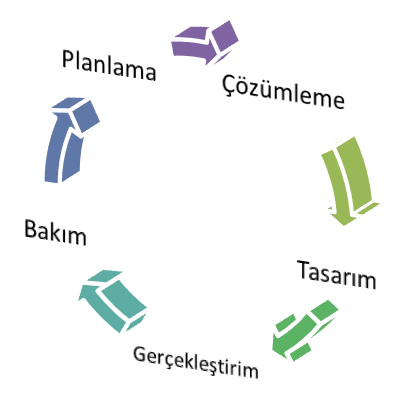
Buket UĞURLU- 210601028

**YAZILIM YAŞAM DÖNGÜ MODELLERİ**

**Yazılım Yaşam Döngüsü Nedir?**

Yazılım yaşam döngüsü, bir yazılım projesinin planlamasından başlanılarak hayata geçirilmesini ve sonrasını kapsayan bir süreçtir. Bu döngü sayesinde yazılımın gerçekleştirilme süreci daha planlı bir şekilde işletilir. Bu aşamalar sırasıyla şu şekildedir: Planlama, Analiz, Tasarım, Gerçekleştirim, Bakım.

****

**Planlama:** İlk aşama gereksinim analizinin yapılmasıyla başlar. Burada müşterinin yazılımdan ne istediği, beklentisinin ne yönde olduğu öğrenilir ve iyice anlaşılır. Buna göre yazılımın üretim süreci için plan ve fizibilite çalışmalarına başlanılır.

**Analiz:** Elde edilen gereksinimlerin detaylı bir şekilde incelenmesi gerçekleştirilir ve buna göre dokümantasyon oluşturulur. UML diyagramları oluşturulur. Projenin riski ve gerçekleştirilmesinin ne kadar süreceği hesaplanır.

**Tasarım:** Bu aşamada yazılımın kodlanmasından ziyade nasıl oluşturulacağı belirlenir. Tasarım mantıksal ve fiziksel olmak üzere ikiye ayrılır. Mantıksal tasarımda önerilen sistem ele alınırken fiziksel tasarımda ise yazılımın detayları incelenir.

**Gerçekleştirim:** Burada projenin kodlanır, test edilir ve kurulumu gerçekleştirilir. Bütün işlemler önceki aşamalarda kararlaştırılan programlama dili, geliştirme ortamları vb. detaylara dikkat edilerek yapılır.

**Bakım:** Artık yazılım tamamlanmıştır ve hayata geçirilmiştir. Ortaya çıkan hatalar veya yazılıma sonradan eklenen yeniliklerin yapıldığı aşamadır. Burada kullanıcılardan geri dönüşlerin alınması yazılım iyileştirilmesi, daha kullanışlı hale gelmesi açısından oldukça önemlidir. Bakım aşaması yazılım var olduğu sürece devam eder.

**Yazılım Yaşam Döngü Modelleri**

Projelerin geliştirilmesi süreci için pek çok model bulunmaktadır. Bu modeller yazılımın gelişim sürecinin aşama aşama takip edilebilmesini sağlar. Geliştiriciler için kolaylık sağlar. Gelişigüzel, Barok, Çağlayan, V Süreç, Spiral (Helezonik), Artımsal Geliştirme Süreç, Kodla ve Düzelt, Evrimsel Geliştirme, Formal Sistem Geliştirme Modelleri örnek olarak gösterilebilir.

1. **Gelişigüzel Model**

Tam olarak bir yöntem olduğu söylenemez. Geliştirici genellikle tek kişidir. Adından da anlaşılabileceği gibi projenin yönetimi karmaşıktır. Bu modelde yazılımın iyileştirilmesi veya bakımı için çalışma yapmak oldukça güçtür. Günümüzde geçerliliği yoktur.

