Yazılım Geliştirmede Erken Aşamalarda Toplanan Verinin Hata Tahmini Performansına Etkisi

Rana Özakıncı[1] ve Ayça Tarhan[2]
[1] rana.ozakinci@tubitak.gov.tr
TÜBİTAK - BİLGEM - Yazılım Teknolojileri Araştırma Enstitüsü (YTE), Ankara, Türkiye
[2] atarhan@hacettepe.edu.tr
Hacettepe Üniversitesi Bilgisayar Mühendisliği Bölümü, Ankara, Türkiye

Özet. Yazılım kalitesinin en önemli göstergelerinden biri geliştirme yaşam döngüsü boyunca gözlenen yazılım hatalarının eğilimidir. Yazılım hata tahmini modelleri, yazılım ürünlerinin hatalılık durumlarını gözlemlemeyi ve ileriye dönük hata eğilimlerini tahmin etmeyi mümkün kılar. Güncel hata tahmini modelleri genellikle yazılım kodlama aşamasına ilişkin dinamikleri ve ölçümleri kullanarak geliştirilmektedir. Bu durum, gereksinim analizi veya tasarım gibi yazılım geliştirme yaşam döngüsünün erken aşamalarına ait bilgilerin işlenememesine, dolayısıyla maliyet düşürme, etkin kaynak planlama gibi önleyici eylemlerden erken aşamalarda faydalanamamaya neden olmaktadır. Bu çalışmada, yazılım geliştirme yaşam döngüsünün gereksinim analizi veya tasarım aşamaları gibi erken aşamalarında hata tahmini yapan çalışmalar sistematik eşleme yöntemi ile incelenmiştir. Daha sonra, erken aşamalarda toplanan verilerin hata tahmin modellerinin performansına olan etkisini araştırmış çalışmalar belirlenmiş ve sistematik literatür tarama yöntemi ile incelenerek çeşitli niteliklerine göre sınıflandırılmıştır.

Anahtar Kelimeler: Erken, hata tahmini, yazılım hatası, yazılım güvenilirliği, yazılım kalitesi, hata tahmin modeli, sistematik eşleme.

Abstract. Defectiveness of the software can be seen as one of the most important factors that address software quality. Software defect prediction models enable to observe the defectiveness of the software as well as to estimate the forward looking trend of defects. Most defect prediction models are built according to the coding phase based metrics. Therefore, chances for taking preventive actions early in the lifecycle such as cost reduction or effective resource planning are missing due to the lack of information analysis in requirement or design phases. This paper provides a systematic mapping research of the literature in order to identify the early defect prediction studies. In addition, studies that examine the impact of the early lifecycle information to the performance of the software defect prediction models were selected and investigated with systematic literature review approach.

Keywords: Early, defect prediction, software defect, software reliability, software quality, prediction model, systematic mapping.

1 Giriş

Yazılım geliştirme, takvim baskısı, limitli bütçe ve farklı insan davranışlarını içeren zor ve zahmetli bir iştir. Yoğun iş gücü gerektiren şartlarda geliştirilen çoğu yazılım, geliştirme sürecinin değişik evrelerinde çeşitli türlerde hata ile karşılaşılma riskini taşır. Bu hataların, önemsiz veya kritik sonuçlara neden olabilecek şekilde çok çeşitli etkileri olabilir. Bu nedenle, yazılım sistemlerinde kalite güvenceyi sağlamak, yazılımın güvenilirliğini garanti altına almak ve hataların etkisini en aza indirebilmek, geliştirme yaşam döngüsünün erken aşamalarından itibaren önem taşır.

Yazılımın en kaliteli şekilde geliştirilmesi hem müşteri hem de geliştiriciler açısından son derece önemlidir. Bununla beraber, yazılımın güvenilirliğinin sağlanabilmesi ve yazılımın çalışması sırasında oluşabilecek hataların en aza indirilebilmesi için, yazılım kalitesini ön aşamalarda tahmin edebilmek, bu doğrultuda önleyici eylemlerde bulunabilmek ve düzeltici önlemler alabilmek, yazılım kalitesinde optimum düzeye ulaşmak için en etkili ve sağlıklı yöntemlerden biridir [1].

Literatürde yazılım kalite tahminine yönelik birçok çalışma mevcuttur. Ayrıca endüstriyel alanda yazılım geliştiren büyük ve kurumsal organizasyonlar da yazılım kalite tahmini modellerini kullanarak projelerin erken safhalarında hataları önleyici etkinliklerde bulunmaktadır [2]. Yazılımın kalitesinin tahmini için uygulanan en temel yöntem yazılım hatalılığın tahmininin yapılmasıdır. Yazılım geliştirme yaşam döngüsü boyunca oluşabilecek hataları önlemek ve/veya en aza indirebilmek, yazılım kalite tahmininin temelini oluşturur [3]. Yazılım güvenilirliğini ve hatalarını tahmin etmeye yönelik mevcut modeller çoğunlukla kodlama ve test aşamalarına ait ölçümleri kullanarak geliştirilmektedir. Fakat yazılım geliştirme yaşam döngüsünde kodlama ve test aşamalarına gelindiğinde, düzeltici ve önleyici faaliyetler planlamak için çok geç kalınmış olabilir. Bu soruna çözüm olarak, yazılım yaşam döngüsünün gereksinim analizi veya tasarım gibi erken aşamalarında hata tahmini yapabilmek, kalite kestirimi ve etkin kaynak, takvim ve maliyet planlaması açısından uygun olacaktır [3]. Bunların yanında, erken tahmin modelleri, geliştirmenin ilk aşamalarında etkin karar destek ve dengeleme analizi (trade-off analysis) mekanizması olarak da kullanılabilir [4].

