HAVACILIK VE UZAY TEKNOLOJİLERİ DERGİSİ
OCAK 2010 CİLT 4 SAYI 3 (57-67)

BİR ÇEVİK YAZILIM GELİŞTİRME SÜRECİNİN UYARLANMASI VE UYGULANMASI

Kadir ÇAMOĞLU* **Derya AKBAYIR** **Fatih YÜCALAR** **Selim BAYRAKLI**

Maltepe Üniv.
Bilişim Bölüm Başkanlığı
Yazılım Geliştirme Böl.
kcamoglu@maltepe.edu.tr

Maltepe Üniv.
Mühendislik Fak.
Bilgisayar Müh. Böl.
dersoy@maltepe.edu.tr

Maltepe Üniv.
Mühendislik Fak.
Bilgisayar Müh. Böl.
fatihy@maltepe.edu.tr

Maltepe Üniv.
Mühendislik Fak.
Bilgisayar Müh. Böl.
selim.bayrakli@maltepe.edu.tr

Geliş Tarihi: *23 Eylül 2009,* ***Kabul Tarihi:*** *22 Ocak 2010*

ÖZET
1990'lı yıllarda "çevik" adı altında toplanan yeni yazılım geliştirme yaklaşımları, kısa süreli ve değişken projeler için yeni metodolojiler önermiştir. Çalışan yazılımı kapsamlı belgelemeye tercih eden, plan izleme yerine değişime açık ve müşteriyle sıkı bir etkileşim içinde olan bu metodolojiler sayesinde klasik yaklaşımlarla başarılması riskli projeler, kabul edilebilir ölçüde başarılı olabilmiştir.

Bu çalışmanın konusunu, hem çevik yaklaşımlarla yazılım geliştirme hem de kullanılması gereken internet tabanlı programlama teknolojilerine aşina olmayan bir yazılım ekibi tarafından üniversite otomasyon sisteminin geliştirilmesine yönelik proje süreci oluşturmuştur. Bu çalışmada, çevik metodolojiler içinden seçilen pratiklerin proje sürecine nasıl uyarlandığı ve uygulandığı, uygulama aşamasında hangi sıkıntılarla karşılaşıldığı ve bu sıkıntıları çözümleme yolları anlatılmıştır. Ayrıca bu pratikleri uygulayacaklar için önerilerde bulunulmuştur.

Anahtar Kelimeler: *Çevik, Çevik Yazılım Geliştirme, Extreme Programlama, Scrum.*

TAILORING AND IMPLEMENTATION OF AN AGILE SOFTWARE DEVELOPMENT PROCESS

ABSTRACT
In 1990s, a new software development approach named "agile", suggested new methodologies for projects of relatively short duration and changing requirements. Such software projects which start with uncertain requirements are considered risky under classical approaches, but they became fairly successful by using these newer methods. Agile methods favor working software to detailed documentation, they are open to change and they prefer strong interaction with customers.

This study reports the results of a project development effort deriving its development process from different agile methods adapted for the circumstances. The project aims to develop a university automation system by a software team which was initially unfamiliar with the agile software development approach and internet based programming technologies. We describe how to tailor and implement the chosen practices from the different agile methods, which challenges were addressed during the implementation, and the ways to solve these challenges. Furthermore, we comment on the experience gained and make recommendations on the implementation of these practices.

Keywords: *Agile, Agile Software Development, Extreme Programming, Scrum.*

*Sorumlu Yazar

1. GİRİŞ

Yazılım geliştirme süreci sıkıntılı ve uzun süren bir dönemdir. Yazılım projeleri yönetimsel eksikliklerden dolayı ancak kısmi başarı ve memnuniyet ile tamamlanabilmektedir. Yazılım sektöründe, yazılım sürümlerinin zamanında ortaya çıkarılamaması, değişiklik isteklerine çabuk cevap verilememesi, yazılım hatalarının geç fark edilmesi ve zaman içerisinde gelen isteklere göre sistemin kendi yapısını geliştirememesi gibi çeşitli sorunlar ortaya çıkmıştır. Bu sorunların aşılmasına yönelik yapılan çalışmalar sonucu, 1990'lı yılların sonlarına doğru "çevik" olarak isimlendirilen metotlar geliştirilmiştir [1].

Çevik metotlar, piyasaya çok çabuk ürün çıkarabilme, değişen isteklere hızla yanıt verme ve en kısa sürede bir yazılım ürününü müşteri hizmetine sunmayı amaçlamaktadırlar [2]. Çevik metotlar, verimliliği yüksek, esnek, hata oranı düşük, hızlı ve ucuz çözümler sağlamaktadır. Bu metotlar, kendi içerisinde özü aynı fakat pratikleri farklılaşan çeşitli metodolojilere ayrılmaktadır. Çalışma içersinde bu metodolojilerden bahsedilmektedir.

Çalışmanın 2. bölümünde çevik yazılım geliştirme kavramı ve prensipleri, 3. bölümünde en yaygın uygulanılan çevik metodolojiler, 4. bölümde uygulama, karşılaşılan zorluklar, çözümleme girişimleri ve öneriler, son bölümde ise ortaya çıkan sonuçlar anlatılmaktadır.

2. ÇEVİK YAZILIM GELİŞTİRME

"Çevik", dünyada yazılım süreçlerini daha esnek ve güçlü kılmak için kullanılan aynı zamanda yazılım süreçlerini de kısaltan kavramsal bir yazılım geliştirme metodolojisidir [3]. Bu metodolojide projenin ölçeği ne olursa olsun, proje küçük yinelemelere ayrılır ve her yineleme başlı başına bir proje gibi ele alınarak geliştirilir. Her yinelemenin sonunda da proje ekibi tarafından müşteriye, projenin ne kadarının gerçekleştirildiğine dair bilgi verilir. "Çevik" ile her bir yinelemenin 2-4 hafta kadar sürmesi planlanmaktadır. Her yinelemenin kendi içerisinde çalışan bir sistem olması sonucu müşteriye sürekli çalışan bir yazılım teslim edilerek, müşteri memnuniyetinin artması sağlanmaktadır. "Çevik" in hızı proje ekibinde çalışan tüm ekip üyelerinin sürekli iletişim halinde olmasından kaynaklanmaktadır. Ayrıca projenin küçük parçalardan oluşması da geriye dönük hataların düzeltilmesini kolaylaştırmaktadır. Genel hatlarıyla çevik metotlar, verimliliği yüksek, esnek, hata oranı düşük, hızlı ve ucuz çözümler sağlamaktadır.

2.1 Çevik Yazılım Geliştirme Manifestosu

Değişik çevik metotların temsilcileri 2001 yılında bir araya gelerek ortak yanlarını ortaya koyan bir manifesto yayınladılar. Bu manifestoda;

- Süreçler ve Araçlar yerine Bireyler ve Etkileşimler,
- Kapsamlı Belgeler yerine Çalışan Yazılım,
- Sözleşme Görüşmeleri yerine Müşteri İlişkileri,
- Plan İzleme yerine Değişikliğe Açıklığın,

daha önemli ve öncelikli olduğu belirtilmektedir [4].

