**YAZILIM YAŞAM DÖNGÜSÜ MODELLERİ AÇIKLANMASI VE GÜNÜMÜZDEKİ ÖNEMİ**

(Ali Berke Kara, İzmir Bakırçay Üniversitesi, Bilgisayar mühendisliği bölümü, 210601013)

**Özet**: Herhangi bir ürünün, yatırımın veya servisin canlı organizmalar gibi varlıklarını sürdürmelerinden hareketle yaşam döngüsü kavramı ortaya çıkmıştır. Maliyetlerin takibi ve kontrolünün yapılabilmesi için döngünün kontrol edilmesi oldukça önemlidir. Yazılımlara gelince, geliştirilmekte ve kullanılmakta olan her program için bir yazılım yaşam döngüsü söz vardır ve bu döngünün de iyi kontrol edilmesi gerekmektedir. Yaşam döngüsünde genel kabul gören altı aşama vardır. Bunlar; ihtiyaçlar, tanımlama, tasarım, gerçekleştirme, uygulama ve bakım aşamalarıdır. Bu genel alan içinde, yaşam döngüsü modellerinden de söz etmek gerekiyor. Bu yaklaşımların ilki ve en yaygın şekilde kullanılanı, yaşam döngüsündeki aşamaların birbiri ardına sıralandığı, birinin çıktılarının bir sonraki adımın girdilerini oluşturduğu Çağlayan Modeli’dir. Bu modelin gerçek hayattaki ortaya çıkardığı birtakım sorunların önüne geçmek için öne sürülen V süreç modeli , çağlayan modelinin bir türevi olan Artışlı Model ve yazılım mühendisliği sürecini her türlü uygulamayıkapsayacak şekilde açıklayan Sarmal Model, Helezonik ve Yap ve Düzelt Modeli yaklaşımlardan belli başlılarıdır. Tüm modellerin kendi içinde zayıf ve kuvvetli yanları bulunmaktadır. Önemli olan bir uygulamada şirket kültürüne en uygun olanı benimseyebilmektir. En sık kullanılan model Çevik yazılım geliştirme modelidir. Extreme Programming (XP) ve SCRUM olmak üzere ikiye ayrılır. Bu modellerin yazılım geliştirme hayatımızdaki önemi oldukça yüksektir. Bu çalışmada yazılım yaşam döngü modelleri açıklanmış, yapılan açıklamalara göre hangi projede hangi model kullanılacağına dair yol gösterilmiş olup, bu modellerin neden önemli olduğu vurgulanmıştır.