1. **Barok Modeli**

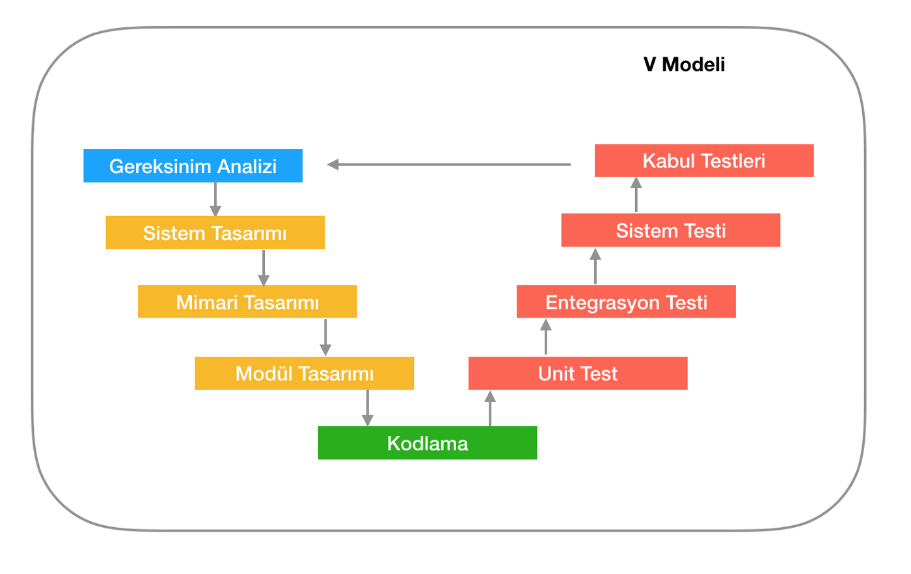
Barok modeli de tıpkı gelişigüzel modeldeki gibi günümüzde pek sıklıkla kullanılmaktadır. Yazılım yaşam döngüsü aşamalarının hepsini kapsar. Buna ek olarak dokümantasyon için ayrı bir aşama yer almaktadır. Daha çok gerçekleştirim aşamasına özellikle de ürünün test edilmesine yoğunluk veren bir modeldir.

1. **Çağlayan Modeli**

Projenin her aşaması bir basamak olarak görülür. Aşamalar mükemmel olarak tamamlanana dek tekrar tekrar gerçekleştirilir. Bir basamak tamamlanmadan sonrakine geçilmez. Bu özelliğiyle Barok modeline göre daha kullanışlıdır. Genellikle projeler sözleşmelerinde bir temel görevi görür. Modelin yönetilmesi basittir. Daha küçük çaplı projelerde kullanılması elverişlidir. Kullanıcı ürünün son aşamasına kadar ürünü göremez. Kullanıcının da sürece dahil edilmesi yazılımın verimi açısından önemlidir bu konuda dezavantajlıdır. Gereksinimlerin çok iyi anlaşılmasını gerektirir. Bu modelin günümüzde kullanımı azalmaktadır.



1. **V Süreç Modeli**



Bu modelde sol taraf yazılımın üretim aşamasıdır. Sağ taraf ise ürünün test edilmesini içerir. Bu modelin üç çıktısı bulunmaktadır: Kullanıcı Modeli, Mimari Model ve Gerçekleştirim Modelidir.

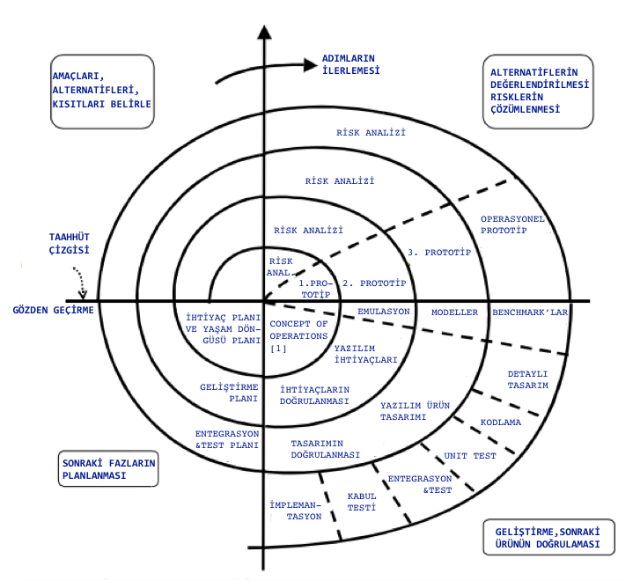
* Kullanıcı Modeli: Gereksinim dokümanının anlaşılır. Kullanıcıyla bir uyuşmazlık yaşanırsa bu aşamada çözüm bulunur.
* Mimari Model: Yazılımın mimarisi belirlenir. Arayüz, veri tabanı tabloları, mimari diyagramlar gerçekleştirilir.
* Gerçekleştirim Modeli: Yazılım modülleri kodlanır ve test edilir.

