ÇEVİK YAZILIM GELİŞTİRME

“AGILE”

acm YAZILIM ÇÖZÜMLERİ

KEEP IT SIMPLE

İÇİNDEKİLER

ÖNSÖZ

Araştırmalara göre ülkemizdeki yazılım projeleri yönetimsel eksiklilerden dolayı ancak %50 başarı ve memnuniyet ile tamamlanabilmektedir. Ne yazık ki, bu ciddi iş gücü kaybı ve bu verimsiz üretim ile Türk yazılım sektörünün dünya devleriyle yarışabilmesi pek mümkün değildir.

Geçmişe bakarsak, Avrupa ve Amerika'daki büyük şirketler de bu dönemi yaşamışlar, daha verimli projeler üretmek üzere çeşitli yöntemler denemişler ve çoğu şirket yönetimde ve uygulamada en başarılı buldukları "Agile" (çevik) yazılım metodolojilerini benimsemişlerdir. Bu metodolojiler sayesinde, artan verimlilik ve esneklik ile projelerin kalitesini arttırmış ve başarı oranlarını %80'lere çıkartmayı başarmışlardır.

"Agile" (çevik), dünya üzerinde kabul edilen yöntemler arasında en hızlı ve güvenli proje geliştirme metodolojisidir. Halen, birçok şirket tarafından hızla kullanıma geçirilmektedir. Ancak, ülkemizde henüz yaygın olarak kullanılmamakta ve birçok şirketin öz kaynakları ve zamanı çeşitli aksaklıklar nedeniyle boşa harcanmaktadır.

ACM olarak, Türkiye'de yaygınlaştırmaya çalıştığımız bu yönetimsel modeller ile ülkemizdeki yazılım geliştirme ve proje yönetimi kültürünün daha başarılı sonuçlar üreten bir hale gelmesi için çalışmalar yürütmekteyiz. Türkiye'de belirli bir yazılım proje kültürünün oluşturularak, ülkemizin, ürettiği yazılımlarla dünyada ön sıralara girmesini hedeflemekteyiz.

Bu doğrultuda ücretsiz olarak dağıtmakta olduğumuz bu kitapçığı, Türk yazılım sektörünü bilinçlendirmek, çevik yazılım yöntemlerinin tanınırlığını arttırmak ve yazılım kalitemizin artırılmasına katkıda bulunmak için sizlere sunmaktayız.

Tamamen kendi imkânlarımız ve iyi niyet doğrultusunda hazırlanan bu kitapçık ile ilgili olarak, her türlü önerileriniz, eklemek istedikleriniz veya varsa düzeltilmesini düşündüğünüz hususlar için bizimle irtibata geçmenizi rica ederiz.

Saygılarımızla,

ACM Yazılım Çözümleri

GİRİŞ

Yazılım geliştirmek, yeni bir ürün yaratmaya benzer ve yazılım projeleri süreç içerisinde değişerek şekillenir. Projelerde karşılaşılacak olan birçok problem, yazılım projesi doğası gereği başlangıçta öngörülebilir olmaktan uzaktır. Bu yüzden, yazılım geliştirirken alışılagelmiş klasik seri üretim metotları birçok koşulda başarılı çözümler üretmekten uzaktır ve başarı için daha esnek metodlara ihtiyaç duyulmakadır.

Yüzeysel bir açıdan, seri üretim yaklaşımının ve yeni bir ürün geliştirmenin doğalarını karşılaştıracak olursak:

	Seri üretim yaklaşımı	Yeni ürün geliştirmek
Gereksinimler:	proje başlangıcında belirli	değişken
Değişiklik:	kısıtlı	en üst seviyede

Anlaşılacağı üzere, seri üretim yaklaşımı proje başlangıcında detaylı bir analiz sonucu gereksinimlerin belirlenmesi ve sonrasında da bu gereksinimler temel alınarak projenin sonlandırılmasına yönelik çalışırlar. Ancak, yeni bir ürün geliştirirken (yazılım geliştiriken) detaylı bir analiz ile birlikte tüm gereksinimlerin proje başlangıcında belirlenmesi pek mümkün değildir. Yeni bir ürün geliştirirken istekler ve bunun doğrultusunda gereksinimler sürekli değişim gösterme eğilimindedir.

Gerçek dünya koşullarında yazılım projelerine bakacak olursak;

- Proje başlangıcında müşteriler, tam olarak ne istediklerinden yüzde yüz emin değildirler.

- Müşteriler isteklerini net bir şekilde dile getirmekte zorlanırlar.

