YAZILIM YAŞAM DÖNGÜ MODELLERİ

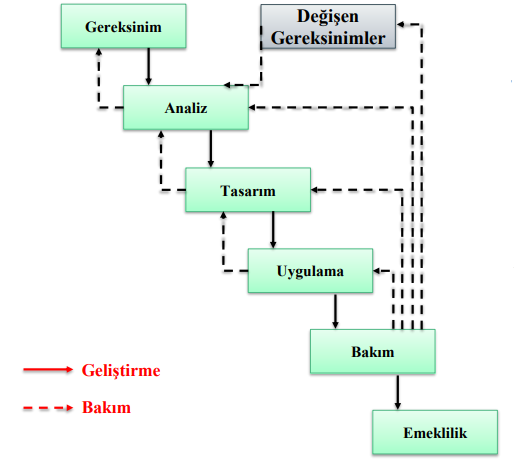
Bir yazılımın üretimi ve kullanımı süresince, geliştirilmesi ve bakımının yapılmasını sağlayan aşamalar topluluğuna yazılım yaşam döngüsü (Software life cycle) denir. Yazılım bir nevi yaşayan bir organizma olduğu için: ihtiyaçları, işlevleri sürekli değişir ve gelişir. Bu yüzden, yazılım yaşam döngüsü tek yönlü (doğrusal) olarak düşünülemez, bir döngü olarak düşünülmelidir. Yazılım yaşam döngüsünün temel adımları şunlardır: Gereksinim (Requirements), Analiz (Analysis)-Çözümleme, Tasarım (Design), Gerçekleştirme (Implementation), Bakım (Maintenance).

•**Gereksinim:** Müşterinin ihtiyaçlarına göre gereksinimlerin belirlendiği ve fizibilite yapılan aşamadır. •**Analiz:** Gereksinimlerin incelendiği, önceliklerin belirlendiği aşamadır. Temel sorunlar burada ortaya çıkar. •**Tasarım:** Gereksinimleri karşılayacak yazılımın temel yapısının oluşturulduğu aşamadır. •**Gerçekleştirme:** Kodlama ve yazılan kodun test edilmesi bu aşamada yapılır. •**Bakım:** Ürünü teslim ettikten sonra, bildirilen hataların düzeltildiği ve ihtiyaç duyulan yeni gereksinimlerin eklendiği aşamadır.

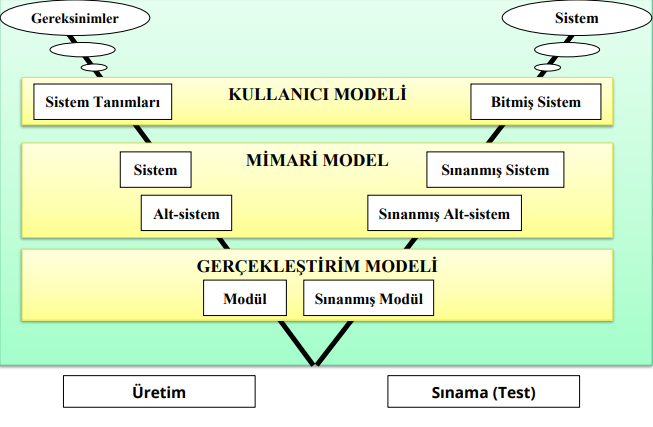
Birçok yazılım yaşam döngü modeli vardır.

**Gelişi Güzel Model:** Kişiye bağlı yazılım geliştirme şeklinde yapılır, bir yöntem ve model yoktur. Kişiye özel yapıldığı için sürecin gözetilmesi, yazılımın izlenebilirliği ve bakım yapılabilirliği oldukça zordur. Genellikle basit programlama içeren ve tek bir kişinin üretimde yer aldığı yöntemdir. 1960’laı yıllarda kullanılmıştır.

**Barok Modeli:** Yazılım yaşam döngüsü temel adımları esas alınır. Doğrusaldır ve döngü yoktur. Döngü olmadığı için aşamalar arası geri dönüşlerin nasıl yapılacağı da tanımlı değildir. Gereksinim adımına daha çok önem veren bu model, belgelemeyi (Dökümantasyon) ayrı bir süreç olarak ele alır ve yazılımın geliştirilmesi ve testinden hemen sonra yapılmasını öngörür. Günümüzde ise belgeleme, yapılan işin doğal bir ürünü olarak görülür. 1970’li yıllarda kullanılan Barok modeli, günümüz yazılım geliştirme projelerinde kullanılan bir model değildir.