V Modeli önceki modellere oranla kullanıcının projeye katılımını arttırır. Proje yönetimi ve kullanımı kolaydır. Fakat risk çözümlemeleri ve aşamaların tekrarı açısından yetersizdir.

1. **Spiral Model**

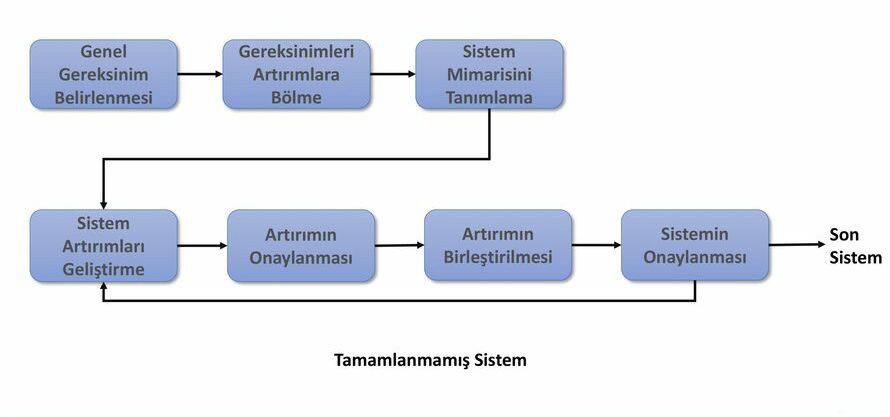
Yazılım yaşam döngüsünün ve prototip modelinin avantajlı özelliklerinin birleşimine benzetebiliriz. Planlama, risk analizi, üretim ve kullanıcı değerlendirmesi safhalarından oluşmaktadır. V modeline oranla risk analizinin yapılmasına daha çok ağırlık verilmiştir. Spiralin ilk en içteki kısmında projeyle ilgili yapılmak istenenler tanımlanır ve gerekli analizler yapılır. Eğer risk analizi yapılırken istenmeyen bir hatayla karşılaşılırsa bununla ilgili düzenlemeler üretim aşamasında gerçekleştirilir.

Büyük çaplı projelerin yönetimi için çok iyidir. Risk değerlendirilmesinin yapılması için uygundur. Ancak risk analizinin yapılması uzmanlık gerektirdiğinden modelin uygulanmasında deneyime ihtiyaç vardır. Müşteriyle etkileşimin hızlı olması projeden istenilen sonucun alınmasına katkı sağlar.

****

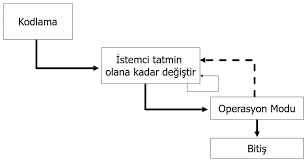
1. **Artırımsal Geliştirme Süreç Modeli**

Çağlayan yaşam döngüsünün daha işlevsel haline benzetebiliriz. Yazılımın her aşaması diğerini kapsayacak şekilde oluşturulmuştur. Her aşama sonunda bir ara ürün oluşur. Gereksinimlerin öncelik sırasına göre artımlar belirlenir. Çağlayan modelinde istenen özelliklerin bütün basamaklarda gerçekleştirilmesi beklenirken artırımsal modelde yazılım fonksiyonel birimlere ayrılarak uygulanır. Yani böl ve yönet yaklaşımı kullanılır. Bu sayede yazılımın geliştirilme sürecinin detaylı bir şekilde incelenmesi sağlanır. Büyük çaplı projelerin başarısız olma olasılığını düşürür. Yazılımın kullanımının ve üretiminin aynı anda gerçekleştirilmesine olanak sağlamaktadır. Müşterinin ihtiyaçlarının karşılanması bakımından oldukça başarılı bir modeldir.



