YAZILIM KALİTESİ İÇİN YİNELEMELİ ÖLÇME YÖNTEMİ

Nurdan CANBAZ, Feza BUZLUCA

Bilgisayar ve Bilişim Fakültesi
İstanbul Teknik Üniversitesi
İstanbul, Türkiye

nurcanbaz@itu.edu.tr, buzluca@itu.edu.tr

Özet-Bilgisayar yazılımlarının insan hayatına her geçen gün daha çok girmesi sonucunda kaliteli yazılım geliştirmek ve bunun paralelinde yazılım kalitesini ölçebilmek önem kazanmaya başlamıştır. Bu çalışmada yinelemeli kalite ölçme metodu tanıtılacaktır. Metodun temel amacı, nesneye dayalı yazılımların proje yaşam döngüsü içinde yer alan kodlama safhasındaki değişimlerini göstermektir. Sınıfların zamanla nasıl değiştiğinin takip edilebilmesi, proje yürütücüleri için projenin zaman içinde ne şekilde ilerlediğinin izlenebilmesi, geliştiriciler için sınıflarla ilgili ortaya çıkabilecek sorunların önceden tahmin edilmesi açısından önem taşımaktadır. Bu kapsamda yazılım kalitesini ölçmeye dayalı metrikler ve istatistiksel analizler kullanılarak yazılım sınıflarının kalitesine ait değerlendirmeler yapılmıştır. Çalışmanın devamında veri madenciliği ve makine öğrenme teknikleri kullanılarak analiz yönteminin geliştirilmesi planlanmaktadır.

I. GİRİŞ

Günümüzde yazılım endüstrisindeki gelişmeler ile birlikte kaliteli yazılım geliştirmenin önemi artmıştır. Küçük ölçekli yazılımlar bile artık tek kişi tarafından geliştirilmemekte bir yazılım ekibi tarafından oluşturulmaktadır. Bu da yazılımın tasarlanmasını, geliştirilmesini ve yönetilmesini zorlaştırmaktadır.

Uygun yöntemler kullanılmadan ve kalite kriterleri dikkate alınmadan geliştirilen yazılımlar firmaları ve yazılımları satın almak isteyen müşterileri zarara uğratmaktadır. Bu doğrultuda yazılım kalitesi farklı açılardan değerlendirilmelidir. Bunlar, geliştirici ekip, proje yönetimi ve yazılım ürününü satın alacak müşteridir.

Müşteri açısından incelendiğinde, kaliteli bir yazılımın temel özellikleri işlevsel isterlere uygunluk, kullanım kolaylığı, güvenilirlik (hata sıklığının az olması) ve güncelleme kolaylığıdır Yazılım ürünün kalitesini önceden bilmek müşterinin ürünü ile ilgili ortaya çıkacak sorunları önceden kestirmesine yardımcı olacaktır. Böylece müşteri pazardaki alternatif ürünlerden uygun olanı seçebilecektir.

Geliştirici ve proje yönetimi açısından bakıldığında ise çıkabilecek hataları veya ilerde sorun yaratabilecek bileşenleri önceden tespit edebilmek, maliyeti azaltmak ve daha başarılı bir projeye imza atabilmek için önemli rol teşkil eder.

Yazılım projelerinde kalite halen soyut bir kavram olma özelliğini taşımaktadır; ancak yapılan çalışmalarla yazılım kalitesinin ölçülebilmesi için farklı metrikler geliştirilmiştir. Bu metrik değerleri ile birlikte yazılımın kalitesi somut değerlere çevrilebilmektedir [1][2].

Yazılım projelerinde kullanılan metrik ölçme araçları ile birlikte sayısal veriler elde edilebilmektedir. Ancak bu veriler üzerinden anlam çıkarabilmek ve çıkarılan bu anlamın doğruluğunu teyit edebilmek hem çaba hem de uzun zaman gerektirmektedir. Literatürde bu verilerden anlam çıkarmaya yönelik araştırmalar hâlihazırda mevcut olmasına karşılık yeni araştırmaların yapılmasına ihtiyaç duyulmaktadır.

Bizim çalışmamız, proje yöneticilerine ve geliştiricilere nesneye dayalı geliştirilen yazılım projelerindeki kodlama safhasının nasıl ilerlediğinin gösterilmesini hedeflemektedir.

II. İLGİLİ ÇALIŞMALAR

Bu bölümde, hata eğilimli sınıfların bulunması ve veri madenciliği tekniklerinin yazılım ölçmede kullanılmasına ilişkin literatürdeki araştırmalar incelenmiştir.

Karar ağaçları, yapay sinir ağları ve destek vektör makineleri hata eğilimli sınıfları bulmakta kullanılmaktadır [3]. Burada kullanılan model ve veri setinin hata eğilimli sınıfları bulmakta performansı etkilediği belirlenmiştir [4][5].

