Tıpkı hayatımızda olan doğma-büyüme-ölme döngüsü gibi yazılım dünyasının yani yazılım sürecinin de böyle bir döngüsü vardır.Buna “*yazılım yaşam döngüsü(Software Development Life Cycle,SDLC)* “denir.Yazılım yaşam döngüsü içerik olarak bölünmüş aşamalardan oluşmaktadır.Yazılım çalışmalarımız hem müşterilerle hem takım arkadaşlarımızla hem de diğer kişilerle sürekli bir etkileşim içinde olmamızı gerektiren bir sisteme dayalıdır.Bunun için bu döngüde ,döngü aşamaları arasında geriye dönmek ve tekrar ileri gitmek gibi hareketler yapılabilir.Yazılım yaşam döngüsünün aşamaları ve bu aşamalarda neler yapıldıkları aşağıdaki gibidir:

1. **Planlama Aşaması** :Döngümüzün ilk aşaması planlamadır.Bu aşamada yapılacak ürünün fikrinin ortaya çıktığı aşamadır.Ürün fikri için müşterinin gereksinimleri alınır ve fizibilite(bir yatırım ya da proje başlatma kararı almadan önce gereklli bütün bilgilerin sistemli bir şekilde elde edilmesidir.) çalışmaları yapılır.
2. **Analiz Aşaması**:Bu aşamada artık belirlediğimiz gereksinimleri açıklığa kavuşturmak için müşteriyle iletişim içinde olup , birlikte çalışarak bu gereksinimler kesinleştikten ,varsa problemleri de çözdükten sonra *dökümante* edilir.
3. **Tasarım Aşaması**:Gereksinimler açıklığa kavuşturulup oturtulduktan sonra tasarım aşamasına geçilir.Bu aşamada adı da üstünde olduğu gibi müşterinin gereksinimlerine bağlı kalınarak ürünün tasarımı yapılır.Yani ürünün özellikleri,yetenekleri ve arayüzlerinin belirlemesi etkinliğidir.Bu aşamada artık ürün ile ilgili bütün karaların verilmiş olması gerekir.Bir sonraki adıma herhangi soru veya karar bırakılmaz.
4. **Gerçekleştirme ve Test Aşaması**:Gerçekleştirme ürünü somut olarak ortaya çıkardığımız aşamadır.Bu aşamada tasarım aşamasından gelen gereksinimler doğrultusunda kodlama yapılır.Test aşamasında ise kodlama yaparken ve kodlama bittikten sonra ürün test edilir.Test aşamasını analiz aşamasından itibaren sıkı tutmanın hata yapma olasılığımızı azaltacağından dolayı bize prestij,zaman,para vb. anlamında katkılar sağlamış olur.
5. **Teslim ve Bakım Aşaması**:Kodlama güzel bir şekilde yazılıp bütün test aşamaları bittikten sonra artık ürünün sahaya verilmesi kalır.O da bu aşamada yapılır. Ürün sahaya verildikten sonra ise bakım aşaması başlar.Mutlaka son kullanıcılar için kullanım kılavuzu ve versiyon fark dokümanı oluşturulmalıdır.Bu doküman sonuçlarına göre de hata giderici ,ürüne yeni özellikler ekleyici,önleyici gibi amaca göre seçilen farklı bakımlar uygulanır.

****

Yazılım yukarıda görüldüğü gibi bazı temel aşamalardan oluşur.Bu aşamalara *çekirdek süreçler(core processes)*denir.Bu süreçlerin gerçekleştirilmesi için *Yazılım Belirtim Yöntemleri* ve *Yazılım Yaşam Döngüsü Modelleri*  kullanılmaktadır.

