**Yazılım Yaşam Döngüsü**

**Yazılım Yaşam Döngüsü (SDLC) Nedir?**

Yazılımın da insan hayatı gibi bir yaşam süreci vardır. Yazılım geliştirildikten sonra kullanılır ve bakımı yapılır yani hayatına devam eder. Bu yazılımın yaşam sürecidir. Biz yazılımı geliştirir, teslim ederiz ardından bu yazılım müşteri ya da müşteriler tarafından kullanır. Kullanım esnasında kullanıcılar çeşitli hatalarla karşılaşabilir. Bu yüzden teslim sonrasında bakım yapılmaya devam edilerek bu hatalar giderilmeye çalışılır. Kısacası yazılımın yaşamını sağlayan geliştirme, teslim ve bakım gibi basamaklarda çeşitli aşamalar içeren bu döngüye yazılım yaşam döngüsü diyoruz. Yazılım yaşam döngüsü süreci ürünün en iyi kalitede, en az kaynak ile ve en kısa sürede ortaya çıkarılmasını amaçlar. Yazılım yaşam döngüsü iyi yapılanmış adımlarla yazılım üreten takımlara kolaylık sağlayarak ortaya hızlı, kaliteli, iyi bir şekilde test edilmiş hatalardan arındırılmış ve kullanılmaya hazır bir yazılım sunmaya yardımcı olur.

Peki bu döngünün aşamaları nedir?

**Gereksinim :** Bu aşamada öncelikle gereksinim analizi yapılır. Yani müşterinin ihtiyaçlarını ve isteklerini bu aşamada dinlenir.

**Planlama :** Planlama kısaca yazılımın üretim aşamasındaki iş dağılımının yapıldığı, yazılım maliyetinin hesaplandığı, yazılım gereksinimlerinin çıkarıldığı, fizibilite çalışmasının yapıldığı ve proje planının oluşturulduğu ilk aşamadır.

**Analiz :** Bu aşamada “Biz ne yapmak istiyoruz?” sorusuna cevap ararız. Bu bağlamda müşterinin tam olarak neye ihtiyacı olduğu belirlenir. Yazılım gereksinimleri analiz edilerek dokümante edilir.

**Tasarım :** Bu aşamada “İstediğimiz şeyi nasıl elde edeceğiz?” sorusuna cevap ararız. Yazılım gereksinimlerini yerine getirecek modelin oluşturulması işlemidir. Tasarım aşamasında UML, nesne modeli, iş akış şeması gibi farklı yöntemler uygulanabilir.

**Gerçekleştirim :** Bu aşamada yazılımımızın kodlaması ve testleri yapılır.

**Teslim :** Yazılımımızın testleri de bittikten sonra yazılımımız teslim edilir yani artık ürün kullanılabilir olur.

**Bakım :** Teslim aşaması da bittikten sonra ürüne sonradan çeşitli bakımlar yapılır.(hataların giderilmesi yeni özellikler eklemek gibi)

**Yazılım Yaşam Döngü Modelleri**

Yazılım yaşam döngü modelleri kişilerin veya projelerin ihtiyaçlarına göre çeşitlenmiştir.

Peki bu modeller nelerdir?

**1. Gelişigüzel Model**

1960’lı yıllarda kullanılan bu yöntem genellikle basit programlama içeriği olan projelerde kullanılan ve belli bir kuralı veya yöntemi olmayan bir modeldir. Genellikle kişiye bağlıdır. Bu yüzden yazılımın takibini ve bakımını yapmak bir hayli zordur. Aslında bu yöntemi bir model olarak adlandırmak pek doğru olmaz çünkü herhangi bir model ya da yöntem bulunmamaktadır.

**2. Barok Modeli**

Barok modeli 70’li yılların ortalarından başlayarak kullanılmaya başlanan aşamalar arası geri dönüşlerin nasıl yapılacağı tanımlı olmayan bir modeldir. Bu modelde yaşam döngüsü temel adımları doğrusal bir şekilde ele alınır ve geliştirilir. Ayrıca bu modelde belgeleme ayrı bir süreç olarak ele alınır ve yazılım bittikten sonra yapılır. Halbuki günümüzde belgeleme yapılan işin doğal bir ürünü olarak görülür. Günümüzde yazılım geliştirme projelerinde kullanılan bir model değildir.

