Yazılım Yaşam Döngüler

Geliştirdiğimiz bir yazılım projesinin başlangıcından bitimine kadar takip ettiği adımlara Yazılım Yaşam Döngü Modeli (SDLC) denir. Yazılım Yaşam Döngü Modeli gereksinimlerin belirlenmesiyle başlar, teslim sonrası bakımla son bulur.

Yazılım dünyasında birçok yaşam döngü modeli bulunur. Bunlar: Çağlayan Model, Gelişigüzel Model, Spiral Model, V Model, Barok Model, Artımsal Geliştirme Süreç Model, Evrimsel Geliştirme Süreç Model, Kodla ve Düzelt Yaşam Model’dir.

Yazılım Yaşam Döngü Modeli belirli aşamalardan oluşur. Bunlar: gereksinimlerin belirlenmesi, analiz safhası, proje tasarımı, gerçekleştirim, teslim sonrası bakım ve emeklilik safhasıdır. Tabi bu adımların sayısı ve harcanan zamanı projeden projeye değişiklik gösterebilir.

Yazılım yaşamı sürekli gelişim ve değişim içinde olduğundan doğrusal olmaktan ziyade bir döngüdür.

**Gereksinim Safhası**

Bu safhada müşterinin projeden beklentileri belirlenir. Müşterinin ihtiyaçları üzerinde durulur. Proje planlanır.

**Analiz Safhası**

Analiz safhasında projenin işevleri belirlenir. Müşteriden alınan gerekli bilgilerle talepler belirlenir. “Yazılım ürününün amacı ne?” sorusuna yanıt aranır. Projenin tüm detayları netleşir. Bu yüzden yazılım yaşam döngü sürecinin en önemli adımlarındandır.

**Tasarım Safhası**

Bu safhada proje planı oluşturulur. Proje detayları gerekli durumlarda modüllere ayrılır, proje içinde yapılacak işlemler belirlenir ve proje detayları oluşturulur. Proje planının yanı sıra proje dokümanı da oluşturulmalıdır. Proje dokümanında projenin amacı ve kapsamı, sistem tasarım bilgileri, kullanıcı arayüz tasarımları, UML diyagramları yer alır. Ayrıca tasarım safhası boyunca iki tür tasarım yapılır: Üst Seviye ve Mimarı Tasarım, Ayrıntılı Tasarım.

Tasarım yazılımın amacı yazılım geliştiricilerinin yazılımı geliştirirken referans alacağı bir kaynağa ihtiyaç duymasıdır.

**Gerçekleştirim Safhası**

Gerçekleştirim safhasında artık analiz ve planlama bitmiş, uygulama kısmına geçilmiştir. Gerçekleştirim safhasında yazılımın ilk testleri gerçekleşir. Tasarım safhasında oluşan plan çerçevesinde ilerlenir. Kabul testi müşteri tarafından yapılır.

**Teslim Sonrası Bakım Safhası**

Proje müşteriye verildikten sonra oluşabilecek hataların giderilmesi, yazılıma yeni özelliklerin eklenmesi ve yazılımın iyileştirilmesi sürecidir. İki tür teslim sonrası bakım vardır:

**1)Düzeltici Bakım:** Sistemdeki hatalar düzeltilir.

**2)Özelliklerin Arttırılması:** Sözleşmede yapılan değişiklikleri yazılıma uyarlayan bakımdır. İki çeşidi vardır:

**a) Mükemmelleştirici Bakım:** Müşterinin düşünceleri doğrultusuna yazılımı yükseltmek.

**b) Uyarlanabilir Bakım:** Ürünü çalışma ortamına uyum sağlayabilecek şekilde geliştirmek.

**Emeklilik Safhası**

Emeklilik safhasında artık yazılımın kullanımı durdurulur. Çünkü yazılım artık zamana ayak uyduramıyordur ve daha yeni ve uyumlu bir yazılım gerekiyordur.

