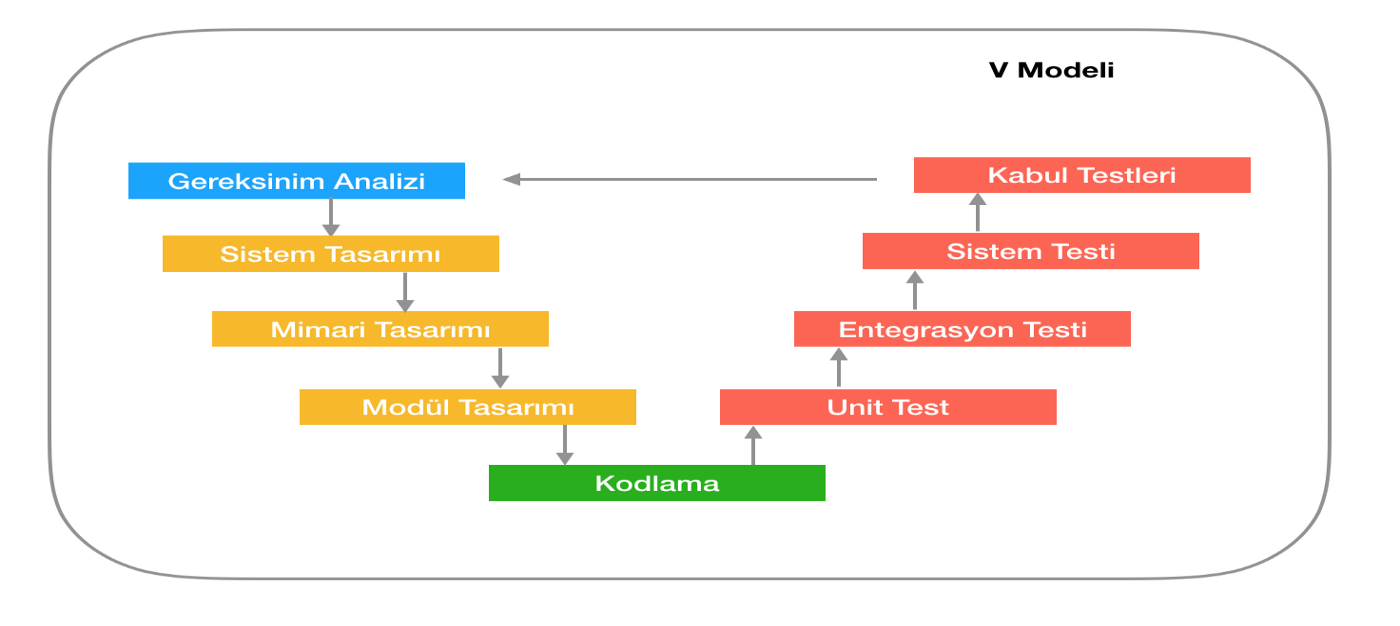
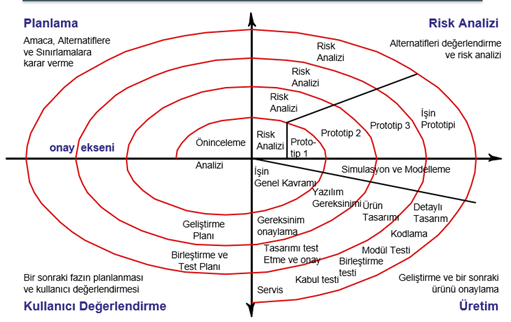
210601057  
Mehmet Enes ARSLAN **Giriş:**  
 Geçmişten günümüze geliştiriciler belirli metot veya modeller geliştirerek yaptıkları programların geliştirme süreçlerini belirli bir plan veya döngü haline getirmişlerdir. Bu süreçte kullanılan modeller de gelişmiş ve etkili olan yöntemler daha çok öne çıkarken kullanışsız olan yöntemler zamanla kullanılmayıp unutulmuştur.   
 Yazılım yaşam döngüsü herhangi bir yazılımın üretim ve kullanım aşaması birlikte olmak üzere geçirdiği tüm aşamalar olarak tanımlanır.  
 **Yazılım yaşam döngüsünün temel adımları:**  
1.Gereksinim:  
Müşteri gereksinimlerinin elde edildiği ve fizibilite çalışmasının yapıldığı aşamadır.  
2.Analiz:  
Sistem gereksinimlerinin ve işlevlerinin ayrıntılı olarak incelendiği aşamadır. Var olan işler incelenir ve temel sorunlar ortaya çıkarılır.  
3.Tasarım: Belirlenen gereksinimlere cevap verecek yazılım sisteminin temel yapısının oluşturulduğu aşamadır.  
4.Gerçekleştirme:  
Modüllerin kodlandığı birleştirildiği test edildiği ve kurulum çalışmalarının yapıldığı aşamadır.  
5.Bakım:  
Hata giderme ve yeni gereksinimlerin eklendiği aşamadır.  
  
