Yazılım Projelerinde Kazanılmış Değer Yönetimi Kullanımı

Pınar Efe[1], Onur Demirörs[2]

[1,2] Enformatik Enstitüsü, Ortadoğu Teknik Üniversitesi, Ankara
[1]pinar.efe@gmail.com, [2]demirors@metu.edu.tr

Özet. Kazanılmış Değer Yönetimi (KDY) proje yönetiminde yaygın olarak kullanılan bir performans ve geri bildirim aracıdır. KDY, projenin mevcut durumunu ve ilerlemesini kapsam, zaman ve maliyet boyutları açısından net bir şekilde ortaya koyar. Projenin gidişatına ve mevcut eğilimlere dayanarak, gerçek maliyeti ve olası tamamlanma zamanını tahmin etmemizi sağlar.

KDY proje yönetiminde yaygın olarak kullanılmasına rağmen, gerek ülkemizde, gerekse dünyada yazılım projelerinde sıkça kullanılan bir yöntem değildir. Bu çalışmada KDY detaylı bir şekilde anlatılmış, yazılım projelerinde KDY kullanılmasının faydaları ve mevcut uygulama zorlukları araştırılmış, iki yazılım projesine uygulanma süreci ve sonuçları da incelenerek yazılım projelerinde KDY kullanımı ile ilgili sorunlar tartışılmıştır.

Anahtar Kelimeler. Kazanılmış Değer Yönetimi, Yazılım Proje Yönetimi, Performans Ölçümü, Proje İzlenmesi

1 Giriş

Kazanılmış Değer Yönetimi, proje yönetiminde sıklıkla kullanılan sayısal bir performans yönetimi aracıdır. İki temel hedefi vardır: ilki projenin maliyeti ve takvimi açısından mevcut durumunu ortaya koymak, ikincisi ise mevcut duruma ve maliyet, zaman eğilimlerine bakarak projenin geleceğini tahminlemek. En basit haliyle "Kazanılmış Değer" (KD) kavramı projenin herhangi bir anında harcadıklarımıza karşı kazandıklarımızı göstermektedir. KDY projenin durumunu ve gidişatını açık ve tarafsızca gösterdiği için, Proje Yönetimi Enstitüsü (PYE) bu yaklaşım için "aydınlıkta yönetim" benzetmesini kullanılmıştır [18]. KDY olası bütçe aşımı ve takvim gecikmeleri için bir erken uyarı sistemidir.

KDY, 60'lı yılların başından itibaren başta Amerika Birleşik Devletleri (ABD) olmak üzere dünyada bir çok ülkede, farklı ölçeklerde ve zorluklarda projeler için kullanılmaktadır. KDY yaklaşımının proje yönetimi dünyasındaki bu yaygın kullanıma rağmen yazılım projelerinde kullanımı nadirdir. Genel olarak, geleneksel proje yönetiminde kullanılan araçlar, teknikler ve yöntemler yazılım projelerinde de kullanılmaktadır. Ancak yazılım projeleri geleneksel projelere göre birçok farklılık barındırdığından, geleneksel proje yönetim araçlarının yazılım projelerinde aynı şekilde kul-

lanılması yeterli olmamaktadır [16]. Yazılım projelerinin gereksinimleri göz önüne alınarak bu yöntemlerin uyarlanması yerinde olacaktır.

Bu çalışmanın amacı yazılım projelerinde KDY kullanımını incelemek, faydaları ve zorluklarını tartışmak, kullanım azlığının nedeni olabilecek farklılıkları araştırmak ve yöntemin yazılım projelerinin gereksinimleri doğrultusunda iyileştirilmesi konusunda yol gösterici olmaktır.

Bildirinin geri kalanı şu şekilde düzenlenmiştir: 2. bölümde KDY'nin özet tarihçesi ve yöntemin detayları, 3. bölümde yazılım projelerinde KDY kullanımı ile ilgili olarak yapılmış literatür araştırmasının sonuçları anlatılmış, 4. bölümde yapılmış olan araştırmalara ve yazarların gerçekleştirdiği durum çalışmasının sonuçlarına dayanarak yazılım projelerinde KDY kullanımının faydaları ve zorlukları tartışılmıştır. Son olarak 5.bölümde çalışmanın sonucu özetlenmiştir.

2 Kazanılmış Değer Yönetimi

2.1 KDY Tarihçesi

KDY, proje yönetiminde resmi olarak 60'lı yıllardan itibaren kullanılmasına rağmen, "Kazanılmış Değer" (KD) kavramının en temel haliyle ilk Amerikan fabrikalarındaki endüstriyel üretimde kullanımı çok daha eskilere, 1800'lü yılların sonlarına uzanmaktadır.