2.2 Çevik Yazılım Geliştirme Prensipleri

Çevik manifestonun altında yatan temel prensipler [5] şunlardır;

- Öncelik, sürekli ve hızlı olarak kaliteli yazılım teslimatıyla müşteri memnuniyetini sağlamaktır.
- Proje ne kadar ilerlemiş olursa olsun değişiklikler kabul edilir. Çevik yazılım süreçleri, değişiklikleri müşteri avantajına dönüştürürler.
- Mümkün olduğunca kısa zaman aralıklarında çalışan, kaliteli yazılım teslimatı yapılır.
- Tüm ekip elemanları yüz yüze iletişim halinde, günlük olarak birlikte çalışırlar.
- Projeler motivasyonu yüksek bireyler etrafında geliştirilirler. Ekip elemanlarına gerekli destek verilmeli, ihtiyaçları karşılanmalı ve yapılan iş ile ilgili onlara güvenilmelidir.
- Ekip içerisinde kaliteli bilgi akışı için yüz yüze iletişim önemlidir.
- Çalışan yazılım, projenin öncelikli gelişim ölçütüdür.
- Çevik süreçler, sürdürülebilir geliştirmeyi destekler.
- Sağlam teknik alt yapı ve tasarım, çevikliği arttırır.
- Basitlik önemlidir.
- En iyi mimariler ve tasarımlar kendini organize edebilen ekipler tarafından yaratılır.
- Düzenli aralıklarla ekip kendi yöntemlerini gözden geçirir ve verimliliği arttırmak için gerekli iyileştirmeleri yapar.

3. ÇEVİK METODOLOJİLER

Extreme Programming (XP), Scrum, Agile Unified Process, Future Driven Development (FDD), LEAN Development, Dynamic System Development Methodology (DSDM) ve Microsoft Solution Framework (MSF) olarak bilinen çevik metodolojiler vardır [1]. Bu metodolojiler arasından en yaygın uygulananları XP ve Scrum'dır.

3.1. Extreme Programming

Extreme Programming (XP), Kent Beck tarafından 1999 yılında bir yazılım geliştirme disiplini olarak ortaya çıkarılmıştır. Yazılım geliştirmede kolaylığı ve esnekliği sağlamak için, 12 farklı pratiği öngören XP, grup içi iletişime önem veren, geri dönüşlerin daha fazla olmasına imkân sağlayan bir yazılım geliştirme yöntemidir [6].

3.2. SCRUM

SCRUM, Jeff Sutjerland ve Ken Schwaber tarafından 1990'ların ortalarında geliştirilen, çevik yazılım geliştirme metodolojileriyle uygulanabilecek bir proje yönetim yaklaşımıdır. Karmaşık yazılım işlerini küçük birimlere (sprint) bölerek geliştirmeyi öngörür. Bu metodoloji, karmaşık ortamlarda adım adım yazılım geliştiren küçük ekipler için uygundur. Gereksinimlerin kolaylıkla tanımlanamadığı ve kaotik durumların beklendiği projeler için en uygun metodolojidir. Bu metodolojide bir yinelemenin tamamlanması 30 günden fazla sürmemekte ve günlük 15 dakikalık toplantılarla sürekli iş takibi yapılmaktadır. SCRUM 9 farklı pratiği öngörür [7].

3.3. Agile Unified Process

Agile Unified Process, Rational Unified Process (RUP)'in basitleştirilmiş bir versiyonudur. Agile Unified Process, RUP'a sadık kalarak çevik teknikleri ve kavramları kullanan yazılım geliştirme yaklaşımının anlaşılmasını kolaylaştıran bir metodolojidir [8].

3.4. Feature Driven Development

Feature Driven Development (FDD), Jeff De Luca tarafından doksanlı yılların sonunda geliştirilmiş bir çevik metodolojidir. FDD, özellik (feature, function) güdümlü çalışır. Sisteme yeni bir özellik kazandırılmadan önce, detaylı bir tasarım çalışması yapılarak bu özelliği kapsayan mimari yapı oluşturulur. Bu yüzden FDD daha çok tasarım odaklı işleyen bir çevik süreçtir [1].

3.5. Lean Development

Bob Charette tarafından geliştiren "Lean Development" 1980'de Japon otomobil endüstrisindeki yeniden yapılanmada kullanılan Lean üretimin prensipleri temel alınarak, 2000'li yılların başında yazılıma uyarlanmış bir çevik metodolojidir. Bob, geleneksel metodolojilerin risk olarak gördüğü değişimi, kısıtlayıcı yönetim uygulamalarıyla kontrol ederek bu değişimin yeni fırsatları üretilebilmesini sağlamıştır [1][9].

3.6. Dynamic System Development Methodology

Dynamic System Development Methodology (DSDM), hızlı yazılım geliştirme adımlarının kontrolünü sağlayan bir metodolojidir. Büyük Britanya' da bir konsorsiyum tarafından 1990'lı yılların ortalarında geliştirilmiştir [1]. Bu yaklaşımda fonksiyonel gereksinimlere öncelik verilmesi gereklidir. Aynı zamanda yüksek kalite ve değişen gereksinimlere adapte olunması önemlidir.

3.7. Microsoft Solution Framework

Microsoft Solution Framework (MSF), ilk olarak Microsoft tarafından 1993 yılında versiyon 1.0 olarak yayınlanmıştır. Bunu sonraki yıllarda sırasıyla; 1997'de versiyon 2.0, 1999'da versiyon 2.5, 2002'de versiyon 3.0 ve en son 2005 yılında ise versiyon 4.0 yayınlanması izlemiştir [10]. MSF başarılı bilişim çözümleri için geliştirme ekibinin nasıl organize edileceği, projelerin nasıl planlanacağı, süreç yapısının nasıl gerçekleştirileceği, risklerin değerlendirilip kurulumun nasıl tamamlanacağıyla ilgili bir süreç yaklaşımıdır. MSF takım ve süreç olmak üzere iki modelden oluşmaktadır [11].

4. PROJE ARKA PLANI

Bu bildirinin konusu, kurum dâhilinde gerçekleştirilen üniversite öğrenci işleri otomasyon sistemidir. Bu sistem standart öğrenci işleri otomasyonlarına ek olarak hazırlık bölümünün ve enstitülerin iş akışlarını ayrıntılı bir şekilde takip etmektedir.

Üniversite üst yönetimi tarafından sistemin web tabanlı olması, modüler olması ve ilişkisel veri tabanı üzerinde çalışması gerekliliği temel teknik özellikler olarak istenmiştir.

Bu proje, çevik yazılım geliştirme ve web uygulamalarında deneyimli bir proje yöneticisi ile bu konularda deneyimi olmayan iki yazılımcı ve bir analistle başlatıldı. Sonrasında projenin ikinci ayından itibaren ekibe bir yazılımcı daha ilave edildi.

Belirlenen temel gereksinimler üzerinden yazılım ekibinin deneyimleri de göz önünde bulundurularak en uygun mimari belirlendi. Mimari yapı belirlenirken, veritabanı, veri erişim katmanı, iş katmanı, XML web servisi ve web uygulaması olarak beş katmalı bir yapı modellenmiştir. Ayrıca katmanlar kendi içlerinde uygulamanın işlevlerine uygun olarak modüllere ayrılmıştır. Üniversitenin elinde bulunan mevcut lisanslar ve ekibin yetenekleri göz önünde bulundurularak; Microsoft Visual C#, programlama dili, Microsoft SQL Server 2005 ilişkisel veritabanı ve Microsoft Team Suite 2008 yazılım geliştirme ortamı kullanılmıştır.

Projenin ilk haftasında proje yöneticisi tarafından ekibe yoğunlaştırılmış bir SCRUM + XP eğitimi verilerek projeye başlandı. Bununla birlikte diğer çevik metodolojilerinden alınan pratikler de anlatıldı ve nasıl uygulanacağı belirlendi.

Ardından ikinci bir haftada da ekibe kullanılacak yazılım geliştirme ortamıyla ilgili eğitim verildi ve örnek uygulamalar yapıldı.

