Yazılım Yaşam Döngü Modelleri

Yazılım ürünlerinin de tıpkı insanlar gibi bir hayatı vardır. Doğarlar, gelişirler ve nihayetinde ölürler (emekli olurlar). Bir yazılım ürününün doğmasından emekli olmasına kadar geçirdiği aşamaların tümüne Yazılım Yaşam Döngüsü (Software Development Life Cycle) denir. Yazılım yaşam döngüsünü gerçekleştirmek için birçok farklı model olsa da bu modellerin hepsi aynı amaca hizmet etmektedir: Müşterinin istediklerini tam olarak karşılayan kaliteli bir yazılım ürünü ortaya koymak. Bu amaç için uygulanan yazılım yaşam döngüsünün temel aşamalarına geçmeden önce belirtmeliyim ki yazılım yaşam döngüsü adından da anlaşılacağı üzere bir döngü biçiminde işler, tek yönlü bir süreç değildir. Şimdi yazılım geliştirmenin temel aşamalarını ele alalım.

* **Planlama:** Yazılım yaşam döngüsünün ilk aşamasındır. Müşteriyle gereksinim analizinin yapıldığı yani müşterinin yazılım ürününden beklediklerinin ilk kez belirlendiği adımdır. Gereksinimler belirlendikten sonra fizibilite çalışmaları yapılır.
* **Analiz:** Müşterinin isteklerinin tam olarak belirlendiği ve doküman haline getirildiği aşamadır. Bu işlem müşteri, yazılım mühendisi ve iş analisti gibi kişilerin oluşturduğu bir grupla yapılır. Müşterinin istedikleri eksiksiz bir şekilde belirlenmelidir ve yanlış anlaşılmalar ortadan kaldırılmalıdır. Eğer mevcut bir yazılım ürünü varsa incelenmelidir ve hataları belirlenmelidir. UML ve use-case diyagramları bu aşamada oluşturulmaya başlanır.
* **Tasarım:** Analiz aşamasında belirlenen gereksinimler doğrultusunda yapılacak yazılım ürününün tasarımı yapılır. Yazılım ürününün bileşenlerinin, arayüzünün ve fonksiyonlarının tasarlandığı aşamadır. Bir sonraki aşama olan gerçekleştirme aşamasına hiçbir soru işareti bırakmadan geçilmelidir. Tasarımı iki farklı şekilde ele alabiliriz. Bunlar ‘Mimari Tasarım’ ve ‘Detaylı Tasarım’dır. Mimari tasarım aşamasında önerilen sistemin yapısı, UML diyagram ve modüller yardımıyla doküman haline getirmeyi amaçlar. Detaylı tasarımdaysa, mimari tasarım sürecinde oluşturduğumuz dokümanlar revize edilerek yazılım ürününün bileşenlerinin detayları belirlenir.
* **Gerçekleştirme:** Tasarım aşamasında bütün hatları iyice belirlenen ürünün kodlanmaya başlandığı aşamadır. Kodlama yaparken dikkat etmemiz gereken kısım kodumuzun okunabilirliğinin yüksek olmasıdır. Karışık if blokları ya da uygun olamayan isimlendirmeler kodun tekrar düzenlenmesini zorlaştıracaktır. Yazdığımız kodu düzenlemek isteyen bir kişi zorlanmadan istediği düzenlemeleri yapabilmelidir. Okunabilir kod yazmak oldukça önemli bir husustur çünkü birazdan değineceğim test ve bakım aşamaları sonrasında çıkabilecek olası hataları düzeltirken bize oldukça fazla zaman kazandırır. Gerçekleştirme aşaması içinde yalnızca kodlamayı barındırmaz. Kodlarımızın testinin de yapılması gerekmektedir. Bir kodun testini ne kadar erken yaparsak oluşabilecek maliyetleri o ölçüde azaltırız (zaman, para, itibar vb.). Yazılan kod birçok farklı testten geçtikten sonra ve gerekli düzenlemeler sağlandıktan sonra artık kuruluma hazırdır.
* **Bakım:** Gerekli testlerden geçen ve kurulumu tamamlanan yazılım ürününün ömrünün sonuna kadar yani emekli oluncaya kadar içinde bulunduğu aşamadır. Yazılım farklı kullanıcılar tarafından kullanıldıkça ortaya yeni hatalar ve yeni istekler çıkacaktır. Bakım aşaması, ortaya çıkan hataların düzeltilmesi ve müşterinin yeni isteklerinin gerçekleştirilmesini kapsar. Bu işlemler yapılırken yazılım ürününün versiyon numarası değişir. Yazılım ürünü genelde piyasaya v1.0.0 olarak sürülür. İlk numaraya majör değer denir ve yapılacak köklü değişimleri temsil eder. İkinci numaraya minör değer denir ve çok fazla farklılık yaratmayan değişimlerle birlikte değişir. Üçüncü değerse yazılım ürünündeki küçük hataların düzeltilmesiyle değişir. Yazılım ürünü üzerinde yapılan değişiklikler sonucu piyasaya sürülecek yeni versiyon hakkında muhakkak versiyon fark dokümanı oluşturulmalıdır.