**Anahtar Kelimeler:** Yazılım yaşam döngüsü, yazılım yaşam döngü modelleri, SCRUM

**YAZILIM YAŞAM DÖNGÜSÜ MODELLERİ VE KARŞILAŞTIRMALARI**

Yaşam döngüsü, tüm canlıların ortak özelliklerinin buluştuğu doğma, büyüme, olgunluk ve sonunda ölüm aşamalarının olduğu bir süreçtir. Tüm canlılar gibi herhangi bir ürünün, materyalin, veya yatırımın da bir yaşam döngüsü olur. Yazılım kısmında ise, kullanılan ve geliştirilen her programın bir yazılım yaşam döngüsü vardır. Yaşam döngüsünde genel olarak kabul edilen altı aşama vardır. Bunlar; planlama, analiz, tasarım, geliştirme, uygulama ve bakım aşamalarıdır. Planlama aşaması yazılım yaşam döngüsünün ilk evresidir. Bu aşamada kullanıcının isteğini, projedeki ihtiyaçlarını tespit ettiği aşamadır. Bu planlamada projenin baştan sona tüm yaşam döngüsü bulunur. Projede yer alacak kişilerin iş bölümü, projenin adımları, maliyeti ve zaman sınırı gibi detaylar, bu aşamada proje yöneticisi ya da analist tarafından yapılır. Analiz aşaması ikinci evresidir. Tasarlanacak olan üründe ne amaçlandığını detaylı olarak dosyaya yazar. Bu şekilde proje daha somut bir hale gelir. Tasarım aşaması üçüncü evredir. Yazılım projesinin iskeleti ve şekli bu evrede oluşturulur. Projenin bütün mantıksal veya fiziksel tasarımı burada gerçekleştirilir. Geliştirme aşaması dördüncü evredir. Burada artık yazılım projemiz hayata geçer. Tüm kodlama, kurulum ve denemeler bu evrede gerçekleşir. Son aşamalar ise test ve bakımdır. Test aşamasının tamamlanması yazılım projesinin son kullanıcılar tarafından kullanılmasının önünde engel bulunmadığını söyler. Son kullanıcıya ulaştıktan sonra oluşabilecek sorunlara karşı bakım ihtiyacı ise yine analistler ve geliştirici personeller tarafından takip edilir. Projenin hedeflerine ulaşabilmesi için her türlü süreç ve yöntemler mevcuttur. Her bir süreç modeli, sağlıklı çalışmak için yazılım geliştirmede bu yaşam döngüsünü takip eder. Yazılım süreç modelleri üretilenlerin kalitesine katkı sağlamanın yanında, projelerin karışıklığını azaltıp karmaşıklıkları engeller. Yazılım Şirketleri projenin amaçlarına ulaşabilmesi için kurulmuştur. Yazılım yaşam döngüsünün toplamda yedi farklı modeli vardır. Bunlar; Gelişigüzel model, Barok modeli, Çağlayan yaşam-döngü modeli, V süreç modeli, Helezonik (spiral) model, Artımsal geliştirme süreç modeli ve Kodla ve düzelt yaşam-döngü modelidir. İlk sırada Gelişigüzel model yer almaktadır. Model olarak adlandırılmasının doğru olmaması ile birlikte herhangi bir yöntem gözetmez. İlgilenen ya da geliştiren kişiye bağlıdır. Takip edilmesi ve bakımı oldukça zor olan bu model 60 ‘lı yıllarda kullanılmıştır. Genel olarak üretim ortamı tek kişiden oluşmaktadır ve çok basit bir programlamaya sahiptir. İkinci model Barok modelidir. Bu model yaşam döngüsü temel adımlarının doğrusal bir şekilde geliştirildiği modeldir. Miadı olarak 70’li yıllarda kalmıştır. Bu model dokümantasyonu ayrı bir zaman dilimi olarak ele alır ve yazılımın geliştirme ve test aşamaları bittikten sonra yapılmasını öngörür. Hâlbuki bugün dokümantasyon, yapılan işin doğal bir ürünü gibi kabul edilmektedir. Aşamalar arasındaki geri dönüşlerin nasıl yapılacağı da tanımlanmamıştır. En kötüsü de gerçekleştirim evresine daha çok önem gösteren bir model olduğundan günümüzde kullanımı tavsiye edilmemektedir. Üçüncü model Çağlayan (şelale) yaşam modelidir. Geçmişte kullanılan en popüler yazılım geliştirme modelidir. Ayrıca en çok tanınan, en eski ve en temel modeldir. Diğer bir ifadeyle geleneksel yazılım geliştirme modelidir. Öncelikle gereksinimlerin tanımlanması yapılır. Sonrasında ise yazılım ve sistem tasarımı gerçekleştirilir. Bununla birlikte kodlama ve kısım kısım test etme evreleri gerçekleştirilir. Bir aşama bitmeden diğer aşamaya geçilmemektedir. Dokümantasyon her aşamada yapılır