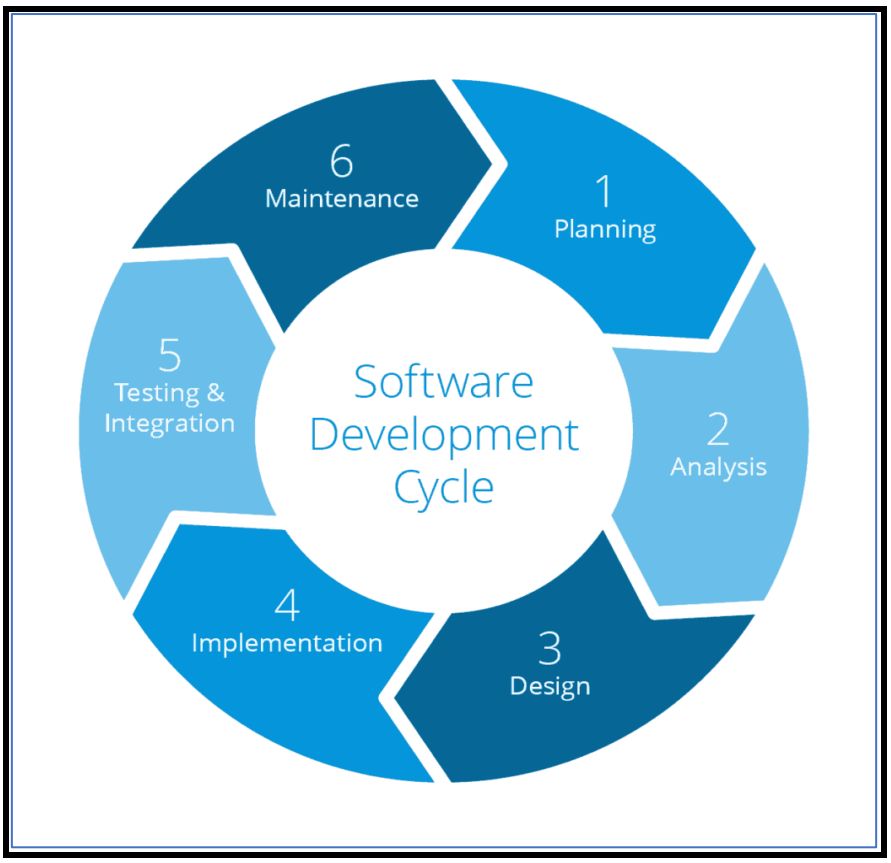
Yazılım Yaşam Döngü Modelleri
(Software Life Cycle Models)

1.Yazılım Yaşam Döngüsü Nedir? (Software Development Life Cyle)

Yazılımların geliştirilmeye başlandığı ilk günden itibaren aşama aşama hangi kurallara itibar edileceğini belirleyen bir şablondur.

Bu gösterime göre, geliştirilen yazılım versiyonunda tespit edilen hatalar giderilir, gerekli güncellemeler yapılır, kritik kararların alınması için toplantılar düzenlenerek yapılması gereken işlemler için fikirlerin masaya yatırılması sağlanır ve sonuç olarak kararlı bir yazılım ürününün üretilmesi hedeflenir.



Görsel 1.Yazılım Yaşam Döngüsü Modeli Şematik Gösterimi

1.1.Yazılım Yaşam Döngüsü Aşamaları

Yazılım yaşam döngüsü 6 aşamadan oluşmaktadır:

1.1.1.Planlama(Planning)

- Üretilecek olan yazılım ürünü için müşteriden bilgilerin toplanarak gereksinimlerin elde edildiği ve bu gereksinimlere göre planlamanın yapıldığı aşamadır.

1.1.2.Analiz(Analysis)

- Projenin bütün fonksiyonlarının ayrıntılı bir şekilde ele alındığı aşamadır.
- Bu aşamada sistem ihtiyaçları kesinleşir ve ihtiyaçlara yönelik talepler dokümante edilir.
- Projenin daha net ve somut hale gelmesini sağlar.
- Projenin ne kadar sürede tamamlanacağını belirler.
- Projeyle ilgili risk durumlarının ortaya atılmasını sağlar.

1.1.3.Tasarım(Design)

- Analiz aşamasından alınan sonuçlar detaylı olarak değerlendirmeye alınır ve bir proje alanı oluşturulur.
- Bir tasarım dokümanı hazırlanır. Bu dokümanda ; sistem tasarımı, veri modeli, kullanıcı arayüz tasarımları ve tasarım ayrıntıları gibi bilgiler yer alır.

1.1.4.Uygulama/Gerçekleştirme(Implementation)

- Projenin gerçekleştirilmeye başlandığı aşamadır.
- Bu aşamadan sonra herhangi bir analiz işlemi yapılmaz.
- Önceki aşamalarda belirlenen kurallara riayet edilerek işlemlere devam edilir.
- Alfa ismi verilen testler yapılabilmektedir.

1.1.5.Test ve Entegrasyon(Testing&Integration)

- Projenin geliştirilme aşamasından sonra belirli yöntemlere testlerin yapıldığı aşamadır.
- Projenin müşteriye teslim edilmeden önce hata ve sorunları ortadan kaldırma amacıyla yapılır.
- Bu aşamanın sonunda oluşan yazılım ürününün müşteriye teslim edilmesi kararlaştırılır.

1.1.6.Maintenance(Bakım)

- Projenin müşteriye teslim edildikten sonra oluşabilecek hatalara yönelik çalışmaların yapıldığı aşamadır.
- Bu aşamada ; yazılımda iyileştirmeler ve yeni fonksiyonların eklenmesi işlemleri gerçekleştirilebilir.

