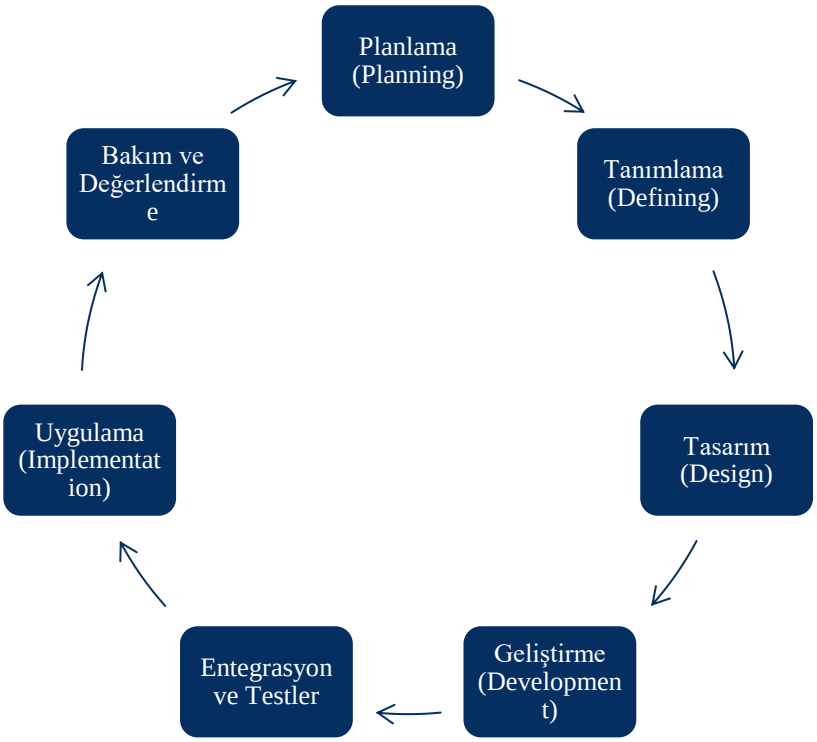
Yazılım Yaşam Döngü Modelleri

Ebrar Şeyma Karakuş

No:220601049

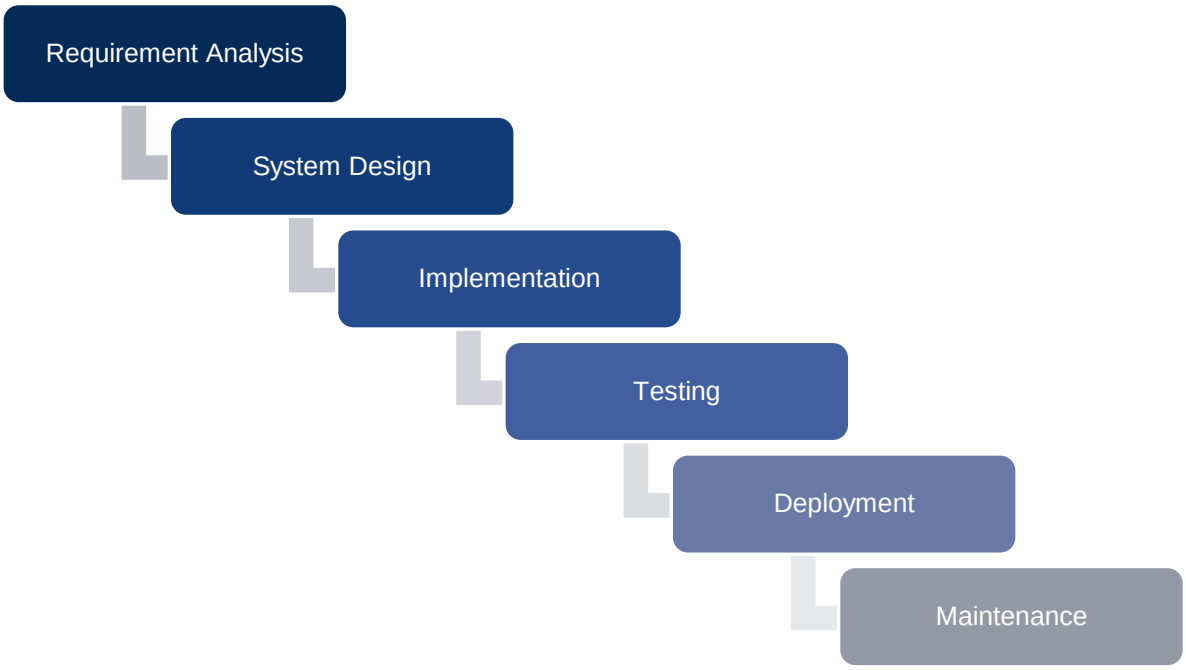
GİRİŞ

Başlatma, analiz, tasarım ve uygulama ile başlayan sistemin bakımı ve sonlandırılması ile devam eden çok adımlı sürece Sistem Geliştirme Yaşam Döngüsü (SDLC) denir. SDLC, müşterinin beklentisini karşılayan, son tarih ve maliyet tahminleri içinde tamamlanmaya ulaşan etkin ve verimli çalışan yüksek kaliteli bir yazılımla sonuçlanmalıdır. Günümüzde pek çok yazılım geliştirme yaşam döngüsü modeli mevcuttur. Çok model olduğundan, gereksinimlerine göre kullanılırlar. Bu nedenle, yazılım yaşam döngüsü modellerinin gereksinimlerin bilinmesi gerekmektedir. Bu makale, yazılım yaşam döngüsü modelleri ve bazı modellerin artıları ve eksileri ile ilgilidir.



Şelale Modeli (Waterfall Model)

Şelale modeli çoğunlukla Gantt şeması kullanılarak planlanır, yani bir aşamayı tamamlanır ve ardından bir sonraki aşamaya geçilir. Tamamlanan bir aşamayı asla tekrar gözden geçirilmez. Şelale yöntemi için temel yaklaşım, gereksinim kararlılığı belirlemesidir. Gereksinimleri, gerekli bilgileri, işlevi, davranışı, performansları ve arayüzleri tanımlar. Tasarım, test kaynağına dayalı yazılım mimarisi, arayüz temsilleri ve uygulamalarından oluşur. Bu model kullanılırken ürün ve teknolojisi iyi anlaşılmıştır. Belirsiz gereksinimler yoktur.



