**Yazılım Yaşam Döngü Modelleri**

**Yazılım Yaşam Döngüsü Nedir?**

Yazılım; günümüz dünyasının her alanında ve her pazarında büyük bir pay sahibi olan, gelişimini her geçen gün arttıran bir daldır. Üretilen somut ya da soyut fark etmeksizin her kavramda olduğu gibi belli başlı aşamaları bulunmaktadır. Bu aşamaların amacı talep edilen yazılımın, talep eden kişi ya da kurumun tüm isteklerini karşılamaktır. Bu döngü modelinde aşamalar doğrusal olarak değil, tekrarlanacak şekilde dairesel olarak ilerler. Aşamalar başlıca beş başlıktan oluşur.

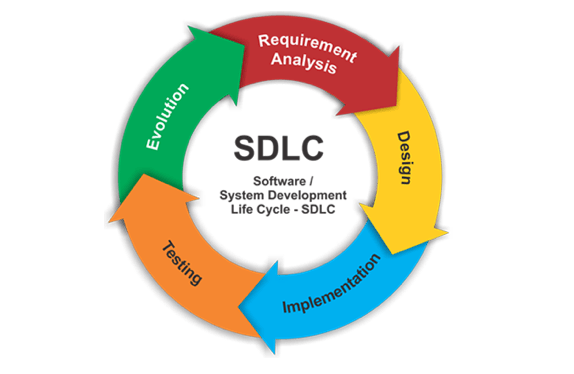
Planlama: Müşterinin istekleri ya da ihtiyaçları doğrultusunda atılan ilk adımdır. Yazılımın omurgasını oluşturur. Ekiple ya da bireysel olarak yol haritası belirlenir. Fizibilite çalışmalarının yapıldığı aşamadır.

Analiz: Planlamadan sonra gelen bölümdür. Proje sürecinin alacağı zaman ve gereksinimler en ince detaylarına kadar incelenir. Karşılaşılabilecek sorunlarla ilgili ön hazırlık yapılır.

Tasarım: Analiz aşaması sonrasında edinilen bilgiler doğrultusunda yapılan tasarlama işlemidir. Üst seviye ve Mimari Tasarım, Detaylı Tasarım olarak ikiye ayrılır.

Gerçekleştirme: İlk üç aşamada planlanan programın uygulandığı kısımdır. Kodlama ve test süreci bu aşamada başlar.

Bakım: Diğer aşamaların aksine bu süreç projenin tesliminden sonra başlar. Alınan geri bildirimler ile birlikte yapılan güncellemelerdir. Üç adım şeklinde gerçekleşir. Bunlar; hata düzeltme, yükseltme ve geliştirme adımlarıdır.



**Modellerin Açıklanması**

Birçok Yazılım Yaşam Döngü Modeli bulunmaktadır. Fakat bunlar arasından birini en iyi olarak adlandırmak doğru olmaz. Yazılım projenizin boyutu, kullanıcı kitlesi gibi değişkenlerine bakarak size uygun modeli tercih edebilirsiniz.

1. **Gelişigüzel Model**

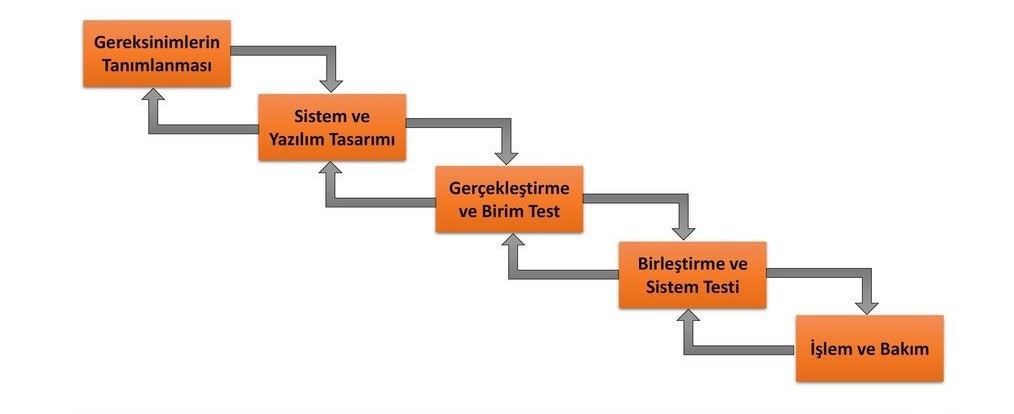
Tam anlamıyla bir “Yazılım Yaşam Döngü Model” i olarak adlandırılması doğru olmaz. Bunun sebebi ise günümüzde kullanılmamasıdır. 1960’lı yıllarda popülerliği olan bir yöntemdir. Bireysellik temel esastır. Kişiye bağlı bir geliştirme yöntemi olduğu için takibi zordur.

