Her ürün satışa sunulması ve sonrası için bir döngüden geçer. Örneğin tarım ürünleri yetiştirilme aşamasında sulama, ilaçlama gibi süreçlerden geçer ve sonra satışa sunulma aşamasında da belli aşamalardan geçer. Satıştan sonra her ürün tekrar üretim için en başa döner ve yine aynı süreçlerden geçer. Bir döngü şeklinde bu aşamalar sürekli tekrarlanır. Yazılım da donanımlarda çalışan program ve çeşitli dokümanlar içeren teknolojik bir ürünüdür. Ve her ürünün olduğu gibi yazılımın da geçtiği aşamalar vardır. Bu aşamaların oluşturduğu döngü, yani Yazılım yaşam döngüsünün temel aşamaları: Planlama, analiz, tasarım, uygulama, test, bakımdır.



***Planlama***: Döngünün ilk aşamasıdır. Yani elimizdeki projenin ana hatları, iş bölümü ve izlenecek yolların tartışılıp oluşturulduğu aşamadır. Bunların yanı sıra bu aşama da maliyet, ihtiyaçlar da üstün körü belirlenir.  
 ***Analiz:*** Bu aşamada hatları oluşturulan projenin bilinmeyen ve eksik notları tamamlanır. Örneğin proje tıbbi bir proje ise tıbbi terimlerin bilinmesi ve ona göre projeye devam edilmesi gerekir. Bu durumda araştırmalar yapılır ve müşteriyle iletişime geçilip terimlerin proje için yeteri derecede açıklaması alınır. Toplanan veriler bir doküman haline getirilir.  
 ***Tasarım***: Projenin ilk taslağı planlama da oluşturulurken detaylandırması tasarımda gerçekleşir. Yazılım modellerinin içeriği bu aşamada belirlenir. Projenin görünümü ara yüzü tasarımları da bu aşamada tasarlanır. Planlama taslak aşaması dersek tasarım aşaması da maket aşaması denebilir. Çünkü projenin ana hatlarının yerine oturtulup detaylandırıldığı aşamadır.   
***Uygulama :*** Tasarım aşamasındaki projenin ana hatları yerine oturtulup detaylanmaya başladığı an uygulama yani kodu yazma aşaması başlar. Bu aşama da yazılan kodlar yeni mezun olan birinin baktığında anlayabileceği kadar basit ve anlaşılır olmalıdır ki ileriki aşamalarda koda bakıldığında her şey anlaşılır ve açık olsun.  
 ***Test:*** Bu aşama projenin yazılımının, tasarımının, görünüşünün, her şeyinin bittiği ve ürünün verimli, mi? Çalışıyor mu? Kullanışlı mı? gibi faktörlerin son kontrol ve testlerinin yapıldığı aşamadır. Bu aşamanın ayrı bir önemi vardır ki projede herhangi bir problem sıkıntı hata varsa ileriki zamanlarda maliyet, zaman ve çeşitli kayıplara neden olmadan sorunu belirleyip çözüm üretilebilsin. Test olaraksa çeşitli testler mevcuttur. Bunlara örnek verecek olursak, birim test yazılan kodun birimlere ayrılıp her birimde ki her fonksiyonun çalışıp çalışmadığının kontrol edildiği testlerdir. Fakat şu da unutulmamalıdır birim testteki test edilen her birimin doğru olması yazılımın bütünün kusursuz ve sıkıntısız çalıştığı anlamına gelmez. Başka test örnekleri verecek olursak, uygulamaya yeni özellikler veya işlevler geliştirildiğinde ve mevcut bir yapıya eklendiğinde testler yapılır. Tüm kritik işlevlerin doğru çalışıp çalışmadığını kontrol eden duman testleri vardır. Yazılımın işlevlerinin kesinlikle iyi çalıştığından emin olmak için yapılan testlerdir. Tüm projeyi test eden geniş bir test olduğu için büyük çaplı hataları tespit eder. Detaylı bir test değildir. Kabul testleri projenin neredeyse bitmiş durumdaki prototipini, ilgili iş birimi ya da kullanıcılar tarafından kullanılıp projenin sınandığı testlerdir. Yük testleri projenin karşılaşacağı durumların ön izlemesi gibidir. Kullanıcı kapasitesine ulaştığında karşılaşacağı hız gibi sorunlara projenin ne kadar dayanabileceği kontrol edilir. Bunlar gibi daha bir sürü kategori veri durumu test eden çeşitli testler uygulanmaktadır.   
***Bakım:*** Bu aşama da bitip kullanıcıya sunulan projenin kullanım sırasındaki oluşan problemleri düzeltmek içindir. Ve eksik olan kısımlar çeşitli güncellemelerle giderilir.