ve olması da gerekir. Eğer bir aşamada dokümantasyon ya da test yapılmamışsa, o aşama tamamlanmamış olarak kabul edilir. Yapıyı birleştirme ve sistemi test etme yapıldıktan sonra da sistemin bakım ve sürdürülebilirliği sağlanmaya çalışılır. Bu modelin bazı sorunları vardır. Gerçek hayattaki projelerin genellikle tekrar edilmesi gerekir. Tümüne bakıldığında yazılımın kullanıcıya ulaşma süresi çok uzundur. İhtiyaçlar çoğu kez açık bir ifadeyle yapılamadığından dolayı, hataların düzeltilme ve eksikleri kapatma maliyetleri yükselir. Üst düzeydekilere ürünün eline ulaşma süresinin uzun olması, projenin sanki hiç bitmeyeceği ve durmaksızın gider merkezi ebatına geldiği düşüncesini giderek artırmaktadır. Ayrıca bu modelde aşamalar arasındaki geri dönüşlerin nasıl yapılacağı da bellidir. Ancak bu modelde bazı noktalara dikkat edilmesi gerekir. Bunlardan en önemlisi, her ne kadar aşamalar arasında geri dönüş müümlün olsa da analiz evresinde mümkün olan tüm detayların tasarıma yansıtılabilmesi için kullanıcı ve sistem ihtiyaçlarının en küçük detayına kadar belirlenmesi gerekir. Tasarım evresi de yazılımın bütün ihtiyaçlarını karşılayabilecek düzeyde ayrıntılı bir çalışma gerektirir. Bu nedenle bu modeli kullanan proje ekipleri en fazla zamanı bu iki evre kullanmak mecburiyetindelerdir. Bütün bu zorlu çalışmalara karşı bilhassa uzun zaman gerektiren projelerde gereksinimlerin değişecek olması mecburidir. Proje ekipleri bu modelle yapılan çalışmalarda moralini bozmakta ve kodlama haricindeki aşamalar fazla önem göstermemektedirler. Dördüncü model V süreç modelidir.  Süreç adımları kodlama yapıldıktan sonra yukarıya ve iki tarafa doğru eğim aldığı ve V şeklini oluşturduğundan bu şekilde adlandırılmıştır. Çağlayan yaşam-döngü modelinin biraz daha gelişmiş halidir. Sol taraf üretim, sağ taraf ise kontrol bölümüdür. Üst ve alt seviye tasarım vardır. Üst seviye daha yaygın bir tasarımdır. Alt seviye ise girdiler-çıktılar, beklentiler gibi daha ayrıntılı bir tasarıma sahiptir. Proje takibi bu yöntemde oldukça kolaydır. Modelin kullanımı ise genel olarak basittir. Lakin aşamalar arası tekrarlamaları kullanmaz. Risk çözümleme ile ilgili faaliyetleri bulundurmaz. V süreç modelinin temel çıktılarını Kullanıcı Modeli, Mimari Model ve Gerçekleştirim Modeli olarak sıralayabiliriz. Kullanıcı modelinde, geliştirme evresinin kullanıcı ile olan ilişkileri belirlenmekte ve sistemin nasıl kabul göreceğiyle alakalı test etiketleri ve planları ortaya çıkarılmaktadır. Mimari model düşük seviyeli tasarım olarak kastedilebilir. Tasarım sistemi küçük parçalara bölünür. Sistem tasarımı ve üretilecek alt sistemle bütün sitemin test süreciyle alakalı işlevler. Gerçekleştirim modeli yazılım parçalarının kodlanması ile test edilmesiyle alakalı fonksiyonlar. Modellerden beşincisi Helezonik modeldir.  Planlama, risk analizi, üretim ve kullanıcı değerlendirmesi olmak üzere 4 ana başlıktan oluşur. Planlama; ortaya çıkacak olan ara ürün için planlama, amaç belirleme, bir önceki işlemde ortaya çıkan ara ürün ile birleştirilme. Risk analizi; risk seçeneklerinin araştırılıp risklerin tanımlanması. Üretim; Ara ürünün oluşturulması. Kullanıcı değerlendirmesi; kullanıcı tarafından ara ürün ile ilgili olarak yapılan testler ve değerlendirmeler. Ve bir spiral oluşturacak biçimde dönerek bu aşamalar tekrarlanır. Şelale modelinde yok sayılan riskleri göz önünde bulundurur. Proje çevrimlere ayrılır ve her bir çevrimin riskleri ayrı ayrı olmak üzere ele alınır. Çağdaş modellere oldukça yakın olan bir modeldir. Modelin en yararlı hususlarından biri de prototipler yardımıyla kullanıcılar sistemi erkenden görebilirler. Riske duyarlı yaklaşımı potansiyel zorlukları engeller. Hataları erken gidermeye fokuslanmıştır. Projelerin büyüklüğüne bağlı olarak prototip sayısı artabilir