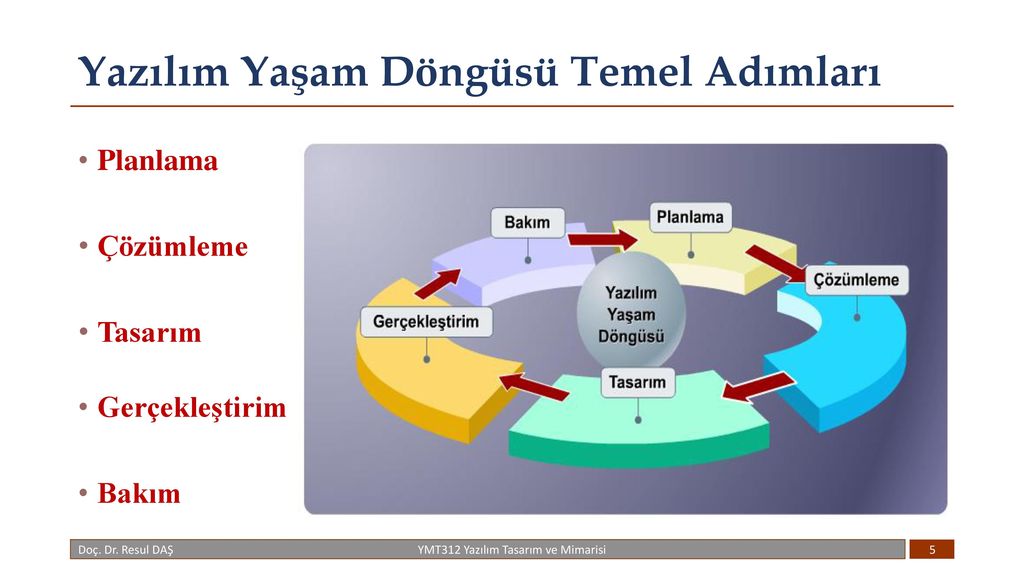
YAZILIM GELİŞTİRME YAŞAM DÖNGÜSÜ (SDLC) ZELİHA ÇABUK  
 200601068

Yazılım yaşam döngüsü , herhangi bir yazılımın,üretim aşaması ve kullanım aşaması ile birlikte olmak üzere geçirdiği tüm aşamalar biçiminde tanımlanır.SDLC aşamalardan oluşmaktadır ve bu aşamalara geri dönüp onlar üzerinde oynama yapılmaya yani değişikliğe açık olmalıdır çünkü yazılım canlıdır ve sürekli bir değişim içinde olabilir.Bazı yazılım yaşam döngüsü temel adımları:



1.Aşama:Planlama-Gereksinim (Requirements)=Gereksinimler ve ihtiyaçlar belirlenir.Projede oluşabilecek sorunlar ,fırsatlar ve projenin bitim zamanının net resmini bize verir.Fizibilite çalışmasının yapıldığı ve projenin oluşturulduğu aşamadır.

2.Aşama:Çözümleme-Analiz(Analysis)=Bu aşamada mevcut sistemdeki işler ayrıntılı biçimde incelenir ve temel sorunlar ortaya çıkarılır.Bu aşamada temel UML diyagramları çizimine başlanır.

3.Aşama:Tasarım(Design)=Çözümleme aşamasından sonra gereksinimlere cevap verecek sistemin temel yapısının oluşturulduğu çalışmaların olduğu aşamadır.2 tür çalışma vardır; -Üst seviye mimari tasarım: Mevcut sistemin değil ,önerilen sistemin yapısı anlatılır. Olası değişikler önerilir. -Detaylı Tasarım: Yazılımı içeren bileşenleri ve bunların ayrıntıları incelenir.

4.Aşama:Gerçekleştirim(Implemantation)=Bu aşamada kodlama,test etme ve kurulum çalışmaları yapılır. SDLC’nin en uzun aşamasıdır.

5.Aşama:Bakım(Maintenance)=Yazılım dağıtıldıktan ve müşteri teslimininden sonra hata düzeltme,yükseltme ve geliştirme olmak üzere etkinlikler gerçekleştirilir.

Yazılım geliştirme yaşam döngüsü (SDLC),müşteri beklentilerini karşılayan yüksek kaliteli yazılımlar üretmeyi amaçlar.Bunun için SDLC’nin farklı modelleri vardır. Birden çok modelin olmasının sebebi ;yazılım projesinin büyüklüğü,yazılım projenin kimler için kullanılacağı gibi durumlarla sıralanabilir.

