Uluslararası, Çoklu Bileşenli Çevik Yazılım Geliştirme Projelerinde 3. Partilerle Çalışma Deneyimleri

Şenay Demirel[1], Yağmur Kırkağaç[1]

Netaş Telekomünikasyon A.Ş., İstanbul, Türkiye
{stuna,yagmur}@netas.com.tr

Özet. İnternet ekonomisinin hızlı değişimi yazılım mühendisliğindeki belirli kuralları da aynı hızla değiştirmektedir. Çevik yazılım geliştirme süreçleri, projede üretilen değerin hızlı ve etkin bir şekilde gerçeklenebilmesi için gelişmiş bir sistemdir. Çevik yazılım geliştirme süreçleri, yazılım projelerinin yaşamsal döngülerinin verimli bir şekilde tamamlanmasını sağlamayı amaçlamaktadır. Netaş Telekomünikasyon A.Ş. çevik yazılım geliştirme süreçlerinin projelerde uygulanmasının uzun yıllardır bilgi birikimine ve tecrübesine sahiptir. Netaş, büyük ölçekli, uluslararası ve çoklu bileşenli projelerini, 3. partilerle ortaklaşa çalışarak yürütebilmektedir. Bu bildiride, uluslararası, büyük ölçekli ve 3.partiler ile birlikte yürütülen çevik yazılım projelerinin yönetiminde edinilen deneyimlerin paylaşılması amaçlanmış, karşılaşılan sorunlar ve bu sorunlara getirilen çözümler çevik yazılım geliştirme prensipleri ışığında incelenmiştir.

Anahtar Kelimeler: Çevik Yazılım Geliştirme, Çoklu Bileşenler, 3.Partilerle Yazılım Çözümleri Geliştirme, Yazılım Proje Yönetimi

Abstract. The enormous change in internet economy, also changes the dynamics of some software engineering rules. Agile software development process is an advanced system to realize key values of the projects fast and effective. Agile software development processes aim to accomplish the life-cycle of software projects efficiently. Netas Telecommunication Corporation has the knowledge and experience of agile software development processes which are applied to their projects. Netas is able to execute projects, which are especially large-scale, international and including multi-components, collaborating with 3rd parties. In this paper, international, large-scale agile software projects are conducted in collaboration with 3rd parties was aimed to share the experience of the management that, encountered problems and the solutions brought to these problems in the light of agile software development principles are examined.

Keywords: Agile Software Development, Multi-Components, Software Solutions Development with 3rd Parties, Software Project Management

1 Giriş

Günümüzde yazılım, birçok endüstrinin işleri için gerekli olmakla beraber, yazılım süreçlerinin başarı ile yürütülmesi için farklı yazılım disiplinleri uygulanmaktadır. Buna rağmen, birçok araştırma yazılım geliştirme projelerinin başarısını ölçebilecek bir formül ortaya koyulamadığını göstermektedir. Özellikle, çevik yazılım projelerinde birçok değişkene bağlı olmak, başarı ölçütlerinin formül ile ifadesini daha da zorlaştırmaktadır.

Standish Grup 1994'ten itibaren her yıl yazılım geliştirme endüstrisinin anlık durum görüntüsünü "CHAOS Raporu" isimli raporları ile yayınlamaktadır [1]. 2015 yılında yayınlanan CHAOS Raporu, çeşitli ülkelerde farklı yapıda projelerin oluşturduğu 50000 projenin analizlerini içermektedir [2]. Standish Grup 2015 yılı CHAOS Raporu'nda, son 5 yılda analiz edilen projelerin zaman, bütçe ve kalite değişkenleri ile tanımlanan proje başarı oranlarını CHAOS Manifestosu'nda tanımlandığı şekilde analiz etmişlerdir. Projelerin, başarı sonuçları Tablo 1'de özetlenmiştir [2].

Tablo 1. 2010-2015 Yılları Arasındaki Tüm Yazılım Projelerinin Başarı Oranları [2]

Başarı Oranı	2011	2012	2013	2014	2015
Başarılı	%29	%27	%31	%28	%29
Zorlanılan	%49	%56	%50	%55	%52
Başarısız	%22	%17	%19	%17	%19

Tablo 1'de görüldüğü üzere, son 5 yılda geliştirilen yazılım projelerinin yaklaşık %70'i hala üstesinden gelmesi zor veya başarısız olmuştur [2]. Araştırmacılar, yazılım projelerinin başarı oranlarını etkileyen faktörleri belirlemeye ve başarı oranlarını iyileştirecek yöntemleri araştırmaya odaklanmışlardır. Bu alanda yapılan araştırmalarda, çevik yazılım geliştirme disiplini diğer yazılım geliştirme disiplinleri arasında en popüler olanlardan biridir.

