# Yazılım Yaşam Döngü Modelleri

Büyük ve karmaşık sistemler geliştiren şirket veya kurumların geliştirme süreçlerini sistematik bir şekilde belirleyip bu düzene uymaları ve yeri geldiğinde bunu geliştirmeleri gereklidir. Yazılım da aynı zamanda bir ürün olduğu için bu sistematik düzen belirlenmeli ve geliştirilmelidir. Bu modeller sayesinde planlı çalışmak kolaylaşacak ve daha kaliteli ürünler ortaya çıkacaktır. Şimdi bu yazılım yasam döngü modellerine bakacağız.

Teoride yazılım geliştirme; Gereksinimleri belirleme, analiz, tasarım ve gerçekleştirme adımlarının sırayla gerçekleştirilmesiyle oluşturulur.

Gereksinimleri belirleme; Müşterinin ihtiyaç ve isteklerinin belirlendiği adımdır.

Analiz; Müşteriden alınan ihtiyaç ve isteklerin ayrıntılı olarak ele alınıp incelendiği bölümdür.

Tasarım; Yazılımın temel yapısının gereksinimlere göre detaylı ve mimari olarak oluşturulmasıdır.

Gerçekleştirme; Yazılımın kodlandığı ve ürünün ortaya çıkarıldığı aşamadır.

Bakım; Hataların giderildiği aşamadır.

## 1.Gelişigüzel Model

Herhangi bir model veya yönteme bağlı kalmadan yapılır. Tamamıyla yapan kişiye bağlıdır. Bakımı oldukça zordur ve yaklaşık olarak 1960lı yıllarda ömrü dolmuştur. Bu nedenle tam bir modeldir diyemeyiz.

## 2. Barok Modeli

Her ne kadar belgeleme bugünlerde işin bir parçası olarak görülse de bu modelde belgeleme ayrı bir süreç olarak ele alınmıştır. Yaşam döngüsünün başlıca adımları bu modelde de uygulanmıştır. Gerçekleşme evresine çok fazla ağırlık veren bir model olduğu için kullanımı günümüzde tavsiye edilmemektedir.

## 3. Çağlayan Yaşam Modeli

Öncelikle gereksinimler belirlenir. Sonrasında sistem ve yazılım tasarımı gerçekleştirilir. Daha sonra ise kodlama ve test etme etapları gerçekleştirilir. Yapıyı birleştirme ve test etme yapıldıktan sonra da sistemin bakım ve yönetilmesi sağlanır. Dokümantasyon çok önemli bir yere sahiptir. Aynı zamanda şelale veya geleneksel model olarak da adlandırılır. Şelale yönteminde yazılım geliştirme süreci analiz, tasarım, ​​ kodlama, test, sürüm ve bakım​​ gibi aşamalardan oluşur. ​​ Geleneksel yazılım yöntemlerinde bu safhalar şelale​​ modelinde olduğu gibi işler. Her aşama, ​​ başlangıç noktasında bir önceki aşamanın ürettiklerini bulur. ​​ Kendi bünyesindeki değişikler doğrultusunda teslim​​ aldıklarını bir sonraki aşamanın kullanabileceği şekilde​​ değiştirir. ​​ Her ne kadar eskiden çok popüler olsa da günümüzde yavaş yavaş etkisini kaybetmektedir.

## 4.V Süreç Modeli

Bu modelde uygulanması gereken adımlar sağ ve sol olmak üzere 2 parçaya ayrılmıştır. Doğrusal bir yönde ilerlemek yerine, süreç adımları kodlama evresinden sonra yukarıya doğru eğim alır ve V şeklini oluşturur. V-Model geliştirme yaşam çevriminin her bir evresi arasındaki ilişkileri gösterir. Yatay ve dikey açılar zaman veya projenin tamamlanabilirliğini ve soyut seviyeyi gösterir. Bu adımlar bir süre sonra birleştiği için V modeli adını almıştır. Bu modelde 3 temel çıktı vardır. Bunlar sırasıyla “kullanıcı modeli”, “mimari model” ve “gerçekleştirim modelidir.

