## Yazılım Geliştirme Modelleri

Yazılım yaşam döngüsü modelleri yazılım geliştirme sürecinde kullanılan yöntemlerin genel adıldır. Bu modeller yazılımın geliştirilmesinden dağıtılması hatta bakımına kadar ki süreçleri kapsayıp yazılım geliştirme sürecindeki verimliliğini ve projenin başarı şansını arttırmak için kullanılır. Çevik yazılım geliştirme modelinin dışında kalan bazı yaşam döngüsü modelleri şunlardır;

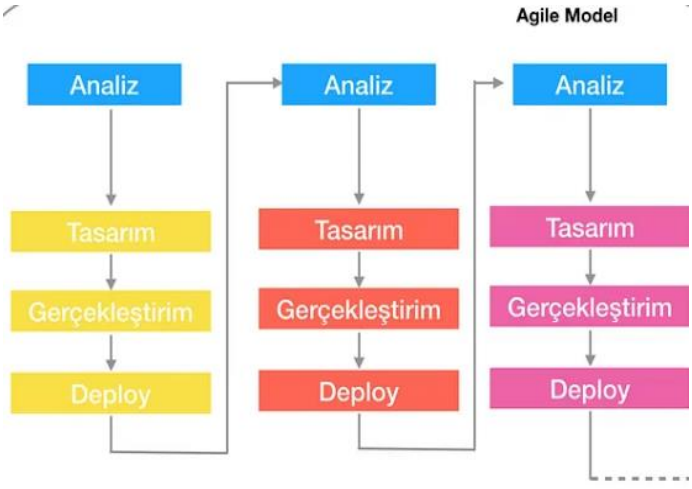
Su dalgası modeli; bu modelde su dalgası gibi bir şekilde ilerleyen yazılım geliştirme süreci her aşama tamamlandığında bir sonraki aşama başlar. Her aşama tamamlandığında geri dönüş yapılamaz. Geleneksel bir modeldir. Bunun sebebi geçmişte en popüler döngü modeli olmasıdır. Yazılımların hazırlanması ve müşteriye sunulması uzun bir süreç olduğu için pek tercih edilmemektedir. Kullanıcı bu sürece dahil edilmediğinden oluşabilecek bir sorunu çok geç fark edebilir. Bu da yazılım maliyetinin artmasına neden olur.

V modeli; Yazılım geliştirme sürecini test merkezli olarak ele alır. Aşamalar teker teker tamamlandığı her zaman test edilir ve geri dönüş yapılır. Bu model yazılım kalitesini artırmak için kullanılır.

Spiral (Helezonik) Model; Yazılım geliştirme sürecini bir spiral şeklinde ele alır. Risk analizinin ön planda olması ve prototip oluşturması ön plandadır. Risk analizi sayesinde hatalar erkenden giderme imkânı tanıyabilir. Kullanıcı her aşamada yazılım projesinin bir parçası halindedir. 4 temel aşamaya sahiptir bunlar; Planlama, risk analizi, üretim ve kullanıcı değerlendirmesidir.

Bahsettiğimiz modellerin dışında birçok model daha bulunmaktadır. Ancak günümüzde en çok tercih edilen Çevik (Agile) Yazılım Geliştirme metotlarıdır.

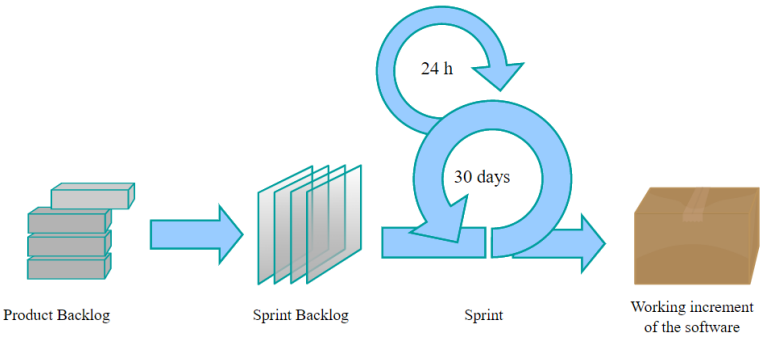
## Çevik yazılım geliştirme




"Çevik", dünyada yazılım süreçlerini daha esnek ve güçlü kılmak için kullanılan aynı zamanda yazılım süreçlerini de kısaltan kavramsal bir yazılım geliştirme metodolojisidir. Bu metodolojide projenin ölçeği ne olursa olsun, proje küçük yinelemelere ayrılır ve her yineleme başlı başına bir proje gibi ele alınarak geliştirilir. Her yinelemenin (iteration) sonunda da proje ekibi tarafından müşteriye, projenin ne kadarının gerçekleştirildiğine dair bilgi verilir.

