Ece KARAALP

220601046

YAZILIM YAŞAM DÖNGÜ MODELLERİ ve SCRUM

Özet

Bilgisayar yazılımlarının ilk geliştirme, hata giderme, iyileştirme, yazılımların güncelleştirilmesi ve bakımı gibi üretim ve kullanım aşamalarının kontrollü bir biçimde yapılması gerekmektedir. Bu süreçte döngü modellerinin avantaj ve dezavantajlarına, projenin büyüklüğüne veya bu projenin kimler tarafından kullanılacağına göre hangi yazılım yaşam döngü modelinin kullanılacağı tespit edilerek o modelin aşamalarına uygun bir şekilde planlamalar yapılmaktadır. Döngü modellerinin birbirleri ile karşılaştırılması sonucu hangisinin daha az riskli, daha maliyetli, bakımının kolay veya karmaşık olmasına göre daha çok tercih edilen modeller tespit edilebilir.

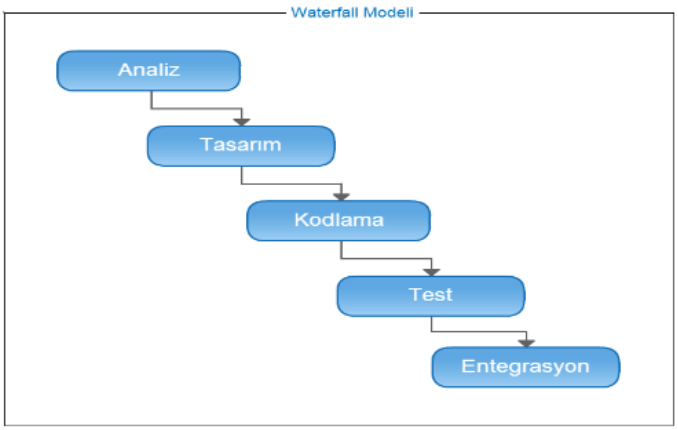
Giriş

Yazılım yaşam döngüsü (SDLC), bir yazılım ürünü geliştirilirken takip edilmesi gereken adımlardır. Bu adımlar;
1. Gereksinim
2. Analiz
3. Tasarım
4. Gerçekleştirme ve Test
5. Bakım
6. Emeklilik.

Yaşam döngü modellerindeki adımların sayısı ve niteliği tercih edilen modele göre değişiklik göstermektedir.

Şelale Modeli

Bu model geleneksel yazılım geliştirme modeli olarak da bilinir. SDLC modellerinden en eski ve en basit olanıdır. Adından da anlaşılacağı üzere yazılım geliştirme sürecinin adımları bir şelalenin dökülmesine benzetilmiştir. Bu açıdan arka arkaya devam eden aşamalardan oluşan bir süreç olarak görülebilir. Şelale modelinde işler aşama aşama yapılır. Bir aşama bitmeden diğerine geçilmez. Planlama ile başlayıp testlere ulaşana kadar sistemde pek bir değişiklik yapılamaz. Ancak sistemde yenilik yapılacaksa tekrar başa dönüp planlama aşaması gerçekleştirilmelidir. Bu da uzun süren projeler için ciddi bir sıkıntı çıkarmaktadır. Hem zaman açısından kayıp hem de maliyet artışına sebep olabilir (ŞEKER,2015).



Avantajları:

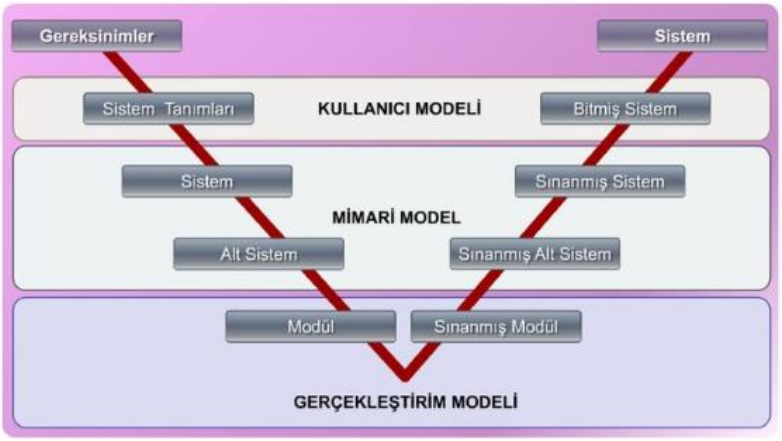
1. Basit veya daha küçük projeler için uygundur.
2. Gereksinimleri iyi anlaşılmıştır.
3. Proje yöneticileri için iş dağılımı yapmak kolaydır.
4. Her aşama için dokümantasyon gerektirdiği için kodun ve testlerin arasındaki mantığın daha iyi anlaşılmasını sağlar.

