Yazılım Hata Kestirimi İçin Veri Analizi Yöntemlerinin Kullanılması

Özkan SARI[1,2], Oya KALIPSIZ[1]

[1]Bilgisayar Mühendisliği Bölümü, Yıldız Teknik Üniversitesi, İstanbul
[2] Provus – A Mastercard Company, Ayazağa, İstanbul
ozkan.sari@provus.com.tr, kalipsiz@yildiz.edu.tr

Özet. Yazılım kalite faaliyetleri içerisinde hataların giderilmesi için gerçekleştirilen test faaliyetleri önemli bir yer tutmaktadır. Bazı test faaliyetleri neticesinde dahi hataların tespit edilemediği durumlar olabilir. Bu nedenle yazılım hatalarının ve kusurlarının tespit edilebilmesi için etkin yöntemlere ihtiyaç vardır. Koddaki hatalı olabilecek yerler kodun özellikleri incelenerek tespit edilebilirse hataların daha erken bulunması ve müdahale edilmesi mümkün olacaktır. Hata kestirim faaliyetleri de bu amaca hizmet ederek, yazılımdaki hataların otomatik bir şekilde ve erken safhada tespitini amaçlamaktadır. Devam eden araştırma çalışmasında, belli başlı hata kestirim yöntemleri incelenmiş ve hata kestirimi için etkin bir model geliştirilmesi amaçlanmıştır. Modelin oluşturulması için lojistik regresyon analizi tabanlı yöntemler üzerinde durulmuştur. Ortaya konulan modelin oluşturulmasında ve sınanmasında açık bir hata veritabanından faydalanılmıştır. Ayrıca model *Provus – A Mastercard Company* bünyesinde geliştirilen yazılımların kodları üzerine uygulanmış ve hata kestirimi konusunda belirli bir seviyede başarı elde edilebilmiştir.

Anahtar Kelimeler: Yazılım kalite yönetimi, Yazılım Hataları, Hata kestirimi, Veri analizi, Lojistik regresyon analizi

1 Giriş

Geliştirilen yazılımların bazı hatalar içermesi doğaldır. Önemli olan bu hataların tespit edilebilmesidir. Hataların tespitinde testler önemli bir yer tutmaktadır ancak her zaman bu yeterli olmamaktadır. Testte tespit edilemeyen hataların sonradan tespit edilmesi veya yazılımdaki hataların mümkün olduğunca erken tespit edebilmesi için etkin yöntemlere ihtiyaç vardır.

Yazılım hatalarının, yazılım geliştirme faaliyetlerinin erken safhalarında tespit edilmesinin daha az masraflı olduğu bilinmektedir. Araştırmalar göstermiştir ki, bir hatanın düzeltilmesi yazılımın teslimi sonrasına kalırsa, aynı hatanın yazılımın geliştirme safhasında düzeltilmesine göre maliyet yüz kat daha artmaktadır [1]. Yazılım hata kestirim yöntemleri sayesinde yazılımdaki hataların, kodun bazı özellikleri incelenerek tespit edilmesi mümkündür.

Bu çalışmanın amacı, yazılımdaki hataların kodun özellikleri incelenerek tespit edilmesi ve hatalı sınıf/dosyaların otomatik bir şekilde ayırt edilebilmesidir. Bu amaçla kodların bazı metrikleri üzerinden lojistik regresyon analizi ile modülün hatalı olma ihtimali tespit edilecektir.

2 Yazılım Hata Kestirimi Yöntemleri

Yazılım mühendisliği alanında yapılmış çalışmalar incelendiğinde, sistemdeki yazılım kusurlarının tespiti ve kestirimi üzerine birçok araştırma yapıldığı görülmektedir [2]. Kaynak kodun metrikleri incelenerek yazılımda gelecekte oluşabilecek hataların tespiti mümkündür. Metrikler, yazılımın kalitesi hakkında bilgi sağlamak ve olası hataların kolayca algılanmasında fayda sağlarlar.

Hata kestirim faaliyetlerinin kod seviyesinde tespiti yönünde çok çeşitli yaklaşımlar bulunmaktadır. Bu yaklaşımlar hem yöntem hem de başarım açısından farklılıklar göstermektedirler [3].

