

Çevik Süreç Ve Türkiye

Göksel ÜÇER¹, Yüksel YURTSEVER²

¹ Netsis Yazılım, Urla, İzmir, goksel.ucer@netsis.com.tr

² Vestel, Gazıemir, İzmir, yukselyurtsever@gmail.com

Özet. Bilişim ve yazılım sektörünün dünyadaki parasal payı büyük olmasına rağmen Türkiye’de henüz istenilen noktaya gelememiştir. Yazılım sektörü, Türkiye’de uzun yıllar hayatını sürdürse de yazılımın projelerindeki temel sorunlardan uzaklaşamamıştır. Yazılım projeleri, genelde ertelenen, iptal edilen ya da olması gereken takvimde eksik fonksiyonlarla çıkarılan projeler olarak bilinmektedir. Yazılım projelerinin yönetilmesindeki en temel sorunlar, Türkiye’deki yazılım evlerinin de genel sorunlarıdır. Sorunları aşmada insan ve yazılım geliştirme yöntemi faktörlerinin önemi büyüktür. Kurumların uygun yöntemi seçmesinde geleneksel ve çevik süreç arasındaki farkları iyi bilmesi gerekir. Farkların göz önüne serilmesi, çevik sürece ait ortak prensiplerin aktarılması, çevik süreçte en yaygın kullanılan XP hakkında genel bilginin verilmesi ve çevik sürecin Türkiye’deki durumunun özetlenmesi bu çalışmayı ortaya çıkarmıştır.

1 Giriş

Bu çalışmada dünyada ve Türkiye’de bilişim ve yazılım sektörlerine ait rakamsal veriler incelenmektedir. Bu verilerin yorumlanması, Türkiye’de yazılım sektörünün ne kadar önemli olduğunu, nasıl geliştirileceğini ve yazılım projelerini yönetirken hangi yöntemlerin kullanılmasına ışık tutmaktadır.

Yaygın olarak kullanılan geleneksel ve çevik yazılım geliştirme yöntemleri arasındaki farklar ortaya çıkarılmıştır. Kurumlar yazılım projelerini yönetirken birçok zorluk yaşamaktadır. Bu zorlukları, iki yöntem arasındaki avantaj ve dezavantajları kullanarak aşabilirler.

Daha sonraki bölümler incelediğinde çevik sürecin “insan merkezli” bir yaklaşım olduğu anlaşılacaktır. Kurumların ve organizasyonların çevik süreci tercih ederken bu yaklaşımı göz ardı etmemesi gerekir. Çevik sürecin en yaygın kullanılan yöntemlerinden birisi XP’dir. XP’nin projelere uyumluluğu, riskleri ve sınırları detaylı olarak anlatılmaktadır.

Yazılım projelerini yönetmek zor ve kalıp yöntemleri olmayan bir işdir. Bu tür projelerin başarılı yönetilmesi, en önemli iki faktörün (insan ve yöntem) uyum içinde yönetilmesi ile mümkün olacaktır.

2 Bilişim Ve Yazılım

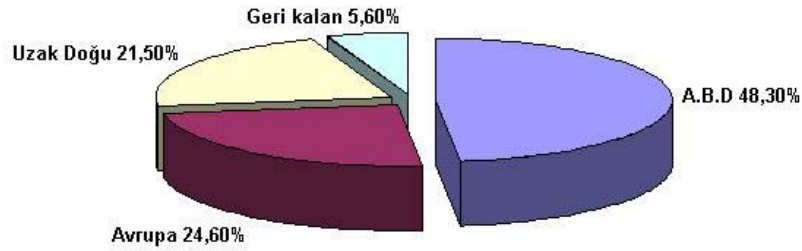
2.1 Sektör Hakkında

Bilişim teknolojilerinde yer alan firmalar üç ana gruba ayrılmaktadır.

- Bilgisayar ve telekomünikasyon ürünleri yapan firmalar.
- Yazılım üreten firmalar.
- Ürünlerin satışını ve hizmetini yapan firmalar.

