Aşırı Programlama İçin Üç Yeni Pratik

Mustafa Yıldız, Gürol Erdoğan, Selahattin Kuru

Enformatik Uygulama ve Araştırma Merkezi, Işık Üniversitesi, İstanbul
{mustafa, gurol, kuru}@isikun.edu.tr

Özet. Aşırı Programlama, özellikle küçük yazılım geliştirme ekipleri için, isterlerin sıklıkla değiştiği ortamlarda, hızlı yazılım geliştirmeye dayalı yalın bir yazılım geliştirme yöntemidir. Bu yöntem, katı ve izlenmesi zor olan kurallar ve standartlar içeren geleneksel yazılım geliştirme yöntemlerinin aksine, basit ve uygulanması kolay pratiklerden ibarettir. Bu çalışmada Aşırı Programlama için üç yeni pratik ortaya konulmuştur. Bu pratikler; *İş Tabanlı Programlama, Önceden Yorumlu Kodlama* ve *Tam Zamanında Kodu Sahiplenme*'dir.

1. Giriş – XP Nedir?

Son dönemde, her alanda değişimin hızı artmış, uzun süreli planlar daha riskli bir hal almıştır. Yazılım mühendisliği alanında da, değişime uyumu çok zor olan, temelinde uzun süreli planlar olan yazılım geliştirme metodları kullanılan projelerdeki başarısızlık oranı yükselmiştir. İşte bu duruma tepki olarak, hızlı program geliştirmeye dayanan ve isterlerin değişimine hızla adapte olabilen yazılım geliştirme teknikleri ortaya atılmıştır. Bu tekniklerin hepsi, "*Manifesto for Agile Software Development*" [2] adı verilen ve aşağıdaki 4 maddeden müteşekkil bir manifestoyu temel almaktadır. XP bu tekniklerin en yoğun olarak kullanılanıdır.

- Ağır süreçler ve araçlar yerine **bireyler ve bireyler arasındaki güçlü iletişim.**
- Yoğun dokümantasyon yerine **doğru çalışan yazılımlar.**
- Katı sözleşmeler yerine **müşteri işbirliği.**
- Bir planı izlemek yerine **değişime uyum sağlamak.**

Basit olarak ifade etmek gerekirse, geleneksel yazılım geliştirme metodlarında şu şekilde bir süreç izlenir: İsterler analizi, sistem tasarımı, kodlama, test ve teslim. Bu aşamaların herbiri son derece detaylı ve yoğun bir dokümantasyon gerektirir. Herhangi bir aşamada, daha önceki aşamalardan birinde değişiklik yapmak çok maliyetli olduğundan geleneksel metodlar bu tür değişimlere cevap vermez denilebilir. Bu yüzden XP'nin öne çıkmasına neden olan en önemli faktör değişime hızla adapte olabilmesidir.

Geleneksel yazılım geliştirme metodları ile XP'yi karşılaştırdığımızda karşımıza çıkan ikinci önemli nokta, XP'nin sürece değil ürüne odaklı bir metod olmasıdır. Diğer bir deyişle, XP kullanan yazılım geliştiriciler için çalışan ve ihtiyaçları doğru şekilde karşılayan bir ürün nasıl geliştirilmiş olursa olsun başarılıdır.

2. XP Çekirdek Pratikleri

Bu bölümde XP'nin temelini oluşturak pratiklerin herbiri kısaca açıklanmaktadır. Zaman içerisinde çeşitli araştırmacılar tarafından değişik pratikler ortaya atılmış ve pratikler listesi artmış olsa da çekirdek pratikleri anlamak XP'yi anlamak için yeterli olacaktır. Bu pratikler dört başlık altında toplanabilir; *planlama pratikleri*, *tasarım pratikleri*, *kodlama pratikleri* ve *test pratikleri*.