Yazılım geliştirme yaşam döngüsünün gereksinim analizi, tasarım ve kodlamanın erken aşamaları gibi evrelerinde, yazılımın sonraki aşamalarında oluşabilecek hataları ve/veya hata-eğimli olabilecek modülleri kestirebilmek, geleceğe yönelik etkin kalite planlamaları yapabilmek adına oldukça önemlidir. Bu çalışmanın temel hedefi, yazılım geliştirme yaşam döngüsünün erken aşamalarına ait bilgileri, kodlama aşamasından toplanan bilgilere ek olarak kullanarak hata tahmini yapmış modellerin performansındaki değişimi incelemektir. Bu kapsamda öncelikle, yazılım geliştirme yaşam döngüsünün erken aşamalarında hata tahmini yapmış olan akademik çalışmalar sistematik eşleme yöntemi ile incelenmiştir. Daha sonra, erken aşamalardan toplanan verilerin kodlama aşamasından elde edilen verilerle oluşturulan modellere girdi olarak kullanılmasının modelin performansına olan etkisi araştırılmıştır. Bu kapsamda belirlenen çalışmalar, sistematik literatür tarama yöntemi ile detaylı bir şekilde incelenmiş ve kategorize edilmiştir.

Bildirinin ikinci bölümünde yazılım hata tahmini alanında yapılmış akademik çalışmalardan ve bulgularından bahsedilmiştir. Üçüncü bölümde, bu bildiride

kullanılmış araştırma yönteminin tasarımına değinilmiştir. Dördüncü bölümde araştırma sorularına ilişkin yanıtlar verilmiş, bulgular paylaşılmıştır. Son bölümde ise çalışma özetlenmiş, sonraki dönemlerde yapılacak çalışmalar aktarılmıştır.

2 Yazılım Hata Tahmini ve İlişkili Çalışmalar

IEEE Standard Classification for Software Anomalies [5] standardında, yazılım hatalarına ilişkin terimler için ortak bir sözlük oluşturulmuştur. Standarda göre hata (defect); yazılım geliştirme yaşam döngüsünün erken aşamalarındaki iş ürünlerinde rastlanabilen, iş ürününün gereksinimleri karşılayamamasına neden olan, düzeltilmesi veya değiştirilmesi gereken çeşitli kusurlar veya eksiklikler olarak tanımlanır. Hata terimiyle benzer anlamda kullanılan diğer terimlere ilişkin tanımlar ise şu şekildedir:

- Yanlışlık (Error): Doğru olmayan sonuçlara neden olabilen insan eylemi
- Kusur (Fault): Yazılım programı içinde arızaya neden olabilen temel yanlışlıklar
- Arıza (Failure): Program davranışının kullanıcı beklentilerinden sapması, üründen beklenen işlevin belirlenen gereksinimler ve limitler altında yerine getirilememesi
- Sorun (Problem): Kişinin bir sistemi kullanırken karşılaştığı zorluk, çözülmesi gereken olumsuz durum

Bu tanımlardan yola çıkarak, yaşam döngüsünün erken aşamalarında karşılaşılabilecekaksaklık, bozukluk ve aykırılıklar çalışma boyunca "hata" terimi kullanılarak ele alınmıştır [6].

Hata tahmin modelleri, yazılım ürününe ve/veya süreçlerine ait çeşitli parametreler (metrikler, hata verileri vb.) yardımıyla gelecekte ortaya çıkabilecek hataların öngörülmesini sağlamaktadır. Yazılım hata tahmininin başarıyla gerçekleştirilmesi sayesinde, yazılımın hataya eğilimli bileşenleri önceden tespit edilebilecek ve böylece bu modüllere daha çok test ve bakım kaynağı ayrılarak uygulamada oluşabilecek kritik hataların önceden düzeltilebilmesi sağlanacaktır [7].

