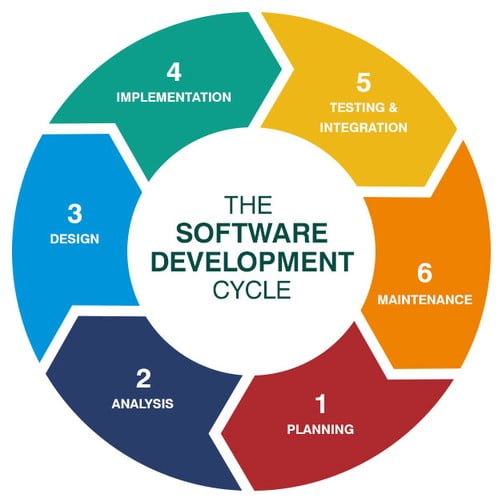
**Yazılım Geliştirme Yaşam Döngüsü(SDLC) ve Modelleri Nelerdir?**

Yazılım Geliştirme Yaşam Döngüsü, yazılımları geliştirmek, tasarlamak ve test etmek amaçlı kullanılan bir süreçtir. Yazılımın nasıl geliştirileceği, sürdürüleceği ve daha iyi hale nasıl getirileceğinin açıklayan bir plandan oluşur. Buradan da yazılımın aslında bir ürün olduğu ve o ürününde bir yaşam süreci olduğunu gözlemleriz.



**Yazılım Yaşam Döngüsü Nedir?**

Yazılım bir üründür ve ürünlerinde bir yaşam süreci vardır. Yazılımın yaşam döngüsü tek yönlü değildir. Bazı aşamalardan oluşur. Bu aşamalar Planlama, Analiz, Gerçekleştirme, Tasarım ve Bakımdır. Yazılım projeleri analiz ve planlanmasından başlayarak kullanımı boyunca süren bir döngüye denir.

Yazılım yaşam döngüsünün temel adımları:

* **Planlama (Planning)**
* **Analiz (Çözümleme) Aşaması**
* **Tasarım (Design)**
* **Gerçekleştirme (Implementation)**
* **Bakım (Maintenance)**

**1. Planlama Aşaması:**

**Yazılım projesinin planlaması ve görev dağılımı yapılır.**

**2. Analiz (Çözümleme) Aşaması:**

Yazılım projesinin ne kadar süreceği ve ne gibi risklerinin olacağı belirlenen aşamadır.

**3. Tasarım Aşaması:**

Yazılım projesinin nasıl sürdürüleceğinin tasarlanması.

**4. Gerçekleştirme Aşaması:**

Yazılım projesinin kodlama ve test kısmıdır.

**5. Bakım Aşaması:**

Yazılım projesi ürün olarak sunulduktan sonra bakım, güncelleme olaylarının yapıldığı aşamadır.

**Yazılım Yaşam Döngüsü Modelleri**

Yazılım yaşam döngüsünde birden fazla model vardır. Yazılım projesinin büyüklüğü, yazılım projesinin kimler için kullanılacağı ve benzeri durumlardan dolayı birçok model ortaya çıkmıştır.

Popüler yazılım yaşam döngüsü modelleri şunlardır:

V Süreç modeli

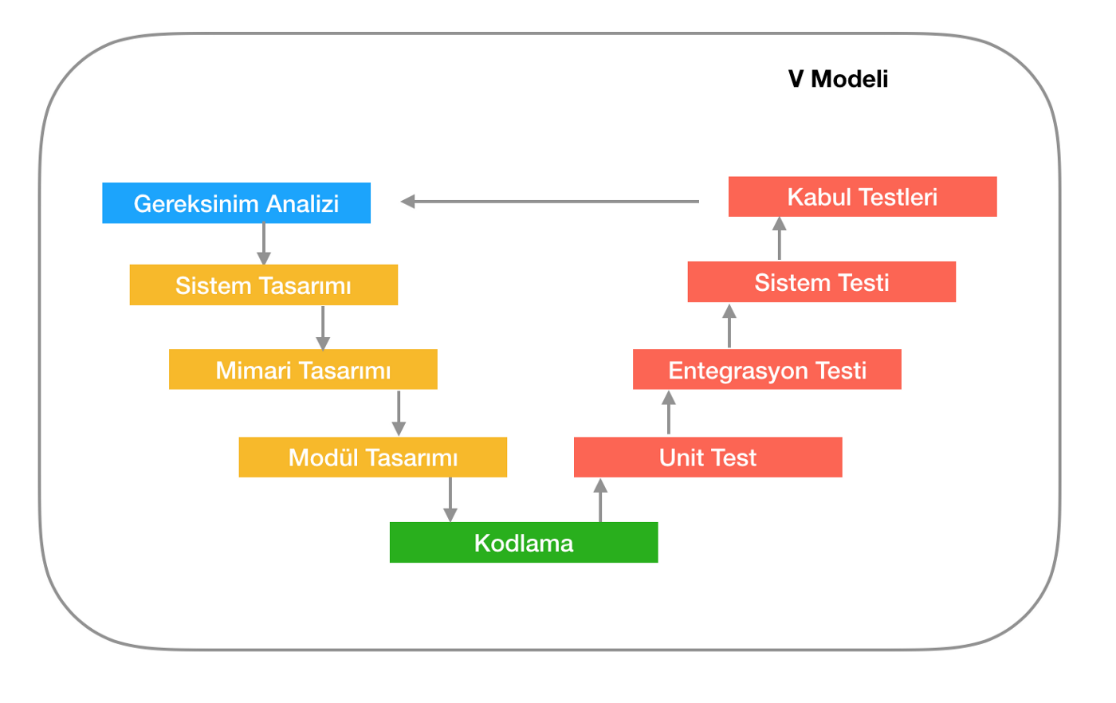
Helezonik (Spiral) Model

Kodla ve düzelt

Çevik modeller

Artırımlı Geliştirme Süreç Modeli

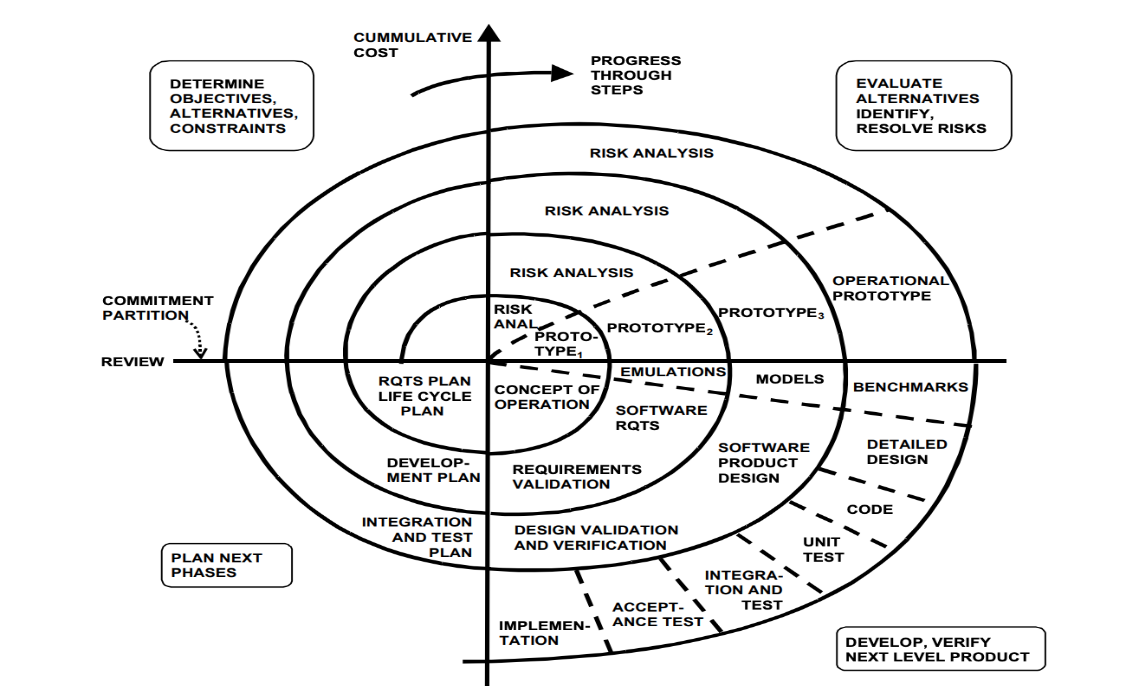
Çağlayan modeli



**1.V Süreç modeli:**

İsmini V şeklinde olmasından almıştır. Şelale modelinin daha gelişmiş halidir. Sol üretim, sağ ise sınama bölümüdür. Üst ve alt seviye tasarım vardır. Üst seviye daha genel bir tasarımdır. Bu yöntemde proje takibi kolaylaşır. Modelin kullanımı genel olarak basittir.