Dezavantajları:

1. Yazılımın son kullanıcıya ulaşması zaman alır.
2. İhtiyaçların sık değiştiği projeler için uygun değildir.
3. Doğru gereksinimleri toplamak zor olabilir.
4. Test aşamasından sonra eksik çıktığı fark edilirse geri dönüp düzeltmek zor ve maliyetlidir.

V Süreç Modeli

V modeli, şelale modelinin gelişmiş hali olarak düşünülebilir. Her aşama kendi kontrol aşamasıyla eşleştirilerek "V" harfine benzer şekilde gösterildiği için bu ismi almıştır. Temel olarak bu model testler arasında oluşan hataların düzeltilmesi için hangi düzeye gidilmesi konusunda yol göstermektedir (Özdemir, Reis ve Erol,2011).



Avantajları:

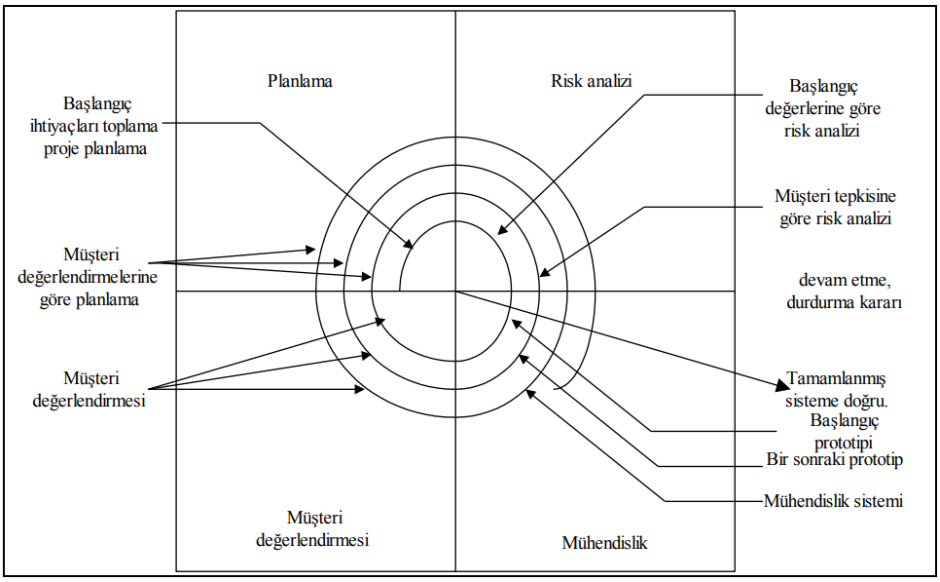
1. Hangi aşamanın ne şekilde test edileceği belirgindir. Böylece proje yönetimi ve takibi kolaylaşmış olur.
2. Kullanıcının projeye olan katkısını arttırmaktadır.

Dezavantajları:

1. Şelale modeline olan benzerliğinden dolayı ihtiyaçların ilerleyen aşamalarda anlaşılması, maliyetli geri dönüşler görülebilmektedir.
2. Aşamalar arasında tekrarlamalar yoktur.
3. Risk çözümleme aktiviteleri yoktur.