1. **Barok Modeli**

Yazılım yaşam döngüsünü Gelişigüzel Model’e göre daha belirgin kullanmaktadır. Ancak bir döngü şeklinde değil, doğrusal bir düzlemde ilerlemektedir. Yeni modellere kıyasla dokümantasyonu ayrı bir başlık altında incelemektedir. 1970’li yıllarda kullanılmıştır. Günümüzde kullanılmadığı gözlemlenmektedir.

1. **Çağlayan Yaşam Döngü Modeli**

Geçtiğimiz iki modelin aksine, hala kullanımına rastlanan bir modeldir. Uzun süreli kullanımı ve sağlamlığı sayesinde nadir de olsa günümüzde hala kullanılmaktadır. Modelin sağlamlığının temelinde her aşamanın en az birer kere tekrarlanması etkili olmuştur. Bu tekrarlama işlemleri modelin ve sürecin sağlamlığını arttırmıştır. En aktif kullanım alanı olarak, az zaman gerektiren ya da sınırlı sürede yapılacak projeler söylenebilir. Fakat aşamaların tekrarlı olması günümüzdeki kullanımını negatif yönde de etkilemiştir. Kullanımının azalmasının bir sebebi adım adım test edilmesidir. Bir adım bitmeden diğer adıma geçilemez. Bu durum uzun süreli projelerde zaman kaybı olarak görülür.



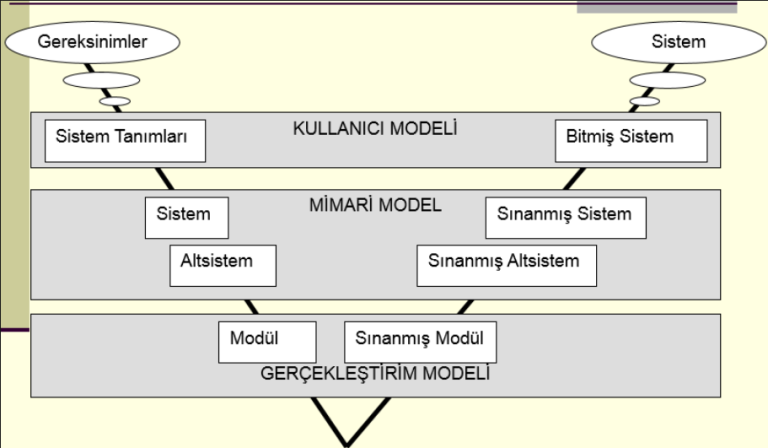
Garantici bir modeldir. Her aşamada dokümantasyonlar yazılmalıdır. Eğer bir diğer adıma geçmek istiyorsak mutlaka o an bulunulan adımın dokümanı hazırlanmalıdır. Barok modelde dokümantasyonlar ayrı bir başlıkken Çağlayan Model’inde sürecin bir parçasıdır. Bu modelin en önemli adımı analiz bölümüdür. Sistem gereksinimleri ve müşteri ihtiyaçları ön planda tutulmalı, hassasiyetle incelenmelidir. Analiz aşamasının ardından elde edilen bilgilerle tasarım aşamasına gelinmelidir. En çok uğraş gerektiren adımlar bu ikisidir. Modelin dezavantajlarından biri ise müşterinin sürece dahil olmamasıdır. Bu sebeple istek-talep ilişkisi tam olarak karşılanamayabilir. Bu da geri dönüşlerin artmasına ve maddi kayıplara yol açar.

