# YAZILIM YAŞAM DÖNGÜSÜ MODELLERİ

**Batuhan SEYMAN**

**Öğrenci Numarası: 220601068**

**İZMİR BAKIRÇAY ÜNİVERSİTESİ/BİLGİSAYAR MÜHENDİSLİĞİ/1. SINIF**

## ÖZET

Bu yazı yazılım yaşam döngüsünün ne olduğu, bu döngünün temel aşamaları, yazılım yaşam döngüsü modellerinden olan çağlayan modeli, artımlı geliştirme süreci ve spiral modelin ne olduğu, çevik yazılım geliştirme metodunun ne olduğu, çevik yazılım geliştirme metotlarını temel alan bir proje yaklaşımı olan scrum hakkında bilgi vermek amacıyla yazılmıştır.

## YAZILIM YAŞAM DÖNGÜSÜ NEDİR?

Yazılım herhangi bir donanımda çalışan bilgisayar programı ve bunlarla ilişkili yapılandırma ve belgeler bütünüdür. Yazılım sadece bilgisayar programı olmaktan ziyade bir sorunu çözmek için tasarlanan bilgisayar programı, planlama ve belgelendirme gibi süreçleri içinde barındıran bir süreçten oluşur. Yazılım ürünü geliştirilirken gereksinimlerinin analizinden emekliliğine kadar uzanan süreçte takip edilen adımlar yazılım yaşam döngüsü olarak adlandırılır. Yazılım yaşam döngüsü doğrusal bir akış olarak düşünülmemelidir, bünyesinde iterasyonlar barındırır. Yaşam döngü modelindeki safhaların sayısı ve niteliği modelden modele değişiklik gösterir ancak genel olarak bu sürecin temel safhaları ve safhalarının içeriği şu şekildedir:

**1.Gereksinim Safhası:** Çözüme ulaşılması hedeflenen problemin ne olduğunun anlaşılmaya çalışıldığı, müşterinin ihtiyaç duyduğu gereksinimlerin toplandığı ve fizibilite çalışmasının yapıldığı aşamadır.

**2.Analiz Safhası:** Elde edilen gereksinimlerin analiz edildiği aşamadır. Analiz sürecinde üretilecek yazılım ürününün tüm işlevleri belirlenir ve projenin detayları ortaya çıkartılır.

**3.Tasarım Safhası:** Elde edilen gereksinimlere yanıt verecek sistemin temel yapısının oluşturulduğu, kodlanacak modüllerin oluşturulduğu, ne tarz algoritmaların kullanılacağı vb.nin belirlendiği aşamadır.

**4.Gerçekleştirme Safhası:** Oluşturulan modüllerin tek tek kodlandığı, birbirlerine entegre edildiği ve testlerinin yapıldığı aşamadır.

**5.Bakım Safhası:** Yazılım ürünü teslim edildikten sonra ortaya çıkan hataların düzeltildiği ve yeni gereksinimlere cevap verilen aşamadır.

## YAZILIM YAŞAM DÖNGÜSÜ MODELLERİ

Yazılımın düşük maliyetle, kısa sürede ve yüksek kalitede üretilmesi için birçok yaşam döngüsü modeli oluşturulmuştur. Bu modellerden bazıları:

## ÇAĞLAYAN/ŞELALE (WATERFALL) MODELİ

Bir projenin tüm safhalarında bir şelale gibi safhadan safhaya akar bu sebeple ismi şelale modelidir. Yazılım yaşam döngüsünün tanımı bu model baz alınarak yapılmaktadır, geleneksel geliştirme modeli olarak da bilinir. Her aşamasında dokümantasyon bulunduğu için dokümantasyon odaklı bir geliştirme modelidir. Yazılım geliştirme süreci gereksinim, analiz, tasarım, uygulama, bakım, emeklilik adımlarından ve bir önceki adımla bir sonraki adım arasında en az birer kez tekrarlarından oluşur.

## MODELDE İZLENEN PRENSİPLER

Bütün işler adım adım yapılır. Geliştirme süreci doğrusaldır, bu sebeple geliştirmenin bir safhası bitmeden diğer safhaya geçilmez. Her safhanın sonunda o safhayla ilgili doküman yazılır. Kullanıcı bu sürece yalnızca başlangıç ve bakım safhasında dahil olabilir.