2.1 Hata Tahmini Üzerine Sistematik Araştırma Çalışmaları

Literatürde yazılım hata tahmini alanında yapılmış sistematik araştırma çalışmaları, genel olarak kodlama aşaması ve sonrasında elde edilebilen verileri kullanarak oluşturulmuş hata tahmini modellerini incelemiştir. Çatal ve Diri [7] 2009 yılında yaptıkları çalışmada, 74 tane çalışmayı metrik tipleri, tahminleme yöntemleri ve verisetleri bazında incelemiş ve PROMISE veri setinin açık olarak yayınlandığı 2005 yılından itibaren makine öğrenmesi yöntemlerinin ve açık verilerin kullanımında bir artış gözlemlendiğini raporlamıştır. Çatal [8] 2011 yılında yaptığı çalışmada yazılım hata tahmini ile ilgili 1990 ve 2009 yılları arasında yayınlanmış 90 tane makaleyi araştırmış ve yayınlandığı yıllara göre sınıflandırmıştır. Bu çalışmanın en önemli katkısı, bu konuda araştırma yapmak isteyen araştırmacılara metrikler, yöntemler, veri kümeleri, performans değerlendirme kriteri ve deneysel sonuç bakış açılarından yardımcı olmasıdır. Ayrıca çalışmada konu ile ilgili son yaklaşımlar anlatılıp

değerlendirilmektedir. 2012 yılında Hall vd. [9], hata tahmini modellerinin performansını araştırmaya yönelik sistematik gözden geçirme makalesinde, 2000 ile 2010 yılları arasında yayınlanmış 208 adet deneysel çalışmayı incelemiştir. Çalışmanın temel amacı, yazılım bağlamının, hata tahmin tekniklerinin ve bağımsız değişkenlerin, hata tahmin modellerinin performansına olan etkilerini değerlendirmek olarak belirtilmiştir. Çalışmanın sonuçlarına göre, Naïve Bayes ve Logistic Regression gibi basit modelleme tekniklerinin daha iyi performans gösterdiği raporlanmıştır. Ayrıca farklı yazılım metriklerinin birlikte kullanımının da model performansını arttırdığı belirtilmiştir. Wahono [10] 2015 yılında araştırma eğilimlerini, veri setlerini, yöntemleri ve çatıları analiz ederek, 2000 ile 2013 yılları arasında yayınlanmış 71 adet hata tahmini çalışmasını incelemiştir. Çalışmaların %77'sinin sınıflandırma modeli kullandığı, %64'nün ise açık veri kümeleri ile deney yaptığı raporlanmıştır. Yazarlara ait bir önceki çalışmada [11] 2000 ile 2015 yılları arasında yayınlanmış 41 adet erken hata tahmini çalışması incelenmiştir. Bu çalışmada modellerde kullanılan süreç özellikleri ve metriklerine odaklanılmıştır. Çalışma sonuçlarına göre 41 adet erken hata tahmin çalışmasından 18 tanesinin süreç bazlı verileri kullanarak hata tahmin modeli oluşturduğu görülmüştür. Ayrıca incelenen çalışmaların %31'inde support vector machine, artificial neural networks, genetic algorithm, K-means clustering ve decision trees gibi makine öğrenmesi metodlarının kullanıldığı raporlanmıştır.

2.2 Erken Aşamalarda Toplanan Veriyi Kullanan Araştırma Çalışmaları

Literatürde bulunan yazılım hata tahmini modellerini sistematik olarak incelemiş araştırmalarda [7][8][9][10], incelenen çalışmalarının sadece %5'i erken aşamalara ilişkin veri kullanarak hata tahmini yapmaktadır. Bu durum, erken aşamaları ele alarak hata tahmini yapan çalışmaların azlığını göstermektedir. Bunlara ek olarak aşağıdaki çalışmalar, yazılım geliştirmenin erken aşamalarında kullanılan verinin elde edilen sonuçlara olumlu etkileri üzerine odaklanmıştır.

Tarhan ve Demirörs [12], gereksinim aşamasını takiben test tasarımı sürecinden ve test geliştirme boyunca yapılan gözden geçirmelerden toplanan süreç verilerinin, süreç üretkenliğine ve ürün kalitesine olan etkisini görmek adına bir çalışma sunmuşlardır. Sonuçlara göre, süreç verisi analizinin süreç üretkenliğine bir etkisi olmazken, test prosedürlerinin kalitesini olumlu şekilde etkilediği görülmüştür.

Aslan vd. [13], küçük ölçekli bir yazılım şirketinde süreç işletim verisinin ürün hatalılığı üzerine olan etkisini araştıran bir durum çalışması raporlamışlardır. Çalışmanın sonuçları, süreç işletim verisini analize dahil etmenin tahmin doğruluğunu arttırdığını göstermiştir.

Söylemez ve Tarhan [14], süreç işletim verisinin toplanması ve analizini, dikey hata sınıflandırması (orthogonal defect classification) tekniği ile birleştiren bir yöntem sunmuştur. Geliştirme performansında ve ürün kalitesinde bir iyileştirme olup olmadığını görebilmek adına, yöntemin uygulanmasından önce ve sonra hata özellikleri analiz edilerek karşılaştırılmıştır. Karşılaştırma sonuçlarına göre, önerilen yöntemin verimli ve etkili olduğu belirlenmiştir. Doğrulama ve geçerleme etkinlikleri gibi hata tetikleyicilerin, geliştirme yaşam döngüsünün erken aşamalarında yazılım

hatalarının tespit edilmesi açısından etkin olduğu, buna bağlı olarak yazılım hatalarının maliyetinde bir düşüş görüldüğü ifade edilmiştir.

