Yazılım Geliştirme Süreçlerinde Şelale Yönteminden Çevik Yaklaşıma Geçiş: Bir Teknoloji Şirketinde Uygulama

Ayfer Başar[1], Ali Özkaya[1], Fatih Kesgin[2]

1: Ziraat Teknoloji A.Ş., ARGE ve Kalite Servisi, Davutpaşa, İstanbul
2: Ziraat Teknoloji A.Ş., Yazılım Araçları Servisi, Davutpaşa, İstanbul
{abasar, aozkaya, fkesgin}@ziraatteknoloji.com

Özet. Yazılım üretiminde, yönetim disiplini ihtiyacı yazılım geliştirme yöntemlerinin geliştirilmesini gerekli kılmıştır. 1970'li yıllarda Winston W. Royce tarafından yazılan bir makalede tarif edilen Şelale ("Waterfall") yöntemi, yazılım geliştirme yöntemlerinin en bilinen örneği olup uzun yıllar birçok firma tarafından kullanılmıştır. Şelale yöntemi, genellikle ihtiyaçların çok iyi tanımlandığı ve ihtiyaç değişkenliğinin düşük olduğu ortamlarda etkili olmuştur. Ancak Şelale yönteminde müşterinin yazılım geliştirme çalışmalarına çoğunlukla dâhil olmaması ve süreç içerisinde değişen gereksinimlerin dikkate alınamaması, müşteri beklentilerinin eksiksiz karşılanmasını zorlaştırmakta, olası hataların geç fark edilmesine ve hata giderme maliyetinin artmasına neden olabilmektedir. Böylece, hızla değişen ihtiyaçlara kısa sürede yanıt verilebilmesi amacıyla 1990'lı yıllardan itibaren Çevik ("Agile") yazılım geliştirme yöntemleri yaygın olarak kullanılmaya başlanmıştır. Bu çalışmada, Şelale yönteminden çevik yöntem kullanımına geçen bir yazılım şirketinin deneyimleri aktarılmıştır. Çevik yöntem olarak öncelikle SCRUM metodolojisi benimsenmiş ve 5 yıl boyunca şirketin tüm yazılım geliştirme servislerinde SCRUM uygulanmıştır. Ancak zaman içerisinde SCRUM yönteminin kullanımında çeşitli sorunlarla karşılaşılmış, daha yalın bir diğer çevik yöntem olan KANBAN'a geçilmesi planlanmıştır. KANBAN yönetiminin şirkete uyarlanmasında, SCRUM yönteminin kullanımından elde edilen deneyimler kullanılmıştır. Bu bildiride 4 yıl Şelale, 5 yıl SCRUM yöntemi ile kazanılan deneyimler ve bu deneyimlere dayalı olarak özelleştirilen KANBAN yönteminin süreç tasarımı açıklanmıştır.

Anahtar Kelimeler: Yazılım Geliştirme, Şelale, Çevik Yöntem, Scrum, Kanban.

1 Yazılım Geliştirme Yöntemleri

Yazılım Yaşam Döngüsü ("Software Life Cycle") üretim ve müşteri tarafından kullanımı sırasında yazılımın geçirdiği tüm evreleri kapsar. Planlama, analiz, tasarım, kodlama, test, teslimat (yaygınlaştırma) ve bakım (saha izleme, sorun giderme) bu döngünün içindeki temel faaliyetlerdir. Planlama, temel ihtiyaçların belirlendiği ve

gerekirse fizibilite çalışmalarının yapıldığı safhadır. Analiz aşamasında; gereksinimler netleştirilmekte ve çıktılar belgelenmektedir. Belirlenen gereksinimlere göre müşterinin beklentilerini karşılamak amacıyla yazılımın özellikleri ve ara yüzü, tasarım adımında kurgulanır. Tasarımın sonrasında müşteriye teslim edilecek ürün kodlanır. Test aşamasında, manuel ya da otomatik olarak, test edilen yazılımın önceden belirlenen gereksinimleri karşılayıp karşılamadığı doğrulanır, beklenen ve gözlenen sonuçlar arasındaki farklar belirlenir. Test çalışmaları tamamlandıktan sonra kullanıcı kabul onayı alınmasını takiben müşteri tarafından kullanılabilmesi için yazılımın yaygınlaştırılması gerçekleştirilir. Bakım aşamasında, yaygınlaştırılan yazılımın kullanımı esnasında ortaya çıkan hatalara yönelik düzeltici aksiyonlar alınır [1].

