Yazılım Yaşam Döngüsü Modelleri

Yazılım Yaşam Döngüsü Nedir?

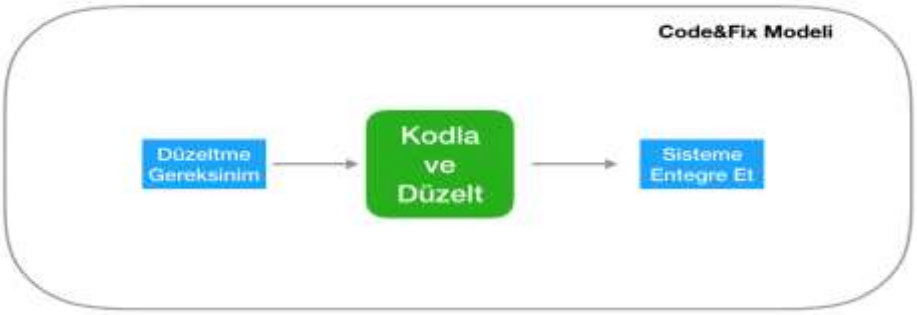
Yazılım yaşam döngüsü(SDLC), yazılımın üretimi ve kullanımını içeren müşterinin isteklerini esas alan yazılım sürecidir. Bu süreç durağan olmadığından gereksinimler değişim gösterir. Örneğin müşteri gereksinimleri sürekli değiştirmek isteyebilir (Değişen Hedef Problemi). Değişimler döngüsel (spiral) şekilde projeye katılır. Yazılım yaşam döngüsü lineer ve tek yöne değildir. Böylelikle maliyet ve zaman kaybı minimuma indirilir Yazılım ürününü insanın evrendeki varlığıyla veyahut su döngüsü ile ilişkilendirebiliriz. İnsan doğar, büyür, ölür ancak kaybolmaz başka bir canlının yaşamına katılır. SDLC de böyledir. Yenileme değişim ve dönüşüm içerisindedir. SDLC' de asıl amaç projeleri üretimini adımlara bölerek uygulamasını kolaylaştırmaktır. Adımların her birinin bitiminde dokümantasyon yapılarak süreç kayıt altına alınır. Yazılım yaşam döngüsü temel adımları:

1- Gereksinim (Requirements): SDLC 'nin en önemli adımıdır. Gereksinimler alınması tüm projenin işleyişinin temelini oluşturur bu nedenle müşteri istekleri iyi dinlenmeli uygulanabilir tarafı müşteriye anlatılmalıdır. Müşteri ile iletişim kaynaklı herhangi bir hata tüm projede maliyet ve zaman kaybına hatta yanlış projelerin oluşmasına neden olabilir. Gereksinim adımda deneyimli kişiler tarafından saha araştırmaları yapılır. İhtiyaçlar belirlenir. Fizibilite çalışmaları yürütülür.
2-Analiz(Analysis): İlk aşamasında topladığımız gereksinimleri ayrıntılı incelediğimiz yazılımın işlevini tahsis ettiğimiz adımdır. ''Bu yazılım projesinde ne yapacağız?'' gibi temel sorular ikinci adımı oluşturur. Aynı zamanda müşterinin kullandığı mevcut sistem göz önüne alarak iyileştirmeyi gerektirir.
3-Tasarım(Design):Bir sonraki aşamanın uygulanması için toplanan ve değerlendirilen gereksinimler ışığında yazılım sisteminin altyapısını oluşturur. Tasarım aşamasının süresi uzadıkça projenin maliyet kaybı azalır. Tasarım ikiye ayrılır:
A- Üst Düzey Tasarım(High-level design): Modüllerin belirlenmesi, isimlendirilmesi, modüller arası ilişkilerin oluşturulması, ara yüz bağlantıları gibi projenin yapısal parçalarının çözülmesidir.
B-Alt Düzey Tasarım(Low-level design): Modül detayları, veri tabanı ve ara yüz detayları gibi yazılımın büyük parçalarının ayrıntılarının çözülmesidir.
4-Gerçekleştirme (Implementation): Yazılım Projesi denildiği zaman akla gelen kısım olan kodlama süreci bu aşamadadır. Bu aşamada, geliştiricilerin kuruluşları tarafından tanımlanan kodlama kurallarına uymaları önemlidir. Kodlama, ilk adımlarda belirlenen programlama dili, geliştirme ortamı ve teknolojiler ile geliştirilir.
5-Test: Test projenin müşteri istekleri doğrultusunda gerçekleşip gerçekleşmediğini ve yazılımdaki hataların tespiti ve düzeltilmesini içerir. Test aşamasında ürün kalite standartlarını karşılayana kadar hatalar rapor edilir, izlenir, düzenlenir, tekrar test edilir.
6-Bakım(Maintenance): Kısmen tamamlanmış projenin müşteri ve kullanıcılara sunularak projenin hatalarının giderilmesi, yeni gereksinimlerinin uygulanması, güncellemelerin gerçekleşmesi durumlarını kapsar. Bakım yazılımın ömrü boyunca devam eden süreçtir. Yeni gereksinimler ortaya çıktıkça bu

