# YAZILIM YAŞAM DÖNGÜSÜ MODELLERİ

Azra GÖKSU – 220601039

İZMİR BAKIRÇAY ÜNİVERSİTESİ Mühendislik ve Mimarlık Fakültesi Bilgisayar Mühendisliği Bölümü 1. Sınıf

## ÖZET:

Bu makalede Yazılım yaşam döngüsü modellerinden şelale modeli, V-Şekil modeli ve spiral modeli hakkında bilgiler verilmiş; bu modellerin avantajları ve dezavantajlarından ve hangi projelere uygun olduklarından bahsedilmiştir. Bu konulardan yanı sıra çevik yazılım geliştirmenin ne olduğu ve tarihçesi incelenmiştir. Bir çevik yazılım geliştirme metodolojisi olan SCRUM açıklanmış, günümüzde popüler olmasının sebeplerinden bahsedilmiş ve üstün ve zayıf yönleri incelenmiştir.

## Giriş – Yazılım Yaşam Döngüsü nedir?

Yazılım yaşam döngüsü (Software Development Life Cycle), bir yazılım projesinin geliştirilmesi sürecindeki bazı aşamalar bütünüdür. Bu aşamalar farklılık gösterse bile genellikle planlama ve gereksinim analizi, gereksinim tanımlama, tasarım, ürün oluşturma, test etme ve bakım aşamalarından oluşur. Bu aşamalar bir daireye ait gibidir. Yani bakım aşaması tamamlandıktan sonra tekrardan planlama aşamasına geçilebilir.

## 1. Yazılım Yaşam Döngüsü Aşamaları:

**1.1 Planlama:** Yazılım yaşam döngüsünün en önemli ve esas adımı planlama ve gereksinim analizi aşamasıdır. Endüstrideki alanında uzman kişiler, müşteriler, satış departmanları ve yapılan anketler sonucu alınan verilerden yola çıkılarak takımdaki kıdemli kişiler tarafından gerçekleştirilir. Bu veriler daha sonra temel proje yaklaşımını ve ekonomik ve teknik alanlarda ürün fizibilite çalışması yapmak için kullanılır. Bu aşamada ayrıca projenin oluşabilecek riskleri de tanımlanır. Teknik fizibilite çalışmasının sonucu, minimum risklerle projeyi başarılı bir şekilde uygulamak için takip edilecek çeşitleri yaklaşımları tanımlar.

**1.2 Gereksinimleri Tanımlama:** Gereksinim analizi tamamlandıktan sonraki aşama, ürün gereksinimlerini açık şekilde tanımlamak ve belgelemektir. Projenin gereksinimleri doküman haline getirildikten sonra bu gereksinimlerin müşteri veya market analizcileri tarafından onay alması gerekmektedir.

**1.3 Tasarım:** Bu aşamada sistemin genel yapısı ve projenin mimarisinin tasarımı yapılır. Gereksinim tanımlama aşamasındaki dokümanlar baz alınarak birden fazla tasarım fikri ortaya çıkabilir ve dokümanlanabilir. Bu fikirler daha sonra bütçe, zaman kısıtlaması, tasarımın taşıdığı risk, modülerliği gibi faktörler dikkate alınarak aralarından en uygun olanı seçilir ve fiziksel tasarım olarak hayata geçer.

**1.4 Ürün Oluşturma:** Bu aşamada ürünün adım adım inşası ve bazı geliştirmeleri gerçekleştirilir. Programlama ve yazılım bu aşamada yapılır. Geliştiriciler kendilerine gönderilen dokümanları ve yönergeler eşliğinde kodlamaları yapar. Dokümanlar ne kadar detaylı ve temiz olursa kodlama aşaması da o kadar sıkıntısız geçer. Python, C, C++, Java gibi gelişmiş programlama dilleri kullanılabilir.

**1.5 Test etme:** Bu aşamada ürün belgelerde belirtilen kalite standartlarına ulaşıncaya kadar testlerden geçer. Ürünün kusurları belirlenir, izlenir ve tekrardan test edilir. Bu süreçte

karşılaşılan sorunlar ikinci aşamadaki gereksinimlere eklenir. Döngü ikinci aşamadan tekrar test aşamasına gelir ve sorun çözülünceye kadar böyle devam eder.

