# Yazılım Mühendisliği Projelerinde Çevik Yaklaşımların Yeri

Turgay Karlıdere[1], Oya Kalıpsız[2]

[1] Deniz Harp Okulu, 34942, Tuzla, İstanbul
[1] tkarlidere@dho.edu.tr
[2] Yıldız Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü, 34349, Yıldız, İstanbul
[2] kalipsiz@yildiz.edu.tr

**Özet.** Son yıllarda iş hayatında rekabetin sertleşmesi ve rakiplerden farklılaşma ihtiyacı, müşteri gereksinimlerinin hızlı değişimine yol açmaktadır. Gereksinimleri hızlı değişen müşterinin yeni isteklerini, yazılım proje ekibinin daha çabuk yerine getirebilmesi amacıyla çevik yazılım geliştirme yöntemleri ortaya atılmıştır.

Bu makalede, çevik yazılım geliştirme ile ilgili şu sorulara yanıtlar aranmaktadır: Çevik yaklaşımlar, ne tip projelere ve hangi şartlara daha uygundur? Çevikliğin, ürün tesliminden sonraki aşamalara da faydası var mıdır? Çevik yaklaşımların benimsenme süreci nedir? Alışılmış metodolojileri terk edip çevik olma zamanı gelmiş midir? Çevik ve geleneksel yöntemler birlikte kullanılabilir mi?

## 1 Giriş

Yazılım geliştirme sürecinin ilerleyen safhalarında ortaya çıkacak değişim isteklerinin getireceği maliyet nedeniyle, geleneksel yaklaşımlar, başlangıçta yeterince kapsamlı çalışıp ihtiyaçları eksiksiz tespit etmeyi ve bu sayede projenin ileri safhalarında ortaya çıkabilecek olası değişimleri önlemeyi esas alır. Ancak hızlı gelişen teknoloji ve artan rekabet ortamında müşteri gereksinimleri artık daha hızlı değişmekte ve değişim isteklerinin daha çabuk yerine getirilmesi gerekmektedir. Proje başlangıcında eksiksiz tespit edilmeleri imkansızlaşan müşteri ihtiyaçlarındaki değişimlere engel olunamadığına göre, yazılım projelerinde kullanılacak yazılım geliştirme yöntemlerinin de bu değişimleri daha kısa sürede ve daha az maliyetle karşılayabilecek özelliklere sahip olması beklenmektedir. Bu nedenle, geleneksel yazılım geliştirme yöntemlerine alternatif olarak çevik yazılım geliştirme yöntemleri ortaya çıkmıştır [1], [2].

Her tip yazılım projesine uygun tek bir yazılım geliştirme yönteminden bahsedilemeyeceği için projeye uygun yazılım geliştirme yönteminin kullanılması, projenin başarısını etkileyecektir. Bu nedenle çevik yöntemlerin hangi tip projelerde kullanıldıklarında daha iyi sonuç vereceklerini bilmek gerekir. Biz de çalışmamızda, çevik yazılım geliştirme yöntemlerini, proje özellikleri açısından inceleyerek, geleneksel yöntemler yerine çevik yöntemlerin kullanılması hakkında karar vermeye faydalı olacak kriterler ortaya koyduk.

İncelemeye esas olması açısından çevik yaklaşımların genel özelliklerinin anlatıldığı ikinci bölümden sonra üçüncü bölümde, bu özellikler esas alınarak tespit edilen, çevik yöntemlerin yazılım projelerine uygunluk kriterleri açıklanmıştır. Dördüncü bölümde, çevik yaklaşımların benimsenme sürecine değinilmiştir. Çevik ve geleneksel yazılım geliştirme yöntemlerinin birlikte kullanılabilirliği konusunun ele alındığı beşinci bölümden sonra çevik yaklaşımlarla ilgili varılan sonuçlara yer verilmiştir.

## 2 Çevik Yaklaşımların Özellikleri

Çevik yaklaşımlar, yazılım geliştirme safhasında geç ortaya çıkan gereksinim değişimlerini çabuk karşılamayı esas alan yöntemlere verilen genel bir isimdir. Bu yöntemlerden adı daha fazla ön plana çıkanlar arasında Scrum, Dynamic Systems Development Methodology (DSDM), Adaptive Software Development (ASD) ve Extreme Programming (XP)'yi sayabiliriz. Her çevik yöntemin kendisine özgü prensipleri [2] olsa da çevik yaklaşım şemsiyesi altında belirlenen genel özellikler; kişiler ve kişiler arasındaki etkileşim, çalışan yazılım, müşteriyle işbirliği ve değişim isteklerini karşılamak şeklinde *"çeviklik bildirisi"*yle [3] duyurulmuştur.

