Pamukkale Univ Muh Bilim Derg, 25(2), 240-246, 2019

Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi

Pamukkale University Journal of Engineering Sciences

# Üç yazılım firmasında yazılım süreç değişimlerinin gözlenen etkileri: Endüstriyel keşif vaka çalışması

# Observed effects of software processes change in three software firms: Industrial exploratory case study

*Murat YILMAZ*[1*]

[1]Bilgisayar Mühendisliği Bölümü, Mühendislik Fakültesi, Çankaya Üniversitesi, Ankara, Türkiye.
myilmaz@cankaya.edu.tr

Geliş Tarihi/Received: 11.01.2018, Kabul Tarihi/Accepted: 23.05.2018
doi: 10.5505/pajes.2018.03708
* Yazışılan yazar/Corresponding author
Araştırma Makalesi/Research Article

**Öz**

*Yazılım geliştirme süreçleri, gelişen yeni teknolojiler ve onun sağladığı imkânlar doğrultusunda sürekli iyileştirme gerektirir. Yazılım müşterilerinin pazarlanabilir fonksiyonlar içeren ürün talepleri üzerine kurgulanmış yeni nesil yazılım geliştirme modelleri ara ürün üretim hızını ve dolayısıyla ara sürüm sayısını arttırmayı hedeflemektedir. Bu ihtiyaçlar ışığında, yazılım şirketlerinin geliştirme süreçlerini müşteriden gelen istekleri karşılamak adına gözden geçirmeleri gerekmektedir. Ancak, daha da önemlisi, şirketler yazılım üretim hattındaki verimi düşürmemek için süreçlerini yenilikçi pratikler doğrultusunda değiştirmek zorunda kalmaktadırlar. Bu makalede, yazılım geliştiren üç şirketin yazılım geliştirme yöntemleri durum çalışması yöntemi ile detaylı olarak incelenerek, süreç değişimi aktiviteleri sistematik bir şekilde detaylandırılmıştır. Elde edilen bilgiler ışığında, üç firmanın da yazılım geliştirme yöntemlerindeki değişimler sorgulanarak edindikleri tecrübeler ve bu edinimlerin süreçler üzerindeki etkileri tartışılmıştır. Çalışmanın sonucunda, yazılım ürün geliştirme başarısının sürecin iyi işletilmesini önemli bir oranda etkilediği, yazılım geliştiren takımların da edindikleri kazanımlar ışığında kendi süreçlerini tasarlamaya çalıştıkları gözlenmiştir.*

**Anahtar kelimeler:** Çevik dönüşüm, yazılım geliştirme süreçleri, Deneysel yazılım mühendisliği, Vaka çalışması

**Abstract**

*Software development processes require continuous improvement in line with emerging new technologies and the possibilities it provides. A new generation of software development models based on product demands of software customers with marketable functions aims to increase the intermediate product production speed and thus the number of interim versions. In the light of these needs, software companies need to oversee their development processes to meet their customers' needs. But more importantly, companies are forced to change their processes in line with innovative practices in order not to cut back on the software production line. In this article, the software development methods of the three companies that develop software are examined in detail by the case study method, and the process change activities are systematically detailed. In the light of the information obtained, the experiences of the three firms in the software development methods are questioned and the effects of these acquisitions on the processes are discussed. As a result of the study, it has been observed that the software development success has a significant impact on the well-being of the process, and the software development teams are trying to design their own processes in the light of the gains they acquire.*

**Keywords:** Agile transformation, software development processes, Empirical software engineering, Case study

## 1 Giriş

Yazılım mühendisliği, yazılım ürününün tasarım aşamasından post-mortum (ürünün piyasadan tamamen yok olması) aşamasına kadar gerçekleşen tüm üretim basamaklarının sistematik bir şekilde yönetilebilmesini sağlayan mühendislik disiplinidir. Ancak, diğer mühendislik disiplinlerine göre ürün kalitesinin arttırılması için farklı yaklaşımlar gerektirir. Örneğin; yazılım geliştirme işleri takım tabanlı sosyal aktiviteler olduğundan takım üretkenliğini arttırmak için yazılım geliştiricilerin hem teknik bilgi birikimlerini hem de iletişim kabiliyetlerini iyileştirmeleri beklenmektedir. Yazılım ürünü, birçok kişinin katkısıyla birlikte zaman içinde daha da karmaşıklaşır. Bu karmaşıklık artışı ile beraber, yazılım geliştirme süreçlerinin genelleştirilmesi mümkün olmamaktadır. Ayrıca, seçilen süreç modelinin verimli olarak işletilebilmesi için yazılım ürünlerinin, yapısal ve sosyal aktivitelerin koordinasyonu da gerekmektedir. Yazılım geliştiren kişilere verilecek işlerin kişilik özellikleri veya davranış karakteristikleri gibi sosyal yapıların dikkate alınarak dağıtılması ürün kalitesini arttıracaktır [1]. Bu bilgiler ışığında, yazılım sürecinin değişen ihtiyaçlara göre güncellenmesi ve ürün entegrasyonu için kullanılan tüm teknik ve sosyal mekanizmaları tek bir üretim bandı üzerinde toplanması gereklidir [2]. Bu gereklilik bilinse de birçok ulusal şirket kazara geliştirme [3] yöntemi ile sistematik bir süreç yapısı olmadan sadece ürün geliştirmeye odaklanmaktadır. Ancak, Humphrey'nin *Gereksinim Belirsizlik İlkesine* [4] göre, yazılım sistemleri, kullanıcıları tarafından test edilmeden, yazılım gereksinimleri hiçbir zaman başarı ile tanımlanamaz.

Cockburn [5] çevik yöntemlerin getirdiği üretim modelinin üretkenliği arttırabilmesi için gerekli çevik pratiklerinin planlanan işlere uygun olarak seçilmesi ile sağlanabileceğini belirtmektedir. Çevik Manifesto [6],

i. İnsanlar arası etkileşimi korumanın öncelikli olduğunu söyler (sürdürülebilir takım çalışması),

ii. Kullanıcılardan gelen geribildirimle sürekli olarak tasarım kalitesi geliştirilmelidir (ürün esnekliği),

iii. Küçük sürümler, hızlı üretim ve geri bildirim döngüleri (değişiklikler için uyarlanabilirlik) ile kuralcı bir yazılım geliştirme yolunun izlenmesinin yerine adaptif bir yöntem yardımıyla değişikliğe gidilebilmesi,

iv. Çalışan ürünün, detaylı dokümantasyondan daha önemli olduğunu savunur [7].