1. Gelişigüzel model:   
Herhangi bir plan veya düzenden bağımsız olarak geliştiricinin ilgi duyduğu yönlerde yapılan bir geliştirme şeklidir. Genellikle tek kişilik geliştirme ortamı vardır. Eskiden kullanıldığı bilinen bu yöntem günümüzde nadiren tercih edilmektedir. Bunun sebebi yazılan kodun takibi ve yapılan bir hatada onarılması çok zordur.   
2. Barok modeli:   
 Bu modelin aşamaları inceleme, çözümleme, tasarım, kodlama, modül testleri, alt sistem testleri, sistem testleri belgeleme ve kurulumdur. 70’li yıllardan kalma bir modeldir. Bu model belgelemeyi ayrı bir süreç olarak ele alır. Bugünlerde ise belgeleme yapılan işte doğal bir ürün olarak görünmektedir. Bu modelin aşamaları doğrusal olarak uygulandığı için aşamalar arasında geçiş veya döngü söz konusu değildir. Gerçekleştirme evresine daha çok ağırlık veren bir model olduğundan bugün kullanılması tavsiye edilmez.   
3. Çağlayan yaşam-döngüsü modeli:  
 Bu modelin başlıca aşamaları gereksinimlerin tanımlanması, analiz yapılması, tasarımın oluşturulması, kodlama yapılması, testler ve bakımdır. Bu aşamaları açıklamak gerekirse:  
 1- Gereksinimlerin tanımlanması:   
 Projenin yapılabilmesi için gerekenlerin tanımlandığı, personel ve donanım açısından yeterliliğin ölçüldüğü, yapılan çalışmalar sonucunda bir yol haritasının oluşturulduğu aşamadır.  
 2- Analiz:   
 Bu aşamada öncelikle mevcut sistemde varolan işler incelenir. Sonrasında yazılımla ilgili daha ayrıntılı işlevler ve gereksinimler çıkartılır. Bunu yapmaktaki asıl amaç mevcut yapıdaki işlerin bir yazılım mühendisi bakış açısıyla oluşturulmasıdır. Sorunun ne kadar doğru algılanıp algılanmadığını belirten bir aşamadır.  
 3- Tasarım:  
 Tanımlanan gereksinimlere ve yapılan analizlere karşılık verecek çalışmaların yapılmaya başlar. Yapılan çalışmalar yazılım ya da bilgi sistemin temel yapısını oluşturan çalışmalardır. Bu çalışmalar mantıksal tasarım ve fiziksel tasarım olmak üzere iki gruba ayrılır. Mantıksal tasarımda mevcut sistemin dışında önerilen başka bir sistemin yapısı anlatılır. Olası değişiklikler önerir. Fiziksel tasarımda ise yazılımı içeren bileşenler ve bunların ayrıntılarını içerir.  
 4- Kodlama:  
 Tasarım aşaması biten projede kodlama yapılmaya başlar. Kodlama önceki aşamalarda elde edilen bilgi ve donanım kullanılarak yapılır.   
 5- Testler:  
 Yazılan kodun test edilerek oluşan hataların giderilmesi sağlanır. Yapılan testlerde limitleri zorlayarak yazılan kodun dayanıklılığını da ölçebiliriz. Yeni oluşan ihtiyaç durumlarında eklemeler yapabiliriz.   
 6- Bakım:  
 Bakım süreci yazılan kodun ömrü boyunca bitmeyen bir süreçtir.   
**Çağlayan modelinin olumlu yanları:**  
 Kullanımı ve anlaşılması kolaydır. Yönetimi kolaydır. Projenin safhaları ayrı olduğundan iş bölümü ve iş planı projenin en başında net bir şekilde bellidir. Bu durum projenin yönetimini de oldukça kolay hale getirir. Çağlayan modeli çok küçük ve gereksinimleri iyi anlaşılmış projelerde iyi çalışır.  
 **Çağlayan modelinin eksiklikleri:**  
 Gerçek yaşamdaki projelerin çok azı yenileme gerektirmez. Yazılımın kullanıcıya ulaşma süresi çok uzundur. Gereksinim tanımları çoğu kez net olarak yapılmaz. Bu konudaki eksiklik ve yanlışın ortaya çıkma zamanı gerçekleştirim sonrasına rastlar. Bu durumda yanlışların düzeltilmesi ya da giderilmesinin maliyeti çok yüksektir. Çağlayan model ise söz konusu işleri üretim aşamasının sonunda öngörmekte, bu aşamaya kadar yapılması gereken işler teknik personel için çoğu kez önemsiz olarak görülebilmektedir. Bu nedenle bu model ile yapılan üretimlerde yazılım ekipleri mutsuz olmakta ve kod yazma dışında olan ve iş yükünün büyük bir bölümünü kapsayan kısma yeterli önemi vermemektedir. Üst düzey yönetimlerin hedeflenen ürünü görme süresinin uzun oluşu projede sorunlara yol açmaktadır.  
  