Bu alanda yapılan çalışmalarda kullanılan farklı yaklaşımlar Marco D'Ambros ve diğerleri tarafından beş ana gurupta toplanmıştır [3]. Çalışmamızda farklı tipteki metrikler incelenmiştir., Kaynak kod metrikleri odaklı yazılım hata kestirimi yöntemlerine odaklanılmakla beraber diğer yöntemlerin güçlü yönlerinden faydalanılmaya çalışılmıştır.

2.1 **Kaynak Kod Metrikleri**

Çalışmamızda temel olarak kaynak kod metriklerini değerlendirilmesine yönelik bir yöntem kullanılmıştır. Kaynak kod metrikleri yöntemi, çeşitli yazılım metrik değerlerinin bir arada kullanılması ile bir model oluşturulması ve bu modelin yazılımdaki hatalarla bağlantısının bulunması esasına dayanır. Literatürdeki birçok çalışma CK (Chidamber & Kemerer) metriklerini kullanmaktadır. Buna ek olarak nesne tabanlı metrikler (Object-Oriented Programming - OOP), kaynak kod satır sayısı (Line of Code - LOC) gibi metrikler de kullanılmaktadır.

Ek olarak OOP metriklerinin CK metrikleri ile birleşimi de hata kestiriminde tercih edilmektedir. CK metrikleri üzerinden hata kestirimi yapan Chidamber & Kemerer modeli ve Gyimothy ve diğerlerinin LOC tabanlı hata kestirim yaklaşımı buna örnek olarak verilebilir [4, 5].

2.2 **Süreç Metrikleri Tabanlı Yaklaşım**

Süreç metrikli tabanlı yaklaşımda hatalara değişikliklerin sebep olduğu tezinden yola çıkılır ve değişikliklerin sayısı, son zamanlardaki değişiklerin sıklığı, değişikliklerin kaç farklı yazılımcı tarafından yapıldığı vb. değerler incelenir. Buradaki yaklaşımın temel mantığı özellikle farklı kişiler tarafından, çok ve sık değişiklik yapılan kodlarda hataların çok olduğu gerçeğidir. Bu yaklaşıma Moser ve diğerlerinin çalışmaları örnek olarak verilebilir [6]. Bu yaklaşım, çalışmada kullanılan kaynak kod metrik odaklı

yaklaşımın özelleşmiş bir tipi olarak düşünülebilir. Diğer metriklerle birlikte süreç tabanlı metriklerin de kullanılması çalışmanın ilerleyen aşamalarında düşünülebilir.

2.3 Önceki Kusurlar

Geçmişteki hatalardan yola çıkarak gelecekteki hataların tahmin edilebileceği düşüncesinden yola çıkılmıştır. Zimmermann ve diğerleri çalışmalarında geçmişteki kod kusurlarının bulunduğu yerlerin bir hata veritabanı oluşturularak belirlenmesinin önemine değinmişlerdir [7]. Çalışmamızda da bu tip bir hata veritabanından faydalanılmıştır.

2.4 Değişikliklerin Entropisi

Kompleks değişikliklerin basit değişikliklere kıyasla hataya daha açık olduğu tezine dayanır. Hassan tarafından yapılan çalışma bu yaklaşıma örnek olarak verilebilir [8]. Bu fikrin temelinde, bir sistemdeki değişikliklerin ne kadar dağınık olduğunun belirli bir zaman aralığında ölçülmesi vardır. Tek bir dosyayı etkileyen değişiklik, birçok dosyayı etkileyen değişiklikten daha basit kabul edilir. Değişikliklerdeki yaygınlık arttıkça, karmaşıklık da artacaktır. Bu tip bir veri elde edilemediği için söz konusu yönteme çalışmamızda yer verilmemiştir.

3 Veri Analizi Yöntemleri

Yazılım hata kestiriminde kullanılan çeşitli yöntemlerden biri de veri analizidir. Çalışmamızda, veri analizi yöntemlerinden lojistik regresyon analizi üzerine odaklanılmıştır..

Lojistik regresyon analizinde temel amaç bir regresyon denklemi oluşturarak, bireylerin hangi grubun üyesi olduğunu kestirmektir [9].

