YAZILIM YAŞAM DÖNGÜSÜ NEDİR VE MODELLERİ NELERDİR?

Yazılımın kodlamadan oluştuğu ve tek seferde yazılıp bittiğini düşünen çoğu kişinin aksine yazılım bir organizmadır ve kodlamadan ziyade yazılımın kalitesi ne kadar süre piyasada kalacağı; ne kadar değişime uyumlu esnek bir yazılım olduğu ve oluşum + bakım sürecinin maliyeti ile doğru orantılıdır. Yazılımı kullanma ihtiyacımız ise bir problemi çözme isteğimizden gelmektedir. Bu yazılımın üretim ve kullanım aşaması birlikte olmak üzere geçirdiği tüm aşamalar yazılım yaşam-döngüsünün tanımı niteliğindedir. Bu ihtiyaçlar sürekli değişip geliştiği için mutlaka döngü biçiminde düşünülmelidir. Tek yönlü yani doğrusal bir üretim ile yazılım elde etmek kesinlikle düşünülmemelidir. Yazılım Yaşam-Döngüsünün temel aşamaları şu şekildedir:



**.** Planlama (Planning)

**.** Analiz (Analysis)

**.** Tasarım (Design)

**.** Gerçekleştirme (Implementation)

**.** Bakım (Maintenance)

1.) PLANLAMA:

İlk olarak müşteri gereksinimlerinin elde edildiği proje yapmaya karar verme süreci boyunca teknik ve finansal açılardan gerekli araştırmalar ile değerlendirmelerin yapıldığı aşamadır. Başlangıç aşamasıdır. Tüm proje planlaması bu aşamada gerçekleşir.

2.) ANALİZ:

Yazılımın gereksinim ve işlevlerinin ayrıntılı bir şekilde incelendiği ve ortaya çıkarıldığı aşamadır. Mevcut işler incelenir, temel sorunlar ortaya çıkarılır. Bu aşamada sistem analisti ve müşterinin daha sık görüşmeler yapması projenin ilerleyişinde gelişebilecek küçük sorunların oluşmadan yok olmasını sağlayabilir. Görüşmeler sonucunda müşteri beklentileri tam olarak anlaşılır ve bu beklentiler doğrultusunda bir harita oluşturulur. Çeşitli yazılım metodolojilerinde bu aşamada kullanım dokümanları ve test plan dokümanları da oluşturulabilir.

3.) TASARIM:

Bu aşama bir süreç şeklindedir. Yazılım sisteminin temel yapısının oluşturulduğu aşama olduğu için mutlaka yeterli ve dikkatli zaman ayrılmalıdır. Gereksinimlerin tamamlanması ile yazılımın ve sistemin tasarımı yapılır. Ve bundan önceki iki aşamadan faydalanılarak gerekli gereksinimleri karşılayacak yazılımın bir projesi çizilir. Mimari Tasarım dokümanları oluşturulur. Buna göre projede hangi arayüzlerin kullanılacağı, yazılım ürününün özellikleri ve yetenekleri ortaya konulur. Tasarımdaki ilk aşama basit ve anlaşılır olmalıdır. Böyle olması ilerleyen aşamalarda işlerimizi kolaylaştırır. Tasarım aşaması iki bölümden oluşur:

3.1) MİMARİ TASARIM: Modüllerin belirlenmesi ve genel olarak bir plan yapılması aşamasıdır. Yazılım modüllerinin süreç içerisindeki etkileşimleri ile ilgilenir. Mimari tasarım dokümanları oluşturulur. Modüller, akış şemaları bu aşamada oluşan doküman örneklerindendir.

3.2) DETAYLI TASARIM: Bu aşamada “Nasıl?’’ ve “Ne?’’ sorularına cevap aranır. Tasarıma bu sorulara cevap verilerek devam edilir. Bu aşamada kullanılan en önemli yöntem soyutlama yöntemidir. Bu kavram daha çok nesneye yönelik programlama ile ön plana çıkar. Nesne tanımlanırken verinin kullanılması veya veriye erişilmesi için gerekli detayların azaltılması işlemidir. Veri yapıları, ekran tasarımları bu aşamada ortaya çıkar.

4.) GERÇEKLEŞTİRME:

Bu aşama çözüme ulaşmış; planlanması, analizi ve tasarımı tamamlanmış olan sistemin kodlanma aşamasıdır. Müşteriye teslim edilecek ürünün yapımının gerçekleştiği aşamadır. Modüllerin tümü kodlanıp birleştirilip test edilir.

