Yazılım projeleri ve Agile(Çevik) Yöntemlerin Uygulanması

Dr. Serdar Biroğul
sbirogul@yahoo.com

Yazılım projeleri ve bunlara yapılan yatırımlar gelişen teknolojinin de etkisiyle artmaktadır. Artık kodlama ve yazılım geliştirme olgusu büyük ve kapsamlı projeler olmaktan çıkıp her yerde olan bir unsur olmuştur. Yazılım projesi diğer proje türlerinde çok daha farklı yapıya sahiptir. Gelişen teknolojinin hızlı değişimi ve tamamen insan beyin emeğini gerektiren yapısı ve öngörülemeyen unsurların çok olmasından dolayı yazılım projeleri ve bunların yönetimine ait süreçlerin iyi seçilmesi ve yönetilmesi gerekmektedir. Dünyada gerçekleştirilen yazılım ve teknoloji tabanlı projelerin büyük çoğunluğu istenen hedeflere ulaşamamaktadır. Bu durum teknolojinin ve proje sahibinin isteklerinin proje sürecinde değişmesi, proje başlangıcındaki isteklerini tam olarak belirleyememesinden dolayı yanlış analizlerin oluşturulmasından kaynaklanmaktadır. Aşağıdaki karikatür yazılım projelerine genel bakışın şaka yollu anlatımı göstermektedir. Bu karikatür her ne kadar komik olsa da aslında şu anki gerçek durumu göstermektedir.

Yazılım Geliştirme Metodolojisi/Yaşam Döngüsü yazılım proje sürecinin geçirdiği tüm aşamaları içermektedir. Yazılımın üretildiği aşamadan itibaren işlevsellik ile gereksinimleri sürekli değişmekte ve bu değişiklikler de yazılımın değişmesine veya genişlemesine neden olmaktadır. Bu döngü doğrusal ve tek yöne doğru ilerleme şeklinde değil herhangi bir aşamada geriye dönmenin mümkün olmasını gerektirmektedir. Yazılım geliştirme metodoloji çeşitlerine bakıldığında şelale modeli, artımsal model, döngüsel model ve spriral modeller öne çıkmaktadır. Bu metodolojilerle birlikte günümüzde yazılım projesi yürüten ve kalitenin müşteri memnuniyeti olduğunu benimsemiş tüm firma ve ekiplerde etkin ve hızlı geri dönüşlere sahip metodoloji olarak Agile/Çevik metotlar gündeme gelmiştir.

