CMMI-DEV Seviye-3 Sertifikasyonuna Sahip Bir Organizasyonda SCRUM Çevik Yazılım Geliştirme Yöntemi'nin Yazılım Geliştirme Çalışmalarında Uygulanması

Esra Şahin[1], İlgi Keskin Kaynak[1], Hakime Koç[2]

[1] Program ve Tasarım Kalite Güvencesi Müdürlüğü, Kalite Yönetim Dir., REHİS Grup Bşk.
ASELSAN A.Ş.

{esrasahin, ikkaynak}@aselsan.com.tr
[2] Görev Yazılımları Müdürlüğü, Tasarım Teknolojileri Dir., REHİS Grup Bşk.
ASELSAN A.Ş.

unsal@aselsan.com.tr

Özet. Çevik yazılım geliştirme yöntemlerinden biri olan SCRUM, yazılım geliştirme faaliyetlerinin yönetilmesi açısından sunduğu esnek yaklaşım ile yazılım yönetimi ve geliştirilmesinde fayda sağlayan bir yöntemdir. CMMI-DEV ise tasarım ve geliştirme yapan firmalar için yol gösteren bir yetenek olgunluk modeli olup, herhangi bir geliştirme yöntemi dayatmadan organizasyonlara proje yönetimi, mühendislik (tasarım ve geliştirme), süreç yönetimi ve destek süreçlerini değerlendirerek performanslarını iyileştirmeleri için kapsamlı bir altyapı sunmaktadır. Yazılım sektöründe SCRUM ve CMMI-DEV'ın birlikte kullanılmasına yönelik araştırma yapıldığında, yorumlama farkından dolayı bu iki kavramın zıt kutuplarda değerlendirilebildiği gözlenmektedir. Oysa ki SCRUM, yazılım geliştirme ve yönetimi alanlarında sağladığı yaklaşım ile CMMI-DEV uygulamalarının gerçekleştirilmesine engel oluşturmamaktadır. Bu çalışmanın amacı SCRUM'ın, CMMI-DEV ile uyumlu olarak kullanılabildiğinin, örnek bir uygulama ile anlatılması ve bu kullanımın sağladığı faydaların değerlendirilmesidir. Ürün ve Sistem Tasarımı, Geliştirmesi ve Platform Entegrasyonu disiplinlerinde CMMI-DEV v1.3 Seviye-3 sertifikasyonuna sahip olan ASELSAN REHİS (Radar, Elektronik Harp ve İstihbarat Sistemleri) Grup Başkanlığı bünyesinde yürütülen bir proje, örnek uygulama kapsamında pilot proje olarak seçilmiştir. Söz konusu projede CMMI-DEV v1.3 Seviye-3 uygulamaları gerçekleştirilirken, gömülü ve kullanıcı arayüzü yazılım konfigürasyon birimleri geliştirmede SCRUM yöntemi uygulanmıştır.

Anahtar Kelimeler. CMMI-DEV, SCRUM, Çevik Yazılım Geliştirme

1 Giriş

Yazılım geliştirme süreci işlerken gereksinimlerin değişiklik göstermesi durumu zaman, maliyet, iş gücü açısından kayıplara neden olmaktadır. Bu değişimleri projenin erken safhalarında fark edebilmek, daha kısa sürede ve daha az maliyetle karşılayabilmek için geleneksel yazılım geliştirme yöntemlerine alternatif olarak çevik yazılım geliştirme yöntemleri ortaya çıkmıştır. Çevik yaklaşım, kişiler arasındaki etkileşimi arttırmayı, çalışan yazılıma odaklanmayı, müşteriyle iş birliği yapmayı ve değişiklik isteklerini karşılamayı öngörür. Çevik yazılım geliştirme yöntemleri kendine özgü prensipleri olan SCRUM, Dynamic Systems Development Methodology (DSDM), Adaptive Software Development (ASD) ve Extreme Programming (XP) gibi alt yöntemlere ayrılmaktadır [1].

Bu çalışma kapsamında, SCRUM'ın CMMI-DEV modeli ile uyumluluğu incelenecektir. Bu inceleme için de CMMI-DEV v1.3 Seviye 3 sertifikasyonuna sahip olan ASELSAN REHİS Grup Başkanlığı'nda yürütülen bir proje, pilot uygulama olarak değerlendirilecektir.

Makalenin organizasyonu şu şekilde özetlenebilir; Giriş Bölümü'nde çalışmanın kapsamına ilişkin bilgi verilmiştir. Girişin ardından gelen Çalışmaya İlişkin Altyapı Bölümü'nde CMMI-DEV, SCRUM ve bu iki kavramın birlikte kullanılabilmesi ile ilgili temel bilgiler anlatılmıştır. Verilen bilgi altyapısını takiben Örnek Uygulama Bölümü'nde SCRUM uygulamasının detayları bir pilot uygulama üzerinden anlatılmıştır. Değerlendirme Bölümü'nde ise pilot uygulamanın değerlendirmesi sunulan bilgi altyapısı üzerinden yapılmış ve Sonuç Bölümü'nde bu çalışmadan elde edilen sonuçlar kaydedilmiştir.

Yazılım geliştirme dünyasında Çevik Yöntemler ile CMMI arasında kutuplaşma oluştuğu gözlenmektedir. CMMI'ın ilk olarak uygulamaya geçirildiği kullanım alanları büyük ölçekli, hata toleransı düşük, emniyet kritik sistemler ve hiyerarşik düzen içeren uygulamalar olmuştur. Çevik Yöntemler ise ilk olarak nispeten daha küçük, değişebilen gereksinimlerden, tek bir ekipten oluşan ve sadece yazılım boyutu olan uygulamalardan oluşmuştur.

