Dilara Ahretlikoğlu
210601060

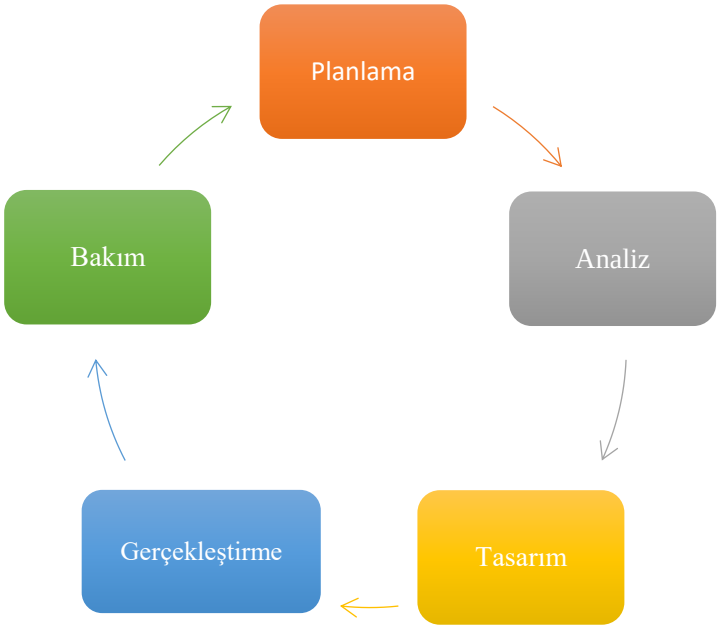
1.YAZILIM YAŞAM DÖNGÜSÜ KAVRAMI

Geliştirilen bir projenin planlanma aşamasından teslim edilme aşamasına kadar gerçekleştirilen belirli adımlara yazılım yaşam döngüsü denir. Yazılım yaşam döngüsü modeli, yaşam döngüsünün 6 adımını konu alır. Bu aşamalar sayesinde maliyet, süreç yönetimi ve diğer yazılım süreçleri daha kolay ve düzenli ilerler.
Yazılım yalnızca kodlama aşamasından oluşmayan oldukça zorlu bir süreçtir. Yaşam döngüsünün gereksinim, analiz, tasarım, gerçekleştirme, bakım ve emeklilik evreleri de bu sürecin bir parçasıdır. Bu aşamalar bir kez gerçekleştirildikten sonra proje tamamlanmayabilir. Bir döngü halinde sürekli gerçekleştirilmeye devam eden safhalardır. Proje tamamlandıktan sonra oluşan hatalar, müşterinin yeni istekleri, projeye eklenecek yeni özellikler vb. gibi konular için bu süreç devam etmektedir. Proje içerisinde bu süreçler tamamlandıktan sonra hiçbir hata ve yeni istek olmadığını varsayarsak bile bu noktada devreye projenin bakım aşaması girmektedir. Geliştirilen bütün yazılım projelerinde ileride doğabilecek hataları ve yeni istek ve talepleri için bakım sürecinin de yazılım yaşam döngüsüne eklenmesi gerekmektedir. Bu döngüye ***Yazılım Yaşam Döngüsü*** denir.

1.1 Yaşam Döngüsünün Aşamaları

Yazılım yaşam döngüsü modeli 6 aşamadan oluşan bir döngü şeklinde tanımlanır. Bu aşamalar şu şekilde listelenebilir:

1. Planlama
2. Analiz
3. Tasarım
4. Gerçekleştirme
5. Bakım
6. Emeklilik Safhası



Bu 6 aşama Yazılım Yaşam Döngüsünün bütününü tanımlayan kavramlardır. Bir sistemi geliştirmek bazen aylar alırken bazen ise yılları alabilen bir süreçtir. Bu aşamalara birçok insan katılabildiği gibi oldukça da maliyetli bir iştir. Bu aşamalar içerisinde en uzun süren Bakım ve Onarım safhasıdır. Kimi zaman birkaç ay kimi zamansa 20-30 yıl kullanılan sistemlerin düzenli bakımlarının yapılması sistemin doğru bir işleyişte devam edebilmesi için şarttır.

1.1.1 Planlama

Yazılım yaşam döngüsünün ilk safhasıdır. Burada "Ne istiyoruz?" sorusuna yanıt aranır. Bu aşamada temel ihtiyaçlar belirlenir. Maliyetler belirlenir, projenin nasıl başarılı bir şekilde faaliyete geçirilebileceği planlanır.