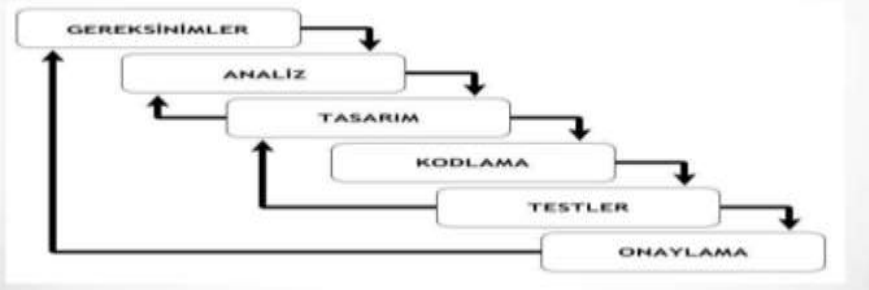
döngü devam eder. Yazılım yaşam döngüsünü etkili kullanmak için yazılım yaşam döngü modelleri ortaya çıkmıştır

Yazılım Yaşam Döngü Modelleri

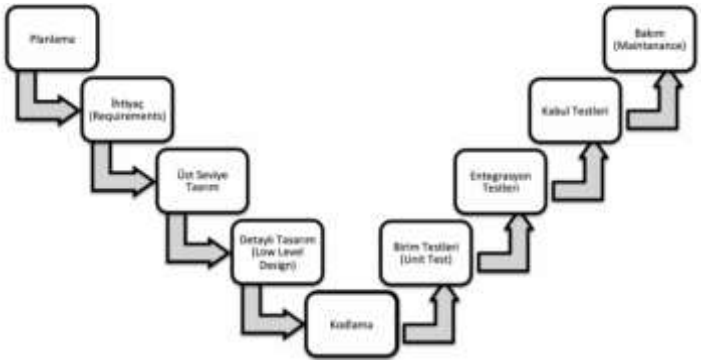
Farklı modellerin ortaya çıkmasının nedeni ihtiyaç çeşitliliğidir. Projenin kitlesi, maliyeti, büyüklüğü, süresi farklılığı projede uygulanacak döngü modelinin değişmesini sağlar.
1-Kodla Ve Düzelt (code and fix): Genellikle az kişiden oluşan resmi olmayan küçük boyutlu projelerde kullanılır. Proje tamamlanana kadar kodla düzelt işlemi sürer. Projeye başlandığı anda oluşturur ve ilk halin üzerine revizeler ile son halini alır. Uygulaması gayet kolaydır ancak hataların anlaşılması zor ve maliyetlidir. Profesyonel hayatta tercih edilmez. Avantajları:
-Planlama yoktur. SDLC aşamaları bir bütün gibi ele alınır. Planlamanın olmaması ve SDLC adımlarının keskin çizgilerle ayrılamaması zaman kazanmayı sağlar.
-Küçük çaplı ve kısa ömürlü projelerde kullanılır.
-Alanında uzman kişilerle çalışmaya gerek yoktur buna bağlı olarak da herkes tarafından kolaylıkla uygulanabilir bir modeldir. Dezavantajları:
-Planlama olmadığından proje bitim tarihi kesin değildir.
- Büyük çaplı projelerde karışıklığa neden olur.
–Dokümantasyon olmadığından hataların ayıklanması zordur. Hatalar bulunsa dahi yüksek maliyete neden olabilir.