**3. Çağlayan/Şelale (Waterfall) Yaşam Döngü Modeli**

Şelale Modeli 1970’li yıllarda Royce tarafından sistem mühendisliği sürecinden türetilen yazılım geliştirme sürecinin ilk yayınlanmış modelidir. Bu model genel olarak geleneksel model olarak adlandırılır. Şelale modeli statik bir yapıya sahiptir. Üretimi fazla zaman gerektirmeyen projeler için oldukça uygundur. Fakat yazılım dünyası genel olarak esnek ve çok sık değiştiğinden şelale modeli yazılım projeleri için kötü yaklaşımlardan biridir. Bu yüzden kullanımı da gün geçtikçe azalmaktadır. Yine de ilk olması ve basit olması ve diğer yaklaşımlara temel teşkil etmesi açısından önemlidir.

* **Şelale Modelinde :**
* Yaşam döngüsü temel adımları baştan sona en az bir kez izlenerek gerçekleştirilir.
* Aşamalar sırasına göre sıra sıra uygulanır. Yani önceki aşama bitmeden diğer aşamaya başlanamaz.
* Her aşamada dokümantasyon yapılmalıdır. Yani her aşamanın sonucu bir ya da birden fazla onaylanan (imzalanan) belgedir.
* Her şeyin bir dokümanının bulunması gerekmektedir ve eğer ki her safha sonunda test ve dokümantasyon olmamışsa o safhanın tamamlandığı kabul edilmez.
* Barok modeline göre dokümantasyon ayrı bir süreç olmak yerine üretimin doğal bir parçası haline gelmiştir.
* Bu modelde aşamalar arasındaki geri dönüşler barok modele göre daha iyi tanımlanmıştır.
* **Şelale Modelinin Avantajları Neler?**
* Proje ilerlemesinin düzgün bir şekilde izlenmesini, muhtemel hataların erken tespitini ve ölçülebilir yazılım geliştirmeyi sağlar.
* Projeler daha kolay yönetilebilir olur ve maliyet aşımı olmadan zamanında teslim edilir.
* Bütçeyi tahmin etme kolaylığı sağlar
* Müşteriler ve son kullanıcılar tarafından da iyi anlaşılabilen adımlardan oluşur.
* Gereksinimleri iyi anlaşılabilen projelerde iyi çalışır.
* **Şelale Modelinin Dezavantajları Neler?**
* Müşteriler, gereklilikleri tamamen, doğru ve net bir şekilde ifade etmelidir. Aksi halde kodlama ve test aşamalarında olabilecek değişikliklerin yazılıma yansıtılması maliyeti çok yüksek olacaktır.
* İleride herhangi bir fazla maliyet olmaması için planlama ve belgeleme için çok fazla zaman ve emek harcanmaktadır.
* Projenin sonunda entegrasyon ve test çok kapsamlı bir çaba gerektirir.
* Esnek değildir.
* Genel olarak yazılımın kullanıcıya teslim edilme zamanı uzundur.

**4. V Süreç Modeli (V-Shaped Model)**

Adımlar V şeklini oluşturduğu için V-Shaped Model denilmiştir.

Bu modelde sol taraf üretimi sağ taraf ise test işlemini ifade eder. Bu modelin çıktıları “Kullanıcı Modeli”, “Mimari Model” ve “Gerçekleştirim Modeli” dir.

Kullanıcı modelinde geliştirme sürecinin kullanıcı ile olan ilişkileri tanımlanmakta ve sistemin nasıl kabul edileceğine ilişkin sınama belirtimleri ve planları ortaya çıkar.

Mimari Modelde sistem tasarımı ve oluşacak alt model ile tüm sistemin sınama işlemlerine ilişkin işlevler ele alınmaktadır.

Gerçekleştirim Modelinde yazılım modüllerinin kodlanması ve sınanmasına ilişkin fonksiyonlar ele alınmaktadır.