KDY, PERT/Cost yönteminin parçası olan bir proje yönetim aracı olarak, ilk defa ABD Donanması tarafından 1962 yılında kullanılmıştır. Daha sonra, 1967 yılında ABD Savunma Bakanlığı KD yaklaşımını 35 kriteri içeren "Cost/Schedule Control Systems Criteria (C/SCSC)" adıyla resmi olarak yayımlamış ve geliştirilen sistemlerde kullanımını zorunlu kılmıştır. Özel sektör ise karmaşıklığından dolayı bu yöntemi benimsememiş, zorunluluklar dışında 90'ların ortasına dek kullanmamıştır. Bu nedenle 1995 yılında ABD Savunma Bakanlığı, kullandığı bu karmaşık KD yaklaşımını incelemek, sadeleştirmek ve yeniden yazmak üzere bir grup oluşturmuş, bu grubun çalışmaları sonucu 1997 yılında sadeleştirilen 32 kriteri içeren yeni KDY Sistemi'ni (KDYS) yayımlamış ve ardından 1998 yılında KDYS, Amerikan Ulusal Standartlar Enstitüsü tarafından standart olarak yayımlanmıştır [1]. Böylece KDY, ABD'de devlet organizasyonları tarafından zorunlu tutulan bir yöntem olmak yerine özel sektör tarafından tercih edilen bir yöntem olurken, bunun yanında özel sektör de yöntemin gelişmesine katkıda bulunmaya başlamıştır.

Sadeleştirilen KDY yönteminin ABD'de kullanımı hızla yayılmış, Amerikan ordusunun yanı sıra Ulusal Havacılık ve Uzay Dairesi (NASA), ABD Enerji Bakanlığı gibi diğer devlet kurumları ve özel sektör tarafından kurumsal ihtiyaçlarına uygun olarak uyarlanmıştır. Yöntem KDY uzmanlarının yanı sıra proje yöneticileri tarafından da anlaşılmaya başlanmıştır. İnşaat sektörü bu yöntemin özel sektördeki ilk kullanıcılardan olmuştur [19].

KD kavramı PYE tarafından yayımlanan 'PMBOK Kılavuzu'nunda ilk sürümlerinden itibaren yer almış ve sonraki sürümlerinde de geliştirilmiştir [18]. 2005 yılında ise PYE tarafından KDY yönteminin kullanımını kolaylaştırmak ve yaygınlaştırmak üzere PMBOK kılavuzuna tamamlayıcı "Practice Standard of Earned Value Manage-

ment" adıyla bir KDY uygulama standardı yayımlamıştır [17]. Bu standart proje performasını ölçmek ve gidişatını tahminlemek üzere proje yöneticilerine basit ve pratik bir yol sunmaktadır.

KDY kullanımı daha sonra Avustralya, Kanada, İsveç gibi ülkelerde de yaygınlaşmış, Avustralya'da 2003 yılında ilk ulusal KDY standardı yayımlanmıştır [2].

Organizasyonların KDY uygulama yetkinliğini ölçmek amacıyla bir olgunluk modeli fikri 2000 yılında ortaya atılmıştır. EVM3 adı verilen bu model 5 seviyeli, kesikli bir modeldir ve organizasyonların KDY kullanımını değerlendirme ve geliştirme olanağı sunmaktadır [25]. ANSI/EIA-748 standardı ile uyumlu KDYS kullanan organizasyonlar KDY metrikleri ve iyileştirme planlarını kullanmak üzere EVM3 kullanabilirler.

Kazanılmış Takvim yöntemi KDY yöntemine ek olarak sunulmuş başka bir yöntemdir [15]. Burada temel amaç, mevcut KDY yönteminde bulunan takvim ile ilgili problemlerin giderilmesidir. Yöntemin yaratıcısı Lipke, mevcut KDY metriklerine ek olarak Kazanılmış Takvim ismini verdiği yeni bir metrik tanımlamış, bu metrikle proje takviminin ilerlemesini ölçerken bütçe yerine zaman ölçülmesi fikrini temel almıştır. Ayrıca, burada *Takvim Sapması* ve *Takvim Performans Endeksi*, *Kazanılmış Takvim* metriğine bağlı olarak hesaplanmaktadır.

2.2 KDY Yöntemi

KDY, PMBOK içerisinde "kapsam, zaman çizelgesi ve kaynakları entegre etmek ve proje performansını ve ilerlemeyi nesnel bir şekilde ölçmek için kullanılan bir yönetim metodolojisi" olarak tanımlanmıştır [18].

PYE, yayımladığı standart içerisinde KDY yaklaşımını en basit ve kolay kullanılabilir formuyla tanımlamaktadır. Bu standart proje yöneticilerine kolayca uygulanabilir bir yöntem sunmaktadır. KDY, projenin başarılı olmasında çok kritik önem taşıyan şu sorulara cevap sunmayı amaçlar [17]:

- Proje takvimin önünde miyiz, gerisinde miyiz?
- Zamanı ne kadar verimli kullanmaktayız?
- Proje tahmini olarak ne zamana tamamlanacak?
- Planlanan bütçeyi aştık mı?
- Bütçeyi ne kadar verimli kullanmaktayız?
- Kalan işler ne kadar bütçeyle tamamlanacak?
- Projenin tamamı ne kadar bütçeyle tamamlanacak?