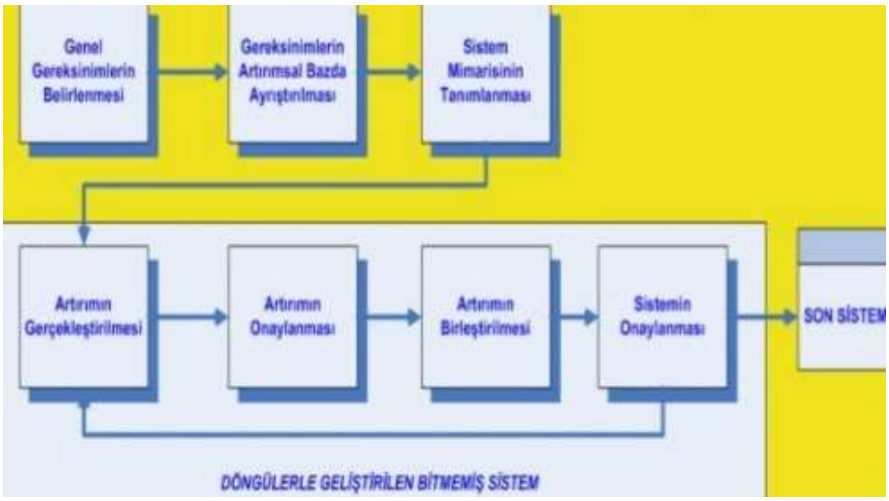
2-Şelale Modeli (Waterfall Model): Proje yönetiminin ilk zamanlarında analiz, tasarım, gerçekleştirme ve testin sıralı olması gerektiği ve bunun dışındaki her modelin riskli olduğu düşünülüyordu. Bu bağlamda şelale modeli ortaya çıkmıştır. Şelale modeli SDLC adımların sırayla ilk düzeyden son düzeye doğru geriye dönüş olmadan ilerleyen ve her adımın dokümantasyonla sonlandığı günümüzde yaygın olarak kullanılmasa da tüm zamanların en çok kullanılan modelidir. Avantajları:
-Proje gerçekleştirme adımları tasarlı olduğundan hata tespiti kolaylıkla yapılabilir. Aynı zamanda proje gerçekleştirme adımları ayrık olduğundan proje ekibi gruplara ayrılarak adımları paylaşabilirler.
–Bitiş ve başlangıç tarihi kesindir.
–Şelale modelinde dokümantasyon adımların sonunda ve zorunludur. Bu nedenle diğer modellerden daha fazla doküman içerir. Dezavantajları:
-Gereksinimler en başta belirlenir. Projenin gereksinimini değiştirmek projeye yeniden başlamayı gerektirir. Bu işlem de maliyeti arttırır.
- Proje bir bütün olarak ele alındığından çıktı tektir. Tek olması hem ekibin hem müşterinin projeden uzaklaşmasına neden olur.
–Şelale modeli sürekli geri dönüşlere izin vermez. Diğer modellerden daha az esnektir.
–Müşteri ekibin bir parçası değildir. Ekibe revize veremez.