Li ve Leung tarafından yapılan çalışmada, hataya yatkınlığın bulunması için denetimsiz öğrenme modelinin geliştirilmesi amaçlanmaktadır [3]. Çalışmadaki çıkış noktası, aynı metrik kümelerinde bulunan bileşenlerin benzer hata yatkınlıklarının olmasıdır. Kullanılan veri seti NASA'dan temin edilen 12 farklı projeye ait hata kayıtlarını ve kaynak kodları içermektedir. Elde edilen veriler üzerinde ön işleme yapılarak veriler normalleştirilmiş ve metrik değerleri hesaplanmıştır. En Yakın Komşu algoritması kullanılarak oluşturulan model ile hata yatkınlıklarının bulunması hedeflenmiştir.

Veri madenciliği teknikleri kullanılarak yapılan bir diğer çalışma yazılımların kalitesinin önceden tahmin edilebilmesini sağlamak amacı ile yapılmıştır[6]. Bu çalışmayı yapan araştırmacılar daha önceki çalışmalarında C4.5 algoritmasını [7] kullanarak yaptıkları çalışmayı J48 [8] algoritmasını kullanarak yapmışlardır. Projenin kalitesinin tahmin edilmesi için QMOOD (Quality Model for Object

Oriented Design) [9] model araştırmacılar tarafından tercih edilmiştir. Burada QMOOD modelinin yapısında bulunan "Çok Biçimlilik, Karmaşıklık, Kalıtım vb." için "İyi, Kötü, Çok İyi" şeklinde tanımlayıcılar atanmış ve Weka açık kaynak kodlu veri madenciliği aracı kullanılarak tahminler yapmaya çalışılmıştır. Bu çalışmada yazılım ürününün sürümleri arasında bir kıyaslama yapılmamakta sadece tek bir sürüm üzerinde inceleme yapılmaktadır.

Makine öğrenmesi ve veri madenciliği teknikleri kullanılarak yapılan benzer bir çalışma, kodda programlama kurallarının çıkarılması, kopyala-yapıştır kısımların ve API kullanımlarının araştırıldığı çalışmadır [10]. Bu çalışmanın amacı bu üç özelliğin bir arada bulunduğu bir yöntem bulmaktır. Çünkü araştırmacılara göre kopyala-yapıştır ile yazılan kodlarda hatalının görülme olasılığı fazladır ve hata tüm kodlarda olacağından bu tarz kodların bakımı zordur.

Veri madenciliği teknikleri dışında bulanık mantık gibi yapay zeka teknikleri de son dönemdeki araştırmalarda kullanılmıştır. Aljahdali ve Sheta yaptığı çalışmada yazılımın geçmiş hatalarına bakılarak projede çıkabilecek hataları bakım safhasına geçmeden tahmin edilmesi amaçlayan model (Software Reliability Growth Model) bulanık mantık kullanılarak geliştirilmiştir. Bu modelin oluşturulmasında 16 projenin hata kayıtları incelenmiş, testi için ise gerçek zamanlı, askeri ve işletim sistemi olmak üzere üç proje kullanılmıştır [11].

Benzer bir çalışma anlaşılabilirliğin (intelligibility) bulunmasını makine öğrenmesi kullanarak bulunmasını amaçlayan çalışmadır. Makine öğrenmesi tekniklerinden olan Bayes Ağları, Olay Tabanlı Öğrenme, Yapay Sinir Ağlarından Weka açık kaynak kodlu yazılım aracılığı ile bu çalışmada yararlanılmıştır. Bu çalışmadaki asıl amaç aynı metriklerin farklı modellerle nasıl sonuç verdiğinin incelenmesi ve anlaşılabilirliğe olan etkilerinin araştırılmasıdır [12].

Yapılan diğer bir çalışma test adımlarından biri olan geleneksel kod gözden geçirmelerinin yerine makine öğrenmesi tekniklerinin kullanılmasının önerildiği çalışmadır [13].

Nesneye dayalı programlarda sınıf değişimlerinin önceden tahmin edilmesinde makine öğrenmesinin kullanıldığı çalışmalar yapılmıştır [14]. Bu çalışmada kaynak kod yerine tersine mühendislik yapılarak çıkarılan UML diyagramlar kullanılmıştır. Amaç sürümde sınıfın ne kadar değişeceğini bulmaktır. Bunun için JFlex (The Fast Scanner Generator for Java) isimli 58 sınıflı açık kaynak kodlu proje incelenmiştir.

Metriklerin gruplamasına yönelik çalışmalar da literatürde mevcuttur. Metriklerin gruplanması ve farklı projelerin sonuçlarının incelenmesine yönelik çalışmalardan biri C++' da nesneye dayalı metriklerin incelenmesine yöneliktir [15]. Bu çalışmada iki tane proje nesneye dayalı metrikler aracılığı ile karşılaştırılmıştır. Karşılaştırılan projelerden biri bir kolejin 9 sınıftan oluşan kütüphane yönetim sistemi, diğeri 8 sınıftan oluşan grafik editör yazılımıdır.

Hata bulmaya yönelik bir çalışma 2 milyon satırlık kodun test verileri ile yapılmıştır [16]. Bu çalışmada amaçlanan, müşteriye teslim edilmeden önce yapılan sağlamlık ve kararlılık testlerinde bulunan hatalı verilerin kullanılarak bundan sonra oluşacak hataların önceden tahmin edilmesini sağlamaktır. Araştırmacıların uzun süren testlerden önce hatayı kısa sürede tahmin eden bir model geliştirmek için poison dağılımından yararlanmışlardır.