2.Yazılım Yaşam Döngü Modelleri

Yazılım yaşam döngüsü kapsamında birçok model geliştirilmiştir. Bu yazıda şu modeller incelenmiştir :

1. V Süreç Modeli
2. Evrimsel Model
3. Spiral(Helezonik) Model
4. SCRUM

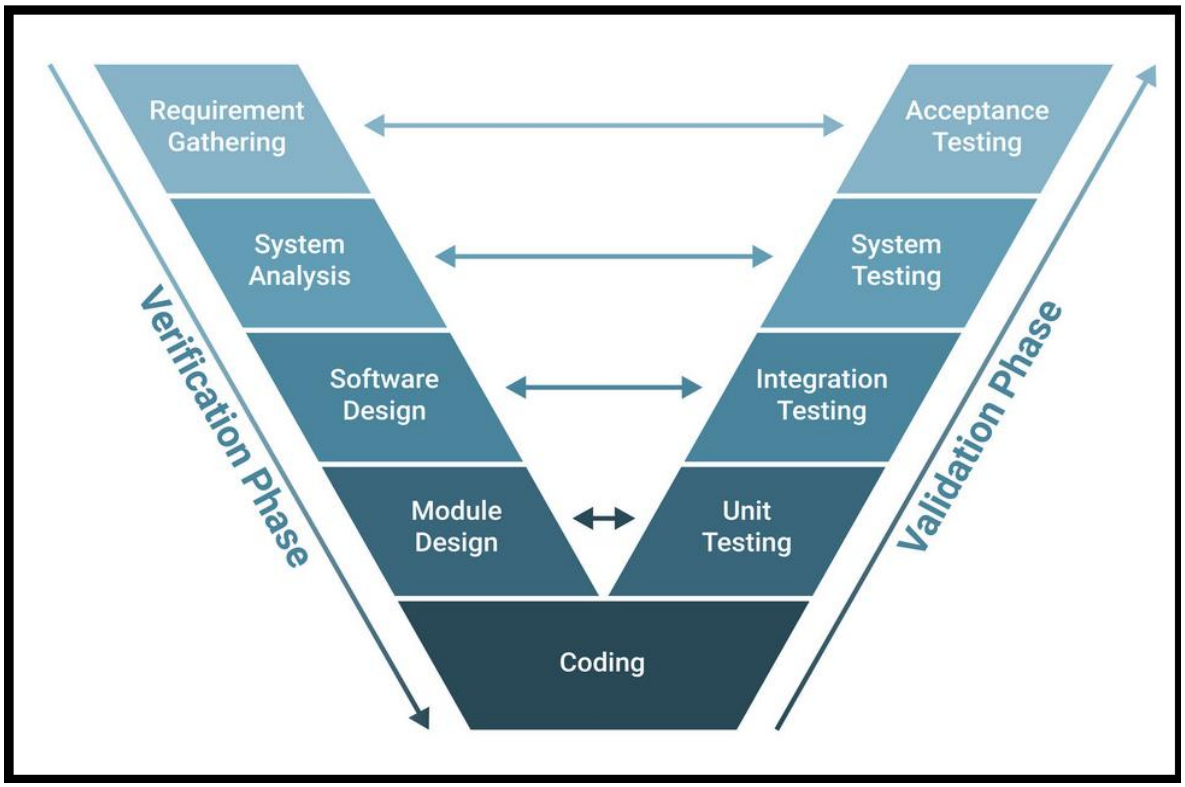
2.1.V Süreç Modeli

Bu modelde yazılım geliştirme döngüsü “V” şeklini oluşturarak ilerlemektedir.

Waterfall(Şelale) modelinin gelişmiş halidir.

Waterfall modelden farkı, yazılım sürecine başlanmadan önce bir test planının oluşturulmasıdır.

Belirsizliğin az olduğu ve iş tanımlamalarının belirgin olduğu projelerde kullanılmaktadır.



Görsel 2.V Süreç Modeli Şematik Gösterimi

2.1.1.V Süreç Modeli Aşamaları

2.1.1.1.Gereksinim Toplama/Analiz Aşaması (Requirement Gathering)

- Bu aşamada kullanıcı ihtiyaçları gözlemlenir ve gereksinimler ihtiyaçlara göre belirlenir.
- Belirlenen kullanıcı gereksinimlerine göre “Kullanıcı Gereksinim Dokümanı” oluşturulur.
- Kullanıcı gereksinim dokümanı , kullanıcının sistemden beklediği ; fonksiyonellik, arayüz işlevselliği, performans yeterliliği ve güvenliği gibi gereksinimleri ifade etmektedir.

Gereksinimleri toplamak için farklı metotlar bulunmaktadır. Bu metotlar :

- Kullanıcıyla yapılan görüşmeler,
- Anketler,
- Doküman analizleri,
- İncelemeler,
- Tek kullanımlık prototipler,
- Kullanıcı senaryoları,
- Statik ve dinamik incelemelerdir.

2.1.1.2.Sistem Tasarımı/Analizi Aşaması (System Analysis)

- Sistem mühendisleri tarafından analiz edilen ve kullanıcı gereksinim dokümanına göre sistemin işleyişinin anlaşıldığı aşamadır.
- Bu aşamada kullanıcı gereksinimleri kontrol edilir. Gereksinimlerde uyuşmazlık oluşması durumunda kullanıcının bilgilendirilmesi sağlanır ve uyuşmazlıklara çözümler üretilir.Üretilen çözümlere göre doküman güncellenir.
- Bu aşamada sistemin test edilmesi için dokümanlar hazırlanır.

2.1.1.3.Yazılım&Mimari Tasarımı Aşaması (Software Design)

- Yüksek seviyeli tasarım olarak ifade edilebilir.
- Bu aşamada mimarinin içereceği modüllerin listesi, her modülün özet olarak fonksiyonelliği, arayüz ilişkisi, bağımlıklar, veritabanı tabloları, mimari diyagramları ve teknoloji detayları sunulur.

2.1.1.4.Modül Tasarımı Aşaması (Module Design)

- Düşük seviyeli tasarım olarak ifade edilebilir.
- Bu aşamada tasarım sistemi daha küçük birimlere ayrılır ve her bir birim programcıya doğrudan kodlamaya başlayacak şekilde açıklanır.

Bu tasarımda :

- Veritabanı tabloları; tüm elemanlarını içerecek şekilde ve elemanların tiplerini ve boyutlarını gösterecek şekilde tasarlanır.
- Arayüz ayrıntıları karmaşık API referanslarıyla birlikte yer alır.
- Tüm bağımlılık konuları bulunur.
- Hata mesaj listesi yer alır.
- Girdi ve çıktılar eksiksiz bir şekilde bulunur.

Birim test tasarımı bu aşamada geliştirilir.

2.1.1.5.Birim Test Aşaması (Unit Testing)

- Birim test planları(Unit Test Plans(UTPs)) modül tasarım aşamasında geliştirilmiştir.
- UTP’ler birim seviyesinde veya kod seviyesindeki hataları ortadan kaldırmak için gerçekleştirilir.
- Bir birim, programda yer alan bir modül gibi en küçük varlık olarak nitelendirilebilir.

- Birim testi yapıldıktan sonra ortaya çıkan hatalar birimden izole edilir ve sorunsuz işleyen bir birim doğrulanmış olur.