Yazılım ve hizmet dünya bilişim sektörünün %38'ini oluşturmaktadır. İki sektörün 1997 yılındaki büyüklüğü yaklaşık 1 trilyon \$. Pazarın ortalama %25'lik kısmı genel programlardan, %20'lik kısmı ise yazılım hizmetleri ve danışmanlıklardan oluşmaktadır.

Şekil 1'de görüldüğü gibi, Türkiye'nin içinde bulunduğu bölge payı çok düşük orandadır.



Şekil 1. Sektörde bölgesel dağılım [1]

Dünyada yazılıma bakıldığında, 2001 yılı verilerine göre Hindistan, 6,2 milyar \$ ihracat, İrlanda, yılda 950 bilgisayar mühendisi ve İsrail ise kendi yazılım paketlerini üreten önemli ülkeler arasında yer almaktadır.

2.2 Türkiye'de Bilişim

1995 ve 2001 yılları arasında Türkiye'de bilişim sektörü %17,8 oranında büyüme göstermiştir ve 2002 yılında 1,4 milyar\$ satış yapabilecek duruma gelmiştir. Yazılım sektörü ise, aynı yıllarda %9,5 oranında büyüme göstermiş ve 2002 yılında 215 milyon\$ satış yapabilir duruma gelmiştir [1]

Yazılım yapan firmaların ürünleri incelediğinde, ürünler iki ana grup altında toplanmaktadır [1].

Altyapı

- İşletim sistemleri
- Güvenlik sistemleri
- Ağ teknolojileri
- Veri tabanları

Uygulama

- ERP (Enterprise resource planning)
- CRM (Customer relationship management)
- LOB (Line of business) (*Turizm, hastanecilik, otelcilik, vb.*)

Sektörde çok az firmanın olması ve yüksek maliyet gereksiniminden dolayı Türkiye'nin alt yapı alanına girmesi zor olacaktır. Türkiye, gelişmekte olan ülkelerin yaptığı gibi uygulama alanında yazılım yapabilir.

Tablo 1'de Türkiye'de uygulama alanında faaliyet gösteren firmaların satış rakamları gösterilmektedir.

Tablo 1. Türkiye'deki ERP sektörünün satış rakamları [2]

Firma	Kartlar
SAP	8
IBM Türk	5,7
Logo	4,6
Oracle	1,8
Meteksan	1,5
Byte&Muhsinoğlu	1,5
IAS	1,5
Bilişim Sanayi	1,3
Koç Sistem	1,1
Diğerleri	11,9
Toplam	38,9

Yazılım Sektörünün Sorunları

Türkiye’de yazılım sektörünün en önemli sorunlarından biri %78 oranında yasa dışı (korsan yazılım) kullanımının olmasıdır. Bu oran, uygulama alanında faaliyet gösteren firmaların satış rakamlarını ciddi derecede etkilemiştir. 1995 yılında yürürlüğü giren yasa ile bu oranın aşağılara çekilmesi hedeflenmektedir. Son yıllarda bu yasaya ait yapılan düzenlemeler, denetimlerin sıklaştırılmasına ve firmaları yasal olmayan ürünlerini güncellemeye zorlamıştır.

3 Çevik Süreç

Çevik kelimesi, hızlı düşünmek ve akıllı bir yolu tercih etmek olarak ifade edilebilir. Geleneksel yazılım geliştirme süreçleri, projelerde değişen gereksinimleri karşılamak ve beklenen kalite faktörlerine erişmek açısından çok katı ve ağırdır. Çevik yazılım geliştirmede ise, süreçlerin katı kurallarla yönetilmesi yerine amaca yönelik verimli ve etkin pratiklerin yapılması esastır.

Çevik süreçte yazılımcılar ile iş mantığı konusunda uzman kişilerin belgelerle iletişim kurması yerine yüz yüze etkileşimli iletişim kurmaları tercih edilmektedir ve bunun sonucunda aradaki gereksiz zaman kaybı ortadan kaldırılmakta ve müşterinin daha sık aralıklarla kullanabileceği ya da test edebileceği ürünlerin ortaya çıkması mümkün olmaktadır.