Bu nedenlerle, uygulanan pratikler kolayca öğrenilebilmeli ve gerekli görülürse değiştirilmelidir. Çevik yöntemlerde, karmaşık bir süreç yapısı olmamasından ve dokümantasyona ayrılan zaman azaltıldığından dolayı, bazı araştırmacılar bu yöntemleri ürün odaklı basitleştirilmiş yöntemler olarak da adlandırdılar [8]. Highsmith ve Cockburn [9], çevik felsefeyle birlikte insanların ve iletişim süreçlerinin başarıya olan etkisinin vurgulanması gerektiğini savunmuşlardır. Çevik yöntemler, sözlü iletişimin ve takım çalışmasının önemini vurgulamakta, kapsamlı belgelere duyulabilecek ihtiyacı en aza indirmeyi amaçlamaktadır [10].

Ülkemizde yazılım geliştirmeyi yönetmek için en fazla kullanılan çevik yazılım geliştirme süreçlerinden biri Scrum yöntemidir. Bu yöntemde yapılan yinelemeler, koşu (sprint) olarak adlandırılır [11]. Tüm diğer çevik süreçlerde olduğu gibi, her koşu sonucunda çalışan bir ürün parçacığı üretilmesi şarttır. Bu yöntem, ayrıca ekip lideri yerine ekip koçu (scrum master) ve kendini organize edebilen bir takım yapısına dayalıdır [12]. Kanban, çevik yaklaşımların sunduğu zaman kutulu iterasyonlara alternatif olarak akış tabanlı bir üretim modeli içeren alternatif bir çevik yöntemdir [13]. Temel çalışma prensibi, iş akışlarının bir pano vasıtası ile görselleştirilmesi ilkesine dayanır. Scrumban yaklaşımı, Scrum ve Kanban yöntemlerinin en uygun pratikleri seçilerek özelleştirilmesi yoluyla, müşteri gereksinimlerinin yinelemeler üzerindeki olumsuz etkisini azaltmak için düşünülmüş karma bir modeldir. Bu yöntem, yazılım geliştiren organizasyonların kaynak kod bakımı veya yazılım onarımı yapmalarını kolaylaştırmaktadır. Ampirik çalışmalar göstermektedir ki yazılım yaşam döngüsünün ilerleyen aşamalarında özellikle koşu uygulamaları bakım süreçlerini güçleştirmektedir [14]. Bazı koşularda işlerini bitiren yazılım geliştiriciler koşu bitene kadar işsiz kalabilmekte, bu durum ekip dinamiklerini olumsuz yönde etkileyebilmektedir. Araştırmacılar, bu sorunu adreslemek için Scrumban yaklaşımını önermektedirler [14].

Bu çalışmanın amacı ulusal yazılım sektöründe yapılmış üç vaka çalışması yardımıyla, yazılım süreç seçimi, uygulaması ve yönetimi hakkında bilgi, fikir ve tecrübe paylaşımını sağlamaktır. Bu çalışma ile özellikle bu konuda akademik ve endüstriyel anlamda edinilen tecrübelerin, vaka çalışması olarak raporlanması hedeflenmiştir. Bu konuda yürütülen çalışmalarda, konu üzerinde danışmanlık yapılan üç şirket seçilmiştir. İncelenen üç şirket, müşterilerden gelen talep ve endüstrideki değişim ve yeniliklerden dolayı kullandıkları yazılım geliştirme yöntemlerinden yenilikçi yazılım geliştirme yöntemlerine geçiş yapmak istemektedirler. Tüm firma bilgileri, güvenlik açısından ismi paylaşılmadan verilecektir. Bu çalışma sırasında toplanan tüm veriler, yazılım şirketlerinin ticari çıkarlarına zarar vermeyecek şekilde, gizlilik içinde tutulacaktır. Birinci şirket (Firma A) güvenlik ürünleri geliştiren, orta ölçekli bir organizasyondur. İkinci şirket (Firma B), mobil uygulamalar geliştiren ve dış pazarı hedefleyen orta ölçekli bir yazılım firmasıdır. Üçüncü şirket (Firma C) oyun ve interaktif uygulamalar geliştiren küçük ölçekli bir firmadır. Bu çalışma sırasında bu üç firmanın yazılım süreç uygulamaları ve üretim performansı incelenerek, önerilen değişimlerin gelişim sürecini ve şirket performansını nasıl etkilediğine ve süreçlerini iyileştiren şirketler için edinimlerin ve tecrübelerin nasıl kazanca dönüştürüldüğü araştırılacaktır.

Yazılım mühendisliğinde yapılan bilimsel araştırmaların bir amacı, yazılım yaşam döngüsünü incelemek, şirketlerde olup biten olayları sistematik yöntemler ışığında anlayabilmek ve daha önemlisi gerektiği zaman müdahale edebilmektir. Bu çalışmada araştırma yöntemi olarak deneysel (empirical) yazılım mühendisliği yöntemlerinden olan keşif vaka çalışması (exploratory case study ) yöntemi uygulanmıştır. Araştırmacı bu konuda uluslararası deneyime ve bilgi birikime sahiptir. Bu çalışmada kullanılan teknik, katılımcıların kendi sözel anlatımlarından çıkartılan zihinsel tutum ve sosyal davranışları inceler ve onları doğal çalışma ortamlarında elde edilen bilgileri sistematik olarak sınıflandırır. Toplanan veriler, sözcükleri yakından incelemek ve katılımcıların görüşünü ve daha ileri denetimi yapmak için bağlamsal bir çerçeve içinde raporlandırılır. Bununla birlikte, tipik olarak nitel araştırmalar küçük gruplar halinde veya sınırlı sayıda katılımcı ile yürütülmektedir. Çalışma sonucunda elde edilen bulgular üzerinden analizler yapılmıştır. Bu makale öğrenilen derslerin diğer araştırmacılarla paylaşılması amacıyla düzenlemiştir.

Bu makalenin devamı şu şekilde yapılandırılmıştır: İkinci bölümde çalışmada kullanılan araştırma yönteminin detayları açıklanmıştır. Üçüncü bölümde yapılan vaka çalışma analizi anlatılmış, kalitatif analiz sonuçları sunulmuştur. Dördüncü bölümde ise yapılan çalışmadan çıkarılan sonuçlar ve öneriler verilmiştir.

