Çevik dönüşümün etkilerini nasıl ölçeriz? Orta ölçekli bir yazılım kurumu için bilgi ihtiyaçları ve metrikler

Feyza Nur Kılıçaslan[1], Ayça Tarhan[1], Haluk Altunel[2]

[1]Yazılım Mühendisliği Araştırma Grubu (HUSE)
Bilgisayar Mühendisliği Bölümü, Hacettepe Üniversitesi, Ankara Türkiye
[2] Softtech, Ankara, Türkiye
l {feyza.kilicaslan, atarhan}@hacettepe.edu.tr
2 haluk_altunel@hotmail.com

Özet. Birçok yazılım kuruluşu çevik dönüşümün avantajlarından yararlanabilmek için çalışma süreçlerini plan güdümlüden çevik yöntemlere doğru değiştirmeye başlamıştır. Böylece çevik yöntemlerin benimsenmesinin, yazılım kuruluşlarını nasıl etkilediği sıklıkla tartışılmaktadır. Çevik dönüşümün etkilerini ölçmek, çevik yöntemlerin yazılım kuruluşlarına ne ölçüde katkı sağladığını değerlendirmek ve anlamak açısından önemlidir. Bu çalışma, çevik dönüşümü gerçekleştiren orta ölçekli bir yazılım kurumunda, dönüşümün etkilerini ölçmek amacıyla belirlenen bilgi ihtiyaçlarını ve metrikleri tanımlamaktadır. Ölçmeye temel olacak tanımları belirlerken öncelikle literatürde mevcut çalışmalar incelenmiş ve bu çalışmalardan çıkartılan metrikler; ölçtükleri iş, süreç, ürün ve kaynak varlıklarına göre gruplanmıştır. Ardından yazılım kurumunda bu varlıklar açısından çevik dönüşümün etkilerini ölçmeyi sağlayacak bilgi ihtiyaçları, ISO 15939 Yazılım Ölçme Standardı rehber alınarak belirlenmiş ve literatürden derlenen metriklerle ilişkilendirilmiştir. Sonuç olarak iş, süreç, ürün ve kaynak açılarından çevik dönüşümün etkilerini ölçmeyi sağlayacak bilgi indikatörleri (grafik, tablo, vb.) ve ilişkili ölçme yapıları (türetilmiş ve temel metrikler, ölçme fonksiyonları, vb.) tanımlanmıştır. Gelecek çalışmamızda bu bilgi ihtiyaçları üzerinden, yazılım kurumundaki çevik dönüşümün etkilerini ölçmeyi planlıyoruz. Oluşturduğumuz ölçüm tasarımının sadece çalıştığımız yazılım kurumu için değil, çevik dönüşümü gerçekleştiren diğer birçok firma için de yol gösterici olacağını umuyoruz.

Anahtar Kelimeler: Çevik dönüşüm, Yazılım metrikleri, Ölçüm tasarımı

How do we measure the effects of agile transformation? Information needs and metrics for a medium-sized software company

Abstract. Many organizations have started to change their working process from plan-driven to agile in order to leverage the benefits of agile transformation, so it

is frequently discussed how adopting agile methodologies affect software organizations. Measuring effects of agile transformation is important in terms of evaluating and understanding to what extent agile methods contribute to software organizations. This study describes a set of information needs and metrics that are designed to measure the effects of agile transformation in a medium-sized software organization that has been going through agile transformation. While defining the base of measurement, firstly related studies in literature are examined, and metrics derived from these studies are grouped by the measured entities of business, process, product and resource. The information needs for measuring effects of agile transformation are determined by the guidance of ISO 15939 standard for Software Measurement Process, and then are correlated with the metrics compiled from the literature. As a result, information indicators (graphs, tables etc.) and associated measurement constructs (derived and base metrics, measurement functions etc.) that enable measuring impacts of agile transformation are described from business, process, product and resource perspectives. In our future work, we plan to measure the effects of agile transformation in the software organization using these information needs. We hope that our measurement design will be guide not only for the software company we work with, but also for many other organizations who adopt agile transformation.

Keywords: Agile transformation, Software metrics, Measurement design

1 Giriş

Çevik yazılım geliştirme yöntemleri, ürün teslimatını hızlandırması, değişen öncelikleri yönetme becerisini geliştirmesi, üretkenliği arttırması ve yazılım kalitesini iyileştirmesi [1] gibi sebeplerle farklı büyüklükteki firmalar arasında oldukça yaygınlaşmakta ve her yıl sektörde artan oranda benimsenmektedir. Bununla birlikte çevik dönüşümün yazılım firmalarını nasıl etkilediği birçok kez tartışma konusunu olmaktadır. Bu bağlamda çevik dönüşümün etkilerinin ölçülmesi, çevik yöntemlerin yazılım firmalarına sağladığı katkıların daha iyi anlaşılabilmesi ve değerlendirilebilmesi bakımından önemli bir yere sahiptir.