Ekip kısmen hazır olduktan sonra müşteri (üniversite yönetimi) gereksinimleriyle ilgili proje hazırlık safhasına başlandı. Bu safhada müşteri gereksinimleri ana hatlarıyla tespit edildi ve müşteriyle birlikte önceliklendirildi. Ekip, yapılması gereken işler için genel bir zaman kestirimi çıkardı ve projenin birinci versiyonu için kabaca 6 aylık bir geliştirme süresi belirledi. Proje yöneticisi ekiple ve müşteriyle yaptığı

toplantılar sonrasında 3'er haftalık yinelemelerle ilerlemeye karar verdi ve sürüm planlamasının müşteri tarafından onaylanmasının ardından ilk yineleme planlama toplantısına oturuldu. Yinelemelerin ilk fazlarında her gün 1 saat ekibin eğitimine ayrıldı. Eğitime ayrılan süre ilerleyen yinelemelerde haftada 2 saate kadar düşürüldü.

5. PROJE SÜRESİNCE KARŞILAŞILAN ZORLUKLAR, ÇÖZÜMLEME GİRİŞİMLERİ VE ÖNERİLER

Projeye başlamadan önce değişik çevik metotlar gözden geçirilmiş ve projenin büyüklüğü, gereksinimlerin karışıklığı, sonuçta ortaya çıkan yazılımın kapsamına ve ekibin yapısına uygun görünen pratiklerin belirli bir çerçeve içinde uyarlanmasına (tailoring) karar verilmiştir. Seçilen bu pratikler üç paragrafta incelenmiştir. Buna göre her bir pratiğin, ilk paragrafında pratiğin karakteristiği ve seçilme nedeni, ikinci paragrafında proje içersinde nasıl uygulandığı, üçüncü paragrafında uygulama aşamasında karşılaşılan sıkıntılar ve bu sıkıntıları çözümleme çabaları, son paragrafında ise bu pratiği uygulayacaklar için öneriler anlatılmaktadır.

5.1. Müşteri Memnuniyeti

Müşteri memnuniyeti, çevik metodolojilerin en önemli pratiklerinden biridir. Bu pratikte, proje ekibi yazılım projesinin başlangıcından itibaren düzenli aralıklarla çalışan programlar oluşturarak bunları müşteriye sunar. Müşteri de bu çalışan programlarda gerekli duyduğu değişiklikleri talep eder. Böylelikle yüksek oranda müşteri gereksinimleriyle örtüşen bir yazılım geliştirilmiş olur. Bu pratiği seçerken müşteri gereksinimleri doğrultusunda müşteriyi tatmin eden bir yazılım projesinin oluşturulması amaçlanmıştır.

Ekip üyeleri, müşteri memnuniyetinin önceliğini kayıtsız şartsız kabul etmiş ve atılan her adımda bu önceliğe göre iş yapmıştır. Düzenli aralıklarla çalışan programlar üniversite yönetimine sunulmuş ve üniversite yönetiminden gelen geri dönüşlere göre öncelikler belirlenerek, gerekli düzenlemeler yapılmıştır.

Ekip üyelerinin zaman zaman teknik yönleri baskın çıkmış ve müşteriye teknik olarak daha iyisini verme güdüsüyle hareket etme çabaları da olmuştur. Böyle durumlarda proje yöneticisi hem ilgili kişiyi hem de ekibin tamamını 5-10 dakikalık kısa toplantılarla müşteri önceliği konusunda yeniden bilgilendirmiştir.

Yazılım projelerinde esas olan müşteri gereksinimleridir. Bu noktada ekip üyeleri, projenin başında yeterince bilgilendirilmeli ve eğitilmelidir. Proje ekibinin görevi müşterinin ihtiyacını tam olarak karşılayan yazılımı geliştirmektir.

5.2. 5-10 Kişilik Takımlar

Bu pratik SCRUM metodolojisinden alınmıştır. Bu pratikte, proje süresince ekip üyelerinin sürekli olarak birbirleriyle iletişim halinde olmaları gerekmektedir. Bu iletişimde oluşabilecek problemleri ortadan kaldırmak amacıyla ekibin 5-10 kişiden oluşturulması öngörülür [12]. Bu pratiği seçerken ekip üyeleri arasında oluşabilecek iletişim problemlerinin en aza indirgenmesi amaçlanmıştır.

Gerçekleştirilen projede ekip, proje yöneticisi dâhil 4 kişiyle başlamış ardından üçüncü yinelemeden hemen önce ekibe 1 kişi daha ilave edilmiştir.

Ekibe yeni katılan kişi nispeten araçlar ve teknolojiler konusunda deneyimli olduğu için proje yöneticisiyle 1 haftalık bire-bir eğitim ve çalışmayla üçüncü yinelemeye aktif olarak dâhil edilebilmiştir.

Proje süresince ekibe yeni bir üye katılacaksa yineleme süreleri dikkate alınmalıdır. Yineleme geçişlerinde ekibe katılacak olan kişinin, araçlar ve teknolojiler konusundaki deneyimi göz önünde bulundurularak eğitim veya adaptasyon süresi belirlenmelidir.

5.3. Yinelemeli/Artırımlı Süreçlerle Geliştirme

Tüm çevik metodolojilerde yer alan bu pratikte, her bir yineleme 2-4 hafta arasında icra edilir [13]. Bir yineleme sonunda üzerine yeni özellikler eklenen ürün, müşteriye teslim edilir ve geri bildirimler alınır. Bu pratikle yinelemeli süreçlerle yazılım geliştirerek müşteri gereksinimleriyle örtüşen bir yazılımın geliştirilmesi amaçlanmıştır.

Müşteri ve ekip üyeleri arasında yapılan görüşmeler sonucunda, proje için en uygun yinelemenin 3 hafta olacağına karar verilmiş, tüm tarafların onay ve taahhütleri alınmıştır.

Uygulama aşamasında bu pratikle ilgili yaşanan en önemli sorun, önceki yinelemelerde çıkan yazılım hatalarını düzeltmek için aktif yineleme içinde zaman ayırma gerekliliği olmuştur. Başlangıçta yineleme planı içinde yer almayan bu unsur, yazılım geliştirme sürecinin ortalarına doğru iterasyona ayrı bir iş kalemi olarak eklenmiş ve gerekli zaman ayrılarak, yineleme planının daha doğru şekilde oluşturulması sağlanmıştır.

Bir yazılım projesinde yineleme süresi proje kapsamına, müşteri gereksinimlerine ve daha da önemlisi iş ürünlerinin büyüklüğüne bağlıdır. Bu faktörler göz önünde bulundurularak anlamlı bir artırım sağlayacak bir süre belirlenmelidir.

5.4. Küçük ve Kısa Aralıklı Sürümler (Release)

Bu pratik XP ve SCRUM metodolojilerinden alınmıştır. Bu pratikle proje, birbirinden ayrı zaman aralıklarına (2-4 hafta) bölünür. Her bir zaman

aralığında yapılacak işin kendine ait son teslim tarihi vardır. Belirlenen son teslim tarihini aşmayarak iş bitirilir ve müşteriye teslim edilir[6][12]. Bu pratikle müşterinin, projenin ilerleyişini her bir yineleme sonrasında özellikleri artan, yaşayan bir uygulamayla takip edebilmesi amaçlanmıştır.

Proje sürümlerinden ilki 4, ikincisi 3 ve sonuncusu ise 2 yinelemeden oluşan 3 sürüm planlanmıştır.

Bu pratik uygulanırken herhangi bir sorunla karşılaşılmamıştır.

Sürüm sayısı planlanırken müşteri gereksinimleri ve zaman kısıdı göz önünde bulundurulmalıdır.

5.5. Kısa Süreli Proje

Bu pratik SCRUM metodolojisinden alınmıştır. Bu pratikle proje süresinin 1 yılı geçmemesi hedeflenir [12].

Projenin ilk tahminlerde 6 ay süreceği öngörülmüş, ancak süreç içerisindeki eklenen ve çıkarılan işlevlerin ardından 7 ay içinde tamamlanmıştır.