Yazılım projelerinde amaç, bütçe ve takvim kısıtına uygun olarak müşteri gereksinimlerine ve beklentilerine uygun (kaliteli) ürünü geliştirmektir. Bu amaca ulaşmak için çeşitli yöntemler kullanılır. Şelale vb. geleneksel yöntemlerde ilk aşamada müşterinin tüm gereksinimleri netleştirilir, belirlenen proje kapsamı üzerinden proje yönetim planı yapılır, projede temel çizgi değişikliği gerektirecek büyüklükteki değişiklikler dışında, proje boyunca bu kapsam sabit tutulmaya çalışılır. Yazılım projesinde, yazılım geliştirme döngüsündeki tüm safhalar ardışık olarak işletilir. Bir sonraki basamağa geçebilmek için önceki adımlardaki tüm faaliyetlerin eksiksiz gerçekleştirilmiş olması gerekir. Genellikle gerçekleştirilen faaliyetler her adımın sonunda belgelenir. Sadece gereksinimlerinin netleştirildiği ilk safhalarda (örn. planlama, analiz) müşteri ile temasa geçilir. Bu nedenle, sonraki adımlarda ihtiyaçlar ile ortaya çıkabilecek olası değişiklikler genellikle göz önünde bulundurulmaz. Analiz, yazılım, test vb. çalışmalar genellikle farklı çalışanlar tarafından yerine getirilir [2].

Gereksinimlerdeki belirsizlik ve kaymanın yüksek olması, teknolojik gelişmelere bağlı olarak hızlı üretimin değer kazanması nedeniyle zamanla Şelale vb. geleneksel yaklaşımlar önemini kaybetmiş ve çevik yöntemler uygulanmaya başlanmıştır. [3] tarafından yayınlanan çevik manifesto; müşteriyi memnun etme, değişen ihtiyaçları karşılama, sık aralıklara ürün teslimi yapma, iş birimleri ve yazılımcıların birlikte çalışmalarını sağlama, ekibi motive etme, yüz yüze iletişime önem verme, çalışan ürünler ortaya çıkarma, sürdürülebilir gelişmeyi destekleme, kendi kendine organize olan takımlar kurma prensibine dayanır. Çevik yöntemlerin en yaygın örneklerinden biri, Ken Schwaber ve Jeff Sutherland tarafından 1990'lı yılların başında geliştirilen Scrum'dır. Şelale yönteminden farklı olarak Scrum'da; müşteri istekleri; 2-4 haftalık dönemler halinde planlanan sprintlerde geliştirilmektedir. Müşterilerin gereksinim tanımları Kullanıcı Hikâyesi ("user story") olarak nitelenir ve Özellikler Listesinde ("backlog list") tutulur. Böylece zaman kutulu ("time boxed") olan Scrum'da, her koşu ("sprint") sonunda yazılımın bir parçası tamamlanır ve müşteriye teslim edilir. Analiz, yazılım, test vb. işlemler için ayrı alt grupların bulunmadığı Scrum ekipleri, tek bir hedefe ulaşmak için mücadele eden ve sürekli iletişim halinde olan

çalışanlardan oluşur. Ürün Sahibi (“product owner”), Takım (“team”) ve Scrum Ustası (“master”) olmak üzere temelde 3 rol bulunur. Ürün sahibi, kullanıcı hikâyelerini belirler; takım yazılım yaşam döngüsü içinde yer alan faaliyetleri gerçekleştirir; Scrum ustası ise takımı organize edip ürün sahibine yardımcı olur. Çalışmalar; sprint planlama, günlük Scrum, gözden geçirme, geçmişe bakış (“retrospective”) vb. toplantılarında değerlendirilir. Scrum, özellikle 7 kişiden küçük ekiplerde daha etkin sonuçlar verir [4].