Spiral (Helezonik) Model

Şelale ve tekrarlı modelin kombinasyonundan oluşur. Spiral model ile geliştirme yapan takım öncelikle küçük bir gereksinim seti ile başlar ve bu gereksinimler tüm geliştirme aşamalarından geçerek ilerler. Uygulama üretime hazır olana kadar giderek artan spirallerde ek gereksinimler için işlevsellik eklenir. Bu modelde risk analizi ön plandadır ve prototip yaklaşımı vardır. Her döngü öncesi içinde bulunduğu fazın risk analizini yapar ve o faz için planlanmış olan prototip geliştirilir. Her döngünün sonunda yeniden planlamalar yapılır, alternatifler ve kısıtlamalar belirlenir. Genellikle önceden geliştirilmiş yazılım bileşenlerinin yeniden kullanıldığı projeler için uygundur.



Avantajları:

1. Sürekli geliştirme sağlandığı için risk yönetimini kolaylaştırır.
2. Yapılacak olan yenilikler veya değişiklikler daha sonraki bir aşamada yapılır.
3. Müşteri geri bildirimi için yer vardır ve yazılım erken üretilir.

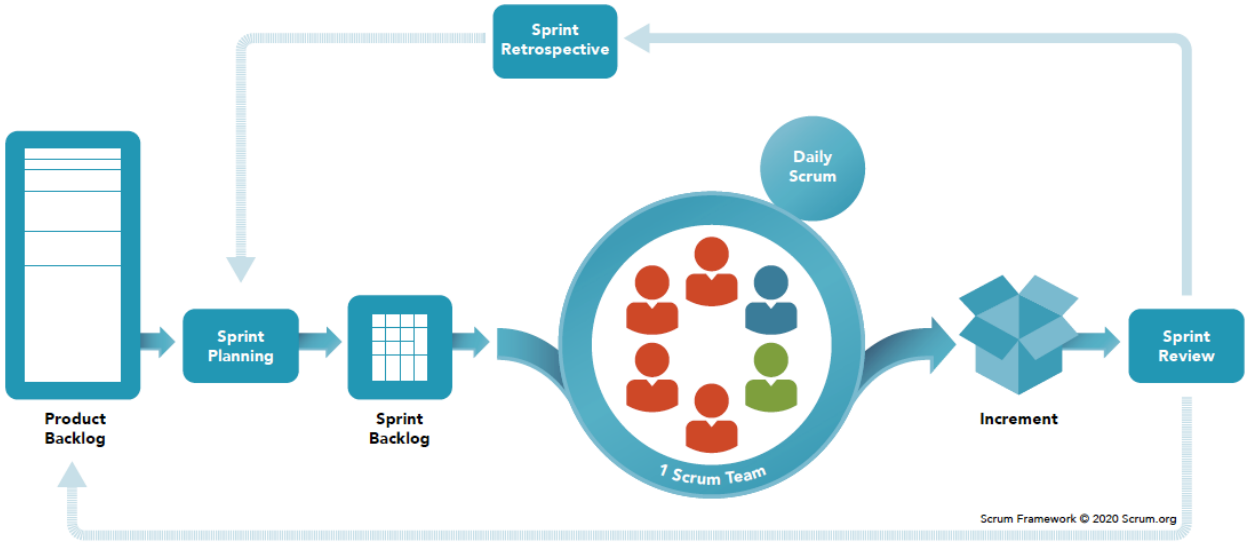
Dezavantajları:

1. Spiral sonsuza gidebilir ve oldukça karmaşıktır.
2. Ara aşamaları olduğu için belgeleme diğer modellere göre daha fazladır.
3. Küçük projeler için maliyetli olmaktadır.

SCRUM

Çevik yazılım geliştirme metodolojilerinden biridir. Kısaca çevik yazılım geliştirme metotları verimliliği yüksek, esnek, hata oranı az, hızlı ve maliyeti az çözümler sunmaktadır. SCRUM, karmaşık ürün ve projelerin gerçekleştirilmesi ve devamlılığını sağlamak için oluşturulmuş bir çerçevedir, işletmelere kendi yöntemlerini uygulama fırsatı sunmaktadır. SCRUM'ın felsefesi deneyciliktir (AYDINER, ESEN, ÖZLÜ,2020).