Yazılım yaşam döngüsünün temel adımlarını işlediğimize göre artık bu temel adımları gerçekleştirmek amacıyla kullanılan Yazılım Süreç Modellerini ele alabiliriz.

Yazılım Süreç Modelleri

Yazılım süreç modelleri, yazılım ürününü oluşturulurken rehberlik eden modellerdir. Başlıca yazılım süreç modellerimiz:

**1. Kodla ve Düzelt Modeli:** Yazılıma yeni başlamış kişilerin sıklıkla kullandığı modeldir. Küçük çaplı projelerde kullanıldığı için dokümantasyona ihtiyaç duymaz. Yazılım son haline gelene kadar kodlama yapılır ve yapılan kodlama üzerinde düzenlemeler yapılır. Ancak bir süre sonra yazdığımız kodlar spagetti koda dönüşeceği için üzerinde bakım yapmak ya da düzenlemek oldukça güç olacaktır. Bu durum ciddi maliyetlere yol açabilir.

**2. Gelişigüzel Modelleme:** Kodla ve düzelt modelinde olduğu gibi herhangi bir dokümantasyon aşaması yoktur. Genelde tek kişinin yapabileceği küçük çaplı projelerde uygulanır. Herhangi bir dokümantasyon aşaması olmadığı için kodda çıkacak olası hataları düzeltmek oldukça zordur.

**3. Barok Modeli:** 1970’li yıllarda hayatımıza girmiştir. Yukarda ele aldığımız yazılım yaşam döngüsünün temel aşamalarını doğrusal olarak işler. Aşamalar arasındaki geri dönüşler tanımlı değildir. Dokümantasyon ayrı bir aşama olarak işlenir. Kodlama ve test aşamaları bittikten sonra dokümantasyon aşaması gerçekleştirilir. Günümüzde artık kullanılmayan bir modeldir.

**4. Şelale Modeli:** En bilinen yazılım süreç modellerinden birisidir. Geleneksel model olarak da bilinir. Analiz, tasarım, kodlama, test ve entegrasyon aşamalarından oluşur. Şelale modeli dokümantasyon temelli bir modeldir. Her aşama sonunda doküman oluşturulur ve sonraki aşamaya geçilir. Bir sonraki aşamaya geçme şartı bulunulan aşamanın eksiksiz bir şekilde tamamlanmasıdır. Gereksinimleri çok iyi belirlenmiş projeler için kullanılır. Belirlenen gereksinimler doğrultusunda yazılım ürünü ortaya çıkar ve ancak bütün aşamalar tamamlandıktan sonra kullanıcılara sunulur. Eğer değişen gereksinimler varsa bu proje maliyetini bir hayli arttırır. Barok modelinin aksine önceki aşamalara dönüş tanımlanmıştır. İstenilen yazılım ürününün tamamlanması için modelin bütün aşamaları en az 1 kez tamamlanmalıdır. Proje geliştirilirken gereksinimlerde muhakkak değişiklikler olacaktır. Günümüzde popülaritesini kaybetmesinin kuşkusuz en büyük nedeni değişen gereksinimler karşısında bu modelin çaresiz kalmasıdır.

