# YAZILIM YAŞAM DÖNGÜ MODELLERİ

## Yazılım Yaşam Döngüsü

Yazılım projelerinin üretim ve kullanım süresi boyunca geçirdikleri aşamalara **yazılım yaşam döngüsü** olarak tanımlanır. Bir yazılımın yaşam süreci doğrusal olarak ilerlemez bu süreç döngü şeklindedir, yani döngü tek yönlü değildir gerektiğinde aşamalar arasında önceki aşamaya dönmek sonra tekrar ilerlemek mümkündür.

## Yazılım Yaşam Döngüsü Temel Adımları-Çekirdek Süreçler (Core Processes)

1. *Gereksinim (Requirements)-Planlama*
   Müşteri gereksinimlerinin toplandığı, fizibilite çalışmasının yapıldığı ve projenin planlandığı adımdır.

2. *Analiz (Analysis)*
   Projede görev alan tüm çalışanlar bir araya gelip, geliştirilecek olan yazılım ürününden istenilen özelliklerin ayrıntılı olarak belirlenip somut bir şekilde yazıldığı aşamadır.

3. *Tasarım (Design)*
   Gereksinimler belirlendikten sonra bu gereksinimleri karşılayabilecek yazılım ürününün genel tasarımının yapıldığı aşamadır. İki tür tasarım mevcuttur.
   **Üst seviye ve mimari tasarım**
   **Detaylı tasarım**

4. *Gerçekleştirme (Implementation)*
   Kod modüllerinin yazıldığı, bu modüllerin birleştirilip çalıştırıldığı, kurulumun yapılıp test edildiği aşamadır.

5. *Bakım (Maintenance)*
   Yazılım ürünü teslim edildikten sonra süreç bitmez ürüne yeni özellik ekleme veya çıkan hataların giderilmesi bu aşamada yapılır.

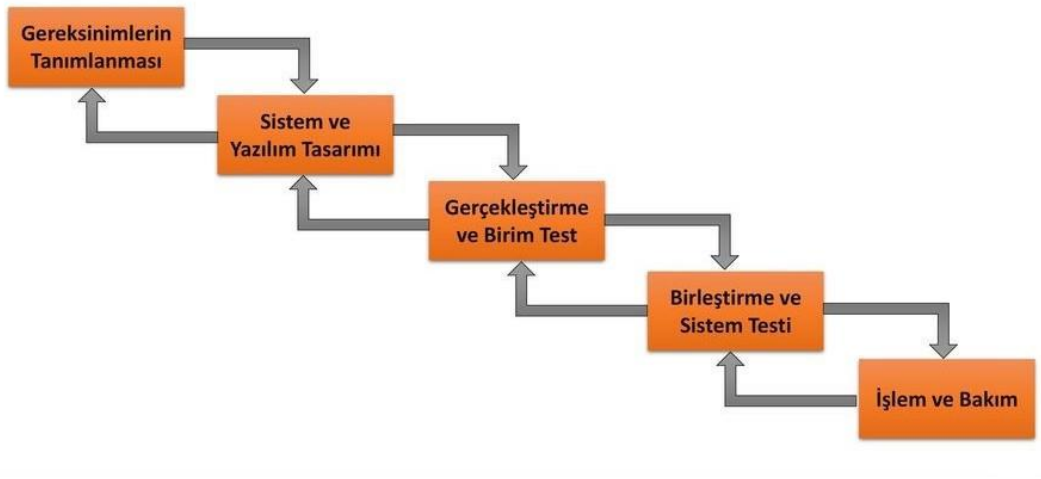
## Bazı Yazılım Yaşam Döngü Modelleri

## Barok Modeli

- İnceleme
- Analiz
- Tasarım
- Kodlama
- Modül Testleri
- Altsistem Testleri
- Sistem Testleri
- **Belgeleme**
- Kurulum