Geçmiş verilerin yazılım kalitesinin belirlenmesinde kullanıldığı bir diğer çalışma ise programda bulunan kalıpların hata bulmada kullanıldığı araştırmadır [17]. Bu araştırmada hata kodları ve hata raporlarının girilerek gelecekte oluşacak hatalı kodların program kalıpları ile otomatik olarak bulunabileceği savunulmaktadır.

III. YAKLAŞIM

Yapılan literatür taramasında hataya yönelik araştırmalarda veri madenciliği tekniklerinin yoğun olarak kullanıldığı görülmüştür. Bu çalışmada yeni bir yaklaşım olarak, sınıfların ilerideki durumlarına ilişkin kestirimlerin sınıfların proje yaşam döngüsü boyunca değişimlerinden çıkarılması hedeflenmiştir.

Diğer bir değişle çalışmamızda amaçlanan, yazılımların sürümleri kullanılarak yazılımın sınıflarının beraberinde projenin bütünün ilerlemesi hakkında karar verilip verilemeyeceğinin bulunmasıdır.

Bilindiği gibi büyük projelerde çok sayıda sınıf bulunmakta ve sayısal değerlere bakılarak bu sınıfların durumları hakkında karar verilmekte proje yönetimi ve geliştiriciler zorlanmaktadır. Üzerinde çalıştığımız yöntemde proje yürütücüleri ve yazılım geliştiricilere hatta ilerleyen aşamalarda yazılımın kalitesini sorgulayan müşteriye yazılım sınıflarının değişimi ve projenin ilerleyişinin sunulması amaçlanmıştır.

Çalışmada bir firmanın geliştirdiği yazılım projesi kullanılmıştır. Proje, C++ programa dili ile yazılmış olan ve Qt grafiksel arayüz kütüphanesi kullanıldığı gerçek zamanlı bir uygulamadır. Projenin tüm test faaliyetlerini tamamlanmış ve müşteriye teslim edilmiştir. Yaşam döngüsünün bakım safhasında olan projenin çalışmaya katkısı, projenin tüm yaşam döngüsünün sürümlerine ulaşılabilmesidir.

Analiz ve tasarım evresi dâhil 30 ayda bitmiş olan projenin son 15 ayı kodlama için ayrılmıştır. Şelale modelinin kurum içi uyarlanmış halini benimseyen ekibin test faaliyetleri tüm yaşam döngüsünde devam etmiştir. Projenin kodlarının incelenmesi aşamasında yanıltıcı olmaması için ilk 5 aylık kod değişimi analiz edilmemiş yazılımın belirli bir aşamaya gelmesi amaçlanmıştır. Son 10 ay içinde sınıfların ne şekilde değiştiğinin incelenmesi üzerine çalışılmıştır.

Çalışmanın hazırlık aşamasında kullanılması planlanan metrikler belirlenmiştir. Kullanılan metrikler ve tanımları Tablo 1'de verilmiştir. Belirlenen metriklerin değerleri SCI Understand aracı [18] ile her ayki sınıf değişimleri için ayrı ayrı hesaplanmıştır.

Çalışmanın diğer aşamasında yazılımın iç özelliklerini etkileyen metrik değerleri belirlenmiştir.

Tablo 1 Tanımlı Metrikler

Metrik	Açıklaması
Ortalama Karmaşıklık (AvgCyclomatic)	Tüm metotlar için karmaşıklık sayısı
Ortalama Değişen Karmaşıklık (AvgCyclomaticModified)	Tüm iç içe metotlar için karmaşıklık sayısı
Ana Sınıf Sayısı (CountClassBase)	Ana sınıf sayısı
Bağımlı Sınıf Sayısı (CountClassCoupled)	Bağımlı sınıfların sayısı
Türemiş Sınıf Sayısı (CountClassDerived)	Alt sınıfların sayısı
Sınıf Method Sayısı (CountDeclClassMethod)	Sınıf metotlarının sayısı
Sınıf Değişken Sayısı (CountDeclClassVariable)	Sınıf değişkenleri sayısı
Kalıtım Ağacının Derinliği (MaxInheritanceTree)	Sınıfın kalıtım ağacının köküne uzaklığı
Uyum Eksikliği (PercentLackOfCohesion)	Sınıflardaki uyum eksikliğinin yüzdesi
Yerel Method Sayısı (CountDeclMethod)	Yerel metotsayısı

Belirlenen iç özelliklerin anlamları ve özellikleri etkileyen metrikler aşağıda verilmiştir. Bunlar;

Bağımlılık (Coupling); İki nesneden en az birinin diğerine etki etmesi olarak tanımlanır [19]. Bir sınıfın bağımlılığı; kendi sınıfları için diğer sınıfları ne kadar kullandığına ve başka sınıflar hakkında sahip olduğu bilgiye bağlıdır.