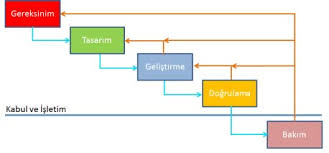
YAZILIM YAŞAM DÖNGÜ MODELLERİ

\*GELİŞİGÜZEL MODEL = Bir yöntem veya planlamadan oluşmaz.Bu yüzden bir model olarak kabul etmek pek doğru olmaz.Geliştiren kişiye bağımlıdır belirli süre sonra yazan kişi bile karmaşıklıktan dolayı anlama zorluğu çekebilir. İzlenebilirliği ve bakımı oldukça zordur.Günümüzdeki karmaşık yazılımlar için kullanışlı olmadığı için günümüzde kullanımı yoktur.Daha çok 60’lı yıllarda tek kişilik üretim ortamlarında kullanılırdı.Basit programlama yöntemidir.

\*BAROK MODEL = Barok modeli 70’li yıllarda yazılım dünyasına girmiştir.Yazılım yaşam döngü temel adımlarını(planlama ,çözümleme...) doğrusal bir şekilde ele alır.Döngü yoktur aşamalar arası geri dönüşün nasıl yapılacağı bilinmemektedir.Belgeleme ayrı bir süreç olarak ele alır tasarım bittikten sonra yapılır yani dökümantasyon son olarak çıkarılır.Günümüzde kullanımdan çıkmış olsa dahi bazı projelerde(göstermelik projelerde) dokümantasyon sonda yapılır.

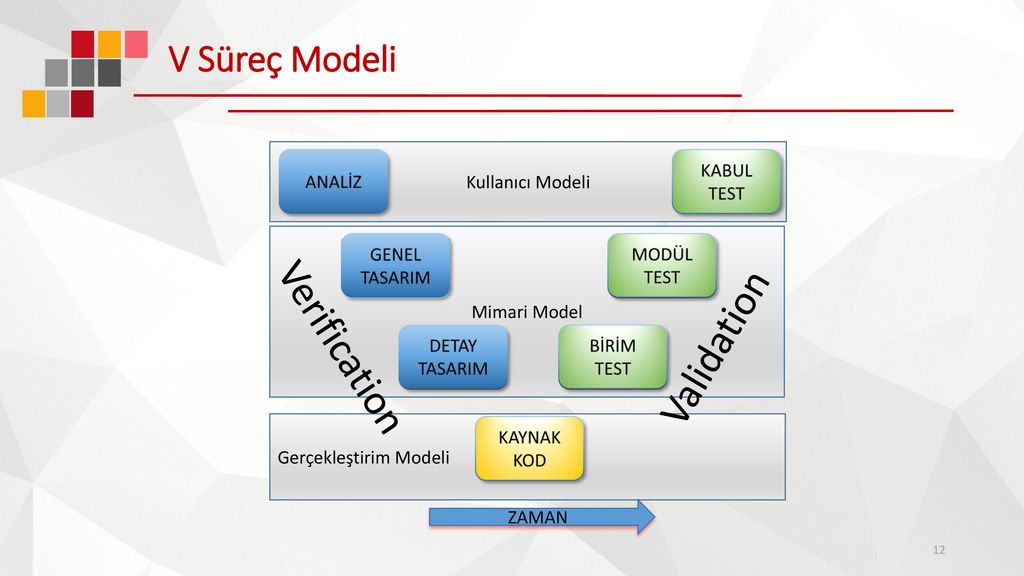
\*ÇAĞLAYAN (WATERFALL) MODELİ

Çağlayan model,tanınmış en eski ve gemişte en popüler olan yazılım yaşam döngü modelidir.Bu model geleneksel yazılım geliştirme modeli olarak da bilinir.Bu model diğer güncel modellerin de temeli diyebiliriz.Bu metotta yazılım aşamaları en az birer kez tekrarlanır.Çağlayan modelde gereksinimlerin kesin olması beklenir.Fakat böyle bir şeyin söz konusu olması beklenemez çünkü yazılım sahası dinamiktir ve sürekli değişikler olmaktadır bundan dolayı bu değişikleri sürekli olarak tanımlamak zorundayız ama çağlayan model bunu desteklemez. Çağlayan model üretimi az zaman gerektiren durağan ve kesin olan projeler için uygundur.