3.1 Regresyon Analizi

Regresyon analizi, aralarında sebep-sonuç ilişkisi bulunan iki veya daha fazla değişken arasındaki ilişkiyi belirlemek ve ölçmek için kullanılır. Hayatın içindeki birçok olayda sebep sonuç ilişkisine rastlamak mümkündür. Örneğin, yaş ile boy, şehir nüfusu ile suç oranı, hayvana verilen yem miktarı ile alınan süt miktarı, çalışanın iş yükü ile stres gibi çeşitli değişkenler arasında ilişki ortaya koymak mümkündür. [10].

Değişkenler arasındaki bu ilişkiyi kullanarak o konu ile ilgili tahminler ya da kestirimler yapabilmek amacıyla regresyon analizi kullanılır.

3.2 Doğrusal Regresyon Analizi ile Lojistik Regresyon Analizinin Kıyaslanması

Doğrusal regresyon analizi ile Lojistik regresyon analizi değişkenlerin ölçüm biçimi yönünden farklılık gösterir. Doğrusal regresyon analizinde bağımlı değişken ve bağımsız değişkenler sayısal (kesikli ya da sürekli) olarak belirtilir. Örneğin, yaş ile kan basıncı arasında bir ilişki aranacaksa; hem yaş, hem de kan basıncı sayısal olarak belirtilmelidir. Lojik regresyonda ise bağımlı değişkenler nitelik ve kategorik bakımdan nitel olarak değerlendirilirler. Cinsiyetin erkek ya da kadın olması buna örnek verilebilir. Lojik regresyonda bağımsız değişkenler sayısal ya da nitel değerler alabilirler [11].

Aşağıdaki örnekte 200 öğrenci, bir kurs bitiminde geçti/kaldı şeklinde değerlendirilmiştir. Bu kursa girişte de bir ölçme sınavı uygulanmıştır [12]. Kursa girişte uygulanan test skoru ile kurstan geçme arasındaki ilişkiyi bulmak için lojistik regresyondan faydalanılabilir. Bu örnekte doğrusal regresyon kullanmak olası değildir. Çünkü bağımlı değişken sonucu 1 ya da 0 değerini alabilir. Doğrusal regresyon ile sınırsız bir aralıkta sayısal değer elde edilebilir.

Şekil 1. Öğrenci giriş sınavı ile kurs geçme arasındaki ilişki araştırması örneği [12]

3.3 Lojistik Regresyon Analizi

Uygulamalı sosyal bilimlerde karşılaşılan ve araştırılan olaylara ilişkin elde edilen verilerde, çoğunlukla bağımlı değişkenin iki mümkün değerinden birine sahip olabileceği varsayılmaktadır. Örneğin bir kişi okuldan mezun olmuştur ya da olmamıştır, bir işçi çalışıyordur ya da işsizdir, bir kişi bir gruba üyedir ya da değildir, bir hasta tedaviye cevap verebilir ya da vermeyebilir. İki olası ve farklı değer içeren bu tür verilere iki değerli veriler denilmektedir. İki değerli veya ikili değişkenler literatürde (0;1) değişkenleri olarak da adlandırılmaktadır. İki değerli değişkenler ile çalışan bir araştırmacının hedefi, bağımsız değişkenlerin koşullu bir kümesine bağımlı olarak, başarı veya başarısızlık olasılığının kestirilmesidir [13].

Lojistik regresyon analizi, bağımlı değişkenin ölçüldüğü ölçek türüne ve bağımlı değişkenin seçenek sayısına göre üçe ayrılmaktadır [9, 10]:

- **İkili (Binominal) Lojistik Regresyon Analizi**: Bağımlı değişkenin iki düzeyi olduğunda kullanılır. Örneğin, hasta/sağlam, yaşıyor/öldü, etkili/etkisiz vb.
- **Sıralı (Ordinal) Lojistik Regresyon Analizi:** Bağımlı değişken sıralı nitel veri tipinde olduğunda kullanılır. Örneğin, hafif/orta/şiddetli, çok etkili/orta derecede, etkili/etkisiz vb.
- **Çok Kategorili/Düzeyli (Multinominal) İsimsel Lojistik Regresyon Analizi:** Bağımlı değişken ikiden çok düzeyli sıralı olmayan nitel veri tipinde olduğunda kullanılır. Örneğin, çalışıyor/çalışmıyor/emekli vb.