Çeviklik, hızlı ve kolay hareket etmenin gücü anlamına gelmektedir. Yazılım projelerinde çeviklik; hızlı, hafif,verimli ve kaliteli yaşam döngüsü sağlayarak, müşteri isteklerini olabildiğince sade fazlar ve hızlı geri dönüşler ile mümkün kılmayı amaçlar. Yazılım geliştirmede uygulanan çevik metodolojiler, esnek bir güç oluşturmayı, hızlı hareket etmeyi ve zamanla oluşan değişikliklere kolay adapte olmayı hedeflemektedir. Çevik yazılım geliştirme metodolojisinin en temel özelliği, dağıtılabilir birimler ile başlayıp, zaman içerisinde ürünün tamamen işlevsel ve ölçeklenebilir birimlerden oluşturulmasını sağlamasıdır. Çevik metodolojisinin döngüsel yapısı, geliştirilecek yazılımın kısa zamanda teslim edilebilir olmasını ve periyotlar halinde geliştirilmelerin tamamlanmasını sağlar.

CHAOS 2015 raporu sonuçlarına göre, şelale (waterfall) ve çevik (agile) yazılım projeleri başarı oranlarının karşılaştırma tablosu Tablo 2'deki gibi su-

nulmuştur. Tablo 2'de de görüldüğü üzere, çevik projeler her proje büyüklüğüne göre geleneksel proje yöntemlerinden daha başarılıdır [2].

Tablo 2. CHAOS 2015 Raporu Çevik ve Şelale Yazılım Geliştirme Metodolojilerinin Yazılım Projelerinin Başarı Oranlarına Etkisinin Karşılaştırma Tablosu [2]

Proje Büyüklüğü	Metodoloji	Başarılı	Zorlanılan	Başarısız
Toplam Projeler	Çevik	%39	%52	%9
	Şelale	%11	%60	%29
Büyük Ölçekli Projeler	Çevik	%18	%59	%23
	Şelale	%3	%55	%42
Orta Ölçekli Projeler	Çevik	%27	%62	%11
	Şelale	%7	%68	%25
Küçük Ölçekli Projeler	Çevik	%58	%38	%4
	Şelale	%44	%45	%11

Yazılım geliştirme işlerindeki faktörler, yazılım mühendisliği ile ilişkili olduğu kadar iş ve proje yönetimi metodolojilerinin de birleşimidir. 3.partilerle yazılım geliştirme genel olarak yazılım geliştirme aktivitelerinin tedarikçiden yazılım geliştiricilerin sağlanması olarak tanımlanabilir. Gelişen internet tabanlı işbirliği servisleri sayesinde Hindistan, Çin, Latin Amerika gibi ülkelerdeki yazılım geliştirme görevleri için tedarikçilerin kullanımı da artmaktadır [3]. Takımlar farklı kültürel altyapılara sahip, dünyanın farklı lokasyonlarında bulunan takım üyelerinden oluşmaktadır. Büyük ölçekli uluslararası projelerde 3. partilerle çalışmanın yeni iş fırsatları oluşturma, pazara daha kolay giriş yapma gibi avantajlarının yanında bazı zorlukları da mevcuttur. Ekip kontrolü, koordinasyon, teknoloji ve insanların birbirleri ile etkileşimi farklı kültürel yapılara sahip, farklı dilleri konuşan, farklı çalışma prensipleri olan insanların bir arada çalışmalarını sağlamak büyük ölçekli, uluslararası projelerde 3. partilerle çalışmanın güçlüklerinden bazıları olarak sayılabilir.

CMMI-Dev v1.3 seviye 3 sertifikasyonuna sahip olan Netaş, yıllardır uluslararası projeler geliştirerek müşterilerine kaliteli yazılım projeleri sunmaktadır. Netaş, uluslararası, çoklu bileşenli, 3.partilerle ortaklaşa geliştirdiği yazılım projelerinde de çevik yazılım metodolojilerini uygulamaktadır.

