T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

AÇIK KAYNAK DAĞITIK KOD GELİŞTİRME ORTAMLARI İÇİN METRİK ÖNERİLERİ

Abdulkadir ŞEKER

DOKTORA TEZİ
Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Programı

Danışman
Prof. Dr. Banu DİRİ

Nisan, 2021

T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

AÇIK KAYNAK DAĞITIK KOD GELİŞTİRME ORTAMLARI İÇİN METRİK ÖNERİLERİ

Abdulkadir ŞEKER tarafından hazırlanan tez çalışması 08.04.2021 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği Programı **DOKTORA TEZİ** olarak kabul edilmiştir.

Prof. Dr. Banu DİRİ
Yıldız Teknik Üniversitesi
Danışman

Jüri Üyeleri

Prof. Dr. Banu DİRİ, Danışman
Yıldız Teknik Üniversitesi _____

Doç. Dr. Mehmet Fatih AMASYALI, Üye
Yıldız Teknik Üniversitesi _____

Dr. Öğr. Üyesi Halil ARSLAN, Üye
Sivas Cumhuriyet Üniversitesi _____

Prof. Dr. Oya KALIPSIZ, Üye
Yıldız Teknik Üniversitesi _____

Prof. Dr. Takuhi Nadia ERDOĞAN, Üye
İstanbul Teknik Üniversitesi _____

Danışmanım Prof. Dr. Banu DİRİ sorumluluğunda tarafımca hazırlanan Açık Kaynak Dağıtık Kod Geliştirme Ortamları için Metrik Önerileri başlıklı çalışmada veri toplama ve veri kullanımında gerekli yasal izinleri aldığımı, diğer kaynaklardan aldığım bilgileri ana metin ve referanslarda eksiksiz gösterdiğimi, araştırma verilerine ve sonuçlarına ilişkin çarpıtma ve/veya sahtecilik yapmadığımı, çalışmam süresince bilimsel araştırma ve etik ilkelerine uygun davrandığımı beyan ederim. Beyanımın aksinin ispatı halinde her türlü yasal sonucu kabul ederim.

Abdulkadir ŞEKER

İmza

Kızım Zeynep Deniz ŞEKER'e ithafen.

TEŞEKKÜR

Akademik kariyerimde yorulduğum ve vazgeçtiğim her an beni tekrar motive eden hayat arkadaşım, eşim Fatma Betül ŞEKER'e, tezim boyunca emeklerini esirgemeyen danışmanım Prof. Dr. Banu DİRİ'ye ve değerli hocam Dr. Öğr. Üyesi Halil ARSLAN'a çok teşekkür ederim.

Abdulkadir ŞEKER

İÇİNDEKİLER

KISALTMA LİSTESİ

API	Uygulama Programlama Arayüzü (Application Programming Interface)
CBS	İçeriğe Bağlı Benzerlik (Context Based Similarity)
CFS	İçerikten Bağımsız Benzerlik (Context Free Similarity)
CI	Sürekli Entegrasyon (Continuous Integration)
CSV	Virgüle Ayrılmış Değerler (Comma Separated Values)
DDY	Dil Deneyimi Yaklaşımı
DVCS	Dağıtık Versiyon Kontrol Sistemi (Distributed Version Control System)
IDF	Ters Belge Sıklığı (Inverse Document Frequency)
OSS	Açık Kaynak Yazılım (Open Source Software)
PGOM	Proje-Geliştirici Oy Matrisi (Project-Developer Rating Matrix)
PR	Birleştirme İsteği (Pull Request)
SLR	Sistematik Literatür İncelemesi (Systematic Literature Review)
SSH	Güvenli Kabuk (Secure Shell)
TF	Terim Sıklığı (Term Frequency)
TİY	Topluluk İlişkisi Yaklaşımı
VCS	Versiyon Kontrol Sistemi (Version Control System)

ŞEKİL LİSTESİ

TABLO LİSTESİ

ÖZET

Açık Kaynak Dağıtık Kod Geliştirme Ortamları için Metrik Önerileri

Abdulkadir ŞEKER

Bilgisayar Mühendisliği Anabilim Dalı
Doktora Tezi

Danışman: Prof. Dr. Banu DİRİ

GitHub, milyonlarca geliştiricinin iş birliği yaptığı bir ortam ve 100 milyondan fazla proje için kod barındırma ve depo hizmeti sunmaktadır. Bu ortamdaki kod geliştirme süreçlerinde geliştiricilerin yürüttüğü aktivitelerden ve ortamdaki ilişkilerinden, geliştiriciler hakkında farklı bilgilerin elde edilmesi sağlanmaktadır. Çok çeşitli özellikleri sayesinde araştırmacılar, geniş yelpazedeki yazılım geliştirme zorluklarını çözmek için GitHub platformundan yararlanmaktadır.

Bu tez kapsamında, farklı yazılım mühendisliği zorlukları için kullanılabilecek, erişimi kolay, uygulama açısından küçük boyutlu ve veri bakımından kapsamlı olan ve GitDataSCP adı verilen açık kaynaklı bir veri kümesi tarafımızdan üretilmiştir.

GitHub gibi milyonlarca projenin bulunduğu platformlarda geliştiricilerin ilgi duyacakları veya katkı sağlayacakları projeleri keşfedebilmeleri oldukça zordur. Özellikle açık kaynaklı büyük projeler ele alındığında, projeler gönüllü geliştiriciler tarafından yürütülmekte ve genişletilmektedir. Geliştiricilerin doğru projeleri bulabilmesi hem geliştiriciye kendi projelerinde takıldığı sorunlarda yardımcı olacak, hem de keşfettiği projelere katkı vermesini sağlayacaktır. Bu noktada, bazı dezavantajlarından dolayı literatürde çok fazla odaklanılmamış olan, açık kaynak kod versiyonlama platformları için proje önerisi problemi ortaya çıkmaktadır.

Geliştirici metriklerinin üretilmesinde, genellikle geliştiricinin geçmiş aktiviteleri kullanılmaktadır. Bu tez kapsamında, GitHub ortamındaki geliştiricilerin *issue*, *commit*, *pull request* süreçlerinde yürüttüğü aktivitelerden geliştirici metrikleri üretmek

amaçlanmıştır. Bu aktivitelerden aday geliştirici metriklerini tespit edebilmek için aktivitelerin GitHub ortamındaki kullanım oranları incelenmiştir. Ayrıca aktivitelerin önemini tespit edebilmek için 130 geliştirici ile bir anket çalışması da yapılmıştır. Bu aşamalardan sonra, çeşitli yaklaşımlar kullanılarak 40 farklı geliştirici metriği önerilmiştir.

Önerilen söz konusu metriklerin proje-geliştirici ilişkisi açısından katkısının değerlendirilmesi için, bir GitHub proje öneri sistemi geliştirilmiştir. GitHub proje sistemini sıradan öneri sistemlerinden daha zorlu kılan taraf, değerlendirme için temel bir gerçeğin olmamasıdır. Bu tez kapsamında, öneri sisteminin doğruluğunun tespit edilebilmesi için topluluk ilişkisi ve dil deneyimi olarak isimlendirdiğimiz 2 farklı değerlendirme yöntemi önerilmiştir.

Öneri sisteminin sonuçlarına göre farklı yaklaşımlar ile elde edilmiş metriklerin başarıları ortaya konulmuştur. Farklı süreçlerde yapılan yorumlar ile ilişkili metriklerin sonuçları, bir projede kod yazmanın yanı sıra yorum yapmanın da proje-geliştirici bağını güçlü bir şekilde temsil ettiğini göstermektedir. Ayrıca, *issue* ve *pull request* süreçlerinde yapılan aktivitelerden elde edilen metriklerin başarısı da bu iki özelliğin önemini göstermektedir.

Önerilen metriklerden, belirli türdeki aktivitelerin birleştirilmesinden oluşan, füzyon metrikler adı verilen geliştirici metriklerinin, diğerlerine oranla daha üstün geldiği görülmüştür. Ayrıca, bir aktivitenin miktarını göz ardı eden, sadece bir projede var olup olmadığını baz alan ikili füzyon metrikleri de şaşırtıcı derecede başarılı sonuçlar vermiştir. Bu tezde önerilen geliştirici metriklerinin iyileştirilebilir ve farklı yazılım mühendisliği problemlerine de uygulanabilir olduğu düşünülmektedir.

Anahtar Kelimeler: Yazılım mühendisliği, açık kaynak yazılım, geliştirici metriği, dağıtık geliştirme, öneri sistemi, GitHub.

ABSTRACT

Proposing Metrics for Open Source Distributed Code Development Platforms

Abdulkadir ŞEKER

Department of Computer Engineering
Doctor of Philosophy Thesis

Advisor: Prof. Dr. Banu DİRİ

GitHub offers a collaborative environment of millions of developers and code hosting and repository for over 100 million projects. In the code development processes in this environment, it is provided that various information about the developers is obtained from the activities carried out by the developers and their relationships in the platforms. Thanks to its wide variety of features, researchers benefit from the GitHub platform to solve a wide range of software development challenges.

Within the scope of this thesis, an open source dataset called GitDataSCP, which can be used for different software engineering challenges, is easy to access, small in size in terms of implementations, and comprehensive in terms of data.

On platforms like GitHub with millions of projects, it is very difficult for developers to discover projects that they will be interested in or contribute to. Projects are run and expanded by volunteer developers, especially when considering large open source projects. Being able to find the correct projects by the developers will not only help the developer with the problems they have in their projects but also contribute to the projects they discover. At this point, the problem of project recommendation arises for open source code versioning platforms, which is not so much focused in the literature due to some disadvantages.

Past activities of the developers are often used to generate developer metrics. Within the scope of this thesis, it is aimed to produce developer metrics from the activities carried out by the developers in the GitHub environment from the *issue*, *commit*,

pull request processes. In order to determine candidate developer metrics from these activities, the usage rates of the activities in the GitHub environment were examined. In addition, a survey was conducted with 130 developers to determine the importance of the activities. After these stages, 40 different developer metrics were proposed using various approaches.

In order to evaluate the contribution of the proposed metrics in terms of project-developer relationship, GitHub project recommender system was developed. The parameter that makes the GitHub project recommender system more challenging than ordinary recommender systems is the lack of a fundamental truth for evaluation. Within the scope of this thesis, two different evaluation methods, which we named as community relationship and language experience, are proposed in order to determine the accuracy of the recommender system.

According to the results of the recommender system, the successes of the metrics obtained with different approaches have been revealed. The results of the metrics associated with comments made in different processes show that on projects, as well as coding, also commenting strongly represents the project-developer connection. In addition, the success of the metrics obtained from the activities performed in the *issue* and *pull request* processes shows the importance of these two features.

From the proposed metrics, combining certain types of activities, developer metrics called fusion metrics have proved to surpass others. Furthermore, binary fusion metrics that ignore the amount of activity, based only on whether it exists in a project, have also yielded surprisingly successful results. The developer metrics proposed in this thesis are thought to be improvable and applicable to different software engineering problems.

Keywords: Software engineering, open source software, developer metric, distributed development, recommender system, GitHub.

1
GİRİŞ

1.1 Literatür Özeti

Yazılım mühendisliği alanında açık kaynak dağıtık kod geliştirme, önemli bir başlık olarak yer almaktadır. Dağıtık kod geliştirme süreçlerinin hızlı ve sorunsuz yürütülebilmesi için dağıtık versiyon kontrol sistemlerine ihtiyaç duyulmaktadır. Bu sistemlerden en yaygın kullanılanı GitHub platformudur. GitHub, milyonlarca geliştiricinin iş birliği yaptığı bir ortam ve 100 milyondan fazla proje için kod barındırma ve depo hizmeti sunmaktadır. Bu ortamdaki kod geliştirme süreçlerinde geliştiricilerin yürüttüğü aktivitelerden ve ortamdaki ilişkilerinden, geliştiriciler hakkında farklı bilgilerin elde edilmesi sağlanmaktadır. Çok çeşitli özellikleri sayesinde araştırmacılar, geniş yelpazedeki yazılım geliştirme zorluklarını çözmek için GitHub platformundan yararlanmaktadır. Bu problemlerin çözümünde ise geliştirici metriği olarak isimlendirilen söz konusu bilgiler kullanılmaktadır. Bu problemlerden önemli bir tanesi de GitHub gibi ortamlardaki kullanıcılar için proje önerisi problemidir.

Bu tez kapsamında, öncelikle, GitHub verilerini kullanan çalışmaları inceleyen bir sistematik literatür taraması yapılmıştır. Bu literatür taraması, dijital kitaplıklarda anahtar kelime tabanlı bir arama yerine, bir GitHub veri kümesi kaynak çalışmasına dayalı olarak gerçekleştirilmiştir. GHTorrent, literatüre göre en çok bilinen GitHub veri kümesi olduğundan, bu veri kümesine atıfta bulunan tüm çalışmalar taramada kullanılmıştır. Veri kümesine atıf yapan 332 çalışmadan belirli kriterlere göre seçilen 172'si ayrıntılı olarak incelenmiştir. Çalışmaların üst verilerinden çıkarılan bilgiler sayesinde, çalışmalar açık kaynak dağıtık yazılım geliştirme zorlukları kapsamında sınıflandırılmıştır. Ayrıca veri kümesi ile ilgili bazı sorunlar ortaya konulmuştur. Literatürde odaklanılmış alanların yanı sıra, araştırmacıları üzerinde çalışmaya teşvik ettiğimiz açık zorluklar sunulmuştur.

GitHub ile ilgili bir başka husus ise ortamındaki büyük veriyi işlemek ve problemlere uygulamak, donanım ve zaman açısından oldukça maliyetli olmaktadır. Bu yüzden,

GitHub verisini kullanan çalışmaların çoğu, bu büyük boyutlu veriyi çeşitli filtreler ile kısıtlayarak, çalışmalara özgü ve paylaşılmayan alt veri kümeleri ile problemlere odaklanmaktadır. Bu durum, benzer problemler üzerinde yapılan çalışmaların birbiri ile adil olarak kıyaslanmasını zorlaştırmaktadır.

Tez boyunca kullanılan bazı terimler şöyle verilebilir;

- Branch: Ana depodan bağımsız çalışan kopya depo
- Commit: Bir depodaki bireysel değişiklik
- Collaborator: Bir depodaki işbirlikçiler
- Collaborator: Bir depodaki işbirlikçiler
- Fork: Bir deponun bağımsız bir depoya kopyalanması
- Issue: Bir depodaki her türlü sorun veya hata
- Hit: Doğru proje önerisi
- Master branch: Projenin başlatıldığı ana dal
- Merge: Yapılan değişikliği ana depoya birleştirme
- Public: Kamuya açık olma
- Pull request: Yapılan değişikliği ana depoya iletme
- Open source: Derlenmemiş okunabilir kod.
- Rebase: Bir deponun geçmişini düzenleme
- Repository: Projeleri barındıran dizin veya (depo).

1.2 Tezin amacı

Açık kaynak dağıtık yazılım geliştirme sayesinde çok sayıda açık kaynaklı proje çok sayıda farklı yazılımcının katkıları ile geliştirilmektedir. Projelerin tüm fazlarında geliştiriciler, bazı protokoller ve platformlar aracılığıyla eş zamanlı, yönetilebilir, ve kontrol edilebilir bir şekilde çalışabilmektedir. Dağıtık geliştirme süreçlerinin doğru yürütülebilmesindeki en önemli rollerden birisi de versiyon kontrol sistemlerine düşmektedir. Bu sistemlerden en yaygın kullanılanlarından biri de *git* protokolü tabanına sahip GitHub platformudur.

GitHub ortamı milyonlarca projeyi depolarında barındıran ve aynı zamanda milyonlarca geliştiriciye eş zamanlı kod geliştirme kolaylığı sunan bir ortamdır. GitHub gibi ortamlarda yazılım geliştirme süreçleri dağıtık yürütülmektedir. Bu durum, çözülmesi gereken çok farklı yazılım mühendisliği problemlerini de beraberinde getirmektedir. Bu tez kapsamında ele alınan ilk problem, açık kaynak dağıtık yazılım geliştirme süreçlerine odaklanan çalışmalarda kullanmak üzere hali hazırda, kolay erişilebilir, açık kaynaklı ve ortak kullanabilecek bir veri kümesinin olmayışıdır. Bu sorunu çözmek için, GitHub ortamına nispeten çok daha küçük (kolay kullanılabilir), aynı zamanda ortamın doğasına benzer bir veri kümesi üretmek amaçlanmaktadır.

GitHub ortamına ait veriler ile belirli süreçlere görevli veya inceleyici atama, bazı özellik ve aktiviteleri sınıflandırma, projeleri ve geliştiricileri karakterize etme, kullanıcılara proje önerme gibi farklı problemlere çözümler sunulmaktadır. Bu problemleri çözerken genellikle geliştiricilerin projelerde ürettikleri kodlar ve diğer aktivitelerden elde edilen bilgiler kullanılmaktadır. Kullanılan bu bilgilere geliştirici metriği adı verilmektedir.

Tez kapsamındaki bir diğer amaç ise, literatürde kullanılan metriklerin yanına yeni, daha kapsayıcı ve hızlı üretilebilir geliştirici metrikler ekleyebilmektir. Bu metriklerin hangi aktivitelerden elde edilmesi gerektiğine karar vermek de önemli bir problemdir. Bu bağlamda, GitHub ortamında kullanılan özeliklerin ve aktivitelerin gerçek yazılımcıların gözündeki önemini ölçmek için bir anket çalışması yapmak amaçlanmaktadır.

En çok projeye sahip olan bir dağıtık kaynak kod versiyonlama platformu olarak GitHub ortamındaki bir diğer problem de kullanıcıların kendi birikim ve yetenekleri doğrultusunda katkı sağlayabilecekleri projeleri bulmalarıdır. Aynı şekilde, geliştiricilerin kendi projelerini zenginleştirmek veya sürdürebilmek için faydalanabilecekleri, kendi projelerine benzer projelerden de haberdar olması da ortam büyüklüğü sebebiyle bir hayli zorlaşmaktadır. Tez kapsamında önerilen geliştirici metrikleri kullanan ve GitHub platformu için geliştiricilere proje önerisi sunacak bir sistem tasarlanması da amaçlanmaktadır.

1.3 Hipotezler

Tez bünyesinde ortaya atılan hipotezler şu şekilde verilebilir.

- Açık kaynak dağıtık yazılım geliştirme problemleri üzerinde çalışan araştırmacılar, ortak veri kümeleri kullanmadığından dolayı yapılan çalışmaları birbiri ile kıyaslamak oldukça zor olmaktadır. Bu bağlamda, farklı yazılım mühendisliği problemleri için kullanılabilecek bir veri kümesi üretilebilir.
- Yazılım mühendisliği problemlerinin çözümünde kullanılan geliştirici metriklerine yeni, daha az bilgi gerektiren, üretilmesi kolay metrikler eklenmesi literatüre önemli bir katkı sağlayabilir.
- Geliştirici metrikleri üretilirken kullanılan aktiviteler, bulundukları süreçlere ve bazı benzer özelliklerine göre gruplandırılarak yeni metrikler oluşturulabilir.
- Geliştirici metrikleri üretilirken kullanılan aktivitelerin, nicel değerleri önemli olmayabilir.
- Geliştirici metrikleri önerisinde, bir aktivitenin bir geliştirici tarafından kaç kez yapıldığını kullanmak yerine, yapılan aktivitenin bir projedeki toplam sayısına oranla ne kadar değerli olduğunu keşfetmek, metrik üretimi açısından önemli olabilir.
- GitHub ortamındaki aktiviteler arasında geliştiricilerin yazılım geliştirmenin herhangi bir aşamasında yaptığı yorumlar, en az kod geliştirme kadar değerli olabilir.
- Önerilen metrikler, literatürde önemli bir problem olarak çalışılan proje öneri sistemi için kullanılabilir.
- Klasik öneri sistemlerinden farklı mantıkla işleyen GitHub proje öneri modelinde, modelin doğruluğunu ölçecek farklı değerlendirme yöntemleri geliştirilebilir.

2
KAYNAK TARAMASI

GitHub, Açık Kaynak Yazılım (OSS) projeleri için en yaygın kod barındırma hizmetidir. Çok çeşitli özellikleri sayesinde araştırmacılar, farklı OSS geliştirme zorluklarını çözmek için GitHub'dan yararlanırlar. Bu bağlamda, literatür çalışmasını GitHub verisi kullanan tüm çalışmalar ile yapmanın önemli olduğu düşünülmüştür. Bu çalışmalara ulaşmak için, dijital kitaplıklarda anahtar kelime tabanlı arama yapmak yerine, içinde bir GitHub veri kümesini referans gösteren çalışmaların bulunması hedeflenmiştir. Literatüre göre en çok bilinen ve kullanılan GitHub veri kümesi olan GHTorrent'e, veri kümesi olarak atıf yapan tüm çalışmalar ile bir Sistematik Literatür Taraması (SLR) yapılmıştır. Bu bölümde, bazı kriterlere göre seçilen 172 çalışma incelenmiştir. Çalışmalar, üst verisinden (metadata) çıkarılan bilgiler sayesinde, OSS geliştirme zorlukları kapsamında sınıflandırılmıştır.

2.1 Araştırma Yöntemi

Literatürdeki diğer sistematik literatür taraması makalelerinde, arama motorlarından veya dijital kitaplıklardan metin tabanlı (anahtar kelime) bir tarama yapılırken bazı çalışmaların gözden kaçabileceği görülmüştür [1, 2]. Bu nedenle, literatür taraması anahtar kelime tabanlı arama yerine, literatürde en çok kullanılan veri kümesini veri kaynağı olarak gösteren çalışmalara dayalı olarak gerçekleştirilmiştir.

Git vb. Dağıtık Versiyon Kontrol Sistemleri (DVCS) sayesinde açık kaynak kod geliştirme platformları önemli sayıda kullanıcıya ulaşmıştır. Bunlardan en yaygın olanlarından biri GitHub platformudur. Yazılım mühendisliğine odaklanan çalışmaların çoğu, kolay erişim, veri boyutu ve özellik çeşitliliği nedeniyle GitHub verisini, bir veri kaynağı olarak kullanmaktadır.

GitHub veri kümelerinin kullanım oranlarının verildiği bir anket çalışmasında, belirli kriterlere göre incelenen makalelerde en çok kullanılan veri kümesinin GHTorrent (%34) olduğu belirtilmiştir [3]. Cosentino'nun sistematik haritalama çalışmasında, GHTorrent veri kümesi %41 kullanım oranıyla lider konumdadır [4, 5]. Başka bir

çalışmada ise, GHTorrent'in en çok atıf yapılan veri kümesi olduğu vurgulanmıştır [6].

Bu bilgiler ışığında, tez kapsamında, bu veri kümesine atıf yapan çalışmalar ile bir SLR yürütülmüştür. Çalışmalar bazı kategorilere ve zorluklara göre sınıflandırılmıştır. Ayrıca, veri kümesini kullanan çalışmalardan bazı dağılımlar (tür, yer, yıl, yöntem, veri tipi, veri boyutu ve konu) elde edilmiştir. Çalışmalarda hangi zorluklara odaklanıldığı ve her bir çalışmanın veri kümesini hangi filtreler ile kullandığı da gösterilmiştir.

Bu tez kapsamında, Kitchenham [7] tarafından önerilen 3 aşamalı SLR protokolü kullanılmıştır (Şekil 2.1).

Şekil 2.1 Sistematik literatür taraması protokolü

2.1.1 Tarama Geliştirilmesi

GHTorrent veri kümesinin tüm atıflarını çıkarmak için bir uygulamadan[1] faydalanılmıştır. GHTorrent veri kümesinin referans çalışması olarak belirtilen çalışmaya [8] atıf yapan 332[2] çalışmanın tümü gözden geçirilmiştir. Çalışmalara, yakın zamanda yayınlanan bir inceleme çalışmasındakine benzer dışlama (exclusion) kriterleri uygulanmıştır [2]. Bu kapsamda, İngilizce dışında herhangi bir dilde yazılmış çalışmalar, erişim problemi olan çalışmalar, raporlar, kitaplar, ve tezler çıkarılmıştır (Tablo 2.1). Ayrıca, veri kümesini kullanmadığı halde sadece bazı bölümlerde kaynak makaleye atıf yapan (ilgili çalışma veya benzer veri kümeleri bölümünde) çalışmalar da çıkarılmıştır. Makale tarama ve sınıflandırma işlemleri çapraz doğrulama yöntemi ile iki araştırmacı tarafından yapılmıştır.

[1] Harzing's Publish or Perish
[2] Son kontrol 24 Temmuz 2019 tarihinde yapılmıştır.

Tablo 2.1 Çalışma dışlama kriterleri

Dışlama Kriteri	Adet
Dil problemi	25
Erişim problemi	18
Kitap veya tez çalışmaları	49
İlgili çalışma atıfı	47
Benzer veri kümesi atıfı	16
Rapor ve demeç çalışmaları	5
TOPLAM	160

Yayın seçme işleminin ardından kalan 49 dergi makalesi ve 123 konferans bildirisi olmak üzere toplam 172 çalışma incelenmiştir. İlk olarak çalışmaların üst verisinden bazı bilgiler elde edilmiştir.

- Başlık, anahtar kelimeler ve özet bölümü
- Yazar bilgileri
- Çalışma amacı, yöntemi ve araştırma soruları
- GHTorrent ile birlikte kullanılan harici veri kümesi bilgisi
- Veri kümesi sürümü (dump)
- Yayınlandığı yer bilgisi
- Çalışmaya yapılan atıf sayısı

2.1.2 Çalışmalara Genel Bir Bakış

Tez çalışmasında 2012-2019 yılları arasındaki yayınlardan 172 tanesi belirlediğimiz kriterler çerçevesinde değerlendirilmeye alınmıştır. Yayınların yıllara göre dağılımları Tablo 2.2'te verilmiştir.

Tablo 2.2 Yıllara göre çalışma sayıları

Yıl	2012	2013	2014	2015	2016	2017	2018	2019
Adet	1	1	22	26	28	36	41	17

Dergilerde yayınlanan çalışmaların bilgileri Tablo 2.3'de verilmiştir. Buna göre, en yüksek yayın sayısı "Empirical Software Engineering" dergisine aittir (ArXiv makaleleri hariç). Sadece 1 makaleye sahip dergi isimleri verilmemiş, bu dergiler "Diğer" kategorisinde toplanmıştır.

Tablo 2.3 İndeksli dergilere göre yayınların dağılımı

Dergi Adı	Adet
Empirical Software Engineering	8
Information and Software Technology	6
IEEE Trans. Software Engineering	4
Journal of System and Software	2
Physica A	2
IEEE Access	2
PeerJ	2
ArXiv	10
Diğer	13
TOPLAM	49

Dergilerin yanı sıra, ilgili çalışmaların bir kısmı da farklı konferanslarda sunulmuş bildirilerdir (Tablo 2.4). Bunların arasında en önde MSR (International Conferences of Mining Software Repositories) ve ICSE (International Confereneces on Software Engineering) konferansları gelmektedir. İki ve daha az çalışma bulunan konferanslar "Diğer" kategorisinde verilmiştir.

Tablo 2.4 Konferans yayınlarının dağılımı

Konferans Adı	Adet
International Conferences of Mining Software Repositories (MSR)	40
International Confereneces on Software Engineering (ICSE)	9
Int. Conf. on Soft. Eng. & Knowledge Eng. (SEKE)	5
Int. Conf. on Soft. Analysis, Evolution and Reengineering (SANER)	5
Int. Conf. on Soft. Maintenance and Evolution (ICSME)	3
Asia-Pacific Software Engineering Conference (APSEC)	3
Symposium on the Foundations of Soft. Eng. (FSE)	3
Int. Conf. on Connected Health: App., Systems and Eng. (CHASE)	3
Int. Workshop on Emotion Awareness in Soft. Eng. (SEmotion)	3
Diğer	49
TOPLAM	123

Tablo 2.5'te yayınlarda problemleri çözmek için kullanılmış olan yöntemler gösterilmiştir. Buna göre, en sık kullanılan yöntem "istatistik" olarak görülmektedir. Bu kategori; istatistik, matematik ve olasılık ile ilgili yöntemleri içermektedir. Veri kümesinin oldukça zengin bir metin içeriği olmasına rağmen, metin madenciliği diğer

yöntemlere nispeten daha az kullanılmıştır. Veri görselleştirme, veri kümesi kullanımı, yeni veri kümeleri üretme, vs. problemlere değinen çalışmalar "Diğer" kategorisinde verilmiştir.

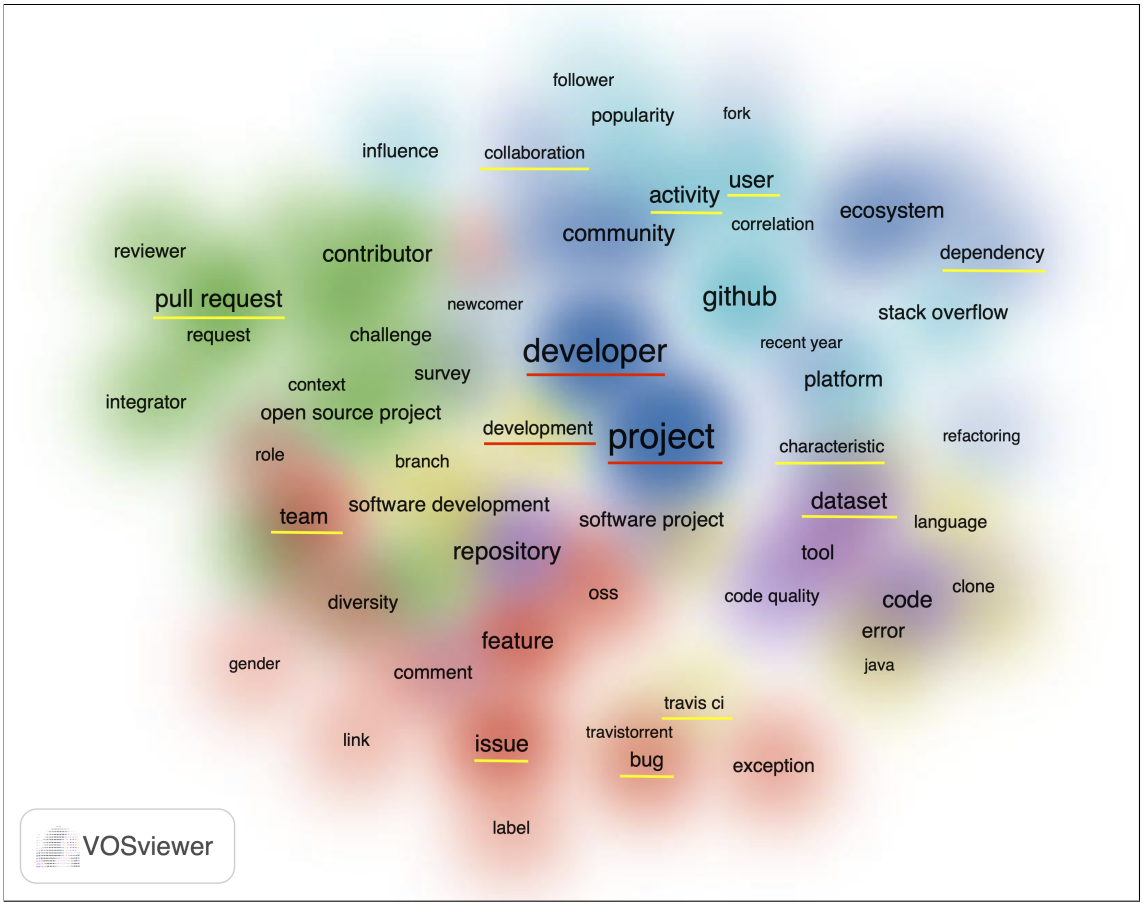
Tablo 2.5 Çalışmalarda kullanılan yöntemlerin dağılımı

Yöntem	Adet
İstatiksel	68
Makine öğrenmesi	41
Anket çalışması	25
Metin madenciliği	25
Diğer	34

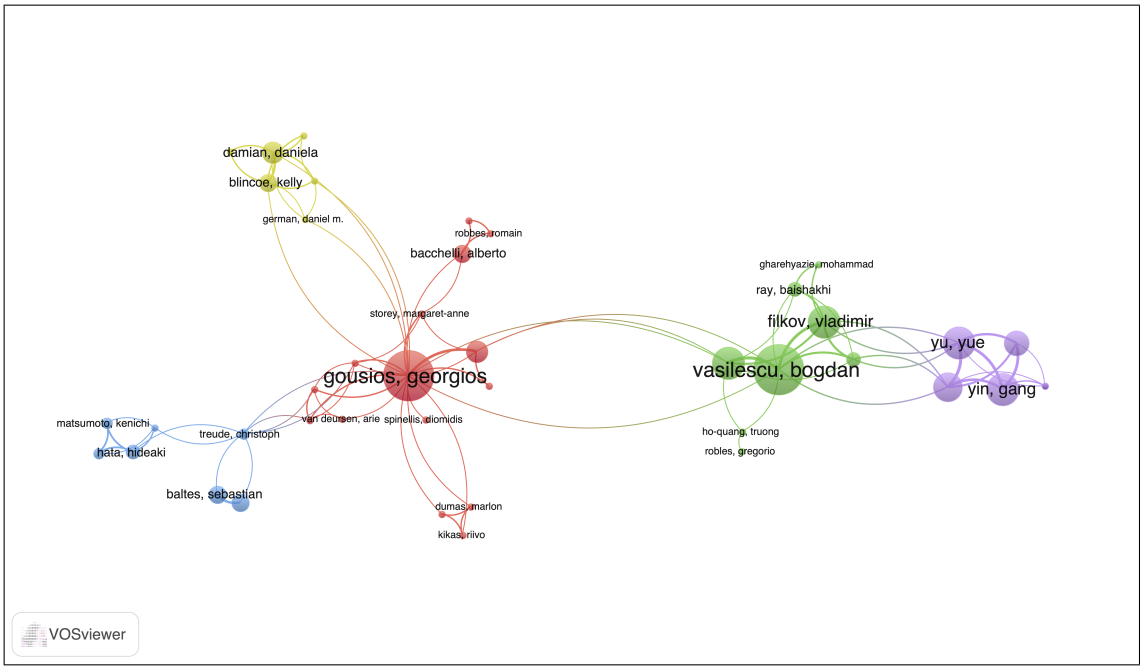
Yayınların başlıkları ve özet bölümleri çalışma hakkında genel olarak bir bilgi vermektedir. Buradan yola çıkarak, çalışmaların bu bölümlerinde bulunan kelimelerden öne çıkan anahtar kelimeleri bulmak hedeflenmiştir. Öne çıkan kelimelerin elde edilmesi için kelimelerin birlikte kullanımına ve kelime frekansına bağlı olarak üretilen bir grafikten faydalanılmıştır. Yoğunluk küme grafı adı verilen grafik VosViewer[1] uygulaması ile üretilmiştir (Şekil 2.2). İlk olarak, kelime fontunun büyüklüğü, kelimenin frekansı ile doğru orantılı olarak hesaplanmıştır. Ardından, kelimenin grafikteki konumunu bulmak için, kelimelerin birlikte kullanım bilgisinden faydalanılmıştır. Örneğin, *developer*, *project* ve *github* kelimeleri diğer kelimelerin çoğu ile birlikte kullanıldığı için grafın ortasında ve büyük font ile gösterilmiştir. Grafikteki renkler ise kelimelerin birlikte kullanımına göre elde edilen grupları ifade etmektedir. Çalışmaların alt sınıflara ayrılmasında bu grafik önemli rol oynamıştır.

Son olarak, incelenen yayınların yazar bilgileri kullanılarak bu alanda çalışan araştırmacılar ve işbirlikleri gösterilmiştir (Şekil 2.3). Buna göre, Georgios Gousios ve Bogdan Vasilescu isimli araştırmacıların ekiplerinin konu ile ilgili olarak öne çıktığı görülmektedir.

[1] https://www.vosviewer.com



Şekil 2.2 Yoğunluk küme grafı ile anahtar kelime tespiti



Şekil 2.3 Yazar işbirliklerinin gösterimi

2.2 Açık Kaynak Dağıtık Yazılım Geliştirme Zorlukları

Yayınları sınıflandırmak için ilk olarak GitHub platformunun kendi doğasından faydalanılmıştır. Bu tür platformlar için "kullanıcı veya geliştirici (developer)" ve "proje (project)" platformun omurgası olarak gösterilebilir. Ardından, yoğunluk küme grafına (Şekil 2.2) bakıldığında, bu iki terimin yanında "geliştirme (development)" kelimesinin de merkezde olduğu görülmektedir. Son olarak, bazı çalışmalar doğrudan veri kümesi ile ilgili olduğu için "veri kümesi" isimli bir başlıkta verilecektir. Veri kümesi ile ilgili çalışmalar Bölüm 5.1'de detaylı olarak incelenecektir. Böylece, yayınlar kullanıcı, geliştirme, proje ve veri kümesi olmak üzere 4 başlık altında toplanmıştır. Bu başlıklar altında sınıflandırılan yayınların hangi problem veya zorluk (challenge) üzerinde çalıştığının da tespit edilmesi gerekmektedir. Bu zorlukları belirlemek için bir başka literatür taraması çalışmasından [5] ve Bölüm 2.1'de verilen yoğunluk küme grafından faydalanılmıştır. Belirlenen kelimeler, problemleri en kapsayıcı şekilde tanımlayan terimlerden seçilmiştir. Böylelikle, çalışmalar ilgili 4 başlık altında 16 zorluk olarak kategorize edilmiştir (Tablo 2.6).

Tablo 2.6 Başlıklara ve zorluklara göre yayınların dağılımı

Kategori	**Adet**	**Zorluk**	**Adet**	**Takma Ad**
Kullanıcı	69	Aktivite	39	ACTV
		Etkileşim	39	INTR
		İnceleme - Atanma	9	REVI
		Karakterizasyon	35	CHAR
Proje	63	Sorun/hata	24	ISSU
		Takım ve Üyelik	11	TEAM
		Bağımlılık	9	DEPE
		Karakterizasyon	36	CHAR
Geliştirme	59	Pull Request (PR)	18	PREQ
		Kaynak Kod	35	CODE
		Sürekli Entegrasyon	17	CONT
		Kalite	12	QUAL
Veri Kümesi	28	Tanım ve Kullanım	4	DEFI
		Alt veri kümesi	4	SUBS
		Genişletme ve türev	7	AUGM
		Yardımcı	13	HELP

İlgi alanlarına göre makale sayılarının yanısıra, her bir kategori ve zorluk için bu bölümde ayrıntılı bilgilerin verildiği tablolar sunulmuştur. Tablolarda çalışmaların referans numarası ve hangi zorluğa odaklandığı gösterilmiştir (Hücrelerdeki "*x*" karakteri çalışmanın o kolondaki zorlukla ilgilendiğine işaret eder).

2.2.1 Kullanıcı Açısından Zorluklar

GitHub kullanıcıları, yazılım yaşam döngüsündeki tüm etkinlikleri gerçekleştirirken, katkıda bulunanlar (yorum, hata bildirimi, vs), kod geliştiriciler, proje yöneticileri, vs. nitelikleri ile yazılım projelerinde başrolde bulunmaktadırlar. Bu bağlamda, kullanıcı başlığı altında çok fazla yayın bulunmaktadır. Kullanıcı başlığı altında aktivite, etkileşim, inceleme/atama ve karakterizasyon olmak üzere dört kategoride zorluklar sunulmuştur (Tablo 2.7).

Aktivite (ACTV): Genel olarak, bu başlık, GitHub kullanıcılarının kodlama geçmişi, yorum, ve performans gibi geçmiş aktivitelerindeki katkılarını kapsamaktadır.

Etkileşim (INTR): GitHub kullanıcıları hem kendi aralarında hem de diğer platformlarda (Stackoverflow, Twitter, vs.) sürekli bir etkileşim halindedirler. Kullanıcılar arasındaki takip etme, izleme, çatallama (forking) gibi olaylar bu başlıkta toplanmıştır.

İnceleme - Atanma (REVI): *Pull request (PR)* değerlendirme, proje sorunlarına veya hatalarına görevlendirilme hakkında yapılan çalışmalar bu başlıkta incelenmiştir.

Karakterizasyon (CHAR): Yukarıdakilerin dışında kalan, kullanıcıların duygusal aktiviteleri, cinsiyet, gönüllülük, etkinlik gibi özelliklerine göre sınıflandırılmaları gibi problemlere odaklanan çalışmalar da bu başlıkta verilmiştir.

Tablo 2.7 Kullanıcı kategorisindeki zorluklara odaklanan çalışmalar

Yayın ID	Adet	ACTV	INTR	REVI	CHAR
[9–12]	4	x			
[13–24]	12		x		
[25, 26]	2			x	
[27–36]	10				x
[37–46]	10	x	x		
[47–49]	3	x		x	
[50–60]	11	x			x
[61–66]	6		x		x
[67–69]	3	x	x	x	
[70–76]	7	x	x		x
[77]	1	x	x	x	x

Kullanıcı kategorisinde en çok çalışılan zorlukların aktivite ve etkileşim başlıklarında olduğu görülmektedir. GitHub platformunun açık kaynak desteği ve topluluk tabanlı proje depolama servisi olmasının doğal bir sonucu olarak bu eğilimlerin ortaya çıktığı söylenebilir.

2.2.2 Proje Açısından Zorluklar

DVCS'lerin temel motivasyonu, hedeflenen ürünlerin açık kaynaklı (public) projeler temelinde geliştirilmesidir. Proje açısından zorluklar; sorun/hata, ekip/üye, bağımlılık ve karakterizasyon olmak üzere 4 başlıkta verilmiştir (Tablo 2.8).

Sorun/hata (ISSU): Bu başlık, devam eden veya tamamlanmış görevler (issue, commit, task, vs.), hata oluşumu ve hata önceliklendirmesi gibi konuları içerir. Ayrıca, bir projeye yapılan katkının türünü tanımlayan sorun-hata-özellik (issue-bug-feature) üçlüsünün farklılıkları ve sınıflandırılması da bu başlık altında yer almaktadır.

Takım/üye (TEAM): Ekip çeşitliliği ile ilgili bazı özellikler (konum, cinsiyet, gönüllülük, kalıcılık vs.), ekip olarak yapılan aktiviteler, bir ekibe katılma veya ekipten ayrılma davranışları, ekiplerdeki çekirdek (core) veya harici (external) üyeler ve ekiplerin yazılım kalitesine etkisi gibi konulara odaklanan çalışmalar bu başlıkta verilmiştir.

Bağımlılık (DEPE): GitHub projelerindeki çeşitli bağımlılıklara, programlama dilleri, kodlar, sorunlar, çatallar, ve kod kopyaları (code clones) açısından odaklanan çalışmalar bu başlıkta toplanmıştır. Ayrıca, projeler arasındaki bazı ilişkiler ve bir projenin hayatta kalmasına dair parametreler de incelenen diğer zorluklardır.

Karakterizasyon (CHAR): Bu başlık, GitHub depo (repository) özellikleri, projelerin benzersiz bölümleri, projelerin dil veya tasarım gibi bazı parametreler açısından çeşitliliği, açık kaynaklı projelerin özellikleri, GitHub ekosistemi ile ilgili bazı konular, depo yapıları (repository artifacts), dallanma (branching) gibi konuları içerir.

Tablo 2.8 Proje kategorisindeki zorluklara odaklanan çalışmalar

Yayın ID	Adet	ISSU	TEAM	DEPE	CHAR
[17, 25, 34, 44, 48, 78–89]	17	x			
[36, 53, 65, 90, 91]	5		x		
[92]	1			x	
[9, 23, 35, 51, 59, 76, 93–110]	24				x
[33, 111]	2	x	x		
[112, 113]	2	x		x	
[60, 114]	2	x			x
[115–118]	4		x		x
[119–123]	5			x	x
[124]	1	x		x	x

Proje kapsamındaki çalışmaların çoğu karakterizasyon ile ilgilidir. GHTorrent'in projeler hakkındaki zengin özelliklerinin bu sonucu ortaya çıkardığı düşünülmektedir.

2.2.3 Yazılım Geliştirme Açısından Zorluklar

Yazılım projelerinde hedeflenen ürünün performansını etkileyen temel faaliyetlerden biri de geliştirme süreci olarak düşünülebilir. Geliştirme alanındaki zorluklar *pull request*, kaynak kod, sürekli entegrasyon ve kalite olmak üzere 4 başlık altında sınıflandırılmıştır (Tablo 2.9).

