**YAZILIM YAŞAM DÖNGÜ MODELLERİ**

**Ayşe Nur DURMAZ**

**210601693**

**İzmir Bakırçay Üniversitesi**

**Mühendislik Fakültesi**

**Bilgisayar Mühendisliği**

**Seyrek, İzmir**

**E-Mail:** [**drmzay12@outlook.com**](mailto:drmzay12@outlook.com)

# YAZILIM YAŞAM DÖNGÜSÜ

Yazılım yaşam döngüsü (Software Development Life Cycle, SDLC), yazılımın her aşamasında gerçekleştirirken ve geliştirirken karşımıza çıkan tüm aşamalardır. Yazılım projeleri hakkında bilinmesi gereken temel bilgileri içerisinde bulundurur. Ürünün “iyi bir yazılım” olması için gerekli olan şartların sağlanmasını ve tüm proje için düzenli ve sağlıklı bir süreç olmasını sağlar. İyi bir yazılımdan bahsedecek olursak, yazılımın zamanla değişebilecek ihtiyaçlara göre esnek ve verimli olması, anlaşılabilir ve kullanışlı olması, kaynakların en verimli şekilde kullanılması, güvenilir ve başka sistemlere uyumlu olması ve tabii ki bunları en az maliyete yapması beklenmektedir. Yazılım yaşam döngüsü ise iyi bir yazılımdan beklenen bu özelliklerin proje sürecinde etkili ve düzenli bir şekilde uygulanmasında rol oynar.

Yaşam döngü modelinde teorik olarak uygulanacak tüm adımlar tanımlanır. Adımların niteliği ve sayısı ifade edilerek kullanılacak model belirlenir. Teoride yazılım geliştirme ise bir projeye en başından gereksinimler, analiz, tasarım ve gerçekleştirme adımlarının doğrusal olarak tanımlanmasıyla gerçekleşmektedir. Gereksinim aşamasında projede üzerine çalışılacak olan temel problemin ne olduğunu anlamaya çalışırız. Müşterinin istekleri ve ihtiyaçları arasındaki farkı göz önüne alarak gereksinimlerini belirleriz. Daha sonra belirlenen gereksinimleri bir araya getirerek analiz etme aşamasına geçilir. Bu aşamada müşterinin ne istediğinden çok neye ihtiyacı olduğu belirlenerek analizin sonucu, bir şartname dokümanı olarak yazılır. Analiz aşamasında belirlenen gereksinimlere göre “Yazılım Proje Yönetim Planı” da hazırlanarak neler yapılacağı detaylı yada kabaca belirtilerek hazırlanır. Tasarım aşamasında ise sistemin parçalarının ilk önce Mimari (Architectural) Tasarımı yapılır. Sistemi daha küçük olan parçalara bölerek büyük problemin çözümünü tasarlarız. Daha sonra Ayrıntılı Tasarım (Detailed Design) yapılarak yazılımın esas içeriği olan veritabanı tasarımı, kullanılacak algoritmalar, sınıflar, aktiviteler, etkileşim diyagramları vs. belirtilir. Yapılan tasarımlar olabildiğince detaylı olmalıdır. Her aşamada müşteri ile iletişim halinde olunmalıdır. Tasarıma kadar olan aşamalar, gerçekleştirme aşaması kadar önemlidir. Hataların ilk aşamalarda tespiti düzeltme maliyetinin çok büyümesini önler. Gerçekleştirme aşaması her bir modül için kod yazılan aşamadır. Yazılan kodların testleri yapılır. Modüller birleştirilerek hatalardan ayıklanarak çalıştırılır. Sistemin bütün halde çalışması gerekmektedir. Gerçek veri kullanarak yapılan testlerden geçen sistem, müşterinin yapacağı kabul testi (acceptance testing) ile tamamlanır. Projeler sistemin teslimi ile sonlanmaz. Sistem müşteriye teslim edildikten sonra “Teslim – Sonrası Bakım (Post-Delivery Maintenance )” ile iki çeşit olan bakım sürecine ihtiyaç duyar. Sistemdeki hataları düzeltmek için “Düzeltici bakım(Corrective Maintenance )” veya Sistemi mükemmelleştirmek ve uyarlanabilirliğini artırmak için “Özelliklerin Artırılması (Enhancement)” yani yazılımın güncellenmesi durumlarında, bakım süreç türleri kullanılır. Son olarak her girişim gibi yazılımların da bir vadesi vardır. Emeklilik safhasında (Retirement Phase ) vadesini tamamlamış yazılımların kullanımı durdurulur.