Planlama Pratikleri

- *Kullanıcı öyküleri :* XP'nin en önemli özelliklerinden birisi artımsal geliştirmedir. Her artım da neyin nasıl geliştirileceğini yazılımın kullanıcısı yani müşteri belirler. Bu işi de geliştirilecek herbir modül için, resmi bir düzeni olmayan öyküler yazar. Bu öyküleri geleneksel yazılım geliştirme metodlarındaki isterlere benzetebiliriz. Kullanıcı öykülerinin nasıl kullanıldığı sonraki pratiklerde görülecektir.
- *Planlama oyunu :* Kullanıcı tarafından yazılan öykülerin herbiri için geliştirme ekibi bir geliştirme süresi öngörür. Geliştirilecek artımların sırası her ne kadar kullanıcı tarafından belirlenir olsa da, şayet bir artımda teknik sebeplerle bir öncelik gereği varsa, geliştirme ekibi bunu kullanıcıya bildirir. Kısacası, planlama oyununda geliştirme ekibi ve müşteri tarafından geliştirilecek artımların sıralaması yapılır ve her bir artımın geliştirme süresi belirlenir.
- *Proje hızını ölçme :* Geliştirme boyunca her bir artımın ne kadar sürede tamamlandığı bilgisi saklanır. Bu bilgi ileriki artımların veya sonraki benzer projelerin süresini saptamada kullanılır.
- *Artımsal geliştirme :* Az önce de ifade etiğimiz gibi, XP'de proje herbiri 2-3 haftalık geliştirme süresi gerektiren artımlar halinde gerçekleştirilir. Herbir artımın sonunda, kullanıcı sistemi kullanmaya başlar ve

sonraki geliştirmelere devam edilir. Bu sayede müşteri için en öncelikli parçalar derhal geliştirilir ve devreye alınır.

- *Rol değiştirme :* Bir XP takımında çalışanlardan herbirinin projenin her aşamasında görev alması ve proje ile ilgili olan bitenin tamamından haberdar olması istenir. Bunu gerçekleştirmek için, takımdaki oyuncuların rolleri sık sık değiştirilir.
- *Ayaküstü toplantılar :* XP takımları, büyük masalar etrafında uzun süreli toplantılar yerine, daha kısa süren ayaküstü toplantılar yaparlar. Geliştirme ekibinin tamamının katıldığı bu toplantılarda hem herkesin herşeyden haberdar olması sağlanır hem de karşılaşılan zor teknik problemler tartışılır.
- *XP'yi uyarla :* XP kullanıyor olmak için, XP'nin tüm pratiklerini kullanmak gerekmez. Bu pratiklerden ekibe ve o anki projeye uygun olmadığı düşünülen pratikler kullanılmayabilir. Bunun yanında kullanılan pratikler de uyarlanıp ekibe ve projeye daha uygun hale getirilebilir.

Tasarım Pratikleri

- *Sade ve basit tasarım :* XP de herşey karmaşıklıktan uzak ve son derece basit olmalıdır. Kullanıcı tarafından şu an için istenmiş olmayan tasarımlar, ileride ihtiyaç duyalacağı düşünülse de yapılmamalı, her bir tasarım ihtiyaca cevap verecek asgari düzeyde olmalıdır.
- *Ortak isimler :* Projede isim verilmek gereken her şey için (modül, tasarım, test, vb.) ortak bir isim verme mekanizması kullanılmalıdır. Böylece takımdaki herkes tarafından konulmuş olan isimler tutarlılık arz eder.
- *CRC kartları :* Nesnelerin ait olduğu *class* (sınıf)'ların herbiri için "*Class, Responsibility and Collaboration*" yani CRC kartları oluşturulur. Bu kartlara ilgili sınıfın ismi, ve bu sınıfın yapması gereken işlevler sıralanır. Bu işlevlerin saptanması işi ise, kartın o sınıf ile ilişkisi olabilecek herkes arasında dolaştırılması ile yapılır. Bu da hemen hemen tüm takım anlamına gelir.
- *Kod tazeleme :* XP'de program kaynak kodlarının temiz ve kolay okunur olması çok önemlidir. Kod her kim tarafından geliştirilmiş olursa olsun, takımdaki herkes tarafından okunabilir ve kolayca değiştirilebilir olmalıdır. "*Refactoring*" yani kod tazeleme bu amaç için kullanılan bir XP pratiğidir. Modülün işlevini ve yapısını değiştirmeden kodun yapısını değiştirme anlamına gelir.