1. **Kodla ve Düzelt Modeli**

Aslında belli başlı bir model yoktur. Müşteri gereksinimleri doğrultusunda yazılımın kodlaması başlanılır ve süreç içerisinde gereksinimler değiştikçe yazılımda tekrar baştan değiştirilir. Bu kodun daha fazla karmaşıklaşmasına sebep olur. Küçük çaplı ve tek kişili geliştiriciler için biraz daha uygun olduğunu söyleyebiliriz. Yine de projenin yapımı çok uzun sürebileceği için projenin maliyeti çok fazladır.



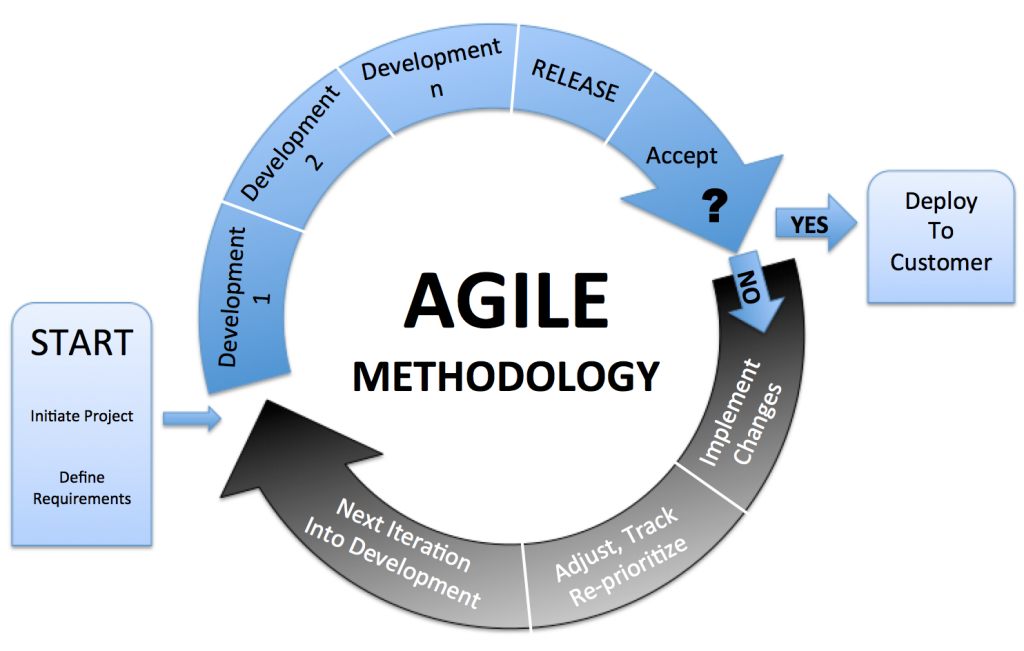
Şekilde de görmüş olduğunuz gibi herhangi bir tasarım, analiz aşaması içermemektedir. Geliştirilme sürecinde müşteriden bir geri dönüş alınmadığı oldukça kullanışsızdır.

**ÇEVİK YAZILIM GELİŞTİRME NEDİR?**

Son yıllarda teknolojinin de hızla gelişimiyle beraber proje yönetimine dair yeni metotlara ihtiyaç olmuştur. Projenin daha hızlı gerçekleştirebilmesi, müşterinin değişen gereksinimlerini daha kolay karşılayabilmesi, büyük proje ekiplerinin yönetilebilmesini sağlamak amacıyla çevik (agile) yöntemi ortaya çıkmıştır.

Bu yöntem ilk defa 1970’li yıllarda ortaya atılmıştır. Çevik yöntemin temeli ilk defa Winston W. Royce’un

“Büyük Yazılım Sistemlerinin Gelişimini Yönetmek” makalesinde atılmıştır. Bu makalede Şelale modeli, yinelemeli model ve çevik modellerini proje yönetim modelleri olarak tanımlamıştır.