Çağlayan modelde adım adım ilerlenir bir aşama bitmeden diğer aşamaya geçilmez.Ama kesin olarak bitti gözüyle bakılan aşamalarda eksik veya hata olunca oraya geri dönmek zorunludur.Bu yanlışlar gereksinim ,tasarım aşamasında olduğunda küçük maliyetler ile hallolurken; geliştirme,test(doğrulama),bakım aşamalarında farkedildiğinde bizlere çok büyük maliyete mal olabilir. Barok modelinden farklı olarak dokümantasyon her adım sonunda çıkarılır ve aşama test edilmediği sürece o aşama bitti olarak bakılmaz .

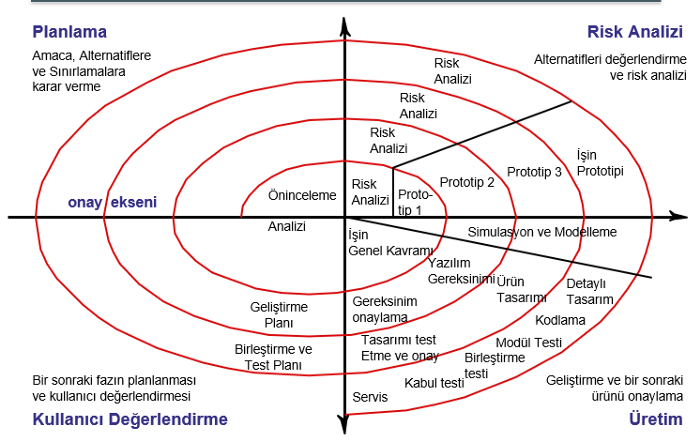
\*V SÜREÇ MODEL



Modele V süreç modeli denilmesinin sebebi adımların V şeklini oluşturmasıdır.V süreç modeli şelale modelinin biraz daha gelişmiş halidir.Adımları şelale modelinin aynısı olmasına ve sırayla gitmesi yönünden şelale modelinin aynısıdır.Kodlama yapıldıktan sonra her adım test edilir en sonunda kullanıcıya test ettirilir.Yani iki kısımdan oluşmaktadır . Birinci kısımda gereksinimlere göre kullanıcı modelini hedef alınarak aşamalar sırası ile uygulanır; ikinci kısımda ise her üretilen modülün testi yapılır.Sonrasında modüllerin birbiri ile çalışma durumu ,entegrasyon testleri yapılır ve kabul testleri aşamasında müşterinin kabul edip etmemesi ile ilgili müşteri gidilir.Müşteri uygulamayı test eder. Bu yönüyle V süreç modeli çağlayan modelinden ayrılır. V süreç modeli test kısmına çok önem vermektedir.Bundan dolayı belirsizlikler V süreç modelinde oldukça azdır.