1.1.2 Analiz

Sistem gereksinimlerinin anlaşılır, kesin ve eksiksiz biçimde belgelenmesi, risk analizinin oluşturulması ve bu dokümanın yazılımın ihtiyaçlarını ortaya koyması gerekir. Müşteriyle en sık iletişime geçilen safhadır. Gereksinim Raporu, müşteriyle yazılımcının resmi düzeyde yaptıkları anlaşmayı belgeler. Bu nedenle, müşteri ya da kullanıcı tarafından çok iyi anlaşılabilmelidir. Bu durumda müşteriye ve kullanıcıya düşen, kendisi için hazırlanan sistemin ne yapacağını rahatça anlayıp onaylayabileceği ya da değişiklik isteyebileceği biçimde ayrıntıların üzerinden gitmektir.

1.1.3 Tasarım

Analiz aşamasında elde edilen detaylar ışığında tasarım aşamasında proje küçük parçalara ayrılır ve proje içerisinde yapılacaklar adım adım planlanır. "İstediğimizi nasıl elde edeceğiz?" sorusunun cevabı bu aşamada aranır. Bir proje planı ve tasarım dokümanı ortaya koyulur. Tasarım dokümanında proje bilgileri (amaç, kapsam vs.), sistem tasarım bilgileri, veri modeli, kullanıcı ara yüz tasarımları, UML Diyagramları gibi içeriklere yer verilir. Tasarım dokümanının amacı, yazılım geliştiricinin yazılımını geliştirirken referans alacağı ve proje sürecinde/sonrasında projeye dahil olacak yeni yazılımcıların projeyi daha kolay anlayabilmesini sağlayacak teknik bir belgelendirmeye sahip olması gerekliliğidir.

1.1.4 Gerçekleştirme

Yazılım geliştirme tamamlandıktan sonra ürün müşteriye teslim edilmeden önce, test ekibi tarafından test aşamasından geçirilir. Bu aşamada ürünün eksik veya hatalı bulunan yanları tespit edilip iyileştirilir.

1.1.5 Bakım

Yazılım ürünü sahaya çıkarılıp müşteriye teslim edildikten sonra bu aşama başlar. Teslim sonrası bakımın 2 çeşidi vardır:

- **İyileştirici Bakım:** Yazılım ürününde sonradan oluşan hata ve kusurların onarıldığı bakım aşamasıdır.
- **Özelliklerin Arttırılması:** Müşterinin istekleri doğrultusunda yazılım ürününe yeni özelliklerin eklendiği bakım aşamasıdır.

1.1.6 Emeklilik Safhası

Vadesini dolduran yazılım emekliye ayrılır. Yani yazılım çalışması durdurulur.

2.YAŞAM DÖNGÜSÜ MODELLERİ

Yazılım yaşam döngüsü birden fazla döngü modelini içerisinde barındırır. Bu modeller şu şekilde listelenebilir:

- Gelişigüzel Yaşam Döngüsü Modeli
- Barok Yaşam Döngüsü Modeli
- Çağlayan Yaşam Döngüsü Modeli
- V Yaşam Döngüsü Modeli
- Spiral Yaşam Döngüsü Modeli
- Evrimsel Yaşam Döngüsü Modeli
- Artırımsal Yaşam Döngüsü Modeli
- Araştırma Tabanlı Yaşam Döngüsü Modeli
- Kodla ve Düzelt Yaşam Döngüsü Modeli
- Çevik Modeller

2.1 Gelişigüzel Yaşam Döngüsü Modeli

Herhangi bir yöntemi yoktur. Kişiye özgü bir modeldir. Genellikle tek kişilik projelerde kullanılır. Kodlaması oldukça basittir ancak takip edilebilirliği ve bakımı zordur. 1960 yıllarında kullanılmıştır.

2.2 Barok Yaşam Döngüsü Modeli

Bu yaşam döngüsü modeli başlıca adımlarının doğrusal bir şekilde geliştirildiği modeldir. Döngü kullanılmaz. Bu sebeple de adımlar arasında geri dönüşler yapılamaz. Ayrıca bu model belgelemeyi ayrı bir süreç olarak işler. Oysaki belgeleme bugün yapılan işin doğal bir süreci olarak ele alınır. Gerçekleştirme aşamasına çok fazla ağırlık vermesinden ötürü günümüzde kullanılması tavsiye edilmez.1970 yıllarında kullanılmıştır.