**5. V Modeli:** Modelin V şeklinde iki kolu vardır, adını da buradan alır. Sol taraf üretim, sağ tarafsa test aşamalarını barındırır. Tanımların çok net olduğu projelerde kullanılır. Temel çıktıları; kullanıcı modeli, mimari model ve gerçekleştirim modelidir. Kullanıcı modeli, kullanıcı ile yapılacak etkileşimleri belirtir. Yazılım ürününün nasıl kabul edileceğine dair planlar ortaya çıkar. Mimari model, yazılım ürününün tasarımını ve yapılacak olan test işlemlerini belirtir. Gerçekleştirim modeli ise yazılım ürününün kodlanmasını ve sınanmasını kapsar. V süreç modeli kullanıcının yazılım ürününe katkısını arttıran bir modeldir. V modelini uygulamak oldukça kolay olsa da risk çözümleme bakımından bir o kadar eksiktir.

**6. Prototipleme:** Gereksinimler belirlendikten sonra hızlı bir tasarım ve planlama aşaması sonucu ortaya model isminden de anlaşılacağı üzere bir ürün çıkarılır. Prototip müşteriye gösterilir ve geri bildirime göre prototip üzerine eklemeler yapılır. Müşterinin istediği son şekli alana kadar prototip düzeltilir. Somut bir şeyler görmek isteyen hızlı takımlar için uygun bir yöntemdir ancak herhangi bir dokümantasyon işlemi yapılmadığı için prototipler çok etkili olmayacaktır. Müşteriyse prototipleri ürünün son şeklinde görmek istediği için iletişimde sıkıntı yaşanabilir.

**7. Helezonik Model:** Helezonik ya da spiral model temel olarak dört aşama içerir. Bu aşamalar planlama, risk yönetimi, üretim ve kullanıcı değerlendirmeleridir. Helezonik model prototip ve dokümantasyon temelli bir modeldir. Genellikle önceden hazırlanmış olan yazılım ürünleri geliştirilir ve istenilen ürün ortaya çıkarılır. Daha çok yüksek bütçeli projelerde kullanılır, küçük projeler için fazla maliyetlidir ve modeli uygulamak profesyonellik gerektirir. Daha demin bahsettiğimiz temel aşamalara dönersek planlama aşaması, yapılacak prototipin tasarımını ve daha önce üretilmiş olan prototiple birleştirme işlemini kapsar. Risk yönetimi aşamasında olası riskler analiz edilip belirlenir. Üretim aşamasında, planlanmış prototip geliştirilir. Kullanıcı değerlendirme aşamasında, geliştirilmiş olan prototipler kullanıcılar tarafından test edilip değerlendirilir. Bu aşamalar istenilen yazılım ürünü ortaya çıkıncaya kadar tekrar edilir. Helezonik modelin en önemli özelliği kuşkusuz risk yönetiminin çok iyi olmasıdır.

**8. Evrimsel Geliştirme Modeli:** Geliştirilmiş ilk tam ölçekli modeldir. Yalnızca kilit gereksinimler ile yazılım ürünü geliştirilmeye başlanır. Müşterinin isteğine göre yazılım ürünü yavaş yavaş geliştirilir. Proje başarısı ilk evrimin başarısına bağlıdır. Evrimsel geliştirme modelini iki ayrı şekilde ele alabiliriz. İlki ‘Keşifçi Geliştirme’dir. Keşifçi geliştirme için iyi tanımlanmış gereksinimlere ihtiyaç duyulur ve yazılım ürününün son şeklini verene kadar müşteriyle birlikte çalışılır. İkincisi ise ‘Atılacak Prototipleme’dir. Gereksinimleri tam olarak belli olmayan projelerde kullanılır. Değerlendirmenin sürekliliği olası maliyetleri ve hataları azaltır. Kullanıcıların tam olarak ihtiyacı olduğu gereksinimleri ortaya çıkarır ancak bakımı zordur.