***Yazılım Yaşam Döngüsü Modelleri:***

***1) Gelişigüzel Model:*** Aslında tam olarak bir yazılım modeli denemez. Çünkü belli bir yöntemi yoktur. Adı gibi yazan kişinin gelişigüzel oluşturduğu yazılımlardır. Hatta üzerinden belli bir süre geçtikten sonra yazan kişi bile kodu anlamada zorluklar çekebilir. Genellikle bir kişinin kullandığı bir modeldir. Bu modelin takibi yani izlenebilirliği ve bakımı çok zordur.

***2)Barok Modeli:*** Bu model 70’li yıllarda kullanılan bir modeldir. Yazılım yaşam döngüsündeki aşamaları alışveriş listesi gibi sırayla tamamlar ve bitirir. Yani geri dönüşün nasıl yapılacağı tanımlanmamıştır. Bir döngü yoktur. Aşamalar yapılmış ve bitmiştir. Ve bu modelde dokümantasyon aşaması günümüzdeki modellerden farklı olarak tüm proje bitince yapılır. Günümüzde kullanımı bırakılmış bir modeldir.

***3)Şelale(Çağlayan)Modeli:*** En temel, en tanınmış, en eski modeldir. Günümüzdeki çoğu modelin temelleri bu modele dayanır. Bu modelde bir aşama bitmeden diğer aşamaya geçilmez. Ve her aşama tamamlandıktan sonra o aşamanın dokümantasyonu yapılır. Dokümantasyon tamamlandıktan sonra diğer aşamaya geçilir.   
**Avantajları:**   
\*Her aşamada test yapılabilir ve bu da kolaylık sağlar.  
\*Bakım yapması kolaydır.  
\*Anlaması ve kullanması kolaydır.  
**Dezavantajları:**  
\* Bir aşama bitmeden diğer aşamaya geçilmez. Bu yüzden projenin teslim süresi uzar.  
\*Geriye dönüşler zor olduğu için önceki aşamalarda yapılan hatalar büyük maliyet sorunlarına yol açabilir.  
\*Uzun ve karmaşık projelerde kullanılması zordur.  
\*Gereksinimlerde yapılan minicik değişimler bile maliyeti arttırabilir.  
\*Gelişim ve değişime çok fazla açık değildir bu da müşteri memnuniyetini yüksek oranda azaltır.

***4)V Modeli:*** Bu model şelale modelinin biraz değişmiş bir modelidir. Aşamalar v şeklinde dizilir. Sol tarafta üretim sağ tarafta test aşamaları vardır. Testin ön planda olduğu bir modeldir. Testler dokümantasyondan sonra başlar. Bu modelde kullanıcı projeye daha çok katılım gösterir.  
**Avantajları:**\*Kullanımı basittir.  
\*Takip etmesi kolaydır.  
**Dezavantajları:**\*Aşamalar arasında tekrarlama olmaz.