Bu sürece dahil olan ekibin uzmanlığı çok önemlidir. Sürecin en önemli kaynağı insan yani uygulamayı geliştiren ekiptir. Bu ekibin, uzmanlık düzeyi ne kadar iyi seviyede ise süreç o kadar başarılı olacaktır. Ekibin kendi kendini yönetebilmesi ve ortaya çıkabilecek bir sorunda alternatif yollar önerebilmesi beklenen davranışlardandır.

En önemli amaçlardan bir tanesi, yazılımın küçük geliştirmelerle yürütülmesi ve böylece ortaya çıkabilecek risklerin azaltılmasıdır.

Çevik süreç için, deneyimli insanlarla çalışmak, çalışanları birbirlerine yakın tutmak ve iletişimi kolay bir ortam sağlamak, müşteri ya da kullanıcılara yakın olmak, geri bildirimleri hızlı değerlendirmek ve çalışmaların hızlı şekilde belgelenmesini sağlamak gerekir.

3.2 Yazılım Projelerine Uygunluk Kriterleri

Ekibin Büyüklüğü

Sözlü iletişim yazılı belgelerden daha önemlidir. Çalışanlar mutlaka kolay iletişim yapabilecek bir ortamda olmalıdır. Bu süreçte görev alacak ekip sayısı maksimum 15 olmalıdır. Eğer eşli programlama uygulanacaksa, yazılım geliştiren programcıların yan yana oturması gerekmektedir.

Uсталık

Tarif gerektirmeyen, alçak gönüllü, bilgi paylaşımını seven çalışanlar olmalıdır.

Kritik Uygulamalar

İnsan hayatını etkileyen kritik projelerde çevik yazılım geliştirme örneklerine rastlanmamıştır. Kritik projelerde birim (*unit*) testler yerine sistem (*integration*) testlerine ağırlık verilmektedir.

Bakım

Yazılım bakımı çevik süreçte, gerektiği kadar belgeleme yapıldığı için daha sonra bakım yapacak ekip açısından sorun yaratabilir.

Çevikliğe Yatkınlık

Çevik yaklaşıma, çalışanların, kurum içindeki anlayışın, alışkanlıkların ve müşterinin uyum sağlaması gerekir. Proje büyüklüğü ya da ekip sayısı çevik süreç için yeterli olabilir ancak ekibe yeterince serbestlik ya da üstünlük kullanma yetkisi verilmemesi sorun olacaktır.

Müşteriler, ara-ara çıkan ürünlerin gerçek ürünler olmadığını bilmeli ve sürece tam ayak uydurabilecek şekilde hazır olmaları gerekir.

3.2 Limitleri

Çevik yazılım geliştirme süreci,

- Kolay iletişimi olumsuz etkilediği için dağıtık ürün geliştirme ortamlarında [3],
- Eşli programlama gereksinimi ve birçok süreçte çoğunluğun görev alması gerektiği için büyük ekiplerde,
- Yazılım güvenliğinin önemli olduğu projelerde [4],
- Ve karmaşık ürün tasarımlarında uygulanamaz.

3.3 Tarihçesi

1990'lı yılların ortalarında çevik yazılım geliştirme yöntemi çok ağır yazılım geliştirme yöntemlerine alternatif olarak geliştirilmiştir. Ağır yazılım geliştirme yöntemlerinde bürokrasinin yoğun olması, işlerin çok yavaş ilerlemesi, yazılımcı yeteneklerinin üst seviyede kullanılamaması ve yazılım mühendislerinin gerçek işlerini (tasarım, analiz, kodlama vb.) yapamaması söz konusu olmaktadır.

2001 yılında Utah'da, 17 kişi bir araya gelip çevik yazılım geliştirme bildirgesini hazırladılar. Bu bildirmede;

- Kişisel ve etkileşimli çalışma,
- Anlaşılır ve kolay belgeleme (*documentation*),
- Müşteri ile birçok aşamada iş birliği,
- Yazılım geliştirme planlarında olabilecek değişikliklere hızlı uyum maddeleri ortak amaç olarak belirlendi.

Çevik yöntemlere, XP (Kent Beck), SCRUM, Adaptive software development, Crystal clear, Dynamic system development, Feature driven development ve Open source software development örnekleri gösterilebilir. Bu bildiriye, en yaygın olarak kullanılan Extreme Programming (XP) incelenecektir.