Kodlama Pratikleri

- *Geliştirme ortamında müşteri :* Müşteri yani yazılımın son kullanıcısı ile takım arasındaki iletişim XP nin en önemli noktalarından bir tanesidir. Bu iletişimi güçlendirmek, kullanıcıyı projenin bir parçası olarak ele almak ve çok hızlı geribesleme alabilmek için, kullanıcının kendisinin veya iş süreçlerini iyi bilen bir temsilcinin geliştirme ekibi ile aynı ortamda bulunması gerekir.
- *Kodlama standartları :* Tüm geliştiricilerin birbirine benzer ve diğer geliştiricilerin kolaylıkla okuyabileceği türden kod yazması için, takım veya proje bazında birtakım kodlama standartları belirlenir.
- *Önce test kodlama :* XP'nin en ilgi çekici kodlama pratiklerinden biri önce test kodlama pratiğidir. Modülün kendisi kodlanmadan önce o modülün çalışıp çalışmadığını, istenilen işleri yapıp yapmadığını test eden küçük test programları yazılır. Test programı doğru sonucu verdiğinde asıl modül bitmiş olur.
- *İkili programlama :* İki programcının bir bilgisayar başında otururak birlikte çalıştıkları bir uygulamadır. Bir ikili, biri klavyeyi kullanarak yazma, diğeri de yazılacak kodu düşünme işini yaparak aslında bir kişi gibi davranır.
- *Sık bütünleştirme :* Tüm yazılım projelerinde, programın çalıştığı ortam ile geliştirmenin yapıldığı ortam ayrıdır. Geliştirilen, test edilen ve kullanılmaya hazır olan her modül derhal çalışma ortamı ile bütünleştirilip kullanıma alınır.
- *Toplu kod sahiplenme :* Yazılım geliştirme projelerinde, genellikle program kodunun her bir parçasının bir veya bir grup geliştirici tarafından sahiplenilmesi ve o kod üzerinde yapılacak değişikliğin, o kodun sahibi tarafından yapılması yolu izlenirdi. XP de kodun her bölümü, tüm takım tarafından sahiplenilir. Sürekli rol değiştirme pratiği de bu konuyu destekleyici niteliktedir.
- *Fazla çalışma yok :* XP ekipleri haftada 40 saatten fazla çalışmamayı bir pratik olarak uygularlar ve fazla mesai yapmanın işleri zamanında yetiştirmeye çalışmanın kötü bir yolu olduğuna inanırlar. Yazılımcıların kendilerini sürekli zinde ve sağlıklı tutmaları ekibin dolayısıyla projenin performansını artırır.

Test Pratikleri

- *Birim (modül) testleri :* Test, XP'nin üzerinde çok fazla ve hassasiyetle durduğu bir konudur. Programın modüllerinin tamamının yüzde yüz doğru çalıştığından emin olunmalıdır. Geliştirilen her modül, programın çalışacağı ortama eklenmeden hemen önce detaylı bir testten geçer. Bu testler modülün kendisi geliştirilmeden önce kodlanmış olan programlardır. Test programlarını geliştirmek için, genellikle bir test çatısı kullanılarak geliştirilir.
- *Geçerlilik testleri :* Bir modül, geliştirme ekibi tarafından yapılan birim testlerinden geçtikten sonra müşteri

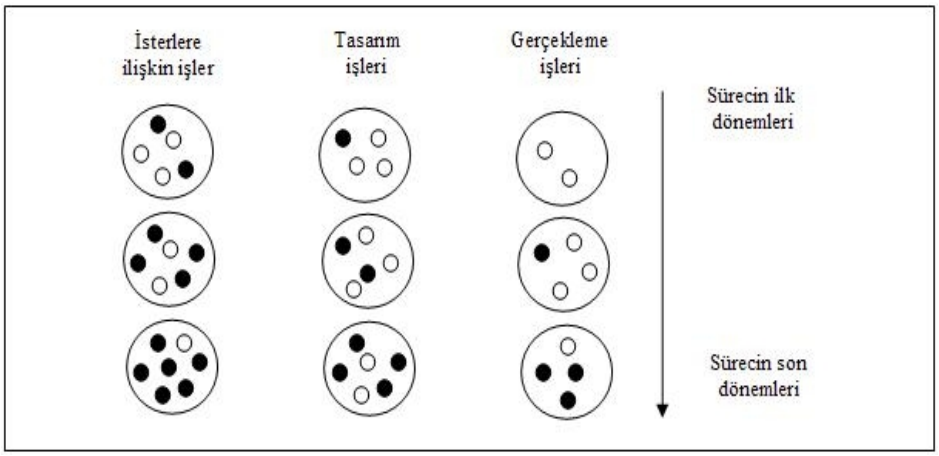
tarafından en başta yazılan kullanıcı öykülerine karşı, yine müşteriler tarafından test edilir. Testlerin her ikisinden de başarı ile geçen modül çalışma ortamı ile bütünleştirilir.

3. Yeni Pratikler

Bu bölümde, bu çalışmada ortaya konan pratikler açıklanacaktır. Bu pratikler; *İş Tabanlı Programlama, Önceden Yorumlu Kodlama* ve *Tam Zamanında Kodu Sahiplenme*'dir.