**Çağlayan Yaşam-Döngü modeli:** Günümüzde geleneksel yazılım geliştirme modeli olarak anılan ve kullanımı git gide azalan bu model, en eski ve en temel model olup geçmişte oldukça popülerdi.

Çağlayan yaşam-döngü modelinin temel adımları baştan sona en az bir kez izleyerek gerçekleştirilir. Gereksinimleri net bir şekilde belirlenmiş, iyi tanımlı projeler ve üretimi az zaman gerektiren yazılım projeleri için uygun bir modeldir. Nesne yönelimli ve karmaşık projeler için uygun değildir. Bir aşama bitmeden sonraki aşamaya geçilmez ve her aşama sonunda dökümantasyon yazılmalıdır, Eğer bir aşama sonunda dökümantasyon ve test olmamışsa o aşama bitmiş kabul edilemez. Bazı aşamalarda hatalar meydana gelir. Bu hataların erken fark edilmesi ile çözüm de daha kolay ve daha ucuz olur. Hatanın kaçıncı aşamada yapıldığı da maliyeti etkiler. Örneğin analiz aşamasındaki bir hatayı test aşamasında düzeltmeye çalışırsak, test aşamasına kadar ki bütün aşamaları kontrol etmemiz gerekecektir. Bu da hem zaman kaybı artacak hem de maliyeti baya yüksek olacak demektir ama analiz aşamasında fark edilirse, çözümü daha ucuz olur. Bu yüzden bu modeli kullanan proje ekipleri en fazla zamanı analiz ve tasarım aşamalarına ayırmalıdır. Maliyeti etkileyen bir diğer etken ise müşteridir. Yazılımı oluşturma sürecinde müşteri ile iletişim halinde olunup ondan geri dönüşler alınmalıdır. Aksi takdirde yazılım ürünü tamamlandıktan sonraki geri dönüşleri dikkate almak çok daha maliyetli olacaktır. Çağlayan yaşam-döngü modeli, barok modelin aksine belgeleme(dökümantasyon) işlevini ayrı bir aşama olarak ele almaz, üretimin doğal bir parçası olarak görür ve aşamalar arasında döngü vardır.

**V Süreç Modeli:**V süreç modelinin; sol tarafı üretim, sağ tarafı ise sınama (test) işlevleri ile ilgilidir. Bu model, bir nevi çağlaya yaşam-döngü modelinin gelişmiş halidir. Yatay ve dikey açılar zaman veya projenin tamamlanabilirliğini ve soyut seviyeyi gösterir. Her aşamada test olması kullanıcın projeye katkısının artmasını sağlar. Doğrusal bir yönde ilerlemek yerine, süreç adımları kodlama evresinden sonra yukarıya doğru eğim alır ve V şeklini oluşturur. Modelin V şeklinde olması, test aşamasında bir hata bulunursa hemen yatay doğrultudan üretim kısmına giderek hatayı çözmemizi kolaylaştırır.

**V Süreç Modelinin Temel Çıktıları:**

•**Kullanıcı Modeli:** Geliştirme sürecinin kullanıcı ile olan ilişkileri tanımlar ve sistemin nasıl kabul edileceğine ilişkin sınama belirtimleri ve planları ortaya çıkarır.

•**Mimari Model:** Sistem tasarımı ve oluşacak alt sistem ile tüm sistemin sınama işlemlerine ilişkin işlevler.

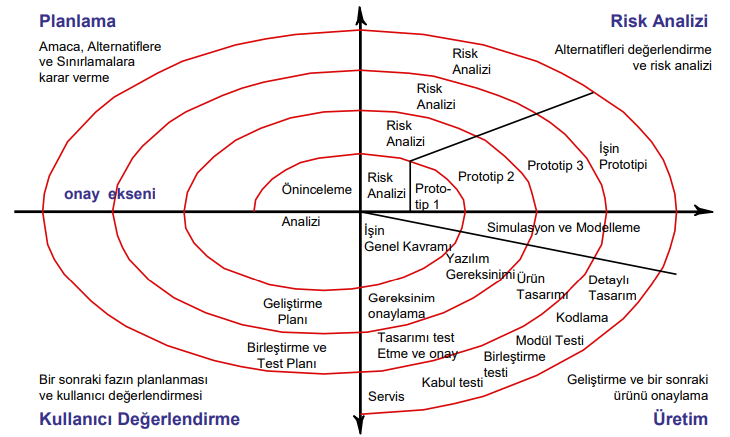
•**Gerçekleştirim Modeli:** Yazılım modüllerinin kodlanması ve sınanmasına ilişkin fonksiyonlar.