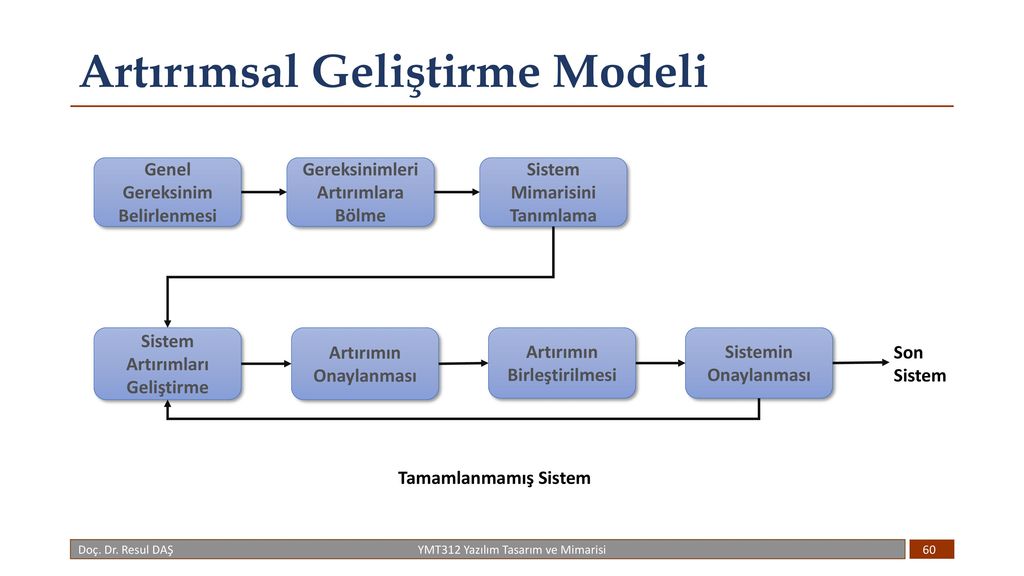
\*HELEZONİK (SPİRAL) MODEL

Helezonik modelde süreci oluşturan aşamalardan tekrar tekrar geçilerek ilerleme kat edilmesini amaçlanmaktadır.İçten en dışa doğru gidilerek tekrarlanarak gitmesinden dolayı helezonik(spiral )bir şekil alır.Helezonik modelde risk analizi modelin temelini oluşturur.Modelde proje incelenir daha sonra risk analizi yapılır ve prototip oluşturulur.Prototipler maket uygulamalardır.Prototipler müşteriye gösterilir müşterin değerlendirmesinden sonra onay ve kabul aşamasına geçilip bir daha yazılım aşamasından geçilir ve işlemler tekrarlanır.



Tekrarlanma yani prototip sayısı işin büyüklüğü ile doğru orantılıdır.İş büyük olunca prototip sayısı da haliyle artar.Bu modelde risk analizi ön planda olması hataları erken giderme ve alternatifleri değerlendirme imkanı sunar.Kullanıcıların sistemi erken görmesi de bir avantajdır.Kullanıcının isteğini en son aşamada dile getirip işleri zorlaştırmak yerine aşama aşama görmesi bize oldukça büyük bir avantaj sağlar.Helezonik model kompleks bir modeldir.Küçük projeler için önerilmez daha çok büyük projelerde kullanılmasını destekler.Kontrot tabanlı yazılımlar adım adım anlaşma esnekliği sağlamayacağından dolayı bu model kontrot tabanlı yazılımlar için uygun değildir.

\*ARTIMSAL GELİŞTİRME SÜREÇ MODELİ

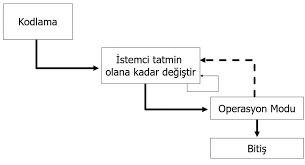
Artımsal geliştirme süreç modelinde kullanım ve yapım aynı anda gerçekleşir. Büyük proje küçük parçalara bölünür .Bu modelde bir döngü söz konusudur.Planlama ile başlar ve daha sonra iterasyonlardan oluşan bir döngüye girer.Her döngüde parçanın biri çözülür. 

\*Bu şekilde de görüldüğü gibi artımsal geliştirme de planlama sürecinden sonra ayrılan parçaya göre bir döngüye girmiştir.Eğer planlama kısmında bir

değişiklik söz konusu olursa o kısma da dönülmek zorunludur. Parçalar projedeki gereksinim ve önem sırasına göre oluşturulur.Önemli olan kısımlara öncelik verilerek projenin tamamının çözümlenmesi amaçlanır.Çözülen her parça bir önceki parçayı kapsayacak ve projeyi büyütecek şekilde olmalıdır.Büyük ve kompleks projelerde bu modelin kullanımı uygundur.Her parçanın bitiminde müşteriye bir görünüm verildiğinden dolayı sistemin işlevsellik daha net görülür.Parçaların gösterimi tüm projenin batma olasılığını azaltır.

\*KODLA VE DÜZELT YAŞAM-DÖNGÜ MODELİ

Kodla ve düzelt modelinde direkt kodlama vardır. Verilen iş direkt olarak kodlandıktan sonra müşteri gereksinim ve isteklerine göre kodlama düzeltilir.Bu süreç yazılımın emekliliğine kadar devam edebilir.Projenin bakımı oldukça zordur çünkü direkt kodlama olduğundan dolayı elimizde prgeliştirme modeliojenin herhangi bir dökümantasyonu yok.Bu model en kolay yazılım geliştirme modelidir fakat aynı zamanda kodlama aşamasından sonra çıkan değişiklikler dolayısıyla en maliyetli modeldir.Yazılımda herhangi bir planlama,test,dökümantasyon vs. yapılmadığından dolayı çok hızlı bitmektedir .Küçük 100-200 satırlık kodlar için uygundur fakat büyük,kompleks ve dinamik projeler için elverişli olmayan bir yöntemdir.