****

Çevik yöntemlerin başlıca amacı zaman, maliyet ve kaynak tüketimini minimuma indirmektir. Sadelik anlayışı vardır. Müşteri memnuniyeti son derece önemlidir. Bu yüzden yazılımın sürekli geliştirilmesi ve müşterinin de bu gelişim aşamalarının bir parçası olması sağlanır. Projenin geliştirilme sürecinde ekibin arasındaki iletişimin yüz yüze olması ve ekibin kendi kendini organize edebilmesi son derece önemlidir. Bu ekibin motivasyonunu da önemli ölçüde arttıracaktır.

Çevik yazılım geliştirme metotlarından yaygın olarak kullanılan birkaç model: Extreme Programming (XP), Dynamic System Development Methodology (DSDM), Adaptive Software Development (ASD), Feature-Driven Development (FDD), Kristal, Scrum örnek gösterilebilir.

1. **Scrum**

Scrum, Ken Schawaber tarafından geliştirilmiş bir çevik metodolojidir. Şu an günümüzde en popüler ve güncel olan yöntemdir. Modern projelerin daha karmaşık hale gelmesi ve değişkenlerin olması aslında Scrum’ın neden bu kadar popüler olmasını açıklıyor.

metin, iş kartı, ekran görüntüsü, küçük resim içeren bir resim

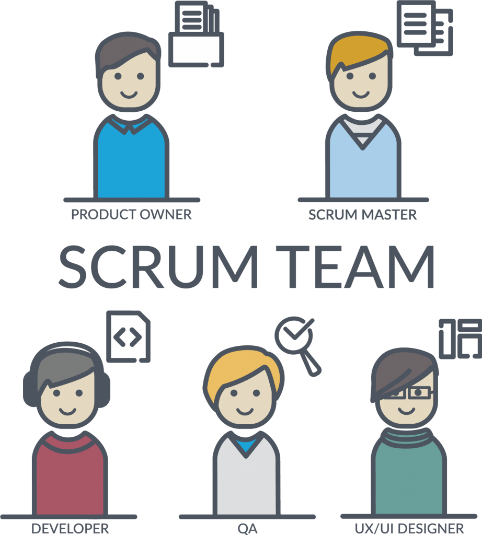
Açıklama otomatik olarak oluşturuldu

Scrum’ın temel unsurları değerleridir (scrum values). Bu değerler doğrultusunda scrum yürütülür.

* Cesaret (Courage)
* Odak (Focus)
* Taahhüt (Commitment)
* Saygı (Respect)
* Açıklık (Opennes)

Bu yöntemde çalışan yazılımın hızlı bir şekilde ve sık sık müşteriye ulaştırılması sağlanarak müşteri memnuniyeti sağlanır. Projenin sürecinin küçük zaman dilimlerine bölünmesine sprint denmektedir. Bu sayede gereksinimler son aşamada değişse bile projeye müdahale kolaylaşır. En başta da bahsettiğimiz gibi ekip üyeleri arasındaki iletişimin yüz yüze olması oldukça önemlidir. Ekibin tüm üyeleri fiziksel olarak her gün bir arada çalışmalıdır. İletişimin yüz yüze olmasıyla amaçlanan aslında ekibin üyeleri aslında güven ortamı yaratmaktır. Bu sayede ekip kendi kendini motive edebilecek ve daha sağlıklı bir çalışma ortamı yaratılacaktır. Sağlam bir mimari ve süreç tasarımının olmasına özen gösterilir. Scrum’da planlanan çalışmalar 24 saat, 30 gün veya aylar süren bir sürece yayılır.