Lojistik regresyon, bağımsız değişkenin sayısına göre "tek değişkenli lojistik regresyon" ve "çok değişkenli lojistik regresyon" olarak da sınıflandırılmaktadır. Lojistik regresyon yönteminin hedefi, bağımlı değişkenin sonucunu tahmin edebilecek en sade modeli bulmaktır. Lojistik regresyon analizi sonucunda elde edilen modelin uygun olup olmadığı "model ki-kare" testiyle, her bir bağımsız değişkenin modelde varlığının anlamlı olup olmadığı ise Wald istatistiği ile sınanır [11].

3.4 Lojistik Regresyon Analizi Terimleri

Lojistik regresyon ile ilgili bazı terimler bu bölümde verilmiştir:

- **Odds:** Başarı ya da görülme olasılığının (p), başarısızlık ya da görülmeme olasılığına (1 - p) oranıdır.
- **Bahis Oranı (Odds Ratio, OR):** İki "odds"un birbirine oranıdır.
- **Lojit:** Bahis oranının doğal logaritmasıdır. Bahis oranı asimetriktir. Bu değerin doğal logaritması alınarak simetrik hale dönüştürülür Lojit katsayıları (lojit) doğrusal regresyon analizindeki "β" katsayısının karşılığıdır.
- **En Çok Olabilirlik Yöntemi (Maksimum Likelihood Estimation):** Lojistik regresyonda model kestiriminde en küçük kareler (ordinary least square) yöntemi yerine, en çok olabilirlik (maximum likelihood) yöntemi kullanılır.

Lojistik regresyonun istatistiksel modeli aşağıda verilmiştir. "β"katsayılarının hesaplanması bölümün devamında anlatılmaktadır.

$$\log \frac{p(x)}{1-p(x)} = \beta_0 + x \cdot \beta \tag{1}$$

Buradan yola çıkarak p, yani bir kategoriye ait olma olasılığı aşağıdaki şekilde bulunur:

$$p(x;b,w)=\frac{e^{\beta_0+x\cdot\beta}}{1+e^{\beta_0+x\cdot\beta}}=\frac{1}{1+e^{-(\beta_0+x\cdot\beta)}} \quad (2)$$

Lojistik regresyon analizi uygulanırken bazı noktalara dikkat edilmelidir. Uygun tüm bağımsız değişkenler modele dâhil edilmelidir. Bazı değişkenlerin modele dâhil edilmemesi hata teriminin büyümesine ve modelin yetersizliğine neden olabilir. Uygun olmayan tüm bağımsız değişkenler de dışlanmalıdır. Nedensel olarak uygun olmayan değişkenlerin modele dâhil edilmesi; modeli karmaşık hale getirebilir, modelin yorumlanmasını zorlaştırabilir, bu değişkenlerin bağımlı değişken üzerinde pay sahibiymiş gibi yanlış izlenim vermesine neden olabilir [11].

Örnek olarak, yaşı 60 yaşından büyük olanları yaşlı, küçük olanları genç kabul ederek, hastalığa yakalanma ile yaşlılık arasındaki ilişkiyi Tablo 1'deki örnek veri setine göre inceleyebiliriz. Burada bağımlı değişken, hasta olup olmama, bağımsız değişken ise yaşlılık durumu olacaktır.

Tablo 1. Yaşlılığa göre hastalığa yakalanma verisi

Yaşlılık (A)	**Hasta**	**Değil**	**Toplam**
Genç (<60)	22	51	73
Yaşlı (>60)	21	6	27

Bu veriden yola çıkarak aşağıdaki lojik regresyon hesaplamaları yapılabilir:

$$\text{Odds(genç)}=22/51=0{,}431 \quad (3)$$

$$y=\ln(0{,}431) = -0{,}841 \quad (4)$$

$$\text{Odds(yasli)}=21/6=3{,}5 \quad (5)$$

$$y=\ln(3{,}5)=1{,}253 \quad (6)$$

$$\text{Bahis Oranı} =3{,}5/0{,}431=8{,}121 \quad (7)$$

Buradaki bahis oranı değerinin anlamı şudur; yaşı 60'tan büyük olanların hasta olma oranları, yaşı küçük olanlara göre 8,121 kat daha fazladır.