**1.6 Bakım:** Bu aşamada ürün resmi olarak piyasaya sürülür ve sürüldükten sonra ortaya çıkabilecek hataların giderilmesi, yazılımın müşterilerden gelen tepkiye göre geliştirilmesi ve yenilenmesi gerekir.

## 2. Yazılım Yaşam Döngüsü Modelleri:

Yazılım yaşam döngüsü modelleri, yazılım projesinin yönetimi sürecinde kullanılan yöntemlerdir. Bu modeller projenin nasıl ilerleyeceği hakkında bir yaklaşım sunar. Her modelin kendine özgü bir işleyişi vardır. Bu modeller, projenin amacı, takımdaki yazılım geliştiricilerin özellikleri, zaman ve bütçe kısıtlamaları gibi etkenler dikkate alınarak en uygun olan seçilir. Seçilen modeller sayesinde proje süresince ortaya çıkabilecek sorunlar önceden tespit edilip önlenebilir veya soruna en uygun çözüm sunulabilir. Bu etkenlere uygun olarak seçilen bir yazılım yaşam döngüsü modeli projenin verimliliğini ve kalitesini artırır. Yazılım yaşam döngüsü modellerinden incelenecek olan bazıları şunlardır:

- Şelale (waterfall) modeli
- V süreç modeli
- Spiral (Helezonik) model

### 2.1 Şelale (Waterfall) Modeli:

Yazılım geliştirmede en eski ve temel modeldir. Günümüzde büyük şirketler ve hükümetler tarafından her türlü projenin yöntemi olarak kabul görse de kullanımı gittikçe azalmaktadır. Şelale modelinde yazılım geliştirme sürecinde uygulanan adımlar ardışık safhalar halinde gerçekleşir. Bu safhalar sırasıyla sistem gereksinimi, yazılım gereksinimi, analiz, tasarım, kodlama, test ve entegrasyon, idame ve bakımdır. Çeşitli safhalar bulunsa da şelale modelinin Royce tarafından önerilen temel safhaları bunlardır. Her safha sonunda belgelendirme sonucu bir çıktı elde edilir ve bu çıktı bir sonraki safhanın girdisini oluşturur. Bu sebepten bir safhaya geçmeden önce bir önceki safhanın tamamlanmış olmasına dikkat edilmelidir. Safhalarda atlama veya geri dönüşlere genellikle izin verilmez çünkü projeye maliyeti yüksek olur. Safha bitişleri kesin ve net bir şekilde belirtilmiştir ve her safha bitiminde katı incelemelerle proje yoğun bir denetim altından geçer. Her safhada doküman ve test olmalıdır. Bu ikisinin olmadığı bir safha tamamlanmış sayılamaz.

Bu model genellikle belirsizlik içermeyen ve değişiklik istemeyen projeler için uygundur. Uzun vadeli projelerde çok ciddi sıkıntılar oluşturabilir. Projenin bir safhası bitmeden diğerine geçilemeyeceğinden uzun süre o safhanın bitmesi beklenmek zorundadır ve müdahale etme gibi bir seçenek yoktur.

*2.1.1 Şelale modelinin faydaları:*

- Başlangıçta gereksinimlerin açık ve anlaşılır şekilde tam olarak yapılması, zaman ve maliyet tahminlerinin daha doğru yapılmasını sağlayarak projedeki belirsizlikleri azaltır.
- Teoride iyi bilinen bir metottur.
- Projedeki talimatlar net bir şekilde belirlendiği için çalışanların kendi başına iş yapma ihtimali azalır ve ne tecrübesi olmayan çalışanlar için bir yol haritası görevi görür. Yöneticiler için görev dağılımı yapmak kolaylaşır.

- Bir önceki safha tamamlanmadan diğerine geçiş olmadığı için proje takibi yapmak kolaydır.
- Müşterinin projeye nerede ve ne zaman katkı yapacağı bellidir.
- Projede elemanların iş yükünün üst üste binmesini önler.