CMMI ve Çevik Yöntemler'in ilk uygulama alanlarındaki farklılıklar, daha sonra oluşan yorumlama farklılıklarının temelini teşkil etmektedir. Aşağıdaki iki faktör bu algının oluşmasında etkendir:

- **Yorumlama/Uygulama Hataları.** CMMI çoğu zaman yanlış anlaşılabildiği üzere bir "standart" değil, ürün kalitesini ve süreç performansını iyileştirmek için kullanılan bir "model"dir. CMMI olgunluk seviyesine sahip olmak isteyen bir organizasyonda SCAMPI (Standard CMMI Appraisal Method for Process Improvement) değerlendirmelerinden başarıyla geçilmesi gerekli olup yeterli olmamaktadır. SCAMPI değerlendirmelerinin başarıyla tamamlanmasının yanı sıra ürün kalitesini ve süreç performansını iyileştirmeye yönelik sunulan altyapı kullanılarak CMMI'da beklenen uygulamaların gerçekleştirilmesi esas olmalıdır. Bunun yanı

sıra SCAMPI değerlendirmeleri tetkik ya da denetim değil, CMMI modelini temel alarak organizasyonun geliştirme uygulamalarını gerçekleştirdiğinin, bu uygulamaların kurumsallaştığının ve iyileştirildiğinin incelendiği bir değerlendirme sürecidir. CMMI süreçlerden değil, süreç alanlarından oluşan bir modeldir. Süreç alanları belli hedeflere ulaşmak için beklenen uygulamalardan oluşur. Süreç alanları kapsamındaki süreçlerin nerede, ne zaman ve hangi sıra ile uygulanacağı ise yapılan işe göre şekillenir. CMMI süreç alanları, süreç olarak algılandığında, bu durum zaman ve efor kaybına neden olur.

- **Bilgi Eksikliği.** Yazılım sektöründe CMMI'ı veya Çevik Yöntemler'i benimseyen toplulukların her ikisinde de bir diğeri ile ilgili bilgi eksikliği olduğu SEI (Software Engineering Institute) tarafından değerlendirilmiş ve bu konuda 2005 yılında NDIA (National Defense Industrial Association) tarafından CMMI "Technology Conference and User Group" Konferansı için, SEI tarafından ise SEPG (Software Engineering Process Group) 2006 için Çevik ve Yalın Yazılım Geliştirme ile ilgili çalışmalar yapılmıştır [2]. CMMI ve Çevik Yöntemler'in birlikte kullanılabilmesine yönelik yayımların 2005 yılı itibariyle arttığı görülmektedir[2]. Örneğin; 2005 yılında CMMI Seviye-3 ile Çevik Yöntemler'in birlikte kullanılabileceğini anlatan bir çalışma [6] yapıldığı, 2007 yılında gerçekleşen Agile konferansı için hazırlanan bir çalışmada [11] ise; CMMI Seviye-5 sertifikasyonuna sahip olan bir kurumda SCRUM'ın başarıyla uygulanmasının aktarıldığı saptanmıştır[2].

2 Çalışmaya İlişkin Altyapı

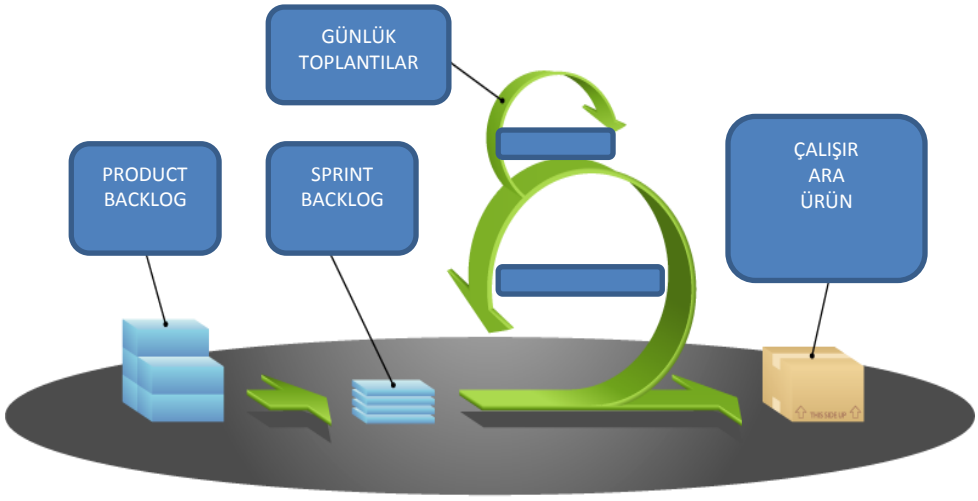
Bu bölümde SCRUM yöntemi ile ilgili bilgi altyapısı verilmiştir. Bu kapsamda öncelikle SCRUM'ın tanımı yapılmış, daha sonra SCRUM'ın temelini oluşturan roller, toplantılar ve kavramlar açıklanmıştır. Son olarak pilot çalışmanın değerlendirmesine temel oluşturmak üzere SCRUM ve CMMI-DEV ilişkisi anlatılmıştır.

