CMMI ve ÇEVİK YAZILIM GELİŞTİRME YÖNTEMLERİNİN BİRLİKTE UYGULANABİLİRLİĞİ

Özden Özcan Top[1] Onur Demirörs [2]

[1]Fujitsu Technology Solutions, Ankara, Türkiye
ozden.top@fujitsu.net.tr
[2] Enformatik Enstitüsü, Orta Doğu Teknik Üniversitesi, Ankara, Türkiye
demirors@metu.edu.tr

Özet. CMMI ve çevik yazılım geliştirme yöntemlerinin birlikte uygulanabilirliği bir süredir devam etmekte olan bir tartışma konusudur. İki yaklaşımın çeşitli açılardan birbirinin karşıtı olduğuna dair yorumların yanında literatürde birlikte uygulanabilirliklerini gösteren çalışmalar da bulunmaktadır. Bu makalede CMMI ve çevik yöntemler üzerine var olan beş efsanenin oluşma nedenleri ve gerçek olmaktan uzak olmalarının nedenleri literatür araştırması sonucu elde edilen aksi örnekler üzerinden açıklanmaktadır.

1 Giriş

Çevik yazılım geliştirme manifestosu yayımlandığından bu yana "Extreme programming (XP)" (Beck, 2000), "Scrum" (Sutherland, Schwaber, Scrum, & Sutherl, 2007), "Adaptive software development" (Highsmith & Orr, 2000), "Crystal family" (Cockburn, 2004) and Lean Software Development (Poppendieck & Poppendieck, 2003) gibi birçok çeviklik odaklı yazılım geliştirme yöntemi geliştirildi ve yazılım organizasyonlarında yaygın olarak kullanılmaya başlandı.

Çevik yazılım geliştirme yöntemleri ve plan odaklı süreç iyileştirme modelleri arasındaki ilişkiler ise bir süredir yazılım dünyasının gündeminde yer almaktadır (B. Boehm & Turner, 2004). Plan odaklı süreç iyileştirme modeli olarak ilk akla gelen 2000 yılında yayımlanmış olan Bütünleşik Yetenek Olgunluk Modelidir (Capability Maturity Model Integrated) (Institute, 2010).

CMMI ile çevik yazılım geliştirme modellerinin birlikte uygulanabilirliği üzerine olumlu bildirimler olmakla birlikte, önyargılar ve soru işaretleri net olarak giderilmiş değildir. Önyargılar zaman içerisinde efsanelere dönüşmekte ve yıkılması güç bir hal almaktadır.

Bu çalışmada CMMI ve Çevik yazılım geliştirme yöntemlerinin birlikte kullanılabilirliği üzerine yanlış algılara ya da yorumlara neden beş efsane ve bu efsanelerin altında yatan nedenler literatür araştırması ile desteklenerek açıklanmıştır. Çalışmanın amacı CMMI ve çevik yazılım geliştirme yöntemlerinin birlikte

kullanılabilirlik sınırlarını net olarak belirlemek ve birlikteliklerinden doğacak faydaları ortaya çıkartmaktır.

Efsaneler aşağıda listelenmiştir:

- CMMI'daki yoğun dokümantasyon çevik yöntemlerin dokümantasyondan uzak yaklaşımı ile çelişir.
- Yüksek CMMI olgunluk seviyeleri çevik yazılım geliştirme pratikleri ile elde edilemez .
- CMMI büyük ölçekli projeler; çevik yazılım geliştirme yöntemleri küçük takımlar ve küçük projeler için uygulanabilirdir.
- CMMI Savunma Sanayii projelerinde uygulanabilirken çevik yöntemler uygulanabilir değildir.
- Çevik yöntemlerin tersine CMMI katı süreçler uygulanmasını zorunlu kılar.

Bu önyargıların/varsayımların bir çoğu yanlış uygulamalardan kaynaklanmaktadır. Makalenin geri kalanında bu varsayımların altında yatan nedenler, gerçek olmaktan uzak olmalarının nedenleri literatür araştırması sonucu elde edilen aksi örnekler üzerinden açıklanmaktadır.

2 Efsaneler

2.1 CMMI'daki yoğun dokümantasyon çevik yöntemlerin dokümantasyondan uzak yaklaşımı ile çelişir

Bu efsanenin ortaya çıkma nedeni hem çevik yöntemlerin hem de CMMI'ın dokümantasyon yaklaşımının yanlış yorumlanmasından kaynaklanmaktadır. İki uç yorumun bir tarafında yoğun dokümantasyon oluşturulacağı diğer tarafında hiç doküman oluşturulmayacağı fikri yer alır.