1. **V Süreç Modeli**

Üretim(sağ), Sınama(sol) şeklinde ikiye ayrılmaktadır. Üç temel çıktısı bulunmaktadır:

* Kullanıcı Modeli: Kullanıcının istekleri doğrultusunda model tamamlanır. Ve bu şekilde kullanıcıya teslim edilir.
* Mimari Model: Projenin tasarımı ve deneme işlemleri bulunur.
* Gerçekleştirim Modeli: Kodlama ve sınama işlemleri bulunur

İş tanımlarının net olduğu, Bilgi Teknolojileri Sektörü için daha uygundur. Çağlayan Modeline göre kullanıcı daha etkin rol oynar bu da projenin verimliliğini arttırmaktadır.



1. **Helezonik (Spiral) Model**

Helezonik Model içerisinde prototip sistemini barındırır. Her adımın bir prototipinin olması da beraberinde risk analizini getirir. Kullanıcının aşama aşama projede yer alması amaçlanmaktadır. Böylece karşılaşılacak problemler ve maddi kaygılar en aza indirgenir. Başlıca dört adımdan oluşur.

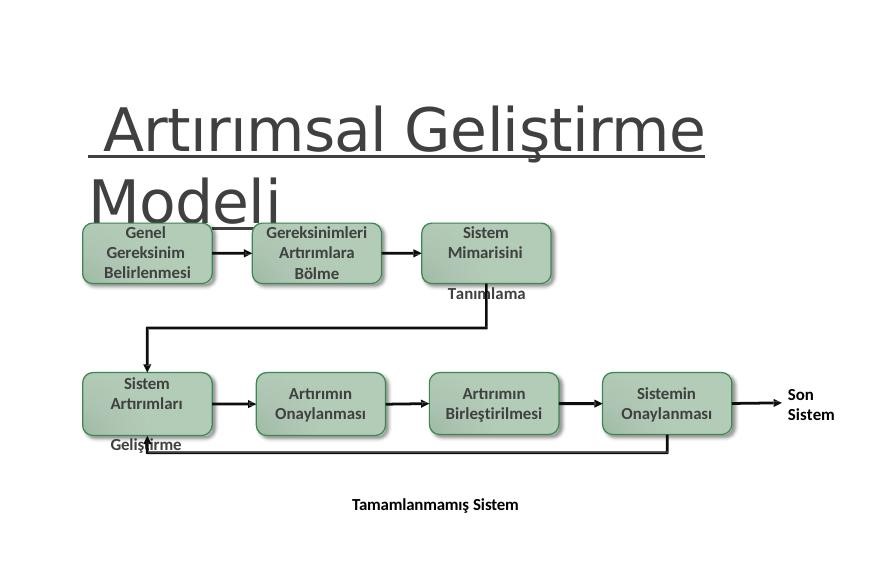
* Planlama: Her bir adımın prototipi için gereklidir. Bir amaç doğrultusunda yapılır. Aşılan her bir adım bir önceki ile birleştirilir.
* Risk Analizi: Risklerin belirlenmesi.
* Üretim: Her bir adımın proje olarak işlenmesi.
* Kullanıcı Değerlendirmesi: Ortaya koyulan prototiplerin kullanıcıya sunulması ve geri bildirim alınmasıdır.

Avantajları: Öne çıkan üç avantajı bulunmaktadır.

* Kullanıcı Katkısı: Kullanıcıların Projeye erkenden adapte olmasını sağlar. Erkenden alınan geri bildirimler ise üretici kişiye ya da kuruma süreç boyunca kolaylık sağlar.
* Yönetici Bakışı: Alınan geri bildirimler sonuncunda yönetici kişi ya da kurumlar proje üzerinden daha kolay hakimiyet sağlar.
* Yazılım Geliştirici – Mühendis Bakışı: Kodlama ve sınama işlemleri üzerindeki yetkinlikleri artar.

