**Yazılım Geliştirme Yaşam Döngü ve Modelleri**

200601015 Özmen KAHVECİ

**1. Giriş ve Tanımı**

Yaşadığımız dünyada pek çok varlık döngüler sonucu oluşmaktadır. Örneğin su döngüsü ve azot döngüsü gibi biyolojik döngüler ile insanların doğup, yaşayıp ölmesi gibi toplumsal döngüler vardır. Yazılımda da bir ürünün ortaya çıkması için bir döngünün oluşması gerekmektedir. Bu döngü, Yazılım Geliştirme Yaşam Döngüsü (SDLC) başlığı altında kavramsallaşmıştır. SDLC, bir ürünün ortaya çıkması için kendi içinde belirli aşamalara bölünmesidir. Bu döngü tek yönlü ve doğrusal olmayıp aşamalar arasında geçiş yapılabilmektedir. SDLC aşamaları şu şekilde sıralanabilir: Gereksinim, analiz, tasarım, gerçekleştirme, bakım.

**1.1. Gereksinim**

Bu aşamada projenin temel ihtiyaçları belirlenir. Maliyet hesaplanması, projenin pratik olarak uygulanabilirliği gibi araştırmaların yapıldığı aşamadır.

**1.2. Analiz**

Projede yer alması gereken hususların neler olduğu, projede karşılaşılabilecek sorunların belirlendiği, gereksinimlerin incelendiği aşamadır.

**1.3. Tasarım**

Projenin biraz daha somutlaştığı aşamadır. Bu aşamada yazılımın arayüzü ve fonksiyonelliği tasarlanır. Bir sonraki aşama olan gerçekleştirme aşaması için gerekli altyapı çalışmaları tamamlanmış olur.

**1.4. Gerçekleştirme**

Tasarımı yapılan projenin kodlandığı ve gerekli testlerin yapıldığı aşamadır.

**1.5. Bakım**

Bu aşama, üzerinde çalışılan projenin ürüne dönüşmüş halinden sonra ürünle ilgili gerekli görülen güncellemelerinin ve bakımlarının yapıldığı aşamadır.

Yazılım Geliştirme Yaşam Döngüsünde belirtilen bu aşamaların gerçekleştirilmesi sırasında hangi düzen ya da sırada, nasıl uygulanacağını tanımlayan belli başlı yazılım süreç modelleri mevcuttur.

**2. Yazılım Süreç Modelleri**

Yazılım süreç modelleri, bir projeyi gerçekleştirirken bize yol gösteren şemalardır. Bazı yazılım süreç modelleri aşağıdaki gibidir.

* Kodla ve Düzelt
* Gelişigüzel Model
* Barok Modeli
* Çağlayan (Şelale) Modeli
* V Modeli
* Helezonik Model
* Arıtımsal Geliştirme Modeli
* Çevik Modeller

**2.1. Kodla ve Düzelt**

Bu modellemenin ilk safhasında direkt yazılım ürününün ilk sürümü gerçekleştirilir. Bu ürün üzerinden yazılım yeterli seviyeye gelene kadar geliştirme devam eder.

**2.1.1. Kullanım Alanları**

Çok küçük projelerde ya da kısa ömürlü prototiplerde kullanılır.

**2.1.2. Avantajları**

Karmaşık ve detaylı bir plan yapmaya gereksinim göstermez. Program çok çabuk bir şekilde geliştirilir. Hızlı bir şekilde ürün elde edilir.

**2.1.3. Dezavantajları**

Kaynak planlamasına ihtiyaç yoktur. Kodlar değiştirilmek için planlanmadığından bakım yapması ve değiştirilmesi oldukça zor ve maliyetlidir.

**2.2. Gelişigüzel Model**

1960’lı yıllarda kullanılan bir modellemedir. Yazılım geliştirme ortamında herhangi bir model ya da yöntem kullanılmaz. Bu sebeplerden ötürü bu yöntemi model olarak adlandırmak doğru değildir.

**2.2.1. Kullanım Alanları**

Tek kişinin üretip geliştirdiği yazılımlarda kullanılır. Basit programlar için kullanılabilecek bir yöntemdir.