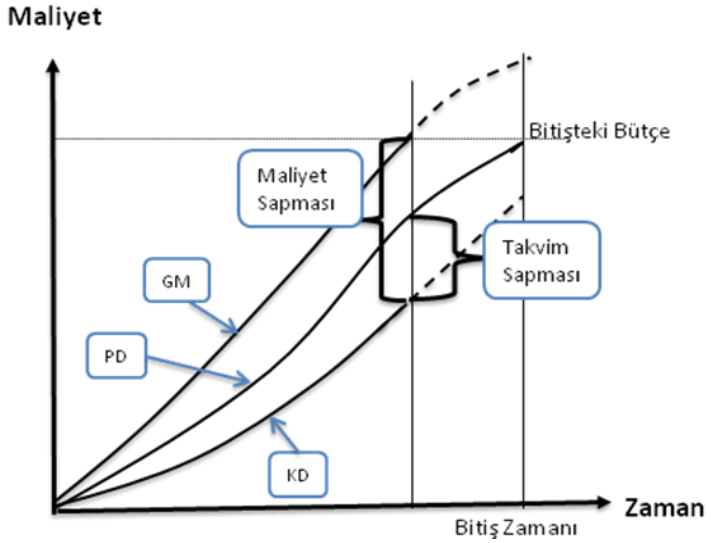
KDY, üç temel veri üzerine inşa edilmiştir; **Planlanmış Değer (PD), Kazanılmış Değer (KD)** ve **Gerçekleşen Maliyet (GM)**.

Planlanmış Değer, proje takviminin herhangi bir tarihine kadar planlanan tüm işlerin toplam bütçesidir. Planlanan İşin Bütçelenmiş Maliyeti olarak da bilinir. Projenin performası PD üzerinden ölçülür. Genel olarak zamana karşı bütçeyi gösteren S şekilli bir eğri grafiği ile gösterilir.

Kazanılmış Değer, projenin herhangi bir anında tamamlanmış olan işlerin toplam değeridir. Yapılan İşin Bütçelenmiş Maliyeti olarak da bilinir. Bu değer PD cinsinden ölçülür.

Gerçekleşen Maliyet, projenin herhangi bir anında tamamlanmış olan tüm işler için harcanmış olan kaynakların toplamıdır. Yapılan İşin Gerçek Maliyeti olarak da bilinir.

KDY'nin diğer metrikleri olan tüm sapmalar, endeksler ve tahminler bu üç temel veriyi kullanarak hesaplanır. Temel KDY metrikleri aşağıda grafiksel olarak gösterilmiştir (bknz. **Şekil 1**). Sapmalar, bu temel verileri karşılaştırarak projenin mevcut durumunu gösterir. Projelerde mevcut durumu değerlendirmek üzere sapmalar düzenli olarak ölçülmeli, raporlanmalı ve düzeltici faaliyetlerle düzenlenmelidir. Endeksler, bütçenin ve zamanın ne kadar verimli kullanıldığının göstergesidir, projedeki ilerleme eğilimini temsil eder. Geçmişteki eğilimler geleceği belirler ilkesinden yola çıkarak, endeksler gelecek tahminlemesi için de kullanılır. Projenin tamamlanma tarihi ve ne kadar bütçeyle tamamlanacağı bilgileri endeksleri kullanarak tahmin edilir.



Şekil 1. Temel KDY metrikleri

Maliyet Sapması (MS) belli bir zaman için tamamlanmış olan işlerin değeri ile gerçekleşen maliyetinin arasındaki farktır. Maliyet Sapması proje maliyetinin planlanmış bütçenin altında ya da üstünde olduğunu gösterir. Maliyet sapmasının eksi değerde olması proje maliyetlerinin planlanan bütçeyi sapma miktarı kadar aştığına işaret eder.

MS= KD - GM

Takvim Sapması (TS) belli bir zaman için planlanmış olan işlerin değeri ile tamamlanmış olan işlerin değeri arasındaki farka karşılık gelir ve projenin takvimin ne kadar önünde ya da arkasında olduğunu gösterir. Takvim sapmasının eksi değeri sadece projenin planan takvimden geride olduğunu gösterir, doğrudan ne kadar geciktiği ile ilgili olarak bilgi vermez.

TS = KD - PD

Maliyet Performans Endeksi (MPE) mali kaynakların ne kadar etkin kullanıldığını gösterir ve tamamlanmış olan işlerin değerinin gerçek maliyetine bölünmesiyle elde edilir.

MPE = KD / GM

MPE = 1 -> Maliyet performansı tam, gerçekleşen işin maliyeti bütçelenen maliyete eşit

MPE < 1 -> Gerçekleşen işin maliyeti, planlanmış bütçenin üstünde

MPE > 1 -> Gerçekleşen işin maliyeti, planlanmış bütçenin altında

Takvim Performans Endeksi (TPE) ise projedeki zamanın ve takvimin ne kadar etkin kullandığını gösterir. TPE, tamamlanmış olan işlerin değerinin planlanmış değere bölünmesiyle hesaplanır.

TPE = KD / PD

TPE = 1 -> Takvim performansı tam; gerçekleşen işin maliyeti planlanmış işin bütçelenen maliyetine eşit

TPE < 1 -> Gerçekleşen ilerleme planlanandan daha yavaş, düzeltmek için işin zamanında gerçekleştirilmesine odaklanılmalı

TPE > 1 -> Gerçekleşen ilerleme planlanandan daha hızlı

Bitişteki Bütçe (BB) projede planlanmış tüm işlerin bütçelerinin toplamıdır, projenin maliyet temelini temsil eder.

Kalan İşin Değeri (KİD) projede tamamlanmamış tüm işlerin tahmini maliyetini temsil eder. Geçmişte planlanmış değerlerin geçerli olmadığı ya da güncelleme gerektirdiği durumlarda hesaplanır.

Tahmini Bitiş Maliyeti (TBM) projenin tamamlanma zamanı için öngörülen maliyetidir, performans endeksi kullanılarak ya da hesaplamanın yapıldığı güne kadar gerçekleşen maliyetlere (GM) kalan işin değeri (KİD) eklenerek hesaplanır.