ve azaltılabilir. Bunlar ile birlikte kompleks bir modeldir. Spiral sonsuza kadar gidebilir. Sürekli olarak prototip çıkarmak ve belgeleme yapmak nedeniyle küçük ve düşük riskli projeler için maliyeti yüksektir. Avantajları olarak üç madde sayılabilir: Kullanıcı katkısı, yönetici bakışı, yazılım geliştirici (mühendis) bakışı. Kullanıcı katkısı; Ara ürün oluşturma ve ara ürünün kullanıcı tarafından üretim süresi boyunca test edilmesine dayanır. Ayrıca yazılımı kullanacak kişinin süre erkenden katılmasıyla gelecekte oluşabilecek bazı sorunları engeller. Yönetici bakışı; Proje sahibi veya yüklenici tarafındaki yöneticiler, proje boyunca çalışan yazılımlarla karşılaştıklarından, daha basit şekilde izleme ve hak etme planlaması yapılır. Yazılım geliştirici (mühendis) bakışı; yazılımın kodlanması ve test edilmesi daha erken bir zamanda başlar. Altıncı model Artımsal geliştirme süreç modelidir. Sistemi bir seferde teslim edeceğine iki aşamaya böler: teslim ve geliştirme. Her teslim beklenilen işlevselliğin bir kısmını tamamlar. Öncelikli gereksinimler erken teslimlerle gösterilir çünkü kullanıcı gereksinimleri önceliklidir. Birparçanın geliştirmesi başlarsa o parçanın gereksinimleri durdurulur. Değişiklikler olursa daha sonraki teslimlerde göz önünde bulundurulur. Planlama, analiz ve tasarım aşamalarından sonra bir çekirdek yapı oluşturularak bir sürüm oluşturulur. Ve sistem artırılması yapılarak sürekli bir şekilde gelişmiş sürümler meydana getirilir. Bu döngü program hazır olana kadar durmadan devam eder. Erken aşamalarda her teslimle birlikte müşteriye bir ürün gelmesi sayesinde sistem erken aşamalarda işlevsellik kazanır. Üzerine ekleye ekleye ve gelişerek ilerlediği için projenin tamamen kullanılmaz hale gelmesi önlenmiş olur yani başarısızlık riski düşer diyebiliriz. Sistem daha fazla test edilme fırsatını bu döngüyle birlikte bulmuş olur. Böl ve yönet (Divide and Conquer) yaklaşım şekline uyabilen bir modeldir. Dezavantajları ise deneyimli personel gerektirir. Artırımları belirleyip tanımlayabilmek için tüm sistemin tanımlanması mecburidir. Kendi içlerinde artırımların tekrarlamalar yapmasını engeller. Son ve yedinci model Kodla ve Düzelt modelidir. Çok uzun satırları olmayan programlar için uygundur. İlk adımda programın ilk sürümü geliştirilir. Ayrıca sistem son şekle gelene kadar geliştirme devam eder.  Genel olarak kesinliğe varamamış olan bir ürün fikriyle başlar ve ürün hazır oluncaya dek kodlama devam eder. Çok küçük projeler veya ömrü kısa olacak prototipler için kullanılabilir. Herhangi bir planlamaya gerek yoktur. Uzmanlık gerektirmediğinden ötürü birçok insan bu modeli kullanabilir. Yazılımı geliştirmesi kolay olduğu için deneyimsiz firmalardaki birçok projede bu model kullanılır. Fakat bu modelin genellikle kontrolü zordur. Kaynak planlaması bulunmamaktadır ve bitiş süresi belli değildir. Ayrıca hataları bulması dokümantasyon yapılmadığı için zordur ve bakım yapılabilirlik açısından düşük seviyededir. Sonuç olarak öğrenciler veya bireysel geliştiriciler için uygun fakat ekip çalışmaları için uygun değildir. En kolay yazılım geliştirme yoludur fakat aynı zamanda en pahalı olanıdır. Tüm bu modellerin karşılaştırılmasına gelecek olursak; Kodla ve düzelt, çağlayan ve v modellerinde gereksinim belirleme başlangıçta yapılır, spiral ve artımlı modellerde belirli sıklıkla yapılır. Çağlayan, V ve spiral modellerinde maliyet yüksektir, Kodla ve düzelt ve Artımlı süreçte maliyet düşüktür. Barşarı garantisi Kodla düzelte ve çağlayan modellerinde düşük; Artımlı ve V modellerinde orta; Spiral modelde yüksektir. Uzmanlık gerekliliği Çağlayan, Artımlı ve V modelinde orta; Spiralde yüksek; Kodla düzeltte düşüktür. Kodla düzelt ve Çağlayan modelleri basit; Artımlı ve V modeli orta; Spiral modelde karışıktır. Zaman süresi