Sınırlı Sayıda Kusur Verisiyle Yazılım Kusur Kestirim Aracı YAKUT

Oral Alan[1] Çağatay Çatal[2] Uğur Sevim[3] Banu Diri[4]

[1,2,3]TÜBİTAK, Marmara Araştırma Merkezi, Bilişim Teknolojileri Enstitüsü, Kocaeli
[4]Yıldız Teknik Üniversitesi Bilgisayar Mühendisliği Bölümü, İstanbul

[1,2] e-posta: {oral.alan,cagatay.catal,ugur.sevim}@bte.mam.gov.tr
[4] e-posta: banu@ce.yildiz.edu.tr

Özetçe

Yazılım kusur kestirimi, kaynak kodlar üzerinden hesaplanan yazılım metriklerini ve kusur bilgilerini kullanarak hata eğilimli modüllerin kestirimini sağlar. Kusur kestirimi için makine öğrenmesi algoritmaları kullanılmaktadır. Bu çalışmada, sınırlı sayıda kusur verisinden Naive Bayes algoritması ile kusur kestirimi için geliştirilen YAKUT isimli araç açıklanmaktadır. YAKUT, JAVA ile yazılmış projelerin metriklerini hesaplayabilmekte ve kullanıcıya çeşitli özellikler sunarak, bu metrikler ve kusur bilgileri üzerinden kusur kestiriminin yapılmasını sağlamaktadır.

1. Giriş

Yazılım mühendisliği araştırma alanında, kusur verilerinden yazılım kusur kestirimi yapabilen modellerin geliştirilmesi son yıllarda aktif bir araştırma alanı haline gelmiştir. Bu alanda yapılan çalışmalar Genetik Programlama [1], Karar Ağaçları [2], Yapay Sinir Ağları [3], Bulanık Mantık [4], Yapay Bağışıklık Sistemleri [5] gibi yöntemler kullanmakta ve bu kapsamda araştırmalar halen sürmektedir. Endüstride genel kabul görmüş bir kusur kestirim aracı henüz geliştirilmemiştir. Yazılım kusur kestirim aracı YAKUT, Eclipse geliştirme platformu tabanlı bir uyumlu ek (plug-in) olup, bu platformda JAVA programlama diliyle geliştirilmiş projeler üzerinde Naive Bayes makine öğrenmesi tabanlı bir model ile kusur kestirimi yapmak amacıyla geliştirilmiştir.

Yazılım kusur kestirim yaklaşımları, yazılım metriklerini ve kusur verisini kullanarak, yazılımın gelecek sürümleri için kusur eğilimli modüllerin tespitini gerçekleştirmektedir. Kusur verisi, bir önceki yazılım sürümünde sistem testleri sırasında modülde hata oluşmuşsa 1 değeri, oluşmamışsa 0 değeri ile temsil edilmektedir. Birçok yazılım şirketi tarafından geliştirilen büyük ölçekli yazılım projelerinde, tüm firmalar kusur verilerini toplamayabilir. Bu durumda, kusur verisi sınırlı sayıda mevcut olabilmektedir. Örneğin; modüllerin %20'sinin kusur bilgileri mevcut olabilir. Tüm kusur bilgilerinin mevcut olduğu durumda, kurulan modeller makine öğrenmesi perspektifi ile değerlendirildiğinde, eğiticili öğrenme kategorisine girmektedir. Sınırlı sayıda kusur verisi olduğu durumda ise kurulan eğiticili modellerin tüm verileri temsil etmesi mümkün olamamakta ve bu nedenle, etiketsiz verilerden de yararlanılması gündeme gelmektedir. Bu araştırma alanı, yarı-eğiticili öğrenme (semi-supervised learning) olarak makine öğrenmesi içinde yer almaktadır.

Menzies ve arkadaşları [6], eğiticili öğrenme söz konusu olduğunda kusur kestirimi için Naive Bayes algoritmasının en iyi performansı verdiğini göstermiştir. Catal ve Diri [7], sınırlı sayıda kusur verisi olduğu durumda, YATSI (yet another two stage idea) tabanlı farklı algoritmaların geliştirilmesi üzerine çalışmış ve Naive Bayes algoritmasının en iyi performansı verdiğini raporlamışlardır.

YAKUT'un hesaplayabildiği metrikler temel ve türetilmiş Halstead metrikleri [8], McCabe metrikleri [9] ve nesneye yönelik Chidamber ve Kemerer metrikleridir [10].