İş tanımlarının belirgin, belirsizliklerin az olduğu BT (Bilgi Teknolojileri) projeleri için uygun bir modeldir

BT projesinin iki aşamalı olarak ihale edilmesi için oldukça uygundur:

• İlk ihalede kullanıcı modeli hedeflenerek, iş analizi ve kabul sınamalarının tanımları yapılmakta,

• İkinci ihalede ise ilkinde elde edilmiş olan kullanıcı modeli tasarlanıp, gerçeklenmektedir.

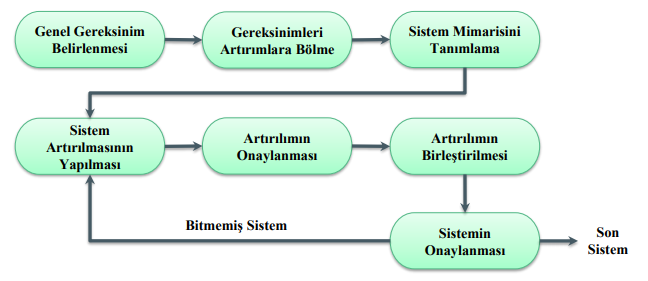
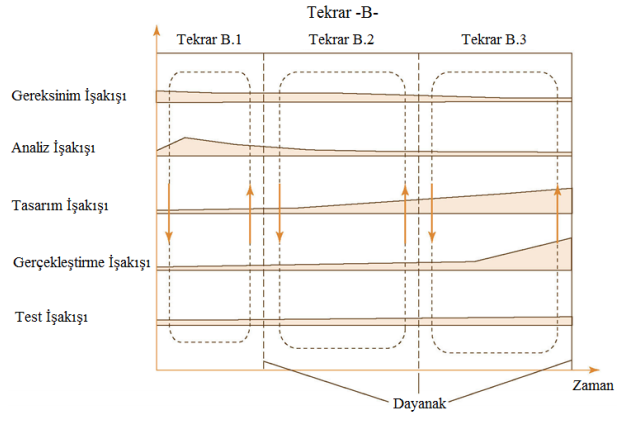
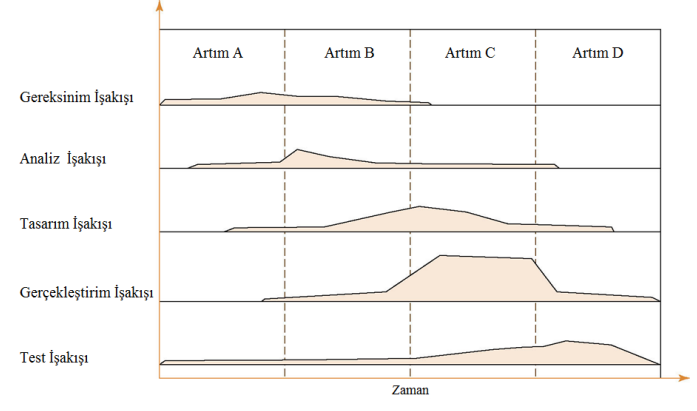
**Helezonik (Spiral) Model:**Helezonik model 4 ana bölümden oluşur.

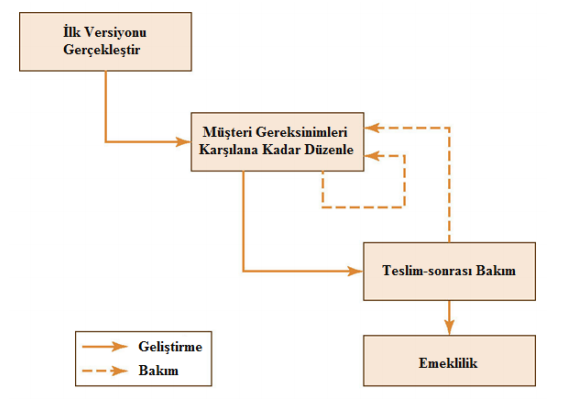
•**Planlama:** Üretilecek ara ürün için işin planlanması, amaç ve kısıt ve alternatiflerin belirlenmesi, bir önceki adımda üretilmiş olan ürün ile tümleştirme yapılması faaliyetlerini içerir •**Risk Analizi**: Risk seçenekleri araştırılır, alternatiflerin değerlendirilir ve riskler belirlenir. •**Üretim:** Planlanmış ara ürün üretilir. •**Kullanıcı Değerlendirme:** Ara ürün ile ilgili, kullanıcı tarafından test ve değerlendirmeler yapılır.