TBM = GM + KİD

TBM = BB / BVİ

Bitişteki Maliyet Sapması (BMS) ise hesaplandığı tarihte projenin tamamlanma tarihi için öngörülen maliyeti ile planlanmış toplam bütçesi arasındaki farktır.

BMS= BB - BTB

3 Yazılım Projelerinde KDY Kullanımı ile İlgili Araştırmalar

Proje yönetim dünyasında sıklıkla kullanılan KDY yaklaşımının, yazılım projelerinde kullanımı çeşitli araştırmaların konusu olmuştur. Bazı araştırmacılar KDY'nin tüm proje çeşitleri için uygulanabilir bir yöntem olduğunu, yazılım projelerinin herhangi bir farklılık içermediğini iddia ederken, diğer yandan daha büyük bir grup ise yazılım projelerine has zorluklara değinmiş ve KDY kullanırken bunlarla ilgili olarak bir takım önerilerde bulunmuşlardır.

En çok kullanılan KDY kitaplarının başında gelen "Earned Value Project Management" kitabının [11] yazarları Fleming ve Koppelman, KDY'nin yazılım projele-

rinde de aynen diğer disiplinlerde olduğu gibi uygulanabileceğini iddia etmiş, herhangi bir uygulama farklılığına değinmemişlerdir [12]. Bunun yerine KDY'nin tüm projelerde başarılı bir şekilde uygulanması için yapılması gereken on adım listesi yayımlamışlar, bu listedeki yapılması zorunlu olarak bahsedilen maddelere uyulduğu sürece büyük ya da küçük ölçekli her disiplinden projenin başarıyla yönetileceğini belirtmişlerdir.

Diğer yandan Ferle [10] yazılım proje yönetiminin tamamen diğer proje disiplinlerden farklı olduğunu iddia etmiş, yazılım projelerine has zorluklardan bahsetmiştir. Yazılım projelerine KDY uygulamanın çok kolay olmadığını ve projenin planlama, izleme ve raporlama gibi çeşitli fazlarındaki karmaşıklıkları KDY açısından değerlendirmiştir. Yazılım projelerinde büyüklük tahminlemenin oldukça zor bir iş olduğunu, hatta bazı durumlarda imkânsız olduğunu iddia etmiş, bu zorlukların temelinde de yazılım projelerinde insan bağımlılığının olduğunu öne sürmüştür. Yazılım projeleri gerek kestirimlerin yapılması sırasında, gerekse gerçekleştirme süresince çalışanların yeteneklerine son derece bağlıdır. Brooks'a göre [6] geliştiricilerin yeteneklerine bağlı olarak verimlilik açısından en iyi ile en kötü arasında on kat fark oluşabilir. Ancak diğer taraftan KDY'nin başarılı bir şekilde uygulanması için de yapılan işlerin izlenmesi ve kontrol edilmesi süreçlerinde ilerlemenin görülebilirliği, nesnel olarak ilerlemenin, performasın ölçülebilmesi ve ilk kestirimlerin doğruluğu oldukça önemlidir [10].

Hanna ise yazılım yönetimine özel zorlukları ve bu zorlukların KDY yöntemine olan etkilerini incelemiştir [13]. Bu zorlukları "İnovasyon ve Prototipler", "Hatanın Bulunması ve Çözümü" ve "Mimari Değişiklikleri" olmak üzere üç ana grup altında toplamıştır. Bunların içerisinde de yazılım mühendisliğin temel zorluklarının kapsamlı bir özetini sunmuş, proje takviminde, kalite faktörlerinde, tasarımsal konularda mevcut olan belirsizlikler ve bu belirsizlikleri çözmek ve ölçmek için kullanılan tekniklerin yazılım projelerinin mevcut en büyük zorlukları olduğunu belirtmiştir. Yazar, daha sonra bu zorluklar bağlamında yazılım projelerinde KDY uygulanmasını daha etkin hale getirmek için çeşitli önerilerde bulunmuştur.

"Performansa Dayalı Kazanılmış Değer" yönteminin yaratıcısı olan Solomon ise KDY yönteminin esas eksikliklerine odaklanmış, Northrop Grumman isimli silah sistemleri geliştiren bir şirkette uygulanan pratiklere ve alınan derslere de dayanarak olası iyileştirmeler üzerine çalışmış, yayımladığı çeşitli makalelerde ve kitabında yöntemini anlatmıştır [20][22][23][24].

Solomon KDY yaklaşımının birçok sınırlandırmaları olduğundan bahsetmiştir [21]. Öncelikle, KD yapılan işin nitelik ölçüsü değil, nicelik ölçüsüdür. Yöntem bu yönde bir şeyi açıkça hesaba katmadığından işin niteliğinin de istenen koşullarda olması proje yöneticisinin sorumluluğundadır. KDY yalnızca projenin kapsamını dikkate alır, ürünün kapsamıyla ilgilenmez. KD türetilmiş bir ölçüdür, dolayısıyla teknik ve mali performansı bütünleştirme verimliliği onu oluşturan temel ölçülerin doğruluğu ve güvenilirliğine bağlıdır. KDY'nin böyle bir amacı olmamasına rağmen bir risk aracı gibi algılanır. KDY çok kesin ve mutlak ölçülebilir ölçülere ihtiyaç duymaz. Örneğin, işin tamamlanma yüzdesi tamamen proje yöneticisinin değerlendirmesi ile belirlenebilir [22].

