YAZILIM NEDİR?

Sözlük anlamı olarak elektronik bir donanımı belirli bir işi yapması için ayarlanmış komutlar bütünü anlamına gelen yazılım, hayatımızın her anında bizimle birlikte. Otobüste, okulda, iş yerinde, AVM’de hatta öyle ki cebimizde bile bulunmakta. Hayatımız kolaylaştırılması için yazılan bunca yazılımı sadece bir ürün olarak kabul etmek yerinebir canlı gibi düşünebiliriz. Ihtiyaçlarımız değiştikçe gelişen, bizim tarzımıza göre tarz değiştiren, uzun süre kullanmadığımız ve kullanmayacağımızda da yaşamını bitiren bir canlı. Her canlının ir döngü içerisinde olduğu dünyamızda yazılımların döngüsü nasıldır peki? Bu sorunun cevabı ise Yazılım Yaşam Döngüsünde.

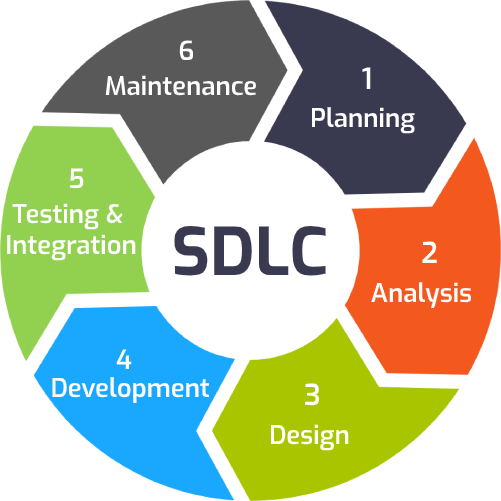
YAZILIM YAŞAM DÖNGÜSÜ (SDLC)

Biz çok farkına varmasakda hayatın her anında aslında bir döngü bulunmaktadır. En basitinden suyun döngüsünü örnek olarak verebiliriz. Yeryüzünde bulunan su çeşitli yollar ile gökyüzüne ulaşır oradan tekrar damla biçiminde yeryüzüne geri döner. İnsan hayatı da döngü halindedir. İnsan doğar, büyür ve ölür. Açıkçası bunca döngünün içerisinde bulunurken ürettiğimiz yazılımların da bir döngü içerisinde olması şaşırılacak bir durum değildir. Peki nedir bu yazılım yaşam döngüleri?

Yazılımın planlanmasından uygulanmasına kadar olan bütün adımlara yazılım yaşam döngüsü (“Software Dvelopment Life Cycle”, “SDLC”) olarak tanımlanır. Bu adımlar genellikle planlama, analiz, tasarım, geliştirme, uygulama ve bakım olarak devam etsede bazı durumlarda bu sayı artabilir veyahut azalabilir. Peki bu adımlarda neler oluyor?

**1.Planlama:** Yazılım için gerekli olan donanım, zaman, maliyet gibi temel ihtiyaçların belirlendiği ve ardından gelen diğer adımların da nasıl bir yol izlenilerek geliştirileceğinin kararlaştırıldığı yazılım yaşam döngüsünün ilk adımıdır.

**2.Analiz:** Bu aşamada yazılımın işlevlerinin karar verildiği, müşteri ile iletişim kurulup müşterinin isteklerinin ve ihtiyacının ne yönde olduğunun karalaştırıldığı ve bunların kağıda aktarıldığı aşamadır. Bu evreden sonra yazılım artık somutlaşmaya başlamıştır.



**3.Tasarım:** Yazılımın ana hatları belirlenir. Müşterinin istek ve ihtiyaçları ile birlikte sürecin nasıl sürdürüleceği, yazılımın özelliklerinin, arayüz tasarımının nasıl olacağının kararlaştırıldığı aşamadır. Bu aşamada use-case ve uml diagramlarının çizimi yapılır.

**4.Geliştirme:** Artık yavaş yavaş kağıt üzerinden bilgisayar ortamına geçilen adımdır. Kodlama bu aşama ile başlar. Genellikle yazılımın ilk testleri de bu aşamada yapılmaktadır.

**5.Uygulama:** Bu adıma ulaşıldığında artık yazılım üstünde son dokunuşların yapılıyor olması ve tamamlanması beklenmektedir. Son hatalar düzeltilir ve yazılım müşteriye teslim edilir. Bu aşamada müşteri ile anlaşmazlığın olmaması için planlama ve analiz aşamalarının iyi şekilde tamamlanmış olmasına dikkat etmeliyiz.