Yazılımın erken aşamalarında hata tahmini yapan çalışmaların incelendiği sistematik bir çalışma olarak sadece bir önceki çalışmamız [11] literatürde yer almaktadır. Bu çalışma dışında erken hata tahmin modellerinin incelendiği sistematik bir gözden geçirme çalışması bulunmamaktadır. Dolayısıyla şimdiki çalışmamız, erken aşamalara özelleştirilmiş modellere ve hata tahmin performansına etkilerine odaklanmak suretiyle yazılım hata tahmini araştırma alanına özgün bir katkı sağlamaktadır.

3 Araştırma Yöntemi

Sistematik eşleme çalışmaları, belirlenen özel bir araştırma alanında yapılmış literatür çalışmalarını yüzeysel bir bakış açısıyla inceleyerek, konu hakkındaki genel kanıtları toplamayı amaçlar [15]. Sistematik literatür tarama çalışmaları ise, konu hakkında daha özel araştırma sorularına odaklanmak suretiyle literatürü detaylı bir şekilde incelemeyi ve sonuçları yorumlamayı amaçlar [16]. Bu çalışmada hedeflenen kapsam için, sistematik eşleme çalışmasını takiben sistematik literatür tarama uygulanmıştır. İlk olarak, sistematik eşleme yöntemi temel alınarak erken aşamalarda uygulanabilen hata tahmin modellerini ele almış akademik yayınlara odaklanılmış, alanda yapılmış temel çalışmalar belirlenmiş ve bunların genel görünüşü verilmiştir. Ardından, erken aşamalara ilişkin bilgilerin mevcut hata tahmin modellerinin performansına etkisini değerlendirmiş olan çalışmalar seçilmiş, sistematik literatür tarama yöntemi temel alınarak daha detaylı şekilde incelenmiştir.

3.1 Araştırma Soruları

Çalışmada, yazılım geliştirme yaşam döngüsünün erken aşamalara ilişkin verileri kullanarak hata tahmini yapmış modeller sistematik eşleme yöntemi ile incelenmiştir. Bu modeller içinden, hem erken aşama hem de kodlama aşamasına ilişkin verileri kullanarak hata tahmini yapmış olan modeller sistematik literatür tarama yöntemi ile daha detaylı bir şekilde incelenmiştir. Bu sayede, ilgili çalışmalarda erken aşamalara ilişkin verileri kullanan modellerin hata tahmini performansına etkisi görülebilecektir. Bu doğrultuda literatürdeki çalışmaları sınıflandırmak için aşağıdaki araştırma soruları belirlenmiştir. AS1 ve AS2 sistematik eşleme çalışması kapsamında, AS3 ve AS4 sistematik literatür taraması kapsamında ele alınmıştır.

AS1: Erken hata tahmini hakkında yapılan çalışmaların eğilimi ne yöndedir?

AS1.1: Yayınların yıllara göre dağılımı nasıldır?

AS2: Erken hata tahmin modelleri hangi yazılım geliştirme yaşam döngüsü (YGYD) aşamaları ile ilişkilidir?

AS3: Gereksinim ve tasarım aşamalarından elde edilen verileri, kod aşamasında elde edilen verilerle birlikte kullanan modellerin tasarımı nasıldır?

AS3.1: Kullanılan hata tahmini yöntemleri nelerdir?

AS3.2: YGYD bazında kullanılan metrikler nelerdir?

AS3.3: Modellerin performans değerlendirmesi hangi ölçümlerle yapılmıştır?
AS3.4: Model hangi araç/ortam(lar) üzerinde uygulanmıştır?
AS3.5: Çalışmalarda kullanılan veri setleri nelerdir?

AS4: Gereksinim ve tasarım aşamalarından elde edilen veriler kod aşamasından elde edilen verilerle birlikte kullanıldığında tahmin modelinin performansına etkisi olmuş mudur? Performansa etkisi varsa ilgili çalışmada raporlanmış mıdır?

3.2 Sorgu Metinleri

Hata tahmin modelinin YGYD'nin erken aşamalarında uygulandığı bilgisini elde edebilmek amacıyla, yayının başlığında, özetinde veya anahtar kelimelerinde "early" ve "earlier" kelimeleri aranmıştır. Sorgu metinleri aşağıda belirtildiği gibidir:

Title: (("early" or "earlier") and
Title – abstract – keywords: ("software defect" or "software fault" or "software reliability") and ("prediction" or "estimation" or "analysis"))

2000 ile 2015 yılları arasında yayınlanmış çalışmalar için aşağıda belirtilen dijital kütüphanelerde aramalar yapılmıştır. İlk olarak dönen sonuç sayısı ve ilk aşamada seçilen yayın sayısı Tablo 1'de verilmiştir.