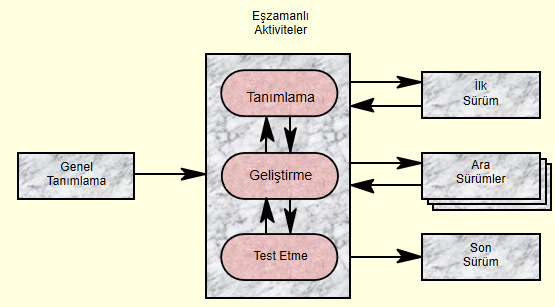
Risk analizi ön plana çıkmıştır, bu sebeple küçük ve düşük riskli projeler için çok pahalı bir modeldir.. Doğrudan gereksinim, analiz, tasarım... vs. gibi bir faz yoktur. Yinelemeli artımsal bir yaklaşım vardır. Helezonik modelde her döngü bir fazı ifade eder ve her tam turda prototipler ortaya çıkar. **Prototip:** Prototip, bir kavram ya da süreci test etmek ya da çoğaltılmış ve öğrenilecek bir şey gibi davranmaya yönelik inşa edilmiş, bir ürünün erken örneği, modeli ya da sürümüne denir. Prototipler final ürünü değildir ama ürünün o aşamada finale en yakın halidir. Oluşan prototipler müşteriye sunulur ve müşteri nelerin eksik ve tam olduğu ile ilgili geri dönüşte bulunur. Bu geri dönüşler dikkate alınarak yeni prototipler oluşturulur ve tekrar müşteriye sunulur. Bu döngü asıl ürün oluşuncaya kadar devam eder.

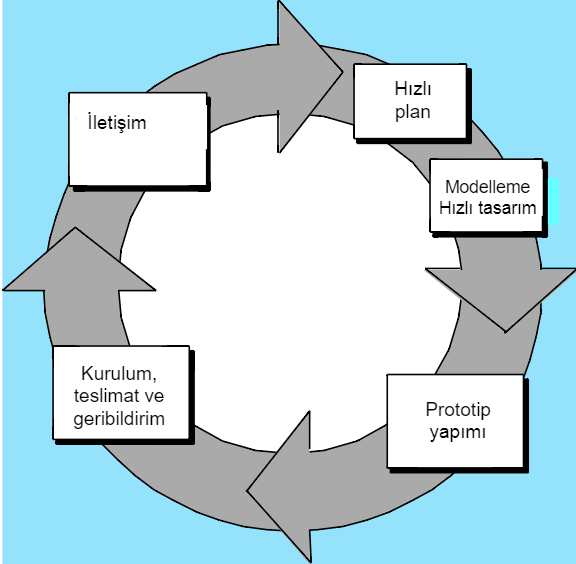
Helozonik modelin birçok avantajı vardır: •Üretim süreci boyunca ara ürün üretme ve üretilen ara ürünün kullanıcı tarafından sınanması temeline dayanır. •Yazılımı kullanacak personelin sürece erken katılması ileride oluşabilecek istenmeyen durumları engeller. •Gerek proje sahibi, gerekse yüklenici tarafındaki yöneticiler, çalışan yazılımlarla proje boyunca karşılaştıkları için daha kolay izleme ve hak ediş planlaması yapılır. •Yazılımın kodlanması ve sınanması daha erken başlar.

**Artımsal Geliştirme Süreç Modeli:** Kullanım ve üretim eş zamanlı gerçekleşir. Proje tek seferde teslim edilmek yerine parçalara bölünür. Kullanıcı gereksinimleri önceliklendirir ve bu gereksinimler erken teslimlere dahil edilir. Erken teslimler, sonraki teslimler için gereksinimleri çıkarmada prototip vazifesi görür. Gereksinimler önemlerine ve birbirine bağımlılıklarına göre sıralanarak her yinelemede bunların bir kısmı tamamlanır.