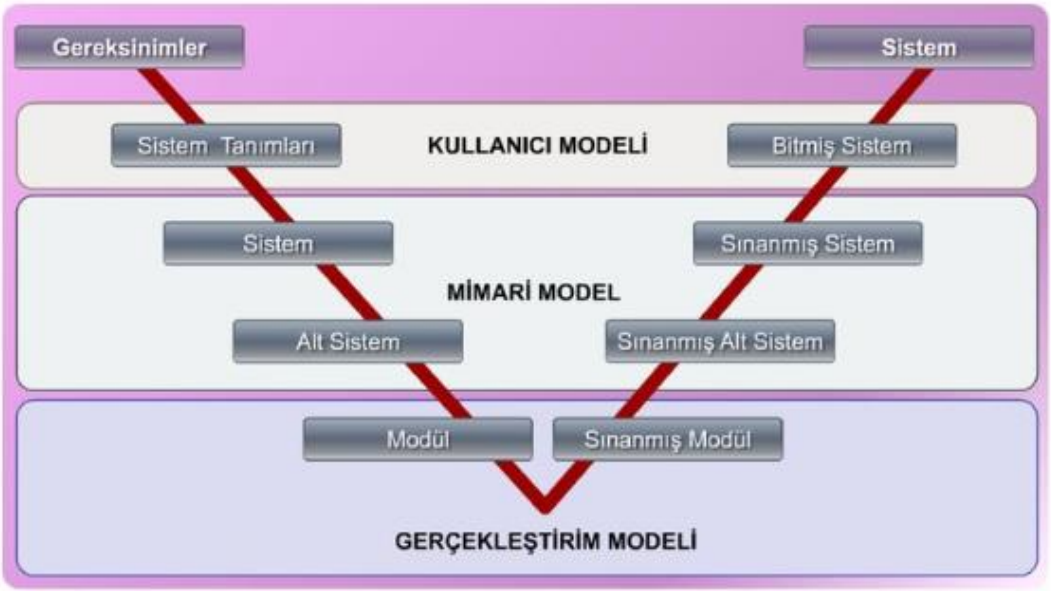
70'li yıllarda ortaya çıkan bu modelin en temel özelliği yazılım yaşam döngüsü adımları arasında doğrusal bir ilerlemenin olmasıdır. Bu modelde adımlar arasında geri dönmek yoktur. Belgeleme **ayrı bir süreç** olarak ele alınır bu modelde. Gerçekleştirme (Implementation) aşamasının üzerinde daha çok durulan bir modeldir. Günümüzde kullanılması önerilmemektedir.

## Çağlayan Yaşam-Döngü Modeli




Bu model adımların en az bir kere tekrarlanması ile gerçekleşir. Aşama aşama ilerlenip her aşamanın dökümantasyonu hazırlanır. Barok Modelinin aksine adımlar arasındaki geri dönüşler iyi tanımlanmıştır ve belgeleme süreci ayrı bir adım olarak ele alınmaz bu modelde. Geleneksel yazılım geliştirme modeli olarak da bilinir. Kullanımı ve yöntemi basittir. Geliştirme sürecinin yönetimini kolaylaştırır. Bu yöntem gereksinimleri iyi tanımlanmış küçük projeler için uygundur. Projede yapılan değişikliklerin maliyeti çok büyüktür bu yüzden bu model kullanılacaksa gereksinimler çok iyi belirlenmelidir. Müşteri sürecin içinde yer almaz değişime açık bir model değildir bu yüzden müşteri memnuniyetini sağlamak oldukça güçtür. Günümüzde kullanımı gittikçe azalmaktadır.

## V Süreç Modeli




Bu modeli Şelale modelinin gelişmiş hali olarak düşünebiliriz. Sol taraf üretim, sağ taraf test işlemleridir.

Bu model genel olarak 3 modelden oluşur;

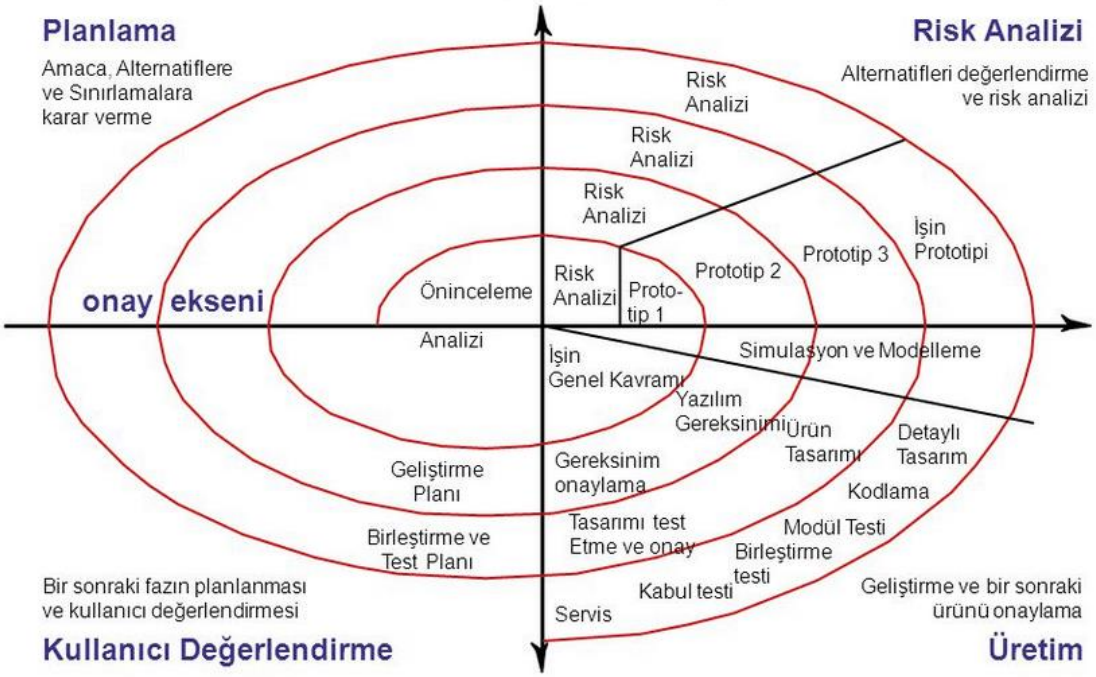
**Kullanıcı Modeli:** Kullanıcının istekleri belirlenir ve bu isteklere göre proje bitirilir.