## BU MODELİN AVANTAJLARI NELERDİR?

Her süreçte dokümantasyon olması sebebiyle iyi yönetilen bir projede sağlam bir temel oluşur. Gereksinimleri iyi tanımlanan ve üretimi kısa sürecek olan projelerde bu model kullanılabilir.

## BU MODELİN DEZAVANTAJLARI NELERDİR?

Eğer yazılımın gereksinimler iyi tanımlanmazsa oluşacak hataların düzeltilme maliyeti çok fazladır. Kullanıcı sürecin başı ve sonu hariç geliştirme sürecinde yer almaz ve kafasındaki son ürünü tam anlamıyla elde edemeyebilir. Her aşamada dokümantasyon olması sebebiyle geliştirilen projeler kullanıcılara çok geç ulaşabilir.

## SPİRAL (HELEZONİK) MODEL

Spiral model 4 ana adım ve bu adımların bir döngü gibi başa dönerek tekrar etmesini temel alır. Tekrarlar bir spiral şeklinde devam eder ve her spirale yazılım geliştirme sürecinin bir aşaması denir. Spiral sayısı projeden projeye göre değişir, kesin bir sayısı yoktur, sonsuza gidebilir. Diğer yazılım yaşam döngüsü modellerinden çok daha karmaşık bir yapıya sahiptir. Spiral sayısının her artımında proje son haline yaklaşmış olur. Bu model risk analizi olgusunu ön plana alır ve her bir spiralin riski ayrı ayrı ele alınır. Model büyük ve karmaşık projelerde kullanılabilir. Bir spiralin safhaları şu şekildedir:

**Planlama:** Projenin kapsamının belirlendiği ve spiralin bir sonraki tekrarı için plan oluşturulan aşamadır. Bu aşamada projenin amacı, alternatifleri ve tasarımın sınırlamaları belirlenir.

**Risk Analizi:** Bu aşamada projede uygulanacak yöntemlerin olası riskleri ve diğer alternatif yöntemler değerlendirilir, proje büyümeye devam ederken her bir spiralde üretim aşamasına geçmeden önce risk analizi yapılır. Ayrıca bir spiralin risk analizi aşamasında programların prototipi oluşturulur örneğin 1. Spiraldeki risk analizi aşamasında 1. prototip, 2. spiralde 2. prototip oluşturulur.

**Üretim:** İlk spiraldeki üretim aşamasında işin genel kavramı belirlenir, bir sonraki spiralde gereksinimler toplanır. Daha sonraki spirallerde ise ürünün tasarımı ve kodlama işlemleri başlar. Özetle projenin kodlama için mimarisinin oluşturulduğu, kodlandığı, her spiralin sonunda test ve onay işlemlerinin gerçekleştirildiği aşamadır.

**Kullanıcı Değerlendirme:** Her spiralin sonunda bulunan bu aşamada projenin gidişatı kullanıcı ve proje yöneticileri tarafından değerlendirilir ve bir sonraki faz planlanır.

## BU MODELİN AVANTAJLARI NELERDİR?

Kullanıcılar her fazın sonunda süreci değerlendirdiği için kullanıcı sistem hakkında erken aşamada fikir sahibi olabilir. Bu model geliştirmeyi parçalara böler ve projeyi geliştirmeye ilk olarak en riskli kısımlarından başlar. Bu yaklaşım mali zarar vb. risklerin meydana gelme ihtimalini azaltır. Model birden fazla geliştirme modelini bünyesinde barındırır ve bu durum geliştirme sürecine sorunların çözümü için daha esnek bir yaklaşım sağlar. Yazılımın kodlanması ve test süreci erken başladığı için hataları gidermeye odaklı bir yapısı da vardır.