## 2 Yöntem

Vaka çalışması, bilimsel araştırmalara cevap bulmak için sıklıkla kullanılan çok boyutlu bir etkinliktir. Bilimsel araştırmanın ilk günlerinden beri, genellikle bir olayın bir veya birden çok örneği sınırları içinde doğal ortamında incelendiğinde kullanılan klasik bir yaklaşımdır. Vaka analizi, yazılım mühendisliği alan araştırmalarında ampirik çalışmalar yapmak için uygun bir metodoloji olarak kabul edilebilir. Creswell'e [15] göre araştırmacı, çeşitli veri toplama prosedürlerini kullanarak, zaman ve etkinlikle (bir olay, süreç, kurum ya da sosyal grup) sınırlı tek bir varlığı ya da olguyu inceleyen vaka incelemeleri ile ilgili ayrıntılı bilgi toplar. Başarılı bir vaka incelemesi araştırması yürütmek için, vaka incelemesinin yapıldığı alanın erişilebilir olması ve kilit kişi ve gerekli kaynakların mevcut olması gerekir. Yin [16], bir vaka çalışmasının, verilen bağlamda belirsiz sınırlar içeren çoklu bilgi kaynaklarına dayanan gerçek yaşam durumlarındaki en modern fenomeni inceleyen ampirik bir araştırma olması gerektiğini ileri sürmektedir. Bir vaka çalışmasının bir araştırma stratejisi olarak ne anlama geldiğine ilişkin olarak Verschuren [17] vaka çalışmaları için şunu iddia etmektedir: "*Bir vaka çalışması, yineleyici ve paralel bir şekilde takip ederek, bütünsel nitelikte olabilecek bir araştırma desenidir.*" Kitchenham ve diğerlerine göre [18], vaka analizlerinin kullanımı birkaç avantaja sahiptir:

i. Yazılım mühendisliği faaliyetleriyle birleştirilebilir,

ii. Endüstriyel projeler kullanılıyorsa, veri toplamak için proje büyüklüğünü artırmaya gerek olmayabilir,

iii. Araştırmacının gerçekleşmiş veya beklenen faydaları daha iyi değerlendirmesini sağlar.

Tipik bir vaka incelemesi, tek bir vaka veya her ikisi de bütünsel (tek) birim veya gömülü (çoklu) analiz birimlerine dayanan çoklu vakalardan oluşturulabilir ve bu nedenle farklı vaka inceleme modelleri mevcuttur:

i. Tek kutuplu bütüncül,

ii. Çok kutu gömülü,

iii. Çok kutu bütünlüklü [16].

Vaka incelemesi, genellikle, toplanan verileri araştırmacıya bir dizi sonuç çıkarmasına olanak tanıyan ilk araştırma soruşturması ile bağlantı kurmak için bir araç olarak kullanmaktadır. Şirketler ile varılan mutabakat sonucunda, vaka çalışmalarına katılan katılımcılar ve şirketlerin isimleri yapılan çalışmanın herhangi bir aşamasında paylaşılmayacaktır. Bu sayede yazılım geliştiricilerin sorular üzerindeki gerçek düşüncülerine ulaşılması hedeflenmektedir.

## 3 Endüstriyel vaka çalışma analiz ve sonuçları

Bu araştırmadaki vaka incelemeleri, 2015-2017 yılları arasında Ankara'da bulunan, yazılım geliştiren üç farklı organizasyon üzerinde yapılan çalışmaları kapsamaktadır. Araştırma, şirketlere yapılan süreç iyileştirme danışmanlığı sırasında her Cuma günü 15.00'ten 18.00'a kadar olmak üzere iki yıllık bir süre içerisinde gerçekleştirilen veri toplama aktivelerine dayalıdır. Bu faaliyet doğrultusunda, araştırmacı yazılım uzman ve geliştiricileri ile yarı yapılandırılmış görüşmeler gerçekleştirilmiştir. Bu tür görüşmeler, katılımcılardan aynı bilgilerin alınması için önceden belirlenen birtakım soruların sorulmasıyla gerçekleştirilir [19]. Veri toplama sırasında amaçlı örnekleme yöntemlerinden ve ölçüt örnekleme yönteminden faydalanılmıştır. Bu yöntem belli kısıtları karşılayan bir örneklem grubu yardımıyla, araştırma açısından daha zengin vakaların detaylı olarak incelenmesine fırsat vermektedir [20]. Bu çalışmaya katılacak olan bireylerin yazılım geliştirme konusunda en az üç yıl deneyimli olmaları, Ankara'da ikamet etmeleri ve yazılım yaşam döngüsünün tüm evrelerinde az da olsa tecrübeli olmaları dikkate alınmıştır.

Bu çalışmaya belirtilen kısıtlar çerçevesinde, vaka çalışması için seçilen üç şirketten toplam 18 kişi katılmıştır. Yazılım geliştiricilerin demografik özellikleri şu şekildedir. Katılımcıların, 5'i kadın, 13'ü erkektir. Araştırma grubunda, 3 yıldan az tecrübeli 3 kişi, 3 ila 5 yıl arası deneyimli 10 kişi, 5 ila 10 yıl arası iş tecrübesine sahip 5 kişi bulunmaktadır. Katılımcılardan 2'si doktora düzeyinde eğitim almıştır. 6 katılımcı yüksek lisans derecesi, geri kalan 10 kişi ise mühendislik fakültesi bilgisayar ve elektronik bölümlerinden lisans diplomasına sahiptir. Vaka araştırması için seçilen firmalar çevik dönüşüm zorlukları ile baş etmek için araştırmacıdan danışmanlık alan firmalardır. Araştırmaya katılan kişiler çevik dönüşüm süreci içerisinde stratejik kararlar alabilen ve dönüşüm sürecini yöneten kişilerden seçilmiştir.

Araştırmanın iç geçerliliğini sağlayabilmek ve tüm verileri kayıt altına almak adına katılımcılara yapılan yarı yapılandırılmış görüşmeler kayıt cihazı ile kaydedilmiştir. Şirketler ve katılımcılar arsındaki bağlantıyı sağlamak için bir kodlama geliştirilmiştir. Ses kayıt transkriptlerinde katılımcılar K, Firma A'dan katılan kişiler AK ile kodlanarak katılımcı numarası ile belirlenmiştir. (Ör: BK3 kısaltması; B firmasından üç numaralı katılımcıyı temsil etmektedir). Kayıt edilen konuşmalar analiz edilmek üzere yaklaşık atmış sayfalık bir dokümana dönüştürülmüştür.

Yapılan görüşmelerde, öncelikle şirketlerin yazılım süreçleri, detaylı olarak tartışılarak şu sorular ile başlanmıştır: "Kullanılan süreç modellerinin organizasyon çapında işlevselliğe olan etkisi nedir?", "Klasik ve çevik geliştirme yöntemlerin kullanımının süreçlerin iyileştirilmesine olan etkileri nelerdir?", "Süreçlerin iyileştirilmesi için yapılan değişiklikleri üretim ve verimlilik üzerine etki sağlıyor mu?"

Bu soruların ardından ikinci aşama olarak, yazılım geliştirme süreçlerindeki değişim faktörü ile baş etme yöntemleri sorgulanmış ve elde edilen bilgi sistematik olarak kıyaslanmıştır: "Süreç iyileştirme çabalarının yazılım gereksinimlerinde sıkça gözlenen değişimle baş etmeye olan etkisi nedir?", "Bu etkinin büyüklüğü için sahip olunan görüşün yazılım geliştiren organizasyon içindeki rolleri ile ilişkisi var mıdır?" sorularına yanıt aranmıştır.