Kullanıcı Modeli; Geliştirme sürecinin kullanıcı ile olan ilişkilerini tanımlar.

Mimari Model; Sistemin tasarımı ve sistemin test edilmesiyle ilgili adımlardır.

Gerçekleştirim Modeli; Yazılımın kodlanması ve test edilmesi aşamasıdır.

## 5. Helezonik Model

Helezonik model aynı adımlara geri dönülmesinin bir zorunluluk olduğunu vurgular, şelale modelinde yok sayılan riskleri göz önünde bulundurur, Proje çevrimlere ayrılır ve her bir çevrimin riskleri ayrı ayrı ele alınır. Risk Analizi ön plana çıkmıştır. Direkt tanımlama, tasarım gibi bir evre yoktur. Artımsal bir yinelemeli yaklaşım vardır. Bu modelde Prototip yaklaşımı mevcuttur. Çağdaş modellere son derece yakındır. ​​ Helezonik modelde 4 temel kavram vardır. Bu kavramlar sırasıyla; “planlama”, “risk analizi”, “üretim”, “kullanıcı değerlendirme” dir.

Planlama; Üretilecek ara ürün için planlama, amaç belirleme, bir önceki adımda üretilen ara ürün ile bütünleştirme

Risk Analizi; Risk seçeneklerinin araştırılması ve risklerin belirlenmesi

Üretim; Ara ürünün üretilmesi

Kullanıcı Değerlendirmesi; Ara ürün ile ilgili olarak kullanıcı tarafından yapılan sınama ve değerlendirmeler

## 6.Artımsal Geliştirme Süreç Modeli

Sistem tek seferde değil parçalara bölünerek oluşturulur. Gereksinimler önemlidir ve önem sırasına göre sıralanırlar. Aynı zamanda hem kullanım hem de üretim yapılır. Yapılan bir aşama bir önceki aşamayı kapsar.

## 7. Kodla ve Düzelt Yaşam-Döngü Modeli

Genellikle yüzlerce satırlık çok uzun kodlarda kullanılır. Yazılım geliştirmenin en kolay yoludur. Bu yüzden küçük firma ve kuruluşlarda en çok kullanılan yöntemdir. Aynı zamanda çok maliyetli bir sistemdir. İlk olarak bir ürün ortaya çıkarılır. Daha sonra bu ürün sürekli olarak geliştirilir. Dokümantasyona çok önem verilmez. Bu sebeplerden dolayı sonradan ortaya çıkan gereklilikleri eklemek veya çıkarmak bu yöntemi daha maliyetli hale getirir.

## 8.Evrimsel Geliştirme Modeli

İlk tam ölçekli modeldir. Coğrafi olarak birbirinden uzak olarak bulunan birimlerin birlikte hareket edebilmesi için bulunmuş bir yöntemdir. İlk olarak bir ürün piyasaya sürülür. Daha sonra bu ürün test edilir ve gerekli güncelleme ve düzenlemeler yapılır. Bu ürünün kalitesi ve işlevselliği ilk ürüne bağlıdır. 2 tür evrimsel geliştirme süreci vardır.

1)Keşifçi Geliştirme; Müşterinin gereksinimlerini incelemek için müşteri ile çalışıp son sistemi teslim etmektir.

İyi anlaşılan gereksinimlerle başlanmalıdır.

2)Atılacak Prototipleme; Sistem gereksinimlerini anlamak için yapılır. Tam anlaşılamamış gereksinimlerle başlanır.

## 9.Prototipleme Modeli

İlk olarak gereksinimler toplanarak işe başlanır. Daha sonra Geliştiriciler ve kullanıcılar buluşarak yazılımdan elde edilecek bütün çıktılara, bu çıktılar için gerekli girdilerin nasıl sağlanacağına, nasıl korunacağına, hangi işlemlere uğrayacağına karar verirler.​​ Sonrasında ise hızlıca prototip bir tasarımla ortaya bir ürün çıkarılır. Kullanıcı bu prototipi bir müddet kullanır. Kullanımı sonra gereksinim ve isteklerine uygunluğuna göre ürünü değerlendirir. Değerlendirme sonucunun geliştiricilere aktarılmasıyla birlikte yazılımda gerekli değişiklikler yapılır. Kullanıcı bu yeni ürünü tekrar kullanır ve dener. Bu süreç kullanıcı ve geliştiriciler ortak bir paydada buluşuncaya kadar devam eder. Bu modele aynı zamanda doğrusal modelin döngüsel halidir de denilebilir.

