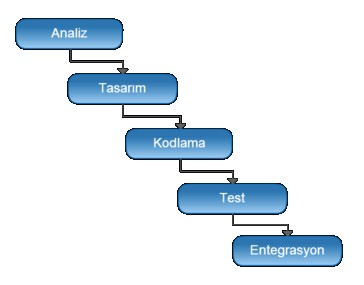
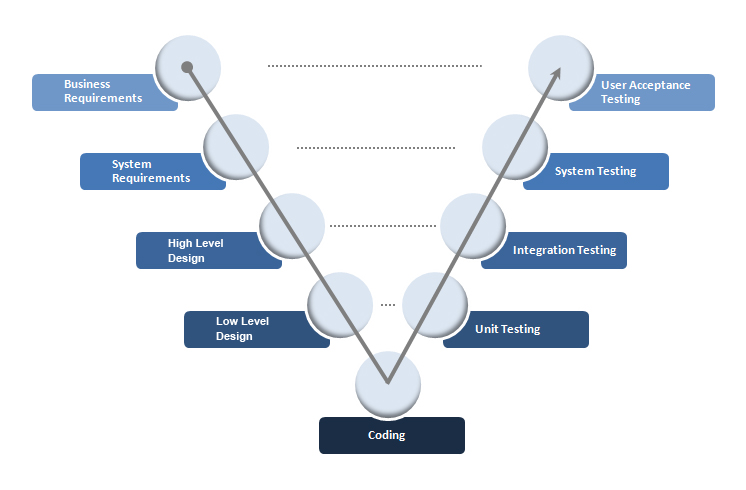
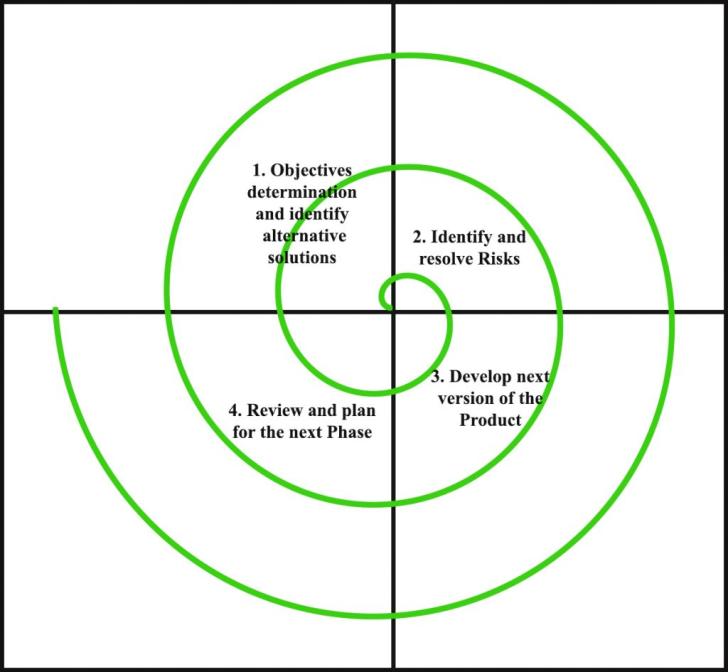
Bir yazılım ürünü oluşturmak için neler gerekmektedir ? Bir müşteri ki her zaman müşteri vardır. Gereksinimler ve amaçlar. Yazılım ürünümüzü oluşturmaya başlarken teoride gereksinimlere göre planlamamızı yaparız ve bu plan doğrultusunda ilerleyerek ürünümüzü oluştururuz. Gereksinimleri belirlemek, analizini yapmak, tasarlamak ve gerçekleştirmek. Fakat bunu hiçbir zaman gerçekleştiremeyiz. Çünkü başarıya giden yol doğrusal değildir birden farklı yoldan geçmek gerekir. Projemizde hatalar oluşabilir, müşterinin istekleri değişebilir dolayısıyla da gereksinimler değişir. Gereksinimler projemizin gidişatına bağlı olarak değiştiğinden hedefimize doğrudan ulaşamıyoruz. Keza karmaşık yazılımlar da buna örnektir. Bu karmaşık yazılım ürünlerini oluşturmanın birden fazla yöntemi vardır. Bunlara yazılım geliştirme modelleri diyoruz. Gelin birlikte yazılım geliştirme modellerine bakalım. 1960 yıllarında çoğunlukla kullanılan gelişigüzel modellemeydi. Bu model bir ekip halinde değil, tek başına çalışan yazılım mühendislerinin basit yazılımlar üzerinde kullandığı yöntemdi. Hatta bir yöntem de sayılmaz bir nevi doğaçlama yapmak gibidir. 1970 yıllarında sıkça kullanılan Barok Modeli. Bu modelde ürün planı doğrusal olarak gider ve aşamalar sırasında nasıl geri dönüş yapılabileceği tanımlı değildir. Bu karmaşık yazılım modellerinde büyük sorun teşkil etmektedir. Buna bağlı olarak plan doğrusal olduğundan dokümantasyon şuandaki gibi aşama aşama değil projeden bağımsız olarak proje bittikten sonra hazırlanabilir. Proje ve dokümantasyon 2 farklı süreç olarak ele alınır ve ayrı ayrı yapılır. Geçmişte kullanılan en popüler yazılım modeli Şelale (waterfall) modeli.