Katılımcılara alınan yanıtlar doğrultusunda açık kodlama yöntemi ile kodlama yapılarak yazılım geliştirme süreç değişimini etkileyen faktörleri irdelemek için kategoriler oluşturulmuştur. Araştırmada elde edilen transkripsiyonlar, çalışmalardaki yanılma payını azaltmak için bir uzman ile birlikte metin, cümle ve paragraf bazında incelenmiştir. Veri parçacıklarının anlamları görüş birliği sağlanana kadar sorgulanmıştır. Sorulan sorulara verilen yanıtlara göre elde edilen veriler incelenmiş, NVivo paket programı yardımıyla analiz edilerek, beş farklı kategori oluşturulmuştur. Bu kategoriler vaka çalışması sırasında en fazla tartışılmış olaylar ile ilgili bilgi birikimini ifade eder. Tablo 1'de yapılan kodlama sonucunda elde edilen bulgular sunulmuştur. Elde edilen bilgiler ışığında, üç firmanın da yazılım gelişime süreçlerindeki beklenen ve beklenmeyen değişiklikler karşılaştırılarak, edindikleri tecrübeler ve bu edinimlerin yazılım geliştiren organizasyonlar üzerindeki etkin faktörler araştırılmıştır. Araştırmacı, yapılan görüşmelerde elde edilen bulguları tema bazlı olarak inceleyerek bir çerçeve model oluşturmuştur.

Vaka çalışmalarında yapılan tüm ziyaretler (toplam 27) görüşmelere başlandığı ilk altı aylık dönemde temalar ve kategoriler oluşturulmaya başlanmış ve devam eden görüşmeler ışığında sürekli olarak güncellenerek, Tablo 1'de sunulmuş olan en güncel haline getirilmiştir.

Birinci vaka çalışması olarak değerlendirdiğimiz Firma A, güvenlik ürünleri geliştiren, 1997 yılında kurulmuş bir firmadır. Firmanın şu anda Ankara ofisinde çalışan 30 elemanı, yazılım bakımı için çalıştırdığı (10 kişilik) gezici bir ekibi bulunmaktadır. Firmanın merkezi Ankara olarak görünse de ürünlerin çoğu Ege ve Akdeniz bölgesinde bulunan şirketler tarafından tercih edilmektedir. Ürünlerin bir kısmı güvenlik ile ilgili çıkabilecek sorunlarla mücadele etmek için bu konuda şirket bazında ürün geliştirmeyen ama ürünlerine güvenlik modülü entegre etmeye çalışan küçük ve orta ölçekli firmalarca tercih edilmektedir. Şirket, şu anda ürettiği çözümleri bir araya toplayan, çok fonksiyonlu servis tabanlı ürün kümesi oluşturmak için çalışmalarını sürdürmektedir.

Şirket ilk yıllarında klasik plan-tabanlı yazılım geliştirme yöntemlerini benimsemiştir. Ancak, müşteri taleplerinin yıllar içinde değişmesi ve piyasa şartlarının hızlı ve teknoloji odaklı üretim istekleri şirketi çevik yöntemleri kullanmaya yöneltmiştir. Şirket, Türkiye'deki diğer birçok yazılım firması gibi Scrum metodolojisini kullanmayı tercih etmiştir. Bu durumun sebepleri şu şekilde sıralanabilir:

i. Üretim için 5-8 kişilik yazılım takımı ihtiyacı,

ii. Müşteri temsilcisinin takımla sürekli bir arada bulunması gerekliliği,

iii. 20 ile 26 günlük koşular üzerinde ürün parçalarının üretim gerekliliği.

Şirket özellikle bazı dönemlerde her bireyin sabit bir rol ile iş yapması yerine herkesin her işi yapabilmesi ilkesine uygun bir kültür geliştirmiştir. Üretim planlaması çevik yöntemler sayesinde dört veya altı aylık teslimat dönemlerinden on veya on iki haftalık dönemlere çekilerek müşteri memnuniyeti sağlanmıştır.

Tablo 1: Yazılım Geliştirme Süreç değişimi etkileyen faktörler.

| Kategori | Tema | Kod |
|---|---|---|
| Organizasyon | Üretkenlik | • Bireysel<br>• Takım<br>• Örgütsel |
| | Sosyal Yapı | • Kişilik<br>• Uyum<br>• Yenilikçi |
| | Roller | • Geliştirici<br>• Testçi<br>• Lider |
| Gereksinim Analizi | İşlevsel | • Maliyet<br>• Memnuniyet<br>• Yeniden kullanma |
| | İşlevsel olmayan | • Kullanılabilirlik<br>• Güvenilirlik<br>• Birlikte çalışılabilirdik<br>• Şirketin büyüklüğü<br>• Entegrasyon<br>• Müdahale |
| İyileştirme aktiviteleri | Model | |
| | Çerçeve | • Kabiliyetler<br>• Tecrübe<br>• Esneklik |
| Çevik Metotlar | Lean<br>Scrum<br>Kanban | • Scrumban uygulaması<br>• Çeviklik Değerleri<br>• Süreç Disiplini |
| Etkileşim | Çevik | • Eşli programlama<br>• Günlük Scrum Toplantıları |
| | Klasik | • Kod gözden geçirme<br>• Toplantı süreleri |

AK1 şu şekilde bir yorumda bulunmuştur.

"Çoğunlukla çevik metodolojileri kullanıyoruz. Birini diğerlerine tercih etmemizin nedenlerinden biri, geliştirme sırasında ürüne müdahalenin daha kolay olmasıdır. Ayrıca, bu yolla değişen gereksinimleri daha hızlı karşılamayı hedefliyoruz. Örneğin, ürünlerimizin çoğunun özellik tabanlı olması, çevik bir metodoloji kullanmanın kolaylıklarından biri, bu aslında çok değişken ve fonksiyonel olmayan gereksinimleri kolayca güncellememiz anlamına geliyor."

Firma A'da yer alan diğer bir şirket çalışanı (AK5), süreç entegrasyonu ile ilgili söyle bir yorum yapmıştır.

"Şirketin bizim takımdan beklentilerini karşılamak adına, ekip olarak alacağımız kararlar doğrultusunda kullandığımız süreç veya yöntemleri değiştirebilmeliyiz. Biz Kanban'ı bir ay denedik. Özellikle ürün destek takımı olarak yararlarını gördük. Ancak, üretim hızımız beklenenin altına düşünce Scrum'a geri dönüş yaptık. Çevik manifestoların getirdiği kuralları benimsemek ve projelerimiz aracılığıyla deneyimlemek için çalışıyoruz. Gerçekte yapmaya çalıştığımız, aslında çevik manifestoya dayanan birçok metodolojinin özelliklerini birleştirerek ekibimizle en iyi örneği oluşturmaktır. Çevik yöntemlerin getirdiği çeviklik sayesinde yeni şeyler deneyimleyebiliyor ve neyin işe yaradığını görebiliyoruz."