Bu bildiride, uluslararası çoklu bileşenli çevik yazılım projelerinde 3.partilerle çalışmanın avantaj ve dezavantajları ele alınmış, çevik yazılım geliştirme süreçlerindeki deneyimler sorunlara getirilen çözümler ile paylaşılmıştır. 2. kısımda Scrum modelinden ve Scrum of Scrum pratiğinden bahsedilmiş, Netaş projelerinden bir örnek uygulama üzerinde açıklanmıştır. 3. kısımda söz konusu olan örnek uygulama sırasında karşılaşılan zorluklardan, 4. kısımda ise bu zorluklar karşısında çevik yazılım geliştirme metodolojilerinden faydalanılarak geliştirilen çözüm yöntemleri açıklanmıştır. 5. kısımda, proje kapsamında çıkarılan sonuçlar değerlendirilmiştir.

2 Uluslararası, Çoklu Bileşenli Çevik Yazılım Geliştirme Projeleri

Çoklu bileşenlerden oluşan sistemler, modüler yapısının yanında geliştirilen modüllerin birbirleri ile uyumlu çalışmasını hedeflemektedir. Bu bildiride konu edilen projede, birbirinden bağımsız çalışan yazılım geliştirmeleri ayrı ürünlerde ayrı fonksiyonellikler sağlamasının yanı sıra, birbirleri ile bağımlı çalışarak teslim edilebilir yeni bir ürün ortaya koyabilmektedir.

Netaş, çevik proje yönetimini Scrum modeli ile uygulamaktadır. Scrum, yazılım geliştiren proje takımlarının; enerjilerini, odak noktalarını, açık ve şeffaf bir şekilde kattıkları bir çevik yazılım geliştirme sürecidir. Scrum modelinde, scrum takımları kendi kendini organize edebilen, girişken takımlardır. Scrum takımları, projedeki teslim mekanizmalarını sprint(koşu) adı verilen kısa süreli (1-4 hafta) süreçler ile sağlarlar. Her sprint süreci, tahminler ve içerik planlaması ile başlar, teslim ve her oturum sonunda çıkarılan sonuçlar ile sonlanır. Ürün iş listesi (backlog), scrum master ve ürün sahibi tarafından tasarım tahminleri ile önemli noktaların ve önceliklendirilmiş görevlerin belirlenmesi gerekmektedir. Günlük scrum toplantıları yapılır. Bu yapı, scrum takımlarını birbirine yakın, hızlı iş geliştirebilen, verimli takımlar haline dönüştürür [4].

Scrum Alliance Grup tarafından, Scrum modelinin pratiklerinden olan Scrum of Scrum en iyi pratiklerden biri olarak önerilmiştir [5]. Bu model, takımlar arası birçok bağlılığı aradan kaldırarak, izole Scrum takımları arasında bölüştürerek çalışır. Scrum masterları (takım liderleri-proje yöneticileri) düzenli olarak lokasyonlar arası buluşur. Bu sayede, Scrum takımları Scrum of Scrum pratiği ile birbirine bağlanır. Bu pratik takımlar arası iletişimi, iş bölümünü ve verimliliği artırır. Bunun yanında, çevik yazılım geliştirme sürecine yeni dahil olanlar için uygun hale getirir.

Bu bildiriye konu olan proje bulut bilişim teknolojileri ile uyumlu, donanım maliyeti düşük uygulama sunucusu ve çoklu ortam video konferans sunucusu geliştirmeleri ile kullanıcı masaüstü ve Web istemcilerinin geliştirilmelerini içermektedir. Büyük ölçekli, çoklu bileşenli bu proje kapsamında geliştirilen sunucu ve istemcilerin entegrasyonu ve testleri de gerçeklenmiştir. Söz konusu proje, son kullanıcıların günlük hayatta sıkça kullandıkları sesli, görüntülü ve yazılı iletişimi bulut bilişim teknolojileri ile sağlamaktadır.

Bu projenin yönetiminde Scrum modeli kullanılmıştır. Bu Scrum takımları, Scrum of Scrum pratiği ile birleştirilmiştir ve scrum masterlar tarafından raporlama ve iş takibi sağlanmaktadır. Scrum takımları aşağıda ayrıntılı bir şekilde açıklanmıştır.

Oluşturulan proje yapısında toplam 5 Scrum takımı oluşturulmuştur. Bu Scrum takımları Şekil 1'de gösterildiği gibidir.