**Mimari Model:** Sistemin mimarisi ve sistem testleri.

**Gerçekleştirim Model:** Kodlama ve kod modüllerinin testleri.

İş tanımın ayrıntılı yapıldığı projelerde kullanılması uygundur. Proje yönetimi kolaydır. Fazlar arasında tekrarlamaların olmaması bu modelin dezavantajlarındandır.

## Helezonik (Spiral) Model




Bu model dört bölüm etrafında döngüler oluşturarak meydana gelir.

**Planlama:** Geliştirilecek yazılım ürünü için planlama, amaç belirleme, bir önceki adımda üretilen ara ürün ile bütünleştirme.

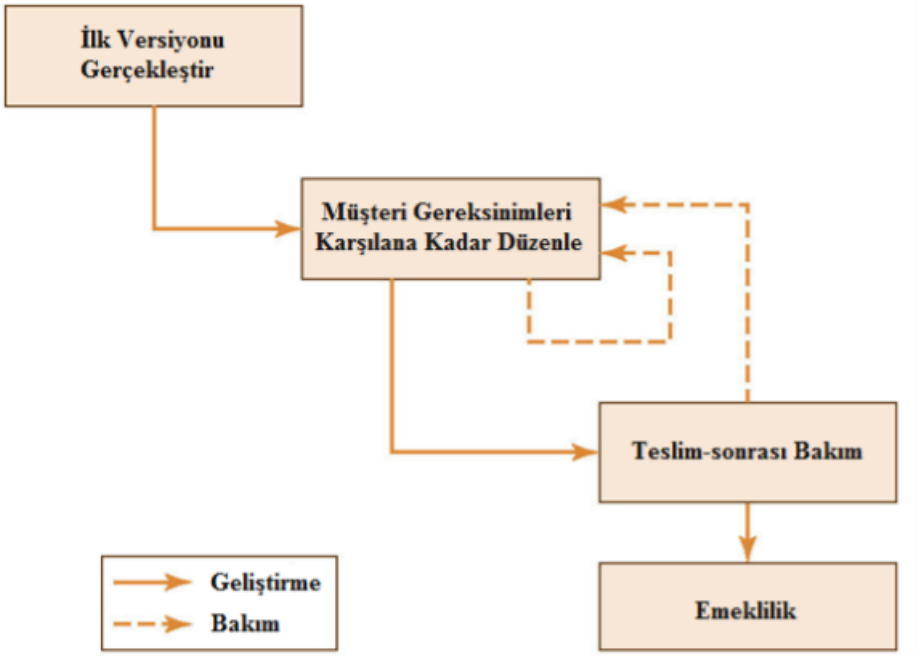
**Risk Analiz:** Risklerin belirlenmesi.