Şirket için önemli bazı konuları AK8 tarafından şu şekilde dile getirilmiştir.

*"…, her proje için geçtiğimiz süreçlerden bahsetmek hoşuma gidiyor. Öncelikle sıraya koyduğumuz işler için 6 aylık bir plan yapıyoruz ve bunları sprint olarak adlandırılan 2 haftalık parçalara ayırıyoruz. Öncelik, genel olarak projelerimizdeki zorlayıcı işlevleri kodlamak, ancak öncelik tanımlamak için bir yol olarak zorluk dışında diğer faktörler de mevcut. Genellikle ilk iki sprint sonrasında ürün parçacığımız bir şeylere benzemeye başlıyor. Bence günlük toplantılar ve retrospektif toplantıları çok önemli. Bu buluşmalar ile bizi gidişat ile ilgili rahatsız ve mutlu eden şeyler tartışılabiliyor. Süreç aslında takımın kültürüne göre evrilmeli, bence, sürekli iyileştirme şart. Bu sayede durmuyoruz ve sürekli daha iyi bir ürün üretmek için takımca, şirketçe çaba harcıyoruz."*

Firma A özellikle servis tabanlı ürün geliştirme konusuna ilerlemek istemektedir. Ama bunun yanında, yapılan tüm görüşmeler sonucunda bazı paydaşların yenilikçi ürün geliştirme yöntemleri konusunda çeşitli hassasiyetleri olduğu gözlenmiştir. Şirket yeni projelerinde bulut tabanlı mimariden yararlanmak istemektedir. Ancak, geleneksel düşüncedeki bazı müşteriler için bu durum bir sorun olabilir. Örneğin, bulutta saklanan verinin güvenliği ve olası veri ihlalleri bazı potansiyel sorunlara işaret etmektedir. Bunun dışında şirket kültürüne paralel olarak, çevik yöntemler, geleneksel yöntemlere tercih edilmekte, bu da iyileştirme süreçlerini etkilemektedir. Şirket çalışanlarının hem fikir olduğu bir başka konu da yazılım geliştirme süreçlerinin ürünlerin bakımını da kolaylaştırmasıdır. Bu durum, bazı yazılım takımlarının Scrum ve Kanban'ın bir karışımı olan Scrumban'a geçme isteklerinde haklı olduğunu göstermektedir.

İkinci olarak değerlendirdiğimiz şirket (Firma B), mobil uygulamalar geliştiren 2013 yılında kurulmuş ve çoğunlukla dış pazarı hedefleyen 20 kişinin çalıştığı, orta ölçekli bir yazılım firmasıdır. Firmanın merkezi ODTU Teknokent'te bulunmaktadır. Geliştirdiği ürünler genel olarak Avrupa yazılım pazarı tarafından tercih edilmektedir. Geliştirdikleri yazılım ürünleri mobil stok kontrol ve envanter sistemlerinin otomasyonu ile ilgilidir. Şirket, kurulduğu ilk yıllarda anlaması ve yönetmesi kolay olduğu düşüncesiyle şelale yöntemini kullanmayı denemiştir. Ancak, bu yöntemle elde edilen ürün tek parça olduğundan, projelerdeki gereksinim değişiklerinde sorunlar yaşanmıştır.

Buna ek olarak, müşteriye ürün üzerindeki değişimleri göstermek ve raporlama konusunda çeşitli sıkıntılar yaşanmış, müşteriden gelen geri bildirimlerin ürüne yansıtılması sırasında sıkıntılar yaşanmıştır. Dolayısıyla, şirket, kültürüne uygun daha farklı bir yazılım geliştirme metodolojisi arayışı başlatmıştır. Bu arayış şirketi çevik yöntemlere itmiş ve bu yöntemle işlerin birkaç haftalık sprintler halinde yapılması düşünülmüştür. Sprintler sırasında elde edilen bilgilerden takım için deneyim birikimi oluşturulmuş, bu sayede yazılım takımları, ürünün hangi parçalarda sorunlar yaşandığını net bir şekilde gözlenebilmiştir. Bu veriler ışığında, firmanın kullandığı

yazılım süreçlerinde gerekli iyileştirmeleri yapmak daha kolay olabilmektedir. Şirket, çevik metodolojiler yardımı ile projenin gelişim yönü değerlendirebilmiştir. Müşterinin istediği değişiklikler kolayca entegre edilebilmiştir. Bu bilgiler ışığında, şirket çevik geliştirme pratiklerini kullanarak, istenilen özelliklerde ürün üretmiş ve müşterilerinin memnuniyetini sağlayabilmiştir.

BK2 şu şekilde bir yorumda bulunmuştur.

*"Şelale yöntemi, bir sorunla karşılaşıldığında derhal düzeltmelere ihtiyaç duyan, rekabet tehdidiyle baş etmek isteyen veya müşterinin isteklerine hızlı bir şekilde cevap vermek isteyen firmalar için uygun değildir. Şelalenin modeli uzun vadeli bir vizyona sahiptir. Hızlı ve sürekli gereksinimlerinin değişen mobil üretim için pek çok sıkıntı oluşabilmektedir. Orta ölçekli projeler, proje süresi boyunca %30'luk bir değişime maruz kalmaktadır. Şelale modeli kullanmak bu tür ürünler için uygun olmayabiliyor, çünkü projede yapılan her değişiklik maliyeti artırıyor. Şelale modeli, gelişime ve değişime uyum sağlayabilen bir model olmadığı için müşteriyi memnun etmek zor! Projenin başında neye ihtiyaç duyduklarını ve ne istediklerini belirleyemeyen müşteriler projenin iptaline neden oluyor."*

Diğer bir B firması çalışanı (BK3) çevik yazılım süreçleri kullanmaya takım olarak bir yıl önce geçtiklerini belirtmiştir. Firma çalışanları, 2 aylık bir çevik yazılım eğitiminden sonra Scrum kullanmaya başlamışlardır.