Dokümantasyonun temel amacı öğrenilen bilgilerin aktarılması, ürünün bakım aşamasında devamlılığının sağlanması, ürün ve süreç gereksinimlerinin görünür kılınması ve iş ürünleri arasında tutarlılık sağlanabilmesidir.

CMMI süreç alanlarının ilişkili iş ürünleri ve pratikler bazında çevik yazılım geliştirme yöntemleri için yorumlanmasına olanak verir.

CMMI v1.3 *gereksinim geliştirme* süreç alanı altında gereksinimlerin dokümantasyon detayının takım içerisindeki koordinasyon ihtiyacı ve sonraki iterasyonlara öğrenilen bilgilerin aktarılma detayı ile belirlenebileceği ifade edilmektedir (Institute, 2010). Bu ihtiyaç müşterinin doğrudan takım içerisinde yer aldığı durumlarda büyük ölçüde azalacak olmasına rağmen farklı çözüm alternatiflerinin değerlendirilmesi için müşteri ve ürün gereksinimlerinin dokümantasyonuna yine de ihtiyaç olabilir.

Yukarıdaki örnekte de görüldüğü gibi CMMI çevik yazılım geliştirme ortamları için sadece "zorunlu" dokümantasyon vurgusu yapar (Kulpa & Johnson, 2008).

Projelerdeki dokümantasyon yoğunluğu diğer tüm parametrelerin yanında proje türüne, kritikliğine ve büyüklüğüne bağlı olarak değişiklik gösterir. CMMI uygulamada yoğun dokümantasyon algısının oluşmasının diğer nedenlerinden biri de

özellikle savunma sanayiinde geliştirilen güvenlik kritik projelerde katı kuralları olan güvenlik standartlarına (örn. DO-178B) uyulması zorunludur. Bu standartlara uyum belirli dokümanlara bağlı kalarak sağlanır.

Diğer taraftan çevik yazılım geliştirme yöntemleri önemli ve acil olmadıkça doküman oluşturmama bunun yerine yüzyüze iletişim yolu ile bilgi ve fikirlerin paylaşılması vurgusunu yapar (Martin, 2003). Fakat bu vurgu nedeniyle önemli dokümantasyonun ne olduğu süreçlere göre değil uygulayıcılara göre değişiklik gösterebilmekte doküman oluşturmamak için çevik yöntemler mazaret olarak kullanılmaktadır.

Sürekli yaşayan dokümanlar oluşturmanın getireceği yükler göz önünde bulundurularak hem bakım hem de esneklik arasında denge sağlayacak doküman yaklaşımları bulmak gerekir. Hem çevikliğin önünde engel olmayacak hem de CMMI süreçleri ile uyumlu dokümantasyon oluşturmak mümkündür.

BBC Worldwide, CMMI olgunluk seviyesi 4'e ulaştığında yalın yazılım geliştirme (Lean Software Development) tekniklerini uygulamaktaydı (Middleton & Joyce, 2012). DTE Enerji Şirketi de çevik pratikleri uygulayarak CMMI Seviye 3'e ulaşmıştır (Baker, 2005), (Baker, 2006). CMMI Maturity Level 5 has been achieved Systematic Yazılım Şirketi 2005 yılında CMMI Seviye 5'e ulaştığında Scrum kullanmaktaydı ve halen Scrum uygulanmaya devam edilmektedir (Sutherland, Ruseng Jakobsen, & Johnson, 2008) (Jakobsen & Johnson, 2008), (Jakobsen & Sutherland, 2009).

2.2 Yüksek CMMI Olgunluk Seviyeleri Çevik Yazılım Geliştirme Pratikleri ile Elde Edilemez

Yazılım geliştirme dünyasında CMMI Seviye 4/5 olgunluk seviyesine kıyasla 2/3 seviyesinde olan çok daha fazla organizasyon bulunmaktadır. Bu da çevik yazılım geliştirme yöntemlerinin CMMI seviye 2/3 için daha çok sayıda uygulanma olasılığını artırmaktadır (Sison & Yang, 2007), (Kähkönen & Abrahamsson, 2004), (Baker, 2005). Her nekadar yazılım organizasyonlarının odağında seviye 4 ve 5'in yer alması bu efsanenin oluşmasında rol oynasa da CMMI seviye 4/5'in özel ve genel hedefleri ile çevik pratikler arasında herhangi bir uyuşmazlık bulunmamaktadır.