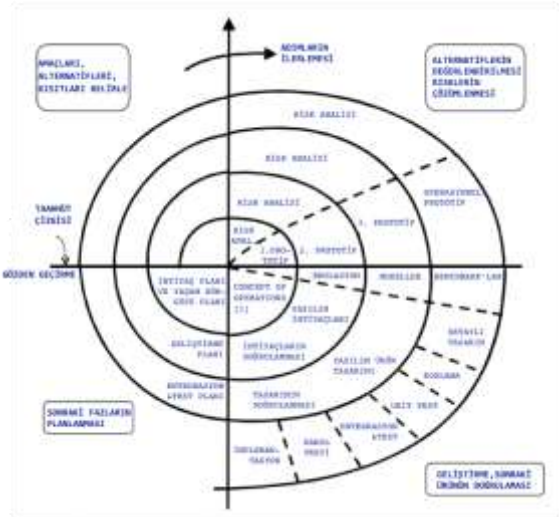
3-V Süreç(Verification and Validation) Modeli: Şelale modeli esas alınarak ortaya çıkmış, gelişmiş bir modeldir. V şeklinin sol kolunda birbirini takip eden adımlara karşılık V şeklinin sağ kolunda test adımları yer alır. Bu sayede her satırdaki adımlar sınanmış olur. Planlama ve ihtiyaç SDLC'nin temel adımlarıdır. Şelale modelindeki gibi planlamanın bitişi ile ihtiyaç başlar. Daha sonra Şelale modelinden farklı olarak ayrıntılı tasarım evresi ardından kodlama yapılır. Kodlama adımından sonra birim test yani üretilen her modülün sınanmasıdır. Devamında entegrasyon adındaki modüllerin uyumlu çalışması üzerine incelemeler yapılır. Kabul testleri müşteriye projenin sunulması ve gereksinimleri karşılayıp karşılamadığını öğrenmektir. Eğer gereksinimleri karşılayan bir proje ise bakıma gönderilir.
Şelale modeline benzer bir diğer yanı geri dönüşlerin kolay olmamasıdır. Gereksinimler bu nedenle doğru ve kesin olmalıdır. Avantajları:
-Aşamalara bölünmesi anlaşılabilirliği arttırdığından proje takibi ve yönetimi kolaydır.
–Dokümantasyon her adım sonunda olduğundan süreç takibi kolaydı. Dezavantaj:
-Aşamalar arası geri dönüşler kolay değildir. İş tanımları bu nedenle doğru ve kesin olmalıdır.
–Esnek değildir.



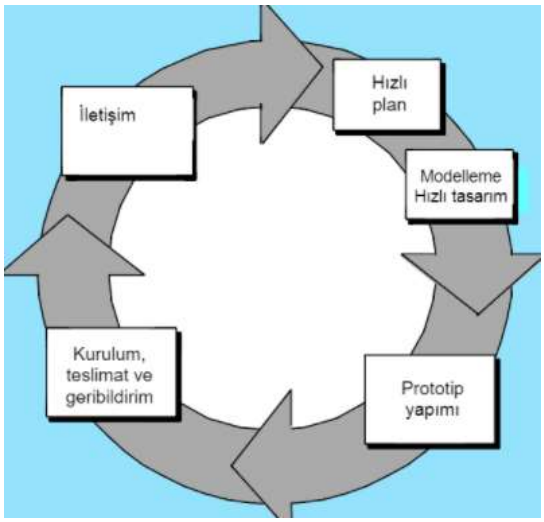
4-Artırımsal Geliştirme Süreç(Incremental Development) Modeli: Artırımsal modelde proje artımlara bölünür ve ilk adımda planlama yapılırken öncelik sırası dikkate alınarak en önemli işlem ilk artım olur. İlk artım geliştirilir ve teslim edilir. Böylelikle teslim edilen artım kullanılırken önem sırasında onun altında olan artım üretilir. Daha sonra üretilen artımlar birleştirilerek yeni bir sürüm ortaya çıkar. Bu sistem değişen müşteri ihtiyaçlarının kolaylıkla projeye dahil edilmesini kolaylaştırır. Artımsal model içerisinde küçük şelale modelleri döngülerinden oluşur. Her döngü bir önceki yazılım sürümü için bakım aşaması görevini görür. Avantajları:
-Artımsal geliştirme kısıtlı sayıda çalışanla gerçekleştirilebilir. Anı zamada yönetilmesi kolaydır.
–Geliştirilen yazılım projesi bir anda ortaya çıkmaz. Proje başlangıcında dahi müşteriye çalışan bir sürüm sunulur. Böylelikle değişiklikler daha az maliyetli olmasını sağlar.
–Bu sistem bize fazlaca test imkanı sunduğundan projenin başarısızlık riski minimumdur.
–Müşteri projenin bir parçasıdır. Dezavantajları:
-Artımları oluşturmak için projenin tamamının anlaşılması gerekmektedir. Artımları doğru şekilde ayırmak zordur. Çünkü gereksinimler değişebilir. Böylelikle artımlar da değişmelidir.
–Artımsal krizlerin oluşması muhtemeldir. Bu yüzden alanında uzman kişilerle çalışılmalıdır.