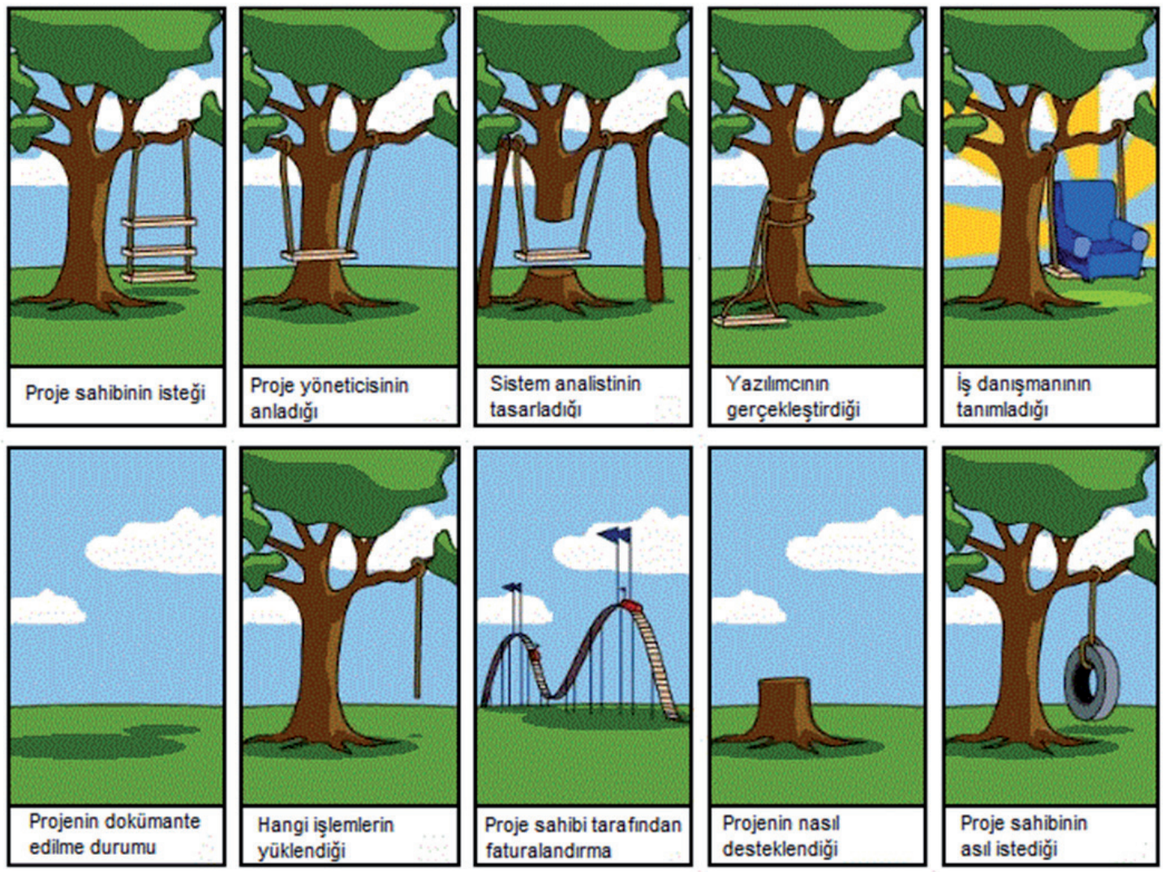
Şelale modelinde yazılım projesine ait aşamalar en az birer kez tekrarlanarak gerçekleştirilmektedir. Çok iyi tanımlanmış ve üretimi az zaman gerektiren projelerde uygulanmaktadır. Şelale modelinin kullanımında dikkat edilmesi gereken önemli hususlardan birisi, aşamalar arasında geri dönüşlerin olmasına karşın analiz aşamasında proje tüm detayının tasarıma yansıtılabilmesi için müşteri ve sistem gereksinimlerinin en ince ayrıntısına kadar belirlenmesi gerekmektedir. Tasarım aşaması da yazılımın tüm gereksinimlerini karşılayacak şekilde detaylı bir çalışma gerektirmektedir. Tüm detaylı çalışmalara rağmen özellikle uzun süren projelerde müşteri ve ilerleyen teknolojiden dolayı değişikliklerin olması kaçınılmazdır. Kodlama ve test aşamlarında olacak bu değişikliklerin sisteme yansıtılması hem proje süresini hem de maliyeti arttırmaktadır. Bu modelde, yazılımın son kullanıcıya ulaşması genel olarak uzun bir zaman almaktadır. Bu durum, hem proje sahibi hem de projenin geliştirildiği kurumun üst yönetiminde belirgin bir memnuniyetsizlik yaratabilmektedir.

Artımsal modelde proje parçalara ayrılmakta ve her bir parça ayrı olarak yönetilmektedir. Yazılım proje süreci bir çok döngüden oluşur ve her döngü sonucu bir çıktı elde edilir. Bu döngülerin devamında projede bir sonraki adıma geçilerek süreç işletilir. Böylelikle teknolojik ve yazılım alanında ortaya çıkan gelişmeler sınırlı da olsa projede uygulanabilir. Her bir döngü sonucunda elde edilen bilgiler aynı zamanda çalışanlar içinde öğrenme sürecini oluşturur. Burada dikkat edilmesi gereken husus öğrenme sürecinin artması projenin ilerleyişinde sorun oluşturmaktadır.

