29.03.2021

Buğra TOPRAKCIOGLU

200601081

Yazılım Yaşam Döngüsü ve Yazılım Süreç Modelleri

Buğra Toprakçıoğlu 200601081

**Yazılım Yaşam Döngüsü ve Yazılım Süreç Modelleri**

**Yazılım Yaşam Döngüsü Nedir?**

Yazılım Yaşam Döngüsü, bir yazılım ürününün planlanmasından gerçekleştirilmesine ve bakımına kadar geçirdiği aşamalardır, Yazılım dediğimizde çoğu kişinin aklına kod kısmı gelir, bu kısım sadece bir aşamanın parçasıdır. Yazılım Yaşam Döngüsü sırasıyla 5 aşamadan oluşmaktadır (Bazı kaynaklarda Gerçekleştirme kısmı Geliştirme ve Test olarak ikiye ayrılır) bunlar:

* Planlama
* Analiz
* Tasarım
* Gerçekleştirme
* Bakım

**Planlama:**

İlk olarak gereksinim analizi yapılmalı, Müşteri dinlenir ve müşteri gereksinimleri elde edilir. “Bu ürün nasıl yararlı olabilir?”, “Bu ürünle neyi başarmak istiyoruz?” gibi soruların cevapları aranır ve fikirler yürütülür. Fizibilite (Yapılabilirlik) çalışmaları yapılır ve proje planlaması gerçekleştirilir.

**Analiz:**

Bu aşamada bu ürünün işlevleri bir netlik kazanır, yine müşteri ile sıkı bir bağlantı içinde olarak yazılımın gereksinimleri de ayrıntılı olarak ortaya çıkarılır, bu aşamada proje süresi belirlenir ve risk analizi yapılır. Ayrıca bu aşama UML diyagramının çizilmeye başladığı ilk adımdır.

**Tasarım:**

Yazılım ürün tasarımı müşterinin isteklerini yerine getirebilecek sistemin temel yapı taşlarını oluşturur. Ürünün özellikleri, yetenekleri ve arayüzlerinin belirlendiği aşamadır. İki tür tasarım vardır bunlar:

* **Üst seviye ve mimari tasarım**: önerilen sistem yapısıdır (Modüller, akış şemaları)
* **Detaylı tasarım**: yazılımın bileşenleri ve detayları (veri yapıları, ekran tasarımları)

**Gerçekleştirme:**

Analizi ve Tasarımı bitmiş olan sistemin koda geçiş aşamasıdır. Daha önce Planlama aşamasında belirlenen Programlama dili, geliştirme ortamı ve belirlenen teknolojilerin kullanımına başlanır. Modüller kodlanır birleştirilip test edilir ve en nihayetinde kurulum çalışmaları yapılarak bu aşama sonlandırılır.

**Bakım:**

Bu aşamamızda servis edilen yazılım ürünümüzde bir hata olduğu durumda hataların giderildiği ve gerektiğinde yazılıma yeni gereksinimlerin eklendiği aşamadır. Bu aşama ürünün tüm yaşamı boyunca sürmektedir

Yazılım yaşam döngüsünün daha aktif ve daha iyi sonuç vermesi için yazılım geliştirme süreç modelleri kullanılabilir, şimdi bu yazılım geliştirme süreç modellerine bir göz atalım. Süreç modelleri temel olarak üçe ayrılırlar (Düzenliyici Süreç Modelleri-Birleşik Süreç modeli- Çevik Yazılım Modelleri)