Aslında genel yapısıyla şelale modeline benzer fakat şöyle bir farkı vardır. Yine şelale modelindeki gibi planlama, ihtiyaç belirleme vb. gibi aşamalar bulunmaktadır. Aşağıya doğru akan bir proje yönetimi söz konusudur. Buradan sonra yukarıya doğru bir çıkış başlar ve birim testlerine başlanır. Her üretilen modülün testi yapılır ve kabul testleri aşamasında müşterinin kabul edip etmemesi ile ilgili müşteriyle iletişime geçilir. Müşteri uygulamayı test eder ve yazılımın bakımına başlanır. Bu modelin bir diğer özelliği ise aynı hizada olan aşamaların birbirini karşılamasıdır. Örnek olarak; detaylı tasarım yapıldı. Bu aşamanın hemen karşısında kodlama bittikten sonra birim testleri yapılır. Yani yapmış olduğumuz tasarıma ait testler yapılır. Her bir alt modülün birbiri ile uyumlu çalışıp çalışmadığı test edilir. V şeklinde model aslında biraz daha detaya inilip sonra yukarı çıkılan ve her aşamanın karşılığının olduğu bir modeldir. “Bir şey yapıldı ama bunun karşılığı doğru mudur?” sorgulamasının yapıldığı bir yapıya sahiptir.

* **V Süreç Modeli :**
* Belirsizliklerin az iş tanımlarının belirgin olduğu bilişim teknolojileri projeleri için uygun olan bir modeldir.
* Model kullanıcının projeye katkısını arttırmaktadır.
* İş ihtiyaçları tam olarak tanımlanmış ve doküman haline getirilmiş projeler için uygundur.
* **V Süreç Modeli Avantajları :**
* Bu model oldukça disiplinli bir modeldir ve aşamaları birer birer tamamlanmıştır.
* Gerekliliklerin tam olarak anlaşıldığı küçük projeler için iyi çalışır.
* Proje yönetimi tarafında takibi kolaydır.
* Kullanımı kolaydır.
* **V Süreç Modeli Dezavantajları :**
* Karmaşık ve nesne yönelimli projeler için iyi bir model değildir.
* Aşamalar arasında tekrarlamaları kullanmaz.
* Uzun ve devam eden projeler için iyi bir model değildir.
* Risk çözümleme ile ilgili aktiviteleri içermez.
* Bir yazılım test aşamasındayken, geriye dönüp bir işlevselliği değiştirmek zordur.

**5. Helezonik (Spiral) Model**

Helezonik (Spiral) Model ilk olarak 80’li yıllarda Barry Boehm tarafından ortaya atılmıştır. Diğer modellerden farklı olarak süreçteki aşamalardan tekrar tekrar geçilmesini ve her geçişte projenin ilerleme kaydetmesi hedeflenmektedir. Örneğin Şelale Modelinde belli ve düz bir ilerleme varken bu modelde durum projedeki aşamaları spiral olarak tekrarlamayı içerir.

Bu modelde bölgeler 4 kısma ayırılır.

**Planlama :** Müşteriden gereksinimlerin alınması, prototip (ara ürün) için planlama, amaç belirleme ve varsa bir önceki adımda üretilen prototiple bütünleştirmeyi içeren adımdır.

**Risk Analizi :** Risklerin belirlenmesini ve risk seçeneklerinin araştırılmasını içeren adımdır.

**Üretim :** Bu adımda prototip üretilir.

**Kullanıcı Değerlendirmesi :** Bu adımda prototip hakkında kullanıcıdan geri dönüşler alınır.

Helezonik (Spiral) Modeli şöyle özetleyebiliriz;

İstekler, ihtiyaçlar belirlenir. Daha sonra risk analizleri yapılıp ortaya bir prototip çıkarılır. Bu prototip kullanıcıya sunulur. Kullanıcıdan bu prototip hakkında geri dönüş alınır. Örneğin müşteri “Şurası olmamış. Böyle istiyorduk ama istediğimiz gibi olmamış.” Diyebilir. Kısacası geri dönüşler alınır ve bu prototiple ilgili ihtiyaçların kontörlü yapılıp doğrulama ve onaylama sürecinden geçirilip bir daha yazılım sürecinden geçirilir. Tekrardan risk analizi yapılıp ortaya 2.prototip çıkarılır. Eğer müşteriden olumsuz bir geri dönüş alınırsa bu adımlar devam ettirilir. Olumlu dönüş alınırsa, detaylı tasarım, kodlama, entegrasyon, testler ve uygulama süreci başlar. Bu son aşama şelale modeline benzer yapıdadır.