3.3 Diğer Yöntemlerle Karşılaştırma

Iterative Development

Çevik ve *iterative* geliştirme yöntemlerinde ortak amaç kısa sürede satılabilir ürünlerin çıkarılmasıdır ancak *iterative* yöntemdeki amaç, ürün çıkarma sıklığının ay bazında olması ve planların tutturulmasıdır. Çevik yöntemde ise, ürün çıkarma sıklığı neredeyse hafta mertebesinde ve önemli olan amaç kısa zaman dilimlerindeki planların tutturulmasıdır.

Waterfall

Waterfall yönteminde, analiz, tasarım, kodlama ve test süreçleri arka arkaya gelmektedir ve her bir süreç planlıdır. Her bir adımda farklı bir ürün çıkmaktadır ve bir sonraki süreç tarafından bu ürün kullanılmaktadır. Bir sürecin tamamlanıp diğerine geçilmesi uzun zaman alabilir. Çevik yöntemde ise her bir adım kısa süreler içinde tamamlanır ve sonuç, ara bir ürün olarak kullanıma hazır hale getirilir. Waterfall yönteminin her bir basamağı çevik süreç içinde yer alan bir döngüde yaşanmaktadır ve sonrasında tekrarlanmaktadır.

3.4 Değerlendirme

Yazılım projelerinde tasarım maliyetleri yüksektir ve tasarımın mutlaka deneyimli uzmanlar tarafından yapılması gerekir. Yazılım tasarımını yapacak uzmanda, ön koşul olarak kodlama yapmış olması beklenmelidir. İnşaat mühendisliğine bakıldığında, daha önceden hiç inşaat yapmamış birisi büyük bir projeyi tasarlayabilir ancak aynı durum yazılım mühendisliği için beklenemez.

Çevik yazılım geliştirme yöntemi kestirimci (*predictive*) değil uyarlanabilir (*adaptive*) bir anlayışla hazırlanmıştır. İç ya da dış müşterilere kısa aralıklarla ürünler teslim edilir ve kullanım sonrasında ortaya çıkan hatalar geri bildirimlerle hızlıca çözülür. Ara sürümlerde çıkan ürünler tüm fonksiyonları kapsamamaktadır. Ürünü kullanacak (sürece dahil olan) müşterilerin bu anlayışa hazır olması gerekir.

Çevik yöntem insan merkezli bir yaklaşım üzerine geliştirilmiştir. Yaratıcılık, deneyim, yeterince inisiyatif kullanma, problem çözme yetenekleri bu süreçte çok daha önemlidir. Usta – çırak ilişkisi ile kişiler arasındaki bilgi seviyesi kapatılır ve uzmanlar, deneyimi eksik olanları yetiştirir.

4 Extremme Programing (XP)

Extremme programlama;

- Basitlik, haberleşme, geri bildirim, cesaret temelleri üzerine kurulu olan,
- Test yapma, kodlama, dinleme ve tasarım yapma aktivitelerinden oluşan,
- Hızlı geri bildirim, basitlik, değişimi benimseyen, artımlı (*incremental*) değişim ve kaliteli prensiplerle çalışan,
- Aktiviteleri başarıyla gerçekleştirmek için kullanılan 12 pratik içeren (planlama oyunu, küçük içerikli sürümler, benzetme, basit tasarım, yeniden düzenleme, önce test geliştirme, ikili programlama, kolektif ortaklık, sürekli bütünleşme, haftada 40 saat çalışma, yerinde müşteri ve kodlama standartları),
- Gerçek hayatta yukarıda belirtilen uygulamaları başarıyla gerçekleştirmek için tanımlanmış stratejilerden oluşan bir yazılım geliştirme yöntemidir [5]

Gereksinimlerin sürekli değişmesi durumunda, ekiplerin 2 ile 10 kişi arasında olması durumunda, yazılımcıların, uzmanların ve müşterilerin aynı ortamda kod geliştirmesi durumunda XP kullanılabilir.

4.1 XP'nin Temel Adımları

İletişim

Projede yer alan tüm çalışanların iyi iletişim kurabilecek bir ortamda olması gerekir. İletişim ile her iki taraf birbirlerine bilgi sunar ve bilgi kazanır [6].

