BİL 102 YAZILIM MÜHENDİSLİĞİ TEMELLERİ
DİLA KEMER
220601055

YAZILIM YAŞAM DÖNGÜSÜ(SDLC)

ÖZET

Yazılım yaşam döngüsü; geliştirdiğimiz yazılım projesinin planlanmasından, teslimat sürecine kadar başından geçen tüm süreci kapsar. Yazılım yaşam döngüsünün 5 temel adımı vardır: planlama, analiz, tasarım, gerçekleştirme/test, bakım.

Yazılım yaşam döngüsünün yanında, bu sürecin işleyişini ve nasıl ilerleyeceğini gösteren yazılım yaşam döngüsü modelleri bulunmaktadır. Yazılım yaşam döngü modelleri, ürünlerin belli standartta olmasını sağlar, kargaşayı azaltır, proje içinde düzeni sağlar. Bu modeller şunlardır: gelişigüzel model, barok model, şelale(waterfall)model, V süreç modeli, helezonik(spiral)model, artırımsal geliştirme, kodla ve düzelt, evrimsel model. Bu modellerden ayrı olarak günümüzde oldukça popüler olan XP(extreme programming) ve SCRUM'ı kapsayan çevik yazılım geliştirme(agile programming) metodu da bulunmaktadır.

GİRİŞ

Bir yazılım ürününün yapılış aşamalarının tümünü kapsayan sürece yazılım yaşam döngüsü denir. Yazılım yaşam döngüsünün temel adımları:

1.GEREKSİNİM/PLANLAMA: Yazılım yaşam döngüsünün ilk adımıdır. Temel ihtiyaçların(gereksinimlerin)belirlendiği ve hangi soruna çözüm arıyoruz? sorusunun cevabının arandığı kısımdır.

2.ANALİZ: Yazılım yaşam döngüsünün en önemli adımlarından biri olan bu aşamada yazılım ürününün işlevleri detaylı ve titiz bir çalışmayla incelenir. Ayrıca yazılım ürününün ne kadar süreceği, hangi noktalar üzerinde durulması gerektiği gibi sorun teşkil edebilecek aşamalar göz önüne alınır. Yazılım proje yönetim planı da bu safhada yazılır. Kısaca "Projenin ne yapmasını istiyoruz?" sorusu sorulur.

3.TASARIM: Analiz safhası sonucu ihtiyaçlara cevap verecek yazılım sistemi oluşturulmadan önce proje bileşenlerine ayrılır ve projede yapılacak işlemlere aşama aşama karar verilir. Yazılım geliştiricinin ihtiyaç duyacağı ve proje aşamasında proje ekibine eklenecek kişilerin de anlayabilmesi için önemli bir adımdır.

4.ÜRETİM/GERÇEKLEŞTİRME/TEST: Gereksinim, analiz ve tasarım adımlarını izleyen adımdır. Kodlama, test ve kurulum çalışmaları yapılır. Önceki adımlarda planlanan aşamalar izlenmeye

devam edilmelidir. Test sırasında hata varsa bu hatalar düzeltildikten sonra proje yayınlanmalıdır.

5.BAKIM: Teslimden sonra hataların giderildiği ya da değişen şartlara göre projeye yeni eklemeler(iyileştirme) yapıldığı kısımdır.

YAZILIM YAŞAM DÖNGÜSÜ MODELLERİ

1.GELİŞİGÜZEL MODEL: Herhangi bir yöntemi yoktur. Geliştirici kişinin kendi oluşturduğu öznel, belli bir standardı olmayan yoldur. Bakımı zordur. Bu modeli genellikle acemi yazılımcılar ve öğrenciler projelerinde bu modeli tercih eder.