Kodla düzeltte ve Çağlayan modelinde çok uzun; V modelinde orta; Artımlı ve Spiral modelde uzundur. Karşılaştırma sonucu​​nda anlaşılmıştır ki​​ yazılım süreç modelleri tamamıyla birbirinden ayrı düşünülmez ve genel olarak birlikte kullanılır. Yazılım yaşam döngüsü modelleri, geliştirilecek yazılım türüne ve kullanım alanına göre seçilmelidir. Yazılım süreç modelinin seçimi için modelin oluşabilecek risklere dayanabilme miktarı, projenin büyüklüğü, projenin karmaşıklığı, gerçekleştirecek firmanın düzeni, zaman, maliyet gibi ölçütlere dikkat edilmelidir. Bu çalışma genel bir değerlendirme çalışması olduğu için çıkarttığı sonuçlar yalnız genel bir çerçevede kabul görür. Üretilecek projede hangi yazılım yaşam döngü modelinin kullanılacağını belirlemekten çok, bu belirleme çalışmasının girdilerinden biridir. Yazılım projelerindeki başarı oranlarını ve ekip içi iletişimi arttırmak amacıyla ortaya Çevik Modeller çıkmıştır. Çevik yazılım geliştirme modelleri şunlardır: Extreme Praogramming, SCRUM. Extreme Preogramming (XP) , Kent Beck ve arkadaşları tarafından 1996 yılında kurulmuştur. 4 ana başlıktan oluşur: Basitlik, Cesaret, Geri Dönüş, İletişim. Basitlik, yazılan kodun basit, sade, anlaşılabilir olmasını sağlar. Dokümantasyon kısmını uzatmayı engeller. Cesaret, yapılan projede korkmadan ilerlemeyi sağlar yani cesur olunmalıdır. Örneğin bir kodun silinmesi gerektiğinde silinip yeniden yazılması gerektiğinde bunu kormadan yapmasını söyler. Geri dönüş sayesinde gelecekte ortaya çıkabilecek hatalar engellenmiş olur. Müsteri ile üretici firma arasında iletişim söz konusudur. İletişim, projelerde büyük sorunlardan bir tanesidir. XP ise bu soruna engel olmaya çalışır. Ekip içi iletişime önem gösterir ve iletişimin sağlıklı bir şekilde devam etmesi için çaba gösterir. Diğer bir model SCRUM’dır. SCRUM, büyük projeleri parçalara ayırarak her bir parçaya sprint adını verir. Projeyi parçalara ayırdıktan sonra her bir sprinti tek tek geliştirmeye çalışır. SCRUM’da ekip içindeki iletişim oldukça önemlidir ve her gün “SCRUM MEETİNGS” adı verilen toplantılar gerçekleştirilir. SCRUM’ın üç temel kavramı vardır: Roller, Toplantılar ve Bileşenler/Araçlar. Roller; Ürün sahibi, SCRUM yöneticisi ve SCRUM ekibinden oluşmaktadır. Ürün sahibi diğer bir deyişle projenin beynidir. SCRUM yöneticisi, ekibe SCRUM etkilerine göre yön verir. SCRUM takımı ise birbiriyle devamlı iletişim halinde olan 5-9 kşiden oluşmaktadır. Toplantılar; SCRUM’ın olmazsa olmazıdır. Her gün SCRUM toplantıları olur ve bu toplantılarda ekibin önceki gün neler yaptığı, ortaya çıkan sorunlar ve bugün neler yapılacağı konuşulur. Ayrıca her sprint için bir toplantı yapılır. Bileşenler; Öncelikle ürün gereksinim dokümanı oluşturulur. Bu dokumanın içinde proje bitinceye dek yapılması gerekenler basit bir şekilde yazılır. Sprint dokümanı oluşturulur. Ürün gereksinim dokümanına takiben yapılan sprint dokumanın amacı her sprintin ona göre ayarlanmasıdır ve bu dokümanı sadece ekip üyeleri değiştirebilir. Yapılan işin hangi seviyede olduğunu ve planlanan zamana göre değerlendirmesi Sprint Kalan Zaman Grafiğinde hazırlanır. Günümüzde en çok kullanılan yazılım geliştirme yöntemi SCRUM’dır. Hatta sadece yazılım geliştirmede değil birçok konuda SCRUM kullanılır. Nedenlerine gelinecek olursa SCRUM; zamandan ve paradan tasarruf eder, güncel gelişmelere ve yüksek teknolojilere uyum sağlar, karmaşık görülen ve gereksinimleri belli olmayan projelere idealdir, ekip içi iletişim güçlü olduğu için hatalar ve sorunlar erken fark edilir, kullanıcıdan devamlı geri bildirim ister ve bu geri bildirimlere göre sorunların giderilmesini sağlar, diğer yazılım geliştirme metotları gibi yinelemelidir, değişen gereksinimlere hızlı bir şekilde cevap oluşturur ve böl fethet yaklaşımına göre hareket eder.