Şelale modelinin avantajları şunlardır:

- Basit ve kullanımı ve anlaşılması kolay.
- Her aşama ayrıdır ve bir aşamada yer alan ekip üyeleri, bir sonraki aşamaya geçmeden önce aşamanın mükemmelleştirilmesini sağlar. Şelale modeli, daha fazla proje çıktısı sağlar.
- Ekip üyeleri farklı yerlere dağıldığında bu yaklaşım çok verimli olabilir.
- Aşamalar çakışmaz ve bu nedenle yönetim çok verimli bir şekilde yapılır.
- Şelale modelini uygulamak için gereken kaynak miktarı diğer yöntemlere göre daha azdır.
- Küçük projeler için çok iyi çalışır
- Ekip üyeleri bağımsız çalışabilir.

Şelale modelinin dezavantajları:

- Bu yaklaşımın en büyük dezavantajı, geri dönüşün olmamasıdır. Bir aşama tamamlandığında, kilitlendiği anlamına gelir.
- Bazen farklı aşamalar için gereken süreyi tahmin etmek gerçekten zordur ve yanlış varsayım, projenin son teslim tarihini tutturamamasına neden olabilir.
- Şelale modeli, müşterinin ihtiyacına göre değişikliklere izin vermez, bu nedenle esnek değildir.
- Şelale sürecinde değişiklik yapılacaksa projeye baştan başlanmalıdır. Bu, bazı şirketler için pahalı olabilir.
- Müşteri listeye gereksinim eklemeye devam ettiğinde, yazılım geliştirme süresini ve masrafını artırır.
- Mevcut çalışma aşamasında olan ekip üyesi hariç, aşamanın geri kalanındaki ekip üyeleri boşta oturur.