Modeli yaşlı olma parametresine göre kurarak A: yaşlı olup olmama (1/0) değerini alır.

$$b_0= \ln(0{,}431) = -0{,}841 \quad (8)$$

$$b_1= \ln(3{,}5)-\ln(0{,}431) = 1{,}253 - (-0{,}841) =2{,}094 \quad (9)$$

$$y = -0{,}841+2{,}094A \quad (10)$$

Lojistik regresyon modeli bu durumda aşağıda gösterilmiştir:

$$p=\exp(-0,841+2,094A) / (1 + \exp(-0,841+2,094A)) \quad (11)$$

Örneğin A=1 yani yaşlı birisi, %77,8 oranında hasta olacaktır.

$$p=\exp(-0,841+2,094 . 1) / (1 + \exp(-0,841+2,094.1)) \quad (12)$$

$$p=3,5008 / 4,5008 = 0,778 \quad (13)$$

4 Veri Seti

Çalışmamızda Marco D'Ambros tarafından hazırlanmış hata kestirimi veritabanından faydalanılmıştır [3]. Bu veritabanı 5 farklı projeden toplamda 5371 örneklem içermektedir. Verilerin elde edilmesinde popüler açık kaynak kodlu bazı projelerin kodlarından faydalanılmıştır. Veritabanında CK, nesneye dayalı programlama, değişim, entropi gibi birçok farklı metrikler ve dosyada/sınıfta hata bulunup bulunmadığı bilgisi verilmektedir. Bu veritabanından sadece belli tipte metrikler seçilerek lojik regresyon analizi modeli oluşturmada kullanılmıştır.

Buradaki metriklerin tamamının kullanılmaması ve sadece belli metriklerin tercihi, bu verilerin *Provus* firmasından elde edilebilen verilerle uyumlu olması amacıyla yapılmıştır. Ayrıca elde edilen 30'a yakın metriğin tamamının bir model oluşturmada kullanılması hem pratikte hem de modelin başarısı için mümkün değildir. Yapılan araştırmalarda da 3 farklı metriğin hata kestirim modellerinde yeterli bir sayı olduğu ortaya konmuştur [14].

Veritabanının seçilen örnek metrikleri içeren bir kesiti Tablo 2'de görülmektedir.

Tablo 2. Kullanılan veri setinden örnek bir kesit

Dosya/Sınıf İsmi	**FAN-IN**	**WMC**	**NOA**	**NOM**	**Hatalı mı?**
IndexBinaryFolder	1	1	1	6	0
CachedIndexEntry	1	2	2	1	0
ASTNode	102	131	131	20	1
MemberTypeBinding	1	0	0	5	0
CodeSnippetParser	1	7	7	41	0
Location	1	12	12	2	0
SingleMemberAnnotation	2	4	4	13	0
NLSTag	2	4	4	2	0

Seçilen veri setindeki belirtilen kısaltmaların ifade ettiği metriklerin anlamları aşağıda verilmiştir:

– *FAN-IN* değeri mevcut sınıfa referans veren diğer sınıfların sayısını gösterir.

- *WMC* (*İng. Weighted Method Count*) CK metriklerinden birisidir [4]. Kodun karmaşıklığı (*İng. cyclomatic complexity*) değerini ifade eder. Metoddaki toplam farklı yollar hesaba katılır. Buradaki değer sınıftaki tüm metodların toplam WMC değerini ifade eder.
- *NOA* (*İng. Number of Attributes*) sınıftaki toplam değişken sayısını ifade eder.
- *NOM* (*İng. Number of Methods*) sınıftaki toplam metod sayısını ifade eder.

5 Lojistik Regresyon Uygulaması

Söz konusu hata kestirimi veritabanı R dili ve *R-Studio* aracı kullanılarak lojistik regresyon analizine tabi tutulmuştur [15]. 5371 örneklemden oluşan verinin %10'u test için ayrılmış, geri kalan kısmı lojistik regresyon modelinin oluşturulmasında kullanılmıştır.

Bir kod dosyasında hata olup olmama durumunun diğer niteliklerle bağlantısı üzerine lojistik regresyon modeli oluşturulmuştur. Modelin katsayıları Tablo 3'te verilmiştir. Burada Z değeri, *Wald* z istatistik testi değeridir. Tahmin katsayı değerinin, standart hata değerine bölünmesiyle elde edilir. Karşılık gelen katsayıların istatistiksel olarak anlamlı olup olmadığını test etmek için kullanılır. Pr(>|z|) ise, Z değerine karşılık gelen olasılık değeridir. Değerin küçük olması istatistiğin anlamlı olduğunu gösterir. Son sütunda da değerin ne kadar anlamlı olduğu simgesel olarak gösterilmiştir.