Yukarıda açıkladığımız 2 yaşam döngüsü modeli günümüzde kullanılmaz. Bu sebeple güncel yaşam döngüsü modelleri içinde yer almadığını söyleyebiliriz.

2.3 Çağlayan (Şelale) Yaşam Döngüsü Modeli
Bu model diğer yaşam modellerinin temelini oluşturan geleneksel bir modeldir. Barok döngü modeliyle karşılaştırıldığında bu süreçte belgeleme daha kısa bir süreç olarak tutulur. Bir adım tamamlanmadan diğer adıma geçilemez. Her adım en az bir kez tekrarlanır. Şelale modeli daha çok iyi tanımlanmış ve üretimi kısa süren projeler için kullanılır. Kullanımı, yönetimi ve anlaması oldukça basittir. Projenin adımları ayrı olduğundan iş bölümü proje başlangıcında bellidir. Bu durum proje yönetimini de kolaylaştırmış olur.
Bu modelin avantajlarının yanında bazı dezavantajları da bulunur.

- Kullanıcı geliştirme sürecinin içinde yer almadığı için olası bir sorun çok geç fark edilebilir bu da yazılım maliyetinin artmasına ve sürecin uzamasına sebebiyet verir.
- Karmaşık ve nesne yönelimli programlar için uygun değildir.
- Model safhalardan oluştuğu için ürün son safhada tamamlanır, gereksinimlerin iyi tanımlanmadığı müşterinin ne istediğinin anlaşılmadığı bir projede bu durum projenin bittikten sonra iptal edilmesine ve başka gerginliklere sebep olmaktadır.

2.4 V Yaşam Döngüsü Modeli
Şelale modelinin gelişmiş hali olarak düşünebiliriz. Bu model belirsizliklerin az olduğu iş tanımlarının ise belirgin olduğu projelerde kullanılır. 'V' harfini andıran modelin sağ tarafı sınama işlemlerinden sol tarafı ise üretim işlemlerinden oluşmaktadır. Her bir üretim aşamasının karşısında bulunan sınama işlemleri sayesinde hata veya kusurlara dönüş yapmak kolaydır. Bu döngü temel olarak 3 modelden oluşur. Kullanıcı Modeli, Mimari Model, Gerçekleştirim Modeli.
Bu modelin dezavantajlarını şu şekilde sıralayabiliriz:

- Adımlar arası tekrarlamaları kullanmaz.
- Risk çözümleme ile ilgili aktiviteleri içermez.

2.5 Spiral (Helezonik) Yaşam Döngü Modeli
Spiral model risk analizi ve prototip üretme konularının özenle üzerinde durur. Risk analizi ön planda olduğu için hataları önceden tespit etmek mümkündür. Prototip oluşturma da her aşamada olduğu için kullanıcı da her aşamada yazılım projesinin bir parçasını görme şansına sahip olur. Bu da sorunların tespit edilmesinde kolaylık sağlar. Planlama, risk analizi, üretim ve kullanıcı değerlendirmesi adında 4 ana başlıktan oluşur. Yaptığı her bir tekrara 'faz' denir. Bu fazlar sırasında risk analizinin ardından kullanıcıdan olumlu ya da olumsuz bir geri dönüş alabilmek için, ürünün prototipi oluşturulur ve kullanıcıya sunulur. Her sunumun ardından alınan geri dönüşlere göre geliştirilmiş planlar ile yeni bir faza başlanır.
Modelin çok fazla avantajı vardır. Kullanıcı katkısının ön planda olması ve bu sayede ürün kullanıcının isteklerine uygun geliştirilebilme şansına sahip olur. Ayrıca hataların erken fark edilmesine olanak sağlayarak olası bütçe aşımları ve maliyet kayıplarını önlemiş olur.
Avantajlarının yanı sıra dezavantajlarını sıralayacak olursak:

- Küçük ve düşük riskli projeler için oldukça maliyetlidir.
- Kullanımı karmaşıktır.
- Çok fazla doküman oluşur.
- Spiral sonsuza gidebilir.