**9. Artımsal Geliştirme Modeli:** En kritik özelliklerin barındığı çekirdek sistemin oluşmasıyla başlar. Sonrasında üretilen bütün versiyonlar birbirini kapsayacak şekilde gelişir. Bunu matruşka bebeklerine benzetebiliriz. Her versiyon bir öncekini kapsar ve daha da geliştirir. Bütün işlevleri tam olmasa da çalışabilecek sistemler için kullanılır. Ürün kullanılırken aynı anda gelişim sağlanır. Çağlayan modeli ve evrimsel geliştirme modelinin arasında bir modeldir. Sistem gereksinimleri müşterilerle belirlenir. Böl ve yönet anlayışında devam ettiği için bütün sistemin başarısız olması pek muhtemel değildir. Profesyonel bir ekiple yönetilmesi gerekmektedir ancak buna rağmen her zaman artım boyutlarını belirlemek kolay olmaz.

**10. Araştırma Tabanlı Model:** Yap at prototipi olarak da bilinen araştırma tabanlı modelin araştırma ortamları tamamıyla belirsizlik içerir. Üretilen bir model birkaç kez kullanıldıktan sonra bir daha kullanılmamak üzere atılır. Sabit bir kullanım süresi olmadığı içi fiyatlandırması da sabit değildir, değişkenlik gösterir. Yazılım ürününden elde edilecek sonuç belli değildir buna örnek vermek gerekirse en büyük asal sayıyı bulan program örneğini verebiliriz.

**Çevik Modeller:** Buraya kadar anlattığımız modellerin çoğunda yani geleneksel yöntemlerde proje başarı oranı çok düşüktür. Bu durumun nedenleri: Hızla değişen gereksinimlere ayak uyduramamak, kullanıcıyla iletişimin çok az olması, yanlış proje yönteminin seçilmesi ve uygun proje ekibinin kurulmaması gibi durumları sıralayabiliriz. Bu sorunları aşmak için 90’lı yılların sonunda çevik modeller geliştirilmiştir. Çevik modellerin özelliklerini sıralamak gerekirse:

* Müşteri memnuniyetini sağlamak için hızlı ve etkili ürünler meydana getirir.
* Programlarda gereksiz özellikler bulundurmamaya özen gösterir, basitlik esastır.
* Müşterilerde çok sık iletişimde bulunulur.
* Küçük gruplardan oluşan ve kendi kendini organize edebilen takımlar vardır.
* Ürünler bir anda meydana gelmez artımlı bir süreç vardır.
* Değişen gereksinimlere ayak uydurabilir.
* Müşterilerin neye ihtiyacı olduğunu keşfettirir.
* Risk yönetimi iyidir, olası krizler önlenir.

Çevik modeller ve geleneksel modellerin başarı oranında çok büyük farklılıklar vardır. Aşağıdaki grafikte de görüldüğü üzere çevik modeller, geleneksel modellere göre çok daha etkilidir.

* **Extreme Programming (XP):** En yaygın çevik süreçlerden birisidir. Kent Beck ve arkadaşları tarafından 90’lı yılların sonlarında ortaya atılmıştır ve çok ses getirmiştir. Müşteri ve değişen gereksinimler sürecin merkezindedir. Tek seferde büyük dokümantasyonlar yapılmaz. Küçük parçalar halinde değişen gereksinimlere ayak uydurarak ilerler.

**XP Temelleri:** XP modelinde 4 temel değer vardır ve model bunlar üzerine kuruludur. Bunlar: Basitlik, geri bildirim, iletişim ve cesarettir. XP modelinin sağlıklı şekilde ilerleyebilmesi için bu dört temel değerin kesinlikle uygulanması gerekmektedir aksi taktirde XP modelinden söz edilemez. Şimdi temel adımları yakından inceleyelim.