\*MODELLERİN KARŞILAŞTIRILMASI - MODELLERİN HANGİ PROJELERDE KULLANILABİLECEĞİ VE SDLC MODELLERİ ÖZET

Barok modeli,gelişigüzel model,kodla düzelt modeli,çağlayan model 🡪 Bu modeller daha çok küçük ,kompleks olmayan,statik projelerde(Sürekli değişikliğe gitmeyecek bir köprü yapımı işinde bu model kullanılabilir bu modelin geçmişte popüler olmasının en temel nedeni de aslında budur geçmişte inşşat alanında ihtiyaçlar varken gelişen günümüzde daha kompleks değişken projeler olduğundan dolayı bu modeller geçmişte kalmıştır.Geçmişte kalsalar da günümüzdeki çoğu modelin temelini atmışlardır.) kullanılırken;Artımsal geliştirme modeli,V süreç modeli,helezonik model🡪Bu modeller de daha büyük ,kompleks ,statik, dinamik projeler için elverişlidir. Barok modelde dökümantasyon en son çıkartılırken çağlayan modelde dökümantasyon sürecin bir parçasıdır.V süreç modeli çağlayan modelinden farklı olarak modülleri test eder .Helezonik model diğer modellerden farklı olarak risk analizi yapar.Artımsal model diğer projelerden farklı olarak projeyi bir bütün olarak almaz onun yerine önem sırasına göre parçalara ayırıp önem sırasıyla işleri hem yapıp hem de kullanır.Helezonik,artımsal,V süreç modellerinde müşteri de sürece biraz da olsa dahilken; barok,gelişigüzel,kodla düzelt,çağlayan modellerde müşteri iş sonunda proje ile tanışır bu da müşteri memnuniyetsizliği, proje bitimi istekler gibi olumsuz etkiler yaratır. Yüksek rekabet koşulları, hızlı değişen teknolojiler, müşterinin de nihai gereksinimi henüz proje başlarken detaylarıyla görememesi, ürün veya hizmetle ilgili algının işin başlamasından sonra karşılıklı oluşması ve gelişmesi eski tip modellerin kullanımını bitirmiştir onun yerine çevik manifestosuna geçiş olmuştur.

ÇEVİK(AGILE) YAZILIM= Çevik yazılımın amacı yazılımın daha esnek ve güçlü olmasını sağlamaktır.Bu metodolojide projenin boyutuna bakılmaksızın proje küçük iterasyonlara ayrılır.Her iterasyonun 2-4 hafta sürmesi beklenir ve her iterasyon bitiminde müşteriye bilgi verilir. Çevik yönetimi felsefesi sürekli gelişim, iyileşim ve yalınlık üzerine kurulmuştur.2001 yılında [“çevik manifestosu”](http://agilemanifesto.org/iso/tr/manifesto.html) yayınlanmıştır.Çevik manifestosu ; İlgili tarafla iletişimi prosedür ve araçlardan daha önemli tutmaktadır,müşteri ile işbirliğinin önemi,işi dondurmaktansa değişikliğe açık olmak,çalışan bir ürün vermeye odaklanmak.Çevik manifestosu sayesinde ürünler daha kısa sürede bitmektedir.Projenin parçalara ayrılması ve müşterinin parçaları görmesi projenin daha az hata ile oluşmasını sağlamaktadır.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.

\*XP(EXTREME PROGRAMMİNG)= XP 4 temel husus üzerine kuruludur.İLETİŞİM,BASİTLİK,GERİ DÖNÜŞ,CESARET. İLETİŞİM🡪Yazılım da yanlış ve eksik iletişim en önemli sorunlardan biridir.XP bu iletişim eksikliğini gidermeyi hedefler.Etkin ve doğru iletişim projeyi daha kısa sürede bitirmeyi ,müşterinin söylediklerini, müşterinin mühendise söyledikleri net anlaşılmasını böylelikle de iletişim hatasından dolayı projede hatalar oluşmasını engeller.XP’de iletişim yüzyüze olmalı,müşteri ile iletişim çok sıkı ve doğru olmalı ,müşteri projenin bir parçası olarak görülmeli iletişim ona göre kurulmalı,istekler hemen alınıp kısa sürede projeye dönüş olmalı .BASİTLİK🡪XP de problem çözümü o an da çözülebilecek en basit çözüm olmalı.Problemin çözümü için “İşe yarayacak en basit şey nedir?”sorusunu sordurmayı hedefler XP.GERİ DÖNÜŞ🡪Geri dönüş bir projedeki en önemli hususlardan birisidir.Projedeki hataların görülmesini sağlar.Müşterinin bize geri dönüşü projenin onun isteği doğrultusunda şekillendirecektir.Ekipler projedeki önceki çalışmaları konusunda birbirine geri dönüş yapmaları projenin ilerlemesi açısından çok büyük katkı sağlar.CESARET🡪Bir proje yaptıkça batıyorsa ve çalışmalar sonucu elle tutulur bir sonuç yoksa o projeyi bir kenara atacak cesarete sahip olmalıyız.Ya da ekip arkadaşlarımıza bunu dile getirmekte cesaretimizi ortaya koymalıyız.Başarısızlıktan korkmak yerine başarısızlığı getiren etkenleri bulup onları çözmek cesaret sayesinde olur. XP hedeflediği esneklik ve basitliği sağlamak için 12 tane pratik sunar.Planlama oyunu,ekipte müşteri,önce test,basit tasarım,çiftli programlama,sürekli entegrasyon,kısa aralıklı sürümler,yeniden yapılandırma,ortak kod sahiplenme ,metafor,kodlama standartı,haftada 40 saat.