### Yazılım yaşam döngüsü aşamalarının her biri önemli olsa da bakım aşamasının yeri ayrıdır. Projeyi kapsamlı olarak inceler, geliştirir ve gerekli görürse bazı noktalarda değişiklik yapabilir. Ayrıca bakımın kritik noktalarından biri de hataları fark edebilmektir. Bir yazılımda hata ne kadar erken fark edilirse, o hatanın çözümü için harcanan zaman ve para da o kadar az olur. Yani bakım aşamasında bulunan bir hata, analiz aşamasında bulunan bir hatadan çok daha pahalıya m[âl](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjE-PHYrbv2AhXTgv0HHTC1B9kQFnoECBAQAQ&url=https%3A%2F%2Ftr.wikipedia.org%2Fwiki%2FK%25C3%25A2%25C4%259F%25C4%25B1t&usg=AOvVaw3wdbXQLChc6P_deErLauGf) olur.

**YAZILIM YAŞAM DÖNGÜ MODELLERİ**

**A) Gelişigüzel Model:** Bu model diğer modellerden farklı olarak daha özneldir. Belli bir kural veya yöntemden bahsedilemez. Yazılımcının kendi tarzıdır. Belli bir yöntemi olmadığından bakımı da diğer yöntemler göre daha zordur. Daha çok 1960’lı yıllarda kullanılmıştır.

**B) Barok Model:** Alışılanın aksine barok model yazılım yaşam döngüsünü doğrusal olarak ele alır. Süreçler arasında nasıl geri dönüşler olacağı konusunda bir açıklaması yoktur. Diğer döngülerden en önemli farkı dokümantasyonun proje bitiminden sonra yapılmasıdır. Daha çok 1970’li yıllarda kullanılmıştır.

Barok ve gelişigüzel modeller günümüzde kullanılmamaktadır.

**C) Çağlayan Model:** Bu model diğer güncel modellerin temelidir. Kısa soluklu projeler için uygundur. Her ne kadar günümüzde kullanımı azalsa da oldukça önemli bir modeldir. Çağlayan modelde her süreç en az bir kere tekrar edilir ve barok modelin aksine dokümantasyon proje bitiminde değil yazılım yaşam döngüsünün her bir safhasında yapılır. Çağlayan modelde gereksinim ve analiz safhası büyük önem taşıdığından bu safhalara uzun bir zaman harcanır. Proje başlangıcında müşterinin bütün ihtiyaçları ve istekleri alınır. Çağlayan model ileriki safhalarda yazılıma eklenti ve değişiklik yapmaya müsait bir model değildir. Çünkü maliyeti yüksek olur, projede çalışan ekip mutsuzlaşır ve verim düşer. Bu nedenle çağlayan model uzun süren projeler için uygun değildir.

**D) V Süreç Model:** V Süreç Model tablosunun sol kısmı üretim, sağ kısmı test işlemleridir. Tablonun kullanıcı model kısmında müşterinin proje için isteklerini ve projenin bitmiş halini oluşturur. Mimari model kısmı projenin tasarım ve sınanma kısmını oluşturur. Gerçekleştirim modeli ise projenin kodlanması ve bunun sınanmasını içerir. V model belirsizliklerin az, iş tanımlarının belli olduğu modeldir. Bilişim Teknolojileri projeleri için uygundur. V süreç modelin en büyük sorunu aşamalarda herhangi bir tekrarın bulunmaması ve risk alanları oluşturulmamasıdır.

**E) Helezonik (Spiral) Model:** Spiral Modeli diğer modellerden ayıran özellikleri prototip oluşturup risk analizini önemsemesidir. Spiral model küçük projeler için pahalı, karmaşık, uzun süren ve fazla dokümantasyondan oluşur. Ürünü tek bir seferde müşteriye teslim etmek yerine belli zamanda prototipler sunup aldığı geri dönüşler sayesinde üründe belli değişiklikler yapılır. Böylece hem müşteri projenin oluşum sürecinde projeyle iç içe olur hem de oluşabilecek maddi zararlar önlenir. Bu model 4 aşamadan oluşur:

Planlama: Her aşamada olan ara ürün(prototip) için planlama yapılır.

Risk Analizi: Oluşabilecek riskler araştırılır, bulunur ve çözülür.

Üretim: Prototip üretilir.

Kullanıcı Değerlendirmesi: Prototiple ilgili kullanıcıdan geri dönüşler alınır.