Pull Request (PR) (PREQ): PR sınıflandırma veya önceliklendirme, PR tanımlama metinleri, bir PR hakkında yapılan yorum/tartışma içerikleri, PR işleminin kabulü veya reddi gibi konular bu başlık kapsamındadır.

Kaynak Kod (CODE): Projelerin programlama dilleri, kodlar arası bağlantılar (orijinal kod ve kod kopyalama), yeniden düzenleme (refactoring), taktik kodlar ve kod çakışması gibi konular bu başlık altındadır.

Sürekli Entegrasyon (CONT): Sürekli entegrasyon kullanımı ve yazılım kalitesine etkisi, yapı kırılma (build breakage) problemleri, test durumları, sürekli entegrasyon sürecinin otomatikleştirilmesi ile ilgili çalışmalar bu başlık altında verilmiştir.

Kalite (QUAL): Asıl amacı yazılım kalitesini irdelemek olan tüm çalışmalar bu başlıkta verilmiştir.

Tablo 2.9 Geliştirme kategorisindeki zorluklara odaklanan çalışmalar

Yayın ID	**Adet**	**PREQ**	**CODE**	**CONT**	**QUAL**
[10, 11, 19, 28, 29, 77, 80, 91, 125–128]	12	x			
[12, 22, 31, 34, 41, 61, 78, 83, 84, 92, 109, 129–135]	18		x		
[32, 43, 136–140]	7			x	
[141–143]	3	x	x		
[144, 145]	2	x			x
[146–151]	6		x	x	
[57, 152–157]	7		x		x
[158–160]	3			x	x
[161]	1	x	x	x	

Bu alanda, çoğu çalışma kaynak kodla ilgilidir. Kaynak kodun incelenmesinin projeler hakkında özgün bilgiler vermesi, projenin kalitesinin ve sürdürülebilirliğinin kodla olan ilişkisi bu sonucu ortaya çıkarmıştır.

Sonuç olarak, bu literatür taraması sayesinde, OSS geliştirme zorlukları sınıflandırılmış, bu alan ile ilgilenen araştırmacılar için odaklandıkları problemler hakkında yapılan çalışmalara bir bakışta ulaşmaları sağlanmıştır. Ayrıca henüz çok odaklanılmamış, ancak önemli olduğu düşünülen problemler de, açık zorluklar olarak ortaya çıkarılmıştır. Tezin bu bölümünde yapılan inceleme çalışması dergi makalesi olarak yayınlanmıştır.

3
AÇIK KAYNAK DAĞITIK YAZILIM GELİŞTİRME

Açık kaynak terimi, tasarımı halka açık olan, insanların değiştirebileceği ve paylaşabileceği ürünleri ifade etmektedir [162]. Açık kaynak yazılım ise, herkesin inceleyebileceği, değiştirebileceği ve geliştirebileceği bir kaynak koda sahip olan yazılımdır. Geliştiriciler, bir başkasına ait açık kaynak yazılımları değiştirirken veya kullanırken bazı lisansları kabul etmek zorundadırlar.

Açık kaynak lisansları, kullanıcıların bir yazılımı kullanma, inceleme, değiştirme ve dağıtma şeklini belirleyen kurallardır. Genel olarak, açık kaynak lisanslar, bilgisayar kullanıcılarına açık kaynak yazılımları diledikleri herhangi bir amaçla kullanma iznini verir. Telif müsaadeli (copyleft) olarak adlandırılan bazı açık kaynak lisanslar, değiştirilmiş bir açık kaynak programı yayınlayan herkesin aynı zamanda o programın kaynak kodunu da yayınlaması gerektiğini şart koşar. Geliştiriciler, bu lisanslar ile açık kaynak yazılımları kendi projeleri için kullanırken, bir yandan da bu yazılımları hep birlikte geliştirmeyi hedeflerler. Açık kaynaklı projelerin bir topluluk tarafından iş birliği içinde geliştirilmesini ise dağıtık kod geliştirme sağlamaktadır.

Dağıtık yazılım geliştirme farklı fiziksel çalışma alanlarında bulunan ve merkezi olmayan ekiplerle bir yazılımın planlanması, tasarlanması, oluşturulması, test edilmesi ve yönetilmesi anlamına gelmektedir [163]. Dünyanın çok farklı konumlarında çalışan ekiplerinden oluşan 80 dağıtık yazılım geliştirme ekibinin performansı üzerine yapılan bir çalışmada, araştırmacılar, uygun yönetim süreçleri uygulandığında, dağıtık ekiplerin (sanal ekipler), aynı konumdaki ekiplere göre oldukça yüksek performans gösterebileceğini keşfetmişlerdir [164]. Dağıtık yazılım geliştirme ekipleri, web tabanlı forumlarını, proje yönetim platformlarını, kaynak kod veritabanlarını, mesajlaşma ve video konferans yazılımlarını kullanarak projelerini yürütürler. Bu bağlamda, açık kaynak yazılımlar ve onları geliştiren ekipler, versiyon kontrolü, dağıtık geliştirme, hata takibi, geliştiricilere bildirim gönderme, yorum yapma ve tartışma gibi çok farklı ihtiyaçlara gereksinim duyarlar. Kullanıcılar, bu gereksinimlerinin birçoğunu dağıtık versiyon kontrol sistemleri ile karşılamaktadırlar.

3.1 Dağıtık Versiyon Kontrol Sistemleri

Açık kaynak yazılım geliştirmenin yaygınlaşması ile Versiyon Kontrol Sistemi (VCS) de önemli bir ihtiyaç haline gelmiştir. Bu sistemler, kod geliştiren ekiplerin eşzamanlı çalışması için kolaylık sağlayacak birçok özelliği beraberinde getirmektedir. Kod versiyonlama ile yazılımın bir parçası için kod yazan bir geliştirici, eski sürümlere kolayca dönebilir, sürümler arasındaki değişiklikleri kontrol edebilir ve diğer geliştiricileri yazılımın veya sadece ilgili kodun yeni sürümünden haberdar edebilir. VCS kullanmanın temel avantajları, kullanıcılar için yazılımdaki tüm değişiklikleri takip etmek, geliştirme sürecini kolaylaştırmak, birden çok proje için kod yönetimi sağlamak olarak ifade edilebilir. VCS'nin dağıtık hale gelmesi ise, geliştiricilerin kod üzerinde birlikte çalışmasına ve görevleri dallara (branches) ayırarak yürütebilmesine imkan vermektedir. Dağıtık Version Kontrol Sistemi (DVCS), tüm kod tabanının (codebase) tam geçmişinin her geliştiricinin yerel bilgisayarına yansıtıldığı bir versiyon kontrol biçimidir. Merkezi yaklaşıma kıyasla DVCS, dallanmayı (branching) ve birleştirmeyi (merging) otomatik olarak yönetmeyi, çoğu işlemi hızlandırmayı, çevrimdışı çalışma yeteneğini geliştirmeyi, ve yedekleme için tek bir konuma güvenmemeyi sağlar [165].

3.1.1 Versiyon kontrol sistemlerinin kıyaslanması

Versiyon kontrol sistemlerinden öne çıkanlar; Concurrent Versions System (CVS), Apache Subversion (SVN), Git, GNU Bazaar ve Azure DevOps olarak verilebilir. Tablo 3.1'de bu sistemlerin farklı özellikler ve komutlar açısından karşılaştırması[1] verilmiştir.

Tablo 3.1 En çok bilinen versiyon kontrol sistemlerinin karşılaştırılması

	CVS	**SVN**	**Git**	**GNU Bazaar**	**Azure DevOps**
a	CVS Team	Apache	Junio Hamano	Canonical	Microsoft
b	1986	2000	2005	2005	2006
c	İS	İS	Dağıtık	İS + Dağıtık	İS + Dağıtık
d	Yok	Var	Var	Var	Var
e	Yok	Var	Var	Var	Var
f	Yok	Yok	Var	Var	Yok
g	Yok	Yok	branch	init-branch	mkstream
h	Yok	Yok	rebase	rebase	chstream
i	Var	Var	Kısmen	-	Var
j	Var	Kısmen	Yok	Yok	-

[1]Karşılaştırma wikipedia sayfasındaki tablolardaki bazı özellikler seçilerek yapılmıştır, tüm karşılaştırma için bu linki kullanılabilir.

a. Destekçi olan kurum/kuruluş/şahıs (maintainer) bilgisini gösterir.

b. İlk versiyonun çıkarıldığı tarihi (first release) verir.

c. Kaynak kod depo kopyaları tutulma biçimi (repository model) 2 şekilde sağlanır. İstemci-sunucu (İS) modelde, kullanıcılar bir istemci aracılığıyla bir ana depoya erişir. Dağıtık modelde ise, depolar eş zamanlı yürütülür ve kullanıcılar genellikle çalışma kopyalarına ek olarak sürüm geçmişinin mevcut olduğu yerel bir depoya sahiptir.

d. En küçük *Commit* (atomik commit) özelliği, tüm değişikliklerin ya tam olarak yapıldığının ya da hiçbir değişikliğin yapılmayacağının garantisini ifade etmektedir.

e. Birleştirme izleme (merge tracking); bir sistemin hangi dallar arasında hangi değişikliklerin birleştirildiğini hatırlama durumunu açıklayan parametresidir.

f. İmzalı revizyon (signed revision); sistemdeki, OpenPGP gibi bir formatta, revizyonların dijital imzalanma özelliğini ifade eder.

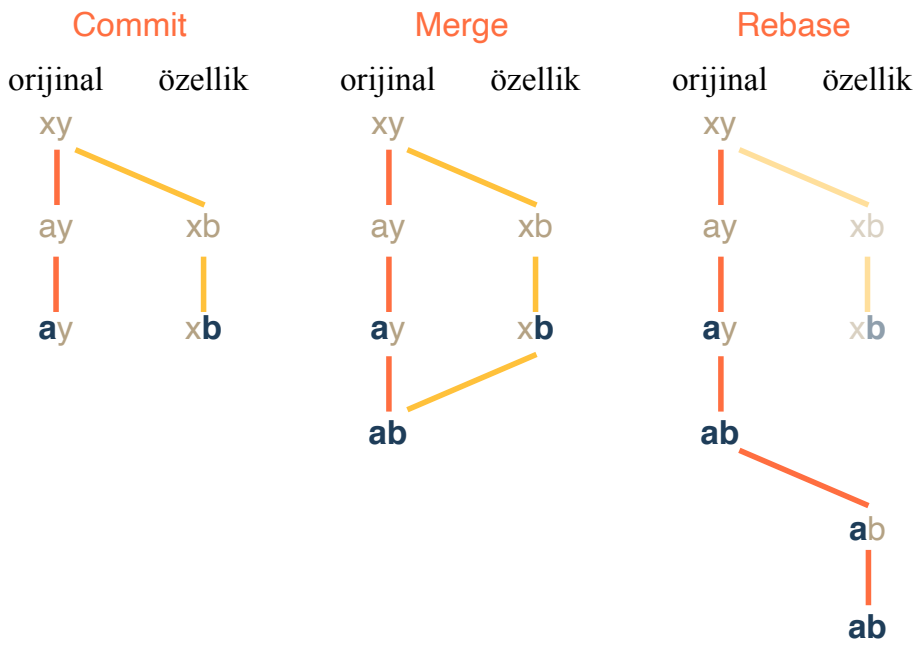
g. Yerel dallanma (local branching) komutu; orijinal uzak depoda bulunmayan bir yerel dal oluşturmaya yarar.

h. Yeniden Tabana Dönme (rebase) komutu; bir deponun geçmişini değiştirerek bir dizi işlemi kolayca değiştirmeye olanak tanır. Mantık olarak birleştirme (merge) işlemine benzer olsa da, *rebase*, geçmiş işlemlerin karmaşasını azaltmak için, bir dalda yapılan aktiviteleri *master* dalda birleştirir (Şekil 3.1).

i. Büyük depo desteği özelliği; sistemin bir gigabayt veya daha büyük depoları etkili bir şekilde kontrol edip edemediğini gösterir.

j. Zaman damgası koruma (timestamp preservation); teslim alma (checkout) sırasında, son değiştirilen dosya sistemi özniteliğinin üzerine yazma süresinin eklenmesini ifade eder.

Bu sistemlerin içerisinde en çok tercih edilenlerden biri de *Git* sistemidir.



Şekil 3.1 *Rebase* komutunun birleştirme işleminden farkı [166]

3.1.2 Git protokolü ve tarihçesi

Linux çekirdeği, oldukça geniş kapsamlı bir açık kaynak yazılım projesidir. Linux çekirdek bakımı (1991–2002) yapılırken yazılımdaki değişiklikler, yamalar ve arşivlenmiş dosyalar olarak aktarılmıştır. 2002'de Linux çekirdek projesi, BitKeeper adlı tescilli bir dağıtık VCS kullanmaya başlamıştır. 2005 yılında, Linux çekirdeğini geliştiren topluluk ile BitKeeper'ı geliştiren ticari şirket arasındaki ilişki bozulmuş ve aracın ücretsiz statüsü iptal edilmiştir. Bu durum, Linux geliştirme topluluğunu (özellikle Linux'un yaratıcısı Linus Torvalds'ı) BitKeeper'ı kullanırken edindiği deneyim ile kendi araçlarını geliştirmeye teşvik etmiştir. Böylelikle hızlı, basit tasarımlı, doğrusal olmayan geliştirme için güçlü destek (binlerce paralel dal) sağlayan, tamamen dağıtık, Linux çekirdeği gibi büyük projeleri verimli bir şekilde idare edebilen (hız ve veri boyutu) *Git* sistemi tasarlanmıştır[1]. 2005'teki ilk sürümünden bu zamana kadar sistem, kullanımı kolay olacak şekilde geliştirilmiştir. *Git*, merkezi bir servis olmadan çalışabilen dağıtık bir VCS'dir [167]. Diğerlerinin arasında *Git* sisteminin öne çıkmasının nedenleri aşağıdaki gibi verilebilir:

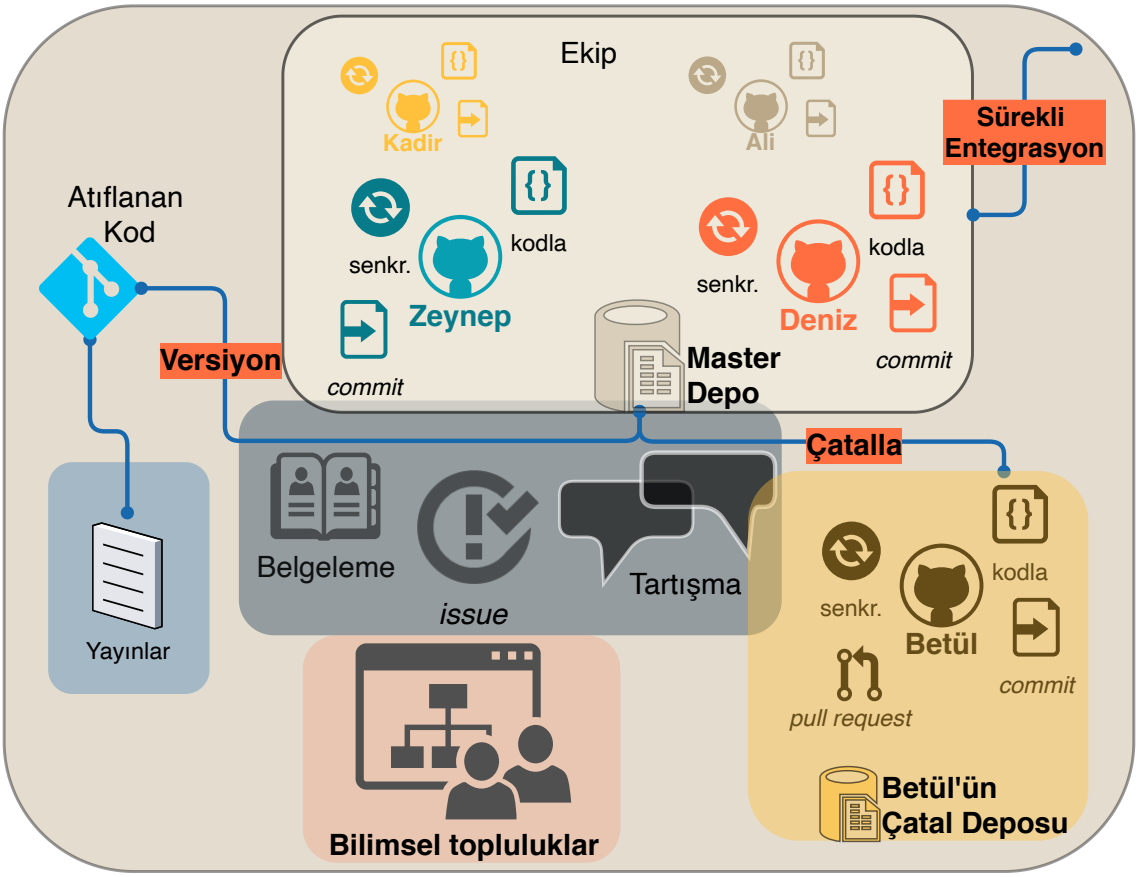
- Oldukça hızlı çalışan bir sistem olması
- Çevrimdışı çalışabilme özelliğine sahip olması
- Her şeyin ihtiyaç duyulduğunda geri alınmasının mümkün olması
- Veri kaybetmenin neredeyse imkansız olması (tam güvenilir)

[1] https://git-scm.com/

- Parçalama (staging) alanı sayesinde parçalı *commit* imkanına sahip olması
- Tüm iş akışlarına uygun ve kolay uygulanabilir olması
- Dallanma (branching) özelliğini tam anlamıyla kolay ve hızlı kullandıran ilk sistem olması
- Referanslarının üst düzey açık kaynak projeler olması (Ruby On Rails, jQuery, Perl, Debian, Linux Kernel, vs.)

3.1.3 GitHub

GitHub, *Git* depoları için merkezi depolama sağlayan bir ortamdır. GitHub, 50 milyondan fazla insanın öğrendiği, paylaştığı, birlikte yazılım geliştirmek için çalıştıkları bir platformdur. GitHub'a açık internet üzerinden ücretsiz olarak erişilebilir. GitHub ismindeki "Hub" kelimesi, *git* ile ilgili herşeyin etrafında olan bir merkez anlamı taşımaktadır. GitHub, Git'e basitleştirilmiş ve entegre bir arayüz sağlar ve ayrıca temel kullanıcı yönetimi, bir sorun izleyici, ilgili wiki'ler, proje barındırma ve diğer özellikleri de kullanıcılarına sunmaktadır.



Şekil 3.2 GitHub genel çalışma mantığı [168]

Github ortamın genel çalışma mantığı ve yapısal özellikleri Şekil 3.2'de verilmiştir [168]. Şekil 3.2'de görüldüğü gibi, GitHub geliştiricileri, platformun zengin özellikleri sayesinde, projelerinin kodlamadan sorunlara, sürekli entegrasyondan sürümlemeye kadar her aşamasını kolay bir şekilde yöneterek sürdürürler. Bu ortamın mantığını anlamak için öncelikle bazı terimler hakkında bilgi sahibi olmak gerekmektedir.

3.1.3.1 Depo (Repository)

GitHub ortamı hakkında bilinmesi gereken ilk terim, belirli bir proje için tüm dosyaların depolandığı konum olarak tanımlanan "*repository*", "*repo*" veya ""*depo*" gibi farklı isimlerle anılan ifadedir. Bir depo, belirli bir proje için tüm dosyaların depolandığı bir konumdur. Her projenin kendi deposu vardır ve ona benzersiz bir URL ile erişilebilmektedir. GitHub ortamında bir depo oluşturma örneği Şekil 3.3'te verilmiştir [1]. Depolar açık (public) veya özel (private) olarak oluşturulabilir. Depolara, içerdiği proje hakkında ilk bilgiyi vermek için *README* dosyası eklemek faydalı olacaktır.

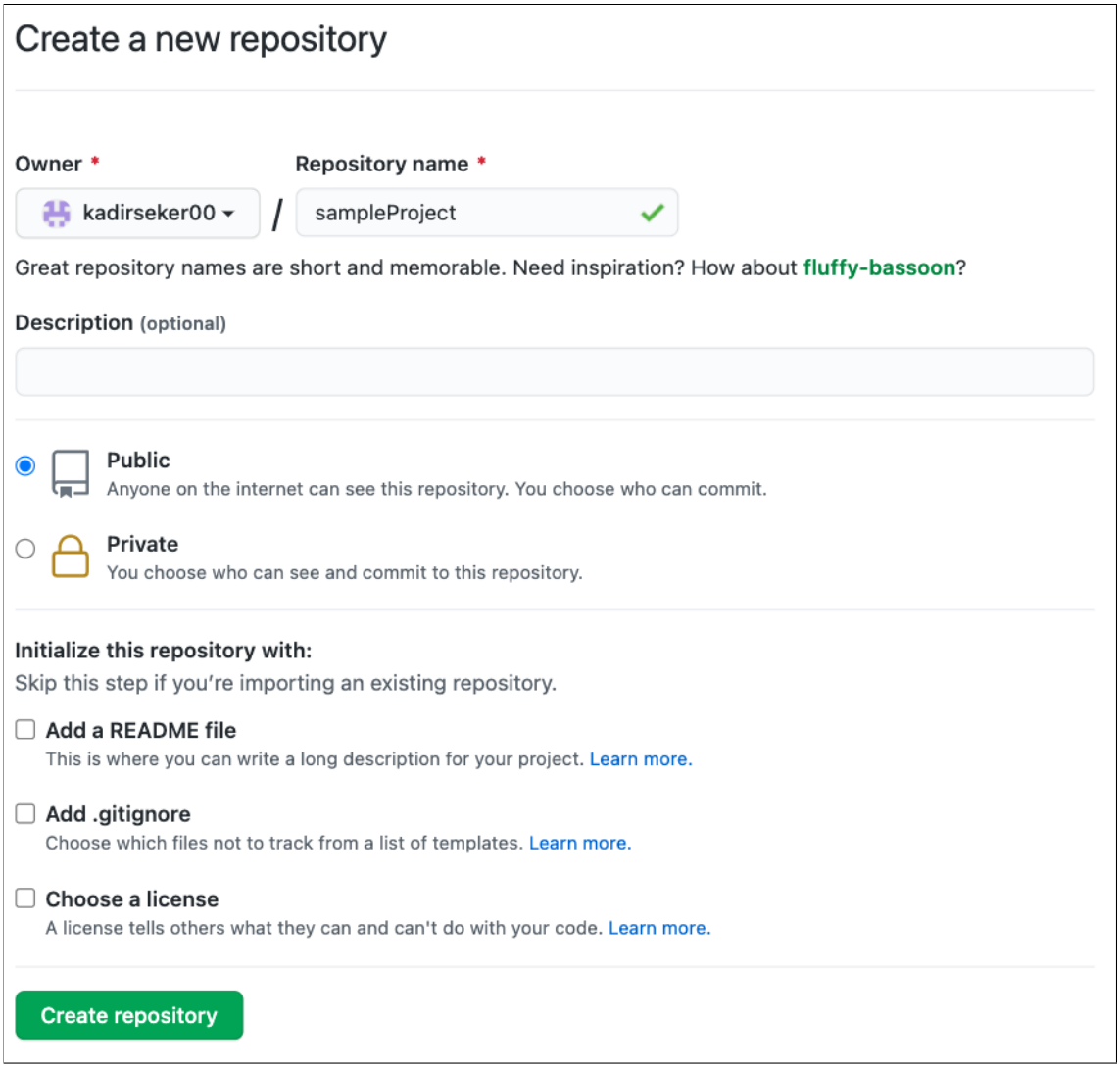
Bir *git* deposunda *commit* yaptığınızda, *git add dosyaadı* ve ardından *git commit* ile hangi dosyaların hazırlanıp yürütüleceği seçilir. Fakat bazı dosyaların yerel ortamdan GitHub deposuna (log dosyaları, API anahtarları, vs.) aktarılmaması -yok sayılması (ignore)- istenebilir. İşte bu aşamada *.gitignore* devreye girer. İlgili dosya isimleri *.gitignore* isimli dosyaya eklenir. Böylelikle, *Git* o dosyaları görmezden gelir ve onları izlememesi gerektiğini bilir.

3.1.3.2 Çatal (Fork)

Bilinmesi gereken bir diğer terim ise çatal (fork) olarak verilebilir. Çatal aslında deponun bir kopyasıdır ve geliştiriciye orijinal projeyi etkilemeden yapacağı değişiklikleri özgürce deneme imkanı sağlamaktadır. Ayrıca çatallanmış projeyi değiştirerek orijinal projeye katkı sağlama, çatallanmış versiyonu paylaşma veya çatallanmış projeden yeni bir ürün oluşturma imkanı da bu şekilde elde edilebilir. Bir repoyu çatallamak ve kopyalamak birbirine benzer olmasına karşın kullanıcıya ait çataldan orijinal repoya (upstream repository) değişiklikler önerme ve senkronizasyon ile orijinal repodan yerel çatala değişiklikleri getirebilme imkanı çatallamanın önemli iki farkıdır.

Kullanıcılar yeni bir depo oluştururken kendi isimleri ile proje ismini birleştirler. Bu şekilde proje tam adı oluşturulur. Bu mantıkta isim oluşturulması, çatallanan projelerin gösterilmesini de sağlamaktadır. Şekil 3.3'te *kadirseker00* kullanıcısının

[1] https://github.com/new

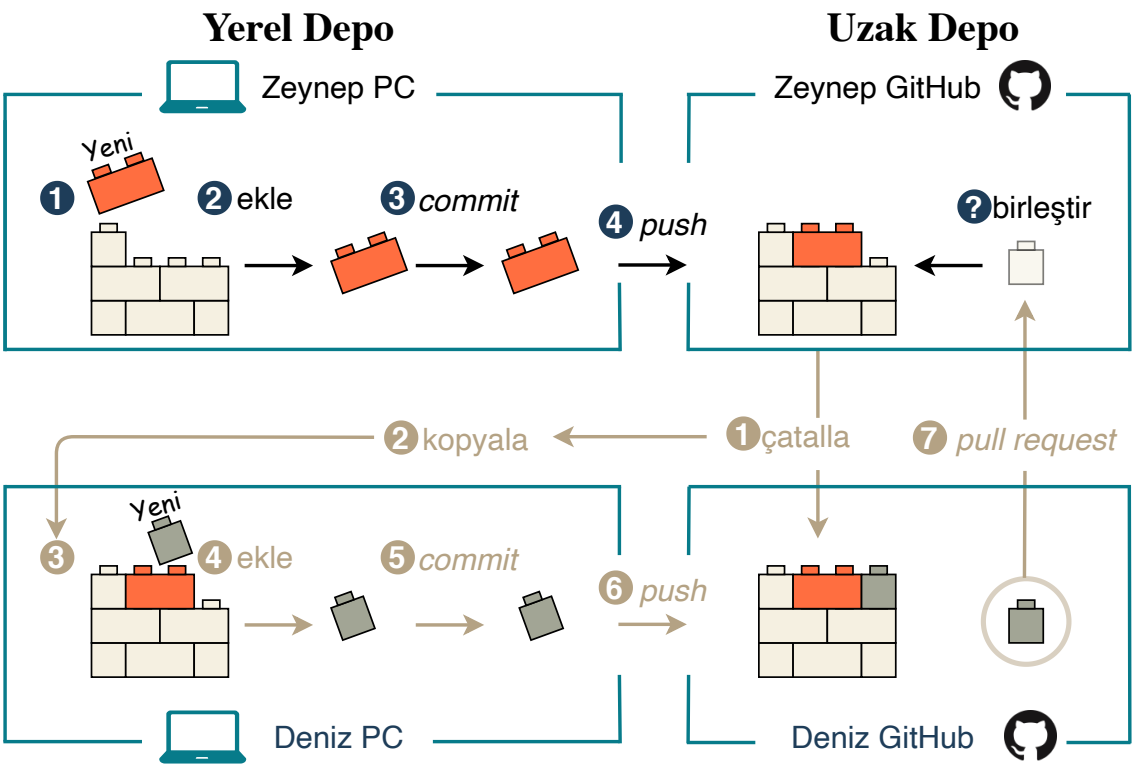


Şekil 3.3 GitHub ortamında bir depo oluşturulması örneği

sampleProject isimli deposunu *Emre* isimli bir kullanıcı çatalladığında, onun proje ismi *emre/sampleProject* olarak yeniden adlandırılacaktır. Bunun yanı sıra, GitHub ortamının esnekliğini gösteren bir başka özellik de, çatallanmış proje, orijinal projeden bağımsız bir ürüne dönüşmüşse isminin de değiştirilebilmesidir. Örnekteki durumda Emre yeni bir ürün üretmişse çatalladığı projenin ismini emre/NewProduct olarak da değiştirebilir. Bu değişiklik orijinal projedeki güncellemeleri almaya bir engel teşkil etmeyecektir.

Şekil 3.4'te çatallama iş akışı verilmiştir[1]. Zeynep, kendi yerel ortamındaki projesi için geliştirdiği bir kod parçasını GitHub deposuna gönderir (push). Deniz adlı bir diğer kullanıcı, bu depoyu kendi GitHub ortamına çatallar. Deniz, bu çatalın bir kopyası (clone) kendi yerel bilgisayarına indirir. Deniz projeye yeni bir ekleme (commit) yapar. Bu eklemeyi önce kendi GitHub deposuna gönderir (push). Bu değişikliği projenin

[1]https://dribbble.com/shots/2389144-Git-GitHub-Tutorial



Şekil 3.4 GitHub ortamında bir çatallama örneği

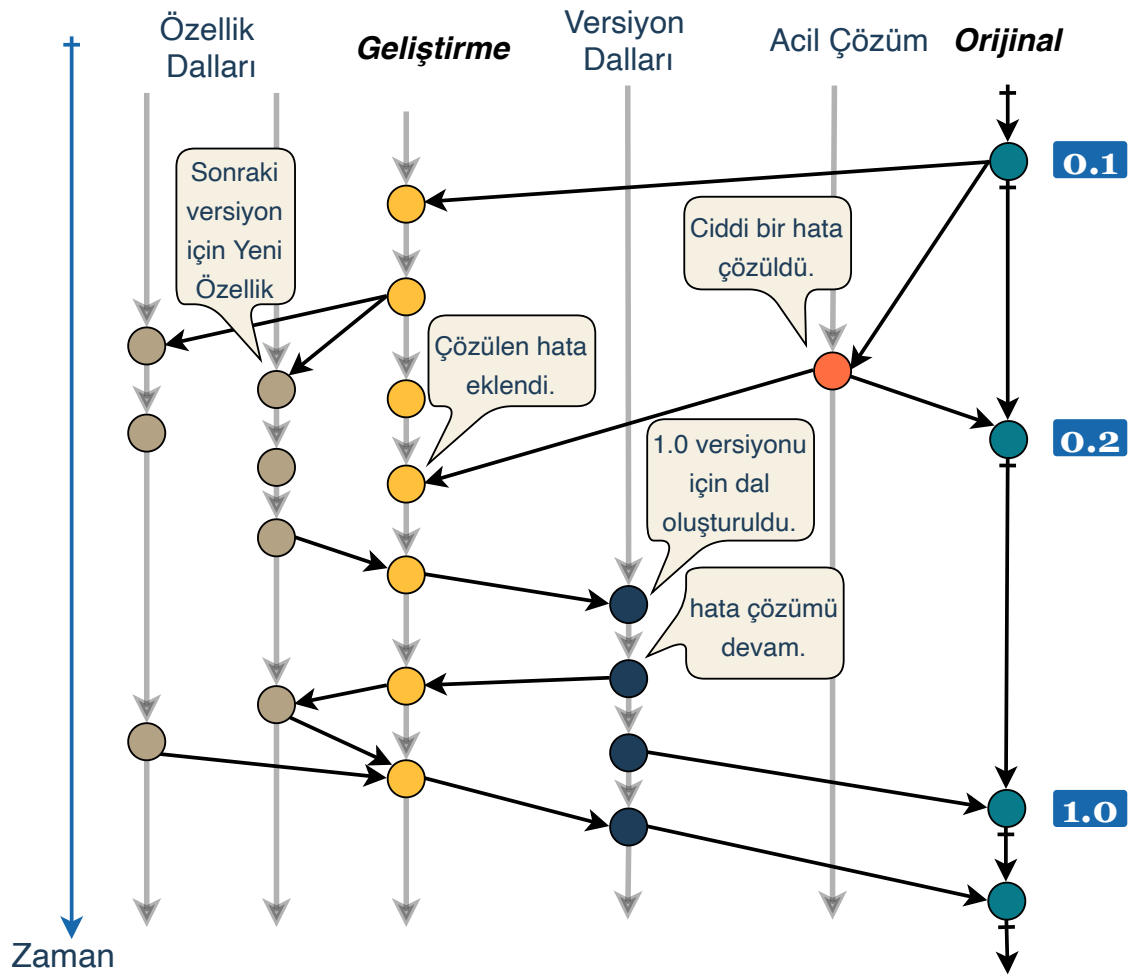
asıl sahibi Zeynep'e bildirmek için bir *pull request* açar. Zeynep, değişikliği inceler ve eğer isterse kendi projesine birleştirir (merge). Bu çalışma şeklinin en önemli avantajı kullanıcıların birbirinden bağımsız olarak güncellemeler yapabilmesidir.

Çatallama ile ilgili bir diğer terim de *push* olarak verilebilir. Açık kaynaklı projelerde, kullanıcılar sahibi oldukları çataldaki değişiklikleri orijinal depo ile karşılaştıran bir *pull request* açtıklarında, orijinal repoya *push* erişimi olan herkese değişiklikleri orijinal depoya gönderme izni verebilir. Bu, depo yöneticilerinin, birleştirme öncesinde kullanıcının sahip olduğu bir çataldan orijinal depoya yerel olarak *commit* yapmasına veya testler çalıştırmasına izin vererek işbirliğini hızlandırır. Aynı şekilde, bir depo üzerinde birden fazla kişi çalışıyorsa, yapılan *commit*, *push* yetkisi olan kişi tarafından depoya aktarılabilir.

3.1.3.3 Dal (branch)

Dal (Branch), aynı projede, kendine özgü bir alanda özellikler geliştirmeye, hataları düzeltmeye veya yeni fikirleri güvenli bir şekilde denemeye olanak tanır. Yeni bir dal, bir deponun varsayılan dalından (genellikle *master*) oluşturulur. Bu şekilde, diğer kişilerden ve varsayılan daldan bağımsız olarak bu yeni dal üzerinde çalışılabilir.

Projeler genellikle *master* ve *develop* olarak adlandırılan iki dalda yürütülür (Şekil 3.5).



Şekil 3.5 GitHub ortamında dallanma örneği [169]

Master dal sunulan bir sonraki sürümü gösterirken, *develop* dalı bir sonraki sürümde sunulacak geliştirmeleri yansıtır. *Develop* dalındaki kaynak kod kararlı hale geldiğinde, tüm değişiklikler *master* dala birleştirilmeli ve ardından bir sürüm numarasıyla etiketlenmelidir. Bu iki dal dışında başka destekleyici dallar da bulunabilir. Gelecekte belli olmayan bir sürüm için bazı özellikler geliştirmek adına özellik dalı (feature branch) kullanılabilir. Sürüm dalları (release branches), yeni bir üretim sürümünün hazırlanmasını destekler. Küçük hata düzeltmelerine ve bir sürüm için meta verilerin hazırlanmasına (sürüm numarası, oluşturma tarihleri, vs.) imkan verir. Tüm bu çalışmaları bir sürüm dalında yaparak, *develop* dalı bir sonraki büyük sürüm için özellikleri almak üzere yetkilendirilir. Bazı testlerin yürütülmesi için test dalları (test branches) oluşturulabilir. Geliştirmenin türüne göre yönetim biçimi de farklı olacaktır. Şekil 3.5'te görüldüğü gibi, bazı ciddi hatalar fark edilir edilmez acil çözüm dalında (hotfix branch) düzeltilir ve proje yeni versiyona doğrudan geçirilir.

3.1.3.4 Versiyonlama

Projeler geliştirilirken, yapılan değişiklikler belirli aralıklar ile canlı projeye birleştirilir. Bu birleştirmeler, yeni versiyonlar halinde sunulur. Versiyonlamanın genellikle bir standartı vardır. Son yıllarda anlamsal versiyonlama (semantic versioning- SemVer) mantığı ön plana çıkmıştır. Aktif projelerde neredeyse her gün yeni eklentiler (plugin/addon), kütüphaneler ve uzantılar oluşturulduğu için, yazılım geliştirme projelerini evrensel bir şekilde versiyonlamak, yazılımda neler olup bittiğini izlemenin en iyi yoludur. Anlamsal Sürüm Oluşturma, *X.Y.Z* biçiminde 3 bileşenli bir sayıdır. Bu sayıda X büyük bir revizyonu (major); Y, küçük bir revizyonu (minör); Z ise bir yamayı (patch) temsil etmektedir.

Burada anlaşılması gereken bir husus, hangi değişikliklerin ardından yeni bir versiyona geçileceğine karar verilmesidir. Bunu anlamak için Mozilla Firefox[1] projesine ait bazı versiyonlar ve bu versiyonlara geçişte yapılan değişiklik adetleri türleri ile birlikte Tablo 3.2'de verilmiştir. Mozilla Firefox projesinde sürüm aralığı 4 hafta olarak belirlenmiştir (acil yama güncellemeleri hariç). Tablo 3.2'de verilen revizyonlarda ilk dikkat çeken durum projeye yeni özellik eklenmesi sadece büyük revizyonlarda görülmektedir. Küçük revizyonlarda yeni özellik eklenmemesine karşı, çözülen hata sayısı oldukça fazladır. Yama (patch) değişimlerinde ise düzeltilen birkaç hata incelendiğinde genellikle kritik seviyede oldukları görülmüştür. Bu yüzden hata çözülür çözülmez versiyon geçişi tamamlanmıştır. Örneğin, 84.0.2 geçişinde çözülen 2 hata da kritik seviye olarak raporlanmıştır.

Tablo 3.2 Mozilla Firefox projesine ait bazı versiyonlar

Versiyon	Tarih	Tür	Değişiklik (Adet)			
			Yeni	Düzeltme	Değiştirme	Diğer
78.0	30-06-2020	Büyük	6	15	5	7
78.0.1	01-07-2020	Yama	0	1	0	1
78.0.2	09-07-2020	Yama	0	5	0	1
78.1.0	28-07-2020	Küçük	0	11	0	1
78.2.0	25-08-2020	Küçük	0	4	0	1
...	...	...	...	...	...	...
84.0	15-12-2020	Büyük	4	15	0	2
84.0.1	22-12-2020	Yama	0	2	0	1
84.0.2	06-01-2021	Yama	0	2	0	1
85.0	26-01-2021	Büyük	3	14	1	2

[1] https://www.mozilla.org/en-US/firefox/releases/

3.1.3.5 İşbirlikçi (Collaborator)

İşbirlikçi (collaborator), kısaca bir projenin *master* deposuna *push* erişimi verilmiş kişidir. Özel depolarda, depo sahipleri yalnızca ortak çalışanlara *push* erişimi verebilirler. Bunun dışında işbirlikçiler, çatallama, dalı yeniden adlandırma, *PR* oluşturma, değerlendirme ve birleştirme, *issue* veya *PR* etiketleme, versiyonları yönetme, projeye belge (wiki) ekleme, paketleri yönetme, yinelenen *issue* veya *PR* bildirme, etiketleme gibi birçok yetkiye sahip olmaktadırlar.

3.1.3.6 Katılımcı (Contributor)

Katkıda bulunan veya katılımcı kişi (contributor), bir projede bazı değişikliklere katkıda bulunmak isteyen, projenin çekirdek geliştirme ekibinde **olmayan** dışarıdan birisidir. İşbirlikçi ise, projenin çekirdek geliştirme ekibinde yer alan ve *master* deposuna *commit* yapma yetkisine sahip kişidir.

3.1.3.7 Diğer Servisler

GitHub dışında benzer özellikler ve çözümler sunan BitBucket ve GitLab gibi platformlar da bulunmaktadır. BitBucket, ekiplere projeleri planlamak, kod üzerinde işbirliği yapmak, test etmek ve devreye almak için ücretsiz özel Git depolarıyla birlikte tek bir yer sağlar. Takımlar, Bitbucket ortamında, üstün bir Jira entegrasyonuna, yerleşik sürekli entegrasyon ve dağıtıma ve 5 kullanıcıya kadar ücretsiz versiyonlara sahip olabilirler. GitLab temel olarak, *git* depo yönetimi, kod incelemeleri, sorun izleme, etkinlik beslemeleri ve wiki'ler sunmaktadır. GitLab şirketler için şirket içi (on-promise) kurulum imkanı da vermektedir. Ayrıca, güvenli kimlik doğrulama ve yetkilendirme için *LDAP* ve *Active Directory* sunucularına bağlanabilir. Tek bir GitLab sunucusu ile 25.000'den fazla kullanıcı idare edebilir. Genel hatlarıyla bu 3 ortamın kıyaslanması Tablo 3.3'te verilmiştir.

Tablo 3.3 GitHub, GitLab, ve Bitbucket platformlarının kıyaslanması

	GitHub	**GitLab**	**Bitbucket**
Kuruluş Tarihi	2008	2011	2008
Sahibi	Microsoft	GitLab	Atlassian
Destek	Git	Git	Git, Mercurial
Ücretsiz Özel Depo	Limitsiz	Limitsiz	Maksimum 5
Entegre CI	Yok	Var	Var
Açık kaynak	Hayır	Evet	Hayır
Tahmini geliştirici sayısı	40 Milyon	...	5 Milyon
Tahmini depo sayısı	100 Milyon	...	28 Milyon

3.2 Katkı Yöntemleri

GitHub ortamında, hataları veya eksikleri ortaya çıkarma, bunlar üzerinden proje izleme, projenin yönetimini sağlama, özellikler hakkında tartışma, kod eklemeleri yapma, yapılan geliştirmeleri onay ve değerlendirme sürecine sunma gibi çok farklı yollarla projelere katkıda bulunulabilir. Dağıtık geliştirmenin en önemli gereksinimlerinden olan bazı özellikler bu bölümde tanımlanmıştır.

3.2.1 Issue

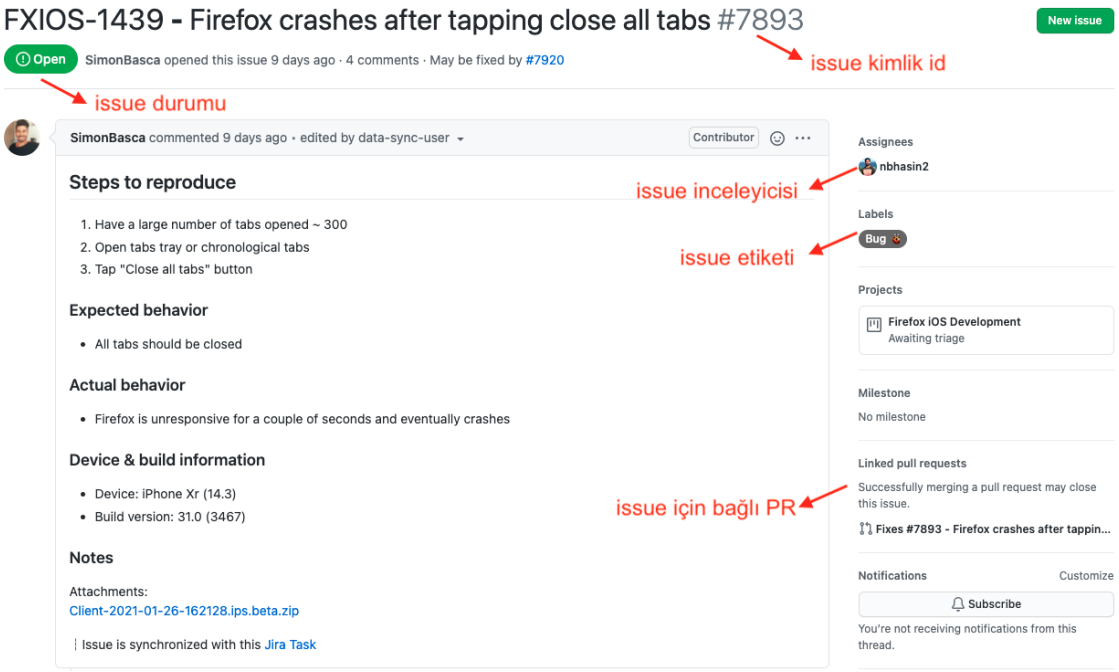
Issue, her kod deposuyla ilgili görevlerin, geliştirmelerin ve sorunların tanımlanarak ekip üyeleri ile paylaşılabildiği ve tartışılabildiği bir ortam sağlar. *Issue*, herkes tarafından oluşturulabilir (açık erişim sağlanan depolar için) ve kod deposu işbirlikçileri tarafından yönetilebilir. Her *issue* kendi tartışma bölümünü içerir. Ayrıca, bir *issue* için bir etiket (label) atanarak kategorisi (bug, enhancement, feature, vs.) tanımlanabilir, kilometre taşlarına (milestones) dönüştürülebilir, bir geliştiriciye incelemesi veya çözmesi için atanabilir.