## 10.Formal Sistem Geliştirme Modeli

Formal Sistem Geliştirme Modeli​​ yazılım tasarım ve gerçekleştirmesiyle ilgili matematiksel bir tekniktir. Bu modelin temelinde karmaşık sistemleri geliştirme ve program geliştirmeye destek yatar. Formal Sistem Geliştirme Metodu​​ kullanıcı sistemi kullanmaya başladığında karşısına çıkan belirtim hatalarını en aza indirir. Formal belirtim, tasarım ve geçerleme kullanarak yazılımda doğruluğun geliştirilmesini vurgular.​​ Yazılım artımlarla geliştirilir.​​ Amacı pahalı hata ayıklama işlemini engellemek için kodu ilk yazarken doğru yazmak ve test aşamasından doğruluğunu sağlamaktır.​​

## 11.Yeniden Kullanıma Yönelik Geliştirme Modeli

Farklı kurum veya şirketler daha önce üretilmiş olan yazılımların alınarak kullanılması ve gerekli görülen parçaların değiştirilmesi veya geliştirmesi yapılarak kullanılan bir modeldir. Son yıllarda popülaritesi gittikçe artmaya başlayan bir yaklaşımdır.

12.Birleşik Süreç Modeli

Nesne tabanlı programlama için daha önceki yöntemlerin iyi yönlerinin birleştirilmesiyle elde edilmiş bir modeldir. Yinelemeli, artımlı ve risk güdümlü olmak üzere 3’e ayrılır;

Yinelemeli; Gereksinimler bütün olarak değil parça olarak ele alınır. İlk olarak bir ürün ortaya çıkarılır. Sonrasında gereksinimler eklendikçe ürün geliştirilir.

Artımlı; Her yinelemenin ardından farklı ve yeni istekler ele alındığı için elde edilen ürünlerin özellikleri artarak gelişir ve istenilen ürüne daha fazla yaklaşılmış olunur.

Risk Güdümlü; Ürünün ilk sürümünde en önce en riskli bölümler ele alınmalıdır. Böylece en büyük sorunlarla projenin başında karşılaşılacağından çözmesi kolay ve düşük maliyetli olacaktır.

# SCRUM NEDİR?

Scrum 1990lı yıllarda bulunan bir proje yönetme aracıdır. Yalnızca yazılım değil diğer birçok alanda da kullanılabilen bir proje yönetme aracıdır. Yazılımda genellikle karmaşık, kompleks ve yönetimi zor olan projelerde tercih edilmektedir. Scrum modelinde iş bir bütün olarak ele alınmayıp, böl ve yönet mantığıyla küçük parçalara ayrılarak işlem yapılır. Böylece projenin hem yönetimi hem de kontrolü daha basit şekilde yapılabilir. Scrum aynı zamanda düzenli geri bildirim ve tekrar ister. Aynı zamanda scrum ekip çalışmasına önem verir ve ekip içi iletişim ve verimliliği arttırır. Ekip uyumu ve birlikteliği konusunda 3 temel kriter vardır. Bu kriterler sırasıyla; “şeffaflık”, “denetleme” ve “uyarlama” dır.

Şeffaflık; Projenin ilerleyişi, sorunlar, gelişmeler herkes tarafından görülebilir olmalıdır.

Denetleme; Projenin ilerleyişi düzenli olarak kontrol edilir.

Uyarlama; Proje, yapılabilecek değişikliklere uyum sağlayabilmelidir.

Şimdi biraz Scrum modelinin yapısını inceleyeceğiz. Scrum modelinin yapısını 3 Temel bileşene ve daha ayrıntılı olarak diğer alt bileşenlere ayırabiliriz. 3 temel bileşenden ilki olan Kavramlar;

1) Product Backlog; Proje için gerekli olan gereksinimler listesidir. Proje sonunda “Ne üretilmek isteniyor?” sorusuna cevap aranır. Müşteriden gereksinimler alınır, öncelik sırasına göre sıralanır. Product owner, değişen ihtiyaçlara göre product backlog’a ekleme veya çıkarma yapabilir. Böylece değişim, projenin her aşamasında projeye kolayca eklenebilir.