2.1 SCRUM Yöntemi

SCRUM; belirli zaman dilimleri halinde ilerleyen, kısa döngülerle çıktı üretme ve geri bildirim düşüncesine dayanan, esnek ve prensipleri olan yazılım geliştirme metodlarından biridir. Yazılımın nasıl geliştirileceğinden çok nasıl yönetileceği ile ilgilenir. Bu yöntem, küçük çaplı projelerde yönetimi kolaylaştırırken, büyük çaplı projelerde ekipleri küçük parçalara ayırmaya yönlendirir ve sürekli gözden geçirmeleri gerektirir. Bu gözden geçirmeler sayesinde ekip işleyişi net olarak görülür ve bu da iş birliğini ve ekip içindeki iletişimi artırır. Aynı şekilde ekip müşteri ile de sürekli iletişim içerisindedir. Bu durum çalışan ürünün müşterinin isteklerine göre en kısa zamanda geliştirilmesini sağlar. [3]

SCRUM yöntemiyle yazılım geliştirme iteratif ve artırımlı olarak yürütülür. Her iterasyona "koşu" adı verilir. Sürecin temel girdisi "ürün gereksinim seti" (product backlog) denilen gereksinim setidir. Koşu toplantılarında her koşuda gerçeklenecek gereksinim seti planlanır ve bu setlere "koşu gereksinim seti" (sprint backlog) adı verilir. Her koşu adımında o koşu adımına giren koşu gereksinim setleri tasarlanır,

gerçeklenir ve test edilir. Koşu sonucunda oluşan çalışır ara ürün müşteri (iş/ürün sahibi) ile birlikte test edilerek döngü yeniden başlatılır ve tüm süreç adımları yeniden uygulanır, bknz. Şekil 1. Bir koşu adımı tamamen tamamlanmadan ikinci bir koşuya başlanılmaz. [4]



Şekil 1. SCRUM Yöntemi

SCRUM'ın temelinde Roller, Toplantılar ve Kavramlar yer alır.

2.2 SCRUM Yönteminde Tanımlı Roller

- **İş/Ürün Sahibi.** Müşteri tarafından görevlendirilen, ortaya çıkacak üründen ne beklenildiğini, ürünün kapsamının ve içeriğinin ne olduğunu belirten kişidir. Gereksinimlerin önceliklendirilmesine karar verebilir.
- **SCRUM Yöneticisi.** Ekibin SCRUM kurallarına bağlı kalmasını sağlayan, iş/ürün sahibine karşı üründen sorumlu olan ve ekip içi iletişimi sağlayan kişidir.
- **Geliştirme Ekibi.** Devamlı iletişim halinde olan ve tek bir hedefe ulaşmayı amaçlayan 5-9 kişiden oluşan topluluktur. Ekip üyeleri her koşu başlangıcında kendilerine bir hedef seçerek belirledikleri koşu hedefine ulaşmak için proje kapsamında çalışan kişilerdir. [1][4]

2.3 SCRUM Yönteminde Tanımlı Toplantılar

- **Planlama.** Bu toplantılar her koşu öncesinde , koşuları ve bu koşularda hedeflenen noktaları planlamak için yapılır. Ekip sıradaki koşu kapsamında yapılacak işi önceliklere göre belirler. Ürün Sahibi ile birlikte koşu amacı belirlenir.

- **Günlük Toplantılar.** Bu toplantılarda her gün yaklaşık 15 dakika Geliştirme Ekibi ve SCRUM Yöneticisi toplanır. Toplantıların amacı sorun çözmek için değil, koşuların durumunu takip edebilmektir.
- **Koşu Değerlendirme Toplantısı.** Koşuların sonunda yapılır. Geliştirme Ekibi koşu boyunca yaptığı çalışmayı İş/Ürün Sahibi'ne sunar. SCRUM Yöneticisi toplantı başında koşu amacını açıklar ve koşu boyunca oluşan ürünün gösterimini yapar. Toplantıda ürün değerlendirilerek kabul/red kararı verilir ve bir sonraki koşunun ana hatları belirlenir. Koşu Gereksinim Setleri'nde gerekli görülen değişiklikler bu toplantılar sonucunda yapılabilir.
- **Koşu Retrospektif Toplantısı.** Her koşu sonrası kısa bir toplantı ile SCRUM Yöneticisi yönetiminde koşu boyunca nelerin planlamaya uygun olduğunu, nelerin iyileştirilebileceği değerlendirilir ve bir sonraki koşu için öneriler oluşturulur. Bir sonraki koşuda bu öneriler dikkate alınır.[1] [4]