**F) Artımsal Geliştirme Süreç Model:** Sistem tek seferde teslim edilmek yerine parçalara bölünüp, ara ürünler oluşturularak teslim edilir. Böylece bir yandan üretim sürerken kullanım da sağlanarak kullanıcıdan geri dönüşler alınır. Uzun sürecek ve işlevsel olarak eksiklikleri olan projeler için uygundur. Bir parçanın geliştirlmesi başlandığında gereksinimler dondurulur. Üretilen her ara ürün birbirini kapsayacak ve giderek artan sayıda işlev içerecek şekilde geliştirilir. Bu modelin eksikliği ise bir parçanın kendi içinde tekrar edilmesine izin verilmez. Bu yüzden de bir ara ürün bitip diğeri başlayana kadar herhangi bir değişiklik yapılmaz.

**G) Kodla ve Düzelt Yaşam-Döngü Model:** Projeyi kodlar ve bitirirsiniz. Herhangi bir dokümantasyon ve ara ürün yoktur. Proje bittikten sonra müşteriye sunulur ve teslim sonrası bakım yapılır. Birçok hata bulunabileceğinden en maliyetli yaşam döngü modelidir. Yazılım geliştirmenin kolay yolu olduğundan genelde tecrübesiz firmalar kullanır. Kısa zamanlı projelerde kullanılır. Emeklilik safhası var.

**H) Evrimsel Geliştirme Süreç Model:** Bu model ilk tam ölçekli modeldir. Başarısı ilk evrimin başarısına bağımlıdır. Her aşamada üretilen her ürün tam işlevselliğe sahiptir. Büyük alanlara yayılmış büyük firmalar için önerilir.

Küçük ve özellikleri iyi tanımlanmış projelerde şelale modeli, fazla maliyetli ve uzun süren projelerde spiral model veya artımsal geliştirme model, belirsizliklerin az olduğu ve iş tanımlarının belli olduğu projelerde V model, büyük kitleler ulaşacak projelerde evrimsel geliştirme model, kişiye özel ve zaman sorunu olmayan projelerde kodla ve düzel modeli uygundur.

**ÇEVİK MODELLER**

Çevik modellerle projelerin piyasaya çok çabuk sunulması, değişimlere hızlı ayak uydurulması ve müşteri hizmetine çok çabuk sunulması amaçlanmaktadır. Çevik yazılım geliştirme metotları ile hızlı, verimliliği yüksek, esnek ve ucuz ürünler amaçlanır. Çevik geliştirme teknikleri, belgeleme yerine projeyle yani yazılıma yoğunlaşır. Teknoloji ve kullanıcı beklentilerinin sürekli değişimine cevap olarak doğmuştur. Değişen istekler yazılımın son aşamalarında bile kabul edilmesi, projenin temelinde motive olmuş bireylerin yer alması, çalışan yazılımın ilerlemenin birincil ölçül olması çevik modellerin prensipleri arasındadır.

**SCRUM**

Scrum projelerde dokümantasyondan ziyade yazılıma, müşteriyle pazarlık yerine müşteri ile iş birliğine, tek bir plana bağlı kalmaktan ziyade değişime ve yeniliğe açık olmaktır. Böylece değişen gereksinimler projenin sonraki safhalarında bile kabul edilebilir, projenin yapımı boyunca yüz yüze iletişim ve bilgi alış-verişi sayesinde yapılan hatalar az maliyetli oldu.

Scrum bir projeyi parçalara bölerek günümüz uzun zamanlı komplike projelerini daha yapılabilir hale getiriyor. Bu parçalara sprint adı verilir. Her scrumda ürün sahibi, scrum lideri ve 5-10 kişiden oluşan geliştirme takımı bulunur. Müşteriler ve satıcılar ise scrumda aktif bir role sahip değildir.