1. Scrum – Sunucu Geliştirmeleri: NETAŞ'ın kendi bünyesinde oluşturulmuş olan ve sunucu üzerindeki geliştirmeleri yapan takımdır. Takımın tamamı NETAŞ mühendislerinden oluşmakta ve aynı lokasyonda çalışmaktadır.

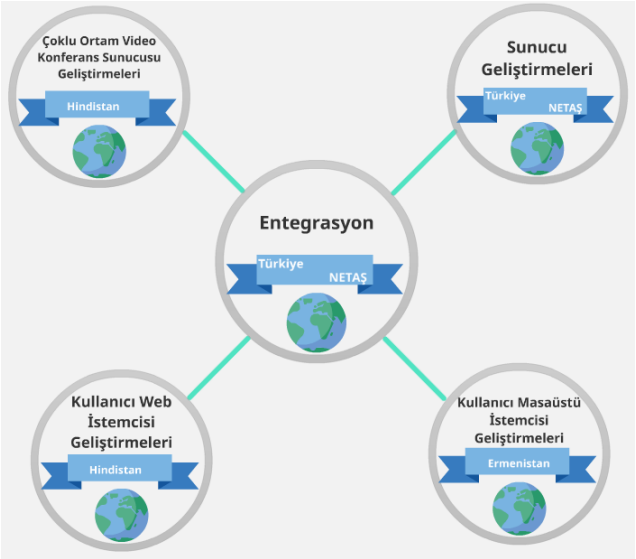
2. Scrum - Kullanıcı Masaüstü İstemcisi Geliştirmeleri: NETAŞ'ın hali hazırda diğer projelerinde de birlikte çalıştığı Ermenistan'da bulunan alt

yüklenici firma tarafındaki takımdır. Bu takım masaüstü istemcisi üzerindeki geliştirmeleri yapmaktadır.

3. Scrum – Kullanıcı Web İstemcisi Geliştirmeleri: Kullanıcıya tarayıcı üzerinden sesli ve görüntülü konferansa katılma imkanı sağlayacak olan istemci geliştirmelerini kapsar. Hindistan'da bulunan bir yüklenici firma tarafından yapılmaktadır.

4. Scrum – Çoklu Ortam Video Konferans Sunucusu Geliştirmeleri: Video konferans sunucusunun farklı istemcilerin isteklerine cevap verebilecek şekilde geliştirilmesini kapsar. Hindistan'da bulunan başka bir yüklenici firma tarafından yapılmaktadır.

5. Scrum – Entegrasyon: Uçtan uca çözüm fonksiyon ve kalite testleri ile birlikte farklı bileşenlerin entegrasyonundan ve projenin ürüne dönüşüm sürecinden sorumludur. NETAŞ bünyesindeki ürün doğrulama ve sistem mühendisleri ve proje mimarlarından oluşmaktadır.



Şekil 1. Uluslararası, Büyük Ölçekli, Çoklu Bileşenli Projelerde Bir Scrum Modeli Örneği

3 Uluslararası, Çoklu Bileşenli Çevik Yazılım Geliştirme Projelerinde 3. Partilerle Çalışma Sırasında Karşılaşılan Zorluklar

Bu bildiriye söz konusu olan projede 3. partilerle çalışma alanlarında karşılaşılan zorluklar ve etkilenen başarı faktörleri şu başlıklar altında incelenebilir.

a. Takım Yapısı ve Dağılımı: Projenin ihtiyaçları doğrultusunda oluşturulan Scrum modeli hem coğrafi olarak farklı lokasyonlarda bulunan 4-8 kişiden oluşan 5 farklı takımı içermektedir hem de farklı ülkelerde yer alması nedeniyle kültürel ve sosyo ekonomik olarak farklı etkenlerden etkilenebilmektedir. Uluslararası bir proje olması projenin hem planlanmasını, hem kontrol edilebilmesini hem de takımlar arası bağımlılıkların ve birleşme noktalarının giderilmesini zorlaştırmaktadır.

b. Proje Planları: Projenin dağıtık ve uluslararası yapısı nedeniyle proje planları yapılırken çeşitli noktalar göz önünde bulundurulmuştur.