* **Helezonik (Spiral) Modelde :**
* Spiral üzerindeki her bir halka bir fazı gösterir.
* Hedefler, istekler ve ihtiyaçlar belirlenir.
* Alternatifler değerlendirilir, riskler belirlenip çözülür.
* Prototipler geliştirilir. Yani prototip yaklaşımı vardır.
* Doğrudan gereksinim, analiz, tasarım vb. aşamalar yoktur.
* Yinelemeli ve artımsal bir yaklaşım vardır.
* Risk analizi olgusu ön plana çıkmıştır.
* Süreç boyunca risklerin değerlendirilmesi ve çözümü yapılır.
* **Helezonik (Spiral) Modeli Avantajları :**
* Gereksinimler daha doğru belirlenebilir.
* Kullanıcılar sistemi erken görebilirler.
* Kullanıcının bu modeldeki katkısı oldukça fazladır ve bu katkı ileride oluşabilecek istenmeyen durumların önüne geçilmesini sağlar.
* Geliştirme süreci daha küçük parçalara bölünebilir bu projede daha riskli kısımların önceden geliştirilmesini sağlar. Bu da bize daha iyi bir risk yönetimi sağlar.
* Hatalar erken giderilir.
* **Helezonik (Spiral) Modeli Dezavantajları :**
* Küçük ve düşük riskli projeler için pahalı bir yöntemdir.
* Yönetimi daha karmaşıktır.
* Projenin sonu başlangıçta bilinmeyebilir. Yani spiral sonsuza da gidebilir.
* Süreç karmaşıktır.
* Çok sayıda ara aşama olduğu için aşırı dokümantasyon gerektirir.

**6. Artımsal Geliştirme Süreç Modeli (Incremental Model)**

Artımsal geliştirme süreç modelinde genel olarak bir döngüden bahsedebiliriz. Bu model de diğer modeller gibi gereksinim, tasarım, uygulama ve test aşamasından geçer. Diğer modellerden farklı olarak döngüler daha küçük, daha kolay yönetilen süreçlere bölünür. Bu küçük süreçleri bir parça olarak düşünebiliriz. Şekiller parça parça eklenir ve her parçanın tamamen bitmesi beklenir. Her yeni parça bittikten sonra yineleme başlar. Her yineleme sonucunda müşteriden geri dönüşler alınır ve bu geri dönüşler sayesinde yineleme yazılımımıza yani ürünümüze yeni özellikler katar. Önceki parçanın üzerine eklemeler yapılıp hatalar düzeltilir. Sonuç olarak her parçanın yani her döngünün bitmesi beklenir. En sonunda tüm parçalar yani modüllerimiz birleştiğinde yazılım tamamen tamamlanmış ve hazır bir hale gelir.

Bu model kısacası müşterinin basit bir ihtiyacını karşılamakla başlayıp ilerledikçe daha da gelişen ve daha fazla ihtiyacı karşılayan en sonunda ise müşterinin istediklerini karşılayan bir ürün karşımıza çıkarır.

* **Artımsal Geliştirme Süreç Modelinde;**
* Bir taraftan müşteri kullanım yaparken, bir taraftan da üretim yapılır.
* Eğer değişiklik yapılacaksa bu değişiklik diğer döngüde ele alınır.
* Her döngünün sonunda üretilen sürümler birbirlerini kapsayacak ve git gide özelliklerini arttıracak şekilde ilerler.
* **Artımsal Geliştirme Süreç Modeli Avantajları;**
* Yazılım için gerekli olan gereksinimler müşteri ile etkileşim halinde bulunulduğu için çok iyi bir şekilde belirlenir.
* Müşteriden geri dönüş alınıldığı için projenin başarısız olma ihtimalini düşürür.
* Sistem özellikleri daha fazla test edilme imkanı bulunur. Bu da hatalarımızın oranını düşürür.
* Sonuçlar çok erken ve düzenli bir şekilde ortaya çıkarılır.
* Gereksinimleri değiştirmek daha az maliyetlidir.
* Riskler her yineleme sırasında tanımlanıp çözülebildiği için risk yönetimi kolaylığı vardır.
* Her döngü sonunda tespit edilen hatalar ve riskler bir sonraki adımda çözülebilir.
* Büyük ve kritik derecede önem arz eden projeler için uygundur.
* **Artımsal Geliştirme Süreç Modeli Dezavantajları;**
* Yinelemeler paralel ve fazla olduğu için yönetim karmaşıklığı daha fazladır.
* Deneyimli personellere ihtiyaç vardır.
* Her ne kadar değişim maliyeti düşük de olsa, sürekli değişen gereksinimler için çok da uygun değildir.
* Küçük projeler için uygun değildir.

**7. Kodla ve Düzelt Yaşam Modeli**

Kodla ve düzelt yaşam modelinde analiz yapılarak sorun üstünde düşünülür ve hemen ardından kodlama başlar. Kodlama aşaması bittikten sonra bir düzeltme başlar. Bu düzeltme müşterinin ihtiyaçları karşılanana kadar devam eder.