**Yazılım Belirtim Yöntemleri(Specifications)**

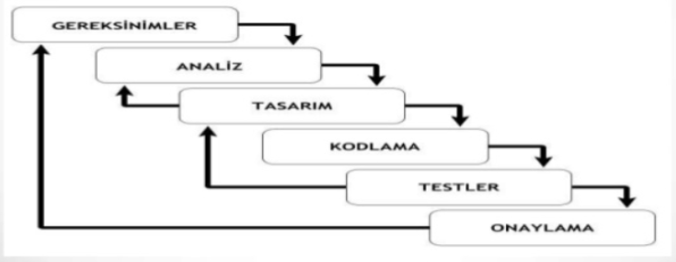
Yukarıda bahsettiğimiz çekirdek sürece ilişkin fonksiyonları yerine getirmek amacıyla kullanılan yöntemlerdir.

* Süreç Akışı İçin Kullanılan Belirtim Yöntemleri:Bu süreçler arasındaki ilişikilerin ve sağlanan iletişim gösterildiği yöntemlerdir(Yapısal Şemalar.Veri akış şemaları,Nesne/Sınıf şemaları(use ase,UML vs.)).
* Süreç Tanımlama Yöntemleri :Yapılan süreçlerin içindeki işleyişini göstermek için kullanılan yöntemlerdir(Algoritma,Karar Tabloları,Karar Ağaçları,Anlatım Dili,Düz Metin vs.).
* Veri Tanımlama Yöntemleri:Süreçlerde kullanılan verilerin tanımlanması için kullanılan yöntemlerdir(Veri Tabanı Tabloları,Veri Sözlüğü,Nesne İlişki Modeli).

**Yazılım Yaşam Döngüsü Modelleri**

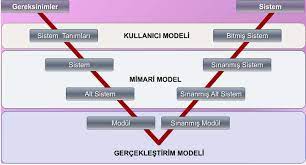
 Bir yazılım ürününün ihtiyacının ortaya çıkmasından kullanımdan kalkmasına kadar geçen döneme *yazılım yaşam döngüsü modeli* denir. Eğer yazılım yaşam döngü modelinde adımları takip edip yazılım projelerimizi yaparsak yazılım krizinde çıkacak hataları minimuma indirdiğimiz için mağduriyetler azalacaktır.Yazılım yaşam döngüsü modelleri sırasıyla şunlardır:

* **Gelişigüzel Model:** İsminden de anlaşılacağı üzere gelişigüzel geliştirmede belli bir model ya da yöntem kullanılmaz.Aslında tam anlamıyla bir model de denemez.Sadece geliştiren kişiye bağımlı olarak yapılır.Geliştiren kişinin bile aradan belirli bir zaman süreci geçtikten sonra anlamayacağı ya da değiştirmede güçlük çekeceği bir model olduğu için bakım ve izlenilebilirliği oldukça zor bir modeldir.
* **Barok Modeli:**Barok modeli, 70’li yılların ortalarında kullanılmaya başlanmıştır.Bu döngü süreçlerin doğrusal bir şekilde gerçekleştirilmesini öngörür.Adımsal arası ilişkiler arası tanımlı olmayan bir yöntem olduğu için günümüzde bu yöntem pek kullanılmaz.En büyük sorunların biri de belgelemenin yapılan bir işin doğal bir ürünü gibi görülmemesidir.Yani “*belgeleme*”adı altında yazılımın geliştirilmesi ve testin ardından ayrı bir adım bulunmasıdır.Günümüzde kullanılmamasının diğer bir nedeni ise *gerçekleştirme* evresine daha çok ağırlık veren bir model olmasıdır.Hem bu hem de belgeleme işleminin en sonda ayrı bir adım olarak ele alınması yapılan üründeki hata yapma olasılığımızı bir hayli arttırır ve bu da presitj,zaman,para gibi maaliyet açısından bir çok şeyi oldukça kötü etkilemiş olur.
* **Çağlayan(Şelale)Modeli:**Bu model,yukarda bahsettiğimiz yazılımın çekirdek süreçlerinin ardışık olarak icra edildiği,yazılım mühendisliğinin en eski ve temel modelidir.Günümüzde hükümetler ve büyük şirketler tarafından her türlü projenin yönetim standardı olarak kabul görmektedir.Yazılım tanımlarında belirsizlik hiç yok ya da az varsa yazılımı üretim aşaması çok uzun zaman alması beklenmiyor ise bu süreç modelinin kullanılması önerilir.Şelale yaşam döngüsü modeli aşamaları şunlardır:Gereksinimler,tasarı,uygulama,test ve bakım.Bunlardan da zaten yukarda detaylı olarak bahsetmiştik.Eksiklikler veya hatalar fark edilirse önceki adımlara geri dönülür.Adımları geride bıraktıkça,ilerleyen aşamalarda karşılaşılan hataların düzeltilmesi maliyet açısından kötü bir durumdur.Yazılımı yazan kişi ya da takım bir yazılımı yazarken bir hata yapmasa bile müşteri, gereksinimleri eksiksiz ve kesin belirtmekte zorlandığı için sonraki süreçte hatalar kaçınılmaz olur.Onun için yapılması gereken ürünü yaparken hem müşterinin sabırlı olması hem de yazılım yazan kişi ya da toplumun aceleci davranmaması, temkinli geriye dönüşler yaparak yazılımı kontrol etmelidir(ler).Her sürecin sonunda dokümantasyondan oluşan bir çıktı elde edilir ve her çıktı kendini takip eden sürecin girdisini oluşturur.Çıktı-girdi ilişkisi nedeniyle yaşanacak olası değişimlerin projeye maliyeti yüksek olacağı için süreçler arasında atlama ve büyük çapta bir geri dönüşe izin verilmez.Şelale modelinde temelden iyi başlama çok önemlidir.Her süreç bitişleri net bir şekilde tanımlanmış her süreç değişimde katı incelemeler,yoğun dokümantasyon ve yönetim onayı gibi kontrollerle proje yoğun bir denetim altında yapılır.Bundan dolayı bu modelin uygulanması katı bir disiplin gerektirir.Bu modelin temel amacı proje süresince değişikliklere izin verilmeyerek kapsam,zaman ve kaynaklar gibi faktörleri baştan güzelce sabitlemek ve büyük risk faktörü olan değişimi etkisiz kılmaktır.

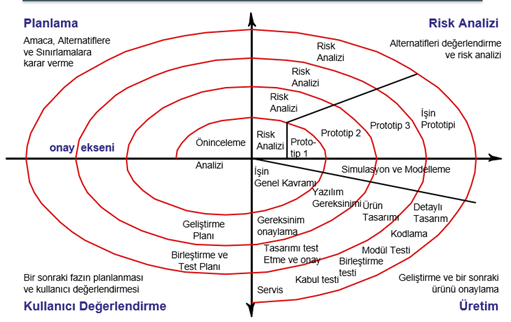
****

* **V Süreç Modeli:** V modelini şelale modelinin gelişmiş hali olarak düşünebiliriz.Şelale modelindeki gibi doğrusal bir yönde ilerlemek yerine,süreç adımları kodlama evresinden sonra yukarıya doğru eğim alır ve tipik V şeklini oluşturur.V model geliştirme yaşam çevriminin her bir evresi arasındaki ilişkileri gösterir.Yatay ve dikey açılar zaman veya projenin tamamlama birliğini ve soyut seviyeyi gösterir.V’nin sol tarafı üretim sağ tarafı ise sınama işleveleri ile ilgilidir.Sırası ile sol tarafta modül,alt sistem,sistem,sistem tanımları,gereksinimler bulunur sağ tarafta ise sırasıyla sınanmış modül,sınanmış alt sistem,sınanmış sistem,bitmiş sistem,sistem bulunur.Model aynı zamanda sınama işlemlerinde hata bulma durumunda nereye dönüleceğini de belirtmektedir.Sağ tarafta yapılan sınama işlemlerinde bulunan bir hata durumunda,yatay olarak karşısına gelen sol taraf işlevlerine dönülmektedir.V süreç modelinin temel çıktıları :Kullanıcı modeli,mimari model ve gerçekleştirim modeli olarak adladırılan üç alt modeldir.