- Bir takımın teslim edeceği çıktı bir başka takım için girdi olduğu durumlar iş listelerinde öncelikli planlanarak proje için toplam zaman ve kaynak faydasının arttırılması planlanmıştır.
- Proje planı oluşturulurken, etkileşim ve bütünleşme noktaları belirlenmiş, bu noktaların olabildiğince erken doğrulanabilmeleri için de çözüm doğrulama döngüleri sürekli ve geliştirme döngülerine paralel olacak şekilde tasarlanmıştır.
- Takımların çevik yazılım geliştirme prensiplerine uyması, 3 ya da 4 er haftalık devinimlerle teslimat yapmaları ve kaliteden sorumlulukları planlanmıştır.
- Projenin uluslararası yapısı nedeniyle farklı ülkelerin tatil, bayram ya da izin dönemleri proje planlarına yansıtılmıştır.

c. İnsan Faktörü ve İletişim: Takımların teknik yeterliliği ve tecrübesi yanında takım olarak yakalanan sinerji, güven, açık iletişim, işbirliği ve yardımlaşma projenin başarıya ulaşması için önemli faktörlerden biridir. Projenin yapısı ve de süresi göz önüne alındığında projenin bütününde çalışan insan sayısının artması da insanların motivasyonunu etkileyen bir etken olarak öne çıkmaktadır. Projenin uluslararası yapısı nedeniyle video konferans ile günlük scrumlar gerçeklenerek iletişim kuvvetlendirilmiştir.

d. Teknik Faktörler: Projenin tanımı ve yapısı gereği teknik olarak kurulan altyapı ve entegrasyon ihtiyacı öne çıkmaktadır. Bu entegrasyonu proje planları dahilinde planlanan sürelerde karşılamak için projede otomasyon ihtiyacı öne çıkmaktadır.

e. Organizasyonel Faktörler: Kurulan scrum takımlarının farklı şirketler bünyesinde yer alması nedeniyle yönetim farklılıkları doğmaktadır. Scrum Masterların scrum of scrum toplantılarında karşılaşılan sorunlara karşı yaklaşım farklılıkları özellikle iletişim noktasında dikkatli olmayı ve proje raporlamalarında ve durum bildirim özetlerinde proje durumunu net ve tarafsız olarak bildirmeyi gerekmektedir.

4 Karşılaşılan Zorluklar Karşısında Kullanılan Yöntemler ve Çevik Yazılım Geliştirmedeki Karşılıkları

Karşılaşılan her bir zorluk karşısında farklı çözüm teknikleri uygulanmış ve projenin başarı ile sonuçlanması için gereken çözümler sağlanmaya çalışılmıştır. Çevik yazılım geliştirme prensipleri ile örtüşen bu metodoloji ve yaklaşımlar Tablo 3 ve Tablo 4'te listelenmiş ve ayrıntıları verilmiştir.

Tablo 3. Karşılaşılan Zorluklar ve Çözüm Yöntemleri

Zorluklar	Alınan Aksiyonlar	Karşılık Gelen Çevik Yazılım Geliştirme Prensibi
Takım Yapısı ve Dağılımı	• Takımların rolleri ve iş listeleri mümkün olduğu kadar ayrıntılı olarak ve erken fazda belirlenmeye çalışılmıştır. (Backlog planlaması) • Bir takımın teslim edeceği çıktı bir başka takım için girdi olduğu durumlar iş listelerinde öncelikli planlanmıştır (Backlog Prioritization) • Takımların teslimat ve kalite sorumlulukları net olarak belirlenmiştir. • Projenin çeşitli aşamalarında çalıştaylar düzenlenmiş (workshop) ve takımların bir araya gelerek birlikte çalışması sağlanmıştır. • Dönemsel ihtiyaçlara göre NETAŞ bünyesinde (on-site) çalışmalar ayarlanmıştır.	• Kendi içerisinde organize takımlar • Yazılımın kısa aralıklarla teslimi (aylar yerine haftalar) • Geliştiriciler ve Scrum Masterların birebir iletişim halinde, birlikte çalışması
Proje Planları	• Kolektif proje planları yapılmış ve adaptasyona izin verecek şekilde entegrasyon noktaları belirlenmiştir. • Ayrı bir scrum takımı entegrasyon ve uçtan uca çözümden sorumlu tutulmuştur • Proje zamanlaması içinde farklı sürüm planları yapılmış ve tamamen ürünleşmeye geçmeden önce alfa, beta gibi farklı sürümler müşterinin kullanımına açılmıştır.	• Plan çerçevesinde değişikliklere gidilmesi, • Değişen durumlara düzenli adaptasyonun sağlanması • Sürekli ve kaliteli yazılım teslimi ile müşteri memnuniyeti sağlanması
İnsan Faktörü ve İletişim	• Projenin çeşitli aşamalarında çalıştaylar düzenlenmiş (workshop) ve takımların bir araya gelerek birlikte çalışması sağlanmıştır. • Dönem dönem yerinde (on-site) çalışma olanakları sağlanmış ve takımların farklı lokasyonlarda buluşması sağlanmıştır. • Takım oluşturma aktiviteleri ile takım sinerjisi oluşturulmaya çalışılmıştır.	• Yüz yüze görüşmelerle daha iyi iletişim kurulması (co-location) • Projelerin birbirine güvenen motive bireyler tarafından geliştirilmesi

