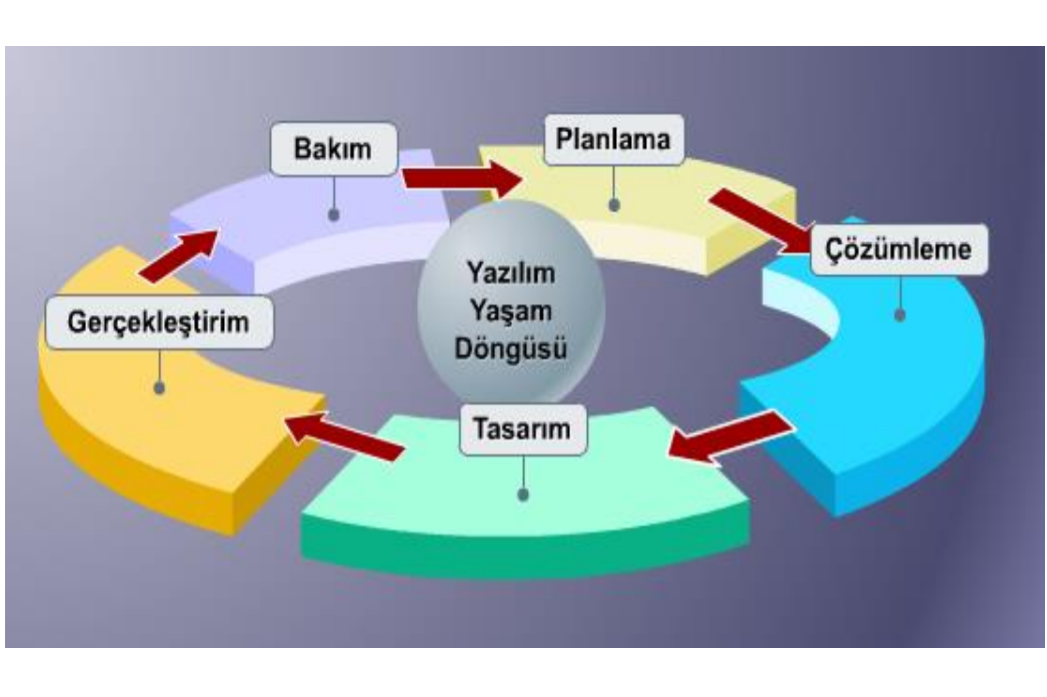
**Yazılım Yaşam Döngüsü ve Modelleri**

**Yazılım Yaşam Döngüsü Nedir ve Neden İhtiyaç Duyarız?**

Bir yazılım ürünü ortaya çıkartmak için sadece yazılım yeterli olmaz. Bunun yanında sürecin planlanması, gereksinimlerin belirlenmesi, gerekli testlerin yapılması, hataların giderilmesi gibi belirli noktaların ele alınması gerekir. Bu süreçler düzenli bir şekilde işlenirse daha verimli ürünler elde edilebilmesi yanında ürünün hazırlanma sürecinin kolaylaşması sağlanır. Bir yazılım ürünü çıkarmak için bu evreleri takip etmemize yazılım yaşam döngüsü denir. Yazılım yaşam döngüsü ürünün hem gelişim hem de kullanım aşamasında mevcuttur.

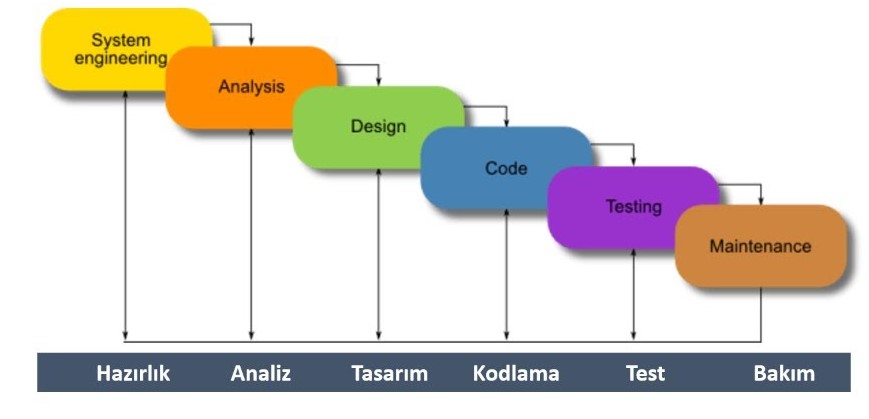
Yazılım yaşam döngüsü aşamaları şunlardır; planlama, analiz, tasarım, gerçekleştirme ve bakım.