2.1.1.6.Entegrasyon Test Aşaması (Integration Testing)

- Mimari tasarım aşamasında geliştirilir.
- Bu aşamada ; birbiriyle haberleşebilen birimler, birbirinden bağımsız şekilde test edilen ve oluşturulan birimler doğrulanır.
- Ortaya çıkan test sonuçları müşteri takımlarıyla paylaşılır.

2.1.1.7.Sistem Test Aşaması (System Testing)

- Sistem test planları, sistem tasarım aşamasında geliştirilir.
- Birim ve entegrasyon test aşamalarına benzemez.
- Sistem test planları,müşterinin iş takımı tarafından birleştirilmektedir.
- Sistem testi, uygulama geliştirme beklentilerinin karşılanıp karşılanmadığını belirlemeye yardımcı olur.
- Bu aşamada; uygulama işlevselliği, merkezi bağımsızlık ve iletişim için testler uygulanır.
- Sistem testi; fonksiyonel ve fonksiyonel olmayan gereksinimlerin karşılandığını doğrular.
- Sistem testi için : yükleme testi, performans testi, stres testi ve regresyon testi uygulanmaktadır.

2.1.1.8.Kullanıcı Kabul Testi (Acceptance Testing)

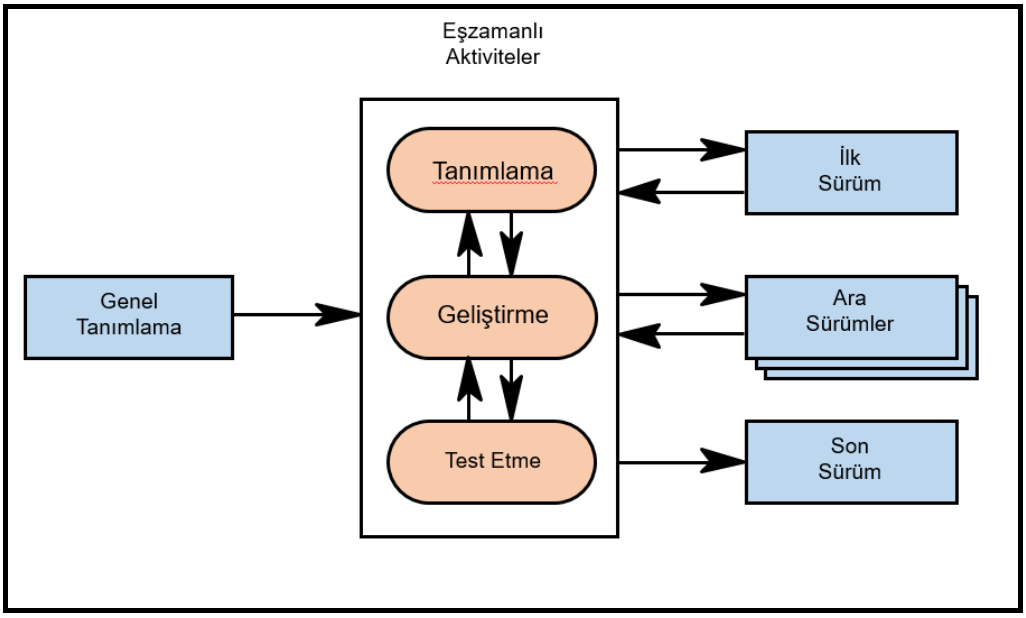
- Kullanıcı kabul testi planları (UAT:User Acceptance Test) gereksinim analizi aşamasında geliştirilir.
- UAT planları iş kullanıcıları tarafından birleştirilir.
- UAT gerçekçi veriyi kullanır ve üretim ortamını kullanıcı ortamına benzeterek çalışır.
- UAT kullanıcı gereksiniminin karşılanıp karşılanmadığını, gerçek zamanda kullanım için sistemin hazır olup olmadığını ve sistemin dağıtıma hazır olup olmadığını doğrular.

2.1.2.V Süreç Modeli Avantajları ve Dezavantajları

Avantajlar	Dezavantajlar
Kullanıcının projeye katkısını arttırmaktadır.	Aşamalar arasında tekrarlamalar bulunmaz.
Proje yönetimi ve takibi kolaydır.	Risk çözümleme uygulamaları bulunmamaktadır.
Verification ve validation planları ilk adımlarda belirtilir ve teslim edilecek olan tüm projelerde uygulanır.	İş ve ürün gereksinimleri değişiklik gösterebilmektedir.

2.2.Evrimsel Geliştirme Modeli(Evolutionary Development)

- Coğrafik olarak geniş alanlara yayılmış, çok birikimli organizasyonlar için önerilen bir modeldir.(banka uygulamaları)
- İlk tam ölçekli modeldir.
- Her aşama sonunda üretilmiş olan ürünler, üretildikleri alana göre tam işlevselliğe sahiptir.
- Diğer modellere göre ilerleyişi yavaştır.
- Modelin başarısı ilk evrimin(aşamanın) başarısına bağlıdır.
- Müşteri isteklerinin belirsiz olduğu durumlarda kullanılır.
- Müşteri ile yapılan çalışmalar doğrultusunda sistem için bir taslak oluşturulur.