Çevik yaklaşımların ilk özelliği, süreçler ve araçlardan çok geliştirme ekibindeki kişilere değer vermeleridir [4]. Yazılım geliştirmede uygulanan yöntemin başarısı, takip edilen sürece ve kullanılan araçlara bağlı olsa da sonuçta süreci uygulayan ve araçları kullananlar insanlardır. Yetenekli ve usta ekiplerle uygulanan süreçlerin başarısız olma olasılığı düşüktür. Çevik yaklaşımlar, çalışanlara gerekli ortamı ve desteği sağlayarak, başarılı olacaklarına güvenmeyi ve kararlarına değer vermeyi gerektirir.

Çevik yaklaşımların bir diğer özelliği, çalışan yazılıma, dokümantasyondan daha fazla değer vermeleridir. Geleneksel yöntemlerdeki kesin kurallar ve dokümantasyon yükü yazılım geliştirme ekibinin süreci uygulamasını zorlaştırdığından, çevik yaklaşımlar ürüne doğrudan faydası olmayan ve kullanılmayan doküman üretmek yerine, gerektiği kadar dokümantasyon hazırlamak ve aslen çalışan yazılıma odaklanmak gereğini savunurlar [1], [5].

Çevik yaklaşımlara göre yazılım geliştirme ekibinin, müşterinin gerçekten ne istediğini anlayabilmesi ve onu üretmesi için proje sonuna kadar müşteriyle birlikte çalışması gerekir. Çevik yöntemlerde detaylı gereksinim analizi yapılmadığından, iyi bir tasarım ve gerçekleştirme için müşteri, geliştirme ekibiyle aynı takımda yer almalıdır. Proje sonunda gereksinimlerin eksiksiz karşılanabilmesi ancak bu sayede mümkün olabilir.

Çevik yaklaşımların belki de en önemli özelliği plandan çok, değişim isteklerini karşılamaya önem vermeleridir. Yazılım proje ekibi ilk planlarda belirlenmiş müşteri ihtiyaçlarını gerçekleştirdiğinde sonuca ulaştığını düşünebilir. Ancak zaman içerisinde müşterinin ihtiyaçları değişebileceğinden, ortaya çıkan ürün gerçekten müşterinin istediği sonuç olmayabilir. Bu nedenle yazılım projelerinde planlara titizlikle uyup değişen ortam şartlarına ve gereksinimleri farklılaşan müşterinin yeni isteklerine kayıtsız kalmak ya da bu istekleri karşılamada isteksiz davranmak veya gecikmek projenin başarısızlığına sebep olur. Çevik yaklaşımlar, değişimlere en kısa zamanda

yanıt verecek şekilde yapılanmayı gerektirir.

## 3 Çevik Yöntemlerin Yazılım Projelerine Uygunluk Kriterleri

Bir yazılım geliştirme yönteminin her tür projede iyi sonuç vermesi beklenmemelidir [6], [7]. Yazılım projelerinde kullanılacak yazılım geliştirme yöntemini belirlerken projenin özelliklerini dikkate almak gerekir. Projeye uygun olmayan yöntemin kendisi bazen başarısızlık sebebi olabilmektedir. Şu halde, belli metodolojilerin veya süreçlerin ne zaman ve hangi şartlar altında en iyi çözüm olacağını ve bulunulan duruma göre nasıl biçimlendirileceğini bilmek önemlidir.

Çevik yöntemlerin hangi tip projelere daha uygun oldukları hakkında bu çalışmada tespit edilen kriterler, yazılım projelerinde uygulanacak yönteme karar verilmesinde faydalı olacaktır. Ekibin büyüklüğü, ekipteki ustalık, müşteri profili, uygulama kritikliği, bakım safhası ve çevikliğe yatkınlık olmak üzere altı başlık altında incelenecek bu kriterlerin belirlenmesinde çevik yaklaşımların, çeviklik bildirisinde [3] tanımlanan genel özellikleri esas alınmıştır.