Basitlik

Uygulama tasarımına başlarken çok temel kavramlar göz önüne alınmalıdır. Uygulamanın daha sonradan destekleyeceği fonksiyonlara ilk tasarım anında değil daha sonraki sürümlerde yer verilmelidir. Uygulama içindeki karmaşık fonksiyonlar parça-parça ve sindirilerek tasarlanmalıdır.

Geribildirim

Geliştirilen her küçük sistem için testler yapılmalıdır ve sonuçlar hızlıca değerlendirmelidir. Modelleme takım halinde yapılmalıdır ve takıma projeyi hızlandıracak doğru kişiler seçilmelidir. Modelin gerçek hayata geçirilmesinde, proje motivasyonun ve bilinirliğin artması için düzenli ya da düzensiz toplantılar gerçekleştirilebilir.

Cesaret

Takım halinde çalışan tüm bireyler kendilerine ve diğer takım arkadaşlarına güven duymalıdır. Cesaretli ekip, denemekten ve yanılmaktan korkmamalıdır ve bir sorun olması durumunda inisiyatif kullanıp sorunu çözmeye ve sonuca erişmeye çalışmalıdır.

Temel adımlardan biri olan cesaretin uygulanabilmesi için kurum içindeki anlayışın ve yönetim şeklinin önemi büyüktür. Cesaret gösterecek ekibin oluşturulması ve projeyi yürütecek hale getirilmesi ancak kurum içindeki tüm süreçlerin uyumlu olması ile gerçekleşebilir.

Alçak Gönüllülük

Bir konuda uzman olmak tüm konulara hakim olmak anlamına gelmemektedir. Ekipte çalışanlar bazı konularda mutlaka eksik olduğunu bilmeli ve bilgi almaya ve yeri geldiğinde bilgi paylaşmaya açık olmalıdır. Projeyi yürüten ekip, kendi eksiklerini bilir ve onları kapatmaya gayret ederse ve sonrasında iyi olduğu konuları da ekip içinde paylaşmaya açık olursa projenin başarıyla tamamlanma oranı yükselecektir.

4.2 Riskler ve XP Çözümleri

Proje Planında Olabilecek Sapmalar

Yazılım projelerinde genelde değişiklikler çok sık ve kısa zamanlı aralıklarla yapılmaktadır. Çevik süreçte, önem derecesine göre sıralanan fonksiyonlar müşteride test edebilir hale getirilmektedir. Bu şekilde yürütülen projelerdeki sapmalar çok ciddi boyutlarda olmayacaktır ve kısa aralıklarla çıkan ürünlerde bir sorun olması durumunda az bir zaman kaybı ile sorunlar düzeltilir.

Projenin İptal Edilmesi

Geleneksel yazılım geliştirme yöntemlerinde ortaya bir ürün çıkmadan da projelerin iptal edildiği örnekler vardır. Projenin tasarım aşamasında bir erteleme olması daha sonradan birbirlerine bağlı olan diğer süreçleri etkileyebilir ve ortaya ürün çıkmadan proje iptal edilebilir. XP’de ise, mutlaka içeriği zengin olmayan ve kullanılabilir bir ürün vardır. Bu ürünler ve artımlı devam eden süreçler projenin iptalini gerektiren riskleri ortadan kaldırır.

Sistemin Kötüye Gitmesi

XP’de her değişiklik sonrasında test işlemleri yapılmakta ve test güdümlü kodların sonuçları başarılı beklenmektedir. Bir değişiklik sonrasında hata veren test sonuçları değişikliği yapan uzman(lar) tarafından kontrol edilir ve düzeltilir.

Birim testlerle, değişimi kontrol altına alan bir süreçte, sistemin kötüye doğru gitmesinden söz edilemez. Sistem ancak birim test zamanında kötü sonuçlar doğurabilir ancak daha sonrasında küçük sürümler öncesindeki kontrollerle bu sorunlar da düzeltilir.