* **Kodla ve Düzelt Yaşam Modelinde;**
* Hiçbir yaşam döngü adımı gerçekleştirilmeden direkt olarak kodlamaya geçilir.
* Yazılım istenilen düzeye gelene kadar devamlı olarak geliştirme yapılır.
* Dokümantasyon bulunmaz.
* Emeklilik safhası içerir.
* **Kodla ve Düzelt Yaşam Modeli Avantajları;**
* Tek gereken yeterli düzeyde istek ve gayrettir.
* Eğer ki ürün onu yapan tarafından kullanılacaksa bu model gayet avantajlıdır.
* Yazılım geliştirmenin en kolay yoludur.
* **Kodla ve Düzelt Yaşam Modeli Dezavantajları;**
* Kodlamaya başlarken herhangi bir değişiklik bilinmediğinden, bir sürü değişiklikten sonra kod karmaşık bir hale gelir ve sonraki düzeltmeleri yapmak oldukça zorlaşır.
* Müşteri sürece dahil olmadığından ihtiyaçlara uygun olmayabilir.
* Yazılım geliştirme süreçlerinden en pahalısıdır.
* Takımlar için uygun değildir.

**Agile (Çevik) Yazılım**

Gelişen ve sürekli değişen yazılım dünyasında diğer modellerden elde edilen tecrübe ve birikim sonrasında, kullanıcıların yazılımı daha hızlı ve ucuz bir şekilde istemesinin üzerine çevik yazılım ortaya çıkmıştır. Çevik geliştirmede, süreç ve belgeleme yerine doğrudan yazılımın kendisine yoğunlaşan ve yinelemeli geliştirmeye dayanan teknikler bulunur. Teknoloji ve kullanıcı beklentilerindeki sürekli değişimlere bir çözüm olarak ortaya çıkmıştır. Agile kısaca bir manifestodur. Esasını 4 temel madde oluşturur ;

* Süreçler ve araçlardan ziyade bireyler ve etkileşimlere
* Kapsamlı dokümantasyondan ziyade çalışan yazılıma
* Plan izlemek yerine değişime karşılık vermeye ve değişime değer vermeye
* Sözleşme görüşmeleri yerine müşteri ile iletişime ve iş birliğine bağlı kalmak daha önemlidir.

ve bu maddelerin altında da 12 alt madde vardır;

* En önemli öncelik, yazılımın hızlı ve devamlı teslimini sağlayarak müşterileri memnun etmektir.
* Gereksinim değişiklikleri yazılım sürecinin ilerleyen aşamalarında bile kabul edilmelidir.
* Çalışan yazılım, kısa zaman aralıklarıyla düzenli bir şekilde müşteriye sunulmalıdır.
* Tüm ekipteki kişiler proje boyunca birlikte çalışmalıdır.
* Projelerin temelinde motivasyonu yüksek kişiler bulunmalıdır. Ekibe ihtiyaçları olan ortam ve desteği sağlamak ve onlara işi başaracakları konusunda güven duymak önemlidir.
* Yazılım takımında bilgi alışverişinin verimli olabilmesi için yüz yüze iletişim önemlidir.
* Çalışan bir yazılım ilerlemenin birincil ölçüsüdür.
* Çevik süreçler sürdürülebilir geliştirmeyi destekler. Sponsorlar, yazılımcılar ve kullanıcılar sabit tempoyu sürekli devam ettirebilmelidir.
* Sağlam teknik altyapı ve tasarım, çevikliği arttırır.
* Sadelik yani basitlik olmazsa olmazlardandır.
* En iyi mimariler ve tasarımlar kendi kendini örgütleyebilen takımlardan ortaya çıkar.
* Takım, düzenli aralıklarla kendi yöntemlerini gözden geçirir. Nasıl daha etkili ve verimli olabileceklerinin üzerinde düşünürler ve buna yönelik gerekli iyileştirmeler yaparlar.