Bir deneyim raporunda [2], Yahoo! Music'deki çevik dönüşümün üretkenliği ve takım moralini önemli ölçüde artırdığı öne sürülmüştür. Başka bir şirket olan Primavera Systems tarafından, çevik yöntemlerin bir türü olan Scrum'ı uyguladıktan sonra müşteri tarafından bildirilen kusur sayısıyla ölçülen kalitede %30 artış olduğu bildirilmiştir [3]. Prochazka ve arkadaşlarının yaptığı bir çalışmada [4] ise "yeni bir çalışma yolu" olarak belirtilen çevik dönüşümün sonuç olarak müşteri ve takım memnuniyetini arttırmaya olanak sağladığı yönünde kanıtlar ortaya koyulmuştur.

Çevik yazılım geliştirmenin birçok açıdan iyileştirmeler sağladığı düşünülse de [5, 6] bazı çalışmalar çevik dönüşüm çabalarından yeterince kazanım sağlanamadığını ya da çabaların başarısızlıkla sonuçlandığını bildirmiştir. Zamana yayılan bir vaka çalışmasında [7], çevik yazılım geliştirme yöntemi benimsendikten sonra takip edilen projedeki hata yoğunluğunda önemli bir azalma ya da hata profillerinde değişiklik gözlemlenmediği raporlanmıştır. Ericsson'daki bir pilot çevik dönüşüm projesi kapsamında

yapılan başka bir çalışmada [8] çevik çalışma yöntemlerinin projelerin başarı oranını arttırmadığı bildirilmiştir.

Yukarıda bahsedilen hususlar çerçevesinde birçok yazılım kurumunun çevik yöntemleri uygulamaya başlamasıyla çevik dönüşümün etkilerini ölçmek mutlak bir ihtiyaç haline gelmiştir. Çevik dönüşümün firmaya sağladığı katkıların objektif bir şekilde ölçülebilmesi de bilgi ihtiyaçlarına uygun metriklerin tanımlanmasıyla yapılabilir. Bu çalışmada, "Çevik dönüşümün etkilerini nasıl ölçeriz?" sorusuyla çevik dönüşüm öncesi ve sonrasını nicel olarak karşılaştırabilmeye olanak sağlayacak metrik tabanlı bir ölçüm tasarımı sunulmuştur. Bu ölçüm tasarımı, çevik dönüşüm gerçekleştirmiş orta ölçekli bir yazılım kurumuyla birlikte ve çevik dönüşümün iş, süreç, ürün ve kaynak üzerindeki etkilerini ölçmek üzere, tekrarlamalı olarak geliştirilmiştir.

Makalenin ilerleyen kısımlarında; Bölüm 2'de literatürdeki ilgili çalışmalara yer verilmiştir. Bölüm 3'de bu çalışmada kullanılan araştırma yöntemi tanımlanmıştır. Bölüm 4'de ölçüm tasarımı sunulmakta, Bölüm 5'de sonuçlar ve gelecek çalışmalara yer verilmektedir.

2 İlgili çalışmalar

Literatürde çevik yazılım ölçümüne [9-12] yönelik çalışmalar bulunsa da; geleneksel (çağlayan, spiral vb.) geliştirme metotları ile çevik geliştirmenin karşılaştırılmasına olanak sağlayarak çevik dönüşümün etkilerini ölçen metrik modeli ya da ölçüm tasarımı sunan az sayıda çalışma bulunmuştur. Bu alanda yapılmış çalışmalardan [13] ve [14] bir yazılım geliştirme organizasyonundaki çevik dönüşümün nicel olarak ölçülmesine yönelik bir metrik modeli sağlamıştır.

Heidenberg ve arkadaşlarının yaptığı bir çalışmada [13], çevik ve yalın dönüşümün yazılım geliştirme organizasyonlarına etkilerini ölçmek için geliştirilen bir metrik modeli sunulmuştur. Olszewska ve arkadaşları yapmış oldukları çalışmada [14], çevik ve yalın dönüşümün etkilerini performans ve kalite bakımından ölçmeye yönelik metrikler sunmuşlardır. Korhonen'in vaka çalışmasında [15], çevik dönüşümün ilk on iki ayı boyunca büyük ölçekli bir firmadaki hata verileri analiz edilmiş; çevik dönüşüm öncesi ve sonrası hata raporlama uygulamasında değişimi sunulmuştur.