2.4 SCRUM Yöntemi Kavramları

- **Ürün Gereksinim Seti (Product Backlog).** İhtiyaç sahibi tarafından belirlenen, geliştirme ekibinin erişimine açık olan ürün gereksinimlerinin listesidir. Bu gereksinimler arasında ürün sahibi tarafından önceliklendirmeler yapılır ve öncelik sırasına göre gerçeklemeler yapılır.
- **Koşu Gereksinim Seti (Sprint Backlog).** Ürün gereksinim seti, koşular için bölünerek koşu gereksinim setleri oluşturulur. Proje ekibi bir sonraki koşu başlangıcına kadar o koşu dahilinde sadece ilgili Koşu Gereksinim Seti'ne odaklanır. Gereksinimlerden alt gereksinimler ya da işler (task) türetilebilir. Bu işlem ekip tarafından yapılır. İş için planlanan süre ve her işin gerçekleşme süresi koşu gereksinim setine günlük olarak işlenmelidir. Ekip üyeleri gereksinimleri seçerek paylaşırlar. Ürün Gereksinim Seti üzerinde yapılan değişiklikler Koşu Gereksinim Setleri'ne yansıtılır ve koşu gereksinim setleri koşular esnasında değiştirilmemek üzere dondurulur.
- **Koşu İlerleme Grafiği (Burndown Chart).** Bu grafik, bir koşuda tamamlanan işlerin planlanan işler ile karşılaştırılabilmesini sağlar. Günlük koşu değerlendirme toplantılarında kalan iş değeri (puanı) güncellenerek kaydedilir. Kalan puan kalan süre değildir, işin değeri üzerinden puanlama yapılır. SCRUM iş gücü ölçümü tabanlı bir uygulama değildir, sonuç üretmeye odaklanır.[1][4]

2.5 Çevik Yöntemler ile CMMI-DEV İlişkisi

CMMI-DEV büyük ölçekli projelerdeki işleyişe değinen uygulamalar içerirken, Çevik Yöntemler küçük ekipleri dikkate alarak ortaya konulmuştur. Fakat detaylı olarak incelendiğinde CMMI-DEV ile Çevik Yöntemler'in birlikte uyumlu olarak uygulanabildiği görülmektedir. Şöyle ki; proje seviyesinde değerlendirildiğinde, CMMI-DEV projede "ne" yapıldığına odaklanırken hangi geliştirme yönteminin uygulanması gerektiğine değinmemektedir [6]. Çevik Yöntemler'de ise projede ürünün "nasıl" geliştirildiğine odaklanılır [7]. Bu bakış açısından yola çıkarak Çevik Yöntemler ile CMMI-DEV'in birlikte uygulanabileceğini söyleyebiliriz. CMMI-DEV'in Mühendis-

lik, Risk Yönetimi ve Entegre Proje Yönetimi süreç alanlarında yer alan uygulamaları, şu konularda Çevik Yöntemler'in büyük ölçekli projelerde uygulanmasını kolaylaştırabilmektedir:

- İş ve ürün hedeflerini ortaya çıkarma,
- Gereksinimlerin ekibe atanmasının yönetimi,
- Süreç ve ürün açısından ekipler arası arayüzlerin ve kısıtların tanımlanması ve idame edilmesi,
- Ürün için entegrasyon, doğrulama ve geçerli kılma stratejilerinin belirlenmesi ve takibi,
- Risk yönetimi [2].

Organizasyon seviyesinde CMMI-DEV'in Çevik Yöntemler'le birlikte kullanılmasını gerektiren bir diğer neden ise Çevik Yöntemler'in kurumsal kapsamda tanımlanmamış olmasıdır [5]. Bu noktada CMMI-DEV; süreç tanımlaması, ölçüm, geri besleme, eğitim ve iyileştirme mekanizmaları ile Çevik Yöntemler'i kurumsal seviyede tamamlamaktadır. CMMI-DEV ile Çevik Yöntemler için, farklı bakış açılarına göre aşağıdaki tabloda belirtilen analiz makalenin yazarları tarafından SEI'nin bu konudaki raporu [2] incelenerek yapılmıştır.

Tablo 1. CMMI-DEV ve Çevik Yöntemler

CMMI-DEV	ÇEVİK YÖNTEMLER
Odaklanma ve perspektif	
CMMI-DEV kurumsal perspektife sahiptir. CMMI-DEV'in odak noktası **kurumsal** seviyededir. Ürün geliştirmeye katkı sağlayan tüm yetenek ve fonksiyonlar süreç iyileştirme ile adreslenir ve bu uygulamalar kurumsal seviyede gerçekleştirildiğinde büyük oranda fayda sağlanır.	Çevik Yöntemler proje perspektifine sahiptir. Çevik Yöntemler'in odak noktası **proje ve proje ekibi**dir.
Yönetim	
CMMI-DEV'de proje başarısının elde edilmesinde **yönetim** önemli bir role sahiptir. Planlama, koordinasyon, bağımlılıkların ve arayüzlerin yönetimi, risk yönetimi oldukça önemlidir [8].	Çevik Yöntemler'de Yönetim geleneksel olarak (komuta-kontrol) değil **koçluk** şeklinde yapılır. Ekibin ilerleyebilmesi için koç rolünü üstlenen kişi ekibin önüne çıkan engelleri ortadan kaldırmaya yardımcı olur.
Planlama	
CMMI-DEV'de projenin hedeflerine ulaşılabilmesi için **proje seviyesinde planlama** yapılması için uygun süreç tanımlanır. CMMI-DEV'de detaylı planlama yapılması doğrudan belirtilmemiştir [9].	Çevik Yöntemler'de birden fazla seviyede planlama yapılabilir. Üst seviyede proje planlaması, detay seviyede ise **koşu planlamaları** yapılır. Bu durum, esneklik ve yeniden plan-