Yanlış Algoritmaların Kodlanması

XP'de müşteri, yazılım gerçekleştiren ekibin önemli bir parçasıdır ve süreç içinde aktif olarak rol almaktadır. Tasarım ya da testlerde müşteri iş mantıklarının yanlış tasarlandığını fark ederse ekip arkadaşlarını uyararak doğru hedefe yönlendirir.

İş Mantığı Değişiklikleri

Ürün tasarlanırken müşteri gereksinimlerinde değişiklikler olabilir. XP'de ürün aralıkları çok kısa olduğu için bu tür değişikliklerin geri dönüş maliyetleri çok küçük olacaktır.

4.3 XP'nin Uygulanması

Planlama

Planlama aşamasında, müşteri ile birlikte çalışma yaparak bir sonraki çıkacak üründe nelerin yapılacağı ve hangi önceliklerde olacağı belirlenmektedir. Yapılacak işler çıktıktan sonra uzman ekip tarafından süreler tahmin edilir ve müşteri bilgilendirilir [7].

Ara Sürümler

Ara sürümlerde içeriği zengin olmayan küçük ürünler tasarlanır ve müşterinin bu ürünleri test ettikten sonra geri bildirimleri hızlıca değerlendirilir.

Basit Tasarım

Basit tasarım, sistem özelliklerini aynı günde kodlayacak şekilde tasarım yapmak olarak düşünülmelidir. Gelecekte kullanılacak özelliklerin baştan tasarlanması, süre kaybına ve ileride tasarımın hatalı olmasına sebep olacaktır.

Test

Test işlemleri, birim testleri (*unit test*) de içermektedir. Birim testler, yazılımcıların müşteri önceliklerine göre yazdıkları ve daha sonra test ettikleri aktivitelerdir. Küçük sürümlerle yapılan testler ürünlerin bitişini göstermektedir. Test sonuçlarının başarılı olması ve müşteri tarafından onaylanması, ara sürümün kullanılabilir olduğunu göstermektedir.

Refactoring

Yazılımcılar mevcut bir tasarımı geliştirmekle sorumludur. Gelişim devam ederken, sistem bir önceki özelliklerini ve davranışlarını aynen korumalıdır ve yeni fonksiyonları çalıştırabilir halde tutmalıdır. *Refactoring*, bu nedenle yazılımcıların günlük aktiviteleri olarak tanımlanır.

Eşli Programlama

Eşli programlamada, ürün iki programcı tarafından kodlanmalıdır. Birisi yazarken diğeri takip eder, muhtemel olabilecek hataları izler ya da eş deęişiminde diğernin kaldığı yerden kolayca ilerleyebilir.

Ortak Kod Mülkiyeti

Ekipte herkes kodun bir yerini deęiştirebilir. Tüm kararlar ve tasarımlar birlikte alındığı için tüm bireyler bu cesareti gösterebilir ve herkes birbirinin kodunu sahiplenir.

Sürekli Tümeleştirme

Sürekli tümeleştirme ile programcılar geliştirdikleri küçük sistemleri gün içinde ya da sonunda birkaç kez birleştirebilir.

Haftada 40 Saat Çalışma

Yazılımcılar yorulduğu yere kadar çalışmalıdır. Normal bir yazılımcının gün aşırı çalışması ve tekrar işine erkenden devam etmesi çok verimli olmaz. Görevlerin yetiştirilmemesi durumunda fazla mesai yapılabilir ancak bununda 1–2 saat ile sınırlı kalmalıdır.

Müşteri Odaklılık

Müşteri yazılım geliştiren ekibin içindedir ve aktif rol almaktadır. Müşteri her zaman tasarımda çıkabilecek sorulara cevap verebilir durumda olmalıdır ve ayrıca ara ürünlerin kabul testlerinden sorumludur.

Kodlama Standartları

Kod standartları, eşli programlama ve ekipte yer alan bireylerin birlerinin koduna bakmasından dolayı önemlidir. Her sınıfın, nesnenin veya yöntemin yazılışının aynı olması kodun anlaşılmasını kolaylaştıracak ve ekip içindeki iletişimi arttıracaktır.