### 3.1 Ekibin Büyüklüğü

Yazılı dokümanlardan çok yüzyüze görüşmeye önem veren çevik yöntemlerde çalışanlar arasında iyi iletişim ve etkileşim şarttır. Bu nedenle çevik yöntemlerin uygulanması kararını etkileyen belki de en önemli faktör çalışanların sayısıdır. Çevik yöntemleri deneyen proje ekipleri çoğunlukla 15 kişiyi geçmemektedir [7], [8]. Kalabalık geliştirme ekiplerinde yüz yüze iletişim ve koordinasyon zorluğu, çevik yöntemlerin başarısını olumsuz yönde etkileyecektir. Bu durumda ekibi küçük takımlara bölmek gibi ölçekleme yöntemleri uygulanmalıdır [7].

Çevik yaklaşımlarda yüzyüze görüşme ve müşteriyle birlikte çalışma ihtiyacı, proje ekibinin aynı ortamda bulunmasını gerektirir. Birbirinden farklı uzak mekanlarda bulunması gereken çalışanlar aynı takımda yer almamalıdır. Teknolojik imkanlar kullanılarak farklı mekanlarda bulunmanın getirdiği iletişim ve koordinasyon eksikliği aşılmaya çalışılsa da programcıların başarımını arttırdığı savunulan *"eşli programlama"* [8], [9], [10] uygulanacaksa programcıların yanyana olmaları gerekir.

### 3.2 Ekipteki Ustalık

Çevikliğin özünde, tarifi gereksiz bilgileri planlara yazmak yerine ekibin kendiliğinden biliyor olmasını beklemek yattığından, çevik yöntemler kullanılacaksa, geliştirme takımında *"tarif gerekmeyen"* yetenekli ve tecrübeli kişiler bulunması gerekir [6]. Proje ekibinin hepsi olmasa bile bir kısmının geçmişte benzer projeler geliştirmiş, kullanılacak teknolojiye hakim ve insan ilişkilerinde iyi özelliklere sahip olması, çevik yöntemlerin uygulanabilirliği için önemlidir. Ekibin organizasyonunda eşli programlama uygulanarak ve proje süresince eşler rotasyonla değiştirilerek, gereken tecrübeli ve yetenekli kişi sayısı azaltılabilir [7]. Ancak bunun için, yardımlaşmayı ve bilgi paylaşmayı seven, alçak gönüllü takım arkadaşlarının önemi büyüktür [11].

### 3.3 Müşteri Profili

Çevik yaklaşımların genel prensibi, müşterinin de geliştirme ekibi ile birlikte aynı takım içinde çalışmasını gerektirir. Başlangıçta detaylı gereksinim analizi yapılmadığından, tasarım ve kodlama yapılabilmesi için analizdeki eksikleri doldurmak üzere müşteriyle birlikte çalışmak önemlidir. Müşteri profilinin buna uygun olmaması halinde çevik yöntemler başarısız olacaktır. Örneğin ihtiyaçların, müşterinin üst yönetimi tarafından bilinmesi ve bu kişilerin de zamanlarının kısıtlı olması nedeniyle birlikte çalışma mümkün olmayabilir. Müşteri uzak bir ortamda bulunabilir veya birlikte çalışmaya istekli olmayabilir. Gerçek ihtiyaç sahibi yerine projenin verildiği yüklenici firma temsilcisi ile birlikte çalışmak gerekebilir [12]. Bu gibi hallerde projenin başarısı olumsuz yönde etkilenecektir.

### 3.4 Kritik Uygulamalar

Çevik yöntemlerin; insan hayatını etkileyen sistemler, güvenlik uygulamaları, nükleer sistemler, uzay çalışmaları ve silah sistemleri gibi kritik ve başarısızlık maliyeti yüksek projelerde uygulama örneğine rastlanmamıştır. Yüksek güvenilirlik beklenen bu tip projeler çok iyi test edilmeden teslim edilemez. Çevik yaklaşımlarda uygulanan iterativ ve kademeli geliştirme yönteminin yeterli sıklıkta test ortamı yarattığı düşünülebilir [7]. Ancak bu tip uygulamalarda sık test yapmak mümkün olmayabileceği gibi birim testlerinden çok sistem (bütünlük) testleri önemlidir. Çevik yöntemlerin hata affetmeyen kritik sistemlerde kullanımları konusunda yorum yapmak için bu yöntemlerle ilgili henüz yeterli olmayan ampirik verilerin [13] artmasını beklemek gerekir.

Kritik sistemlerin geliştirilmesinde tamamen çevik yaklaşımlar kullanmak yerine, geleneksel yöntemlerin içinde, çevik yaklaşımların eşli programlama, önce test kodunu hazırlama ve kademeli geliştirme gibi pratiklerinden faydalanılabilir [14].