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

<https://ubys.bakircay.edu.tr//AIS/Common/Content/PdfJs/web/viewer.html?file=https%3a%2f%2fubys.bakircay.edu.tr%2f%2fAIS%2fCommon%2fFile%2fGetEncryptedFileAsPDF%3fefai%3d!xBBx!d2GHJhH3kzUAemhZdiMBw!xGGx!!xGGx>!

<https://ubys.bakircay.edu.tr//AIS/Common/Content/PdfJs/web/viewer.html?file=https%3a%2f%2fubys.bakircay.edu.tr%2f%2fAIS%2fCommon%2fFile%2fGetEncryptedFileAsPDF%3fefai%3dNjBGKkTTLeovHAijlE7bVg!xGGx!!xGGx>!

<https://polen.itu.edu.tr/bitstream/11527/5973/1/2137.pdf>

<https://hayririzacimen.medium.com/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-ve-s%C3%BCre%C3%A7-modelleri-70fdfb2f8f77>

<https://www.codex.com.tr/yazilim-gelistirme-modelleri>

<https://medium.com/@omerharuncetin/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BC-modelleri-543c7879a742>

Medium hesabım: <https://medium.com/@karaaliberke>

GitHub hesabım: <https://github.com/aliberkekara-ceng>

Linkedin hesabım: <https://www.linkedin.com/in/ali-berke-kara-b29499235/>