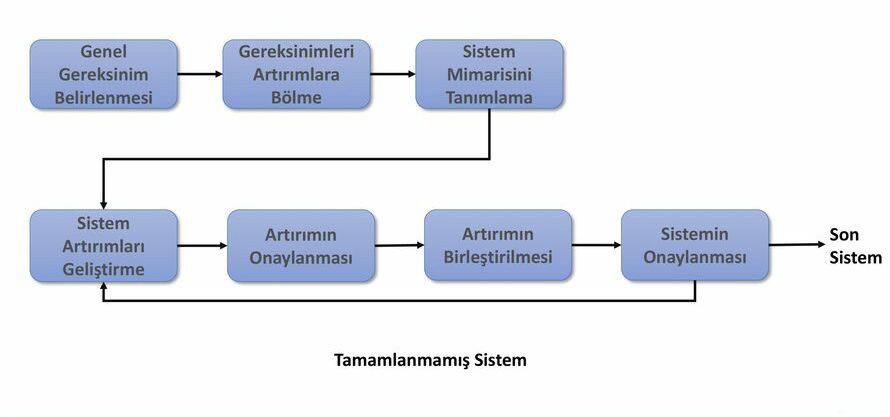
1. **Kullanıcı modeli:**Buradageliştirme sürecinin kullanıcı ile olan ilişkilerini tanımlamakta ve sistemin nasıl kabul edileceğine ilişkin sınama belirtimleri ve planları ortaya çıkarılmaktadır.
2. **Mimari model:**Burada sistem tasarımı ve oluşacak alt sistem ile tüm sistemin sınama işlemlerine ilişkin işlevler yer alır.
3. **Gerçekleştirim modeli:**Burada yazılım modüllerinin kodlanması ve sınanmasına ilişkin fonksiyonlar yer alır.

****

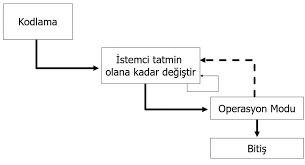
* **Helezonik Model:**Helezonik modelin üzerinde durduğu en temel konular risk analizi ve prototip üretmedir.Her döngü öncesinde içerisinde bulunan bölümün risk analizi yapılır ve ayrıca yeniden planlama yapılarak kısıt,ister ve hedefler yeniden hesaplanır.Bu modelin bize en çok katkı sağlayan yönü risk analizinin detaylı ele alınmasıdır.Risk analizinin detaylı ele alınmasından dolayı özellikle güvenlik yazılımlarının oluşumunda bu model kullanılmaktadır.4 sektörden oluşur bunlar sırasıyla planlama,risk değerlendirme ve azaltma,geliştirme ve doğrulama, kullanıcı değerlendirmesidir.Hepsini açıklamak gerekirse bir projenin temelini iyi atmak ve aşamanın başarımı için somut hedefler belirlenmesine *planlama* denir.Riskleri değerlendirerek ,bunları için azaltıcı eylemler gerçekleştirilmesine *risk değerlendirme ve azaltma* denir.Genel modeller içinden geliştirme için bir model seçilmesine *geliştirme ve doğrulama* denir.Ara ürün ile ilgili olarak kullanıcı tarafından yapılan sınama ve değerlendirmelere *kullanıcı değerlendirmesi* denir.Helezonik modelin bazı avantajları vardır bunlardan biri kullanıcıya katkısı 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.Risk analizi her sürecin sonunda olduğu için gerek proje sahibi gerekse yüklenici tarafındaki yöneticiler projeyi daha kolay izleme ve hak ediş planması yaparlar.