1. **Ürün Sahibi:** Müşteri ve scrum takımı arasında iletişimi sağlar. Projenin özelliklerini belirler. Sprintleri inceler ve gerekirse müşterinin ihtiyaçlarına göre sprinti değiştirir. Ayrıca sprinti iptal yetkisine sahiptir. Eğer bir sprintin işlevleri artık projede ihtiyaç duyulmuyorsa veya o sprintin işlevlerinde daha önemli işlevler ortaya çıkmışsa o sprint iptal edilebilir.
2. **Scrum Lideri:** Scrum teorilerini, kurallarını ve pratiklerini iyi bilir. Takımın bu kuraları uygulamasını sağlar. Takımı rahatsız eden, verimsiz çalışmaya sebep olan durumları engeller. Takımın yöneticisi değildir.
3. Geliştirme takımı: Yaklaşık 5-10 kişiden oluşur. Bir sprinte tanımlanan bütün işlevleri tamamlarlar. Sprint backlogu oluştururlar. İş kendilerine verilmez, kendileri işi üstlenir. Herkes her şeyi yapabilicek konumdadır.

**TOPLANTILAR**

1)Günlük Scrum Toplantısı: Ekip üyeleri tarafından her gün yapılan toplantıdır ve 15 dakika ile sınırlıdır. Her üye bu toplantıya katılmalı ve bu üç soruya yanıt aramalıdır: “Dün ne yaptım?” , ”Bugün ne yapacağım?”, “İşlemimiz için herhangi bir sorun var mı?”.

2)Sprint Planlama Toplantısı: Bu toplantı, gelecek sprintin tamamlanması için ürün gereksinim listesindeki işlerden seçilen seçilen sprint listesi hazırlanır. Bu toplantının hedefi sprintin amacını ve sprint iş listesini belirlemektir.

3)Sprint İnceleme Toplantısı: Bu toplantıda ürün içinde çalışan bir yazılım parçacığı sunulmaktadır. Toplantıda bir önceki sprint hakkında inceleme olmalıdır. Bu toplantının en önemli sorusu “Sprint hedefine ulaşıldı mı?” olmalıdır.

4)Sprint Gözden Geçirme Toplantısı: Bu toplantıda üyeler, önceki sprintleri tartışır ve bundan sonraki koşuda neyi daha iyi yapabileceklerini tartışırlar. Bu toplantı için etkili yollardan bir başlama-durma-toplantıya devam etme şeklindedir. Başlamada ekibin daha üretken, motive edilmiş ve işlevsel olmaya nasıl başlaması gerektiği, durmada yararlı olmayan bir şeyi yapma ve problem oluşması, toplantıya devam etme ise geliştirme ekibinin toplantıya verimli ve motive olmuş şekilde devam etmesidir.

Kısa sprintler ve sürekli geri bildirim nedeniyle değişimlerle kolay bir şekilde baş etmek, döngüsel olması, müşteriyle sık ve yüz yüze iletişim, prototipler, zaman ve paradan tasarruf, ekip üyelerinin üretkenliğini geliştirmesi, konuların önceden tanımlanması sayesinde hızlı bir şekilde çözülmesi scrum’ın avantajları arasındadır.

Ekip üyelerinden birinin proje geliştirme sürecinde projeden ayrılmasıyla projeye büyük bir ters etki yaratması, iyi çalışan küçük gruplar için uygun olması, test ekibinin her sprintten sonra test yapmaz ise proje kalite yönetimini ölçmek ve uygulamanın zorlaşması gibi dezavantajları vardır.

**EXTREME PROGRAMMING(XP)**

XP Kent Beck tarafından 1999 yılında ortaya atılan bir disiplindir. XP grup içi iletişime önem veren, pratik ve geri dönüşlerin daha fazla olmasına olanak sağlayan bir disiplindir. XP’nin 4 temel değeri vardır: İletişim, basitlik geri bildirim ve cesaret. İletişim proje çalışanları arasındaki bağlantısın yüz yüze ve sık olmasını temel alır. Basitlikte karmaşık çözümlere yer verilmez. Yapılması gereken zorunlu işlerdir. Esnek ve basit bir sistem geliştirmeye çalışır. Geri bildirim ise ortaya çıkabilecek problemler ortadan kaldırılır. Müşterilerle yazılım ekibi belli zaman aralıklarında buluşup, sistemi test ederler. Böylece sonradan doğabilecek anlaşmazlıklar belirlenip ortadan kaldırılır. Cesaret projenin üzerine pes etmeden gidilmesini temel alır. Yaptığınız şeyleri çöpe atmayı korkmamayı prensip alır.