1. **Artımsal Geliştirme Süreç Modeli**

Sistem tek parça halinde değil, parça parça teslim edilir. Her teslim amaca bir adım daha yaklaşmak demektir. Kullanıcı gereksinimleri ön planda tutulur ve ihtiyaç halinde erken teslim edilebilir. Bir parça üretim sürecine girdiği zaman, o süreç içerisinde gereksinimler dikkate alınmaz. Parçanın üretimi bitirilir ve gerekli duyulan değişiklikler bir sonraki üretime dahil edilir. Üretilen sürümler bir çığ gibi düşünülebilir. Gelen her yeni sürüm bir öncekini kapsar ve giderek büyür. Bu modelde kullanım ve üretim birbirine bağlı kavramlardır. Aynı zaman diliminde gerçekleşir.



Erken teslimler aynı zamanda bir prototip işlevi görmektedir. Bu sebeple Helezonik model ile özdeşlik göstermektedir. Sürecin adım adım ilerlemesi ise projenin başarısız olma ihtimalini azaltmaktadır.

1. **Kodla ve Düzelt Yaşam Döngü Modeli**

Öncelikle yazılımın ilk sürümü ortaya koyulur. Bu döngü modelinde dokümantasyon olmadığı için bakım safhası zordur. Emeklilik safhasını içermektedir. Düz bir doğrultuda durmadan ilerlendiği ve özgür bir alan oluşturduğu için kolay ve tercih edilebilir bir seçenektir. Ancak üretici – kullanıcı ilişkisi göz ardı edildiği için alınan geri bildirimler maddi olarak oldukça pahalıya mâl olabilir. Kolay olduğu için birçok yazılım projesinde kullanılmaktadır.

1. **Evrimsel Geliştirme Modeli**

Bu model, tam ölçekli modellerin ilkidir. Modelin başarılı olabilmesi ilk evrimin başarılı ilerlemesine bağlıdır. Müşterinin gereksinimlerinin anlaşılmadığı durumlarda daha çok bu modele başvurulur. İzlenilebilirlik, yapılandırma, sürüm ve değişiklik yönetiminde sorunlar sık görülür.

1. **Big Bang Modeli**

Big Bang Modeli, diğer modellerin aksine az planlama ile gereksinimleri yeri geldikçe karşılayan bir modeldir. Planlama olmadığından dolayı gereksinimlerin tam bilinmediği ve teslim tarihi konusunda esnek olan projeler için daha uygundur. Kontrol daha çok program geliştiricisinde olduğu için karmaşık projelerde kullanılması riskli olabilir.

1. **Prototipleme Modeli**

Modelin iskeleti prototipler üzerine kuruludur. Sürecin başından sonuna kadar, her bir adımda prototip oluşturulabilir. Aynı zamanda oluşturulan bu prototipler kullanıcıya iletilir. Kullanıcının sürecin içinde yer alması, olası negatif geri bildirimleri ve maddi kayıpları önlemektedir. Ancak sürekli olarak prototip üzerine ilerleyen bu döngünün yetersiz ihtiyaç analizi gibi dezavantajları da bulunmaktadır.

1. **Çevik Yazılım Geliştirme Modeli**

Hızlı üretim-tüketim ilişkileri için oldukça uygundur. Piyasaya yönelik, hızlı ve maliyeti az işler ortaya koyulabilir. Verimliliği yüksek ve hata oranı düşüktür. Bu model kendi içerisinde temeli aynı ancak işleyişi farklı olan metodolojilere ayrılır. Bu metodolojide projenin boyutu önemsenmeksizin projenin parçalara ayrılan her bölümü ayrı bir proje gibi dikkatle incelenir. Ve bu ilerleme sonuçları devamlı olarak çalışan bir yazılım halinde müşteriye aktarılır. Müşterinin, sürece ve projenin ilerleyişine hâkim olması hata oranını azaltarak müşteri memnuniyetine katkı sağlar. Bu parçalama işlemi aynı zamanda bir hataya rastlandığında çözüm üretme konusunda da olumlu etkiye sahiptir. Bu modelde kullanılan bir diğer önemli yöntem de ekibin her bireyinin devamlı iletişim halinde olmasıdır. Bu prensipler projenin başarısında yüksek oranda katkı sağlar.