2.6 Evrimsel Yaşam Döngüsü
Müşterinin isteklerinin anlaşılmadığı zamanlarda kullanılır. Müşteri ile ortak bir taslak sistem oluşturulur. Gereksinimlerden en net olanı ile başlanır, müşterinin talepleri doğrultusunda zamanla diğer özelliklerde eklenir. Bu sayede gereksinimi anlamayı kolaylaştırır. Modelin başarısı ilk evrimin başarısına bağlıdır. Diğer modeller ile karşılaştırıldığında ilerleyişi daha yavaştır. Sürekli yapılan değerlendirmeler sonucunda risk ve hatalar en aza indirgenir.
Dezavantajlarını şu şekilde listeleyebiliriz:

- Bakımı zordur.
- Düzenli bir oluşum yoktur.
- Gereksinimleri sürekli yenilemek gerekebilir.
- Sürecin izlenebilirliği yoktur.
- Sürekli bir değişim söz konusu olduğu için yazılımın yapısı hasar görebilir.

2.7 Artırımsal Yaşam Döngüsü Modeli

Eğer bir projede değişikliğe ihtiyaç varsa, bu model bu ihtiyaca ayak uydurur. Uzun zamanlı projeler için eksikler çıktıkça yama yapmak tipindeki projelerde kullanılır. Artırımsal model belli bir zaman dilimi içerisinde yazılımın bölüm bölüm geliştirilip teslim edildiği bir yapıya sahiptir. Bu modelde bir taraftan üretim bir taraftan da kullanım yapılır. Önceki modellerde ürünlerdeki değişiklikler göz önünde bulundurulmaz. Bu model doğal olarak yinelemeli bir yapıya sahiptir. Avantaj ve dezavantajları şu şekildedir:

Avantajları

- Gereksinimlerin önemine göre teslim edilecek artımlar belirlenir.
- Öncelikle en önemli gereksinimleri karşılayan çekirdek bir sistem geliştirilir.
- Projenin başarısız olma riskini azaltır.
- En önemli sistem gereksinimleri daha çok sınanma şansı bulmuş olur.
- Divide and Conquer (Böl ve Yönet) prensibiyle çalışılır.

Dezavantajları

- Deneyimli personel gerektirir.
- İyi tanımlama gerektirir.
- Tekrar yoktur.

2.8 Araştırma Tabanlı Yaşam Döngüsü Modeli

Yap- At tarzı proje olarak da adlandırılır. Çoğunlukla proje ödevleri, yarışma projeleri, TUBITAK, KOSGEB projeleri gibi sistemlerde kullanılır. Proje sonucunda ulaşılacak sonuç belli olmadığından maliyet hesabı da yapılamaz. Projeyle ilgili sonuç dışında pek bir işlevi yoktur. Sınırlı sayıda üretim olur. Bu projeye bir örnek olarak "En hızlı çalışan tek sayı bulma uygulaması" gibi bir örnek verilebilir.

2.9 Kodla ve Düzelt Yaşam Döngüsü Modeli

Küçük yapıdaki programlarda kullanılır. Ürünün direkt gerçekleştirilmesine odaklanılır. Belgeleme yapılmaz. Bu sebeple az bakım gerektirmesine rağmen bakımı zordur.