**Düzenleyici Süreç Modelleri:**

**1)Gelişigüzel Model:**

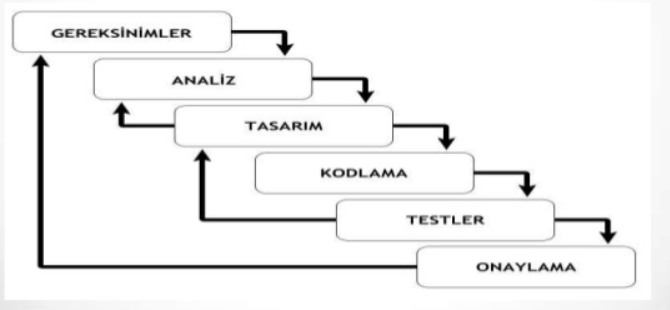
Bu yöntemi her ne kadar model olduğunu söylemek yanlış olsa da ona bir yer verelim, belirlenmiş yöntem ve modeli yoktur. Daha çok tek kişilik basit projelerde kullanılan bu yöntem 60lı yıllarda ortaya çıkmıştır. Bu yöntem tek kişiye bağlı bir geliştirme türü olduğu için ortaya bakımı ve okunurluğu zor bir kod çıkarmaktadır.

Miadı doldurmuş olan bu model göz kararı ölçüsüz bir yemek gibidir sürekli başarı getirmemektedir.

**2)Barok Modeli:**

Yine eski bir model olan Barok modeli 70’li yılların ortasında kullanılmaya başladı. Barok modeli Yazılım yaşam döngüsü adımlarını doğrusal bir şekilde almaktadır. Dokümantasyonu olması bir önceki Gelişigüzel Model’ e göre onu daha iyi bir hale getirse de günümüzdeki modellerden farklı olarak Belgeleme yi ayrı bir süreç olarak ele alınması (Geliştirme ve Test faaliyetleri yapıldıktan sonra öngörülmesi), Geliştirme aşamasına çok ağırlık vermesi ve aşamalar arasında geri dönüşlerin tanımlı olmaması onu günümüz yazılım projelerinde uygulanan bir model olmaktan çıkarmıştır.