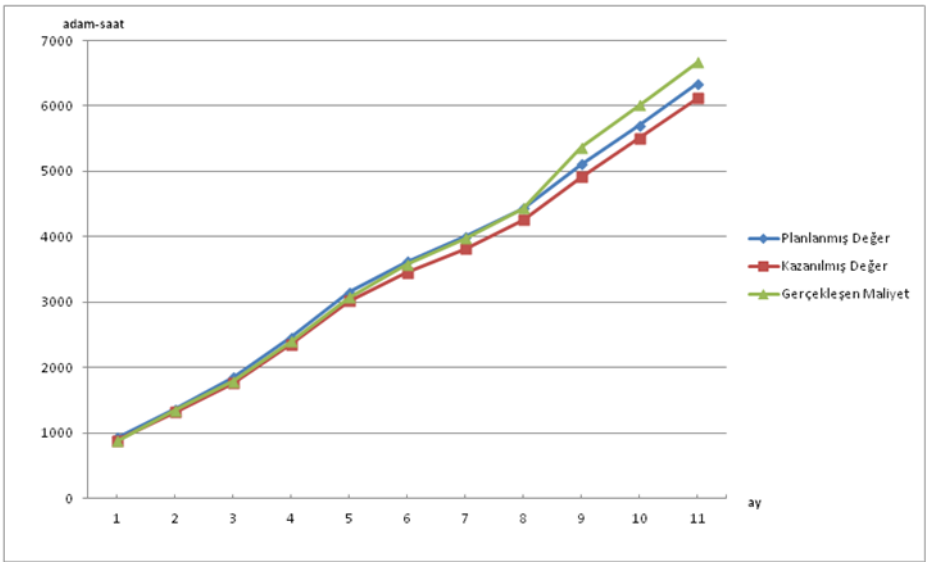
Bu kısıtlamaların ve eksikliklerin giderimesi amacıyla Solomon CMMI, PMBOK, EIA 632 ve IEEE 1120 standartlarını kullarak kendi deneyimlerinden faydalanarak, oluşturduğu "Performansa Dayalı Kazanılmış Değer" yönteminin aktivitelerini adım adım tanımlayarak bir kılavuz kitap oluşturmuştur. Bu yöntemin KDY'den ayırt edici özelliği müşteri gereksinimlerine odaklanması, ürün kapsamını ve kalite gereksinimlerini de dikkate almasıdır. Proje planında, kapsam gereksinimlerinin yanı sıra kalite gereksinimleri de yer alır ve ilerlemeyi ölçerken ürünün kalite gereksinimlerini de yerine getirecek performansa dayalı ölçüler tanımlanır, bu ölçüler KDY'nin temel ölçüleri olup, projedeki ilerleme ürün kapsamı ve kalite gereksinimleri üzerinden bu temel ölçüleri kullanarak ölçülür [23].

KDY'nin yazılım projelerinde kullanılan başka bir uzantısı ise bilinen çevik yöntemlerle beraber kullanılan Çevik KDY yaklaşımıdır. Çevik yöntemlerle KDY kullanımı ilk olarak 1998 yılında Lockheed-Martin firmasından Steven H. Lett tarafından önerilmiş olup [14], daha sonra Alleman, Henderson ve Cockburn'ün katkılarıyla geliştirilmiştir [3][4][7]. Çevik KDY, klasik KDY'nin Scrum'da tanımlanan değerler kullanılarak uyarlanmasıyla oluşturulmuş olup klasik KDY'den daha basit KD hesaplamaları içerir. Sulaiman ve dig ise Çevik KYD'yi detaylı olarak tanımlayıp klasik KDY ile karşılaştırmış, terimleri ve hipotezleri açıklamış, bunun yanında da Çevik KDY'yi iki ayrı projede uygulayarak elde ettiği deneysel doğrulama testlerinin sonuçları sunmuşlardır [26].

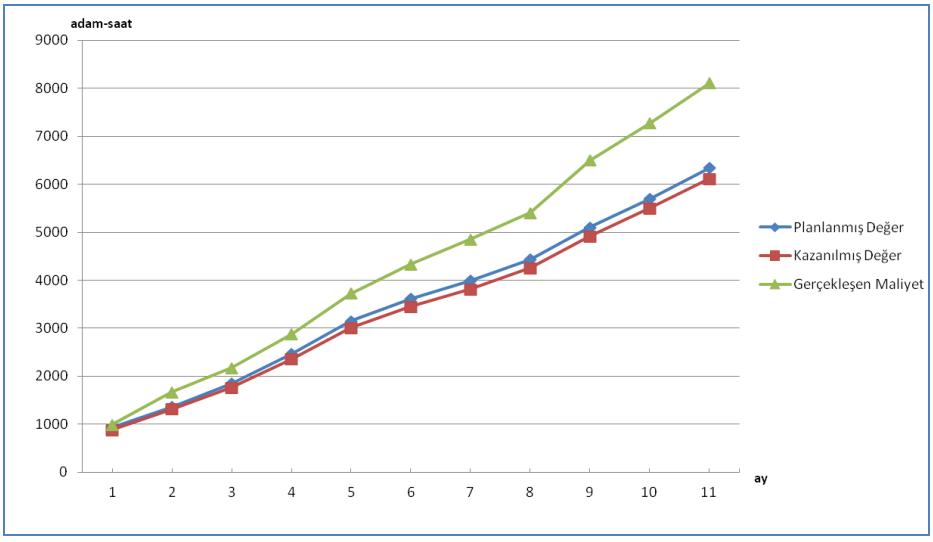
4 Yazılım Projelerinde KDY

Yazılım projelerinde KDY kullanımı ile ilgili problemleri analiz etmek üzere yapılmış başka bir çalışmada [9], bu bildirinin yazarları iki farklı tip yazılım projesinde KDY uygulamış ve sonuçları analiz etmişlerdir. Projelerin biri yinelemeli, artırımlı bir ürün geliştirme projesi olup, diğeri ise bir şelale modeli ile geliştirilmiş bir uygulamadır.