Diğer çevik yazılım geliştirme yöntemi olan Kanban ise, yazılım yaşam döngüsünün her safhasında anlık olarak üzerinde çalışılan işlerin sayısını kısıtlayarak sürekli üretim ve teslimata odaklanır. Scrum’dan farklı olarak Kanban zaman kutulu değildir. Böylece anlık talep ya da değişikliklerin dikkate alınması için Scrum’da olduğu gibi mevcut sprintin tamamlanmasına ihtiyaç duyulmamaktadır. Ayrıca Kanban’da ürün sahibi, takım ve usta gibi roller bulunmamakta, ekibi yönlendirmek amacıyla gerekirse çevik koç (“agile coach”) rolü atanabilmektedir. Scrum’dan farklı olarak Kanban hız yerine çevrim / teslim zamanına odaklanmaktadır [5].

Proje planlamanın başarıyla yürütülmesi, değişiklik isteklerinin düzenli teslimatı ve müşteriye sık sık yapılan geri bildirimler açısından Scrum, Kanban’a kıyasla avantajlı bulunmaktadır. Öte yandan devam eden iş (“work in progress”) sayısının azaltılması, sürecin görselleştirilmesi ve değişiklik yönetiminin etkin bir şekilde yürütülebilmesi açısından Kanban, Scrum’a tercih edilmektedir [6]. 330 personeli ile 12 ülkedeki 400 müşterisine hizmet veren İskandinavya’daki “Software Innovation” adlı yazılım işletmesinde 2009-2011 yılları arasında yapılan ve 12.000 isteğin analiz edildiği çalışmanın sonuçlarına göre; Kanban sayesinde Scrum’a kıyasla çevrim süresinin %50 ve hata (“bug”) sayısının %11 azaldığı, üretkenliğin %21 arttığı görülmüştür [7].

2 Bir Teknoloji Şirketinde Uygulanan Farklı Yazılım Geliştirme Yöntemleri

Bu çalışmada, yaklaşık 500 adet çalışanı olan bir bilgi işlem şirketinin son 10 yılda kullandığı yazılım geliştirme teknikleri, uygulama şekilleri ve karşılaşılan problemler detaylandırılıp sonuçlar analiz edilmiştir. Şirkette 2004-2009 yılları arasında Şelale yöntemi ile yazılım üretilmiştir. Geliştirilecek ana uygulamanın başlangıç düzeyindeki temel gereksinimlerinin net olduğu bu dönemde, klasik proje disiplini ile uyumlu olması ve yeni oluşturulmuş mühendislik ekipleri tarafından bilinen yöntem olması gibi pratik nedenlerle Şelale yöntemi benimsenmiştir. Yöntemin yapısına uygun olarak Analiz, Yazılım ve Test fonksiyonlarının her biri için farklı müdürlükler kurulmuştur. Tüm süreç ve kayıtlar Şelale yöntemine uygun olarak tasarlanmıştır. Buna göre her biri ayrı prosedür ile takip edilen süreçler tanımlanmıştır: *Gereksinim*