Petersen ve Wohlin tarafından 2010 yılında yapılan vaka çalışmasında [16], plan güdümlü yazılım geliştirmeden çevik yönteme geçişin etkileri karşılaştırılmıştır. Bu vaka çalışmasında, firma çalışanlarıyla yapılan görüşmelerden toplanan nitel verilere odaklanılmıştır. Çevik yazılım geliştirmede Scrum master, test yöneticileri ve test görevlilerine odaklanan çalışmada [17], test metrikleri kullanılarak çağlayan modelden çevik modele geçişin etkileri karşılaştırılmıştır. Carnegie Mellon Üniversitesinde yapılan kontrollü deneysel çalışmada [18] ise sonuçlar ve takımlar tarafından üretilen metrikler, şelale modelinden uçdeğer programlama (İng. Extreme Programming-XP) modeline geçişin avantaj ve dezavantajlarına bakmak için kullanılmıştır.

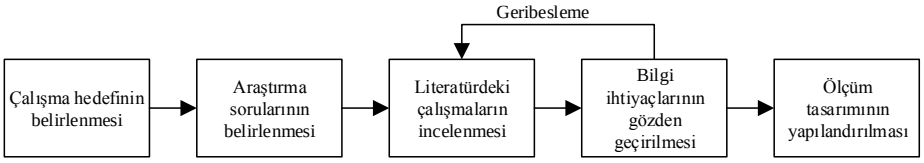
3 Araştırma yöntemi

Bu çalışmada, Hedef-Soru-Metrik (İng. Goal-Question-Metric) yaklaşımı [19] benimsenerek halen devam etmekte olan çevik dönüşüm ile ilgili sistematik literatür haritalama çalışmamızın makale havuzundan, ölçüm ile ilişkilendirilen makaleler [20] incelenmiştir. Çalışmanın hedefi, çevik dönüşümün etkilerini nicel olarak ölçmeyi sağlayacak bir ölçüm tasarımı ortaya çıkarmaktır. Bu hedefi adresleyen dört araştırma sorusu (AS) tanımlanmıştır.

AS1: Çevik dönüşüm yazılım kurumunda işi (İng. Business) nasıl etkilemiştir?
AS2: Çevik dönüşüm yazılım kurumunda süreci nasıl etkilemiştir?
AS3: Çevik dönüşüm yazılım kurumunda ürünü nasıl etkilemiştir?
AS4: Çevik dönüşüm yazılım kurumunda kaynağı nasıl etkilemiştir?

İncelenen makalelerden, araştırma soruları kapsamında çevik dönüşümün etkilerini ölçmeye yönelik metrikler çıkarılmış ve bu metrikler ölçtükleri iş, süreç, ürün ve kaynak varlıklarına göre gruplandırılmıştır. Daha sonra çalıştığımız yazılım kurumunun bilgi ihtiyaçları göz önünde bulundurularak ve ISO 15939 Yazılım Ölçme Standardı [21] rehberliğinde, yazılım kurumunun işbirliği ile tekrarlamalı çalışarak bir ölçüm tasarımı ortaya çıkarılmıştır. Ölçüm tasarımının oluşturulmasında izlenen adımlar şeması Şekil 1'de sunulmuştur. Çevik dönüşüm öncesinde ve sonrasında ölçüm tasarımının durum karşılaştırmasında yararlı olmasını sağlamak amacıyla, tasarım metrikleri aşağıdaki kriterler göz önünde bulundurularak ortaya konulmuştur.

- Metrikler, hem plan güdümlü (şelale vb.) hem de çevik geliştirme yöntemine uygulanabilir olmalı.
- Metrikler, geçmişteki (çevik dönüşüm öncesindeki) ve devam etmekte olan projelerden toplanabilir olmalı.
- Metrikler, herhangi bir kapsam, boyut ve karmaşıklıktaki projelerden toplanabilir ve kullanılabilir olmalı.
- Metrikler objektif olarak değerlendirmeye olanak sağlamalı.



Şekil 1. Ölçüm tasarımının oluşturulmasında izlenen adımlar

4 Ölçüm tasarımı

Ölçüm tasarımı yapılandırılırken; iş, süreç, ürün ve kaynak açılarından çevik dönüşümün etkilerini ölçmeyi sağlayacak bilgi indikatörleri (grafik, tablo, vb.) ve ilişkili ölçme yapıları (türetilmiş ve temel metrikler, ölçme fonksiyonları, vb.) ISO 15939 Yazılım Ölçme Standardı rehberliğinde tanımlanmıştır. Bu ölçüm yapısı, ilgili yazılım özelliklerinin (İng. Software Attribute) nasıl nicelleştirildiğini ve karar verme sürecine temel oluşturan indikatörlere nasıl dönüştürüldüğünü açıklamaktadır [22]. Ölçüm yapısının,

geleneksel yöntemlerle geliştirilmeye başlanılmış, sonrasında çevik dönüşüm gerçekleştirilmiş ürünler üzerinde uygulanması hedeflenmektedir. Aşağıda ölçüm yapısıyla ilişkili kavramların açıklamalarına yer verilmiştir.