*2.1.2 Şelale modelinin dezavantajları:*

- Değişiklik veya geliştirme amaçlı büyük geri dönüşler oldukça zor ve maliyetlidir. Bu da proje süresince değişikliklere ve yeniliklere olumlu cevap verilmez.
- Bir aşamada ortaya çıkan sorun sonraki bütün aşamaları olumsuz etkiler.
- Katı kurallar ve yoğun denetim nedeniyle proje yavaş ilerler.
- Gereksinimler proje başında tam olarak tespit edilmeyebilir veya müşteri istekleri doğru anlaşılmayabilir bu da ortaya yanlış ve ihtiyacı karşılamayan bir ürün çıkmasına sebep olabilir.
- Sistemdeki hatalar test aşamasına gelene kadar test edilemez.

**2.2 V Süreç Modeli:**

Aşamaları V harfini oluşturduğu için V süreç modeli veya V şekil modeli denmiştir. Aslında şelale modeline benzer olmakla birlikte şöyle bir farklılığı da vardır. Planlama, gereksinim belirleme, üst ve alt seviye aşamaları vardır. Bu aşamalar şelale modelindeki gibi ardışık ve alt alta gerçekleşir. Fakat V modelinde en son aşama tamamlandıktan sonra tekrardan yukarı çıkılarak adımlar tekrarlanır. V şeklinin sol tarafı üretim, sağ tarafı ise test etme işlemleri içindir. Bu modelin bir özelliği de aynı hizada olan adımların birbirini karşılamasıdır. Bir adım gerçekleştirildikten hemen sonra bu adımın karşılığının ne olacağı tespit edilebilir. Bu modelin en uygun olduğu projeler belirsizliklerin az, iş tanımlarının belirgin olduğu küçük projelerdir.

*2.2.1 V Şekil Modelinin avantajları:*

- Bu model sayesinde kullanıcının projeye katkısı artırılır.
- Basit ve kullanılması kolay bir modeldir. Yöneticilerin proje takibi yapması kolaydır.
- Test aşamaları kodlamadan önce yapılır ve bu da büyük bir zaman kazandırır.
- Tasdik planları erken aşamalarda vurgulanır ve tüm teslim edilebilir ürünlere uygulanır.
- Test aşamalarının döngüde erken yapılması nedeniyle başarı şansı yüksektir.
- Hata denetimi kolaydır ve yapılan adımların karşılığının ne olacağı hemen tespit edilebilir.

*2.2.2 V Şekil Modelinin dezavantajları:*

- Aşamalar arasında tekrarlama yoktur.
- Risk çözümleme aktiviteleri yoktur.
- İş ve ürün gereksinimleri projeye göre değişkenlik gösterebilir.
- Şelale modelinde olduğu gibi esnek değildir ve büyük değişimlere sıcak bakılmaz.

**2.3 Spiral Model:**

Spiral model yazılım yaşam döngüsünün en önemli modellerinden biridir. Yazılım geliştirimine sistematik ve yinelemeli bir yaklaşım sunar. Spiral modelde diğer modellerden farklı olarak aşamalardan tekrar tekrar geçilir ve her geçişte projenin ilerleme kat etmesi hedeflenir. Spiral fikrini temel alır ve spiralin her bir tekrarı tamamlanmış bir yazılım

geliştirme döngüsünü temsil eder. Spiral model risk odaklı bir modeldir. Her bir tekrarla riskler olabildiğince aza indirilmeye çalışılır. Spiral model aşağıdaki aşamalardan oluşur:

1. Planlama: Projenin genel kapsamı hakkında plan yapılır.
2. Risk analizi: Projeyle alakalı riskler belirlenir.
3. Mühendislik: Bir önceki adımdaki gereksinimler dikkate alınarak yazılım geliştirilir.
4. Değerlendirme: Müşterilerin isteklerine uyup uymadığı ve iyi kalitede olup olmadığı denetlenir.
5. Planlama: Yineleme adımı değerlendirme aşamasındaki sonuçlara göre yeni bir planlama adımı ile başlar.