**Üretim:** Ara ürünün üretilmesi.

**Kullanıcı Değerlendirmesi:** Ara ürün ile ilgili olarak kullanıcı tarafından yapılan sınama ve değerlendirmeler.

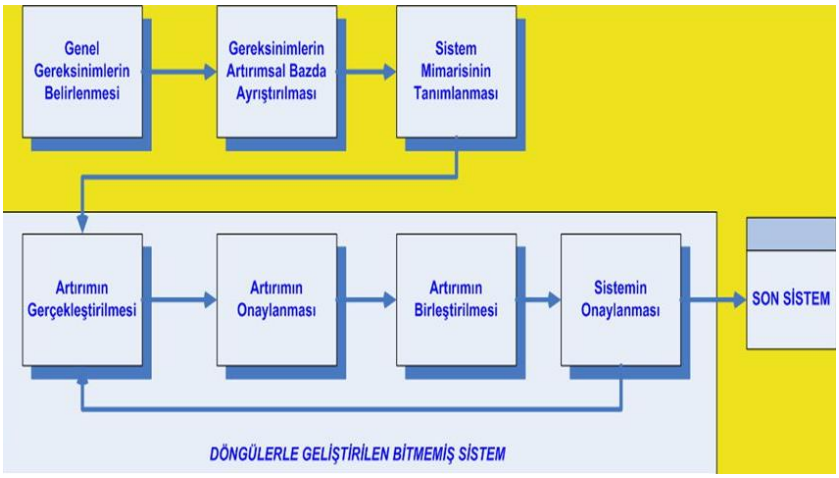
Bu modelin en önemli özelliği kullanıcı ürünü erken safhada görüp deneyebilir bu sayede hatalar erkenden görülür ve hata giderme maliyeti düşürülmüş olur. Spiraldeki her döngü bir fazı ifade eder. Riske duyarlı yaklaşımı potansiyel zorlukları engeller. Bu model geliştirmeyi parçalara böler ve en riskli parçaları önce gerçekleştirir böylelikle riskli parçalarda çıkabilecek sorunları gidermek daha az maliyete sebebiyet verir. Bu modelin küçük ve düşük riskli projelerde kullanılması önerilmez büyük ve risk faktörünün yüksek olduğu projelerde kullanılmaya yöneliktir.

## Kodla ve Düzelt Yaşam-Döngü Modeli




Bu model yazılım geliştirmenin en kolay yoludur. Süreç bir fikirle başlar ve ürüne ulaşılıncaya kadar kodlama yapılarak devam edilir. Kolay bir model olduğu için tecrübesiz ve küçük firmalar tarafından sıkça tercih edilir ancak yapılacak değişiklikler düşünülünce bu model oldukça maliyetli olabilir. Çok küçük projelerde veya prototiplerde kullanılabilir.

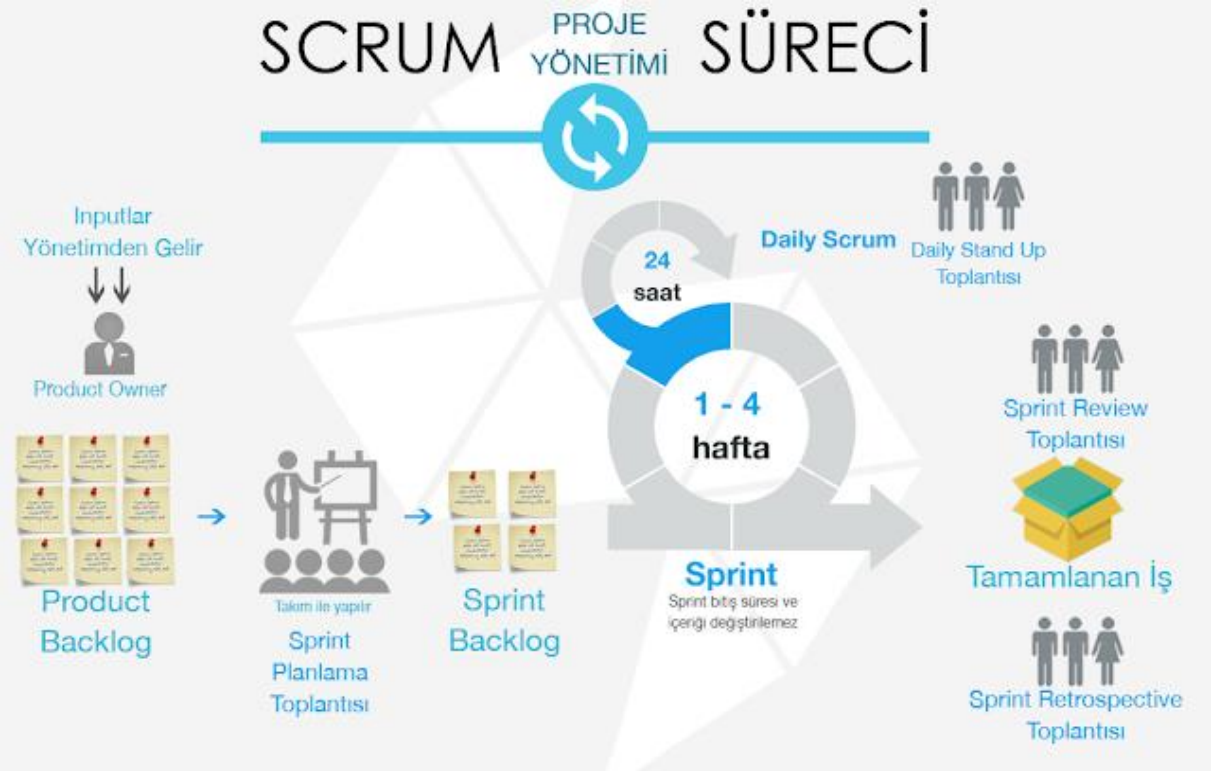
## Artımlı Geliştirme (Incremental Development)