**2.2.2. Avantajları**

Program tek kişi üzerinden geliştirildiği için böylelikle olası bir planlama daha hızlı yapılır.

**2.2.3. Dezavantajları**

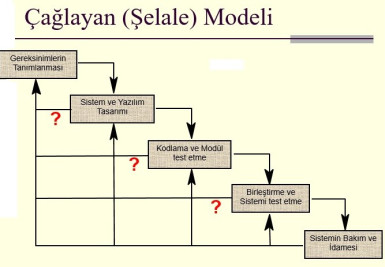
Projenin takip edilebilirliği, geliştirilmesi, bakımı oldukça zor olduğundan proje maliyeti yüksek olur. Bu sebeple üretim zaman alır.

**2.3. Barok Modeli**

1970’li yıllarda kullanılan bu yöntem Yazılım Geliştirme Yaşam Döngüsünün (SDLC) adımlarını doğrusal şekilde geliştirildiği bir modeldir. Bu modelde belgeleme işlemi ayrı bir süreç olarak ele alınır ve yazılımın gerçekleştirme aşamasından sonra yapılır. Günümüzde kullanımı azalmaktadır.

**2.4. Çağlayan (Şelale) Modeli**

En eski, en tanınmış ve en temel modeldir. Geçmişte sıkça kullanılan ve popüler olan bu model günümüzde pek rağbet görmemektedir. Çağlayan (Şelale) Modeli belli safhalara ayrılmıştır. Bir safha bitirilmeden diğer safhaya geçilemez. Her safhada belgeleme ve test işlemleri olmak zorundadır. Safhalar arasında geriye dönüşler olabilir.



**2.4.1. Kullanım Alanları**

Küçük, basit ve gereksinimleri anlaşılır projelerde kullanılır.

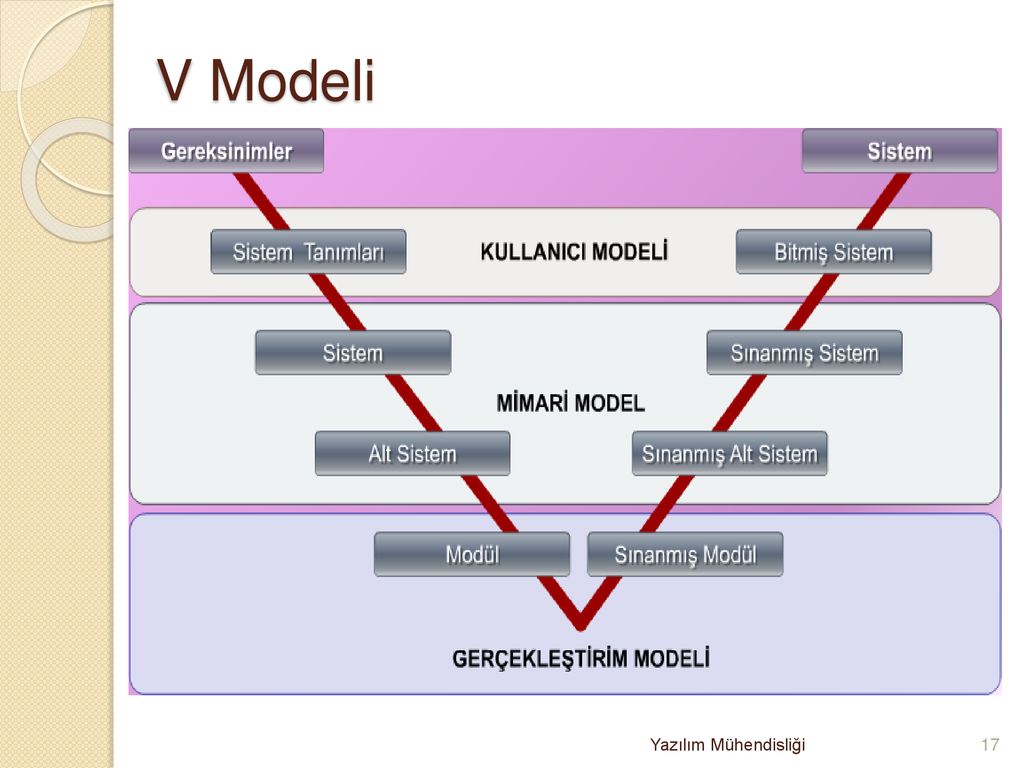
**2.4.2. Avantajları**

Belli başlı safhalara ayrıldığı için anlaşılması ve yönetimi kolaydır. Her safhada belgeleme işlemi yapıldığı için proje takibi kolaydır.