*"Çevik yöntemler kullanmaya başladık, çünkü onlar yazılım geliştirme işlerimiz için daha uygundu. Çoğunlukla uluslararası piyasa için yazılımlar üretiyoruz rakiplerimiz çok güçlü; bu nedenle, hızlı prototip üretmemiz ve mümkün olduğunca çabuk test etmemiz gerekiyor. Taktir edersiniz ki çevik süreçler bizim için daha iyi bir seçenektir. Çevik süreçleri kullanmayı tercih etmemizin bir başka nedeni, çok uzun aralıklarda yazılan kod parçalarında hata olasılığı artmaktadır. Bu hataların teslim edilmeden önce bulunması ve düzeltilmesi şarttır. Dolayısıyla çoğunlukla aylık sürümler ve yeni eklenen fonksiyonların kontrolü gerekir. Ayrıca, bizim işlerde müşteri sık aralıklarla gereksinimlerini değiştirebilir veya ürüne yeni bir şeyler eklemek ister. Değişim fazla olduğunda, çevik süreçler daha uygundur. Çevikliğin en önemli avantajlarından biri hem müşteri hem de takımlar ve ekipler arasındaki kurulması gereken sürekli iletişimdir. Bizim ekipler, müşteri ile daha sık iletişim kurmaya başladıklarında ne yapılması gerektiğini ve müşterinin ne tür değişiklikler istediğini daha kolay anlıyorlar. Doğru bir iletişim varsa başarılı bir ürün geliştirme olasılığınız artar. Sonuç olarak, çeviklik süreçleri kullanmaya başladıktan sonra müşterinin yazılım ürünlerimizden daha memnun olduğunu fark ettik."*

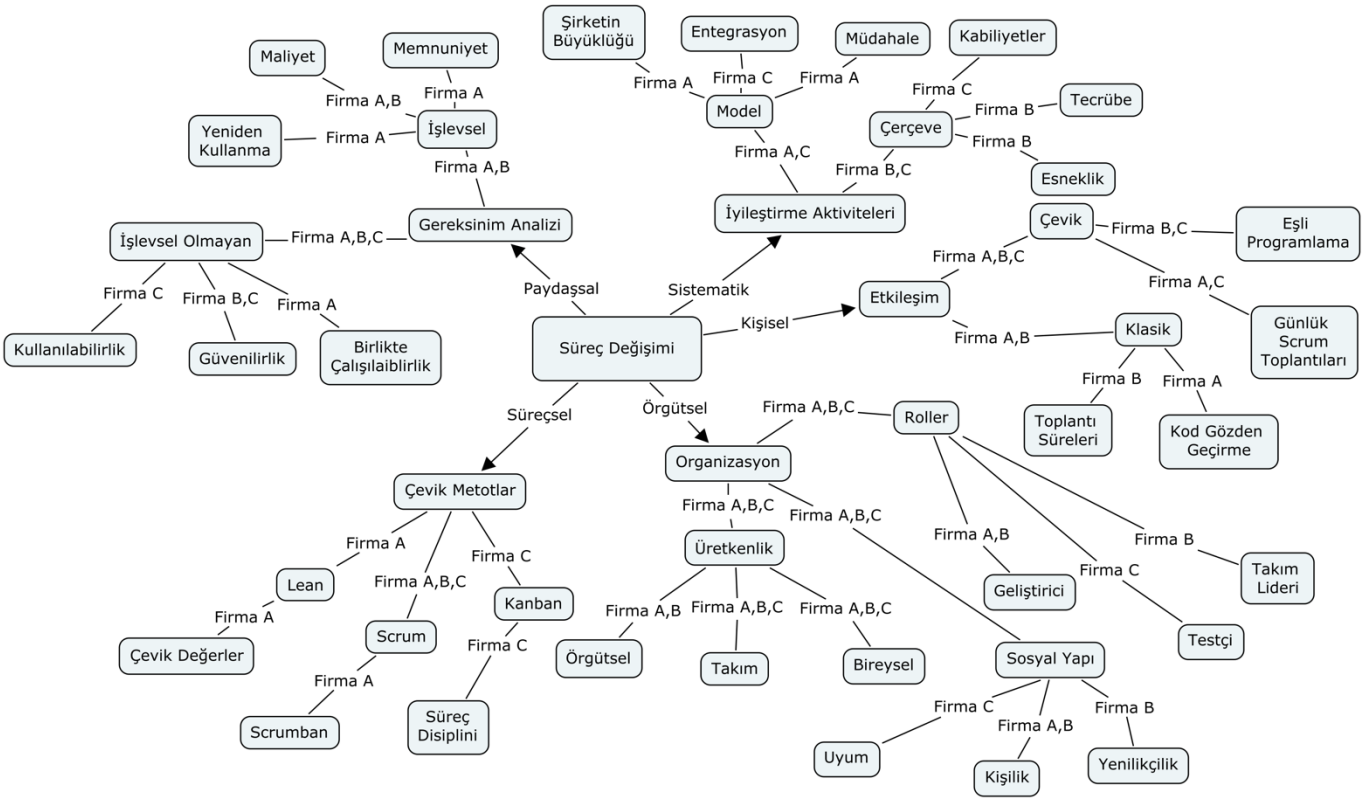
Üçüncü şirket (Firma C) merkezi ODTU Teknokent'te bulunan, oyun ve interaktif uygulamalar geliştiren, 2007 yılında kurulmuş ve 10 kişinin çalıştığı, küçük ölçekli bir yazılım geliştiren organizasyondur. Şirket, küçük çaplı oyun geliştirme projelerinin yanı sıra, yapı ve mimari işleri sunum ve simülasyon modelleri ile ilgili projeler yürütmektedir. Şirketleri oyun ve multimedya sektöründeki akışkan piyasa şartlarından dolayı, yazılım geliştirme için çevik süreçleri tercih etmektedir. Firma başarılarını tanımlarken, ürünlerindeki özelliklerden; güvenlik, hız, kullanılabilirlik gibi işlevsel olmayan gereksinimleri ön plana çıkartmaktadır. Çevik yazılım süreçleri konusundaki tercihlerini özellikle müşteri isteklerine hızlı karşılık verebilme kabiliyetlerine dayandırmaktadırlar. Firma için üretim süreçlerinin her aşamasında geribildirim çok önemlidir. Dolayısıyla, projenin aşamaları sırasında ortaya çıkan değişiklik istekleri veya sorunlar daha erken fark edilebilmektedir.

*"…, performansı artırmak ve dinamik sahne tasarımı için ürünün ve işlevselliğinin sürekli olarak yenilenmesi gerekiyor. Değişiklikleri yaptıkça genel yapıda çıkabilecek aksaklıklar artsa da aslında bizim başka bir çaremiz yok. Bizim işlerde bazen bulut üzerinde aynı proje ile birkaç kişi birden güncelleme yapabilir. Bu bence sadece çevik yöntemler kullanılarak mümkün olabilir. Projelerde gözlediğim bir başka konu ise: uzun süre etkileşim kurmayan yazılım pratisyenleri zamanla yaptıkları işe olan bağlılık ve duydukları ilgiyi kaybettikleri gibi aynı zamanda büyük bir motivasyon kaybına uğrayabiliyorlar. Tüm bu gerçekler göz önüne alınırsa çevik yazılım süreçleri bizim işlerimizi oldukça kolaylaştırıyor."*

Araştırma sırasında elde edilen bilgiler incelendiğinde, katılımcılar çevik yazılım geliştirme süreçleri iyi planlanıp, doğru uygulandığı durumlarda etkili bir geliştirme yöntemi olduğu yönünde hemfikirdir. Çalışmaya katılan uzmanlar, süreç değişim çabalarını motive edici bulmasalar da yazılım geliştirme sırasında yeni yöntemlere yer verilmesi ve varsa bu süreçlerin uygulama kolaylıklarını ortaya çıkaracak yollar bulunması gerektiğini belirtmişlerdir. Ayrıca katılımcılar bu tür aktiviteler için geliştirilecek oyun destekli uygulamaların özellikle çevik yazılım geliştiren organizasyonlara katkı sağlayacağı ve karşılaşılabilecek aksaklıklara karşı hazırlıklı olma, yinelemeleri planlama açısından önemli bir etkisi olabileceği konusunda görüş bildirmişlerdir. Bununla birlikte, birçok katılımcı ortaya çıkan bulguların ve öğrenilen derslerin yazılım geliştirme süreç değişimi çalışmaları dışında da kullanılması ve şirket içi erişime açık olacak şekilde sunulmasının şirket hafızasına katkı sağlayacağını düşünmektedirler.