Üniversite yönetimi yazılım geliştirme sürecinde, yeni işlevler eklenmesine ve proje kapsamı dâhilindeki bazı işlevlerin çıkarılmasına karar vermiştir. Bu kararlar proje süresinin değişmesine neden olduğundan üniversite yönetiminin onayıyla proje süresi uzatılmıştır.

Projenin geliştirilmesi esnasında isterler değişmeyecek büyüklükte olmalıdır. Böylece başarıyla sonuçlandırılan kısa süreli projeler hem yönetimin, hem kullanıcıların hem de yazılım ekibinin motivasyonunu ve inancını artırır. Aynı zamanda ortaya çalışan bir uygulama çıkar ve hedeflenen büyük projenin bir kısmı bile olsa, hayata geçmiş olur.

5.6. Genel Bir Model/Mimari Geliştirme

Bu pratik DSDM metodolojisinden alınmıştır. İsterler ve kısıtlar ana hatlarıyla belli olduktan sonra projenin tamamı için genel bir mimari belirlenir. Bu mimari daha sonra her bir yineleme safhasında detaylandırılır. Pratiği seçmemizin nedeni, projenin genel yapı ve kapsamını önceden görebilmektir.

Uygulamanın tamamı için beş katmanlı ve her bir katman içinde işlevlere bağlı modüllere ayrılmış bir genel mimari oluşturulmuştur. Katmanlar sırasıyla en arka uçta ilişkisel veritabanı ve veritabanı programlama nesneleri, ardından veritabanı erişim katmanı, iş katmanı, XML Web Servisi ve Web Uygulaması şeklinde yapılandırılmıştır. Genel çözüm modeli ve mimari teknoloji seçimiyle birlikte müşteriye sunularak onay alınmıştır. Ardından mimari ve kullanılacak teknolojiler ekibe tanıtılmış ve tartışılmıştır.

Genel bir mimari model geliştirirken karşılaşılan en önemli sıkıntı, ekibin çok katmanlı mimari modelle ve servis temelli yaklaşımla çalışmamış olmasıydı. Ancak modülerliği ve müşteri isterlerini karşılamak amacıyla mimari model, ekibin yeteneklerinden ziyade projenin gereksinimlerine en uygun şekilde tasarlanmıştır. Ekibin mimariye uyum sağlaması için yoğun bir eğitim ve örnek uygulama programı hazırlanmış ve uygulanmıştır.

Müşteri gereksinimleri ve kısıtları ana hatlarıyla belli olduğunda, projenin geneli için bir mimari ve altyapı tasarımı oluşturmak projenin başarımına büyük oranda katkı sağlamaktadır. Bu noktada müşterinin sahip olduğu donanım, lisans vb. faktörlerin göz önünde bulundurulmasında fayda vardır.

5.7. Teknolojiyi Gereksinimlerin Belirlemesi

Bu pratik Lean metodolojisinden alınmıştır. Öncelikli olarak uygulama sahasına yönelik gereksinimler belirlenir. Daha sonra bu gereksinimlere uygun teknolojiler seçilir [1]. Bu pratik, müşterinin istediği projenin, bütçesi dâhilinde gerçekleştirilebilmesi için önemlidir.

Bir önceki pratikte belirtilen genel modeli geliştirebilmek için müşterinin sahip olduğu lisanslar, kullanıcı alışkanlıkları, ekip deneyimi gibi kısıtlar göz önünde bulundurularak teknoloji tercihleri belirlenmiştir. Mevcut durum gereği yazılım geliştirme için Microsoft Visual Studio Team Suite, ilişkisel veritabanı olarak Microsoft SQL Server 2005 kullanımı tercih edilmiştir. Uygulamanın Web sunucusu katmanı Asp.Net ile programlanmış, iş ve veri erişim katmanları SOA (XML Web Servisleri) ile servis edilmiştir.

Yazılımın analiz aşamasında uyguladığımız bu süreçte karşılaştığımız sorun, müşterinin uygulamanın tüm arayüzlerinin Web tabanlı olmasını istemekle birlikte, daha önceden alışık olunan masa üstü yazılım seviyesinde bir performans beklemesiydi. Sorunu çözümlemek için müşteri bu iki teknoloji hakkında detaylı olarak bilgilendirildi ve her bir sayfa için en yüksek bekleme süresi olarak özel sayfalar ve raporlar hariç 4 saniyede karar kılındı.

Teknoloji belirlenmeden önce müşteri gereksinimleri ve kısıtları ana hatlarıyla belirlenmeli ve genel mimariye en uygun teknolojiler seçilerek müşterinin beklentilerinin bu şekilde ne kadar karşılanabileceği anlatılmalı, sonra da müşterinin onayı alınmalıdır. Teknoloji seçimi yaparken göz önünde bulundurulması gereken bir diğer faktör de ekibin bilgi düzeyi ve deneyimidir.

5.8. Sürekli Günlük Entegrasyon

Bu pratik XP ve MSF metodolojilerinden alınmıştır. Yazılım geliştirilirken yapılan sistem değişiklikleri ve yeni bileşenler hemen sisteme entegre edilerek günlük

derlemelerle test edilir. Sürekli entegrasyon sayesinde programcıların sistem üzerinde yapılan değişiklikleri görmeleri ve oluşabilecek hataları erken tespit edebilmeleri sağlanır [6][11][14].

Uygulama geliştirme için tüm ekip üyelerine Micsoroft Visual Studio Team Suite kurulmuştur. Kodları ortak bir noktadan yönetmek ve eş zamanlı ekip çalışmasını kontrol edebilmek için de Microsoft Team Foundation Server kullanılmıştır.

Kullanılan yazılım geliştirme ortamı (Microsoft Visual Studio Team Suite ve Microsoft Team Foundation Server), kodları tek bir yerde tutma ve günlük otomatik olarak derleme özelliklerine sahip olduğu için bu pratiğin uygulanmasında teknik herhangi bir sıkıntıyla karşılaşılmamıştır. Ancak araçların kullanımına yabancı olan ekibin eğitilmesi gerekmiştir.

Teknoloji seçiminde bütçe kısıtlarının el verdiği ölçüde otomatik entegrasyon sağlayan ürünler tercih edilmelidir.

5.9. Kodlama Standartları

Bu pratik XP metodolojisinden alınmıştır. Ekip üyeleri önceden tanımlanmış kodlama standartlarına göre yazılımı geliştirirler [6]. Bu pratiği seçerken, yazılan kodun karmaşıklığını azaltarak, bütün ekip üyeleri tarafından kolaylıkla anlaşılabilmesi amaçlanmıştır.

Proje başlamadan önce ekiple beraber kodlama, isimlendirme ve kod yorumları yazımı vb. üzerine standartlar belirlenip, belgelenmiştir. Proje boyunca ekip üyelerinin bu standartlara uyması istenmiştir.

Ekibin kodlama standartlarına başlangıçta dikkatlice uymasına rağmen, zamanın daralmasından dolayı bazı noktalarda standartlara uymadığı gözlenmiştir. Bu noktada, projenin önceliği belirlenen işlevleri zamanında ve çalışır halde teslim etmek olduğu için bu sapmalar göz ardı edilmiştir.

Proje geliştirme süresince ekibin mutlaka bir kodlama standardına uyması gereklidir.

5.10. Planlama Oyunu

Bu pratik XP metodolojisinden alınmıştır. Planlama oyunu, müşterinin belirlediği her bir yineleme için yazılım ekibinin müşterinin de içinde bulunduğu bir toplantıda, o işin ne kadar zamanda yapılacağıyla ilgili kestirimde bulunmasıdır. Böylece müşteri bir sonraki yinelemede hangi işlerin yapılacağına karar verecek ve yazılım ekibi de belirttiği sürelere mümkün olduğunca bağlı kalarak, bu işleri ilgili yinelemede yapacaktır [6].

