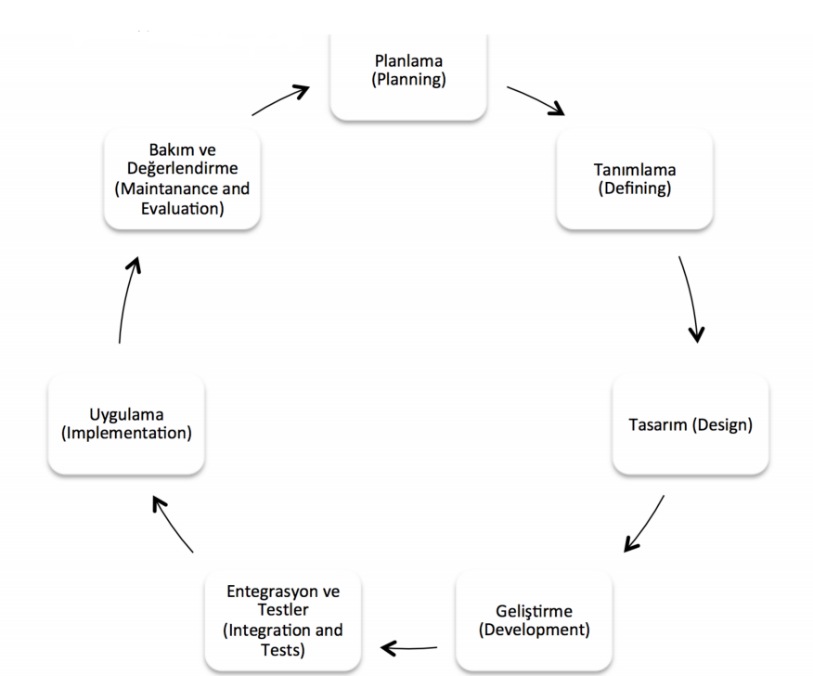
**Yazılım Yaşam Döngüleri**

**Halil Kocaman**

**200601084**

**Yazılım yaşam döngüsü denince ilk akla gelen yazılım dünyası olsa da aslında hayatın her alanında bu döngüyü kullanıyoruz. SDLC (Sofware Devolopment Life Cycle) kavramı olarak bilinen bu döngü Türkçeye Yazılım Yaşam Döngüsü olarak çevrilmiştir. Örneğin yemek yapmaya başlamadan önce ne yapılacağına karar verip buna uygun malzeme listesini çıkarıp aşama aşama yemeğimizi hazırladıktan sonra bir sonraki yemek için de aynı şeyleri yapmamız gerekir. Tıpkı bu örnekteki gibi Yazılım Yaşam Döngülerinin de bir yaşam süreci vardır. Ancak bu sürecin doğru analiz edilmesi durumunda ortaya başarılı bir yazılım ürünü ortaya çıkar. Süreci oluşturan birkaç temel aşama vardır. Bu aşamalar; Planlama, Tanımlama, Tasarım, Geliştirme, Entegrasyon ve testler, Uygulama, Bakım ve değerlendirme olarak sınıflandırılabilir. Yazılım ürünlerinin analiz ve planlanmasından başlayarak hayata geçirilmesi boyunca süren bu döngüye Yazılım Yaşam Döngüsü diyebiliriz. Aşağıdaki şekilde bu döngü basitçe gösterilmiştir.**

****

**Planlama : Yazılım ürününü bir proje olarak kabul edersek bu aşama projenin ilk ve en önemli aşamasıdır. Projeden beklentileri , İhtiyaçların ve sorunların değerlendirilip yol haritasına dönüştürüldüğü kısımdır. Diğer adımların başlangıç aşamasıdır.Proje ekibinin görev dağılımı da bu aşamada yapılır.Örneğin bir baraj yapılacak, baraj için gerekli alanın bulunması , zemin etütlerinin yapılması , çevreye vereceği etkilerin hesaplanması , kapasitesinin bütçesinin getiri ve götürülerinin hesaplanması planlama aşamasındadır.Kısacası İşin projelendirildiği, fikrin ortaya çıkarıldığı ve fikrin tartışıldığı aşamadır.**