### 3.5 Bakım Safhası

Bazı projelerde bakım safhası, geliştirme safhasından daha önemlidir. Proje teslim edildikten sonraki süreçte geliştirme ekibinde olası değişiklikler ve dokümantasyon eksiklikleri nedeniyle bakım işlemleri maliyeti arttırarak projenin yeniden yapılmasına bile yol açabilmektedir. Çevik yöntemlerde, “gerektiği kadar dokümantasyon” ilkesi sebebiyle, bakım safhasında lazım olacak dokümanlardan feragat edilme riski bulunur. Bu nedenle projenin tesliminden sonraki aşamalar da düşünülerek bakım safhasında gerekli olabilecek dokümantasyon hazırlanmalıdır.

Çevik yazılım geliştirme yöntemlerinin bakım süreçlerini nasıl etkileyeceği henüz fazla gündeme gelmemekle [8] birlikte, farklı bir ekip tarafından daha önce çevik yöntemlerle geliştirilmiş bir uygulamanın bakım projesi, uygulama hakkında yeterli bilgi ve belge olmazsa bilinen yazılım mühendisliği sorunlarıyla karşı karşıya kalacaktır.

### 3.6 Çevikliğe Yatkınlık

Çevik yaklaşımları kullanmayı düşünenlerin, öncelikle kurumları içindeki anlayış ve

alışkanlıkları çevik yaklaşımlara uydurmaları gerektiği savunulmaktadır [12]. Çevik yöntemler, yazılım geliştirme teknikleri ve zamanlama gibi teknik konularda karar verme yetkisini üst yönetimden alıp geliştirme ekibinin sorumluluğuna verdiklerinden [15], kurum içinde bu değişimin benimsenmiş olması gereklidir.

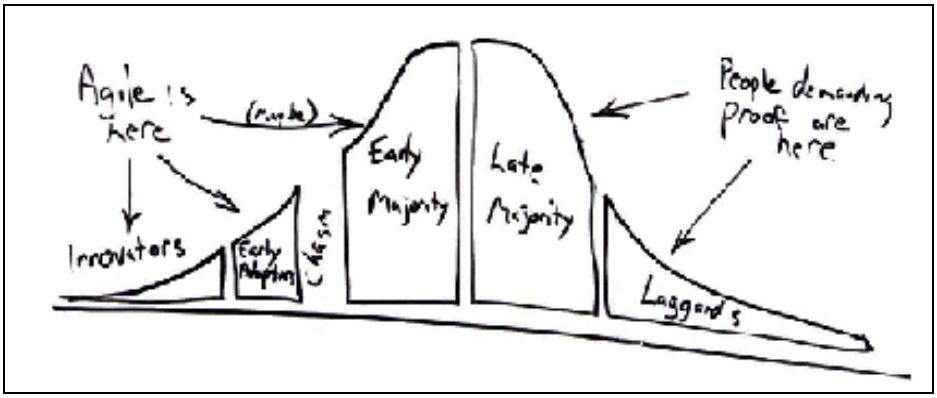
Projenin kendisi çevik yöntemlerin uygulanmasına çok elverişli olabilir. Ancak, geliştirme ekibine yeterince serbestlik tanınmayan, ekibin çalışma şekillerinin katı kurallar ve yöntemlerle kısıtlandığı ortamlarda çevik yaklaşımların başarı şansı azalacaktır.

Müşteriye, proje başlangıcında gereksinimleri formal bir şekilde imza altına almayan bir sözleşmeyi kabul ettirmenin zor olacağı dikkate alınarak, müşterilerin de çevik yöntemler konusunda eğitilmesi gerekebilir [12].

## 4 Çevik Yaklaşımların Benimsenme Süreci

Çevik yazılım geliştirme yöntemlerinin başarılı olabilecekleri projelerin özelliklerindeki kısıtlamalar, uzun yıllar denenmiş geleneksel yöntemler dururken çevik yöntemlerin benimsenmesini etkilemektedir. Çevik yöntemlerle ilgili birçok başarı hikayesi anlatılmasına rağmen bu yöntemlerin beninsenme oranının yükselmesi, ampirik verilerin toplanıp analiz edilmesine bağlıdır.

Bir taraftan, metriklerin oluşturulabilmesi için çevik yöntemlerin denenmesi ve gerçek projelerde kullanılması gerekirken diğer taraftan, bu yöntemlerin yeni olmaları, endişeyle karşılanmalarına neden olmaktadır.