Tablo 4. Karşılaşılan Zorluklar ve Çözüm Yöntemleri

Zorluklar	Alınan Aksiyonlar	Karşılık Gelen Çevik Yazılım Geliştirme Prensibi
Teknik Faktörler	• Sistemde farklı takımlar tarafından sağlanan yazılım birimlerini birleştirmek üzere öne çıkan otomasyon ihtiyacına uygun araçlar kullanılarak doğrulama testleri tamamen otomatize edilmiştir.	• Teknik olarak iyi tasarımların çıkarılmasına sürekli önem verilmesi
	• Tasarımın birçok bileşen tarafından değiştirilebilir olması nedeniyle sürekli teslim altyapısı hızlı ve etkili bir şekilde kullanılmıştır. Bu sayede kalite ve tasarım hedefleri düzenli bir şekilde takip edilebilmiştir.	• Geliştirilen yazılım ilerlemenin başlıca ölçüsüdür
	• Sunucu tarafındaki kodlar mümkün olan en erken aşamada dondurulmuş ve çok acil durumlar dışında değişiklikten kaçınılmıştır.	• Sürekli ve kaliteli yazılım teslimi ile müşteri memnuniyeti sağlanması
Organizasyonel Faktörler	• Projenin çeşitli aşamalarında çalıştaylar düzenlenmiş (workshop) ve takımların bir araya gelerek birlikte çalışması sağlanmıştır.	• Geliştiriciler ve Scrum Masterların birebir iletişim halinde, birlikte çalışması
	• Dönemsel ihtiyaçlara göre değişen haftalık ya da günlük senkronizasyon toplantıları yapılmıştır • Açık İletişim yöntemi ile güven ve yardımlaşma olgusu oluşturulmuştur.	• Yüz yüze görüşmelerle daha iyi iletişim kurulması (co-location)

5 Sonuçlar

Netaş Telekomünikasyon A.Ş., çevik yazılım geliştirme süreçlerini projelerinde uygulamaktadır. Ar-Ge projelerinde oluşan süreç, kısa zamanda değişebilen, modüler, hızlı ve esnek proje geliştirme yöntemlerini gerektirmektedir. Bu projenin gelişiminde, şelale yazılım geliştirme ile çevik yazılım geliştirme metodolojisi arasında seçim yaparken göz önünde bulundurulan en önemli etkenlerinden birkaçı şu şekilde sıralanabilir.

- Çoklu bileşenden oluşan bu projenin bileşenlerine ayrılarak yüklenici firmalar da dahil olmak üzere uzmanlık alanlarına göre paylaşılması,

- Bileşenlerin paralel olarak yazılım geliştirmelerinin sağlanması ve modüler bir yapıya sahip olması,

- Her bileşenin proje geliştirilirken testlerinin sağlanıyor olması,

- Farklı firmalar tarafından gerçeklenen her parçanın entegrasyon ihtiyacı ve bu entegrasyonda gereken süreçlerin kısa zaman aralıklarında tamamlanması,

- Süreçlerin değişen ihtiyaçları kısa zaman içerisinde karşılayabilecek şekilde yöntemler sunması ve bu değişikliklerin projenin diğer bileşenleri üzerindeki etkilerini planlayabilmeyi sağlaması gerekmektedir.