Bu model yazılım ürününü parça parça geliştirip teslim etmeye dayanır. Her yeni parça eski parçanın üstüne yeni özellikler eklenerek oluşturulur yani eski parçalar prototip görevi görmektedir. Bu modelle az sayıda çalışanla yazılım geliştirmek mümkündür. Hem üretim ve kullanım aynı anda yapılabilir. Her artımda çalışanların projeye ilişkin deneyimleri artar. Öncelikle en önemli sistem gereksinimlerini karşılayan bir çekirdek sistem geliştirilir ve bunun üzerine artımlar yapılır böylelikle önemli gereksinimlerin test edilme sayısı artar ve projenin başarısız olma ihtimali düşer. Divide and Conquer (Böl ve Yönet) yaklaşımıdır.

## Scrum

Öncelikle scrum'un kendi rehberindeki tanımı şu şekildedir: "**İnsanların mümkün olan en yüksek değere sahip ürünleri üretken ve yaratıcı bir şekilde geliştirirken, karmaşık ve adaptasyona açık sorunları ele alabildikleri bir çerçeve**"




Scrum; Agile (çevik) proje yönetim metodolojilerinden biridir. SCRUM metodolojisi, karmaşık ortamlarda yazılım inşa etmek için kullanılan arttırımlı bir süreçtir. Bu metodoloji, esnek bir geliştirme süreci sürdürebilmek için tasarlanmıştır. Scrum ile proje üzerinde çalışmaya devam edilirken değişkenler yeniden yönetilebilmektedir. Böylelikle proje gereksinimlerinin karşılanması konusunda ciddi oranda başarı sağlar. Bu yöntemde projenin durumu, ilerlemeler, hatalar herkes tarafından görülebilir olmalıdır. Projenin ilerleyişi ve hedefler **günlük** toplantılarla sürekli takip edilmelir (Meetings). Bu toplantılarda her ekip üyesi şu sorulara cevap verir;

*Dün ne yaptım?

*Bugün neyapacam?

*Önümde olan engeller ve karşılaştığım sorunlar neler?

SCRUM' un ***Müşteri açısından avantajları***; Diğer modellerde müşteri proje başlangıcında gereksinimleri sunar ve proje sonunda ürünü alır. Scrumda ise müşteri yazılım sürecinin içindedir ve belli zaman aralıklarıyla aldığı ara ürünleri inceleyip aklına gelen yeni gereksinimleri projeye dahil edebilir veya bazı gereksinimlerin gereksiz olduğunu fark edip bunu aktarabilir. Bu ara kontroller projenin başarıya ulaşmasında büyük etkiye sahiptir.

***Şirketler açısından;*** bu yöntem iş tekrarını önler ve çalışma ekibinin verimliliğini arttırır. Müşteri memnuniyetinin sağlandığı bu yöntemle birlikte memnun müşteriler birer referansa dönüşür ve şirketin pazar potansiyeli yükselir.

## Sonuç

Başta SCRUM olmak üzere Çevik Yöntemlerin yapısı gereği proje süresince gerçekleşen değişikliklere kolay ayak uydurması ve yinelemeli süreç yapısı sayesinde esnek bir proje süreci izlenmektedir bu durum da başarı oranını arttırır.

Belli aralıklarla teslim edilen ara ürünler sayesinde proje ilerleyişi somut bir şekilde görülür, riskler düşürülür ve hataların erken tespiti sağlanır.

Proje sürecinin bir parçası olan müşteriler üründen beklenen özelliklerin tam olarak belirtilmesini sağlar. Bu durum yüksek müşteri memnuniyetini ve başarıyı kendinde getirir.

Tablo 4. Uygulanan Metodolojiye Göre Yazılım Projeleri Başarı Oranları [8]
(Software Projects Success Rates According to the Methodology Implemented)

| | **Şelale Modeli** | **Çevik Yöntemler** |
|---|---|---|
| **Başarılı Projeler** | % 11 | % 39 |
| **Sorunlu Projeler** | % 60 | % 52 |
| **Başarısız Projeler** | % 29 | % 9 |