	lama yaklaşımını getirir.
Öğrenme	
CMMI-DEV'de öğrenme aşağıdaki yöntemlerle gerçekleşebilir: 1. Kurumsal eğitimler 2. Proje aktiviteleri 3. Süreçlerin, öğrenilen derslerin ve ölçümlerin kurum içinde paylaşılması 4. Süreç performansı analizi	Çevik Yöntemler'de öğrenme, proje ya da koşu seviyesinde ve faaliyetler esnasında gerçekleşir.
Değerlendirme	
SCAMPI yöntemi ile kurumun süreçleri CMMI-DEV uygulamalarına göre incelenerek iş hedeflerine ulaşabilmek için geliştirilen süreçler değerlendirilir.	Çevik Yöntemler'de değerlendirme, sonuçlar ve oluşan ürün üzerinden yapılır (i.e., müşteri memnuniyeti, proje getirileri).
Kişisel Gelişim	
CMMI-DEV'de proje başarısının bireysel başarıya doğrudan bağlı olmasını önlemek için insan ve teknolojiyi dengeleyen bir süreç altyapısı öne sürülmektedir.	Çevik Yöntemler ekip ve birey odaklıdır (i.e., insan sürecin üstündedir). Ekip, işinde iyi olan insanlar seçilerek oluşturulur.
Proje Yaşam Döngüsü Faaliyetleri	
CMMI-DEV ürünü geliştirirken gözden geçirmenin gerekliliğini vurgular. Yüksek maaliyetli ürün hatalarından, testin yanı sıra dokümantasyon, analiz ve gözden geçirme faaliyetleri ile kaçınılır. Doğru ürünün geliştirildiğinden emin olmak için ara ürünler için de geçerli kılma faaliyetleri yapılmalıdır.	Çevik Yöntemler gerektikçe eş zamanlı geliştirme, test koşuları ve eş gözden geçirmeleri destekler. Önemli olan, ürünün çalışmadığı durumların mümkün olduğunca erken aşamada tespit edilmesidir. Gecikme ve hata maaliyetinin düşük olacağı varsayımı ile artımsal teslim yöntemi benimsenir.
Tahminleme	
Tahminleme kalite ve süreç performansını ölçmede kritik olan alt süreçler için istatistiksel yöntemlerle desteklenerek yapılır.	Geliştirme faaliyetlerinin hızlanması için geliştirme fazı koşulara ayrılır, tasarım ve çözümler evrilir; böylece tahmin edilebilir bir geliştirme hızı elde etmeye çalışılır. Burada amaç, koşu sonlarında fark edilen zaman ve takvim kısıtını koşu başında irdeleyip koşu kapsamını sonradan daraltmaya gerek duymamaktır.

3 Örnek Uygulama

ASELSAN REHİS Grup Başkanlığı'nda yürütülen bir projede CMMI-DEV v1.3 Seviye-3 uygulamaları gerçekleştirilirken, gömülü ve Kullanıcı Arayüzü (KA) yazılım konfigürasyon birimleri geliştirilen bir projede SCRUM yöntemi uygulanmıştır. Proje kapsamında bir Elektronik Destek Sistemi ve bu sisteme ait kullanıcı arayüzü ve gömülü yazılımlar geliştirilmiştir. Elektronik Destek (ED) Sistemi, harekât alanında konuşlanmış olan belli bir frekans bandındaki radarların tespiti, parametrelerinin ölçümü, kimliklendirilmesi ve yönünün belirlenmesi için kullanılan bir sistemdir. ED Sistemi yüksek olasılıkla yakalama, yön bulma için yayın bazında analiz, tanımlı bir veri kütüphanesi ile birlikte çalışarak hedef kimliklendirmesi gibi özelliklere sahiptir.

Yazılım Geliştirme Ekibi, 9 kişilik bir ekip olarak çalışmıştır. KA Yazılımları Geliştirme Ekibi 4 kişi, Gömülü Yazılımlar Geliştirme Ekibi 5 kişi olarak ayrılmıştır.

Yazılım Konfigürasyon Birimi gereksinim dokümanları CMMI "gereksinim geliştirme" ve "gereksinim yönetimi" uygulamaları dikkate alınarak ASELSAN REHİS Grubu Süreç ve Yönergeleri'ne göre hazırlanmıştır. Gerekler için öncelikler belirlenmiş ve koşular bu öncelikler temel alınarak planlanmıştır. Her koşu öncesi gerekler incelenerek bir sonraki koşu için gerekiyorsa İş/Ürün Sahibi ya da SCRUM Yöneticisi tarafından ürün gereksinim setinde güncellemeler yapılmıştır. Koşu süreleri 2 hafta olarak planlanmıştır. Proje yönetiminin temsilcisi olarak, Yazılım Ekip Lideri, "İş/Ürün Sahibi" rolünü üstlenmiştir. Yazılım Ekip Lideri ürün gereksinim setinde ürün özelliklerini tanımlamıştır. Bu setteki her gereğin önceliği belirlenmiştir. İş/Ürün Sahibi planlama aşamasını bu şekilde tamamladıktan sonra her koşu sonrası geliştirme ekibinin çıkardığı iş ürününü, beklenen işlevselliği karşılayıp karşılamadığı konusunda test etmiş ve iş ürünü için kabul ya da red kararını vermiştir. Ürün her gösterim sonrası kabul edilmiş ancak eksiklik ya da hatalar bir sonraki koşuya yeni bir madde olarak eklenmiştir.