\*SCRUM= Çevik yazılım geliştirme metodolojilerinden biri olan scrum büyük projeleri sprint adı verilen küçük parçalara bölerek projeyi geliştirmeyi hedefler.Bu yazılım geliştirme metodolojisinde bir sprintin bitme tarihi en fazla 30 iş günüdür.Scrumun özü küçük takımlardır.Bu takımlar esnek ve yüksek adaptasyon yeteneğine sahiptir.Bu sayede kompleks projeler ürünlenebilmektedir.Scrum iterasyon ve artımlı bilgi aktarımı ile gelişim göstermedir.Scrumda deneysel sürecin kontrolü de oldukça önemli bir yer kaplar.Üç yöntemle bu kontrol yapılır.Şeffaflık,gözlem,adaptasyon.Şeffaflık;ürünü çıkarmada sorumlu kişiler diğerlerinin ne yaptığını anlamalı.örneğin kişiler aynı dili kullanmalı.Gözlem;yetkin gözlemciler tarafından sprintlerin oluşturduğu scrumu gözetmesi lazım .Hedeften sapma var mı yapılan ve istenen arasında fark artıyor mu bunları arada bir gözlemlemesi gerekmektedir.Adaptasyon;gözlemci istenen ürün ve gidişat arasında bir sorun görürse bunu hemen düzeltmeli,gidişatı ürüne adapte etmeli.

\*Şekilde görüldüğü gibi bilgiler yönetimden gelir,tüm iş (product backlog) sprintlere bölünür daha sonra 1-4 hafta içinde sprintler çözülmeye