**Tanımlama: Bu aşama analiz aşaması olarak da düşünülebilir. Problemin tanımlandığı aşamadır. Tanım aşamasında projede nelerin istenildiği ile ilgili analiz çalışmaları yapılır. Yazılım projesinin ne kadar süreceği, ne gibi risklerinin olacağı belirlenir.**

**Tasarım: Yazılım projesinin nasıl devam edeceğini belirleyen aşamadır.** **Yazılımın veya sistemin tasarımları yapılır. Planlama ve tanımlaya göre bir proje çizilir. Bir sonraki adım olan geliştirme aşamasına bütün kararlar verilmiş olarak geçilmesi beklenir. Örneğin yazılımın Arayüzü belirlenir, fonksiyonları tartışılır , ekipler oluşturulur,hedef kitlesi belirlenir, geliştirme aşamasına herhangi bir soru veya karar bırakılmaz.**

**Geliştirme: Bu aşama, yazılım projelerinin kodlama aşaması olarak bilinir. Bir önceki tasarım aşamasında planlanmış ortama uygun olarak alınan kararlar doğrultusunda projenin gerçekleştirilmesine başlanır. Projenin en çok zaman alan aşamasıdır.**

**Entegrasyon ve Testler : Yazılımın artık gerçek hayata geçtiği aşamadır. Sistem veya yazılım Test amacıyla piyasaya sürülür. Günümüzde pek çok yerde karşılaştığımız Beta versiyonlar bu aşamanın birer ürünüdür. Sistemin veya yazılımın Farklı donanımlara uyumunun sağlanması için çalışmaların yapıldığı aşamadır. Ayrıca sistemi veya yazılımı kullanacak olan kişi veya kurumların programla ilgili eğitimlerin verilmesi de yine bu aşamada gerçekleşir.**

**Uygulama:** **Projenin tam anlamıyla hayata geçirildiği aşamadır. Yazılım veya sistem kullanıldıkça yeni problemler ortaya çıkar. Bu problemlere karşılık yeni fikirler, yeni çözümler, yeni ihtiyaçlar ortaya çıkar ve bunlar geribildirim olarak proje ekibine döner. Bu geribildirimler sonucunda bakım aşamasına geçilir.**

**Bakım: Yazılım veya sistem ürününün hayata geçirilmesinden sonra güncelleme ve bakım olaylarının yapılmasıdır. Bu aşamada** **yazılımın veya sistemin bakımı yapılırken farklı aşamalardan geçirilir. Yeni tanımlar ortaya çıkar, yeni tasarımlar yapılır ve süreç bu şekilde hiç durmadan devam eden döngü haline gelir. Bu döngüye de SDLC (Sofware Devolopment Life Cycle) yani Yazılım Yaşam Döngüsü adı verilir.**

**Yazılım Yaşam Döngü Modelleri**

**Yazılım yaşam döngüsünde birden fazla model kullanılır. Bu modellerinin birden fazla olmasının nedeni yazılım projesinin büyüklüğü, yazılım projesinin kimler için kullanılacağı gibi durumlardır. Bu modeller kısaca Barok ve Gelişigüzel Modeli, Şelale (Waterfall) , V Süreç Modeli, Spiral Model, Artırımsal Geliştirme Modeli, Kodla ve Düzelt (Code & Fix), Evrimsel Geliştirme Modeli, Çevik Modeller (XP Ve Scrum Modeli) olarak Sıralanabilir.**

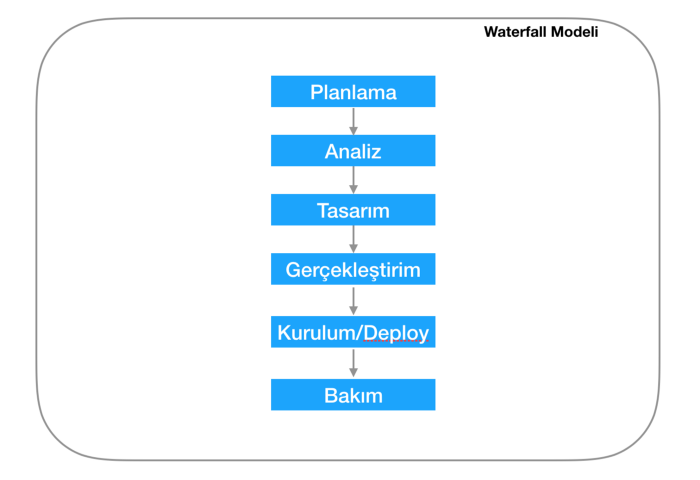
**•Barok ve Gelişigüzel Yaşam Döngü Modelleri**