**Çevik Yazılım’ın Geleneksel Yöntemlerden Farkı Nedir?**

|  |
| --- |
| Sağa Doğru Şapka Karakteri düz dolguylaGeleneksel Yöntemler Çevik Yöntemler  Sağa Doğru Şapka Karakteri düz dolguylaMüşteriler ne istediğini iyi bilir. Müşteriler ne istediğini keşfeder.  Geliştiriciler neyi, ne şekilde üreteceklerini iyi bilir. Geliştiriciler neyi nasıl üreteceğini keşfeder.  Sağa Doğru Şapka Karakteri düz dolguylaYazılım süreci boyunca hiçbir şey değişmeyecektir. Yazılım süreci boyunca birçok değişim yapılabilir. |

Agile (Çevik) Yazılımda farklı geliştirme yöntemleri bulunur. Bunlardan bazıları; Extreme Programming (XP), SCRUM, Kristal Yöntem, Dinamik Sistem Geliştirme Yöntemi (DSDM), LEAN Development, Özellik Güdümlü Geliştirme (FDD) dir. En çok kullanılanlar ise **Extreme Programming(XP)** ve **SCRUM**’dır.

**Extreme Programming (XP)**

Ward Cunningham, Kent Beck ve Ron Jeffries tarafından ortaya atılan, Kent Beck’in yayınladığı kitapla tüm dünyaya duyurulan Extreme Programming modeli çok geniş bir uygulama alanına sahiptir. Xp yazılım süreci boyunca kaliteli, çalıştırılabilir kod üretmeye odaklanmış, basit, grup içi iletişime önem veren, geri dönüşlerin daha fazla olmasına imkan sağlayan bir yazılım geliştirme yöntemidir. Xp 4 aşamadan oluşur; Planlama, Tasarlama, Kodlama, Test. Bu aşamalarla geleneksel yaklaşıma benzese de farklı olarak küçük sürümlere sahiptir ve bu parçalar bir araya getirildiğinde ana resme erişilir. Bu, geleneksel yaklaşımdan önemli bir farktır.

**1.Planlama :** Kullanıcı senaryoları yazılır. Yazılım sürüm yayınlaması için zaman planlamaları yapılır. Proje küçük parçalara bölünerek, hızı sürekli olarak ölçülür. Küçük sürümler genellikle bu aşamada çıkarılır.

**2.Tasarlama :** Bu aşamada her şey basitleştirilmeye çalışılır. Doğru zaman olmadan herhangi bir işlev eklemesi yapılamaz fakat herhangi bir zamanda ya da çoğunlukla ihtiyaç halinde yeniden düzenleme yapılabilir.

**3.Kodlama :** Kodlama standartlarında bir kodlama görülür. Bu kodlamalar birim testleriyle sürekli kontrol edilir. Verimliliğin düşmemesi için bu aşamada fazla mesai yapılmaz. Kodun sahipliği projedeki ekibe aittir.

**4.Test :** Tüm kodların birim testi artık oluşmuştur. Bir hata ile karşılaşıldığında yeni testler hazırlanır. Kabul testleri ise genellikle müşteri tarafından hazırlanır ve takip edilir.

Extreme programming basitlik, iletişim, geribildirim ve cesaret olarak dört temel değere dayanır. Bu dört temel değeri özetlemek istersek;

Bir yazılıma başlanırken asla ama asla başarısızlıktan korkmamalıyız. Bunun yerine projelerin üzerine gidilmesi ve geliştirilmesi projeyi başarıya götürür. Proje, daha iyi çalışabilmesi için kolaylıkla değiştirilebilir. Çünkü testler çok sık ve sağlam bir şekilde yapılır. Kodda beğenilmeyen bir yer olması halinde o modül çöpe atılır ve yeniden yazılır. Bu şekilde korkmadan bu değişiklikleri yapmamız gerekir. Bir diğer önemli değer ise iletişimdir. Bir projede sağlıklı iletişim olmazsa projedeki ekip birbirleriyle tam olarak anlaşamaz ve projede bir kopukluk olur. Öte yandan projenin sürekliliği, projenin hızı, projenin verimliliği düşer ve bu sorunlar projenin başarısız olmasına sebep olabilir. İletişim kadar geri bildirim de çok önemlidir. Geri bildirim sayesinde ortaya çıkabilecek hatalar ortadan kaldırılır. Son olarak XP’nin yapı taşlarından biri olan basitlik, çok zor sağlansa da yapılması en gerekli işlemdir. Çünkü karmaşık yapı ve çözümler XP’nin mantığına aykırı olmakla birlikte bizi yavaşlatacaktır. Xp bu bakımdan esnek ve basit bir sistem hedefler.