- Müşterilerin asıl istekleriyle ilgili detaylar proje başında fark edilebilir değildirler, süreç içerisinde yüzeye çıkarlar.

- Müşteriler isteklerinin gerçekleştiğini gördükçe, proje gözle görülür sonuçlar üretmeye başladıkça, isteklerinde değişiklik yapma eğilimindedirler.

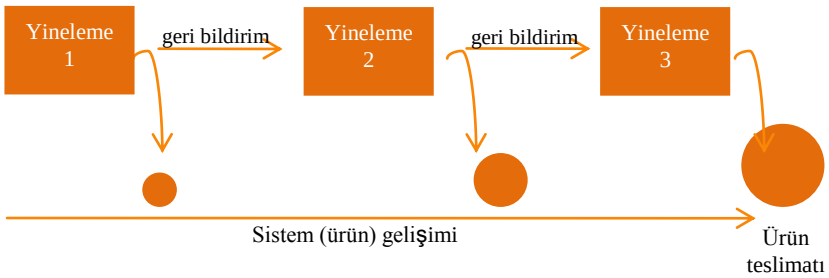
- Dış koşullar değişim içerisindedir ve projelerin bunlardan etkilenerek değişime uğramaları kaçınılmazdır.

İşte bu yüzden, yazılım geliştirmek müşteri odaklı olmayı ve değişikliklere uyum sağlayabilmeyi gerektirir. Bunun aksine klasik yöntemlerle ele alınan yazılım projeleri değişime uyum sağlamakta güçlük çekerek kırılacak ve istenen başarıyı elde etmekte zorlanacaktır.

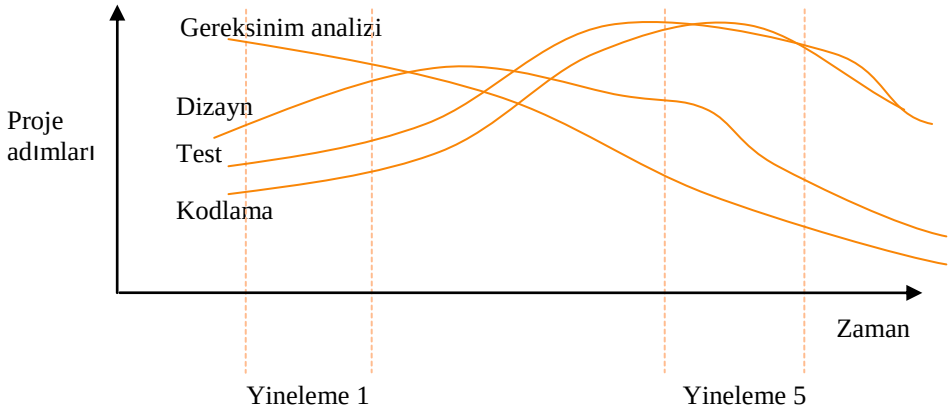
Sonuç olarak, yazılım projeleri çoğunlukla yeni bir ürün yaratmaktır ve yeni bir ürün yaratma sürecinde değişiklikler kaçınılmazdır. Yazılım projelerinin bu gerçeklikler doğrultusunda ele alınarak yönetilmeleri gerekir.

TEKRARLANAN YAZILIM GELİŞTİRME METODU

Tekrarlanan yazılım geliştirme metodu, yazılım projelerinin sıralı yinelemelerle oluşturulduğu bir yazılım gelişirme metodolojisidir. Tekrarlanan yazılım metodolojilerinde yazılım projeleri kendi içerisinde parçalara bölünerek ele alınır. Bu metodolojilerde, oluşturulan her bir parça kendi içinde küçük bir proje gibi düşünebilinir. Asıl proje hedefi, bu küçük projelerin birbirine eklenmesiyle elde edilmektedir.



Tekrarlanan yazılım geliştirme metodundaki her bir yineleme kendi içinde bir yazılım projesinin gereksinim analizi, dizayn, kodlama ve test gibi adımlarını bünyesinde bulundurur. Ancak, her bir yinelemenin içerisindeki bu proje adımlarının ağırlığı değişkendir. Öyle ki, ilk yinelemelerde gereksinim analizi kodlamaya göre daha yoğunken ileriki yinelemelerde durum değişerek gereksinim analizinin içeriği küçülmekte ve kodlamanın ağırlığı artmaktadır.