3.1. İş Tabanlı Programlama

İş tabanlı programlama[3], yukarıda yapılan sınıflandırma düşünüldüğünde bir tasarım pratiği olarak ele alınabilir. Bu pratikte, esas yapılacak olan işler ve bu işler arasındaki öncelik sıralamasıdır. Geliştirme sürecinin herhangi bir anında, geliştirmenin tüm fazları ile ilgili tamamlanmış (kapalı) ve tamamlanmamış (açık) işler bulunabilir. Daha önceden tamamlanmış olan bir iş, bir sebeple daha sonra açık hale gelebilir. Genellikle sürecin ilk dönemlerinde isterlere ait işler daha fazla, tasarım ve gerçekleme işleri daha azdır, son dönemlerde ise tasarım ve gerçekleme işleri, isterler ile ilgili işlere oranla daha fazladır. Fakat sürecin her anında az ya da çok her faza ait iş bulunabilir. Şekil 1, geliştirme sürecinde işlerin dağılımına ilişkin bir örnek içermektedir.



Şekil 1. Geliştirme sürecinde örnek iş dağılımı

Bu işler üç farklı –uzun dönem, haftalık, günlük- iş listesine kaydedilir. Yeni isterleri oluşması veya var olan isterlerin değişmesi durumunda bu listeler güncellenir.

3.2. Önceden Yorumlu Kodlama

Bir kodlama pratiği olan önceden yorumlu kodlama[3], temelde kodlamanın iki aşmaya ayrılmasına dayanmaktadır. Bu aşamalardan ilki kodun yapacağı işin ortaya konulması, diğer bir deyişle kodun semantiği, ikinci aşama ise kodun, bir programlama dilinde gerçeklenmesidir. Önce, algoritma editöre yorum satırları olarak yazılır. Bu aşamada yorum satırları, blokların yapısı gözetilerek dıştan içe doğru hizalı yazılır. İlk aşamanın bitmesiyle birlikte, aynı yazılım geliştirici veya ikili programlama eşi yorum satırları olarak yazılmış olan satırları programlama dilinin komutlarına çevirir. Bu işi yaparken önceden yazılmış yorumları da program kodunun içerisinde bırakır. Kodlama işlemi sırasında, satır satır değil, dıştan içe bir yol izlenir. Bu pratiğin, neticede yorumlu ve temiz bir kod elde ediyor olmaktan başka, kodlama işini paylaşabilmek, yarım bırakılan işe daha kısa sürede adapte olabilmek ve daha iyi bir kodlama planı yapabilmek gibi faydaları da vardır. Şekil 2 birinci aşaması tamamlanmış örnek bir kod içermektedir. Şekil 3 ise aynı programın ikinci aşamanın ortasındaki bir görüntüsünü içerir.

```
<%
'iki çevrimli döngü
    'birinci çevrimde zayıf önşartları çek ve dersler değişkenine ata
    'ikinci çevrimde katı önşartları çek ve dersler değişkenine ata
    'dersler değişkeninin sonuna kadar bir çevrim
        'hata değişkenini "true" olarak ata
        'öğrenci bu dersten geçtiyse hata değişkenini "false" olarak değiştir ve çık
        'geçmediyse dersin eşdeğerlerini çek ve esdeger değişkenine ata
        'esdeger değişkeninin sonuna kadar bir çevrim
            'öğrenci bu dersten geçtiyse hata değişkenini "false" olarak değiştir ve çık
        'esdeger çevrimi sonu
        'hata değişkeninin değeri "true" ise
            'hatayı döndür ve çık
    'dersler çevrimi sonu
'dış çevrim sonu
%>
```