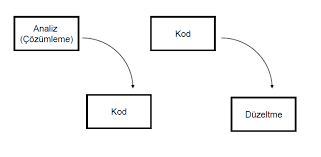
**6.Bakım:** Yazılımınızı teslim ettikten sonra ortaya çıkan sorunların düzeltilmesi için bulunan adımdır.

SDLC nin temel basamaklarını bu şekilde tanımlayabiliriz. Bu basamakları içerisinde bulunduran, yazılımın hangi düzende veya sırayla, nasıl uygulanacağını gösteren yapılara ise **YAZILIM SÜREÇ MODELLERİ** denir.

YAZILIM SÜREÇ MODELLERİ

Dışarıdan bakıldığında aynı süreçlerden geçen yazılımların aslında o kadarda benzer durumlar yaşamadığını görebiliriz. Zaman, maliyet, bilgi gibi bir çok neden birleşince her yazılımın kendine özgü bir döngüsü oluşmaktadır. Bunların sonucunda da birden fazla YAZILIM SÜREÇ MODELLERİ ortaya çıkmakta. Sıkça kullanılan ve bilinen süreç modellerini tanıyalım.

**KODLA VE DÜZELT (CODE AND FIX):** Planlaması olmayan genellikle küçük projelerde kullanılan bir yöntemdir. Planlama, süre, maliyet gibi hesaplamalar çok yapılmaz.



**GELİŞİGÜZEL MODEL:** Basit programlama yöntemidir. Genellikle tek kişilik projelerde kullanılır. İzlenebilirlik ve bakımı zordur. Tam bir yol haritası olmadığı için bir model olarak tanımlamak çok doğru sayılmaz.

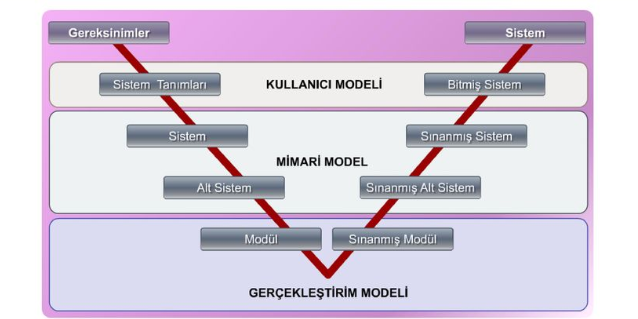
**BAROK MODEL:** Yazılım yaşam döngüsünün temel adımlarının sırasıyla takip edildiği modeldir. Aşamalar arasında gerekli olan geri dönüşler tanımlı değildir. Belgelemeyi ayrı bir süreç olarak ele alıp geliştirme ve testler bittikten sonra gerçekleştirilir. Gerçekleştirme evresine çok önem verildiği için günümüzde kullanımı tavsiye edilmemektedir.

**ŞELALE (WATERFALL) Model:** Geçmiş dönemlerde en popüler model olan şelale modeli en eski, en tanınmış, en temel modeldir. Bu yüzdendir ki **GELENEKSEL YAZILIM GELİŞTİRME MODELİ** olarak da bilinmektedir.bu modelde adımların en az birer kez tekrarlanması gerekmektedir. Analizi çok iyi yapılmış ve az zaman gerektiren projelere uygun olan bu model günümüzde gittikçe kullanımını yitirmektedir. Barok modelinin aksine belgelemeyi ayrı bir aşama olarak görmez, üretimin doğal bir parçasıymış gibi ele alır. Şelale modelinin en önemli özelliklerinden biri de bir adımın tamamlanmadan diğer adıma geçilemez olmasıdır. Tüm aşamalar hiçbir eksik olmadan tamamlanması gereklidir ve hiçbir şekilde önceki aşamalara geri dönüş tavsiye edilmez. Geri dönülmesi gerektiren bir durum olduğunda ise proje sanki sıfırdan başlıyormuş gibi düşünülür. Uzun ve karmaşık projelere uygun değildir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**V ŞEKİLLİ (V-SHAPED) MODEL:** Şelale modelinin daha gelişmiş halidir. Her adımın bir karşılığı bulunmaktadır. En büyük avantajlarından biri proje yönetiminin basit ve kullanımının oldukça basit olmasıdır. Dezavantajı ise aynı zamanda gerçekleştirilebilecek olaylara pek imkan tanımaz ve risk çözümleme aktiviteleri içermez.En belirgin özelliği gereksinimlere ve ürün tanımlamasına net olarak ihtiyacı olması ve bununla beraber test süreçlerinin ön planla çıkmasıdır.V modeli isminden anlaşılacağı üzere V şeklinde devam eden basamaklara sahiptir. V yapıyı ortadan simetrik olarak ayırdığımızda sol bacağı üretim işlemlerini sağ bacağı ise test işlemleri temsil etmektedir. V modelinin 3 temel çıktıya sahiptir. Bunlar sırasıyla “KULLANICI MODELİ, MİMARİ MODEL, GERÇEKLİŞTİRME MODELİ” olarak adlandırılır.