1. **Extreme Programming (XP) Modeli**

Bu model 4 temel değere dayanır. Bunlar; iletişim, basitlik, geri bildirim ve cesarettir. İletişim sorunları projenin ilerleyişini olumsuz yönde etkilemektedir. Bu olumsuzluğun engellenmesi için ekip üyelerinin yüz yüze iletişim kurması gerektiğini savunur. Kısa sürede geribildirim alınarak teslim süresinin azaltılması böylece sağlanır. Bir diğer temel değer olan basitlik konusuna gelirsek. XP modeli projenin anlık ihtiyaçları üzerinde yoğunlaşarak esnek bir sistem kurmaya çalışır. Zorlu ve karmaşık çözümler bu modele aykırı yöntemlerdir. Bu modelde müşteri de bir ekip üyesi olarak görülür. Projenin ilerleyişine belirli zaman dilimlerinde görüşmelere katılarak katkıda bulunur. Bu geribildirimler sayesinde hataların önüne geçilmiş olur. Bu temel değerlerin en zorlusu cesarettir. Bu model bizlere, eğer işten memnun kalmadıysak o projeyi bir kenara bırakıp tamamen sıfırdan başlamaya teşvik eder. Bu sayede içimize sinmeyen bir işle vakit kaybetmemiş oluruz.

XP modelinde basitliği ve esnekliği sağlamak için 12 pratik bulunur. Bunlar; planlama oyunu, ekipte müşteri, önce test, basit tasarım, çiftli programlama, sürekli entegrasyon, kısa aralıklı sürümler, yeniden yapılandırma, ortak kod sahiplenme, metafor, kodlama standardı ve haftada 40 saat yöntemleridir.

**SCRUM**

Scrum, büyük çaplı projelerde kodlamayı küçük parçalara bölerek adım adım süreci işler. Bu birimlerin her birine de “sprint” adı verilir. Genellikle karmaşık problemlerin çözümlerinde kullanılan bir yöntemdir. Günlük kısa süreli toplantılarla sürekli olarak ekip içi takip sağlanır. Üç temel kavramı bulunur.

* Roller: Ürün Sahibi: Projenin gelişiminden sorumludur. Ayrıca şirketin ekonomik çıkarlarını göz önünde bulundurarak tasarımı ve ana amaçları belirler.

Scrum Yöneticisi: Takımda disiplini sağlayan kişidir. Kuralları ve uyumu disipline ederek sürecin yolunda ilerlemesini sağlar.

Scrum Takımı: Projeyi birbirleri ile devamlı iletişim halinde olarak geliştirmek ile yükümlü 5-9 kişiden oluşur.

* Toplantılar: Sprint Planlama Toplantısı: Bu toplantıda gereksinimler belirlenerek parçalara ayrılmış sprint süreçlerinde yapılması gereken işler incelenir. Gereksinimlerin gözden geçirilmesi sonrasında ekip üyeleri gruplandırılarak uygun şekilde görev dağılımı tamamlanır. Risk değerlendirilmesi ve pazar maliyetlerinin araştırılması da bu toplantı sırasında gerçekleşir.