Şekil 1'de görüldüğü üzere yazılım süreç değişimlerine yönelik görüşler paydaşsal, sistematik, kişisel, örgütsel ve süreçsel bazlı 5 temel kategori ve 28 tema yardımıyla sınıflandırılmıştır. Katılımcıların yarı yapılandırılmış anket sorularına verdikleri yanıtların kodlanmasıyla oluşturulan kategoriler ve kodların sınıflandırılması ile elde edilen temalar, görsel olarak sunulmaktadır. Şekil 1'de üç firma ile yapılan görüşmeler sırasında firmaların hangi kategorileri etkilediği görülmektedir. Buna ek olarak, ortaya çıkan temaların hangi şirket ile yapılan görüşmelerden elde edilen bulgulara dayandığı da izlenebilir. Yazılım geliştirme süreç değişimi katılımcıların birçoğu tarafından gereksinim analizi ile ilişkilendirilmiştir. Gereksinimlerin yeniden kullanılması, maliyet, müşteri memnuniyeti, kullanılabilirlik, güvenilirlik ve birlikte çalışabilirlik yazılım geliştirme süreç değişimini etkileyen önemli etmenler olarak göze çarpmaktadır. Model ve çerçeve açısından iyileştirme faaliyetlerinin özellikle firma büyüklüğü, entegrasyon ve müdahale kabiliyeti katılımcılar tarafından önemsenmektedir.

Yazılım geliştirme süreç değişimi için önerilen bir yöntem deneme yanılma yoluyla çevik pratiklerden işe yarayanların bulunması ve gerekli görülen aktiviteler için kullanılmasıdır. Eşli programlama, günlük Scrum toplantıları, kod gözden geçirme yazılım geliştirme süreç değişimi etkileyen pratikler olarak belirlenmiştir. Süreç aktivitelerini en çok etkileyen roller yazılım geliştirici, yazılım testçisi ve takım lideri olarak göze çarpmaktadır. İlginç bir çıkarım olarak, sistem analizcisi görevi katılımcılar tarafından yazılım geliştirme süreç değişimi etkileyen bir rol olarak önerilmemiştir.




Şekil 1: Yazılım süreç değişimlerine yönelik görüşlerin firma bazlı düzenlenmiş kategori ve temaları.

Katılımcılara göre verimliliği arttırmak için örgütsel bireysel ve takımsal faaliyetler ayrı ayrı değerlendirilmelidir. Ek olarak, Şekil 1'e göre, çevik değerler ve süreç disiplini yazılım geliştirme süreç değişimi için kontrol altında tutulması gereken önemli parametreler olarak kabul edilebilir.

Katılımcıların yorumlarına göre, iyileştirme faaliyetleri yazılım geliştirme modeli ve kullanılan çerçeveye dayalı olarak incelenmelidir. Şirketin büyüklüğü, sürekli entegrasyon ve ürüne ani müdahale edebilme kabiliyeti yazılım geliştirme süreç değişimi sırasında, yazılım yaşam döngüsünü etkileyen önemli etmenler olarak görülmektedir. Ayrıca, iyileştirme aktiviteleri için kullanılacak olan çerçeve yapısının seçiminin şirketin sahip olduğu kabiliyetler ve tecrübelerin yanı sıra çerçevenin esnekliği konusunu da dikkate alınarak seçilmesi gerekliliği anlaşılmaktadır.

## 4 Sonuçlar

Bu çalışma yazılım süreç değişimlerinin ulusal yazılım endüstrisindeki etkilerini tartışmak ve uzman katılımcıların süreçler hakkındaki görüşlerini ortaya koymak amacıyla yapılmıştır. Yazılım süreçlerini değiştirmekte olan üç farklı şirketten sağlanan katılımcılarla yarı yapılandırılmış mülakatlar yapılmıştır. Katılımcılardan şirketlerindeki süreç değişimlerini ve bu durumdan etkilenen yazılım geliştirme faaliyetlerini değerlendirmeleri istenmiştir. Elde edilen bulgular süreç değişimlerinin sancılı bir dönem olduğunu göstermiştir. Birçok katılımcı, bu vaka çalışmasının değişim süreç yönetimine olumlu katkı sağlayacağını düşündüğünü belirtmiştir. Değişim ve iyileştirme faaliyetlerinin yazılım geliştiricileri hem bireysel hem de örgütsel bağlamda etkileyen farklı faktörler olduğu anlaşılmaktadır. Bulunan faktörlerin daha detaylı analizi ile süreç içindeki olumsuzlukların giderilmesinde etkili olacağı anlaşılmaktadır.

Bu araştırmada, üç yazılım şirketinin yazılım geliştirme süreçlerini ayrıntılı olarak incelenmeye çalışılmıştır. Bu inceleme sırasında her üç şirkette de yazılım ürününün beklenen ara çıktılarının hızlandırılması ile ilgili paydaşlardan talep geldiği fark edilmiştir. Müşteriler, yeni sürüme daha hızla kavuşmak, değişimi kısa yinelemeler ile gerçekleştirmek ve her hafta yeni özellikler görmek istemektedirler. Yazılım geliştirme araçlarındaki özellikle takım ve kaynak kod otomasyonundaki yenilikler ara çıktıların daha hızlı üretilebilmesini kolay hale getirmektedir. Bu amaçla, üç şirkette yapılmış olan analiz, yazılım geliştirme süreç değişiminin organizasyonun (verim, roller ve sosyal yapısının), gereksinim analizi (işlevsel veya işlevsel olmayan fonksiyonların), iyileştirme süreçleri, çevik metotlar ve etkileşim özelliklerine bağlı durumsal faktörlerle değiştiğini ortaya çıkartmıştır. Yazılım geliştirmek için kurulan süreç yapısının bu faktörlere göre iyileştirilmesi mümkün olabilmektedir. Yapılan vaka çalışması ile ulaşılan bazı sonuçlar şu şekildedir.