2.BAROK MODEL: Yazılım yaşam döngüsü adımlarını doğrusal bir yol izlenerek, kompleks olmayacak şekilde gerçekleştirilmesidir. Eski bir modeldir ve aşamalar arası geri dönüş hakkında belirsizlik mevcuttur. Geliştirme ve test aşamasından sonra yapılan dokümantasyon, ayrı bir işlem olarak ele alınır. Üretim kısmına daha çok öncelik verilir. Günümüzde kullanımı yaygın bir model değildir.

3.ÇAĞLAYAN(WATERFALL-ŞELALE)MODELİ: Yazılım yaşam döngüsü temel adımlarının baştan sona en az bir defa üstünden geçilerek gerçekleştirildiği modeldir. Kullanımı ve anlaşılması kolaydır. Barok modelden farkı; dokümantasyon ayrı bir adım olarak değil, doğal bir süreç olarak ele alınmasıdır. Barok modelden ayrılan bir diğer yönü ise adımlar arası geri dönüşlerin nasıl yapılacağının belli olmasıdır. Geri dönmemek için her adımı doğru bir şekilde yapmak gerekir. Bu modelin dezavantajları:

-Durağan olup değişimlere elverişsizdir.

-Karışık projelerde geri dönüşler zordur.

-Yazılım ürününün müşteriye sunulma süresi uzundur.

-Müşteri yazılım geliştirme sürecinde olmadığı için geri dönüşler(feedback)sonradan alınır ve sonradan çıkan sorunları ve ihtiyaçları karşılamak zorlaşır, zaman ve para kaybına yol açar.

Bu model gereksinimleri iyi tanımlanmış, karışık olmayan projeler için uygundur. Kullanımı gittikçe azalmaktadır. Diğer modellere temel teşkil etmesi bakımından önemlidir. Bu modelin avantajları:

-Müşteri ve kullanıcıların anlayabileceği adımlardan oluşur.

-Değişiklik süreci, kolay yönetilebilmesi için birimlere ayrılmıştır.

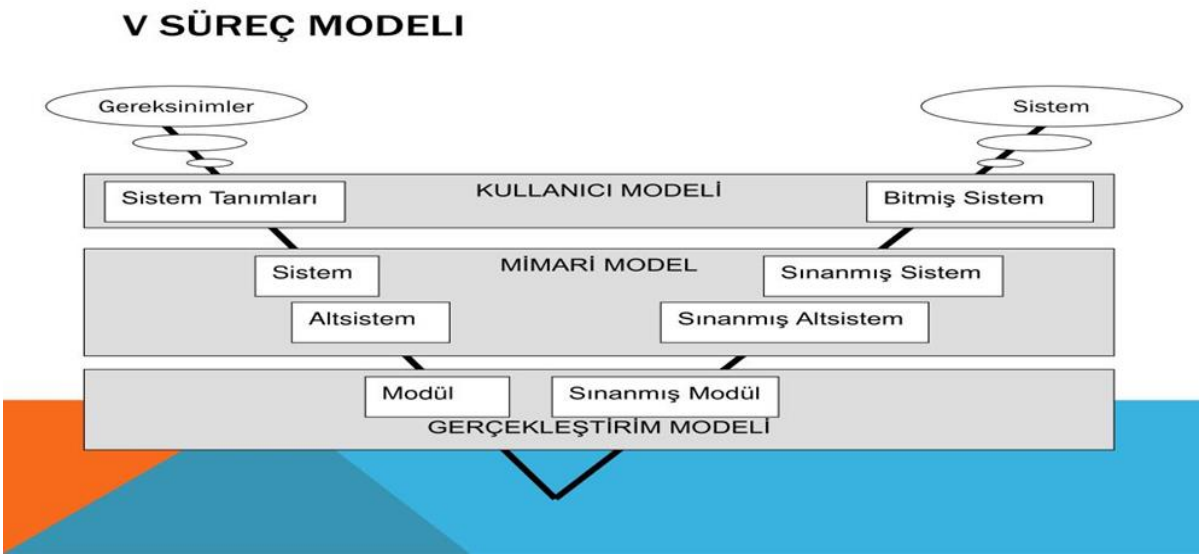
-Proje yöneticilerinin kolay görev dağılımı yapmasını sağlar.