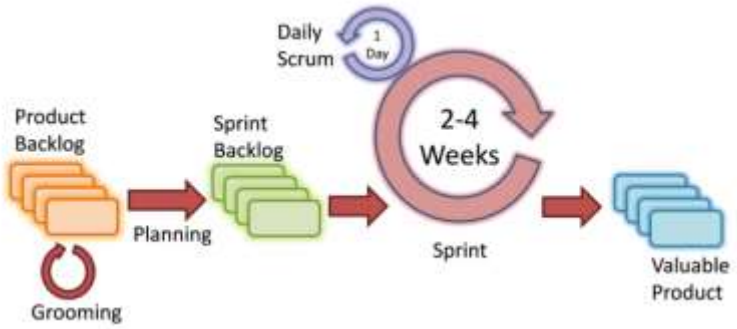
5-Spiral Model: Bu modelde amaç diğer modellerdeki projenin oluşumu sırasındaki hataları minimuma indirmek için sürecin aşamalarına birden fazla sayıda dönüş yaparak ilerleme sağlamaktır.En esnek SDLC modellerindendir. Bu modelde artırımlı modelden ilham alınmıştır. Proje döngülere ayrılır ve her döngünün riskleri ayrı ayrı hesaplanır. Döngü 4 temel adımdan oluşur:
a-Planlama: Amaçların,alternatiflerin, kısıtlamaların belirlenmesi
b-Risk Analizi: Alternatiflerin değerlendirilmesi, risklerin çözümlenmesi
c-Üretim:Geliştirme, sonraki ürünün doğrulaması
d-Müşteri Değerlendirilmesi: mühendislerin sonraki fazları planlaması. Avantajları:
-Gereksinim değişiklikler yaşam döngüsünün ilk aşamasından sonraki aşamalarında ortaya çıkarsa bu sorun başarıyla giderilir.
–Geliştirme küçük parçalara bölünür.En riskli kısımlar ilk gerçekleştirilir.
–Yüksek riskli projelerde kolaylıkla uygulanabilir. Dezavantaj:
-Modelin uygulanması risk kestirme deneyimi gerektirdiğinden yaygın kullanılaz.
–Küçük ve düşük riskli projeler için pahalı bir yöntemdir.
–Karmaşıktır. Döngü sonsuza gidebilir.
–Adım içi adım sayısı fazlalığından dolayı dokümantasyon fazladır.