4.V süreç modeli:  
 Çağlayan modelinin gelişmiş hali olarak düşünebiliriz. İzlenilen aşamalar doğrusal gitmek yerine, kodlama evresinden sonra aşamalar yukarıya doğru ilerlemeye başlar. Bu ilerleyiş V şeklini oluşturduğundan V süreç modeli olarak bilinir. Modelin aşamaları gereksinim analizi, sistem tasarımı, mimari tasarım, modül tasarımı, birim test etme, entegrasyon test etme, sistem test etme, kullanıcı kabul test etmedir.  
 **Doğrulama Evresi:** 1. Gereksinim analizi:  
 Gereksinim analizinde ilk evre doğrulamadır. Sistemin gereksinimleri kullanıcıların isteklerine göre analiz edilerek toplanır. Bu aşamada yapması zorunlu olan işlemlerin kurgulanmasıyla ilgilenilir. Ancak yazılımın nasıl tasarlanacağı ve inşa edileceğini belirmez. Kullanıcıyla görüşülür ve kullanıcı gereksinim dokümanı oluşturulur.   
 Hem yazılımsal hem de donanımsal gereksinimleri toplamak için farklı yollar vardır, bunlar görüşmeler, anketler, doküman analizleri, incelemeler, tek kullanımlık prototipler, kullanıcı senaryoları, statik ve dinamik incelemelerdir.   
 2. Sistem tasarımı:   
 Sistem tasarımı sistem mühendislerinin analiz ettiği ve kullanıcı gereksinim dokümanlarını çalışarak sundukları sistemin işleyişini anladıkları evredir. Olası durumları kullanıcı gereksinimlerini kullanarak anlamaya çalışırlar. Eğer herhangi bir gereksinim uyuşmazlığı varsa, kullanıcı bununla ilgili olarak bilgilendirilir.  
  Çözüm bulunur ve kullanıcı gereksinim dokümanı düzenlenir.   
 3. Mimari tasarım:  
 Seçilen mimaride temel tipik olarak içereceği modüllerin listesi, her bir modülün özet fonksiyonelliğini, arayüz ilişkisini, bağımlılıklarını, veri tabanı tablolarını, mimari diyagramlarını, teknoloji detaylarını sunmalıdır. Entegrasyon test etme tasarımı özel bir evre içinde gerçekleştirilir.  
 4.Modül tasarımı:    
 Tasarım sistemi daha küçük birimlere veya modüllere ayrılır ve her biri programcıya doğrudan kodlamaya başlayacak şekilde açıklanır.   
-Veritabanı tabloları, tüm elemanları ile birlikte, tiplerini ve boyutlarını içerir.  
-Tüm arayüz ayrıntıları karmaşık API referanslarıyla birlikte  
-Tüm bağımlılık konuları  
-Hata mesaj listesi  
-Eksiksiz girdi ve çıktılar  
Birim test tasarımı bu evrede gerçekleşir.  
 **Geçerli Kılma Evresi:**  
 V modeli içerisinde doğrulama evresinin her bir seviyesi geçerleme evresindeki her bir seviye ye karşılık gelmektedir. Bazıları diğer isimleriyle de bilinir.  
1. Birim Test Etme:  
 Birim test planları modül tasarım evresi boyunca geliştirilir. Bu planlar birim seviyesindeki veya kod seviyesindeki hataları ortadan kaldırmak için yürütülür. Birim testi kodun/birimin geri kalanından izole edildiğinde düzgünce işleyebilen en küçük varlığı doğrular.   
2. Entegrasyon test etme:  
 Entegrasyon test planları mimari tasarım evresi boyunca devam eder. Bu testler birimler arasında haberleşebilen ve aynı zamanda birbirinden bağımsız bir şekilde hareket eden birimlerden oluşur. Test sonuçları müşteriler ile paylaşılır.  
3. Sistem test etme:  
 Sistem test planları sistem tasarım evresi boyunca geliştirilmeye devam eder. Birim ve entegrasyon test planlarına benzemez. Sistem testi uygulama geliştirimi beklentilerinin karşılandığından emin olur. Tüm uygulama işlevselliği, merkezi bağımsızlığı ve iletişimi test edilir.  Sistem testi işlevsel ve işlevsel olmayan gereksinimlerin karşılandığını kontrol eder ve doğrular. Yükleme ve performans test etme, stres test etme, regresyon test etme gibi alt kümeler sistem testi içindir.  