4. olgunluk seviyesinde süreç performansı istatistiksel ya da diğer nicel yöntemler teknikler kullanılarak ölçülür ve süreçler daha tahmin edilebilir bir hal alır. 5. olgunluk seviyesinde ise organizasyon yenilikçi teknikler kullanarak süreçlerini sürekli iyileştirmeye odaklanır (Institute, 2010). Süreçleri iyileştirmek üzere düzenli olarak metrikler toplanır, doğrulukları kontrol edilir. Çevik yazılım geliştirme yöntemlerindeki kontrol mekanizmaları tahmin edilebilirliği ve risklerin kontrol edilebilirliğini artırır (Schwaber, 1997) Süreç izleme faaliyetlerinin geleneksel yöntemlere kıyasla çevik pratiklerle daha iyi gerçekleştirildiği McMahon tarafından ifade edilmiştir (McMahon, 2010).

2.3 CMMI Büyük Ölçekli Projeler; Çevik Yazılım Geliştirme Yöntemleri Küçük Takımlar Ve Küçük Projeler için Uygulanabilirdir

Çevik yazılım geliştirme yöntemlerinin küçük takımlar üzerindeki olumlu etkisi kanıtlanmıştır (Turk, France, & Rumpe, 2002). Geleneksel methodolojilerin hantal yapısının aksine, çevik yöntemlerin büyük ölçekli projelerde belirli uyarlamalar yapılarak uygulanabilir olduğu belirtilmektedir. Boehm, Cockburn ve Highsmith tarafından 250 kişilik bir projede başarıyla uygulanan çevik yöntemlere referans vermektedir (Barry Boehm, 2002). Literatürde ayrıca çevik yöntemlerin büyük ölçekli projeler için nasıl uyarlanacağını, başarılı uyarlamaları açıklayan çalışmalar bulunmaktadır (Cao, Mohan, Xu, & Ramesh, 2004) and (Paasivaara, Durasiewicz, & Lassenius, 2008).
Diğer taraftan süreçlerin etkinliğinin artırılması sadece büyük değil küçük orta ölçekli organizasyonların da ihtiyacıdır. CMMI, küçük ve orta ölçekli projelerde de belirli uyarlama zorluklarına ragmen uygulanabilmektedir (Garcia, 2005).

2.4 CMMI Savunma Sanayii Projelerinde Uygulanabilirken Çevik Yöntemler Uygulanabilir Değildir

CMMI'ın ortaya çıkış gerekçesi Amerikan Savunma Bakalığının savunma sanayii projelerindeki başarısızlıkları önlemek ve projelerde belirli bir kaliteye ulaşmak istemesidir. Bunun sonucu olarak CMMI uzun bir süre büyük ve yaşam kritik projelere cevap veren bir model olarak görülmüştür (McMahon, 2010). Fakat ilerleyen zamanla birlikte CMMI farklı sektörlerde uygulanmaya başlanıp dünya çapında bir süreç iyileştirme standardına dönüşmüştür (Institute, 2010). SEI tarafından yayımlanan rapora göre CMMI'ın telekominikasyon, finans, üretim ve savunma sektörlerinde uygulandığını görüyoruz (Gibson, Goldenson, & Kost, 2006).

Diğer taraftan çevik yazılım geliştirme yöntemleri projelerin karakteristiklerine göre farklı sektörlerde uygulanmaktadır. Kritiklik, güvenlik ve güvenilirlik savunma sanayii projelerinin temel özellikleridir. Çevik yazılım geliştirme yöntemlerinin bu tür projelerde uygulanabilirliği hakkında karşıt fikirler bulunmaktadır. Bazı araştırmacılar test odaklı geliştirme (test driven development) yaklaşımının bu gereksinimleri karşıladığını savunurken (Lindvall et al., 2002) savunma sanayii projelerindeki geliştirme ortamının çevik projelere özgü sinerjiyi yok ettiği de savunulmaktadır.