2) Sprint Backlog; Geliştirme takımı tarafından işlemler öncelik sırasına göre sprint içerisine alınırlar. Bir sprint boyunca yapılacak adımların listesini oluşturur.

3) Scrum Daily Meeting: Her gün Scrum takımıyla, (maksimum 30 dk. ve ayakta) düzenleyerek genel durum hakkında bilgi alınır.

Scrum metodundaki roller;

1)Product Owner (Ürün sahibi); Geliştirme takımı ve müşteri arasındaki iletişimi sağlar. Projenin özelliklerini tanımlar. Projenin önceliklerine göre product backlogu oluşturur. Sprint’i iptal yetkisine sahiptir.

2) Scrum Master (scrum yöneticisi); Scrum kurallarını, teorilerini ve pratiklerini iyi bilir ve takımın bu kurallarını uygulamasından sorumlu kişidir. Takımın yöneticisi değildir. Takımı rahatsız eden, verimli çalışmalarını engelleyen durumları ortadan kaldırır.

3) ScrumTakımı; Bir Sprint’e alınan bütün işleri tamamlayan kişilerdir. Kendi kendini yönetir. İşin verilmesini beklemezler, işi kendileri alır ve geliştirirler. Kişilerin tek bir görevi yoktur, çapraz görev dağılımı yaparlar, herkes her şeyi yapabilir konumdadır. 5–9 kişi arasında değişir.

Scrum metodunda toplantılar;

1)Sprint (koşu) Planlama; Product backlog ile belirtilen gereksinimler, bu toplantı ile scrum takımı tarafından küçük görevlere ayrılır. Takımdaki her bir kişi kendi hızına göre bu görevleri kendilerine alır. Bu toplantıya product owner, geliştirme takımı ve scrum master katılır. Risk değerlendirmesi ve kontrolleri yapılır.

2)Sprint Review (gözden geçirme); Her sprint’in sonunda yapılır. Yapılan sprint gözden geçirilir, ortaya çıkan ürün değerlendirilir. Amaç yazılımın ürün sahibinin gereksinimlerine uygun olarak geliştirildiğinden emin olmaktır. Eğer bir hata var ise fark edilir ve düzeltilir.

3) Daily Scrum; Her gün aynı yerde aynı saatte ayak üstü yapılan 15 dakikalık toplantılardır. Üyeler davet edilmeyi beklemezler. Takımdaki her üye dün ne yaptım, bugün ne yapacağım, işimi engelleyen herhangi bir sorun var mı sorularına cevap verir. Bu sayede herhangi bir sorunu var ise scrum master bu problemi ortadan kaldırır.

Scrum modelinde ekip çalışması, toplantılar, iş bölümü ne kadar ön planda olsa da en az onlar kadar ön planda olan bir diğer şey de dokümantasyon kısmıdır. Bunlara aynı zamanda scrum modelinin bileşenleri de diyebiliriz.

1)Ürün Gereksinim Dokümanı; Projede yapılacak işlerin, uygulanacak adımların ve gereksinimlerin projeden önce belirlenip listelendiği dokümandır. Eklenebilecek birçok yeni eleman için güncel tutulmalı ve bakım yapılmalıdır.

2)Sprint (koşu) Dokümanı; Sprint backlog’a cok benzemektedir. Daha doğrusu Sprint backlog tan alınan işleri kapsar.

3)Sprint Zaman Grafiği; Bu grafikte daha önceki adımlarda belirlenmiş olan iş bitiş süreleri ve sprint-zman ilişkileri bulunur. Belirtilen zamana kadar yapılması gereken işlerin ne kadarını yapılıp yapılmadığı buraya işlenir. Scrum takımındakiler belirli zaman aralıklarıyla bu verileri girmelidirler.