4. Kullanıcı kabul test etme:  
 Kullanıcı kabul testi planları gereksinim analiz evresi boyunca geliştirilir. Test planları iş kullanıcı tarafından birleştirilir. Kullanıcı kabul testi gerçekçi veriyi kullanarak üretim ortamını benzeterek bir kullanıcı ortamında çalışır. Kullanıcı kabul testi kullanıcının gereksiniminin karşılandığını ve gerçek zamanda kullanım için sistemin hazır olduğunu ve sistemin dağıtıma uygun olduğunu doğrular.  
V modeli avantajları:  
 Doğrulama ve onaylama planları erken aşamalarda vurgulanır. Doğrulama ve onaylama sadece son üründe değil bütün teslim edilebilir ürünlerde kullanılır. Proje yönetimi açısından takibi kolaydır. Kullanımı kolaydır.  
V modeli dezavantajları:  
 Aynı zamanda gerçekleştirilebilecek olaylara kolay imkân tanımaz. Fazlar arasında tekrarlamaları kullanmaz. Risk çözümleme ile ilgili aktiviteleri içermez. Yazılım da diğer sistemler gibi zamanla evrimleşir. Geliştirme devam ettikçe iş ve ürün gereksinimleri de değişiklik gösterebilir. Son ürüne ulaşma düz bir çizgi ile ifade edilemez.  
5.Helezonik (spiral) model:  
 Spiral yazılım geliştirme modeli temel olarak dört ana bölümden oluşur. Bunlar planlama, risk yönetimi, üretim ve kullanıcı değerlendirmeleri olarak tanımlanabilir.   
1. Planlama:  
 Üretilecek ürünleri planlama ve bir önceki adımda üretilen ürünlerle birleştirme yapılır.  
2. Risk analizi:  
 Olası hataların ve risk seçeneklerinin araştırılması risklerin belirlenmesidir.  
3. Üretim:  
 Ara ürünlerin üretilmesidir.  
4. Kullanıcı değerlendirmesi:  
 Üretilen ara ürünlerin kullanıcı tarafından kullanılarak kontrol edilmesi ve değerlendirilmesidir.  
Helezonik model, risk analizi ve prototip üretme üzerine kurulmuştur. Her döngü öncesi, içinde bulunulan fazın risk analizi yapılır ve o faz için planlanmış olan prototip geliştirilir. Her döngünün sonunda, yeniden planlama yapılarak hedefler, alternatifler ve kısıtlamalar belirlenir.   
Helezonik modelin olumlu yanları:  
**Kullanıcı katkısı:**  
 Helezonik modele göre oluşturulan metodolojiler üretim süreci boyunca ara ürün üretme ve üretilen ara ürünün kullanıcı tarafından test edilmesi temeline dayanır. Özellikle uzun projelerde kullanıcının erken dönemde sürecin içerisine katılması, sürekli geri bildirimler verilmesi ve proje bitiminde istenilmeyen durumlarla karşılaşılmaması gibi olgular üretim başarısına katkı sağlar.  
**Yönetici bakışı:**  
 Gerek proje sahibi gerekse yüklenici tarafındaki yöneticiler çalışan yazılımlarla proje süreci boyunca karşılaştıklarından dolayı daha kolay izleme ve planlama kolaylığı elde eder.  
**Yazılım geliştirici (Mühendis) bakışı:**  
 Yazılımın kodlanma ve sınanması çağlayan modeline kıyasla daha erken başlar.  
Helezonik modelin dezavantajları:  
 Küçük ve düşük riskli projeler için pahalı bir yöntemdir. Komplekstir. Spiral sonsuza gidebilir. Ara adımların fazlalığı nedeniyle çok fazla dokümantasyon gerekir. Kontrat tabanlı yazılıma uymaz. Öznel risk değerlendirme deneyimine dayanır. Kontrat tabanlı yazılımlar adım adım anlaşma esnekliğini sağlamaz. Yüksek riskli ögelerin doğru belirlenmesini gerektirir.   
  