Spiral Model

Spiral model, oluşabilecek riskler için destek sağlayan en önemli Yazılım Geliştirme Yaşam Döngüsü modellerinden biridir. Şematik gösteriminde, birçok ilmeği olan bir spirale benzer. Spiralin tam ilmek sayısı bilinmez ve projeden projeye değişebilir. Sarmalın her döngüsü, yazılım geliştirme sürecinin bir aşaması olarak adlandırılır. Ürünü geliştirmek için gereken aşamaların tam sayısı, proje risklerine bağlı olarak proje yöneticisi tarafından değiştirilebilir. Proje yöneticisi faz sayısını dinamik olarak belirlediğinden, sarmal modeli kullanarak bir ürün geliştirmek için proje yöneticisinin rolü önemlidir. Yazılım geliştirmeye sistematik ve yinelemeli bir yaklaşım sağlayan bir yazılım geliştirme yaşam döngüsü (SDLC) modelidir.

Spiral modelin avantajları:

- Spiral yaşam döngüsü modeli, en esnek SDLC modellerinden biridir. Geliştirme aşamaları, projenin karmaşıklığına göre proje yöneticisi tarafından belirlenebilir.
- Proje izleme, her yinelemenin her aşamasında yapıldığı için çok kolay ve etkilidir.
- Bu izleme uzmanlar veya ilgili kişiler tarafından yapılır. Bu, modeli daha şeffaf hale getirir.
- Risk yönetimi, bu modeli diğer modellere kıyasla daha çekici kılan temel özelliğidir.
- Değişiklikler yaşam döngüsünün sonraki aşamalarında yapılırsa, bu değişikliklerle baş etmek proje yöneticisi için çok büyük bir sorun değildir.
- İş ihtiyaçlarının istikrarsız olabileceği yüksek riskli projeler için uygundur.
- Bu model kullanılarak son derece özelleştirilmiş bir ürün geliştirilebilir.
- Müşterinin ihtiyaçlarını karşılayan ürünün sorunsuz bir şekilde geliştirilmesini sağlar.

Spiral modelin dezavantajları:

- Bu modelde yer alan maliyet genellikle çok yüksektir.
- Özellikle net bir SRS'ye (sistem) proje için karmaşık bir yaklaşımdır.
- Bu modeli etkili bir şekilde uygulamak için kurallar ve protokoller uygun şekilde izlenmelidir. Proje yaşam döngüsü gelişimi boyunca, kuralları ve protokolleri takip etmek çok zordur.
- Düşük riskli projeler için uygun değildir.
- Bu yazılım geliştirme sürecinde bütçe ve zamanlama gereksinimlerini karşılamak çok zordur.
- Müşteri tarafından izin verilen çeşitli özelleştirmeler nedeniyle, aynı prototipi başka projelerde kullanmak çok zordur.
- Proje ve bunların azaltılması ile ilgili belirsizlikleri veya riskleri değerlendirmek için kapsamlı bir beceri gerektirir.
- Modeller, yalnızca ilgili maliyetlerin çok daha yüksek olduğu ve sistem ön gereksinimlerinin daha yüksek düzeyde karmaşıklık içerdiği büyük projeler için en iyi sonucu verir.
- Risk değerlendirme uzmanlığı gereklidir.

Scrum Modeli

Scrum , Çevik çerçeve türüdür . Üretkenlik ve ürün sunma yaratıcılığı mümkün olan en yüksek değerlerdeyken, insanların karmaşık uyarlanabilir sorunları ele alabileceği bir çerçevedir. Scrum, yinelemeli süreci kullanır. Süreç sprintlerden oluşur. Odak, bağımsız ve değerli bazı özellikler sunmaktır. Her özellik diğerinden bağımsızdır, herhangi bir özellik arızalanırsa diğer özellikler etkilenmez.