**Gelişigüzel Model: Çok eski bir modeldir. Yöntem veya model yoktur. Kişiye özel yapılır ve takip edilmesi oldukça zordur. 1960’larda kullanılmıştır.**

**Barok Modeli: Çok eski bir modeldir. Yaşam döngü aşamaları temel olarak doğrusal bir şekilde ilerler. Döngü yoktur. Bu model belgelemeyi ayrı bir süreç olarak ele alır.1970’lerde kullanılmıştır.**

**Bu iki model günümüzde kullanılmamaktadır. Ancak tarihin ilk yazılım modelleri olarak kabul edilmektedir.**

**•Şelale (Waterfall) Yaşam Döngü Modeli**

****

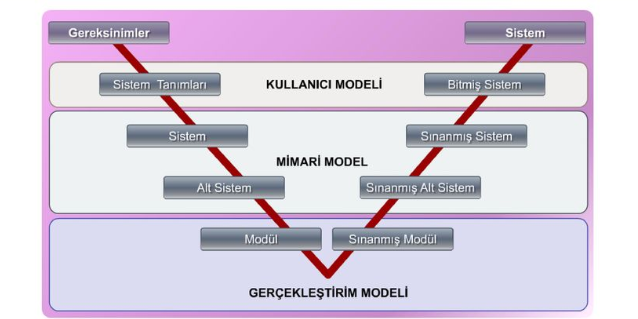
**Bu model güncel yaşam döngü modellerinin temelidir. Modelde uygulanan adımlar en az bir kere tekrar edilir. Şelale modeli iyi planlanmış ve kısa zamanda bitirilebilecek projeler için kullanılır. Barok modeli ile kıyaslanacak olduğunda belgeleme işlemi sürecin içindedir ve bir aşama bitmeden diğerine geçilmez. Bu modelin dezavantajları olarak:**

**•Uzun süreçte hazırlanan projeler için uygun değildir.**

**•Kullanıcı değerlendirmeleri sürece dahil olmadığı için sorunlar çok daha uzun sürede çözüme kavuşur ve bu sebeple maliyet artar.**

**•Tüm aşamalar tamamlandıktan sonra kontroller yapıldığı için herhangi bir aşamada hata çıkması durumunda modelin en başına dönmek zorunda kalınabilir.**

**• V Süreç Modeli**

****

**Waterfall modelinin geliştirilmiş halidir. Şelale modeline ek olarak Doğrula ve Onayla aşamaları modele dahil edilmiştir. Sol taraf üretim sağ taraf ise test yani sınanma aşamalarıdır. Örneğin Sınanmış Sistem testinden geçiyorsanız Sistem ve Alt sistem aşamalarını başarılı bir şekilde tamamlamış olduğunuzu gösterir. Bu döngü temelde 3 modelden oluşmaktadır. Bunlar Kullanıcı Modeli, Mimari Model, Gerçekleştirim Modeli olarak yukarıda şekilde sınıflandırılmıştır.**

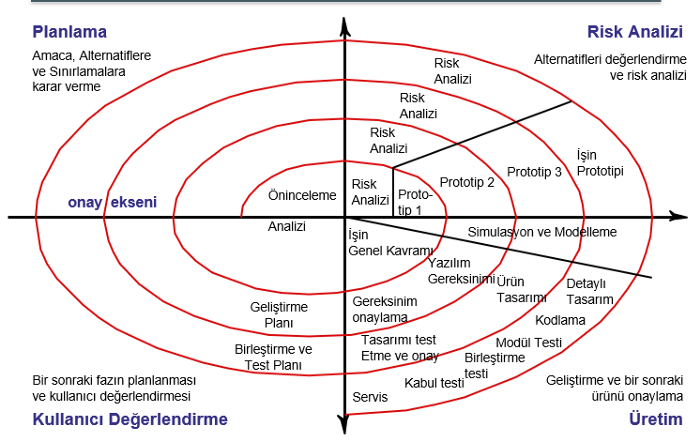
**Kullanıcı Modeli: Döngünün bu kısmında kullanıcıların istekleri ve tanımları kullanılarak proje kullanıcıya bitmiş haliyle teslim edilir.**