Tablo 1. Dijital Kütüphane Aramalarından Çekilen Çalışma Sayıları

Dijital Kütüphane	İlk Arama Sonucu	İlk Aşamada Seçilenler
ACM	5	1
ScienceDirect	3	1
Scopus	64	20
SpringerLink	28	16
Web of Science	25	9
Wiley	4	1
Toplam	**129**	**48**

3.3 Çalışma Seçim Kriterleri

Sistematik eşleme için kullanılacak birincil çalışmaları dahil etme ve eleme kriterlerimiz Tablo 2'de belirtilmiştir. Bu kriterlere göre 41 adet yayın sistematik eşleme çalışmasına dahil edilmiştir. Daha sonra AS3 ve AS4 araştırma sorularına yanıt vermek üzere 6 adet çalışma seçilmiş ve bu çalışmalar sistematik literatür tarama yöntemiyle detaylı olarak incelenmiştir.

Tablo 2. Dahil Etme/Eleme Kriterleri

Dahil Etme Kriterleri		Eleme Kriterleri
K1	Hata sayısı, hata eğilimi, hatalılık oranı gibi hata odaklı tahmin yapan yayınlar	Var olan hataları tespit eden veya yerini belirleyen yayınlar
K2	Gereksinim ve/veya tasarım aşamasına değinmiş yayınlar	Gereksinim ve/veya tasarım aşamasına hiç değinmemiş, sadece kodlama ve sonrasına ilişkin verileri ele alan yayınlar

3.4 Veri Çıkarımı

Araştırma sorularını yanıtlamaya yönelik verileri çıkarma işlemi, çalışmaların çeşitli özelliklere göre sınıflandırılmasıyla yapılmıştır. İlk olarak AS2'yi yanıtlamak amacıyla tüm çalışmalar gözden geçirilmiş ve YGYD fazına göre sınıflandırılmıştır. Daha sonra bu çalışmalar içerisinden kodlama aşamasına ilişkin verileri kullanan modellere, erken aşama (gereksinim ve/veya tasarım) verisini de ekleyerek hata tahmini yapan çalışmalar belirlenmiştir. Bu çalışmalar AS3 ve AS4'ü yanıtlamak üzere detaylıca incelenmiş ve Tablo 3'de görülen ilgili kategorilere göre sınıflandırılmıştır.

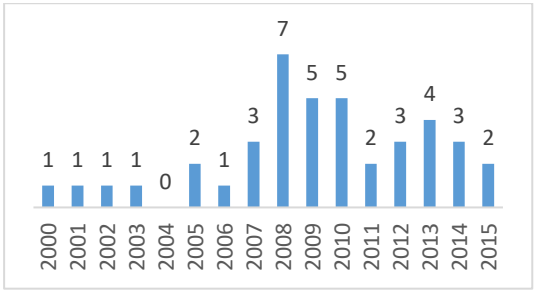
Tablo 3. Analiz için çıkarılan veriler

Veri Tipi	Veri Kümesindeki Değerler	Açıklama	İlişkili AS
YGYD Fazı	Gereksinim, Tasarım, Kodlama	Hata tahmin modeli için gerekli girdilerin elde edildiği yazılım geliştirme yaşam döngüsü fazı	AS2
Tahmin Yöntemleri	Çeşitli Makine Öğrenmesi Yöntemleri	Hata tahmin modelinde kullanılan makine öğrenmesi yöntemleri	AS3.1
Metrikler	Gereksinim, Tasarım, Kod Metrikleri	Hata tahmin modelinde girdi olarak kullanılan metrikler	AS3.2
Performans Değerlendirme	ROC, AUC, Probability of Detection, Probability of False Alarms, Precision, Accuracy, MAE, RMSE	Hata tahmin modelinin performansını ölçmek için kullanılan performans değerlendirme ölçümleri	AS3.3
Modelin Uygulandığı Araç/Ortam	Visual Basic, Weka, MATLAB	Hata tahmin modelinin uygulandığı araç/ortam	AS3.4
Veri Seti	NASA MDP (Metric data program), PROMISE	Hata tahmin modelinin uygulandığı veri setleri	AS3.5

4 Bulgular

AS1: Erken hata tahmini hakkında yapılan çalışmaların eğilimi ne yöndedir?
AS1.1: Yayınların yıllara göre dağılımı nasıldır?

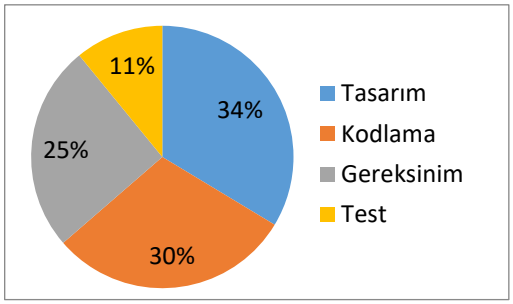
Çalışmaların yıllara göre dağılımının düzensiz olduğu söylenebilir. 2005 yılından itibaren yazılım hata tahmin çalışmalarına olan ilginin arttığı söylenmektedir [7] [10], fakat erken aşamalara ilişkin hata tahmini yapan çalışmaların sayısı farklılık göstermektedir (Şekil 1).



Şekil 1. Erken hata tahmini çalışmalarının yıllara göre dağılımı

AS2: Erken hata tahmin modelleri hangi yazılım geliştirme yaşam döngüsü (YGYD) aşamaları ile ilişkilidir?