Yapılan durum çalışmalarına detaylı bakacak olursak ilk projede, her ay tamamlanan ürün sürümleri içerisinde önceki sürümlerde tamamlanmış özelliklere dair hatalar ve değişiklikler gözlenmektedir. Proje planında öngörülmeyen bu hatalar ve değişiklikler, kimi zaman projede o ay için harcanan zamanın % 40'ına eşit ciddi bir zamanda çözülmektedir. Çalışmada, bu hataları dikkate almayarak yalnızca projede o sürümlerde planlanan yeni özellikler için KDY uygulandığında, sonuçlar oldukça başarılıdır: Mevcut proje durumu ve ilerlemesi açıkça görülmekte, olası proje tamamlanma tarihleri, bütçesi hakkında başarılı bir sonuç ortaya çıkmaktadır. Ancak KDY, bu uygulama sonradan ortaya çıkan hatalar ve değişiklikler için herhangi bir ipucu dahi vermemekte, bu hatalardan ve değişikliklerden kaynaklanan gecikmeleri ve bütçe aşımlarını ölçememekte, dolayısıyla bu faaliyetleri kontrol altında tutamamaktadır (bknz. Şekil 2). Hatalar ve değişiklikler hesaba katılarak KDY uygulandığında ise projede bütçe açısından bir performans problemi gözlenmekte, ancak bunun nedeni anlaşılamamakta ya da yanlış yorumlanabilmektedir (bknz. Şekil 3). Bir diğer problem ise herhangi bir anda tamamen bittiği düşünülen bir iş için daha sonra yeniden zaman ve emek harcanması, dolayısıyla o iş için ilk hesaplanan kazanılmış değerin yanlış olmasıdır. KDY, yeniden yapma faaliyetleri konusunda yetersiz kalmaktadır.



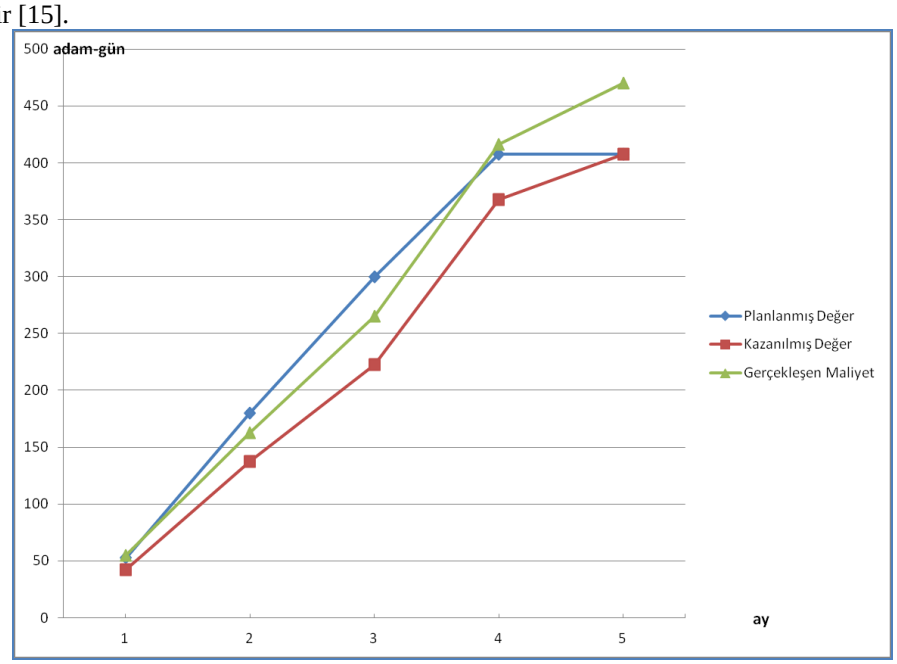
Şekil 2. İlk proje yeni özellikler için KDY uygulaması



Şekil 3. İlk proje yeniden yapma faliyetlerini de içeren KDY uygulaması

İkinci durum çalışmasında ise şelale modeli dolayısıyla uzun bir geliştirme safhası bulunur ve ardından test safhası gelir. Burada geliştirme safhası boyunca projedeki kötü gidişatı KDY açıkça göstermiş, takvimdeki gecikmeyi ve bütçe aşımını önceden hesaplayabilme imkânı sunmuştur. (bknz. Şekil 4) Ancak bu resimde de geliştirme

safhasının 1 aylık ertelenmesi sonrasında geliştirme safhası sonunda projede bütçe aşımı olmasına rağmen zamanında bitmiş gibi görünmektedir. Bu da KDY uygulamasının geciken projelerdeki takvim konusundaki başka bir problemini göstermektedir. Lipke tarafından sunulan Kazanılmış Takvim yöntemi de bu problemi adreslemektedir [15].