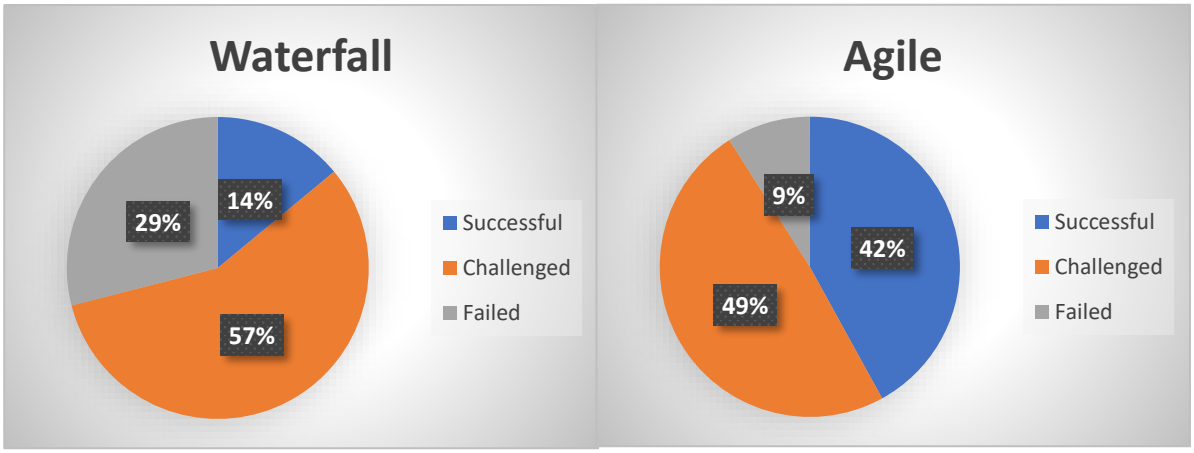
2.10 Çevik Modeller

Yenilemeli geliştirmeyi temel alan bazı yazılım geliştirme metodolojilerine dayanarak harmanlanmış, yenilikçi bir yazılım geliştirme süreçleri topluluğudur. Eski, yavaş ve bürokratik sistemlere tepki olarak doğmuştur. Genellikle takım çalışmalarının, sık denetimlerin, düzenli ürün sunmanın ve müşteri gereksinimleriyle firmanın yeteneklerinin uyumlu bir şekilde bir araya gelerek hızlı ve kaliteli bir şekilde ürün ortaya koyan süreçtir. Takımların kendi kendilerine organize olmaları çok önemlidir. Bunun için takımlar kendilerini değerlendirerek nasıl daha etkili olabileceklerini planlarlar.
Çok sık çıktı üretildiği için beğeniye sunularak müşterinin ihtiyaçlarına göre revize edilir. Kısa sürede müşteri memnuniyeti sağlanır. Fazla eğitim gerektirmez, kolay adapte olunabilir. Değişime açıktır ve esnektir.
Dezavantajları

- Kurumsal bir yapıda uygulamak zordur.
- Sürekli değişsen ihtiyaçlardan dolayı sürekli çalışma gerektirir.
- Ürünün riski ile projenin riski ortak olduğundan kariyer riski doğurur.
- Takımlar üzerinde hedefe ulaşmak için baskı oluşturur.

Günümüzde yaygın olarak kullanılan Çevik Yazılım Metotlarından bazıları:

- Uç Değer Programlama (Extreme Programing — XP)
- SCRUM
- Çevik Tümleşik Süreç (Agile Unified Process -AUP)
- Özellik Güdümlü Geliştirme (Feature-Driven Development — FDD)



3. SCRUM

Scrum, kelime olarak rugby oyununda oluşturulan küçük ekiplere verilen isimdir. Günümüzde oldukça yaygın bir kullanıma sahiptir. Eski yöntemlerdeki gibi projede izlenmesi gereken adımlar belirtilmez, bunun yerine basit ama önemli birkaç kural ile esnek ama üretken bir iş ortaya koyulur.

- Scrum' ın genel olarak çalışma mantığına bakacak olursak öncelikle yapılacak iş **(Product Backlog)** parçalara bölünür.
- Daha sonra işin **(Sprint Backlog)** bir parçası ele alınır ve 2-4 hafta süreyle tamamlanması planlanır. Genelde 4 hafta süren bu programa "**Sprint**" denir.

Klasik metodojilerde kullandığımız sıralı yaklaşımda gereksinim, analiz, tasarım, gerçekleştirme gibi aşamalar vardır. Bu metodolojiler bir aşama bitmeden diğerine geçişe izin vermez. Scrum'ın bize getirdiği **overlapping** (örtüşme) yaklaşımıyla ise tüm sprinte bakıldığında bu 4 aşama bir bütünmüş gibi ilerler. Scrum'da 3 temel kavram vardır. Bunlar; "Roller", "Toplantılar" ve "Araçlar" dır.

3.1 Roller

3.1.1 Ürün Sahibi (Product Owner)

Roller ile başlayacak olursak her projede bir ürün sahibi olmalıdır. Bu kişi müşterinin kendisi olmak zorunda değildir. Müşteri temsilcisi veya scrum ekibinin bir parçası olabilir. Önemli olan bu kişinin müşterinin isteklerini en doğru şekilde anlamasıdır. Bunun için müşteriyle doğru bir iletişim kurmalıdır. Bu kişi projenin içinde yer alır ve ekiple birlikte hareket eder.

3.1.2 Scrum Yöneticisi (Scrum Master)

Scrum yöneticisinin ne olduğundan önce ne olmadığından bahsetmek gerekirse, scrum yöneticisi ekibe görevlerini dağıtan ve emirler veren kişi değildir. Tersine ekibe yardımcı olan, problemleri çözen, takımlarla ve müşteriyle iletişime yardımcı olan takımın üretkenliğinin artmasına yardımcı olan kişidir.