Planlama aşaması ile başlanır ve temel adımlardan oluşan bir döngüye giriş yapılır. Uygulama aşamasına gelince sistem; bir taraftan uygulamayı müşteriye göstermek bir taraftan da uygulamayı test etmek üzere ikiye ayrılır. Müşteriden gelen geri dönüşler ve test sonuçları doğrultusunda proje tekrar döngü içerisine konulup eksiklikler tamamlanır. Üretilen her yazılım sürümü birbirini kapsayacak ve giderek artan sayıda işlev içerecek şekilde geliştirilir. Bu süreçte planlamada oluşan yeni gereksinimler de artırımlı modelde her seferinde arttırıla arttırıla bir iterasyon şeklinde devam eder ve her yenilik bir sonraki iterasyonda sürece dâhil edilir. Uzun zaman alabilecek, eksiklikler çıktıkça üzerine ekleme yapabileceğimiz ve sistemin eksik işlevlikle çalışabileceği türdeki projeler bu modele uygun olabilir. Artımsal modelin çağlayan yaşam döngü modelinden en büyük farkı ise hata veya yeni gereksinimler çıkması sonucu maliyetinde büyük bir değişim olmamasıdır.

**Kodla ve Düzelt Yaşam-Döngü Modeli:**  Büyük projeler için kullanılmaz, birkaç yüz satırdan oluşan küçük projelerde ya da kısa ömürlü prototipler için kullanılabilir. Direkt olarak yazılım ürünü gerçekleştirilir. Sistem, en son istenilen şekle gelinceye kadar devamlı geliştirilir. İlk safhada yazılım ürünün ilk sürümü gerçekleştirilir. Bakım safhası vardır ama sisteme ait dökümantasyon olmadığı için çok zordur. Emeklilik safhası da vardır. Teknik kullanılmadığı için spagetti kod örnekleriyle karşılaşılabilir. Yazılım geliştirmenin en kolay ancak en maliyetli yoludur. Genellikle küçük ve tecrübesiz firmalar tarafından kullanılır. 

**Evrimsel Geliştirme Süreç Modeli:** İlk tam ölçekli modeldir. Coğrafik olarak geniş alana yayılmış, çok birimli organizasyonlar için önerilmektedir (banka uygulamaları). Her aşamada üretilen ürünler, üretildikleri alan için tam işlevselliği içermektedirler.

**Prototipleme:** Gereksinimler hızlıca toplanır. Geliştiriciler ve kullanıcılar bir araya gelip yazılımdan elde edilecek bütün çıktılara, gerekli girdilerin nasıl sağlanacağına, nasıl korunacağına, hangi işlemlere uğrayacağına karar verir. Bir ilk örnek üretilir ve kullanıcının değerlendirmesine sunulur. Kullanıcıdan alınan geri dönüşler doğrultusunda ilk örnekte değişiklikler yapılır ve tekrar kullanıcı değerlendirmesine sunulur. Böylelikle kullanıcının istediği yazılıma yakın bir ilk örnek üzerinde yazılımın işlevleri konusunda, kullanıcı ile anlaşılır. Doğrusal modelin döngü versiyonudur. Bu modelde, gereksinim analizi ve prototipleme için tasarım yapıldıktan sonra geliştirme süreci başlar. Prototipleme yapıldıktan sonra kullanıcıya iletilir ve müşteri paketi test edip geri bildirimde bulunur. Sınırlı sayıdaki yinelemelerden sonra, son yazılım paketi üreticiye verilir. 

**Çevik Yazılım Geliştirme:** 1990’lı yılların sonuna doğru “çevik (agile)” denen metotlar geliştirilmiştir. Çevik yazılım geliştirme metotları ürünün piyasaya çıkma süresini kısaltma ve değişen isteklere hızlı yanıt vermeyi amaçlar. Çevik yazılım geliştirme metotları, verimliliği yüksek, esnek, hata oranı düşük, hızlı ve ucuz çözümler sağlamaktadırlar. Bu metotlar kendi içerisinde özü aynı fakat pratikleri farklılaşan çeşitli metodolojilere ayrılmaktadırlar. “Çevik” zamanda yazılım süreçlerini de kısaltan kavramsal bir yazılım geliştirme metodolojisidir. Bu metodolojide ana fikir projenin parçalanarak yinelemelere ayrılması ve her yinelemenin ayrı bir projeymiş gibi ele alınmasıdır. Projenin parçalanması geriye dönük hataların çözümleri daha kolaylaştırır. İletişim çok önemlidir, proje ekipleri hem kendi aralarında hem de müşteri ile sürekli iletişim halindedir. Büyük şirketlerin karmaşık, uzun projelerinde kullanılır.