Tablo 3. Lojistik Regresyon Modeli Hesaplama Sonuçları ve Katsayıları

	Katsayı Tahmin	**Std. Hata**	**Z değeri**	**Pr(>|z|)**	**Anlamlılık**
(Sabit Terimi)	-2,251056	0,058578	-38,428	< 2e-16	***
FAN-IN	0,007283	0,003001	2,426	0,01525	*
WMC	0,006744	0,001177	5,728	1,02e-08	***
NOA	0,025149	0,004061	6,193	5,91e-10	***
NOM	0,015507	0,005223	2,969	0,00299	**
Anlamlılık (Significance) Kodları: 0 '***' 0,001 '**' 0,01 '*' 0,05 '.' 0,1 ' ' 1					
Hata ~ FAN-IN + WMC + NOA + NOM					

Yapılan hesaplamalar neticesinde elde edilen lojistik model denklemi aşağıda verilmiştir.

$$y = \text{logit}(p) = -2{,}251056+0{,}007283\text{x}\mathbf{FAN\text{-}IN}+0{,}006744\text{x}\mathbf{WMC}+0{,}025149\text{x}\mathbf{NOA}+0{,}015507\text{x}\mathbf{NOM} \quad (14)$$

Buradan yola çıkarak, kod dosyasının hatalı kategorisine ait olma olasılığı aşağıdaki şekilde hesaplanabilir:

$$p=\exp(y) / (1 + \exp(y)) \quad (15)$$

Tablo 3'te verilen katsayı tahmin değerleri için R dilindeki *confint* fonksiyonu ile hesaplanmış, % 95 güven aralığı değerleri Tablo 4'de gösterilmiştir.

Tablo 4. Lojistik Model Güven Aralıkları

	% 2,5	**% 97,5**
(Sabit Terimi)	-2,365867297	-2,136243951
FAN-IN	0,001400130	0,013165179
WMC	0,004435944	0,009051245
NOA	0,017189900	0,033108933
NOM	0,005270617	0,025742899

Burada katsayıların eksponansiyeli alınarak, bahis oranı (odds ratio, OR) değerleri de yorumlanmış ve Tablo 5'te gösterilmiştir.

Tablo 5. Lojistik Model Bahis Oranı Değerleri

(Sabit Terimi)	**FAN-IN**	**WMC**	**NOA**	**NOM**
0,105288	1,007309	1,006766	1,025468	1,015628

Tablo 5'deki sonuçlara bakarak, diğer değişkenler sabit tutulmak üzere değişkenlerdeki değişimin ihtimaller üzerindeki etkisi hakkında aşağıdaki yorumlar yapılabilir:

- FAN-IN değerindeki 1 birim artış, kabul ihtimalini %0,73 arttırır.
- WMC değerindeki 1 birim artış, kabul ihtimalini %0,67 arttırır.
- NOA değerindeki 1 birim artış, kabul ihtimalini %2,55 arttırır.
- NOM değerindeki 1 birim artış, kabul ihtimalini %1,56 arttırır.

Lojistik regresyon denkleminin elde edilmesinden sonra, test için ayrılan verinin modele göre nasıl sonuç verdiği kıyaslanmıştır. Modelin %84 oranında başarılı olduğu görülmüştür. Test sonuçları Tablo 6'da verilmiştir.

Tablo 6. Lojistik Model Test Sonuçları

Test Veri Sayısı	573
Başarılı	486
Başarısız	87
Başarı Oranı	% 84,817

6 Modelin Kullanımı

Çalışmamızda bir sonraki adım, modelin *Provus* bünyesinde geliştirilen yazılımların kodları üzerine uygulanmasıdır. Çalışmada kullanılan, hata veritabanı genelde popüler açık kaynak kodlu bazı projelerin kodlarından derlenmiştir. Bu veri seti üzerinden oluşturulan modelin, tamamen farklı bir kod ortamında sınanması ve başarısının görülmesi önemlidir. Bu nedenle *Provus* bünyesinde geliştirilen bir proje üzerinde uygulama gerçekleştirilmiştir.