**2.4.3. Dezavantajları**

Bu süreçlerde kullanıcı yer almadığı için proje bitiminde çok fazla geri dönüş olabilir. Bu durum maliyeti yükselten bir durumdur. Uygulanması zaman alan bir modellemedir. Dolayısıyla uzun ve karmaşık projelerde ekstra maliyet ve zaman kaybı gibi olumsuz sonuçlar doğurabilir. Hatta projenin iptal olmasına sebebiyet verebilir. Yazılım üretim ekipleri, hızlı bir şekilde yazılım ürünü oluşturma eğilimde oldukları için bu modelleme ekibi bunalıma bile sokabilir. Nesne yönelimli projeler için uygun değildir.

**2.5. V Modeli**



Sol tarafı üretim, sağ tarafı test işlemleridir. Bu model kullanıcının projeye olan katkısını arttırmaktadır.

**2.5.1. Kullanım Alanları**

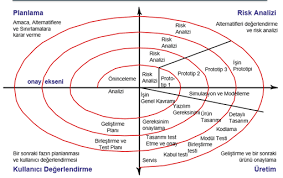
Belirsizliğin az olduğu, iş tanımlarının kesinlikle belirgin olduğu projelerde kullanılır. BT (Bilgi Teknolojileri) projelerinin iki aşamalı olarak gerçekleşebilmesi için uygun bir modeldir. İlk aşamada iş analizi ve kabul sınamalarının tanımları yapılıp kullanıcı modeli hedeflenir. İkinci aşamada ise hedeflenen model gerçekleştirilmektedir.

**2.5.2. Avantajları**

Takibi ve kullanımı kolay bir modeldir. Doğrulama ve onaylama sadece son üründe değil tüm teslim edilebilir ürünlerde uygulanır.

**2.5.3. Dezavantajları**

Aşamalarda geri dönüşler yoktur.

**2.6. Helezonik** **Model**

Bu model aynı safhalara tekrar dönülmesini zorunlu kılar. Prototip üretme ve risk analizini ön planda tutar. Her döngü bir faza karşılık gelir. Çağdaş modellere son derece yakındır.

**2.6.1. Planlama**

Üretilecek ara ürün için planlama yapılır, amaç belirlenir ve bir önceki adımda üretilen ara ürün ile bütünleştirilir.

**2.6.2. Risk Analizi**

Risk seçenekleri araştırılıp riskler belirlenir.

**2.6.3. Üretim**

Ara ürün üretilir.

**2.6.4. Kullanıcı Değerlendirmesi**

Kullanıcı ara ürün ile ilgili testleri ve değerlendirmeleri yapar.

**2.6.5. Kullanım Alanları**

Önceden geliştirilen yazılımların yeniden kullanıldığı projelerde kullanılabilir. Güvenlik yazılımlarında tercih edilir.

**2.6.6. Avantajları**

Kullanıcı sistemi erken görme şansına sahiptir. Bu model projeyi parçalara böler ve önce risk taşıyanı çözer. Böylelikle yaşanabilecek zorlukları engeller. Projenin sürekli bir prototip üretmesinden dolayı proje takibi kolay olur. Yazılımın üretilmesi ve test edilmesi gibi aşamalara erken başlanır.

**2.6.7. Dezavantajları**

Kompleks bir yapıya sahip olduğundan küçük ve düşük riskli projeler için oldukça maliyetlidir.