Belirleme ve Geliştirme, Analiz ve Fonksiyonel Tasarım, Tasarım Geliştirme, Kodlama, Fonksiyonel Test, Yazılım Bakım, Ölçme ve Değerlendirme, Uygulama Yazılımları Müşteri Kabulü. Her sürecin çıktısı bir sonraki adımda kullanılmak üzere belgelenmiş, böylece müşterinin isteklerinin karşılanmasına yönelik gerçekleştirilen tüm çalışmalar kayıt altına alınmıştır: *Kapsam Dokümanı, Fizibilite Raporu, Sözleşme (İmzalı Teklif), Analiz ve Fonksiyonel Tasarım Dokümanı, Fiziksel-Teknik-Fonksiyonel Tasarım Dokümanı, Test Talep Formu, Test Senaryo ve Bulgu Formu, Test-Fonksiyonel Tasarım Kontrol Listesi, Test Onay Formu, Analiz ve Tasarım Dokümanı, Revizyon Formu, Müşteri Kabul Onay Formu.* Hazırlanan dokümanların bir kısmı kurum içinde ilgili yönetici, bir kısmı ise müşterinin onayına tabi tutulmuştur. Ancak Şelale yönteminin uygulanması sırasında işletmede aşağıdaki sorunlar ile karşılaşılmıştır:

- Gereksinimlerin geliştirilmesi, analiz ve fonksiyonel tasarım çalışmalarının tamamlanması ve müşteri onayı alınması için geçen zamanın fazla olması,
- Yöntemin doküman ve kayıt odaklı olması nedeniyle oluşan çalışan memnuniyetsizliği,
- Gereksinimleri sabitleme gayretinin yarattığı müşteri memnuniyetsizliği,
- Basit taleplerin büyük paketler içinde karşılanmasına neden olan üretim yönteminin yarattığı taleplerin geç karşılanması sorununun oluşturduğu müşteri memnuniyetsizliği.

Öte yandan Şelale yönteminin uygulandığı 2008 yılında saptanan sorun sayısının yaygınlaştırılan talep sayısına oranının %6,82 ve obje sayısına oranının %0,75 olduğu görülmüştür. Ayrıca uygulama geliştirme kaynaklı çağrıların toplam çağrı sayısına oranı %3,12 olmuştur. Bu sorunların etkisiyle müşteri memnuniyeti beklenen düzeyin altında kalmış ve 2009 yılında işletmede yeni yöntem arayışına girilmiştir. Bu dönemdeki müşteri talepleri analiz edildiğinde çoğunlukla, Şelale yöntemi uygulanarak geliştirilen temel bankacılık sisteminin fonksiyonel olarak geliştirilmesinin hedeflendiği, müşteri gereksinimlerinin genellikle küçük veya nadiren büyük, ancak yüksek seviyede belirsizlik gösterdiği izlenmiştir. Gerek küçük taleplerin hızla karşılanması, gerekse belirsizliği yüksek büyük taleplerin kullanılabilir parçalarla geliştirilmesi için çevik yönteme ihtiyaç duyulduğu ve en uygun yaklaşımın Scrum olduğu düşünülmüştür. Böylece, 2009 yılında ilk olarak bir altyapı yazılımı servisinde yaklaşık 3 ay boyunca pilot olarak denendikten sonra Şelale yöntemine kıyasla başarılı sonuçlar verdiği görülen Scrum, kısa bir zaman içinde şirketin tüm yazılım geliştirme ekipleri tarafından kademeli olarak uygulanmaya başlanmıştır. Scrum'la birlikte analiz, tasarım, kodlama, test vb. adımlar ayrı müdürlükler tarafından değil, aynı takımlar içinde gerçekleştirilmesi planlanmıştır. Böylelikle şirketin organizasyon yapısı Scrum ile uyumlu hale getirilerek, fonksiyonel olarak uzmanlaştırılmış küçük servisler oluşturulmuştur. Servisteki toplam personel sayısına bağlı olarak her servis bir ya da iki Scrum takımı oluşturulacak şekilde teşkil edilmiş, böylece Scrum takımlarında görev alan personel sayısı 6 ila 12 arasında değişmek kaydıyla toplam 29 yazılım geliştirme servisi için 36 adet Scrum takımı kurulmuştur. Her servisin yöneticisi, ürün sahibi olarak