6.Artımlı Geliştirme Süreç Modeli:  
 Modelde üretilen ve uygulamaya alınan her ürün sürümü birbirini içerecek şekilde giderek artan sayıda işlev içerecek biçimde geliştirilmektedir. Öncelikle ürüne ilişkin çekirdek bir kısım geliştirilir. Çekirdeğin etrafında üretilen ürünler çekirdeğe eklenerek hedef ürün elde edilir.   
 Yapılacak işlerden edinilecek sonuçlar belirgin değildir ve bir sonraki adımın iş tanımları büyük ölçüde bir önceki adımın sonuçlarına bağlıdır. Bu nedenle gerek zaman gerekse işgücü planlaması yaparken birçok engellerle karşılaşılır. Geliştirilen yazılımlar genellikle sınırlı kez kullanılır ve kullanım bittikten sonra işe yaramaz hale gelir ve atılırlar. Kullanıcı gereksinimleri önceliklendirilir ve öncelikli gereksinimler erken teslime dahil edilir. Gereksinimler önemlerine ve birbirine bağlılıklarına göre sıralanarak her yenilemede bunlarda biri tamamlanır. Bir parçanın gelişmesi başladığında gereksinimleri dondurulur. Olası değişiklikler sonraki teslimlerde ele alınır. Üretilen her yazılım sürümü birbirini kapatacak giderek artan sayıda işlev içerecek şekilde geliştirilir. Uzun zaman alabilecek ve sistemin eksik işlevsellikle çalışabileceği projelere bu model daha uygundur. Bir taraftan kullanım diğer taraftan üretim yapılır. Her teslimle birlikte müşteriye bir sonuç ulaştığından sistemin işlevselliği kısa sürede ortaya çıkar. Erken teslimler sonraki teslimler için gereksinim çıkarılmasında prototip vazifesi görür. Projenin toptan batma riskini azaltır.  
Artımlı geliştirmenin avantajları:  
 Sistem için gerekli olan gereksinimlere müşterilerle birlikte belirlenir. Gereksinimlerin önemine göre teslim edilecek artımlar belirlenir. Erken artımlar prototip gibi davranarak, gereksinimlerin daha iyi anlaşılmasını sağlar. Böl ve yönet yaklaşımı benimsenmiştir.  
Artımlı geliştirmenin dezavantajları:  
 artımları tanımlamak için tüm sistemi tanımlanmasına ihtiyaç vardır. Gereksinimleri doğru boyuttaki artımlara atamak bazen zor olabilir. Deneyimli personel gerektirir. Artımların kendi içerilerinde tekrarlamalara izin vermez.  
7.Araştırma tabanlı süreç modeli:  
 Araştırma ortamları bütünüyle belirsizlik üzerine çalışan ortamlardır. Yapılan işlerden elde edilecek sonuçlar belirsizdir ve bir sonraki işin tanımı genellikle bir önceki adımın sonuca bağlıdır. Bu nedenle gerek zaman gerek işgücü planlanmasında zorluklarla karşılaşılır. Helezonik modelin küçük dönümleri biçiminde gerçekleştirilir. Geliştirilen yazılımlar genellikle sınırlı kez kullanılır ve kullanım bittikten sonra işe yaramaz hale gelir ve atılır.  
8.Kodla ve düzelt yaşam döngüsü:  
 Genellikle kesinleşmemiş olan bir ürün fikriyle başlayıp ürün hazır oluncaya kadar devam eden bir süreçtir. Küçük projeler veya kısa ömürlü prototipler için uygundur. Belirli bir planlamaya ihtiyaç duymaz. Uzmanlık gerektirmediği için herkes tarafından rahatlıkla kullanılabilir.  
 Sistem, en son istenilen şekle gelinceye kadar devamlı geliştirilir. Bakım safhası vardır ama çok zordur. Çünkü sisteme ait dokümantasyon yoktur. Ayrıca emeklilik safhası vardır. Yazılım geliştirmenin en kolay yoludur. Ancak en pahalı olanıdır. Yazılım geliştirmek kolay olduğundan ne yazık ki küçük firmalardaki projelerde bu model kullanılır.   
Bu modelin avantajları:  
 Herhangi bir planlamaya ihtiyaç duymaz. Küçük projelerde kullanılabilir. Program aşamaları çabuk geçilir. Uzman görüşüne ihtiyaç düşüktür.  
Bu modelin dezavantajları:  
 Kontrollü değildir. Kaynak planlaması yoktur. Bitiş süresi belli değildir. Hataların bulunması ve doğrulanması zordur. Kodların düzeltilmesi maliyetli olabilir. Kodlar kullanıcıların ihtiyacını karşılamayabilir. Kodlar sonradan değiştirmek için planlanmadığından esnek değildir, değiştirilmesi zordur.   
  