***5)Spiral(Helezonik) Model:*** Bu modelde belli zaman aralıklarıyla yazılım gereksinimleri belirlenir. Karmaşık bir yapısı vardır. Ve yapımı maliyetlidir. Fakat başarı oranı yüksektir. Bu modeldeki en temel özellik risk analizi ve prototip üretmedir. Her döngü sonunda planlamalar yeniden yapılır. 4 bölümden oluşur bunlar sırasıyla planlama, risk değerlendirme ve azaltma, geliştirme ve doğrulama, kullanıcı değerlendirmesidir. Planlamada aşamanın başarımı için bazı hedefler belirlenir. Riskleri belirleyip bu risklerin ortadan kaldırılması için çözümler üretilmesi risk değerlendirme ve azaltma bölümünde yapılır. Modeller arasından model seçilmesi geliştirme ve doğrulama bölümünde yapılır. Daha proje tamamen sonlanmadan ortaya çıkan ilk versiyona ara ürün denir. Bu ara ürünün kullanıcı tarafından değerlendirilmesi de kullanıcı değerlendirmesi bölümünde yapılır. Bunlar spiral modelin temel kısımlarıdır. Tanımlama tasarım gibi aşamaları yoktur. Her aşamada ihtiyaç ve gereksinimlere göre karar verilir.  
**Avantajları:**\*Bir çok yazılım modelini içinde bulundurur bu yüzden en çağdaş yazılım modellerinden biridir.  
\*Risk analizine önem verilir. Bu yüzden ileriki aşamalarda zaman ve maliyet avantajı sağlar.  
\* Kodlama ve test süreci projenin başlarında başlar.  
\*Pek çok yazılım modelini içinde bulundurur.  
**Dezavantajları:**\*Küçük projeler için maliyeti çok yüksektir.  
\*Diğer modellere göre karmaşık bir yapıya sahiptir.  
\* Dokümantasyon işlemi fazladır

***6) Artımsal Geliştirme Süreç Modeli:*** Genellikle uzun zaman alabilecek ve sistemin eksik işlevlerle çalışabileceği projelerde kullanılır. Bu modelde gereksinimler parça parça artarak tamamlanır. Bir ana proje vardır ve onun üzerinden eksikler ve güncellemelerle ana projenin üzerine eklenip sürekli artan bir yapıya sahiptir. Gereksinimler müşterinin verdiği değerlendirmelere göre öncelik sırasına dizilir ve öyle gerçekleştirilir. Müşterinin değerlendirmeleri dikkate alınarak yapılan bir modeldir.

**Avantajlar:** \*Gereksinimler müşteriyle birlikte belirlenir.  
\*Gereksinimi giderecek en iyi sistem seçilir ve böylece sistemin başarısızlık oranı azaltılır.   
\*Her artırım sonrası müşteriye teslim edilen ürün erken verildiği için olumlu ve olumsuz yönler ortaya çıkıp gereksinimler bunun üzerinden sağlanır.  
**Dezavantajları:**\*Eklenecek gereksinimleri anlamak için tüm sistemi iyice anlamak gerekir.  
\*Eklenecek gereksinimler kendi içlerinde tekrar kullanılamazlar.   
***7)Kodla Ve Düzelt Yazılım Döngü Modeli:*** Yazılım süreci kısa. Bir kaç yüz satırdan oluşan yazılımlar için kullanılır. İlk yazılan kod ilk üründür. Ve sürekli geliştirilerek devam ettirilir. Yani güncellenir.   
**Avantajlar:**\*Aşamalar çok hızlı bir şekilde geçilir.  
\*Yazılımı gerçekleştirmek için en kolay yollardan biridir.  
\*Uzman görüşüne çok gerek duyulmaz herkesin kullanabileceği bir modeldir.  
**Dezavantajları:**\*Emeklilik safhası vardır.  
\*Bitiş süresi belli değildir.  
\*Hatların bulunması ve hataları düzeltmesi zordur.  
\*Kodlardaki sorunları sonradan gidermek maliyet artışına sebep olabilir.  
\*Bakım safhası çok zordur çünkü bu modelde dokümantasyon yoktur.  
\*En pahalı yazılım geliştirme modelidir  
\*Yazılımı kolay olduğu için küçük veya tecrübesiz firmalarda sıkça kullanılır.