Spiral model genellikle karmaşık ve büyük yazılım geliştirme projelerinde tercih edilir. Daha esnek ve uyum sağlayabilen bir yaklaşım sunar. Ayrıca önemli belirsizliklere ve büyük risklere sahip projeler için çok uygundur.

*2.3.1 Spiral Modelin avantajları:*

- Yazılım yaşam döngü modellerinden en esnek çalışılabilen modellerden birisidir.
- Proje takibi her yineleme fazında yapıldığından yöneticiler için oldukça kolay ve efektif bir yöntemdir.
- Risk yönetimi bu modelin anahtar özelliğidir ve diğer modellere göre daha çekici kılar.
- Proje sonlarında bir değişiklik gerekir veya önerilirse bu değişikliği gerçekleştirmek az maliyetli ve kolaydır.
- Son derece özelleştirilmiş ürünler ortaya konulabilir.
- Kullanıcılar sistemi erkenden görebilir.

*2.3.2 Spiral Modelin dezavantajları:*

- Diğer modellere göre oldukça karmaşıktır, spiral sonsuza kadar gidebilir.
- Maliyetli olduğu için küçük projelerde kullanılamaz.
- Düşük risk içeren modeller için uygun değildir.
- Müşteriden gelen çeşitli özelleştirmelerden dolayı aynı prototipi başka projelerde kullanmak zordur.
- Belirsizlikleri ve riskleri değerlendirmek için yüksek tecrübe gerekir.
- Proje başında fazların sayısı belirli olmadığından süre yönetimi oldukça zordur.

# 3. Çevik Yazılım ve SCRUM:

## 3.1 Çevik Yazılım Geliştirme nedir?

Çevik yazılım geliştirme, basit prensiplere dayanan yazılım geliştirme gruplarının genel adına denir. Bu yaklaşımda kendi kendini örgütleme, resmiyet içeren etkili bir yönetim, yüksek kalitedeki yazılımların hızlı dağıtımı, uygun maliyetli, müşterilerin ve çalışanların ihtiyaçlarını karşılayan çözümler üreten yinelemeli (iterational) ve artırımlı (incremental) bir proje hedeflenir. Çevik metotlar, geleneksel yöntemlerin yetersiz kaldığı durumlarda alternatif bir yöntem olarak ortaya çıkmıştır.

Çevik yazılım metodolojilerinin kökeni 1957 yılında IBM'deki yazılım geliştirme çalışmalarına dayanmaktadır. Bu çalışmalar daha sonra E. A. Edmonds tarafından artırımlı yazılım geliştirme olarak tanımlanmış ve 1974 yılında "Uyumlu Sistemler İçin Yazılım Geliştirme" başlıklı bir makale ile tanıtılmıştır. 1990'lı yıllar ise Şelale modelinin hantal bir

sistem olduğu iddia edilerek onun yerine daha hızlı ve çevik yazılım geliştirme metodolojileri sunmaya çalışılan yıllar olarak geçmiştir. 2001 yılının şubat ayında yazılım dünyasının önde gelen 17 ismi Utah'ta bir araya gelerek yazılım geliştirme üretkenliğini arttırmaya yönelik 2 günlük beyin fırtınası yapmışlar ve "Çevik Yazılım Geliştirme Manifestosu" ve "Çevik Yazılımın Prensiplerini" yayınlamışlardır.

### 3.2 SCRUM Nedir?

Scrum, efektif planlama, kollektif çalışma ve devamlı ilerleme unsurlarına öncelik vererek takımlara kademeli olarak ilerlemeli bir yapı sunan bir çevik yazılım metodolojisidir. Çevik yönetim ve proje yönetiminde karmaşık bir ortamda ürünleri geliştirmek, sürdürmek ve sunmak için bir yaklaşımdır. Bu metodolojinin temel özelliği, gözlemci, geliştirmeci ve tekrara dayalı olmasıdır. Birçok modern yazılım projesinin karmaşık olduğu ve en baştan tüm projenin planlanmasının zor olduğu şeklindeki bir varsayım üzerine kurulmuştur. Scrum, bu karmaşıklıkları azaltmak için üç ilkeye sahiptir:

1. *Şeffaflık:* Projedeki sorunlar ve ilerlemeler günlük olarak tutulur ve herkes tarafından takip edilebilir olması sağlanır.
2. *Gözlem:* Ürünün parçaları ya da fonksiyonları düzenli aralıklarla teslim edilir ve değerlendirilir.
3. *Uyumlama*: Ürün için gereksinimler en baştan bir defalığına belirlenmez, aksine her teslimde tekrar değerlendirilir ve duruma göre düzenlemeler yapılır.