Yukarıda bahsedilen bu etkenler doğrultusunda çevik yazılım geliştirme süreçlerinin bildiride sözü geçen proje için uygun olduğu görülmüştür. Çevik yazılım geliştirme süreçleri yerine, şelale yazılım geliştirme süreçleri uygulansaydı, projenin kullanıcı arayüzündeki belirsizliklerin projeye başlanmadan karar verilip tasarımlarının oluşturulması projeye başlangıç zamanını uzatabilirdi. Projenin başlangıcında tasarımı yapılmış yapının projenin ilerleyen safhalarında değişmesi gerekseydi projede değiştirilecek kısımlar daha maliyetli ve yine süreci uzatacak şekilde olabilirdi. Uluslararası ve çoklu bileşenden oluşan bu proje şelale yazılım geliştirme metodolojisi ile geliştirilseydi, dinamik olarak müşteri isteğine veya proje performansına göre değişmesi gereken kısımların değiştirilmesi zaman ve yöntem açısından verimsiz olabilirdi. Bileşenlerin ilerleyen safhalarında ilk başta geliştirilmiş bileşenin yapısının sonrasında geliştirilen yapıya uygun olmaması değişimi ve süreci zorlayıcı kılabilirdi. Projenin çoklu bileşenden oluşması ve uluslararası nedeniyle, projedeki iletişim, yardımlaşma ve işbirliği ihtiyacı çevik yazılım geliştirme süreci ile daha kolay karşılanabilirdi.

Özellikle çevik yazılım geliştirme süreci ile birlikte kullanılan sürekli teslim altyapısı projede kilit bir rol oynamaktadır. Proje bileşenlerinin çoklu olması nedeniyle yapılan herhangi bir değişikliğe bağlı olarak karşılaşılan problemlerin adreslenmesinde otomasyon altyapısından faydalanılmıştır.İlk kalite testleri otomatik birim testler ile sağlanmıştır. Kalite testlerini geçen yazılımlar için otomatik test senaryoları oluşturulmuştur. Otomatik test senaryoları fonksiyonel testlerdir. Fonksiyonel testler kullanılarak sık kod girişi olduğunda geliştiriciye hatalarla ilgili hızlı geri dönüş sağlanması ve düzeltilmesi için Jenkins sürekli entegrasyon sunucusu kullanılmıştır. Bu fonksiyonel testler sayesinde otomatik derleme yapılmış ve hatasız olması koşuluyla müşteriye verilecek dosyalar otomatik üretilmiştir.

Bu ve benzer projelerde kazanılan deneyimlere dayanarak bu çalışmaya konu olan proje gibi çok partili projelerde çevik yazılım geliştirme metodolojilerinin fayda sağladığı ve projenin başarılı olmasında etkili olduğu gözlemlenmiştir.

Dağıtık yapılı ve birbirine çeşitli açılardan bağımlı olan projelerde takımların ihtiyaçları, bağımlılıkları, üretimleri ve dolayısıyla planları sıklıkla değişebilmekte ve çevik prensipler bu değişimlerin planlar dahilinde ele alınabilmesini ve yönetilebilmesini sağlamaktadır.

Müşteriye yapılan sürekli ve hızlı teslimat prensibi nedeniyle otomasyon ihtiyaçları ayrıca öne çıkmakta ve proje başlangıcından itibaren takımların hepsinde benzer otomasyon altyapılarının kurulmuş ve fonksiyonel olması gerekmektedir.

Ayrıca proje planlarında takımların entegrasyon ihtiyaçlarından, teknik bağımlılıklarına, sorumluluk tanımlamalarından, lokasyon farklılıklarından ileri gelen tatil planlamalarına kadar çok çeşitli faktörlerin göz önüne alınması ve planlanması gerektiği deneyimlenmiştir.

Bu ve benzeri birçok nokta göz önüne alındığında çevik yazılım geliştirme metodolojisinin, şelale yazılım geliştirme metodolojisine göre daha uygun, esnek ve hızlı planlar yapabilmeye olanak tanıdığı deneyimlenmiştir ve şirketimiz tarafından uluslararası, çoklu partili projelerde yoğunlukla kullanılmaktadır.

Kaynaklar

1. The Standish Group, `http://www.standishgroup.com/`
2. Standish Group 2015 Chaos Report, `https://www.infoq.com/articles/standish-chaos-2015`
3. Buslovic, M., Deribe, S.: A Multiple Case Study on Contradictions and Preconditions for Outsourcing Agile Software Development Projects. Linköping University (2012)
4. Ambler, S.: Scaling Agile: An Executive Guide. White paper, IBM Agility at Scale (2010)
5. Scrum Alliance, `https://www.scrumalliance.org/`