri

Süreçlerin iç işleyişini göstermek için kullanılan yöntemlerdir

(Düz Metin,Algoritma,Karar Tabloları,Karar Ağaçları,Anlatım Dili)

#### 3.Veri Tanımlama Yöntemleri

Süreçler tarafından kullanılan verilerin tanımlanması için kullanılan yöntemlerdir.

(Nesne İlişkisi Modeli,Veri Taban Tabloları,Veri Sözlüğü)

# YAZILIM SÜREÇ MODELLERİ

Yazılım yaşam döngüsünde belirtilen süreçlerin geliştirme aşamasında hangi düzen ya da sırada,nasıl uygulanacağını tanımlayan modellerdir.Süreçlere ilişkin ayrıntılara ya da süreçler arası ilişkilerle ilgilenmezler.

Yazılım süreç modelleri 9 tanedir :

* Kodla ve Düzelt Model Yaşam Döngü Modelleri
* Çağlayan Yaşam Döngü Modeli
* V Süreç Modeli
* Evrimsel Geliştirme Süreç Modeli
* Prototipleme Yaşam Döngü Model
* Spiral Model
* Formal Sistem Geliştirme Modeli
* Yeniden Kullanıma Yönelik Geliştirme Modeli
* Artırımsal Geliştirme Süreç Modeli

## 1.Kodla Ve Düzelt Yaşam Döngü Modeli

Genellikle resmi olmayan bir ürün fikriyle başlar ve program ürün hazır olana kadar ya da gerekli zaman bitene kadar kodlama yapılarak devam edilir.

**Avantajları**

* Herhangi bir planlamaya ihtiyaç duyulmaz
* Çok küçük projelerde ya da kısa ömürlü prototiplerde uygulanabilir.
* Program aşamaları çabuk geçilir.
* Uzman görüşüne ihtiyaç düşüktür, herkes bu modeli kullanabilir.

**Dezavantajları**

* Kontrollü değildir.
* Kaynak planlaması yoktur.
* Bitiş süresi belli değildir.
* Hataların bulunması ve doğrulanması zordur.
* Kodları düzeltmek maliyetli olabilir.
* Kodlar kullanıcının ihtiyacını karşılamayabilir.
* Kodlar sonradan değiştirmek için planlanmadığından esnek değildir ve değiştirilmesi zordur.

## 2.Çağlayan Modeli

Şelale yönteminde yazılım geliştirme süreci analiz, tasarım, kodlama, test, sürüm ve bakım gibi safhalardan oluşur.

**Avantajları**

* Kullanılması ve anlaşılması basittir.
* Yönetimi kolaydır.
* Projenin safhaları ayrı ayrı olduğundan iş bölümü ve iş planı projenin en başında net bir şekilde bellidir. Bu durum projenin yönetimini de oldukça kolay hale getirir.
* Çok küçük ve gereksinimleri çok iyi anlaşılmış projelerde iyi çalışır.

**Dezavantajları**

* Karmaşık ve nesne yönelimli projeler için uygun değildir.
* Devam eden ve uzun projeler için zayıftır.
* Projede oluşabilecek her türlü değişime elverişsiz, katı bir modeldir. Yapılan her değişiklik maliyeti büyük oranda artırır.
* Müşteri memnuniyetini sağlamak çok zordur ; çünkü gelişim ve değişime açık bir model değildir
* Model safhalardan oluştuğu için ürün son safhada tamamlanır, gereksinimlerinin iyi tanımlanmadığı ,müşterinin ne istediğinin anlaşılmadığı bir projede bu durum proje bittikten sonra projenin iptal edilmesine ve başka gerginliklere sebep olmaktadır.

## 3.V Modeli

Şelale modelinin gelişmiş hali olarak düşünülebilecek bir yazılım geliştirme süreci sunar .Doğrusal bir yönde ilerlemek yerine , süreç adımları kodlama evresinden sonra yukarıya doğru eğim alarak tipik V şeklini oluşturur.