YAKUT, açık kaynaklı bir proje olan Lachesis [11] referans alınarak geliştirilmiş bir projedir. Lachesis, nesneye yönelik programların yazılım karmaşıklıklarını ölçen bir projedir ve Java kaynak kodlarının analizini yapabilmektedir. Lachesis kendi başına bir yazılım kusur kestirim aracı olmayıp, kullanıcılara metrikleri hakkında bilgi vermektedir. Proje kapsamında, Lachesis kaynak kodları, temel Halstead metriklerini hesaplayabilecek şekilde değiştirilmiş ve grafik arayüzündeki bazı bağımlılık metrikleri çıkartılmıştır. Lachesis üzerinde tespit edilen hatalar, düzeltilerek YAKUT'un doğru şekilde metrikleri hesaplaması sağlanmıştır. YAKUT tarafından hesaplanabilen yazılım metrikleri Tablo 1'de gösterilmektedir.

Tablo 1: YAKUT Metrikleri

Metrik Tipi	Metrik Grupları	Metrikler
Method-Seviyesindeki Metrikler	**Türetilmiş Halstead Metrikleri**	Vocabulary
		Implementation Length
		Time Equation
		Program Level
		Program Volume
		Program Difficulty
		Program Effort
		Intelligence Content
	Temel Halstead Metrikleri	Total Operand
		Total Operator
		Unique Operand
		Unique Operator
	McCabe Metrikleri	Cyclomatic Complexity
		Lines of Code
Sınıf Seviyesindeki Metrikler	**Chidamber-Kemerer Metrikleri**	Weighted Methods per Class (WMC)
		Depth of Inheritance Tree (DIT)
		Coupling between Object Classes (CBO)
		Number of Children (NOC)
		Response for a Class (RFC)
		Lack of Cohesion of Methods (LCOM)
	Sınıf Analizi	Lines of Code for Class

2. Ürünün Hedefi

Sınırlı sayıda kusur verisiyle kusur kestirimi için iyi performans verdiği raporlanan [7] Naive Bayes algoritmasının Eclipse tabanlı bir uyumlu ek haline getirilmesi hedeflenmiştir. Bu kapsamda, Java projelerinden istenen metrikleri hesaplayabilmek üzere açık kaynaklı Lachesis projesinden yararlanılmış, Lachesis içinde tespit edilen hatalar düzeltilmiş, kusur bilgilerinin editör üzerinden girilebilmesi için farklı eylemler (actions) tanımlanmış, Eclipse Preferences sayfalarına YAKUT ayarları için yeni bir sayfa eklenmiştir. Aynı zamanda, hiç kusur verisi olmadığı durumda metrik eşik seviyelerine bağlı olarak kestirim yapılabilmesi için oluşturulan YAKUT preferences sayfasına metrik eşik seviyelerinin girilebileceği arayüzler oluşturulmuştur. Metrik eşik seviyelerine bağlı olarak kusur kestiriminin yapılabilmesi için geliştirilen yöntem, NASA veri kümelerinde analiz edilmiş ve kabul edilebilir sonuçlar üretilmiştir. Bu araç, sınırlı sayıda ya da varolmayan kusur verisiyle kusur kestirimin yapılabilmesi için geliştirilmiştir.

3. Araştırma Alanı

Yazılım kusur kestirimi probleminde, kusur bilgileri tamamen mevcutsa, ilgili problem makine öğrenmesi perspektifinden "eğiticili öğrenme" kategorisine girmektedir. Literatürdeki çalışmaların tamamına yakını bu kategori üzerine yoğunlaşmıştır. Kusur verilerinin çok az sayıda bulunması durumunda ise bu problem "yarı-eğiticili öğrenme" kapsamında değerlendirilebilmektedir. Kusur bilgileri mevcut olmayan modüllerin yazılım metriklerinden de yararlanarak yeni modeller kurgulanabilmektedir ancak sınırlı sayıda kusur verisi ile model oluşturmak da diğer bir alternatif yöntemdir. Catal ve Diri [7], yaptıkları çalışmada Naive Bayes algoritmasının sınırlı sayıda kusur verisi için de iyi performans verdiğini, YATSI (yet another two stage idea) ile birlikte kullanıldığında Naive Bayes'in performansının azalabildiğini de tespit etmişlerdir. Bu çalışmada da gösterildiği gibi, her durumda etiketsiz verilerden yararlanmak iyi sonuçlar üretmeyebilir. Kusur verisi hiç mevcut değil ise, problem "eğiticisiz öğrenme" bakış açısı ile ele alınabilir. Bu durumda, bir yöntem olarak tüm modüller kümelere ayrılabilir (clustering) ve her kümenin temsilcisi bir uzman tarafından incelenerek kümenin kusur eğilimli ya da değil şeklinde etiketlenmesi sağlanabilir. Catal ve arkadaşları [12], kümeleme algoritmasının ardından her kümenin ortalama vektörü olarak ifade edilen temsilcisini, metrik eşik vektörü ile karşılaştırarak eşiklerin aşıldığı kümeleri kusur eğilimli olarak işaretlemişlerdir. Geliştirildikleri modeli, Boğaziçi Üniversitesi SOFTLAB'ın topladığı veri kümesinde analiz ederek başarılı sonuçlar elde etmişlerdir.