Çevik Yazılım:  
Yaptığımız yazılımın bir tekniği olmalı hele ki bir ekip halinde geliştirilen yazılımlarda ki iletişim kopuklukları belirli bir standart sağlayamama gibi durumlar ortaya çıkıyor. Bu durum hem şirketimizi hem de müşterimizi zor duruma düşürebilmektedir.  
 Müşteri proje sırasında yeni bir şey ekleme isteyebilir veya ne istediğini bilmiyor olabilir. Bu nedenle süreç devam ederken kayıpsız devam etmesi gerekir. Projenin bir yandan sorunsuz ilerlemesini sağlayan bir metodolojidir. Bu metodolojinin genel özellikleri şunlardır:  
 Basitlik, günlük çalışma, haftalık program, çalışan iletişimi, müşteri iletişimi, modül bazlı çalışma ve modül bazlı testtir.   
  
Scrum:  
 Scrum özellikle yazılım geliştirme süreçlerinde çokça tercih edilen, çevik bir proje yönetim tekniğidir. Daha yüksek kaliteli yazılımların nasıl daha hızlı geliştirileceğine dair kararlara rehberlik etmek için bir dizi değer ve   
ilkeyi uygulayabilmek için yardımcı olur.   
 Scrum, günümüzde kullanılan en popüler agile frameworklerden birisidir ve karmaşık ürünler, sistemler geliştirmek için kullanılır.  
 Scrum kelimesi aslında bir rugby terimidir. Rugby de bir Scrum, topu almaya çalışan bir grup oyuncudur.  
 Proje yönetimi alanında Scrum, ekip üyelerinin bir projeyle ilgili başarıları, ne kadar ileri gittikleri, sonraki adımların ne olduğu ve bekledikleri gelecekteki çıkabilecek sorunlar hakkında konuşmak için bir araya geldikleri kısa toplantıları ifade eder. Toplantılar kısa ve konsantredir.  
Scrum değerleri:  
Scrum aşağıdaki değerler üzerine kurulmuştur. Bunların olmadığı bir ortamda uygulanamaz.  
**Taahhüt:** Ekip üyeleri kişisel olarak ekip hedeflerine ulaşmayı taahhüt eder.  
**Cesaret:** Ekip üyeleri doğru olanı yapar ve zorlu sorunlar üzerinde çalışır.  
**Odaklanma:** Belirlenen işe ve takımın hedeflerine odaklanılır.  
**Açıklık:** Ekip üyeleri, ekibin karşılaştığı tüm işler ve sorunlarda birbirlerine açıktır.  
**Saygı:** Ekip üyeleri, yetenekli ve bağımsız olmak için birbirlerine saygı duyarlar.  
  