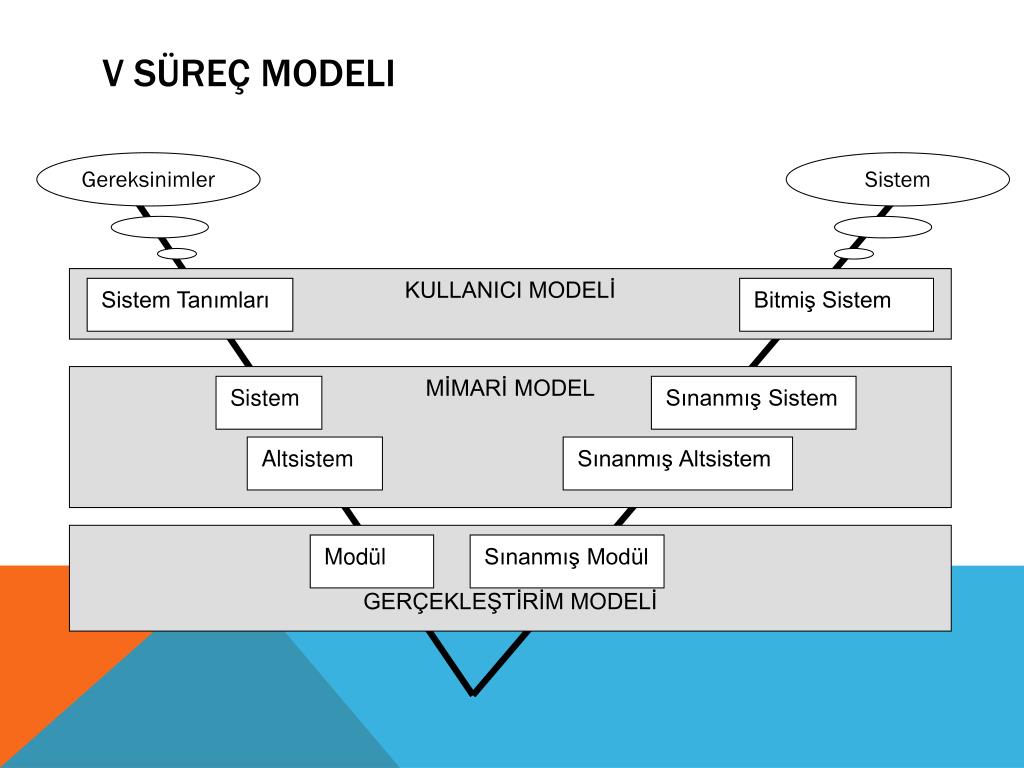
**3)Çağlayan (Şelale) Modeli:**

****

Geleneksel yazılım geliştirme modeli olarak da bilinen Çağlayan modeli geçmişte en popüler model olarak görülmüştür. En tanınmış model olan Çağlayan modelinde yazılım en az bir kez tekrarlanması ile geliştirilmektedir, bu modelden yarar sağlamak için çok iyi tanımlı ve kısa projeler uygundur ve bir önceki Barok Model’ine göre en büyük avantajları dokümantasyon sürecin içinde olması ve aşamalar arasındaki geri dönüşlerin nasıl olacağı da tanımlı olmasıdır. Çağlayan modelinde işler aşama aşama yapılmakta ve bir aşama bitmeden diğer adıma geçilmemektedir. Her aşamada dokümantasyon olmalıdır

, Dokümantasyon veya Test olmadığı takdirde o aşama bitmiş sayılmaz. Analiz ve Tasarım aşamaları Çağlayan Yaşam modeli için çok önemlidir. Analiz kısmında sistem gereksinimleri en ince ayrıntısına kadar elde edilmeli ve Tasarım aşamasında bu gereksinimleri karşılayabilecek bir tasarım gerekmektedir. Maalesef genellikle gerekli gereksinim bilgileri yeterli şekilde elde edilemez ve bu sorun gerçekleştirme aşamasından sonra anlaşılır. Bu tür hataların düzeltilmesi oldukça pahalıya patlar. Dolayısı ile ürünün kullanıcıya ulaşması uzun zaman almaktadır. Tüm bu durumlar Yazılım Üreticilerinden ekip olarak mutsuz olmasına kodlama aşamasına daha çok önem vermelerine sebep olur. Bu kadar sorunun en büyük sebebi kullanıcının süreç içerisinde olmaması desek yanlış olmaz, bu sebeplerden ötürü Çağlayan modelinin kullanımı günümüzde gittikçe azalmaktadır.

**4)V Süreç Modeli:**

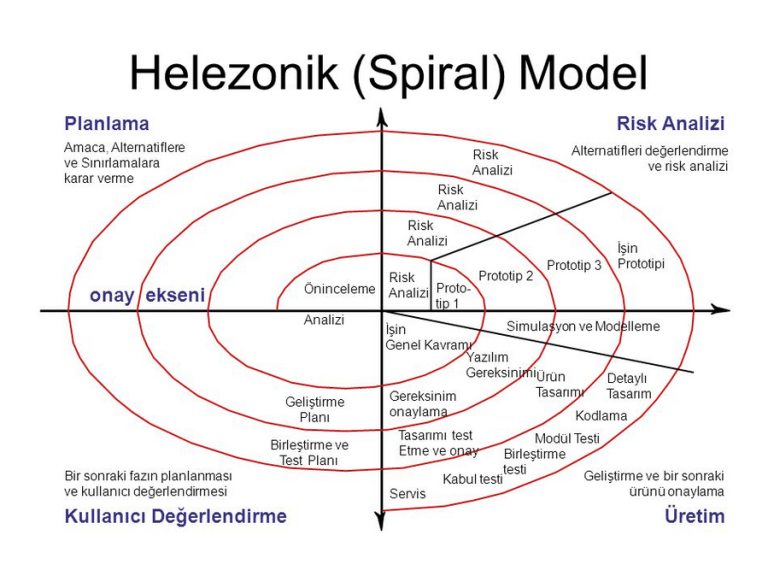


V Süreç modeli için Çağlayan modelinin geliştirilmiş bir versiyonu diyebiliriz. Sol taraf üretim, sağ taraf sınama(test) işlevleri ile ilgilidir.

V süreç modelinin temel çıktıları şunlardır:

* **Kullanıcı Modeli:** Kullanıcının tanımları projenin oluşturulması ve Bitmiş Sistemin teslim edilmesi.
* **Mimari Modeli:** Sistem tasarımı ve oluşacak sistemin sınanma aşamasıdır.
* **Gerçekleştirme Modeli:** Yazılım Modüllerinin kodlanması ve sınanması aşamasıdır.