**Avantajları**

* Doğrulama ve onaylama planları erken aşamalarda vurgulanır.
* Doğrulama ve onaylama sadece son üründe değil tüm teslim edilebilir ürünlerde uygulanır.
* Proje yönetimi tarafından takibi kolaydır.
* Kullanımı kolaydır.

**Dezavantajları**

* Aynı zamanda gerçekleştirilecek olaylara kolay imkan tanımaz.
* Fazlar arasında tekrarlamaları kullanılmaz.
* Risk çözümleme ile ilgili aktiviteleri içermez.
* Yazılım da diğer sitemler gibi zamanla evrimleşir.
* Geliştirme devam ettikçe iş ve ürün gereksinimleri de değişkenlik gösterebilir.
* Son ürüne ulaşma düz bir çizgi ile ifade edilmez.

## 4.Evrimsel Geliştirme Süreç Modeli

İlk tam ölçekli modeldir. Coğrafik olarak geniş alana yayılmış, çok birimli organizasyonlarda kullanılmak üzere geliştirilecek projeler için önerilir. Modelin başarısı ilk evrimin başarısına bağımlıdır.

**Avantajları**

* Kullanıcıların kendi gereksinimlerini daha iyi anlamlarını sağlar.
* Sürekli değerlendirme erken aşamalardaki geliştirme risklerini azaltır.
* Hatalar azalır.

**Dezavantajları**

* Sürecin görünürlüğü azdır. (Düzenli teslim edilebilir ürün yoktur.)
* Sistemler sıklıkla iyi yapılandırılmaz. (Sürekli değişiklik yazılımın yapısına zarar verir.
* Bakımı zordur.
* Yazılımın gereksinimini yenilemek gerekebilir.

## 5.Prototipleme

Bir yazılım sisteminin ilk sürümüdür. Bir problem ve olası çözümleri hakkında daha fazla bilgi edinmek,

Tasarım seçeneklerini denemek ve temel kavramları göstermek için prototip modeller kullanılmaktadır. Burada amaç; kullanıcının gereksinimlerini ortaya çıkarmaktır. Ana fonksiyonları gösterecek olan bir ürün geliştirilir, burada detaylar yoktur, sadece anahtar fonksiyonlar vardır.

**Avantajları**

* Kullanıcı sistem gereksinimlerini görebilir.
* Karmaşa ve yanlış anlaşılmaları engeller.
* Yeni ve beklenmeyen gereksinimler netleştirilebilir.
* Risk kontrolü sağlanır.

**Dezavantajları**

* Prototip hedefleri net değilse kod hackleme ya da jenga başlar.
* Düzeltme aşaması atlanır ise düşük performansa yol açar.
* Müşteri prototipten de son ürün gibi görünüm ve etki bekler.

## 6.Spiral Model

Sarmal modeli aynı safhalara geri dönülmesinin bir zorunluluk olduğunu vurgular, şelale modelinde yok sayılan riskleri göz önünde bulundurur .Proje çevrimlere ayrılır ve her bir çevrimin riskleri ayrı ayrı ele alınır .Çağdaş modellere son derece yakındır.

**Avantajları**

* -Kullanıcılar sistemi erken görebilirler.
* -Geliştirmeyi küçük parçalara böler. En riskli kısımlar önce gerçekleştirilir.
* -Pek çok yazılım modelini içinde bulundurur.
* -Riske duyarlı yaklaşımı potansiyel zorlukları engeller.
* -Hataları erken gidermeye odaklanır.
* -Yazılım-donanım sistemi geliştirmek için bir çerçeve sağlar.

**Dezavantajları**

* Küçük ve düşük riskli projeler için pahalı bir yöntemdir.
* Karmaşıktır.
* Ara adımların fazlalığı nedeniyle çok fazla dokümantasyon gerektirir.

## 7.Formal Sistem Geliştirme Modeli

Yazılım tasarım ve gerçekleştirmesiyle ilgili matematiksel bir tekniktir .Bu modelin temelinde karmaşık sistemleri geliştirme ve program geliştirmeye destek yatar .Kullanıcı sistemi kullanmaya başladığında karşısına çıkan belirtim hatalarını minimize eder.