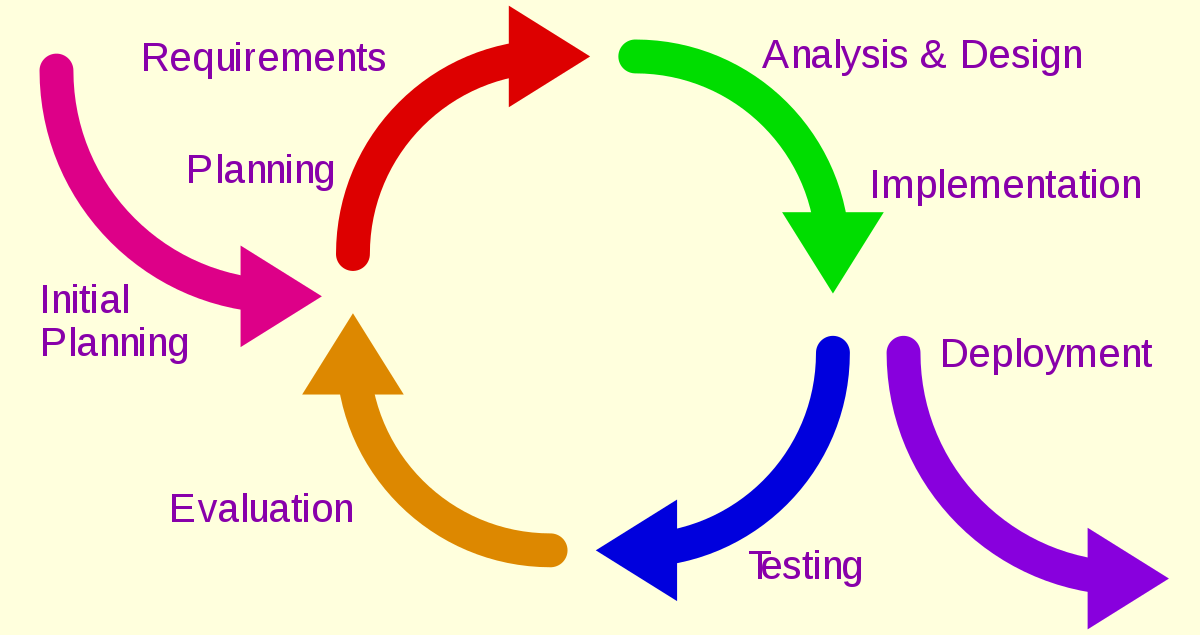
Bu modelde plan akışı tıpkı bir şelale gibi doğrusal şekilde ilerler. Fakat Barok modelinden farklı olarak başta bir plan yapılır ve dokümantasyona girilebilir. İlk olarak plan yapılır. Plana bağlı olarak yapım aşamasına geçilir. Yapım aşaması bittikten sonra test aşamasına geçilir. Proje bitmeden test edilemediği için bazı sorunları erkenden görme ve çözüm üretebilme fırsatı kaçırılmış olunur. Proje bitmeden değişim gerçekleşemez. Gereksinimlerin değişmesi burada büyük sorun teşkil eder. Plan temelli olduğundan dinamikliği az bir yöntemdir. Proje sürecini domine eden tek kişi vardır (Proje yöneticisi). Yinelemeli Şelale modeli de şelale modeline kıyasla daha az risklidir fakat çevik yöntemlerle karşılaştırdığımızda yine de çok daha riskli ve verimliliği düşüktür. Bu modeldeki en çok yaşanan sorun dar boğaz oluşmasıdır. Örneğin son ana kadar kodunuzda sorun çıkmaz ve en sonda sorun çıkar fakat tüm paketi ele alabilmeniz mümkün olmayabilir. Diğer bir sorun ise proje bitmeden herhangi bir ürün elde edilemediğinden sonuç elde etmek oldukça uzun sürer. Bu durum özellikle de sabırsız müşterileri kötü etkiler. Aynı şekilde proje üstünde çalışan insanlar da çalıştıklarının karşılığını alamadıkları için motivasyonları düşebilmektedir. Proje üstünde çalışanlar için önlerindeki yolu görmek oldukça zordur. Yolun başındalar mı sonundalar mı bilemezler. Çünkü gereksinimlerin her değişiminde veyahut da her bir hatada şelalenin başına geri dönmek zorundadırlar. Şelale modelindeki hatalar proje bitiminde anlaşılır. Bu yüzden V modeli denilen bir modelleme ortaya çıkmıştır.