## BU MODELİN DEZAVANTAJLARI NELERDİR?

Model karmaşık bir yapıya sahiptir ve spirallerin sayısı projeden projeye değişir hatta sonsuza gidebilir. Ara adım fazla olduğu için çok fazla dokümantasyon gerekir. Bu da zaman kaybına sebep olur. Projenin başarısının büyük bir kısmı risk analizine dayanır eğer risk analizi aşamaları başarılı bir şekilde gerçekleşmezse proje de hüsrana uğrar. Bu sebeple çok deneyimli uzmanlar gerektirir. Kullanıcılar ara ürün olan prototiplerde de son üründeki işlevselliği görmeyi umar. Geliştirme süreci pahalıdır, bu sebeple küçük projelerde kullanılamaz.

## ARTIMLI GELİŞTİRME SÜREÇ MODELİ

Bu modelde oluşturulacak sistem tek seferde teslim edilmek yerine her biri sistemde olması gereken işlevselliklerden bazılarına sahip olan küçük parçalara bölünerek ve her teslimde bu parçalara sistemin bir önceki yapısına yeni işlevler eklenerek teslim edilir. Bu yaklaşım bir bebeğin zamanla emeklemeyi, yürümeyi, konuşmayı, dik oturabilme yetilerini kazanması vb. şekilde zamanla belirli yetiler elde etmesi gibi düşünülebilir. Proje artımlar elde ederek son halini alır. Prototip yaklaşımı vardır. Her teslimle birlikte müşteriye görünen bir değer döndüğü için sistemin işlevselliği erken aşamalarda ortaya çıkar. Temel yaşam döngüsü adımları her artım sürecinde tekrar eder yani bu modelin bünyesinde iterasyonlar vardır. Model çağlayan modelinin iterasyon içeren üst modeli gibi de düşünülebilir çünkü temel adımlar sistem tüm işlevselliğini kazanana kadar sürekli tekrar eder. Spiral modele benzer yanları olsa da aralarındaki en büyük fark spiral model risk olgusunu temel alırken, artımlı geliştirme süreç modelinin prototipleri temel almasıdır. Bu model bir taraftan kullanımı, diğer taraftan da geliştirmeyi temel alır. Geliştirmesi uzun zaman alabilecek ve eksik işlevsellikle çalışabilecek büyük ve karmaşık projelerde kullanıma uygun bir modeldir.

## BU MODELİN AVANTAJLARI NELERDİR?

Müşteri ihtiyaçlarının değişmesi durumunda daha esnek ve rahat cevap verebilen bir yaklaşım sunar. Test etme ve hata ayıklama işlemleri diğer yöntemlere göre çok daha kolaydır çünkü geliştirmenin erken ve orta aşamalarında büyük bir sistem yerine küçük modüller test edilir. Böl ve yönet yaklaşımı vardır. Gereksinimlerin önemine göre teslim edilecek artımlar belirlenir, bu durum tüm projenin başarısız olma riskini azaltır çünkü tüm gereksinimlere cevap veremese bile en azından bazı gereksinimlere cevap veren bir ürün elde edilir. Artımlı yaklaşımı sebebiyle sistemin tüm bileşenleri tekrar tekrar test edilmiş olur ve hata payı azalır.

## BU MODELİN DEZAVANTAJLARI NELERDİR?

Artımları tanımlamak için bütün sistem tanımlanmalıdır. Gereksinimler doğru artımlara tanımlanmayabilir, bu durum kullanıcının önceliklerine uymaz. Deneyimli personele ihtiyaç vardır. Ürüne yeni işlevler eklendikçe önceki artımlarda oluşturulan prototiplerde sıkıntılar meydana gelebilir. Artımlı geliştirme süreci sürekli tekrarlanan bir süreçtir, her bir artım ayrı bir iş olarak değerlendirilir. Bu durum sürecin maliyetinin bir hayli artmasına sebep olabilir. Bu süreçte artımlar kendi içlerinde tekrar etmezler, bu durum hatalara sebep olabilir.