****

* **Artımsal Geliştirme Süreç Modeli:**Artımsal model bir takvime bağlı olarak projeyi kesim kesim, üstüne ekleye ekleye geliştirlp teslim etmeye dayanır.Artımsal modelin kısıtlı sayıda çalışanla işin yapılmasını sağlama gibi bir üstünlüğü vardır.Artırımsal geliştirme süreç modeli birinci adımda genel gereksinimler belirlenir,ikinci adımda gereksinimleri artırımlara bölme,üçüncü adımda sistem mimarisini tanımlama,dördüncü adımda sistem artırılmasının yapılması,beşinci adımda artırılımın onaylanması,altıncı adımda artılımın birleştirilmesi,yedinci adımda ise sistemin onaylanmasıdır.Eğer sistem bitmemiş ise dördüncü adıma geri dönülür.Her teslimle birlikte müşteriye bir değer döndüğünden,sistemin işlevselliği erken aşamalarda oraya çıkar.Bu da maliyet açısından faydalı bir model olduğunu gösterir.Önceki teslim sonraki teslime göre bir prototip vazifesi görür.Bu sayede projenin tümden batma olaslığı azalmış olur.



* **Kodla ve Düzelt Yaşam-Döngü Modeli:**Çok uzun programalar için kullanılabilir. İlk etapta yazılımın ilk sürümü gerçekleştirilir.Yazılım ürünü gerekleştirilir ve bu siste en son istenilen şekle gelinceye kadar devam eder.Belgeleme yoktur.Bakım safhası vardır ve belgeleme olmadığı için çok zordur.Bu anlatılanlara da bakılınca yapımı en kolay ama maliyeti çok yüksek olan bir modeldir.Emeklilik safhası da vardır.İşlem basamakları sırasıyla ilk adım ilk versiyonun gerçekleştirilmesi,ikinci adım müşteri gereksinimleri karşılana kadar düzenle,üçüncü adımda teslim sonrası bakımdır bu adımdan gerekli görüldüğü taktirde ikinci adıma geri dönülür.Dördüncü adım ise son adım olan emekliliktir.Bireysel geliştiriciler için uygundur,takım için uygun değildir.Ama tabi ki kullanılmasını sağlayan bazı avantajları da vardır. Bunlar şunlardır:Herhangi bir planlamaya ihtiyaç duyulmaz,çok küçük projelerde çok bir gerektirmediği için uygulanabilir.Bazı da dejavantajları vardır.Onlar da şöyledir:Kontrollü değildir,kaynak planlaması yoktur,bitiş süresi belli değildir,belgeleme olmadığı için hataların bulunması ve doğrulaması zordur,kodları düzeltmek maliyetli olabilir,kodları sonradan değiştirmek gerçekten uğraş gerektiren bir iştir.

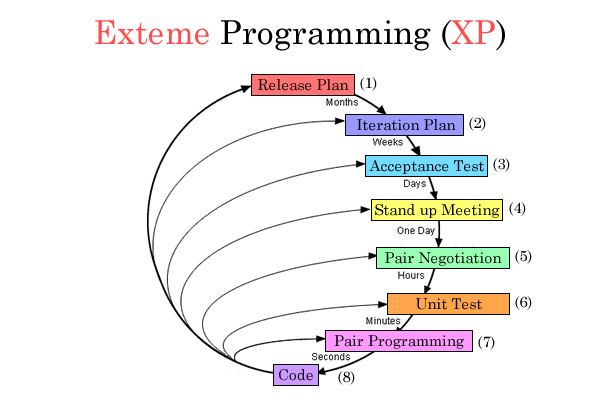
****

* **Çevik Yazılım Geliştirme:**2001 yılında Kent Beck ve 16 arkadaşı tarafından kurulmuştur.Bu modelleme biçimi kapsadığı değerler,prensipler ve pratikler sayesinde yazılımlara daha esnek ve kullanışlı biçimde uygulanabilir.Bu metoda göre:

1. Bireyler arasındaki etkileşim her şeyden ön plandadır.
2. Çalışan yazılım,detaylı bilgilerden daha önemlidir.
3. Müşteri ile iş birliği,sözleşmedeki kesin kurallardan daha önemli önceliktir.
4. Değişliklere uyum sağlayabilmek, mevcut planı takip etmekten daha önemli önceliktir.