Avantajları:  
Kullanımı kolay bir modeldir.  
Proje yönetiminde takibi kolaydır.  
Dezavantajları:  
Aynı zamanda gerçekleşebilecek olaylara fazla imkan tanımaz.  
Risk çözümleme aktivitesini içermez.

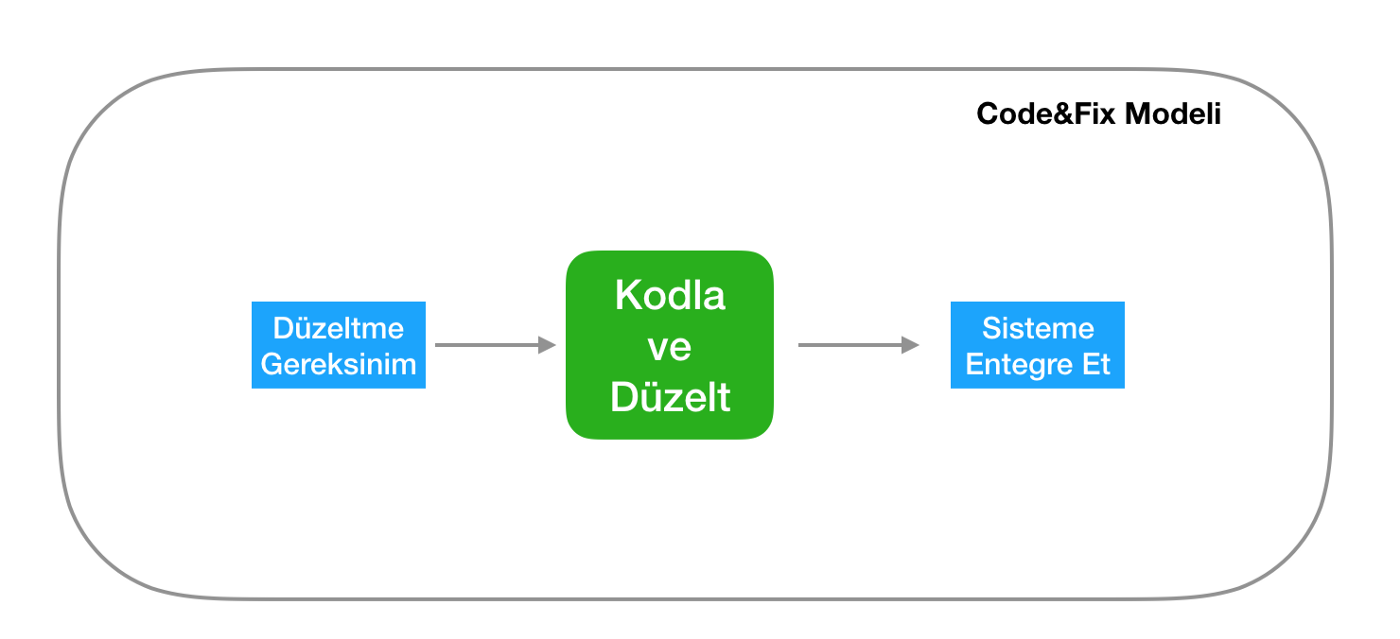


**2. Helezonik (Spiral) Model:**

 Planlama, üretim, risk analizi ve kullanıcı değerlendirme olarak 4 aşamadan oluşmaktadır.  Bu model komplike bir yapıya sahiptir. Spiral bazı durumlarda sonsuza dek sürebilir. Bu süreçte karmaşıklık daha da artacaktır. Küçük projeler için pahalı bir modeldir. Çağdaş modellere yakın bir modeldir.

Altı temel görev vardır:  
Müşteri iletişimi  
Planlama  
Risk analizi  
Yazılım tasarımı  
Üretim-Dağıtım  
Müşteri onayı

Avantajları:  
İhtiyaçları daha iyi belirlemektedir. Bu sayede ileride oluşabilecek istenmeyen durumları engellemektedir.  
Yazılımın sınanması ve kodlaması daha erken başlamaktadır  
Dezavantajları:  
Risk analizinde yüksek uzmanlık gerektirir.  
Ara adımların fazlalığı nedeniyle aşırı dokümantasyon gerektirir.  
Maliyeti yüksektir.  
Küçük projeler için uygun değildir, büyük yazılım projelerini yönetmek ve tasarlamak için daha uygun bir modeldir.