2001 yılında yazılım dünyasının önde gelen isimlerinden Kent Beck ve 16 arkadaşı “Çevik Yazılım Geliştirme Manifestosu” nu yayınlamışlardır.

**Çevik Manifesto:** Yazılım geliştirme amacıyla üretilmiş bir modelleme biçimidir, kapsadığı değerler, prensipler ve pratikler sayesinde geleneksel modelleme metotlarına göre yazılımlara daha esnek ve kullanışlı biçimde uygulanabilir.

Bu manifestoda:

1. Bireyler ve etkileşimi, süreç ve araca tercih edilir.

2. Çalışan bir yazılımı, detaylı belgelendirmeye tercih edilir.

3. Müşteri ile iş birliğini, sözleşmedeki kesin kurallara tercih edilir.

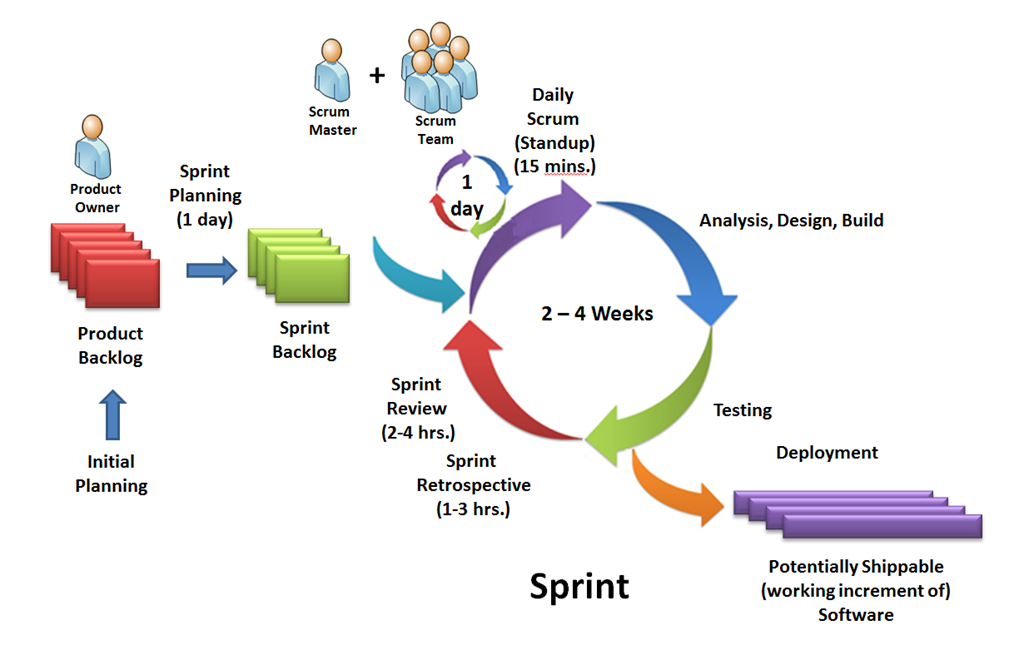
4. Değişikliklere uyum sağlayabilmeyi, belirli bir plana tercih edilir.

**En yaygın uygulanan çevik metodolojiler:**

Extreme Programming (XP), SCRUM, Agile Unified Process, Feature-Driven Development (FDD), Test-Driven Development (TDD),LEAN Development, Dynamic System Development Methodology (DSDM),Microsoft Solution Framework (MSF) olarak bilinen çevik metodolojiler vardır.

**Extreme Programming (XP):** XP kolay, grup içi iletişime önem veren, geri dönüşlerin daha fazla olmasına imkân sağlayan bir yazılım geliştirme yöntemidir.Dört temel değerden oluşur, bunlar: İletişim, basitlik, geri bildirim, cesarettir.