## Scrum

Kelime kökeni olarak Rugby sporundaki bir hücum taktiğinin adıdır. Bu taktikte top tüm oyucularla birlikte karşı sahaya taşınarak atak yapmaktır. Scrum Jeff Sutjerland ve Ken Schawaber tarafından 1990'kların ortalarında geliştirilen bir proje yönetim metodudur. Aslında sadece yazılım geliştirmeye değil, her yere uygulanılabilir. Karmaşık yazılım işlerini küçük birimlere bölerek geliştirmeyi hedefler. Bu metot karmaşık ortamlarda adım adım yazılım geliştiren ekipler için uygundur.




*Scrum süreci*

Scrum yazılım geliştirme projelerinde kullanılan bir yazılım geliştirme modelidir. Angile Manifesto'nun bir parçası olarak ortaya çıkan bir yöntem olan Scrum'ın temel amacı ekiplerin daha hızlı ve etkili bir şekilde çalışmasını sağlamaktır. Scrum'ın ana amacı, felsefesi; ölçeklenebilir, tekrarlanabilir ve öngörülebilir bir süreç yaratmaktır. Bunların yanı sıra tüm yazılım geliştirme aşamaların planlı bir şekilde yürütülmesini sağlamanın zor olacağı düşüncesini ve karmaşık düzeni azaltmaya çalışmak Scrum ile mümkün olmaktadır. Bu modelde projenin başarılı bir şekilde ilerleyebilmesi için şeffaflık en temel özelliklerinden biridir. Şeffaflıktan bahsedecek olursan projenin ilerlemesinin ve çıkan sorunların bir kütüphanede kod vs. olarak tutulmasını sağlamak ve herkes tarafından izlenebilir olmasını sağlamak her zaman artı bir değer kazandırır. Yazılımcıların asli görevlerinden olan araştırma ve geliştirme çalışmalarını sistem üzerine uygulamalara dökülmesini sağlamaktır. Bu görevde gerekli denetlemelerin de zamanında yapılmasına özen göstermek gereklidir. Ürünler ile ilgi olarak parça ya da fonksiyon sistemlerin düzenli bir şeklide kontrollerini sağlamak gereklidir. Bu kontroller sayesinde sistemler üzerinde meydana gelecek olan hataların, hata payları minimize edilecektir. Yani çok daha iyi bir yazılım elde etmek mümkün olacaktır. Bir başka süreç olan uyarlama süreci yani ar-ge faaliyetlerinin yazılım sisteminin üzerindeki en temel sonucu bize verir.

Scrum üç temel bileşene sahiptir. Bu bileşenlere genel olarak "Pig Roller" yani Scrum sürecinde dahil olan, projede asıl işi yapan kişilerdir bunlar;

1. Product Owner; kısacası ürün sahibi, proje hedeflerini belirleyen ve ekipden gerekli olan özellikleri isteyen kişidir. Projenin özelliklerini tanımlayarak geliştirme takımı ve müşteri arasındaki temel iletişimi sağlar.

2. Scrum Master; Temelde ekip üyelerinin sorunlarını çözmek ve ekip uyumunu sağlamak süreci yönetmekle sorumludur. Ayrıca Scrum Master Scrum kurallarını, pratiklerini ve teorilerini iyi bilir. Bahsi geçen bu kuralları sahip olduğu bilgiyi takıma ve takımın uygulamasından sorumlu kişidir. Takımı rahatsız eden, verimli çalışmalarını engelleyen sorunları ortadan kaldırmakla görevlidir.
3. Development Team; Türkçe çevirisi gibi geliştirme ekibidir. Ürünün geliştirilmesi için gereken kodlama, test etme ve diğer görevleri yerine getiren kişilerden oluşur. Bir Sprint'e alınan bütün işleri tamamlayacak özelliklere sahip kişilerdir. Kendi kendilerini yönetirler. İşi alır ve geliştirirler, Projenin geliştirilmesi ile ilgili sorumluluk geliştirme takımına aittir.