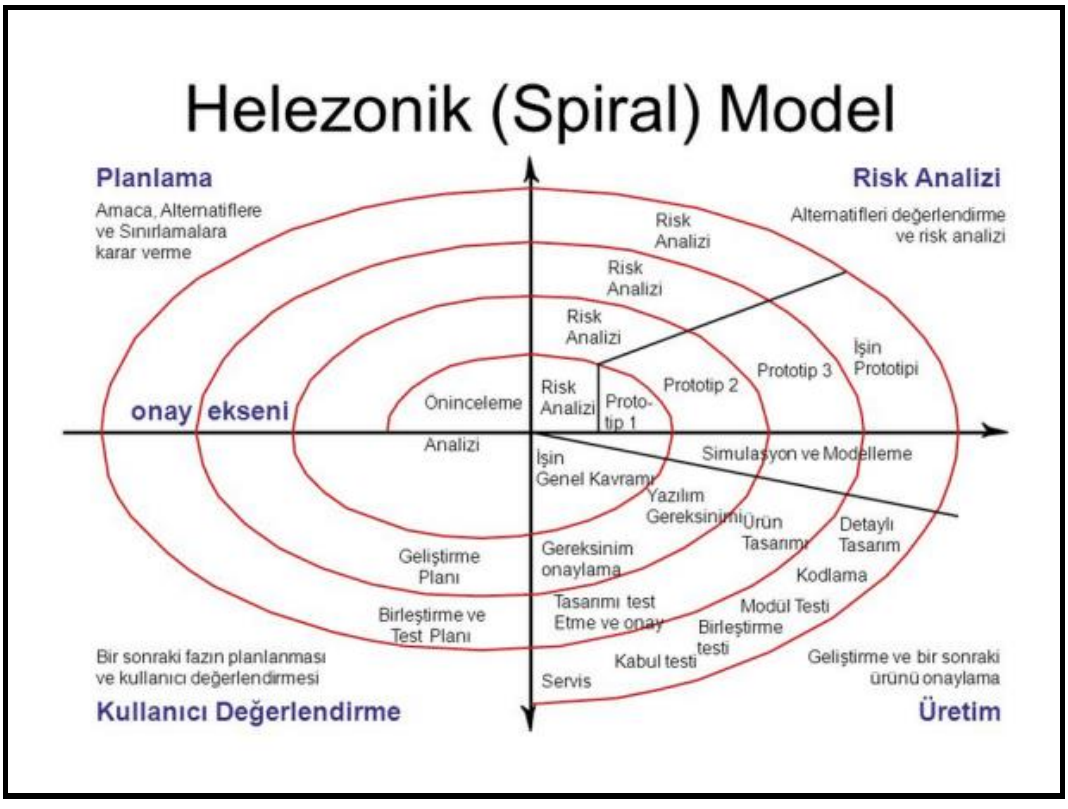
Görsel 3.Evrimsel Geliştirme Modeli Şematik Gösterimi

2.2.1.Evrimsel Geliştirme Modeli Avantajları ve Dezavantajları

Avantajlar	Dezavantajlar
Kullanıcıların kendi gereksinimlerini daha iyi anlamalarını sağlar.	Sürecin görünür kısmı azdır yani periyotlar halinde müşteriye teslim edilebilir bir ürün bulunmamaktadır.
Belirli periyotlar halinde yapılan değerlendirmeler erken aşamalarda ortaya çıkabilecek geliştirme risklerini ortadan kaldırır.	Sürekli yapılan yazılım değişiklikleri sistemin yapısına zarar verir.
Hata oranı azdır.	Bakım yapmak zordur.
	Yazılım gereksinimlerini güncellemek gerekebilmektedir.

2.3.Spiral(Helezonik) Model

- Bu model özellikle risk analizi ve prototip üretme konularının üzerinde durmaktadır.
- Ortaya çıkardığı prototipler sayesinde yinelemeli-arttırımlı bir yaklaşıma sahip olmuştur.
- Bu modelde her döngü bir aşamayı(fazı) ifade etmektedir.
- Her bir döngünün riskleri ayrı ayrı ele alınmaktadır.
- Her döngü sonu geldiğinde yeni bir planlama yapılarak kısıtlamalar istenir ve hedefler yeniden hesaplanır.
- Bu model, daha önceden geliştirilmiş olan yazılım bileşenlerinin tekrar kullanıldığı projeler için çok uygundur.
- Risk analizi detaylı olarak ele alınması zaman ve maliyetin daha kolay hesaplanmasını sağlamaktadır.
- Güvenlik alanında geliştirilen yazılımların oluşturulmasında bu döngü modeli tercih edilmektedir.
- Planlama, risk analizi, üretim ve kullanıcı değerlendirmesi aşamalarından oluşmakta ve bu aşamalar tekrarlanmaktadır.
- Modelin en büyük avantajı kullanıcının üretime katkısının olmasıdır. Bu katkı sayesinde ürün kullanıcının isteklerinin tamamına yanıt verebilme işlevine sahip olmaktadır.
- Bu model yürütülürken bazen sonsuza kadar gidebilme durumu doğabilmektedir. Bu durumda proje için hazırlanan dokümanların sayısı oldukça artış gösterecek ve proje karmaşıklığı da doğru orantılı olarak artış gösterecektir.



Görsel 4.Spiral(Helezonik) Model Şematik Gösterimi

2.3.1.Spiral Model Avantajları ve Dezavantajları

AVANTAJLAR	DEZAVANTAJLAR
Kullanıcı sistemi daha erken görebilmektedir.	Küçük ve düşük riskli projeler için çok pahalı bir modeldir.
Ürün parçalar haline getirilir ve öncelikler risk taşıyan parçalar çözümlenir.	Karmaşık bir yapıya sahiptir.
Birden fazla yazılım modelini içinde bulundurmaktadır.	Gereğinden fazla doküman oluşabilmektedir.
Oluşan hatalar vakit geçmeden ortadan kaldırılır.	Kontrat tabanlı yazılıma uyum sağlamaz.
Yazılım-donanım sistemi geliştirmeye olanak sağlar	Öznel risk değerlendirme deneyimine dayanır.

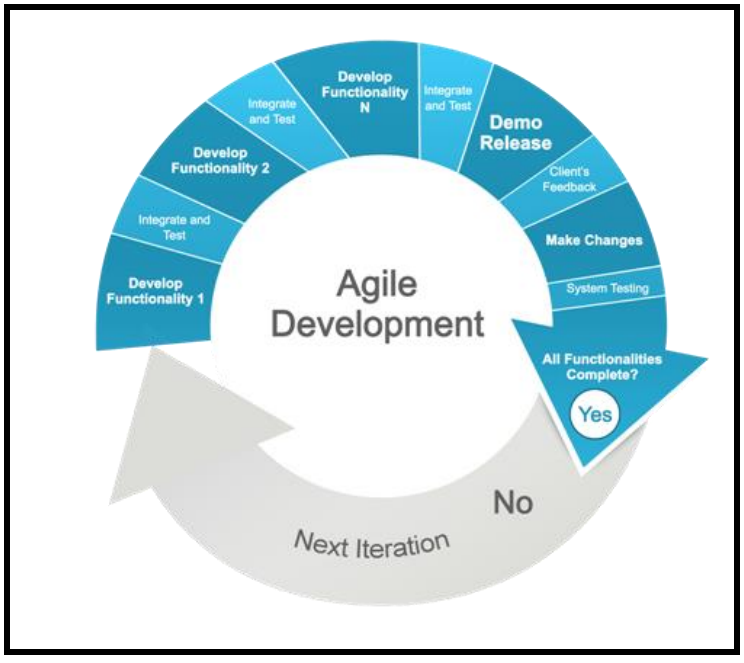
2.4.Çevik Yazılım Geliştirme ve SCRUM

Bir yazılım projesinin geliştirilme süreci uzun sürebilmekte ve stres altında gerçekleşebilmektedir.