Bağımlılık için seçilen metrik: Bağımlı Sınıf Sayısı (CountClassCoupled)

Uyum (Cohesion); Uyumluluk bir sınıftaki metot ve değişkenlerin bir biri ile ne kadar ilişkili olduğuna bağlıdır. Eğer sınıf uyumluluğu düşükse sınıfın yeniden kullanılabilirliği ve bakımı zordur. Bu bağlamda müşteriden gelen değişik isteklerinde bu sınıfların değiştirilmesi hataya yol açabilir.

Uyum için seçilen metrikler: Uyum Eksikliği (PercentLackOfCohesion), Yerel Metot Sayısı (CountDeclMethod)

Kalıtım; Bir sınıfın özelliklerini taşıyıp yeni ek özellikler içeren sınıflar türetilmesidir. Nesneye dayalı yazılım geliştirmede önemli yer taşımasına karşılık dikkatli kullanılması gerekmektedir. Kalıtım ağacının derin olması durumda yazılımın karmaşıklığı ve bakım zorluğu, test edilecek durum sayısı artmaktadır. Ayrıca derin kalıtıma sahip sınıflarda esneklik azdır.

Kalıtım için seçilen metrikler: Ana Sınıf Sayısı (CountClassBase), Türemiş Sınıf Sayısı (CountClassDerived), Kalıtım Ağacının Derinliği (MaxInheritanceTree)

Karmaşıklık; Bir sınıfın anlaşılabilirliğinin zor olmasıdır. Anlaşılabilirliğin zor olması sınıfın ihtiva ettiği özelliklerin çokluğundan kaynaklanmaktadır. Karmaşıklığı çok olan bir sınıfın testi zordur. Beraberinde yeniden kullanılabilirliği azaltmıştır.

Karmaşıklık için seçilen metrikler: OrtalamaKarmaşıklık (AvgCyclomatic), Ortalama Değişen Karmaşıklık (AvgCyclomaticModified), Sınıf Method Sayısı (CountDeclClassMethod), Sınıf Method Sayısı (CountDeclClassMethod), Sınıf Değişken Sayısı (CountDeclClassVariable)

Yukarıda tanımlanan yazılımın iç özellikleri gruplandırılarak daha üst seviyede örneğin proje yürütücüsü olarak bakıldığında projenin gidişatı hakkında aylık bilgi sahibi olunması amaçlanarak; yazılımın esnekliği, yeniden kullanılabilirliği ve bakım kolaylığını etkileyen yazılım iç özellikleri belirlenmiştir. Bu özelliklerin belirlenmesinde McCall [20] ve ISO/IEC 9126 [21] kalite modellerinden yararlanılmıştır. Tanımlanan yazılım iç özelliklerin yazılımda etkilediği kalite özellikleri Tablo 2'de verilmektedir.

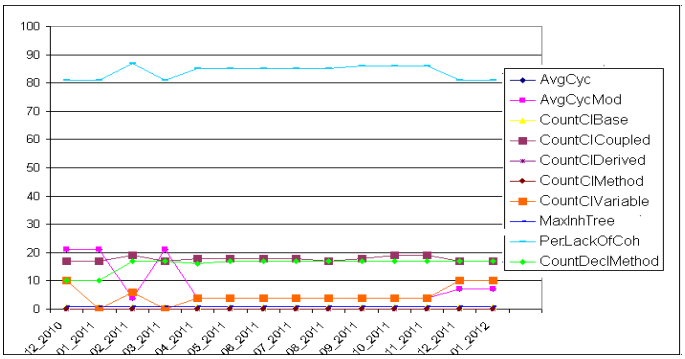
Tablo 2 Yazılım Kalite Değişkenleri

	Kalite Değişkenleri		
	Yeniden Kullanılabilirlik	Bakım Kolaylığı	Esneklik
Bağımlılık	*	*	
Uyumluluk	*	*	
Kalıtım	*		*
Karmaşıklık	*		*

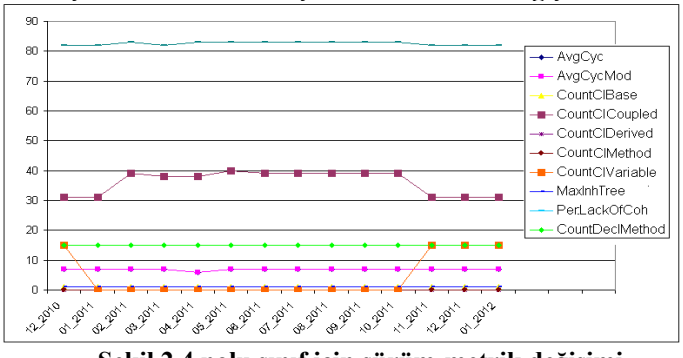
Burada özelden genele ilerleyen bir yaklaşım belirlenerek model oluşturulmuştur. Çalışmanın ilk adımında nesneye dayalı metriklerin tanımlanmasının ardından yazılımda bu metriklerin iç özellikleri nasıl etkilediğine karar verilmiş, ardından yazılımın kalite özellikleri üzerindeki etkilerine karar verilmiştir.

Çalışmanın deneysel boyutunda proje ekibinden geliştiricilerle yapılan görüşmeler sonucunda en kritik dokuz sınıf belirlenip bunların aylık kod değişimlerine ait metrik ölçümleri çıkarılmıştır ve bunun sonucu olarak sınıflara ait metrik değerlerinin aylık olarak nasıl değiştiği bilgisine ulaşılmıştır.