Bu aşamalar projelerin doğuşundan kullanımının durdurulmasına kadar her temel adımın özenle ve düzenle gerçekleştirilmesi için zemin niteliğindedir. Bunun için yazılım yaşam modeli döngü halindedir. Belirli aralıklarla güncelleme, bakım, sürdürebilirlik açılarından kendini tekrarlar. Yazılım geliştirme modelleri kullanım alanları, maliyetleri ve temel özellikleri ile birbirinden ayrılır.

1. Çağlayan / Şelale Modeli (Waterfall Model) : En temel modeldir. Geleneksel yazılım geliştirme modeli olarak da bilinir Yazılım geliştirme süreci analiz, tasarım, kodlama, test, sürüm ve bakım aşamalarından oluşur. Doğrusal olarak işler. Her bir aşamada bir önceki aşamanın sonuçlarından elde edilenler kullanılarak kendinden sonraki aşamada kullanılmak üzere değişikliğe uğrar. Bir sonraki aşama için önceki aşamaların titizlikle tamamlanması ve geçerli sonuçların elde edilmesi esastır . Kullanıcı ile iletişim ilk aşamalarda dahil olsa da tasarım ve kodlama aşamaları da dahil olmak üzere sonraki aşamalarda iletişim dahil edilmediği için hata riski artar. Hataların geç fark edilmesi ise hatanın onarım maliyetini büyülteceği için riskli bir modeldir. Bu sebepten günümüzde çok iyi tanımlanmış ve üretimi az zaman gerektiren projeler için uygun olmakla birlikte kullanımı giderek azalmaktadır. Temelde maliyeti düşük bir modeldir. Döngü halinde olmadığı için aşamaların tekrarından doğan maliyet den kurtulur.
2. Kodla ve Düzelt Yaşam-Döngü Modeli: Herhangi bir planlama yapılmadan ürün hazır olana kadar kodlama üzerinde çalışılarak devam eden küçük bir döngü şeklinde çalışır. Projede aşamalar arası geçiş hızlı olduğu için kontrollü değildir. Kontrollü olmaması ise uzmanlığa ihtiyaç durumunu azaltır. Deneme- yanılma yöntemi ile ilerler. Bu nedenle projenin bitiş süresi belirsiz olacağı için kodları düzeltmek maliyetli olabilir. Bunun için çok küçük projelerde, kısa ömürlü prototiplerde kullanımı uygun olabilir.
3. V Süreç Modeli: Doğrulama ve geçerleme (onaylama) modeli olarak da bilinir. Şelale modelinde olduğu gibi yazılım yaşam döngüsü adımları doğrusal ilerlerken V şeklini alır. Sol tarafına üretim , sağ tarafına test aşamaları yazılır. Bu modelde de bir aşama tamamlanmadan bir sonrakine geçilemez. En belirgin özelliği “ürünün test edilmesi kendisine karşılık gelen geliştirme aşamasına paralel olarak planlanmaktadır.” Planlama ve test tasarımı aşamaları kodlama aşamasına gelene kadar tamamlanması gereken aşamalarla paralel olarak gerçekleştirildiği için zamandan kazandırır. Şelale modelinin paralel eklentili hali gibi düşünüldüğünde şelale modeline göre daha başarılı bir modeli olarak düşünülebilir. Paralel aşamaları bulunduğu için kesin kurallara bağlı olarak uygulanır. Paralel olması, gereksinimler üzerinde bir değişiklik yapılmak istendiğinde test belgelerinin de diğer belgelerle birlikte güncellenmesi gerekeceğinden maliyeti artırması ise dezavantajıdır. İyi tanımlanmış sistemlerde Bilgi Teknoloji projeleri için uygundur.
4. Helezonik (Spiral) Model: Spiral şeklinde görselleştirilen bu modelde spiralin ilk çeyreğine planlama, ikinci çeyreğine risk analizi, üçüncü çeyreğine üretim ve son çeyreğine ise kullanıcı değerlendirmesi aşamaları yazılır. Dört aşamadan da geçerek ilerleyen iç içe döngülerden her biri bir fazı temsil eder. Yinelemeli ve artımsal bir yapıyı temsil etmektedir.