İncelenen çalışmalarda, verilerin toplandığı projelerde tercih edilen YGYD metodolojisi net olarak belirtilmemiştir. YGYD fazları olarak; gereksinim analizi, tasarım, kodlama, test ve sürüme açma aşamalarından bahsedilmektedir. Burdan yola çıkarak, kullanılan YGYD metodolojisinin şelale yöntemi olduğu varsayımı yapılmıştır. Şekil 2'de, erken hata tahmin modellerinde en çok tasarım aşamasına ilişkin verilerin (%34) kullanıldığı görülmektedir. Gereksinim aşamasına ilişkin veriler ise %25 oranında kullanılmıştır. Çalışmaya dahil edilen modellerin bir kısmında, gereksinim ve/veya tasarım aşamasına ek olarak kodlama ve/veya test aşamasına ilişkin verilerin de kullanıldığı görülmüştür. Erken aşamaları ele alan bu modellerden, kodlama aşamasına ilişkin verileri de modele dahil edenlerin oranının %30 olduğu söylenebilir. Test aşamasına ilişkin verileri de ele alan çalışmalar (%11) düşük oranda da olsa mevcuttur.



Şekil 2. İlişkili YGYD Aşaması Dağılımı

AS3: Gereksinim ve tasarım aşamalarından elde edilen verileri, kod aşamasında elde edilen verilerle birlikte kullanan modellerin tasarımı nasıldır?

İlgili araştırma sorusunu ve alt sorularını yanıtlamak için toplanan veriler Tablo 4'de verilmiştir.

Tablo 4. Gereksinim/tasarım aşamalarından elde edilen verileri, kod aşamasından elde edilen verilerle birlikte kullanan modellerin tasarımı

Ref No	YGYD Fazı	Hata Tahmin Yöntemi	Tahmin Aracı	Veri Seti	Performans Ölçütü	Kullanılan Metrikler
[17]	Gereksinim, Kod	OneR, NaiveBayes with kernel, VotedPerceptron, Logistic, J48, VFI, Ibk ve Random Forest	Weka	CM1, JM1 ve PC1	ROC	Gereksinim: action, conditional, continuance, imperative, incomplete, option, risk level, source, weak phrase Kod: büyüklük ve karmaşıklık metrikleri
[18]	Tasarım, Kod	Random Forest, Bagging, Logistic regression, Boosting ve Naivebayes	Weka	CM1, KC1, KC3, KC4, PC1, PC3, PC4, MW1, MC2, JM1, MC1, PC2, PC5	ROC, AUC, BOX plot, Statistical	Kod: büyüklük ve karmaşıklık metrikleri Tasarım: complexity, edge, node, brach count
[19]	Gereksinim, Kod	K-Means clustering	MATLAB 7.4.	CM1, JM1 ve PC1	ROC, Probability of Detection (PD), Probability of False Alarms (PF)	Gereksinim: action, conditional, continuance, imperative, incomplete, option, risk level, source, weak phrase Kod: büyüklük ve karmaşıklık metrikleri
[20]	Gereksinim, Kod	Genetic Algorithm	Visual Basic 6.0	PC1	Accuracy, MAE, RMSE	Gereksinim: module, action, conditional, continuance, imperative, option, risk level, source, weak phrase Kod: büyüklük ve karmaşıklık metrikleri
[21]	Gereksinim, Kod	Decision Tree C4.5 ve K-means clustering	Rapidminer	CM1	ROC	Gereksinim: action, conditional, continuance, imperative, incomplete, option, risk level, source, weak phrase Kod: büyüklük ve karmaşıklık metrikleri
[22]	Gereksinim, Kod	Density Based Clustering	MATLAB 7.4	PC1	Accuracy, Probability of detection, Probability of false alarm, Precision	Gereksinim: action, conditional, continuance, imperative, incomplete, option, risk level, source, weak phrase Kod: büyüklük ve karmaşıklık metrikleri

AS4: Gereksinim ve tasarım aşamalarından elde edilen veriler kod aşamasından elde edilen verilerle birlikte kullanıldığında tahmin modelinin performansına etkisi olmuş mudur? Performansa etkisi varsa ilgili çalışmada raporlanmış mıdır?

Gereksinim ve tasarım aşamalarından elde edilen veriler, kodlama verisi ile oluşturulan modellere eklendiğinde, modellerin performansında artış görüldüğü raporlanmıştır. Performanstaki artışlar seçilen 6 adet çalışmada hem sayısal (ROC eğrisine ait grafiklerin karşılaştırılması, doğruluk değerleri gibi), hem de sözel olarak raporlanmıştır. Detaylı sonuçlar Tablo 5'de verilmiştir.