**Mimari Model: Projenin tasarım ve tasarımların sınanması kısımlarından oluşur.**

**Gerçekleştirim Modeli : Projenin kodlanması ve kodların sınanması kısımlarından oluşur.**

**V-Modelin dezavantajları ise ; Aşamalar arasında tekrar bulunmaz, risk çözümleme için ayrılan bölüm yoktur.**

**•Spiral Model**

****

**Spiral Modeli bir kar topu gibi giderek büyüyen bir geliştirme sistemi olarak görebilirsiniz. Spiral modeli diğer modellerden ayıran en belirgin fark ise Risk Analizinin ön planda olmasıdır. Ayrıca her aşamada prototip oluşturarak projenin kolay takibini, hata tespitlerini ve kullanıcı ile test edilme imkanını sağlar. Temel olarak 4 aşamadan oluşmaktadır. Bunlar:**

**Planlama: Her aşamada ara ürün için bir planlama yapılır.**

**Risk Analizi: Risklerin araştırılması , tespit edilmesi ve çözümlenmesini sağlar.**

**Kullanıcı Değerlendirme: Her aşama sonunda oluşturulan ara ürün için kullanıcı geribildirimleri alınarak ürüne müdahale edilir ve bir sonraki aşamaya geçilir.**

**Üretim: Ara ürünlerin ve Bitmiş ürünlerin üretilme aşamasıdır.**

**Spiral Modelin dezavantajları olarak:**

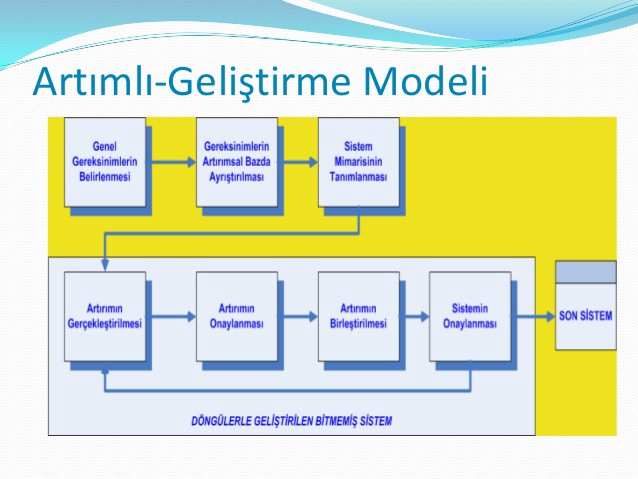
**•Düşük bütçeli projeler için uygun olmaması**

**•Karmaşık bir içeriğe sahip olması**

**•Dokümantasyonun fazla olması**

**•Modelin uygulama sürecinin diğer modellere göre çok uzun sürmesi gösterilebilir.**

**•Artımlı Geliştirme Modeli**

****

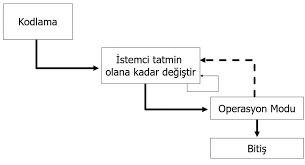
**Artırımlı Geliştirme Modelinde proje parçalara ayrılır. Kullanıcıların önceliklerine göre ayrılan parçalar sıralanır. Sıralanan bu parçalar tek tek ara ürün olarak ortaya çıkar. Ortaya çıkan bu ürünler kullanıcılar tarafından kullanılır. Ara ürünler her seferinde üzerine katarak ortaya çıkar. Bu modelde üretim ile geliştirme aynı anda gerçekleşir. Bu nedenle uzun sürebilecek ve işlev eksikliğine neden olabilecek projeler için uygundur. Proje bir yandan devam ederken diğer yandan kullanılır. Bu modelde ürünlerin başarısızlık olasılığı düşüktür. Ara ürünler yazılımın geliştirilmesinde önemli rol oynarlar.**