- Bilgi ihtiyacı: Amaç ve hedeflere ulaşılabilmesi, risk ve sorunlarla baş edilebilmesi için gerekli olan bilgiler
- Analiz birimi: Üzerinde analiz yapılacak temel öğe
- Ölçme birimi: Aynı türdeki iki büyüklüğün oranlarını bir sayı olarak ifade ederek bunların karşılaştırılmasını sağlayan, genel kabul ile tanımlanmış büyüklük
- Türetilmiş metrik: İki veya daha fazla temel metrik değerinin bir fonksiyonu
- Temel metrik: Bir özellik (İng. Attribute) ve onu nicelleştirme yöntemi açısından tanımlanan metrik
- Ölçme fonksiyonu: İki veya daha fazla temel metriği birleştirmek için uygulanan algoritma veya hesaplama
- İndikatör yorumu: Beklenen durumların ya da çıktıların tanımlanması

4.1 Çevik dönüşüm yazılım kurumunda işi nasıl etkilemiştir?

Çevik dönüşümü gerçekleştiren kurumların hedeflerinden biri de müşteriye dönüş hızını artırmaktır. Müşteriden gelen talebe karşılık verme esnekliğinin ölçülmesi, çevik dönüşümün işe yansıyan etkilerinin açıkça görülmesini mümkün kılar.

İndikatör 1- Çevrim süresi ve bekleme süresi.

Bu çalışmada çevrim süresi (İng. Cycle Time); müşteriden gelen talebin işleme konulduğu andan, tamamlamasına kadar geçen süre olarak ele alınmaktadır. Bekleme süresi ise talebin işleme konulmadan önce işin yapılmaya hazır hale gelmesi için beklenen zaman dilimidir.

Bilgi İhtiyacı	Çevrim süresini ve bekleme süresini değerlendirmek
Analiz Birimi	Ürün
Ölçme Birimi	Zaman
Türetilmiş Metrik(ler)	1.Ortalama çevrim süresi, 2.Ortalama bekleme süresi
Ölçme Fonksiyonu	1. Toplam çevrim süresi / çevrim sayısı 2. Toplam bekleme süresi / bekleme sayısı
Temel Metrik(ler)	Çevrim süresi (ürüne yönelik talep başına), bekleme süresi (ürüne yönelik talep başına)
İndikatör Yorumu	Çevik yöntemlerle geliştirilen üründe, çevrim ve bekleme süresinin kısa olması; ayrıca bekleme süresinin çevrim süresine oranının düşük olması beklenir.

İndikatör 2- Akış zamanı.

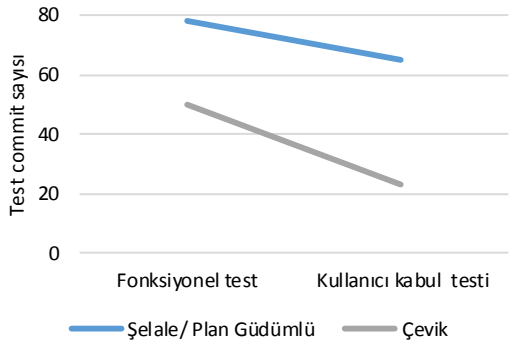
Müşteriden talebin geldiği andan, talebin işlenip ürünün müşteriye teslim edildiği ana kadar geçen zaman dilimi akış zamanı (İng. Lead Time) olarak değerlendirilmiştir. Akış zamanı, talebin önceliklendirilmesi, doğru ürüne yönlendirilmesi, bekleme süresi, çevrim süresi gibi süreçlerin toplam zamanı olarak hesaplanır.

Bilgi İhtiyacı	Akış zamanını değerlendirmek
Analiz Birimi	Ürün
Ölçme Birimi	Zaman
Türetilmiş Metrik(ler)	Ortalama akış zamanı
Ölçme Fonksiyonu	Toplam akış zamanı / akış sayısı
Temel Metrik(ler)	Akış zamanı (ürüne yönelik talep başına)
İndikatör Yorumu	Çevik yöntemlerle geliştirilen üründe, akış zamanının kısa süreli olması beklenir.

4.2 Çevik dönüşüm yazılım kurumunda süreci nasıl etkilemiştir?

İndikatör 3- Test commit.

Fonksiyonel test, bir doğrulama (İng. Verification) aktivitesi olarak yazılımın tanımlı gereksinimleri sağlayıp sağlanmadığının teyit edilmesi için yapılır. Yazılım test sürecinin son aşaması olan kullanıcı kabul testi, bir geçerleme (İng. Validation) aktivitesi olarak yazılımın kullanılması beklenen tipik senaryolarını kapsar. İndikatör 3 yapısına uygun temsili karşılaştırma grafiği Şekil 2'deki gibidir.