4. Ürünün Mimarisi

Bu bölümde Eclipse tabanlı yazılım kusur kestirim uyumlu eki olan YAKUT'un mimarisi anlatılmaktadır. Bölüm 4.1 Eclipse platformu hakkında bilgi vermekte, Bölüm 4.2 YAKUT uyumlu eki ve seçeneklerini, Bölüm 4.3. YAKUT akış diyagramını açıklamaktadır.

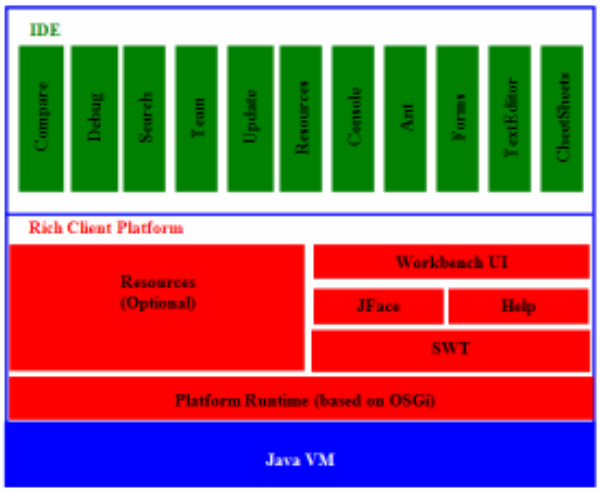
4.1. Eclipse Platformu

Eclipse, F/OSS (*Free and Open Source Software*) topluluğu tarafından geliştirilmiş, genişletilebilir bir yazılım geliştirme platformudur. IBM, Object Technology International (*OTI*) ve sekiz firma Kasım 2001 yılında Eclipse platformunu duyurmuşlardır. Eclipse uyumlu eklerle genişletilebilen bir platformdur.

Şekil 1'de Eclipse platformunun üç katmanı görülmektedir: tümleşik geliştirme ortamı (*IDE*), zengin istemci platformu (*Rich Client Platform*) ve Java sanal makinesi.

Zengin istemci platformunun içerisinde şu katmanlar yer almaktadır:

- *Platform runtime:* Çalışma zamanı OSGi (Open Services Gateway Initiative) teknolojisi üzerine kuruludur ve gömülü yazılımlar için Java bileşenleri ve servis modelini sunar.
- *SWT (Standart Widget Toolkit) :* SWT, SWING gibi bir grafiksel kütüphane olup, kullanıcı arayüzü bileşenleri, butonlar, menüler ve renkler gibi temel grafiksel kullanıcı arayüzü bileşenlerini sağlamaktadır.
- *JFace:* Jface, sıkça kullanılan kullanıcı arayüzü bileşenlerini daha üst düzeyde tanımlamaktadır. Örneğin; Preferences, sihirbazlar, dialog pencereleri gibi bileşenler JFace ile tanımlanabilmektedir.
- *Help:* Yardım bileşeni, araçlar ve uyumlu eklerle ilgili yardım alt yapısını sağlamaktadır.
- *Workbench User Interface (UI):* Editörler, görünümler (*view*) ve perspektifler bu katmanda yer alır. Eğer kullanıcı bir penceredeki veriyi değiştirebiliyorsa, bu pencereler editör olarak adlandırılmakta, aksi durumda bu pencerelere görünüm adı verilmektedir. Perspektif ise birçok editör ve görünümün bir araya gelmesiyle oluşur.
- *Resources:* Bu uyumlu ek, dosya, dizin, proje gibi elemanların çalışma uzayında (workspace) oluşturulmasını sağlayan alt yapıdır.