1. **Planlama Aşaması:** Bu aşamada müşterinin istekleri doğrultusunda ürünün gereksinimleri belirlenir, fizibilite çalışmaları ve görev dağılımları yapılır. Proje ile ilgili detaylı planlamalar yapılır.
2. **Analiz Aşaması:** Proje gereksinimleri ve işlevleri detaylı olarak incelenir. Proje hakkındaki temel sorunlar ortaya çıkarılır. Bu aşamada hangi programlama dilinin kullanılacağı belirlenir.
3. **Tasarım Aşaması:** Belirlenen gereksinimler doğrultusunda geliştirilen yazılımın temel yapısı oluşturulur. Bu aşamaya verilen zaman arttıkça hata oranının minimuma inmesi beklenir.
4. **Gerçekleştirme Aşaması:** Kodlamanın, testlerin ve kurulumun yapıldığı aşamadır.
5. **Bakım Aşaması:** Projenin ürün olarak sunulduktan sonra ortaya çıkan hataların düzeltildiği, gerekli güncellemeler getirilerek yeni gereksinimlerin karşılandığı aşamadır.

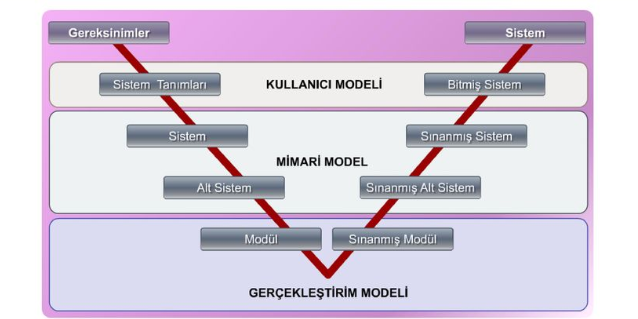
****

**Yazılım Süreci Modelleri**

Bu modeller yazılım geliştirme faaliyetlerinin nasıl yapılacağı ve geliştirme düzenin nasıl olacağı açısından rehber niteliği taşır. Birden fazla yazılım yaşam döngüsü modeli vardır. Bunlara değinecek olursak;

1. **Gelişigüzel Model:** Aslında bu yöntemi model olarak almamız doğru bir kullanım değildir. Çünkü, belirlenmiş bir yol veya model yoktur. Gidişat geliştiren kişiye bağlıdır. İzlenebilirliği ve bakımı oldukça zordur. 1960’lı yıllarda kullanılan bu yöntem basit programlama içerir.
2. **Barok Model:** 1970’li yıllarda ortaya çıkan bu modelde yazılım yaşam döngüsünün temel adımları doğrusal bir şekilde ele alınır ve geliştirilir. Bu modelde belgeleme işlemleri günümüz modellerinden farklı olarak ayrı bir süreç halinde ele alınır. Adımlar arasındaki gerekli geri dönüşlerin nasıl yapılacağı tanımlanmamıştır. Gerçekleştirim evresine çok fazla önem veren bir modeldir. Günümüzde uygulanan bir model olmaktan çıkmıştır.
3. **Çağlayan (Şelale) Modeli:** 1970 yılındaWinston Royce tarafından tanıtılmıştır. Bu model geçmişteki en popüler modeldir. Geleneksel geliştirme modeli olarak da bilinir. Yazılım yaşam döngüsü adımları baştan sona ardışık olarak gerçekleştirilir. Aşamalardan biri tamamen bitmenden öbürüne geçilmez bu şekilde aşamalar arası çakışmalarda önlenmiş ****olur.