4.V SÜREÇ MODELİ: Çağlayan modeline verification(doğrulama) ve validation(onaylama) aşamalarının eklenmiş halidir. V süreç modelinin 3 temel çıktısı bulunur:

a)Kullanıcı Modeli: Geliştirme süreci ile kullanıcının ilişkilerinin tanımlandığı ve sistemin nasıl kabul edileceğine dair sınama ve planları içerir.

b)Mimari Model: Sistem tasarımı ile tüm sistemin sınama sürecine dair işlevleri kapsar.

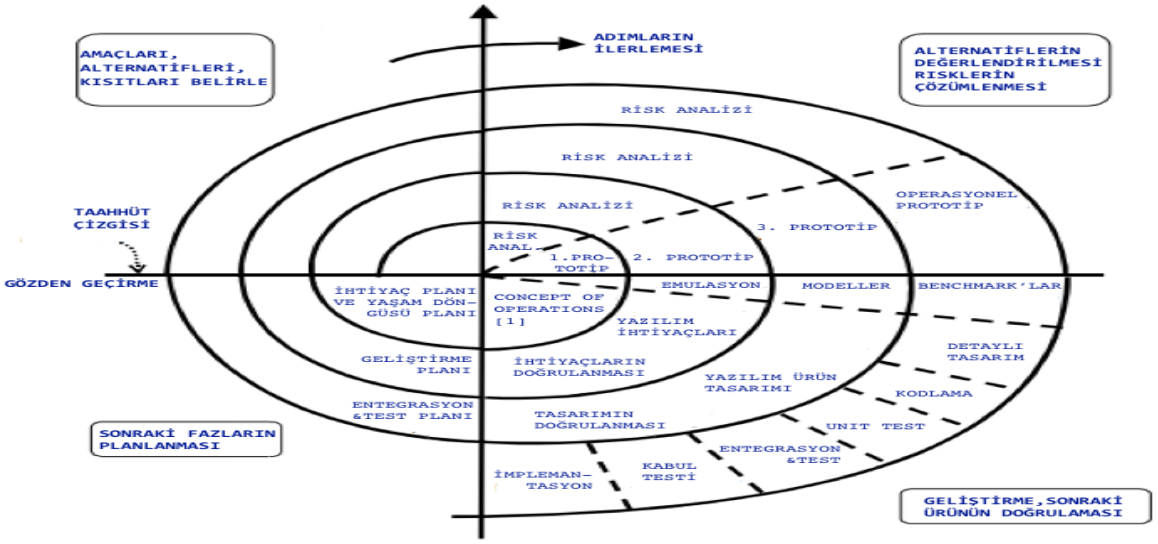
c)Gerçekleştirim Modeli: Yazılım modüllerinin kodlanması ve sınanması sürecini kapsar.



Sol taraf üretim, sağ taraf sınama işlemlerini temsil eder. V süreç modelinin kullanımı kolaydır, anlaşılması ve proje takibi basittir ama aşamalar arası tekrar yoktur. Genellikle CRM(müşteri ilişkileri yönetimi) projelerinde kullanılır.

5.EVRİMSEL MODEL: Belirlenen gereksinimleri anlamayı kolaylaştıran bu modelde, risk ve hata oranı azdır. Ancak sürecin görünürlüğü az ve ürünün bakımı zordur. Modelin başarısı ilk evrimin başarısına bağlıdır. Genellikle çok birimli organizasyonlar(bankacılık uygulamaları)için önerilir.

6.SPİRAL(HELEZONİK) MODEL: 4 adımı içerir. Bunlar: planlama, risk analizi, kullanıcı değerlendirme ve üretim adımlarıdır. Bu modelde risk analizi ön plandadır ve aşamalar spiral oluşturacak şekilde dönerek tekrar eder. Her döngü bir fazı ifade eder. Geliştirme aşaması iterasyonlara ayrılır ve en çok risk içeren bölümler önce gerçekleştirilir. Genellikle askeri savunma yazılım sistemleri için kullanılır.