görevlendirilmiş ve kendi alanındaki müşteri ihtiyaçlarının özellik listesine dönüştürülmesinden sorumlu tutulmuştur. Başlangıçta, her Scrum takımındaki en deneyimli personel Scrum ustası olarak seçilip takım ve ürün sahibine yön vermiştir. Ancak zaman içinde, deneyimli personel üzerinde yarattığı iş yükü nedeniyle Scrum ustalığı, nispeten daha az deneyimli personele de yaptırılabilmiştir. Müşteri taleplerinin karşılanması için; Scrum yöntemine uygun olarak 2-4 haftalık dönemler halinde koşular (sprint) planlanmıştır. Takım içi iletişim ve Scrum'ın etkinliğini arttırmak amacıyla koşu planlama, gözden geçirme ve iyileştirme toplantıları ile ayaküstü günlük Scrum toplantıları gerçekleştirilmiştir. Sprint planlama toplantısı, Scrum ustası yönetiminde bir tam gün süreli olarak her sprintin başında düzenlenen ve ilgili sprintte hangi işler üzerinde çalışılacağına karar verilen toplantıdır. Sprint gözden geçirme toplantısı, sprint tamamlandıktan sonra, ürün sahibi (ilgili servis yöneticisi), Scrum ustası ve takımın katılımı ile gerçekleştirilen sprintte tamamlanan ürün özelliklerinin gözden geçirilerek kabul edildiği toplantıdır. Sprint gözden geçirme toplantısından sonra takımın, sprint boyunca yaptığı faaliyetleri değerlendirmek amacıyla sprint iyileştirme toplantıları düzenlenmiştir. Ayrıca ürün sahibi yönetiminde, her sabah tüm takımın katıldığı 10-15 dakika süreli ayaküstü günlük Scrum toplantıları gerçekleştirilmiştir. Şelale yönteminden farklı olarak yazılım yaşam döngüsündeki analiz, tasarım, kodlama vb. adımlar için ayrı ve detaylı doküman üretilmemiş; uygulama üzerinde ürün özellik listesi ve sprint indirgeme grafiği ("burndown chart") tutulmuş, böylece tüm takımların sprint iş listesinde yer alan çalışmalarının tamamlanma derecesi gösterilmiştir. Öte yandan, her talep için test senaryoları da uygulama üzerinden takip edilmiştir. Scrum yönteminin gereği olarak, sprint indirgeme grafiği ve kaynak bilgisine bağlı olarak hesaplanan hız ("velocity") yardımıyla tüm takımların performansı ölçülmüştür.

Yöntemin uygulamaya alındığı ilk yılda, Şelale yöntemine nazaran hızlı bir iyileşme sağlanmıştır. Şöyle ki; saptanan sorun sayısının yaygınlaştırılan talep sayısına oranı %6,82'den %6,12'ye ve obje sayısına oranı %0,75'ten %0,46'ya, uygulama geliştirme kaynaklı çağrıların toplam çağrı sayısına oranı %3,12'den %2,22'ye düşmüştür. Böylece üretim hızı ve müşteri memnuniyeti artmıştır.

Ancak son yıllarda yeniden müşteri memnuniyeti açısından sorun yaşanmıştır. Bu nedenle, 2014 yılı Eylül ayında şirket çapında Scrum ile ilgili kapsamlı bir değerlendirme çalışması yürütülmüştür. Çalışma sonucunda, Scrum yönteminin uygulamasında çeşitli sorunlar geliştiği belirlenmiştir. Şöyle ki;

- Takımların bir kısmının, Scrum Rehberi'nde tavsiye edilen en fazla 7 kişiden oluşma kuralına uymadığı görülmüştür.
- Scrum ustası rolü ek sorumluluk olarak görülmeye başlanmış ve istenmeyen bir görev haline gelmiştir.