1. **Basitlik:** XP modelinde basit çözümler bulmak son derece önemlidir. Basit çözümler projenin hızlı ve düşük maliyette ilerlemesini sağlar. Buna ek olarak basit çözümlerin düzenlenmesi oldukça kolaydır.
2. **Geri Bildirim:** Kalite kontrolü geri bildirim üzerinden sağlanır. Ortaya çıkan ürünler kullanıcılara sunulur ve geri bildirim alınır. Gereksinimlere göre düzenlenen yazılım ürünü tekrar olarak kullanıcıya sunulur ve tekrardan geri bildirim alınır. Kullanıcılar, yazılım ekibinin önemli bir parçasıdır. Kullanıcının geri bildirimlerine göre yazılım ürünü son şeklini alır. Bundan dolayı geri bildirim bu süreç için olmazsa olmazdır.
3. **İletişim:** Hem takım içinde hem kullanıcıyla iletişim kurmak hayati önem taşır. Yüz yüze iletişim kurulmalıdır. Yüz yüze kurulan iletişim sayesinde taraflar birbirini çok daha iyi anlar. Olası krizler önlenmiş olur. Eğer takım içindeki iletişim çok güçlüyse bazı dokümantasyon işlemlerine çok daha az zaman harcanır.
4. **Cesaret:** Dört temel değer arasında uygulanması en zor olanıdır. Takım üyeleri yazılım ürününün üzerine gidebilmeli ve gerekirse sil baştan başlamayı göze almalıdır. Farklı çözümlere yanaşmaktan ve başarısızlıktan korkmamalıdır.

**XP Pratikleri:** Yazılım geliştirme sürecinin daha başarılı olmasını sağlamak için 12 farklı pratik mevcuttur. 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’tir.

* **Planlama Oyunu:** Müşterinin gereksinimleri belirlemesine, yazılım ekibininse bu gereksinimlerin sürelerini belirlenmesine dayanan oyundur.
* **Ekipte Müşteri:** Gereksinimlerin hızlı ve tam istenildiği gibi anlaşılması için müşterinin yazılım ekibiyle aynı ortamda bulunması gerekir.
* **Önce Test:** Adından da anlaşılacağı üzere koddan önce test programının yazılmasıdır. Gelecekte çıkabilecek olası sorunları önceden çözmeyi sağlar.
* **Basit Tasarım:** Müşteri gereksinimleri en basit yöntemle çözülmelidir. Yazılımı değiştirmesi ve yönetmesi daha basit olur.
* **Çiftli Programlama:** Yazılım ekibindeki yazılımcıların ikişerli gruplar oluşturup yazılım ürününü geliştirmesidir. Geliştirilen yazılım ürünü farklı bakış açılarından incelendiği için olası hatalar önlenir. Usta-çırak ilişkisine benzetilebilir.
* **Sürekli Entegrasyon:** Geliştirilen ürünler mevcut ürünle hemen birleştirilip test edilir. Bu durum gelecekte oluşabilecek olası hataların önüne geçer.
* **Kısa Aralıklı Sürümler:** Parçalara ayrılmış projenin belirlenen kısmının birkaç hafta içinde bitirilip müşteriye teslim edilmesidir. Bu sayede müşteri yazılım ürününün gelişimini takip edebilmektedir.
* **Yeniden Yapılandırma:** Geliştirilen kod ve tasarımlar sürekli olarak daha da basitleştirilmeye çalışılır. Müşterinin gereksinimlerine göre tekrardan şekillendirilir.
* **Ortak Kod Sahiplenme:** Yazılan kod bütün ekibin ortak malıdır. Bireyselliğe yer vermez. Projenin gelişimi için başkasının yazdığı kod düzenlenebilir.
* **Metafor:** Yazılım ürününün parçalara ayrılarak her bir parçanın başka sistemlere benzetilmesidir.
* Kodlama Standardı: Yazılım ekibinin önceden belirlemiş olan kodlama standartlarına uygun kod yazmasıdır. Yazılım ekibindeki herkes, bir başkasının yazdığı kodu rahatça anlayabilmelidir.
* **Haftada 40 Saat:** Hafta içi her gün 8 saat olmak üzere haftada toplamda 40 saat çalışılmalıdır. Zorunda kalmadıkça fazla mesai yapılmamalıdır yapılacaksa da en fazla haftada bir gün yapılmalıdır. Aksi takdirde verim düşecektir.
* **SCRUM:** Adını rugby sporundaki bir hücum taktiğinden almıştır. 90’lı yılların ortalarında Jeff Sutjerland ve Ken Schawaber tarafından geliştirilmiştir ve yalnızca yazılım projelerinde değil bütün projelerde uygulanabilir. Scrum, karmaşık işlerin daha az insanla, daha hızlı bir şekilde, daha az maliyetlerle, çok daha kaliteli bir şekilde geliştirilmesi fikrine dayanır. Yapılan günlük toplantılarla projenin ne aşamada olduğu takip edilir ve olası sorunlar tespit edilir. Şimdi Scrum sürecine daha yakından bakalım.
* **Product Backlog:** Proje sahibi ürünün gereksinimlerinden oluşan bir liste oluşturur. Product Backlog bu listedeki gereksinimlerin önceliklerine göre sıralanmış halidir. Süreç içinde liste sürekli olarak güncel kalmalıdır.
* **Sprint Backlog:** Gerçekleştirilen toplantılarda, bir sonraki toplantıya kadar yapılması gereken işler belirlenir. Bu aralık genel olarak 15-30 gün arasıdır. Bir ayı geçmeyecek şekilde yapılması önerilir. Proje büyüklüğüne göre yapılacak Sprint’ler artar.
* **Scrum Daily Meeting:** Günlük olarak ayakta yapılan kısa (15 dakikayı geçmemesi önerilir.) toplantılardır. Bu toplantılarda üç temel soru sorulur ve eğer bir sorun varsa çözümü aranır. Bu sorular: “Dün ne yaptın?”, “Bugün ne yapacaksın?”, “Seni engelleyen neydi?”. Takımdaki herkesin bu sorulara cevap vermesi beklenir bu sayede takımdaki insanlar birbirinden haberdar olur.