Uygulamada çoğunlukla proje yöneticisi önce geliştirilecek modülle ilgili yöneticiyle toplantı yaptı, sonra toplantıdan elde edilen bilgiler doğrultusunda ekiple birlikte "backlog" lar oluşturuldu.

Müşteriyle birlikte bir sonraki yinelemede yapılacak işlerin belirlendiği bu süreçte ekibin görevi en doğru kestirimleri yapmaktı. Planlama oyununda karşılaşılan en büyük iki sıkıntı ekibin doğru kestirimler konusunda yetersiz oluşu ve müşterinin gereksinimlerini aktarmada iş süreçleri yazılı ve net olmadığı için tam bilgi aktaramayışıydı. Kestirim hataları özellikle ilk yinelemelerde planlanandan daha az "backlog" tamamlanmasıyla sonuçlandı. Eksikliğin tamamlanması için ve daha sonradan bildirilen modül güncellemeleri için fazla mesai yapılmak zorunda kalındı.

Doğru kestirim tecrübeyle doğrudan ilişkili bir özelliktir. Planlama oyununda başarı için tecrübeli eleman faktörü çok önemlidir. Ekibin mümkün olduğunca, yapılacak işi ne kadar zamanda yapabileceğini bilen elemanlardan oluşturulması gerekmektedir. Ayrıca gereksinimlerin zamanında ve tam olarak alınabilmesi için, yazılım sürecinin başlamasından önce tam kapsamlı bir iş analizinin yapılması, yazılımın süresini kısaltacak ve kalitesini artıracaktır.

5.11. Yineleme İçerisinde Takım Dokunulmazdır

Bu pratik SCRUM metodolojisinden alınmıştır. Bu pratiği almamızın nedeni, belirli bir modüle konsantre olmuş takımın dikkatini dağıtmadan uygulama geliştirebilmesini sağlamak istememizdir. Bu özellik hem motivasyon hem de yazılımın kalitesi için önem arz etmektedir.

Pratiği uygulanma aşamasında proje yöneticisi müşteriyle gerekli görüşmeleri yapıp, sonraki sprint için gerekli bilgileri toplarken, gereksinimlerin devam eden sürece sızmasını engelleme yönünde çaba göstermiştir.

Müşteri tarafından genellikle bu pratiğe uyulmuştur. Ancak müşterinin tam olarak ne istediğini bilmemesinden dolayı, gereksinimleri hatalı belirtmesiyle ilgili olarak yaşanan bir istisna, yeni bir yinelemeye başladıktan sonra, önceki yinelemelerle ilgili kritik hata geri bildirimlerinin bir an önce çözümlenmesi gereksinimi olmuştur. Bu durumla ilgili alınan iki önlem; daha az hatalı kodlamaya teşvik ve yinelemelerde %20'lik bir zaman dilimini hata ayıklamaya ayırmak şeklinde gerçekleştirilmiştir.

Yineleme gerçekleştirilirken ek gereksinimler talep edilirse, bu gereksinimler kesinlikle yinelemeye dâhil edilmeyip bir sonraki yinelemeye bırakılmalıdır. Yineleme gerçekleştirilirken istisnai durumlar dışında ekipten üye çıkarılmamalıdır ya da eklenmemelidir. Ayrıca bir önceki yinelemede hata çıkma olasılığı göz önünde bulundurularak hata ayıklamaya makul bir zaman ayrılmalıdır. Bu zamanı en aza indirebilmek

için kod gözden geçirmeleri ve mümkünse eşli programlama yapılmalıdır.

5.12. Müşteriyle Birlikte Geliştirme

Bu pratik XP, MSF ve DSDM metodolojilerinden alınmıştır. Çevik projelerde müşteri ile proje ekibinin beraber çalışması kaçınılmazdır. Programdan istenilenleri en iyi müşteri bileceği için, sürekli müşteriden geri bildirim alınması gerekmektedir. Bunun en kolay yolu müşteri ile proje ekibinin beraber çalışmasıdır [12]. Müşteri ile proje ekibi arasındaki etkileşimi sağlayan ürün yöneticisidir. Ürün yöneticisi müşteri tarafında çalışan, müşterinin isteklerini anlayan ve bunu proje ekibine anlatan kişidir. Bu sayede projenin müşteriyle birlikte geliştirilmesi, gereksinimlerin doğru şekilde anlaşılması ve uygulanması sağlanmış olur.

Pratiğin uygulanması için sık sık geliştirilecek modüllerle ilgili kişiler ekiple toplantıya davet edilmiştir. Başlangıçta birkaç toplantı pratiğe uygun yapılabilse de, sonraki toplantılar da pratiği aslına uygun şekilde işletmek mümkün olmamıştır.

Proje boyunca yaşanan sıkıntılardan biri de müşterinin aktif olarak takımla birlikte olamayışıydı. Ancak müşteri yineleme planlamada ekibe dâhil olmuş, sonrasında yineleme boyunca ya süreçteki sorulara cevap vermiş ya da cevap verebilecek kişilere yönlendirmiştir.

Müşterinin ekiple birlikte çalışmaması bazı noktalarda yavaşlamaya neden olabilir. Bu durumun önüne geçebilmek için projenin başında müşterinin ikna edilmesi ve sürekli ekiple çalışabilecek yetkin birinin atanmasının sağlanması önemlidir.

5.13. Açık İletişim ve Ortak Dil

Bu pratik MSF ve DSDM metodolojilerinden alınmıştır. Bu pratikte projenin başarılı olabilmesi için ekip üyeleri arasında açık ve samimi bir iletişimin olması gerekmektedir. Bu pratiği seçerken, ekip üyelerinin projeyle ilgili endişelerini, yetersiz ve deneyimsiz olduğu alanlarla ilgili düşüncelerini diğer ekip üyeleriyle paylaşabilmesi amaçlanmıştır. Ekipteki iletişim eksikliği proje işlevlerinde eksikliklere yol açabilir. Bu problemin önüne geçilmesi için tüm ekibin aynı jargonu konuşması gerekir.

Ekip her toplantıda ve her fırsatta açık olmaya ve sürekli iletişime teşvik edilmiştir. Projenin başlarından ortalarına kadar açık iletişim yeterince yapılamamıştır.

Ekipteki kişilerin karakteristik özellikleri ve çevik metodolojiyle ilk defa tanışmaları nedeniyle en çok sıkıntı yaşanan pratik bu olmuştur. Ekip üyeleri arasında ki iletişim problemlerinin büyük bir kısmını yanlış anlaşılmalar oluşturmuştur. Bu yanlış anlaşılmalar belirli bir oranda düzeltilene kadar yazılı iletişim yolu kullanılmak zorunda kalınmıştır.

Projede ekip üyelerinin karakteristik özelliklerinden kaynaklanan yanlış anlaşılmalar varsa; ya bu projede uygulandığı gibi yazılı iletişim seçilebilir ya da ekip üyesine bir iş verildiğinde bu işin doğru anlaşıldığını kontrol etmek için proje yöneticisi geribildirim isteyebilir.

5.14. Ortak Vizyon

Bu pratik MSF metodolojisinden alınmıştır. Ortak vizyon, çözümü elde etmek için yapılması gerekenleri açık bir şekilde anlamaktır [15]. Bu pratikle hem ekip üyelerinin hem de müşterinin geliştirilecek olan yazılımın işleyişi hakkında hemfikir olması amaçlanmıştır.

Projeyle ilgili müşteri tarafından belirlenen vizyon ekip tarafından tamamıyla sahiplenilmiştir.

Bu pratik ile olarak proje süresince hiçbir sorun yaşanmamıştır.

Ekip, proje vizyonunu mutlaka benimsemelidir.