# ÇEVİK YAZILIM GELİŞTİRME

Bir önceki bölümlerde standart yazılım geliştirme modellerinden bahsettik. Bu kısımda çevik yazılımın ne olduğu hakkında bilgi verip çevik yazılım geliştirme metotlarını benimseyen bir proje yönetim yaklaşımı olan scrumdan bahsedeceğiz. İlk olarak çevik yazılım nedir? Çevik yazılım hızlı ürün çıkarabilme, değişen gereksinim ve isteklere hızla cevap verebilme, yazılım ürününü en kısa sürede müşteriyle buluşturma metotlarını amaçlayan bir yaklaşım türüdür. Çevik yazılım metotları verimi yüksek, hata oranı düşük, esnek bir yapıya sahip yazılım üretmek için çeşitli çözümler ortaya koyar. Standart modellerin birçoğunda gereksinimlerin tümünün toplanması projenin en başında yapılır ve gereksinim değişikliklerine kolayca cevap verilemez. Standart modellerin aksine çevik yazılım geliştirme metodu gereksinim değişikliklerine daha esnek bir cevap verebilme özelliğine sahiptir. Metotta projenin büyüklüğü gözetilmeksizin proje küçük iterasyonlara ayrılır ve her iterasyon sonunda müşteriye projenin işleyişi hakkında bilgi verilir. Bu sayede geliştirme sürecine müşteri de dahil edilmiş olur ve gereksinim değişiklikleri daha iyi analiz edilip daha hızlı cevap verilebilir. Bölünmüş her bir iterasyon 2-4 hafta arasında tamamlanır ve diğer bölüme geçilir.

## SCRUM

Scrum çevik yazılım geliştirme metotlarını ilke alan ve sadece yazılım geliştirmeye değil, uygun olan her projeye uygulanabilecek bir proje yönetim şeklidir. Yazılım geliştirme işlerini sprint adı verilen küçük parçalara bölerek çalışır. Günümüzde yazılım projeleri büyük ve karmaşık olduğundan, yazılım gereksinimleri eskiye kıyasla çok fazla ve çok daha hızlı değiştiğinden dolayı bu metot oluşturulmuş ve kullanımı çok fazla yaygınlaşmıştır. Google, Microsoft ve Yahoo! gibi büyük şirketler de dahil olmak üzere birçok şirket bu metodu kullanmaktadır. Scrum günümüz projelerinin karmaşıklığını azaltmak için 3 adet ilke benimsemiştir: “Şeffaflık, denetleme, uyarlama.” Şeffaflık Scrum’da projedeki ilerlemelerin kayıt tutulmasını, sürecin herkes tarafından izlenilebilir olmasını sağlayan ilkedir. Denetleme projenin ilerleyişinin düzenli olarak kontrol edilmesini sağlayan ilkesidir. Uyarlama ise projenin gereksinimlerinin en başta bir defa belirlenmesinden ziyade her teslimatta gereksinimlerin tekrar değerlendirilip o anki duruma göre uyarlamalar yapabilmeyi sağlayan ilkedir. Bu ilkelere dayanarak scrum, gereksinimlerin rahatlıkla tanımlanamadığı ve kaotik durumların beklendiği büyük ve karmaşık projeler için en uygun metottur.

## SCRUM NASIL ÇALIŞIR?

Burada öncelikle Scrumda üretim sürecine ait olan terimlerden bazılarının tanımlarından bahsedelim. Bu terimlerden yazının ilerleyen kısımlarında tekrar bahsedilecektir.
**Product Backlog:** Müşterinin gereksinimlerine göre önceliklendirilmiş ve sürekli olarak incelenen, gereksinim değişikliklerine uyacak şekilde tekrar önceliklendirilen bir yapılacaklar listesidir.
**Sprint Backlog:** Projenin o anki sprintindeki yapılacak işlerin bir listesidir. Bu listedeki işler 2-4 hafta zaman aralığını kapsar ve bir sprint bittiğinde diğer sprinte geçilir.
**Scrum Daily Meeting:** Projenin gidişatı hakkında görüşülen ve her gün 15-30 dk gibi kısa sürelerde biten toplantılardır.

Yapılacak işlerin tamamı sprint adı verilen küçük parçalara bölünür ve her bir sprint en fazla 30 günde bitirilir. Her sprintin sonunda yazılımda istenen fonksiyonel bir parça oluşturulmuş ve müşteriye teslim edilebilir bir durumda olur. Bir sprint bittikten sonra diğer sprinte geçilir ve bu durum proje bitene kadar devam eder. Her gün projenin gidişatının görüşüldüğü ve genelde sabahları olan toplantılar yapılır. Scrum takımında birden fazla rol bulunur. Bunlardan biri de müşteri tarafından görevlendirilen ürün sahibidir. Ürün sahibi müşteriye projenin gidişatıyla ilgili bilgiler verir, ondan geri dönüşler alır, yeni gereksinimler hakkında bilgiler toplar. Scrumın çalışma yapısını bu şekilde özetleyebiliriz. Scrum 3 temel bileşenden oluşur: "Roller, toplantılar, bileşenler."