Kodlama yaparken kodlama kalite standartlarına uymamız gerekmektedir. Buna uymamız hem bize hem de ürünü inceleyecek olan başka bir yazılımcıya kolaylık sağlar. Kodlama yaparken kodları birbirine sıkı sıkıya bağlamamamız daha esnek bir kodlama yapmamız daha faydalı olacaktır. Bunu gerçekleştirmemiz sonradan karşılaşabileceğimiz herhangi bir sorunda kodu yeniden yazmamızı ve sorunu gidermemizi kolaylaştırır. Yazılımın mümkün olduğunca erken test edilip sorunların giderilmesi para kayıpları ve müşteri ile olumsuz tablolar sonuçlanmasından bizi kurtarır. Bu aşamada kurulum çalışmaları yapılır ve sistem artık hayata geçer. Ürün piyasaya veya müşteriye sunulur.

5.) BAKIM:

Yazılım ürünü piyasaya veya müşteriye sunulduktan sonra bakım aşaması başlar ve bu programın ömrü boyunca devam eder. Bakımın temel amacı var olan hataları gidermek veya oluşabilecek sorunları engellemek ve programın eksikliklerini tamamlamak bununla birlikte ihtiyaç duyulan yeni özellikler ekleyerek ürünü geliştirmektir. Yapılan bu bakım ve çeşitli güncellemeler sonucunda da versiyonlar oluşur. (54.12.22-54.12.24 e geçiş gibi) Bu aşama sürekli devam eden bir aşamadır. Yani yaşayan bu organizma ölene dek.

YAZILIM YAŞAM DÖNGÜ MODELLERİ

1.) Gelişigüzel Model:

Bir yöntem veya model olarak adlandırmak doğru değildir. Gelişigüzel geliştirmede yöntem ve model bulunmaz. Kişiye özel yazılım geliştirme yapılır ve gözetilmesi ve bakım yapılması oldukça zordur. Bütçe kişiye bağlı olarak değiştiğinden geliştirilebilirlik kısıtlıdır.1960’lı yıllarda kullanılmış olup genellikle basit programlama içerip çoğunlukla tek kişinin üretim yaptığı yöntemdir. Basit öğrenci projelerinde genel olarak kullanılır. Onun dışında yazılım dünyasında pek rastlanmaz.

2.) Barok Model:

Yazılım yaşam döngü adımları doğrusal bir biçimde ele alınır ve gözden geçirilir.1970’li yıllarda ortaya çıkmış Döngü yoktur bu yüzden aşamalar arasında geri dönüşlerin nasıl yapılacağı belirsizdir. Bu modelde dokümantasyon günümüz modellerinden farklı olarak ayrı bir süreç olarak ele alınır. Yazılım tamamlanıp test aşaması bittikten sonra yapılır. Gerçekleştirim aşamasına ağırlık veren bir model olup günümüzde kullanımı görülmemektedir. Bunun en önemli nedeni müşteri dönütlerinin cevapsız kalma ihtimalinin büyüklüğüdür.

3.) Şelale Yaşam-Döngü Modeli (Waterfall):

Çağlayan modeli olarak da bilinen şelale modeli en eski, en tanınmış ve en temel modeldir. Bu model bazı hükümet standartlarına girmiştir. Baştan tanımlanmış sistemler için daha çok tercih edilmektedir. Bu modelde oluşturulacak sistemlerin her birini bir proje olarak ele almak gerekmektedir. Bu modelde yazılımın yaşam döngüsü aşamaları lineer bir şekilde uygulanır. Ve aşamaların en az bir kez tekrarlanması ile geliştirilir. Kullanımı ve anlaması basit bir uygulamadır. Yönetimi kolaydır. Barok modelinin aksine burda aşamalara nasıl geri dönüşlerin yapılacağı tanımlıdır. Ve dokümantasyon üretimin bir parçası halinde seyreder. Diğer birçok modelin temelini oluşturan bu yapı yazılımda büyük değişiklik ve yeniliklere pek geri dönüş sağlayamadığından günümüzde kullanımı gittikçe azalmaktadır. Sürekli kendinden önceki aşamalara dönüş sağlayan bu yapıda işler aşama aşama gerçekleşir.