**3. Kodla ve düzelt:**

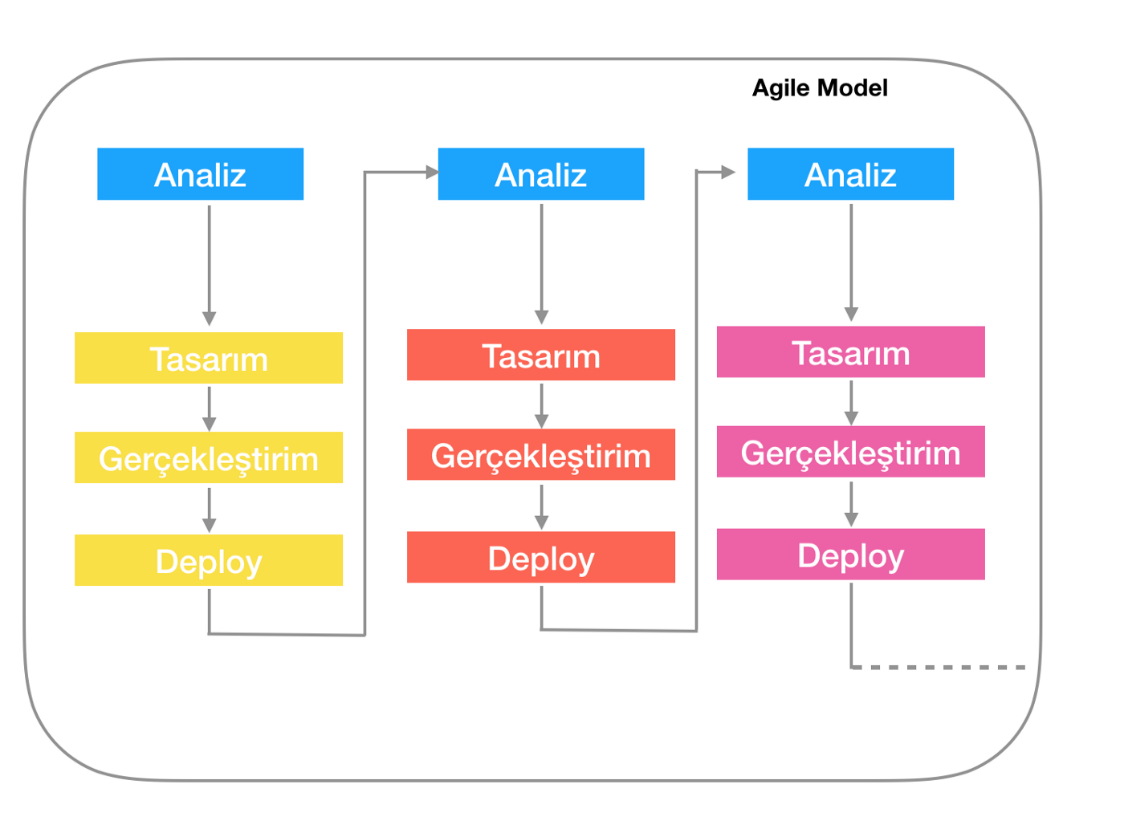
Belli bir sistemden veya düzenden söz edilemez.Bu modelde program ürün hazır olana kadar veyahut gerekli zaman bitene kadar kodlama yapılarak devam eder.

Avantajları:

Bir planlamaya ihtiyaç duyulmaz.  
Program aşamaları çabuk geçilir.  
Bu modeli amatörler bile rahatlıkla kullanabilirler.

Dezavantajları:

Kontrollü değildir.  
Bitiş süresi belli değildir.  
Kaynak planlaması yoktur.  
Hatanın doğrulaması ve bulunması zordur.  
Kodu düzeltmek maliyetli olabilme ihtimali vardır.  
Kodlar kullanıcının ihtiyacını karşılamazsa ve değiştirilmesi gerekirse değiştirilmesi zor bir yapıya sahiptir. Bu model küçük projelere uygundur.



**4. Çevik modeller:**

Bu yöntemlerin amacı, geliştirme sürecinde bazı yüklerden kurtularak sonuca hızlı ulaşmasını amaçlamaktadır. Yüksek performans sağlar. Hata riski en az ve ucuz yollu çözümler oluşturur.