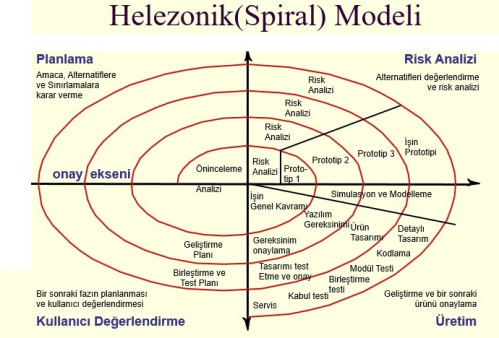
* Üretimi az zaman isteyen ve çok iyi tanımlanmış ürünler için uygundur. Belirsiz gereklilikler yoktur ve gereksinimler çok iyi belgelenmiştir. Sert bir modeldir ve gereksinimleri belirgindir, dolayısıyla kolay yönetilebilir. Barok modelinden farklı olarak belgeleme işlemi ayrı bir süreç olarak ele alınmaz üretimin bir parçasıdır. Her safhada dokümanlar yazılır. Dokümantasyonu ve testi tamamlanmayan bir adım tamamlanmış sayılmaz ve bir adım tamamlanmadan sonraki adıma geçilemez.
* Planlama ve analiz aşamalarında geliştirilecek ürününün tüm gereksinimleri en ince ayrıntısına kadar belirlenir ve belgelenir. Ardından tasarım aşamasında belirlenen gereksinimlerin özellikleri incelenir, gereksinimlerin karşılanması için detaylı çalışmalar yapılır. Bu durum tasarım ve analiz aşamasının çok uzun sürmesine sebebiyet verir.
* Uzun zamana yayılan projelerde gereksinimlerin değişim durumu olabilir. Kullanıcı süreç içerisine dahil edilmez, dolayısıyla ilerleyen dönemlerde yeni gereksinimler ortaya çıkabilir. Bu durumlarda gereksinimlerin karşılanması zordur ve maliyeti çok fazla arttırır.
* Yazılımcılar genellikle direkt kodlama kısmına geçmek ister ve bu kısma geçmek uzun zaman aldığı için moral bozukluğu yaşayabilirler.

1. **V Süreç Modeli:** Şelale modeline benzer özellikler gösterir, sıralı tasarım süreci takip eder. Her aşama bir sonraki aşamaya geçmeden önce tamamlanır. 

Şekilde sol taraftaki kol üretimi, sağ taraftaki ise testi ifade eder. Üretim ve test aşamaları v şeklinde şekildeki gibi birleşir. Bu yüzden ismi V süreç modeli olmuştur.