Issue, kod geliştirme sürecinde yapılan kod bağlamaları ile ilişkilendirilerek geliştirme faaliyetlerini gözlemlemeyi sağlar. Projenin tüm versiyonlarında çözülen *issue*'lar değişim günlükleri (change log) ile takip edilebilir. Ayrıca, *issue* ile ilgili yapılan bir katkının hangi geliştirici tarafından gerçekleştirildiği ve süreç içerisinde ilgili *issue* için çözüm aşamaları da izlenebilir.

Issue'ların yönetimini kolaylaştıracak bir diğer fonksiyon da #numara (hashtag) ile tanımlanan *issue* kimlik numaralarıdır. Bu numara, katkıda bulunma sürecinde GitHub sisteminin tüm varlıkları ile ilişkilendirilerek kullanılabilir (*Commit*, *PR*, kilometretaşı, günlükler, vs.). Mozilla projesine ait bir *issue* örneği Şekil 3.6'da gösterilmiştir[1]. Henüz çözülmemiş olan (durumu açık) bu *issue* için bir kimlik numarası verilmiş, nbhasin2 adlı kullanıcıya atanmış, başlığından da anlaşılacağı üzere hata (bug) olarak etiketlenmiş, kimlik numarası ile bir *PR*'a bağlanmıştır.

Trello, Asana gibi araçlar ile birlikte çevik yazılım geliştirme yöntemi ile yürütülen projelerde ToDo (Yapılacaklar) listeleri içeren modüller kullanılmaktadır. Bu modül sayesinde önceliklendirme, tarihlendirme, kapatma, yeniden açma gibi fonksiyonlar ile *issue* yönetimi daha kolay hale gelmektedir. İş (task) listesi olarak da ifade edilebilecek bu modül üzerinden mevcut iş, bir projeyle ilişkilendirilebilir ve bu iş üzerinden yeni *issue*'lar oluşturulabilir.

[1] https://github.com/mozilla-mobile/firefox-ios/issues/7893



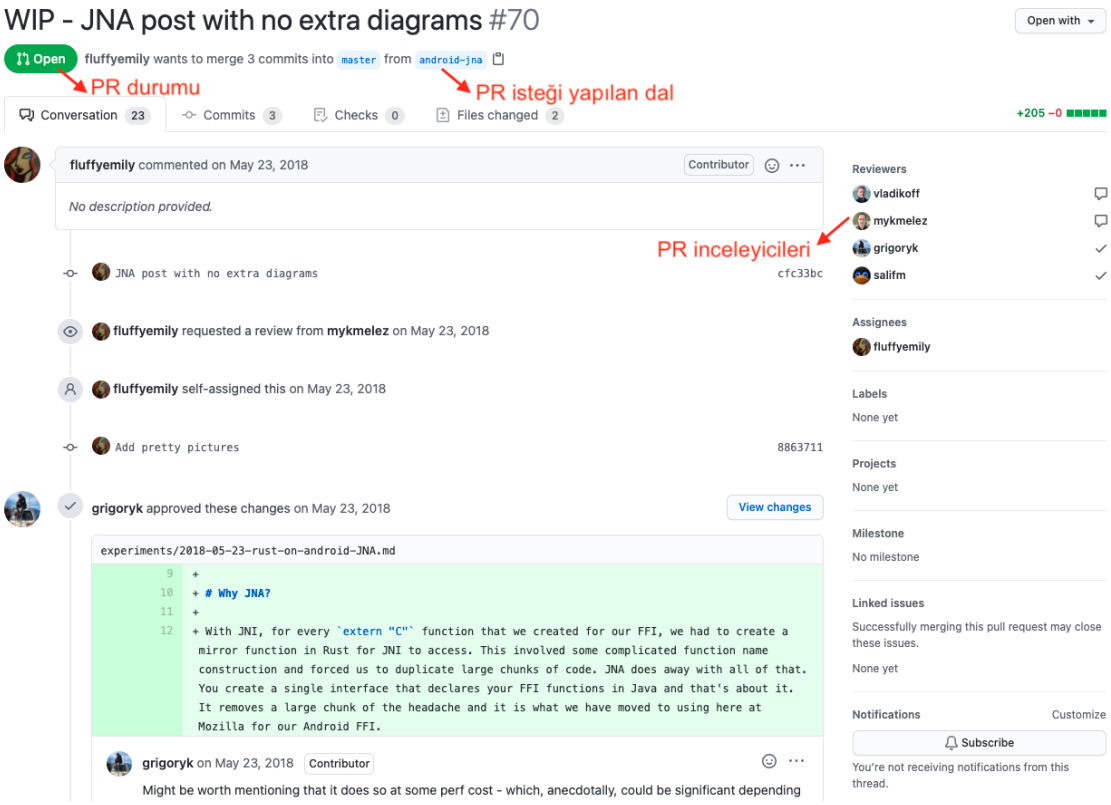
Şekil 3.6 GitHub ortamında bir *issue* örneği

3.2.2 Pull Request

Pull request (PR) (birleştirme isteği), bir kod deposundaki bir daldan, yapılan bazı değişikliklerin ana dala (genellikle master) bağlanması için gönderilen bir inceleme isteğidir. Bir *PR* ile yapılan değişiklikler işbirlikçiler tarafından tartışılabilir, etiketlenebilir, kilometre taşlarına bağlanabilir, tartışmalarda geliştiricilerden bahsedilebilir (mention), yeni geliştirmeler eklenebilir ve tüm bu değerlendirmeler sonucunda değişiklikler gözden geçirilerek ana dala bağlanabilir veya reddedilebilir. *Issue* gibi, her *PR* için de kendi tartışma bölümü vardır. Bir *PR* oluşturduktan sonra, ilgili dal ile kod deposunun ana dalı arasındaki değişikliklerin özeti sunularak bir inceleme yapılması istenir. Sunulan *PR*, ekip üyeleri (genellikle çekirdek geliştiricilerden yetki verilmiş olanlar) tarafından kabul edilirse koda bağlanabilir (merge).

Bir *PR* sonrasında, mevcut *PR*'a çalışılan daldan istenilen geliştirme faaliyetleri bir *commit*'e bağlanarak iletilebilir. Bu geliştirme faaliyetleri ilgili *PR*'da görüntülenebilir ve değişikliğe uğrayan tüm dosyalar incelenebilir. Diğer geliştiriciler, *PR* ile önerilen değişiklikleri inceleyebilir, yorumlar ekleyebilir, ve bu *PR*'a yeni *commit*'ler ekleyebilirler. Bir *issue* bir *PR*'a bağlanabilir. Böylece ilgili *PR* ile bir *issue* çözümlendiği ve kapatıldığı belirtilebilir. Mozilla Firefox projesinden bir *PR* örneği Şekil 3.7 gösterilmiştir[1]. İlgili *PR* henüz master dala birleştirilmemiştir. İnceleme yapmak üzere 4 geliştirici bu *PR* için atanmıştır.

[1]https://github.com/mozilla/firefox-browser-architecture/pull/70

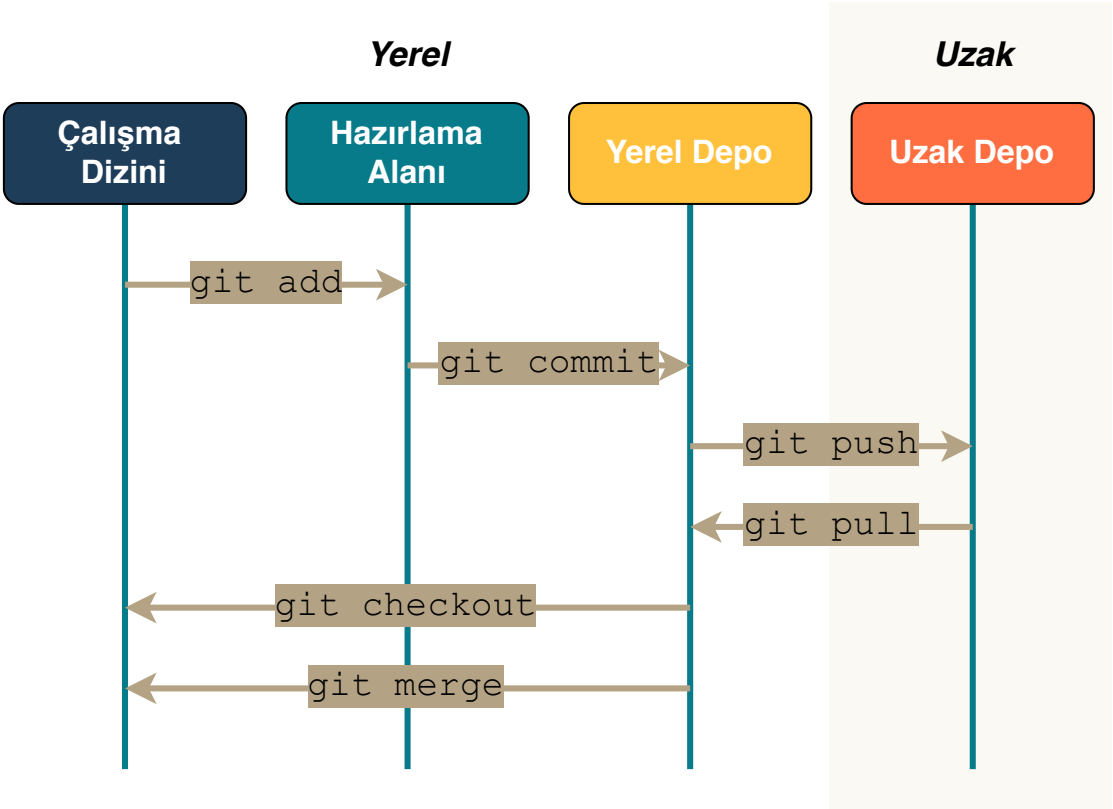


Şekil 3.7 GitHub ortamında bir *PR* örneği

3.2.3 Commit

Commit, bir dosyada (veya dosya kümesinde) yapılan bireysel bir değişikliktir. Başka bir deyişle bu değişikliği kaydetmek için ilgili depoya yapılan bir teslimat işlemidir.

Git tabanlı geliştirme yapılan ortamlarda süreçler dört temel alan üzerine inşa edilmiştir. Bu alanlar Şekil 3.8'de gösterilmiştir. Çalışma dizini (working directory) şu anda çalışılmakta olan alanı ifade etmektedir. Yerel bilgisayarda proje dosyalarının bulunduğu yerdir. Bu alan, aynı zamanda *Git*'in izlemediği alan olarak da ifade edilebilir. Dosyalardaki herhangi bir değişiklik, çalışma dizininde değiştirilmiş/eklenmiş/silinmiş olarak işaretlenir. Burada yapılan değişiklikler *Git*'e bildirilmediği (indekslenmediği) sürece yapılan değişiklikler kod depolarına yansıtılmaz. Çalışma dizinindeki dosyalarda yapılan değişiklikler *Git* protokolü tarafından bilinmektedir ancak bunlar sadece çakışma (conflict) kontrolü için çalışma dizinindeki kayıpları önlemek amacıyla kullanılır. Geliştirici *git status* isteği ile çalışma dizininde yapılan değişikliklerin durumunu ve bu değişikliklerin *Git* tarafından izlenip/izlenilmediğini kontrol edebilir. Çalışma dizininde yer alan dosyaların *Git* tarafından izlenmesi için hazırlama alanına (staging area) eklenmesi gerekir. Hazırlama alanı, *Git*'in dosyalarda meydana gelen değişiklikleri izlemeye ve kaydetmeye başladığı alandır. Kaydedilen değişiklikler *.git* dizinine yansıtılır. Hazırlama alanına bir dosya eklendikten sonra başka değişiklikler yapılırsa bu



Şekil 3.8 GitHub ortamında çalışma ağacı mantığı [170]

değişikliklerin de tekrar indekslenmesi için hazırlama alanına yeniden eklenmesi gerekir. Çalışma alanındaki bir dosyanın hazırlama alanına eklenmesi için *git add dosyaadı* komutu ile *Git*'e bildirilmesi gerekir. Dolayısıyla bu alan çalışma dizininde yapılan değişikliklerin *Git* yerel kod depolarına sunulması için hazırlandığı alan olarak ifade edilebilir.

Yerel kod deposu (local repository), *Git* sisteminin aktif dizini olarak ifade edilebilir. Yani geliştirme sürecindeki her şeyin kaydedildiği alandır. Hazırlama alanından yerel kod deposuna bir nesne taşındığında hazırlama alanındaki tüm değişiklikler bir araya getirilir ve yerel kod deposuna atanır. İşte bu süreç *commit* olarak ifade edilir. Basitçe bir *commit*, *Git*'e son kaydetmeden o ana kadar gerçekleşen tüm değişiklikleri (hazırlama alanındaki) taramasını söyleyen bir kontrol noktasıdır. Yerel kod deposuna *commit* yapıldıktan sonra hazırlama alanı temizlenecektir. Hazırlama alanında indekslenmiş nesnelerin yerel kod deposuna taşınması için *git commit –m* "*commit mesajı*" şeklinde bir bildirimde bulunmak gerekir.

Git sistemi, yapılan *commit* işlemlerini kimin ne zaman yaptığını kayıt altına almak adına, her *commit* için benzersiz bir kimlik (SHA veya hash) oluşturur. Bu kimlik üzerinden ilgili değişiklik geri alınabilir ya da değişikliğin takibi yapılabilir. *Commit* mesajında, yapılan geliştirmeleri ifade eden bir açıklama yer alabilir, yapılan işin hangi

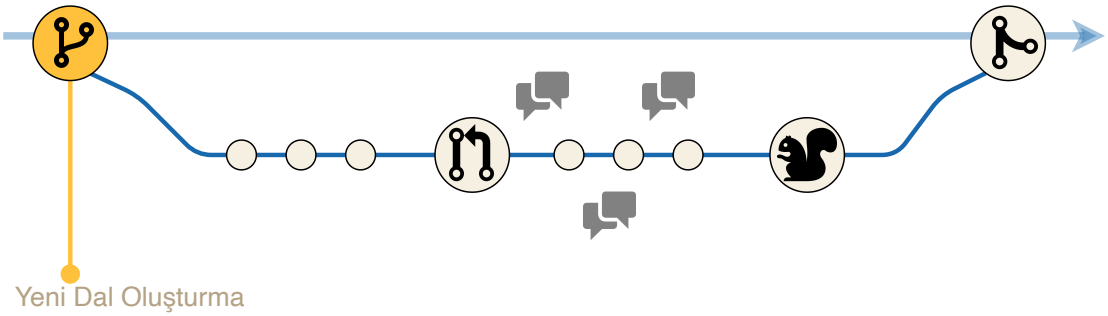
issue ile ilgili olduğunu belirtmek için bir *#issue_numarası* eklenebilir. Ayrıca başka anahtar kelimelerle (fixes, fixed, resolve, resolved, close, closes vb.) ilgili *issue* durumu tanımlanabilir veya *issue* doğrudan bir *commit* ile kapatılabilir.

Bu aşamadan sonra herhangi bir *Git* projesinde işbirliği yapılabilmesi için, uzak kod depolarıyla çalışmak gerekmektedir. Uzak kod depoları, bir projenin internet ya da yerel ağ üzerinde bir konumda barındırılan versiyonlarıdır. Yerel kod deposundaki bir *commit* uzak kod deposuna iletilirken *git push #parametreler* komutu kullanılır. Böylece yerel kod deposundaki değişikliklerin uzak sunucudaki ilgili dal üzerinden paydaşlara teslimatı yapılmış olur. Ekip üyeleri benzer şekilde uzak kod deposundaki değişiklikleri yerel kod deposuna almak için *git pull #parametreler* komutunu kullanabilir. Böylece her geliştirici için kaynak kod versiyonlanmış ve uzak kod depoları üzerinden etkileşimleri sağlanmış olur. Uzak kod depolarındaki dallar veya çatallanmış proje depoları üzerindeki değişikliklerin ana dala bağlanması için *PR* ve birleştirme adımları gerçekleştirilir.

3.3 GitHub İş Akışı

GitHub, dağıtımların düzenli olarak yapıldığı, ekipleri ve projeleri destekleyen, hafif, dal tabanlı bir iş akışına sahiptir. Bu bölümde, önceki bölümlerde de genel olarak anlatılan özellikler ve aktiviteler bir araya getirilmiş ve GitHub iş akışının nasıl çalıştığı açıklanmıştır [171].

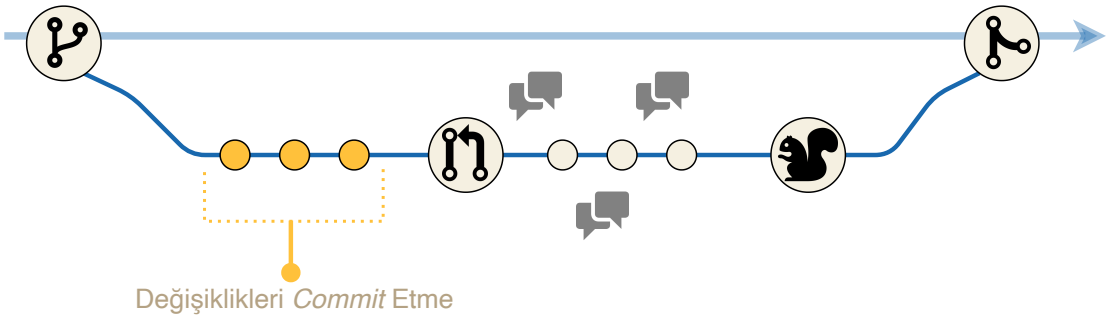
Öncelikle, bir proje üzerinde çalışırken yeni eklemeleri daha kolay ve bağımsız yönetebilmek için yeni bir dal yaratılır (Şekil 3.9). Orijinal projeyi etkilemeden değişiklikler yapmak bu sayede mümkün olacaktır.



Şekil 3.9 GitHub ortamında yeni bir dal oluşturulması [171]

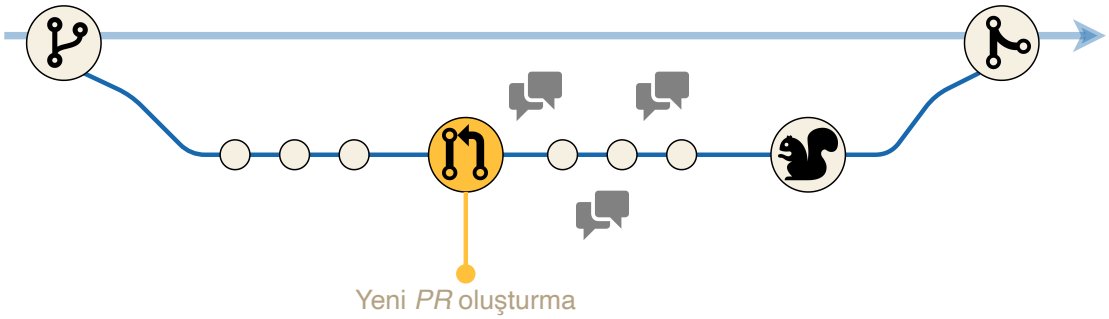
Yeni dal oluşturulduktan sonra, bir dosya ekleme, düzenleme veya silme işlemi için bir *commit* yapılır ve ilgili dala eklenir (Şekil 3.10). Bu *commit* ekleme işlemi, bir özellik dalı üzerinde çalışırken ilerlemelerin kaydını tutar. *Commit* aynı zamanda, geliştiricinin neyi/neden yaptığını başkalarının anlaması için onların görebilecekleri

şeffaf bir proje geçmişi oluşturur. Bu nedenle her *commit*, değişikliğin neden yapıldığını açıklayan bir tanımlama metni de içerir.



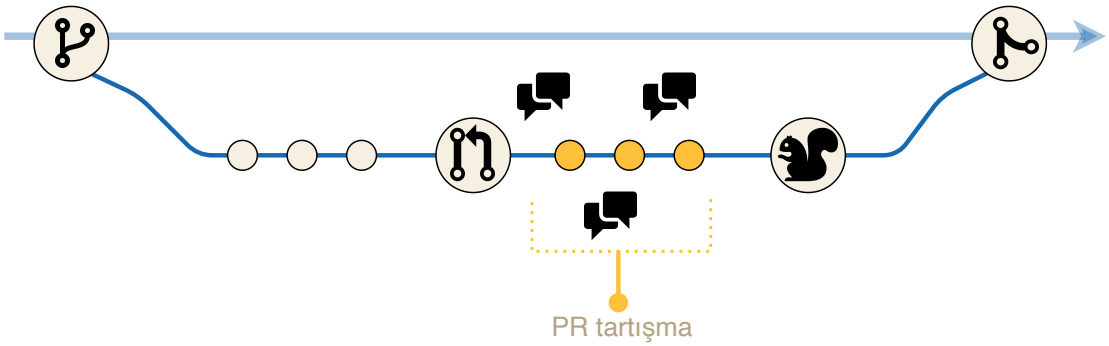
Şekil 3.10 GitHub ortamında *commit* etme işlemi [171]

Devamında, bir *PR* ile *commit* işlemleri orijinal depoya gönderilmek istenir (Şekil 3.11). Geliştirme sürecinin herhangi bir noktasında bir *PR* açılabilir.



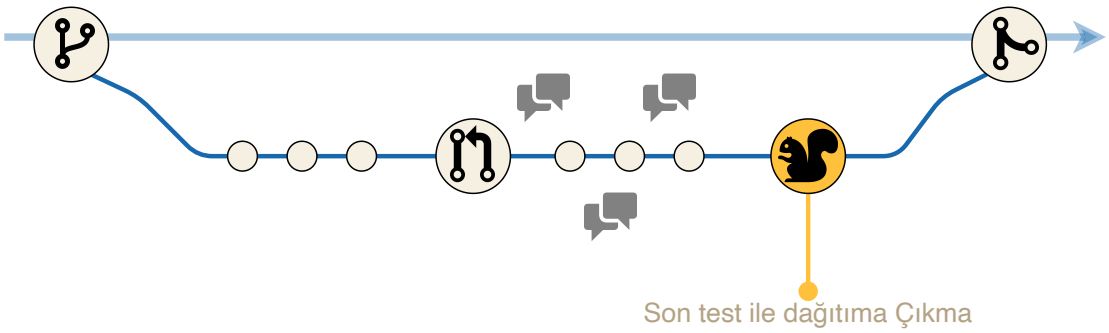
Şekil 3.11 GitHub ortamında yeni bir *PR* açılması [171]

Bir *PR* açıldığında, değişiklikleri inceleyen kişilerin veya ekibin soruları/yorumları olabilir. Kodlama stilinin proje yönergelerine uymaması, değişiklik birim testlerinin eksik olması veya herşeyin yolunda olması gibi durumlar tartışılabilir (Şekil 3.12).



Şekil 3.12 GitHub ortamında *PR*'ın tartışılması [171]

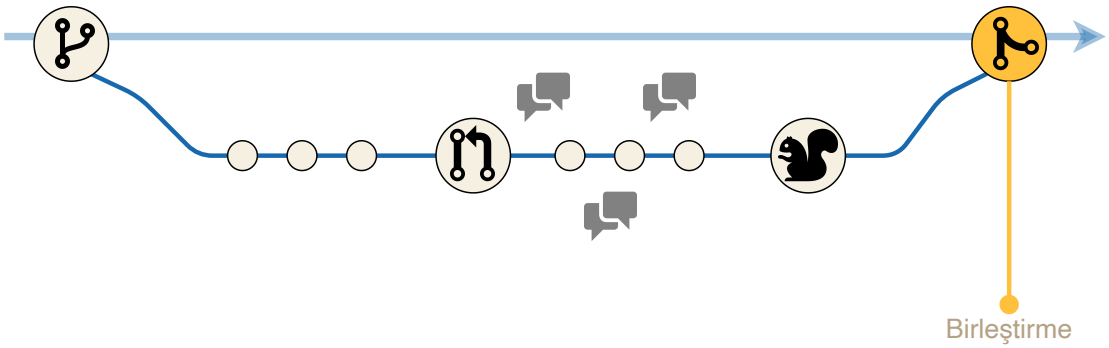
Orijinal depo ile birleştirmeden önce üretimdeki son testler için daldan dağıtım (deployment) yapılabilir (Şekil 3.13). *PR* incelendikten ve dal testlerini geçtikten



Şekil 3.13 GitHub ortamında son testler için dağıtım yapılması [171]

sonra, değişiklikler dağıtabilir. Genellikle, bir test ortamına dağıtmak en iyisidir. İş akışlarına göre bazen, doğrudan üretime dağıtmak da daha iyi bir seçim olabilir.

Son olarak değişiklikler üretimde doğrulandıktan sonra, orijinal depoya birleştirilir (Şekil 3.14). *PR* birleştirildikten sonra, kodun geçmiş değişikliklerinin bir kaydını tutar. Böylece, herkesin bir kararın neden ve nasıl verildiğini anlamasına izin verilir.



Şekil 3.14 GitHub ortamında değişikliğin birleştirilmesi [171]

4
GELİŞTİRİCİ METRİKLERİ

Geliştirici metrikleri, yazılım projelerine katkı sağlayan kullanıcıların ürettikleri kodlardan, yazdıkları yorumlara kadar yaptıkları tüm aktivitelerden çıkarılan ölçümlerdir. Yazılım projelerinin ve geliştiricilerin performanslarını değerlendirme için geliştirici metrikleri kullanılabilir. Projelerin başarı şansı, işbirliği, öğrenme becerisi bu aktivitelerden çıkarılabilir [172]. Bu aktivitelerden çıkarılan üst veri (metadata) geliştiricilerin performansları ve ünleri ile doğrudan ilişkilidir [173]. Genellikle bu metrikler dağıtık kod geliştirmenin *issue*, *commit* ve *pull request* gibi süreçleri ile ilgilidir. Bu bölümde, literatürde kullanılan ve bu 3 süreçle ilgili bazı geliştirici metrikleri ve çalışmalarda bu metrikler ile odaklanılan problemler verilmiştir.

4.1 Literatürdeki Metrikler

Pull request (PR) farklı geliştiricilerin bir projeye katkı sağlamasında anahtar rol oynar [174]. *PR* aşamasında isteğin projeye birleştirilmesi için bir inceleyici tarafından gözden geçirilmesi gerekmektedir. Bu değerlendirmenin sonucu olumlu ise, *PR* ana projeye (genellikle *master branch*) birleştirilir. *PR* değerlendirmesinin adil ve hızlı yapılması için doğru inceleyicinin bulunması önemli bir parametredir. Bu bağlamda, otomatik *PR* inceleyici ataması problemi için farklı metrikler ve yöntemler kullanılmıştır. Literatürde, bu problemin çözümü için bir projede *PR* kabul oranı, bir projede aktif kod yazan geliştiriciler [49], *PR* yapılan dosya dizini [175], *PR* göndericilerinin (*pull requesters*) kimliği, bazı sosyal özellikler [176], ve *PR* içeriğine ait metin verileri [177] gibi çok farklı metrikler kullanılmıştır.

Bir projeye aynı anda birden fazla *PR* gelebilir. Bu durumda hangisinin önce değerlendirilmesi gerektiği de önemlidir. Bir *issue* için *PR* açılması, *PR* yaşı veya *PR* yorumlarında bir kullanıcıdan bahsedilmesi (mention) gibi metrikler de *PR* önceliklendirme için kullanılmıştır [141].

Cosentino ve arkadaşları projelerin açıklığını (openness) keşfetmek için, topluluk birleştirme, kabul oranları ve işbirlikçi olma isimleriyle tanımladıkları üç geliştirici metriği önermişlerdir. Araştırmacılar, proje sahiplerinin bu metrikleri projelerinin çekiciliğini değerlendirmek için kullandıklarını vurgulamışlardır [9].

Geliştirici metriklerinin kullanıldığı bir başka problem de yazılım hata tespitidir. Yapılan bir çalışmada, hata tespiti için dosya ve *commit* seviyesi olarak gruplandırılan farklı metrikler önerilmiştir. Bir *commit*'teki düzenlenen dosya sayısı, bir *commit*'in tüm dosyaları arasında en fazla düzenlenmiş dosya adı, ilk ve son *commit* arasında geçen süre, bir geliştiricinin *commit* ettiği bir dosya üzerindeki deneyimi önemli metrikler olarak verilmiştir [178].

Yazılım ürünlerinin güvenilirliğinin açıklanabilmesi için güvenilirlik metrikleri ortaya atılmıştır [179]. Konu ile ilgili bir çalışmada, bir projede güvenilirlik ölçümü için katkı sağlayıcı (contributor) sayısı, *commit* sayısı, *commit* yapılan kod satırı sayısı, gibi metrikler kullanılmıştır. Tiwari ve arkadaşları, güvenilirlik için 1000 kod satırı başına düşen *commit* ve katkı sağlayıcı sayısı ismini verdikleri iki önemli metrik önermişlerdir [180].

Açık kaynaklı yazılım projeleri için kod sahipliğinin tespiti de önemli bir problemdir. Kod katkısına göre sahipliğin derecelendirildiği bir çalışmada, bir geliştirici tarafından düzenlenen/dokunulan dosya (touch) sayısı kullanılmıştır [181]. Bir başka çalışmada, bir dosyada değiştirilen satır sayısı (churn) bu problemi ele alırken kullanılmıştır [182]. Bu metriklerin önemini vurgulamak için Foucault, kod sahiplik metrikleri ile yazılım kalitesi arasında bir ilişki olduğunu ispatlamıştır [183].

Öneri sistemleri, yazılım mühendisliğinin dikkat çeken konularından biridir [184, 185]. Klasik öneri sistemlerinde önceden bilinen, bir kullanıcı-nesne (user-item) matrisi kullanılır. Bu matris kullanıcıların her bir nesneye verdikleri oyları (ratings) barındırmaktadır. Bu tür sistemlerde araştırmacılar kullanıcının hali hazırda verdiği oyu tahmin etmek üzerine model ve algoritmalar kurarlar [186]. GitHub gibi işbirliği platformlarında ise problem oldukça farklıdır. Bu ortamlarda, kullanıcı yerine geliştiricinin, nesne yerine ise projenin geçtiği düşünüldüğünde, geliştiricilerin projelere verdiği kesin bilinen bir oy bulunmamaktadır. Bu bağlamda, çözülmesi gereken ilk problem en doğru geliştirici-proje matrisini oluşturmaktır. İşte bu noktada farklı geliştirici metrikleri devreye girer.

Kullanıcılara proje önermenin farklı bir bakış açısıyla değerlendirildiği bir çalışmada, geliştiricilerin gelecekte hangi projelere katılacakları tahmin edilmiştir. Geliştiricilerin GitHub yaşı (kaç yıl önce hesap açtıkları), katıldıkları proje sayısı, *commit* ettikleri programlama dilleri, kaç projeye yıldız verdikleri, katıldıkları projelerdeki geliştirici sayısı, bir projedeki *commit* sayısı gibi farklı metrikler problemin çözümünde kullanılmıştır [71]. Yine bir başka çalışmada, kullanıcıların bir projeye katılmalarını etkileyen faktörler incelenmiştir. Bu çalışmada, geliştiricinin sosyal bağlantıları, diğer geliştiriciler ile ortak kodlama dilleri, aynı projelere veya bir projedeki aynı dosyalara katkı sağlama durumu gibi metrikler kullanılmıştır [70]. Liu ve diğerleri, aynı şirkette çalışma, proje sahibi ile geçmiş işbirliklerine sahip olma ve bir projeden çıkarılan zamana dayalı bazı özellikleri, tasarladıkları sinir ağı tabanlı öneri sisteminde kullanmışlardır [23]. Bir diğer çalışmada, GitHub'daki temel kullanıcı aktiviteleri proje önerisi için kullanılmıştır. Bir projeye verilecek oyu belirlemek için oluşturma-çatallama-yıldız verme (create-fork-star) aktivitelerinden faydalanılmıştır [66].

Literatürü özetlemek için, bütün bu çalışmalarda öne çıkan geliştirici metrikleri Tablo 4.1'de verilmiştir. Metrikler, kullanıldıkları yayın, ilişkili oldukları özellik, tanımları ve hedefledikleri problem ile birlikte sunulmuştur. Araştırmacılar, odaklandıkları problemi çözerken, bu metrikleri genellikle geliştiricileri geçmiş aktiviteleri aracılığıyla bir şekilde karakterize etmek için kullanırlar [187]. Geliştirici metrikleri, başta proje karakterizasyonu, otomatik *issue* veya *PR* inceleyici atama, proje önerisi olmak üzere çok çeşitli OSS geliştirme probleminin çözümü için önemli rol oynamaktadır [59, 69, 88, 188].

Bir sonraki bölümde, GitHub (tüm) ortam verilerine dayanarak, geliştirici metriği olarak kullanılabilecek aktivitelerin kullanım oranları analiz edilmiştir. Ayrıca, buradan elde edilen sonuçların doğrulanabilmesi için bir anket çalışması yapılmış, aktivitelerin gerçek yazılımcıların gözündeki değeri de irdelenmiştir. Sonuç olarak; farklı özelliklere ait önemli aktiviteler olduğu ve bu aktivitelerden elde edilecek bilgi ile farklı metrikler üretilebileceği kanısına varılmıştır.

Tablo 4.1 Literatürde kullanılmış öne çıkan geliştirici metrikleri

.	Metrik Adı	Tanımı	Hedeflediği Problem	Referans
Issue	*Maintenance Type*	Bir *issue* türü (özellik, hata, vs.)	*Issue* çözme zamanı analizi	[79]
	A_D_issue_rep_assign	Geliştiricilere atanan toplam *issue* sayısı	*Issue* kapanma oranlarını anlama	[48]
	ContainsFix	Bir *issue* için *PR* bağlanma durumu	*PR* önceliklendirme	[141]
	Owned_issues	Proje sahibinin açtığı *issue* sayısı	Projeye katılma durumu tahmini	[71]
Commit	*D_languages*	Bir geliştiricinin *commit* ettiği diller	Projeye katılma durumu tahmini	[71]
	Contibutors_SLOC	1K satır başına düşen katkı sağlayıcı sayısı	Güvenilirlik metrikleri keşfi	[180]
	TotalLoc	Son *commit* bünyesinde kod satır sayısı	Yazılım hata tespiti	[178]
	ExpLoc	Geliştiricinin düzenlediği dosyadaki deneyimi	Yazılım hata tespiti	[178]
	NumofFile	Son *commit* bünyesinde değiştirilen dosya sayısı	Yazılım hata tespiti	[178]
	Past Experience	Aynı dile ait dosyalardaki deneyim.	Projeye katılma durumu tahmini	[70]
	Readme TF_IDF	Readme ve kod dosyalarındaki *TF-IDF*	Proje önerisi	[66]
Pull Request	*NumCommits*	Bir *PR* bünyesindeki *commit* sayısı	Otomatik *PR* inceleyicisi atama	[49]
	AcceptanceRate	Bir projedeki *PR* kabul oranı	Otomatik *PR* inceleyicisi atama	[49]
	TotalLines	Bir *PR* bünyesinde toplam kod sayısı	Otomatik *PR* inceleyicisi atama	[49]
	Age	*PR* açma-kapama arasındaki zaman	*PR* önceliklendirme	[141]
	FileLocation	*PR* gönderilen dosya dizini	Otomatik *PR* inceleyicisi atama	[175]
	DecisionTime	*PR* değerlendirme süresi	Yazılım proje açıklığı	[9]
	Files_changes	Değiştirilen dosya sayısı	*PR* inceleyicisi atama faktörleri	[77]
	Comment Network	*PR* yorumlarındaki bağlantılar	*PR* süreç analizi	[69]

4.2 Aday Metriklerin Belirlenmesi

Bu bölümde, literatürden elde edilen bilgi birikimi ile GitHub ortamında geliştirici metriği olarak kullanılabilecek aktiviteler hakkında bir inceleme yapılmıştır. Öncelikle, bu aktivitelerin kullanım oranları ortaya çıkarılmış, ardından, bu aktivitelerin yazılımcıların gözündeki değerinin anlaşılabilmesi için yapılan bir anket çalışmasının sonuçları verilmiştir.

4.2.1 Özellik ve Aktivitelerin Kullanım Oranları

Açık kaynak kod geliştirme özelliklerinin analiz edilebilmesi için GHTorrent veri kümesi kullanılmıştır. GitHub ortamında kullanılabilecek metrikler, *issue*, *commit* ve *PR* özellikleri ile ilişkili aktivitelerden çıkarılmıştır. Veri kümesi, MongoDB ortamında işlenmiş ve özelliklere ait aktivite sayıları elde edilmiştir. Bu bölümde aktivitelerin kullanım sıklığını sunarken geliştirici ve proje başına her bir faaliyetin oransal değeri verilmiştir. Böylece her bir aktivitenin geliştiricilere veya projelere göre kullanım sıklığı ortaya çıkarılmıştır. Denklem 4.1'de görüldüğü üzere, ilgili aktivitenin tüm platformdaki toplam sayısı, proje ve kullanıcı sayısına bölünerek bir aktivitenin kullanma oranı hesaplanmıştır.

$$Kullanma\_Orani(aktivite\_X) = \begin{cases} \frac{\#\_aktivite\_X}{\#\_proje}, & \text{her proje için} \\ \frac{\#\_aktivite\_X}{\#\_kullanßcß}, & \text{her kullanıcı için} \end{cases} \tag{4.1}$$

4.2.1.1 *Issue* ile ilişkili aktiviteler

Issue; projelerde hataları, iyileştirmeleri veya diğer istekleri izlemek için kullanılır. Bir hatayı çözmek için veya projeye yeni bir özellik eklemek için *issue* açılabilir. Açılan *issue* tanımlanırken bir etiket (label) veya kilometre taşı (milestone) ile daha açıklayıcı hale getirilebilir. Açılan *issue* hakkında farklı kullanıcılar yorum yapabilir. Bir *issue* için inceleme yapacak veya çözümü üzerinde çalışacak bir geliştirici ona atanabilir. Bir *issue* doğrudan bir problem ile ilgili ise ona bir *PR* eklenebilir. Tüm bu aktivitelerden elde edilen bilgiler, *issue* özelliği ile ilişkili metrikleri üretmek için kullanılmaktadır. *Issue* ile ilgili aktivitelerin kullanım oranları, proje ve geliştirici başına düşen aktivite sayıları şeklinde Tablo 4.2'de verilmiştir.

Geliştiriciler genellikle sorunun tam olarak ne olduğunu bulmaya çalışırken birbirleriyle tartışma yaparlar. Bu bağlamda, en yaygın kullanılan aktivite bir *issue* hakkında yorum yapmak olarak ortaya çıkmıştır. *Issue* için etiket ekleme önemli bir özellik olmasına rağmen bu özelliğin çok kullanılmadığı görülmüştür. Aslında, bu durum, *issue* sınıflandırılması problemini ortaya çıkarmıştır [88, 112]. Projelerde bir

Tablo 4.2 *issue* ile ilişkili aktivitelerin kullanım oranları

ISSUE	Kullanıcı başına	Proje başına
***issue* yorum sayısı**	**5,94**	**1,53**
issue açma sayısı	3,37	0,87
issue kapatma sayısı	1,20	0,31
PR bağlı *issue* sayısı	1,01	0,26
issue etiket sayısı	0,93	0,24
issue inceleme sayısı	0,45	0,11

issue'nun takibi ve çözümü için bir inceleyici atanır (assignee). Sonuçlara bakıldığında, atanmış bir geliştirici tarafından *issue* inceleme aktivitesi az kullanıldığından dolayı, otomatik *issue* inceleyici ataması problemi de literatürde çalışılan bir konu olmuştur [188].

4.2.1.2 *Commit* ile ilişkili aktiviteler

Commit; bir projede yapılan kod katkılarını ilgili depo içine kaydetmektir. Kullanıcılar yazdıkları kodları *commit* ile projeye eklerler, kodları yazmaya yazarlık (authoring), ilgili *commit*'i projeye eklemeye işleme (committing) adı verilir. Ayrıca, kullanıcılar bir *commit* hakkında yorum yazarak da katkı sağlayabilirler. *Commit* özelliğine ait farklı aktivitelerin oransal değerleri Tablo 4.3'te verilmiştir.

Tablo 4.3 *Commit* ile ilişkili aktivitelerin kullanım oranları

COMMIT	Kullanıcı başına	Proje başına
***commit* etme sayısı**	**7,97**	**2,05**
commit yazarlık sayısı	5,95	1,53
commit yorum sayısı	0,38	0,09

Literatürde, *commit* özelliğinden, bu aktivitelerin haricinde, kodlama ile ilgili parametreler (satır sayısı, dosya sayısı, kodlama dili, vs.) üzerinden daha fazla metrik üretildiği görülmüştür. Ayrıca, Tablo 4.3 ışığında, geliştiricilerin bir *commit* üzerine nadiren tartıştıkları görülmektedir. Bu durum GitHub ortamındaki geliştiricilerin ekstrem programlamanın eşli programlama mantığını kullanmadıklarını da göstermektedir.

4.2.1.3 *PR* ile ilişkili aktiviteler

GitHub'da *PR*, yapılan katkıları depo sahiplerine gönderme işlemidir. Bu durumda, bir *PR* öncelikle projenin bir çekirdek üyesi tarafından *PR* gözden geçirilme (review) sürecine tabi tutulur. Daha sonra, *PR* kabul edilirse, değiştirilen kod projeye birleştirilir (merge). *PR* özelliğine ait aktiviteler Tablo 4.4'te verilmiştir.

Tablo 4.4 *PR* ile ilişkili aktivitelerin kullanım oranları

PULL REQUEST	Kullanıcı başına	Proje başına
***PR* sayısı**	**1,60**	**0,41**
PR yorum sayısı	0,92	0,23
PR birleştirme sayısı	0,64	0,16
PR inceleyici sayısı	0,04	0,01

En çok kullanılan aktivite *PR* oluşturma olarak görülmektedir. Ayrıca, açılan *PR* sayısının sadece yaklaşık 1/3'ünün kabul edildiği de görülmektedir. Bu durum, yapılan çoğu katkının projelere kabul edilmediğini göstermektedir. Bunun bir sebebi gelen değişiklik taleplerinin doğru kişiler tarafından incelenmemesi olabilir. Bu bağlamda, *PR* incelemenin önemli bir aktivite olduğu düşünülmektedir. Öyle ki literatür incelendiğinde, *PR* ile ilgili zorlukların genellikle *PR* inceleme hakkında olduğu görülmüştür [68, 69].

4.2.2 Anket Çalışması

Bölüm 4.2.1'de verilen ve geliştirici metriği olarak kullanılabilecek bazı aktivitelerin, gerçek yazılım dünyasındaki karşılığının araştırılması için bir anket çalışması yapılmıştır. Bu anket ile, ilgili aktivitelerin yazılımcılar bakımından önemi ortaya konulmuştur. Anket aşağıdaki 3 araştırma sorusu kapsamında geliştirilmiştir;

1. Geliştiriciler hangi OSS platformlarını kullanıyor?
2. Geliştiriciler için OSS platformlarında yaptıkları en önemli aktivite nedir?
3. Geliştiriciler için kod yazmak kadar yorum yapmak (tartışmak) da önemli midir?

4.2.2.1 Amaç ve Tasarım

Anket çalışması Google forms üzerinden yapılmıştır. Anket, Türkiye'de farklı şirketlerde çalışan yazılımcılara doğrudan e-posta yoluyla ve aktif yüzlerce üyesi olan İstanbulCoders adında bir topluluğun e-posta grubuna gönderilmiştir. Anketin hedef kitlesi açık kaynak yazılım platformları ile uygulama geliştiren veya bu ortamlardaki

projeleri kullanan, yazılım sektöründe aktif çalışan geliştiricilerdir. Ankete, farklı şirketlerde ve pozisyonlarda çalışan, farklı derecede deneyime sahip 130 geliştiriciden cevap alınmıştır.