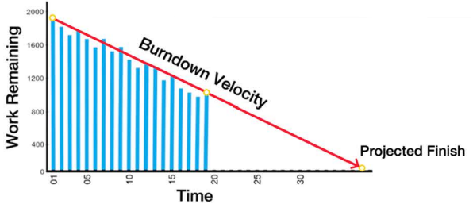
•**İletişim:** XP iletişim eksikliğini ortadan kaldırmayı amaçlar. XP’ de yazılım ekibi ile kullanıcılar arasında iletişim yüz yüze olmalıdır.•**Basitlik:** Karmaşık çözümler yerine, gereksinimleri karşılayan en basit çözüm kullanılmalıdır. •**Geri bildirim:** Müşteri, yazılım grubunun bir üyesi olarak görülür. Müşterilerle yazılım ekibi belli zamanlarda buluşup, o ana kadar gerçekleştirilen yazılım çalıştırılarak gelinen noktaya bakarlar. Müşteriden alınan geri bildirim sayesinde ortaya çıkabilecek hatalar/anlaşmazlıklar önlenir. •**Cesaret:** XP’nin en zor değeridir. Başarısızlıklardan korkmak yerine onları telafi etmeyi önerir.

**Scrum**: Rugby sporundaki bir hücum taktiğinden adını almaktadır. Scrum bir proje yönetim yaklaşımıdır. Yani sadece yazılım geliştirmeye değil, her yere uygulanabilir. Kompleks yazılım süreçlerinin yönetilmesi için kullanılır. Karmaşık yazılım işlerini küçük birimlere (sprint) bölerek geliştiren bu metodoloji, karmaşık ortamlarda adım adım yazılım geliştiren ekipler ve gereksinimlerin kolaylıkla tanımlanamadığı, kaotik durumların beklendiği projeler için en uygundur. **Product Backlog:** müşteri ile anlaşıp önceliklendirilmiş yüksek seviyeli gereksinim listesi olarak düşünülebilir.**Sprint Backlog:** Genelde 15-30 günlük (Time-Boxed) proje zaman dilimidir ve bir proje 1'den fazla Sprint'ten oluşur. Örnek olarak 3 aylık bir proje.**Scrum Daily Meeting:** Her gün Scrum takımıyla, Scrum Daily Meeting (maksimum 30 dk. ve ayakta) düzenleyerek genel durum hakkında bilgi alırsınız. Dün ne yaptın? Bugün ne yapacaksın? Seni engelleyen ne idi?

Scrum’da 3 temel kavram vardır:

**Roller (Roles)** •**Ürün Sahibi (Product Owner)** Geliştirme takımı ve müşteri arasındaki iletişimi sağlar. Projenin özelliklerini tanımlar. Projenin iş değeri açısından geri dönüşü ile sorumludur.

•**Scrum Yöneticisi (Scrum Master)** Scrum kurallarını, teorilerini ve pratiklerini iyi bilir ve takımın bu kurallarını uygulamasından sorumlu kişidir. Takımı ve organizasyonu Scrum’a adapte eder.•**Scrum Takımı (Scrum Team)**  Scrum Takımı, devamlı iletişim halinde olan ve tek bir hedefe ulaşmak için mücadele eden kişilerden oluşur. 5 – 9 kişiden oluşur.