7.ARTIRIMSAL GELİŞTİRME MODELİ: Sistem iki parçaya bölünür: Geliştirme ve teslim. Kullanıcı gereksinimlerinin ön planda tutulduğu bu modelde gereksinimler erken teslime dahil edilir. Her teslimle beraber sistem erken aşamada işlevsellik kazanır. Sistem daha fazla test edilme imkanı bulmuş olur. Böl ve yönet (Divide and Conquer) yaklaşımına uygun bir modeldir. Öğrenci bitirme projelerinde kullanımı tercih edilir.

8.KODLA VE DÜZELT: Dokümantasyonun bulunmadığı bu modelde, yazılım ürünü direkt olarak gerçekleştirilir. Ürün istenilen hale gelene dek yani müşteri gereksinimlerini karşılayana kadar düzenlenir. Bakım safhası oldukça zordur ve emeklilik(retirement) vardır. Yazılım geliştirmenin en pahalı ve en kolay yoludur.

ÇEVİK YAZILIM GELİŞTİRME

Yazılım yaşam döngü modelleri günümüzde güncelliğini koruyamadığı için ve yazılımdaki başarı oranı düşük olduğu için yeni yöntemler geliştirilmeye başlandı. Bu yöntemler çevik yazılım geliştirme adı altında toplandı. Çevik yazılım geliştirme amaçları:

-Olabildiğince kısa zamanda ürün çıkarıp müşteriye sunma

-Değişen gereksinimlere hızlı dönütler

-Verimi yüksek, hata oranı düşük, hızlı ve az maliyetli çözümler içerir. Günümüzde kullanımı oldukça yaygın olan çevik yazılım geliştirme ile değişen gereksinimlere kolayca adapte olmak mümkündür.

Projenin küçük parçalara(iterasyon) ayrılarak planlama ve yürütmenin gerçekleştiği bu modelde yöntemler gözden geçirilip süreç iyileştirme yapılır. Her süreçte olduğu gibi çevik yazılımda da dezavantajlar vardır. Sürekli değişen gereksinimler çalışma şartlarını zorlaştırır. Ekip üzerinde hedefe ulaşma baskısı görülür.

YAYGIN KULLANILAN ÇEVİK YAZILIM GELİŞTİRME METODOLOJİLERİ

-XP(EXTREME PROGRAMMİNG): En popüler süreçlerden biri olan XP, yazılım kalitesini bir üst seviyeye çıkarmak ve değişen gereksinimlere daha hızlı cevap vermeyi hedefleyen bir metottur. XP'yi besleyen 4 temel ilke vardır: İletişim, basitlik, geri bildirim, cesaret.

a)**İletişim:** Analist, yazılımcı, proje yöneticisi, testçi, kullanıcı, müşteriden oluşan bir ekip oluşturulur. XP, ekip içindekilerin birbirleriyle olan iletişimine son derece önem verir. Kişiler arası iletişim yüz yüze olmalıdır.

b)**Basitlik:** Tasarım ve kodlama kısımlarında basitlik ilkesi göz önünde bulundurulmalıdır. Ancak basitliği sağlamak zordur; çünkü basitleştirmek isterken ayrıntı gibi görünen önemli hususları atlamak, başka problemlere sebebiyet verebilir.

c)**Geri Bildirim:** Yazılım ekibi ve müşterinin belli zamanlarda toplanarak durum değerlendirmesi yapması sırasında projenin gidişatı hakkında konuşulur. Projenin hedeflenen yere gelmesi için oldukça önemli bir ilkedir.

d)**Cesaret:** Zaman zaman çalışılan projedeki aksaklıklar ve zorluklar nedeniyle ekip üyelerinin umutsuzluğa kapılmaması ve başarısızlıktan korkmadan, yeri geldiğinde baştan başlayabilmelilerdir.