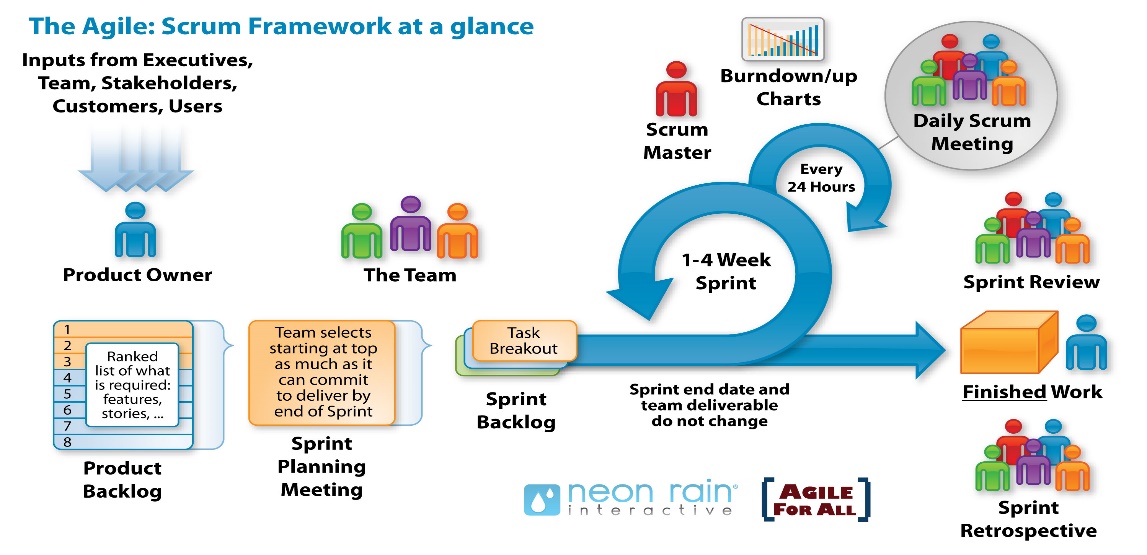
Sprint Gözden Geçirme Toplantısı: Her sprint sonrasında toplantılar yapılarak müşteri tarafından belirlenen gereksinimlerdeki ilerleme gözden geçirilir.

Günlük Scrum Toplantısı: Her iş gününde belirlenen saatlerde yapılan bu kısa toplantılar ekibin birbiri ile etkili iletişim halinde olmasını sağlar. Bu şekilde ekip üyeleri diğer üyelerin yaptıkları işlerden haberdar olur ve takımın ilerleyişinde kontrol sağlanır.

* Bileşenler: Ürün Gereksinim Dokümanı: Kullanıcının bakış açısından bakılarak hazırlanılmış, süreç boyu yapılması gereken iş elemanlarını içeren basit listedir.

Sprint Dokümanı: Mevcut sprint için backlog’dan elde edilen iş ve görev listesidir.

Sprint Kalan Zaman Grafiği: Yapılan ve yapılması gereken işleri gösteren grafiktir. Planlamaya uygun olarak ilerlenip ilerlenmediğini kontrol etmek için kullanılır.



**Scrum Günümüzde Neden Popüler?**

Yöntemin günümüzde hala popüler olmasının sebepleri; karmaşık gözüken projeleri bile parçalayarak basite indirgemesi ve her parçayı ayrı bir proje gibi incelemesidir. Bu yöntem ekiple ve kullanıcı ile sürekli etkileşim halinde olunmasıyla birlikte hata oranında büyük düşüş sağlamaktadır. Diğer metotların aksine yinelemeli olmayan bu yöntemde zamandan, dolayısıyla maliyetten kazanç sağlanır.

**Yazılım Yaşam Döngü Modellerinin Karşılaştırılması**

Gelişigüzel ve Barok Modeli dokümantasyonu sürece dahil etmeyen modellerdir. Döngü şeklinde olmamaları mevcut hataların tespitini engellediği için karmaşık projelerde sorun yaratmaktadır. Günümüzde de bu sebeplerden dolayı tercih edilmeyen modellerdir.

Çağlayan Modeli diğer iki modelin aksine hala kullanılsa da çok tercih edilen bir model değildir. Müşterinin sürece dahil olmaması ve her aşamanın tekrarlanması kapsamlı projelerde zaman kaybı ve isteklerin tam karşılanamamasına sebep olmaktadır.