Döngüsel model artımsal modele çok benzemektedir. Artırımsal modelde her döngü tasarım, kodlama, test ve entegrasyon süreçlerini içerir. Döngüsel model ise planlamadan başlayarak tüm proje süreçlerini içeren döngülerden oluşur. Döngüsel modelde döngü sayısı ve süresinin doğru belirlenmesi gerekmektedir. Eğer döngü sayısı ve süresi doğru belirlenemezse sondaki döngüler tekrarlayan işler haline gelmekte ve zaman yetersizliği ortaya çıkmaktadır.

Spiral model risk analizine ve prototip üretmeye üzerine kurulmuş bir metodolojidir. Her döngü öncesi, içinde olunan fazın riskleri analiz edilmektedir. Bir sonraki döngü içinde yeniden aynı süreç uygulanmaktadır. Böylece yazılım geliştirilirken oluşan riskler gözönünde bulundurulur. Spiral model önceden geliştirilmiş yazılımların yeniden kullanıldığı projelerde mevcut bilgiler kullanılarak risk analizi ve geliştirmeler daha kolay yapılabilir. Spiral modelde her döngünün başında yapılan risk analizi sebebiyle zaman ve maliyetin kolayca tahmin edilebilmektedir. Bu model büyük projeler için uygun olsa da uygulamanın karışık olduğu durumlarda ve süreçlerde tecrübe gerekliliğini gerektirmektedir.

Bu modellere bakıldığında proje sahibi/kullanıcı/müşteri, yazılımın geliştirilme aşamasında süreç içerisinde genellikle yer almaz ve bu durum yazılım tamamlandıktan sonra geri dönüşlerin artmasına neden olmaktadır. Bu geri dönüşler, yazılım geliştirme maliyetini büyük oranda yükselten bir durumdur. Tasarım aşamasında fark edilen hata ve eksiklikler küçük bir maliyet ile giderilebilir. Ancak, bu hata ve/veya eksiklikler entegrasyon veya bakım aşamalarında farkedilirse bunları gidermenin maliyeti katlanarak artacaktır. Yazılım geliştirme sürecinin ilerleyen safhalarında müşteri tarafından veya ilerleyen teknolojinin zorlayıcı etkilerinden dolayı değişim istekleri maliyet ve zaman kaybına neden olmaktadır. Geleneksel yazılım geliştirme yaklaşımları projenin başlangıcında yeterince kapsamlı çalışılarak ihtiyaçları eksiksiz tespit etmeyi ve bu sayede projenin ileri safhalarında ortaya çıkabilecek olası değişimleri önlemeyi esas almaktadır. Ancak hızla değişen ve gelişen teknoloji proje sahibi/müşteri gereksinimlerinin de proje gelişim sürecinde değişmesine neden olmaktadır. Bu durumdan dolayı proje başlangıcında, proje sahibi isteklerinin eksiksiz tespit edilmesi imkansız hale gelmiştir.



Değişen proje sahibi ihtiyaçlarına kısa süre de cevap verebilecek ve az maliyetle karşılanabilecek olan Agile/Çevik Yazılım Geliştirme modeli etkinliği göstermektedir. Agile/Çevik Bildirimi [1];

- Bireylere ve etkileşimlere, süreçlerden ve araçlardan daha fazla değer verir.
- Çalışan yazılıma, kapsamlı dokümantasyondan daha fazla değer verir.
- Müşteri işbirliğine, kontrat görüşmelerinden daha fazla değer verir.
- Değişime yanıt vermeye, bir planı izlemekten daha fazla değer verir.