Görüşülen yazılım geliştiricilerin büyük çoğunluğunun yeni metodolojileri takip etmeye çalıştıkları gözlenmiştir. Bununla birlikte, yazılım geliştiricilerin birçoğu yeni süreçlere geçme konusunda sakıngan görünmektedirler. Görüşülen yazılım geliştiricilerin önemli bir bölümü, yeni çıkan yazılım geliştirme araçlarını kurup kullanmak istediklerini belirtmişlerdir. Yazılım geliştiren organizasyonların çoğunun çevik yöntemler kullanmak istediği ancak iş teslim aralıklarının kısalması ve buna bağlı olarak kalite parametrelerinin bozulması durumlarından ötürü şikâyetçi oldukları görülmüştür. Şikâyetlerden bir diğeri ise; şirketler herhangi bir çevik yazılım sürecini seçerken hangi pratiklerin uygulanması gerektiğine

dair örgütsel bir karar almamalarıdır. Ayrıca, katılımcılar yazılım geliştirme süreç değişimi etkileyen farklı faktörler olabileceği göz önüne alınırsa daha doğru sonuca ulaşmanın mümkün olacağını düşünmektedirler. Süreç değişimleri iyileştirme amaçlı olarak yapılan ve yönetilmesi zor aktivitelerdir. Yazılım geliştiren organizasyonlar, (değişen) teknoloji, iş baskıları, müşteri beklentileri ve personel gibi kilit yönleri dengelemek için çok boyutlu sorunlarla uğraşmaktadırlar. Ayrıca bazı etmenler, zamanla proje ve süreçler içinde sürekli evirilerek karmaşıklığı arttıran baskın faktörlere dönüşebilmektedir. Yapığımız araştırma sırasında gözlediğimiz kadarıyla şirketler proje bazında edindikleri bilgi ve pratiklere dayanarak kendi süreçlerini inşa etme eğilimindedirler. Bu da hiçbir yazılım metodolojisinin projeye göre özelleştirilmeden yapılacak yazılım geliştirme işleri ile birebir uyumlu bir çözüm olamayacağına dair kanıt oluşturmaktadır. Önerilecek çözümler, projenin işlevsel olmayan gereksinimleri ile ilişkilendirebileceğimiz kalite parametrelerini göze alarak değerlendirilmelidir. Ayrıca, yazılım geliştirme sürecinin seçilmesi sırasında bu faktörler dikkatlice incelenerek sürece olan etkileri detaylı olarak araştırılmalıdır. Ancak, günümüzde yazılım geliştirme yöntemleri, süreç mühendislerinin tavsiyesine göre, projelere uygunluğu tam değerlendirmeden seçilmektedir. Bu tür tercihler yazılım ürün kalitesini olumsuz yönde etkilemektedir.

Bu araştırmada, yarı yapılandırılmış görüşmeler kullanılarak yazılım uzmanlarının süreç değişimi ile ilgili görüşlerini raporlanmıştır. Nitel çalışmaların doğasına paralel olarak katılımcıların görüşleri veya gerçekleşen olayları algılama şekilleri farklı olabilmektedir. Ayrıca araştırmacı, yazılım geliştiricilerin çalışma sırasında verdikleri bilgileri gerçek görüşleri olarak kabul etmektedir. Dolayısıyla, bu tip ampirik çalışmalarda katılımcıların görüşlerini doğru ifade etmeleri bir hayli önem taşımaktadır. Bu çalışma, sadece Ankara'da bulunan üç şirket ile yürütülmüştür. Diğer şehirlerde ve hatta ülkelerde bulunan yazılım şirketlerini aynı yöntemle inceleyen ileriki bir çalışma, yapılan çıkarımların geçerliliğini arttıracaktır.

## 5 Teşekkür

Dr. Emrecan ÇUBUKÇU'ya ve Dr. Özgür ERGÜL'e katkıları nedeniyle teşekkür ederim.

## 6 Kaynaklar

[1] Yilmaz M. A Software Process Engineering Approach to Understanding Software Productivity and Team Personality Characteristics: An Empirical Investigation. PhD Thesis, Dublin City University, Dublin, Ireland, 2013.

[2] Highsmith J. *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. 1st ed. New York, USA, Addison-Wesley, 2013.

[3] Brooks FP. *The Mythical Man-Month Essays on Software Engineering*. Boston, Addison-Wesley, 2013.

[4] Humphrey WS. *A Discipline for Software Engineering*. New York, USA, Addison-Wesley, 1995.

[5] Cockburn A. *Agile Software Development: The Cooperative Game*. Boston, USA, Pearson Education, 2006.

[6] Agile Alliance, "Agile Manifesto". http://www.agilemanifesto.org (30.07.2017).

[7] Boehm B, Turner R. "Balancing agility and discipline: Evaluating and integrating agile and plan-driven methods". *International Conference on Software Engineering*, Edinburg, Scotland, 23-28 May 2004.

[8] Boehm B. "Get ready for agile methods, with care". *Computer*, 35(1), 64-69, 2002.

[9] Highsmith J, Cockburn A. "Agile software development: The business of innovation". *Computer*, 34(9), 120-127, 2001.

[10] Goodpasture JC. "Project Management the Agile way: Making it Work in the Enterprise". Florida, USA, J. Ross Publishing, 2010.

[11] Schwaber K, Beedle M. *Agile Software Development with Scrum*. Vol. 1. New Jersey, USA, Prentice Hall, 2002.

[12] Schwaber K. *Agile project management with Scrum*. Washington, USA, Microsoft Press, 2004.

[13] Anderson DJ. *Kanban: Successful Evolutionary Change for Your Technology Business*. Washington, USA, Blue Hole Press, 2010.

[14] Yilmaz M, O'Connor R. "A scrumban integrated gamification approach to guide software process improvement: a Turkish case study". *Tehnicki Vjesnik (Technical Gazette),* 23(1), 237-245, 2016.

[15] Creswell JW. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. New York, USA, Sage Publications, 2013.

[16] Yin RK. *Case Study Research: Design and methods*. New York, USA, Sage Publications, 2013.

[17] Verschuren P. "Case study as a research strategy: some ambiguities and opportunities". *International Journal of Social Research Methodology*, 6(2), 121-139, 2003.

[18] Kitchenham BA, Brereton P, Turner M, Niazi MK, Linkman S, Pretorius R, Budgen D. "Refining the systematic literature review process-two participant-observer case studies". *Empirical Software Engineering*, 15(6), 618-653, 2010.

[19] Hove SE, Anda B. "Experiences from conducting semi-structured interviews in empirical software engineering research". *11th IEEE International Software Metrics Symposium*, Como, Italy, 19-22 September 2005.

[20] Shull F, Singer J, Sjøberg DI. *Guide to Advanced Empirical Software Engineering*. New York, USA, Springer Science & Business Media, 2007.