Bu modelde bir aşama tam anlamıyla bitirilmeden diğer aşamaya geçmek söz konusu değildir. V Model, doğrusal modellerden biridir ve şelale modelinin test bölgesinin daha da planlanmış hali olarak görülebilir. Her bir aşama kendi test aşamasıyla eşleştirerek “V” harfine benzer şekilde gösterildiği için bu ismi almıştır. V’nin sol kolu üretim kısmı iken sağ tarafı sınanma kısmıdır. Yani her adımımızı kontrol ederiz ve böylece ortaya çıkacak riskler, sorunlar azaltılır. Daha güvenli bir şekilde projemizi oluşturabiliriz. Dezavantajları ise tıpkı şelale modelinde olduğu gibi doğrusal olması ve müşterinin gereksinimlerine bağlı olarak değişememesidir. Her ne kadar bir nebze de olsa güvenli bir şekilde ilerlenilse de dinamik ve verimli değildir. Bir diğer yazılım modelimiz Spiral(Helezonik).

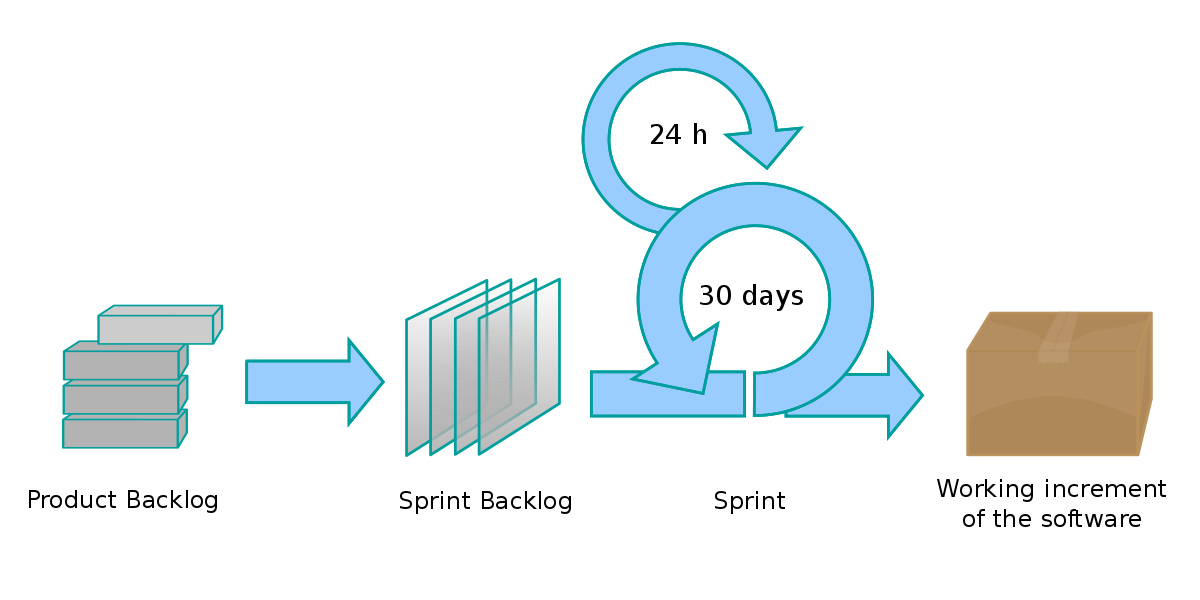


Spiral modellemede Şelale modeli belli iterasyonlara (yinelemelere) bölünür. Bu sayede doğrusallıktan kurtuluruz ve daha dinamik bir yönteme başvurmuş oluruz. Her iterasyon parçası için planlama ve risk analizi yapılır. Her bir iterasyondan çıkan sonuçlara göre testler yapılır. Bu testler sonucunda ortaya çıkan hatalar, tüm projeyi değil, o iterasyon parçasını etkiler. Bu sayede proje hem daha hızlı gerçekleşir hem sorunlar daha kolay çözülür hem de daha hızlı sonuç alınır. Fakat projemizde bölmüş olduğumuz parçaları birleştirirken hatalar oluşabilir, tümleştirme cehennemi ortaya çıkabilir. Proje birçok parçaya bölündüğünden bu modellemenin maliyeti oldukça artmaktadır. Basit ve düşük riskli projeler için maliyetli bir yöntemdir. Daha karmaşıktır. Spiral sonsuza kadar sürebilir. Ara adımların fazlalığı nedeniyle çok fazla dokümantasyon gerektirir. Bu sayede zaman kaybı da artar. Yazılımın içten gelişeceğini öngörür. Öznel risk değerlendirme deneyimine dayanması objektif olarak değerlendirilememesine yol açar. Diğer bir modelleme çeşidi de Yinelemeli modeldir(Iterative Model).



Proje parçalara bölünmez. Bu parçaları birleştirirken oluşabilecek sorunları ortadan kaldırır. Çalışabilecek, en küçük yazılım kısa sürede üretilir. Daha sonra sürekli olarak kendini geliştirerek ortaya çıkan hatalar, eksiklikler ve yanlışlar düzeltilir. Devamında da bir önceki aşamadan dolayı ortaya çıkan hatalar, bozukluklar giderilir, sırası gelen diğer gereksinimler eklenir. Her iterasyonda projede çalışan tüm ekip işlerini birleştirirler. Böylelikle "Tümleştirme Cehennemi" denilen durumla karşılaşılmaz. Yani Spiral modeldeki birleştirmeler sonucu ortaya çıkan yeni sorunlardan da kurtulunmuş olunur. Tümleştirme cehennemini açıklamamız gerekirse projeyi oluşturan parçaların birleşince çalışamaması durumdur. Her bir parça, fonksiyon kendi görevini gerçekleştirir. Fakat bir bütün haline geldiklerinde ise istendiği gibi çalışmadıkları, hatta ve hatta hiçbir bütünlük