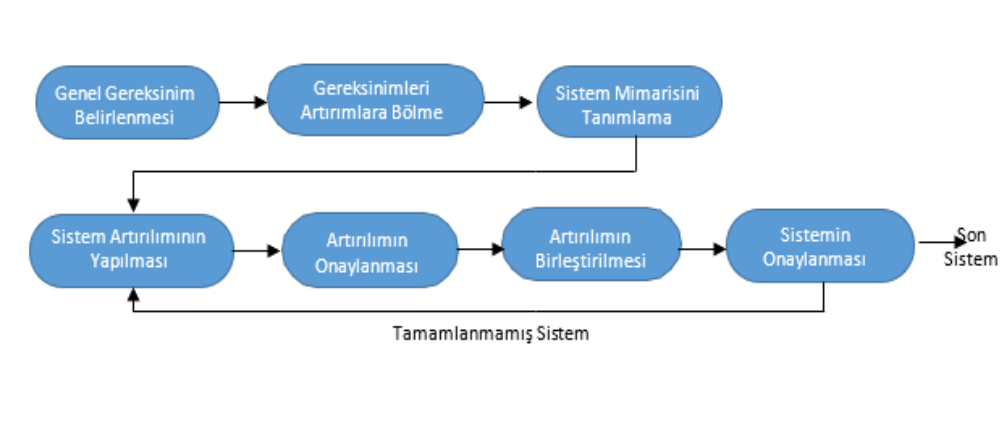
* **Kullanıcı Modeli**: Gereksinimlerin ve beklentilerin tam olarak anlaşılması için müşteri ile iletişim halinde olunur.
* **Sistem Tasarımı**: Kullanıcı gereksinim belgeleri incelenir ve önerilen sistemin işlevleri analiz edilir.
* **Mimari Modeli**: Sistemin tasarımı ve işlevselliği gibi olayların, oluşacak alt sistemlerin test ve entegrasyon aşamalarına ilişkin işlevler.
* **Gerçekleştirim Modeli**: Gereksinimlere göre bir programlama dili belirlenir. Ürün kodlanır. Son yapı daha iyi performans için entegre edilir. Performans kontrolü için ürün birçok kod incelemesinden geçirilir.
* Belirsizliklerin az olduğu ve iş tanımının belirgin olduğu projeler için uygundur. Kullanıcı şelale modelinin tersine kullanıcıyı sürece dahil eder. Küçük ve orta ölçekli projeler için uygundur. Anlaması kolaydır ve kusurların yukarıdan aşağı doğru akışı şelale modeline göre daha rahat önlenir.
* Karmaşık projeler için kullanışlı bir modül değildir. Erken prototipler üretilemez. Projenin ilerleyen dönemlerinde değişiklikler olması gerekirse test belgeleri güncellenmelidir. Aşamalar arası tekrar yoktur ve risk çözüm aktiviteleri içermez.

1. **Helezonik (Spiral) Model:** Risk odaklı bir modeldir. Şelale modelinde ele alınmayan riskler ele alınır. 4 ana aşamadan oluşmaktadır. Bunlar planlama, risk analizi, üretim ve kullanıcı değerlendirmesidir. Bu aşamalar bir spiral şeklinde birbirlerini takip ederler.



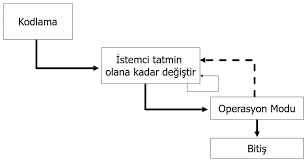
Üretilecek ürün için müşteriden gereksinimler toplanır ve amaç belirlenir. Alternatif çözümler önerilir ve gerekli planlamalar yapılır. Tüm olası çözümler değerlendirilir. Karşılaşa bilinecek riskler belirlenir ve minimize edilir. Ardından belirlenen en iyi çözüm için prototip yapılır. Tanımlanan özellikler test edilir, geliştirilir ve ürünün bir sonraki sürümü çıkarılır. En son ise kullanıcıdan ara ürün için sınama ve değerlendirme alınır.

* Prototip ve yinelemeli artımsal yaklaşımı vardır. Doğrudan gereksinim tanımlanması ve analiz gibi bir yaklaşım yoktur.
* Yazılım kodlaması ve testleri erkenden başlatılır. Kullanıcının aktif olarak sürece dahil olması ileride ortaya çıkabilecek sorunları oldukça azaltır.
* Uzun süreçli bir modüldür ve hata payı küçük olan projeler için fazla masraflıdır.

1. **Artımsal Geliştirme Modeli:** Şelale modeli unsurlarının yinelemeli prototipleme ile birleşmesidir. Ürün her biri ayrı tasarlanmış yapıların birleşmesi ile oluşur. Her bileşen tamamlandıktan sonra bir sonraki bileşen beklenmeden teslim edilir yani parçalar halinde teslim vardır. Her teslim beklenen işlevin bir parçasını karşılar. Ara ürünler her seferinde sistemi geliştirir. Yani yinelemeli bir artış vardır. Ürün bütün olarak teslim edilmediği için eksik işlevlerle çalışabilecek sistemler için uygun bir modeldir. Sistemin tamamlanması uzun bir süreçte gerçekleşir. Bir taraftan üretim devam ederken bir taraftan da kullanım olur.****