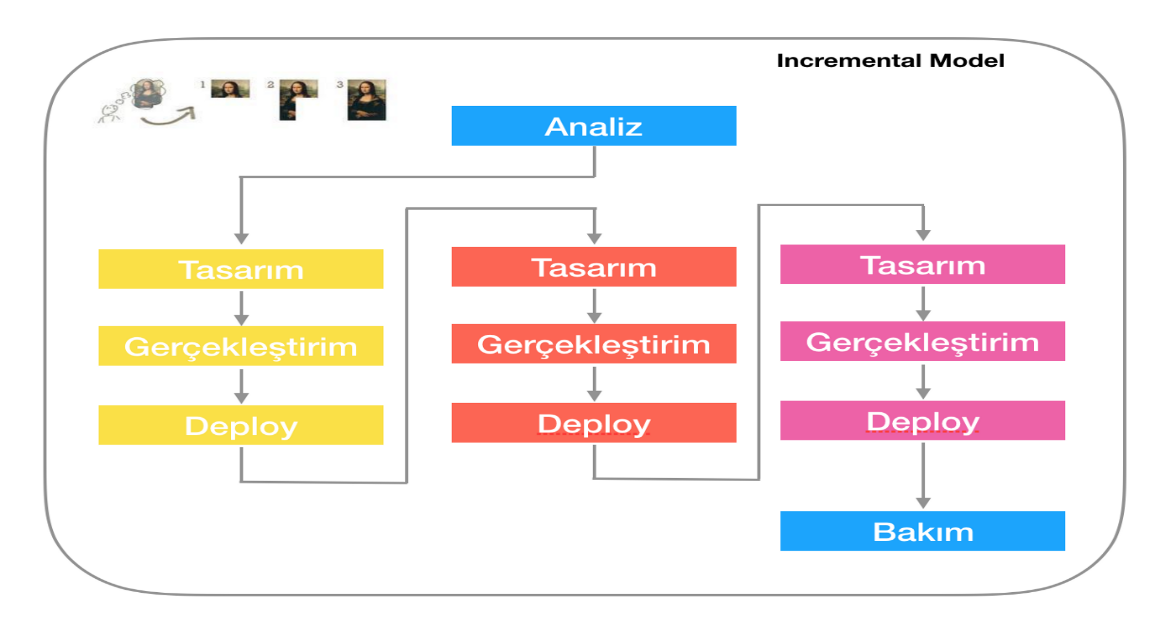
Uç Değer Programlama (Extreme Programing), Scrum, Çevik Tümleşik Süreç(Agile Unified Process) ve Özellik Güdümlü Geliştirme (Feature-Driven Development) günümüzde yaygın kullanılan Çevik Yazılım Metotlarından bazılarıdır.

Avantajları:

Takım elemanlarının motivasyonu yüksek olur. Bu nedenle verim artar.  
Değişime açıklık ve esneklik üst düzeydedir.  
Kalite sürdürülebilirdir.  
Proje yürütme ve planlama bir aradadır.  
Takım oyunu vardır.

Dezavantajları:,

Kurumsal yapıda uygulanması zordur.  
Modellerde dokümantasyona önem verilmez, gerektiği taktirde dokümantasyon yapılır.  
Sürekli değişebilen ihtiyaçlardan dolayı aşırı çalışmak gerektirebilir.  
Takımda hedef baskısı vardır.