3.1.3 Scrum Ekibi

Birbirleriyle iletişim halinde olan ve programlama kısmını yapan ekiptir. Genellikle 5-10 kişiden oluşur.

3.2 Toplantılar

3.2.1 Sprint Planning

Sprintlerde neler yapılacağı, gereksinimlerin listesi, takımların belirlenmesi, risk analizleri ve maliyet hesaplamaları yapılır.

3.2.2 Daily Scrum Meeting

Tüm sprint boyunca her gün takım üyeleri bir araya gelerek 15 dakika süren ayakta toplantılar yaparlar. Bu toplantılarda önceki gün ne yapıldığı, nasıl sorunlarla karşılaşıldığı, o gün neler yapılacağı hakkında kısa konuşmalar yapılır.

3.2.3 Sprint Review

Her sprint ardından bir Sprint Gözden Geçirme (Sprint Review) yapılır. Yapılanlar gözden geçirilir. Bu toplantılar uzun sürebilir. Ardından bir sonraki sprinte hazırlık yapılır.

3.3 Araçlar

3.3.1 Ürün Gereksinim Dokümanı

Proje boyunca tamamlanması gereken işlerin listesidir. Listeyi product owner yönetir. Kullanıcının… isteğine göre yeni istekler eklenip çıkarılabilir.

3.3.2 Sprint Listesi

Bir sprint turunda takımın yapması gereken işlerin listesidir.

3.3.3 Sprint Kalan Zaman Grafiği (Burndown Chart)

Bir sprint turunda görevler yerine getirildikçe yapılan iş ile kalan iş arasındaki bağıntıyı ortaya çıkaran grafiktir. Belirlen sürenin yeterli olup olamadığı, işin gidişatında bir sorun olup olamadığı ve işin zamanında bitip bitmeyeceği hakkında yöneticilere bilgi veren grafiktir.

3.DÖNGÜ MODELLERİNİN KARŞILAŞTIRILMASI

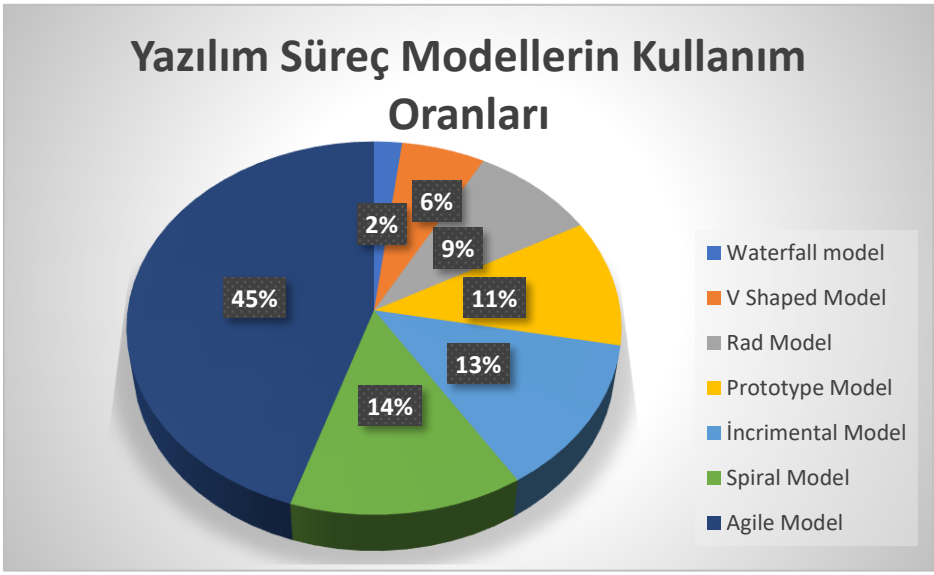
Karşılaştırma sonucu anlaşılacaktır ki yaşam döngü modelleri birbirinden ayrı değildir hatta birlikte kullanılabilir. Yazılım süreç modelleri yazılımın türüne göre seçilmelidir. Projenin özellikleri maliyet, risk analizi gibi faktörlere göre değerlendirilmeli ve elde edilen sonuçlara göre projeye en uygun yaşam döngü modeli seçilmelidir.

Tablo 1: Döngü Modellerinin Karşılaştırılması.