* Projenin tamamen batma riski azdır. Diğer geliştirme sistemelrine göre daha kolay test edilir ve hata ayıklanır. Ürün içindeki her bir öğe daha titiz bir şekilde değerlendirilir. İlk ürün teslimi oldukça hızlıdır.
* Ortaya çıkan maliyet müşteri bütçesini aşabilir. Sisteme eklemeler yapıldıkça önceden eklenen prototiplerde görülmeyen yeni sorunlar ortaya çıkabilir.

1. **Kodla ve Düzelt Modeli:** En basit ürün geliştirme modelidir. Kapsamlı bir planlama, somut bir strateji kullanılmaz. Direkt olarak yazılım ürünü gerçekleştirilir. Dokümantasyon kullanılmaz bu sebepten ötürü bakım aşaması olsa bile bakımlar zor gerçekleştirilir. Ürün teslim edildikten sonra bakım aşamasının olmasının yanı sıra emeklilik aşamasında vardır.

****

* Kolay olmasının yanı sıra çok pahalı bir modeldir. Kolay olduğu için tecrübesiz firmalar tarafından da tercih edilmektedir. Kodlama sonrası değişikliklerin maliyeti de oldukça yüksektir.

1. **Evrimsel Geliştirme Metodu:** Yinelemeli ve artımlı modellerin birleşimidir. Yazılım gereksinimleri aşamalı olarak oluşturula bilinen ve teslim edile bilinen modüllere ayrılır. Geliştirmeye çekirdek yani diğer modüllere ihtiyaç duymayan modüllerden başlanır. Her yeni gelen sürüm bir önceki sürümün gelişmiş versiyonu olacak şekilde artımlı ilerleme sağlanır. Müşteriler ürünü aşamalı olarak almak istediğinde kullanılan bir modeldir.

* Büyük projeler için kullanışlıdır. Kullanıcı sürümün tamamı bitmeden ürünü deneme şansına sahip olur. Çekirdek modüller kapsamlı şekilde test edilir buda hata oranını azaltır.
* Süreci müşteri için kabul edile bilinir ve aşamalı olarak uygulana bilinecek şekilde parçalara bölmek zordur.

1. **Prototipleme modeli:** Bu model müşterilerin sistem gereksinimlerini tam olarak bilmediği zaman kullanılır. Ürünün temeli olarak kabul edilen prototiplerin ortaya çıkarılması ve müşteri geri bildirimine göre prototiplerin geliştirilmesi şeklinde ilerler. İşlemler kullanıcı prototipi yeterli buluncaya kadar ve onaylayıncaya kadar devam eder.

* Yeni gereksinimler ve eksik işlevler kolayca karşılana bilinir. Hatalar çok rahat tespit edilir ve bu sayede kalite artar, maliyet düşer. Geliştirilen prototipler ileride daha karmaşık projeler için tekrardan kullanıla bilinir. Esnek bir modeldir. Karmaşık projeleri karşılaya bilen bir modeldir.
* Gereksinimlerde çok fazla değişiklikler olur. Gereksinimler çok fazla değiştiği için yeterli dokümantasyon bulmak zordur. Müşteri tarafından istenilen tüm değişimleri gerçekleştirmek zordur. Müşteri ilk prototipten memnun kalmazsa ürüne karşı ilgisini kaybedebilir.

1. **Çevik geliştirme modeli:** 2001 yılında Kent Beck ve 16 arkadaşı tarafından oluşturulmuştur.Yinelemeli geliştirmeye dayanan bir modeldir. Proje küçük parçalara bölünür ve bu sayede riskler minimize edilir. Yinelemelerin sayısı, kapsamı ve süresi önceden belirlenir. Her yinelemenin sonunda müşteriye projenin ne kadarının gerçekleştiği hakkında bilgi verilir. Esnek ve değişime açıktır bir modeldir. Proje ekibi sürekli olarak iletişim halindedir ve bu durum projenin hızını arttırır. Ekip içerisindeki kaliteli bilgi akışı ve yüz yüze iletişim önemlidir.