İş tanımının belirgin olduğu, belirsizliklerin az olduğu projeler için uygun bir modeldir, kullanıcının projeye katkısını arttırmaktadır. V Süreç Modelinin takibi kolay olması, sistemin kullanımı kolay olması vb. avantajı olduğu gibi aşamalarda tekrar olmaması ve risk çözümleme için ayrılan yer olmaması dezavantajdır.

**5)Helezonik (Spiral)Model:**

Helezonik Model de ana konular risk analizi ve prototiplerdir. Her döngü öncesinde içerisinde bulunan bölümün risk analizi yapılır, her döngü sonunda yeniden planlama yapılabilir. Helezonik model de risk analizinin detaylı hesaplanması zamanı ve maliyeti kolay hesaplamamızı sağlar ayrıca her döngü sonunda kullanıcı sistemi görebilir ve böylece erken safhada hatalar çözülebilmektedir. Avantajlarından biri ise kodlama aşamasına erken safha da geçilebilmesidir. Şimdi de biraz dezavantajlarından bahsedelim öncellikle bu model özellikle güvenlik yazılımlarında kullanılır. Eğer küçük çaplı riski az bir proje için bu model tercih edilmemelidir. Gereksiz bir maliyete sebep olabilir, çok fazla dokümantasyon oluşturur ve bir dezavantajı ise kompleks bir yapıya sahip olmasıdır.

**6)Artımsal Geliştirme Süreç Modeli:**

Uzun zaman alan ve eksik işlev ile çalışabilen Artımsal Geliştirme Süreç Modelimizde, ürün parçalara ayrılarak üretilir ve parça parça teslim edilir. Kullanıcı gereksinimlerinin önem sırasına göre erken teslim edilecek parça belirlenir. Geliştirilmeye başlanan parçanın gereksinimleri değiştirilemez. Üretilen her parça giderek artan sayıda işlev içerir yani burada artımsal bir geliştirme söz konusudur. Müşterinin yazılımı erken safhada deneyimlemesi sağlanır, bu yapılan erken teslimler gereksinimleri belirlemede yardımcı olmaktadır. Sistemin fazla sınanması ve Başarısızlık riskinin az olması gibi avantajları bulunur. Tekrar olmaması, Deneyimli personel gerektirmesi gibi dikkate alınması gereken dezavantajları da bulunabilir.

**7)Prototipleme**:

Hızlı Gereksinim, Hızlı Tasarım, Hızlı Kodlamanın önplanda olduğu bu modelde öncelikle geliştiriciler ve müşteriler bir toplantı ile gereksinimleri belirler, daha sonra bu belirlenen gereksinimlere göre bir prototip tasarlanıp müşteriye sunulur, bu örnek prototip üzerinden gerekli değişiklikler var ise yapılır prototipin beğenilip onay alınması durumunda geliştirme aşamasına geçilir. Sonuç olarak karmaşa ve yanlış anlaşılmaları engelleyen, değişime çok açık bir modeldir fakat belgeleme olmaksızın hızlı ve kirli prototipler oluşturulmaktadır.

**8)Evrimsel Geliştirme Modeli:**

Ürün burada Artımsal Geliştirme Sürecinde olduğu gibi aşama aşama üretilir fakat parçalar halinde teslim edilmez, İlk tam ölçekli modeldir modelin başarısı ilk evrimin başarısına bağlıdır. Nasıl bir projede bu modelleme kullanılır sorusunu düşünürsek birçok şehirde şubesi olan bir bankanın uygulaması gayet güzel bir örnek olur. Avantajları ve dezavantajlarını incelersek, gereksinimi anlamayı kolaylaştırır, risk ve hata azdır. Dezavantaj olarak ise bakım zordur ve sürekli gereksinim yenilemek gerekebilir.

**9)Araştırma Tabanlı Model:**

Araştırma Tabanlı model proje ödevi, yazılım yarışma projesi gibi sonucu ve maliyeti belli olmayan, sonucu belli olduktan sonra hiçbir işe yaramayan Yap-At tarzı projelerde kullanılan yazılım süreç modelidir.