oluşturmadıkları anlaşılır. Yinelemeli geliştirme(iterative model) bunu engeller. Çünkü proje sürekli olarak bütünleştiği için, her bir parça eşzamanlı olarak tümleştirmede ortaya çıkabilecek tüm sorunlar kısa sürede farkedilir ve düzeltilir. Tümleştirme sorunları projenin sonlarına doğru değil ilk aşamasından itibaren farkedilebilir. Yinelemeli geliştirmenin bir faydası da müşterimizden Feedback alınmasıdır. Bütün proje müşterinin gözetimi altında ilerler ve sürekli olarak gereksinimlere göre yazılım oluşturulabilir. Böylece müşterinin hayal kırıklığı yaşaması büyük beklentilere girmesi önlenmektedir. Çünkü proje müşterimizin ihtiyaçlarına isteklerine bağlı olarak sürekli olarak değişir ve proje daha güvenilir bir biçimde oluşturulur.

Müşteri, isteklerini en başta tam olarak sunamasa bile bu yöntem ile kolay bir şekilde çözülür. Yinelemeli geliştirmenin bir diğer faydası da proje üzerinde çalışan ekibin proje ile ilgili bilgilerini arttırmasıdır. Uzun bir vade sonunda ortaya çıkabilecek bir projenin bir parçasını yapmak, geliştiricinin fotoğrafa uzaktan bakmasını engeller. Bütünü göremeyen kişi de doğal olarak sadece kendisinden isteneni yapar ve yaptığı işin diğer ekip üyeleriyle entegre çalışıp çalışmayacağını dikkate alamaz. Çünkü oluşturulan bir parçanın istendiği gibi olup olmadığı diğerleriyle birleştirilmeden bilinemez. Halbuki sık sık entegre olan projelerde ortaya çıkarılmış olan bu ürün herkes tarafından görüldüğü sürekli olarak düzeltilir, tüm ekip üyeleri kendi görevini bilir. Yinelemeli modelleme en çok da son teknolojiden yararlanırken kullanılır. Son teknolojik ürünler hakkında ekibimiz tam bir bilgi sahibi olmadığından, en doğru modellemelerden biridir yinelemeli modelleme. Ürün ilk seferde oluşturulur ve test edilir. Yineleme sonucu oluşan hatalar giderilir, son teknolojik ürünümüz hakkında oluşan yeni bilgilere göre ürün sürekli olarak gelişir, kendini günceller. Fakat her modelde olduğu gibi bu modelin de dezavantajları bulunmaktadır. Bazı müşteriler alışkanlıklarından vazgeçemez. Projedeki hatanın değişmesi için bazı noktaları değiştirmeniz gerekmektedir. Fakat müşteri bunu istemeyebilir. Bu modelleme özellikle öğrenmeye ve çalışmaya meraklı olan müşteriler için uygundur. Günümüzde popüler olan çoğu projede, özellikle sosyal medyada bunu görebiliriz. Kullanıcılar sevdiği platformu merak eder ve her güncelleme geldiğinde kullandıkları ürünü anlamaya çalışırlar. Fakat sürekli olan bu güncellemeler, yeni yazılım donanım gereksinimleri ortaya çıkarabilir. 1980-1990'larda ortaya çıkan ve oldukça popüler olan bu modelleme çeşidimiz ise Çevik yazılım geliştirme süreçlerinin bir çeşididir(Agile Development). 1990'ların başlarında ortaya çıkarılan Scrum bir SDLC(Yazılım geliştirme yaşam döngüsü) olmayıp, bize bir Çevik Altyapı (Agile Framework) sunmaktadır.