Bazı yazılım geliştirme modellerinden bahsedecek olursak;

* **Ekstrem Programlama (XP):** Geri dönüş sayısının çok fazla , grup içi iletişimin önemli olduğu bir modeldir. 4 temel değeri vardır. İletişim, basitlik, geri bildirim, cesaret.

**İletişim:** İletişimler yüz yüze gerçekleştirilir. Yazılım ekibi ile kullanıcılar arasında sıkı iletişim esası vardır. Bu sayede sorunlar erken safhalarda fark edilir. Kullanıcılardan bilgi alınması gerekildiğinde proje aksatılmadan gerekli bilgiler alınır ve yazılım gelişme hızı kesilmez.

**Basitlik:** Gereksinimleri karşılayan en basit çözümler kullanılır. Bu mantıktan dolayı karmaşık çözümlere uygun bir model değildir.

**Geri Bildirim:** Müşterilerde proje grubunun bir üyesidir. Müşteriler ve yazılım ekibi belli zamanlarda bir araya gelir ve yazılımın geldiği noktayı değerlendirir. Bu sayede ileride ortaya çıkabilecek anlaşmazlıkların önüne geçilir.

**Cesaret:** 4 temel değerin en zoru cesarettir. Projenin gelişimi açısından oldukça önemlidir. Sorunlardan ve yapılan yanlışlardan kaçmak yerine üzerlerine gitmek ve çözümlerini aramak esas alınır. Yapılan çalışmadan memnun kalınmazsa sil baştan yapıla bilinmelidir. Başarısızlıklardan korkmak sürecin yavaşlamasına sebebiyet verir.

* **Scrum:** Scrum bir proje yönetimi yaklaşımıdır. Yazılım dışındaki alanlarda da kullanılır. Karmaşık problemlerin sprint denilen parçalara ayrılarak çözümünü sağlar. Günlük 15 dakikalık ayak üstü toplantılar ile takibi sürdürülür. Scrum 3 elemandan oluşur. Bunlar roller, toplantılar ve bileşenlerdir.

**Roller:** Ürün sahibi, scrum takımı, scrum yöneticilerinden oluşur.

Ürün sahibi ürün artışlarını kabul veya reddeder, gelişmeye devam edilip edilmeyeceğine karar verir yani projenin geri dönüş sorumlusudur.

Scrum takımı 5-9 kişiden oluşan, sürekli iletişim halinde olan takımdır.

Scrum yöneticileri, Takımı scruma adapte eder, ekibi dikkat dağıtıcı unsurlardan korur, scrum eserlerini görünür kılar. Takım üzerinde bir yönetim yetkisi yoktur yani koordinatör değildir.

**Toplantılar:** Her sprintin öncesindegeniş kapsamlı gereksinimlerin çıkarılması, takımların belirlenmesi, risk değerlendirmesi için sprint planlama toplantısı yapılır. Toplantının sonunda teslim edilecek ürünün özelliklerinin listesi ve başarı hedeflerinin kısa bir açıklaması yapılır.

Sprint gözden geçirme toplantılarında, Her sprintin sonunda elde edilen artışın değerlendirilmesi yapılır. Nihai ürün belirlenen kriterlere göre değerlendirilir.

Her gün ayak üstü 15 dakikalık herkesin gün içerisinde proje hakkında neler yaptıklarını anlattığı toplantılar yapılır. Ekip üyeleri, dün ne yaptım? Bu gün ne yapacağım? Önümdeki engeller ve karşılaşacağım sorunlar neler? Sorularını yanıtlar.

**Bileşenler:** Projenin gereksinimlerinin basit bir listesi çıkarılır ve buna ürün gereksinim dokümanı denir. Geçerli ve kullanışlı olabilmesi için sürekli bakımı yapılır. Listeye sonradan gereksinim eklemesi veya çıkarılması yapılabilir. Bu liste genellikle kullanıcı hikayelerinden oluşur.