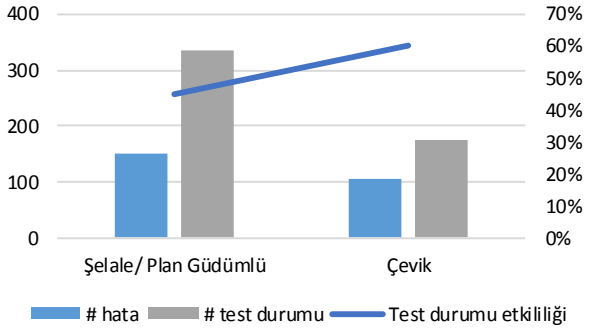


Şekil 2. Fonksiyonel ve kullanıcı kabul test commit sayıları

Bilgi İhtiyacı	Fonksiyonel ve kullanıcı kabul testi "commit"lerini değerlendirmek
Analiz Birimi	Ürün
Ölçme Birimi	Commit
Türetilmiş Metrik(ler)	1. Toplam fonksiyonel test "commit" sayısı 2. Toplam kullanıcı kabul test "commit" sayısı
Ölçme Fonksiyonu	1.Fonksiyonel test commit sayılarının toplanması 2.Kullanıcı kabul test commit sayılarının toplanması
Temel Metrik(ler)	Fonksiyonel ve kullanıcı kabul testlerindeki "commit" sayıları
İndikatör Yorumu	Çevik yöntemlerle geliştirilmiş üründe; fonksiyonel test "commit" sayısının, kullanıcı kabul test "commit" sayısına göre daha az olması ve bu azalış eğiminin de daha fazla olması beklenir.

İndikatör 4- Test durumu etkililiği.

Test durumu (İng. Test Case), test edilen bir sistemin gereksinimlerini yerine getirip getirmediğini ya da doğru bir şekilde çalışıp çalışmadığının gösterilmesi için kullanılan girdiler, gerçekleşmesi beklenen adımlar ve beklenen sonuçlardan oluşan bir dizi koşul ya da değişkendir. İndikatör 4 yapısına uygun temsili karşılaştırma grafiği Şekil 3'deki gibidir.



Şekil 3. Test durumu etkililiği

Bilgi İhtiyacı	Test durumu etkililiğini değerlendirmek
Analiz Birimi	Ürün
Ölçme Birimi	Hata, test durumu
Türetilmiş Metrik(ler)	Test durum etkililiği
Ölçme Fonksiyonu	Kaydedilen hata sayısı / Toplam test durum sayısı
Temel Metrik(ler)	Kaydedilen hata sayısı, test durum sayısı
İndikatör Yorumu	Çevik yöntemlerin kullanımında az sayıda test durumu ile çok sayıda hata yakalanması hedeflenir.

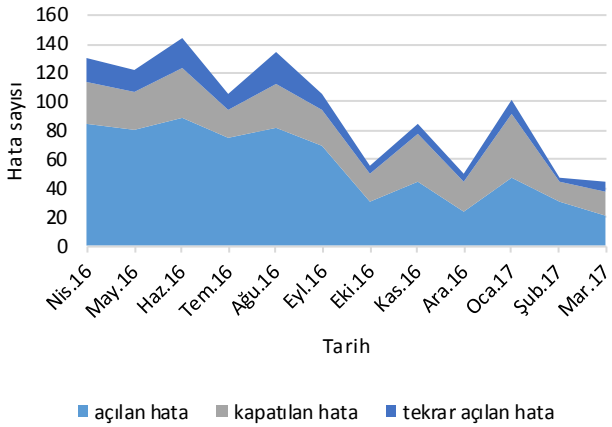
İndikatör 5-Hata çözme başarısı.

Müşterinin bildirdiği hataların ne kadarının çözüme kavuşturulduğunun ölçülmesi, hata çözme başarısı olarak değerlendirilmiştir.

Bilgi İhtiyacı	Hata çözme başarısını değerlendirmek
Analiz Birimi	Ürün
Ölçme Birimi	Hata
Türetilmiş Metrik(ler)	Hata çözme başarı oranı
Ölçme Fonksiyonu	Çözülen hata sayısı / Müşterinin bildirdiği hata sayısı
Temel Metrik(ler)	Müşteriden dönen hata sayısı, çözülen hata sayısı
İndikatör Yorumu	Çevik yöntemlerin hata çözme başarı oranının daha yüksek olması beklenir.

İndikatör 6- Açılan, kapatılan ve yeniden açılan hataların durumu.

Geliştirme süreci boyunca ilk defa karşılaşılan sorunlar açılan hata, sorunların çözümlenmesi kapatılan hata [23], sorunların tam çözümlenmemesi ya da yeniden ortaya çıkması yeniden açılan hata olarak ele alınmaktadır. İndikatör 6 yapısına uygun temsili karşılaştırma grafiği Şekil 4'deki gibidir.