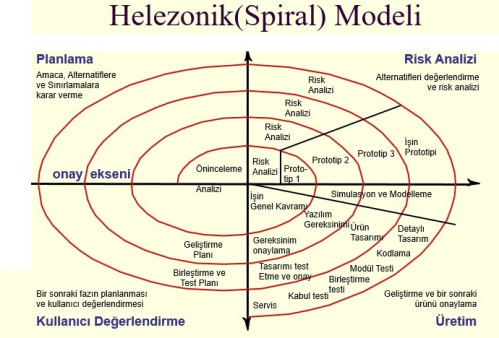
●**Kullanıcı Modeli:** Geliştirme sürecinin kullanıcı ile nasıl bir ilişkisinin olacağının ve sistemin nasıl kabul edileceğine ilişkin belirtilerin ve planların yapıldığı modeldir.

●**Mimari Model:** Sistemin tasarımını ve daha sonradan oluşabilecek alt sistem ile tüm sistemin sınama işlemlerine ilişkin işlemlerin barındırıldığı modeldir.

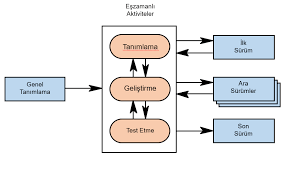
●**Gerçekleştirme Modeli**: Yazılımın kodlanması ve test aşamalarını kapsayan modeldir.

**PROTOTİPLEME (PROTYPING):** Hızlı bir planla sonrasında tasarım ve prototip yapımına geçilir. Gereksinimler değiştikçe veya arttıkça prototip üzerinde değişikliklere gidilir. Müşteri ile sürekli iletişim halinde olunur. Bu sayede yanlış anlaşılmaların önüne geçilmiş olur. Müşteri prototipi test ettikçe onaylar ve proje son haline getirilmeye çalışılır. Belgelendirmeye çok dikkat edilmez bu yüzden hızlı ve karmaşık bir sürece sahiptir. Düzeltme aşamasında yapılan hatalar projenin performansını önemli ölçüde etkiler.

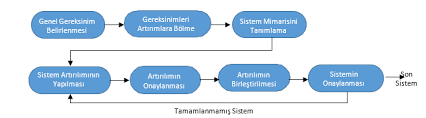
**HELEZONİK (SPİRAL) MODEL:** Planlama, risk analizi, üretim ve kullanıcı değerlendirme adında 4 ana başlığa sahiptir. Planlama yapıldıktan sonra risk analizi ile birlikte prototiplerde oluşmaya başlar. Üretim evresinde yazılımın tasarımı, gereksinimleri, kodlaması ve testleri yapılır. Daha sonrasında ürün kullanıcı değerlendirmesine gider. Çıkan sonuca göre evreler tekrarlanır. Özellikle güvenlik yazılımlarının oluşumunda bu model kullanılmaktadır. Yinelemeli bir artış vardır. Geliştirme parçalara bölünür. Risk analizi ön plandadır. Riske duyarlı olduğu için potansiyel zorlıkları engeller. Hataları daha erken gidermeye odaklanır. Küçük ve düşük riskli projeler için fazla maliyetlidir. Çok fazla belgelemeye ihtiyaç duyar ve karmaşık bir yapıya sahiptir. Sonsuza kadar devam edebilme potansiyeline sahiptir



**EVRİMSEL GELİŞTİRME (EVOLUTIONARY DEVELOPMENT) MODELİ:** Tam ölçekli ilk modeldir. İstek ve gereksinimlerin tam anlaşılmadığı ya da çok fazla alt parçalara sahip olan büyük projelerde kullanılır. Gereksinimlerin ne olduğunun anlaşılmasına yardımcı olur. Sürekli değerlendirmeler yapılıdığı için geliştirme risklerini ve hataları azaltır. Ilk çıkan **“evrim”** egöre projenin uzunluğu, maliyeti değişiklik gösterebilir. Modelin izlenebirliği ve bakımı zordur.