En yaygın kullanılan çevik metodojiler 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 metodojiler vardır.

* **EXTREME PROGRAMMING**:En popüler çevik süreçlerden(Agile Process)birisi XP olarak bilinen Extreme Programming’dir.Kent Beck ve arkadaşları tarafından 1996 yılında Chrysler firmasında yapılan proje bünyesinde oluşan XP,ihtiva ettiği basit ama bir o kadar etkili yöntemlerle yazılım sektörüne bir canlılık katmıştır.XP kolay,grup içi iletişime çok önem veren,geri dönüşlerinin çok olup grup içi iletişimini hiç koparmayıp çok sağlam tutan bir yazılım geliştirme yöntemidir.XP’nin dört temel değeri vardır:İletişim,Basitlik,Geri Bildirim,Cesaret.Projenin başarıya ulaşabilmesi için ekibin iyi iletişimi olmalı.XP iletişim eksikliğini ortadan kaldırmayı amaçlar.XP’de iletişim yüz yüze olmalıdır.Yazılım ekibi ile yazılım kullanıcıları arasında iyi bir ilitişim olması önemli ve gereklidir.Basitlik XP’nin temel değeridir.Karmaşıklık XP’nin mantığına aykırıdır.XP esnek ve basit bir sistem gerçekleştirmeye çalışır.Geri bildirim ortaya çıkabilecek hataları oratadan kaldırır.Müşteri proje grubunun üyesidir.Müşteri yazılım ekibiyle sık sık buluşarak ilerde olabilecek büyük anlaşılmazlıklar daha önceden giderilir.En zoru cesarettir.Başarısızlıktan korkmak projenin hızının düşmesine neden olur.Yaptığınız işi müşteri memnun kalmazsa yeniden yazmalısınız.Yazılım geliştirmede kolaylığı ve esnekliği sağlamak için XP ,12 farklı pratiği ön görür;planlama oyunu,ekipte müşteri,basit tasarım,önce test,çiftli programlama,sürekli entegrasyon,kısa aralıklı sürümler,yeniden yapılandırma,ortak kod sahiplenme,metafor,kodlama standardı,haftada 40 saat.

****

* **SCRUM**:Jeff Sutjerland ve Ken Schawaber tarafından 1990’ların ortalarında geliştirilen,çevik yazılımla uygulanabilecek bir proje yönetim yaklaşımıdır.SCRUM,büyük projeler parçalara bölünerek ve her bir parçaya da “sprint” adı verilerek sonra da her bir sprintin teker teker geliştirildiği bir proje yönetimidir.Bu yönetim yaklaşımında amaç ekip içi bağı ve müşteri arasındaki bağı da hiç koparmayacak şekilde ilerlemektir.Bir diğer amaç ise iletişim kuvvetli olduğundan dolayı ve her bir sprint ayrı ayrı ele alınıp geliştireceğinden dolayı hata yapma olasılığımız ve maliyet olasılığımız düşük bir çalışma yapılmış olur.SCRUM’da da XP gibi temel prensipler vardır.Scrum 3 temel üzeriine dayanır.

1. **Uyarlama**:Proje,yapılacak bir takım değişimlere uyum sağlamalıdır.
2. **Denetleme**:Projenin ilerleyişi durumunun sık sık kontrol edilmesi.
3. **Şeffaflık**:Projenin ilerleyişi,sorunları ve gelişmeleri herkes tarafından görülebilir olmalıdır.