***Hangi Projede Hangi Yazılım Modeli Kullanmalıyız? :***

***1-)Kodla ve Düzelt (Code and Fix) Modeli:*** Bu model en basit modellerdendir çünkü bu modelde amaç en kısa sürede bir sonuç yani program elde etmektir. Planlama, analiz yapma gibi kısımlar ile çok uğraşmadan hemen sonuca gitme hedeflenir. Bu yüzden de küçük çaplı bir kişinin yazdığı basit programlarda kullanılır.  
\*Gereksinimler başlangıçta belirlenir.  
\*Maliyeti düşüktür.  
\*Başarı garantisi düşüktür.  
\*Aşamalar çakışmaz.  
\*Uzmanlık gerekliliği düşüktür.  
\*Basittir.  
\*Bakım oranı düşüktür.  
\*Değişiklik yapmak kolaydır.  
\*Yeniden kullanılma oranı düşüktür.  
\*Uygulamak kolaydır.  
\*Çok uzun sürer.  
***2-) Şelale(Çağlayan)Modeli:*** Analiz aşamaların sırayla yapıldığı ve bir aşama bitmeden diğer aşamaya geçilmeyen bir modeldir. Küçük ve gereksinimleri çok iyi anlaşılmış projelerde kullanılması daha rahattır. Eski askeri yazılımlar ve projelerde sıkça kullanılmış bir modeldir. Ama sonradan kullanışlı çokta basit olmadığı anlaşılmıştır. Analizin süreç içinde değişiklik gösterdiği, kodlama sırasında tasarımın değiştiği fark edilmiştir. Zamanla bu modeli temel alan daha uygun farklı modeller uygulanmıştır.   
\*Gereksinimler başlangıçta belirlenir.   
\* Maliyeti yüksektir.   
\*Başarı garantisi düşüktür.   
\*Uzman gerekliliği orta düzeydedir.   
\*Aşamalar çakışmaz.  
\*Basittir.   
\*Bakım oranı düşüktür.  
\*Değişiklik yapmak zordur.   
\*Yeniden kullanılma oranı düşüktür.  
\*Uygulamak kolaydır.   
\*Çok uzun sürer.

***3-) V Modeli***: Şelale Modelinin doğrulama ve onaylama aşamaları eklenmiş hali diyebiliriz. Şelale modelinde kodlama ve test aşamasında test yapılmaktadır. Fakat burada aşamaların tekrar tekrar test edilip ileride çıkacak sorunları ortadan kaldırmaktır.  
 **Özellikler:**  
\*Gereksinimler başlangıçta belirlenir.  
 \*Maliyeti yüksektir.   
\* Başarı garantisi orta düzeydedir.   
\* Uzmanlık gerekliliği orta düzeydedir.   
\*Aşamalar çakışmaz.   
\* Basitlik derecesi ortadır.  
\*Değişiklik yapmak zordur.   
\*Yeniden kullanma oranı düşüktür.   
\* Uygulamak kolaydır.  
\*Uzun sürer.  
 ***4-)Spiral Model:*** Bu model büyük ve uzun projelerde planlama ve gereksinim analizi için çok kolaylık sağlar. Bu modelin amacı gereksinim analizi planlama aşamasında ortaya çıkan riskleri ve sorunları aşama aşama düzeltmeye yöneliktir.   
**Özellikler:**  
\*Gereksinimler belirli aralıklarla belirlenir.   
\* Maliyeti yüksektir.  
\* Başarı garantisi yüksektir.   
\* Uzmanlık gerekliliği yüksektir.   
\*Aşamalar örtüşmez.   
\* Karmaşıktır.   
\* Bakım yapılır.   
\*Değişiklik yapmak kolaydır.  
\*Yeniden kullanılabilir.   
\*Uygulamak karmaşıktır.   
\*Uzun sürer.  
***5-)Artımsal Geliştirme Süreç Modeli:***   
**Özellikler:**  
\*Gereksinimler belirli aralıklarla belirlenir.  
\*Maliyeti düşüktür.  
\*Başarı garantisi yüksektir.  
\*Uzmanlık gereklilik düzeyi ortadır.  
\*Aşamalar ötüşür   
\*Basitlik düzeyi orta derecededir.  
\*Bakım yapılır.   
\*Değişiklik yapılabilir.   
\*Yeniden kullanılabilir.   
\*Uygulamak kolaydır  
\*Uzun sürer.