Scrum'ın amacı, başta hayal edilip tasarlanan bir ürünün hızlı, ucuz ve kaliteli şekilde üretilmesidir. Tasarlanan ürünün hayata geçirilmesi, müşteri tarafından detaylıca hazırlanmış talep listesinin aşama aşama gerçekleştirilmesi yerine, müşteri tarafından tanımlanan işlevler iki veya dört haftalık "sprint" adı verilen dönemler içerisinde gerçekleştirilir ve gözden geçirilir.

#### *3.2.1 SCRUM'ın avantajları:*

- Çevikliği sayesinde şirketlere zaman ve paradan tasarruf ettirir.
- Scrum metodolojisi, iş gereksinimleri belgelerinin elde edilmesinin zor olduğu durumlarda projenin başarılı bir şekilde geliştirilmesine olanak tanır.
- Hızlı hareket eden gelişmeler hataların kolayca çözümlenmesiyle hızlıca kodlanabilir.
- Diğer çevik metodolojiler gibi yinelemeli bir yapısı vardır ve kullanıcıdan devamlı geri dönüt ister.
- Kısa atlamalar sayesinde değişiklik yapmak kolaydır.
- Sorunlar günlük yapılan toplantılarla önceden belirlenir bu şekilde hızlıca çözüme kavuşturulur.
- Scrum herhangi bir teknoloji ve programlama diliyle kolayca çalışabilir.

#### *3.2.2 SCRUM'ın dezavantajları:*

- Bir görev iyi tanımlanmazsa tahmin edilen projenin süreceği zaman ve maliyet doğru olmayabilir ve birçok sprint dönemlerinin oluşmasına neden olabilir.
- Tecrübeli takım çalışanları gerektirir. Yeni başlamış bir çalışan projenin alacağı vakti artırabilir.
- Proje sırasında takımdan birisi çıkarsa tüm projeye büyük bir yan etkisi olur.
- Küçük ve hızlı hareket eden projeler için en uygundur.

*3.2.3 SCRUM Günümüzde neden bu kadar tercih edilir?*

Scrum, basitliği ve yüksek performans kalitesinden dolayı günümüzde en popüler çevik yazılım geliştirme metodolojileri arasındadır. Başarma hissiyatı, olumlu geri bildirim ve ekip çalışması ortamında yapılan işi sahiplenme ihtiyacından avantaj sağlar. Bir proje boyunca değeri hızlıca sunmak için tasarlanmış uyarlanabilir, tekrarlı, hızlı ve esnek bir metodoloji olması Scrum'ın popüler olma nedenlerindendir.

## KAYNAKÇA


- https://ybsansiklopedi.com/wp-content/uploads/2015/08/Yaz%C4%B1l%C4%B1m-Geli%C5%9Ftirme-Modelleri-Yaz%C4%B1l%C4%B1m-Ya%C5%9Fam-D%C3%B6ng%C3%BCs%C3%BCSDLCYBS.pdf
- https://dergipark.org.tr/en/download/article-file/328845
- https://www.geeksforgeeks.org/software-engineering-spiral-model/
- http://www.ijcait.com/IJCAIT/13/1334.pdf
- https://tr.wikipedia.org/wiki/Atik_yaz%C4%B1l%C4%B1m_geli%C5%9Ftirme
- https://www.wrike.com/scrum-guide/scrum-methodology/
- https://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/



ULAŞIM LİNKLERİ

Linkedin: https://www.linkedin.com/in/azra-g%C3%B6ksu-b8b2b125a/

Medium: https://medium.com/@azragoksuni

Github: https://github.com/azragks