Scrum, üç temel kavram içerir. Bunlar: roller, toplantılar ve bileşenlerdir.

1. **Roller:**
   1. **Ürün Sahibi (Product Owner):** Projenin iş değeri hakkında geri bildirim yapmalıdır. Yapılacak işle ilgili vizyon sahibi olmalıdır.
   2. **Scrum Yöneticisi (Scrum Master):** Scrum sürecinin sağlık bir şekilde ilerlemesini sağlar. Çıkabilecek olası problemlerin çözülmesinde yardımcı olur.
   3. **Scrum Takımı (Scrum Team):** Kendi kendine yetebilen, 5-9 kişiden oluşan takımlardır. Takım içindeki herkes tek bir amaca odaklanır ve bunun için çaba gösterir.
2. **Toplantılar:** 
   1. **Sprint Planlama (Sprint Planning):** Belirlenen gereksinimlerin küçük parçalara ayrılır ve bu parçalar için uygun takımların oluşturulur. Yapılacak risk kontrolleri belirleştirilir. Maliyet hesaplaması, geliştirme araçları ve altyapısının onaylanması gibi işlemler bu toplantılarda yapılır.
   2. **Sprint Gözden Geçirme (Sprint Review):** Her sprint sonunda yapılır. Ürün sahibi ve takımın, gereksinim listesini analiz etmesiyle başlar. Takım üyeleri gereksinim listesinden “Koşu Gereksinim Listesi” oluşturur.
   3. **Günlük Scrum Toplantısı (Daily Scrum):** Günlük olarak ayakta yapılan kısa toplantılardır (15 dakikaya geçmemelidir). “Dün ne yaptın?”, “Bugün ne yapacağım?”, “Bana engel olan bir durumlar ne?” sorularının cevabının verildiği toplantılardır. Bir problem varsa çözüm aranır.
3. **Bileşenler:** 
   1. **Ürün Gereksinim Dokümanı (Product Backlog):** Proje boyunca yapılacak olan işlerin listesidir. Kullanıcı hikayelerine (user stroy) göre sürekli olarak güncellenir. İhtiyaçlara göre gereksinim eklenir, çıkartılır ya da düzenlenir.
   2. **Sprint Dokümanı (Sprint Backlog):** Product Backlog’da belirlenen gereksinimlerden, sprint sonunda çalışabilir bir parça elde etmeyi amaçlar.
   3. **Sprint Kalan Zaman Grafiği (Burndown Chart):** Yapılması gereken toplam işlerin ne kadarının yapıldığını ne kadarının yapılmadığını gösteren grafiktir.