Şekil 1: Eclipse Platformu

4.2. YAKUT Aracı ve Seçenekleri

YAKUT kusur kestirim aracı, aşağıdaki uyumlu ekleri içermektedir:

- *gov.mam.bte.ruby.eclipse.plugin:* Bu uyumlu ek metrikleri hesaplar ve kullanıcıyla etkileşimi sağlar. Aracın temel uyumlu ekidir.
- *gov.mam.bte.ruby.thirdPartyLibraries:* Bu uyumlu-ek varolan dış kaynaklı jar arşiv dosylarının bir araya getirilmesiyle oluşturulmuştur. Bu arşiv dosyaları SableCC ve log4j projelerini içerir. log4j programlarda kayıt alma için SableCC ise Java sözcüksel analizi (*lexical analysis*) ve ayrıştırma (*parsing*) işlemleri için kullanılır.
- *gov.mam.bte.ruby.help:* Bu uyumlu ek kullanıcıya html dosyaları üzerinden yardım sağlar.

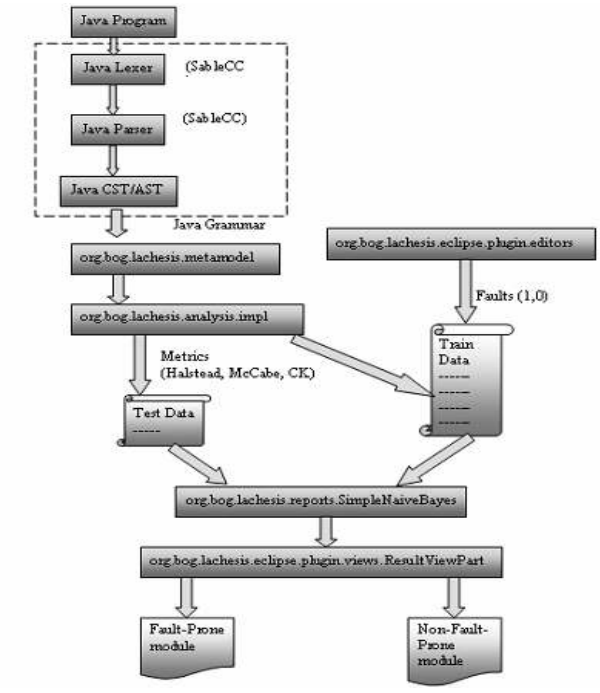
Kullanıcılar YAKUT ayarlarını *Eclipse Preference Pages* menüsünden yapabilirler. Bu amaçla YAKUT aracında *RUBY Metrics* isimli bir sayfa hazırlanmıştır:

- *RUBY Metrics:* Bu sayfa dosyalarda hangi metriklerin saklanacağının belirlendiği sayfadır. Eğer *train* dosyası sistemde henüz yoksa "*Create Train Files*" seçeneği seçilmelidir.
- *Additional Settings:* Bu sayfa, metriklerin uzun halleri ya da kısaltmaları ile gösterilmesini sağlar .
- *Analysis Packages:* Kusur kestiriminde kullanılacak metrikler bu sayfadan seçilir.
- *Class Metric Levels:* Kusur kestirim modelleri çoğu durumda modüllerin bir önceki kusur bilgilerini kullanır, ancak bazen hiç kusur bilgisi bulunmayabilir. Bu durumda, metrik eşik seviyesi aşılan metrikler kırmızı işaretlerle kullanıcılara bildirilir. Bu sayfa sınıf seviyesinde metriklerin eşik seviyelerinin belirlendiği sayfadır. YAKUT aracı McCabeIQ aracıyla aynı eşik seviyeleri kullanmaktadır. Bu değerler, kullanıcı tarafından değiştirilebilmektedir.
- *Method Metric Limits:* Bu sayfa benzer şekilde metod metrikleri için seviyelerin ayarlandığı sayfadır. YAKUT aracı Predictive aracıyla aynı seviyeleri kullanmaktadır. Bu değerler, kullanıcı tarafından değiştirilebilmektedir.
- *Prediction Modes:* Bu sayfadan, kusur kestiriminin hangi seviyede (sınıf veya metot) yapılacağı seçilmektedir.