Agile/Çevik Yazılım Geliştirme Sürecinde ise;
- En büyük öncelik, yazılımı zamanında teslim ederek Kalite sürecinin temel unsuru olan müşteri memnuniyetini sağlamaktır.
- İsteklerdeki değişiklikler, geliştirmenin son aşamasında olsa dahi hoş karşılanır ve gerektirdikleri sağlanmaya çalışılır. Ancak istenen değişiklik yazılımda köklü farklılıklara/değişimlere neden olacaksa bu değişiklik isteğinin proje sahibine tam olarak anlatılması ve değişikliğin farklı açılardan ele alınması sağlanır. (Uygulanan Çevik süreçlerde, değişimden proje sahibinin rekabetteki avantajına olacak konulardan yararlanılır.)
- Çalışan yazılım sık sık, bir kaç haftadan bir kaç aya kadar sürelerde kısa bir zaman diliminde teslim edilerek ilerlemeler sağlanır.
- Proje sahibi ve proje ekibi proje boyunca sık sık, mümkünse günlük olarak birlikte çalışılır. Bu toplantılar proje özelliğine göre haftalık olarak da yapılmaktadır.
- Projeler istekli bireylerle yapılmalıdır düşüncesiyle onlara ortam ve ihtiyaç duydukları desteği verir ve onların yapacakları işe güvenilir.
- Dışarıdan, proje geliştirme takımındakilere ve birbirleri arasındaki bilgi taşınmasının en etkili ve etkin yolu yüz yüze iletişimdir.
- İstenilenleri gerçekleştiren yazılım, gelişmenin birincil ölçütüdür.
- Çevik süreçler sürekli devam ettirilebilir. Destekleyenler, geliştiriciler ve kullanıcılar sabit bir hızı sağlayabilmelidir.
- Teknik üstünlüğü ve iyi tasarıma sürekli dikkat etmek çevikliği arttırır.
- Basitlik esastır.
- En iyi mimariler, istekler ve tasarımlar kendi kendine örgütlenen takımlardan çıkmaktadır.
- Düzenli aralıklarla takım nasıl daha etkili olabileceklerini düşünür ve davranışlarını ona göre ayarlamakta ve düzenlemektedir.

Agile/Çevik Yazılım Geliştirme Süreci ile günümüz teknolojik yapısında proje sahibi memnuniyetini az maliyet ve hızlı çözüm gerçekleştirilebilir. Proje sahibinden gelen geri beslemelere uygun olarak geliştirmeye devam edilir. Geliştirme artımlı, işbirlikçi, doğrudan

(değiştirilmesi kolay, iyi dokümante edilmiş) ve uyumlu (son dakika değişiklikleri sağlayabilen) özellikleri sağlayan yazılım geliştirme süreci ile sağlanmaktadır.

Değişiklik yönetimi proje sahibinin memnuniyetinin sağlanması için etkin olarak kullanılmaktadır. Büyüyen yazılım projelerinde artan karmaşıklık seviyesinin düşürülmesi için proje, alt projelere bölünür. Böylelikle karmaşıklık seviyesi düşürülür. Yineleme sonunda çalışır programcıklar elde edilir. Projenin başlangıcından itibaren giderek büyüyen bu parçacıklar proje sahibine sunularak memnuniyetinin arttırılması sağlanır. Değişiklikler arasında gerçekleştirilen fikir alış verişleri proje sahibi/müşteri odaklı olunmayı sağlar.

Proje sahipleri çoğunlukla proje başlangıcında tam olarak ne istediklerine dair kesin fikirleri olmadığından projenin gelişim süreci proje sahibinin değişiklikleri ile şekillendirilir. Bu sayede proje sahibi/müşteri odaklı ve esnek yapıyla memnuniyet en üst seviyede tutulmaya çalışılır.

Agile/Çevik metodoloji yaklaşımı ile proje riskleri azaltılır, başarı arttırılır ve verim yükseltilir. Parçalı program geliştirme ile proje sürecinde yazılım hızla şekillenmekte ve proje başlangıcında fark edilmeyen riskler ciddi sorunlara yol açmadan giderilmektedir. Yazılım projelerindeki değişikliğin kaçınılmaz olmasının farkındalığıyla Agile/Çevik sürecinin uygulanması doğru bir yaklaşım olarak gözükmektedir. Etkinliği ve yöntemi onaylanmış olunan diğer yazılım geliştirme metodolojilerinin uygulanmasının zorunlu olduğu durumlarda yazılım geliştirme sürecinin Agile/Çevik metotlar ile desteklenmesi uygun olacaktır.

[1] http://agilemanifesto.org