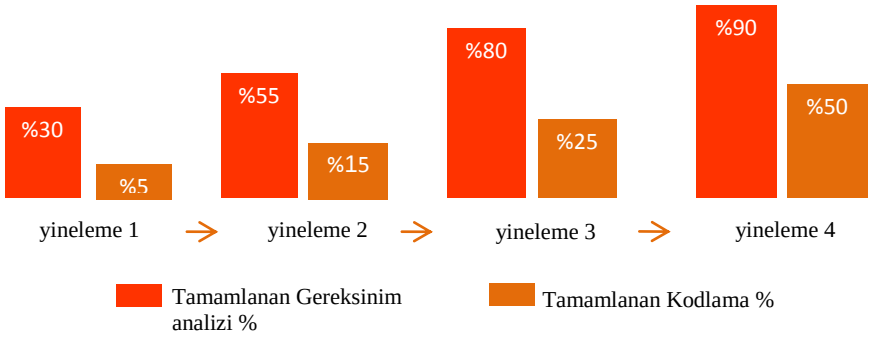


Bir projedeki yineleme sayısı projeden projeye ve kullanılan modellere (XP, Scrum, vs) göre değişiklik gösterir. Ancak, bugün yaygın olarak kullanılan modellere bakıldığında yineleme uzunluklarının genellikle 1-6 hafta arasında değişiklik gösterdiği görülmektedir.

Araştırmalar göstermektedir ki, yazılım projeleri büyüdükçe karmaşıklıkları artmakta ve projelerin başarılı olma oranları azalmaktadır. Diğer yandan projeleri parçalara ayıran tekrarlanan yazılım geliştirme metodolojileri, projelerin karmaşıklığını ve riski azaltmakta, verimliliği ve başarı oranlarını arttırmaktadır.

Tekrarlanan yazılım geliştirme metodunda projenin küçük parçalara ayrılmasının yanında, bu yinelemer arasındaki geri bildirimler de büyük önem taşımaktadır. Geri bildirimler, proje gelişim sürecinin esnekliğini ve proje başarısını arttırmaktadır. Projeyle hedeflenen sonucun tam olarak anlaşılması ve elde edilebilmesi için geri bildirimler oldukça kritikir.

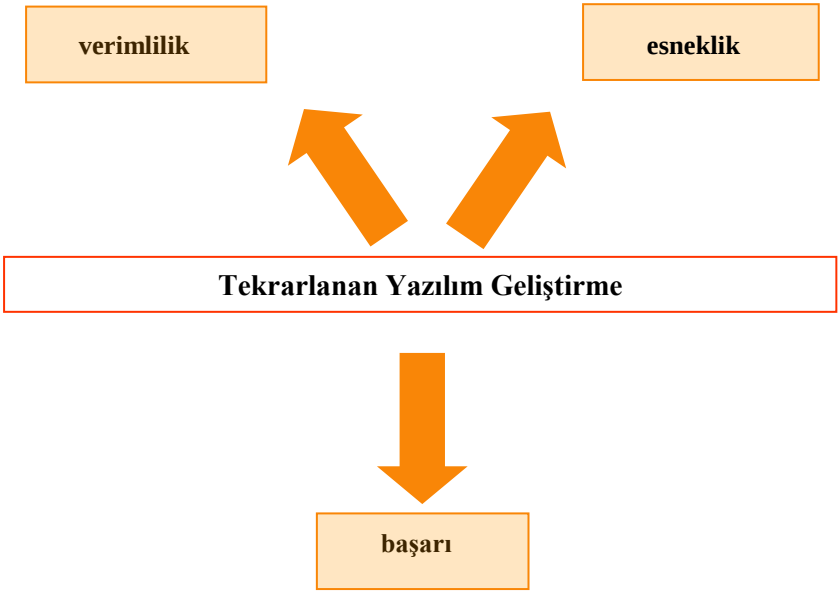
Her ne kadar bu metodolojiyle esneklik destekleniyor olsa da, esneklik sınırsız değildir ve değişiklik oranı proje ilerledikçe azalmalıdır. Aksi takdirde, proje sonlandırılamayacak ve kargaşa ortamı doğacaktır.



Özetleyecek olursak, tekrarlanan yazılım geliştirme metodu:

- ana projeyi parçalara bölerek karmaşıklığı azaltmakta,
- bünyesinde bulundurduğu geri bildirimler sayesinde değişimi desteklemekte,
- riskleri azaltmakta,
- ve projelerin başarı oranını arttırmaktadır.