KA ve Gömülü Yazılım çalışmalarını paralel şekilde yürütecek iki ayrı alt ekip ve iki ayrı SCRUM Yöneticisi belirlenmiştir. SCRUM Yöneticileri KA ve Gömülü Yazılım geliştirme ekiplerinin temsilcisi olarak çalışmışlar ve geliştirme ekibinin koşu hedefine ulaşmasını sağlamaktan sorumlu olmuşlardır. SCRUM Yöneticileri her koşu için planlanan hikayelerin ekip içinde dağılımının yapılmasını sağlamış, koşu gereksinim setini güncel tutmuşlar ve koşu sonunda ürün gösterimlerini planlamak ve hazırlamaktan sorumlu olmuşlardır. Diğer paydaşlardan gereken bilgiler, SCRUM Yöneticisi ya da İş/Ürün Sahibi tarafından sağlanmıştır.

Koşu planlamayı İş/Ürün Sahibi yönlendirmiştir, koşu dosyalarında bir sonraki koşu sayfası gösterimlerden sonra hızlı bir şekilde oluşturulmuştur. Koşu gereksinim setinde her koşunun ayrıntısı kaydedilmiştir. Planlama esnasında, KA ve Gömülü Yazılımlar'a verilecek hikayeler, ekiplerin koordinasyonu ile belirlenmiştir. KA ile Gömülü Yazılımlar'ın arayüzünü etkileyen hikayeler aynı koşu kapsamına alınmıştır. Koşu gerekleri belirlendikten sonra, koşu geliştirme ekibi, hikaye ya da gerekleri koşu gereksinim setlerinde alt hikaye ya da işlevlere bölmüştür. Hikayeler ekip içinde paylaşılmış ve her hikaye puanlanmıştır. Her gün yapılan değerlendirmelerle kalan iş puanı koşu gereksinim seti üzerinde güncellenmiştir.

SCRUM Yöneticileri'nin moderator rolünü üstlendiği 10 – 15 dakikalık günlük toplantılarda ekip üyelerinden "Ne yapıldı, ne yapılacak, sorun var mı?" sorularının kısa cevapları alınmaya çalışılmıştır.

Günlük olarak kimin hangi konu üzerinde çalıştığı koşu gereksinim setinden takip edilebilmiştir. SCRUM uygulamasında tecrübe kazanıldıkça iş dağılımı daha kolay yapılmıştır. İş dağılımı sonrası kişiler hangi işleri önce yapacaklarına kendileri karar vermişlerdir. Koşu değerlendirme toplantısı öncesinde, koşu çalışmaları kapsamında, yazılım arayüzlerinin entegrasyonu için hikayeler planlanmıştır.

Her koşu sonrası yapılan koşu gösterimleri, çalışan yazılımlar üzerinden simülatörler kullanılarak yapılmıştır. Gösterimler ortalama 1 gün sürmüş, tüm ekip katılımı ile yapılmıştır. KA ve Gömülü Yazılım Geliştirme Ekipleri tarafından ortak ve entegre gösterimler yapılmıştır. Gösterimler sırasında koşu gereksinim setinin tüm maddeleri üzerinden gidilmiştir. Tamamlanan ve gösterimde başarıyla gösterilen maddeler "tamamlandı" olarak belirlenmiş ve daha sonra bu durum koşu gereksinim setine yansıtılmıştır. Tamamlanamayan ya da gösterim sırasında sorun çıkan maddeler genellikle bir sonraki koşu planına aktarılmıştır. Gösterim toplantı duyurusu İş/Ürün Sahibi tarafından yapılmıştır, gösterimin içeriği ve düzenlemesi SCRUM Yöneticileri tarafından yapılmıştır.

Koşu ilerleme grafiklerini hazırlamak için öncelikle tüm gereksinimlerin puanlanması gerekir. Pilot proje uygulamasında tüm gereksinimler işin başında puanlanmamış, koşu kapsamında puanlama yapılmıştır. Her koşuda ne kadar iş planlanıp ne kadar gerçeklendiği ile ilgili grafikler hazırlanmıştır. Puanlamalarda standart yaklaşımlar belirlemek zaman içinde kazanılan tecrübeye bağlı olarak sağlanmaktadır.

İş/Ürün Sahibi kimin ne kadar puanlık iş yaptığı gibi konulardan çok, hangi işlerin tamamlandığı, hangi işlerin tamamlanamadığı ve nedenlerini sorgulamaya çalışmıştır. Bir iş için harcanan zaman ölçümleri yerine toplam ne kadarlık iş yapılabildiği ve sonraki koşu için ne kadarlık planlama yapılması gerektiği izlenmiştir.

Tanımlanan sürüm planına göre koşular sonrası Yazılım Konfigürasyon Birimi'nin sürümü oluşturulmuş ve geliştirme ekiplerinden bağımsız olarak çalışan yazılım test ekibine dokümanları ile birlikte aktarılmıştır, sürüm dokümanlarının hazırlanması da koşu gereksinim seti iş maddelerine eklenmiştir. Yazılım test ekibi, yeterlilik testleri CMMI "geçerlileme" uygulamaları dikkate alınarak ASELSAN REHİS Grubu Süreç ve Yönergeleri'ne göre gerçekleştirmiştir.