Scrumun genel olarak birkaç prensip içerir bunlar bazı kaynaklarda toplantılar olarak da geçmektedir bu prensipler şunlardır;

Sprint Planning; Scrum sprint adı verilen kısa zaman dilimleri içinde çalışır. Sprintler genellikle iki ila dört hafta arasında sürer. Product backlog ile belirtilen gereksinimler bu toplantılarla Development Team tarafından küçük görevlere ayrılır. Takımdaki herkes kendi becerilerine ve hızlarına göre bu görevlerden edinir. Toplantıya Product Owner, Dev team ve Scrum master katılım gösterir.

Daily Scrum; Ekip üyelerinin günlük olarak yaptığı kısa toplantıdır. Bu toplantıda ekip üyeleri, sorunları ve ilerlemelerini paylaşır. Bu toplantıda gelecek 24 saat planlanır. Karşılıklı olarak ekip üyeleri etkileşime geçerek dün ne yaptıklarını bugün ne yapacaklarını ya da işini engelleyecek herhangi bir sorunun olup olmadığını sorularına cevap verir. Eğer herhangi bir sorun var ise Scrum Master bu sorunla, problemle ilgilenerek ortadan kaldırmaya çalışır. Bu toplantı her zaman yapılmalıdır.

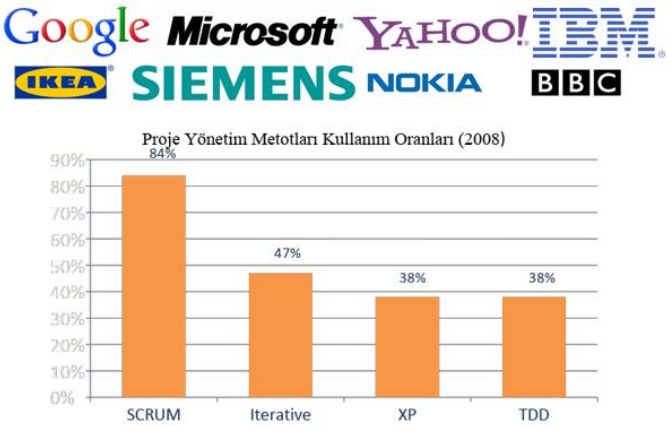
Sprint Review; Sprint'in sonunda gerçekleştirilen toplantıdır. Bu toplantıya ürün sahibi dahil olarak ekip üyelerinin yaptıklarını değerlendirir. Bu toplantıda yazılımın ürün sahibinin gereksinimlerine ve isteklerine uygun olarak geliştirildiğinden emin olmaktır. Tespit edilen hata var iste düzeltilir.

Sprint Retrospective; Ekip üyeleri tarafından bir önceki sptrinteki çalışmalarını değerlendirildiği toplantılardır. Bu toplantıda ekip üyeleri hangi konularda daha iyi olabileceklerini ve gelişimlerine neyin daha çok etkileyeceğini tartışırlar. Sprint boyunca yapılan işlerin doğrularını, yanlışlarını ve yapılan işlerin kalitesinin değerlendirilmesi yapılır.

Scrum birçok avantaja sahiptir. Sahip olduğu bu avantajlar sayesinde günümüzde en çok tercih edilen modellerin başında gelir. Gerek teknolojinin gerekse artan nüfusa bağlı olarak ihtiyaçların artmasından dolayı insanların kısa zamanda daha hızlı ürün teslimatı ihtiyacı duyduğundan Scrum sprint adı verilen kısa zaman dilimleri içinde çalışarak ürün teslimatını hızlandırmasından dolayı tercih edilmektedir. Bunun yanı sıra daha iyi proje yönetimi ile süreci yönetmek için iyi bir çerçeve sağlar.

Neden Scrum?