İşte bu yüzden, artık tüm dünyada birçok yazılım projesi bu yöntem ile geliştirilmektedir.


verimlilik
esneklik
Tekrarlanan Yazılım Geliştirme
başarı

ÇEVİK YAZILIM GELİŞTİRME METODU

Çevik yazılım süreçleri, 1950'lerdeki üretim alanında verimliliğin artırılması için geliştirilen yalın yaklaşımların, yazılım sektöründe bir uzantısı olarak ortaya çıkmıştır. Yazılım dünyasında çeşitli çevik yaklaşımlara 1970'lerden itibaren rastlanabilmekle birlikte, çevik yazılım metodolojilerinin kullanımı 1990'larda hız kazanmış ve geçtiğimiz son 7-8 yıl içerisinde de tüm dünyada başarılarını kanıtlayarak popülaritesini arttırmıştır. Şu anda, dünyadaki birçok yazılım şirketinde ve birçok yazılım projesinde yazılımlar, çevik yaklaşımlarla geliştirilmektedir.

Çevik yazılım geliştirme metodu, tekrarlanan yazılım geliştirme metodu taban alınarak geliştirilmiş, sık aralıklarla parça parça yazılım teslimatını ve değişikliği teşvik eden bir yazılım geliştirme metodolojisidir. Çevik geliştirme;

- değişimi,
- takım içerisindeki iletişimin arttırılmasını,
- parça parça yazılım teslimatını,
- test odaklı yazılım geliştirilmesini,
- ve uyumlu planlamayı teşvik eder.

Çevik Yazılım Geliştirme Manifestosu

2001 yılında, dünyanın önde gelen çevik yazılım geliştiricileri (XP, Scrum gibi metodolojilerin yaratıcıları) ortak bir zeminde buluşabilmek adına bir araya gelerek "çevik yazılım geliştirme manifestosu" nu ve "çevik yazılımın prensipleri" ni yayınlamışlardır. Böylelikle çevik metotların projelere genel bakış açıları ifade edilmiştir.

Bu manifestoda;

Bireyler ve aralarındaki etkileşimlerin, kullanılan araç ve süreçlerden;
Çalışan yazılımın, detaylı dokümantasyondan;
Müşteri ile işbirliğinin, sözleşmedeki kesin kurallardan;
Değişikliklere uyum sağlayabilmenin, mevcut planı takip etmekten;

daha önemli ve öncelikli olduğu belirtilmektedir.

Çevik Yazılımın Prensipleri:

1- İlk öncelik, sürekli, kaliteli yazılım teslimatıyla müşteri memnuniyetini sağlamaktır.

2- Proje ne kadar ilerlemiş olursa olsun değişiklikler kabul edilir. Çevik yazılım süreçleri değişiklikleri müşteri avantajına dönüştürürler.

3- Mümkün olduğunca kısa zaman aralıklarıyla (2-6 hafta arası) çalışan, kaliteli yazılım teslimatı yapılır.

4- Analistler, uzmanlar, yazılımcılar, testçiler vs. tüm ekip elemanları bire bir iletişim halinde, birlikte çalışırlar.

5- İyi projeler motivasyonu yüksek bireyler etrafında kurulur. Ekip elemanlarına gerekli destek verilmeli, ihtiyaçları karşılanarak proje ile ilgili ekiplere tam güvenilmelidir.

6- Ekip içerisinde kaliteli bilgi akışı için yüz yüze iletişim önemlidir.

7- Çalışan yazılım, projenin ilk gelişim ölçütüdür.

8- Çevik süreçler mümkün olduğunca sabit hızlı, sürdürülebilir geliştirmeye önem verir.

9- Güçlü teknik alt yapı ve tasarım çevikliği arttırır.

10- Basitlik önemlidir.

11- En iyi mimariler, gereksinimler ve tasarımlar kendi kendini organize edebilen ekipler tarafından yaratılır.

12- Düzenli aralıklarla ekipler kendi yöntemlerini gözden geçirerek verimliliği arttırmak için gerekli iyileştirmeleri yaparlar.

Çevik Metodolojilerin Getirileri:

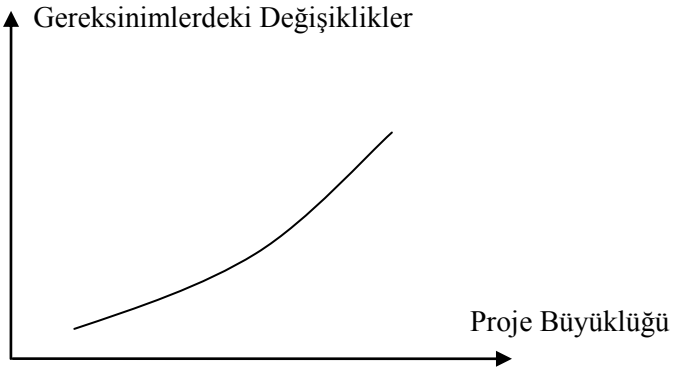
Düşük Risk:

Tekrarlanan yazılım geliştirme metotları, proje risklerini azaltıp başarıyı arttırmakta ve hata oranlarını düşürerek verimliliği yükseltmektedir. Bunun arkasında yatan en temel etken, daha projenin başlarında geliştirilen program parçacıkları sayesinde, proje ekibinin yetkinliklerinin ve projede kullanılan her türlü araçların önceden denenerek eksiklerinin görülebilmesidir. Ayrıca, parçalı geliştirme süreci içerisinde proje hızlı olarak şekillenmekte ve proje başlangıcında fark edilemeyen riskler ciddi sorunlara yol açmadan önce görülebilir hale gelmektedir.

Değişimin Teşvik Edilmesi:

En başta belirttiğimiz gibi, yazılım projelerinde değişiklik kaçınılmazdır. Orta çaplı projeler dahi proje süresince başlangıçlarına göre %30 oranlarında değişeme uğramaktadır. Bu nedenle, değişim yazılım projelerinin doğasıdır ve bu gerçeklik çevik metodolojilerin de üzerinde önemle durduğu bir etkendir.

Çevik metodolojiler değişime karşı gelmek yerine değişimi müşteri avantajına dönüştürmeye yönelik olarak çalışırlar. Çevik metodolojiler, önerdiği parçalı yazılım üretimi ve her adımdaki güçlü bilgi alış verişiyle, değişim gereksinimlerinin mümkün olduğunca projelerin başlangıç adımlarında fark edilmesini ve projenin değişime hızlı bir şekilde adapte olabilmesini sağlarlar.



Karmaşıklığın Yönetimi:

Yazılım projelerinin hacimleri büyüdükçe karmaşıklıkları artar ve buna bağlı olarak hata oranları da artar. Çevik yöntemler ise projeleri, daha kolay yönetilebilir küçük parçalara bölerek ele alırlar. Böylece, projelerin büyüklüğü ne olursa olsun, küçük parçalara ayrılarak ele alınan projeler için karmaşıklık en düşük seviyeye indirilir.

Sürekli Yazılım Teslimi:

Çevik metodolojilerde her yineleme sonunda çalışır bir programcık meydana getirilmektedir. Projenin başlarından itibaren devam ederek büyüyen bu parçacıklar, müşterilere elle tutulur aşamalar olarak sunulmakta ve bu da müşteri memnuniyetini arttırmaktadır. Yapılan araştırmalar, müşterilerin tamamlanmış ama çalışmayan programlar yerine, çalışır durumda ama tamamlanmamış programları tercih ettiklerini göstermiştir.

Yüksek Kalite:

Tekrarlanan yazılım geliştirme metotlarının bünyesinde, test odaklı yazılım geliştirme mantığı bulunmaktadır. Proje başından başlayarak tüm test süreçlerinin yazılım geliştirme süreci içerisinde beraber yürütülmesi sayesinde, hatalar büyümeden fark edilerek hızlı bir şekilde düzeltilebilmektedir.

Proje hacimleri büyüdükçe hata oranları da artmaktadır. Ancak çevik metodolojiler, projeleri parçalara bölerek ve her bir parçayı kendi gelişimi içerisinde de test ederek hata oranlarının düşürülmesini sağlamaktadır.

Müşteri ihtiyaçlarına daha iyi cevap veren çözümler:

Yinelemeler arasında gerçekleştirilen fikir alış verişleri, çevik yöntemlerin değişikliğe olan yatkınlığı ve müşteri odaklı yaklaşımları sayesinde, projeler, müşteriler ile birlikte değişerek gelişmekte ve proje sonunda da müşteri ihtiyacını en iyi derecede karşılayabilecek programlar ortaya çıkmaktadır.

Müşterilerin çoğunlukla proje başlangıcında tam olarak ne istediklerine dair net birer fikirleri yoktur ve istekler, projenin gelişim süreci ile birlikte şekillenmektedir. Bu gerçeklik karşısında, çevik yöntemler müşteri odaklı ve esnek yapısıyla, müşteri memnuniyetini en üst seviyede tutmayı başarabilmektedirler.

REFERANSLAR

www.agilealliance.com

www.scrumalliance.org

www.apln.org

Agile&Iterative Development, Craig Larman, Addison-Wesley 2007

Extreme Programming Explained, Kent Beck with Cynthia Andres, Addison-Wesley 2007

Agile Project Management With Scrum, Ken Schwaber, Microsoft 2003