**10)Kodla-Düzelt Modeli:**

Genelde öğrenci projelerinde kullanılan herhangi bir yöntem ve dokümasyan içermeyen, direk yazılım ürünü çıkarılmaya çalışılan, İstenilen duruma getirilinceye kadar geliştirilmeye devam edilen projelerde kullanılan modellerdir. Emeklilik safhası bulunur ve en kolay yöntemlerden biridir fakat dokümantasyon olmadığı için proje kodunun bakımı çok zordur. Tecrübesiz yazılım projelerinde de bu yöntem kullanılır.

**Birleşik Süreç Modeli:**

Yazılım süreç modellerinin en iyi özelliklerinden birleştirilip bütünleştirilmiş yazılım süreç modeline Birleşik Süreç Modeli denir.

**Çevik Yazılım Metotları:**

Yazılım Sektöründe bir ürünün katı kurallar ile çok uzun vadede müşteriye sunulması, zaman içerisinde yeni gelen değişim isteklerine sistemin yetersiz kalması gibi sorunlar yaşanıyordu, bu sorunların aşılması için 1990’lı yıllarda “Çevik” (Agile) olarak adlandırılan metotlar geliştirildi. Çevik metotların amacı başta anlattığımız sorunların çözümü üzerinedir yani piyasa çabuk ürün çıkarabilmeye ve değişiklik isteklerine çabuk yanıt vermeyi amaçlamaktadır. Çevik metotları verimliliği üst düzey ve esnek, hata oranının düşük olduğu, hızlı ve ucuz çözümler sağlamaktadırlar. Çevik yazılım geliştirme metotları çeşitli metodolojilere ayrılmaktadırlar.

**Çevik Yazılım Geliştirme:**

Çevik yazılım süreçlerini daha esnek, güçlü ve kısa yapan kavramsal bir yazılım geliştirme metodolojisidir. Bu metodolojide proje küçük yinelemelere ayrılır ve her yinelemeye başka bir proje gözüyle bakılır ve geliştirilir bu durum ayrıca hataların çözümlerinde büyük bir kolaylık sağlar, yinelemelerin her birine 2-4 hafta tamamlama süresi verilir her tamamlanan yinelemede müşteri ile iletişime geçilir bilgi verilir. Her yineleme kendi içinde çalışan bir sistem olduğu için müşteriye sürekli çalışan bir sistem verilir ve bu da müşteri memnuniyetini beraberinde getirir.

**Çevik Yazılım Geliştirme Manifestosu:**

2001 yılında dünyanın önde gelen çevik modellerin temsilcileri ortak bir zeminde buluşabilmek adına bir araya gelerek “Çevik Yazılım Geliştirme Manifestosu” nu duyurdular. Bu manifestoya göre;

* Bireyler ve aralarındaki iletişim, kullanılan araç ve gereçlerden;
* Çalışan yazılım, detaylı belgelerden;
* Müşteri ile iş birliği, sözleşmedeki kesin kurallardan;
* Değişikliklere uyum sağlayabilmek, mevcut planı takip etmekten; daha önemli ve önceliklidir.

Çevik yazılım geliştirme ‘nin avantajlarından bahsedelim, Yazılım projelerinin başarısı %55 ‘e kadar arttığı görülmüştür, bunun yanında üretkenlik artar, moral motivasyon artar, müşteri memnuniyeti kısa sürede sağlanır, yazılım kalitesi artar ve maliyetler düşer.

Dezavantajlarına değinmek gerekir ise kurumsal bir yapıda uygulamak zordur ve bazen hedefler kısa süreli olduğu için çalışanlarda bir baskı yaratabilir.