Yazılım projesinin başarısı ve müşteri memnuniyeti yazılım projesini yürüten yönetim kadrosunun yeterlilik durumuyla doğru orantılı olarak şekillenmektedir.

Yazılım sektöründe gerçekleştirilen iş ve işlemlerle ilgili ortaya çıkması muhtemel sorunlar şunlardır :

- Yazılımda meydana gelen hataların zamanında fark edilmemesi,
- Yazılım ürününde istenilen değişiklik, güncelleme gibi durumlara geri dönüş verilmesinin uzun sürmesi,
- Yazılım versiyonlarının istenilen zamanda yayımlanmaması,
- Yazılım ürününe yönelik sonradan eklenmesi istenen taleplere göre sistemin kendi yapısını geliştirememesi.



Görsel 5.Çevik Yazılım Geliştirme Şematik Gösterimi

2.4.1.Çevik Yazılım Geliştirme

Yazılım sektöründe meydana gelen sorunlara yönelik olarak 1990'lı yılların sonlarına doğru **Çevik(Agile) Yazılım Geliştirme** metotları geliştirilmiştir.

Çevik metotların geliştirilme amacı yazılım projelerinde sonuca daha hızlı ulaşarak değişen ihtiyaçlara daha erken cevap vermektir.

2.4.1.1.Çevik Yazılım Geliştirme Metotlarının Özellikleri

- Yüksek performansa sahiptirler
- Yazılımlarda hata riski azdır.
- Daha ekonomik çözümler üretir.
- Verimlilik yüksektir.
- Esnektirler.
- Test odaklı yazılım geliştirmeyi destekler.
- Ekip çalışması ön plandadır.

2.4.1.2.Çevik Yazılım Geliştirme Avantajları ve Dezavantajları

AVANTAJLAR	DEZAVANTAJLAR
Öğrenmek kolaydır.	Kurumsal alanda uygulanması zordur.
Ekip üyelerinin motivasyonu çok yüksektir.	Ürün başarısı projenin başarısını etkilediği için kariyer riski bulunmaktadır.
Planlar adım adım yapılır.	Takım üzerinde hedefe ulaşma stresinden dolayı baskı bulunmaktadır.
Takım çalışması ön plandadır.	İhtiyaçlar sürekli değiştiği için çalışma süresi artış göstermektedir.
Müşteri memnuniyet oranı üst seviyededir.	
Planlar iterasyonlara göre ilerler.	
Projenin planlaması ve uygulanması eş zamanlıdır.	

2.4.2.SCRUM

Yazılım mühendisliğinde ve yazılım geliştirme sürecinde kullanılan bir çevik yazılım geliştirme metodudur.

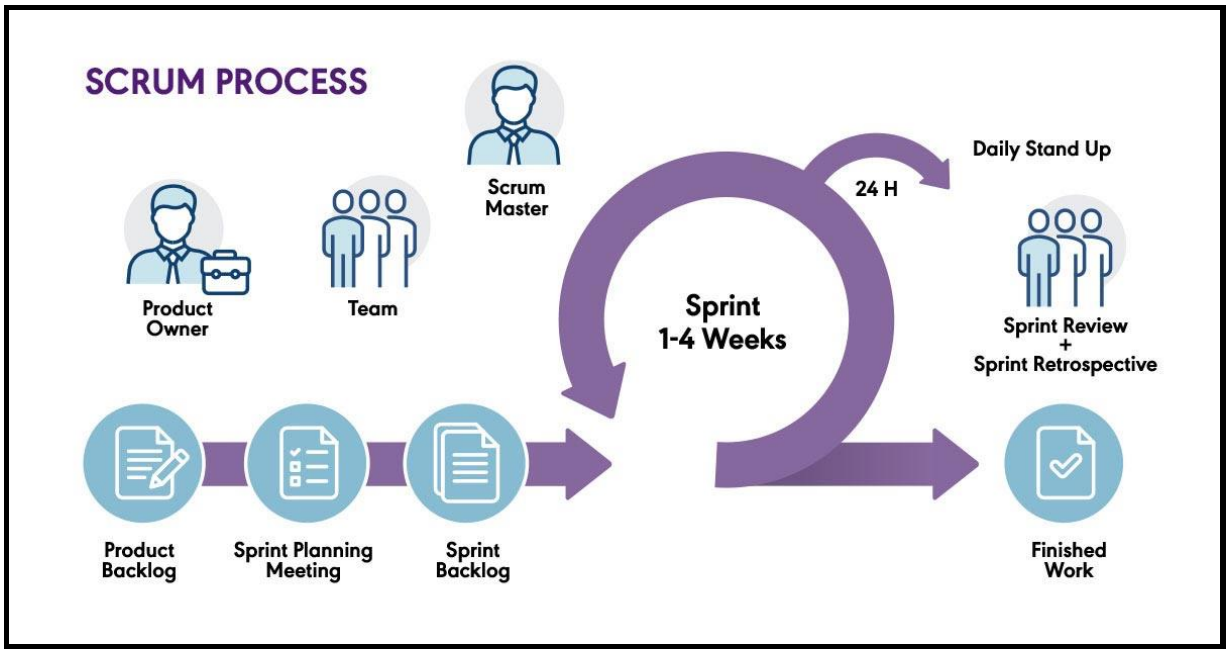
SCRUM metodunun genel özelliği ; gözleme dayalı olması, geliştirmeye odaklı olması ve tekrara dayalı olmasıdır.

SCRUM metodu, karmaşık yazılım proje süreçlerini küçük birimlere(sprint) böler ve işlemlerini bu birimler üstünde yaparak yazılım geliştirme sürecini tamamlamaya çalışır.

SCRUM metodu, karmaşık ortamlarda yazılım geliştirme işiyle meşgul olan yazılım ekipleri için kullanışlıdır.

SCRUM, gereksinimlerin kolaylıkla tanımlanamadığı ve karmaşık durumlara sahip olduğu tahmin edilen projeler için kullanışlı olabilecek bir metottur.

SCRUM metoduyla yürütülen bir projede ekipler günlük kısa toplantılar yaparak iş takibini sağlarlar.



Görsel 6.SCRUM Süreci Şematik Gösterimi

2.4.2.1.SCRUM Metodunun Prensipleri

SCRUM metodu yazılım projelerindeki karmaşıklıkları azaltmak adına 3 temel prensibin oluşmasını sağlamıştır:

2.4.2.1.1.Şeffaflık

Projedeki ilerlemeler ve karşılaşılan sorunlar günlük olarak tutulur. Bu sayede projenin herkes tarafından izlenebilir olması sağlanır.