Tablo 5. Çalışmalarda raporlanan sonuçlar

Ref No	Sonuçlar
[17]	Model karşılaştırma sonuçları, gereksinimi ölçümlerinin hata tahmini için son derece yararlı olabileceğini göstermektedir. Gereksinim + kod metrik grubuyla ve Random Forest tahminleme yöntemiyle oluşturulan modelin en iyi ROC eğrisine sahip olduğu grafiksel sonuçlarla raporlanmıştır.
[18]	Çalışmada tasarım, kod, tasarım + kod metriklerinden oluşturulan modellerin en başarılısı tasarım + kod metriklerinden oluşturulan model olarak belirtilmiştir. Modellerin performans karşılaştırılması iki farklı istatistiksel teste göre yapılmıştır. Friedman testinin sonuçlarına göre, model performansları arasında istatistiksel olarak anlamlı farkın olduğu (p−value < 0,05) ispatlanmıştır. Wilcoxon testinin sonuçlarına göre ise, 7 veri setinde tüm metrik grubundan (tasarım + kod metrikleri) yapılandırılan modellerin performansa önemli ölçüde bir katkılarının olmadığı gözlenmiştir. Fakat geri kalan 6 veri setinde, tasarım ve kod metrikleri ile yapılandırılan modellerin performansının önemli ölçüde arttığı (p−value < 0,05) ispatlanmıştır.
[19]	Bu çalışmada, test eforunu azaltmak ve daha kaliteli projeleri oluşturmak için gereksinim aşamasında mevcut verilerin de kullanıldığı hata tahmin modellerinin yararlı olacağı vurgulanmıştır. Füzyon modelin tahmin performansının CM1 veriseti için 0,99729 (probability of detection) ve 0,79518 (probability of false alarm) olduğu, PC1 veriseti için ise PD değeri 1, PF değeri 0,99724 olarak raporlanmıştır.
[20]	Kodlama aşamasından toplanan verilerle oluşturulan hata tahmin modeline, gereksinim aşamasından toplanan veriler de eklendiği zaman, hata tahmin modelinin performansının yaklaşık %4 oranında arttığı raporlanmıştır.
[21]	Tahmin modellerinin kod aşamasına ek olarak erken aşamalara ilişkin verileri de kullanacak şekilde yapılandırımlasının yararlı olabileceği belirtilmiştir. Kod ve gereksinim metriklerinin bir arada kullanıldığı modelde, çapraz geçerleme sonuçlarına göre hata tahmin performansı %100 kesinlik (Precision) ve %100 duyarlılık (Recall) oranında raporlanmıştır.
[22]	Füzyon (gereksinim ve kod aşamasına ilişkin verilerin birlikte kullanıldığı) modellerin daha yüksek performanslı sonuçlar verdiği ve projenin erken aşamalarında kullanılabilir olduğu söylenmiştir. Kod bazlı modelin doğruluk (accuracy) oranı 0,844 iken, füzyon modelin doğruluk oranının 0,928 olduğu raporlanmıştır.

İncelenen çalışmalarda, hata tahmin modelinin performansındaki artışların nedeninin erken aşama bilgisi olduğu söylenmektedir. Performansın artışına sebep olan etkenin kullanılan metrik sayısındaki artış olup olmadığı çalışmalar tarafından tartışılmamıştır.

5 Sonuç ve Öneriler

Bu çalışmada, yazılım geliştirme yaşam döngüsünün gereksinim ve tasarım gibi erken aşamalarına ilişkin bilgileri hata tahmin modeline dahil eden akademik çalışmalar sistematik yöntemlerle taranmıştır. Erken aşama bilgisini, kodlama aşaması bilgileri ile kullanarak tahmin modeli yapılandırmış çalışmalardaki performans değişimleri incelenerek literatüre özgün bir katkı sağlanmıştır.

Detaylı olarak incelenen çalışmalarda, gereksinim ve tasarım aşamalarından toplanan ölçümlerin, kod aşamasından toplanan ölçümlerle birlikte kullanıldığında, hata tahmin modellerinin performansını arttırdığına ilişkin sonuçların raporlandığı tespit edilmiştir. Bu çalışmalarda, performansın artışına sebep olan etkenin metrik sayısındaki artış olup olmadığı incelenmemiştir. Gelecekte, erken aşamalardan ve kod aşamasından toplanan verilerle hata tahmin modelleri oluşturmayı, tahmin performanslarını karşılaştırmayı ve performansta artış görülmesi durumunda bunun nedenini araştırmayı planlamaktayız.

Erken aşamalarda kullanılan verilerin performans artışına katkısı olduğu kabul edildiğinde, sadece erken aşamalardan toplanan veriler ile hata tahmininin yapılabilir olduğunu görmek önemli olacaktır. Sistematik eşleme kısmına dahil ettiğimiz çalışmalarda, sadece erken aşama bilgisini kullanan modellerin detaylıca incelenmesi gerektiğini düşünmekteyiz.

İncelenen çalışmalarda, erken aşamalarda yazılım hata tahmini yapan modellerin, çok çeşitli yöntemler ve parametreler ile yapılandırıldığı görülmüştür. Yazılım projesinin parametreleri (proje ekibi, geliştirme süreçleri, takvim, bütçe, yöneticilerin/proje personelinin ihtiyaçları vb.) göz önünde bulundurularak, bağlama en uygun yöntemi seçebilmek önem taşımaktadır. Bu doğrultuda, yazılım parametrelerine göre kullanılması gereken metriklerin, tahmin yöntemlerinin ve model performansını değerlendirme kriterlerinin belirtildiği bir kılavuz, yazılım hata tahmini yapacak kişiler için yönlendirici olacak ve fayda sağlayacaktır. Sonraki çalışmalarımızda, erken aşamaları adresleyebileceğini ve hata tahmin performansını arttıracağını belirlediğimiz yazılım metriklerini ve tahmin yöntemlerini kullanarak, çoklu durum çalışmaları yapmayı ve sonuçlarına göre erken aşamalarda hata tahminine ilişkin yönlendirici bir kılavuz hazırlamayı planlamaktayız.