- Gereğinden uzun süren Scrum toplantıları amaca hizmet etmekten uzaklaşıp durum değerlendirme şeklinde ilerlemeye başlamıştır.
- Scrum takımları içerisinde yetkinlik dengesi bozulmuş, sprint içerisindeki iş dağılımı homojen olmaktan uzaklaşmıştır.
- Anlık/acil müşteri talepleri nedeniyle ile genellikle sprint iş listesi bozulmuştur.
- Aynı takım için sprint süreleri sıkça değiştirilmiş ve takvime uyulmamıştır.
- Sprint planlama sırasında kullanılan öykü puanı ("story point"), işin zorluk derecesi yerine adam.gün maliyetine dayandırılmıştır.
- Sprint indirgeme grafiği çoğunlukla gerçek durumu yansıtmamış, takımların hızı yanlış ölçülmüştür.

Bu sorunların yanı sıra, Scrum ile birlikte özellikle test konusunda Şelale yöntemi kullanımına nazaran geriye gidildiği saha izleme sonuçları ile gösterilmiştir. Müşteri ile yapılan görüşmelerde elde edilen bilgiler analiz edildiğinde; kapsam, analiz, tasarım, test vb. çalışmaların klasik yöntemlerle belgelenmemesinin, müşteri tarafından test faaliyetlerinin yetersiz ve ürün kalitesinin düşük bulunmasına neden olduğu sonucuna varılmıştır.

Ayrıca son dönem müşteri gereksinimleri detaylı analiz edilmiştir. Tüm taleplerin yaklaşık %85'inin çoğunlukla eskimiş ana uygulama için iyileştirme nitelikli küçük işler olduğu görülmüştür. Scrum uygulamasındaki sorunlar ve talep yapısının zamanla büyük ölçüde iyileştirme nitelikli taleplere dönüşmesi nedeniyle sürekli ve hızlı üretim için yeni bir yöntem arayışına gidilmiştir.

Böylece işletmede 5 yıl boyunca uygulanan Scrum yönteminden vazgeçme ve yeni bir yazılım geliştirme yöntemi arama kararı alınarak, uygulama geliştirme servis yöneticilerinden bir çalışma grubu oluşturulmuş ve yaklaşık 3 ay süreli bir çalışma sonunda Kanban yönteminden uyarlanan yeni süreçler tasarlanmıştır. Kanban, Scrum'dan farklı olarak zaman kutulu olmayıp sürekli üretimi desteklemektedir. Bu nedenle, yeni bir işin planlanabilmesi için mevut sprintin tamamlanması gereken Scrum'dan farklı olarak takımların iş yüküne göre herhangi bir zamanda yeni bir çalışma yürütülebilmekte, böylece verimlilik arttırabilmektedir. Ayrıca daha esnek bir üretim yöntemi olan Kanban'da, Scrum'dan farklı olarak ürün sahibi, takım, usta vb. roller bulunmamakta ve toplantılar düzenlenmemektedir. İşletmede Kanban uygulaması olarak; her bir uygulama geliştirme servisinin havuzunda (kuyrukta bekleyen işler arasında) yer alan taleplerin öncelikle ilgili servis yöneticisi tarafından özellik listesine dönüştürülmesi; sonra analiz, yazılım, test ve yaygınlaştırma faaliyetlerine göre görev listesinin oluşturulması, her bir görevin adam.gün cinsinden maliyetinin belirlenerek iş planına dahil edilmesi önerilmiştir. Planlama döneminde ilgili servis çalışanlarının iş yükü ve devam eden iş sayısı kısıtına göre görevlerin atanması, böylece üretimin esnek ve hızlı bir şekilde gerçekleştirilmesi düşünülmüştür. Kanban yönteminde, her iş için bir analiz dokümanı hazırlanması, bu dokümanda yer alan bilgilerin yazılım çalışmaları öncesinde talep sahibine onaylatılması tasarlanmıştır. Öte yandan test senaryo bilgilerinin uygulama üzerinden takip edilmesi planlanmıştır. Böylece bir taraftan Şelale yönteminde olduğu gibi çok

sayıda doküman üretmekten kaçınılmış ve olası zaman kayıplarının engellenmesi hedeflenmiş; diğer taraftan Scrum yönteminde hiç doküman üretilmemiş olması nedeniyle ortaya çıkan müşteri memnuniyetsizliğinin bertaraf edilmesi planlanmıştır. İşletme, 2015 yılı Mayıs ayı itibariyle Kanban yöntemi ile yazılım geliştirmeye başlamıştır.