2.4.2.1.2.Gözlem/Denetleme

Projenin parça halleri veya fonksiyonları periyotlar halinde müşteriye teslim edilir ve değerlendirmeleri yapılır.

2.4.2.1.3.Uyarlama/Uyumlama

Proje için oluşturulan gereksinimler yalnızca bir kez belirlenmez. Her teslimat yeniden değerlendirme sürecine alınır ve oluşan durum senaryolarına göre uyarlamalar yapılır.

SCRUM'ın amacı, projenin başında hayal ürünü olarak tasarlanan prototip ürünün; hızlı, ucuz ve kaliteli bir şekilde üretilmesidir.

Tasarlanan ürünün üretilmesi için müşteri tarafından tanımlanmış olan işlevler 2-4 haftalık "Sprint" isimli küçük dönemlere ayrılır ve bu dönemlerde üretilmeye çalışılır. Her sprint sonunda yazılımın bitmiş olan fonksiyonel kısımları müşteriye teslim edilmektedir.

SCRUM, Çevik Yazılım Geliştirme prensiplerini hayata geçiren bir metottur.

SCRUM, karmaşık projeleri daha iyi yönetmek için bir çatı görevi üstlenmektedir.

2.4.2.2.SCRUM'ın Tarihsel Temelleri

SCRUM, Ken Schwaber ve Jeff Sutherland tarafından 1990'ların başında geliştirilmiş ve esas fikirleri Schwaber tarafından ortaya konmuştur.

Çevik Yazılım Geliştirme Metodolojisi(Agile 2001) SCRUM sayesinde kendisine uygulama alanı bulmuştur.

Çevik Yazılım Geliştirme Metodolojisine göre :

- Süreçler ve araçlar yerine bireyler ve etkileşimler,
- Kapsamlı dokümantasyon yerine çalışan yazılım,
- Sözleşme pazarlığı yerine müşteri ile işbirliği,
- Herhangi bir plana takılı kalmak yerine değişimlere çözüm üretmek daha değerlidir.

2.4.2.3.SCRUM'da Roller

2.4.2.3.1.Ürün Sahibi(Product Owner)

Stratejik ürün geliştirmeden sorumludur.

Ürün vizyonunun tasarımı, iletişimi, özelliklerin tanımı ve önceliklendirilmesi,teslim edilen sprintin işlevselliği ve kabul edilebilirliği, şirketin ekonomisine uygun ürün tasarımı ve ana amaçları belirler.

Sadece ürün sahibi teslimat, işlevsellik ve maliyet gibi kararlardan sorumludur.

2.4.2.3.2.SCRUM Takımı (SCRUM Team)

Ürün sahibinin isteklerine ve sıralamasına uygun olarak ürün işlevselliğini sağlamaktan ve belirlenmiş olan kalite standartlarına uyarak ürünü teslim etmekten sorumludur.

Sprinte dahil edilecek işlevlerin seçiminden ve miktarına karar verirler.

Oluşabilecek iyi-kötü sonuçlar takım elemanlarına mal edilmez.Takım bir birim olarak algılanır. Bir takım 5 ila 9 kişiden oluşmaktadır.

Bir SCRUM takımında olması gereken özellikler şunlardır:

- Kendi kendine organize olabilmeli
- Küçük olmalı
- Multidisipliner bir yapıya riayet etmeli
- Özerk(özgür) olmalı

2.4.2.3.3.SCRUM Yöneticisi (SCRUM Master)

SCRUM'ın başarılı olmasını sağlamaktan sorumludur.Bunun için takım ile birlikte çalışır fakat takıma tabii olmaz.

SCRUM kurallarını bildirir,uyumu kontrol eder,toplantı moderatörlüğü yapar ve SCRUM sürecindeki düzensizlikler ile ilgilenir.

Takımın önündeki engelleri kaldırmak amacıyla Impediment Backlog (Engel Biriktirme Listesi) kullanır ve bu anlamda sorumluluk üstlenir.

Karşılaşılabilecek engeller:

- Takım içi iletişim eksikliği,
- Kişisel çelişkiler,
- Takım-Ürün Sahibi arasındaki iletişim,
- Dış kaynaklı rahatsızlıkların(ek işlevler) giderilmesi

SCRUM ustasının yazılım takımına karşı yürütme yetkisi bulunur fakat şeflik yetkisi bulunmamaktadır.Bu yüzden hüküm veremez ve disiplin kovuşturması yapamaz.

SCRUM Ustası; süreçten sorumlu,takımın arkadaşı ve yardımcısı,sürecin doğruluğunun denetleyicisi, takım ile birebir bağlantılı ve takımın yanında çalışan kişidir.

SCRUM ustasının görevleri:

- SCRUM'ı uygulamak ve ürün sahibi ile yazılım takımına destek olmak,
- Engelleri kaldırmak ve süreçteki sapmaları düzenlemek,
- Takıma hizmet etmek ve meslektaş bir yönetim tarzı ile yönetmek.

2.4.2.3.4.Kullanıcı(User)

Yazılım ürününün kullanıcısıdır.

Ürünün nasıl kullanılacağı hakkında kendi bakış açısıyla fikirlerini verir.

Proje yürütülürken esas hedef kitlesidir.

Kullanıcı, sprint başlangıcı ve sonunda ürünü test etmek amacıyla proje içerisinde yer alır ve geri bilgi akışı(feedback) sağlar. Vermiş olacağı feedback ile :

- Kullanılabilirlik(Usability) konusunda takıma değerli ipuçları verir.
- Takım ve ürün sahibi kullanıcı bilgileri doğrultusunda Sprint planını uyarlar ve son durum tekrar kullanıcı tarafından test edilir.

2.4.2.3.5.Yönetici(Manager)

SCRUM'ın yapısal çerçevesini belirler.

Ek kaynaklar(ressourcen) oluşturur.(yer,araç-gereç vs..)

SCRUM ustasına destek olur ve mesleki personel organizasyonu sağlar.

SCRUM ekibini dış taleplerden korumaktan sorumludur.

Görevi : projenin çerçevesiyle ilgilenmek ve SCRUM ustasının proje alanı içerisindeki belirlemiş olduğu problemlerin çözümüyle meşgul olmak.

2.4.2.3.6.Takım ve Etkileşim

Rol dağılımı sırasında takım kendi kendisini organize edebilmektedir.