XP PRATİKLERİ:

1-PLANLAMA OYUNU(PLANNİNG GAME): Küçük kağıtlara yazılmış öneriler olarak tanımlanır. Planlama aşamasının gelişim sürecini gösterir. Projede nasıl bir yol izlenecek, ne kadar bütçe gerekiyor? gibi soruları yanıtlamaya çalışır.

2-METAFOR: Genellikle iş dünyasında teknik altyapısı olmayan insanların anlayabilmesi için basitleştirilmiş sistem mimarisidir.

3-BASİT TASARIM: XP'de projeye başlanmadan önce güvenli yazılım ve hata tespiti için test yazılımı oluşturulur.

4-EKİPTE MÜŞTERİ: Projenin gerçekleştirme aşamasında yazılım ekibinin yanında müşteri temsilcisi bulunur. Böylece herhangi bir geri dönüte ihtiyaç duyulduğunda müşteri ile iletişime geçmek kolaylaşır.

5-ÇİFTLİ PROGRAMLAMA(PAİR PROGRAMMING): Bu yöntemde iki kişi tek bilgisayar üzerinde çalışır, yazılımcılardan biri "sürücü"(driver) diğeri "gözlemci"(observer) rolünü üstlenir. Sürücü kod yazar, gözlemci ise yazılımı yazan kişinin hata yapması durumunda hemen müdahale edebilir. Bu yöntem, ekibe yeni gelen yazılımcının ekibe ve ekibin çalışma kültürüne adapte olabilmesi açısından önemlidir.

6-SÜREKLİ ENTEGRASYON(CONTINUOUS INTEGRATION-CI): XP modelinde iterasyonlara ayrılmış kodun çalışıp çalışmadığının kontrol edildiği pratiktir. Proje ekibindekilerin hataları daha çabuk telafi etmelerini sağlar.

7-KISA ARALIKLI SÜRÜMLER: 2 ya da 6 aylık sürümler çıkar. En çok kullanılacak olan yerler ilk başa konur, risk azaltılır.

8-ÖNCE TEST: Projeden önce güvenli yazılım ve hata tespiti için test yazılımı oluşturulur. Bu yöntemde yazılımcı, birim teste sahip olmayan kod üzerinde çalışmayı reddeder. Bu yöntem sayesinde tasarım iyileştirme faaliyetleri daha kolay ve güvenilir bir şekilde gerçekleştirilir.

9-YENİDEN YAPILANDIRMA(REFACTORING): Yazılmış bir koda ekleme veya çıkarma yapılsa dahi kodun çalışma fonksiyonlarını kaybetmemesi gerekir. Bu pratik, kod bloğunda daha kolay değişiklik yapabilmemizi sağlar.

10-ORTAK KOD SAHİPLENME: Tüm çalışmalar sonucu geliştirilen yazılım ürününün üzerinde tüm ekibin emeği vardır ve bir ekip üyesi diğer ekip üyesinin yazdığı kod bloğu üzerinde çalışma hakkına sahiptir.

11-KODLAMA STANDARTI: Geliştiriciler, ortak bir kodlama kültürüne sahip olmalıdır. Örneğin, her bir geliştiricinin yazdığı kodların syntax'ı (sözdizimi) birbirinden farklı olursa kod karmaşıklaşır ve anlaşılması zorlaşır.

12-HAFTADA 40 SAAT: Ekip üyelerinin zihnini temiz tutması yapılacak projenin başarıya ulaşması için oldukça önemlidir, bu yüzden haftada 40 saat çalışma standartı vardır. Fazlası verimsizlik, azı ise projenin zamanında teslim edilememesine yol açabilir.