5.15. Ortak Kod Sahiplenme

Bu pratik XP metodolojisinden alınmıştır. Geliştirilen yazılım kodu, bütün ekip üyelerinin ortak malıdır. Bu pratikle bir ekip üyesinin belirli kurallar çerçevesinde herhangi bir ekip üyesinin ürettiği kodlara erişerek, mevcut yazılımı daha iyiye götürme adına değişiklikler yapabilme imkânına sahip olmaları amaçlanmıştır.

Ekip üyelerine katmanlı mimaride yazılan projenin her katmanında dönüşümlü olarak görevler verilerek projeye hakim olması sağlanmıştır.

Başlangıçta ekibin ortak kod sahiplenmeyi kabullenmesine rağmen süreç içinde çeşitli anlaşmazlıklar yaşanmış, proje yöneticisinin müdahaleleriyle bu anlaşmazlıklar giderilmiştir.

Ekip üyelerinin her biri, kodun herhangi bir yerini gerektiğinde değiştirebilmelidir. Ekip içinde tüm kararlar ve tasarımlar birlikte ele alındığı için, herkes birbirinin kodunu sahiplenmelidir.

5.16. Kendi-Kendini Düzenleyen, Yönlendiren, Motivasyonu Yüksek Takım

Bu pratik SCRUM metodolojisinden alınmıştır. Yazılım projesini gerçekleştiren ekip üyelerinin farklı kişilik ve iş yeteneklerine sahip kişilerden oluşması doğaldır. Ekip, motivasyonu ve iletişim becerisi yüksek, olumlu düşünce yapısına sahip, disiplinli ve kararlı kişilerden oluşturulmalıdır.

Ekip üyelerinin motivasyonunu yüksek tutmak ve kendi öz motivasyonları ile çalışmalarını sağlamak

için, kodlama ayrıntılarına müdahale edilmemiş ve yapacakları işleri (alt modülleri) kendilerinin belirlemesi sağlanmıştır. Aynı zamanda ekip içinde yardımlaşmanın gerekliliği ve önemi sürekli olarak vurgulanmış ve cesaretlendirilmiştir.

Ekip daha önceki bireysel uygulama geliştirme tecrübelerinden dolayı motivasyonu yüksek elemanlardan oluşmaktaydı. Ancak ekip olarak çalışma konusunda zaman zaman sıkıntılar yaşandı. Bu tip sorunlar proje yöneticisinin motive edici ve yönlendirici yaklaşımlarıyla çözülmeye çalışıldı.

Ekip ruhu oluşturmak ve motivasyonu yüksek tutmak için sosyal faaliyetler düzenlenmeli, ekip içindeki uygunsuzluklar ve çatışmalar olabildiğince erken fark edilerek çözümlenmelidir.

5.17. Risk Yönetimi

Bu pratik MSF metodolojisinden alınmıştır. Projenin başından sonuna kadar karşımıza çıkabilecek, gerek projenin iptal olmasını gerekse de projenin yavaşlamasına yol açacak, bütün riskler belirlenir ve bu riskler önem sırasına göre sıralanır. Risk yönetim planı hazırlanarak, proje süresince oluşabilecek olağan dışı durumlara karşı önlemler alınmış olunur [17].

Projenin hazırlık safhasında muhtemel riskler tespit edildi. Bu risklerden bazıları müşteri onayı alınarak proje kapsamı dışında tutuldu. Geri kalan riskler içinse önleyici tedbirler almak yerine risklerin gerçekleşmesi göze alındı. Bunun nedeni risk gerçekleşirse, düzeltmek için önleyici eylemlerin maliyetinin daha düşük olacağının tespit edilmesiydi.

Bu pratikle ilgili karşılaşılan en önemli zorluk, projenin isterlerini her bir modül için ayrı biri hatta bazen birden fazla kişiyle çalışılması nedeniyle riskleri toplu olarak ön görmenin sorumluluğunun tamamen proje yöneticisi üzerinde olmasıydı. Bu noktada proje yöneticisi önceki yazılım projesi deneyimlerini de kullanarak olası tüm riskleri masaya yatırdı.

Proje hazırlık safhasında mutlaka riskler tespit edilmelidir. Tespit edilen risklerle ilgili olarak müşteri ile toplantı yapılmalı ve hangi risklerin proje kapsamı dışında tutulacağına karar verilmelidir. Kapsam dışında tutulmayacak riskler varsa, bu risklerin önceliğini yüksek tutarak önleyici tedbirler alınmalıdır.

5.18. Günlük Toplantılar

Bu pratik SCRUM metodolojisinden alınmıştır. Yinelemenin başında belirlenen yer ve saatte, günlük olarak 15 dakikalık ayaküstü toplantılar yapılır. Toplantılarda bir önceki gün neler yapıldığı, o gün içinde neler yapılacağı ve ekibin varsa ihtiyaçları ve karşılaştıkları güçlükler konuşulur [12]. Bu pratikle ekip üyelerinin birbirleriyle eş zamanlı çalışması ve sorunların çok hızlı bir şekilde çözülmesi sağlanır.

Proje boyunca her sabah saat 9.00-9.15 arasında günlük toplantılar yapıldı. Bu toplantılarda genel olarak bir önceki gün yapılanlar ve o gün yapılacaklar konuşuldu. Ayrıca ekibin varsa karşılaştığı zorluklar ve sıkıntılar tartışıldı. Proje yöneticisi bu sıkıntılarla ilgili olarak çözüm önerileri getirerek gerekli yönlendirmeleri ve takviyeleri yaptı.

Toplantılar başlangıçta SCRUM pratiklerine uygun olarak ayakta yapılsa da ekip buna pek alışamadı. Bir süre sonra ekibin yerleşiminin de uygunluğu nedeniyle toplantılar oturularak yapılmaya başlandı.

Günlük toplantıların düzenli yapılması ve belirli bir süreyi aşmaması önemle tavsiye edilir. Bizim tecrübelerimize göre, pratiğinde önerdiği gibi, bu toplantılar için 15 dakika uygun bir süredir. Bu toplantılarda ekip özellikle geliştirme süreciyle ilgili sıkıntılarını paylaşma yönünde cesaretlendirilmelidir. Ekibin tamamen toplantıya odaklanması için eğer imkân varsa toplantının çalışma ortamı dışında olması tavsiye edilir.

5.19. Kabul Testi

Kabul testi kullanıcıdan alınan gereksinimlerin "backlog" olarak tanımlanma safhasında, işin gerçekleşme kriterini ve bunun nasıl test edileceğinin tanımlanmasıyla başlar. Ardından yazılım ekibi tarafından geliştirilen işlev müşteri tarafından test edilir, varsa eksiklikler söylenir veya işlev gerçekleştirilmiş olarak kabul edilir. Müşterinin istediğini elde edip etmediğini ölçmek için en etkili yollardan biridir.

Proje boyunca kabul testleri oluşturulmuş, bu testler hem ekip içinde alfa testlerinde hem de müşteri tarafından kabul testlerinde kullanılmıştır.

Bu konuda yaşanan en büyük sıkıntı, müşterinin analiz aşamasında kabul testlerini yeterince tarif edememesinden kaynaklanmıştır. Bu nedenle test aşamasında geliştirilen işlevlerle ilgili bazı yetersizlikler olduğu ileri sürülmüş, ancak bunlar başlangıçta kabul testleri tanımında yer almadığı için ekip tarafından ek talep olarak değerlendirilmiştir.

Kabul testleri, proje ve ekip başarısının müşteri tarafından onaylandığı en önemli işlemdir. Bu yüzden projenin başında bu testler mümkün olduğunca ayrıntılı tanımlanmalı ve proje sürecindeki değişiklikler mutlaka kayıt altına alınmalıdır. Ayrıca kabul testi yapıldıktan sonra müşteri "tamamdır" dediğinde testi yapanın ve mümkünse bir yetkilinin imzası alınmalıdır.