**ARTIMSAL GELİŞTİRME (INCREMENTAL DEVELOPMENT) MODELİ:** Projeyi kısım kısım geliştirip teslim etmeye dayanır. Üretilen her sürüm birbirini kapsayacak ve gideren artan işlev içerecek şeklinde geliştirilir. Uzun zaman isteyen ve sistem eksik olsa dahi çalışabilecek projeler için uygundur. Bu modelde ilk adım olarak genel gereksinimler belirlenir. İkinci adımda gereksinimler daha küçük parçalara bölünür. Daha sonrasında sistem miramirisi tanımlanır. 4.adıma gelindiğinde artırım kodlanmaya başlanır. Kodlama işlemi sonrasında o parçanın onayı alınır. Onay sonrası diğer parçalar ile birleştirilip sistemin onaylanması beklenir. Sistem bitmemiş ve istenilirse 4. adıma dönülür. Her artım sonucu müşteri ile görüşüldüğü için sistemin işlevselliği erken aşamalarda ortaya çıkar. Projenin tamamen iptal edilme riskini azaltır.



**ARAŞTIRMA TABANLI MODEL:** YAP-AT prototipi olarak da bilinmektedir. Belirsizlik üzerine çalışan ortamlarda kullanılmaktadır. Bundan dolayıdır ki yapılan işlerin sonuçları belirgin değildir. Sonuçlarının belirsiz olduğu gibi zaman, maliyet miktarlarının tahmini yapılamaz. Sabit bütçeli işler için uygun değildir. Geliştirilen yazılımlar genellikle sınırlı sayıda kullanılır ve kullanım bittikten sonra işe yaramaz hale gelir.

**ÇEVİK MODELLER (AGILE PROGRAMMING):** Eski, yavaş ve sabit sınırları olan modellere göre daha modern daha esnek ve gelişmiş olan çevik modeler, günümüzde daha fazla uygulanması beklenilen modellerdir. Projenin daha kısa sürede bitmesi ve sonrasında oluşacak problemlere karşın daha kolay bir çözüme ulaşılması için tasarlanmıştır. Klasik yazılım süreç modellerine göre daha rahat bir yol haritası vardır diyebiliriz. Ekip içi iletişime oldukça önem verilmekle beraber ekibin motivasyonunu da maksimum seviyede tutmayı amaçlar. Süreçler ile müşteri arasında sürekli bağlantı bulunmaktadır. Müşteri memnuniyeti için sürekli bir çıktı alınır. Müşteriden gelen geri bildirimler ile proje geliştirilir. Belgeleme oldukça önemli ve fazla sayıdadır.

●**EXTREME PROGRAMING (XP):** En popüler çevik süreçlerden biri olan XP “İyi olanın en iyisini yap” prensibi ile tasarlanmıştır. 4 temel maddesi bulunur bunlar: İletişim, basitlik, geri bildirim ve cesarettir. Xp nin sahip olduğu 4 temel değere bakıldığında günümüzde hala güncelliğini koruyan temel sorunların cevaplarını bulabiliriz.

**○İLETİŞİM:** Başta ekip içerisindeki iletişime büyük önem verilmektedir. Ekip içerisindeki anlaşmazlık projenin uzamasına, bozulmasına hatta bitmesine dahi yol açabilir. XP de böyle bir durumun olması istenilmemekle beraber beklenilmemektedir. İletişimin yüzyüze olmasını isteyen XP sadece ekip içerisinde değil müşteri ile olan iletişime de çok önem vermektedir. Projenin hem daha hızlı hem de çok daha performanslı bir şekilde tamamlanması için iki tarafında birbirini iyi anlaması gerekmektedir.

**○BASİTLİK:** KISS (KEEP IT SIMPLE) yazılım dünyasında önemli bir yeri olan basitlik XP model için ana prensiplerden biridir. Başta yaptığınız proje kodunun okunabilirliğini arttıran bu yöntem, projenin izlenebilirliğini de arttırmaktadır. Eğer detaylı düşünecek olursak, karmaşık bir sorundan çıkmanın ne kadar zor olduğunu çoğumuz bilir. Hangi yönde nasıl gideceğimizi bilememek proje için en kötü seneryolardan biridir. KISS sayesinde daha ufak ve sade sorunlarla karşılaşmamız ve bunlara karşılık olarak daha basit çözümler oluşturmamızın olasılığı fazladır. Bu da hem ekibimizin motivasyonunu hem de projenin geleceğini etkilemektedir.