Şekil 4. Şelale (Nis.16-Eyl.16) ve çevik (Eki.16-Mar.17) yöntemlerde hata durumu izleme

Bilgi İhtiyacı	Hataların durumunu değerlendirmek
Analiz Birimi	Ürün
Ölçme Birimi	Hata
Türetilmiş Metrik(ler)	1. Ortalama açılan hata sayısı 2. Ortalama kapatılan hata sayısı 3. Ortalama yeniden açılan hata sayısı
Ölçme Fonksiyonu	1. Toplam açılan hata sayısı / Ürün sayısı 2. Toplam kapatılan hata sayısı / Ürün sayısı 3. Toplam yeniden açılan hata sayısı / Ürün sayısı
Temel Metrik(ler)	Açılan hata sayısı, Kapatılan hata sayısı, Yeniden açılan hata sayısı
İndikatör Yorumu	Çevik yöntemlerle geliştirmede kapatılan hata sayısının açılan hata sayısına daha yakın bir değerde ve tekrar açılan hata sayısının da kapatılan hata sayısına daha yakın bir değerde olması beklenir.

4.3 Çevik dönüşüm yazılım kurumunda ürünü nasıl etkilemiştir?

İndikatör 7- Hata yoğunluğu.

Hata yoğunluğu, ürünlerin kalite değerlendirmesinde kullanılan önemli göstergelerden biridir. Hata yoğunluğunun yüksek olması, düşük kaliteli ürün olarak değerlendirilmektedir. Yazılım büyüklük ölçümü için çalıştığımız firmanın işlev puanı (İng. Function Point) verilerine erişilememesi sebebiyle ölçme birimi olarak kod satır sayısı kullanılmıştır.

Bilgi İhtiyacı	Hata yoğunluklarını değerlendirmek
Analiz Birimi	Sürüm
Ölçme Birimi	Hata, kod satırı
Türetilmiş Metrik(ler)	Hata yoğunluğu
Ölçme Fonksiyonu	Sürümdeki toplam hata sayısı / Sürümün kod satır sayısı
Temel Metrik(ler)	Hata sayısı, sürüm sayısı, sürümün kod satır sayısı
İndikatör Yorumu	Çevik yöntemlerle geliştirmede hata yoğunluğunun düşmesi beklenir.

İndikatör 8- Hatanın önem derecesi.

Hatalar, yazılım kalitesi üzerinde oluşturduğu olumsuz etkinin şiddetini vurgulamak için önem derecesine göre sınıflandırılır. Bu çalışmada hatalar; düşük, orta, yüksek ve kritik olmak üzere dört sınıfa ayrılmıştır. Düşük önem derecesi; ürünün işlevselliğini ya da veriyi etkilemeyen, yazım denetimi/ dilbilgisi hataları ya da tasarımdaki yüzeysel hataları temsil eder. Orta önem derecesindeki hatalar; ürünün birincil özelliklerini veya işlevselliğini engellemezken, düşük dereceli hatalara göre daha büyük bir olumsuz etki oluştururlar. Ürünün tanımlı gereksinimlerini karşılamasını veya bir özelliğini yerine getirmesini engelleyen sorunlar, yüksek önem dereceli hata olarak sınıflandırılır. Kritik önem derecesindeki hatalar, çözümlenmedikçe ürünün işlevselliğini yerine getiremediği, veri kaybına veya bozulmasına sebep olan hatalar olarak değerlendirilir.

Bilgi İhtiyacı	Hataların önem derecesini değerlendirmek
Analiz Birimi	Ürün
Ölçme Birimi	Hata
Türetilmiş Metrik(ler)	Toplam Hata Sayısı, her hata tipinin oranı
Ölçme Fonksiyonu	Hata tipine ait toplam hata sayısı / Tüm tipteki toplam hata sayısı
Temel Metrik(ler)	Hata tipine göre hata sayısı
İndikatör Yorumu	Çevik yöntemlerle geliştirilen üründe her bir üst hata seviyesinin, bir alt hata seviyesine göre daha düşük oranda hataya sahip olması beklenir. Örneğin kritik hata derecesindeki hata oranının, diğer hata seviyelerine göre en düşük oranda olması beklenir.

İndikatör 9- Müşteriden dönen hata (Gizli hata).

Fonksiyonel test aşamasında tespit edilemeyip kullanıcı kabul testinde ortaya çıkan hatalar, ürün içindeki gizli hata veya müşteriden dönen hata olarak ele alınmaktadır.

Bilgi İhtiyacı	Müşteriden dönen hata durumunu değerlendirmek
Analiz Birimi	Ürün
Ölçme Birimi	Hata
Türetilmiş Metrik(ler)	Gizli hata sayısı yoğunluğu
Ölçme Fonksiyonu	Toplam gizli hata sayısı / Toplam hata sayısı
Temel Metrik(ler)	Gizli hata sayısı, Tespit edilen hataların toplam sayısı

İndikatör Yorumu	Çevik yöntemlerle geliştirilmiş üründe, geliştirme sürecinde tespit edilen hata sayısı ile gizli hata sayısının toplamından ortaya çıkan toplam hata sayısı içinde gizli hata sayısının az olması beklenir.