Planlama aşamasında başlayan döngü, projenin yapısına göre tekrar ederek her turu tamamladıktan sonra kullanıcı değerlendirme aşamasında son bulur. Planlama üretilecek ara ürün için işin planlaması, ihtiyaçların incelenmesi ve toplanması işlerinin gerçekleştirildiği aşamadır. Gereksinimleri kolaylaştırmak için açıklamalı incelemeler yapılır. Bu aşamada gereksinimleri anlama belgesi ve nihai gereksinim listesi oluşturulur. Bu aşama tamamlanınca ikinci çeyrek olan risk analizi aşamasına geçilir. Risk aşamasında potansiyel risklerin belirlenmesi için nihai gereksinim listesinin çıktıları incelenerek üzerinde çıkarımlar yapılır. Risk belirlendikten sonra riski azaltabilecek stratejiler planlanır ve sonuçlandırılır. Bu aşamada tüm riskleri ve azaltma planlarını anlatan belge çıkarılır. Üretim aşamasına geçmeden önce bu aşamanın titizlikle tamamlanmış olması gerekir. Üretim aşamasında gerçek geliştirme, kodlama ve test yapılır. Sistemin kodunun yanı sıra test senaryoları ve test sonuçları oluşturularak test özet raporu ve kusur raporu oluşturulur. Kusurlar üzerinde çalışıldıktan sonra son çeyrek olan kullanıcı değerlendirmesi aşamasına geçilir. Bu aşamada müşteriler yazılımı değerlendirir ve geri bildirimlerini bildirirler. Gerekirse ilk çeyreğe dönülerek sistem istenilen hale gelene kadar süreç tekrarlanır. Son değerlendirme aşamasında proje tamamlanmış olur.

Model birçok döngü oluşturulabilme potansiyelinde olduğu için büyük projelerde kullanılır. Yazılımların sürekli risk değerlendirmesine ihtiyaç duyduğu projelerde kullanılır. Gereksinimleri karmaşık olan projelerde sürekli açıklama getirilerek önemli değişikliklerin yapılmasını kolaylaştırır. Bu sebeple geliştirme hızlıdır. Dolayısıyla güncelleme ile yeni özellikler eklenmesini destekler. Belgeleme, ara işlemler yüzünden daha fazla olacağından küçük projeler için maliyetlidir. En önemli özelliği spiralin sonsuza kadar devam etmemesi gerekmesidir.

1. Artımsal Geliştirme Yaşam-Döngü Modeli: Artımsal modelin özü, projede değişiklik yapılacağı zaman yazılımı küçük parçalara bölerek parçaları geliştirdikten sonra değişikliğe ayak uydurmasını sağlamaktır. Diğer modellerden temel farkı artımsal model yazılım geliştirmenin kısıtlı sayıda çalışanla işin yapılmasını sağlama üstünlüğünün olmasıdır. Modelde yapısı gereği bir taraftan üretim bir taraftan da kullanım yapılır. Tüm döngülerin sonunda fonksiyonellik sağlanmış bir ürün ortaya çıkarılır.