**En Yaygın Uygulanan Çevik Modeller:**

* Extreme Programming (XP)
* SCRUM
* Agile Unified Process

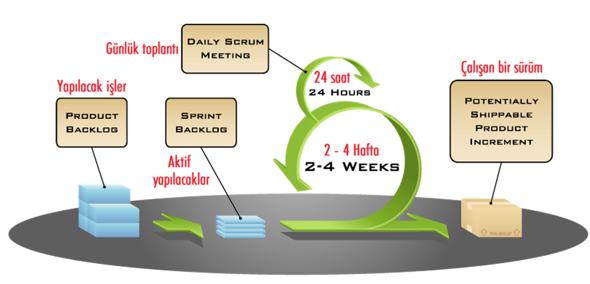
**Uç Programlama (XP):**

XP, Kent Beck tarafından 1999 yılında yazılım geliştirme disiplini olarak ortaya çıkarıldı. Xp ‘nin amacı grup içi iletişime önem vermek ve geri dönüşlerin daha fazla olmasını sağlamaktır. Bu disiplin yazılım geliştirmede kolaylığı ve esnekliği sağlamak için 12 farklı pratiği öngörmektir bunlar; Planlama Oyunu, Ekipte Müşteri, Önce Test, Basit Tasarım, Çiftli Programlama, Sürekli Entegrasyon, Kısa Aralıklı Sürümler, Yeniden Yapılandırma, Ortak Kod Sahiplenme, Metafor, Kodlama Standardı, Haftada 40 Saattir. Ayrıca Uç programlamanın dört temel değeri vardır bunlar;

* **İletişim:** Bir projenin başarılı olması için ekip iyi bir iletişim halinde olmalıdır, çünkü ortaya çıkan birçok problemin iletişim yetersizliği ve sağlıksız iletişimden kaynaklandığı çoğu kez görülmüştür. Xp ’nin bu değeri ise iyi ve sağlıklı iletişimi sağlamak amacındadır. Bu yüzden Xp ’de iletişim yüzyüze olmalıdır. Bu konuya yazılımcı ile yazılımı kullanan kişi arasındaki iletişimde dahil edilebilir. Bu sayede sorunlar daha erken fark edilebilir.Bir bilgi alınması gerekiyorsayüz yüze iletişim sayesinde

Kısa bir süre içerisinde bu bilgi elde edilir ve yazılımın gelişme hızı kesilmeden projenin devamı sağlanır.

* **Basitlik:** Basitlik Zorunlu işlerin yapılmasıdır**.** Karmaşık çözümler XP ‘nin mantığına aykırıdır. XP Basitliği sağlamak günün ihtiyaçlarını hedef alarak esnek ve basit bir sistem gerçekleştirmeye çalışmaktadır. Basitlik isminin aksine sağlanması zor bir değerdir.
* **Geri Bildirim:** 2-4 hafta aralıklar ile sistemin durumu belirlenir ve Yazılım ekibi ile müşterilerin toplanıp sistemin son durumuna bakarlar. Sonradan olabilecek hatalar veya anlaşmazlıklar büyümeden giderilir. Yazılımcılar, sistemin mantıksal yapısını içeren birim testler oluştururlar ve sistem ile ilgili somut bilgiler toplarlar ayrıca müşterilerin, proje çalışanlarının ve yöneticinin görüş ve düşünceleri geri bildirim aşamasında alınır.
* **Cesaret:** XP’nin saydığımız dört noktasından en zoru Cesarettir. Yazılımcı projesinin üzerine gitmeli bunu yaparken yılmamalı ve asla korkmamalıdır gerektiğinde projeyi çöpe atıp tekrar yazabilmelidir. Başarısızlıktan korkmak yazılımcıyı yavaşlatır ve projenin hızı düşer elinizi korkak alıştırmayın.

**SCRUM:**

Scrum Çevik proje yönetimlerinden biridir. Gereksinimlerin kolayca tanımlanamadığı kompleks şüreçlerin yönetilmesi için kullanılır bütünü parçalama ve tekrara dayalı bir yöntem izler. Düzenli bir geri bildirim ve planlamalarla hedefe ulaşır. Müşteri isteklerine göre şekillenen Scrum müşterinin geri bildirimine bağlı bir yapılanma sağlar. İletişim ve takım çalışmanın önemi büyüktür. Scrum un 3 temel prensibi bunlar ;