Scrum uygulanan 1 Mayıs – 15 Haziran 2014 tarihi arasında yapılan yaygınlaştırmalardaki 3.015 nesne üzerinden 5.791 atama için 3.223 kod kalitesine dair bulgu tespit edilmiştir. Kanban uygulanan 2015 yılının aynı döneminde ise, yaygınlaştırılan 1.594 nesne üzerinden 2.507 atama için 818 kod kalitesi bulgusu ortaya çıkmıştır. Böylece Kanban sayesinde nesne başına bulgu sayısının %107'den %51'e, atama başına bulgu sayısının ise %56'dan %33'e düştüğü görülmüştür.

3 Sonuç

Bu çalışmada, değişen müşteri gereksinimlerinin niteliğine bağlı olarak yazılım geliştirme yöntemi açısından Şelale yaklaşımından çevik yapıya yönelen bir işletmenin deneyimleri aktarılmıştır. Ana uygulamanın ömür döngüsü içinde yeni oluşturulan taleplerin nitelik değiştirmesi ile Şelale yöntemi ile müşterinin hızlı üretim beklentisi karşılanamamıştır. Bu tespitle, çevik yöntem ile yazılım geliştirme kararı alan işletme, değişkenliğin yükselmeye başladığı ve daha çok küçük ek iyileştirme taleplerinin karşılandığı 5 yıl boyunca Scrum yöntemini uygulamıştır. Scrum'da ise, temelde doküman görmeyen müşterinin kalite beklentisi karşılanamamış, yöntemin yeniden gözden geçirilmesine ihtiyaç duyulmuştur. Uzun yıllardır kullanılan temel üretim uygulaması üzerinde alınan müşteri taleplerinin artık çoğunlukla küçük eforlu işler olması nedeniyle ve müşterinin hızlı sonuç, daha yüksek kalite beklentisini karşılamak için Kanban yöntemi uyarlanarak yeni süreç tasarımına geçilmiştir. Kanban uygulaması ile birlikte kalite ölçümü de yapılan bir performans yönetim sistemi kurulmuştur. Kanban yöntemi ile alınan ilk sonuçlar kalite artışını göstermektedir.

Kaynaklar

[1] Pressman, R. S., "Software Engineering: A Practitioner's Approach" Mcgraw/Hill, 6th Ed., 2005.

[2] Tsui, F., Karam, O., Bernal, B., "Essentials of Software Engineering", Jones and Bartlett Publishers, Inc., USA, 2013.

[3] Beck, Kent ve diğerleri, "Principles Behind the Agile Manifesto" Adres: http://www.agilemanifesto.org/principles.html, Erişim Tarihi: 28.04.2015.

[4] Cohen, D., Lindvall, M., Costa, P., "An Introduction to Agile Methods. In Advances in Computers", 1–66, New York: Elsevier Science, 2004.

[5] Anderson, David, “Kanban - Successful Evolutionary Change for your Technology Business.” Blue Hole Press, 2013.
[6] Mahnic, V, “Improving Software Development through Combination of Scrum and Kanban”, Recent Advances in Computer Engineering, Communications and Information Technology, Proceedings of the 8th WSEAS International Conference on Computer Engineering and Applications, 281-288, 2014.
[7] Sjøberg, D. I. K, Johnsen, A., Solberg, J, “Quantifying the Effect of Using Kanban versus Scrum: A Case Study”, IEEE Software, Vol. 29, No: 5, pp. 47-53, 2012.