Mevcut Sprintin iş ve görevlerini kapsayan dokümana sprint dokümanı denir.

Sprintin ne kadarının yapıldığının ve ilerlemenin zaman açısından nasıl gerçekleştiğinin anlaşılması için sprint kalan zaman grafiği kullanılır.

**Scrum Günümüzde Neden Popüler ?**

Scrum günümüzde en çok tercih edilen modeldir. Bunun sebeplerine değinecek olursak;

* İş birliği ve iletişimi ön plana çıkartarak daha doğru ve başarılı ürünler elde edilir.
* Karmaşık ve gereksinimleri tam belirlenememiş projeler için uygundur.
* Maliyeti düşüktür ve entegre edilmiş takım çalışmaları sayesinden hızlı ilerler.
* Kullanıcılarla sürekli iletişim halinde olunur buda hata oranını azaltır.
* Yinelene bilir bir modeldir.
* Büyük ihtiyaçlar küçük parçalara bölünerek giderilir yani böl ve fethet prensibini esas alır.

**Hangi Projede Hangi Modeli Kullanmalıyız ?**

* Gereksinimlerin ve iş tanımının net olarak belirlendiği, belirsizliklerin az olduğu durumlarda V modeli tercih edile bilinir. Eğer küçük bir proje ise şelale modeli de uygundur.
* Ekip çalışmasından ziyade bireysel çalışmaların yapıldığı küçük projelerde kodla ve düzelt modeli uygundur.
* Gereksinimleri tam belirlenememiş, kapsamlı, karmaşık ve orta-kısa büyüklükte uzun sürmeyen projelerde çevik modeller uygundur.
* Maliyetli ve büyük projelerde spiral veya artımsal modeller uygundur.
* Coğrafi olarak geniş alanlara yayılmış, geniş kitleye sahip organizasyonlar için önerilen modeldir.

Kaynaklar:

* <http://www.volkantay.com/2020/11/02/yazilim-yasam-dongusu/>
* <https://www.elektrikport.com/teknik-kutuphane/yazilim-gelistirme-yasam-dongusu/11471#ad-image-0>
* <https://medium.com/@omerharuncetin/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BC-modelleri-543c7879a742>
* <https://muhendisportali.com/yazilim-yasam-dongusu/>
* <http://muhammetbaykara.com/wp-content/uploads/2020/03/2.pdf>
* <https://www.codex.com.tr/yazilim-gelistirme-modelleri>
* <https://teknoliva.com/yazilim-gelistirme/>
* <https://systemanalize.wordpress.com/gelistirme-metotolari/>
* <https://fikirjeneratoru.com/yazilim-proje-yonetimi-yontemleri/>
* <https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm>
* <https://www.guru99.com/what-is-sdlc-or-waterfall-model.html>
* <https://www.javatpoint.com/software-engineering-v-model>
* <https://www.geeksforgeeks.org/software-engineering-spiral-model/>
* <https://en.wikipedia.org/wiki/Incremental_build_model>
* <https://productcoalition.com/the-code-and-fix-model-2cabd4c48166>
* <https://www.geeksforgeeks.org/software-engineering-evolutionary-model/>
* <https://www.geektonight.com/evolutionary-model-software-engineering/>
* <https://www.geeksforgeeks.org/software-engineering-prototyping-model/>
* <https://www.quora.com/Why-is-the-Scrum-process-so-popular-in-the-software-industry>
* <http://www.ilkimdilara.com/scrum-meetings/>
* Doç. Dr. Deniz Kılınç, Bakırçay üniversitesi, Yazılım Mühendisliği Temelleri dersi hafta 3-4 ders notları.

Medium linki: https://turgutefe-aksit.medium.com/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-ve-modelleri-19a92b0154be

Turgutefe Akşit

200601005

Bilgisayar Mühendisliği