Ankette önce katılımcılara işleri ile ilgili bazı ön bilgileri toplayan 7 soru sorulmuştur. Ardından, geliştirici metrikleri hakkında biri açık uçlu diğerleri çoktan seçmeli 6 soru daha sorulmuştur. Açık uçlu soruya 70 katılımcı cevap vermiş, diğer soruları tüm katılımcılar yanıtlamıştır.

A. İş hakkında sorular

   (1) Çalıştığınız firmadaki pozisyonunuz nedir?

   (2) Yazılım sektöründeki deneyiminiz ne kadardır?

   (3) Çalıştığınız firmadaki toplam yazılımcı sayısı nedir?

   (4) OSS geliştirme platformlarından hangisini/hangilerini kullanıyorsunuz? (GitHub, GitLab, Bitbucket, vs.)

   (5) OSS geliştirme topluluklarında yer aldınız mı?

   (6) Açık kaynak bir proje/yazılım kullandınız mı?

   (7) Açık kaynak bir proje/yazılım geliştirdiniz mi?

B. Kod geliştirme özellikleri hakkında sorular

   (1) Bir geliştiricinin aşağıdaki aktivitelerden hangisini yapıyor olması onun aktiviteyi yaptığı proje ile **ilgisini** daha çok gösterir?

   (2) Bir geliştiricinin aşağıdaki aktivitelerden hangisini yapıyor olması onun aktiviteyi yaptığı projede **yetkin** (projeye hakim) olduğunu daha çok gösterir?

   (3) *Issue* ile ilgili aşağıdakilerden hangisi daha önemli bir aktivitedir?

   (4) *Commit* ile ilgili aşağıdakilerden hangisi daha önemli bir aktivitedir?

   (5) *PR* ile ilgili aşağıdakilerden hangisi daha önemli bir aktivitedir?

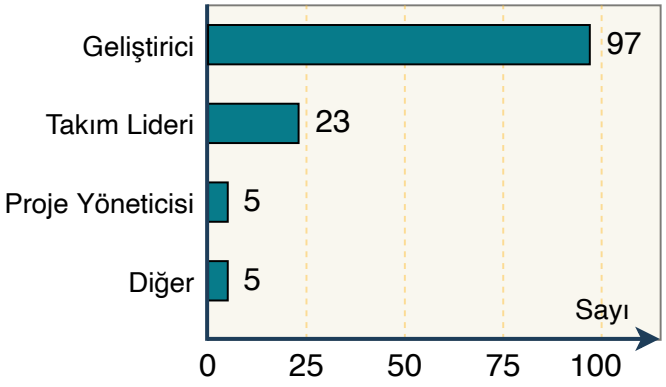
   (6) Yukarıdaki özellikler bağlamında bir projede yorum ile katkı yapmanın, o projede kod geliştirmek kadar önemli olduğunu düşünüyor musunuz? Neden?

4.2.2.2 Anket yapılan hedef kitle hakkında bilgiler

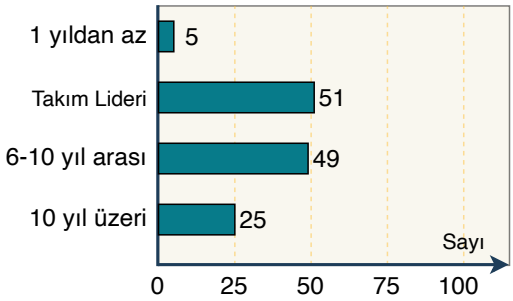
Ankete 130 farklı yazılımcı katılmıştır. Katılımcılar farklı seviyede deneyime sahip, çeşitli pozisyonlarda çalışan ve farklı büyüklükteki şirketlerde çalışan yazılımcılardır.

4.2.2.3 Alınan cevapların değerlendirilmesi

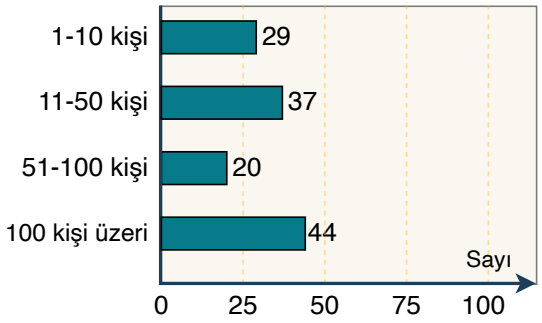
Katılımcılara ilk olarak yazılım sektöründeki deneyimleri ve çalıştıkları firmalarındaki pozisyonları ve çalışan sayıları sorulmuştur (Şekil 4.1).



(a) Pozisyon



(b) Deneyim

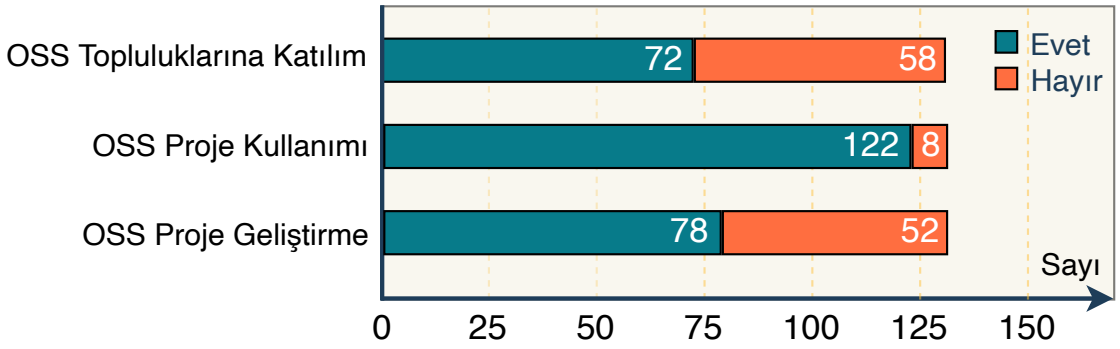


(c) Büyüklük

Şekil 4.1 Katılımcılar hakkında bazı bilgiler

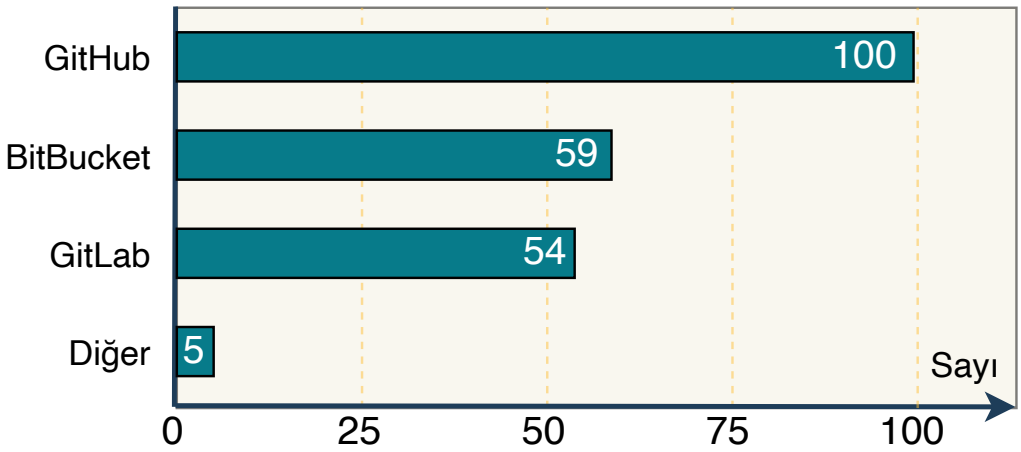
Bunun sonucunda, ankete katılanların 97'si yazılım geliştirici, 23'ü takım lideri, 5'i proje yöneticisi, kalan 5 kişi ise bunların dışında bir pozisyonda olduklarını belirtmiştir (Şekil 4.1a). Böylece, ankette çeşitli rollerden geliştiricilerin fikirleri alınarak, her bakış açısından gelen cevaplar değerlendirilmiştir. Katılımcıların yazılım sektöründeki deneyimlerine verdikleri cevaba göre ise; yeni başlayan 5, 1-5 yıl arasında 51, 5-10 yıl arasında 49, 10 yıldan fazladır sektörde olanların sayısı 25 kişi olarak ortaya çıkmıştır (Şekil 4.1b). Şirket büyüklükleri ile ilgili verilen cevaplarda ise; çoğunluğun 100'den fazla çalışanı olan şirketlerde çalıştığı ortaya çıkmıştır (Şekil 4.1c).

Katılımcıların OSS konusunda birikimini ölçmek için sorulan sorulara verdikleri cevaplara göre büyük çoğunluğun konuyla ilişkili olduğu görülmüştür (Şekil 4.2). Katılımcıların yarısından fazlasının OSS topluluklarında yer aldığı, %93'ünün OSS kullandığı (122 kişi), %60'ının OSS geliştirdiği öğrenilmiştir. Sonuçlara bakıldığında, katılımcıların konuya olan ilgileri görülmüş, verilen cevapların kayda değer olacağı düşünülmüştür.



Şekil 4.2 Katılımcıların OSS hakkında birikimleri

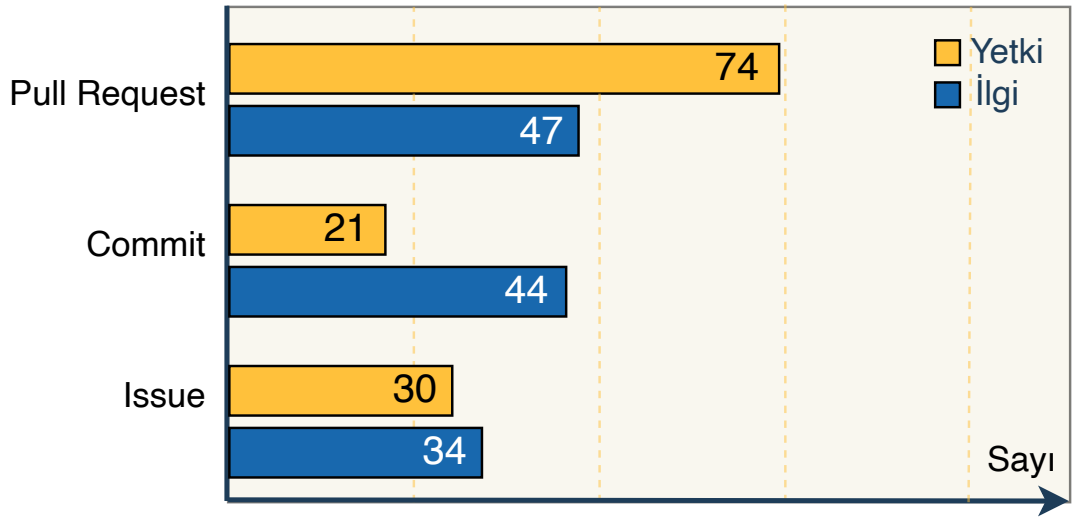
Bir sonraki soruda, kullanıcıların OSS hakkında önceki sorulara verdikleri cevapların doğruluğunu teyit etmek ve hangi platformların kimler tarafından tercih edildiğini görmek için, katılımcılara kullandıkları veya proje geliştirdikleri OSS platformları da sorulmuştur. Alınan cevaplara göre en çok GitHub platformunun kullanıldığı ortaya çıkmıştır (Şekil 4.3). Çoğu kullanıcı GitHub yanında diğer platformları da kullandıklarını belirtmiştir. 50'den fazla geliştiriciye sahip şirketlerde çalışan 64 katılımcıdan 28'inin GitHub yanında Gitlab'a yöneldiği de görülmüştür. Bu durum, şirketlerin maliyet kaynaklı problemlerden dolayı ücretsiz bir araç olan ve şirketlerin yerel ortamlarında (on-premise) kullanabildikleri GitLab ortamına yöneldiklerini göstermektedir.



Şekil 4.3 Katılımcıların kod geliştirme için kullandıkları platformlar

OSS ile alakalı 3 sorunun hepsine "Hayır" cevabı veren ve araç kullanım sorusunu boş bırakan veya soruya ilgisiz yanıtlar veren 5 kişinin bilgileri anketten çıkarılmıştır. Bu çalışmanın hedef kitlesi OSS konusunda tecrübesi olan geliştiricilerdir. Bu bağlamda, OSS kullanmayan, geliştirmeyen, iş birliği platformları kullanmayan kişilerin sonraki bölümdeki cevaplarının dikkate alınmamasının daha doğru olacağı kanısına varılmıştır.

Bu bölümde, katılımcılara bir projede önemli olduğunu düşündüğümüz 3 farklı özelliğe ait aktivitelerden hangisinin daha önemli olduğu sorulmuştur (Şekil 4.4). Sorulan iki sorudan birinde, yapılan aktivitenin geliştiricinin o projeye olan ilgisi açısından, diğerinde ise o projedeki yetkinliği açısından önemi sorulmuştur. Gelen cevaplarda projedeki yetkinliği *PR* ile alakalı aktiviteler (%60) ile ölçülebileceği sonucu açık bir şekilde ortaya çıkmıştır. Proje ile ilgilendiğinin göstergesi olarak verilen sonuçlar daha bulanık olmakla birlikte yine *PR* öne çıkmaktadır. Bu sonuç, özellikle değerlendirici atamalarında literatürde *PR* ile ilişkili aktivitelerden üretilen geliştirici metriklerinin kullanılmasını da desteklemektedir.

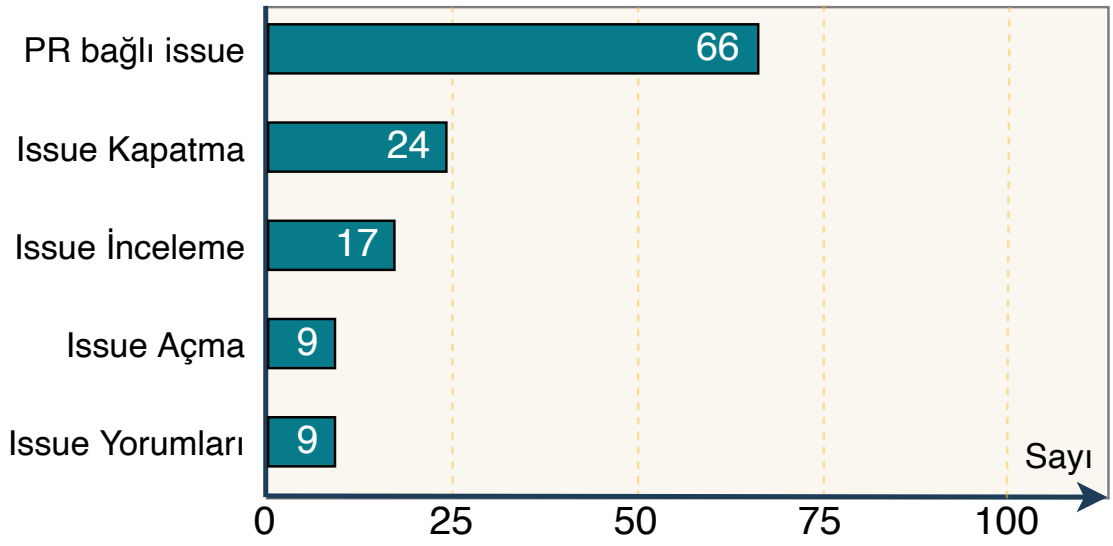


Şekil 4.4 Katılımcılar için yetki ve ilgi bağlamında önemli olan özellikler

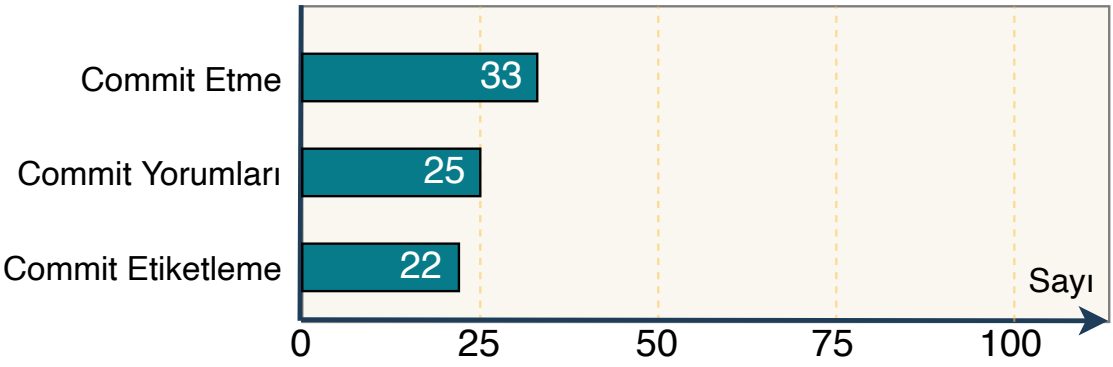
Katılımcılara *issue* özelliği ile ilgili en önemli aktivitenin sorulduğu soruda, katılımcılar bir *issue* için bir *PR* bağlanmasının en önemli aktivite olduğunu söylemişlerdir (Şekil 4.5). Bu aktivite, ilgili *issue*'nun projeye gerçek bir katkısı olduğunun göstergesidir. Geliştirici-proje ilişkisinin ölçülmesinde, geliştiricinin açtığı kaç *issue*'nun bir *PR*'a bağlandığının sayısı önemli bir metrik olarak kullanılmaktadır [141]. Bu durum, katılımcıların verdiği cevabı da açıkça desteklemektedir.

Commit ile ilgili aktivitelerde aslında doğrudan kullanılan tek aktivitenin *commit* etmek olduğu görülmektedir (Şekil 4.6). *Commit* yorumları da katılımcıların yaklaşık %30'u tarafından öne çıkarılmıştır. Bunun yanında *commit* etiketlemek de bir diğer önemli aktivite olarak verilebilir.

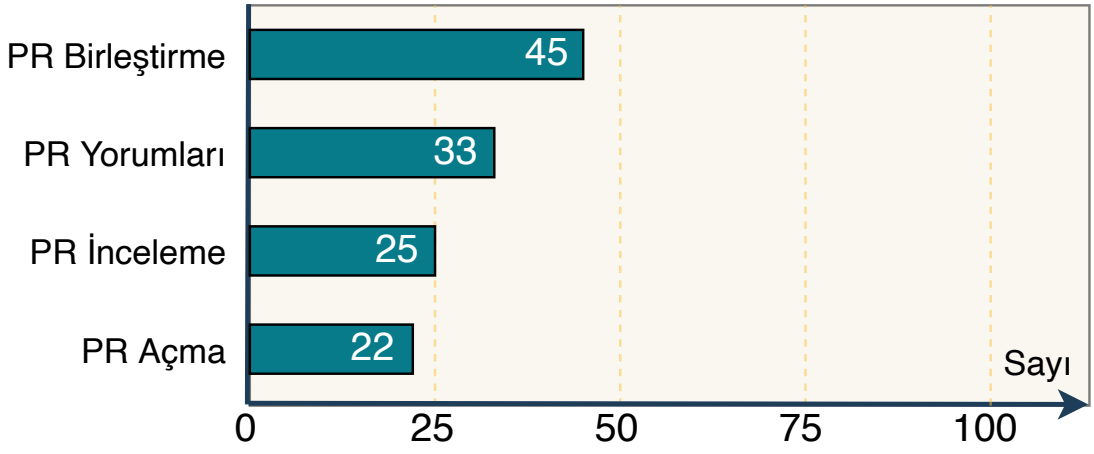
PR aktivitelerinde kullanım oranları ile uyuşmayan sonuçlar ortaya çıkmıştır (Şekil 4.7). En çok kullanılan aktivite *PR* açmak iken, katılımcılar *PR* birleştirmeyi en önemli aktivite olarak görmüşlerdir. Bu bağlamda, bu aktivite, bir geliştiricinin projeye olan yetkinliğini de ölçmek için kullanılabilir. Geliştirici ne kadar projeye hakim ise gelen PR'ları birleştirme yetkisine ve birikimine de o denli sahiptir.



Şekil 4.5 Katılımcılar için *issue* ile ilişkili aktivitelerin önemi



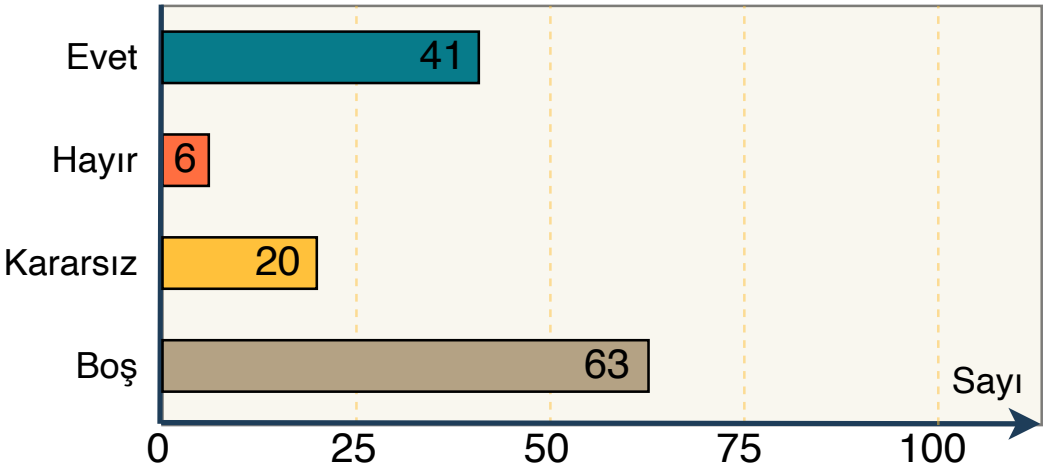
Şekil 4.6 Katılımcılar için *commit* ile ilişkili aktivitelerin önemi



Şekil 4.7 Katılımcılar için *PR* ile ilişkili aktivitelerin önemi

Son olarak, geliştiricilere bir proje için kod yazmakla, yorum yapmanın aynı derecede öneme sahip olup olmadığı sorulmuştur. Açık uçlu olarak sorulan bu soruya verilen cevaplar 4 kategoride birleştirilmiştir. Soruya cevap verenlerin

%62'si yorum yazmanın da kod yazmak kadar önemli olduğunu açıkça belirtmiştir (Şekil 4.8). Verilen olumlu cevaplarda genel kanı, yorum yapmanın hem projeyi anlamak konusunda fayda sağladığı hem de yapılan yorumlar (tartışma) sayesinde projenin hatalarının ve eksik özelliklerinin daha kolay ortaya çıkarıldığı şeklindedir. Olumsuz cevaplarda ise kod yazmanın daha kritik olduğu ve daha uzman geliştiriciler tarafından yapıldığı, bu sebeple yorum yazmanın çok da önemli olmadığı görüşü savunulmuştur.



Şekil 4.8 Katılımcılar için projelerde yorum yapmanın önemi

Sonuç olarak en yoğun kullanılan aktiviteler *issue* yorum eklemek ve *commit* etmek olarak belirlenmiştir. Kullanım oranlarının aksine, *PR* özelliğinin ve onunla ilişkili aktivitelerin yazılımcıların gözünde daha önemli olduğu ortaya çıkmıştır. Bu bölümde yapılan araştırmalar ve anket çalışması bir konferans bildirisi olarak yayınlanmıştır.

4.3 Önerilen Geliştirici Metrikleri

Tez kapsamında, geniş bir yelpazedeki yazılım mühendisliği problemlerinin çözümü için 40 farklı geliştirici metriği önerilmiştir. Bu metrikler GitDataSCP veri kümesi kullanılarak elde edilmiş, yazılım mühendisliğindeki önemli bir problem olarak görülen proje öneri problemi üzerinde test edilmiştir.

Metrikler elde edilirken geliştiricilerin proje başına yaptıkları aktiviteler göz önüne alınmıştır. Yapılan aktivite sayısı, bazı aktivite sayıları arasındaki oran ve bir aktivite sadece var olup olmadığının durumu metrik oluştururken kullanılmıştır. İlk olarak, aktiviteler ayrı ayrı değerlendirilmiş ve tekil metrik oluşturulmuştur. Ardından tekil metriklerin değerleri bazı ortak özelliklere göre toplanmış ve füzyon metrikler üretilmiştir. Son olarak, bir projede bir aktivitenin sadece varlığı dikkate alınarak ikili füzyon metrikler oluşturulmuştur.

4.3.1 Tekil Geliştirici Metrikleri

Geliştiricilerin projelerin çeşitli süreçlerindeki aktiviteleri birer metrik olarak ele alınmıştır. Aktivite, her türlü yorumu, kod katkılarını, incelemeleri, vs. tüm faaliyetleri içermektedir. Önerilen metriklerin tanımları Tablo 4.5'te verilmiştir. Metrikler bir geliştiricinin bir projede yaptığı aktiviteleri göstermektedir. Örneğin, *issue_opened* metriği bir geliştiricinin ilgili projedeki açtığı toplam *issue* sayısını ifade etmektedir.

Tablo 4.5 Tekil geliştirici metrikleri tanımları

	Metrik Adı	Tanımı
1	issue_opened	*issue* açma sayısı
2	issue_commented	*issue* yorum sayısı
3	issue_closed	*issue* kapatma sayısı
4	issue_has*PR*	*PR* ile bağlanmış *issue* sayısı
5	issue_assigned	Tarafına atanmış *issue* sayısı
6	commit_commented	*commit* yorum sayısı
7	commit_authored	*commit* yazarlık yapma sayısı
8	commit_committed	*commit* işlemi sayısı
9	pr_opened	*PR* açma sayısı
10	pr_merged	*PR* birleştirme sayısı
11	pr_assigned	Tarafına atanmış *PR* sayısı
12	pr_commented	*PR* yorum sayısı

Bu metriklere ek olarak, tekil metrik değerlerinin optimize edilmiş versiyonları[1] da oluşturulmuştur. Burada, geliştiricinin ilgili projedeki oransal katkısını bulmak

[1]Bu metrikler 'O_' ön eki ile sunulmuştur. Örneğin, *issue_closed* metriğinin optimize versiyonu, *O_issue_closed* olarak verilmiştir.

hedeflenmiştir. Örneğin, Deniz adlı geliştirici A ve B projelerinde 10'ar adet *issue* kapatmış olsun. A projesindeki toplam kapatılan *issue* sayısı 100 iken, B projesinde bu sayı 1000 olarak verilsin. Bu durumda, Deniz oransal olarak daha çok *issue* kapattığı için, A projesinde (%10 katkı) B'ye (%1 katkı) kıyasla daha çok katkı sağladığı düşünülebilir. Sonuç olarak, Deniz'in A projesindeki *issue_closed* değeri 10 iken, *O_issue_closed* değeri 0,01 olarak hesaplanacaktır. Optimize metrikler hesaplanırken Denklem 4.2 kullanılmıştır. Denklem 4.2'de k_i i. kullanıcıyı, p_n ise n. projeyi temsil etmektedir.

$$O\_metrikX_{p_n}^{k_i} = \frac{\#\_metrikX_{p_n}^{k_i}}{\#\_toplam\_metrikX_{p_n}} \tag{4.2}$$

4.3.2 Füzyon Geliştirici Metrikleri

Tekil metrikler bazı özelliklerine ve aktivite türlerine göre bir araya getirilebilir. Kod ile alakalı tekil metrikler birlikte, projenin tüm süreçlerinde yapılan yorumlar bir arada düşünülebilir. Bu bakış açısından yola çıkılarak belirli tekil metrik değerleri toplanmış, aşağıda tanımları verilen füzyon geliştirici metrikleri oluşturulmuştur.

1. *code_contributions:* Kod katkısı ile ilgili tüm metriklerin toplamından oluşturulmuştur.

$$code\_contributions = pr\_opened + issue\_opened + issue\_hasPR + pr\_merged + commit\_committed \tag{4.3}$$

2. *comments:* Yorum ile ilgili tüm metriklerin toplamından oluşturulmuştur.

$$comments = issue\_commented + commit\_commented + pr\_commented \tag{4.4}$$

3. *issue_related: Issue* ile ilgili tüm metriklerin toplamından oluşturulmuştur.

$$issue\_related = issue\_opened + issue\_hasPR + issue\_commented + issue\_assigned \tag{4.5}$$

4. *pr_related PR* ile ilgili tüm metriklerin toplamından oluşturulmuştur.

$$pr\_related = pr\_opened + pr\_merged + pr\_commented + pr\_assigned$$

(4.6)

5. *commit_related: Commit* ile ilgili tüm metriklerin toplamından oluşturulmuştur.

$$commit\_related = commit\_commented + commit\_authored + commit\_committed \tag{4.7}$$

6. *commit2comment: Commit* sayısı ile *commit* yorumu sayısına oranından oluşturulmuştur.

$$commit2comment = \frac{commit\_committed}{commit\_commented} \tag{4.8}$$

7. *issue2comment:* Açılan *issue* ile *issue* yorumu sayısına oranından oluşturulmuştur.

$$issue2comment = \frac{issue\_opened}{issue\_commented} \tag{4.9}$$

8. *pr2comment: PR* sayısı ile *PR* yorumu sayısına oranından oluşturulmuştur.

$$pr2comment = \frac{pr\_opened}{pr\_commented} \tag{4.10}$$

9. *code2comment:* Toplam kodlama aktiviteleri ile yorumların oranından oluşturulmuştur.

$$code2comment = \frac{code\_contributions}{comments} \tag{4.11}$$

4.3.3 İkili Füzyon Geliştirici Metrikleri

Bölüm 4.3.1'de verilen metriklerde, bir aktivitenin kaç kez yapıldığı bilgisi kullanılmıştır. Örneğin, Deniz adlı geliştirici, X projesinde 18 *issue* açmış ise, Deniz'in X projesi için oyu 18 olacaktır. Bu metriklere alternatif olarak, basit bir şekilde bir aktivitenin sadece var olup olmadığını sorgulayan bir dizi metrik daha önerilmiştir. Örneğin, Deniz, X projesinde tek bir *issue* açmış olsa bile, Deniz'in X'e oyu 1, aksi durumda 0 olacaktır. Böylece, tekil geliştirici metriklerinden Denklem 4.12 ile ikili

geliştirici metrikleri elde edilmiştir.

$$ikili\_metrik = \begin{cases} 1, & tekil\_metrik > 0 \text{ ise} \\ 0, & tekil\_metrik = 0 \text{ ise} \end{cases} \tag{4.12}$$

İkili geliştici metrikleri sadece 0 ve 1 değerleri içerdiğinden, onları doğrudan kullanmak anlamsız sonuçlar vermiştir. Bu yüzden, onların yerine Bölüm 4.3.2'de uygulandığı gibi birleştirilmiş versiyonları metrik olarak önerilmiştir. İkili füzyon geliştirici metrikler[1] ikili geliştirici metriklerinin toplamlarından elde edilmiştir.

Tüm bu metriklerin yanına, bir karşılaştırma daha yapmak için Sun ve diğerlerinin proje öneri yayınında sunulan başka bir metrik de eklenmiştir [189]. İlgili çalışmada, geliştirici-proje oy değeri *oluşturma-çatallama-yıldız verme (create-fork-star)* aktiviteleri ile hesaplanmıştır. Aktivitelerin varlık durumuna bakarak, bir projeyi oluşturmak 10, çatallamak 5, yıldız vermek 3 puan olarak belirlenmiştir. Bir projede birden fazla aktivite bulunursa en yüksek puan, projeye ilgili geliştirici tarafından verilmiş oy olarak değerlendirilmiştir. Tez kapsamında da bu duruma benzer aktiviteler aynı puanlama mantığı ile hesaplanmış *Sun_metric* adı verilen metrik diğerlerinin arasına eklenmiştir.

[1] Bu metrikler 'binary_' ön eki ile sunulmuştur. Örneğin, *comments* metriğinin ikili versiyonu *binary_comments* olarak verilmiştir.

5
VERİ KÜMESİ

Bu bölümde öncelikle, aday metriklerin keşfinde kullanılan GHTorrent veri kümesi tanıtılmıştır. Ardından, farklı yazılım mühendisliği problemlerinde ortak olarak kullanılabilecek, tez kapsamında üretilen ve kullanılan, GitDataSCP adı verilen açık kaynak veri kümesi tanıtılmıştır. GitDataSCP veri kümesinin oluşturulma aşamaları, içeriği ve paylaşımı hakkında ayrıntılar verilmiştir.

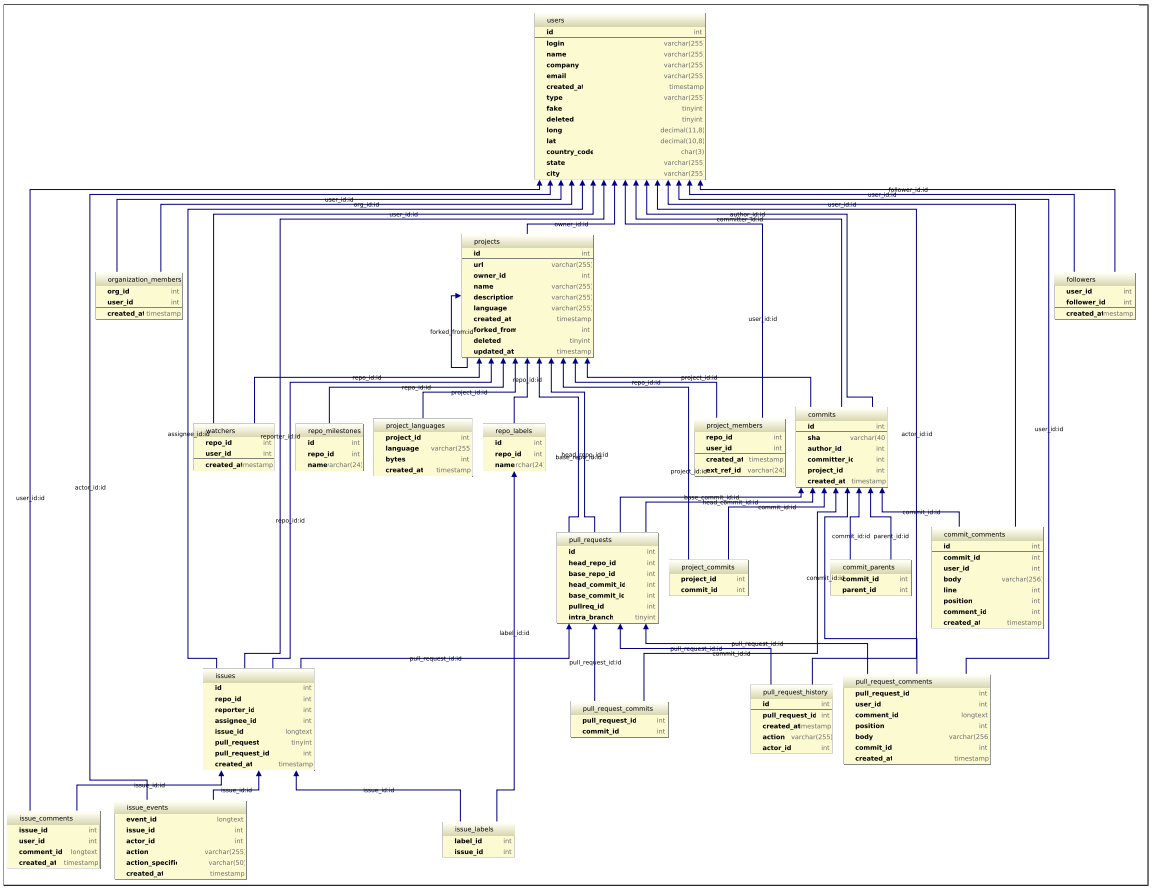
5.1 Veri Kümesi: GHTorrent

Bulut tabanlı açık kaynak kod depolama ve versiyonlama sistemleri arasında en yaygın olanlardan biri GitHub'tır. GitHub, 100 milyondan fazla geliştiriciyi ve 40 milyondan fazla depoyu barındıran dünyanın en büyük kod sunucusudur[1]. GitHub ve benzeri platformlarda süreçler dağıtık olarak yönetilir. Bu sayede, çok farklı lokasyondan, birbirinden bağımsız birçok geliştirici aynı projelere katkı sağlayabilirler. Bu katkılar; kod yazma, projelerdeki sorunları ortaya çıkarma veya düzeltme, projenin farklı süreçlerinde yorum ve tartışmalar üretme yoluyla yapılabilmektedir. İşte tüm bu aktiviteler, araştırmacılar için, çok farklı yazılım mühendisliği problemleri için kullanılabilecek büyük miktarda veri anlamına gelmektedir. Bu aktivitelerin dışında, geliştiricilerin veya projelerin kendi aralarındaki bağlantıları, ilişkileri ve farklılıkları da ortamın sosyal verilerini oluşturmaktadır. GitHub platformunun verisi büyüklüğü ve çeşitliliği ile birçok çalışma için veri kaynağı olarak kullanılmaktadır.

GitHub verileriyle yapılan çalışmalarda, araştırmacılar verileri GitHub API aracılığıyla çekerler veya bazı hazır veri kümelerini kullanırlar. Bir tarama çalışmasında GitHub veri kümeleri arasında en çok kullanılan veri kümesi GHTorrent olarak verilmiştir [3]. Diğer benzer çalışmalarda da bu veri kümesinin farklı kriterlere göre öncü olduğu belirtilmiştir [4–6]. GHTorrent veri kümesi, Georgios Gousios ve ekibi tarafından Delft Üniversitesi, Yazılım Mühendisliği Bölümü'nde geliştirilmiştir [8]. Veri kümesi, GitHub API ile sistematik olarak toplanmış, platformdaki tüm açık projeler ve kullanıcılar

[1] https://en.wikipedia.org/wiki/GitHub

hakkında çeşitli bilgileri içermektedir. GHTorrent ilişkisel 26 tabloda bu bilgileri araştırmacılar için sunmaktadır (Şekil 5.1)[1]. Veri kümesi, platformdaki tüm verileri indirilebilir dökümler (dump) halinde barındırmaktadır. Kullanıcılar için *Users*, projeler için *Repos* adı verilen tabloları, kalan tablolarda ise farklı aktivite ve özellikleri barındıran veri kümesi, platforma ait çoğu bilgiyi içermektedir.



Şekil 5.1 GHTorrent veri kümesine ait ilişkisel şema

5.1.1 GHTorrent için Detaylı Bir İnceleme

Github verilerini hızlı ve tutarlı bir şekilde elde etmenin en mantıklı yolu GitHub API kullanmaktadır. Ancak, API ile veri çekerken saatlik 5000 istek sınırı olması büyük verilerin çekilmesindeki en önemli kısıtlamadır. Buna karşılık, GHTorrent indirebilir dökümleri sayesinde kısıtlama olmaksızın istenilen tarihe ait verilerin tamamını elde etmeye imkan verir. Aynı şekilde, GHTorrent web sitesindeki yönergeleri takip ederek güncel GitHub verisine MongoDB ve MySQL sunucuları aracılığı ile erişmek mümkündür.

GHTorrent verileri esnek biçimde servis etmektedir. Veriler, hem MongoDB veritabanı ile ham JSON formatta hem de MySQL veritabanı ile ilişkisel tablolarda sunulmaktadır.

[1]Büyük boyut için bu ***linki*** kullanabilirsiniz.

MongoDB formatı için:

a. İki aylık frekanslarla hazırlanmış MongoDB koleksiyonlar arşivi indirilebilir. (2015'e kadar ayrı koleksiyon dosyaları halinde).

b. Web sitesindeki yönerge ile uzak MongoDB sunucusuna bağlanılarak istenilen veri indirilebilir[1]

MySQL formatı için:

a. Tüm ilişkisel tablolar toplu bir MySQL dökümü halinde indirilebilir. (2015'e kadar).

b. Tüm ilişkisel tablolar, her biri ayrı ayrı CSV dosyaları halinde indirilebilir. (2015'ten sonrası için)

c. En güncel MySQL veritabanı için GHTorrent web sitesinde kullanılan online SQLite aracı kullanılabilir.

Bu farklı seçeneklere rağmen, son yıllarda yapılan çalışmalar da dahil olmak üzere, çoğu çalışmanın veri kümesinin eski versiyonlarını kullandığı görülmüştür. Tablo 5.1'de son iki yılda (2018-2019) veri kümesini kullanan çalışmaların hangi versiyondan faydalandıkları verilmiştir. İncelenen 55 çalışmadan 25 tanesi açıkça hangi versiyonu kullandıklarını belirtmemişlerdir. Diğerlerinin çoğunun yapıldığı yıla göre çok daha önceki yıllara ait veri kümesi versiyonlarını kullandıkları görülmüştür.

Tablo 5.1 Son iki yıla ait çalışmalarda kullanılan GHTorrent versiyonları

GHTorrent Versiyon Yılı	**Çalışmanın Yılı**	
	2019	**2018**
2019	1	0
2018	4	1
2017	5	4
2016 ve öncesi	5	5
Bilinmeyen	6	19

[1] Uzak MongoDB sunucusu güncel verileri vermiyor olabilir.

GHTorrent veri kümesinin güncel versiyonlarını elde edebilmek için iki seçenek bulunmaktadır.

1. Çok büyük boyuttaki CSV türündeki dosyaları indirip, yerel veri tabanına import etme
2. Günlük MongoDB depolarını indirip, web sitesinde belirtilen yönerge ile yerel ortamdaki koleksiyonları güncelleme

Bu seçeneklerin ikisi için de çok yüksek boyutlardaki dosyaların aktarılması ve işlenmesi, çok fazla çaba sarf etmeyi gerektirmekte ve ciddi zaman kaybına sebep olmaktadır [4]. Bu bağlamda, araştırmacıların -eski ve daha küçük boyutlu olan- önceki versiyonları tercih ettikleri düşünülmektedir. Her ne kadar eski verilerin kullanılması bir problem teşkil etmese de, veri kümesinin daha kolay elde edilebilir güncel versiyonları olmasının, GitHub platformunun sürekli gelişen ve değişen yapısı göz önüne alındığında, önemli olduğu düşünülmektedir.

GHTorrent veri kümesi getirdiği kolaylıkların ve avantajların yanı sıra bazı problemlere ve eksilere de sahiptir. Tez kapsamında deneyimlenen ve incelenen bazı çalışmalarda raporlanan problemler aşağıda verilmiştir.

- GHTorrent çoğu koleksiyonunda tekrarlayan (duplicated) kayıtlar içermektedir [87, 135]. Bu durum, *Repos* ve *Users* koleksiyonlarında *id* değeri aynı olan kayıtlardan fark edilmiştir.
- Veri kümesinde koleksiyonlar arasında bağlayıcı alan olarak kullanılacak bazı alanların eksik olduğu görülmüştür. Örneğin, *CommitComments* ve *Commits* koleksiyonlarında depo bilgisi verecek *id* veya *fullname* alanları bulunmamaktadır. Bu bilgi ancak *url* alanı parse edilerek üretilebilir.
- Bir başka çalışmada, GHTorrent veri kümesinin geliştirici hesaplarının GitHub'daki ekiplerin üyesi olup olmadığına dair doğru veri sağlamadığı bildirilmiştir [65].
- Sun ve arkadaşları çalışmalarında, GHTorrent'in hangi dosyayı kimin düzenlediği bilgisini içermediğini belirtmişlerdir [66].

Veri kümesi ile ilgili gözden kaçan bir diğer husus daha olduğu düşünülmektedir. Veri kümesini kullanan çalışmaların veri seçme kriterleri incelenmiştir. Veri görselleştirme veya veri genişletme çalışmaları haricinde, neredeyse tüm çalışmalarda, veri seçmek

için veri kümesine farklı filtreler uygulanmıştır. Filtreler genellikle proje veya kullanıcı özellikleri ile ilişkili eşik değerleri olarak seçilmiştir (*commit* sayısı, takipçi sayısı, kodlama dili, vs.). Bu yüzden, odaklanılan problemler aynı olsa bile, kullanılan veri kümeleri her çalışmaya özgü olduğundan, çalışmalarda iddia edilen başarı oranlarının doğruluğu tartışmalı hale gelmektedir.

Tablo 5.2'de verilen çalışmalarda *PR* ile ilgili problemlere odaklanılmıştır. Tüm çalışmalarda GHTorrent veri kümesi kullanılmasına rağmen, hepsinde farklı filtreler ile veri kümesinin farklı büyüklükteki alt kümeleri kullanılmıştır. Bu durum, ortak bir veri kümesi üretme zorunluğunu da beraberinde getirmiştir.