4 Değerlendirme

CMMI-DEV ve Çevik Yöntemler'in karşılaştırıldığı Tablo-1'de verilen farklı bakış açılarına göre yapılan pilot çalışma incelendiğinde aşağıdaki gibi bir değerlendirme ortaya çıkmıştır:

- **Odaklanma ve Perspektif.** SCRUM yöntemi ile yazılım faaliyetleri yönetilen pilot projenin odak noktası öncelikle müşteri ihtiyaçlarını karşılayan ürünü teslim tarihinde elde etmek olmuştur. Gereksinimlerini önceliklendirmek, müşterinin istediği gereklere öncelik verilmesini sağlamış ve ürünün ilk versiyonlarında temel

isteklerin karşılanması hedeflenmiş, her koşuda kazandırılan iş değerine (gerçeklenen gereksinimler) odaklanılmıştır. Her koşu sonunda yapılan retrospektif toplantılarına ek olarak proje sonunda genel bir değerlendirme toplantısı yapılmıştır. Projede edinilen bilgi birikimi ve tecrübe, proje ekibi tarafından hazırlanan proje kapanış raporu ve REHİS GYM (Görev Yazılımları Müdürlüğü) seviyesinde yapılan bilgilendirme sunuşu ile diğer proje ekipleri ile paylaşılmııştır. Bu bilgi paylaşımı ile perspektif, proje seviyesinden REHİS GYM seviyesine taşınmıştır.

- **Yönetim.** Geliştirilen ürünün yeterliliğini doğrulama faaliyetleri, koşular boyunca Yazılım Ekip Lideri tarafından İş/Ürün Sahibi rolü ile yürütülmüştür. KA ve Gömülü Yazılım Ekiplerine atanan SCRUM yöneticileri ise sorumlu oldukları yazılım ekiplerine iş atamalarını yapmış ve ekip koçu olarak ekip üyelerinin karşılaştığı bağımlılık, arayüz yönetimi vs. gibi sorunların çözülmesine yardımcı olmuşlardır. Bu anlamda geliştirme sürecinin yönetimi, geleneksel (komuta-kontrol) yöntem yanısıra, koçluk şeklinde de yapılmıştır. Süreçlerle tanımlanan yazılım yeterlilik testleri faaliyetleri de oluşturulan sürümler üzerinden yazılım test ekibi tarafından gerçekleştirilmiştir. Sürüm planlaması Yazılım Ekip Lideri tarafından koordine edilmiştir. Yazılım ekip lideri koçluk mekanizmasına paralel şekilde süreçlerle tanımlı yöntemlerin uygulanmasını da koordine etmiştir. Süreçlerin gerektirdiği faaliyetler gerektiğinde koşu gereksinim setine "işler" olarak eklenmiş ve takibi yapılmıştır.
- **Planlama.** Yazılımlara yönelik çalışmaların planlanması, iş gücü tahminlemesi, konfigürasyon yönetimi, kalite faaliyetleri gibi süreçsel faaliyetlerin planlanması, proje planlama süreçlerine uygun olarak Yazılım Ekip Lideri tarafından gerçekleştirilmiş ve Yazılım Geliştirme Planı dokümanı ile yayınlanmıştır. Plan, süreçlere uygun olarak takip edilmiş, dönemsel ölçümler alınmış ve raporlanmıştır.
- Geliştirme çalışmasının planlanması kapsamında, gereksinimler arasında yüksek öncelik verilen yeteneklerin önce gerçeklenmesine çalışılmıştır. Artımsal planlama yaklaşımı ile her koşuda yapılacak işler koşu planlama toplantısında planlanmış ve proje ekibi ile paylaşılmıştır. Koşu süresi 2 hafta olarak belirlenmiştir, her koşu sonunda çalışan ürün geliştirilmiştir.
- **Öğrenme.** Koşular sırasında öğrenilen bilgiler koşu değerlendirme ve retrospektif toplantılarında paylaşılmıştır. Proje sonunda gerçekleştirilen bilgilendirme sunuşu ile projede edinilen bilgi REHİS GYM seviyesine taşınmıştır.
- Projede gerçekleştirilen iyi uygulamalar ve öğrenilen dersler, CMMI uygulaması gereği olarak, proje sonunda proje ekibi tarafından hazırlanan proje kapanış raporu ve yapılan bilgilendirme sunuşu ile REHİS GYM bünyesindeki diğer proje ekiplerine aktarılmıştır. Böylece projede edinilen bilgi birikimi proje seviyesinden REHİS GYM seviyesine taşınmıştır.
- **Değerlendirme.** Her sabah yapılan kısa toplantılarla işlerin ilerleme durumu izlenmiş, tüm ekip üyelerinin yaptığı iş görünür kılınarak işler ve sorumluluklar netlik kazanmıştır. Pilot çalışma sonucunda elde edilen sonuca göre değerlendirme yapılarak buradaki uygulamaların Kalite yönetim sisteminde "SCRUM Uygulama Rehberi" olarak kaydedilmesi önerilmiştir.

- Yazılım Ekip Lideri, katıldığı, süreçlerle tanımlı, genel proje toplantılarında, gelinen aşama hakkında üst yönetime gerekli bilgileri aktararak çalışma sonuçlarının diğer proje ekibi üyeleri tarafından da izlenebilmesini sağlamaya çalışmıştır.
- **Kişisel Gelişim.** İteratif çalışmada tamamlanan hikayeler ile her koşu sonunda çalışır ürünün olması ekip motivasyonunu sağlamıştır. Geliştirme ekibi KA ve Gömülü Yazılım geliştirme konularında tecrübeli kişilerden oluşturulmuştur. Ekibin çalışmaları Microsoft Excel aracı ile desteklenmiştir.
- **Proje Yaşam Döngüsü Faaliyetleri.** Pilot proje boyunca her koşu sonunda ürünün çalışıp çalışmadığı doğrulanmıştır. Koşu planına göre her koşuda yapılacak işler belirlenmiştir. KA ve Gömülü Yazılım Ekipleri'nin koşu sonunda ortak gösterimler hazırlamaları sistem entegrasyonu öncesinde yazılım konfigürasyon birimi entegrasyonlarının yapılması ve test edilmesini sağlamıştır. KA ve Gömülü Yazılımlar'ın ortak senaryoları paralel şekilde gerçeklemesi teslim edilebilir bir ürünün sürekli hazır olmasını sağlamıştır. Projede gerçekleştirilen dokümantasyon, analiz ve gözden geçirme faaliyetleri kurumsal seviyede tanımlı süreçlere göre uygulanmıştır.
- **Tahminleme.** Ürünün kalite performansını ölçmek için her koşu sonunda testler yapılmış, hata yoğunluğu takip edilmiş ve koşu ilerleme grafiğinde ürünün ne kadarının tamamlandığı incelenmiş, buna göre sonraki koşu için planlama yapılmıştır.