çalışılır.Her gün yapılan daily stand up meeting (en fazla 30 dakikalık ve ayakta yapılan toplantı) sprintlerin durumunun kontrolü açısından önem taşır.Her sprintle bitiminde iterasyon olur ta ki sürecin bitimine kadar.Scrum üç temel kavramdan oluşmaktadır.Roller(scrum takımı-scrum yöneticisi-ürün sahibi),toplantılar(sprint planlama-sprint gözden geçirme-günlük sprint toplantıları),araçlar(ürün gereksinim dokümanı-sprint dokümanı-sprint kalan zaman grafiği).Ürün sahibi(product owner);projenin iş dönüşü açısından geri dönüşü ile sorumludur buna müşteri ve iş arasındaki bir köprü de diyebiliriz onlara sunar geri dönşleri alır ve takıma iletir.Scrum yöneticisi(master);takımın scrum teorisini ,pratiklerini,kurallarını ve temel değerlerini anlamasına yardımcı olur.Scrum master takımın scrum uygulamalarını planlamak,üretkenliğini arttırmak için yapılacak değişikleri yaptırır.Scrum takımı; devamlı iletişim halinde olan ve projenin bitimi için uğraşan 7-9 kişilik insanlardan oluşur.Sprint planlama;geniş bir gereksinim listesi çıkarımı yapılır,riskler gözetilir oluşabilecek problemler,çıkacak maliyet hesaplanır.Sprint gözden geçirme ; her sprint başlangıcından önce yapılan genel planlama konuşmasıdır. Toplantının ilk kısmında Ürün Sahibi ile Takım, Ürün Gereksinim Listesi’ni gözden geçirir ve gereksinim elemanlarının hedeflerini ve içeriklerini belirler.Toplantının ikinci kısmında, takım üyeleri Ürün Gereksinim Listesi’nin en üstünden başlayarak koşu sonucunda gerçekleştirilecek gereksinimlerden oluşan Koşu Gereksinim Listesi’ni oluştururlar.Daily sprint meetings;günlük olarak genelde sabahları yapılan ayak üstü 15-30 dakikalık toplantılardır.Yapılacak işin gidişatı konuşulur dün ne yaptın, bugün ne yapacaksın, seni engelleyen sorunlar nelerdi? Bu üç temel soru sorulur ve toplantı biter.Takımın ilerleyişini görmek için mükkemmel bir yöntemdir.Takımın üstündeki olumsuz etkileri farkedip onları çözmeye de yardımcıdır.Ürün gereksinim dokümanı ;proje boyunca yapılması gereken iş elemanlarının listesidir.Statik değil dinamiktir.Sürekli olarak eklemeler,çıkartımlar ve bölümler olur.Genellikle kullanıcı hikayelerinden oluşur ve kullanıcının bakış açısından oluşur.Sprint dökümanı; Sprint dokümanı, Geliştirme Takımının Sprint Hedefine ulaşmak için gerekli gördüğü tüm işleri görünür kılar. Sprint dokümanı, sürekli iyileştirmeyi sağlamak için, bir önceki Retrospektif toplantısında belirlenen en az bir yüksek öncelikli süreç iyileştirme maddesini içerir.Sprint dokümanı, Günlük Scrumda ilerlemenin anlaşılabilmesi için yeterli ayrıntıyı içeren bir plandır.Geliştirme Takımı, Sprint boyunca Sprint dokümanı değiştirir; Geliştirme Takımı Sprint içerisinde plana uygun çalıştıkça ve Sprint Hedefine ulaşmak için gerekli olan işi daha fazla anladıkça Sprint dokümanı belirginlik kazanır.Sprint kalan zaman grafiği ;bu grafik sprint boyyunca işlerinin ne kadarının yapıldığını ve normalde ne kadar yapılması gerektiğini göstermesi açısından önem taşır.

\*SCRUM GÜNÜMÜZDE NEDEN POPÜLER?

Scrum projeleri yukarıda anlatıldığı gibi şeffaflık,gözetim,sürekli iletişim,basitlik ,geri dönüş vs. gibi temeller üzerine kuruludur.Bu da scrum ile yapılan projelerin daha net ve açık olmasını sağlar.Projenin daha açık ve net olması hataların hızlı bir şekilde çözümlenmesini sağlar.Günümüzde eskideki gibi durağan ,küçük projeler yerine kompleks ve uzun projeler vardır.Bu gibi projeler scrum mantığında küçük parçalara bölünüp bir takım insanın bu parçaları çözümleme çalışması , scrumun iletişime verdiği önemden sayesinde yüksek iletişim bağı ile bu takımın kendi içinde ve müşteriyle arasında iletişim hatasını en aza indirgemesi,daily sprint toplantılar sayesinde ürünün gidişatını engelleyen konuları görmek ve buna çözüm bulmak,sprint çalışmaların geri dönüşe izin vermesinden dolayı müşterinin değişken istekleri gözetilir ve değişen istekler yerine getirilir.Ekip çalışması olduğundan dolayı insanların %80 ‘nın işe verimliliği artar bir an önce küçük parçaları bitirmeyi hedefler ve böylelikle büyük parça kendiliğinden ortaya çıkar .

KAYNAKÇA

<https://www.researchgate.net/profile/Fatih-Yucalar-2/publication/265790432_Tailoring_and_Implementation_of_an_Agile_Software_Development_Process/links/541be6cf0cf203f155b33b24/Tailoring-and-Implementation-of-an-Agile-Software-Development-Process.pdf>

<https://slideplayer.biz.tr/slide/12386328/>

<https://docplayer.biz.tr/5816900-Yazilim-gelistirme-modelleri-ve-sistem-yazilim-yasam-dongusu.html>

<https://www.agilealliance.org/glossary/xp/#q=~(infinite~false~filters~(postType~(~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'xp))~searchTerm~'~sort~false~sortDirection~'asc~page~1)>

<https://en.wikipedia.org/wiki/Extreme_programming>

Doç. Dr. Deniz Kılınç, Bakırçay Üniversitesi Yazılım Mühendisliğine Giriş Dersi 2- 3 ve 4. Hafta Sunumları