Tablo 5.2 Veri kümesinde veri seçmek için kullanılan farklı filtreler

Yayın ID	Depo Sayısı	Depo Seçim Filtresi
[125]	3587	100'den fazla PR içerenler
[141]	475	—
[161]	40	Çatallanmamış ve 100'den fazla PR içerenler
[11]	333	Aylık en az 5 PR içerenler
[77]	3	—
[49]	32	En az 5 geliştiricisi ve 100'den fazla PR içerenler
[145]	74	Java ile geliştirilenler

5.1.2 GHTorrent ile Üretilen Bazı Veri Kümeleri

Literatürde incelenen çalışmaların bazılarında veri kümesi bir problemin çözümü için veri kaynağı olarak kullanılmamıştır. Sadece veri kümesi kullanımının anlatıldığı, GHTorrent'in bazı özellikleri seçilerek alt veri kümeleri oluşturulduğu, GHTorrent'e eklemeler yapılarak genişletilmiş veri kümelerinin üretildiği çalışmaların olduğu görülmüştür. Veri kümesini bu amaçlarla kullanan çalışmalar, 4 alt başlıkta Tablo 5.3'te verilmiştir.

Tablo 5.3 Veri kümesi odaklı çalışmalar

Yayın ID	Adet	DEFI	SUBS	AUGM	HELP
[4, 174, 190, 191]	4	x			
[89, 90, 101, 104, 110, 135, 192]	4		x		
[18, 20, 137, 193]	7			x	
[194–206]	13				x

Tanım-Kullanım (DEFI): Bu başlık, veri kümesinin tanıtımını yapan çalışmaları ve GHTorrent'in nasıl kullanılacağı ile ilgili ipuçları, yönergeleri ve araçları sunan çalışmaları içermektedir.

Alt Veri Kümeleri (SUBS): GHTorrent veri kümesinden bazı özellikler veya veriler seçilerek alt veri kümeleri oluşturan çalışmalara bu başlıkta yer verilmiştir. Genellikle proje ve geliştirici bazlı filtreler ile bazı özel problemler için kullanılabilecek ManySStuBs4J, GE852, 50K-C gibi alt veri kümeleri oluşturulmuştur.

Artırım-Türev (AUGM): GHTorrent veri kümesini taban olarak kullanıp, farklı ortamlardaki veriler ile birleştirme yoluyla genişletilmiş veri kümeleri üreten çalışmalar bu başlıkta toplanmıştır. Sürekli entegrasyon bilgileri ile TravisTorrent, Stackoverflow sitesindeki bilgiler ile SOTorrent gibi yeni genişletilmiş veri kümeleri üretilmiştir.

Yardımcı (HELP): Bu başlık altındaki çalışmalar ise çok farklı konulara odaklanan, GHTorrent'den kullanıcılara veya projelere ait, sadece birkaç bilgiyi (kullanıcı maili, proje dili vs.) almayı hedeflemiştir. Son olarak, GHTorrent veri kümesi ile oluşturulan bazı araç ve çerçeveler Tablo 5.4'te verilmiştir.

Tablo 5.4 GHTorrent ile üretilmiş özel amaçlı depolar ve çerçeveler

GitHub Depo Adı	**Amacı**
OSSHealth/ghdata	Augur: Sürdürülebilirlik ve OSS sağlığı için veri toplama aracı
TestRoots/travistorrent-tools	TravisTorrent için veri üretme aracı
RepoReapers/reaper	Depo skoru hesaplama aracı
SOM-Research/Gitana	GitHub proje aktiviteleri için SQL-tabanlı bir araç
DevMine/ght2dm	ght2dm: GHTorrent'i DevMine'a aktarmak için bir araç
valeriocos/selective-importer-4-ghtorrent	GHTorrent tablo seçme aracı

5.2 Önerilen Veri Kümesi: GitDataSCP

Bölüm 5.1.1'de belirtildiği gibi her araştırmacı kendi oluşturduğu veri kümesi üzerinde çalıştığı için benzer konulardaki çalışmaların bile başarı oranlarının karşılaştırılması bir hayli zor olmaktadır. Tez kapsamında, literatürdeki bu sorunu gidermek için 2015 yılına kadar GitHub verilerini içeren GHTorrent veri kümesi versiyonunun bir kopyası yerel ortama aktarılmış ve belirli kriterlere göre daraltılarak açık kaynaklı bir veri kümesi olan GitDataSCP (GitHub Dataset for Software Collaboration Platforms) oluşturulmuştur [207]. Veri kümesi, tasarlanan bir web sayfasında hem CSV formatındaki koleksiyonlar halinde, hem de bir MongoDB arşivi olarak paylaşıma açılmıştır[1]. Veri kümesi oluşturulurken, GHTorrent'in Bölüm 5.1.1'de verilen problemlerini gidermek için tekrarlı verilerin çıkarılması, bağlantı alanları eklenmesi, gibi bazı ön işlemler uygulanmıştır. Veri kümesine uygulanan bu işlemlerde kullanılan sorgular kadirseker00/GitDataSCP isimli GitHub deposunda[2] paylaşılmıştır.

Veri kümesi kullanıcı, proje, yazılım geliştirme gibi birçok başlıktaki yazılım mühendisliği problemleri üzerinde çalışma imkanı sunacak içerikte veriler barındırmaktadır.

5.2.1 GHTorrent Veri Kümesinin İndirilmesi

Veri kümesinin elde edilmesi için, ilk olarak GHTorrent web sitesinde verilen uzak MongoDB sunucusuna bağlanma işlemi denenmiştir. Bu sunucuya bağlanmak için web sitesinde verilen yönerge uygulanmıştır.

- SSH anahtar üretilmesi
- GHTorrent GitHub sayfasındaki bir dosyaya açık anahtar (public key) bilgisi eklenerek, bir pull request işlemi yapılması
- Bu dosyaya tarafımızca eklenen anahtarın birleştirilmesi (merge)
- Terminal ortamından sunucu ile bağlantı sağlanılması
- Gelen cevaba SSH anahtar bilgisinin iletilmesi
- Yeni bir terminal ortamından MongoDB veritabanına bağlantı yapılması
- Veri çekme işleminin yapılması

[1] http://ceng1.cumhuriyet.edu.tr/datasets/githubdata/
[2] *https://github.com/kadirseker00/GitDataSCP*

Örnek bazı sorgular şu şekilde verilebilir.

- *Repos* koleksiyonundaki ilk 10 dokümanı indirme;

```
mongoexport --collection = repos
            --limit = 10
            --db = github
            --out = repo10.json
```

- *Users* koleksiyonundan 3'ten fazla takipçisi olan ilk 10 dokümanı indirme;

```
mongoexport -d = github
            -c = users
            -q = '{ "followers": { "$gte": 3 } }'
            --limit = 10
            --out = user10fol3.json
```

Her ne kadar bu sorgular başarılı bir şekilde çalışsa da, çekilen veri boyutu büyüdükçe problemler çıkmaya başlamıştır. Sorguların çok uzun sürmesi, bağlantı kopma problemleri, büyük veya daha karmaşık sorguların terminal ortamında yazılamaması ve sorgular için maksimum süre sınırı olması gibi problemler sebebiyle bu yöntem ile veri kümesinin alınması mümkün olmamıştır.

Bu başarısız denememenin ardından uzak bağlantı yerine eski sürümlerin doğrudan indirilmesine karar verilmiştir. Bu aşamada önce MySQL ortamı için veriler indirilmiş ancak, büyük boyutlardaki tabloların yerel ortama aktarılmasında (import) sorunlar çıkmıştır.

Bunun ardından, son seçenek olarak 2015 yılına ait MongoDB veritabanı koleksiyonları indirilerek (yaklaşık 750 GB'lık ham veri), veri kümesi yerel ortama aktarılmıştır. Bu aktarma işlemi için Studio 3t[1] programının 1 yıllık lisansı alınmış ve işlemlere bu program üzerinden devam edilmiştir.

5.2.2 Ön işlemler

GHTorrent veri kümesi elde edildikten sonra, veri seçmek ve hataları düzeltmek için veri kümesine bazı ön işlemler uygulanmıştır.

[1] https://studio3t.com/

5.2.2.1 Verilerin Filtrelenmesi

Veri kümesi oluşturulurken, platformdaki en aktif 100 kullanıcı[1] etkinlikleri baz alınmıştır (Şekil 5.2). Bu kullanıcılar seçilerek *sUsers* koleksiyonu oluşturulmuştur. Kullanıcılardan sonra depoların[2] da daraltılması gerekmektedir. Bu işlem için öncelikle, bu kullanıcıların sahip olduğu depolar seçilmiştir.

```
db.getCollection('Repos').aggregate([
    {$lookup:
        {
            from: "sUsers", localField: "owner.id",
            foreignField : "id", as: "fromUsers"
        }
    },
    {$match: {"fromUsers.0": {$exists: true}}},
    {$project: {fromUsers: 0}},
    {$out: "sReposOwned"}
]);
```

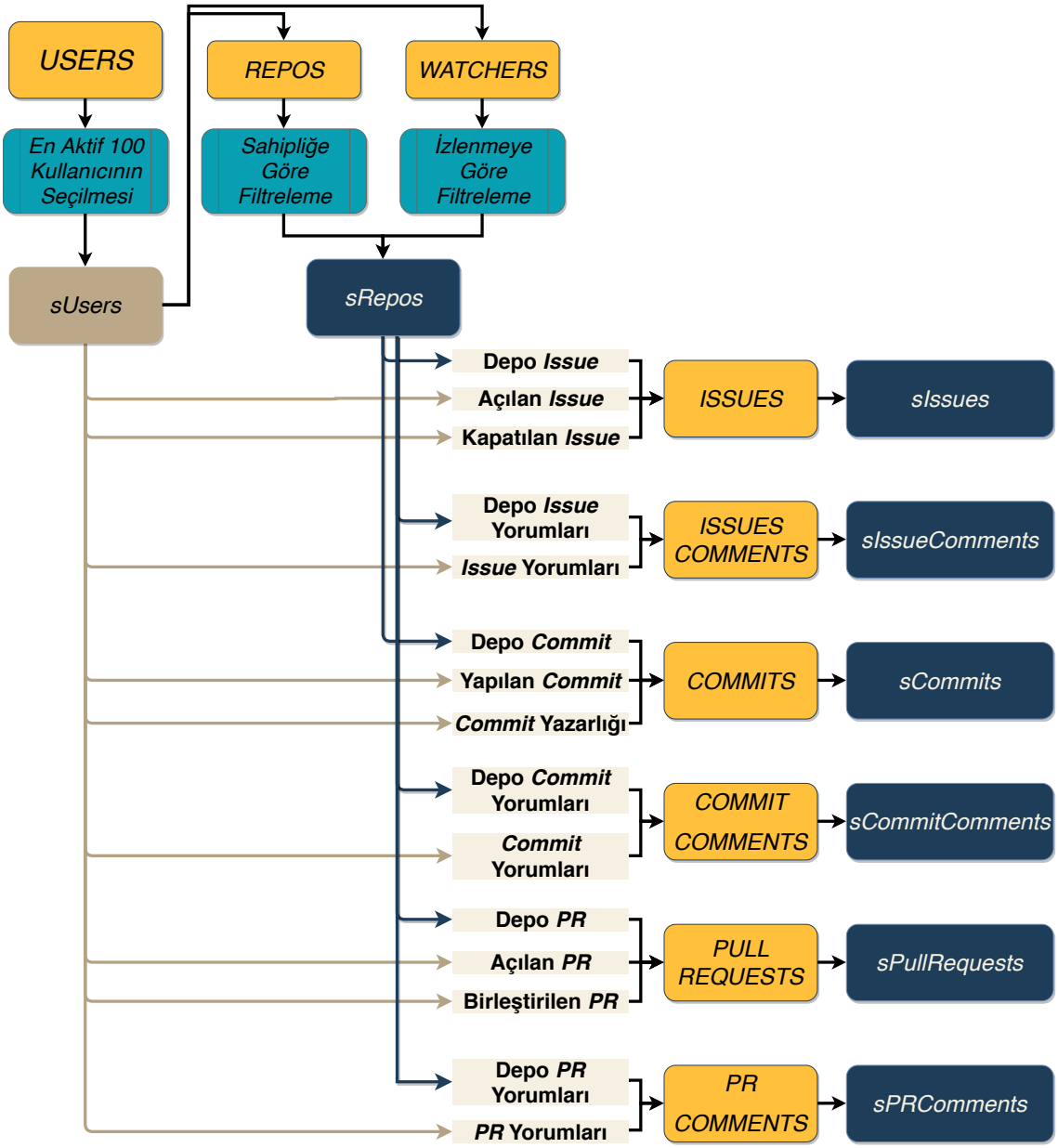
Ardından, ilgili kullanıcıların izlediği depolar elde edilmiş, oluşturulan *sReposOwned* ve *sReposWatched* koleksiyonları birleştirilerek *sRepos* koleksiyonu oluşturulmuştur.

```
db.getCollection('Watchers').aggregate([
    {$lookup:
        {
            from: "sUsers", localField: "user.id",
            foreignField : "id", as: "fromUsers"
        }
    },
    {$match: {"fromUsers.0": {$exists: true}}},
    {$project: {fromUsers: 0}},
    {$out: "sReposWatched"}
]);
```

Daha sonra *commit*, *issue* ve *pull request (PR)* özelliklerine ait koleksiyonlar *sUsers* ve *sRepos* koleksiyonlarına göre filtrelenmiş ve daraltılmış koleksiyonlar oluşturulmuştur.

[1] https://gist.github.com/paulmillr/2657075

[2] Tez kapsamında, GitHub *repository* terimi için *depo* veya *repo* kelimesi kullanılmıştır.



Şekil 5.2 GitDataSCP veri kümesinin oluşturulması

5.2.2.2 Hatalı veya Eksik Verilerin Düzeltilmesi

Veri kümesinin hatasız olarak üretilmesi için bazı işlemlerden geçirilmesi gerekmektedir. Bu bölümde, her bir sorunun çözümünde kullanılan ilgili örnek MongoDB sorguları da problem ile birlikte verilmiştir. Veri kümesinde seçilen veriler kullanıcı bazında planlandığından, tüm koleksiyonlar için birincil anahtar görevini üstlenecek alanların belirlenmesi gerekmektedir. Bu bağlamda, birincil anahtar (veya ayırt edici özellik) olarak kullanıcılarla ilgili verilerde *user_id*, depolarla ilişkili verilerde de benzer şekilde *repo_id* kullanılacaktır. Bu nedenle, GHTorrent veri kümesinde, bu alanlarla ilgili mevcut sorunların giderilmesi gerekmektedir.

İlk olarak, dokümanlardaki ilgili *id* alanı, *null* değere sahip veya hiç bulunmayan kayıtlar silinmiştir.

```
db.Pull_requests.deleteMany({user_id: null});
```

Veri kümesinin —kullanıcılar tarafından üretildiği veya GitHub'dan yanlışlıkla çekildiği düşünülen— tekrarlı kayıtlar içerdiği fark edilmiştir. *user_id*, *repo_id* veya *full_name* gibi anahtar alanlarda tekrarlanan veriler içeren kayıtlar da silinmiştir.

```
db.Issues.aggregate([
    {$match: {"issue_id": {$nin: [null]}}},
    {$group:
        {"_id": "$issue_id", "doc": {"$first": "$$ROOT"}}
    },
    {$replaceRoot: {"newRoot": "$doc"}},
    {$out: "Issues"}
    ],
{allowDiskUse: true}
);
```

Bazı koleksiyonlarda, diğer koleksiyonlar ile ilişki kurmak için kullanılacak anahtar alanlar olmadığı fark edilmiştir. Bu alanlar farklı yollarla elde edilmiştir.

1. *Issues* ve *Watchers* koleksiyonlarında *repo_id* veya *full_name* alanları bulunmamaktadır. Burada *full_name* alanı; depo adı ve depo sahip isminin birleştirilmesi ile üretilmiştir.

    ```
    db.Watchers.aggregate([
        {$addFields:
            {full_name:
                {$concat: ["$owner", "/", "$repo"]}
            }
        },
        {$out:"Watchers"}
    ]);
    ```

2. *CommitComments* koleksiyonunda *full_name* oluşturmak için depo ve sahip adı alanları da bulunmamaktadır. Bu koleksiyon için *full_name* alanı depo *url*[1] alanı parse edilerek üretilmiştir.

[1] Örnek depo url; https://github.com/johndue/projectX

```
db.CommentCommits.aggregate([
    {$addFields:
    {full_name:
        {$concat: [
            {$arrayElemAt: [{$split: ["$url", "/"]}, 4]},
            "/",
            {$arrayElemAt: [{$split: ["$url", "/"]}, 5]}
        ]}
    }},
    {$out: "CommentCommits"}
]);
```

3. Bu aşamada A ve B adımlarında elde edilen *full_name* alanı kullanılarak *repo_id* alanının koleksiyonlara eklenmesi işlemi gerçekleştirilmiştir.

```
db.CommitComments.aggregate([
    {$lookup:
        {from: "Repos", let: {item: "$full_name"},
            pipeline: [
            {$match: {$expr:
                        {$eq: ["$full_name","$$item"]}
                    }
            },
            {$project: {"repo_id": 1}}
                ],
            as: "fromComments"}
    },
    {$replaceRoot:
        {newRoot:
            {$mergeObjects:
                [{$arrayElemAt:
                    ["$fromComments",0]},"$$ROOT"]
            }
        }
    },
    {$project: {'fromComments': 0}},
    {$out: "CommitComments"}
]);
```

4. Benzer şekilde, *user_id* olmayan ilgili koleksiyonlara da bu alan eklenmiştir.

Bu aşamadan sonra veri seçme işlemi gerçekleştirilmiştir. Seçilmiş depo ve kullanıcılarla ilişkili veriler her bir koleksiyon için ayrı ayrı elde edilmiştir. Örneğin, *sUsers* koleksiyonundaki kullanıcıların açtığı veya kapattığı *issue* kayıtları ve *sRepos* koleksiyonundaki depolara ait *issue* kayıtları birleştirilmiş ve *sIssues* koleksiyonu oluşturulmuştur.

Kullanıcılar tarafından açılan *issue* bulunması;

```
db.getCollection('Issues').aggregate([
    {$lookup:
        {
            from: "sUsers", localField: "user.id",
            foreignField : "id", as: "fromUsers"
        }
    },
    {$match: {"fromUsers.0": {$exists: true}}},
    {$project: {fromUsers: 0}},
    {$out: "IssuesOpened"}
])
```

Kullanıcılar tarafından kapatılan *issue* bulunması;

```
db.getCollection('Issues').aggregate([
    {$lookup:
        {
            from: "sUsers", localField: "closed_by.id",
            foreignField : "id", as: "fromUsers"
        }
    },
    {$match: {"fromUsers.0": {$exists: true}}},
    {$project: {fromUsers: 0}},
    {$out: "IssuesClosed"}
])
```

İlgili depolarda açılan *issue* bulunması;

```
db.getCollection('Issues').aggregate([
    {$lookup:
        {
            from: "sRepos", localField: "repo_id",
            foreignField : "id", as: "fromRepos"
        }
```

```
    },
    {$match: {"fromRepos.0": {$exists: true}}},
    {$project: {fromRepos: 0}},
    {$out: "IssuesRepos"}
])
```

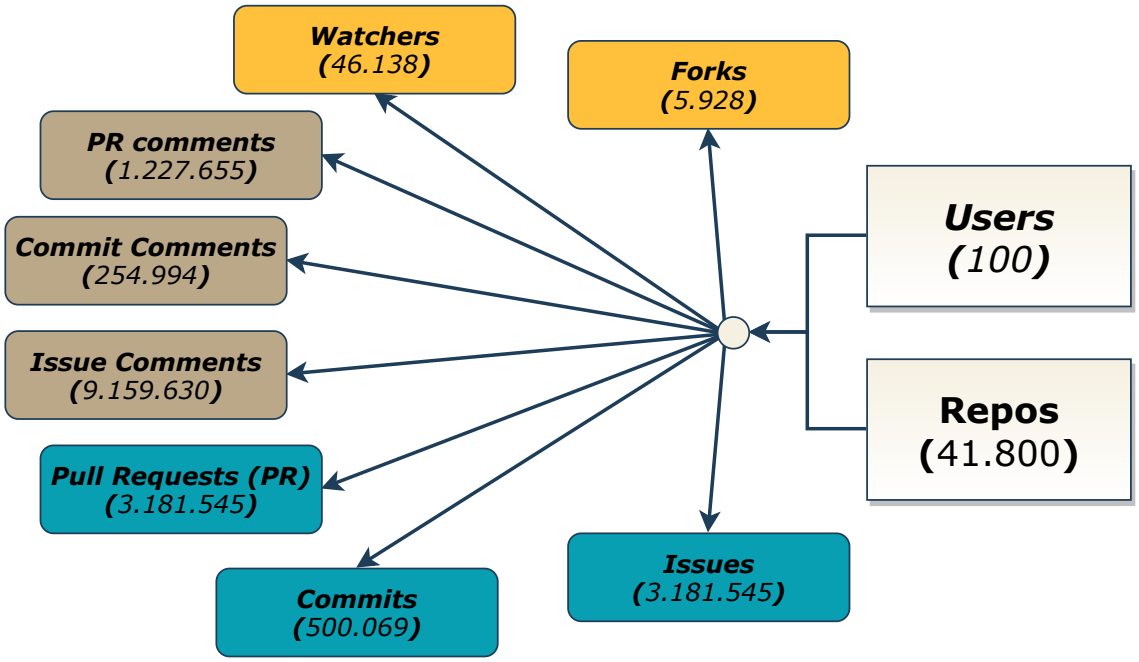
Bu 3 geçici koleksiyon birleştirilmiş ve *sIssues* koleksiyonu elde edilmiştir. Benzer şekilde, diğer koleksiyonlar da Şekil 5.2'de gösterildiği gibi elde edilmiştir.

Hatalı veri düzeltme, bağlantı alanları ekleme ve GHTorrent koleksiyonlarından veri seçme işlemi yapıldıktan sonra yeni alt veri kümesi oluşturulmuştur. GHTorrent ile önerilen filtrelenmiş veri seti arasındaki boyut karşılaştırması Tablo 5.5'te verilmiştir. Tablo 5.5'ten görülebileceği gibi, çok büyük verilerle çalışmak, özellikle algoritma veya model geliştirme aşamasında ciddi zaman kayıplarına neden olmaktadır. Bu kapsamda, önerilen filtrelenmiş veri setinin araştırmacılara ortak bir veri havuzu sağlayacağı, aynı zamanda çalışmalarında zaman kazandıracağı düşünülmektedir.

Tablo 5.5 GHTorrent ile önerilen veri kümesi GitDataSCP karşılaştırılması

	~Disk Boyutu		**~Doküman Sayısı**	
Koleksiyon Adı	GHTorrent	GitDataSCP	GHTorrent	GitDataSCP
Users	1GB	0.8MB	5.000.000	100
Repos	30GB	45MB	20.000.000	40.000
Commits	298GB	4GB	41.000.000	500.000
Commit Comments	1GB	90MB	2.000.000	250.000
Issues	11GB	3GB	17.000.000	3.000.000
Issue Comments	15GB	4GB	31.000.000	9.000.000
Pull Requests	23GB	5GB	8.000.000	1.500.000
Pull Request Comments	6 GB	1GB	5.000.000	1.200.000
Followers	1 GB	20 MB	7.000.000	130.000
Forks	8 GB	5 MB	11.000.000	7.000
Watchers	7 GB	8 MB	38.000.000	50.000
RepoCollaborators	1 GB	2 MB	5.000.000	4.000

Sonuç olarak, bu tez bünyesinde araştırmacılar için çok geniş yelpazede yazılım mühendisliği problemlerinde kullanılabilecek —GitHub platformuna nispeten küçük boyutlu— platformdaki seyreklik (sparsity) problemini barındıran kamuya açık bir veri kümesi olan GitDataSCP üretilmiştir (Şekil 5.3).



Şekil 5.3 GitDataSCP veri kümesine genel bir bakış

İncelenen çalışmalarda kullanılan veri kümelerindeki kullanıcı-proje oranlarına göre, önerilen veri kümesi platformun doğasında var olan ölçüye yakın bir seyreklik değerine sahiptir (Tablo 5.6). Bu seyreklik oranının yüksek olması geliştirilen algoritmaların başarısına negatif bir etki göstermesine rağmen, gerçek dünya problemlerine de bir o kadar yakın sonuçlar vermesini sağlamaktadır. Bu bağlamda, veri kümesinin yüksek seyreklik içermesinin özgünlük ve gerçeklik açısından önemli olduğu düşünülmektedir.

Tablo 5.6 İlgili çalışmalarda kullanılan veri kümelerinin seyreklik oranları

Yayın ID	Kullanıcı Sayısı	Proje Sayısı	Seyreklik Oranı (~)
[66]	1.700	22.000	0,07
[70]	1.255	58.092	0,02
[23]	1.070	1.600	0,66
[71]	62.607	9.447	0,15
[48]	62.607	9.447	0,15
GitDataSCP	100	41.280	**0,002**

Tezin bu bölümünde üretilen veri kümesi bir dergi makalesi olarak yayınlanmıştır.

6
UYGULAMA

OSS geliştirme araçlarının artan yetenekleri sayesinde, açık kaynak kullanıcı ve proje sayısı her yıl artmaktadır. Bunlar arasında en çok tercih edilenlerden biri olan GitHub, her biri farklı bir karaktere ve beceriye sahip milyonlarca geliştirici ile birlikte, farklı sorunlara çözümler sunan çok çeşitli projeleri barındırmaktadır. Bu kadar büyük miktarda veriye sahip platformlarda karşılaşılan zorluklardan biri, geliştiricilerin kendilerininkine benzer projeleri bulmaları, ilgilerini çekecek projeleri keşfetmeleri veya katkıda bulunabilecekleri projelere ulaşmalarıdır.

Geliştiriciler ilgilerini çekebilecekleri projeleri bulmak için arama motorlarını veya platform içi arama menülerini kullandıklarında, metin tabanlı aramanın kısıtlamaları [208] ve doğru anahtar kelimeleri bulma problemi nedeniyle onlar için önemli olabilecek çoğu projeyi gözden kaçırmaktadırlar [209]. Bu sorunun üstesinden gelmek için çeşitli proje öneri sistemleri geliştirilirken, öneri modellerinin düzgün çalışması için projelerin kullanıcılar tarafından derecelendirilmesi gerekmektedir. Nasıl ki sinema izleyicileri izledikleri filmlere oy (rating) veriyorsa, geliştiriciler de ilgilendikleri projeleri derecelendirmelidirler. Ancak, hali hazırda GitHub gibi yazılım işbirliği platformlarında böyle bir kayıt veya bilgi mevcut değildir. Bir başka deyişle, bu problem üzerinde çalışmak için etiketli veri kümesi bulunmamaktadır. Bu bağlamda, bir kullanıcının projelere verdiği oyu hesaplamak için geliştiricilere ve projelere ait bazı aktivitelerinden veya özelliklerinden elde edilen geliştirici metrikleri kullanılabilir.

Tezin bu bölümünde, Bölüm 4.3'te önerilen geliştirici metriklerinin değerlendirilmesi için bir proje öneri sistemi tasarlanmıştır. Öneri sistemi, GitHub verilerini içeren, tez kapsamında üretilmiş, GitDataSCP adı verilen bir veri kümesi ile geliştirilmiştir.

6.1 Öneri Model Tasarımı

GitHub proje önerisi için, ürün-ürün İşbirliğine Dayalı Filtreleme (item-item Collaborative Filtering) kullanan bir öneri modeli oluşturulmuştur. İşbirlikçi filtreleme yöntemi genellikle bir kullanıcının davranışı, etkinlikleri veya tercihleri hakkında bilgi toplamayı, bu bilgiyi analiz etmeyi ve kullanıcının diğer kullanıcılara benzerliklerine göre neyi seveceğini tahmin etmeyi hedefler [210]. İşbirlikçi filtreleme, geçmişte benzer tercihlere sahip olan farklı bireylerin, gelecekte aynı seçimleri yapacağı varsayımına dayanmaktadır. Örneğin; Zeynep adlı bir kullanıcı A, B ve C ürünlerini, Deniz adlı bir kullanıcı ise B, C ve D ürünlerini tercih etmiş ise; gelecekte Zeynep muhtemelen D ürününü ve Deniz ise A ürününü tercih edecektir. Örnekteki gibi, ürün-ürün mantığı, bir kullanıcıya ona benzer bir kullanıcının tükettiği ürünlere benzer ürünler önerir. Kitap satış sitelerindeki "incelenen kitaba bakanların satın aldığı diğer kitaplar" önerisi bu mantığa örnek olarak verilebilir.

Proje öneri sistemi, belirli bir metrik ile bir Proje-Geliştirici Oy Matrisi'nin (PGOM) oluşturulması, projeler arasında benzerliklerin bulunması, geliştiriciler için önerilerin üretilmesi ve önerilerin değerlendirilmesi aşamalarını içerir. İlk olarak, seçilen metrikler kullanılarak bir PGOM (Table 6.1) oluşturulmuştur. Tablo 6.1'de görüldüğü gibi, sütunlar kullanıcıları, satırlar projeleri (depoları) temsil etmektedir. Bir metrik seçilir, ardından metriğin değerleri (miktar, oran veya ikili) hücrelere girilir. Örneğin, Tablo 6.1'de görüldüğü gibi, *k-1* geliştiricisi, *p-2* deposunda 7 adet *issue* açmıştır.

	k-1	**k-2**	...	**k-i**
p-1	0	51	...	96
p-2	7	0	...	0
p-3	0	0	...	4
...	...	...	...	...
p-n	0	34	...	0

(a) Metriklerin gerçek değerleri

	k-1	**k-2**	...	**k-i**
p-1	0	4	...	10
p-2	0,08	0	...	0
p-3	0	0	...	0,5
...	...	...	...	...
p-n	0	3	...	0

(b) Metriklerin normalize değerleri

Tablo 6.1 *issue_opened* metriğine ait PGOM örneğinden bir kesit

Orijinal PGOM değerleri *min-max* normalizasyonu ile normalleştirilmiştir. Böylece değerler 0-10 arasına ölçeklenmiştir. Aynı sinema-izleyici modelindeki gibi, her geliştiricinin projelere 0-10 aralığında bir oy verdiği varsayılmıştır (Tablo 6.1b). Ardından, projeler arasındaki benzerlik iki farklı yöntem ile hesaplanmış (kosinüs benzerliği ve TF-IDF) ve her bir geliştiriciye onun için en yüksek oya sahip ilk *n* proje önerilmiştir. Son olarak, iki değerlendirme yöntemi ile yapılan öneriler değerlendirilmiştir.

6.2 Benzerlik Yöntemleri

Geliştiricilere proje önermek için öncelikle birbirine benzer projeler bulunmuştur. Hafıza-bazlı (memory-based) işbirlikçi filtreleme öneri sistemi bir ürün için kendisine yakın komşuların bir kümesini seçmektedir [211]. Önerilen modelde de bu yaklaşım kullanılmıştır.

6.2.1 Kosinüs Benzerliği (İçerikten-Bağımsız)

Bu yöntemde, İçerikten-bağımsız Benzerlik (CFS) yöntemi kullanılmıştır. Projeler arasındaki benzerliği skorlamak için proje öneri sistemlerinde sıklıkla kullanılan kosinüs benzerliği kullanılmıştır.

1. PGOM'dan bir proje vektörünü içeren ilgili satır örneği (Denklem 6.1).

$$\vec{P}_{metricX}(i) = (u_1, u_2, u_3, \cdots, u_{100}) \tag{6.1}$$

2. *Proje-i* (P_i) ile *Proje-j* (P_j) arasındaki skoru *metrikX*'e göre hesaplamak için, Denklem 6.2 kullanılmıştır.

$$\begin{aligned} CFS(P_i, P_j) &= \cos(\vec{P}_{metrikX}(i), \vec{P}_{metrikX}(j)) \\ &= \frac{\vec{P}_{metrikX}(i) \cdot \vec{P}_{metrikX}(j)}{\|\vec{P}_{metrikX}(i)\| * \|\vec{P}_{metrikX}(j)\|} \end{aligned} \tag{6.2}$$

6.2.2 TF-IDF Benzerliği (İçeriğe-Bağlı)

İkinci olarak, İçeriğe-bağlı Benzerlik (CBS) yöntemi kullanılmıştır. Burada "İçeriğe-bağlı" teriminin anlamı metin bazlı bir benzerlik yöntemi kullanılmasıdır. Projeler arasındaki benzerliği hesaplamak için bir projeye ait tüm yorumlar ele alınmıştır. Şekil 6.1'de, üzerinde çalışılan korpusu kaba bir şekilde anlamak için yorumlarda sık kullanılan kelimeler, bir kelime bulutu grafiği ile gösterilmiştir.

Şekil 6.1 Tüm süreçlere ait yorumlarda geçen en sık kelimeler

1. Projelerdeki tüm *commit*, *issue* ve *PR* özelliklerine ait yorumların hepsi elde edilmiştir. Tablo 6.2'de, örnek olarak *issue* yorumlarından bazıları verilmiştir.

Tablo 6.2 *Issue* yorumlarından bazı örnekler

Kullanıcı id	***Issue* id**	**Proje id**	**İçerik**
55859	138	1	I'd like to see this commit, or at least this feature, included. I'm trying to build something which uses the time information for a distributed team and I'd like to retain the timezone information ...
145667	629	27	Here is a simpler example that I think shows the problem better.\r \n require ŕubinius/debugger ...
...	...	...	...

2. Elde edilen metinlerin hepsine metin ön işlemleri uygulanılmıştır.

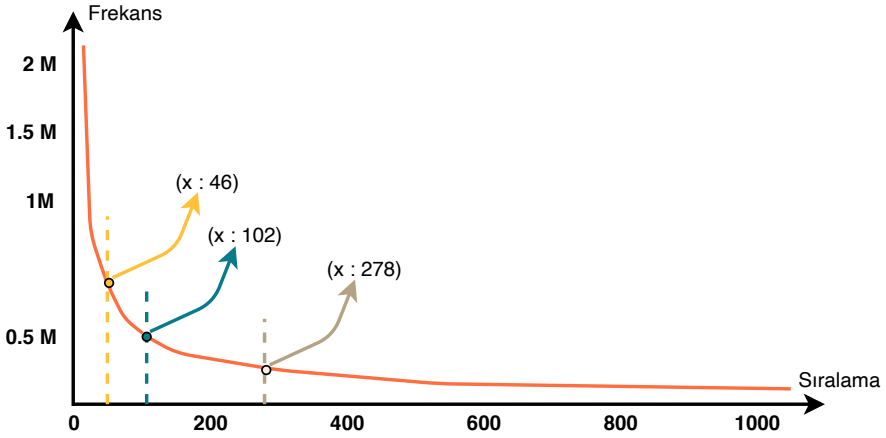
   - Tüm metin küçük harfe çevrildi.
   - Tüm sayısal değerler silindi.
   - Tüm noktalama işaretleri silindi.
   - Tüm gereksiz kelimeler (stopwords) silindi.
   - Fazla olan boşluklar silindi.

3. Tüm yorumlar projelere göre gruplanmış ve tek bir sütunda birleştirilmiştir (Tablo 6.3'teki *Tüm Yorumlar* sütunu). Böylece her bir proje için tek bir yorum dokümanı oluşturulmuştur.

Tablo 6.3 Tüm yorumların bir araya getirilmesi

	Tüm Yorumlar	***Commit* Y.**	***Issue* Y.**	***PR* Y.**
Proje-1	like included exception ...	exception ...	like included ...	*NaN*
Proje-2	...	...	...	...
...	...	...	...	*NaN*
Proje-n	...	*NaN*	...	...

4. Bu dokümanlar sıradan bir bilgisayar ile işlenemeyecek kadar büyük boyuttadırlar[1]. Bu yüzden her bir yorum dokümanı için en sık kullanılan n kelimenin seçilmesine karar verilmiştir. Kesim noktasını (n) belirlemek için korpusa Zipf yasası uygulanmıştır [212]. Şekil 6.2'de görüldüğü gibi, ikinci dirsekten daha yüksek dereceye sahip kelimeler (işlevsel kelimeler) dikkate alınmıştır. Böylece, her bir proje için en fazla 100 kelime içeren dokümanlar üretilmiştir.



Şekil 6.2 Yorumlardaki kelimelerin Zipf yasasına göre dağılımı

5. Bütün projeler için Terim Sıklığı-Ters Belge Sıklığı (*TF-IDF*) ile benzerlik skorları bir önceki aşamada üretilen dokümanlar üzerinden hesaplanmıştır (Denklem 6.3 ve 6.4). Denklemlerde t bir kelimeyi (terim), d kelimeleri içeren dokümanı, p ise depoları (proje) temsil etmektedir.

$$tf\_idf_{(t,d,p_i)} = tf(t,p_i) * idf(t,p_i) \tag{6.3}$$

$$CBS_{(P_i,P_j)} = tf\_idf(p_i) * tf\_idf(p_j) \tag{6.4}$$

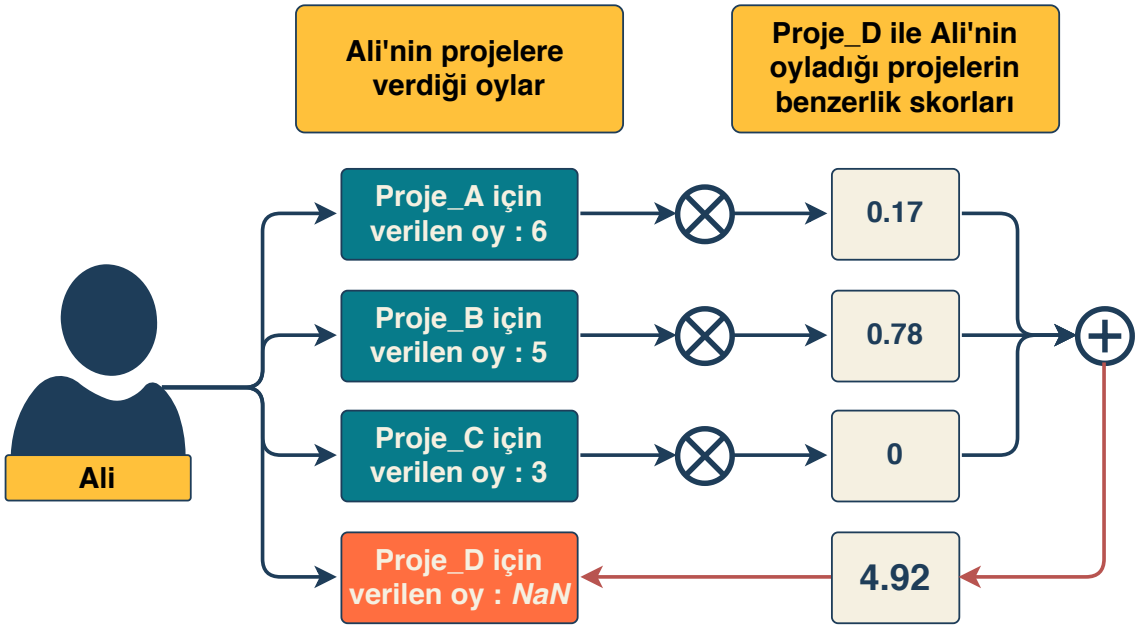
[1]Örneğin, *issue* yorumlarında, her bir proje için ortalama kelime sayısı yaklaşık olarak 11.000'dir.

6.2.3 Bilinmeyen Projelerin Üstesinden Gelinmesi

Yukarıdaki iki yöntem ile proje benzerliklerini hesapladıktan sonra, her geliştirici için oylamadığı projelere (bilinmeyen[1]) oy vermek (üretmek) için bir algoritma ile kullanılmıştır. Bilinmeyen projeler ile bilinenler arasındaki benzerliği bulmak için bilinen projelerin oyları kullanılmıştır [66]. Bilinmeyen bir projenin oyu hesaplanırken, geliştiricinin bildiği projeler ile bilmediği projeler arasındaki benzerlik skorlarının nokta çarpımı (dot product) kullanılmıştır (Denklem 6.5).

$$bilinmeyen_{oy} = \sum_{n=0}^{i} bilinen^{i}_{oy} * benzerlik_{bilinen^{i},bilinmeyen} \tag{6.5}$$

Bu hesaplamayı gösteren örnek bir senaryo Şekil 6.3'te verilmiştir.



Şekil 6.3 Bilinmeyen projelerin skorlarını katkı yapılan projeler ile hesaplama

6.3 Öneri Değerlendirme Yaklaşımları

Her bir geliştiriciye bilmediği projeler arasından en yüksek oya sahip ilk n proje önerilmiştir. Ardından, bu sonuçlar önerilen iki farklı yaklaşım ile değerlendirilmiştir.

Geliştiricilere proje önerirken, önerilen projeleri değerlendirmek için bir temel gerçek (ground truth) olmalıdır. Klasik öneri sistemlerinin aksine, bu problemde eğitilmemiş (unsupervised) bir durum söz konusudur. Literatürdeki bazı benzer çalışmalardaki değerlendirme kriterlerine aşağıda yer verilmiştir.

[1]Geliştiricinin hiç bir katkı yapmadığı ve bir ilişkisi olmadığı projeler *bilinmeyen* olarak varsayılmıştır.

- PGOM rastgele olarak test ve eğitim kümelerine bölünmüştür. Test ve eğitim kümelerinde önerilen ilk *n* projenin [66] arasındaki kesişimden doğruluk değerleri hesaplanmıştır. Ancak, başka bir çalışmada, bu yöntemin zamanın önemli bir parametre olduğu GitHub gibi platformlarda kullanılmaması gerektiği belirtilmiş, verileri rastgele bölerek çapraz doğrulama yönteminin uygulanmasının geçmişteki bir etkinliği gelecekteki verilerle tahmin etme sorununu ortaya çıkaracağı işaret edilmiştir [49].

- Başka bir çalışmada, bir geliştiricinin ilgili projeye ilişkin geçmiş *commit* aktivitesi kullanılarak yapılan öneriler değerlendirilmiştir. Projede belirli bir geliştiriciye ait *commit* sayısı belirlenen bir eşik değerin üzerinde ise, yapılan önerinin doğru olduğu varsayılmıştır. Proje başına ortalama *commit* sayısı, veri kümesinde eşik değer olarak belirlenmiştir [59]. Tez kapsamında, *commit* özelliği zaten bir metrik olarak kullanıldığı için bu yaklaşım tercih edilmemiştir.

- Bir geliştiricinin gelecekte bir projeye katılıp katılmayacağı tahmin edilen bir başka çalışmada, veri kümesi zamana göre iki farklı kümeye bölünmüştür. Bu şekilde, tahmin edilen sonuç gerçek gelecek verilerle doğrulanmıştır [71].

- Ankete dayalı bir diğer çalışmada, katılımcılara hangi özelliklerin bir öneri aracı olarak kullanılabileceği sorulmuştur. Katılımcıların çoğunluğu, geliştiricilerin önceden kodladığı veya bildiği dillerin öneri için önemli olduğunu belirtmiştir [213].

Literatürdeki bu değerlendirmeler ışığında, tez kapsamında iki farklı değerlendirme yaklaşımı önerilmiştir.

6.3.1 Topluluk İlişkisi Yaklaşımı (TİY)

Bu yaklaşımda, GitHub'ın *izleme (watching)* ve *çatallama (forking)* özellikleri temel gerçek olarak kullanılmıştır. GitHub kullanıcıları geliştirmelerini gözlemlemek istedikleri projeleri izleyebilirler [13]. Yani bir geliştirici bir projeyi izliyorsa, bu durum, onun o projeyle ilgili olduğunu göstermektedir. Benzer şekilde, çatallamada kullanıcının ilgilendiği projelere bağımsız olarak katkı yapmasını sağlayan bir fonksiyondur [214]. Çoğu proje dışı geliştirici ilgilendikleri projelere katkı sağlamak için çatalla-çek (fork-pull) mekanizmasını kullanmaktadırlar. Bu bağlamda, bu iki özelliğin öneri için önemli olduğu düşünülmüştür. İzleme ve çatallama özellikleri kullanılırken, aşağıda detayları verilen bazı iyileştirmeler yapılmıştır.

Bir GitHub deposunun tam adı, sahip adı (login) ve depo adı (repo name) ile birleştirilerek oluşturulur (Örneğin; davidteather/handtracking). Sonuçlar analiz

İlk-*5* öneri	Tam	Yarım
visionmedia/co		**co**
fengmk2/emoji		**fengmk2**
iojs/io.js	iojs/io.j	
julgru/co-read	julgru/co-read	
koajs/compose/		

(a) Ali için önerilenler

Skor	Depo tam adı
⊕	adamwiggins/**co**
⊕	**fengmk2**/parameter
⊕	**iojs/io.js**
⊕	**julgru/co-read**
⊖	fengmk2/cnpmjs.org
....	...