Pilot uygulama sonucunda, daha iyi uygulanabileceği düşünülen konular test aşamasında yoğunlaşmıştır. Her koşu sonrası yapılan gösterimlerde test akışını iyi planlamak, test durumları ve senaryolarını iyi belirlemek gerekmektedir. Ayrıca, her koşu öncesi kısa bir analiz çalışması ve ürün gereksinim seti ile tutarlılık kontrolü yapılmasının faydalı olacağı değerlendirilmektedir. Aynı şekilde tasarım faaliyetleri için de, koşu içinde belirli bir zaman ayrılması gerektiği değerlendirilmektedir. Pilot uygulama sonucunda iyileştirme önerileri verilmiştir. Şöyle ki; Ekip farklı uzmanlık alanlarından gelen kişilerle zenginleştirilebilir[1]. Örneğin yazılım test ekibinden geliştirme ekibine katılım, İş/Ürün Sahibi'nin Sistem Mühendisliği ya da Proje Yönetimi'nden tanımlanması değerlendirilebilir. Ayrıca geliştirme ekibinin çalışmalarını desteklemek için Microsoft Excel'e alternatif olarak Çevik Yöntemler'in uygulanmasını destekleyen Agilo, IBM Rational Jazz gibi platformların kullanılması da önerilmiştir.

5 Sonuç

Sonuç olarak SCRUM'ın CMMI-DEV Seviye-3 olan bir organizasyonda herhangi bir uyumluluk sorunu yaşanmadan uygulanabildiği gözlemlenmiştir. Burada önemli olan CMMI-DEV'in sağladığı altyapının beklentilerini net olarak algılayıp, SCRUM'ın bu modelde nasıl bir yere sahip olması gerektiğini netleştirebilmek olmuştur. Örnek uygulama bölümünde anlatıldığı gibi pilot proje hem kullanıcı arayüzü hem de gömülü sistemleri oluşturan yazılım konfigürasyon birimlerinden oluşmuştur. Sistem seviyesinde arayüzleri olan bir projede SCRUM'ın kolaylıkla uygulanabilmesinin

nedeni CMMI-DEV'ın Entegre Proje Yönetimi, Risk Yönetimi ve Mühendislik (tasarım ve geliştirme) süreç alanları ile proje yönetimi ve entegrasyonun sağlanmasına yönelik gerekli altyapının kurulmasına destek sağlaması olarak değerlendirilmiştir [10].

Kaynaklar

1. Study and practice of Import Scrum agile software development", Hu Guangyong, Nanjing Institute of Industry Technology, Nanjing 210046, China
2. CMMI or Agile: Why Not Embrace Both!, Hillel Glazer, Entinex, Inc., Jeff Dalton, Broadsword Solutions Corporation, David Anderson, David J. Anderson & Associates, Inc., Mike Konrad, SEI, Sandy Shrum, SEI, November 2008.
3. "The New New Product Development Game" by Takeuchi and Nonaka. HarvardBusiness Review, January 1986.
4. Rafael E. Landaeta "Strategic Management of Scrum Projects: An Organizational Learning Perspective" , Old Dominion University 241 Kaufman Hall, Norfolk, VA 23462, Rlandaet@odu.edu, Stacia Viscardi, Agile Evolution, Inc.
5. Stephen Swoyer. "Agile Programming and the CMMI: Irreconcilable Differences?" Application Development Trends, February 2005.
6. David Anderson. "Stretching Agile to Fit CMMI Level 3: The Story of Creating MSF for CMMI Process Improvement at Microsoft Corporation." Agile Conference. Denver, CO, 2005. http://ieeexplore.ieee.org/iel5/10705/33795/01609821.pdf
7. Mark Paulk. "Extreme Programming from a CMM Perspective." IEEE Software, November/December 2001.
8. Pablo Santos. "SCRUM Meets CMMi: Agility and Discipline Combined." Dr. Dobb's Portal: The World of Software Development, August 2007.
9. M. B. Chrissis, M. Konrad, S. Shrum, CMMI-DEV Version 1.3 "CMMI for Development Guidelines for Process Integration and Product Improvement" Third Edition, pp.287-303.
10. Neil Potter, The Process Group, Comparing Scrum And CMMI, How Can They Work Together, version 1.6.
11. Jeff Sutherland, Carsten Ruseng, Jacobsen, & Kent Johnson. "Scrum and CMMI Level 5: A Magic Potion for Code Warriors." Agile Conference. Denver, CO, July, 2005. http://jeffsutherland.com/2007/09/scrum-and-cmmi-level-5-magic-potion-for.html