**SCRUM Günümüzde Neden Popüler?**

Google, Microsoft, IBM gibi yazılım devlerinden Riot Games’e kadar birçok şirket Scrum modelini tercih eder. Günümüzde en çok tercih edilen proje yönetim metodudur. Karmaşık projeleri parçalayarak daha az kişiyle, daha az maliyetle ve daha hızlı tamamlanmasını sağlar. İletişim ve takım çalışması Scrum’ın olmazsa olmazlarıdır. Müşteri merkezli bir yönetim biçimi olduğu için gelecekte yaşanabilecek olası krizeler engellenir ve müşterinin tam olarak istediği gereksinimlere göre proje ilerler. Bu gibi özellikleri sayesinde günümüzde Scrum oldukça çok tercih edilir.

**Kaynakça:**

* [**https://medium.com/@denizkilinc/yazılım-yaşam-döngüsü-temel-aşamaları-software-development-life-cycle-core-processes-197a4b503696**](https://medium.com/@denizkilinc/yazılım-yaşam-döngüsü-temel-aşamaları-software-development-life-cycle-core-processes-197a4b503696)
* [**https://caglartelef.com/yazilim-yasam-dongusu/**](https://caglartelef.com/yazilim-yasam-dongusu/)
* [**https://www.youtube.com/watch?v=6Ao75g13wjY**](https://www.youtube.com/watch?v=6Ao75g13wjY)
* **Doç. Dr. Deniz KILINÇ İzmir Bakırçay Üniversitesi 2. ve 3. hafta ders sunumları.**
* [**http://www.kurumsaljava.com/2008/11/21/extreme-programming-nedir/**](http://www.kurumsaljava.com/2008/11/21/extreme-programming-nedir/)
* [**http://web.firat.edu.tr/mbaykara/CevikYazilim.pdf**](http://web.firat.edu.tr/mbaykara/CevikYazilim.pdf)
* [**https://eratilmis.com/agile-ve-scrum-nedir/**](https://eratilmis.com/agile-ve-scrum-nedir/)
* [**https://medium.com/@secilcor/scrum-nedi̇r-6a4326951dd8**](https://medium.com/@secilcor/scrum-nedi̇r-6a4326951dd8)
* [**https://furkanalniak.com/yazilim-muhendisligi-yazilim-surec-modelleri/**](https://furkanalniak.com/yazilim-muhendisligi-yazilim-surec-modelleri/)
* [**http://ybsansiklopedi.com/wp-content/uploads/2015/08/Yazılım-Geliştirme-Modelleri-Yazılım-Yaşam-DöngüsüSDLCYBS.pdf**](http://ybsansiklopedi.com/wp-content/uploads/2015/08/Yazılım-Geliştirme-Modelleri-Yazılım-Yaşam-DöngüsüSDLCYBS.pdf)
* [**https://medium.com/yazılım-yaşam-döngüsü/yazılım-yaşam-döngüsü-b6f3f4da3146**](https://medium.com/yazılım-yaşam-döngüsü/yazılım-yaşam-döngüsü-b6f3f4da3146)
* [**https://medium.com/@burakguven351999/yazılım-yaşam-döngüsü-nedir-4abd49bf86ea**](https://medium.com/@burakguven351999/yazılım-yaşam-döngüsü-nedir-4abd49bf86ea)

**Hazırlayan: Dinçer Kaan Turanlı - 200601037**