**SCRUM**

1990’ların başında Jeff Sutherland ve Ken Schawaber tarafından tasarlanan çevik bir yazılım geliştirme yöntemidir. Scrum birbiri arasında tekrar eden, aşamalı bir çevik geliştirme yazılım yöntemidir. Scrum projeyi en başından sprint adı verilen mümkün olan en küçük birimlere böler ve bu sayede verimli zaman kullanımını maksimum seviyeye çıkarılır. Müşterilerin ihtiyaçlarındaki değişimlere hızlı cevap verir. Bu yönüyle gözleme dayalı bir yaklaşım olarak tanımlanır. Scrum şeffaflık, denetleme ve uyarlama prensiplerini temel alarak yazılım geliştirmeyi öngörür. Bu temel prensiplerin yanında scrum 3 temel kavrama dayanır. Bunlar Roller, Toplantılar ve Bileşenlerdir. Gelin bunlara bir göz atalım.

**1.Roller**

**Ürün Sahibi :** Ürünü tanımlayan, kapsamını belirleyen, ihtiyaç duyulması halinde her sprintin önceliği ve özelliği üzerinde değişiklik yapan, ürünün sonucundan sorumlu olan ayrıca müşteri ile geliştirme ekibi arasındaki iletişimden sorumlu olan kişidir.

**Scrum Yöneticisi :** Scrum’ın temel değerlerini, kurallarını ve pratiklerini iyi bilen ve takımın bu kuralları uygulamasını sağlayan kişidir.

**Scrum Takımı :** Kendi kendilerini örgütleyebilen ve bir sprinte alınan bütün işleri tamamlayacak özelliklere sahip kişilerdir. Kısacası projenin geliştirilmesinden sorumlu kişilerdir.

**2.Toplantılar**

**Günlük Scrum Toplantısı :** Ekip üyeleri tarafından her gün yapılan 15 dakikalık toplantılardır. Her üye bu toplantıya katılır ve takımdaki herkes kendisine dün ne yaptım, bugün ne yapacağım, herhangi bir sorun var mı? sorularına yanıt ararlar.

**Koşu (Sprint) Planlama Toplantısı :** Product backlog ile belirtilen gereksinimler, bu toplantıda küçük görevlere ayrılır. Bu görevler kişilerin kendi hızlarına ve isteğine göre dağıtılır. Kısacası görev dağılımı yapılır.

**Koşu (Sprint) Gözden Geçirme Toplantısı :** Her sprintin sonunda yapılan bu toplantıda, yapılan sprintler gözden geçirilir ve ortaya çıkan ürün değerlendirilir. Bu toplantıdaki amaç yazılımın gereksinimlere uygun olarak geliştirildiğinden emin olmaktır.

**3.Bileşenler**

**Product Backlog :** Proje için gerekli olan gereksinimler listesidir.

**Sprint Backlog :** Sprint içerisinde yapılacak işlerin öncelik sırasına göre sıralanmış listesidir. Scrum takımı tarafından yapılır.

**Burndown Chart :** Sprintin günlerini ve kalan işi gösteren bu grafik, işlerin ne kadarının yapıldığını ve normalde ne kadar yapılması gerektiğini gösterir. Scrum’ın prensiplerinden olan şeffaflığı da sağlar.

|  |
| --- |
| SCRUM XP (EXTREME PROGRAMMING)  Sağa Doğru Şapka Karakteri düz dolguylaSprint (2 hafta – 1 ay) Sprint (1 – 2 hafta)  Sağa Doğru Şapka Karakteri düz dolguylaSprintler en son halinde gelip, Sprintler değişebilir. toplantı yapıldıktan sonra değişmez.  Sağa Doğru Şapka Karakteri düz dolguylaHerhangi bir mühendislik pratiği Mühendislik pratikleri tanımlar. tanımlamaz. (Eşli programlama, otomatik test vs.) |

**Scrum Neden Popüler?**

Scrum'ın günümüzdeki popülerliği için birçok etkenden bahsetmek mümkündür. Özellikle scrum sadece yazılım alanında değil de her türlü alanda uygulanabilir bir model olması bence bu modelin popülerliğini arttırmasının altında yatan sebeplerden biridir. Gelin Scrum'ın neden popüler olduğuna maddeler halinde göz atalım;