Scrum prensipleri:  
**Şeffaflık:** Ekip, herkesin diğer ekip üyelerinin karşılaştığı sorunların farkında olduğu bir ortamda çalışmalıdır. Ekipler, organizasyon içinde, genellikle uzun süredir mevcut olan ve ekibin başarısının önüne geçen sorunları ortaya çıkarır. Ekip üyeleri şeffaf bir şekilde bütün sorunları belirlemelidir.  
**Düzenli takip:** Ekibe, sürecin nasıl işlediğini yansıtma fırsatı vermek için çerçeveye yerleştirilmiş sık denetim noktalarıdır. Bu inceleme noktaları günlük Scrum toplantısını ve sprint gözden geçirme toplantısını içerir.  
**Adaptasyon:** Ekip sürekli olarak işlerin nasıl gittiğini araştırır ve mantıklı görünmeyen öğeleri gözden geçirir.  
Scrum Günümüzde Neden Popüler:  
 Scrum, rol uyumunu hesaba katan çok disiplinli bir metodolojidir. Bu özellik, tüm ekip üyeleri tarafından oynanan bireysel rollerin daha iyi anlaşılmasını ve takdir edilmesini olanak sağlar.  
 Scrum sayesinde devamlı geri bildirimler alarak, çok hızlı ve kaliteli projeler geliştirebilirsiniz.  
 Ekip üyelerinin gerek projenin gidişatını anlamada gerekse ekip arkadaşlarının yaptıkları çalışmaları anlamada çok yardımcıdır. Bu özellik çalışmaların daha hızlı aynı zamanda da kontrollü bir şekilde devam etmesini sağlamaktadır. Takım olmanın önemli olduğu bu süreçte ekibe uyum sağlayamayan üyeler de olabilir. Bunun gibi durumlar sürecin verimsizleşmesine ve yavaşlamasına yol açar.  
   
**KAYNAKÇA:**

<http://w3.bilecik.edu.tr/bilgisayar/wp-content/uploads/sites/75/2019/02/ders-2.pdf>

<https://medium.com/@tunaytoksoz/yazilim-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-sdlc-ve-modelleri%CC%87-c3fe40f6e4e8>

<https://toptalent.co/agile-yazilim-gelistirme-alaninda-bir-metod-scrum-framework-nedir>

<https://www.argenova.com.tr/scrum-nedir>

<https://stringfixer.com/tr/V-Model_(software_development)>

<https://hayririzacimen.medium.com/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-ve-s%C3%BCre%C3%A7-modelleri-70fdfb2f8f77#:~:text=Kodla%20ve%20D%C3%BCzelt%20(Code%20and,i%C3%A7in%20herkes%20bu%20modeli%20kullanabilir>.

<https://www.codex.com.tr/yazilim-gelistirme-modelleri>

<https://fikirjeneratoru.com/yazilim-proje-yonetimi-yontemleri/>

Linkedin:  
<https://www.linkedin.com/in/mehmet-enes-arslan-0433a7235/>

Github:  
<https://github.com/MehmetEnesARSLAN/MehmetEnesARSLAN/settings>

Medium:  
<https://medium.com/@arslanmehmetenesceng>