No	Model/Özellik	Kodla ve Düzelt	Çağlayan Modeli	V Modeli	Evrimsel Yaşam Modeli	Spiral Model	Artımlı Süreç	Çevik Modeller
1	**Gereksinim Belirleme**	Başlangıç	Başlangıç	Başlangıç	Başlangıç	Belirli Sıklıkla	Belirli Sıklıkla	Belirli Sıklıkla
2	**Maliyet**	Düşük	Yüksek	Yüksek	Düşük	Maliyetli	Düşük	Çok Yüksek
3	**Başarı Garantisi**	Düşük	Düşük	Orta	-	Yüksek	Yüksek	Çok Yüksek
4	**Uzmanlık Gerekliliği**	Düşük	Orta	Orta	-	Yüksek	Orta	Çok Yüksek
5	**Fazların Örtüşmesi**	Hayır	Hayır	Hayır	Hayır	Hayır	Evet	Evet
6	**Maliyet Kontrolü**	Hayır	Evet	Evet	Evet	Evet	Hayır	Evet
7	**Basitlik**	Basit	Basit	Orta	Karmaşık	Karmaşık	Orta	Karmaşık
8	**Risk Duyarlılığı**	Yüksek	Yüksek	Yüksek Değil	Yüksek	Düşük	Düşük	Azaltılmış
9	**Esneklik**	Katı	Katı	Düşük	Düşük	Çok esnek	Çok Esnek	Çok Esnek
10	**Bakım**	Düşük	Düşük	Düşük	Düşük	Evet	Evet	Evet
11	**Değişiklik Yapma**	Kolay	Zor	Zor	Zor	Kolay	Kolay	Zor
12	**Yeniden Kullanılabilirlik**	Düşük	Düşük	Düşük	Düşük	Mümkün	Mümkün	Evet
13	**Dokümantasyon ve Eğitim Gerekliliği**	Evet	Zorunluluk	Evet	Evet	Evet Ama Çok Değil	Evet Ama Çok Değil	Evet
14	**Zaman**	Çok Uzun	Çok Uzun	Uzun	Uzun	Uzun	Uzun	Kısa
15	**Uygulama**	Kolay	Kolay	Kolay	Kolay	Karmaşık	Kolay	Kolay

Yukarıdaki tabloda döngü modelleri gereksinim belirleme, maliyet, başarı garantisi, basitlik, esneklik, bakım ve daha birçok kritere göre değerlendirilmiştir. Bu tablo ile yaşam döngü modellerinin benzerlik ve farklılıkları, avantaj ve dezavantajları daha net anlaşılacaktır.

Sağ tarafta yazılım döngü modellerinin kullanım oranlarının genel bir çalışması verilmiştir. Bu çalışma genel bir değerlendirme olduğundan ürettiği sonuçlarda genel bir çerçevede değerlendirilmelidir.
Günümüzde en çok kullanılan çevik modeldir. Kısa sürede esnek bir şekilde geliştirilerek yüksek başarımla sonuçlandığından en çok tercih edilen model olması muhtemeldir.



REFERANS

1. Ersoy, E. (2002). *Yaşam Döngüsü Yönetimi Kavramının Yazılım Ürünlerine Uygulanması* (Doctoral dissertation, Fen Bilimleri Enstitüsü).
2. Macit, Y., & Tüzün, E. (2015). Uygulama Yaşam Döngüsü Yönetimi Karşılaştırmalı Süreç İncelemesi. In *UYMS*
3. URL 1- https://stackify.com/what-is-sdlc/ 18.03.2020 tarihinde saat 18.02 de alındı.
4. URL 2- https://www.codex.com.tr/yazilim-gelistirme-modelleri 18.03.2020 tarihinde saat 18.03 de alındı.
5. URL 3- https://fikirjeneratoru.com/yazilim-proje-yonetimi-yontemleri/ 18.03.2020 tarihinde saat 18.04 de alındı
6. URL 4- https://iskulubu.com/yazilim/yazilim-gelistirme-yasam-dongusu/ 18.03.2020 tarihinde saat 18.08 de alındı.
7. URL 5- https://furkanalniak.com/yazilim-muhendisligi-yazilim-surec-modelleri/ 18.03.2020 tarihinde saat 18.08 de alındı.
8. Kılınç, D. (2016). Yazılım Yaşam Döngüsü Temel Aşamaları (Software Development Life Cycle Core Processes) (https://medium.com/@denizkilinc/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-temel-a%C5%9Famalar%C4%B1-software-development-life-cycle-core-processes-197a4b503696) 18.03.2022 tarihinde saat 18.08 de alındı.