2.4.2.3.7.Rollerin İstismar Edilme Riski

SCRUM'da proje yöneticisi bulunmaz.Bu sebeple deneyimsiz bir SCRUM ekibinde SCRUM ustası ve ürün sahibi bu rolü üstlenmeye çalışabilmektedir.Bu durum roller içinde tehlike yaratır ve takımın özerklik bilincine zarar verir.Dolayısıyla SCRUM'da sapmalar yaşanabilir.

Bu tehlikeyi azaltmak için SCRUM ustası ve ürün sahibi SCRUM Expert'inden(Uzman) yardım alarak bu sorunun üstesinden gelmelidir.

2.4.2.4.SCRUM'da Toplantılar

2.4.2.4.1.Sprint Planlama Toplantısı 1

Bu toplantıda "NE?" sorusu ön plandadır.

Ürün sahibi kendi yazılım takımına ürün içeriğinde(Product Backlog) kararlaştırılmış olan kullanıcı hikayelerini(User Stories) öncelik sırasına göre belirtir ve gereksinimler takım tarafından netleştirilip yazılı olarak kaydedilir.

Kullanıcı işlevsellik konusunda bilgiler verir.

Ürün sahibi ve takım sprint içerisinde olması gerekli olan işlevler ve kriterler üzerinde anlaşma yaparlar.

Toplantının amacı kullanılabilir yazılım etmek için kararlar almaktır.Yazılım;test edilmiş,entegre olmuş ve kullanıcıya açılmış olmalıdır.

Sprintin kabul edilme şartları yazılımın kabul kriterleri ile etkileşimlidir.Alınan kararlar toplantı sonunda net bir şekilde belirlenir ve belirlenen fonksiyonların içeriği incelenir.

Açıklamalar yapıldıktan sonra SCRUM ustası takımına bir sonraki sprint de kaç tane kullanıcı hikayesi olacağını sorar ve tek tek değerlendirmeler yapılarak dış baskı olmadan erişilebilirliği incelenir.

Bu toplantı haftalık olarak yapılır ve süresi 60 dakikadır.

2.4.2.4.2.Sprint Planlama Toplantısı 2

Bu toplantıda "NASIL?" sorusu ön plandadır.

Yazılım takımı kullanıcı hikayelerinden hangilerinin sprinte dahil olacağını bilmektedir ve uygulamanın teknik boyutu açıklanır.

Bu toplantı takımın kendi sorumluluğunda organize edilir.

Genelde küçük gruplar oluşturulur ve yapı,test,açık gibi konulara açıklık getirilir.Alınması beklenen sonuç görevlerin 1 günlük süreyi aşmayacak şekilde bitirilmesini planlamaktır.

Görevler, belirlenmiş olan kullanıcı hikayelerinden yola çıkılarak duvara veya beyaz tahtaya asılır.Bu sayede hangi görevin işlemde veya sırada olduğu bilinir.

Bu toplantı haftalık olarak yapılır ve süresi 60 dakikadır.

Sprint planlama toplantısı 1 ve 2 aynı gün içerisinde yapılmalıdır.

2.4.2.4.3.Günlük SCRUM (Daily SCRUM)

Her iş günü başlamadan önce 15 dakikalık bilgi paylaşımı için günlük SCRUM toplantısı yapılır.

Bu görüşmede herhangi bir problem değerlendirilmez.Yalnızca 3 tema işlenir : dün ne yaptım,bugün ne yapacağım,beni ne engelliyor.

Eğer belirlenen görevin 1 günde bitmesi mümkün değilse görev parçalanıp takıma dağıtılır.

15 dakika içinde bazı sorulara cevap bulunamadığı durumlarda SCRUM ustası not alıp bir sonraki toplantıya taşır veya kendisi bir çözüm yolu bulur.

Görsel 7.Günlük Scrum Toplantısı

2.4.2.4.4.Sprint Değerlendirmesi (Sprint Review)

Sprintin sonunda takım tarafından yapılır ve başlangıçta belirlenen hedeflerin kapsamında olup olmadığı ürün sahibi tarafından değerlendirilir.

Eğer teslim edilen işlevde eksiklik varsa kullanıcı hikayesi yeniden ürün sahibi tarafından ürün içeriğine gönderilir ve öncelik sırası verilir.

Kullanıcının değerlendirmeye katılması işlevin testi,ürün tasarımı ve kullanıcı bakışı açısından çok önemlidir.Kullanıcı hikayelerinde bir eksiklik varsa SCRUM ustası tarafından not alınır ve ürün sahibi tarafından ürün içeriğine aktarılır.

Sprint değerlendirmesi 1 ay biter fakat en fazla 4 saat sürmelidir.

2.4.2.4.5.Sprint Retrospektif (Geçmişe Bakış)

Geçmişe bakış toplantıları olarak da bilinir.

Sprint değerlendirme toplantılarından sonra ve sprint planlama toplantılarından önce yapılırlar.

Geçmiş sprintteki tecrübeler masaya yatırılır ve yapılacak iyileştirmeler belirlenir.

Scrum yönetiminin en önemli özelliklerinden birisi bu süreçte suçlu/suçsuz eleştirilerinin yapılmamasıdır.

2.4.2.5.SCRUM ve Sprint

Sprint ürünün geliştirildiği yapıdır.

Her sprint başlangıcında 2 kez sprint planlama toplantıları yapılır.

Planlamada belirtilen ürün işlevleri sprintin sonunda teslim edilir.

Geliştirme ekibinin çalışmaları,görev panosuna dayanır ve kullanıcı hikayeleri önceliklerine göre ele alınarak tüm takım bir hikaye üzerinde çalışır.Böylece katılımcılar hangi işlevin geliştirildiğini bilir ve karşılıklı katkılarda bulunur :

- Yazılımcı,testçi ile birlikte testi başlatır.
- İşyeri analisti kullanıcılara işlevin parçalarını tanıtır.
- Ürün geliştirmedeki gerekli kriterler işlevde mevcutsa yazılım takımı yeni işlev üzerinde çalışmalarına başlar.
- Sprint süreci teslim edilecek işlevin üretilmesini hızlandırır.
- Belirlenen hedefe erişim ve sorumluluk sadece yazılım takımındadır.
- Dışarıdan gelebilecek ekstra görev ve taleplerden ve takımı hedeften şaşırtacak her türlü engelin kaldırılmasından Scrum Ustası sorumludur.
- Geliştirme ekipleri içerisinde yaşanabilecek problemlere karşı Scrum Ustası müdahele etmelidir.