5.20. Önce Test
Bu pratik XP metodolojisinden alınmıştır. Programın kodlanmasına başlamadan önce, birim testleri için test kodları yazılır [12]. Bu pratik geliştirilen programın kalitesini yüksek tutmak amacıyla alınmıştır.

Projede kullanılan yazılım geliştirme ortamı, birim testlerini, bütünleşik yazılım geliştirmeyi ve derlemeyi desteklediği için bu pratik önce testi kodla, sonra işlevi geliştir mantığıyla kodlanmıştır.

Ancak projenin sonlarına doğru zaman darlığı nedeniyle hata çıkma riski göze alınarak birim testinden çok, işlev geliştirmeye zaman ayrılmıştır. Kalitenin korunması için yazılan kodlar gözden geçirmeyle doğrulanmaya çalışılmış ve kalitenin korunması için çaba gösterilmiştir.

Sonraki yinelemelerde hatalarla uğraşmamak ve kaynakların daha verimli bir şekilde kullanılmasını sağlamak amacıyla birim testlerinin mümkün olduğunca yapılmasında fayda vardır.

5.21. Basit Tasarım
Bu pratik XP metodolojisinden alınmıştır. Bu pratikle, müşterilerin gereksinimlerini karşılayacak en basit tasarım gerçekleştirilir. Böylece kısa sürede anlaşılması, yönetilmesi ve değiştirilebilmesi kolay bir yazılım geliştirilmesi sağlanır [6][12].

Genel mimari üzerine yapılacak olan her bir işlev için, müşteri beklentisini karşılayan minimum tasarımla ilerlenmeye çalışılmıştır.

Bu pratik çoğunlukla ekip tarafından uygulansa da bazen ileriye doğru müşterinin değişebilecek gereksinimleri göz önünde bulundurularak kodlama eğilimi gösterilmiştir. Böyle durumlarda proje yöneticisi devreye girerek sonraki versiyonlar için konuyla ilgili not alıp, daha basit bir tasarımla ilerlenmesini istemiştir.

Ekip içinde mümkün olduğunca basit tasarıma önem verilmeli ve müşterinin beklentisinden daha fazlası için kaynak harcanmamalıdır.

5.22. Zamanında Teslim
Bu pratik DSDM ve SCRUM metodolojilerinden alınmıştır. Ekip üyeleri her iki veya dört haftada bir müşteriye çalışan bir program teslim eder. Ürünün teslim zamanı değiştirilemez fakat zamanında teslim etmek için teslim edilecek işlev/iş sayısı azaltılabilir [12].

Proje boyunca tüm çalışma zaman kutusuyla (time boxing) sınırlandırılmıştır. Böylece her bir kabul testi belirlenen zamanlarda gerçekleştirilmiştir.

Proje ekibinin kestirim hataları ya da önceki yinelemelerden gelen hataların çözümlemeleri nedeniyle tamamlanamayan işlevler olmuştur. Bu işlevler sonraki yinelemelere devredilmiştir.

Yinelemeler kesinlikle uzatılmamalı ve tamamlanmayan işler bir sonraki yinelemeye bırakılmalıdır. Zamanlama kutusuna mutlaka sağdık kalınmalıdır.

5.23. Sürekli Gelişim
Bu pratik MSF metodolojisinden alınmıştır. Ekip üyeleri her zaman değişime ve yeniliğe açık olarak, ortaya çıkan hatalardan ders alırlar ve aynı zamanda aralarındaki bilgi paylaşımıyla da kendilerini sürekli geliştirirler. Böylece her geçen gün daha da çok şey öğrenen ve kendini geliştiren birey ekibe ve dolayısıyla da projeye daha fazla katkıda bulunur.

Proje süresince hem proje yöneticisinin belirlediği konulardaki eğitimler hem de günlük toplantılarda paylaşılan güçlü ve zayıf yönler üzerine sürekli iyileştirmeler yapılmıştır. Bir taraftan ekip üyeleri bireysel olarak programlamada ve çevik metodolojilerde ilerleme sağlarken diğer taraftan ekip olarak çalışma konusunda da gelişme sağlanmıştır.

Bu konuda yaşanan tek sıkıntı ekip üyelerinin bilmedikleri konularda bilgi eksikliklerini kabul etmemeleri ve gizlemeleri olmuştur. Proje yöneticisi böyle bir durumu fark ettiğinde ilgili kişiye eksiğini fark etmesini sağlayacak bir görev vererek, az da olsa zaman kaybını göze alarak ekip üyesinin eksiğini fark etmesini ve tamamlayıcı aksiyonlar almasını sağlamaya çalışmıştır.

Ekip üyeleri sürekli öğrenme konusunda açık olmalı, eksik ya da hatalarını yeni bir şey öğrenebilme konusunda fırsat olarak görmelidir. Ekip kurulurken bu faktörlere dikkat edilmelidir.

5.24. Az da Olsa Belgele
Bu pratik Lean metodolojisinden alınmıştır. Çevik programlamada az da olsa belgeleme vardır. Kapsamlı belgelemeyle uğraşıp zaman kaybetmek yerine amaca uygun olarak sadece gerekli görülen yerlerde belgeleme yapılır [18]. Gerektiğinde raporlama amacıyla kullanmak ve ekip içinde isterler ve projenin akışıyla ilgili gerekli bilgiyi sağlamak amacıyla belge oluşturulmuştur.

Çevik metodolojilerin doğası gereği detaylı belgelendirme yapılmamış, ancak hem mimari hem kodlama tarafında mutlaka yeterince belgeleme yapılmıştır. Özellikle kodlama standartlarıyla getirilen kurallardan kod açıklama satırları, yazılan kodla ilgili gerekli belgelemeyi sağlamaktadır. Aynı şekilde veritabanı tarafında da hem yerleşik yordamlarda hem de tablolarda gerekli açıklamalara yer verilmiştir.

Belgelemeyle ilgili karşılaşılan zorluk, üniversitedeki birimleri yazılı iletişimden çok işleri sözlü iletişimle

yapma yönündeki ısrarı olmuştur. Bu zorluğu çözmek için gerekli bilgiler ve bildirimler kısa e-postalar aracılığıyla yapılmıştır.

Projenin başından itibaren tanımlanmış mimari, kodlama standartları ve kabul testleriyle ilgili mutlaka belgeleme yapılması gerekmektedir.

5.25. İş İhtiyaçlarına Odaklanma

DSDM metodolojisinden alınmış bu pratik, daha geç teslim edilecek mükemmel bir yazılım yerine önceden teslim edilmiş iyi bir iş ihtiyacı çözümüne odaklanmayı öngörür. Bu da kritik iş ihtiyaçlarına odaklanmayı gerektirir. Bu noktada her yineleme öncesi müşterinin yapılacak işler için önceliklerini belirlemesi gerekir [19].

Müşterinin belirlediği önceliklendirmeye uyularak kritik gereksinimlere odaklanılmıştır. Genel ihtiyaçlarda da gelecekte yapılma ihtimali olan işleyiş ve işlemlerden çok, şu an gerçekleşen iş süreçlerine ve bu ihtiyaçlara cevap verebilecek işlevler geliştirmeye odaklanılmıştır.

Bu pratikte karşılaşılan zorluk, her projede olduğu gibi müşterinin önceliklerinin beklenmedik durumlar nedeniyle değişmesi yönünde olmuştur. Müşterinin öncelik değişikliği sprintler arasında ekibe yansıtılmamış, yeni bir sprinte başlarken değişen önceliğin de uygulanan metodoloji nedeniyle çok doğal karşılandığı için ekibin ilerleyişine olumsuz bir etkisi olmamıştır.

Öncelikleri belirlemek müşterinin yetkisi dâhilindedir. Bu konudaki sorumluluğu müşterinin üstlenmesi sağlanmalıdır.