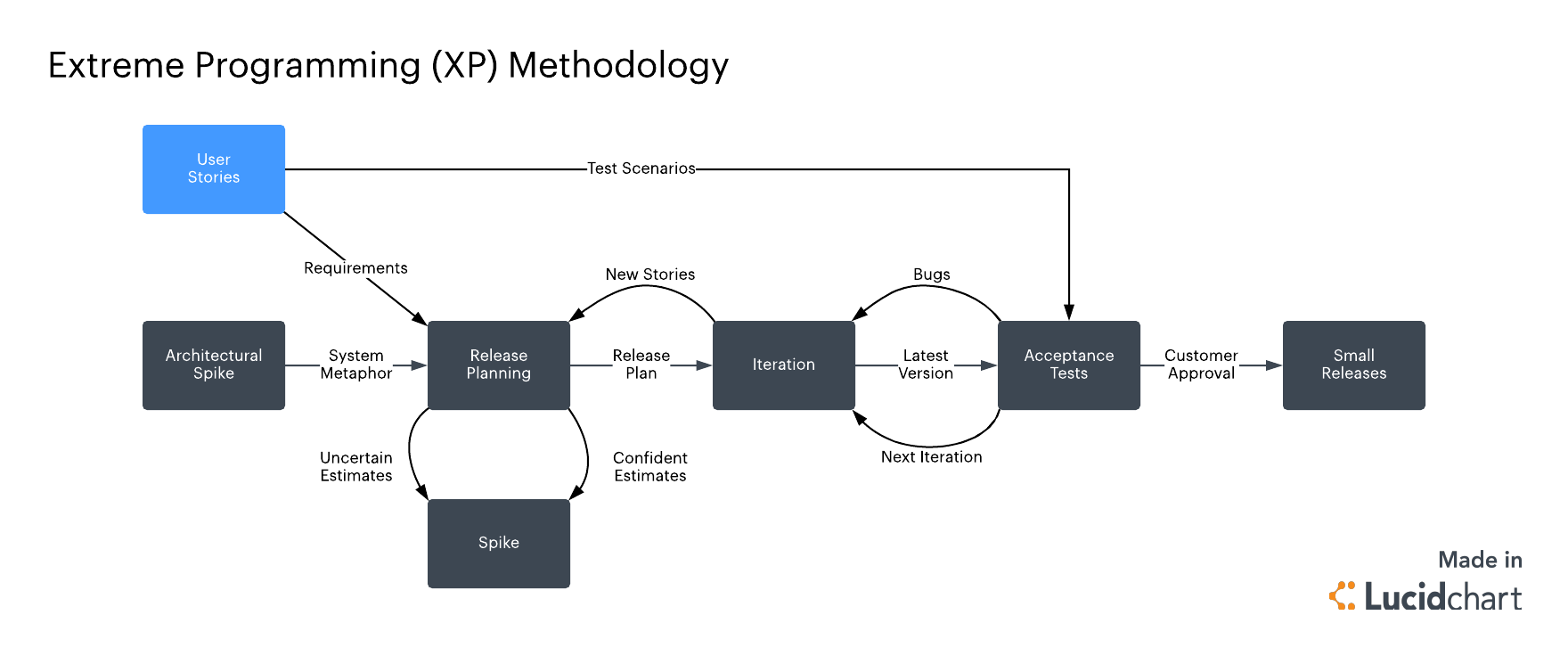
Scrum’da günlük toplantılar ve sprint değerlendirmeleri önemlidir. Bu toplantılarda ekibin daha verimli nasıl çalışabileceği ve bir sonraki sprintte neler yapılacağı planlanır. Geçmişteki hatalara bakılarak çıkarımlar yapılır. Bu sayede gelecekte aynı hataların tekrarlanmaması ve daha etkin bir çalışma yapılması hedeflenir. Toplantılar genellikle 15 dakika gibi az bir zaman diliminde gerçekleştirilir. Bu toplantılarda 3 soruya odaklanılır: “Dün ne yaptım, bugün ne yapacağım, beni engelleyen bir sorun var mı?”



Scrum Master nedir? Ekibin içerisinde Scrum’a en hâkim olan, bildikleriyle diğer ekip üyelerine de yardımcı olan ve rehberlik eden kişidir. Scrum master toplantıdaki sorulara verilen cevapları iyi dinlemelidir. Bir sorunla karşılaşan ekip üyesine yardımcı olmalı ve bir takım halinde çalıştıklarını üyelere hissettirmelidir.