Veriler karşılaştırıldığında hiçbir yazılım modeli birbirinden tamamen farklı değildir. Ya birbirinin geliştirilmiş hali ya da belli bir model temel alarak o modelin eksikliklerini tamamlamak amaçlı ortaya çıkan modellerdir. Proje için hangi modelin uygulanacağı belli kriterlere dikkat edilerek seçilmelidir. Projenin maliyetini teslim süresine ve projenin büyüklüğü ve karmaşıklığına göre ilerideki çıkacak sorunları ve eksiklikleri en az zararla atlatabilecek yani bakım yapılabilecek modeller tercih edilmelidir. Zaman, maliyet veya başka kayıpların yaşanmaması için süreç boyunca sürekli test edilip gereksinimler oluşan ihtiyaçlara göre belirlenirse projenin başarı oranı daha yükselir.

Yine de sektörde hala projeyi zamanında teslim edememe değişik istek ve gerekliliklere zamanında cevap verememek hataların tespitinde gecikmeler olması ve sistemin gereksinimlere göre kendini yeterince geliştirememe gibi sorunlardan dolayı farklı modeller ve yollar aranıyordu. Bu sorunlara karşı 90’lı yılların sonlarında **çevik(agile)** şeklinde adlandırılan metotlar geliştirildi.   
Bu metotların amacı:  
 \*Çok hızlı ürün oluşturabilmek.   
\*Değişsen isteklere olabildiğince hızlı yanıt verebilme.   
\*Projeyi olabildikçe kısa sürede kullanıma sunmak.

Çevik yazılımda hızı proje ekibinin sürekli irtibat halinde olmasından kaynaklanır. Ve proje küçük küçük paketlere ayrıldığı için geriye dönük hataları tespit etmek daha kolaylaşır.

***Çevik yazlımın prensipleri:***\*Teslimatı hızlı ve sık sık yaparak müşteri memnuniyeti sağlamak.   
\* Teslim sürecini hızlandırmak.  
\* Gereksinimlere sonradan yapılan eklenmeler kesinlikle dikkate alınır   
\*Yüz yüze iletişim önemlidir.  
\*Değişen koşullara adapte olabilmek önemlidir.  
**Avantajları:**\*Adaptasyon sağlamak kolaydır.   
\*Kısa döngüler ekibin motivasyonunu artırır. Bu da verimi artırır.  
\*Sık sık çıktı oluşturulup müşteriden değerlendirmeler aldığından ihtiyaç ve istekleri sağlayabilmek daha da kolaylaşır.  
\*Ekip çalışması vardır.  
**Dezavantajlar:**\*Kurumsal yapıda uygulaması zordur .  
\*Sürekli değişen ve artan ihtiyaç ve istekler sebebiyle aşırı çalışma.   
\*Ekip üzerinde hedefe ulaşma baskısı vardır.  
 **Özellikler:**\*Belirli sıklıklarla gereksinimler belirlenir.   
\*Maliyeti çok yüksektir.   
\*Başarı garantisi oldukça yüksektir.  
\*Uzmanlık gerektirme oranı çok yüksektir.   
\*Aşamalar örtüşür   
\*Karmaşık bir yapısı vardır.  
\*Bakım yapılır.   
\*Değişiklik yapılması zordur.   
\*Yeniden kullanılabilir.   
\*Uygulamak kolaydır.  
\*Kısa sürer.

<https://medium.com/@stuvsuzoglu/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCleri-c6de1177d916>

REFERANSLAR

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

<https://denizkilinc.com/yazilim-yasam-dongusu-temel-asamalari-software-development-life-cycle-core-processes/>

<https://medium.com/architectural-patterns/yaz%C4%B1l%C4%B1m-geli%C5%9Ftirme-modelleri-62915545c51e>

<https://github.com/esracaglak/Makale1>

<https://github.com/eserelci/Yaz-l-m-Ya-am-D-ng-s-Modelleri-Ve-Scrum>

<https://erhankiyak.medium.com/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-modelleri-5f509164e99c>

<https://medium.com/@ahmetonol/yaz%C4%B1l%C4%B1m-geli%C5%9Ftirme-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-sdlc-8b39d913986d>