5 Türkiye’de Çevik Süreç

5.1 Bimar

Türkiye’de çevik süreç yöntemlerine örnek olması için Bimar’ın yapmış olduğu çalışmalar tercih edilmiştir. Bimar, Aralık 2003 tarihinde bu çalışmaya başlamıştır. Aynı anda CMMI seviye 2 ve XP pratiklerin uygulanması amaçlanmıştır. Bu çalışma sonrasındaki Bimar’ın beklentisi, %20 oranında zaman ve maliyet değerlerinin indirgenmesi ve %30 oranında müşteriye gidecek son ürünlerdeki hata sayısının azaltılması olmuştur.

XP' de önemli bir faktör olan müşteri ile birlikte çalışma faaliyetleri şirketin kendi tasarladığı kartlar üzerinden götürülmüştür. Müşteriler, kartlara uygulamalardaki beklentilerini yazmakta ve daha sonra uygulamayı geliştiren ekip bu kartları inceleyerek planlama ve ara sürümün çıkış tarihlerini kestirmektedir.

Kartlarda yer alan kullanıcı hikayeleri, eşli programlama ve test güdümlü yöntemlerle kodlanmaktadır. XP pratiklerine uyumlu yürütülen pilot projenin sonrasında,

- Tek programcıya göre daha hızlı yazılım üretimi,
- Olası hatalarda azalma,
- Ve yazılımcılar arasındaki bilgi düzeyinin hızlıca kapandığı izlenmiştir.

Kodlamaya başlamadan önce test kodlarının yazılması ve daha sonra kodlamaya geçilmesi ile fonksiyonlardaki girdi ve çıktı değerleri önceden test edildiği için ortaya çıkabilecek hatalar da azaltılmıştır.

Tablo 2'de Bimar'ın pilot proje sonrasında yapmış olduğu değerlendirme notları gösterilmektedir.

Tablo 2. Bazı CMMI süreçleri ve XP pratikleri arasındaki ilişki. [9]

	Kartlar	Eşli Programlama	Test Güdümlü prog.
<i>Requirement Managements</i>	++	{ }	+++
<i>Project Planning</i>	+++	++	++
<i>Configuration management</i>	--	{ }	{ }

5.2 Genel

Türkiye'de yazılım yapan firmalarda kullanılan yazılım geliştirme yöntemleri, çalışan sayısına ve üretilecek yazılımın tiplerine göre tercih edilmektedir. İhracat yapabilen ve ekip sayısının çok olduğu firmalarda genel eğilim, CMMI/SPICE yöntemlerine üzerine olmuştur.

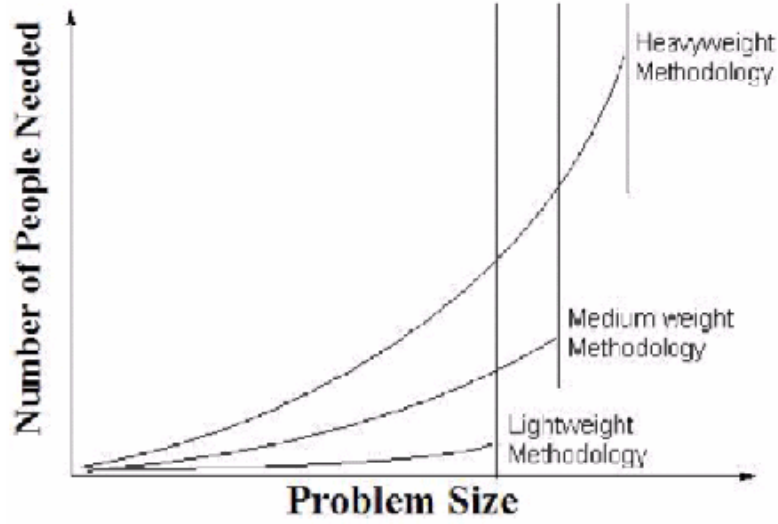
Çevik sürecin Türkiye'de yaygın olarak kullanılması, yazılım mühendisliği disiplinin gelişmesine, çevik süreç deneyimlerinin artmasına, danışmanlık yapan kuruluşların iyi bir deneye sahip olmasına ve başarılı örneklerin kendilerini göstermesine bağlıdır.