2.5 Çevik Yöntemlerin Tersine CMMI Katı Süreçler Uygulanmasını Zorunlu Kılar

Çevik yazılım geliştirme yöntemlerinin zaman zaman belirli bir kurala ya da sürece bağlı olmayan geliştirme yaklaşımları ile karıştırıldığına tanıklık ediyoruz. Fakat çevik yöntemler bu yanılgının aksine kaos ortamından uzak, sağlam ilkelere dayanan bir dizi prensipten/süreçten oluşmaktadır (Cao & Ramesh, 2007).

CMMI ise yazılım geliştirme süreçlerinin belirli bir disipline bağlı kalarak uygulanmasında yol gösterici olan bir süreç iyileştirme modelidir. Burada çeviklik ile disiplin arasında çelişki olup olmadığını değelendirmek gerekir. Rong ve arkadaşları

plan ve çeviklik odaklı süreçlerin birbirinin tamamlayıcısı olduğu belirtmiş; çalışmalarında Kişisel Yazılım Geliştirme (Personel Software Process) ve Scrum yaklaşımları örneğini vermişlerdir (Rong, Shao, & Zhang, 2010).

Katı süreçler, esnek olmayan hiyerarşi ve disiplin çevik yaklaşımların karşıtı olarak görülebilir fakat belirli bir seviyede disiplin uzun vadede başarı elde etmek için gereklidir. Çevik yaklaşımlar kendi içsel disiplinlerine bağlı kalarak uygulanmalıdır (Paulk, 2002). Çevik yöntemlere ait süreçlerin eksik ya da tutarsız uygulanışı projelerin başarısız olması ile sonuçlanabilir. Bu nedenle CMMI, çevik yöntemlerin karşıtı olmamanın yanında süreçlerin belirli bir disiplin altında uygulanarak içselleştirilmesine olanak sağlar.

3 Sonuç

CMMI ile çevik yazılım geliştirme yöntemlerinin birlikte uygulanabilirlikleri bir süredir devam etmekte olan ve üzerinde bir dizi varsayım ve efsanenin dolaştığı bir tartışma konusudur. Bu çalışmada bir süreç iyileştirme modeli olarak CMMI ve çevik yazılım geliştirme yöntemlerinin birlikte uygulanabilirlikleri kişisel deneyimler ve literatür araştırması sonuçları bir araya getirilerek değerlendirilmiştir.

Efsanelerin ortaya çıkmasının temel nedenlerinden birinin her iki yaklaşımın farklı açılardan yanlış yorumlanması olduğu görülmüştür. CMMI-Dev kılavuzunda her süreç alanı için süreç alanının çevik yöntemlere özgü yorumlandığı açıklayıcı bilgilere yer verilmektedir (Institute, 2010).

Çevik yazılım geliştirme yöntemleri 2000'li yıllların başından beri yoğun olarak tercih edilmekte olsa da zaman zaman disiplinsiz uygulamalar, doküman üretmeme gibi durumlar için özür niteliğinde uygulandığı da gözlenmiştir. Bu tür yanlışların önüne geçerek en yüksek faydayı elde etmek için çevik geliştirme yöntemleri üzerine olgunluk modelleri geliştirilmektedir. Agile Adoption Framework (Sidky, Arthur, & Bohner, 2007) ve Scrum Maturity Model (Yin, Figueiredo, & Mira da Silva, 2011) var olan olgunluk modelleri arasında yaptığımız değerlendirme sonucu en başarılı bulduğumuz modellerdir (Özcan Top & Demirörs, 2013).

Doğru uyarlamalar sonrasında CMMI odaklı süreç iyileştirme sonrası yürütülen projelerde çeviklikten ödün vermemek mümkündür. Burada da tanımlanmış olan ön yargıları somut olarak yıkmanın en önemli aracı CMMI ve çevik yöntemlerin bir arada başarı ile uygulandığını açıklayan bilimsel araştırma sonuçlarıdır. CMMI ve çevik yöntemlerin bir arada uygulandığı daha fazla çalışmanın sonuçlarını görmeye ihtiyaç vardır.

Kaynaklar

1. Baker, S. W. (2005). *Formalizing agility: an agile organization's journey toward CMMI accreditation.* Paper presented at the Agile Conference
2. Baker, S. W. (2006). *Formalizing agility, part 2: how an agile organization embraced the cmmi*.