* **Şeffaflık:** Projenin ilerleyişi ve sorunları gibi gelişmelerden herkes haberdar olmalıdır.
* **Denetleme**: Projenin düzenli olarak kontrol edilme prensibidir
* **Uyarlama**: Proje olası herhangi değişikliğe uyum sağlayabilmelidir

**Scrum Kavramları:**

**1)Product Backlog:**

Proje için gerekli olan gereksinimlerin listesi. Product Owner tarafından müşteriden gereksinimler alınır ve alınan gereksinimler öncelik sırasına göre sıralanır. Product Owner değişen gereksinimlere göre Product Backlog’ a ekleme veya çıkarma yapabilir. Böylece değişim her aşamada kolayca projeye entegre edilebilir hale gelir.

**2)Product Backlog Items:**

Product Backlog içindeki her bir gereksinime verilen isimdir.

**3)Sprint (Koşma):**

Proje Sprint denilen küçük kısımlara ayrılmaktadır burası scrum içindeki aktivitelerin gerçekleştiği yerdir 1 veya 2 haftalık süreçlerden oluşur.

**4)Sprint Backlog:**

Geliştiriciler tarafından Product backlog itemlar öncelik sırasına göre sprint içine alınır ve İşlerin zaman çizelgesi çizilir

**5)Scrum Board:**

Bir sprint içerisinde yapılacak olan maddeler burada yönetilir. Yapılacak olan sorumluluklar “TO DO” bölümüne atılır, bir takım üyesi işe başladığında “IN PROGRESS” bölümüne gönderilir. Bir iş test aşamasına hazır olduğu zaman “TO VERIFY” durumuna getirilir. İş kontrolü yapıldığında ise “DONE” kısmına yollanır.

Scrum toplantılarında bu maddelerin duruma göre yerleri değiştirilir.

**6)Burndown Chart:**

Sprintte kalan işi ve geçen günleri gösteren grafiktir. Scrum’un Şeffaflık ilkesi burada sağlanır.

**Scrum Rolleri:**

**1)Product Owner:**

Müşteriler ve Ekip arasındaki iletişimi sağlar, gereksinimleri toplar projenin özelliklerini tanımlar projenin önceliklerine göre product backlog oluşturur. Sprint’i iptal edebilme yetkisine sahiptir neden bir sprint silinir ki?

Belli başlı cevapları vardır sprint in içindeki işlere gerek kalmamış olabilir veya daha önemli işler çıktığı için sprint silinebilir.

**2)Scrum Master:**

Scrum kuralları, teorileri ve pratiklerine hakim bir kişidir ve bu kuralların uygulanmasından sorumlu kişidir. Takım kaptanı veya yöneticisi değildir görevi takımı rahatsız eden verimli çalışmalarını engelleyen durumları ortadan kaldırmaktır.

**3)Geliştirme Takımı:**

Sprint içine alınan bütün işleri tamamlayabilecek donanıma sahiptirler. Herhangi birinin onlara iş vermesini beklemezler işi kendileri alır ve yaparlar. Sprint backlog hazırlayan kişilerdir tek bir görevleri yoktur 5-7 kişiden oluşan bu grup kendi aralarında işi paylaşırlar. Projenin geliştirme sorumluluğu bu takıma aittir.

**Chicken Roller:**

Scrum yapısında aktif olarak rol almayan kişilerdir (Müşteriler,Satıcılar vs.)

**Scrum Toplantıları:**

**1)Sprint Planning:**

Product Backlog ile belirtilen gereksinimler Geliştirme takımı tarafından görevlere (Task) ayrılır. Takımdaki her eleman taskleri hızına göre kendisine alır. Sprintler belirlenir ve Product Owner ‘a sunulmak üzere yazılım geliştirmeyi hedefleyecek şekilde belirlenir 1- 3 haftalık sprintler oluşturulur.

**2)Daily Scrum:**

Her gün aynı saatte 15 dk lık ayakta yapılan toplantılardır. Bu toplantı gelecek 24 saat üzerine yapılır takımdaki tüm üyeler “Dün ne yaptım?”, ”Bugün ne yapacağım?”, ”İşimi yapmama engel olan sorun var mı?” sorularına cevap verirler. Eğer bir sorun var ise Scrum Master sorunu çözmek için devreye girer.