**Artırımlı Geliştirme Modelinin dezavantajları ise :**

**•Ara ürünler tamamlanana kadar herhangi bir değişiklik yapılamaz.**

**•Parçaları tamamlamak için projenin detaylı bir biçimde tamamlanması gereklidir.**

**•Kodla ve Düzelt Yaşam Döngü Modeli**

****

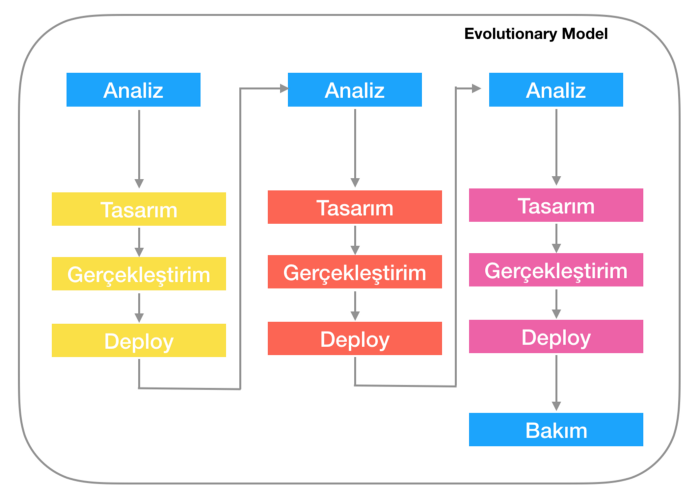
**Küçük ve basit projeler için kullanılır. Direkt olarak ürün odaklı çalışılır. Operasyon aşamasında istenilen ürün çıkana dek kontroller ve düzeltmeler devam eder. Ürünü hazırlar ve kullanıcıya sunarsınız. Düşük maliyetli bir modeldir.**

**Dezavantajları ise:**

**•Büyük projelerde kullanılamaz.**

**•Bakımları nadir fakat zordur.**

**•Evrimsel Geliştirme Modeli**

****

**İlk tam ölçekli yazılım modelidir.** **Her aşamada üretilen ürünler, üretildikleri alan​​ için tam işlevsellikte olmalıdır.​​​ Modelin başarısı ilk evrimin başarısına bağımlıdır. Bu modelin en önemli farkı kullanıcıyı yazılım geliştirme safhasına dahil etmektir. Bu sayede kullanıcıların kendi gereksinimlerini daha iyi anlamasını sağlar. Sürekli analiz yapıldığı için erken aşamalardaki geliştirme risklerini azaltır.**

**Dezavantajları:**

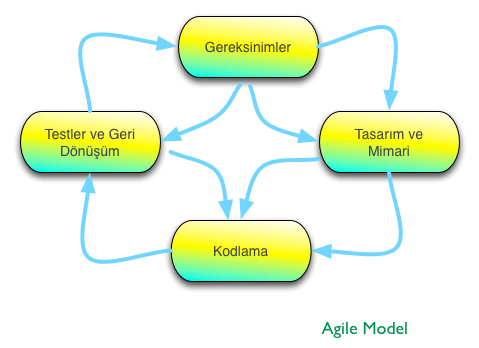
**•Sistemde sürekli olarak değişim yapmak yazılımın yapısına zarar verir.**