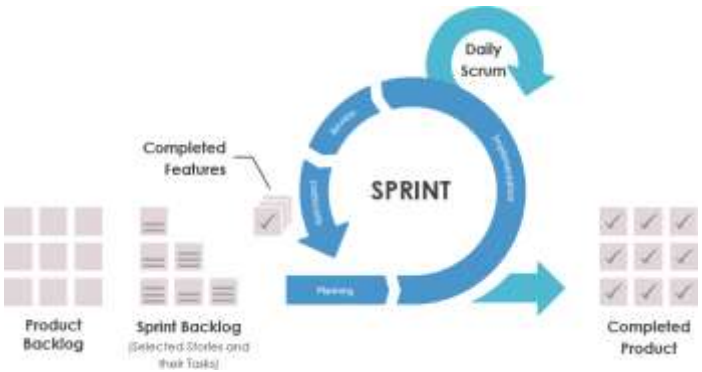
6-Prototip Oluşturma Modeli: Şelale modelinin hız kazandırılmış ve döngüsel halidir.Proje hızlıca yapılan gereksinim analizi ve müşteri isteklerinin dinlenmesinin ardından hızlı plan olarak adlandırılan proje ekibinin çalışmaları nasıl uygulayacakları hangi çıktı üretilecegi konuşulur. Sonra hızlı tasarım ardından prototipleme ile hiçbir modelde oldadığı kadar hızlı bir ürün ortaya çıkar. Bu ürün ilk prototiptir. Müşteri isteklerine göre aynı döngü tekrar gerçekleşir.Böylelikle müşteri ilerlemelere şahitlik ederek projenin içinde tutulur. Müşteri proje içinde kulanıcı etkileriyle yüksek maliyet kayıpları olmadan ürün oluşmasını sağlar.



7-Çevik (Agile) Süreç Modeli: Değişen gereksinimlere kolaylıkla ayak uyduran proje içi iletişimin yüksek olduğu hızlı ürün geliştirme metodudur. Ölçek fark etmeksizin tüm projelerde uygulanabilir. Yinelemeli geliştirmeyi esas alan metod projeyi basit ve küçük parçalara bölerek gerçekleştirir. Yineleme adındaki parça 2-4 hafta arası sürede geliştirilerek müşteriyle buluşturulur. Böylelikle müşteriyle olan güçlü iletişim projeyi olumlu yönde etkilemektedir. Çevik modelin en temek özelliği ekip, müşteri ve kullanıcı ilişkisinin canlı olmasıdır. Böylece hatalar çabucak kontrol edilir ve projeler hızla oluşur. Avantajları:
-Kısa döngüler olduğundan hem programı geliştiren ekip elemanlarını motive eder hem de müşteriyi projenin varlığı konusundaki endişelerinin sona ermesini sağlar.
–Oluşturulan yinelemeler ile müşteri ihtiyaçlarını mükemmele yakın çözer.
–Sürdürülebilir kaliteli geliştirme yöntemidir.
–Projenin planlanması ve geliştirilmesinin bir arada olduğu geliştirme sürecidir. Dezavantajlar:
-Dokümantasyon sürecin sonunda olduğundan doküman çıkarma zordur.
–Sürekli değişen müşteri ihtiyaçlarına ayak uydurmak için gereğinden fazla çalışma.