**Avantajları**

* Yazılımdaki belirsizlikleri, eksiklikleri ve uyumsuzlukları saptar.
* Hatasız yazılım geliştirme imkanları sunar.
* Karmaşık değildir.

**Dezavantajları**

* Çok zaman alan ve pahalı bir yöntemdir.
* Kullanımı esnasında teknik olmayan personelle iletişim mekanizması zor işler.
* Sadece birkaç geliştirici bu modelin uygulamasıyla ilgili temel bilgilere sahip olması için yaygın eğitim gerektirir.

## 8.Yeniden Kullanıma Yönelik Geliştirme

Organizasyon tarafından daha önce hazırlanmış veya dışarıdan temin edilmiş yazılımların kullanılması ile geliştirme yapılması son yıllarda popülaritesi artan bir yaklaşımdır .Organizasyonların olgunlukları artıkça , bu tür uygulamalar yapabilmek için alt yapı kurulmuş olmaktadır.

**Avantajları**

* Kaynak kontrolü mümkündür.
* Maliyet denetimi yapmak mümkündür.
* Kısa sürede yazılım geliştirilebilir.
* Basit ve anlaşılırdır.
* Önceden oluşturulmuş sınıflar tekrardan kullanılabilir.

**Dezavantajları**

* Gereksinimleri anlamak güçtür.
* Pahalıdır.
* Uzmanlık gerektirir.
* Başarı garantisi yoktur.

## 9.Artırımsal Geliştirme Süreç Modeli

Artırımsal geliştirme süreç modeli, çağlayan yaşam döngü modeli ile evrimsel geliştirme süreç modelini temel almaktadır. Bir yazılım ürünü geliştirilirken sürekli olarak değişen müşteri gereksinimlerini karşılayan ve ortaya çıkan kaçınılmaz hataların zamanında düzeltilmesini sağlayan bir süreç modelidir.

**Avantajları**

* Sistem için gerekli olan gereksinimler müşterilerle belirlenir.
* Gereksinimlerin önemine göre teslim edilecek artırımlar belirlenir.
* Öncelikle en önemli gereksinimleri karşılayan çekirdek bir sistem geliştirilir.
* Tüm projenin başarısız olma riskini azaltır.
* En önemli sistem özellikleri daha fazla sınanma (test edilme ) imkanı bulmuş olur.

**Dezavantajları**

* Yapılan işlerden edinilecek sonuçlar belli değildir.
* Geliştirilen yazılımlar genellikle sınırlı sayıda kullanılır ve kullanımı bittikten sonra işe yaramaz hale gelir ve atılır.
* Model -zaman fiyat kestirimi belli olmadığı için sabit fiyat sözleşmelerinde uygun değildir.

# ÇEVİK YAZILIM GELİŞTİRME METODOLOJİLERİ

## Çevik Yazılım Geliştirme

Yazılım geliştirme süreci sıkıntılı ve uzun süren bir dönemdir. Yazılım geliştirme projeleri yönetimsel eksikliklerden dolayı ancak kısmi bir başarı memnuniyet ile tamamlanabilmektedir. Yazılım sektöründe , yazılım sürümlerinin zamanında ortaya çıkarılmaması , değişiklik isteklerine çabuk cevap verilmemesi ,yazılım hatalarının geç fark edilmesi ve zaman içinde gelen isteklere göre sistemin kendi yapısını geliştirememesi gibi çeşitli sorunlar ortaya çıkmıştır.

Bu sorunların aşılmasına yönelik yapılan çalışmalar sonucu ,1990 ‘lı yılların sonlarına doğru çevik “çevik “olarak adlandırılan metotlar ortaya çıkmıştır .Çevik yazılım geliştirme metotları ; piyasaya çabuk ürün çıkarabilme , değişen isteklere hızlı yanıt verme ve en kısa sürede bir yazılım ürünün müşteri hizmetine sunmayı amaçlamaktadır .Çevik yazılım geliştirme metotları, verimliliği yüksek ,esnek, hata oranı düşük, hızlı ve ucuz çözümler sunmaktadır .Çevik yazılım geliştirme metotları ,kendi içerisinde özü aynı fakat pratikleri farklılaşan çeşitli metodolojilere ayrılmaktadır.