**○GERİ BİLDİRİM:** Müşteri ile iletişimin ne kadar önemli olduğunu çokca kez tekrar etmiştik. XP modele göre de bu iletişim ana bir prensiptir. Müşteri artık projenin bir üyesi gibidir. Müşteriden aldığınız geri bildirimler sonucunda projenizi şekillendirebilirsiniz. Müşteri ile sürekli iletişim halinde olmanız onlar için de sevindirici bir olaydır.

**○CESARET:** Her olayın ilk ve en büyük adımı olan cesarettir. Her ne olursa olsun pes etmemek, devam edebilmek çok önemlidir. Aksi takdirde başta ekibinizi sonrasında projenizi kötü etkiyelebilirsiniz. Ne olursa olsun çalışmayı bırakmamalı gerekirse, sıfırdan başlayabilmelisiniz.

Extreme Programing bu 4 temel başlığı esas alarak 1999 yılında Kent BECK tarafından bir yazılım geliştirme disiplini olarak ortaya çıkmıştır. Bu evrelere daha kolay adapte olunması için 12 farklı pratik bulunmaktadır. Planlama oyunu, ekipte müşteri, önce test, basit tasarım, çiftli programlama, sürekli entegrasyon, kısa aralıklı sürümler, yeniden yapılandırma, ortak kod sahiplenme, metafor, kodlama standardı, haftada 40 saat.

**●SCRUM:** Günümüzde popülerliğini gittikçe arttran SCRUM modeli kelime anlamı olarak rugby oyunundaki bir saldırı çeşitinden almaktadır. Jeff SUTJERLAND ve Ken SCHAWABER tarafından geliştirilen bir proje yönetim yaklaşımıdır. SCRUM ‘ı diğer modellerden ayıran en büyük özelliği sadece yazılım projeleri için değil bütün projeler için uygulanabilir olmasıdır. “BÖL VE FETHET” mantığı hakimdir. Zor projeleri daha küçük ve kolay projelere bölerek tamamlanması amaçlanır. Karmaşık ortamlarda ve gereksinimlerin tam yapılamadığı projelerde çalışan ekipler için uygundur. 30 günü aşmayan günlük toplantılar sayesinde projenin durumu ve geleceği tasarlanır. SCRUM adını da aldığı spor dalındaki gibi tam bir takım oyunudur. Her oyuncunun bir rolü ve bir görevi vardır. Takım halinde herkes görevini yaptığı takdirde güzel sonuçlar elde edilebilir. SCRUM’ın 3 temel kavramı vardır. Bunlar roller, toplantılar ve araçlardır.



**○ROLLER**

**Ürün Sahibi:** Projede geri dönüşlerden sorumlu olan müşteri veya müşteri temsilcisi.

**Scrum Yöneticisi:** Ekibin SCRUM metoduna bağlı kalmasını sağlayan ve gerektiğinde işleri SCRUM’a adapte eden kişidir. Rugbyde ki takım koçuna benzetebiliriz.

**Scrum Takımı:** Sürekli iletişim halinde olan ekip üyeleri.

**○TOPLANTILAR**

**Koşu (Sprint) Planlama:** Projenin ana hatlarının çizildiği, gereksinimlerin belirlenip iş bölümünün yapıldığı toplantı.

**Koşu (Sprint) Gözden Geçirme:** Ürün kontrolünün yapıldığı toplantılardır.

**Günlük SCRUM Toplantısı:** Günlük 15 dakikalık toplantılarda her üyeden 3 sorunun cevabı beklenir. “Dün ne yaptın?”, “Bugün ne yapacağım?” ve “İşimi engelleyen herhangi bir sorun var mı?” sorun var ise diğer takım üyelerinden yardım istenir.

**○ARAÇLAR**

**Ürün Gereksinim Dokümanı:** Proje boyunca yapılacak işlerin listesi.

**Koşu (Sprint) Dokümanı:** Mevcut sprintte gerekli olan gereksinimlerin olduğu liste.

**Sprint Kalan Zaman Grafiği:** Geçen ve kalan zamanı gösteren grafik

SCRUM NEDEN POPÜLER?