(b) Ali'nin izledikleri

⊕: tam, ⊕: yarım, ⊖:yanlış

Tablo 6.4 Önerilerdeki tam ve yarım eşleşme durumları için örnek bir senaryo

edilirken, modelin bazı önerilerinde sadece sahip adını veya depo adını doğru tahmin ettiği görülmüştür. Bu durumlarda model, doğru depo sahibinin yanlış bir projesini veya tam tersini önermiş olmaktadır. Bu tip öneriler model tarafından yarım puan olarak değerlendirilmiştir. Çünkü, bir geliştiriciye yalnızca doğru depo sahibini önermek, o geliştiricinin ilgili kişinin diğer projelerinden haberdar olmasını sağlayacaktır. Benzer şekilde, model, yanlış depo sahibi ile doğru bir repo adı önermesi durumunda, aslında yapılan öneri doğru bir deponun çatallanmış bir deposunun önerildiği anlamına gelmektedir. Böylece ilgili geliştirici ana depoyu da keşfedebilir.

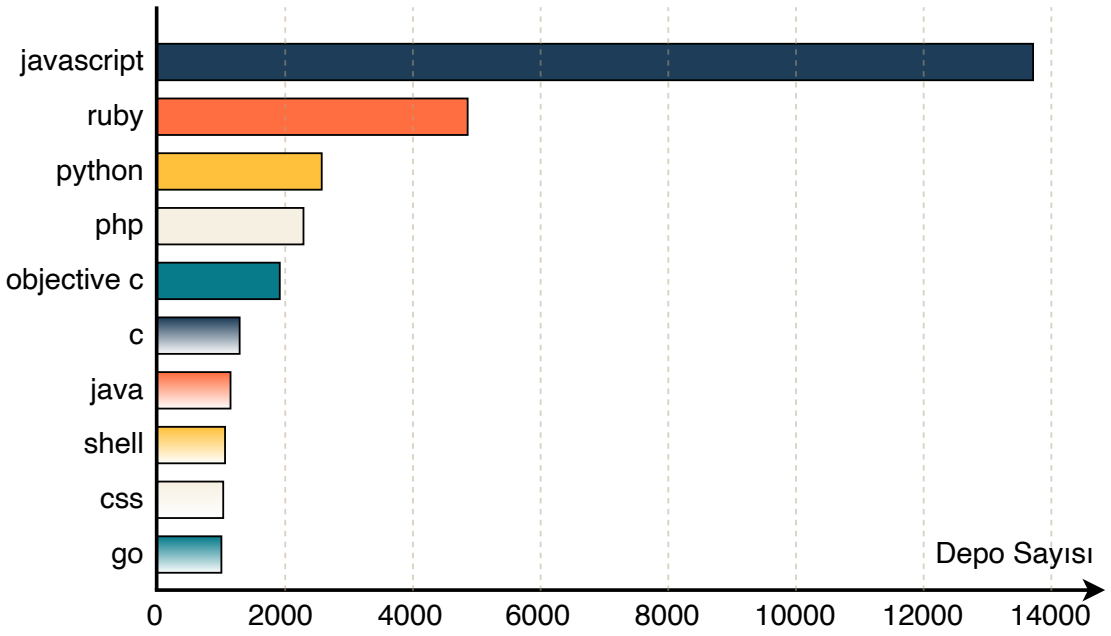
Bu duruma örnek bir senaryo Tablo 6.4'te verilmiştir. Ali için önerilen projeler Tablo 6.4a'daki ilk sütunda verilmiştir. Önerilenlerden ikisi Ali tarafından izlenen projelerdir, böylece -iki tam doğru ile- başlangıç skoru ***2*** olur. *Fengmk2* adlı kullanıcıya ait olan (*fengmk2/parameter* ve *fengmk2/cnpmjs.org*) isimli iki depo Ali'nin izlediği depolar arasındadır (Tablo 6.4b). Model Ali'ye, tanıdığı bir geliştiricinin takip etmediği bir projesi *fengmk2/emoji* önermiştir. Benzer şekilde, Ali *visionmedia/co* projesinin bir çatallanmış deposunu izlemektedir. Böylece, iki yarım-skor daha başlangıç skoruna eklenir ve son skor ***3 (2 + 0.5 + 0.5)*** olarak elde edilir.

$$hit\_skor_{TiY} = \begin{cases} 100 * \dfrac{hit_{tam} + (hit_{yarim} * 0.5)}{n}, & \#\_ilgili \geqslant n \text{ ise} \\ 100 * \dfrac{hit_{tam} + (hit_{yarim} * 0.5)}{\#\_ilgili}, & \text{aksi takdirde} \end{cases} \tag{6.6}$$

Hit skoru (Doğru öneri skoru) hesaplamak için Denklem 6.6 kullanılmıştır. Sonuçlara detaylı bakıldığında bazı geliştiricilerin *n*'den daha az proje ile ilgili oldukları görülmüş, bu durum hesaplamada dikkate alınmıştır. Denklem 6.6'da, "*#_ilgili*" terimi, bir geliştiricinin izlediği veya çatalladığı toplam depo sayısını temsil etmektedir.

6.3.2 Dil Deneyimi Yaklaşımı (DDY)

İkinci bir yaklaşım olarak, geliştiricilerin kodlama dili deneyiminden faydalanılmıştır. Veri kümesindeki projelerin dilleri işlenmiş, onlardan edinilen bilgiler ile bir değerlendirme kriteri oluşturulmuştur. Veri kümesindeki projelerin 94 farklı kodlama diline sahip olduğu görülmüştür. En çok kullanılan 20 dil, Şekil 6.4'te verilmiştir. Veri kümesindeki dil çeşitliliği sayesinde dil parametresinin geçerli bir değerlendirme kriteri olarak kullanılabileceği düşünülmüştür.



Şekil 6.4 Veri kümesinde çok kullanılan programlama dilleri

Geliştiricilerin sahip olduğu, izlediği veya *commit* yaptığı tüm proje dilleri ortaya çıkarılmıştır (Tablo 6.5). Böylece, bir geliştiricinin herhangi bir faaliyeti olan dillerin keşfedilmesi amaçlanılmıştır. Ardından, bunlar kullanım sıklıklarına göre sıralanmış ve en çok kullanılan üç dil elde edilmiştir (Tablo 6.5 'teki "uzmanlaşılan diller" sütunu).

Tablo 6.5 Her bir geliştirici için uzmanlaştığı kodlama dillerinin keşfedilmesi

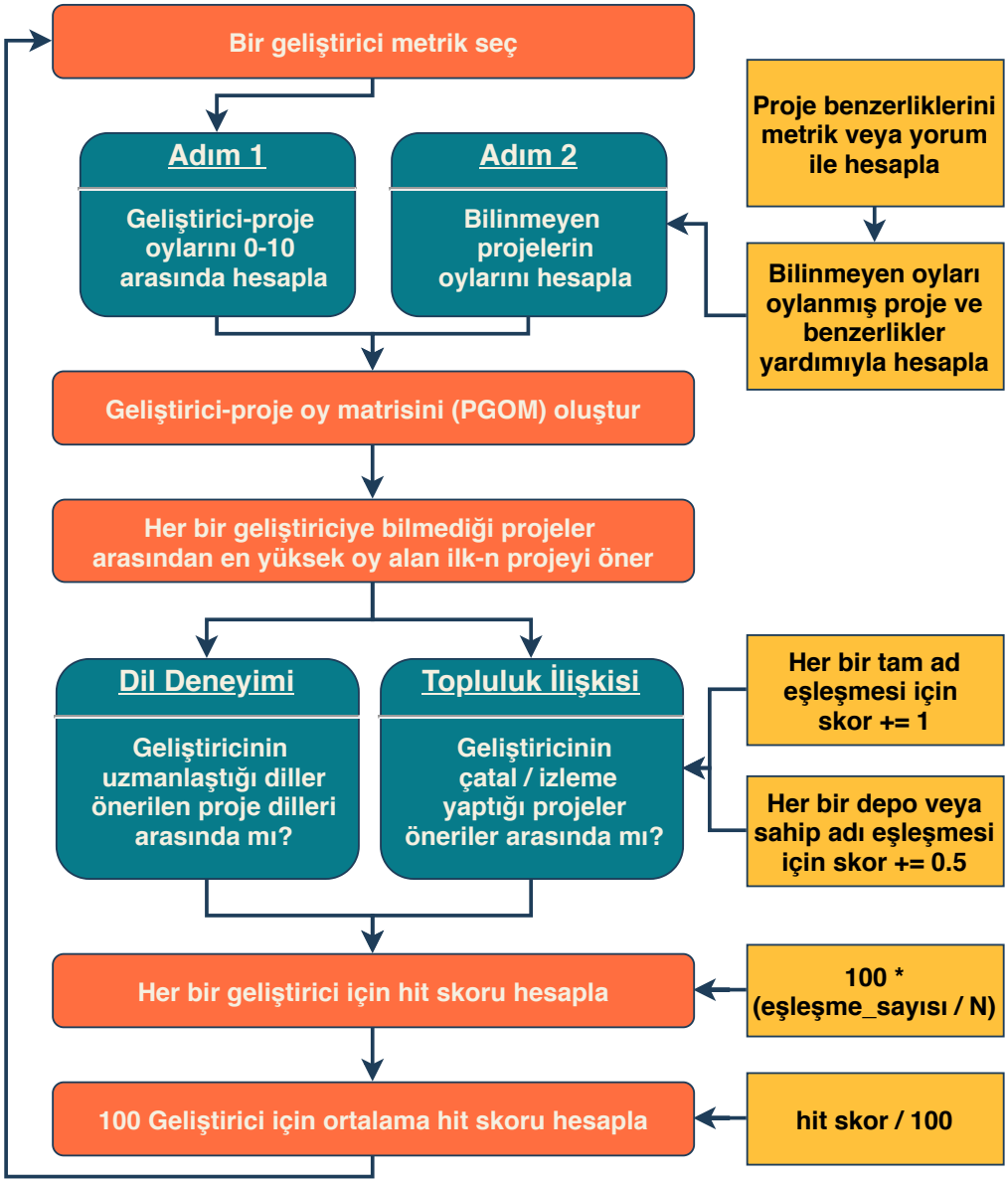
Kullanıcı ID	Tüm Kodlama Dilleri	Uzmanlaşılan Diller
21	[c, ruby, ruby, ruby, go, ruby, ruby, ...]	[ruby, javascript, go]
13760	[objective-c, objective-c, c, haskell, ...]	[objective-c, c, ruby]
3346407	[css, python, python, lua, c++, ...]	[python, javascript, css]
...	...	...

Dil deneyimi değerlendirme yönteminde, önerilen projenin dili, geliştiricinin uzmanlaştığı diller arasında ise, o proje bir *hit* olarak kabul edilmiştir (Denklem 6.7).

$$hit\_skor_{DDY} = 100 * \frac{hit_{dil}}{n} \tag{6.7}$$

Bu yaklaşımın skorları ilk yaklaşıma göre oldukça yüksektir. Fakat, odaklanılan asıl amaç dikkat çekici metriklerin 2 farklı yöntem ile doğrulanmasıdır. Yapılan çıkarımlarda iki yaklaşım bağımsız olarak değerlendirilmiştir.

Sonuç olarak bu tez kapsamında, GitHub platformu için bir proje öneri modeli oluşturulmuştur. Bir özelliğin metrik olarak seçilmesiyle başlayıp **hit** skorlarının hesaplanmasıyla sona eren bu modelin algoritması, Şekil 6.5'te sunulmuştur.



Şekil 6.5 Proje öneri sisteminin akış şeması

Bu algoritmaların geliştirilmesinde kullanılan kodlar bir GitHub deposunda paylaşılmıştır[1].

[1] *https://github.com/kadirseker00/githubProjectRecommender*

6.3.3 Geçerliliğe Yönelik Tehditler

Bu bölümde geliştirilen uygulamanın kapsamı GitHub'daki aktif geliştiricilerle sınırlıdır. İlk zorluk, bu metriklerin etkin olmayan geliştiriciler için iyi çalışıp çalışmayacağıdır. Bu durum, klasik öneri sistemlerindeki soğuk başlatma (cold start) problemine benzemektedir. Etkin olmayan geliştiriciler için öneri yapmak, haklarındaki bilgi eksikliği nedeniyle, çok daha zordur. Önerilen metriklerin, bu tür geliştiricileri içeren başka veri kümelerinde analiz edilmesinin önemli olduğu düşünülmektedir.

Proje önerisi problemi, GitHub gibi yüksek sayıda proje içeren ortamlar için oldukça önemlidir. Bunun kanıtı olarak, GitHub kısa süre önce bazı kullanıcı etkinliklerine dayalı olarak "Keşfet" sayfasında proje önerileri sunmaya başlamıştır. Önerilen metriklerin dışında, yazılım mühendisliğinin diğer zorluklarına uygulanan metrikler öneri problemi için başarıyla kullanılabilir. Bu bağlamda, araştırmacıların farklı metrikler kullanarak bu sorun üzerinde çalışması önemli görülmektedir.

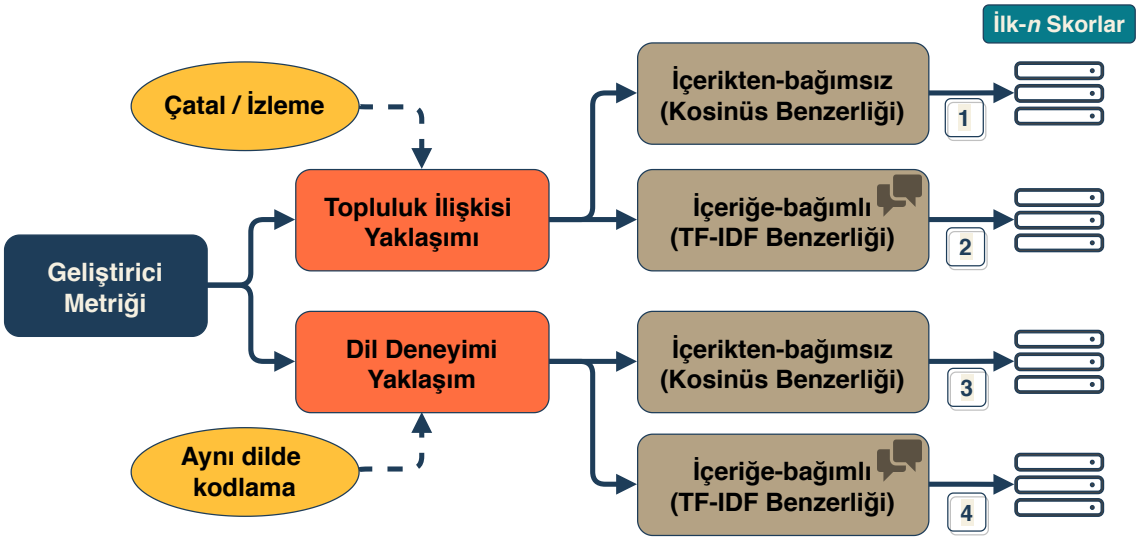
Son olarak, klasik tavsiye sistemlerinden farklı olarak, sorunumuz için etiketli veriler (temel gerçek) bulunmamaktadır. Bu nedenle, GitHub gibi platformlar için proje önerisi üzerinde çalışmak üzere etiketli bir veri kümesi oluşturmanın önemli olduğu düşünülmektedir.

6.4 Deneysel Sonuçlar

Bölüm 4.3'te önerilen 40 farklı geliştirici metriği bu modelde kullanılmıştır. Bu metrikler, bir geliştiricinin bir projedeki geçmiş etkinlikleri hakkında bilgiler içermektedir. Kullanılan tüm metrikler, *min-max* normalizasyon tekniği kullanılarak 0-10 arasına ölçeklenmiştir. Ardından, tüm bu metrikler proje öneri modeline girdi olarak verilmiştir. Modelin sonuçları, ilk 1-3-5-10-20 öneri *hit* skorları ile değerlendirilmiştir.

Bu bölümde, seçilen bazı metriklerin ilk 1-5-10 *hit* skorları gösterilmiştir. Oransal tabanlı üretilen füzyon metriklerinin çoğu ve optimize edilmiş tekil metrikler tüm sonuçlar açısından değerlendirildiğinde oldukça zayıf kalmıştır. Bu nedenle, bu metrikler aşağıdaki skor tablolarından kaldırılmıştır. 40 metriğin hepsine ait sonuçlar Tüm Sonuçlar A bölümünde, tüm ilk n değerleri ile birlikte verilmiştir.

Projelerin benzerlik skorlarını hesaplamak için iki benzerlik yöntemi kullanılmıştır. Geliştirici metriklerinin doğruluğunu değerlendirmek için ise iki farklı yaklaşımdan faydalanılmıştır. Böylelikle, benzerlik yöntemlerinin ve değerlendirme yaklaşımlarının kombinasyonları ile öneri sisteminin 4 farklı sonucu incelenecektir (Şekil 6.6).



Şekil 6.6 Dört parametrenin kombinasyonu ile tüm *hit* skorların hesaplanması

Tablo 6.6'da ve 6.7'de iki farklı değerlendirme yaklaşımına göre elde edilen *hit* skor yüzdelikleri sunulmuştur.

Bu tablolarda, ilk-*n* sütunları yüzdesel olarak *hit* skorları vermektedir. Ayrıca, yeşil (1.), sarı (2.) ve kırmızı (3.) olarak boyanan hücreler kendi sütunlarındaki sırasıyla en iyi skorları göstermektedir. Bunun yanı sıra, her sutüna ve tüm skorların ortalamasına göre en iyi 5 metrik de **kalın** font ile gösterilmiştir. Tüm skorlara göre ortalama değer hesaplanırken, soru cevaplama sistemlerinde sıklıkla kullanılan Ortalama Karşılıklı Sıra (Mean Reciprocal Rank-MRR) tekniği kullanılmıştır. Burada, orijinal denklemdeki sorgu terimi yerine model sayısı kullanılmıştır (Denklem 6.8). Denklem 6.8'deki n değeri, farklı parametreler ile ürettiğimiz model sayısını temsil etmektedir *(n=6)*.

$$MRR_{skor} = \frac{1}{n}\sum_{i=0}^{n}\frac{1}{derece_i} \tag{6.8}$$

Örneğin, *comments* metriği için, Tablo 6.6'da verilen MRR değerinin nasıl hesaplandığı Denklem 6.9 ile gösterilmiştir. Hesaplama yapılırken 6 farklı model için ilgili metriğin sıra (rank) değerleri kullanılmıştır.

$$MRR_{skor}^{comments} = \frac{1}{6} * (\frac{1}{2} + \frac{1}{1} + \frac{1}{2} + \frac{1}{15} + \frac{1}{16} + \frac{1}{18}) = 0.36 \tag{6.9}$$

Tablo 6.6'ya ve Tablo 6.7'ye ek olarak, tüm ilk-*n* değerlerine göre en başarılı 5 metriğin (*Sun_metric* ile birlikte) *hit* skorları iki farklı yaklaşım için de Şekil 6.7'de ve Şekil 6.8'de verilmiştir.

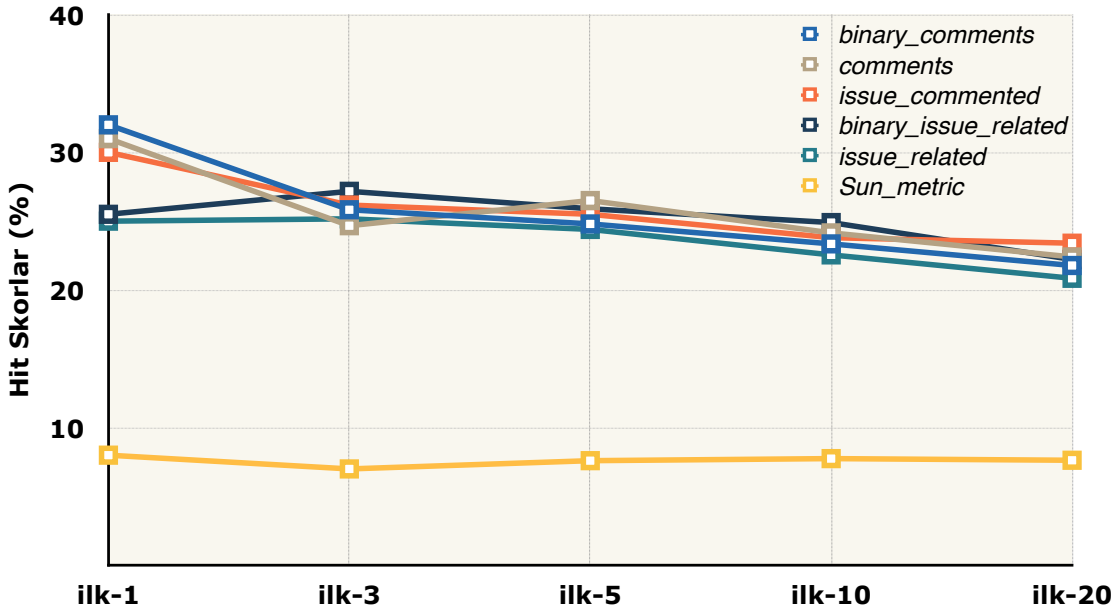
Tablo 6.6 Topluluk ilişkisi yaklaşımıyla geliştirici metriklerinin hit skorları

Topluluk İlişkisi Değerlendirme Yaklaşımı

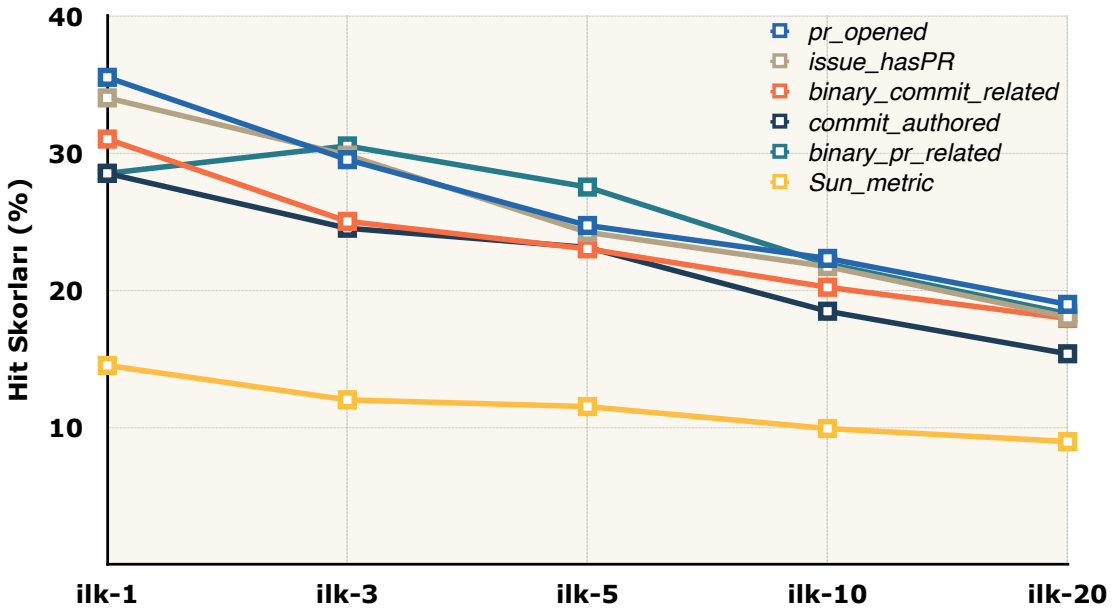
	Geliştirici Metrikleri	İçerik (bağımsız) ilk1	ilk5	ilk10	İçerik (bağlı) ilk1	ilk5	ilk10	MRR
1	*commit_authored*	14,0	12,7	11,6	**28,5**	**23,1**	18,5	0,10
2	*commit_commented*	22,0	19,2	17,0	26,0	20,5	18,5	0,08
3	*commit_committed*	15,5	12,9	11,6	**29,0**	22,6	18,5	0,11
4	*issue_assigned*	7,0	8,3	7,5	26,5	17,4	14,3	0,06
5	*issue_closed*	11,5	8,1	6,3	20,0	14,8	13,2	0,05
6	*issue_hasPR*	**26,0**	17,9	16,2	**34,0**	**24,2**	**21,7**	0,25
7	*issue_commented*	**30,0**	**25,5**	**23,8**	22,0	18,2	17,0	0,20
8	*issue_opened*	23,0	20,7	18,7	26,0	22,1	18,7	0,10
9	*pr_assigned*	4,0	4,2	4,3	12,5	9,3	8,5	0,04
10	*pr_commented*	20,0	19,6	17,9	20,0	20,1	18,6	0,07
11	*pr_merged*	16,0	12,8	11,8	28,0	20,7	18,9	0,09
12	***pr_opened***	**26,1**	19,2	16,0	**35,5**	**24,7**	**22,3**	**0,48**
13	***comments***	**31,0**	**26,5**	**24,2**	22,0	19,3	17,0	**0,36**
14	*code_contributions*	23,0	22,2	21,0	24,0	20,5	18,3	0,09
15	*commit_related*	23,0	18,4	17,8	26,0	19,4	19,0	0,08
16	*pr_related*	25,0	23,6	22,1	22,0	**23,9**	**20,1**	0,15
17	*issue_related*	25,0	**24,4**	**23,4**	18,5	17,0	15,2	0,11
18	***binary_comments***	**32,0**	**24,8**	**23,4**	17,5	18,8	17,6	**0,26**
19	*binary_code_contributions*	24,0	**24,4**	**23,6**	22,0	20,7	18,1	0,12
20	*binary_commit_related*	24,0	19,3	19,1	**31,0**	23,0	**20,2**	0,17
21	***binary_pr_related***	23,5	22,5	22,4	**28,5**	**27,5**	**21,9**	**0,34**
22	***binary_issue_related***	25,5	**25,9**	**24,9**	15,0	17,7	16,4	**0,30**
23	*O_issue_commented*	22,0	21,2	19,3	15,0	13,0	12,3	0,06
24	*code_2_comment*	22,5	19,8	17,9	27,5	22,2	19,4	0,11
25	*Sun_metric*	8,0	7,6	7,8	14,5	11,5	9,9	0,04

Renklendirme Dereceleri: **1.** **2.** **3.**

Tablo 6.6, topluluk ilişkisi yaklaşımının değerlendirilmesi ile içerikten-bağımsız ve içeriğe-bağlı benzerlik yöntemine göre hit skor sonuçlarını göstermektedir. Sonuçlar analiz edildiğinde, *pr_opened* MRR puanına göre en başarılı olan metriktir. *PR*, geliştiricinin doğrudan katkıda bulunduğu projeleri gösterdiği için, proje önerisinin *PR* oluşturmaya dayalı modellenmesinin sistemin başarısını artırdığı düşünülmüştür. Ayrıca, füzyon metriklerinden *comments*, *binary_pr_related*, *binary_issue_related* ve *binary_comments* de dikkat çeken diğer metriklerdir. Tüm bu metriklerin ortak özelliği, hepsinin yorumlama ile ilgili olmasıdır. Bu bağlamda, bu sonuçlar OSS platformlarında tartışmanın veya yorum yapmanın önemini de açıkça göstermektedir.



(a) İçerikten-bağımsız benzerlik yöntemi



(b) İçeriğe-bağlı benzerlik yöntemi

Şekil 6.7 En başarılı 5 geliştirici metriğine ait tüm skorlar (TİY)

Şekil 6.7'de en başarılı 5 metriği ait tüm ilk-*n* skorları verilmiştir.

Tablo 6.7 Dil Deneyimi yaklaşımıyla geliştirici metriklerinin hit skorları

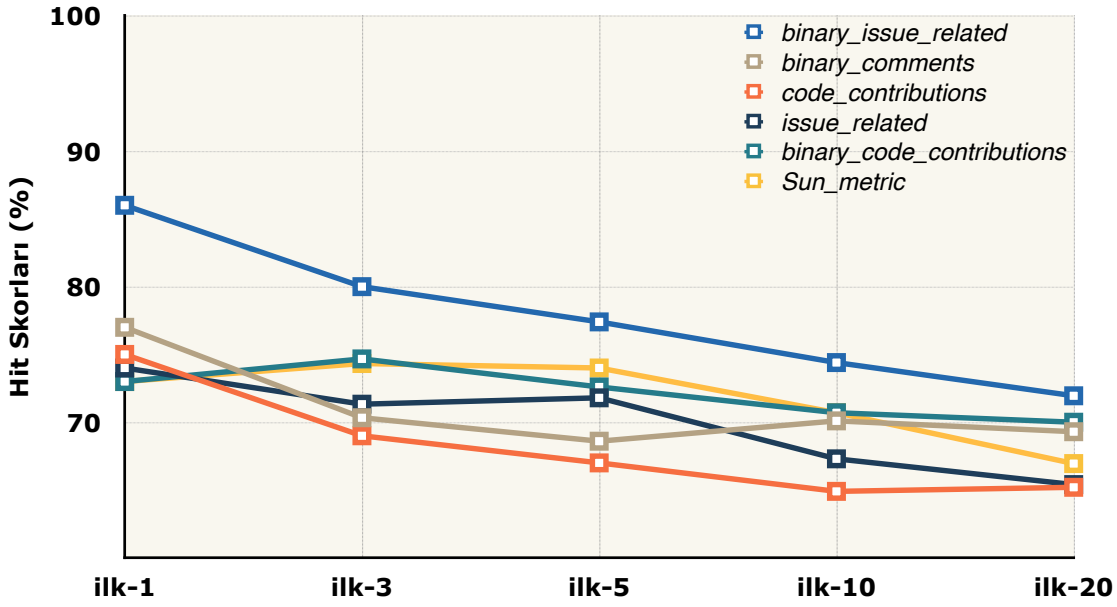
Dil Deneyimi Değerlendirme Yaklaşımı

	Geliştirici Metrikleri	İçerik (bağımsız) ilk1	ilk5	ilk10	İçerik (bağlı) ilk1	ilk5	ilk10	MRR
1	*commit_authored*	23,0	35,6	36,3	65,0	58,8	57,0	0,05
2	*commit_commented*	74,0	57,6	51,9	72,0	66,6	63,9	0,10
3	*commit_committed*	23,0	35,6	35,3	65,0	57,6	55,7	0,04
4	*issue_assigned*	64,0	52,2	53,8	59,0	57,8	53,3	0,05
5	*issue_closed*	68,0	59,6	55,4	50,0	49,4	46,4	0,05
6	*issue_hasPR*	72,0	62,6	61,1	**73,0**	70,8	69,0	0,09
7	*issue_commented*	66,0	66,8	65,6	**75,1**	70,6	70,2	0,12
8	*issue_opened*	68,0	67,6	66,4	**75,0**	**71,8**	70,7	0,14
9	*pr_assigned*	44,0	26,6	20,5	36,0	30,0	29,8	0,04
10	*pr_commented*	48,0	47,8	48,0	60,0	61,4	60,1	0,05
11	*pr_merged*	**74,0**	50,4	45,0	65,0	62,2	60,3	0,08
12	*pr_opened*	70,0	61,8	61,1	70,0	69,2	68,3	0,08
13	*comments*	67,0	63,4	63,7	**76,0**	**72,0**	70,6	0,18
14	*code_contributions*	**75,0**	67,0	64,9	70,0	70,4	70,8	0,15
15	*commit_related*	69,0	61,8	60,1	72,0	67,6	64,5	0,07
16	*pr_related*	63,0	62,8	64,0	69,0	71,2	67,6	0,08
17	*issue_related*	**74,0**	**71,8**	**67,3**	**76,0**	71,4	69,6	0,20
18	***binary_comments***	**77,0**	**68,6**	**70,1**	75,0	**74,4**	**75,2**	**0,43**
19	***binary_code_contributions***	73,0	**72,6**	**70,7**	**76,0**	**73,4**	**74,3**	**0,30**
20	*binary_commit_related*	72,0	64,6	60,4	70,0	69,2	65,9	0,08
21	***binary_pr_related***	70,0	67,0	**66,5**	**78,0**	**77,4**	**74,4**	**0,45**
22	***binary_issue_related***	**86,0**	**77,4**	**74,4**	75,0	72,6	**75,0**	**0,65**
23	*code_2_comment*	52,0	61,0	61,2	65,0	70,2	68,9	0,07
24	*O_issue_hasPR*	**73,0**	55,8	51,2	60,0	56,6	57,2	0,06
25	***Sun_metric***	**73,0**	**74,0**	**70,7**	68,0	71,4	**72,0**	**0,22**

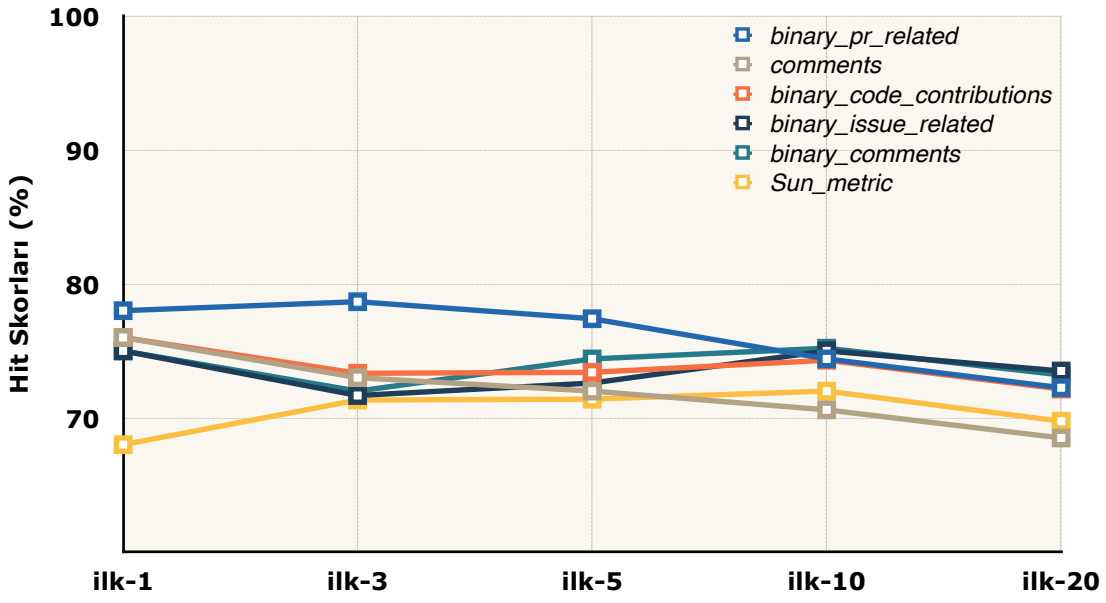
Renklendirme Dereceleri: **1.** **2.** **3.**

Tablo 6.7, dil deneyimi yaklaşımının değerlendirilmesi ile içerikten-bağımsız ve içeriğe-bağlı benzerlik yöntemine göre *hit* skor sonuçlarını göstermektedir. Bu yaklaşımda, *hit* skorları beklenildiği gibi ilk yaklaşıma göre daha yüksek çıkmıştır. Model, topluluk yaklaşımında, yaklaşık 40.000 proje arasından ilk-n projeyi önermeye çalışırken, deneyim yaklaşımında sanki 94 farklı proje arasında öneri sunmaya çalışıyor gibi düşünülebilir. Bu yaklaşımın sonucu olarak, ikili füzyon metrikleri açık bir şekilde öne çıkmıştır. Ayrıca, füzyon metriklerin çoğu, tekil olanlardan daha iyi sonuçlar vermiştir. Bunun sebebi olarak, füzyon metriklerin geliştirici ile ilgili birçok özelliği temsil etmesi ve bu sayede, geliştiricinin uzmanlaştığı

dillerde geliştirilen projelerin daha iyi keşfedildiği düşünülmüştür. Son olarak, bu yaklaşımda, *binary_issue_related* füzyon metrikler arasında en başarılı metrik olarak göze çarpmaktadır. Böylece, proje yönetim operasyonlarının birincil fonksiyonu olan *issue*'nun, geliştirici-proje korelasyonu açısından en önemli süreçlerden biri olduğu da gösterilmiştir. Bu değerlendirme yönteminde, *Sun_metric* sonuçları da, önceki yaklaşımın aksine dikkat çekici değerler vermiştir.



(a) İçerikten-bağımsız benzerlik yöntemi



(b) İçeriğe-bağlı benzerlik yöntemi

Şekil 6.8 En başarılı 5 geliştirici metriğine ait tüm skorlar (DDY)

Şekil 6.8'de en başarılı 5 metriği ait tüm ilk-*n* skorları verilmiştir.

Tablo 6.6 ve Tablo 6.7 birlikte analiz edildiğinde, ikili metriklerin oldukça başarılı olduğu görülmüştür. Ayrıca, yorum tabanlı metriklerin kod etkinliğine dayalı metriklerden daha yüksek skorlar elde ettiği de dikkate değer bir çıkarım olarak verilebilir.

Bu tez kapsamında üretilen geliştirici metrikleri ve proje öneri sistemi uygulaması bir dergi makalesi olarak yayınlanmıştır.

7
SONUÇLAR VE ÖNERİLER

Tez kapsamında ortaya atılan hipotezler doğrultusunda 3 ana hedefe ulaşmak amaçlanmıştır. İlk olarak, GitHub verilerini içeren ve literatürdeki çalışmaları birbiri ile adil bir şekilde kıyaslayabilmek için ortak kullanılması önerilen ve açık kaynak olarak paylaşılan bir veri kümesi üretilmesi hedeflenmiştir. Ayrıca, çeşitli yazılım mühendisliği problemlerinin çözümü için farklı geliştirici metrikleri önerilmesi de tezin bir diğer ana hedefidir. Son olarak, yazılım mühendisliğinin önemli problemlerinden biri olan proje öneri sisteminin önerilen bu metrikler ile geliştirilmesi amaçlanmıştır.

Literatürde yazılım mühendisliğine odaklanan çalışmaların çoğu, erişim kolaylığı, veri miktarı ve özellik çeşitliliği nedeniyle çalışmalarında veri kaynağı olarak GitHub platformunu kullanmaktadır. Bu bağlamda, tez kapsamında yapılan ilk iş, GitHub verilerini kullanan çalışmalar üzerinden bir SLR hazırlamaktır. Yapılan SLR'de, çalışmaların üst verilerinden keşfettiğimiz bilgiler doğrultusunda, çalışmalar 4 üst başlık ve 16 zorluk olarak sınıflandırılmıştır. Bu sayede, özellikle OSS geliştirme zorluklarıyla ilgilenen araştırmacılara açıkca kategorize edilmiş bir değerlendirme sunulmuştur. İncelen çalışmalara göre, araştırmacılar genellikle proje ve geliştirici karakterizasyon zorluğuna odaklanmaktadırlar. Ayrıca, geliştiricilerin geçmiş etkinliklerini ve birbirleriyle olan ilişkilerini (GitHub veya diğer sosyal ağlarda) analiz etmek literatürde odaklanılmış bir diğer zorluk olarak görülmüştür.

Literatürde incelenen çalışmalarda, GitHub veri kümelerinin çalışmalara özgü ve paylaşılmayan alt parçaları ile kullanıldığı görülmüştür. Veri kümelerini bu şekilde kullanmak çalışmaların birbiri ile adil şekilde kıyaslanmasına engel olmaktadır. Tez kapsamında, ortak bir veri kümesi üretmenin literatüre önemli bir katkı sağlayacağı hipotezi öne sürülmüştür. Bu doğrultuda, öncelikle en sık kullanılan GitHub veri kümesi olan GHTorrent'in avantajları, kullanımı, içerik açısından bazı problemleri ve eksiklikleri ortaya çıkarılmıştır. Veri kümesini kullanan çalışmaların çoğunda veri kümesinin özelleştirildiği, her çalışmaya özgü filtreler ile özel (paylaşılmayan) alt veri kümeleri kullanılması ortaya atılan hipotezi doğrulamıştır.

Sonuç olarak, hem GHTorrent veri kümesinin var olan problemlerini gideren, hem de farklı yazılım mühendisliği problemlerinde ortak kullanılabilecek, erişimi ve indirmesi kolay, hızlı uygulama için imkan verecek kadar küçük boyutta ve bir o kadar da kapsamlı bilgiler içeren bir veri kümesi üretilmiştir. GitDataSCP adı verilen bu veri kümesi farklı formatlarda paylaşılarak, araştırmacıların hizmetine sunulmuştur.

GitHub platformunda en yaygın kullanılan özellikler ile dağıtık kod geliştirme süreçlerinde yürütülen aktivitelerden elde edilen bilgiler, çeşitli yazılım mühendisliği zorluklarını çözmek için geliştirici metriği olarak kullanılmaktadır. Bu tez kapsamında, bu özelliklerden *issue*, *commit* ve *pull request (PR)* üzerinden yürütülen aktivitelerin geliştirici ve proje başına kullanım oranları incelenmiştir. Yapılan inceleme sonucunda bazı özelliklerin ve etkinliklerin daha yoğun kullanıldığı görülmüştür. En çok kullanılan özellik *commit*, aktivite ise *commit* etmek ve *issue*'ya yorum yapmak olarak ortaya çıkarılmıştır.

Bir aktivitenin daha çok yapılmasının, onun önemi ile doğru orantılı olup olmadığı önemli bir sorudur. Bu kapsamda, gerçek dünya yazılım geliştiricileri açısından bu aktivitelerin önemini ölçmek için bir anket çalışması yapılmıştır. Katılımcıların tamamına yakınının OSS geliştirme hakkında bilgi birikimine sahip olduğu verdikleri cevaplar ile kanıtlanmıştır. Bu anket sonucunda, GitHub platformunda diğerlerine göre çok daha az kullanılmasına rağmen katılımcılar tarafından önemli olduğu vurgulanan bazı aktiviteler olduğu ortaya çıkarılmıştır. Katılımcılar, kullanım oranının aksine, bir *issue*'ya *PR* bağlamayı, *issue* incelemeyi, *PR* birleştirmeyi, *PR* hakkında yorum yapmayı önemli aktiviteler olarak gördüklerini belirtmişlerdir. Böylece, aktivitelerin nicel değerlerinin önemli olmayabileceği hipotezi de sektörde kod geliştiren veya proje yöneten kişiler tarafından doğrulanmıştır. Tüm bu aktivitelerin geliştirici-proje ilişkisine odaklanan birçok problem için geliştirici metriği olarak kullanılabileceği sonucuna varılmıştır.

GitDataSCP veri kümesi ile GitHub geliştiricilerinin farklı aktivitelerinden geliştirici metrikleri üretilmiştir. Bir aktivitenin yapılma sayısı, bazı aktivitelerinin birbirleri arasındaki oranlar ve bir aktivitenin sadece var olma durumu tekil geliştirici metriklerinin üretiminde kullanılmıştır. Ayrıca, belirli özelliklerine ve türüne göre bazı aktiviteler bir araya getirilerek füzyon geliştirici metrikleri de oluşturulmuştur. Tüm bu aktivitelerden, dikkat çekici ve iyileştirilebilir 40 farklı geliştirici metriği önerilmiştir.

Bu metriklerin değerlendirilmesi için, bir kullanıcının geçmişte etkileşimde bulunduğu depolara benzer projeler öneren işbirlikçi filtrelemeye dayalı bir ilk-*n* proje öneri sistemi geliştirilmiştir. Metrikler, bu problem üzerinde başarılı sonuçlar vermiştir. Bir öneri sisteminde, temel gerçek, kullanıcıların ürünlere verdiği gerçek oylardır. GitHub proje öneri sistemi için bunun gibi ortak bir değerlendirme verisi yoktur. Bu sebeple, tez kapsamında bir hipotez olarak da öne sürülen, proje öneri sistemini değerlendirmek için topluluk ilişkisi ve dil deneyimi olarak tanımlanan iki yeni yaklaşım önerilmiştir. Ayrıca, proje benzerliklerinin bulunması için metrikleri (içerikten-bağımsız) ve projenin tüm süreçlerindeki yorumları (içeriğe-bağlı) kullanan iki benzerlik yöntemi kullanılmıştır.