## SCRUM'DAKİ ROLLER

**Ürün Sahibi (Product Owner):** Bir nevi müşterinin temsilcisi olan ve projenin müşteriye iş değeri açısından dönüşünü temin altına alan, müşteriyle scrum takımı arasında köprü görevi gören kişidir.
**Scrum Yöneticisi (Scrum Master):** Scrum ekibinin verimliliğini sağlarlar. Scrum süreçlerini iyileştirmek ve teslimatı optimize etmek için ekiplere, ürün sahiplerine ve işletmeye koçluk yaparlar. Her bir Sprint için gereken kaynakları planlama, diğer sprint etkinliklerini ve ekip toplantılarını kolaylaştırma, ekip dahilinde dijital dönüşüme öncülük etme gibi görevleri de vardır.
**Scrum Takımı (Scrum Team):** Birbirleri ile devamlı iletişim halinde bulunan ve projenin geliştirilmesini üstlenen, 5-9 kişiden oluşan takımdır. Bu takımda yazılımcı, testçi ve tasarımcı bulunur.

## SCRUM'DAKİ TOPLANTILAR

**Sprint Planlama Toplantısı (Sprint Planning):** Gereksinim listesinin çıkarıldığı, uygun dağıtım gereksinimlerinin belirlendiği, maliyet hesabının vb. işlerin yapıldığı toplantıdır.
**Sprint Gözden Geçirme Toplantısı (Sprint Review):** Bir sprintin sonunda o sprint için planlanan işlerin ne kadar tamamlandığı, nelerin başarılı nelerin başarısız olduğunun değerlendirilmesinin yapıldığı toplantılardır.
**Günlük Scrum Toplantısı (Scrum Daily Meeting):** Her iş günü başlamadan önce 15-30 dakikalık bilgi paylaşımı için günlük Scrum toplantısı yapılır. Bu görüşmede herhangi bir problem değerlendirilmez, takım üyeleri bir önceki gün ne yaptığından, bugün ne yapacağından ne gibi sorunlarla karşılaştığından bahseder ve böylece bu süreçte takım üyeleri birbirlerinin ne yaptığını daha kapsamlı bir şekilde anladıkları için her biri projeye daha hâkim olur.

## SCRUM'DAKİ BİLEŞENLER

**Ürün Gereksinim Dokümanı (Product Backlog):** Kullanıcı hikayelerine göre oluşturulmuş, onların gereksinimlerine göre önceliklendirilmiş, yapılacak tüm işlerin bir listesini içeren, devamlı bakım yapılan ve güncellenen canlı bir dokümandır.
**Sprint Dokümanı (Sprint Backlog):** Mevcut sprint için ürün gereksinim dokümanından elde edilmiş iş ve görevleri kapsar. Her sprint'ten önce takım üyeleri ürün iş listesinden üzerinde çalışacağı öğeleri seçer. Sprint dokümanı esnektir, sprint sırasında değişikliğe uğrayabilir.

**Sprint Kalan Zaman Grafiği (Burndown Chart):** Sprint boyunca işlerin ne kadarının yapıldığını, normalde ne kadarının yapılması gerektiğini karşılaştırabilmeyi sağlayan grafiktir. Bu grafik incelenerek günlük iş dağılımları duruma göre arttırılıp azaltılabilir.

## KAYNAKÇA


- https://tr.linkedin.com/pulse/yazılım-yaşam-döngüsü-nedir-veysel-ugur-kizmaz
- https://www.scrum.org/learning-series/what-is-scrum/what-is-scrum
- https://aws.amazon.com/tr/what-is/scrum/

## HESAPLAR

- https://github.com/Batuhanseyman
- https://medium.com/@batuhan.seyman
- https://www.linkedin.com/in/batuhan-seyman-a741ab26b/