**•Düzenli olarak verilebilecek ürün yoktur.**

**•Bakımları oldukça zordur.**

**•Yazılım gereksinimlerini yenilemek gerekebilir.**

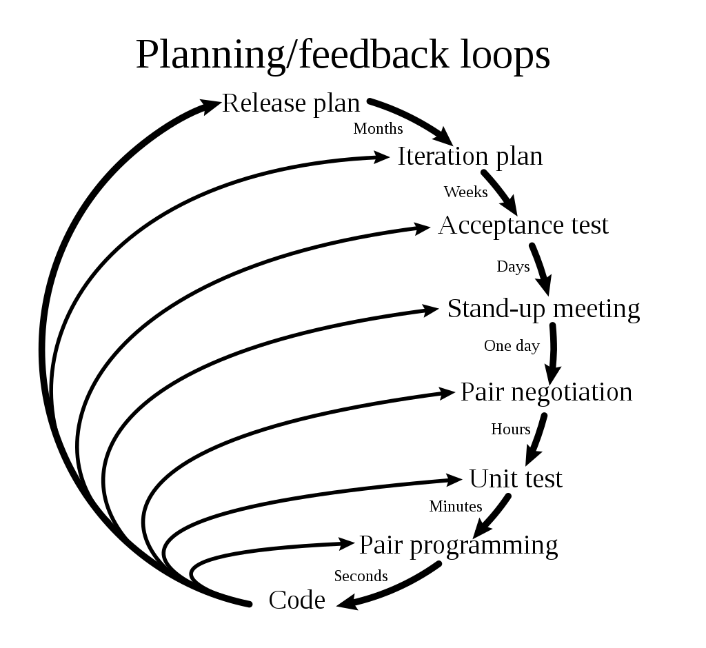
**•Çevik Modeller(Agile Model)**

****

**Bu modellerde gereksinimler ve çözümler farklı grupların ortak çalışmasıyla meydana gelir.Çevik modellerin ortaya çıkış amacı denetim ve adaptasyonu, takım çalışmasını, kendinden örgütlenmeyi ve kaliteli yazılımların geliştirilmesini artırmaktır.​​**

**Başlıca Çevik Yazılım Geliştirme Modelleri :**

**•Extreme Programming (XP)**

****

**Extreme Programming (XP) Yazılım Geliştirme Modelinde geribildirim oldukça ön plandadır. Geri bildirimler projenin gidişatını olumlu etkiler. Kullanıcılar ile üretim ekibi sürekli iletişim halindedir. İletişimi en üst düzeyde tutmaya çalışılır. Takım çalışması ön plandadır. Sorunlara çözüm üretilirken basitlik tercih edilir.** **Basit çözümlerin kavranması ve anlatılması daha kolaydır. Uzun dokümantasyonlardan uzak durulur.**

**Dezavantajları:**

**•Takım çalışmasının fazla baskıya dönüşmesi.**

**•Değişen ihtiyaçlar dolayısıyla aşırı çalışma.**

**•Ara ürünlerin değiştirilme şansının olmaması.**

**•Scrum**

****

**Bu model günümüzün en çok tercih edilen yazılım geliştirme modelidir. Ürün sahibi direkt olarak geliştirme sürecine dahildir. 5-9 kişilik Scrum yazılım ekipleri ve ekibin Scrum yöneticisi bulunur. Bu kişiler yazılım geliştirme esnasında sürekli birbirleriyle iletişim halindedirler. Proje sprint denilen küçük kısımlara ayrılır. Scrum içerisindeki tüm aktiviteler sprint içerisinde gerçekleşir.1–2 haftalık süreçlerdir. Scrumun en önemli farklılıklarından biri olan Toplantılar 4 parçaya ayrılır.**

**1) Sprint Planning; Yazılım ekibinin kendi içindeki görev dağılımı yapılır. Bu toplantıya ürün sahibi, yazılım ekibi ve scrum yöneticisi katılır. Yapılacak işler planlanır ve 1–2 haftalık sprintler oluşturulur.**

**2) Daily Scrum; Her gün belirli saatlerde ayak üstü yapılan 15 dakikalık toplantılardır. Bu toplantı ertesi gün yapılacakları planlamak üzere yapılır. Scrum yöneticisiyle ne yaptık, ne yapacağız ve ne sorun çıktı gibi sorular ele alınır ve gerekilirse müdahale edilir. Yazılım ekibinin büyük çoğunluğu olduğu sürece toplantıya her koşulda devam edilir.**

**3) Sprint Review; Her sprintin sonunda yapılan toplantıdır. Yapılan iş değerlendirilir, ortaya çıkan ürün gözden geçirilir. Bu toplantının amacı yazılımın ürün sahibinin isteklerine uygun olarak geliştirildiğinden emin olmaktır. Eğer bir sorun varsa düzeltilir.**

**4) Sprint Retrospective; Sprint boyunca yapılan işlerin kalitesinin, doğruların ve yanlışların değerlendirildiği toplantıdır. Bu toplantı scrum takımının kendini geliştirebilmesi için önemlidir. Bu aşamadan sonra bir sonraki sprint planning toplantısı yapılarak bu döngü devam eder.**