Önerilen tekil geliştirici metrikleri arasında *PR* açma, *issue*'ya *PR* bağlama, *issue* için yorum yapma metrikleri öne çıkmıştır. Bu bağlamda, geliştiriciler hakkında farklı çıkarımlar yapabilmek için bu metriklerin tek başlarına bile değerli oldukları düşünülmektedir. Böylece, az bilgi ile elde edilen metriklerin de katkı sağlayabileceği hipotezi doğrulanmıştır. Ayrıca, tez kapsamında yorumlar hakkında ortaya atılan hipotezi de doğrulayan sonuçlar elde edilmiştir. Bir projede yorum yapma aktivitesinin kod yazmak kadar değerli olabileceği ortaya çıkmıştır. Füzyon metriklerden öne çıkanların çoğunun çeşitli süreçlerde yorum yapma aktiviteleri ile alakalı metrikler olduğu görülmüştür.

Tezin hipotezlerinden biri olan aktivitelerin nicel değerlerinin öneminin olmayabileceği, önerilen ikili (0-1) metriklerin başarısı ile bir kez daha doğrulanmıştır. Yapılan aktivitelerin sadece varlık-yokluk durumlarını inceleyen ikili metrikler arasında *issue* ile ilgili aktivitelerin birleşiminden oluşan metrik ve tüm süreçlerdeki yorumlardan oluşan metrik, diğer metriklerin çoğuna üstünlük sağlamıştır. Özellikle, *issue* ve *PR* ile ilişkili metrikler nicel değerinden bağımsız metrikler olarak ortaya çıkmıştır.

Tez kapsamında ortaya atılan bir diğer hipotez ise, oransal bilgiler ile üretilen metriklerin daha başarılı olacağı yönündeydi. Ancak, sonuçlara bakıldığında, bu hipotezin tersi bir durum ortaya çıkmıştır. Tekil metriklerin birbiri ile oranlarından elde edilen füzyon metrikler ve bir geliştiricinin yaptığı bir aktivitenin, o aktivitenin ilgili projedeki toplam adedine oranını baz alan optimize metriklerin çoğu başarısız olmuştur. Bunun temel nedeni, GitHub geliştiricilerinin aktif bir şekilde çok az sayıda proje ile ilişkili olması gösterilebilir. Buna bağlı olarak, oransal metriklerin çok küçük ve birbirine yakın değerlerden (çoğu 0'a yakın) oluşmasından dolayı benzer projeler doğru tespit edilememiş olabilir.

Tekil metriklerin belirli özelliklerine göre gruplandırılmasında elde edilecek füzyon metriklerin başarılı olacağı hipotezi de proje öneri sistemi sonuçlarına göre doğrulanmıştır. Füzyon metriklerin çoğu tekil metriklere göre daha yüksek *hit* skorlar elde etmiştir.

Tez bünyesinde üretilen GitDataSCP veri kümesinin, farklı yazılım geliştirme problemlerinin çözümünde kullanılması hedeflenmektedir. Bu tez kapsamında edinilen birikim ile gelecek çalışmalarda, proje benzerliklerini bulmak için anlamsal kelime temsil yöntemlerinin kullanılması planlanmaktadır. Ayrıca, önerilen metriklerin, farklı veri kümeleri üzerinde aynı problem için uygulanılması da düşünülmektedir. Bu problem dışında, proje sahipliğinin keşfi, projelerde katkı sağlayan tüm geliştiricilerin projeye katkı oranlarının hesaplanması, yapılan aktivitelerin programlama dillerine göre karakteristiklerinin çıkarılması gibi farklı problemler üzerinde de denenmesi planlanmaktadır.

KAYNAKÇA

[1] A. A. Khan, J. Keung, *Systematic review of success factors and barriers for software process improvement in global software development*, Oct. 2016. DOI: 10.1049/iet-sen.2015.0038.

[2] R. R. Schreiber, M. P. Zylka, *Social Network Analysis in Software Development Projects: A Systematic Literature Review*, Mar. 2020. DOI: 10.1142/S021819402050014X.

[3] V. Cosentino, J. Luis, J. Cabot, "Findings from GitHub: methods, datasets and limitations," in *Proceedings of the 13th International Workshop on Mining Software Repositories*, New York, New York, USA: ACM Press, 2016, pp. 137–141, ISBN: 9781450341868. DOI: 10.1145/2901739.2901776. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2901739.2901776.

[4] A. S. Badashian, V. Shah, E. Stroulia, "GitHub's big data adaptor: an eclipse plugin," in *CASCON 15*, 2015, pp. 265–268. [Online]. Available: https://dl.acm.org/citation.cfm?id=2886490.

[5] V. Cosentino, J. L. Canovas Izquierdo, J. Cabot, "A Systematic Mapping Study of Software Development With GitHub," *IEEE Access*, vol. 5, pp. 7173–7192, 2017, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2682323. [Online]. Available: http://ieeexplore.ieee.org/document/7887704/.

[6] Z. Kotti, D. Spinellis, "Standing on shoulders or feet?: the usage of the MSR data papers," in *Proceedings of the 16th International Conference on Mining Software Repositories*, IEEE Press, 2019, pp. 565–576. DOI: 10.1109/MSR.2019.00085. [Online]. Available: https://dl.acm.org/citation.cfm?id=3341974.

[7] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," *Journal of Systems and Software*, vol. 80, no. 4, pp. 571–583, 2007, ISSN: 01641212. DOI: 10.1016/j.jss.2006.07.009.

[8] G. Gousios, "The GHTorrent dataset and tool suite," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, ser. MSR '13, Piscataway, NJ, USA: IEEE Press, 2013, pp. 233–236, ISBN: 978-1-4673-2936-1. DOI: 10.1109/MSR.2013.6624034. [Online]. Available: http://dl.acm.org/citation.cfm?id=2487085.2487132.

[9] "Three Metrics to Explore the Openness of GitHub projects," *arXiv preprint arXiv:1409.4253*, Sep. 2014. arXiv: 1409.4253. [Online]. Available: http://arxiv.org/abs/1409.4253.

[10] M. M. Rahman, C. K. Roy, "An insight into the pull requests of GitHub," in *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, New York, New York, USA: ACM Press, 2014, pp. 364–367, ISBN: 9781450328630. DOI: 10.1145/2597073.2597121. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2597073.2597121.

[11] Y. Saito, K. Fujiwara, H. Igaki, N. Yoshida, H. Iida, "How do GitHub Users Feel with Pull-Based Development?" İn *2016 7th International Workshop on Empirical Software Engineering in Practice (IWESEP)*, IEEE, Mar. 2016, pp. 7–11, ISBN: 978-1-5090-1851-2. DOI: 10.1109/IWESEP.2016.19. [Online]. Available: http://ieeexplore.ieee.org/document/7464545/.

[12] S. Horschig, T. Mattis, R. Hirschfeld, "Do Java programmers write better Python? Studying off-language code quality on GitHub," in *Conference Companion of the 2nd International Conference on Art, Science, and Engineering of Programming - Programming'18 Companion*, New York, New York, USA: ACM Press, 2018, pp. 127–134, ISBN: 9781450355131. DOI: 10.1145/3191697.3214341. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3191697.3214341.

[13] J. Sheoran, K. Blincoe, E. Kalliamvakou, D. Damian, J. Ell, "Understanding watchers on GitHub," in *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, New York, New York, USA: ACM Press, 2014, pp. 336–339, ISBN: 9781450328630. DOI: 10.1145/2597073.2597114. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2597073.2597114.

[14] Y. Yu, G. Yin, H. Wang, T. Wang, "Exploring the patterns of social behavior in GitHub," in *Proceedings of the 1st International Workshop on Crowd-based Software Development Methods and Technologies - CrowdSoft 2014*, New York, New York, USA: ACM Press, 2014, pp. 31–36, ISBN: 9781450332248. DOI: 10.1145/2666539.2666571. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2666539.2666571.

[15] G. Silvestri, J. Yang, A. Bozzon, A. Tagarelli, "Linking Accounts across Social Networks: the Case of StackOverflow, Github and Twitter," in *KDWeb*, 2015. [Online]. Available: https://www.semanticscholar.org/paper/Linking-Accounts-across-Social-Networks%7B%5C%%7D3A-the-Case-Silvestri-Yang/351b86ffc19cb02e51466522a0b4b199ac1dbd06.

[16] G. B. Alves, M. A. Brandão, D. M. Santana, A. P. C. da Silva, M. M. Moro, "The Strength of Social Coding Collaboration on GitHub," in *SBBD 2016*, 2016. [Online]. Available: https://www.semanticscholar.org/paper/The-Strength-of-Social-Coding-Collaboration-on-Alves-Brand%7B%5C~%7Ba%7D%7Do/15dc574702e4e61233e04b1e8ea3f5ad38dc0cc6.

[17] R. Abdalkareem, E. Shihab, J. Rilling, "What Do Developers Use the Crowd For? A Study Using Stack Overflow," *IEEE Software*, vol. 34, no. 2, pp. 53–60, Mar. 2017. DOI: 10.1109/MS.2017.31. [Online]. Available: http://ieeexplore.ieee.org/document/7888410/.

[18] N. A. Batista, M. A. Brandão, G. B. Alves, A. P. C. da Silva, M. M. Moro, "Collaboration strength metrics and analyses on GitHub," in *Proceedings of the International Conference on Web Intelligence - WI '17*, New York, New York, USA: ACM Press, 2017, pp. 170–178, ISBN: 9781450349512. DOI: 10.1145/3106426.3106480. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3106426.3106480.

[19] F. Calefato, F. Lanubile, N. Novielli, "A Preliminary Analysis on the Effects of Propensity to Trust in Distributed Software Development," in *2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)*, IEEE, May 2017, pp. 56–60, ISBN: 978-1-5386-1587-4. DOI: 10.1109/ICGSE.2017.1. [Online]. Available: http://ieeexplore.ieee.org/document/7976688/.

[20] T. Komamizu, Y. Hayase, T. Amagasa, H. Kitagawa, "Exploring Identical Users on GitHub and Stack Overflow," in *SEKE*, Jul. 2017, pp. 584–589. DOI: 10.18293/SEKE2017-109. [Online]. Available: http://ksiresearchorg.ipage.com/seke/seke17paper/seke17paper%7B%5C_%7D109.pdf.

[21] D.-C. Yan, B.-H. Wang, "Collaborative similarity analysis of multilayer developer-project bipartite network," *arXiv preprint arXiv:1703.03093*, Mar. 2017. arXiv: 1703.03093. [Online]. Available: http://arxiv.org/abs/1703.03093.

[22] D. Yang, P. Martins, V. Saini, C. Lopes, "Stack Overflow in Github: Any Snippets There?" İn *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, IEEE, May 2017, pp. 280–290, ISBN: 978-1-5386-1544-7. DOI: 10.1109/MSR.2017.13. [Online]. Available: http://ieeexplore.ieee.org/document/7962378/.

[23] J. Liu, Z. Li, T. Wang, Y. Yu, G. Yin, "Adaptive software search toward users' customized requirements in GitHub," in *SEKE*, Jul. 2018, pp. 143–181. DOI: 10.18293/SEKE2018-064. [Online]. Available: http://ksiresearchorg.ipage.com/seke/seke18paper/seke18paper%7B%5C_%7D64.pdf.

[24] E. Kopczyński, D. Celińska-Kopczyńska, "Hyperbolic triangulations and discrete random graphs," *arXiv preprint arXiv:1707.01124*, Jul. 2017. arXiv: 1707.01124. [Online]. Available: http://arxiv.org/abs/1707.01124.

[25] A. S. Badashian, A. Hindle, E. Stroulia, "Crowdsourced bug triaging," in *2015 IEEE 31st International Conference on Software Maintenance and Evolution, ICSME 2015 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., Nov. 2015, pp. 506–510. DOI: 10.1109/ICSM.2015.7332503. [Online]. Available: https://ieeexplore.ieee.org/document/7332503/.

[26] M. L. De Lima, D. M. Soares, A. Plastino, L. Murta, M. L. de Lima Júnior, D. M. Soares, A. Plastino, L. Murta, "Developers assignment for analyzing pull requests," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing - SAC '15*, vol. 13-17-Apri, New York, New York, USA, Apr. 2015, pp. 1567–1572, ISBN: 9781450331968. DOI: 10.1145/2695664.2695884. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2695664.2695884.

[27] C. Hauff, G. Gousios, "Matching GitHub Developer Profiles to Job Advertisements," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, IEEE, May 2015, pp. 362–366, ISBN: 978-0-7695-5594-2. DOI: 10.1109/MSR.2015.41. [Online]. Available: http://ieeexplore.ieee.org/document/7180095/.

[28] P. Brokmeier, "Project level effects of gender on contribution evaluation on GitHub," *PeerJ PrePrints*, vol. 5, no. e2989v1, May 2017. DOI: 10.7287/peerj.preprints.2989v1. [Online]. Available: https://peerj.com/preprints/2989/.

[29] J. Terrell, A. Kofink, J. Middleton, C. Rainear, E. Murphy-Hill, C. Parnin, J. Stallings, "Gender differences and bias in open source: pull request acceptance of women versus men," *PeerJ Computer Science*, vol. 3, e111, May 2017. DOI: 10.7717/peerj-cs.111. [Online]. Available: https://peerj.com/articles/cs-111.

[30] S. Bayati, "Understanding newcomers success in open source community," in *Proceedings of the 40th International Conference on Software Engineering Companion Proceeedings - ICSE '18*, New York, New York, USA: ACM Press, 2018, pp. 224–225, ISBN: 9781450356633. DOI: 10.1145/3183440.3195073. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3183440.3195073.

[31] N. Cassee, G. Pinto, F. Castor, A. Serebrenik, "How swift developers handle errors," in *Proceedings of the 15th International Conference on Mining Software Repositories - MSR '18*, New York, New York, USA: ACM Press, 2018, pp. 292–302, ISBN: 9781450357166. DOI: 10.1145/3196398.3196428. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3196398.3196428.

[32] M. Ortu, A. Pinna, R. Tonelli, M. Marchesi, D. Bowes, G. Destefanis, "Angry-builds: An Empirical Study Of Affect Metrics and Builds Success on GitHub Ecosystem," in *Proceedings of the 19th International Conference on Agile Software Development Companion - XP '18*, New York, New York, USA: ACM Press, 2018, pp. 1–2, ISBN: 9781450364225. DOI: 10.1145/3234152.3234160. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3234152.3234160.

[33] M. Ortu, T. Hall, M. Marchesi, R. Tonelli, D. Bowes, G. Destefanis, "Mining Communication Patterns in Software Development: A GitHub Analysis," in *Proceedings of the 14th International Conference on Predictive Models and Data Analytics in Software Engineering - PROMISE'18*, New York, New York, USA: ACM Press, 2018, pp. 70–79, ISBN: 9781450365932. DOI: 10.1145/3273934.3273943. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3273934.3273943.

[34] S. Saha, G. M. Muradul Bashir, M. Raihan Talukder, J. Karmaker, M. Saiful Islam, "Which Programming Language and Platform Developers Prefer for the Development? A Study Using Stack Overflow," in *2018 International Conference on Innovations in Science, Engineering and Technology (ICISET)*, IEEE, Oct. 2018, pp. 305–310, ISBN: 978-1-5386-8524-2. DOI: 10.1109/ICISET.

2018.8745630. [Online]. Available: https://ieeexplore.ieee.org/document/8745630/.

[35] Y. Lu, X. Mao, T. Wang, G. Yin, Z. Li, W. Wang, "Studying in the "Bazaar": an Exploratory Study of Crowdsourced Learning in GitHub," *IEEE Access*, pp. 1–1, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2915247. [Online]. Available: https://ieeexplore.ieee.org/document/8708224/.

[36] H. S. Qiu, A. Nolte, A. Brown, A. Serebrenik, B. Vasilescu, "Going farther together: the impact of social capital on sustained participation in open source," in *41st ACM/IEEE International Conference on Software Engineering, (ICSE2019)*, IEEE Computer Society, May 2019, pp. 688–699. [Online]. Available: https://research.tue.nl/en/publications/going-farther-together-the-impact-of-social-capital-on-sustained-.

[37] B. Vasilescu, V. Filkov, A. Serebrenik, "StackOverflow and GitHub: Associations between Software Development and Crowdsourced Knowledge," in *2013 International Conference on Social Computing*, IEEE, Sep. 2013, pp. 188–195, ISBN: 978-0-7695-5137-1. DOI: 10.1109/SocialCom.2013.35. [Online]. Available: http://ieeexplore.ieee.org/document/6693332/.

[38] A. S. Badashian, A. Esteki, A. Gholipour, A. Hindle, E. Stroulia, "Involvement, contribution and influence in GitHub and stack overflow," in *Proceedings of 24th Annual International Conference on Computer Science and Software Engineering*, IBM Corp., 2014, pp. 19–33. [Online]. Available: https://dl.acm.org/citation.cfm?id=2735527.

[39] E. Constantinou, G. M. Kapitsaki, "Identifying Developers' Expertise in Social Coding Platforms," in *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, IEEE, Aug. 2016, pp. 63–67, ISBN: 978-1-5090-2820-7. DOI: 10.1109/SEAA.2016.18. [Online]. Available: http://ieeexplore.ieee.org/document/7592778/.

[40] J. Jiang, F. Feng, X. Lian, L. Zhang, "Long-Term Active Integrator Prediction in the Evaluation of Code Contributions," in *International conference on software engineering and knowledge engineering (SEKE)*, 2016, pp. 177–182. DOI: 10.18293/SEKE2016-030. [Online]. Available: http://developer.github.com/v3/.

[41] E. Constantinou, T. Mens, "An empirical comparison of developer retention in the RubyGems and npm software ecosystems," *Innovations in Systems and Software Engineering*, vol. 13, no. 2-3, pp. 101–115, Sep. 2017. DOI: 10.1007/s11334-017-0303-4. [Online]. Available: http://link.springer.com/10.1007/s11334-017-0303-4.

[42] Y. Hu, S. Wang, Y. Ren, K.-K. R. Choo, "User influence analysis for Github developer social networks," *Expert Systems with Applications*, vol. 108, pp. 108–118, Oct. 2018, ISSN: 0957-4174. DOI: 10.1016/J.ESWA.2018.05.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417418302793.

[43] S. Baltes, J. Knack, D. Anastasiou, R. Tymann, S. Diehl, "(No) influence of continuous integration on the commit activity in GitHub projects," in *Proceedings of the 4th ACM SIGSOFT International Workshop on Software Analytics - SWAN 2018*, New York, New York, USA: ACM Press, 2018, pp. 1–7, ISBN: 9781450360562. DOI: 10.1145/3278142.3278143. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3278142.3278143.

[44] G. Destefanis, M. Ortu, D. Bowes, M. Marchesi, R. Tonelli, "On measuring affects of github issues' commenters," in *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering - SEmotion '18*, New York, New York, USA, Jun. 2018, pp. 14–19, ISBN: 9781450357517. DOI: 10.1145/3194932.3194936. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3194932.3194936.

[45] R. K.-W. Lee, D. Lo, "Wisdom in Sum of Parts: Multi-Platform Activity Prediction in Social Collaborative Sites," in *Proceedings of the 10th ACM Conference on Web Science - WebSci '18*, New York, New York, USA: ACM Press, 2018, pp. 77–86, ISBN: 9781450355636. DOI: 10.1145/3201064.3201067. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3201064.3201067.

[46] S. S. Manes, O. Baysal, "How often and what StackOverflow posts do developers reference in their GitHub projects?" İn *Proceedings of the 16th International Conference on Mining Software Repositories*, IEEE Press, 2019, pp. 235–239. DOI: 10.1109/MSR.2019.00047. [Online]. Available: https://dl.acm.org/citation.cfm?id=3341926.

[47] J. Xavier, A. Macedo, M. d. A. Maia, "Understanding the popularity of reporters and assignees in the Github," in *International conference on software engineering and knowledge engineering (SEKE)*, 2014, pp. 484–489. [Online]. Available: https://www.semanticscholar.org/paper/Understanding-the-popularity-of-reporters-and-in-Xavier-Macedo/a113516ff1ca5ff4ebdbb4a2b90c972158cc3763.

[48] O. Jarczyk, S. Jaroszewicz, A. Wierzbicki, K. Pawlak, M. Jankowski-Lorek, "Surgical teams on GitHub: Modeling performance of GitHub project development processes," *Information and Software Technology*, vol. 100, pp. 32–46, Aug. 2018, ISSN: 0950-5849. DOI: 10.1016/J.INFSOF.2018.03.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095058491730304X.

[49] M. L. d. L. Júnior, D. M. Soares, A. Plastino, L. Murta, "Automatic assignment of integrators to pull requests: The importance of selecting appropriate attributes," *Journal of Systems and Software*, vol. 144, pp. 181–196, Oct. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121218301122?via%7B%5C%%7D3Dihub%7B%5C#%7Dbib0014.

[50] J. Brunet, G. C. Murphy, R. Terra, J. Figueiredo, D. Serey, "Do developers discuss design?" İn *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, New York, New York, USA: ACM Press, 2014, pp. 340–343, ISBN: 9781450328630. DOI: 10.1145/2597073.2597115.

[Online]. Available: http://dl.acm.org/citation.cfm?doid=2597073.2597115.

[51] G. Burlet, A. Hindle, "An Empirical Study of End-User Programmers in the Computer Music Community," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, IEEE, May 2015, pp. 292–302, ISBN: 978-0-7695-5594-2. DOI: 10.1109/MSR.2015.34. [Online]. Available: http://ieeexplore.ieee.org/document/7180088/.

[52] M. Sayagh, N. Kerzazi, B. Adams, F. Petrillo, "Software Configuration Engineering in Practice: Interviews, Survey, and Systematic Literature Review," *IEEE Transactions on Software Engineering*, pp. 1–1, 2018. DOI: 10.1109/TSE.2018.2867847. [Online]. Available: https://ieeexplore.ieee.org/document/8451922/.

[53] B. Vasilescu, V. Filkov, A. Serebrenik, "Perceptions of diversity on GitHub: a user survey," in *Proceedings of the 8th International Workshop on Cooperative and Human Aspects of Software Engineering*, IEEE Press, 2015, pp. 50–56. [Online]. Available: https://dl.acm.org/citation.cfm?id=2819330.

[54] K. Yamashita, S. McIntosh, Y. Kamei, A. E. Hassan, N. Ubayashi, "Revisiting the applicability of the pareto principle to core development teams in open source software projects," in *Proceedings of the 14th International Workshop on Principles of Software Evolution - IWPSE 2015*, New York, New York, USA: ACM Press, 2015, pp. 46–55, ISBN: 9781450338165. DOI: 10.1145/2804360.2804366. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2804360.2804366.

[55] T. Jaruchotrattanasakul, X. Yang, E. Makihara, K. Fujiwara, H. Iida, "Open Source Resume (OSR): A Visualization Tool for Presenting OSS Biographies of Developers," in *2016 7th International Workshop on Empirical Software Engineering in Practice (IWESEP)*, IEEE, Mar. 2016, pp. 57–62, ISBN: 978-1-5090-1851-2. DOI: 10.1109/IWESEP.2016.17. [Online]. Available: http://ieeexplore.ieee.org/document/7464554/.

[56] D.-C. Yan, Z.-W. Wei, X.-P. Han, B.-H. Wang, "Empirical analysis on the human dynamics of blogging behavior on GitHub," *Physica A: Statistical Mechanics and its Applications*, vol. 465, pp. 775–781, Jan. 2017, ISSN: 0378-4371. DOI: 10.1016/J.PHYSA.2016.08.054. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378437116305714.

[57] S. Baltes, S. Diehl, "Usage and attribution of Stack Overflow code snippets in GitHub projects," *Empirical Software Engineering*, pp. 1–37, Oct. 2018, ISSN: 1382-3256. DOI: 10.1007/s10664-018-9650-5. [Online]. Available: http://link.springer.com/10.1007/s10664-018-9650-5.

[58] ——, "Towards a theory of software development expertise," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering - ESEC/FSE 2018*, New York, New York, USA: ACM Press, 2018, pp. 187–200, ISBN: 9781450355735. DOI: 10.1145/3236024.3236061. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3236024.3236061.

[59] C. Liu, D. Yang, X. Zhang, B. Ray, M. M. Rahman, "Recommending GitHub Projects for Developer Onboarding," *IEEE Access*, vol. 6, pp. 52 082–52 094, Sep. 2018, ISSN: 21693536. DOI: 10.1109/ACCESS.2018.2869207.

[60] P. Zhang, F. Xiong, H. K. Leung, W. Song, "FunkR-pDAE: Personalized Project Recommendation Using Deep Learning," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2018. DOI: 10.1109/TETC.2018.2870734. [Online]. Available: https://ieeexplore.ieee.org/document/8466656/.

[61] W. Wisse, C. Veenman, "Scripting DNA: Identifying the JavaScript programmer," *Digital Investigation*, vol. 15, pp. 61–71, Dec. 2015. DOI: 10.1016/J.DIIN.2015.09.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1742287615000973.

[62] K. Blincoe, J. Sheoran, S. Goggins, E. Petakovic, D. Damian, "Understanding the popular users: Following, affiliation influence and leadership on GitHub," *Information and Software Technology*, vol. 70, pp. 30–39, Feb. 2016, ISSN: 0950-5849. DOI: 10.1016/J.INFSOF.2015.10.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584915001688.

[63] A. Rastogi, N. Nagappan, "On the Personality Traits of GitHub Contributors," in *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, IEEE, Oct. 2016, pp. 77–86, ISBN: 978-1-4673-9002-6. DOI: 10.1109/ISSRE.2016.43. [Online]. Available: http://ieeexplore.ieee.org/document/7774509/.

[64] D.-C. Yan, M. Li, B.-H. Wang, "Dependence centrality similarity: Measuring the diversity of profession levels of interests," *Physica A: Statistical Mechanics and its Applications*, vol. 479, pp. 118–127, Aug. 2017, ISSN: 0378-4371. DOI: 10.1016/J.PHYSA.2017.02.082. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378437117301656.

[65] J. Middleton, E. Murphy-Hill, D. Green, A. Meade, R. Mayer, D. White, S. McDonald, "Which contributions predict whether developers are accepted into github teams," in *Proceedings of the 15th International Conference on Mining Software Repositories - MSR '18*, New York, New York, USA: ACM Press, 2018, pp. 403–413, ISBN: 9781450357166. DOI: 10.1145/3196398.3196429. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3196398.3196429.

[66] M. M. Sun, A. Ghosh, R. Sharma, S. K. Kuttal, "Birds of a Feather Flock Together? A Study of Developers' Flocking and Migration Behavior in GitHub and Stack Overflow," *arXiv preprint arXiv:1810.13062*, Oct. 2018. arXiv: 1810.13062. [Online]. Available: http://arxiv.org/abs/1810.13062.

[67] Y. Yu, H. Wang, G. Yin, C. X. Ling, "Who Should Review this Pull-Request Reviewer Recommendation to Expedite Crowd Collaboration," in *2014 21st Asia-Pacific Software Engineering Conference*, IEEE, Dec. 2014, pp. 335–342, ISBN: 978-1-4799-7425-2. DOI: 10.1109/APSEC.2014.57. [Online]. Available: http://ieeexplore.ieee.org/document/7091328/.

[68] H. Ying, L. Chen, T. Liang, J. Wu, "EARec: Leveraging Expertise and Authority for Pull-Request Reviewer Recommendation in GitHub," in *Proceedings of the 3rd International Workshop on CrowdSourcing in Software Engineering - CSI-SE '16*, New York, New York, USA: ACM Press, 2016, pp. 29–35, ISBN: 9781450341585. DOI: 10.1145/2897659.2897660. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2897659.2897660.

[69] Y. Yu, H. Wang, G. Yin, T. Wang, "Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment?" *Information and Software Technology*, vol. 74, pp. 204–218, Jun. 2016, ISSN: 0950-5849. DOI: 10.1016/J.INFSOF.2016.01.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584916000069%20https://www.sciencedirect.com/science/article/abs/pii/S0950584916000069.

[70] C. Casalnuovo, B. Vasilescu, P. Devanbu, V. Filkov, "Developer Onboarding in GitHub: The Role of Prior Social Links and Language Experience," in *2015 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015 - Proceedings*, New York, New York, USA: Association for Computing Machinery, Inc, Aug. 2015, pp. 817–828, ISBN: 9781450336758. DOI: 10.1145/2786805.2786854. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2786805.2786854.

[71] R. Nielek, O. Jarczyk, K. Pawlak, L. Bukowski, R. Bartusiak, A. Wierzbicki, "Choose a Job You Love: Predicting Choices of GitHub Developers," in *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, 2016, pp. 200–207. DOI: 10.1109/WI.2016.0037. [Online]. Available: http://ieeexplore.ieee.org/document/7817054/.

[72] E. Constantinou, G. M. Kapitsaki, "Developers Expertise and Roles on Software Technologies," in *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*, IEEE, 2016, pp. 365–368, ISBN: 978-1-5090-5575-3. DOI: 10.1109/APSEC.2016.061. [Online]. Available: http://ieeexplore.ieee.org/document/7890613/.

[73] Z. Wang, Y. Wang, D. Redmiles, "Competence-confidence gap: a threat to female developers' contribution on github," in *Proceedings of the 40th International Conference on Software Engineering Software Engineering in Society - ICSE-SEIS '18*, New York, New York, USA: ACM Press, 2018, pp. 81–90, ISBN: 9781450356619. DOI: 10.1145/3183428.3183437. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3183428.3183437.

[74] J. Jiang, D. Lo, Y. Yang, J. Li, L. Zhang, "A first look at unfollowing behavior on GitHub," *Information and Software Technology*, vol. 105, pp. 150–160, Jan. 2019. DOI: 10.1016/J.INFSOF.2018.08.012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095058491830185X.

[75] L. Bao, X. Xia, D. Lo, G. C. Murphy, "A Large Scale Study of Long-Time Contributor Prediction for GitHub Projects," *IEEE Transactions on Software Engineering*, pp. 1–1, 2019. DOI: 10.1109/TSE.2019.2918536. [Online]. Available: https://ieeexplore.ieee.org/document/8721092/.

[76] S. Bayati, "Effect of newcomers supportive strategies on open source projects socio-technical activities," in *Proceedings of the 12th International Workshop on Cooperative and Human Aspects of Software Engineering CHASE*, IEEE Press, 2019, pp. 49–50. DOI: 10.1109/CHASE.2019.00020. [Online]. Available: https://dl.acm.org/citation.cfm?id=3338731.

[77] D. M. Soares, M. L. de Lima Júnior, A. Plastino, L. Murta, "What factors influence the reviewer assignment to pull requests?" *Information and Software Technology*, vol. 98, pp. 32–43, Jun. 2018. DOI: 10.1016/J.INFSOF.2018.01.015. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0950584917303804.

[78] E. Guzman, D. Azócar, Y. Li, "Sentiment Analysis of Commit Comments in GitHub: An Empirical Study," New York, New York, USA, May 2014, pp. 352–355, ISBN: 9781450328630. DOI: 10.1145/2597073.2597118. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2597073.2597118.

[79] A. Murgia, G. Concas, R. Tonelli, M. Ortu, S. Demeyer, M. Marchesi, "On the influence of maintenance activity types on the issue resolution time," in *Proceedings of the 10th International Conference on Predictive Models in Software Engineering - PROMISE '14*, New York, New York, USA, 2014, pp. 12–21, ISBN: 9781450328982. DOI: 10.1145/2639490.2639506. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2639490.2639506.

[80] D. Pletea, B. Vasilescu, A. Serebrenik, "Security and emotion: sentiment analysis of security discussions on GitHub," in *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, New York, New York, USA: ACM Press, 2014, pp. 348–351, ISBN: 9781450328630. DOI: 10.1145/2597073.2597117. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2597073.2597117.

[81] J. Cabot, J. L. Canovas Izquierdo, V. Cosentino, B. Rolandi, "Exploring the use of labels to categorize issues in Open-Source Software projects," in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, IEEE, Mar. 2015, pp. 550–554, ISBN: 978-1-4799-8469-5. DOI: 10.1109/SANER.2015.7081875. [Online]. Available: http://ieeexplore.ieee.org/document/7081875/.

[82] R. Kikas, M. Dumas, D. Pfahl, "Using dynamic and contextual features to predict issue lifetime in GitHub projects," in *Proceedings of the 13th International Workshop on Mining Software Repositories - MSR '16*, New York, New York, USA: ACM Press, 2016, pp. 291–302, ISBN: 9781450341868. DOI: 10.1145/2901739.2901751. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2901739.2901751.

[83] D. Li, L. Li, D. Kim, T. F. Bissyandé, D. Lo, Y. L. Traon, "Watch out for This Commit! A Study of Influential Software Changes," *arXiv preprint arXiv:1606.03266*, Jun. 2016. arXiv: 1606.03266. [Online]. Available: http://arxiv.org/abs/1606.03266.

[84] X. Liu, B. Shen, H. Zhong, J. Zhu, "EXPSOL: Recommending Online Threads for Exception-Related Bug Reports," in *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*, IEEE, 2016, pp. 25–32, ISBN: 978-1-5090-5575-3. DOI: 10.1109/APSEC.2016.015. [Online]. Available: http://ieeexplore.ieee.org/document/7890567/.

[85] Q. Fan, Y. Yu, G. Yin, T. Wang, H. Wang, "Where Is the Road for Issue Reports Classification Based on Text Mining?" İn *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, IEEE, Nov. 2017, pp. 121–130, ISBN: 978-1-5090-4039-1. DOI: 10.1109/ESEM.2017.19. [Online]. Available: http://ieeexplore.ieee.org/document/8170092/.

[86] D. Hu, T. Wang, J. Chang, Y. Zhang, G. Yin, "Bugs and features, do developers treat them differently?" İn *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*, IEEE, May 2018, pp. 250–255, ISBN: 978-1-5386-6987-7. DOI: 10.1109/ICAIBD.2018.8396204. [Online]. Available: https://ieeexplore.ieee.org/document/8396204/.

[87] K. Werder, S. Brinkkemper, "MEME - Toward a Method for EMotions Extraction from GitHub," in *Emotion 2018 : 2018 ACM/IEEE 3rd International Workshop on Emotion Awareness in Software Engineering*, 2018, ISBN: 9781450357517. [Online]. Available: https://ieeexplore.ieee.org/document/8595354/keywords%7B%5C#%7Dkeywords.

[88] J. M. Alonso-Abad, C. López-Nozal, J. M. Maudes-Raedo, R. Marticorena-Sánchez, "Label prediction on issue tracking systems using text mining," *Progress in Artificial Intelligence*, vol. 8, no. 3, pp. 325–342, Mar. 2019, ISSN: 21926360. DOI: 10.1007/s13748-019-00182-2. [Online]. Available: https://doi.org/10.1007/s13748-019-00182-2%20http://link.springer.com/10.1007/s13748-019-00182-2.

[89] R.-M. Karampatsis, C. Sutton, "How Often Do Single-Statement Bugs Occur? The ManySStuBs4J Dataset," *arXiv preprint arXiv:1905.13334*, May 2019. arXiv: 1905.13334. [Online]. Available: http://arxiv.org/abs/1905.13334.

[90] B. Vasilescu, A. Serebrenik, V. Filkov, "A Data Set for Social Diversity Studies of GitHub Teams," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, IEEE, May 2015, pp. 514–517, ISBN: 978-0-7695-5594-2. DOI: 10.1109/MSR.2015.77. [Online]. Available: http://ieeexplore.ieee.org/document/7180131/.

[91] M. El Mezouar, F. Zhang, Y. Zou, "An empirical study on the teams structures in social coding using GitHub projects," *Empirical Software Engineering*, pp. 1–34, May 2019. DOI: 10.1007/s10664-019-09700-1. [Online]. Available: http://link.springer.com/10.1007/s10664-019-09700-1.

[92] M. Gharehyazie, B. Ray, V. Filkov, "Some from Here, Some from There: Cross-Project Code Reuse in GitHub," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, IEEE, May 2017, pp. 291–301, ISBN: 978-1-5386-1544-7. DOI: 10.1109/MSR.2017.15. [Online]. Available: http://ieeexplore.ieee.org/document/7962379/.

[93] K. Aggarwal, A. Hindle, E. Stroulia, "Co-evolution of project documentation and popularity within github," in *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, New York, New York, USA: ACM Press, 2014, pp. 360–363, ISBN: 9781450328630. DOI: 10.1145/2597073.2597120. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2597073.2597120.

[94] S. Onoue, H. Hata, K. Matsumoto, "Software population pyramids: The Current and the Future of OSS Development Communities," in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '14*, New York, New York, USA: ACM Press, 2014, pp. 1–4, ISBN: 9781450327749. DOI: 10.1145/2652524.2652565. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2652524.2652565.

[95] K. Yamashita, S. McIntosh, Y. Kamei, N. Ubayashi, "Magnet or sticky? an OSS project-by-project typology," in *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, New York, New York, USA: ACM Press, 2014, pp. 344–347, ISBN: 9781450328630. DOI: 10.1145/2597073.2597116. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2597073.2597116.

[96] E. Knauss, D. Damian, J. Cleland-Huang, R. Helms, "Patterns of continuous requirements clarification," *Requirements Engineering*, vol. 20, no. 4, pp. 383–403, Nov. 2015. DOI: 10.1007/s00766-014-0205-z. [Online]. Available: http://link.springer.com/10.1007/s00766-014-0205-z.

[97] W. Wang, G. Poo-Caamano, E. Wilde, D. M. German, "What Is the Gist? Understanding the Use of Public Gists on GitHub," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, IEEE, May 2015, pp. 314–323, ISBN: 978-0-7695-5594-2. DOI: 10.1109/MSR.2015.36. [Online]. Available: http://ieeexplore.ieee.org/document/7180090/.

[98] A. Eck, F. Uebernickel, "Reconstructing Open Source Software Ecosystems: Finding Structure in Digital Traces," in *ICIS*, 2016. [Online]. Available: https://www.semanticscholar.org/paper/Reconstructing-Open-Source-Software-Ecosystems%7B%5C%%7D3A-in-Eck-Uebernickel/60ace7d37a292da6b40e0ac468b326f2e0f524af.

[99] A. Rastogi, N. Nagappan, "Forking and the Sustainability of the Developer Community Participation – An Empirical Investigation on Outcomes and Reasons," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, IEEE, Mar. 2016, pp. 102–111, ISBN: 978-1-5090-1855-0. DOI: 10.1109/SANER.2016.27. [Online]. Available: http://ieeexplore.ieee.org/document/7476634/.

[100] M. A. Fernández, G. Robles, G. /. Libresoft, U. Rey, J. Carlos, "Extracting software development information from FLOSS Projects in GitHub," in *SATToSE*, 2017. [Online]. Available: http://ghtorrent.org/.

[101] G. Robles, T. Ho-Quang, R. Hebig, M. R. Chaudron, M. A. Fernandez, "An Extensive Dataset of UML Models in GitHub," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, IEEE, May 2017, pp. 519–522, ISBN: 978-1-5386-1544-7. DOI: 10.1109/MSR.2017.48. [Online]. Available: http://ieeexplore.ieee.org/document/7962411/.

[102] E. Constantinou, T. Mens, "Socio-technical evolution of the Ruby ecosystem in GitHub," in *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, IEEE, Feb. 2017, pp. 34–44, ISBN: 978-1-5090-5501-2. DOI: 10.1109/SANER.2017.7884607. [Online]. Available: http://ieeexplore.ieee.org/document/7884607/.

[103] S. Onoue, R. G. Kula, H. Hata, K. Matsumoto, "The Health and Wealth of OSS Projects: Evidence from Community Activities and Product Evolution," *arXiv preprint arXiv:1709.10324*, Sep. 2017. arXiv: 1709.10324. [Online]. Available: http://arxiv.org/abs/1709.10324.

[104] C. Cheng, B. Li, Z. Li, P. Liang, "Automatic Detection of Public Development Projects in Large Open Source Ecosystems: An Exploratory Study on GitHub," *arXiv preprint arXiv:1803.03175*, Mar. 2018. DOI: 10.18293/SEKE2018-085. arXiv: 1803.03175. [Online]. Available: http://arxiv.org/abs/1803.03175.

[105] V. Kovalenko, F. Palomba, A. Bacchelli, "Mining file histories: should we consider branches?" İn *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering - ASE 2018*, New York, New York, USA: ACM Press, 2018, pp. 202–213, ISBN: 9781450359375. DOI: 10.1145/3238147.3238169. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3238147.3238169.

[106] Y. Ma, S. Fakhoury, M. Christensen, V. Arnaoudova, W. Zogaan, M. Mirakhorli, "Automatic classification of software artifacts in open-source applications," in *Proceedings of the 15th International Conference on Mining Software Repositories - MSR '18*, New York, New York, USA: ACM Press, 2018, pp. 414–425, ISBN: 9781450357166. DOI: 10.1145/3196398.3196446. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3196398.3196446.

[107] F. Miranda, L. Lins, J. T. Klosowski, C. T. Silva, "TopKube: A Rank-Aware Data Cube for Real-Time Exploration of Spatiotemporal Data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 3, pp. 1394–1407, Mar. 2018. DOI: 10.1109/TVCG.2017.2671341. [Online]. Available: http://ieeexplore.ieee.org/document/7858782/.

[108] A. Raghuraman, T. Ho-Quang, M. R. V Chaudron Chalmers, A. Serebrenik, B. Vasilescu, "Does UML Modeling Associate with Lower Defect Proneness?: A Preliminary Empirical Investigation," in *16th International Conference on Mining Software Repositories*, 2019. [Online]. Available: https://pypi.org/project/langdetect/.

[109] M. Gharehyazie, B. Ray, M. Keshani, M. S. Zavosht, A. Heydarnoori, V. Filkov, "Cross-project code clones in GitHub," *Empirical Software Engineering*, vol. 24, no. 3, pp. 1538–1573, Jun. 2019. DOI: 10.1007/s10664-018-9648-z. [Online]. Available: http://link.springer.com/10.1007/s10664-018-9648-z.

[110] C. S. Lakkundi, V. Agrahari, S. Chimalakonda, "GE852: A Dataset of 852 Game Engines," *arXiv preprint arXiv:1905.04482*, May 2019. arXiv: 1905.04482. [Online]. Available: http://arxiv.org/abs/1905.04482.

[111] M. Ortu, G. Destefanis, S. Counsell, S. Swift, R. Tonelli, M. Marchesi, "How diverse is your team? Investigating gender and nationality diversity in GitHub teams," *Journal of Software Engineering Research and Development*, vol. 5, no. 1, p. 9, Dec. 2017. DOI: 10.1186/s40411-017-0044-y. [Online]. Available: https://jserd.springeropen.com/articles/10.1186/s40411-017-0044-y.

[112] J. L. C. Izquierdo, V. Cosentino, B. Rolandi, A. Bergel, J. Cabot, "GiLA: GitHub label analyzer," in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering, SANER 2015 - Proceedings*, 2015, ISBN: 9781479984695. DOI: 10.1109/SANER.2015.7081860.

[113] K. Blincoe, F. Harrison, N. Kaur, D. Damian, "Reference Coupling: An exploration of inter-project technical dependencies and their characteristics within large software ecosystems," *Information and Software Technology*, vol. 110, pp. 174–189, Jun. 2019, ISSN: 0950-5849. DOI: 10.1016/J.INFSOF.2019.03.005. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0950584919300527.

[114] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, D. Damian, "The promises and perils of mining GitHub," in *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, New York, New York, USA: ACM Press, 2014, pp. 92–101, ISBN: 9781450328630. DOI: 10.1145/2597073.2597074. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2597073.2597074.

[115] N. Matragkas, J. R. Williams, D. S. Kolovos, R. F. Paige, "Analysing the 'biodiversity' of open source ecosystems: the GitHub case," in *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, New York, New York, USA: ACM Press, 2014, pp. 356–359, ISBN: 9781450328630. DOI: 10.1145/2597073.2597119. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2597073.2597119.

[116] J. F. Low, T. Yathog, D. Svetinovic, "Software analytics study of Open-Source system survivability through social contagion," in *2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, IEEE, Dec. 2015, pp. 1213–1217, ISBN: 978-1-4673-8066-9. DOI: 10.1109/IEEM.2015.7385840. [Online]. Available: http://ieeexplore.ieee.org/document/7385840/.

[117] E. Constantinou, T. Mens, "Social and technical evolution of software ecosystems: A Case Study of Rails," in *Proccedings of the 10th European Conference on Software Architecture Workshops - ECSAW '16*, New York, New York, USA: ACM Press, 2016, pp. 1–4, ISBN: 9781450347815. DOI: 10.1145/2993412.3003384. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2993412.3003384.

[118] K. Werder, "The evolution of emotional displays in open source software development teams: An Individual Growth Curve Analysis.," in *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering - SEmotion '18*, New York, New York, USA: ACM Press, 2018, pp. 1–6, ISBN: 9781450357517. DOI: 10.1145/3194932.3194934. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3194932.3194934.