Scrum modelinin avantajları:

- Scrum, şirketin zamandan ve paradan tasarruf etmesine yardımcı olur.
- Scrum modeli, iş gereksinimleri belgelerinin ölçülmesinin zor olduğu projelerin başarılı bir şekilde geliştirilmesini sağlar.
- Hızlı hareket eden gelişmeler, bir hata kolayca düzeltilebileceğinden, bu yöntem kullanılarak hızlı bir şekilde kodlanabilir ve test edilebilir.
- Düzenli toplantılar yoluyla işteki ilerlemenin sık sık güncellenmesinde ısrar eden, kontrollü bir yöntemdir. Böylece, proje geliştirmenin net bir görünürlüğü vardır.
- Diğer herhangi bir çevik metodoloji gibi, bu da doğası gereği yinelemelidir. Kullanıcıdan sürekli geri bildirim gerektirir.
- Kısa sprintler ve sürekli geri bildirim nedeniyle değişikliklerle baş etmek daha kolay hale gelir.
- Günlük toplantılar bireysel verimliliği ölçmeyi mümkün kılar. Bu, ekip üyelerinin her birinin üretkenliğinin artmasına yol açar.
- Günlük toplantılarla sorunlar önceden belirlenir ve bu nedenle hızlı bir şekilde çözülebilir.
- Kaliteli bir ürünü planlanan zamanda teslim etmek daha kolaydır.
- Scrum herhangi bir teknoloji / programlama dili ile çalışabilir, ancak hızlı hareket eden web teknolojisi veya yeni medya projeleri için özellikle yararlıdır.
- Süreç ve yönetim açısından genel gider maliyeti minimumdur, böylece daha hızlı bir sonuca yol açar.

Scrum modelinin dezavantajları:

- Scrum, önde gelen yazılım geliştirme modellerinden biridir çünkü kesin bir bitiş tarihi yoksa, proje yönetimi paydaşları teslim edilecek yeni işlevleri talep etmeye devam etmek için kullanılacaktır.
- Bir görev iyi tanımlanmamışsa, proje maliyetlerini ve süresini tahmin etmek doğru olmayacaktır. Böyle bir durumda, görev birkaç sprint'e yayılabilir.
- Ekip üyeleri taahhütte bulunmazlarsa, proje ya asla tamamlanmaz ya da başarısız olur.
- Sadece küçük ekiplerle iyi çalıştığı için küçük, hızlı hareket eden projeler için iyidir.
- Bu metodoloji sadece deneyimli ekip üyelerine ihtiyaç duyar. Ekip bu teknolojide yeni olan kişilerden oluşuyorsa proje zamanında tamamlanamaz.
- Scrum Master, yönettiği takıma güvendiğinde Scrum iyi çalışır. Ekip üyeleri üzerinde çok sıkı kontrol uygularlarsa, bu onlar için son derece sinir bozucu olabilir, moral bozukluğuna ve projenin başarısızlığına yol açabilir.
- Bir geliştirme sırasında ekip üyelerinden herhangi biri ayrılırsa, bunun proje geliştirme üzerinde büyük bir ters etkisi olabilir

- Test ekibi her sprintten sonra başarısızlık testi yapamıyorsa, proje kalite yönetiminin uygulanması ve ölçülmesi zordur.

V Model

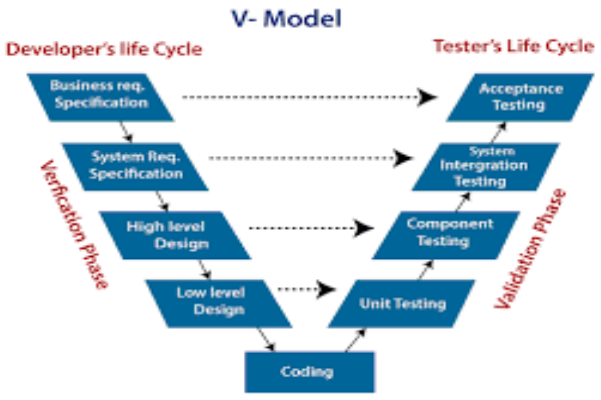
Bu modelde yazılımı her aşamada tersten test edilir. Ürünü her aşamanın gereksinimlerine göre doğrulamak ve doğrulamak için her aşamada test planları ve test senaryoları oluşturulur. Bu nedenle, doğrulama ve doğrulama paralel gider.