6 Sonuç

Yazılım projelerinin yürütülmesi, tamamlanması ve başarı ile sonuçlandırılması genelde çok zor görünmektedir. Yazılım mühendisliği disiplininin gelişimi ile ara aşamalarda çıkan birçok sorun her ne kadar giderilse de projenin yönetilme şekli de çok etkilidir.

Yazılım geliştirme yöntemlerinde insanın ya da standartların önde olması projeye, kuruma, çalışanlara ya da müşterinin beklentilerine göre değişkenlik göstermektedir. “Firmalar için en iyi yazılım geliştirme yöntemi” gibi bir sabit model belirlenemez ancak firmalar kendi kültürlerine, organizasyon yapılarına ve proje gereksinimlerine göre en iyi yöntemi seçer ya da birkaç yöntemin kendilerine uyarlanmış şeklini benimser.

Yazılım projelerinde insan faktörü çok önemlidir. Projelerin kapsamı büyüdükçe insanları yönetmek, etkin kullanmak ya da görev alan bireylerin yaratıcılıklarını sergilemesini beklemek ve inisiyatif kullanmalarını beklemek çok güçtür. Şekil 2’de gösterildiği gibi projeler büyüdükçe daha çok insan gerekir ve bu insanları yönetmek için belirli standartların olması da zorunludur. Küçük projelerde ise daha az insan görev alabilir ve bu insanlar etkin kullanılabilir.



Şekil 2. Problem büyüklüğü ve insan gereksinimi arasındaki ilişki [8]

Yazılım projelerinde insan çok önemli bir faktör olsa da, projeyi başarılı şekilde tamamlamak için diğer süreçlerin de başarılı olması gerekmektedir. Tablo 3’de 250 yazılım firmasının projeleri incelenmiş ve başarılı/başarısız olduğu noktalar gözlemlenmiştir. Bir yazılım projesinin yönetmek için tercih edilecek yöntemin iyi yanlarını ve kötü yanlarını bilmek ve ileride ortaya çıkabilecek riskleri tahmin etmek doğru yazılım geliştirme yöntemini seçmede kolaylık sağlayacaktır. Doğru model ya da temel alınan modelin ortaya çıkabilecek risklere göre düzenlenmiş hali projenin başarılı olmasında önemli rol oynayacaktır.

Tablo 3. 1995–2004 yıllarında 250 yazılım projesinin değerlendirilmesi (Jones, 2004)

Başarılı projeler	Başarısız projeler
Etkin proje planı	Yetersiz proje planı
Etkin proje maliyet tahminleri	Yetersiz proje maliyet tahminleri
Etkin proje ölçütleri	Yetersiz proje ölçütleri
Etkin proje değişiklik yönetimi	Verimsiz proje değişiklik kontrolü
Etkin proje kalite kontrol	Yetersiz proje kalite kontrol
Etkin proje ara ürün izleme	Yetersiz proje ara ürün izleme

Yazılım geliştirme yöntemlerinde ana ve ara yöntemlerin kullanılması da söz konusudur. Projenin kapsamlı ve büyük olmasında CMMI/SPICE gibi yaygın yöntemler kullanılırken ara aşamalarda (alt projelerde) XP gibi çevik yöntemler tercih edilebilir.

Kaynakça

1. Alican, F., Türk yazılım sektörü: sorunlar ve çözüm önerileri, Türkiye, 2004.
2. Bilişim 500, 2001.
3. Cohn, M., Introducing an agile process to an organization, 2001.
4. Turk, D., France R. ve Rumpel, B., Limitations of agile software processes, 2002.
5. Süloğlu, S., Yöntem çevik olunca, <http://uyms.emo.org.tr/bildiriler/26.pdf>, 2003.
6. Kuru, İ., Extreme programming, <http://www.istanbul.edu.tr/Bolumler/enformatik/seminer/2004/seminer/XP.pps>, 2004.
7. Newkirk, J. Introduction to agile process and extreme Programming, USA, <http://www.thoughtworks.com>, 2002.
8. Tortamış, P., Light vs. Heavy: Which To Choose?, Impact of software process on quality workshop, Türkiye, 2004.
9. Kalaycı, O., Real life Experience – How to develop CMMI process, Impact of software process on quality workshop, Türkiye, 2004.