İndikatör 10- Kurallara uyum indeksi

Kurallara Uyum İndeksi (İng. Rules Compliance Index) teknik kalite değerlendirmesinde kullanılır. Sonar Qube [24] aracı üzerinden verimlilik (İng. Efficiency) , sürdürülebilirlik (İng. Maintainability), taşınabilirlik (İng. Portability), güvenilirlik (İng. Reliability), kullanılabilirlik (İng. Usability) kategorileri için belirlenen kural setine göre ihlal yoğunluğu hesaplanır.

Bilgi İhtiyacı	Kurallara uyum indeksini değerlendirmek
Analiz Birimi	Ürün
Ölçme Birimi	Ağırlıklandırılmış ihlal, kod satırı
Türetilmiş Metrik(ler) Ölçme Fonksiyonu	İhlal yoğunluğu 100-(Ağırlıklandırılmış ihlaller / (Kod satır sayısı *100))
Temel Metrik(ler)	Verimlilik, sürdürülebilirlik, taşınabilirlik, güvenilirlik, kullanılabilirlik
İndikatör Yorumu	Çevik yöntemlerle geliştirilmiş üründe; kurallara uyum indeksinin yüksek olması, ihlal yoğunluğunun düşük olması beklenir.

İndikatör 11- Çevrimsel karmaşıklık.

McCabe'in çevrimsel karmaşıklığı (İng. Cyclomatic Complexity), bir yazılım programının karmaşıklığını nicelleştiren bir yazılım kalitesi metriğidir [25]. Karmaşıklık, program aracılığıyla kontrol akış diyagramını kullanarak doğrusal bağımsız yolların sayısının ölçülmesiyle çıkarılır.

Bilgi İhtiyacı	Ürünün karmaşıklığını değerlendirmek
Analiz Birimi	Ürün
Ölçme Birimi	Kontrol akış diyagramı
Türetilmiş Metrik(ler) Ölçme Fonksiyonu	Çevrimsel karmaşıklık Kenar sayısı-Düğüm sayısı+2*Programdaki bileşen sayısı
Temel Metrik(ler)	Kenar sayısı, düğüm sayısı, programdaki bileşen sayısı
İndikatör Yorumu	Çevrimsel karmaşıklık değeri ne kadar yüksekse, program o kadar karmaşık ve hata eğilimli olarak değerlendirilir. Çevrimsel karmaşıklık değerinin yüksek olması, programların bakım yapılabilirliğini ve test edilebilirliğini zorlaştırması sebebiyle istenmeyen bir durumdur. Çevik yöntemlerle geliştirilmiş üründe, çevrimsel karmaşıklığın düşük olması beklenir [26].

4.4 Çevik dönüşüm yazılım kurumunda kaynağı nasıl etkilemiştir?

İndikatör 12- Üretkenlik.

Üretkenlik, takımın performans değerlendirmesinde kullanılan göstergelerden biridir. [27]

Bilgi İhtiyacı	Takımın üretkenliğini değerlendirmek
Analiz Birimi	Takım
Ölçme Birimi	Kod satırı, zaman
Türetilmiş Metrik(ler) Ölçme Fonksiyonu	Ortalama Üretkenlik Kod satır sayısı / Adam ay
Temel Metrik(ler)	Satır sayısı, adam ay
İndikatör Yorumu	Çevik geliştirme yöntemleriyle çalışan takımların üretkenliğinin yüksek olması beklenir. Takımın üretkenliğin yüksek olması, en az kaynak ve çaba sonuncunda en fazla çıktının elde edilmesi olarak değerlendirilir.

5 Sonuçlar ve gelecek çalışmalar

Çevik yöntemlerin, yazılım kuruluşlarına ne ölçüde katkı sağladığını değerlendirmek ve anlamak açısından çevik dönüşümün etkilerini ölçmek bir ihtiyaç haline gelmiştir. Bu çalışmada, çevik dönüşümü gerçekleştiren orta ölçekli bir yazılım kurumunda, dönüşümün etkilerini ölçmek amacıyla belirlenen bilgi ihtiyaçları ve metrikler üzerinden bir ölçüm tasarımı tanımlanmıştır. Ölçüm tasarımı, literatürde yapılan araştırmalar ve çalıştığımız yazılım kurumunun iş birliği ile ISO 15939 Yazılım Ölçme Standardı rehber alınarak oluşturulmuştur. Ölçüm tasarımında kullanılan indikatör 7 ve 12 için yazılım büyüklüğünü ölçme birimlerinin çalıştığımız firma baz alınarak oluşturulması, kısıt olarak değerlendirilebilir. Gelecek çalışmamızda sunduğumuz bu ölçüm tasarımı ile yazılım kurumundaki çevik dönüşümün etkilerini ölçmeyi planlıyoruz. Oluşturduğumuz ölçüm tasarımının sadece çalıştığımız yazılım kurumu için değil, çevik dönüşümü gerçekleştiren diğer birçok firma için de yol gösterici olacağını umuyoruz.