**Şekil 1.** Çevik yöntem benimseme yaşam döngüsü [16]

Yeni teknolojileri kabullenme konusunda insanları ve kurumları beş profile ayıran Geoffrey Moore'un "Crossing the Chasm" isimli kitabındaki *"teknoloji benimseme*

*yaşam döngüsü"*, çevik yöntemlerin benimsenme sürecine uygulandığında, şekil 1'de olduğu gibi *"hemen benimseyenler"* ile *"erken çoğunluk"* arasındaki uçurumun henüz atlanamadığı görülmektedir. Erken çoğunluk, çevik yöntemlerle ilgili başarı örneklerinin sayısı yeterli seviyeye ulaşmadan bu uçurumdan atlamak istemeyecektir. Yeni teknolojileri kabullenme konusunda daha temkinli yaklaşan son gruplar ise çevik yöntemleri uygulamak için kesin verileri bekleyecektir.

## 5 Çevik ve Geleneksel Yöntemlerin Birlikte Kullanılması

Bütün yazılım projeleri aynı özellikte olmadığı gibi geleneksel yöntemlerle çevik yöntemlerin de kendilerine özgü uygulama alanları veya daha iyi sonuç verecekleri proje tipleri vardır. Örneğin çevik yazılım geliştirme yöntemlerinin asıl hedefi, yazılım geliştirme aşamasında değişim beklentisinin yüksek olduğu projelerdir. Ancak, proje özelliklerinin yazılım geliştirme yönteminin özellikleriyle tam olarak örtüşmediği durumlarda, çevik ve geleneksel yöntemler, biri diğerinin güçsüz olduğu tarafları destekleyecek şekilde, birlikte kullanılabilirler. Projenin tamamına çevik yöntemler uygulanamıyorsa, sadece gerektiği yerlerde çevik olunabilir veya çevik yaklaşımlar, projeye ve geliştirme ekibinin yapısına göre biçimlendirilebilir [17], [18], [19].

Üçüncü bölümde anlatılan kriterler esas alınarak çevik yaklaşımların projeye uygun olup olmadıkları ve yetersiz oldukları taraflar tespit edildikten sonra, geleneksel yöntemlerle desteklenmiş çevik yaklaşımların veya gerekli kısımları çevikleştirilmiş geleneksel yöntemlerin kullanılması tercih edilmelidir [20].

## 6 Sonuç

Müşteri ihtiyaçlarındaki değişimlerin hızı artmakla birlikte yazılım projelerinden beklentiler de artmaktadır. Hızlı olmak uğruna kaliteden ödün vermemek gerekir. Çevik yöntemlerin iterativ yapısı, projenin erken safhalarında başlayan testler ve onların sonucundaki geri beslemeler sayesinde kaliteli ürün teslimini sağlayabilir görünmektedir. Ancak programcılar ve müşteri haricinde kalite danışmanı veya gözlemcisi gibi kişilere ekipte yer verilmemiş olması bir eksiklik olarak değerlendirilmektedir [15].

Yıllardır uygulanan plana dayalı yöntemlerin yetenekli ve tecrübeli kişiler tarafından kullanılması sırasında bazı adımlar sıradan hale gelmiş olabilir. Muhasebe veya bankacılık uygulamaları gibi çok sayıda yazılım projesinde çalışmış tecrübeli uzmanlar, benzer yeni bir yazılıma başlarken detaylı analiz ve tasarım yapmaktan sıkılabilirler. Çünkü bu kişiler, ihtiyaçları çoğu zaman müşteriden daha iyi tarif edebilme ustalığına ulaşmışlardır. Bu kişilerin, planlar ve dokümantasyonla uğraşmak yerine, doğrudan çalışır bir ürün ortaya koymalarını ve onun üzerinde iterativ ve kademeli bir geliştirme yöntemi uygulamak istemelerini, yani çevik olmalarını, doğal karşılamak gerekir. Ancak bu yaklaşım tecrübeli, yetenekli ve usta kişiler için ve onlara göre çok sıradan hale gelen ve kritik olmayan uygulama alanlarıyla sınırlı tutulmalıdır.

Yazılım projelerinin, üçüncü bölümde önerilen kriterlere göre değerlendirilmesi, çevik

yöntemlerin projeye uygulanabilirliği hakkında karar verilmesine yardımcı olacaktır. Bu değerlendirme sonucunda çevik yaklaşımların yetersiz kaldığı görülen taraflar varsa geleneksel ve çevik yöntemlerin dengeli bir karışımı uygulanmalıdır.