Kaynaklar

1. C. Smidts, R. W. Stoddard and M. Stutzke. Software reliability models: an approach to early reliability prediction. Software Reliability Engineering, 1996. Proceedings., 7th International Symposium on, NY, 1996, pp. 132-141.
2. Q. Song, Z. Jia, M. Shepperd, S. Ying and J. Liu. A General Software Defect-Proneness Prediction Framework. In: IEEE Transactions on Software Engineering, 37(3), pp. 356-370, (2011)
3. Pandey, A.K., Goyal, N.K.: Early Software Reliability Prediction. Springer, New Delhi (2013)
4. Norman Fenton, Martin Neil, William Marsh, Peter Hearty, Łukasz Radliński, and Paul Krause. 2008. On the effectiveness of early life cycle defect prediction with Bayesian Nets. Empirical Softw. Engg. 13, 5 (October 2008), 499-537.
5. IEEE Standard Classification for Software Anomalies, in IEEE Std 1044-2009 (Revision of IEEE Std 1044-1993), vol., no., pp.1-23, Jan 7 2010
6. IEEE Recommended Practice on Software Reliability, in IEEE STD 1633-2008 , vol., no., pp.c1-72, June 27 2008
7. Çatal, C., & Diri, B. (2009). A Systematic Review Of Software Fault Prediction Studies, Expert Systems with Applications, 36, 7346–7354.

8. Çatal, Ç. (2011). Software Fault Prediction: A Literature Review and Current Trends, Expert Systems with Applications, 38, 4626-4636.
9. Hall, T., Beecham, S., Bowes, D., Gray, D., Counsell, S., (2012). A systematic literature review on fault prediction performance in software engineering. IEEE Trans. Softw. Eng. 38(6), 1276–1304
10. Wahono, R.S. (2015). A systematic literature review of software defect prediction: research trends, datasets, methods and frameworks. J. Softw. Eng. 1, 1–16
11. Özakıncı, R. and Tarhan, A. (2016). The Role of Process in Early Software Defect Prediction: Methods, Attributes and Metrics. In: 16th International Conference on Software Process Improvement and Capability Determination, SPICE 2016, Volume 609, pp. 287-300.
12. Tarhan, A., & Demirors, O. (2011). Investigating the effect of variations in the test development process: A case from a safety-critical system. Software Quality Journal, 19(4), 615–642.
13. Aslan, D., Tarhan, A., & Demirors, O. (2014). How process enactment data affects product defectiveness prediction—a case study. In R. Lee (Ed.), Software engineering research, management and applications (pp. 151–166). Heidelberg: Springer International Publishing.
14. Söylemez, M. & Tarhan, A. (2016). Challenges of Software Process and Product Quality Improvement: Catalyzing Defect Root-Cause Investigation by Process Enactment Data Analysis. Software Quality Journal.
15. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic mapping studies in software engineering. In: 12th International Conference on Evaluation and Assessment in Software Engineering. Volume 17. (2008)
16. B. Kitchenham and S. Charters, "Guidelines for Performing Systematic Literature Reviews in Software Engineering (Version 2.3)," Technical Report EBSE-2007-01, Keele Univ., EBSE, 2007.
17. Jiang, Y., Cukic, B., & Menzies, T. (2007). Fault prediction using early lifecycle data. In Eighteenth IEEE international symposium on software reliability (pp. 237–246). Trollhättan, Sweden: IEEE Computer Society.
18. Jiang, Y., Cukic, B., Menzies, T., & Bartlow, N. (2008). Comparing design and code metrics for software quality prediction. 4th International Workshop on Predictor Models in Software Engineering PROMISE 08, 12, 11–18.
19. Kaur, A., Sandhu, P. S., & Bra, A. S. (2009). Early Software Fault Prediction Using Real Time Defect Data. 2009 Second International Conference on Machine Vision, 242–245.
20. P.Sandhu, S.Khullar, S.Singh, S.Bains, M.Kaur and G.Singh, A Study on Early Prediction of Fault Proneness in Software Modules using Genetic Algorithm, 2010, World Academy of Science, Engineering and Technology.
21. Sandhu, P. S., Goel, R., Brar, A. S., Kaur, J., & Anand, S. (2010). A model for early prediction of faults in software systems. 2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE), 4, 281–285.
22. Sandhu, P. S., Kaur, M., & Kaur, a. (2010). A density based clustering approach for early detection of fault prone modules. International Conference on Electronics and Information Engineering, Proceedings, 2(Iceie), V2525–V2530.