Gereksinim aşaması tamamlanmış olan projede birden çok tasarım ve geliştirme aşaması oluşturularak parça parça bölünmüş sistemin her bir parçası için ayrı ayrı gereksinimleri artırımlara bölme ve sonrasında sistem mimarisini tanımlama aşamaları uygulanır. Gereksinimler müşteri ile birlikte oluşturulduktan sonra gereksinimlerin önemine göre teslim edilecek artımlar belirlenir. En önemli gereksinimleri sağlayacak çekirdek bir sistem oluşturulduktan sonra da erken artımlar prototip gibi davranarak gereksinimlerin daha iyi anlaşılmasını sağlar. Bu sayede projenin bütünüyle başarısız olma riskini azaltır. Böylece önem derecesine göre sistem özellikleri daha fazla test edilir. Bu süreçte dikkat edilmesi gereken gereksinimleri doğru boyuttaki artımlara atanması, artımları tanımlamak için tüm sistemin tanımlanmasına ihtiyaç duyulması, artımların kendi içerisinde tekrar etmesinin önlenmesi gibi birçok hassas nokta olduğundan model deneyimli personele ihtiyaç duyar. Böylece model, Divide and Conquer (Böl ve Yönet) yaklaşımını kullanarak sistemin değişikliklere uyum sağlamasını sağlamış olur.

Sistemler oluşturulurken zamanlama problemleri, yeni sürümler için değişiklik isteklerine cevap verilememesi, hataların geç fark edilmesi, zamanla sistemin yapısının eskimesi, projede üretilen fonksiyonların birçoğunun kullanılmaması gibi problemleri ortaya çıkmıştır. Bu problemlere çözüm olması amacıyla1990’lı yılların sonunda “Çevik (Agile)” adında metotlar oluşturulmuştur. Çevik yazılım geliştirme metotları kavramsal bir yazılım geliştirme metodolojisi olarak daha esnek, hata oranları düşük, hızlı ve maliyetleri düşük sistemler oluşturmayı amaçlamıştır.

Yöntem olarak küçük veya büyük projelerde proje küçük parçalara bölünerek her bir parça için küçük yinelemeler yapılır . her bir yenileme kendi başına proje gibi davranarak yinelemenin sonunda müşteriye bilgilendirme yapılır. Bu şekilde müşteri de 2-4 haftalık yineleme süreçlerinin sonunda dahil edilmiş olur. Bu yöntem ile müşterinin memnuniyeti artırılır. Küçük döngüler halinde devam eden süreçte hatalar erken fark edilerek maliyet açısından tasarruf edilmiş olunur. Müşterinin sürece sıklıkla dâhil edilmesi ile verimlilik artırılır ve esnek projelerin oluşumu kolaylaşır.

Çevik yazılım geliştirme metodu sadece proje için değil personel içinde kaliteli bilgi akışı, kısa periyotlu döngüler sayesinde motivasyonun düşmemesini, ekibin belirli aralıklarla kendi yöntemlerini gözden geçirmesini sağlar. Özetle üretkenlik artarken yazılım kalitesi yükselir ve maliyet düşer.

* + Extreme Programming (XP),
  + SCRUM,
  + Agile Unified Process,
  + Feature-Driven Development (FDD),
  + Test-Driven Development (TDD)
  + LEAN Development,
  + Dynamic System Development Methodology (DSDM),
  + Microsoft Solution Framework (MSF) olarak bilinen çevik metodolojiler vardır.

**SCRUM**

Çevik yazılım geliştirme metodolojileriyle uygulanabilecek bir proje yönetim yaklaşımıdır. Scrum, temelde karmaşık yazılım süreçlerinde kullanılan bütünü parçalayarak tekrara dayanan bir yöntemdir. İhtiyaca göre şekillenerek müşterinin geri dönüşlerine göre yapılanma sağlar. Şeffaflık, denetleme ve uyarlama prensipleri üzerine kurulmuştur. Burada şeffaflık, projenin sorunlarının ve gelişiminin herkes tarafından bilinebilir oluşudur. Denetleme de projenin ilerleyişinin kontrolünün sağlanmasıdır. Uyarlama ise zamanla projede yapılabilecek değişikliklere uyum sağlayabilir oluşudur.