6.1 Modelin Uygulandığı Projenin Özellikleri

Provus bünyesinde geliştirilmekte olan ATM yönetim sistemi yazılımı birçok farklı modülden oluşan büyük ölçekli bir yazılımdır. Bu yazılım 7-24 ATM'lerle iletişim halinde çalışan canlı bir sistemdir [16]. Bu yazılımı oluşturan proje kodları Java ve C programla teknolojileri ile geliştirilmiştir ve 250 bin satırdan fazla kod içermektedir. İlk aşamada projede aktif olarak kullanılan temel bazı sınıflar incelenerek sonuçları paylaşılmıştır.

İncelenen sınıflar ve projede gerçekleştirdikleri işlevler aşağıda verilmiştir:

- **FileUtil:** Projenin ortak kütüphanesinde bulunan bir sınıftır. Genel dosya okuma ve yazma işlemleri bu sınıftan yönetilir.
- **MailMessageHelper:** Tüm modüller tarafından e-posta göndermek için ortak olarak kullanılır.
- **ProvusScheduledThreadPool:** Paralel gerçekleştirilecek işlemlerin yönetilmesinde kullanılan, yoğun iş yüklerini yöneten temel sınıflardır.
- **ModuleInitializer:** Modüllerin ilklendirme işlemlerini yöneten ortak sınıflardır. Projenin ortak kütüphanesinde bulunur.
- **ConfigUtil:** Projelerin kullandığı ayar dosyalarının okunması ve değerlerinin ortak bir hafıza alanında tutulmasını yöneten sınıftır.
- **EjournalParser:** ATM'ler üzerinden alınan jurnal denilen log dosyalarının işlenmesini yöneten ana sınıftır.
- **DieboldEJournalDownloadAtmRunner:** Diebold marka ATM'lerdeki jurnal denilen log dosyalarının ATM'den alınması işlemlerinin yönetildiği sınıftır.
- **PAYSFTPClient:** Projelerdeki ATM'ye FTP ile bağlanma gibi temel işlemleri yürüten sınıftır.

6.2 Metrik Değerlerinin Elde Edilmesi

Söz konusu *Provus* bünyesinde geliştirilen proje kodlarından modele uyumlu metrik değerlerinin elde edilmesinde *Eclispe IDE* [13] *Google Code Pro Analytix* [14] ve *Eclipse Metrics Eklentisi* [18] araçlarından faydalanılmıştır.

6.3 İlk Sonuçlar ve Bulgular

Elde edilen örnek veri seti ve modele göre hesaplanan hata ihtimali oranı Tablo 7'de görülmektedir. İncelenen her bir dosya için Tablo 3'te verilen modele uygun metrik değerleri çıkartılmış ve modelin sonucunda tespit edilen hata ihtimali oranı ve gerçekte dosyada hata bulunup bulunmadığı bilgisi verilmiştir.

Tablo 7. Elde Edilen Modelin Provus Proje Kodlarına Uygulanması

Dosya/Sınıf İsmi	FAN-IN	WMC	NOA	NOM	Hata İhtimali	Hata Var mı?
FileUtil	95	143	2	44	%53.44	Evet
MailMessageHelper	2	22	0	3	%11.49	Hayır
ProvusScheduledThreadPool	2	20	5	9	%13.75	Hayır
ModuleInitializer	11	7	3	3	%11.90	Hayır
ConfigUtil	4	8	3	1	%11.14	Hayır
EjournalParser	1	185	24	37	%54.52	Evet
DieboldEJournalDownloadAtmRunner	1	61	12	15	%21.45	Hayır
PAYSFTPClient	17	52	12	26	%25.51	Evet

Tablo 7'de verilen dosya/sınıflar için hata ihtimalleri incelendiğinde `FileUtil` ve `EjournalParser` sınıflarında diğerlerinden çok yüksek olarak %50 üzeri bir hata oranı tespit edilmiştir. Bu sınıflarda gerçekten de raporlanan hatalar mevcuttur. Bu iki sınıf dışında `PAYSFTPClient` sınıfında da raporlanmış bir hata olmasına rağmen, model tarafından hata ihtimali %25 gibi daha düşük bir değer olarak gösterilmiştir. Diğer sınıflar için modelin tespit ettiği düşük hata oranları, gerçek sonuçlarla uyumluluk göstermektedir.