Şekil 4. İkinci proje için KDY uygulaması

Bu projede yeniden yapma faaliyetleri en başından öngörülmüş ve proje planına dahil edilmiştir. Fakat burada öngörülenden çok daha fazla bir hata çözme emeği gerekli olmuş, bu gereklilik de ancak projede hata çözme faaliyetleri içerisinde anlaşılabilmiştir. KDY metrikleri bununla ilgili herhangi bir ipucu verememektedir. İkinci projede geliştirme safhası boyunca ciddi kalite problemleri oluşmuş, bu problemler hem geliştirme safhası içerisinde hem de test safhası boyunca yeniden yapma faaliyetlerinin temel nedeni olmuş ve projenin ilerlemesini büyük ölçüde etkilemiştir. Yöntem bu yeniden yapma faaliyetleri dolayısıyla bu projenin yönetiminde de başarılı olamamıştır.

KDY diğer proje disiplinlerinde olduğu gibi yazılım projelerinde de projenin durumunu net bir şekilde ortaya koyarak proje kontrolüne ciddi katkılar sağlar. Yapılan durum çalışmalarında da görüldüğü üzere, özellikle yeni geliştiren özellikler için projenin nasıl ilerlediğini, projedeki gecikmeleri ve bu gecikmelerin bitişi nasıl etkileyebileceğini açıkça gösterir, proje yöneticisine gerekli önlemleri alma fırsatı sunar. Buna rağmen yazılım projelerinde KDY uygulanması, yazılım projelerinin doğasından kaynaklanan farklılıklar nedeniyle özellikle yeniden yapma faaliyetleri açısından önemli bir takım sorunlar barındırır.

Öncelikle KDY'nin başarıyla uygulanmasının temel koşulu organizasyonun proje yönetim süreçlerinin belirli olgunluğa ulaşmış olmasıdır. Proje planlanması, izlenmesini ve raporlanması, KDY verilerini doğru bir şekilde elde edebilmek için oldukça önemlidir.

Uygulanan durum çalışmalarında da görüldüğü gibi KDY'nin yazılım projelerinde başarıyla uygulanamamasının en temel nedeni tamamlanan işlerin yeniden yapılması faaliyetleridir. Yukarıda iki proje uygulamasında da KDY yeni özelliklerin geliştirmesinde başarıyla uygulanabilirken, yeniden yapma faaliyetleri de düşünüldüğünde proje yöneticilerini proje gidişatı hakkında yanlış yönlendirebilir. Herhangi bir anda proje tamamen yolundaymış gibi görünmesine rağmen, daha sonra burada tamamlanmış özelliklere ait yeniden zaman ve emek ihtiyacı ortaya çıkabilir.

Yeniden yapma yazılım projelerinde oldukça ciddi bir problem olup, yazılımın ayrılmaz bir parçası haline gelmiştir. Yazılım projelerinin esas özellikleri ve kalite konusu düşünüldüğünde, yeniden yapma yazılım projelerinin doğal bir parçası olarak kabul edilir. Kimi zaman toplam proje bütçesinin % 50'sine yaklaşan yeniden yapma faaliyetleri[5][8] yazılım projelerinde dikkatle planlanıp önlem alınması ihtiyacını doğurmaktadır. Diğer proje disiplinlerinde örneğin bir inşaat projesinde somut bir ürün var olduğundan bu tamamen farklıdır. Özellikle belli bir zaman sonrasında yeniden yapma kabul edilemez. Dolayısıyla yeniden yapma konusu diğer proje disiplinleri için önemsenecek ciddi bir problem değildir. KDY de bu açıdan bir sorun içermez. Yeniden yapma faaliyetlerinin ana nedeni olan kalite konusu ise bize bu faaliyetler hakkında bilgi verebilir.

Yazılım projelerinin KDY ile ilgili diğer bir problemi ise yapılan işin değeri ve buna harcanan emek arasındaki ilişkinin belirsizliğidir. Yazılım projelerinde standart bir büyüklük ölçme yöntemi olmadığından, işlerin büyüklük ve çaba tahminleri son derece öznel ve insana bağlıdır. Dolayısıyla sıkça gözlenen yanlış tahminlemeler KDY yönteminin de yanlış sonuçlar vermesine neden olmaktadır. Bir diğer belirsizlik ise yapılan ve kalan işin ölçümü konusudur. Burada % 50-% 50 ya da % 0-% 100 gibi çeşitli yöntemler kullanılsa da yapılan işin tüm işin ne kadarı olduğu ölçmek ise ayrı bir zorluk ve belirsizlik içermektedir.

5 Sonuçlar

Bu çalışma, esas olarak etkili bir proje yönetim aracı olan KDY ve yazılım projelerinde kullanımı hakkında bilgi vermeyi amaçlamıştır. Literatür araştırması ile beraber iki yazılım projesine KDY uygulama sonuçları da değerlendirilerek yazılım projelerinde KDY uygulamasının fayda ve eksiklikleri tartışılmıştır.