Şekil 2. Birinci aşaması tamamlanmış örnek kod

3.3. Tam zamanında kodu sahiplenme

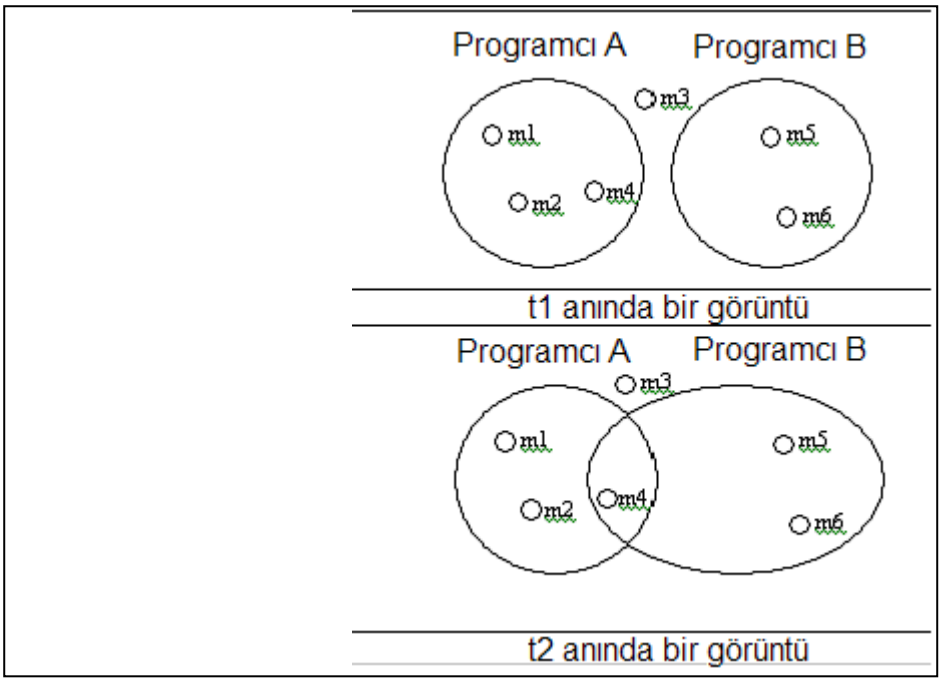
Tam zamanında kodu sahiplenme[3], aşırı programlamanın çekirdek pratikleri arasında olan toplu kod sahiplenmenin bir türevi olan, kodlama pratiğidir. Kodun sahiplenilmesinin, gerekmedikçe tüm takım tarafından sahiplenilmemesi gerektiğini vurgular. Kodun birden fazla geliştirici tarafından sahiplenilmesi, şu faktörlerden bir veya birkaçının oluşması ile olabilir:

- Kodun zorluğu
- Takım elemanlarının değişmesi
- Termin tarihinin yakınlığı

```
<%
For i=0 To 1'iki çevrimli döngü
    'birinci çevrimde zayıf onşartları çek ve dersler değişkenine ata
    If i=0 Then
        presql = "SELECT Pre_Req FROM Prerequisites WHERE Course_Code='"& ders &"' AND Min_Grade
    'ikinci çevrimde katı önşartları çek ve dersler değişkenine ata
    Else
        presql = "SELECT Pre_Req FROM Prerequisites WHERE Course_Code='"& ders &"' AND Min_Grade
    End If
    dersler = Conn.Execute(presql)
    'dersler değişkeninin sonuna kadar bir çevrim
    Do Until dersler.eol
        'hata değişkenini "true" olarak ata
        'öğrenci bu dersten geçtiyse hata değişkenini "false" olarak değiştir ve çık
        'geçmediyse dersin eşdeğerlerini çek ve esdeger değişkenine ata
        'esdeger değişkeninin sonuna kadar bir çevrim
            'öğrenci bu dersten geçtiyse hata değişkenini "false" olarak değiştir ve çık
        'esdeger çevrimi sonu
        'hata değişkeninin değeri "true" ise
            'hatayı dondur ve çık
    dersler.movenext
    Loop'dersler çevrimi sonu
Next'dış çevrim sonu
%>
```

Şekil 3. İkinci aşamanın ortasında örnek kod

Şekil 4'de tam zamanında kod sahiplenme ile ilgili bir örnek gösterilmiştir. Örnekte, t1 anında Programcı A tarafından sahiplenilen m4 kodu, t2 anında bir sebepten dolayı Programcı B ile müşterek olarak sahiplenilmiştir.



Şekil 4. Tam zamanında kod sahiplenme ile ilgili bir örnek

4. Sonuç

Bu çalışmada, aşırı programlama, geleneksel yazılım mühendisliği yaklaşımlarından farklılıkları ve çekirder pratikleri üzerinden anlatılmış ve üç yeni aşırı programlama pratiği geliştirilmiştir. Bu pratikler orta büyüklükte, web tabanlı bir yazılım geliştirme projesinde uygulanmış ve başarılı bulunmuştur. Yeni pratikler, kodun okunabilirliğini artırmış, kodu geliştirmek veya değiştirmek için her zaman birinin bulunabilmesine katkıda bulunmuş ve kod yazma verimliliğini artırmıştır.

5. Kaynakça

[1] Wake W.C., *The XP series: Extreme Programming Explored*, Addison-Wesley, NJ, 2002

[2] http://www.agilemanifesto.org

[3] M. Yıldız, "New Practices for Extreme Programming Applied in Campus ON-LINE, a Large Web Based Application Development Project," M.S. Thesis, Işık Üniversitesi, İstanbul, 2003