Sprintin uygulanma zamanı ve süresi her zaman aynı olmalıdır.Bir sprint en az 1 hafta ve en fazla 4 hafta sürmelidir.Eğer hedeflenen sonuca Scrum sürecinde erişmek imkansızsa takım veya ürün sahibi sprinti durdurabilir.Sprintin durdurulması durumunda inceleme yerine retrospektif yapılır ve gelecek sprintin planlaması yapılır.

2.4.2.6.SCRUM Yapıtaşları

2.4.2.6.1.Ürün Gereksinim Dökümanı(Product Backlog)

Proje süresince yapılması gereken işlemlerin yer aldığı bir dokümandır.

İsteklerin oluşturulması ve yönetilmesi için merkezi bir belge görevi üstlenir.

Teslim edilecek işlevsel elementler yönetilir.

Kullanıcı hikayelerinden oluşur ve kullanıcının bakış açısından izler taşır.

Toplanan kullanıcı hikayeleri ürün sahibi tarafından öncelik sırasına göre düzenlemeye tabi tutulur ve gerçekleştirme zamanı geldiğinde takım tarafından uygulamaya konur.

Değişim talepleri yalnızca bu dokümanda yer alır.Ekleme,çıkarma gibi işlemler ürün sahibi tarafından yapılır.

Ürün içeriği hiçbir zaman eksiksiz bırakılmaz.Önceden tanımlanmış ve iyi anlaşılmış gereksinimleri içerir.

Ürün içeriğine eklenen istekler mesleki ve kullanıcı odaklı olmalıdır.İyi bir kullanıcı hikayesi üç soruya yanıt vermelidir:

- Kim? -→ Kullanıcı olarak,
- Neyi? → Bu işlevi,
- Neden İstiyor? → Şu faydalar için istiyorum(….)

2.4.2.6.2.Sprint (Koşu) Dökümanı (Sprint Backlog)

Ürün gereksinim dokümanından elde edilen iş ve görevleri barındırır.

Buradaki işlerin amacı : sprintin sonunda son ürünün bir işlevselliği ya da çalışabilir bir parçayı elde etmektir.

Görevleri bitirmek için 4 sütunlu bir görev tahtası kullanılır:

- 1.sütunda : Sprint'de bulunan iş parçacıkları("Stories"),
- 2.sütunda : Görevler ("ToDo"),
- 3.sütunda : Çalışma Parçacıkları("In Progress"),
- 4.sütunda : Teslime hazır iş parçacıkları("Done") bulunur.

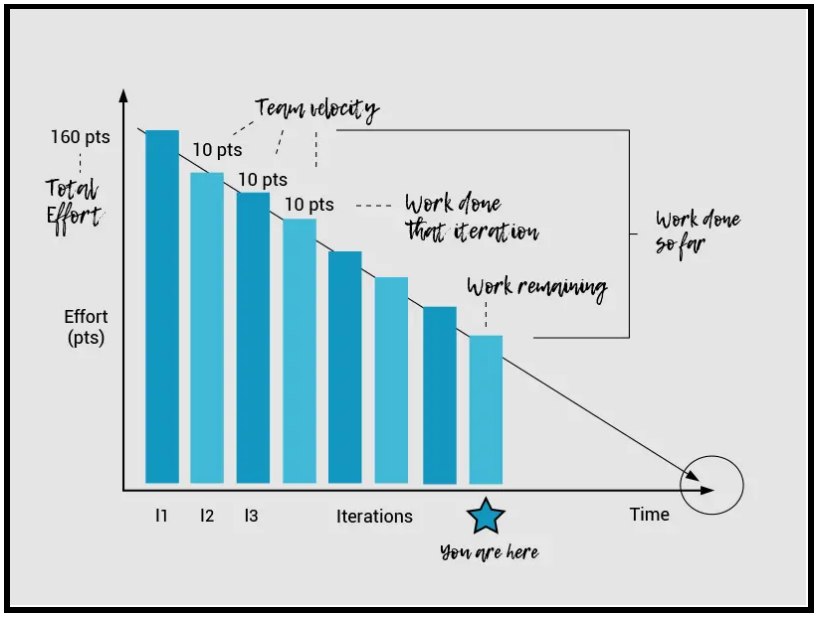
Yazılım ekibi elemanları Günlük Scrum toplantısında bir önceki gün hangi görevin üzerinde çalışıldığını ve bitip bitmediği hakkında bilgiler verir. Bir günde bitmeyen görevler kırmızı nokta ile işaretlenir. Böylelikle engeller kolay bir şekilde tespit edilmiş olur

2.4.2.6.3.Sprint Kalan Zaman Grafiği (Burndown Chart)

İş-bitim grafikleri yapılmış olan ve geri kalan çalışmayı görselleştirmek için kullanılır

Bir sprint kalan zaman grafiği x ekseninde günlük zamanı y ekseninde ise bitirilmemiş görevleri gösterir.Bu grafik sprintin belirtilen zaman birimi içinde daha iyi tahmin edilmesini sağlar.

Grafik eğilimleri merdiven şeklini alır.Her azalma değeri bir hikayenin bitişini gösterir.



Görsel 8.Burndown Chart

2.4.2.6.4.Engel Biriktirme Listesi (Impediment Backlog)
Scrum ustası tarafından kısa bir problem tanımı ve tarih etiketiyle oluşturulur.Ek olarak Günlük Scrum sonunda Scrum Ustası karşılaşılan engelleri listeye ekler.

2.4.2.6.5.Bitti Tanımı (Definition of Done)
Bir kullanıcı hikayesinin uygulanması ve yazılımda yer alan etkinliklerin kontrol listesidir.

Bitti tanımı projenin başlangıcında katılımcılar tarafından kararlaştırılır.

Sprintin başlangıcında görevlerin sayısı ve kapsamı hakkında yardımcı olur.Tüm hikayelerde uygulanmak zorunda değildir.

Bitti tanımı,sprintin sonunda belirlenmiş olan bir hikayenin ayrıntılı taleplerini belirttiği için sprintin kabul edilmesine hizmet etmektedir.

2.4.2.7.SCRUM Araçları
SCRUM'ın uygulanması ve SCRUM sürecinin kolaylaştırılması amacıyla kullanılabilecek araçlar(uygulamalar) aşağıdaki gibidir :

- Agilo
- Pangoscrum
- AgileZen
- Tinypm
- ThoughtWorks Studios
- Greenhopper
- VersionOne
- ScrumWorks Pro
- Banana Scrum
- ScrumTable