**Toplantılar (Meetings)** •**Sprint (Koşu) Planlama (Sprint Planning)** Geniş kapsamlı gereksinim listesi çıkarılır. Dağıtımlar için gereksinimler eşleştirilir, takımlar belirlenir. Dağıtım, geliştirme ve pazarlama maliyetleri hesaplanır •**Sprint (Koşu) Gözden Geçirme (Sprint Review)** Her sprint başlangıcında planlama toplantısı yapılır. Ürün sahibi ile takım, ürün gereksinim listesini gözden geçirir, gereksinim elemanlarının hedeflerini ve içeriklerini belirler, Listenin en başından başlanarak sprint sonucunda gerçekleştirilecek gereksinimlerden oluşan sprint gereksinim listesi oluşturulur. Bu scrum’ın anahtar uygulamasıdır. Takım ürün sahibi tarafından belirtilen önceliklendirilmiş gereksinimlerden ne kadarını yapacağını belirler ve taahhüt eder. •**Günlük Scrum Toplantısı (Daily Scrum)** Sprint başladıktan sonra takım sürecin başka bir anahtar aktivitesi olan Günlük Scrum Toplantılarını gerçekleştirir. Tüm takımın katıldığı bu toplantı her iş gününde belirlenen saatte gerçekleşir ve 15dk’dir. Takımın ilerleyişini ve karşılaştıkları engelleri görmek için önemli bir fırsattır. Tüm ekip üyeleri teker teker şu sorulara cevap verir; “Dün ne yaptım?”, ”Bugün ne yapacağım?”, “Önümde olan engeller ve karşılaştığım sorunlar neler?”. **Bileşenler/Araçlar (Artifacts)** •**Ürün Gereksinim Dokümanı (Product Backlog)**  Proje boyunca yapılması gereken iş elemanlarının basit bir listesidir.Ürün Gereksinim Listesi canlı bir dokümandır, geçerli ve kullanışlı olması için devamlı bakım gerekir. Birçok yeni eleman zamanla eklenebilir, birçok eleman daha küçük parçalara ayrıştırılabilir, bazı gereksinimlerin artık gereksiz veya anlamsız olduğu fark edilip silinebilir. Listedeki sıralamalar her gün değişiklik gösterebilir. Ürün Gereksinim Listesi, genellikle kullanıcı hikayelerinden (user story) oluşur ve kullanıcı bakış açısından bakılır. •**Sprint (Koşu) Dokümanı (Sprint Backlog)** Mevcut sprint için product backlog’dan elde edilmiş iş ve görevleri kapsar. Sprint backlog’daki işlerin amacı, sprint sonunda son ürünün bir parçası olan bir işlevselliği ya da çalışabilir bir parçayı elde etmektir. Sprint backlog sadece takım tarafından değiştirilmektedir. •**Sprint Kalan Zaman Grafiği (Burndown Chart)**  Bu grafik, iterasyon (sprint) boyunca işlerin ne kadarının yapıldığı ile normalde ne kadar yapılması gerektiğini karşılaştırılabilmesini sağlar. Bir iterasyonun toplam 100 saatten ve 20 günden oluştuğunu farz edelim. Normal olarak beklenen her gün 5 saatlik bir işin yapılmasıdır. Takım elemanları her gün ne kadarlık bir iş gerçekleştikleri bilgisini girerler. 

**SCRUM GÜNÜMÜZDE NEDEN POPÜLER?**

Büyük projeleri parçalayarak, karmaşık yapıları basite indirgeyen bir metottur. Günlük yapılan kısa toplantılar ile karşılaşılan sorunları hemen çözülmesini ve ekiplerin iletişim halinde kalmasını sağlar. Müşteriyi de sürece dahil edip alınan geri dönüşleri dikkate alarak müşteri memnuniyetini sağlar. Böylelikle süreç sonunda istenen projeyi oluşturmuş olup, zaman ve para kaybının da önüne geçer. Bu yüzden scrum günümüzde fazlasıyla tercih edilen bir metottur.

**MODELLERİN KARŞILAŞTIRILMASI:**

KAYNAKLAR:

<https://medium.com/@secilcor/scrum-nedi%CC%87r-6a4326951dd8>

<https://medium.com/@ryoldash/agile-%C3%A7evik-yaz%C4%B1l%C4%B1m-geli%C5%9Ftirme-nedir-ve-nas%C4%B1l-uygulan%C4%B1r-93e85ffc866>

<http://web.iku.edu.tr/~gyilmaz/Notes/YazilimMuhendisligiYonetimi/Bolum-02.ppt>

<https://www.techwell.com/2013/02/why-scrum-so-popular>

<https://medium.com/@halilertas88/yazilim-ya%C5%9Fam-d%C3%B6ng%C3%BC-model-6b23478bde38>

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

<https://www.wikipedia.org/>

<https://www.youtube.com/watch?v=Lk7N1zd7Mv8>

<http://www.yazilimheryerde.com/2014/09/agile-metodoloji-scrum.html#:~:text=Agile%20Metodoloji%20(%C3%87evik%20Metodoloji)%20yaz%C4%B1l%C4%B1m,itibaren%20yaz%C4%B1l%C4%B1m%20sekt%C3%B6r%C3%BCnde%20uygulanmaya%20ba%C5%9Flanm%C4%B1%C5%9Ft%C4%B1r>.