**5. Artırımlı Geliştirme Süreç Modeli:**

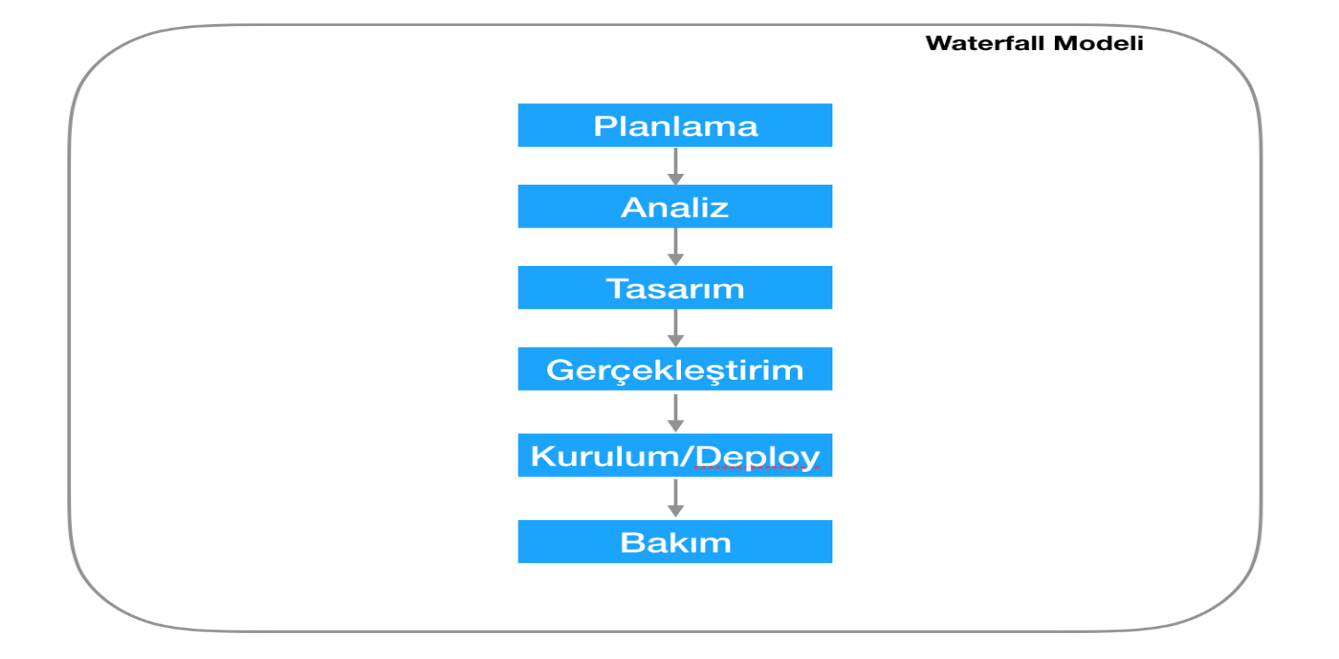
Bu modelde yazılım, bir takvime bağlı olarak geliştirilip teslim edilir. belli döngüler ile projeyi geliştirmeniz gerekir. Değişikliğe ihtiyaç varsa model ayak uydurabilir.  
Bu model kısıtlı sayıda çalışanla işin yapılmasına imkan sağlar.

Avantajları:

Gerekli gereksinimler müşteri ile belirlenir. Daha sonra gereksinimler net olarak belirlenmediği için karşılaşılan ürünün yetersiz kalması ve benzeri sorunların olmasını engeller. En önemlisi sistem özelliklerinin daha fazla sınanma imkanı bulmasını sağlar.

Dezavantajları:

Bu modeli uygulamak için deneyimli personele ihtiyaç vardır.  
Bu model uzun zaman alan projeler için uygundur.



**6. Çağlayan modeli:**

SDLC aşamaları lineer şekilde uygulanır. Kullanımı ve anlaması basittir. Yönetimi kolaydır. Küçük projelerde etkili çalışır. Çağlayan modeli değişimlere uygun değildir. Büyük yazılım projelerinde sürekli farklı ve yeni fikirler ortaya çıktığı için geri dönüşler zordur. Müşterinin istekleri ve sistem gereksinimleri detaylı ve net bir şekilde belirlenmelidir. Günümüzde ise kullanımı her geçen gün azalmaktadır.

Avantajları:  
Projede disiplin sağlanabilir.  
Proje dokümantasyonu üretmek kolaydır.  
Test işlemi her aşamada yapılabilir.  
Bakım işlemi kolaydır.  
Dezavantajları:  
Bu model geri dönüşe açık bir model değildir.  
Proje büyüdükçe karmaşıklık artar ve zorluklarla karşılaşılmaya başlanır.  
Aşamalar tek tek yapıldığı için tasarım ve analiz aşamasının çok detaylı hazırlanması gerekir. Daha sonradan gereksinimlerin değişebilir. Bu nedenle projenin maliyeti ve tamamlanma süresi artar.

**Kaynaklar:**

* [**https://medium.com/architectural-patterns/yaz%C4%B1l%C4%B1m-geli%C5%9Ftirme-modelleri-62915545c51e**](https://medium.com/architectural-patterns/yaz%C4%B1l%C4%B1m-geli%C5%9Ftirme-modelleri-62915545c51e)
* [**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://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**
* Doç. Dr. Deniz KILINÇ Yazılım Mühendisliği ders notları