Günümüzün dev şirketlerinden Google,Microsoftunda içinde bulunduğu ve gün geçtikçe artan SCRUM kullanımı,zaman ve maliyet açısından tasarruflu olması,kolaylıkla yeniliklere adapte olabilmesi, karmaşık ve zor görünen projeleri daha kolay bir şekilde sonuçlandırılması, ekip içi iletişimin yüksek olması ve bu sayede hataların daha önceden çözülebilmesi, müşteri ile sürekli temas halinde bulunup anlaşmazlıkların önüne geçilmesi ve değişen gereksinmler karşısında hızlı cevap verebilmesi ile birlikte popülerliğini arttırmaktadır.

NEREDE, NE KULLANILIR?

Yazılım süreç modelleri için bu bunun içindir şu şunun içindir diyerek tamamen ayırmak uygun değildir. Kişinin kendisinin geliştirdiği yöntem belki de onun için en iyi olandır. Çoğu şirkette tek bir yöntem yerine birden fazla modelin birleştirilmesi sonucu oluşturdukları kendi modellerini kullanmaktadır. Fakat yine genel olarak ayırma yapmak istersek şu şekilde yapabilir.

1.Belirsizliklerin az olduğu, gereksinimlerin tam olarak belirlendiği teknolojik projeler için V Şekilli Model

2.Büyük kitlelere ulaşabileceği düşünülen projeler için Evrimsel Geliştirme Modeli

3.Maliyetli, büyük ve uzun süreceüi düşünülen projeler için Helezonik ya da Artımsal Geliştirme Modeli

4.Kısa süreli, orta ve küçük projeler için Çevik Modeller

5.Kişisel, zaman kısıtlaması bulunmayan, küçük projeler için Kodla ve Düzelt Modeli

6.Küçük, özellikleri ve gereksinimleri iyi tanımlanmış projeler için Şelale Modeli

KAPANIŞ

Yazılım dünyasında bir kez olsun karşınıza çıkacak olan yazılım yaşam döngüsü ve süreç modellerini öğrenmek konusunda umarım yardımcı olabilmişimdir. Süreçler nasıl devam ederse etsin pes etmeden devam etmeyi unutmayın. Artık siz de yeni bir döngünün içersindesiniz. “KODLA ve YAŞA” döngüsüne hoşgeldiniz.

M.Furkan İNCE

2021

**KAYNAKÇA**

[**http://ybsansiklopedi.com/wp-content/uploads/2015/08/Yazılım-Geliştirme-Modelleri-Yazılım-Yaşam-DöngüsüSDLCYBS.pdf**](http://ybsansiklopedi.com/wp-content/uploads/2015/08/Yazılım-Geliştirme-Modelleri-Yazılım-Yaşam-DöngüsüSDLCYBS.pdf)

[**https://medium.com/@omerharuncetin/yazılım-yaşam-döngü-modelleri-543c7879a742**](https://medium.com/@omerharuncetin/yazılım-yaşam-döngü-modelleri-543c7879a742)

[**https://medium.com/@denizkilinc/yazılım-yaşam-döngüsü-temel-aşamaları-software-development-life-cycle-core-processes-197a4b503696**](https://medium.com/@denizkilinc/yazılım-yaşam-döngüsü-temel-aşamaları-software-development-life-cycle-core-processes-197a4b503696)

[**https://slideplayer.biz.tr/slide/12386328/**](https://slideplayer.biz.tr/slide/12386328/)

[**https://osmanozaydin.com/yazilim-yasam-dongusu-ve-agile-yazilim-gelistirme/**](https://osmanozaydin.com/yazilim-yasam-dongusu-ve-agile-yazilim-gelistirme/)

[**https://medium.com/@solobasay/yazılım-yaşam-döngüsü-modellemeleri-857af10373bb**](https://medium.com/@solobasay/yazılım-yaşam-döngüsü-modellemeleri-857af10373bb)

[**https://medium.com/@burakguven351999/yazılım-yaşam-döngüsü-nedir-4abd49bf86ea**](https://medium.com/@burakguven351999/yazılım-yaşam-döngüsü-nedir-4abd49bf86ea)

[**https://erhankiyak.medium.com/yazılım-yaşam-döngüsü-modelleri-5f509164e99c**](https://erhankiyak.medium.com/yazılım-yaşam-döngüsü-modelleri-5f509164e99c)

Doç. Dr. Deniz KILINÇ Yazılım Mühendisliği Temelleri ders notları

[**https://www.youtube.com/watch?v=Lk7N1zd7Mv8**](https://www.youtube.com/watch?v=Lk7N1zd7Mv8)