* Karmaşık yapıları basite indirgemesi sayesinde şirketin hem boşa zaman kaybetmesini engeller hem de boşa harcanılan kaynaktan tasarruf etmemizi sağlar.
* İş ihtiyaç listesindeki iş miktarının belirlenmesinin zor olduğu projelerde başarılı bir şekilde geliştirme sağlar.
* Özellikle geleneksel yöntemlerdeki bakım sürecinde karşılaşılan hatalar aşırı derecede maliyetli bir hal alır. Fakat Scrum'da sürekli olarak yapılan kısa toplantılar yoluyla işteki ilerlemenin sık sık güncellenmesi sayesinde hem kolaylıkla proje kontrol edilebilir olur hem de sonradan gün yüzüne çıkıp bize sıkıntı ve maliyet çıkaracak hatalar önceden giderilir.
* Kullanıcıdan sürekli geri bildirim sağladığı için tercih edilme nedenlerinden biridir.
* Günlük toplantılar bireysel üretkenliğin ölçülmesini mümkün kılar ve bu şekilde ekip üyelerinin her birinin üretkenliğinde iyileşme olur. Bu yönüyle de hem projenin daha başarılı olmasına hem de ekip çalışmasına doğrudan katkı sağlar.
* Scrumda tüm zamanlar planlanmış olduğu için kaliteli bir ürün sunmak daha kolaydır.
* Süreç ve yönetim bakımından genel maliyet minimum düzeydedir. Bu sayede hızlı ve daha ucuz bir sonuç elde edilir.

Scrum bu denli fazla avantajı bulunması sayesinde günümüzde yazılım geliştiriciler arasında popülerliğini gittikçe arttırmıştır.

**Hangi Projede Hangi Modeli Kullanmalıyız?**

Her türlü projeye farklı farklı modeller uygulanabilir. Fakat projenin yapısına uygun bir model seçilmediği zaman projenin maliyeti, verimliliği, harcanan zaman, kaynak tüketimi fazlaca artar. Böylece projenin başarısız olma ihtimali de artar. Bu nedenle projeye uygun bir model seçmek çok önemlidir.

Projelerin genel yapısına bakılarak hangi modelin uygun olduğu modellerin tanımından, avantajları ve dezavantajlarından belirlenebilir.

* Üretimi fazla zaman gerektirmeyen, gereksinimleri çok iyi tanımlanmış projeler için Şelale Modeli uygun olan bir modeldir.
* Belirsizliklerin az iş tanımlarının belirgin olduğu bilişim teknolojileri projeleri için V Süreç Modeli uygun olan bir modeldir.
* Yüksek hata riskli, maliyeti yüksek ve uzun zaman alabilecek projeler için Helezonik (Spiral) Model uygun olan bir modeldir.
* Büyük ve kritik derecede önem arz eden projeler için Artımsal Geliştirme Süreç Modeli uygun olan bir modeldir.
* Küçük yapıdaki kişisel yazılımların geliştirilmesi için Kodla ve Düzelt Yaşam Modeli uygun olan bir modeldir.
* Küçük ve orta çaplı, kısa bir zamanda bitirilmesi hedeflenen projeler için Agile (Çevik) Yöntemler uygundur.

Kaynaklar :

[1]J.Glenn Brookshear Bilgisayar Bilimine Giriş

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

[3]<https://caglartelef.com/yazilim-yasam-dongusu/>

[4]<https://acikders.ankara.edu.tr/mod/resource/view.php?id=70050>

[5]<https://webdosya.csb.gov.tr/db/cbs/icerikler/salihsoylu_tez_v10-20180925134450.pdf>

[6]<https://akademiksunum.com/index.jsp?modul=document&folder=a93e3a2fccf8eb56a557c55c5f0d5cf10789abe2>

[7]<http://ybsansiklopedi.com/wp-content/uploads/2015/08/Yaz%C4%B1l%C4%B1m-Geli%C5%9Ftirme-Modelleri-Yaz%C4%B1l%C4%B1m-Ya%C5%9Fam-D%C3%B6ng%C3%BCs%C3%BCSDLCYBS.pdf>

[8]<https://stackify.com/what-is-sdlc/>

[9]<http://web.firat.edu.tr/mbaykara/CevikYazilim.pdf>

[10]<http://www.yilmazcihan.com/agile-modelleme-cevik-modelleme/>

Alperen Çorak

İzmir Bakırçay Üniversitesi Bilgisayar Mühendisliği 1.Sınıf Öğrencisi

LinkedIn : <https://www.linkedin.com/in/alperencorak/>

Makalenin Medium Linki: <https://medium.com/@alofdean/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-ve-modelleri-f8a1ac9d26f2>