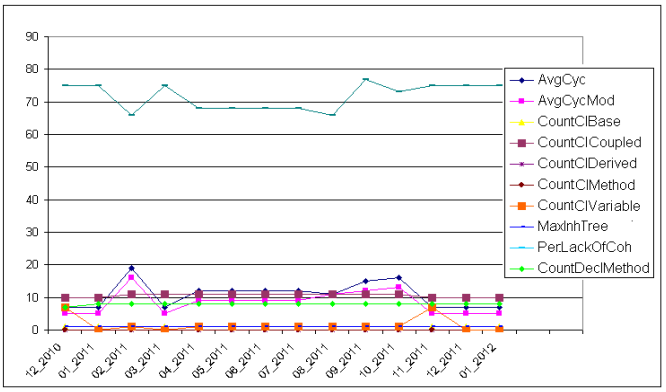
Burada amaçlanan sınıfların değişimlerinin anlaşılabilir şekilde sunulmasıdır. Seçilen 5 sınıfa ait metriklerin değişim grafikleri, Şekil 1, Şekil 2, Şekil 3, Şekil 4 ve Şekil 5'te verilmiştir.



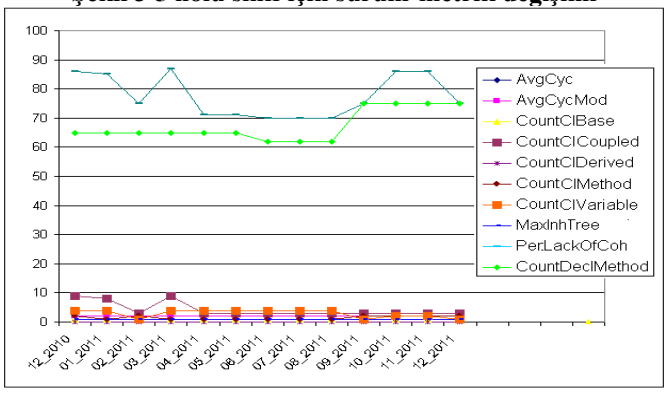
Şekil 1 2 nolu sınıf için sürüm-metrik değişimi



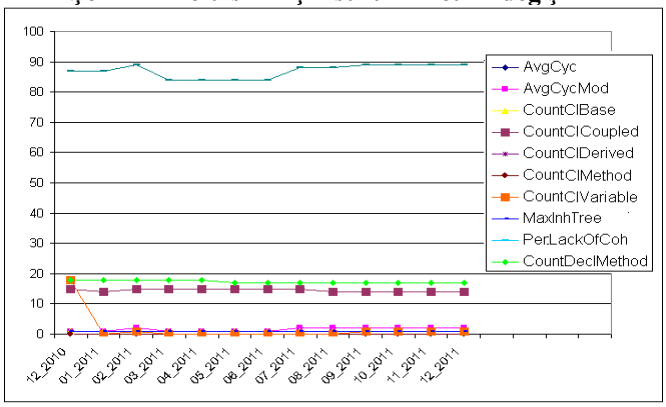
Şekil 2 4 nolu sınıf için sürüm-metrik değişimi



Şekil 3 5 nolu sınıf için sürüm-metrik değişimi



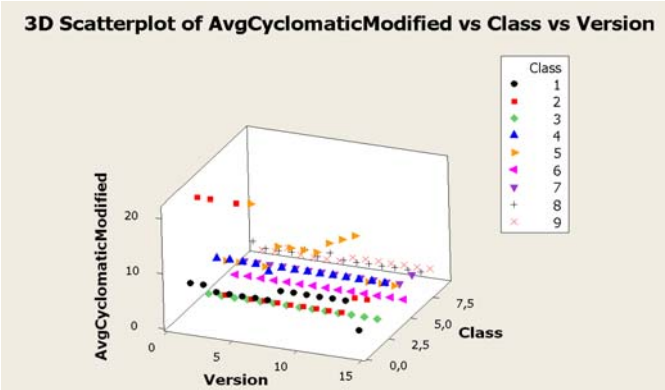
Şekil 4 7 nolu sınıf için sürüm-metrik değişimi



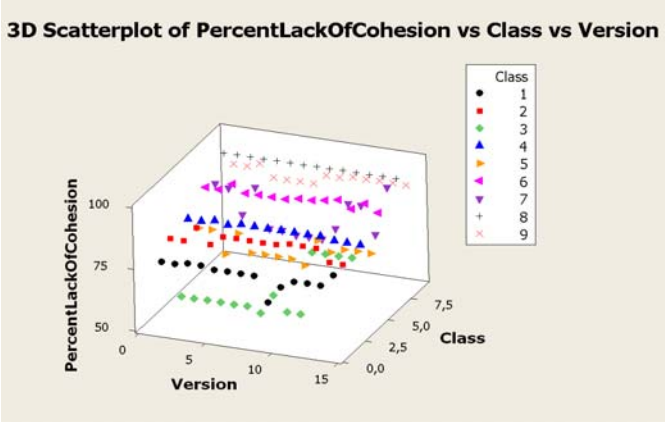
Şekil 5 9 nolu sınıf için sürüm-metrik değişimi

Sınıf ve sürüm arasında metrik değerinin nasıl değiştiğinin analizini için geliştirilen araçla metrik değerleri her sürüm için sınıf değerlerine ait metriklerin olduğu tablolar haline getirilmiş bu veriler veri madenciliği ve istatistiksel analiz araç olan MiniTAB [22] kullanılarak 3D serpme çizim (Scattter plot) matematiksel analizi ile dağılımı hesaplanmıştır.