Çevik model metodlardan oluşur bunlardan bazıları; Extreme Programming, Scrum ve Lean Developmenttır.
a-Extreme Programming(XP): İletişim ve geri dönüşlerin maksimum seviyede olduğu geliştirme modelidir. Dört temel unsur etrafında şekillenir. İletişim, basitlik, cesaret ve geri bildirim. Çevik modelin en önemli özelliği olan iletişim XP'de de yoğun olarak bulunur. İletişim iyi olduğu sürece projede sorunlar erken tespit edilir ve sonuca ulaşır. Yinelemeler gerçekleştirirken en basit parçalarına ayrılarak kolayca halledilir. Müşteri, yönetici ve tüm ekip elemanları 2-4 haftalık yinelemelerle ilgili fikir beyan eder ve geri bildirimler ile proje ilerlemeye devam eder. Projeyi gerçekleştirirken önemli bir diğer şey cesarettir. Başarısızlıktan korkulmamalı aksine çözüm aranmalıdır.
b-Scrum: Gereksinimlerin karışık olduğu yazılım projelerini sprint adı verilen parçalara bölerek uygulamayı kolaylaştıran günümüzde en çok kullanılan modeldir. Scrum müşteri, scrum yöneticisi ve scrum takımından oluşur. Müşteri projede ürünün kendi isteklerine uygun olup olmadığını kontrol eder. Scrum yöneticisi takımı projeye uygun çalışmasını sağlar. Takımlar 5-9 kişiden oluşan ve sürekli iletişim halinde olan bir topluluktur. Scrumlar sprintlerden oluşur sprintlerin çalışmasında da öncesinde sprint planlama toplantısı arada da sprint gözden geçirme yapılır. Sprint planlamada gereksinimler listelenir, riskler değerlendirilir, altyapı ve geliştirme araçları ayarlanır, maliyet hesaplanır. Sprint gözden geçirmelerinde gereksinimlerin ne kadarı uygulandı, yapılanların ne kadarı istenene uygun gözden geçirilir. Bunlar dışında günlük scrum toplantıları ile ekibin 'Dün ne kadar iş yaptı?', 'Bugün ne kadar yapacak?', 'Dün engelleyenler neler?' gibi sorularla geliştirmeye odaklanılır. Bu toplantılar 15 dakika olmalıdır.(30 dakikaya kadar da olabilir. Ancak önerilmez.) Srumın gerçekleştirilmesinde bir diğer etken 'Ürün gereksinim dökümanı'dır. Sürekli değişen gereksinimlerle beraber bu doküman revizeler alır.



Hangi Projede Hangi Modeli Kullanalım?
-Kurumsal olmayan, zaman sınırı olmayan, küçük boyutlu projelerde kodla ve düzelt kullanılabilir.
-Küçük boyutlu ve gereksinimleri iyi tanımlanmış projelerde şelale modeli kullanılabilir
-Belirsizliklerin az olduğu, iş tanımlarının belirgin olduğu bilgi teknolojileri projeleri için V Modeli kullanılabilir.
-Büyük boyuttaki, maliyeti yüksek ve uzun süren projelerde spiral model veya artımsal geliştirme modeli uygundur.
-Orta ve küçük büyüklükte projelerde çevik modeller uygundur.

Kaynaklar:

- https://www.elektrikport.com/teknik-kutuphane/yazilim-gelistirme-yasam-dongusu/11471#ad-image-0
- https://medium.com/@ahmetonol/yaz%C4%B1l%C4%B1m-geli%C5%9Ftirme-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-sdlc-8b39d913986d
- https://online.husson.edu/software-development-cycle/
- https://medium.com/@melsatar/software-development-life-cycle-models-and-methodologies-297cfe616a3a
- 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
- https://www.guru99.com/software-development-life-cycle-tutorial.html
- https://www.techwell.com/2013/02/why-scrum-so-popular
- https://fikirjeneratoru.com/yazilim-proje-yonetimi-yontemleri/
- http://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://polen.itu.edu.tr/bitstream/11527/5973/1/2137.pdf
- http://ybsansiklopedi.com/wp-content/uploads/2014/10/YBS-Ansiklopedi-Ekim-p2-6.pdf
- http://www.ijcait.com/IJCAIT/13/1334.pdf
- https://www.roberthalf.com.au/blog/employers/6-basic-sdlc-methodologies-which-one
- BROOKSHEAR J.G. Bilgisayar Bilimine Giriş. ISBN 978–605–320–361–2.
- Doç. Dr. Deniz Kılınç, Bakırçay Üniversitesi Yazılım Mühendisliğine Giriş Dersi 2. ve 3. Hafta Sunumları