## Günümüzde Scrum

Scrum metodunun günümüzde daha popüler olduğunu ve en çok kullanılan metodlardan biri olduğundan yazımızın öncesinde bahsetmiştik. Bunun en büyük nedenlerin biri olarak scrum metodunun Agile(çevik) yazılım metodlarından biri olmasını gösterebiliriz. Günümüzde dünya her geçen gün dijitalleşmekte ve hemen her sektörde rekabet ön plana çıkmaktadır. Bu rekabet ortamında ayakta kalmak ve bir şeyler başarmak için çıkardığınız ürünün kalitesinin yanı sıra çıkış zamanı ve süresi de önemlidir. Pazarda rakiplerininizin önüne geçmek ve onlardan önce piyasaya atılmak bizler için daha iyi sonuçlar doğuracaktır. İşte bu ve bunun gibi nedenlerden dolayı günümüzde ürünün kalitesinin yanı sıra üretiminin de hızlı olması gerekmektedir. Firmalar bu hızlı üretimi yönetebilmek için Agile (çevik) ürün geliştirme yöntemlerini ortaya çıkarmışlardır. Tabii ki bizlerin üzerinde çalıştığı yazılım da bir ürün olduğu için yukarıdaki koşul ve şartları sağlamalı aynı zamanda çevik olmak mecburiyetindedir. Bu nedenle bizler de Çevik yazılım metodlarını ürünlerimizi ortaya çıkarırken kullanırız. Scrum metodu ise yine bu Agile (Çevik) metodlardan bir tanesidir. Bu nedenle günümüzde diğer yöntemlere daha çok ve sık kullanılmaktadır. Agile yönetim metodunun üzerinde çok durduk. Peki nedir bu Agile Metodu?

# Agile (Çevik) Yazılım Geliştirme

İlk kez 1990 li yıllarda ortaya atılan bir proje yönetim biçimidir. Yazılım ortaya çıkış süreci uzun ve zahmetli bir süreçti. Buna karşılık çıkan ürünler de müşterileri yeterince tatmin edemiyordu. Artan rekabet ve pazarda yar tutma isteği firmaları bu yöne itmiştir. Çevik yazılım geliştirme metotları ürünü piyasaya bir an önce çıkarmayı ve müşteri memnuniyetini artırmayı amaçlamıştır. Aynı zamanda zamanla değişen ve gelişen isteklere çabuk yanıt verebilen yazılım geliştirmeyi de amaçlamışlardır. Bu metotlar bizlere hızlı, ucuz, verimli yazılımlar çıkarabilmenin yollarını göstermiştir. Yazılım bir bütün olarak ele alınmaktan ziyade parçalara bölünmüş her bir parça tamamlanıp müşteri ile iletişime geçilmiş böylece yazılımın çevik olması sağlanmıştır. Ayrıca çevik yazılım geliştirmenin bazı prensipleri de belirlenmiştir.

• Müşteriyi memnun etmek ve sürekli yazılım geliştirmek önemlidir.

• Müşterinin rekabet avantajı için değişen gereksinimler benimsenmelidir.

• Teslimat, mümkün olan en kısa sürede yapılmalıdır.

• Geliştiriciler ve müşteri tüm proje boyunca birlikte çalışmalıdır.

• Yüz yüze iletişim, bir takıma bilgi aktarmanın en iyi yoludur.

• Çalışan yazılım, ilerlemenin birincil ölçümüdür.

• Teknik mükemmellik ve iyi tasarıma sürekli dikkat etmek çevikliği artıracaktır.

• Sadelik, yapılmayan işi en üst düzeye çıkarma sanatı olarak kabul edilir ve esastır.

• Kendi kendine organize ekipler genellikle en iyi tasarımları oluşturur.

• Düzenli aralıklarla, takımın nasıl daha etkili olacağına dair düşünülmeli ve davranışlar buna göre düzenlemelidir.

Burada da görüldüğü üzere çevik yazılımlar müşteri memnuniyeti sağlaması, maliyeti azaltması, üretim ve yazılım kalitesini arttırması vb. avantajları olduğu için günümüzde scrum gibi agile yazılım metodları çok fazla kullanılmaktadır.

Şakir UÇAR

190601026