Üretkenliği maksimuma ve takım üyelerinin işine daha çok sahip çıkmasına sebep olarak, takımları deneyimler yoluyla öğrenmeye bir sorun üzerinde çalışırken kendi kendine organize olmaya ve sürekli gelişmek için kazançları ve kayıpları üzerinde düşünmeye teşvik eder. Developer team yani geliştirme ekibi tarafından yapılan işler aynı anda gerçekleşir. Projenin ömrü boyunca ve sonrasında bile her şey esnek ve değiştirilebilirdir. Scrum takımın kendi kendisini yönetmesine olanak sağlayan bir yapıdadır




## Extreme Programing (XP)

Kent Beck tarafından geliştirilen bu disiplin grup içi iletişime oldukça önem veren bir yazılım geliştirme yöntemidir. Extreme programing (XP) ile proje yaşam döngüsü çerçevesinde proje geliştirimi sağlanır. XP in 4 temel noktası vardır bunlardan kısaca bahsedecek olursak; iletişim temel taşıdır. Çoğu projenin en önemli probleminin çıkma sebebi insanların birbiriyle tam olarak anlaşamaması nedeniyle olduğu görülür. XP de iletişim yüz yüze olmalıdır. Yazılım ekibi ile yazılımı kullanacaklar arasında sıkı bir iletişim bağı olması esastır. Bu sayede sorunlar erken fark edilir. Basitlik iste 2. Temel taşı diyebiliriz. Zorunlu işlerin yapılması basitlik olarak görülür. Gerekli olmayan bir şey varsa kesinlikle bu konu basitlik ilkesi içerisinde olmamalıdır. Zaman ve maliyeti göz önüne alınmış bir yapı oluşturmak gerekmektedir. XP en iyi şekilde bu basitliği sağlamak için bugünün ihtiyaçlarını temel hedeflerden biri alarak esnek basit bir sistem gerçekleştirmeye çalışır. Geri besleme; feedbackler sayesinde optimizasyonun oluşmasına engel olan problemler ortadan kaldırılır. Cesaret projelerin üzerine yılmadan gidilmesi projelerin geliştirmesi açısından son derece önemlidir. Cesaretin olmadığı projelerde korku gelişir ve gelişen bu korku projeyi başarısızlığa iter. Yazılımda başarısızlık yazılımın çöpe gitmesidir.

XP'nin temel noktalarının yanında 12 temel uygulama ilkesi de mevcuttur. Bu uygulama ilkeleri şunlardır;

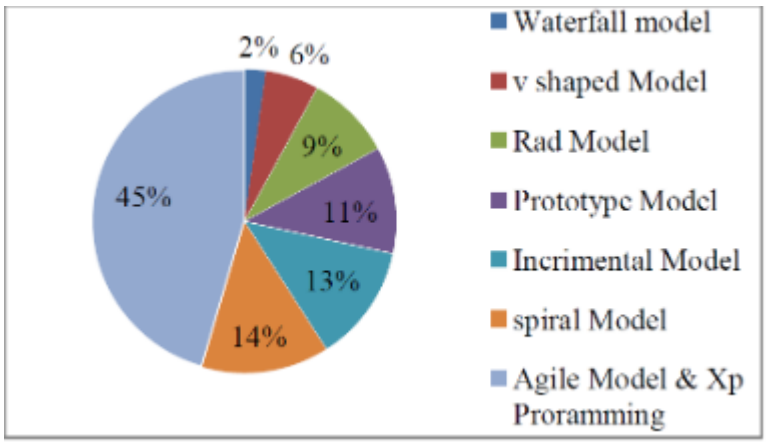
1. Planlama Oyunları: Proje planlaması, tüm takım üyelerinin katılımıyla gerçekleştirilir.
2. Küçük Sürümler: Geliştirme süreci, küçük ve sık sık yapılan sürümler halinde ilerler.