-SCRUM: Günümüzde oldukça popüler bir metottur. Karmaşık yazılım süreçlerini yönetmeye yarayan bu metodoloji, sadece yazılım değil tüm sektörlerde kullanıma uygundur. Çalışma prensibi, süreci parçalara (sprint) ayıran ve tekrara dayalı bir yol izler. Gereksinimlere kolay adapte olan bir yapısı vardır. Klasik metodolojilerdeki gibi gereksinim analiz, tasarım, gerçekleştirim aşamalarını içerir ancak klasik metodolojilerde aşamalar sırayla biri bitmeden diğerine geçilmeyecek şekilde ilerler. Scrum'ın overlapping(örtüşme)prensibiyle bu aşamalar iç içe ilerler. Her gün kısa toplantılarla proje takibi yapılır. Scrum karmaşık ortamda yazılım geliştiren ekipler tarafından kullanılır. Bu karmaşıklığı engellemek için 3 temel ilke vardır:

1)Şeffaflık: İşleyiş sürecini görünür hale getirerek aksaklıkları ortaya çıkarır. Böylelikle proje ekibinin hatalarını erken fark ederek sürekli olarak iyileştirme yapmalarına olanak sağlar.

2)Denetleme: Projenin gidişatının düzenli aralıklarla denetlenmesidir.

3)Uygunluk: Proje değişen gereksinimlere ayak uydurabilmelidir.

SCRUM'DA 3 TEMEL KAVRAM

1-Roller(Roles)

a)Ürün Sahibi(Product Owner): Ekip ve müşteri arasında aracılık eder. Projenin geri dönütlerini vermekle yükümlüdür.

b)Scrum Yöneticisi(Scrum Master): Ekibi organize etmekle sorumludur. Scrum kurallarını iyi bilir ve proje sırasında ekibin kurallara bağlılığını test eder.

c)Scrum Takımı(Scrum Team): Genelde 5-9 kişiden oluşan hep beraber bir hedefi gerçekleştirmek için çalışan ekiptir. Sürekli iletişim halindedirler.

2-Toplantılar(Meetings)

a)Sprint Planlama(Sprint Planning): Öncelikle kapsamlı bir gereksinim listesi hazırlanır, geliştirme için dağıtım gereksinimi planlanır. Dağıtım, geliştirme, planlama bütçeleri gözden geçirilir. Bu toplantıya geliştirme takımı, ürün sahibi ve ekip de katılır.

b)Sprint Gözden Geçirme(Sprint Review): Gereksinimler önem sırasına göre sıralanır ve sprint gereksinim listesi oluşturulur. Scrum ekibi bu gereksinimlerin hangilerinin gerçekleşip gerçekleşemeyeceğini bu uzun toplantı sırasında konuşurlar ve sonraki sprinte hazırlık yapılır.