XP’nin planlama oyunu, ekipte müşteri, önce test, basit tasarım, çiftli programlama, sürekli entegrasyon, kısa aralıklı sürümler, yeniden yapılandırma, ortak kod sahiplenme, metafor, kodlama standardı, haftada kırk saat olmak üzere 12 pratiği vardır. Planlama oyununda müşterinin belirlediği her bir yineleme için, yazılım ekibinin müşterinin de içinde bulunduğu toplantıda, o işin ne kadar zamanda yapılacağıyla ilgili kestirimde bulunmasıdır. Böylece müşteri bir sonraki yinelemede hangi işlerin yapılacağına karar verecek yazılım ekibi de bu işleri belirtilen sürelerde yinelemede yapacaktır. Ekipte müşteri ise müşteri temsilcisi yazılım gelişimi boyunca yazılım ekibi ile birlikte aynı ortamda bulunur. Bu sayede yazılım ihtiyaç duyduğu bilgilere kısa zamanda erişilir. Önce test yazılımın test edilip sorunların önceden belirlenmesidir. Basit tasarımda müşterilerin gereksinimlerini karşılayacak en basit tasarımın yapılmasıdır. Her bir yazılımcının sahip olduğu yetenekler ve bilgi birikimi sayesinde yazılımda ivme kazanılması çiftli programladır. Bu sayede proje hızlanır. Sürekli entegrasyonda yapılan sistem değişiklikleri ve yeni bileşenler hemen sisteme uygulanarak yazılımın güncel tutulmasıdır. Kısa aralıklı sürümlerde prototipler belli tarihlerde müşteriye teslim edilir. Böylece yazılım yaşayan bir uygulamaya dönüşür. Yeniden yapılandırmada kod sürekli gözden geçirilerek yazılım daha kolay bir biçimde yapılır ve “Müşteriden gelen istekler mevcut duruma nasıl uygulanır?” sorusuna yanıt aranır. Ortak kod sahiplenme geliştirilen yazılım kodunun bütün ekip üyelerinin ortak malı olmasıdır. Böylece yazılım bir başka ekip üyesi tarafından iyileştirilebilir. Gerçekleştirilecek yazılımda sistemlerin birbirine benzetilerek yazılımın geliştirilmeye çalışılması metafordur. Kodlama standardında ekip üyeleri önceden tanımlanmış kodlama standartlarına göre yazılım geliştirirler. Böylece yazılan kodun karmaşıklığı azaltılır. Haftada kırk saatte adından da anlaşılacağı üzere günlük sekiz saat olmak üzere haftada kırk saat mesai yapılır. Fazla mesai kavramı yoktur.

Boran AKÇA

210601704

**KAYNAKLAR**

[salihsoylu\_tez\_v10-20180925134450.pdf (csb.gov.tr)](https://webdosya.csb.gov.tr/db/cbs/icerikler/salihsoylu_tez_v10-20180925134450.pdf)

[Yazılım Yaşam Döngü Modelleri. Yazılım Yaşam Döngüsü Nedir? | by Ömer Harun Çetin | Medium](https://medium.com/@omerharuncetin/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BC-modelleri-543c7879a742)

[Agile ve Scrum Nedir? (mshowto.org)](https://www.mshowto.org/agile-ve-scrum-nedir.html)

[Scrum Nedir? (argenova.com.tr)](https://www.argenova.com.tr/scrum-nedir)

[Why Is Scrum So Popular? | TechWell](https://www.techwell.com/2013/02/why-scrum-so-popular)

[YAZILIM GELİŞTİRME VE SÜREÇ MODELLERİ | by Berfin Kaçar | Medium](https://medium.com/@brfn.kcr26/yazilim-geli%CC%87%C5%9Fti%CC%87rme-ve-s%C3%BCre%C3%A7-modelleri%CC%87-2131ea5f09b2)

**Profiller**

[Yazılım Yaşam Döngüler. Geliştirdiğimiz bir yazılım projesinin… | by Boran Akça | Mar, 2022 | Medium](https://medium.com/@borangf66/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCler-a849e1cbbb4f)