7 Gelecek Çalışmalar

Bildiride konu edilen araştırma çalışması devam etmektedir. Yapılan temel çalışmada kayda değer bir ilerleme kaydedilmiştir. Bununla birlikte daha farklı ve çeşitli metrikleri dikkate alan bir model kurmak amacıyla çalışmaların ilerletilmesi düşünülmektedir.

Yapılan çalışmalarda hem modelin oluşturulmasında, hem de test için eldeki veri setlerinin önemi kritiktir. Bunun için daha çeşitli projelerden kodların analiz edilerek bunların metrikleri değerlendirilmelidir. Ayrıca modelden edilen başarımın daha farklı yazılım hata kestirimi yöntemlerinin başarımlarıyla kıyaslanması faydalı olacaktır.

Yapılacak çalışmanın neticesinde kodun analizi ve hata kestiriminin otomatik hale getirilmesi için, çalışmada ortaya konacak nihai modelin bir yazılım veya araç haline getirilmesi düşünülebilir.

8 Teşekkür

Çalışmamız *Provus – A Mastercard Company* tarafından desteklenmiştir. Çalışmaya konu olan model, firma bünyesinde geliştirilen yazılımların kodları üzerine uygulanmıştır.

9 Kaynakça

1. Boehm, B., Basili V.R., "Software Defect Reduction Top 10 List", IEEE Computer, Vol. 34, No. 1, s. 135-137, 2001.
2. Thilo Mende and Rainer Koschke, Effort-aware defect prediction models, In Proceeding of the 14th European Conference on Software Maintenance and Reengineering (CSMR 2010), s. 109–118. IEEE Computer Society, 2010.
3. Marco D'Ambros, Michele Lanza, and Romain Robbes. An extensive comparison of bug prediction approaches. In MSR '10: Proceedings of the 7th International Working Conference on Mining Software Repositories, s. 31-41, 2010
4. Raimund Moser, Witold Pedrycz, and Giancarlo Succi. A comparative analysis of the effciency of change metrics and static code attributes for defect prediction. In Proceedings of ICSE 2008, sayfa 181-190, 2008.
5. Thomas Zimmermann, Rahul Premraj, and Andreas Zeller. Predicting defects for eclipse. In Proceedings of PROMISE 2007, sayfa 76. IEEE CS, 2007.
6. Ahmed E. Hassan. Predicting faults using the complexity of code changes. In Proceedings of ICSE 2009, s. 78-88, 2009.
7. Shyam R. Chidamber and Chris F. Kemerer. A metrics suite for object oriented design. IEEE Trans. Software Eng., 20(6):476-493, 1994.
8. Tibor Gyimothy, Rudolf Ferenc, and Istvan Siket., Empirical validation of object-oriented metrics on open source software for fault prediction. IEEE Trans. Software Eng., 31(10):897-910, 2005.
9. Larose, D. T. (2006) Logistic Regression, in Data Mining Methods and Models, John Wiley & Sons, Inc., Hoboken, NJ,
10. Fikret Gültekin, Regresyon Analizi, Türkistatistik.net, http://turkistatistik.net/upload/dosya/reganaliz.pdf , Erişim Tarihi: 9 Mayıs 2014.
11. Sümbüloglu, K., Sümbüloglu, Biyoistatistik, Hatiboglu Yayınevi, 2007.
12. Robert Burns & Richard Burns, Business Research Methods and Statistics Using SPSS, Chapter 24, Sage Publications, 2008.
13. Eclipse Projesi Websitesi. http://www.eclipse.org. , Erişim Tarihi: 9 Mayıs 2014.
14. Wang, H., Khoshgoftaar, T. M. ve Seliya, N., "How Many Software Metrics Should be Selected for Defect Prediction?", FLAIRS Conference, 2011.
15. R-Studio Web Sitesi, https://www.rstudio.com/, Erişim Tarihi: 9 Mayıs 2014.
16. Provus ATM Operasyonları, http://www.provus.com.tr/hizmetler/atm-operasyon.html, Erişim Tarihi: 9 Mayıs 2014.
17. Google Code Pro Analytix Projesi web sitesi. https://developers.google.com/java-dev-tools/codepro/doc/, Erişim Tarihi: 9 Mayıs 2014.
18. Eclipse Metrics. http://metrics2.sourceforge.net, Erişim Tarihi: 9 Mayıs 2014.