Bu aşamalar ve dönüşler şu şekilde gerçekleşmektedir:

**Değişen İhtiyaçlar**

**ANALİZ**

**PLANLAMA**

**TASARIM**

**KODLAMA**

**TEST/BAKIM**

**EMEKLİLİK**

Modelde her aşamada dokümantasyon yapılır her şeyin dokümanının olması gerekir ve bu durum da işlerin karmaşık bir hal almasını engeller. Aşamaların tamamlanması için o safhada dökümantasyon test işlemlerinin tamamlanmış olması gerekir. Modelin avantajları ve dezavantajları şu şekilde söylenebilir:

Avantajları

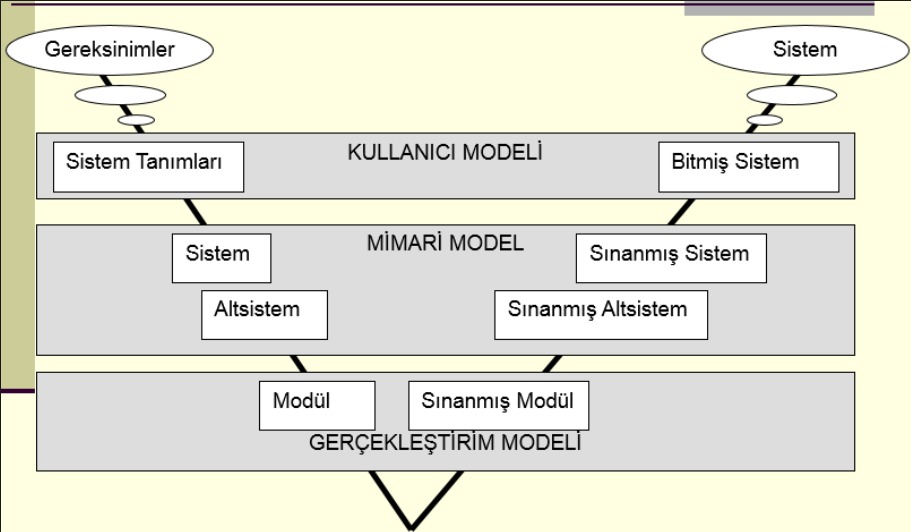
* Eski model olduğu için yaygın kullanımdadır.
* İyi tanımlanmış adımlar mevcuttur her adım dokümante edilir.
* Çok iyi tanımlanmış ve kısa sürede bitebilecek projelerde kullanımı rahattır.
* Öncesinde defalarca başarıya ulaştırılmış yani rutin olan projelerin yürütülmesi için kullanılabilinir.

Dezavantajları

* Yazılımın kullanıcıya ulaşma serüveni uzun sürer.
* Yazılım üretim ekipleri aşamalardan biri olan kod yazma programı çalıştırma hemen sonucu görme heyecanında olduklarından bu model ile yapılan üretimlerde ekip mutsuzlaşmakta ve geri kalan aşamalara önem vermemektedir.
* Kullanıcı süreçte yer almaz ve bu durum yazılım teslime geldikten sonra geri dönüşleri talepleri arttırabilir. Bu talepler büyük mali kayıplarla sonuçlanan durumlardır. Bu durum müşteri ve üretici arasında da memnuniyetsizlik yaratır.

4.) V Süreç Modeli:

Aşamalar arasında kodlama aşamasından sonra yukarıya doğru ilerlediği ve V şeklini almasından ismini alır. Aşağıdaki görselde Üretim ve Sınama ile sırasıyla sola ve sağa ayrılmış aşamalar görülür:



**Sınama (Test)**

**Üretim**

Modelin kullanımının rahat olmasına karşın aşamalar arası tekrarlama gerçekleşmez. Bu durum ise riskli yazılımların başarıyla sonuçlanma ve düzeltilme ihtimalini azaltır. Belirsizliklerin az iş tanımlamalarının net olduğu projelerde rahatlıkla kullanılabilinir. Model kullanıcıyı sürece dahil etmekte olup kullanıcıya projeye katkı imkanı sunmaktadır. Üç modelden oluşmaktadır. Bunlar:

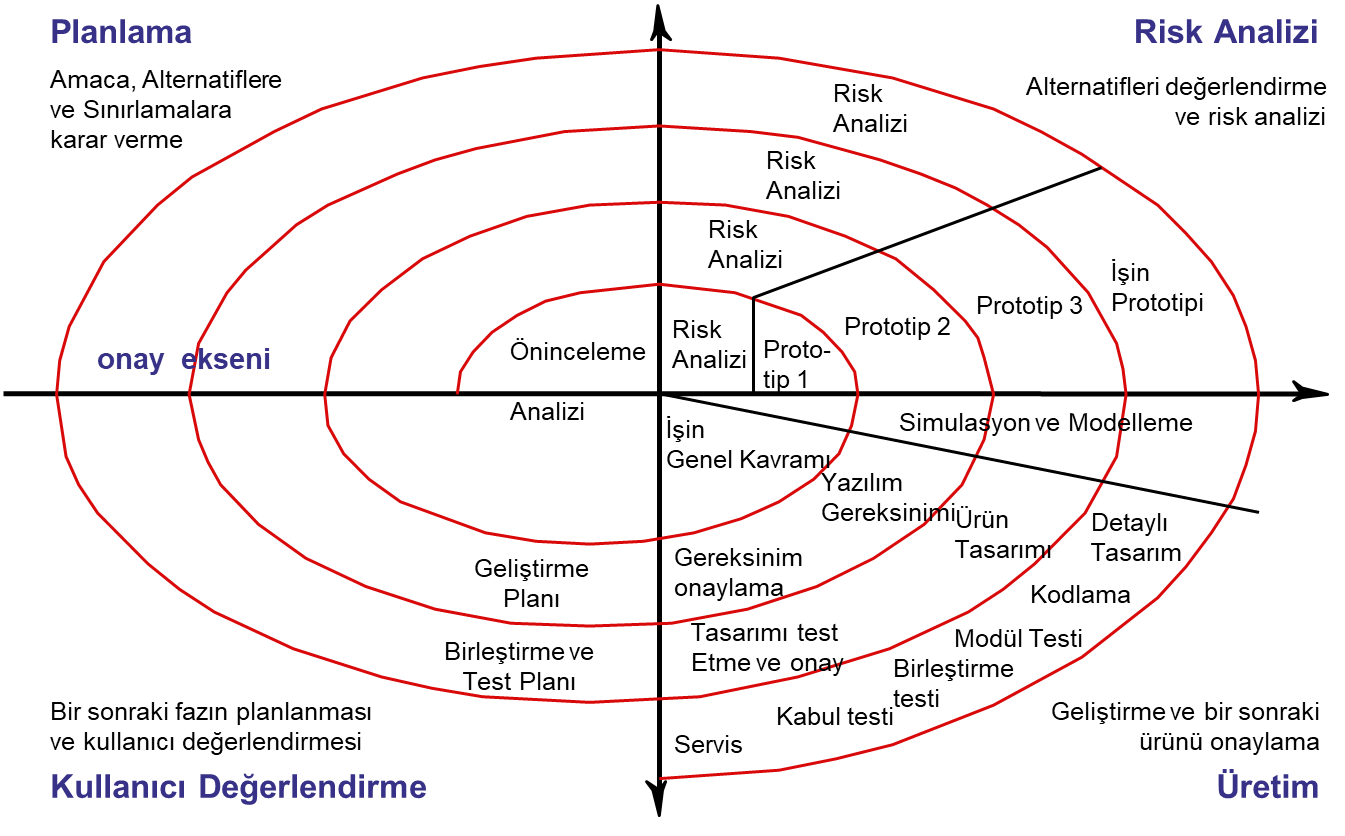
Kullanıcı Modeli: Gelişme sürecinde kullanıcının da yer aldığı modeldir. Burada sürecin kullanıcı ile ilişkisinin tespiti sağlanmakta ve sistemin kullanıcıdan nasıl bir tepki alacağına dair anketler yapılmakta ve kullanıcının beklentileri ortaya çıkarılmaktadır.

Mimari Model: Projenin sistem tasarımı ve sistemde oluşacak sıkıntılara karşın tüm sınama işlemlerine ilişkin işlevler çıktı verilir.

Gerçekleştirim Modeli: Yazılım ürünlerinin kodlanması ve test edilmesine ilişkin verileri çıktılar.

V Süreç Modeli Bilgi Teknolojileri projesinin iki aşamalı olarak ihale edilmesi için çokça uygundur.