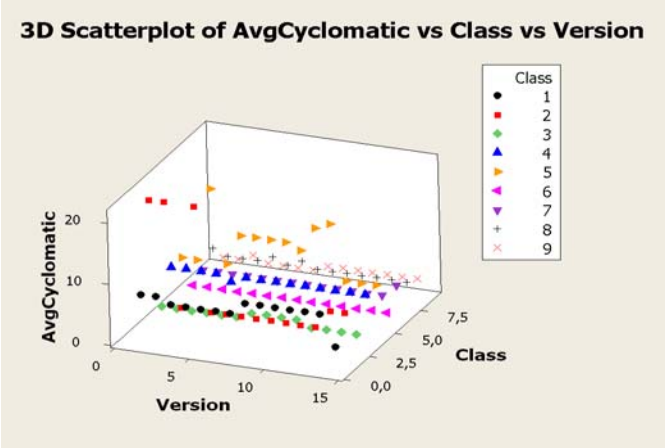
Her metriğe ait sınıf ve sürüm arasındaki korelasyon bağlı değişim Şekil 6, Şekil 7, Şekil 8, Şekil 9, Şekil 10, Şekil 11, Şekil 12, Şekil 13, Şekil 14 ve Şekil 15'te gösterilmektedir.



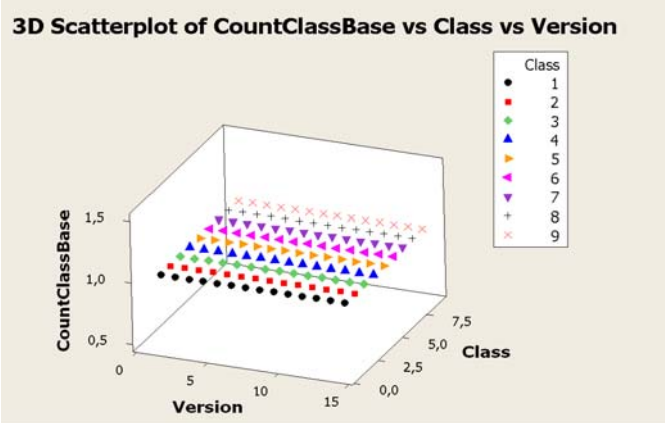
Şekil 6 Ortalama Değişen Karmaşıklık-Sınıf-Sürüm



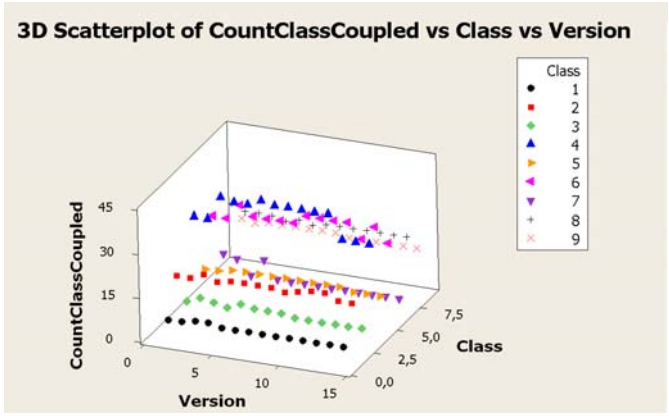
Şekil 7 Uyum Eksikliği-Sınıf-Sürüm



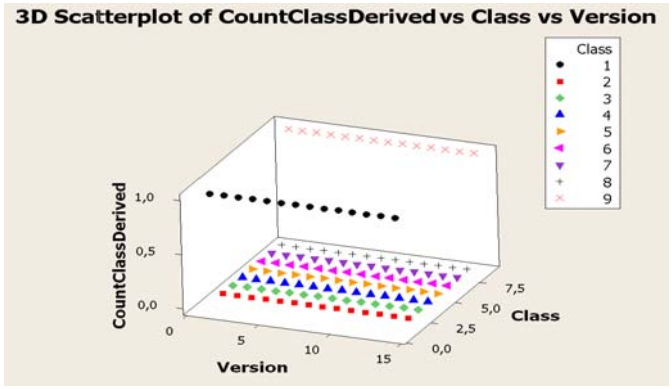
Şekil 8 Ortalama Karmaşıklık-Sınıf-Sürüm