1. **Extreme Programming (XP)**

Extreme Programming, Kent Beck tarafından bulunan ve günümüzde oldukça popüler yöntemlerden birisidir. Basitlik (Simplicity), iletişim (Communication), geri dönüm (Feedback) ve cesaret (courage) değerleri üzerine kuruludur. Planlama oyunu, metafor, kısa aralıklarla yeni sürüm, ayakta toplantı (stand-up meeting), geriye bakış, ortak sorumluluk, sürekli entegrasyon, kod standartları, kalıcı tempo, test etme, sade tasarım, yeniden yapılandırma, eşli programlama olmak üzere on beş teknikten oluşmaktadır.



Extreme programming dört aşamadan meydana gelir.

* Keşif Aşaması: Bu aşamada müşterinin hikayesi oluşturulur. Programın teknik altyapısı için gerekli deney ve araştırmalar yapılır.
* Planlama Aşaması: Müşterinin hikayesinin öncelik sırasından yola çıkılarak iterasyon için planlamalar oluşturulur. Gerçekleştirimin ne kadar süreceğiyle ilgili gerekli bilgi müşteriye verilir. Eğer gerçekleştirimin ne kadar süreceği belirsizse basit bir çözüm getirilmeye çalışılır. Gerçek implementasyon için gerekli süre hesaplanmaya çalışılır.
* İterasyon ve Sürüm Aşaması: Kullanıcının hikayesinin implementasyonu bu aşamada gerçekleştirilir. Ürünün testi gerçekleştirilir. İterasyon sonunda müşteriden feedback alınır. Hata oluşmuşsa hataların değerlendirilmesi gerçekleşir.
* Bakım Aşaması: Projenin bitiminden sonra yazılımın bakımı yapılır. Müşterinin varsa yeni istekleri implemente edilir. Yazılımın sürümü güncellenir.

**KAYNAKÇA**

* <https://tr.wikipedia.org/wiki/V-Model_(Yaz%C4%B1l%C4%B1m_geli%C5%9Ftirme)>
* Fırat Üniversitesi Yazılım Mühendisliği Bölümü YMT 312 Yazılım Tasarımı ve Mimarisi, 2019 <https://akademiksunum.com/index.jsp?modul=document&folder=a93e3a2fccf8eb56a557c55c5f0d5cf10789abe2>
* <https://www.codex.com.tr/yazilim-gelistirme-modelleri>
* <https://osmanozaydin.com/yazilim-yasam-dongusu-ve-agile-yazilim-gelistirme/>
* <https://iskulubu.com/yazilim/yazilim-gelistirme-yasam-dongusu/>
* <https://caglartelef.com/yazilim-yasam-dongusu/>
* <https://toptalent.co/agile-nedir-agile-metodu-nasil-uygulanir>
* <https://nttdata-solutions.com/tr/local-blog/agile-cagimizin-proje-yonetimi-metodolojisi/>
* <https://tr.wikipedia.org/wiki/Winston_W._Royce>
* <https://batuhanakpunar.medium.com/b%C3%B6l%C3%BCm-11-sprint-nedir-nas%C4%B1l-y%C3%B6netilir-%EF%B8%8F-912ae2472adc>
* KARLIDERE Turgay, KALIPSIZ Oya, Yazılım Mühendisliği Projelerinde Çevik Yaklaşımların Yeri, 2003
* DÖNMEZ Ercan, Çevik Yazılım Geliştirme Sürecinde Kritik Başarı Faktörlerinin Belirlenmesi ve Önceliklendirilmesine Yönelik Bir Örnek Çalışma, 2009
* ERSOY Elif, Yaşam Döngüsü Yönetimi Kavramının Yazılım Ürünlerine Uygulanması, Kasım 2002

<https://github.com/buketugurlu>

<https://medium.com/@buketugurlu>

<https://www.linkedin.com/in/buket-u%C4%9Furlu-aa4482223/>