4.3. YAKUT Akış Diyagramı

YAKUT aracı Lachesis uyumlu ekini temel alarak geliştirilmiş Java uyumlu ekidir. Java grameri açık kaynaklı bir proje olan SableCC [13] kütüphanesi üzerine kurulmuştur. SableCC Java dili için bir ayrıştırıcıdır (*parser*). YAKUT akış diyagramı Şekil 2'de görülmektedir.

Java Lexer bileşeni karakter dizilerini Java token'lara dönüştürür, *Java Parser* bileşeni ise SableCC kütüphanesini kullanarak sözdizimsel analiz yapmaya yarar. *Java CST/AST (Concrete / Abstract Syntax Tree)* bileşeni, üzerinde çalıştığımız soyut sözdizimsel ağacı (*abstract syntax tree*) oluşturmayı sağlar. Metrikler *org.bog.lachesis.impl* paketinde hesaplanır ve kullanıcı etkileşimi *org.bog.lachesis.eclipse.plugin.editors* paketiyle gerçekleşir. Kullanıcıdan gelen kusur verisi, *train* dosyasına, editör üzerinde tanımlanmış eylemler (*actions*) kullanılarak yazılır. Naive Bayes gerçeklemesi *train* ve *test* dosyalarını değerlendirir ve sonuçların *ResultViewPart* bölümünde görüntülenmesini sağlar.



Şekil 2: YAKUT Akış Diyagramı

5. Teşekkür

Bu bildiri, TÜBİTAK tarafından desteklenen 107E213 numaralı araştırma projesinde elde edilen bilgiler kullanılarak hazırlanmıştır. Yayındaki hiçbir görüş, tespit ve kanaat, TÜBİTAK'ın resmi görüşü değildir.

6. Kaynakça

[1] Evett, M., Khoshgoftaar, T., Chien, P., Allen, E., "GP-based software quality prediction", *Proceedings of the 3rd Annual Genetic Programming Conference,* San Francisco, CA, sf. 60-65, 1998.

[2] Khoshgoftaar, T.M. ve Seliya, N., "Software quality classification modelling using the SPRINT decision tree algorithm", *Proceedings of the 4th IEEE International Conference on Tools with Artificial Intelligence*, Washington, DC., sf. 365-374, 2002.

[3] Thwin, M.M. ve Quah, T., "Application of neural networks for software quality prediction using object-oriented metrics", *Proceedings of the 19th International Conference on Software Maintenance*, Amsterdam, The Netherlands, sf. 113-122,2003.

[4] Yuan, X., Khoshgoftaar, T.M., Allen, E.B., Ganesan, K., "An application of fuzzy clustering to software quality prediction", *Proceedings of the 3rd IEEE Symp. On Application-Specific Systems and Software Eng. Technology*, IEEE Computer Society, Washington, DC, sf. 85-90, 2000.

[5] Catal, C. ve Diri, B., "Software fault prediction with object-oriented metrics based artificial immune recognition system", *Proceedings of the 8th International Conference on Product Focused Software Process Improvement,* Lecture Notes in Computer Science 4589, Riga, Latvia, sf. 300-314, 2007.

[6] Menzies, T., Greenwald, J., Frank, A., Data mining static code attributes tolearn defect predictors, IEEE Transactions on Software Engineering, 33(1) 2-13, 2007.

[7] Catal, C. ve Diri, B., "Unlabeled extra data does not always mean extra performance for semi-supervised fault prediction", *Expert Systems: The Journal of Knowledge Engineering*, Kabul Tarihi: Ekim 2008, Baskı Aşamasında.

[8] Halstead, M., Elements of Software Science. Elsevier, New York, 1977.

[9] McCabe, T., "A complexity measure", *IEEE Transactions on Software Engineering*, 2(4), 308-320, 1976.

[10] Chidamber, S.R., Kemerer, C.F., A Metrics suite for Object Oriented design. M.I.T. Sloan School of Management, E53-315, 1993.

[11] Vatkov B.I., Program for automatic analysis of software metrics, Diploma Thesis, Faculty of Computer Systems and Control, Technical University of Sofia, Sofia, Bulgaria, 2005, http://lachesis.sourceforge.net

[12] Catal, C., Sevim, U., Diri, B. "Clustering and Metrics Thresholds based Software Fault Prediction of Unlabeled Program Modules", *6th International Conference on Information Technology: New Generations, Software Engineering Track,* 2009, Las Vegas, Nevada, U.S.A

[13] www.sablecc.org