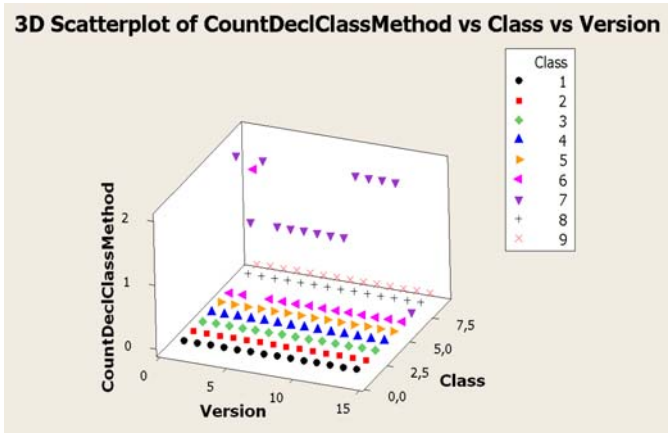
Şekil 9 Ana Sınıf Sayısı-Sınıf-Sürüm



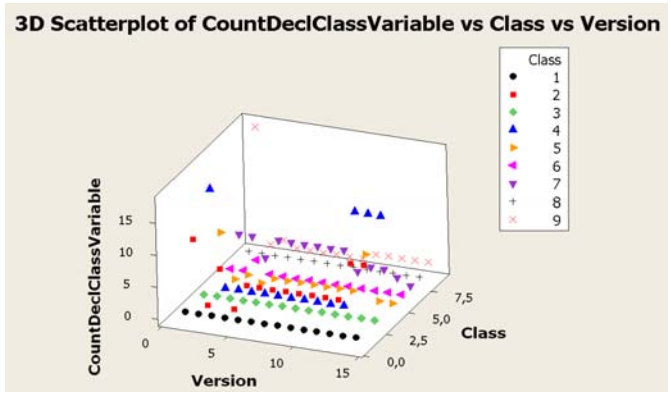
Şekil 10 Bağımlı Sınıf Sayısı-Sınıf-Sürüm



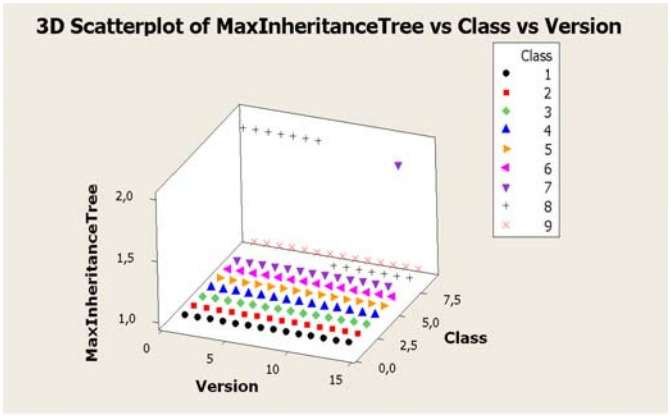
Şekil 11 Türemiş Sınıf Sayısı-Sınıf-Sürüm



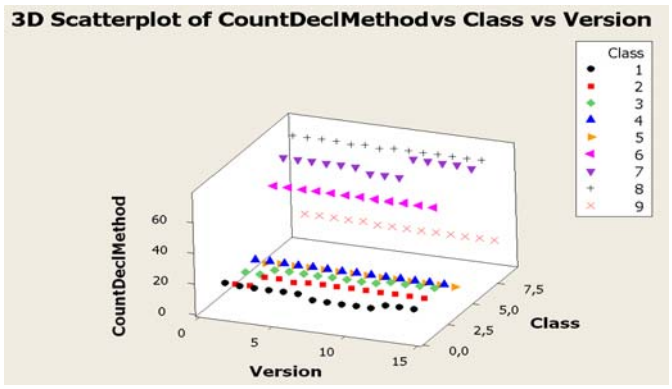
Şekil 12 Sınıf Metot Sayısı-Sınıf-Sürüm



Şekil 13 Sınıf Değişken Sayısı-Sınıf-Sürüm



Şekil 14 Kalıtım Ağacının Derinliği-Sınıf-Sürüm



Şekil 15 Yerel Metot Sayısı-Sınıf-Sürüm

Sonuçlar analiz edildiğinde ortaya çıkan analiz sonuçları, yaşam döngüsü süresince kodlama safhası için 15 ay ayrılmış projenin 10. ayından itibaren gelişiminin genel hatları ile durduğunu ve metrik değerlerinin belirli seviyede sabitlendiğini göstermektedir. Bu durumdan projenin gidişatının olumlu yönde olduğu sonucuna varılmıştır. Geliştiriciler ile konuşulduğunda tahminlerin doğru olduğu son 5 ayın test için ayrıldığı belirtilmiştir.

Yapılan incelemelerde projenin esnekliğini olumsuz yönde etkileyen sınıflar; Sınıf 4 ve Sınıf 5
Yeniden kullanılabilirliğini olumsuz yönde etkileyen sınıflar; Sınıf 2 ve Sınıf 5
Bakım kolaylığını olumsuz yönde etkileyen sınıflar; Sınıf 2 ve Sınıf 7 olarak belirlenmiştir.