**2.7. Artımsal Geliştirme Modeli**

Ürünler üretilirken çekirdek yapının korunması esas alınmıştır. Yazılım parça parça geliştirilip teslim edilir. Bu modelde bir taraftan üretim bir taraftan kullanım yapılabilir.

**2.7.1. Kullanım Alanları**

Daha önceden oluşturulmuş bir üründe değişikliğe ihtiyaç varsa bu model kullanılabilir.

**2.7.2. Avantajları**

Gereksinimler kullanıcılarla birlikte belirlenir. Bu durum oluşan ürün üzerindeki geri dönüşleri azaltır. Sistem sürekli sınandığı ve iskeletin korunduğu için projenin başarısız olma ihtimali azalır.

**2.7.3. Dezavantajları**

İlgili personellerin deneyimli olması beklenir. Projenin iyi tanımlanması gerekir.

**2.8. Çevik Modeller**

Yazılım geliştirme süreci iteratif (yinelemeli) bir süreçtir. Bu süreçte kimi zaman uzun olmakla beraber üstlenilen projenin zamanında teslim edilememesi, hataların geç fark edilmesi, istenilen memnuniyetin sağlanamaması gibi durumlar söz konusu olabilir.

Yazılım geliştirenlerin bu tür sorunları aşmak amacıyla yaptıkları çalışmalar sonucu 1990’lı yılların sonlarına doğru “çevik (agile)” olarak isimlendirilen metotlar geliştirmişlerdir. Çevik yazılım geliştirme metotları kısa sürede bir yazılım ürününü müşteri hizmetine sunmayı ve değişen isteklere hızlı bir şekilde geri dönüş yapabilmeyi amaçlamıştır.

**2.8.1. Çevik Yazılım Geliştirme Manifestosu**

2001 yılında Kent Beck ve 16 arkadaşı tarafından oluşturulmuştur. Bu manifestoda:

* Süreçler ve Araçlar yerine Bireyler ve Etkileşimler,
* Kapsamlı Belgeler yerine Çalışan Yazılım,
* Sözleşme Görüşmeleri yerine Müşteri İlişkileri,
* Plan İzleme yerine Değişikliğe Açıklığın, daha önemli ve öncelikli olduğu belirtilmektedir.

**2.8.2. Çevik Yazılım Geliştirme Prensipleri**

Çevik yazılım geliştirme manifestosunun altında yatan temel prensipler şunlardır:

* Müşteriye hızlı ve sürekli kullanılabilir yazılım teslimatı yapmak,
* Kodlamanın ilerleyen safhalarında bile gereksinim değişiklikleri kabul edilir,
* Kısa zaman aralıklarında çalışan kaliteli yazılım teslimatı yapılır,
* Tüm yazılım geliştirme ekibi birebir iletişim halinde olmak şartıyla birlikte çalışırlar,
* Ekibin motivasyonunu yüksek tutmak önemlidir,
* Düzenli aralıklarla ekip kendi yöntemlerini gözden geçirip verimliliği arttırmak adına gerekli iyileştirmeleri yapar.

**2.8.3. Kullanım Alanları**

Karmaşık ve büyük projelerde kullanılması uygun olmakla birlikte her türlü proje için uygulanabilir.

**2.8.4. Avantajları**

Kısa döngüler dolayısıyla ekip içi motivasyonu yüksek tutar. Sık ara ürün üretilmesi ve müşterilerden geri bildirim alınmasından ötürü müşteri ihtiyaçlarını karşılamada başarılıdır. Değişime açık ve esnektir. Kısa zamanda kaliteli ürün elde edilir.

**2.8.5. Dezavantajları**

Kurumsal yapılarda uygulanması zordur. İhtiyaçların sürekli değişmesinden ötürü fazla mesai harcanır.

Çevik yazılım geliştirme metotları kendi içerisinde özü aynı, pratikleri farklı olan metodolojilere ayrılmaktadır.