KDY proje ilerlemesini kapsam, zaman ve bütçe açısından ölçmekte, projenin performasını değerlendirmekte ve geleceği hakkında tahmin yapabilme olanağı sunmaktadır. Fakat KDY en temelde yapılan işlerin niteliğinden çok niceliğine odaklanmaktadır. Diğer bir deyişle standart KDY yöntemi içerisinde yapılan işlerin kalitesine dair herhangi bir değerlendirme yapılmamaktadır. Yazılım projeleri ise diğer proje disiplinlerine oranla çok daha fazla, kimi zaman proje bütçesinin % 50'sine yaklaşan oranda yeniden yapma faaliyetleri içermektedir. Yazılım projelerinin ayrılmaz bir parçası

olan yeniden yapma faaliyetlerinin temel nedeni de yapılan işin kalitesidir. Dolayısıyla yazılım projelerinde proje ilerlemesini ölçerken kalite faktörü de göz önünde bulundurulduğunda, proje gidişatı hakkında daha sağlıklı sonuçlar elde edilebilecektir.

Sonraki çalışmalarda KDY yöntemini geliştirmek ve yazılım projeleri için daha uygun hale getirmek için kalite boyutu da göz önünde bulundurulmalıdır. Böylelikle yöntem yazılım proje yöneticilerine daha doğru sonuçlar verecek, yazılım projelerinde kullanımı yaygınlaşacaktır.

Kaynaklar

1. ANSI/EIA -748A (1998), American National Standard Institute / Electronic Industries Alliance/ Standard for Earned Value Management Systems
2. AS 4817-2003, Australian Standard 4817-2003: Project performance measurement using Earned Value.
3. Alleman, G. B. (2003). Project Management = Herding Cats, A Field Report, Agile Project Management, PMFORUM, PMFORUM.ORG, February
4. Alleman, G. B. & Henderson, M. (2003). Making Agile Development Work in a Government Contracting Environment – Using Earned Value to Measure Velocity, Agile Development , Salt Lake City, Utah, June 25 – 27
5. Boehm, B. & Papaccio, C (1988). Understanding and Controlling Software Costs, *IEEE Transactions of Software Engineering*, 14, 1462 - 1477
6. Brooks, F. P. (1995). *Mythical Man Month: Essays on Software Engineering*, Reading, MA: Addison-Wesley.
7. Cockburn, A. (2005). Crystal Clear: A Human-Powered Methodology for Small Teams, Pearson Education Inc. Upper Saddle River, NJ, USA
8. Dion, R. (1993). Process Improvement and the Corporate Balance Sheet, *IEEE Software*, 28-35.
9. Efe, P. & Demirors, O. (2013). Applying EVM in a Software Company: Benefits & Difficulties, unpublished.
10. Ferle, M. (2006). Implementing Earned Value Management on IT Projects, *19th International Cost Engineering Congress*, Ljubljana, Slovenia
11. Fleming, Q.W. & Koppelman, J.M. (2005). *Earned Value Project Management*, 3rd Ed., Newtown Square, PA, USA : Project Management Institute.
12. Fleming, Q.W. & Koppelman, J.M. (1998). Earned Value Project Management: A Powerful Tool for Software Projects, *CrossTalk, The Journal of Defense Software Engineering*, 4, 19-23.
13. Hanna, R.A. (2009). Earned Value Management Software Projects, Proceedings of the *3rd IEEE International Conference on Space Mission Challenges for Information Technology*
14. Lett, S. H. (1998). An Earned Value Tracking System for Self-Directed Software Teams, *Proceedings of European SEPG 98*
15. Lipke, W. (2003). Schedule is Different, The Measurable News, 10-15. Retrieved from http://earnedschedule.com/Docs/Schedule%20is%20Different.pdf.
16. Plekhanova, V. (1998). On Project Management Scheduling where Human Resource is a Critical Variable. *Proceedings of the 6th European Workshop on Software Process Technology* (EWSPT-6), Lecture Notes in ComputerScience series, Springer-Verlag, London, UK, pp. 116–121
17. Project Management Institute. (2005). *Practice standard for earned value management*. Newtown Square, PA, USA: Project Management Institute (PMI).

18. Project Management Institute. (2008). A Guide to the Project Management Body of Knowledge (PMBOK® Guide). 4th ed. Newtown Square, PA, USA: Project Management Institute (PMI).
19. Solanki, P. (2009). Earned Value Management: Integrated View of Cost and Schedule Performance. New Delhi, India: Global India Publications Pvt Ltd.
20. Solomon, P. J. (2001). Practical Software Measurement, Performance-Based Earned Value. *CrossTalk*, September, 25-29
21. Solomon, P. J. (2004). Integrating Systems Engineering with Earned Value Management, *Defense AT&L*, 33, 42-46
22. Solomon, Paul J. (2005) "Performance-Based Earned Value." *CrossTalk , The Journal of Defense Software Engineering*, August, 22-26
23. Solomon, Paul J. (2006) "Performance-Based Earned Value" *CrossTalk The Journal of Defense Software Engineering*, May, 20-24
24. Solomon, P. J. & Young, R. (2007). *Performance-Based Earned Value*, Hoboken, NJ:Wiley & Sons
25. Stratton, Ray W. (2006). The Earned Value Management Maturity Model, Vienna, VA, USA : Management Concepts Inc.
26. Sulaiman, T., Barton, B. & Blackburn, T. (2006). AgileEVM – Earned Value Management in Scrum Projects, *Proceedings of AGILE 2006 Conference (AGILE'06)*, IEEE Computer Society, Minneapolis, Minnesota, USA