5.) Helezonik (Spiral) Model:



Adından da anlaşılacağı gibi spiral bir döngü içerir. Döngü bir adımın son ürününün diğerinin ham ürünü olacak şekilde ilerlemesinden oluşur. Sık sık tekrarlara yer verilmesi sayesinde hata oldukça aza iner ve hatalar en erken şekilde tanımlanır. Temel dört aşaması ise şu şekildedir:

**1.PLANLAMA:** Her aşamada o aşamadaki ara ürün için bir planlama yapılır ve bir önceki ara ürün ile bütünleştirme sağlanır.

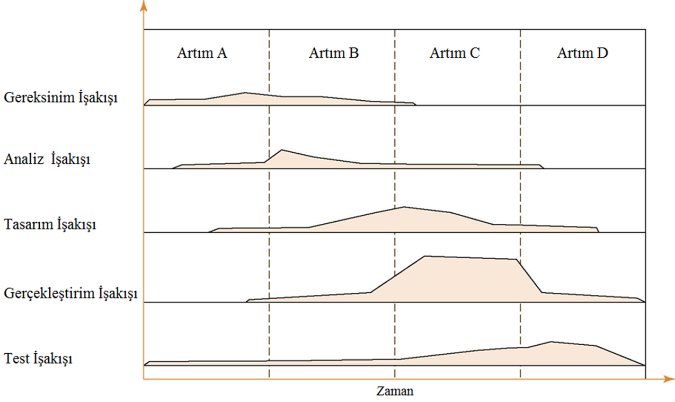
**2.RİSK ANALİZİ:** Sistemdeki risklerin araştırılması,tespiti ve çözülmesi sağlanır.

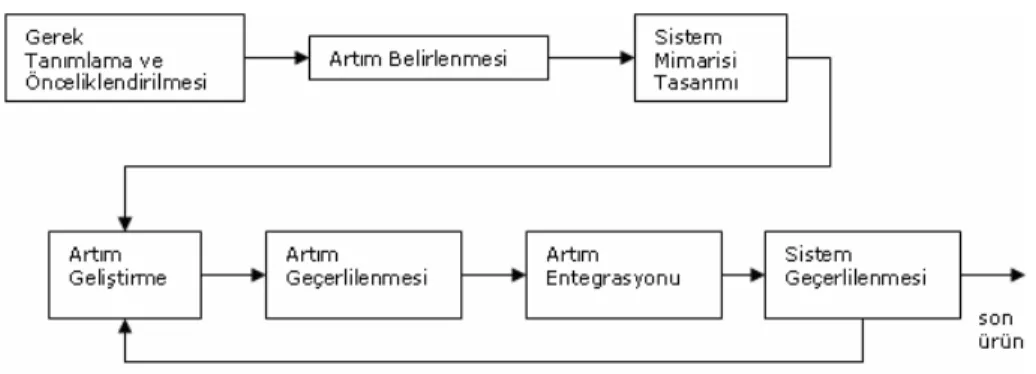
**3.ÜRETİM:** Ara ürünün üretilmesi sağlanır.

**4.KULLANICI DEĞERLENDİRMESİ:** Ara ürün ile ilgili kullanıcı tarafından test edilme ve gerekli değerlendirme/yorumlar alınır ve spiralin ilk adımına dönülür.

Bu modelde risk analizi ön plana çıkarılmıştır. Doğrudan olan yazılım geliştirme fazları mevcut değildir.(gereksinim, analiz, tasarım…vs. gibi) Yinelemeli ve artımsal (Iterative and Incremental ) yaklaşım sergiler. Prototip yaklaşımı mevcuttur. Kullanıcı, yönetici ve yazılım geliştirici tarafından farklı farklı avantajlara sahiptir.

6.) Artımsal Geliştirme Süreç Modeli:

Parça parça ihtiyaçların giderilmesi ile başlayarak süreç sonunda ana hedefe ulaşan bir ürüne ulaşma imkanı sunar. Her aşamada beklenen sonuç için bir adımı tamamlar ve amaca uygun bir işlev kazandırır. Üretim ve kullanıcıya ulaşılan süre beraber işler. Kullanıcı ilk aşama sonucunu kullanıyorken daha da geliştirilip imkanlar dahilinde işlevsellik sürekli bir veya birkaç adım ileriye taşınır. Kullanıcı gereksinimleri önceliklendirilip gereksinimlerin birbirlerine bağlılıkları ile gerçekleştirim aşamasına ulaşmak için hangi yollardan geçileceği sıralanır. Öncelikli gereksinimler erken teslim edilip kullanıcılardan çeşitli sınamalar ile dönütler alınır. Sonraki aşamaya bu dönütler doğrultusunda ilerleme haritası şekillenir. Bir parçanın geliştirilmesi başladığında gereksinimleri dondurulur. Eklenen gereksinimler sonraki teslimlerde ele alınır. Bu süreç aşağıdaki gibi ilerler:



Bitmemiş sistem

Sağdaki grafikte de görüldüğü üzere Arttırımlar ilk aşamalardan başlayıp sırayla devam etmiş olup gelişerek son bulmuştur. Her arttırımda kendinden önceki arttırımın kaldığı aşamadan başlayıp devam etme şeklinde ilerlemiştir. Bu ise yazılımın iteratif bir süreç olduğunun en güzel kanıtıdır. Her aşamada sonuca biraz daha yaklaşılmıştır. Tekrar olmadan yazılım mühendisliği çalışmaz. Erken teslimler protatip vazifesi görür.Projenin komple batma riskini azaltması modelin avantajıdır.

7.) Kodla ve Düzelt Yaşam-Döngü Modeli:

İlk bölümde yazılım ürününün ilk sürümü geliştirilir. Direkt olarak yazılım ürünü kodlanır. Sistem en son istenen şekle gelinceye kadar devamlı geliştirmeye devam edilir. Sisteme ait dokümantasyon olmadığından bakım safhası olsa da çok zor adeta işlevsizdir. Ayrıca emeklilik safhası da mevcuttur.

Yazılım geliştirmenin en kolay yoludur. Ancak en pahalısıdır.(Yazılım ürününde değişikliklerin maliyeti göze alındığında)Genellikle öğrencilerin de kullandığı bu model ne yazık ki yazılım geliştirmek kolay olduğu için küçük ya da tecrübesiz firmalarda kullanılır.

SCRUM GÜNÜMÜZDE NEDEN POPÜLER?

Öncelikle Scrum adı, Rubgy sporundaki bir hücum taktiğinin adıdır. Bu taktikte top, tüm oyuncularla birlikte karşı sahaya taşınarak atak yapılmaktadır. Çevik yazılım teknikleri arasında en çok tercih edilen yazılım geliştirme tekniğidir. Scrum, Jeff Sutjerland ve Ken Schawaber tarafından 1990’lı yılların ortalarında geliştirilen, çevik yazılım geliştirme metodolojileriyle uygulanabilecek bir proje yönetim yaklaşımıdır. Yani sadece yazılım geliştirmeye değil hayatın her alanına uygulanabilinecek bir tekniktir.

Scrum tekniğinde projeniz bağımsız olarak ele alınan daha küçük sprintlere ayrılır. Ve bu sprintlerin her biri yaklaşık 2-4 hafta sürer.Ve bu süreçte günlük 15 dakikalık toplantılarla sürekli iş takibi yapılır. Bu metodoloji kompleks ortamlarda adım adım yazılım ürünü geliştiren ekipler için uygundur. Her sprint tamamlandıktan sonra size geri bildirimlerini veren başka bireylere sunulur. Ardından bir sonraki sprinte geçmeden önce onların önerileri alınıp uygulanır. Bu yöntem klasik metodolojideki (şelale) herhangi bir proje yönetiminden çok farklıdır.

Scrum müşteri isteklerine hızlı cevap verebilen esnek kalitesi yüksek çıktılar üretilebilinmesine ve Scrum ekibinin kendi kendini organize ve motive etmesine imkan sağlayan bir yöntemdir.

SCRUMDA TEMEL KAVRAMLAR:

1-Roller:

* **Ürün Sahibi (Product Owner) :**Projenin iş açısından geri dönüşünden sorumlu olan kişidir.Ekibin bir parçası,müşterinin kendisi,müşteri veya müşteri temsilcisi tarafından görevlendirilmiş herhangi biri biridir.Ekibin bir parçasıdır. Ekiple birlikte hareket eder ve projede bizzat yer alır. Gerekli geri dönüşleri müşteriye iletir veya müşteriden alır.
* **Scrum Yöneticisi (Scrum Master) :** Takımın Scrum’ın temel değerlerine, pratiklerine ve kurallarına bağlı kalmasını garanti altına alır. Takımı ve organizasyonu scruma adapte eder. Görevi takıma emirler verip yönetmek değil bütünlüğü sağlamak, bir problem olduğunda onu çözmektir. Takımların birbiri arasında ve ürün sahibiyle olan iletişimine yardımcı olan,takımın ve organizasyonun Scrum’a adapte olmasını sağlayarak üretkenliğin artmasına katkıda bulunan kişidir.
* **Scrum Takımı (Scrum Team) :** Scrum takımı, sürekli iletişim halinde olan ve tek bir hedefe ulaşmak için müdahale edilen kişilerden oluşur. Her ne kadar bu ekibi kod yazan kişilerin oluşturduğunu düşünsek de Development Team üyeleri arasında Designers, Analyst ve Testerlar da bulunabilir. 5-9 kişiden oluşur.

2-Toplantılar (Meetings):

1. **Koşu Planlama ( Sprint Planning):** Geniş kapsamlı gereksinim listesinin çıkarılması, başarılı geliştirme için uygun dağıtım gereksinimlerinin belirlenmesi, dağıtımlar için gereksinimlerin eşleştirilmenin yapılması, dağıtımlar için takımların belirlenmesi, risk değerlendirmesi ve risk kontrollerinin belirlenmesi, gözden geçirmeler ve olası gereksinim değişikliklerinin belirlenmesi, geliştirme araçları ve altyapısının onaylanması, dağıtım, geliştirme ve pazarlama maliyetlerinin hesaplanması, yönetimi ve destekleri gözden geçirme ve onaylama gibi işlemler bu planlama toplantısı sırasında gerçekleştirilir.
2. **Koşu Gözden Geçirme (Sprint Review):** Her sprint (koşu) başlangıcında planlama toplantısı yapılır. Toplantının ilk kısmında ürün sahibi ile takım ürün gereksinim listesini gözden geçirir ve gereksinim elemanlarını hedeflerini ve içeriklerini belirler. Toplantını ikinci kısmında takım üyeleri ürün gereksinim listesinin en üstünden başlayarak bu sprintte gerçekleştirilecek gereksinimlerden oluşan sprint gereksinim listesini oluştururlar. Bu scrumın anahtar uygulamasıdır. Takım, ürün sahibi tarafından belirtilen önceliklendirilmiş gereksinimlerden ne kadarını yapacağını belirler ve taahhüt eder.
3. **Günlük Scrum Toplantısı (Daily Scrum Meeting):** Sprint başladıktan sonra takım sürecin başka bir anahtar aktivitesi olan günlük scrum toplantılarını gerçekleştirir. Bu toplantılar her iş gününde genelde belirlenen sabah saatlerinde tüm takımın katılımıyla kısa süreli olmak suretiyle ayakta gerçekleştirilir. Takımın ilerleyişini ve karşılaştıkları engelleri görmek için önemli bir fırsattır. Teker teker tüm ekip üyeleri şu soruların cevaplarını verir;

-Dün ne yaptım?

-Bugün ne yapacağım?

-Önümde olan engeller ve karşılaştığım sorunlar neler?

Scrum Master bu günlük toplantılarda not tutar ve sorun yaşayanlara yardımcı olur veya gereksinim listesine uymayan bir durumda toplantı sonrasında bu üye ile sorunun nasıl çözüleceğini konuşur.

3-Bileşenler/ Araçlar (Artiacts)

1. **Ürün gereksinim Dokümanı (Product Backlog):** Proje boyunca yapılması gereken iş elemanlarının basit bir listesidir. Ürün gereksinim listesi canlı bir dokümandır, geçerli ve kullanışlı olması için devamlı bakım gerektirir. Birçok yeni eleman zamanla eklenebilir, birçok eleman daha küçük parçalara ayrılabilir, bazı gereksinimlerin artık gereksiz veya anlamsız olduğu fark edilip silinebilir. Listedeki dokümanlar her gün değişiklik gösterebilir. Ürün gereksinim listesi genellikle kullanıcı hikayelerinden oluşur ve kullanıcı bakış açısından bakılır.
2. **Koşu Dokümanı (Sprint Backlog):** Sprint backlog, mevcut sprint için product backlogdan iş ve görevleri kapsar. Sprint backlogdaki işlerin amacı, sprint sonunda son ürünün bir parçası olan bir işlevselliği ya da çalışabilir bir parçayı elde etmektir. Sprint backlog sadece takım tarafından değiştirilmektedir.
3. **Sprint Kalan Zaman Grafiği (Burndown Chart):** Bu grafik iterasyon (sprint) boyunca işlerin ne kadarının yapıldığı ile normalde ne kadar yapılması gerektiğinin karşılaştırılabilmesini sağlar. Bir iterasyonun toplam 100 saatten ve 20 günden oluştuğu farz edilirse normal olarak beklenen her gün 5 saatlik iş yapılmasıdır. Takım elemanları her gün ne kadarlık iş gerçekleştirdikleri bilgisini girerler.