Proje ekibi ile yapılan görüşmeler sonucunda Sınıf 5'e yeni bir özellik eklenmesinin hatalara yol açtığı sonucuna varılmıştır. Benzer şekilde Sınıf 2'den dolayı testlerde hata çıktığını ve üzerinde değişiklikler yapıldığı belirtilmiştir.

IV. SONUÇLAR VE GELECEK ÇALIŞMALAR

Gerçek dünyada yazılım projesinin tek bir metrik ile değerlendirilmesinin doğru olmadığı görülmektedir. Bu çalışmada yazılımın ölçülmesinde belirlenen kalite özellikleri için metrik bileşimleri kullanılarak bir model oluşturulmuş ve değerlendirmeler bunun üzerinden gerçekleştirilmiştir.

Çalışmanın sonuçlarında incelenen proje için model üzerinde oluşturulan tahminlerin gerçek dünyadaki sonuçlarla örtüştüğü gözlenmiştir. Çalışmanın devamında farklı veri madenciliği teknikleri kullanılarak otomatik olarak analiz

edebilen bilgisayar destekli yazılım aracı geliştirilmesi ve proje için belirlenen kalite özellikleri için puan vermesi amaçlanmaktadır. Böylelikle proje yürütücülerinin projenin geneline tam olarak bakabilmesine, bununla birlikte müşterinin yazılımın ne durumda olduğunu takip edebilmesine olanak sağlanması planlanmaktadır.

KAYNAKLAR

[1] Hakim Lounis,Tamer Fares Gayed, Mounir Boukadoum, "Using Efficient Machine-Learning Models to Assess Two Important Quality Factors:Maintainabilitiy and Reusability,21st International Workshop on Software Measurement,2011

[2] Tibor Bakota, Péter Heged˝us, Péter Körtvélyesi, Rudolf Ferenc, and Tibor Gyimóthy, "A Probabilistic Software Quality Model", 27th International Conference on Software Maintenance(ICSM),2011

[3] Liafna Li, Hareton Leung, "Mining Static Code Metrics for a Robust Prediction of Software Defect-Proneness", Internal Symposium on Empirical Software Engineering and Measurement,2011

[4] E.Arisholm,L.C.Briand and M.J.Fuglerud, "Data mining techniques for building fault-proneness models in telecom java software, in simula Technical Report,2007

[5] V.R.Basili,L.C.Briand and W.L.Melo, "A validation of object oriented design metrics as quality indicators", IEEE Transaction on Software Engineering, vol.22,no.10,p.751-761,Oct,1996

[6] J.Osbeck, Waverly, "Investigaton of Automatic Prediction of Software Quality", Fuzzy Information Processing Society Annual Meeting of the North America, 2011

[7] J. Ross Quinlan. C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[8] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. Witten, "The WEKA Data Mining Software: An Update", SIGKDD Explorations, vol. 11, no. 1, pp. 10-18, 2009.

[9] J. Bansiya and C.G. Davis, "A hierarchical model for objectoriented design quality assessment", IEEE Trans. Softw. Eng., vol. 28, no. 1, pp.4-17, 2002.

[10] Shaheen Khatoon, Azhar Mahmood, "An Evaluation of Source Code Mining Techniques", Eight International Conference on Fuzzy System and Knowledge Discovery (FSKD), 2011

[11] Aljahdali, Sheta , "Predicting the Reliability of Software System Using Fuzzy Logic", Information Technology: New Generations (ITNG), 2011 Eighth International Conference,2011

[12] Hakim Lounis, Tamer Fares Gayed, Mounir Boukadoum, "Machine-Learning Models for Software Quality: a Compromise Between Performance and Intelligibility", 23rd IEE E International Conference on Tools with Artificial Intelligence, 2011

[13] Ceylan, E. ,"Software Defect Identification Using Machine Learning Technique", Software Engineering and Advanced Applications, 2006. SEAA '06. 32nd EUROMICRO Conference,2006

[14] Ali R. Sharafat, Ladan Tahvildari, "A Probabilistic Approach to Predict Changes in Object-Oriented Software System", , Software Maintenance and Reengineering, 2007. CSMR '07. 11th European Conference, 2007

[15] Kayarvizhy, N. , "Analysis of Quality of Object Oriented System using Object Oriented Metrics" , Electronics Computer Technology (ICECT), 2011 3rd International Conference,2011

[16] Okumoto, K. ,"Software Defect Prediction Based on Stability Test Data", Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE), 2011 International Conference on,2011

[17] Ko-Li Cheng ,"Software Fault Detection using Program Patterns", Software Engineering and Service Science (ICSESS), 2011 IEEE 2nd International Conference on,2011

[18] http://www.scitools.com/features/metrics.php

[19] Ural Erdemir, Umut Tekin, Feza Buzluca, "Nesneye Dayalı Yazılım Metrikleri ve Yazılım Kalitesi", Yazılım Kalitesi ve Yazılım Geliştirme Araçları Sempozyumu 2008

[20] McCall,J.A.,Richards,P.K and Walters, G.F.,"Factors in Software Quality",Nat'1 Tech.Information Service,no. Vol. 1,2 and 3,1997

[21] ISO "ISO,IEC 9126",http://www.iso.org,2006

[22] http://www.minitab.com/en-US/default.aspx