**Scrum Neden Günümüzde Popüler Bir Model ?**

**Karmaşık yazılım projelerinin yönetilmesi için kullanılır. Projeyi parçalara ayıran; tekrara dayalı bir yöntem izler. Düzenli geri bildirim ve planlamalarla hedefe ulaşmayı sağlar. Tamamen ihtiyaca yönelik ve esnek bir yapısı vardır. Müşteri ihtiyacına göre şekillendiği için müşterinin geri bildirimine göre yapılanmayı sağlar. Bu nedenle günümüzde en çok tercih edilen modellerden birisidir. Bu modelin avantajlarını maddeler halinde sıralayacak olursak;**

**•Vakit ve maliyetten oldukça fazla tasarruf edilmesi,**

**•Yeniliklere kolay uyum sağlayabilmesi,**

**•Karmaşık projeler için uygun olması,**

**•Hata tespit ve kontrolünün kolay ve hızlı olması,**

**•Ekip içi iletişimin yüksek düzeyde olması,**

**•Geribildirim sayesinde hataların çabuk fark edilmesi,**

**•Yinelemeli bir model olması gibi birçok olumlu faktör Scrum Modelinin popüler olmasını sağlamaktadır.**

**Peki Hangi Projede Hangi Modeli Kullanmalıyız ?**

**•Şelale modelinin kullanımı kolay, iş bölümü ve iş planlaması detaylıdır. Eskiden yaygın olarak kullanılan bu model kullanıcı ile iletişimin az olması, büyük projelerde yetersiz kalması, değişimlere uygun olmaması gibi sorunlar yüzünden günümüzde fazla tercih edilmemektedir. Bu nedenle küçük ve özellikleri iyi tanımlanmış projelerde şelale modeli kullanılabilir.**

**•V Modeli şelale modelinin gelişmiş versiyonudur. Kullanımı ve takibi kolay olması, onaylama ve doğrulama işlemlerinin planın erken aşamasında uygulanabilir olması avantajlıdır ancak fazla tekrarlamalara imkân tanımaması, risk çözümleme basamaklarının bulunmaması gibi sorunlar bulundurur. Bu nedenle belirsizliklerin az olduğu, iş tanımlarının belirgin olduğu bilgi teknolojileri projeleri için V Modeli uygun bir modeldir.**

**•Evrimsel Geliştirme Metodunda kullanıcılar gereksinimleri daha iyi anlarlar, hatalar azalır ancak sürecin görünürlüğü azdır ve bakımı zordur. Büyük müşteri kitlesine hitap edecek projelerde (banka, kargo, postane vb.) evrimsel geliştirme metodu kullanılabilir.**

**•Artımlı Geliştirme modelinde her teslimde müşterinin geri dönütü alındığı için işlevsellik erken ortaya çıkar, proje daha güvenli bir hal alır, gereksinimler müşteri ile belirlenir, özellikler daha fazla test edilmiş olur ancak her artımı tanımlayabilmek için sistemin detaylı bir şekilde tanımlanması lazımdır, deneyim gerektirir ve artımlar kendi içlerinde tekrarlanamaz. Büyük, maliyetli ve uzun süren projelerde artırımlı geliştirme modeli uygundur.**

**•Çevik modellerde müşterilerle ve ekip içi iletişim yüksek olduğu için hatalar minimuma indirilir, adaptasyonu kolaydır, değişime açıktır, takım önemli bir konumdadır. Ancak dokümantasyon çok detaylı değildir. İhtiyaçlar sürekli değiştiğinden çalışma saatleri fazla olabilir. Günümüzde çokça kullanılan bir modeldir. Bu nedenle orta ve küçük büyüklükte, uzun sürmeyen projelerde çevik modeller uygundur.**

**•Kodla ve düzelt modeli küçük programlarda kullanılır, bakım safhası oldukça zordur, emeklilik safhası bulunur; büyük projeler için kullanılamaz çok maliyetlidir, genelde tek bir kişinin isteğine göre program en son halini alana kadar sadece kodlama yapılır ve teslim edilir, dokümantasyon gibi olaylar yoktur o yüzden kullanımı sadece kişiye özel ile sınırlıdır.**