V Süreç Modeli müşteri ile iletişimi içinde barındırır. Bu yönden Çağlayan modeline göre daha tercih edilebilir bir yöntemdir. Takibinin kolay olması ve onaylama işlemlerinin erken süreçte yapılması hataları önlese de risk çözümleme basamaklarının bulunmaması sorunlara yol açar.

Helezonik Model diğerlerinin aksine büyük projelerde kullanıma da uygundur. Bunun sebebi müşterinin ürünü aşama aşama görmesi ve bu sebeple hataların erken fark edilmesi olarak gösterilebilir. Fakat yöntemin karmaşıklığı ve dokümantasyonların fazla tutulması nedeniyle harcanan zaman gibi dezavantajları da bulunmaktadır.

Artımsal Geliştirme Modelinde projenin adım adım ilerlemesi ve parçalar halinde müşteriye teslimi hataların önüne geçmektedir. Müşterinin ön planda tutulması gibi avantajları bulunmaktadır. Fakat deneyim gerektiren bir yöntemdir.

Kodla ve Düzelt Modeli, Gelişigüzel ve Barok Modelleri ile benzerlikler göstermektedir. Dokümantasyon olmadığı için bakım aşamasında zorluklar bulunmaktadır. Ayrıca müşteri ilişkilerinin de göz ardı edilmesinden dolayı hatalarla karşılaşılma ihtimali yüksek bir modeldir. Bu durumlardan dolayı büyük projelerde kullanılamaz.

Çevik Yazılım Geliştirme Modelinde iletişim önem arz ettiğinden dolayı hata minimum seviyededir. Değişime açık ve esnek olmasından dolayı günümüzde çokça kullanılan bir modeldir.

XP Modelinde müşteri ile ekip sık sık iletişim halindedir. Bu sayede alıcının istekleri net bir şekilde anlaşılır ve katıldığı görüşmeler ile sürece dahil edilmiş olur. Aynı zamanda ekip de kendi arasında sürekli olarak görüşme halindedir. İletişimin ön planda tutulması ile hataların da önüne geçilmiş olur.

**Hangi Projede Hangi Modeli Kullanalım?**

Çağlayan (Şelale) Modeli; küçük ve isteklerin net olduğu projelerde kullanılabilir.

V Süreç Modeli; dokümantasyonların net olduğu bilgi teknoloji projeleri için uygun bir yöntemdir.

Evrimsel Geliştirme Modeli, büyük kitlelere ulaşacak, coğrafik olarak geniş alana yayılmış projeler için kullanılabilir.

Helezonik Model; büyük, maliyetli ve uzun süren projeler için uygun bir yöntemdir.

Artımsal Geliştirme Modeli; karmaşık ve büyük projeler için daha uygundur.

Kodla ve Düzelt; bireysel, teslim zamanı belirsiz projeler için daha uygundur.

Çevik Model; kısa süreli ve karmaşık olmayan projeler için tercih edilebilir.

**Kaynakça**

* <https://webdosya.csb.gov.tr/db/cbs/icerikler/salihsoylu_tez_v10-20180925134450.pdf>
* <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>
* [Yazılım yaşam döngü modelleri. YAZILIM GELİŞTİRME MODELLERİ VE… | by BATIN ÇETİN | Medium](https://medium.com/@batincetin44/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BC-modelleri-2342547d0840)
* <https://tr.wikipedia.org/wiki/Scrum>
* <https://www.linkedin.com/pulse/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-nedir-veysel-ugur-kizmaz/?originalSubdomain=tr>
* <https://iskulubu.com/yazilim/yazilim-gelistirme-yasam-dongusu/>

Medium Hesap: <https://medium.com/@ekinkaya20>

Linkedin Hesap: <https://www.linkedin.com/in/ekin-kaya-303277213/>

Github Hesap: <https://github.com/EkinKaya>