Yani sadece yazılım geliştirme süreçlerinde değil ekip diğer tüm süreçlerde kullanılabilir. Başarısı kanıtlanmış bir yazılım geliştirme yöntemidir. Microsoft, Google, Facebook gibi şirketlerde kullanılmaktadır. Değişen gereksinimlerle daha etkin bir yoldan başa çıkmak, çalışanların motivasyonunu artırmak ve müşteri ile proje arasındaki iletişimi geliştirmek istiyorsak Scrum kullanılması kaçınılmazdır. Scrum yazılım mühendisliğinde uygulama geliştirme yöntemidir. Bir yazılım projesinin karmaşıklılığını azaltmak için kullanılan bir yöntemdir ve 3 ilke ile bu sağlanır. 1)Şeffaflık: Projedeki sorunlar ve çözümler günlük olarak not edilir ve bu bilgilere herkes ulaşabilir. 2)Denetleme: Proje parça parça oluşturulur, teslim edilir ve teker teker kontrol edilir. 3)Uyarlama: Gereksinimleri en baştan belirlemek ve bu gereksinimlerin yetmesi beklenemez. Keza her parça tesliminde yeni gereksinimler ortaya çıkabilir. Bu yöntemde tıpkı yinelemeli modelleme gibidir. Fakat işin iyi tarafı, geliştirici gereksiz masraflardan kaçınır ve çok daha ekonomik bir şekilde ürün oluşturulur. Bu hem finansal açıdan müşteriyi daha da memnun eder, hem de daha çok tercih edilmenizi sağlar. Bu yöntemde kapsamlı dokümantasyonlardan çok çalışan işlevli yazılıma önem verilir. Kağıt üzerindeki anlaşmalardan çok müşterinin ihtiyaçlarını dinler ve buna göre hareket edilir. Tıpkı iterative de olduğu gibi bu da oldukça dinamik bir yöntemdir. Peki günümüzde bu kadar popüler olması sağlayan sebepler nedir? Şöyle ki Agile yöntemi ile müşteri doğrudan doğruya geliştirme sürecine dahil ediliyor. Ekip çalışmasını ön planda tuttuğu için ekip içi iletişimi geliştirerek sorunların kısa sürede çözümlenmesini sağlıyor. Bunun gibi faktörler günümüzde projelerinin başarı oranının %80’lere ulaşmasını sağlamıştır. Scrum yaklaşımı en popüler Agile yöntemidir. Büyük ve karmaşık projelerde oldukça sık kullanılmaktadır. Projede bütün parçalanır ve yinelemeye dayanır. Müşteri ile sürekli bir iletişim halinde proje süreci devam ettirilir. Peki bu Scrum yönteminin faydaları nelerdir ? İlk olarak proje oldukça net ve anlaşılırdır. Bu sayede zaman kazanılır. Proje başarı ile sonuçlanır. Bütün ekibin, projenin tüm gidişatından haberdar olması ekip içindeki iletişimi artırır. Müşteri ile ekip arasında güven oluşur ve dolayısıyla projenin başarılı sonuçlanması beklenir. Scrum yöntemini başarıyla uygulanmasını sağlayan kişiye scrum master denir. Bu kişinin sorumluluğu ekibinin başarıya ulaşabilmesidir. Ayrıca ekibin karşı karşıya geldiği problemlerde de yardımcı olabilmektedir.

Furkan Çankaya

190601047