V modelin avantajları:

- Basit ve kullanımı kolay
- Test faaliyetleri kodlamadan önce planlanır. Bu çok zaman kazandırır ve aynı zamanda projenin başlangıç durumunda çok iyi anlaşılmasına yardımcı olur.
- Her aşamanın belirli çıktıları vardır.
- Gereksinimlerin kolayca anlaşıldığı yerlerde iyi çalışır.
- Küçük projeler için çalışır.
- Yaşam döngüsünün başlarında test planlarının geliştirilmesi nedeniyle daha yüksek başarı şansı.
- Model, tüm dahili ve harici çıktıların doğrulanmasını ve doğrulanmasını teşvik eder. Sadece yazılım ürünleri değildir.
- V şeklindeki model, sistemi tasarlamadan önce gereksinimlerin tanımlanmasını ve bileşenleri oluşturmadan önce yazılımın tasarlanmasını teşvik eder.
- Geliştirme sürecinin üretmesi gereken ürünleri tanımlar, her çıktı test edilebilir olmalıdır.

V modelin dezavantajları:

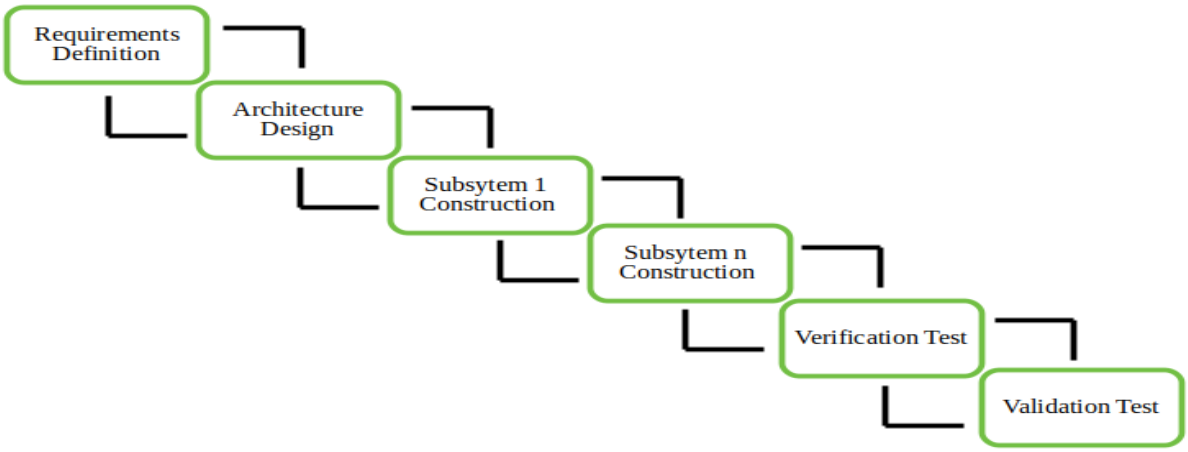
- Esnek değildir; değişime cevap verme yeteneği yoktur. Çok katıdır.
- Verimsiz test metodolojileri üretir.
- Ortada herhangi bir değişiklik olursa, ihtiyaç belgesi ile birlikte en iyi belgeler güncellenmelidir.
- Kapsamı ayarlamak zor ve pahalıdır.
- Yazılım, uygulama aşamasında geliştirilir, dolayısıyla yazılımın ilk prototipleri üretilmez.
- Model, test aşamalarında bulunan sorunlar için net bir yol sağlamaz.
- Eşzamanlı olayı işlemez.