6. SONUÇ

Doğru yapılan bir uyarlama ile yazılım gerçekleştirildiğinde, müşteri ve ekip deneyimsiz olsa bile çevik metodolojileri kullanarak başarılı olmanın mümkün olduğu görülmüştür.

Bu çalışmada elde edilen bulgulardan en önemlisi; ekibin teknoloji ve araçları kullanma konusundaki becerilerindeki iyileşmenin, metodolojiyi uygulama konusundaki ilerlemeden çok daha yüksek oluşudur. Bunun nedeniyse genel olarak yeni teknolojileri uygulamaya açık olan yazılımcıların, iş yapış alışkanlıklarını değiştirmeye aynı derecede açık olmayışlarıdır.

Proje sonunda elde edilen bulgulara dayanarak şu sonuçlara varılmıştır:

1. Projede uygulanacak mimari modele ve kullanılacak teknolojilere hakim, çevik metodoloji deneyimi olan kişilerden oluşturulan bir ekip, proje için en ideal olanıdır.

2. Ekip oluşturulurken hem teknolojiye hem de metodolojiye hâkim olan elemanlar bulunamıyorsa, metodolojiye hâkim olanlar öncelikle tercih edilmelidir. Ekibin teknoloji ve mimari model eksikliği proje başında eğitimle tamamlandığında, ideal bir ekip kurulmuş olacaktır.

3. Müşteri ya da müşterinin atayacağı yetkin biri projeye aktif olarak dâhil edilmelidir.

4. Uygun çevik pratikleri belirlerken ekip değil, proje baz alınmalıdır.

5. Müşteri, proje yöneticisi ve ekip tam bir takım olmalı ve herkesin yürekten inandığı ortak bir proje vizyonu oluşturulmalıdır.

Çevik metodolojilerde başarının anahtarı müşteri gereksinimleri, kısıtlar, ekip ve çevik pratiklerin en uygun şekilde bir araya getirilmesidir.

Çevik metodoloji kullanılarak gerçekleştirilecek bir yazılım projesinde esas alınacak nokta, müşteri gereksinimleri ve kısıtlarıdır. Gereksinim ve kısıtları belirleyip odak noktasına aldıktan sonra yapılacaklar hazırda bir ekibin olup olmamasıyla ilgilidir. Eğer hazırda bir ekibiniz yoksa, öncelikle projeye uygun çevik metodoloji pratiklerini, projede kullanacağınız teknoloji ve mimariyi belirleyin. Ardın da tüm bu proje bileşenlerine (çevik pratikler, mimari, teknoloji) en uygun ekibi kurun. Böylece projenin başarı şansını olası en üst seviyeye çıkartmış olursunuz. Eğer ekip önceden belirlenmişse ve ekibe müdahale şansınız yoksa, çevik pratiklerini seçerken proje ihtiyaçları ve kısıtlarının ötesinde ekibin deneyimi, bilgi ve beceri durumlarını da göz önünde bulundurmalısınız.

7. KAYNAKLAR

[1] Highsmith, J., *"Agile Software Development Ecosystems"*, Addison Wesley, 2002.

[2] Boehm, B., Turner, R., *"Observation on Balancing Discipline and Agility"*, Proceedings of the Agile Development Conference, IEEE Computer Society, 2003.

[3] Cockburn, A., *"Agile Software Development"*, Addison-Wesley Longman, 2001.

[4] Agile Manifesto, Mart 2009, http://agilemanifesto.org

[5] Principles Behind the Agile Manifesto, Mart 2009, http://agilemanifesto.org/principles.html

[6] Beck, K., "*Extreme Programming"*, Addison Wesley, 2002.

[7] Schwaber, K., Agile Project Management with Scrum, Microsoft Press, 2004

[8] Agile Unified Process, Mart 2009, http://www.ambysoft.com/unifiedprocess/agileUP.html.

[9] Poppendieck, M., Poppendieck, T., Lean Software Development, Addison-Wesley, 2003

[10] Microsoft Solution Framework, Mart 2009, http://en.wikipedia.org/wiki/Microsoft_Solutions_Framework

[11] Turner, M., Microsoft Solutions Framework Essentials, Microsoft Press, 2006

[12] Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J., "*Agile Software Development Methods, Review and Analysis"*, Espoo 2002, VTT Publications 478. 107 p.

[13] Augustine, S., *"Managing Agile Projects"*, Prentice Hall, 2005.

[14] Fowler, M., Continuous Integration, Mart 2009, http://martinfowler.com/articles/continuousIntegration.html

[15] MSF Principles, Mart 2009, http://www.stromboli.co.uk/msf.aspx

[16] Hayes, S., Andrews, M., An Introduction to Agile Methods, Mart 2009 http://www.scribd.com/doc/4465171/Intro-To-Agile-Methods

[17] MSF Risk Management Discipline v.1.1, Mart 2009,

[18] Software Intensive Systems Lean Development, Mart 2009, http://www.itabhi.com/ld.htm

[19] DSDM Principles, Mart 2009, http://www.dsdm.org/atern/introduction/principles/

ÖZGEÇMİŞLER

Kadir ÇAMOĞLU

1974 yılında İstanbul'da doğmuştur. 1996 yılı Anadolu Üniversitesi İktisat Fakültesi İktisat Bölümü mezunudur. 1995 yılından bu yana çeşitli yazılım projelerinde görev almış, veritabanı ve yazılım geliştirme alanlarında eğitmenlik ve danışmanlık yapmıştır. Veritabanı tasarımı ve yazılım geliştirme üzerine çeşitli yayınevlerinden yayınlanmış altı teknik kitabın yazarı olan Çamoğlu, Maltepe Üniversitesinde Bilgisayar Mühendisliği programında yüksek lisans yapmaktadır.

Derya AKBAYIR

1984 yılında İstanbul'da doğmuştur. 2007 yılı Maltepe Üniv. Bilgisayar Mühendisliği Bölümü mezunudur. Yazılım Mühendisliği, Veri Madenciliği konularıyla ilgilenmektedir. Şuanda Maltepe Üniv. Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans programında tez aşamasındadır. Tez konusu çevik metodolojilerle ilgilidir. Ayrıca ikinci üniversite olarak Anadolu Üniv. İşletme Bölümünde 4. sınıfta okumaktadır. 2007 yılından beri Maltepe Üniversitesi'nde Arş. Gör. olarak akademik çalışmalarına devam etmektedir.

Fatih YÜCALAR

1980 yılında İstanbul'da doğmuştur. 2002 yılında Maltepe Üniversitesi Bilgisayar Mühendisliği Bölümü'nden bölüm birincisi olarak mezun oldu ve 2005 yılına kadar Maltepe Üniversitesi Bilişim Bölüm Başkanlığı'nda Yazılım Uzmanı olarak görev aldı. 2006 yılında Maltepe Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda Yüksek Lisans Eğitimini tamamladı. 2005-2008 yılları arasında Maltepe Üniversitesi Bilgisayar Mühendisliği Bölümü'nde Araştırma Görevlisi olarak yer aldı. Trakya Üniversitesi'nde, Bilgisayar Mühendisliği alanında doktora eğitimini sürdürmekte olup, Maltepe Üniversitesi Bilgisayar Mühendisliği Bölümü'nde Öğretim Görevlisi olarak akademik çalışmalarına devam etmektedir.

Selim BAYRAKLI

1984 yılında İstanbul'da doğmuştur. 2006 yılında Maltepe Üniversitesi Bilgisayar Mühendisliği Bölümünden lisans, 2008 yılında da Maltepe Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Ana Bilim Dalından yüksek lisans derecesi almıştır. Yazılım Mühendisliği ve Veri Madenciliği konularıyla ilgilenmektedir. Şu anda Doktora eğitimine Maltepe Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Ana Bilim Dalında devam etmektedir.