3. Beck, K. (2000). *Extreme programming explained: embrace change*: Addison-Wesley Professional.
4. Boehm, B. (2002). Get ready for agile methods, with care. *Computer, 35* (1), 64-69.
5. Boehm, B., & Turner, R. (2004). *Balancing agility and discipline: Evaluating and integrating agile and plan-driven methods*.
6. Cao, L., Mohan, K., Xu, P., & Ramesh, B. (2004). *How extreme does extreme programming have to be? Adapting XP practices to large-scale projects*.
7. Cao, L., & Ramesh, B. (2007). Agile Software Development: Ad Hoc Practices or Sound Principles? *It Professional, 9*(2), 41-47.
8. Cockburn, A. (2004). *Crystal clear: a human-powered methodology for small teams*: Addison-Wesley Professional.
9. Garcia, S. Z. (2005). Thoughts on applying CMMI in small settings. *Software Engineering Institute*.
10. Gibson, D., Goldenson, D., & Kost, K. (2006). *Performance Results of CMMI-Based Process Improvement*: Software Engineering Institute.
11. Highsmith, J. A., & Orr, K. (2000). *Adaptive software development: a collaborative approach to managing complex systems*: Dorset House Pub.
12. Institute, C. (2010). Capability Maturity Model Integrated-Development.
13. Jakobsen, C. R., & Johnson, K. A. (2008). *Mature Agile with a twist of CMMI*.
14. Jakobsen, C. R., & Sutherland, J. (2009). *Scrum and CMMI going from good to great*.
15. Kulpa, M. K., & Johnson, K. A. (2008). *Interpreting the CMMI: a process improvement approach*: Auerbach Publications.
16. Kähkönen, T., & Abrahamsson, P. (2004). Achieving CMMI level 2 with enhanced extreme programming approach. *Product Focused Software Process Improvement*, 378-392.
17. Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., et al. (2002). Empirical findings in agile methods. *Extreme Programming and Agile Methods—XP/Agile Universe 2002*, 81-92.
18. Martin, R. C. (2003). *Agile software development: principles, patterns, and practices*: Prentice Hall PTR.
19. McMahon, P. E. (2010). *Integrating CMMI and Agile Development: Case Studies and Proven Techniques for Faster Performance Improvement*: Addison-Wesley Professional.
20. Middleton, P., & Joyce, D. (2012). Lean Software Management: BBC Worldwide Case Study. *Engineering Management, IEEE Transactions on, 59*(1), 20-32.
21. Paasivaara, M., Durasiewicz, S., & Lassenius, C. (2008). *Distributed agile development: Using Scrum in a large project*.
22. Paulk, M. C. (2002). Agile Methodologies and Process Discipline, Institute for Software Research*, Paper 3*.
23. Poppendieck, M., & Poppendieck, T. (2003). *Lean software development: An agile toolkit*: Addison-Wesley Professional.
24. Rong, G., Shao, D., & Zhang, H. (2010). *SCRUM-PSP: Embracing Process Agility and Discipline*.
25. Schwaber, K. (1997). Scrum development process *Business Object Design and Implementation* (pp. 117-134): Springer
26. Sidky, A., Arthur, J., & Bohner, S. (2007). A disciplined approach to adopting agile practices: the agile adoption framework. *Innovations in systems and software engineering, 3*(3), 203-216.

27. Sison, R., & Yang, T. (2007). *Use of Agile Methods and Practices in the Philippines.* Paper presented at the Software Engineering Conference, 2007. APSEC 2007. 14th Asia-Pacific.
28. Sutherland, J., Ruseng Jakobsen, C., & Johnson, K. (2008). *Scrum and CMMI Level 5: The Magic Potion for Code Warriors*.
29. Sutherland, J., Schwaber, K., Scrum, C. O., & Sutherl, C. J. (2007). The scrum papers: Nuts, bolts, and origins of an agile process.
30. Turk, D., France, R., & Rumpe, B. (2002). *Limitations of agile software processes*.
31. Yin, A., Figueiredo, S., & Mira da Silva, M. (2011). *Scrum Maturity Model: Validation for IT organizations' roadmap to develop software centered on the client role.* Paper presented at the ICSEA 2011, The Sixth International Conference on Software Engineering Advances.
32. Özcan Top , Ö., & Demirörs, O. (2013). *Assessment of Agile Maturity Models: A Multiple Case Study.* Paper presented at the Software Process Improvement and Capability Determination, Bremen, Germany.