Bu metodoloji karmaşık yazılım işlerini küçük birimlere (sprint) bölerek geliştirmek için tasarlanmıştır. Söz edilen karmaşıklık şeffaflık, denetleme ve uyarlama ile azaltılmaya çalışılmaktadır. Ürün çıktısını planlamayı ve proje ilerlerken değişiklikleri yönetebilmeyi mümkün kılmaktadır. Böylece herhangi bir zaman dilimi içerisinde üründe ve teslimatta değişiklikler yapılmasını, en kısa sürede en uygun ürün çıktısını ortaya koymamızı sağlamaktadır.



SCRUM'un çalışma mantığı, müşteri tarafından istenilen işlevleri iki veya dört haftalık sprint adı verilen küçük birimlere bölerek geliştirilir ve yeniden kontrol edilir. Her sprint sonunda yazılım müşteriye teslim edilebilir bir durumda olmaktadır. Belki de en önemli kısım ekibin günlük kısa toplantılar yapması olabilir. Günlük toplantılar sayesinde ekip arasında paylaşım ve katkılar yoğun olacağı için ortaya çıkan sorunlar daha kısa sürede çözülmektedir. Sorunun ekip olarak çözülmesi sayesinde yazılımcının bireysel stresi daha düşük olur ve daha verimli işler ortaya koyabilir (İREN, KANTARCI).

Neden SCRUM Tercih Edilmeli?

Yaklaşımlarda, geliştirme ekibi tarafından yapılan işler ardışık olarak değil, aynı zaman dilimi içerisinde gerçekleşir. Yazılım geliştirmeye başlamadan önce geleneksel yazılım geliştirme yaklaşımlarındaki gibi ürün gereksinimlerinin ve tasarımın eksiksiz tamamlanmış olmasını beklemez, hemen geliştirmeye başlar. Projenin ömrü boyunca ve tamamlandıktan sonra bile değiştirilebilir. Ekip üyelerini kendi kendine organize olmaya ve işine adapte olmayı kazandırır.

Sonuç

Yazılım geliştirme döngü modellerine bakıldığında temelde benzer aşamalara sahiptir. Bazı modeller birbirinin gelişmiş hali de olabilir. Ama her döngü modelinin avantaj ve dezavantajlarına veya özelliklerine bakacak olursak üzerinde çalıştığımız projenin hangi modeli uygulayarak geliştirmeye yatkın olduğunu bulabiliriz. Böylece modellerin yazılımcılara planlı çalışmada kolaylık sağlaması zaman, maliyet ve müşteri memnuniyeti açısından büyük önem arz etmektedir.

KAYNAKÇA

1. SEKER, S. E. (2015). Yazılım geliştirme modelleri ve sistem/yazılım yaşam döngüsü. *Software development models and system/software lifecycle", YBS Ansiklopedi*, *2*(3), 18-29.
2. Özdemir, Ş., Reis, Z. A., & Erol, Ç. Yazılım Ürünü Geliştirme Sürecinin Örneklenmesi.
3. https://tr.wikipedia.org/wiki/V-Model_(Yaz%C4%B1l%C4%B1m_geli%C5%9Ftirme)
4. http://www.aspmvcnet.com/tr/m/yazilim-muhendisligi/helezonik-tasarim-spiral-tasarim.html
5. AYDINER, A. S., ESEN, M. F., & Erhan, Ö. Z. L. Ü. (2020). Türkiye'de Çevik Yazılım Geliştirme Süreçlerinde Scrum Yöntemini Uygulayan İşletmelerin Başarı Faktörleri. *Bilişim Teknolojileri Dergisi*, *13*(4), 463-477.
6. İren, E., & Kantarcı, A. SCRUM Yazılım Geliştirme Metodu Üzerine Bir İnceleme ve Değerlendirme.
7. https://batuhanakpunar.medium.com/b%C3%B6l%C3%BCm-1-scrum-nedir-neden-scrum-b00b48f25e54

Hesaplarım

1. https://medium.com/@ecekrlp/yazilim-ya%C5%9Fam-d%C3%B6ng%C3%BC-modelleri%CC%87-ve-scrum-3f368ec37fc0https://medium.com/@ecekrlp
2. https://github.com/EceKaraalp?tab=repositories
3. https://www.linkedin.com/in/ece-karaalp-46b88a25a/