Bunlar:

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

**2.8.6. SCRUM**

Bu metodojiler içerisinde popüler olanlarından SCRUM’ın adı Rugby sporundaki bir hücum taktiğinden gelmektedir. Bu taktikte top, tüm oyuncularla birlikte karşı sahaya taşınarak atak yapılmaktadır. Jeff Sutjerland ve Ken Schawaber tarafından 1990’lı yılların ortalarında geliştirilen bu metodoloji her türlü projeye uygulanabilmekte olan bir proje yönetim yaklaşımıdır. Bu metodolojide bir yinelemenin tamamlanması 30 günden fazla sürmemekte ve günlük 15 dakikalık toplantılarla sürekli iş takibi yapılabilmektedir.

**2.8.6.1. Kullanım Alanları**

Bu metodoloji gereksinimleri belli olmayan, değişime açık, karmaşık yazılım projeleri için kullanılır. Ayrıca Microsoft, Facebook, Google gibi dev firmaların da tercih ettiği bir metodolojidir.

**2.8.6.2 Avantajları**

Ekibin sürekli iletişim halinde olması projenin takibini kolaylaştırır. Projenin parçalara ayrılması olası sorunların hızlı tespit edilip düzeltilmesinde zaman kazandırır. Müşteriler ve geliştiricilerin doğrudan temasa geçmedikleri için yazılım geliştiren ekibin üzerinde bir baskı oluşmayıp bir güven ortamı oluşur.

SCRUM’da üç temel kavram vardır:

1. Roller
2. Toplantılar
3. Bileşenler/Araçlar

**3. Karşılaştırma**

Belirlenmiş modeller arasında bir karşılaştırma yapmak oldukça zor olacaktır. Çünkü bütün modeller birbirlerinin açığını kapatmak üzere oluşturulmuştur. Buna rağmen bir genel kıyaslama yapacak olursak:

* Tek kişinin çalıştığı basit projeler için Kodla ve Düzelt Modeli uygun olacaktır.
* Basit, sade, anlaşılır projeler için Çağlayan (Şelale) Modeli ve V Modeli uygun olacaktır.
* Daha büyük projeler ile gereksinimleri daha karışık projeler için uygulanması daha kolay olan Çevik Modeller uygun olacaktır.
* Daha önceden yapılmış bir ürünün geliştirilmesi, istenilenler doğrultusunda değiştirmesi gibi projelerde Helezonik Model ve Artımsal Geliştirme Modeli uygun olacaktır.
* Zamana ihtiyacı olan projeler için düşük maliyetli ve başarı oranı yüksek olan, Artımsal Geliştirme Modeli uygun olacaktır.
* Maddi kaynak sıkıntısı olmayan, kısa zamanda teslim edilmesi gereken projeler için yüksek maliyetli ve başarı oranı yüksek olan Çevik Modelleri kullanmak uygun olacaktır.

**4. Sonuç**

Bu çalışma kapsamında, yazılım proje yönetimi açısından önemli bir yere sahip olan yazılım geliştirme yaşam döngüsü ve modelleri incelenmiştir. Bu modellerin karşılaştırılması yapılmıştır. Uygulanacak olan projeye göre en uygun olan modelin tercih edilmesinin daha doğru olacağı sonucuna varılmıştır.

**Yararlanılan Kaynaklar**

**Resimler:**

<https://systemanalize.files.wordpress.com/2016/03/c59felale.jpg?w=385&h=267>

<https://slideplayer.biz.tr/slide/15855698/88/images/17/V+Modeli+Yaz%C4%B1l%C4%B1m+M%C3%BChendisli%C4%9Fi.jpg>

**Kaynaklar:**

<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>

<https://furkanalniak.com/yazilim-muhendisligi-yazilim-surec-modelleri/>

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

<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>

**Hesaplar**

**Medium**

<https://ozmenkahveci.medium.com/yaz%C4%B1l%C4%B1m-geli%C5%9Ftirme-ya%C5%9Fam-d%C3%B6ng%C3%BC-ve-modelleri-c2dff08c02c7>

**GitHub**

<https://github.com/ozmenkahveci/yazilim_gelistirme/wiki/Yaz%C4%B1l%C4%B1m-Geli%C5%9Ftirme-Ya%C5%9Fam-D%C3%B6ng%C3%BC-ve-Modelleri>