Yazılım geliştirme projelerinde çevik yaklaşımların yeri, halen devam etmekte olan akademik çalışmaların ve uygulama örneklerinin artmasıyla daha iyi belirlenebilecektir. Çevik yaklaşımlarda yazılım ölçüm yöntemleri, yazılım bakımı ve CMMI gibi süreç iyileştirme kriterlerinin sağlanması konularında yapılacak çalışmaların, bu alana katkı sağlayacağı değerlendirilmektedir.

## Kaynakça


1. Highsmith, J. ve Cockburn, A., "Agile Software Development: The Business of Innovation", IEEE Computer, Eylül 2001, s. 120-122.
2. Highsmith, J., "What is Agile Software Development?" CrossTalk The Journal of Defense Software Engineering, Ekim 2002, s. 4-9.
3. Agile Alliance, Agile Software Development Manifesto, 13 Şubat 2001, http://www.agilemanifesto.org
4. Cockburn, A. ve Highsmith, J., "Agile Software Development: The People Factor", IEEE Computer, Kasım 2001, s. 131-133.
5. Tekinerdoğan, B., "Formalizing Agile Software Development Methods", Impact of Software Process on Quality (IMPROQ 2003 Workshop), 6 Haziran 2003, http://www.cs.bilkent.edu.tr/improq03/Papers/Tekinerdogan.pdf
6. Boehm, B., "Get Ready for Agile Methods, with Care", IEEE Computer, Ocak 2002, s. 64-69.
7. Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., Tesoriero, R., Williams, L. ve Zelkowitz, M., "Empirical Findings in Agile Methods", Proceedings of Extreme Programming and Agile Methods – XP/Agile Universe 2002, s. 197-207.
8. Cohen, D., Lindvall, M. ve Costa, P., "Agile Software Development", The Data & Analysis Center for Software (DACS) Tech Report Number: DACS-SOAR-11, 31 Ocak 2003.
9. Wood, W. A. ve Kleb, W. L., "Exploring XP for Scientific Research", IEEE Software, Mayıs/Haziran 2003, s. 30-36.
10. McDowell, C., Werner, L., Bullock, H.E. ve Fernald, J., "The Impact of Pair Programming on Student Performance, Perception and Persistence", Proceedings of the 25th International Conference on Software Engineering, 3-10 Mayıs 2003, s. 602-607.
11. Rasmusson, J., "Introducing XP into Greenfield Projects : Lessons Learned", IEEE Software, Mayıs/Haziran 2003, s. 21-28.
12. Murru, O., Deias, R. ve Mugheddu, G., "Assessing XP at a European Internet Company", IEEE Software, Mayıs/Haziran 2003, s. 37-43.
13. Abrahamsson, P., Warsta, J., Siponen, M.T. ve Ronkainen, J., "New Directions on Agile Methods: A Comparative Analysis", Proceedings of the 25th International Conference on Software Engineering, 3-10 Mayıs 2003, s. 244-254.
14. Turk, D., France, R. ve Rumpe, B., "Limitations of Agile Software Processes", 3rd

International Conference on Extreme Programming and Flexible Processes in Software Engineering, 26-30 Mayıs 2002, s. 43-46.

15. Laurie, W. ve Alistair, C., “Agile Software Development: It’s About Feedback and Change”, IEEE Computer, Haziran 2003, s. 39-43.
16. Ambler, S. W., “Answering the ‘Where is the Proof That Agile Methods Work’ Question”, Official Agile Modeling Site, http://www.agilemodeling.com/essays/proof.htm
17. Reifer, D. J., Maurer, F. ve Erdogmus, H., “Scaling Agile Methods”, IEEE Software, Temmuz/Ağustos 2003, s. 12-14.
18. Lippert, M., Becker-Pechau, P., Breitling, H., Koch, J., Kornstadt, A., Roock, S., Schmolitzky, A., Wolf, H. ve Züllighoven, H., “Developing Complex Projects Using XP with Extensions”, IEEE Computer, Haziran 2003, s. 67-73.
19. Grenning, J., “Launching Extreme Programming at a Process-Intensive Company”, IEEE Software, Kasım/Aralık 2001, s. 27-33.
20. Boehm, B. ve Turner, R., “Using Risk to Balance Agile and Plan-Driven Methods”, IEEE Computer, Haziran 2003, s. 57-66.