Çevik yazılım metodu, kısa vadeli planlar ve küçük parçalar halinde yazılımın geliştirilmesini ön görür. Yazılımın geliştirilmesindeki geri dönüşler (feedback) ve değişikliklere uyum sağlamak son derece önemlidir. Her yapılan yineleme yazılımı hedeflenen adıma bir adım daha yakınlaştırır. İstenilen sonuca ulaşmak adına birden çok yineleme gereklidir. İlgili dokümanlar gereken durumlarda hazırlanır. Çevik süreç değişimi baz alır ve onunla yaşamayı kolaylaştırmak için yeni yazılım metotları sunar. Amaç dokümantasyonun en az insan çabası ile yeterli seviyede üretilmesidir. Scrum, uzmanlık gerektiren ve maliyeti yüksek bir modeldir. Ancak kısa sürmesi, kolay uygulanması ve başarı garantisinin yüksek olması nedeniyle günümüzde kullanımı oldukça artmaktadır. Google, Microsoft, IBM ve Yahoo gibi büyük şirketler tarafından tercih edilmektedir.

**Referanslar:**

<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://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://docplayer.biz.tr/56373865-Yaz-muh-ders-notlari-1.html>

<https://medium.com/@senadtp/yazilim-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-3475534dddd9>

<https://medium.com/@fatihazir/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BC-modelleri-ve-scrum-a17aef990c4c>

<https://medium.com/@umutakter/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-yaz%C4%B1l%C4%B1m%C4%B1n-%C3%BCretim-ve-kullan%C4%B1m%C4%B1ndaki-t%C3%BCm-a%C5%9Famalar%C4%B1-kapsar-e2b0f0690588>

<https://i0.wp.com/furkanalniak.com/wp-content/uploads/2017/05/vModeli.png>

<https://9lib.net/document/6qm8554z-xp-ve-uml-nin-yazilim-gelistirme-suerecine-etkileri.html>

<https://fikirjeneratoru.com/yazilim-proje-yonetimi-yontemleri/>

<https://batuhanakpunar.medium.com/b%C3%B6l%C3%BCm-1-scrum-nedir-neden-scrum-b00b48f25e54#:~:text=Peki%20neden%20Scrum%3F&text=Scrum%2C%20tak%C4%B1mlar%C4%B1%20deneyimler%20yoluyla%20%C3%B6%C4%9Frenmeye,%E2%80%9Cadapte%20olabilme%E2%80%9D%20yetene%C4%9Fi%20kazand%C4%B1r%C4%B1r>.

<https://enprobilisim.com/yazilim-gelistirme-sureci-modelleri-sdmp/>

<https://medium.com/@omerharuncetin/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BC-modelleri-543c7879a742>

<https://www.canva.com/design/play?category=tADWs0Cq50o&type=TACQ-lCLuV8&uid=ec398745-ccc1-4d68-ae81-d1f28116b5c0&_branch_match_id=975121252272796084&utm_source=landing-page&utm_campaign=cycle-diagram&utm_medium=graphs-pages&_branch_referrer=H4sIAAAAAAAAAw3HUQqAIAwA0BMtW6lZEJ1lbZZChqQF3b7e3wu15jIpxXQ%2B1CSv9otyKMAvHx4k0v%2B03FFmz%2F3oBm2AmRG0WAfkHYLg1jlEuxpuPw97QUlOAAAA>

Medium Hesabım: <https://medium.com/@yaseminbahsi> YASEMİN BAHŞİ

Linked In Hesabım: <https://www.linkedin.com/in/yasemin-bah%C5%9Fi-5ba897222/> 210601050

GitHub Hesabım : <https://github.com/ysmnbhs>