Kaynaklar

1. One, V., 11th Annual State of Agile Report. 2017.
2. Cloke, G. Get your agile freak on! agile adoption at yahoo! music. Agile Conference (AGILE), IEEE, 2007.
3. Schatz, B. and Abdelshafi, I., Primavera gets agile: a successful transition to agile development. IEEE, 22(3): p. 36-42, 2005.
4. Prochazka, J., et al. Keeping the Spin--From Idea to Cash in 6 Weeks: Success Story of Agile/Lean Transformation. Global Software Engineering (ICGSE), 6th IEEE International Conference, 2011.
5. Tarhan, A. and Yilmaz, S.G., Systematic analyses and comparison of development performance and product quality of Incremental Process and Agile Process. Information and Software Technology, 56(5): p. 477-494, 2014.

6. Ozkan, N., Tarhan, A., and Kucuk, C., Scrum at Scale in a COBIT Compliant Environment: The Case of Turkiye Finans IT. 2017.
7. Li, J., Moe, N.B., and Dybå, T. Transition from a plan-driven process to scrum: a longitudinal case study on software quality. Proceedings of the ACM-IEEE international symposium on empirical software engineering and measurement, 2010.
8. Lagerberg, L. and Skude, T., The impact of agile principles and practices on large-scale software development projects: A multiple-case study of two software development projects at Ericsson. 2013.
9. Kunz, M., Dumke, R.R., and Schmietendorf, A., How to measure agile software development. Software Process and Product Measurement, Springer, p. 95-101, 2008.
10. Concas, G., et al. An agile development process and its assessment using quantitative object-oriented metrics. International Conference on Agile Processes and Extreme Programming in Software Engineering, Springer, 2008.
11. Dubinsky, Y., et al. Agile metrics at the israeli air force. Agile Conference Proceedings, IEEE, 2005.
12. Hayes, W., et al., Agile metrics: Progress monitoring of agile contractors. DTIC Document, 2014.
13. Heidenberg, J., et al. A metrics model to measure the impact of an agile transformation in large software development organizations. International Conference on Agile Software Development, Springer, 2013.
14. Olszewska, M., et al., Quantitatively measuring a large-scale agile transformation. Journal of Systems and Software, 117: p. 258-273, 2016.
15. Korhonen, K. Evaluating the effect of agile methods on software defect data and defect reporting practices-a case study. Quality of Information and Communications Technology (QUATIC), Seventh International Conference, IEEE, 2010.
16. Petersen, K. and Wohlin, C., The effect of moving from a plan-driven to an incremental software development approach with agile practices. Empirical Software Eng., 15(6): p. 654-693, 2010.
17. Gupta, R.K., Manikreddy, P., and Abhinandan, G. Challenges in Adapting Agile Testing in a Legacy Product. Global Software Eng. (ICGSE), IEEE 11th International Conference, 2016.
18. Ji, F. and Sedano, T. Comparing extreme programming and Waterfall project results. Software Engineering Education and Training (CSEE&T), 24th IEEE-CS Conference, 2011.
19. Van Solingen, R., et al., Goal question metric (gqm) approach. Encyclopedia of software engineering, 2002.
20. Kılıçaslan, F.N. and Tarhan, A. Online Paper Repository for 'Measuring the effects of agile transformation'. cited 2017; Available from: https://goo.gl/ZShQ16.
21. Standardization, I.O.f. ISO/IEC 15939:2002 Software engineering -- Software measurement process. Available from: https://www.iso.org/standard/29572.html.
22. McGarry, J., Practical software measurement: objective information for decision makers. Addison-Wesley Professional, 2002.
23. Korhonen, K., Evaluating the impact of an agile transformation: a longitudinal case study in a distributed context. Software Quality Journal, 21(4): p. 599-624, 2013
24. SonarQube. The leading product for Continuous Code Quality. cited 2017; Available from: https://www.sonarqube.org/.
25. McCabe, T.J., A complexity measure. IEEE Transactions on Software Eng. 1976(4): p. 308-320.
26. Sato, D., Goldman, A., and Kon, F. Tracking the evolution of object-oriented quality metrics on agile projects. International Conference on Extreme Programming and Agile Processes in Software Engineering. Springer, 2007.
27. Sureshchandra, K. and Shrinivasavadhani, J. Adopting agile in distributed development. Global Software Engineering, ICGSE, IEEE International Conference, 2008.