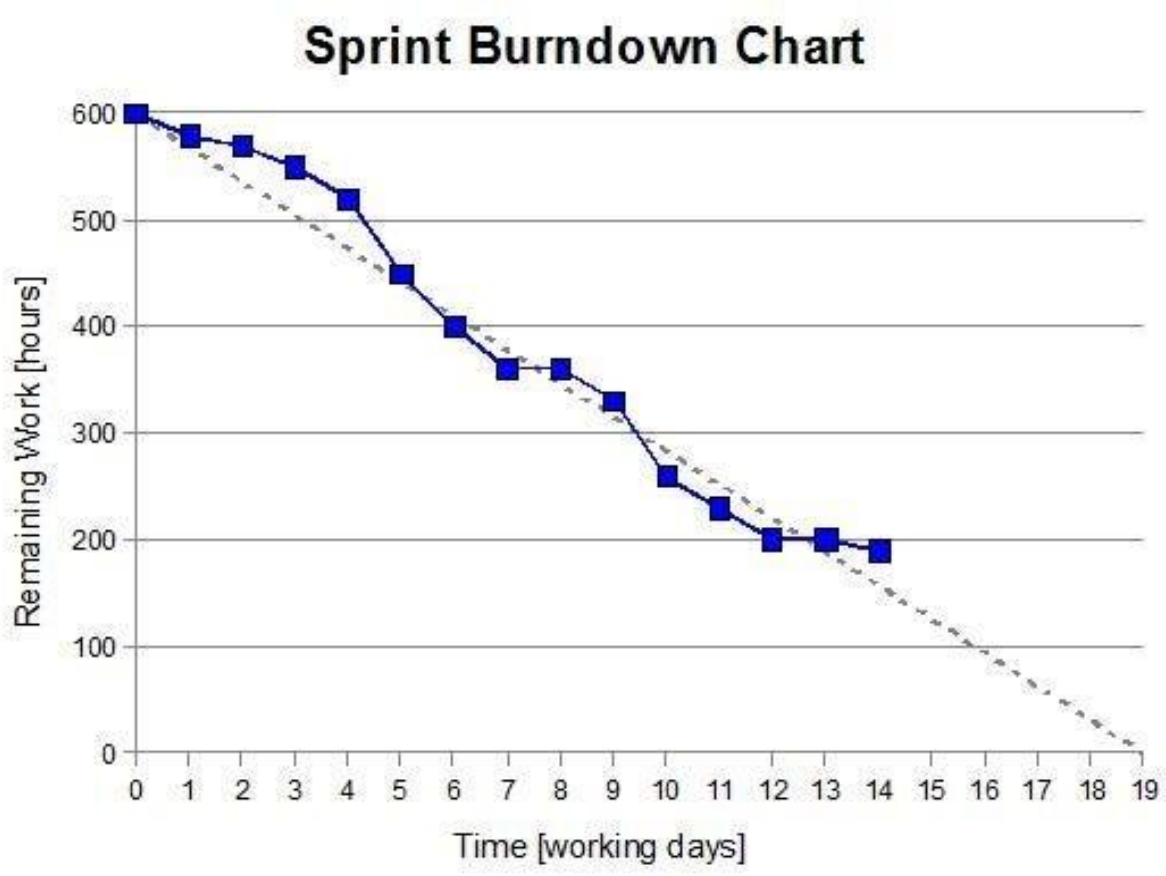
c)Günlük Scrum Toplantısı(Daily Scrum): Yaklaşık 15-20 dakika süren ve her iş gününde belli zaman aralıklarında gerçekleşen bu toplantıya tüm scrum takımı katılır. Önceki gün neler yapıldığını, bugün neler yapılacağının konuşulduğu ayaküstü toplantıdır.

3-Bileşenler/Araçlar(Artifacts)

a)Ürün Gereksinim Dokümanı(Product Backlog): Sprint planlama toplantısında oluşturduğumuz gereksinim listesidir. Ürün sahibi müşteriden gereksinimleri alır ve değişen gereksinimlere göre üretim gereksinim listesini düzenler.

b)Sprint Dokümanı(Sprint Backlog): Ürün dokümanından sprint içerisinde yapılması gereken işler belirlenir. Proje teslim edilmeden önce son kontrol niteliğindedir, ayrıntılar gözden geçirilir.

c)Kalan Zaman Grafiği(Burndown Chart): Sprint boyunca yapılan işin, yapılacak işle kıyaslanmasıdır. Grafikte x ekseni sprint günlerini, y ekseni yapılacak işi gösterir.



Kaynakça

https://akademiksunum.com/index.jsp?modul=document&folder=a93e3a2fccf8eb56a557c55c5f0d5cf10789abe2

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

https://silo.tips/download/9ders-yazlm-gelitirme-modelleri

https://dergipark.org.tr/tr/pub/gazibtd/issue/57610/598346

https://medium.com/@ahmetuyar/extreme-programming-xp-nedir-ddc003a515c4

Hesaplar

https://github.com/Dilakemer

https://www.linkedin.com/in/dila-kemer-956b2425a/

https://medium.com/@dilakemer1