**3)Sprint Review:**

Her sprint sonunda yapılan toplantıdır yapılan sprint analiz edilir ve ortaya çıkan ürün değerlendirilir. Buradaki amaç ortaya çıkan ürünün Product owner’ın gereksinimlerine uygun olup olmadığını kontrol etmektir. Uygun değil ise farkedilir ve düzeltilir.

**4)Sprint Retrospective:**

Sprint boyunca yapılan işlerin kalitesi, doğruların, yanlışların değerlendirildiği bir toplantıdır. Bu toplantı tüm scrum ekibinin “Neleri daha iyi yapabiliriz?” ve “Nasıl daha iyi yapabiliriz?” sorularına cevap aradığı toplantıdır. Projenin son toplantısıdır.

Scrum günümüzde en çok kullanılan yazılım geliştirme modeli hatta sadece yazılım değil sistem geliştirme modeli bile diyebiliriz bunun nedenleri için böl ve fethet yaklaşımına uygun olması ve değişen gereksinimlere daha hızlı tepki vermesi, Karmaşık görünen projeler için etkili bir çözüm yöntemi olması, Olağanüstü bir iletişim olması bu sayede hataların erkenden farkedilmesi , Gerek maddi gerekse zaman açısından çok kazançlı olması diyebiliriz tabiki bu kadar tercih edilmesinin farklı sebepleri vardır ama şuan en akla yatkın sebepler bunlardır.

Araştırdığım makalelerde ve kaynaklarda çok gördüğüm bir başlık var “Hangi Yazılım Süreç Modeli daha iyi ?” sorusu öncellikle x modeli en iyisi y modeli en iyisi demek bence yanlış bu yazımızda birçok model inceledik ve hepsinin avantajları olduğu kadar dezavantajları vardı. O yüzden hangi projede hangi model kullanılmalı sorusunu cevaplamak daha doğru olur.

**Hangi projede hangi model kullanılmalı?**

* Bir kişinin yazdığı küçük projelerde Kodla ve düzelt modeli tercih edilir.
* Proje küçük ve gereksinimleri çok iyi belirlenmişse biraz eski olan Şelale modeli uygundur.
* Genelde orta ve küçük düzey olan ve uzun sürmeyen projelerde ise çevik modeller uygun olabilir.
* Belirsizliklerin az ve iş tanımlamalarının belirgin olduğu Bilişim teknolojileri projelerinde V süreci modeli uygundur.
* Maliyetli, Büyük ve uzun projelerde spiral veya artımsal model kullanılabilir.
* Birden çok şehirde şubesi olan geniş bir alana yayılmış şirketlerin projelerinde evrimsel geliştirme modu kullanılabilir.

**Yardımcı Kaynaklar**

* <https://furkanalniak.com/yazilim-muhendisligi-yazilim-surec-modelleri/>
* <https://caglartelef.com/yazilim-yasam-dongusu/>
* <https://medium.com/@denizkilinc/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-temel-a%C5%9Famalar%C4%B1-software-development-life-cycle-core-processes-197a4b503696>
* <https://www.codex.com.tr/yazilim-gelistirme-modelleri>
* <https://fikirjeneratoru.com/yazilim-proje-yonetimi-yontemleri/>
* <https://medium.com/@omerharuncetin/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BC-modelleri-543c7879a742#:~:text=Yaz%C4%B1l%C4%B1m%C4%B1n%20ya%C5%9Fam%20d%C3%B6ng%C3%BCs%C3%BC%20tek%20y%C3%B6nl%C3%BC,%2C%20Tasar%C4%B1m%2C%20Ger%C3%A7ekle%C5%9Ftirme%20ve%20Bak%C4%B1m.>
* <https://medium.com/@secilcor/scrum-nedi%CC%87r-6a4326951dd8>
* Doç.Dr.Deniz KILINÇ, Bakırçay Üniversitesi Yazılım Mühendisliğine Giriş dersi 2. ve 3. Hafta sunumları.