**Modellerin Karşılaştırılması**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **​​​​ Model/Özellik** | **Artımlı Süreç** | **Birleşik Modelleme** | **Çevik Modeller** | **Spiral Model** | **Kodla ve Düzelt** | **Waterfall Modeli** | **V Modeli** | **Evrimsel Geliştirme** |
| **1** | **Gereksinin Belirleme** | **Belirli Sıklıkla** | **Başlangıç** | **Belirli Sıklıkla** | **Belirli Sıklıkla** | **Başlangıç** | **Başlangıç** | **Başlangıç** | **Başlangıç** |
| **2** | **Maliyet** | **Düşük** | **Pahalı** | **Çok Yüksek** | **Maliyetli** | **Düşük** | **Maliyetli** | **Maliyetli** | **Düşük** |
| **3** | **Başarı Garantisi** | **Yüksek** | **Yüksek** | **Çok Yüksek** | **Yüksek** | **Düşük** | **Düşük** | **Orta** | **-** |
| **4** | **Uzmanlık Gerekliliği** | **Orta** | **Orta** | **Çok Yüksek** | **Yüksek** | **Düşük** | **Orta** | **Orta** | **-** |
| **5** | **Fazların Örtüşmesi** | **Evet** | **Evet** | **Evet** | **Hayır** | **Hayır** | **Hayır** | **Hayır** | **Hayır** |
| **6** | **Maliyet Kontrolü** | **Hayır** | **Hayır** | **Evet** | **Evet** | **Hayır** | **Evet** | **Evet** | **Evet** |
| **7** | **Basitlik** | **Orta** | **Karmaşık** | **Karmaşık** | **Karmaşık** | **Basit** | **Basit** | **Orta** | **Karmaşık** |
| **8** | **Risk Duyarlılığı** | **Düşük** | **Düşük** | **Azaltılmış** | **Düşük** | **Yüksek** | **Yüksek** | **Yüksek Değil** | **Yüksek** |
| **9** | **Esneklik** | **Çok Esnek** | **Çok esnek** | **Çok Esnek** | **Çok Esnek** | **Katı** | **Katı** | **Düşük​​** | **Düşük** |
| **10** | **Bakım** | **Evet** | **Evet** | **Evet** | **Evet** | **Düşük** | **Düşük Bakımlı** | **Düşük Bakımlı** | **Düşük** |
| **11** | **Değişiklik Yapma** | **Kolay** | **Evet** | **Zor** | **Kolay** | **Kolay** | **Zor** | **Zor** | **Zor** |
| **12** | **Yeniden Kullanılabilirlik** | **Mümkün** | **Mümkün** | **Evet** | **Mümkün** | **Düşük** | **Düşük İhtimal** | **Düşük İhtimal** | **Düşük** |
|
| **13** | **Dökümantasyon ve Eğitim Gerekliliği** | **Evet Ama Çok Değil** | **Evet** | **Evet** | **Evet Ama Çok Değil** | **Evet** | **Zorunluluk** | **Evet** | **Evet** |
| **14** | **Zaman** | **Uzun** | **Uzun** | **Kısa** | **Uzun** | **Çok Uzun** | **Çok uzun** | **Uzun** | **Uzun** |
| **15** | **Uygulama** | **Kolay** | **Kolay** | **Kolay** | **Karmaşık** | **Kolay** | **Kolay** | **Kolay** | **Kolay** |

**KAYNAKÇA**

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

<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>

<https://medium.com/architectural-patterns/yaz%C4%B1l%C4%B1m-geli%C5%9Ftirme-modelleri-62915545c51e>

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

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

<https://medium.com/@secilcor/scrum-nedi%CC%87r-6a4326951dd8>

<https://tr.wikipedia.org/wiki/Yaz%C4%B1l%C4%B1m_yay%C4%B1m%C4%B1_ya%C5%9Fam_d%C3%B6ng%C3%BCs%C3%BC>

[*https://furkanalniak.com/yazilim-muhendisligi-yazilim-surec-modelleri/*](https://furkanalniak.com/yazilim-muhendisligi-yazilim-surec-modelleri/)

[*https://www.codex.com.tr/yazilim-gelistirme-modelleri*](https://www.codex.com.tr/yazilim-gelistirme-modelleri)