## Çevik Yazılım Geliştirme Manifestosu

Kent Beck ve değişik çevik metotların temsilcileri olan 16 arkadaş bir araya gelerek ortak yanlarını ortaya koyan bir çevik manifestosu yayınladılar. Bu manifestoda ;

* Süreçler ve Araçlar yerine **Bireyler ve Etkileşimler,**
* Kapsamlı Belgeler yerine **Çalışan Yazılım**,
* Sözleşme Görüşmeler yerine **Müşteri İlişkileri,**
* Plan İzleme yerine **Değişikliğe Açıklığın,**

daha önemli ve öncelikli olduğu belirtilmektedir.

## Çevik Yazılım Geliştirme Prensipleri

Çevik yazılım geliştirme manifestosunun altında yatan temel prensipler şunlardır;

* Müşteriye hızlı ve sürekli olarak kullanılabilir yazılım teslimatı yapmaktır.
* Kodlamanın ilerleyen safhalarında bile gereksinim değişiklikleri kabul edilir, esnektir.
* Mümkün olduğunca kısa zaman aralıklarında çalışan, kaliteli yazılım teslimatı yapılır.
* Analistler, uzmanlar, yazılımcılar, tesisatçılar vs. tüm ekip elamanları birebir işetişim halinde birlikte çalışırlar.
* İyi projeler motivasyonu yüksek kişiler etrafında kurulabilir.
* Ekip içerisinde kaliteli bilgi akışı için yüz yüze iletişim önemlidir.
* İlerlemenin birincil ölçüsü çalışan bir yazılımdır.
* Çevik süreçler, sürdürülebilir geliştirmeyi destekler.
* Sağlam teknik alt yapı ve tasarım, çevikliği artırır.
* Basitlik önemlidir.
* En iyi mimariler ve tasarımlar kendini organize edebilen ekipler tarafından yaratılır.
* Düzenli aralıklarla ekip kendi yöntemlerini gözden geçirir ve verimliliğini artırmak için gerekli iyileştirmeleri yapar.

## Yazılım Projelerinde Yaygın Uygulanan Çevik Metodolojiler

* Extreme Programming (XP)
* Rational Unified Process
* Feature-Driven Development (FDD)
* Test-Driven Development (TDD)
* LEAN Development
* Dynamic System Development Methodology (DSDM)
* Microsoft Solution Framework (MSF)
* SCRUM

### SCRUM

Çevik yazılım yöntemi kendi içerisinde özü aynı, fakat süreçlerinde farklılaşan alt kollara ayrılmaktadır Bu popüler kollardan bir tanesi de SCRUM’ dır. SCRUM, kelime olarak Rugby oyununda oluşturulan küçük ekiplere verilen isimdir. Bu yöntem 1990’ lı yılların ortalarında Jeff Sutjerland ve Ken Schawaber tarafından oluşturulmuş ve günümüze sürekli bir gelişme halinde gelmiştir.

SCRUM, gereksinimleri açıkça belli olmayan, değişime açık, karmaşık yazılım projelerin yönetimi için uygulanması en ideal yöntemdir. İhtiyaca yönelik, esnek, bir geliştirme sürecidir. Düzenli geri bildirim ve kısa dönemli planlamalarla hedefe ulaşmayı sağlıyor. En önemli özelliği ise müşteri ve geliştirici doğrudan temas halinde olmadıkları için yazılım geliştirici baskı altında olmaz, yapması gereken işe yoğunlaşır. Başarısı kanıtlanmış bir yazılım geliştirme yöntemi olan Microsoft, Google, Facebook gibi şirketler tarafından kullanılmaktadır.

SCRUM’ da 3 temel kavram vardır:

1. Roller
2. Toplantılar
3. Bileşenler/ Araçlar

**KAYNAKÇA**

<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>

<https://www.codex.com.tr/yazilim-gelistirme-modelleri>

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

<http://www.acm-software.com/scrum/>

Yazılım Mühendisliğinde Modern Yaklaşımlar,Dr. Fatih Yücalar,Dr. Emin Borandağ,2018

***Berfin KAÇAR 180601007***