Arttırımlı Model (Incremental Model)

Modelde bir döngü söz konusudur. Planlama aşaması ile başlar. Planlamalardan geçip bu planlamanın ardından bir döngüye girilir. İstek analizleri, analizler, tasarımlar yapılır ve uygulamaya geçilir. Uygulamadan sonra iki durum söz konusudur. Birincisi canlıya geçiş durumudur. Deployment edilir ve tabi problemler her aşamada olduğu gibi bu aşamada da yaşanır. Ama bu problemler testlerin birer parçası olarak görülebilir. Bir yandan da testlerimiz devam eder. Bu problemler sistemi besler. Testlerden ve canlıdan gelen bilgilere göre tekrar değerlendirilmeye tabii tutulur. Sonra tekrar istek analiz, analizler, canlıya bir bilgi verilmesi ve tekrar testler gibi süren bir döngü vardır. Bu sırada planlamada değişiklikler olabilir.

Örneğin; tekstil firmasına yazılım yazılıyor olsun. Tekstil firması iç süreçler değişebilir. Web sayfası üzerinden satış yapılıyor olsun. Web sayfası ile ilgili yeni ihtiyaçlar gelmeye başlar. Bir yandan yazılımın çalışan hali ile ilgili bir problemler ortaya çıkar. Bir yandan da "Orada şu modüle şu ekranı da ekleyelim" gibi fikirler, yen istekler çıkmaya başlar. İşte bütün bunlar arttırımlı modelde her seferinde arttırıla arttırıla bir iterasyon şeklinde devam eder ve her yenilik bir sonraki iterasyonda sürece dâhil edilir.



Arttırımlı modelin avantajları:

- Sürümler, artımlı model yinelendikten sonra sağlanır.
- İlk yinelenen modeli kullandıktan sonra, kullanıcı öneri ve değişiklik taleplerini verebilir.
- Bu model, müşterinin ilk yinelemeden sonra ihtiyaç duyduğu yazılımın çekirdeğini sağladığı için kimsenin iş değerlerini etkilemez. Müşteriye yazılım, işini yürütmeye devam etmesine yardımcı olacaktır.
- Müşterinin gereksinimlerine göre esnek ve yönetimi kolay bir modeldir.
- Bu modelde daha iyi risk yönetimi vardır çünkü her sürüm plana göre hazırlandığı için her sürümden sonra müşteri tarafından sonuç teyit edilebilir.
- Test, ihtiyaca göre yinelemede yapıldığından test edilmesi kolaydır.

- Bu model, gereksinimler bir dereceye kadar net olduğunda ancak proje kapsamı saf doğrusal yaklaşım gerektirdiğinde kullanılır.
- Kararlaştırılan son tarihe kadar uygulamayı tamamlayın.
- Bazen erken artışlar daha az kişiyle uygulanabilir.
- Diğer yaklaşımlara kıyasla projenin başarısız olma riskini sever.
- Sonuçlar erken ve periyodik olarak alınır.
- Paralel gelişim planlanabilir.
- İlerleme, dönüm noktası belirlenerek ölçülebilir.
- Daha küçük yineleme sırasında test etme ve hata ayıklama kolaydır.

Arttılımlı modelin dezavantajları:

- Yinelemenin her aşaması çok katıdır ve birbiriyle örtüşmez.
- Geliştirme sürecinin ilk aşamasında tüm gereksinimler bir araya getirilmediği için sistem mimarisi ile ilgili sorunlar ortaya çıkabilir.
- Her artışın nispeten küçük olması gerekir.
- Gereksinimleri artışlarla eşleştirmek kolay olmayabilir, bu nedenle belgeleri yönetmek çok zordur.
- Yazılımın ortak özelliklerini belirlemek zordur.
- Geliştirme sürecinde değişiklikler ilk iterasyonda yapılıyor. Sanki değişmeye devam ediyor ve hiç bitmiyor.
- Daha fazla kaynak gerekebilir.
- Gereksinimlerdeki sık değişiklikler nedeniyle yönetimin daha fazla dikkat etmesi gerekiyor.
- Bir artış içinde yinelemelere izin vermez.

KAYNAKÇA

[1] www.en.wikipedia.org/wiki/Systems_development_lifecycle

[2] www.en.wikipedia.org/wiki/SDLC

[3]www.waterfall-model.com/sdlc/
[4]www.allinterview.com › Categories › Software