**SCRUM Kavramları:**

1. **Product Backlog:**Türkçesi ürün gereksinimleri listesi olan bu kavram,adından da anlaşılacağı üzere müşterinin ihtiyaçlarına göre gereksinimlerin hazırlaması veya düzenlenmesidir.Gereksinimler müşteriye bağlı olduğu için müşteri,gereksinimleri istediği zaman değiştirme hakkına sahiptir.
2. **Sprint Backlog:**Proje yukarıda da bahsettiğimiz üzere sprint denilen küçük kısımlara ayrılır.Scrum içesindeki tüm aktiviteler sprint içerisinde gerçekleşir.Bu sprintler 2-4 hafta arasında sürer.
3. **Daily Scrum:**Her gün ayakta 15 dakika yapılan toplantılardır.Bu toplantıda gelecek 24 saat planlanır.Gruptaki her üye dün ne yaptım,bugün ne yapacağım ve beni engelleyen sorun var mı var ise neler sorularına cevap arar.Eğer üyelerden biri bir sorunla karşılaşırsa *scrum master* duruma müdahele eder.

****

**SCRUM Temelleri:**

* **Roller**

1. **Product Owner:**Geliştime takımı ile müşteri arasındaki iletişimi sağlayan kişidir.Product Backlog’u oluşturan bu kişi her şeyi eline alır.İsterse sprinti dahi iptal edebilir.
2. **Scrum Master:**Scrumun kurallarını eksik bilerek bu projede ekibin kurallara sadık kalmasına yardımcı olur.Takımdaki sıkıntıları ve engelleri çözüme götürür.
3. **Daily Scrum:**Sprintteki işlerin hepsini bilen 5-7 kişi aralığında oluşan takımdır.Sprint Backlog’u oluştururlar.Kişilere tek bir görev verilmemiştir, herkes her işi yapabilecek donanımdadır.

* **Toplantılar:**4 aşamadan oluşur:**Sprint Planning,Daily Scrum,Sprint Retrespective**

**Sprint Planning** kısmında Product Backlog kısmında belirtilen gereksinimler, bu toplantıyla küçük parçalara ayrılır.Sprintlerin planaması yapılır.Bu toplantıya Product Owner,Scrum Master ve Geliştirme Takımı katılır.2-4 haftalık sprintler oluşturarak toplantıya son verilir.

**Sprint Review:**Türkçesi sprinti gözden geçirkmektir. Her sprintin sonunda sonunda yapılır.Sprintteki yazılanların , yazılım ürünü sahibinin gereksinimlerine uygun olup olmadığı kontrol edilir.Herhangi bir hatayla karşılaşılırsa ,bu hata düzeltilir.

**Sprint Retrospective**:Sprint boyunca yapılan işlerin kalitesi,doğruları ve yanlışları hakkında bilgi alış-verişi gerçekleşir.Sprint değerlendirilir.Bundan sonra neleri daha iyi yapabiliriz sorusuna cevap aranır.

****

Scrum’un günümüzde bu kadar popüler olma nedenleri şunlardır:

* Sürekli hem ekip kendi arasında hem de müşteriyke sıkı bir bağ oluşurduğu için değişen gereksinimlere hızlı bir şekilde tepki verilmiş olması,
* Bu sıkı bağ, proje maliyetinden büyük ölçüde tasarruf edilmesini sağlar,
* Karmaşık ve gereksinimleri tam belirlenmemiş olan projeler için güzel bir yapı olması,
* Ekip arasını iletişim düzeyinin sürekli olan toplantılarla yüksek olması , oluşan ya da oluşacak olan hataları hızlıca gidermeyi sağlar,
* Diğer yazılım geliştirme metodojilerinde de bahsettiğimiz gibi sürekli yinelemeli ve aktif bir yapıdır,
* Büyük bir projeyi sprint denilen küçük parçalara ayırarak hem hatadan kaçınmayı hem de projenin hızlıca ilerlemesi sağlanır,
* Her teknolojiye uyum sağlayabilir olmasıdır.

Halil İbrahim HATUN

200601014