Scrum yapısında kullanılan kavramlar ve detayları şu şekildedir:

* Product Backlog: Gereksinimler listesidir. Product owner tarafından alınan gereksinimler öncelik ve önem sırasına göre dizilir. Product owner değişen ihtiyaçlara göre gereksinim listesine ekleme çıkarma yapabilir. Böylece yapılacak değişiklikler projenin her aşamasına adapte edilir.
* Sprint(Koşma): Proje sprint denilen küçük parçalardan oluşturulur. 2-4 haftalık süreçler ile Scrum içerisindeki tüm aktiviteler sprint içerisinde gerçekleştirilir.
* Scrum Backlog: Geliştirme takımı tarafından product backlog itemlar (Product backlog da bulunan her bir gereksinim) önem ve öncelik sıralamasına göre listelendikten sonra işlerin detaylı zaman çizelgesi oluşturulur.
* Scrum Board: Oluşturulan sprintler içerisindeki yapılacak işler burada yönetilir. Yapılacak işler “TO DO” bölümüne alınır. Ekip üyesi bir işe başladığında “IN PROGRESS” bölümüne taşınır. Eğer iş test için hazır hale getirilmişse “TO VERIFY” durumuna getirilir. İş için kontroller bittikten sonra “DONE” bölümüne getirilir. Scrum toplantılarında bu maddeler duruma göre düzenlenir.
* Burndown Chart: X ekseni üzerinde sprint günlerinin, Y ekseninde toplamda bulunan işten kalan işi gösteren bir grafiktir. Başlangıçta belirtilen şeffaflık ilkesi gereğince herkesin ulaşabileceği bir bilgilendirme şablonudur.

Scrum sürecinde kişilerin aldıkları roller şu şekildedir:

* Product Owner: Ekip ile müşteri arasındaki iletişimi sağlayan, projenin özelliklerini tanımlayan kişidir. Proje için oluşturulacak gereksinimler listesini müşteriden alır ve değişikliklere göre listeyi düzenleyebilir. Projenin önceliklerine göre product backlogu oluşturur. Sprinte, ihtiyaç kalmaması gibi durum değişikliğine göre iptal etme yetkisine de sahiptir.
* Scrum Master: Takımın verimini kontrol eden, scrum’ un kurallarının uygulanmasından sorumlu kişidir.
* Geliştirme Takım: Bir sprintte bulunan tüm işleri gerçekleştirecek kişilerin bulunduğu ekiptir. Sprint backlog’u oluşturarak işin verilmesini beklemeden çapraz görev dağılımı ile işleri tamamlarlar. Projenin geliştirilmesi ile ilgili sorumluluklar bu 5-7 kişilik takıma aittir.
* Chicken Roller: Scrum işleyişinde aktif olarak yer almayan kişilerdir. Müşteriler, satıcılar vb.

Toplantılar hakkında bilinmesi gereken kavramlar şu şekildedir:

* Sprint Planning: Gereksinimler, bu toplantı ile geliştirme takımı tarafından küçük görevlere (task) ayrılarak takımdakiler arasında paylaştırılır. Bu toplantıda product owner, geliştirme takımı ve scrum master bulunur.
* Sprint Review: Her sprint başlangıcında yapılan planlama toplantısında ilk önce product owner ile takım, ürün gereksinim listesi üzerinde çalışarak gereksinim listesindeki elemanların hedefleri ve içeriklerinin belirlenmesini sağlarlar. Daha sonra ürün gereksinim listesinin önem sırasına göre sprint sonucunda oluşturulacak gereksinimlerden “Sprint Gereksinim Listesi” oluşturulur. Scrum için önemli bir uygulamadır. Takım, product owner tarafından belirlenen önem sırasına göre sıralanmış olan gereksinim listesini taahhüt eder.
* Daily Scrum: Scrum takımının günlük olarak yaptığı toplantılarıdır. Bir önceki gün neler yapıldığını, o gün neler yapılacağını, verimliliği artırabilecek unsurlar neler vb. konularında bilgilendirme yapılır.

**KAYNAKÇA**

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

Yücalar, F., (2013-2018), "**Yazılım Mühendisliğine Giriş Ders Notları**", Celal Bayar Üniversitesi.

<https://medium.com/@secilcor/scrum-nedi%CC%87r-6a4326951dd8>

<https://tr.wikipedia.org/wiki/Waterfall_model>

Tarhan, A., (2010), "**Yazılım Mühendisliği Ders Notları**", Hacettepe Üniversitesi.

<https://www.yazilimtestmerkezi.com/post/yazilim-gelistirme-modeli-nedir-cesitleri-nelerdir>