3. Basit Tasarım: Tasarım, mümkün olan en basit ve anlaşılır şekilde yapılır.
4. Metinsel İşlevler: İşlevler, müşteri ve geliştirme takımı tarafından belirlenen metinsel özellikler şeklinde tanımlanır.
5. Test İlk İlke: Kod yazmadan önce testler yazılır ve bu testler geliştirme sürecinin bir parçası olarak devamlı olarak güncellenir.
6. Sürekli Entegrasyon: Kodlar sürekli olarak entegre edilir ve her entegrasyondan sonra testler çalıştırılır.
7. Ortak Kod: Tüm geliştirme takımı ortak bir kod tabanı kullanır.
8. Yeniden Yapılanma: Kod her zaman yeniden yapılanabilir olmalıdır.
9. Küçük Ekipler: Küçük ekipler, iletişim ve iş birliğini artırmak için tercih edilir.
10. Standartlaştırılmış Kodlama: Tüm kodlama işlemleri belirlenmiş standartlar ve kurallara göre yapılır.
11. Günlük İletişim: Geliştirme takımı, günlük olarak bir araya gelir ve ilerleme raporlarını paylaşır.
12. 40 Saat Haftalık Çalışma: Takım üyelerinin aşırı çalışması ve iş yükü altında ezilmesi engellenir.

Bu modelleri karşılaştıracak olursak;

| No | Model/Özellik | Çevik Modeller | Su Dalgası Modeli | V Modeli | Evrimsel Geliştirme | Spiral Model |
|---|---|---|---|---|---|---|
| 1 | **Gereksinin Belirleme** | Belirli sıklıkla | Başlangıç | Başlangıç | Başlangıç | Belirli Sıklıkla |
| 2 | **Maliyet** | Çok Yüksek | Maliyetli | Maliyetli | Düşük | Maliyetli |
| 3 | **Başarı Garantisi** | Çok yüksek | Düşük | Orta | - | Yüksek |
| 4 | **Uzmanlık Gerekliliği** | Çok Yüksek | Orta | Orta | - | Yüksek |
| 5 | **Fazların Örtüşmesi** | Evet | Hayır | Hayır | Hayır | Hayır |
| 6 | **Maliyet Kontrolü** | Evet | Evet | Evet | Evet | Evet |
| 7 | **Basitlik** | Karmaşık | Basit | Orta | Karmaşık | Karmaşık |
| 8 | **Risk Duyarlılığı** | Azaltılmış | Yüksek | Yüksek Değil | Yüksek | Düşük |
| 9 | **Esneklik** | Çok esnek | Katı | Düşük | Düşük | Çok esnek |
| 10 | **Bakım** | Evet | Düşük Bakımlı | Düşük Bakımlı | Düşük | Evet |

| | | | | | | |
|---|---|---|---|---|---|---|
| 11 | **Değişiklik Yapma** | Zor | Zor | Zor | Zor | Kolay |
| 12 | **Yeniden Kullanılabilirlik** | Evet | Düşük İhtimal | Düşük İhtimal | Düşük | Mümkün |
| 13 | **Dokümantasyon ve Eğitim Gerekliliği** | Evet | Zorunluluk | Evet | Evet | Evet |
| 14 | **Zaman** | Kısa | Çok uzun | Uzun | Uzun | Uzun |
| 15 | **Uygulama** | Kolay | Kolay | Kolay | Kolay | Karmaşık |




Sonuç olarak yazılım süreç modelleri tamtamıyla birbirinden ayrı değildir ve çoğu zaman beraber kullanılır. Kullanılacak model yazılım türüne ve kullanım alanına göre seçilmelidir. Modelin seçimi için projenin karmaşıklığı, gerçekleştirilecek kurumun yapısı, zaman, maliyet ve oluşabilecek riskleri tolere edebilme kapasitesine dikkat edilmelidir. Günümüzde en çok kullanılan model Çevik modeldir. Kısa sürede esnek bir şekilde geliştirilerek, yüksek başarımla sonuçlanan bir model olduğundan tercih edilir. Büyük kitleler hedefleniyorsa evrimsel; büyük, maliyetli ve uzun süren projelerde spiral model ya ada artımsal model uygundur. Orta ve küçük büyüklükte, uzun sürmeyen projelerde çevik modeller; kişiye özel, zaman sorunu olmayan, küçük programlarda kodla ve düzelt kullanılabilir.

Vahap Can Çakır 200603010

https://medium.com/@vahapcakr1_71719

https://medium.com/@vahapcakr1_71719/yaz%C4%B1l%C4%B1m-geli%C5%9Ftirme-modelleri-c4ac535e92f5

https://github.com/vcncakr

https://www.linkedin.com/in/vahap-can-cakir-9254031ba/