[119] R. Padhye, S. Mani, V. S. Sinha, "A study of external community contribution to open-source projects on GitHub," in *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, New York, New York, USA: ACM Press, 2014, pp. 332–335, ISBN: 9781450328630. DOI: 10.1145/2597073.2597113. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2597073.2597113.

[120] H. Hata, T. Todo, S. Onoue, K. Matsumoto, "Characteristics of Sustainable OSS Projects: A Theoretical and Empirical Study," in *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering*, IEEE, May 2015, pp. 15–21, ISBN: 978-1-4673-7031-8. DOI: 10.1109/CHASE.2015.9. [Online]. Available: http://ieeexplore.ieee.org/document/7166083/.

[121] J. Aue, M. Haisma, K. F. Tomasdottir, A. Bacchelli, "Social Diversity and Growth Levels of Open Source Software Projects on GitHub," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '16*, New York, New York, USA: ACM Press, 2016, pp. 1–6, ISBN: 9781450344272. DOI: 10.1145/2961111.2962633. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2961111.2962633.

[122] K. Yamashita, Y. Kamei, S. McIntosh, A. E. Hassan, N. Ubayashi, "Magnet or Sticky? Measuring Project Characteristics from the Perspective of Developer Attraction and Retention," *Journal of Information Processing*, vol. 24, no. 2, pp. 339–348, 2016. DOI: 10.2197/ipsjjip.24.339. [Online]. Available: https://www.jstage.jst.go.jp/article/ipsjjip/24/2/24%7B%5C_%7D339/%7B%5C_%7Darticle.

[123] Z. Liao, N. Wang, S. Liu, Y. Zhang, H. Liu, Q. Zhang, "Identification-Method Research for Open-Source Software Ecosystems," *Symmetry*, vol. 11, no. 2, p. 182, Feb. 2019. DOI: 10.3390/sym11020182. [Online]. Available: http://www.mdpi.com/2073-8994/11/2/182.

[124] K. Blincoe, F. Harrison, D. Damian, "Ecosystems in GitHub and a Method for Ecosystem Identification Using Reference Coupling," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, IEEE, May 2015, pp. 202–211, ISBN: 978-0-7695-5594-2. DOI: 10.1109/MSR.2015.26. [Online]. Available: http://ieeexplore.ieee.org/document/7180080/.

[125] Y. Zhang, G. Yin, Y. Yu, H. Wang, "Investigating social media in GitHub's pull-requests: a case study on Ruby on Rails," in *Proceedings of the 1st International Workshop on Crowd-based Software Development Methods and Technologies - CrowdSoft 2014*, New York, New York, USA: ACM Press, 2014, pp. 37–41, ISBN: 9781450332248. DOI: 10.1145/2666539.2666572. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2666539.2666572.

[126] D. M. Soares, M. L. de Lima Júnior, L. Murta, A. Plastino, "Acceptance factors of pull requests in open-source projects," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing - SAC '15*, New York, New York, USA: ACM Press, 2015, pp. 1541–1546, ISBN: 9781450331968. DOI: 10.1145/2695664.2695856. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2695664.2695856.

[127] D. M. Soares, M. L. d. L. Junior, L. Murta, A. Plastino, "Rejection Factors of Pull Requests Filed by Core Team Developers in Software Projects with High Acceptance Rates," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, IEEE, Dec. 2015, pp. 960–965, ISBN: 978-1-5090-0287-0. DOI: 10.1109/ICMLA.2015.41. [Online]. Available: http://ieeexplore.ieee.org/document/7424445/.

[128] D. Chen, K. T. Stolee, T. Menzies, "Replicating and Scaling up Qualitative Analysis using Crowdsourcing: A Github-based Case Study," *arXiv preprint arXiv:1702.08571*, Feb. 2017. arXiv: 1702.08571. [Online]. Available: http://arxiv.org/abs/1702.08571.

[129] R. Coelho, L. Almeida, G. Gousios, A. Van Deursen, "Unveiling exception handling bug hazards in android based on GitHub and Google code issues," in *IEEE International Working Conference on Mining Software Repositories*, 2015, ISBN: 9780769555942. DOI: 10.1109/MSR.2015.20.

[130] R. Coelho, L. Almeida, G. Gousios, A. van Deursen, C. Treude, "Exception handling bug hazards in Android Results from a mining study and an exploratory survey," *Empirical Software Engineering*, vol. 22, no. 3, pp. 1264–1304, Jun. 2017. DOI: 10.1007/s10664-016-9443-7. [Online]. Available: http://link.springer.com/10.1007/s10664-016-9443-7.

[131] E. Wittern, P. Suter, S. Rajagopalan, "A look at the dynamics of the JavaScript package ecosystem," in *Proceedings of the 13th International Conference on Mining Software Repositories*, 2016, pp. 351–361. DOI: 10.1145/2901739.2901743.

[132] D. Celińska, E. Kopczyński, "Programming Languages in GitHub: A Visualization in Hyperbolic Plane," in *11. International AAAI Conference on Web and Social Media*, May 2017. [Online]. Available: https://www.aaai.org/ocs/index.php/ICWSM/ICWSM17/paper/viewPaper/15583.

[133] C. V. Lopes, P. Maj, P. Martins, V. Saini, D. Yang, J. Zitny, H. Sajnani, J. Vitek, "DéjàVu: a map of code duplicates on GitHub," *Proceedings of the ACM on Programming Languages*, vol. 1, no. OOPSLA, pp. 1–28, Oct. 2017. DOI: 10.1145/3133908. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3152284.3133908.

[134] D. Gonzalez, S. Prentice, M. Mirakhorli, "A fine-grained approach for automated conversion of JUnit assertions to English," in *Proceedings of the 4th ACM SIGSOFT International Workshop on NLP for Software Engineering - NL4SE 2018*, New York, New York, USA: ACM Press, 2018, pp. 14–17, ISBN: 9781450360555. DOI: 10.1145/3283812.3283819. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3283812.3283819.

[135] P. Martins, R. Achar, C. V. Lopes, "50K-C a dataset of compilable, and compiled, Java projects," in *Proceedings of the 15th International Conference on Mining Software Repositories - MSR '18*, New York, New York, USA: ACM Press, 2018, pp. 1–5, ISBN: 9781450357166. DOI: 10.1145/3196398.3196450. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3196398.3196450.

[136] M. Hilton, T. Tunnell, K. Huang, D. Marinov, D. Dig, "Usage, costs, and benefits of continuous integration in open-source projects," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering - ASE 2016*, New York, New York, USA: ACM Press, 2016, pp. 426–437, ISBN: 9781450338455. DOI: 10.1145/2970276.2970358. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2970276.2970358.

[137] M. Beller, G. Gousios, A. Zaidman, "TravisTorrent: Synthesizing Travis CI and GitHub for Full-Stack Research on Continuous Integration," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, IEEE, May 2017, pp. 447–450, ISBN: 978-1-5386-1544-7. DOI: 10.1109/MSR.2017.24. [Online]. Available: http://ieeexplore.ieee.org/document/7962393/.

[138] L. Madeyski, M. Kawalerowicz, "Continuous Defect Prediction: The Idea and a Related Dataset," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, IEEE, May 2017, pp. 515–518, ISBN: 978-1-5386-1544-7. DOI: 10.1109/MSR.2017.46. [Online]. Available: http://ieeexplore.ieee.org/document/7962410/.

[139] J. Xia, Y. Li, "Could We Predict the Result of a Continuous Integration Build? An Empirical Study," in *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, IEEE, Jul. 2017, pp. 311–315, ISBN: 978-1-5386-2072-4. DOI: 10.1109/QRS-C.2017.59. [Online]. Available: http://ieeexplore.ieee.org/document/8004336/.

[140] C. Vassallo, S. Proksch, H. C. Gall, M. Di Penta, "Automated reporting of anti-patterns and decay in continuous integration," in *Proceedings of the 41st International Conference on Software Engineering ICSE*, IEEE Press, 2019, pp. 105–115. DOI: 10.1109/ICSE.2019.00028. [Online]. Available: https://dl.acm.org/citation.cfm?id=3339520.

[141] E. Van Der Veen, G. Gousios, A. Zaidman, E. van der Veen, G. Gousios, A. Zaidman, "Automatically Prioritizing Pull Requests," in *2015 12th Working Conference on Mining Software Repositories*, vol. 2015-Augus, 2015, pp. 357–361. DOI: 10.1109/MSR.2015.40. [Online]. Available: http://ieeexplore.ieee.org/document/7180094/.

[142] F. Medeiros, G. Lima, G. Amaral, S. Apel, C. Kästner, M. Ribeiro, R. Gheyi, "An investigation of misunderstanding code patterns in C open-source software projects," *Empirical Software Engineering*, pp. 1–34, Nov. 2018. DOI: 10.1007/s10664-018-9666-x. [Online]. Available: http://link.springer.com/10.1007/s10664-018-9666-x.

[143] H. Hata, C. Treude, R. G. Kula, T. Ishio, "9.6 Million Links in Source Code Comments: Purpose, Evolution, and Decay," *International Conference on Software Engineering*, Jan. 2019. arXiv: 1901.07440. [Online]. Available: http://arxiv.org/abs/1901.07440.

[144] Y. Yu, G. Yin, T. Wang, C. Yang, H. Wang, "Determinants of pull-based development in the context of continuous integration," *Science China Information Sciences*, vol. 59, no. 8, p. 080 104, Aug. 2016. DOI: 10.1007/s11432-016-5595-8. [Online]. Available: http://link.springer.com/10.1007/s11432-016-5595-8.

[145] G. Zhao, D. A. da Costa, Y. Zou, "Improving the Pull Requests Review Process Using Learning-to-rank Algorithms," *Empirical Software Engineering*, pp. 1–31, 2019. [Online]. Available: https://danielcalencar.github.io/journal%20papers/2019/05/01/emse-19-Guoliang.html.

[146] W. Muylaert, W. D. Meuter, C. D. Roover, "An Exploratory Study Into the Prevalence of Botched Code Integrations," in *SATToSE*, 2016. [Online]. Available: http://sattose.wdfiles.com/local--files/2016:alltalks/SATTOSE2016%7B%5C_%7Dpaper%7B%5C_%7D8.pdf.

[147] P. Dimitropoulos, Z. Aung, D. Svetinovic, "Continuous integration build breakage rationale: Travis data case study," in *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)*, IEEE, Dec. 2017, pp. 639–645, ISBN: 978-1-5386-0514-1. DOI: 10.1109/ICTUS.2017.8286087. [Online]. Available: http://ieeexplore.ieee.org/document/8286087/.

[148] Y. Luo, Y. Zhao, W. Ma, L. Chen, "What are the Factors Impacting Build Breakage?" İn *2017 14th Web Information Systems and Applications Conference (WISA)*, IEEE, Nov. 2017, pp. 139–142, ISBN: 978-1-5386-4806-3. DOI: 10.1109/WISA.2017.17. [Online]. Available: http://ieeexplore.ieee.org/document/8332602/.

[149] W. Muylaert, C. De Roover, "Prevalence of Botched Code Integrations," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, IEEE, May 2017, pp. 503–506, ISBN: 978-1-5386-1544-7. DOI: 10.1109/MSR.2017.40. [Online]. Available: http://ieeexplore.ieee.org/document/7962407/.

[150] M. Beller, G. Gousios, A. Zaidman, "Oops, My Tests Broke the Build: An Explorative Analysis of Travis CI with GitHub," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, IEEE, May 2017, pp. 356–367, ISBN: 978-1-5386-1544-7. DOI: 10.1109/MSR.2017.62. [Online]. Available: http://ieeexplore.ieee.org/document/7962385/.

[151] D. Gonzalez, J. C. Santos, A. Popovich, M. Mirakhorli, M. Nagappan, "A Large-Scale Study on the Usage of Testing Patterns That Address Maintainability Attributes: Patterns for Ease of Modification, Diagnoses, and Comprehension," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, IEEE, May 2017, pp. 391–401, ISBN: 978-1-5386-1544-7. DOI: 10.1109/MSR.2017.8. [Online]. Available: http://ieeexplore.ieee.org/document/7962388/.

[152] T. Sharma, M. Fragkoulis, D. Spinellis, "Does your configuration code smell?" İn *Proceedings of the 13th International Workshop on Mining Software Repositories - MSR '16*, New York, New York, USA: ACM Press, 2016, pp. 189–200, ISBN: 9781450341868. DOI: 10.1145/2901739.2901761. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2901739.2901761.

[153] I. J. Mujhid, J. C. S. Santos, R. Gopalakrishnan, M. Mirakhorli, "A search engine for finding and reusing architecturally significant code," *Journal of Systems and Software*, vol. 130, pp. 81–93, Aug. 2017, ISSN: 0164-1212. DOI: 10.1016/J.JSS.2016.11.034. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121216302345.

[154] R. Kikas, G. Gousios, M. Dumas, D. Pfahl, "Structure and Evolution of Package Dependency Networks," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, IEEE, May 2017, pp. 102–112, ISBN: 978-1-5386-1544-7. DOI: 10.1109/MSR.2017.55. [Online]. Available: http://ieeexplore.ieee.org/document/7962360/.

[155] E. A. Santos, J. C. Campbell, A. Hindle, J. N. Amaral, "Finding and correcting syntax errors using recurrent neural networks," *PeerJ Preprints*, Aug. 2017. DOI: 10.7287/peerj.preprints.3123v1. [Online]. Available: https://peerj.com/preprints/3123/.

[156] F. Medeiros, M. Ribeiro, R. Gheyi, S. Apel, C. Kastner, B. Ferreira, L. Carvalho, B. Fonseca, "Discipline Matters: Refactoring of Preprocessor Directives in the #ifdef Hell," *IEEE Transactions on Software Engineering*, vol. 44, no. 5, pp. 453–469, May 2018. DOI: 10.1109/TSE.2017.2688333. [Online]. Available: https://ieeexplore.ieee.org/document/7888579/.

[157] K. F. Tomasdottir, M. Aniche, A. Van Deursen, "The Adoption of JavaScript Linters in Practice: A Case Study on ESLint," *IEEE Transactions on Software Engineering*, pp. 1–1, 2018. DOI: 10.1109/TSE.2018.2871058. [Online]. Available: https://ieeexplore.ieee.org/document/8468105/.

[158] B. Vasilescu, S. Van Schuylenburg, J. Wulms, A. Serebrenik, M. G. J. van den Brand, "Continuous Integration in a Social-Coding World: Empirical Evidence from GitHub," in *2014 IEEE International Conference on Software Maintenance and Evolution*, IEEE, Sep. 2014, pp. 401–405, ISBN: 978-1-4799-6146-7. DOI: 10.1109/ICSME.2014.62. [Online]. Available: http://ieeexplore.ieee.org/document/6976106/.

[159] B. Vasilescu, Y. Yu, H. Wang, P. Devanbu, V. Filkov, "Quality and productivity outcomes relating to continuous integration in GitHub," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, New York, New York, USA: ACM, 2015, pp. 805–816, ISBN: 978-1-4503-3675-8. DOI:

10.1145/2786805.2786850. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2786805.2786850.

[160] S. Urli, Z. Yu, L. Seinturier, M. Monperrus, "How to design a program repair bot? Insights from the Repairnator Project," in *Proceedings of the 40th International Conference on Software Engineering Software Engineering in Practice - ICSE-SEIP '18*, New York, New York, USA: ACM Press, 2018, pp. 95–104, ISBN: 9781450356596. DOI: 10.1145/3183519.3183540. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3183519.3183540.

[161] Y. Yu, H. Wang, V. Filkov, P. Devanbu, B. Vasilescu, "Wait for It: Determinants of Pull Request Evaluation Latency on GitHub," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, IEEE, May 2015, pp. 367–371, ISBN: 978-0-7695-5594-2. DOI: 10.1109/MSR.2015.42. [Online]. Available: http://ieeexplore.ieee.org/document/7180096/.

[162] *What is open source software?* [Online]. Available: https://opensource.com/resources/what-open-source (visited on 01/28/2021).

[163] M. Thomson, Stark B, "RFC 8874 - Working Group GitHub Usage Guidance," Internet Engineering Task Force (IETF), Tech. Rep., 2020. [Online]. Available: https://tools.ietf.org/html/rfc8874.

[164] F. Siebdrat, M. Hoegl, H. Ernst, "How to Manage Virtual Teams," *MIT Sloan Management Review*, vol. 50, no. 4, pp. 63–68, 2009.

[165] S. Chacon, B. Straub, *Pro git*. Springer Nature, 2014.

[166] M. Aranda, *Git Merge vs. Rebase: What's the Diff? | Hacker Noon*, Sep. 2017. [Online]. Available: https://hackernoon.com/git-merge-vs-rebase-whats-the-diff-76413c117333 (visited on 02/02/2021).

[167] GitHub, *Git Handbook - GitHub Guides*, Jun. 2020. [Online]. Available: https://guides.github.com/introduction/git-handbook/ (visited on 01/28/2021).

[168] Y. Perez-Riverol, L. Gatto, R. Wang, T. Sachsenberg, J. Uszkoreit, F. d. V. Leprevost, C. Fufezan, T. Ternent, S. J. Eglen, D. S. Katz, T. J. Pollard, A. Konovalov, R. M. Flight, K. Blin, J. A. Vizcaíno, "Ten Simple Rules for Taking Advantage of Git and GitHub," *PLOS Computational Biology*, vol. 12, no. 7, S. Markel, Ed., Jul. 2016, ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1004947. [Online]. Available: https://dx.plos.org/10.1371/journal.pcbi.1004947.

[169] V. Driessen, *A successful Git branching model » nvie.com*, Jan. 2010. [Online]. Available: https://nvie.com/posts/a-successful-git-branching-model/ (visited on 02/10/2021).

[170] I. Green, *Git 101- Part 2 (A bit More Advance)*, Aug. 2014. [Online]. Available: https://greenido.wordpress.com/tag/github/ (visited on 02/10/2021).

[171] GitHub, *Understanding the GitHub flow - GitHub Guides*, 2020. [Online]. Available: https://guides.github.com/introduction/flow/ (visited on 01/28/2021).

[172] L. Dabbish, C. Stuart, J. Tsay, J. Herbsleb, "Social coding in GitHub: Transparency and collaboration in an open software repository," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work, CSCW*, New York, New York, USA: ACM Press, 2012, pp. 1277–1286, ISBN: 9781450310864. DOI: 10.1145/2145204.2145396. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2145204.2145396.

[173] G. M. Alarcon, A. M. Gibson, C. Walter, R. F. Gamble, T. J. Ryan, S. A. Jessup, B. E. Boyd, A. Capiola, "Trust Perceptions of Metadata in Open-Source Software: The Role of Performance and Reputation," *Systems*, vol. 8, no. 3, p. 28, Aug. 2020, ISSN: 2079-8954. DOI: 10.3390/systems8030028. [Online]. Available: https://www.mdpi.com/2079-8954/8/3/28.

[174] G. Gousios, B. Vasilescu, A. Serebrenik, A. Zaidman, "Lean GHTorrent: GitHub data on demand," in *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, New York, New York, USA: ACM Press, 2014, pp. 384–387, ISBN: 9781450328630. DOI: 10.1145/2597073.2597126. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2597073.2597126.

[175] P. Thongtanunam, C. Tantithamthavorn, R. G. Kula, N. Yoshida, H. Iida, K.-i. I. Matsumoto, "Who should review my code? A file location-based code-reviewer recommendation approach for Modern Code Review," in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering, SANER 2015 - Proceedings*, 2015, pp. 141–150. DOI: 10.1109/SANER.2015.7081824. [Online]. Available: http://ieeexplore.ieee.org/document/7081824/.

[176] J. Tsay, L. Dabbish, J. Herbsleb, "Influence of social and technical factors for evaluating contribution in GitHub," in *Proceedings - International Conference on Software Engineering*, New York, New York, USA: IEEE Computer Society, May 2014, pp. 356–366. DOI: 10.1145/2568225.2568315. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2568225.2568315.

[177] Y. Yu, H. Wang, G. Yin, C. X. Ling, "Reviewer recommender of pull-requests in GitHub," in *Proceedings - 30th International Conference on Software Maintenance and Evolution, ICSME 2014*, Institute of Electrical and Electronics Engineers Inc., Dec. 2014, pp. 609–612, ISBN: 9780769553030. DOI: 10.1109/ICSME.2014.107.

[178] S. Ozcan Kini, A. Tosun, "Periodic developer metrics in software defect prediction," in *Proceedings - 18th IEEE International Working Conference on Source Code Analysis and Manipulation, SCAM 2018*, Institute of Electrical and Electronics Engineers Inc., Nov. 2018, pp. 72–81, ISBN: 9781538682906. DOI: 10.1109/SCAM.2018.00016.

[179] G. Kaur, K. Bahl, "Software Reliability, Metrics, Reliability Improvement Using Agile Process," *International Journal of Innovative Science, Engineering & Technology*, vol. 1, no. 3, pp. 143–147, 2014, ISSN: 2348-7968. [Online]. Available: www.ijiset.com.

[180] V. Tiwari, R. K. Pandey, *Open Source Software and Reliability Metrics*, 2012. [Online]. Available: https://pdfs.semanticscholar.org/ce3f/2129c3c735ce4e0d4ac048ea36f990903022.pdf%20www.ijarcce.com.

[181] C. Bird, N. Nagappan, B. Murphy, H. Gall, P. Devanbu, "Don't touch my code! Examining the effects of ownership on software quality," in *Proceedings of the 19th ACM SIGSOFT Symposium on Foundations of Software Engineering*, New York, New York, USA: ACM Press, 2011, pp. 4–14, ISBN: 9781450304436. DOI: 10.1145/2025113.2025119. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2025113.2025119.

[182] J. C. Munson, S. G. Elbaum, "Code churn: a measure for estimating the impact of code change," in *International Conference on Software Maintenance*, IEEE, 1998, pp. 24–31. DOI: 10.1109/icsm.1998.738486.

[183] M. Foucault, C. Teyton, D. Lo, X. Blanc, J. R. Falleri, "On the usefulness of ownership metrics in open-source software projects," in *Information and Software Technology*, vol. 64, Elsevier, Aug. 2015, pp. 102–112. DOI: 10.1016/j.infsof.2015.01.013.

[184] H. J. Happel, W. Maalej, "Potentials and challenges of recommendation systems for software development," in *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, New York, New York, USA: ACM Press, 2008, pp. 11–15, ISBN: 9781605582283. DOI: 10.1145/1454247.1454251. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1454247.1454251.

[185] M. Robillard, R. Walker, T. Zimmermann, "Recommendation systems for software engineering," *IEEE Software*, vol. 27, no. 4, pp. 80–86, Jul. 2010, ISSN: 07407459. DOI: 10.1109/MS.2009.161.

[186] L. Sharma, A. Gera, "A Survey of Recommendation System: Research Challenges," *International Journal of Engineering Trends and Technology*, vol. 4, no. 5, pp. 1989–1992, 2013.

[187] S. Onoue, H. Hata, K. I. Matsumoto, "A study of the characteristics of developers' activities in GitHub," in *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*, vol. 2, IEEE Computer Society, 2013, pp. 7–12, ISBN: 9781479921430. DOI: 10.1109/APSEC.2013.104.

[188] G. Jeong, S. Kim, T. Zimmermann, "Improving Bug Triage with Bug Tossing Graphs," in *Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, ser. ESEC/FSE '09, New York, NY, USA: ACM, 2009, pp. 111–120, ISBN: 978-1-60558-001-2. DOI: 10.1145/1595696.1595715. [Online]. Available: http://doi.acm.org/10.1145/1595696.1595715.

[189] X. Sun, W. Xu, X. Xia, X. Chen, B. Li, "Personalized project recommendation on GitHub," *Science China Information Sciences*, vol. 61, no. 5, pp. 1–14, May 2018, ISSN: 18691919. DOI: 10.1007/s11432-017-9419-x.

[190] G. Gousios, D. Spinellis, "GHTorrent: Github's data from a firehose," in *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*, IEEE, Jun. 2012, pp. 12–21, ISBN: 978-1-4673-1761-0. DOI: 10.1109/MSR.2012.6224294. [Online]. Available: http://ieeexplore.ieee.org/document/6224294/.

[191] V. Markovtsev, W. Long, "Public git archive: a big code dataset for all," in *15th International Conference on Mining Software Repositories*, ACM, 2018, pp. 34–37. [Online]. Available: https://dl.acm.org/citation.cfm?id=3196464.

[192] G. Gousios, A. Zaidman, "A dataset for pull-based development research," in *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, New York, New York, USA: ACM Press, 2014, pp. 368–371, ISBN: 9781450328630. DOI: 10.1145/2597073.2597122. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2597073.2597122.

[193] S. Baltes, L. Dumani, C. Treude, S. Diehl, "The Evolution of Stack Overflow Posts: Reconstruction and Analysis," *arXiv preprint arXiv:1811.00804*, Nov. 2018. arXiv: 1811.00804. [Online]. Available: http://arxiv.org/abs/1811.00804.

[194] B. Vasilescu, "Human aspects, gamification, and social media in collaborative software engineering," in *Companion Proceedings of the 36th International Conference on Software Engineering - ICSE Companion 2014*, New York, New York, USA: ACM Press, 2014, pp. 646–649, ISBN: 9781450327688. DOI: 10.1145/2591062.2591091. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2591062.2591091.

[195] G. Gousios, A. Zaidman, M.-A. Storey, A. van Deursen, "Work Practices and Challenges in Pull-Based Development: The Integrator's Perspective," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, IEEE, May 2015, pp. 358–368, ISBN: 978-1-4799-1934-5. DOI: 10.1109/ICSE.2015.55. [Online]. Available: http://ieeexplore.ieee.org/document/7194588/.

[196] B. Vasilescu, D. Posnett, B. Ray, M. G. van den Brand, A. Serebrenik, P. Devanbu, V. Filkov, "Gender and Tenure Diversity in GitHub Teams," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*, New York, New York, USA: ACM Press, 2015, pp. 3789–3798, ISBN: 9781450331456. DOI: 10.1145/2702123.2702549. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2702123.2702549.

[197] S. Baltes, S. Diehl, "Worse Than Spam: Issues In Sampling Software Developers," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '16*, New York, New York, USA: ACM Press, 2016, pp. 1–6, ISBN: 9781450344272. DOI: 10.1145/2961111.2962628. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2961111.2962628.

[198] G. Gousios, M.-A. Storey, A. Bacchelli, "Work practices and challenges in pull-based development: The Contributor's Perspective," in *Proceedings of the 38th International Conference on Software Engineering - ICSE '16*, New York, New York, USA: ACM Press, 2016, pp. 285–296, ISBN: 9781450339001. DOI: 10.1145/2884781.2884826. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2884781.2884826.

[199] A. Filippova, H. Cho, "The Effects and Antecedents of Conflict in Free and Open Source Software Development," in *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing - CSCW '16*, New York, New York, USA: ACM Press, 2016, pp. 703–714, ISBN: 9781450335928. DOI: 10.1145/2818048.2820018. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2818048.2820018.

[200] A. A. Sawant, R. Robbes, A. Bacchelli, "On the Reaction to Deprecation of 25,357 Clients of 4+1 Popular Java APIs," in *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, IEEE, Oct. 2016, pp. 400–410, ISBN: 978-1-5090-3806-0. DOI: 10.1109/ICSME.2016.64. [Online]. Available: http://ieeexplore.ieee.org/document/7816485/.

[201] Y. Li, N. R. Katsipoulakis, B. Chandramouli, J. Goldstein, D. Kossmann, "Mison: A Fast JSON Parser for Data Analytics," *Proceedings of the VLDB Endowment*, vol. 10, no. 10, pp. 1118–1129, Jun. 2017. DOI: 10.14778/3115404.3115416. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3115404.3115416.

[202] N. Munaiah, S. Kroh, C. Cabrey, M. Nagappan, "Curating GitHub for engineered software projects," *Empirical Software Engineering*, vol. 22, no. 6, pp. 3219–3253, Dec. 2017. DOI: 10.1007/s10664-017-9512-6. [Online]. Available: http://link.springer.com/10.1007/s10664-017-9512-6.

[203] E. van der Bent, J. Hage, J. Visser, G. Gousios, "How good is your puppet? An empirically defined and validated quality model for puppet," in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, IEEE, Mar. 2018, pp. 164–174, ISBN: 978-1-5386-4969-5. DOI: 10.1109/SANER.2018.8330206. [Online]. Available: http://ieeexplore.ieee.org/document/8330206/.

[204] A. Jaffe, J. Lacomis, E. J. Schwartz, C. L. Goues, B. Vasilescu, "Meaningful variable names for decompiled code: a machine translation approach," in *Proceedings of the 26th Conference on Program Comprehension - ICPC '18*, New York, New York, USA: ACM Press, 2018, pp. 20–30, ISBN: 9781450357142. DOI: 10.1145/3196321.3196330. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3196321.3196330.

[205] P. Martins, R. Achar, C. V. Lopes, "The Java Build Framework: Large Scale Compilation," *arXiv preprint arXiv:1804.04621*, Apr. 2018. arXiv: 1804.04621. [Online]. Available: http://arxiv.org/abs/1804.04621.

[206] A. A. Sawant, R. Robbes, A. Bacchelli, "On the reaction to deprecation of clients of 4 + 1 popular Java APIs and the JDK," *Empirical Software Engineering*, vol. 23, no. 4, pp. 2158–2197, Aug. 2018. DOI: 10.1007/s10664-017-9554-9. [Online]. Available: http://link.springer.com/10.1007/s10664-017-9554-9.

[207] A. Şeker, B. Diri, H. Arslan, M. F. Amasyalı, "Summarising big data: public GitHub dataset for software engineering challenges," *Cumhuriyet Science Journal*, vol. 41, no. 3, pp. 720–724, Sep. 2020, ISSN: 2587-2680. DOI: 10.17776/csj.728932. [Online]. Available: https://dergipark.org.tr/en/doi/10.17776/csj.728932.

[208] C. McMillan, M. Grechanik, D. Poshyvanyk, "Detecting similar software applications," in *Proceedings - International Conference on Software Engineering*, 2012, pp. 364–374, ISBN: 9781467310673. DOI: 10.1109/ICSE.2012.6227178.

[209] J. Hu, X. Sun, D. Lo, B. Li, "Modeling the evolution of development topics using Dynamic Topic Models," in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering, SANER 2015 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., Apr. 2015, pp. 3–12, ISBN: 9781479984695. DOI: 10.1109/SANER.2015.7081810.

[210] F. Ricci, L. Rokach, B. Shapira, "Introduction to recommender systems handbook," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, P. B. Kantor, Eds. Boston, MA: Springer US, 2011, pp. 1–35, ISBN: 978-0-387-85820-3. DOI: 10.1007/978-0-387-85820-3_1. [Online]. Available: https://doi.org/10.1007/978-0-387-85820-3_1.

[211] N. E. I. Karabadji, S. Beldjoudi, H. Seridi, S. Aridhi, W. Dhifli, "Improving memory-based user collaborative filtering with evolutionary multi-objective optimization," *Expert Systems with Applications*, vol. 98, pp. 153–165, May 2018, ISSN: 09574174. DOI: 10.1016/j.eswa.2018.01.015.

[212] S. T. Piantadosi, "Zipf's word frequency law in natural language: A critical review and future directions," *Psychonomic Bulletin and Review*, vol. 21, no. 5, pp. 1112–1130, Oct. 2014, ISSN: 15315320. DOI: 10.3758/s13423-014-0585-6. [Online]. Available: https://link.springer.com/article/10.3758/s13423-014-0585-6.

[213] J. Jiang, D. Lo, J. He, X. Xia, P. S. Kochhar, L. Zhang, "Why and how developers fork what from whom in GitHub," *Empirical Software Engineering*, vol. 22, no. 1, pp. 547–578, Feb. 2017, ISSN: 15737616. DOI: 10.1007/s10664-016-9436-6. [Online]. Available: http://github.com.

[214] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, D. Damian, "An in-depth study of the promises and perils of mining GitHub," *Empirical Software Engineering*, vol. 21, no. 5, pp. 2035–2071, Oct. 2016, ISSN: 15737616. DOI: 10.1007/s10664-015-9393-5. [Online]. Available: https://github.com/features.

A
TÜM SONUÇLAR

Tablo A.1 Topluluk İlişkisi yaklaşımıyla değerlendirmede tüm geliştirici metriklerin tüm skorları (içerikten-bağımsız)

Topluluk İlişkisi Değerlendirme Yaklaşımı

ID	Geliştirici Metrikleri	İçerikten-Bağımsız Benzerlik				
		ilk-1	ilk-3	ilk-5	ilk-10	ilk-20
1	*binary_code_contributions*	24,0	25,3	24,4	23,6	22,1
2	*binary_comments*	32,0	25,8	24,8	23,4	21,8
3	*binary_commit_related*	24,0	21,8	19,3	19,1	17,3
4	*binary_issue_related*	25,5	27,2	25,9	24,9	22,2
5	*binary_pr_related*	23,5	21,8	22,5	22,4	21,1
6	*code_2_comment*	22,5	21,0	19,8	17,9	15,8
7	*code_contributions*	23,0	22,3	22,2	21,0	19,2
8	*comment_2_code*	24,0	18,2	17,2	16,4	14,7
9	*comments*	31,0	24,7	26,5	24,2	22,4
10	*commit_2_comment*	14,0	10,3	10,3	9,8	7,7
11	*commit_authored*	14,0	12,2	12,7	11,6	10,5
12	*commit_commented*	22,0	20,2	19,2	17,0	14,7
13	*commit_committed*	15,5	12,3	12,9	11,6	10,2
14	*commit_related*	23,0	20,2	18,4	17,8	16,2
15	*issue_2_comment*	23,5	20,2	17,5	16,5	14,6
16	*issue_assigned*	7,0	8,8	8,3	7,5	6,0
17	*issue_closed*	11,5	9,5	8,1	6,3	6,0
18	*issue_hasPR*	26,0	20,8	17,9	16,2	14,9
19	*issue_commented*	30,0	26,2	25,5	23,8	23,4
20	*issue_opened*	23,0	21,7	20,7	18,7	16,8
21	*issue_related*	25,0	25,2	24,4	22,6	20,9
22	*O_commit_authored*	14,5	11,7	12,6	11,5	10,2
23	*O_commit_commented*	19,5	18,3	18,2	17,3	15,2
24	*O_commit_committed*	15,0	11,7	12,8	11,2	9,9
25	*O_issue_assigned*	7,0	8,8	8,3	7,5	6,0
26	*O_issue_closed*	11,5	9,5	8,1	6,3	6,0
27	*O_issue_closedPR*	16,0	16,0	14,4	14,0	12,9
28	*O_issue_commented*	22,0	23,2	21,2	19,3	17,6
29	*O_issue_opened*	15,0	15,2	15,8	15,4	15,5
30	*O_pr_assigned*	4,0	5,3	4,2	4,3	4,0
31	*O_pr_commented*	21,5	19,0	20,1	18,2	17,1
32	*O_pr_merged*	15,5	12,3	12,5	10,9	10,7
33	*O_pr_opened*	20,0	16,5	15,3	13,6	12,1
34	*pr_2_comment*	14,5	14,8	15,0	14,1	11,6
35	*pr_assigned*	4,0	5,3	4,2	4,3	4,0
36	*pr_commented*	20,0	18,0	19,6	17,9	15,5
37	*pr_merged*	16,0	13,0	12,8	11,8	11,2
38	*pr_opened*	26,0	19,8	19,2	16,0	14,2
39	*pr_related*	25,0	24,2	23,6	22,1	20,8
40	*Sun_metric*	8,0	7,0	7,6	7,8	7,6

Tablo A.2 Topluluk İlişkisi yaklaşımıyla değerlendirmede tüm geliştirici metriklerin tüm skorları (içeriğe-bağlı)

Topluluk İlişkisi Değerlendirme Yaklaşımı

ID	Geliştirici Metrikleri	İçeriğe-bağlı Benzerlik				
		ilk-1	ilk-3	ilk-5	ilk-10	ilk-20
1	*binary_code_contributions*	22,0	23,0	20,7	18,1	16,5
2	*binary_comments*	17,5	18,2	18,8	17,6	15,4
3	*binary_commit_related*	31,0	25,0	23,0	20,2	17,9
4	*binary_issue_related*	15,0	17,0	17,7	16,4	15,0
5	*binary_pr_related*	28,5	30,5	27,5	21,9	18,3
6	*code_2_comment*	27,5	22,2	22,2	19,4	17,1
7	*code_contributions*	24,0	22,2	20,5	18,3	15,2
8	*comment_2_code*	25,0	22,8	23,2	20,3	18,1
9	*comments*	22,0	20,0	19,3	17,0	14,8
10	*commit_2_comment*	25,0	20,8	18,3	14,4	12,2
11	*commit_authored*	28,5	24,5	23,1	18,5	15,4
12	*commit_commented*	26,0	20,5	20,5	18,5	15,6
13	*commit_committed*	29,0	24,5	22,6	18,5	15,3
14	*commit_related*	26,0	20,2	19,4	19,0	16,1
15	*issue_2_comment*	24,5	18,7	19,8	17,4	16,7
16	*issue_assigned*	26,5	19,5	17,4	14,3	11,9
17	*issue_closed*	20,0	17,7	14,8	13,2	11,6
18	*issue_hasPR*	34,0	29,8	24,2	21,7	18,0
19	*issue_commented*	22,0	18,8	18,2	17,0	15,1
20	*issue_opened*	26,0	22,3	22,1	18,7	16,1
21	*issue_related*	18,5	16,5	17,0	15,2	13,7
22	*O_commit_authored*	28,0	22,2	20,1	17,7	15,1
23	*O_commit_commented*	21,5	19,3	17,8	16,7	15,8
24	*O_commit_committed*	25,5	24,3	20,5	18,1	15,2
25	*O_issue_assigned*	17,0	17,5	14,4	12,7	10,3
26	*O_issue_closed*	13,0	10,8	10,2	10,4	9,0
27	*O_issue_closedPR*	21,0	18,3	16,6	14,5	12,8
28	*O_issue_commented*	15,0	13,7	13,0	12,3	11,1
29	*O_issue_opened*	16,5	15,8	15,6	13,0	12,7
30	*O_pr_assigned*	14,0	10,0	9,1	9,6	7,8
31	*O_pr_commented*	24,5	24,3	21,5	19,7	16,7
32	*O_pr_merged*	19,5	16,7	14,7	13,5	13,4
33	*O_pr_opened*	17,5	18,2	17,0	15,3	13,4
34	*pr_2_comment*	23,0	21,0	18,9	16,2	15,1
35	*pr_assigned*	12,5	10,7	9,3	8,5	7,6
36	*pr_commented*	20,0	21,8	20,1	18,6	16,2
37	*pr_merged*	28,0	21,3	20,7	18,9	17,1
38	*pr_opened*	35,5	29,5	24,7	22,3	19,0
39	*pr_related*	22,0	26,5	23,9	20,1	17,0
40	*Sun_metric*	14,5	12,0	11,5	9,9	9,0

Tablo A.3 Dil Deneyimi yaklaşımıyla değerlendirmede tüm geliştirici metriklerin tüm skorları (içerikten-bağımsız)

Dil Deneyimi Değerlendirme Yaklaşımı

ID	Geliştirici Metrikleri	İçerikten-Bağımsız Benzerlik				
		ilk-1	ilk-3	ilk-5	ilk-10	ilk-20
1	*binary_code_contributions*	73,0	74,7	72,6	70,7	70,0
2	*binary_comments*	77,0	70,3	68,6	70,1	69,3
3	*binary_commit_related*	72,0	65,3	64,6	60,4	57,2
4	*binary_issue_related*	86,0	80,0	77,4	74,4	72,0
5	*binary_pr_related*	70,0	69,0	67,0	66,5	64,9
6	*code_2_comment*	52,0	62,0	61,0	61,2	60,9
7	*code_contributions*	75,0	69,0	67,0	64,9	65,2
8	*comment_2_code*	59,0	61,7	61,4	62,3	61,2
9	*comments*	67,0	62,7	63,4	63,7	63,3
10	*commit_2_comment*	13,0	28,7	32,0	28,5	39,2
11	*commit_authored*	23,0	31,0	35,6	36,3	34,0
12	*commit_commented*	74,0	62,7	57,6	51,9	53,0
13	*commit_committed*	23,0	32,7	35,6	35,3	33,6
14	*commit_related*	69,0	63,7	61,8	60,1	57,5
15	*issue_2_comment*	65,0	66,7	64,6	64,5	63,1
16	*issue_assigned*	64,0	54,7	52,2	53,8	48,2
17	*issue_closed*	68,0	54,3	59,6	55,4	49,1
18	*issue_hasPR*	72,0	64,7	62,6	61,1	58,8
19	*issue_commented*	66,0	68,0	66,8	65,6	65,8
20	*issue_opened*	68,0	67,7	67,6	66,4	64,2
21	*issue_related*	74,0	71,3	71,8	67,3	65,4
22	*O_commit_authored*	23,0	31,7	36,0	37,1	34,1
23	*O_commit_commented*	73,0	60,3	55,8	51,2	53,2
24	*O_commit_committed*	22,0	32,0	35,2	34,8	33,1
25	*O_issue_assigned*	64,0	54,7	52,2	53,8	48,2
26	*O_issue_closed*	68,0	54,3	59,6	55,4	49,1
27	*O_issue_closedPR*	63,0	65,3	63,2	61,3	60,4
28	*O_issue_commented*	68,0	64,0	62,4	63,0	62,7
29	*O_issue_opened*	73,0	67,3	64,6	64,3	63,6
30	*O_pr_assigned*	44,0	29,3	26,6	20,5	21,3
31	*O_pr_commented*	48,0	45,7	47,8	48,0	50,2
32	*O_pr_merged*	74,0	55,7	50,4	45,0	47,4
33	*O_pr_opened*	70,0	63,3	61,8	61,1	62,0
34	*pr_2_comment*	58,0	53,3	54,8	56,6	53,0
35	*pr_assigned*	44,0	29,3	26,6	20,5	21,3
36	*pr_commented*	49,0	45,3	45,8	48,5	49,8
37	*pr_merged*	72,0	54,0	49,4	44,5	47,5
38	*pr_opened*	67,0	64,0	61,2	60,4	61,0
39	*pr_related*	63,0	62,3	62,8	64,0	63,7
40	*Sun_metric*	73,0	74,3	74,0	70,7	67,0

Tablo A.4 Dil Deneyimi yaklaşımıyla değerlendirmede tüm geliştirici metriklerin tüm skorları (içeriğe-bağlı)

Dil Deneyimi Değerlendirme Yaklaşımı						
ID	**Geliştirici Metrikleri**	**İçerikten-Bağımsız Benzerlik**				
		ilk-1	**ilk-3**	**ilk-5**	**ilk-10**	**ilk-20**
1	*binary_code_contributions*	76,0	73,3	73,4	74,3	72,2
2	*binary_comments*	75,0	72,0	74,4	75,2	73,2
3	*binary_commit_related*	70,0	67,7	69,2	65,9	64,5
4	*binary_issue_related*	75,0	71,7	72,6	75,0	73,5
5	*binary_pr_related*	78,0	78,7	77,4	74,4	72,3
6	*code_2_comment*	65,0	66,7	70,2	68,9	67,0
7	*code_contributions*	70,0	71,3	70,4	70,8	69,2
8	*comment_2_code*	60,0	68,3	67,6	67,4	67,0
9	*comments*	76,0	73,0	72,0	70,6	68,5
10	*commit_2_comment*	55,0	49,0	48,8	47,4	44,8
11	*commit_authored*	65,0	57,7	58,8	57,0	53,6
12	*commit_commented*	72,0	66,7	66,6	63,9	60,9
13	*commit_committed*	65,0	57,7	57,6	55,7	52,5
14	*commit_related*	72,0	67,7	67,6	64,5	63,6
15	*issue_2_comment*	66,0	68,7	70,8	70,5	68,8
16	*issue_assigned*	59,0	58,7	57,8	53,3	52,0
17	*issue_closed*	50,0	50,0	49,4	46,4	43,8
18	*issue_hasPR*	73,0	71,7	70,8	69,0	66,0
19	*issue_commented*	75,0	71,0	70,6	70,2	69,0
20	*issue_opened*	75,0	71,7	71,8	70,7	68,1
21	*issue_related*	76,0	73,0	71,4	69,6	68,7
22	*O_commit_authored*	61,0	57,0	57,4	53,1	51,2
23	*O_commit_commented*	65,0	69,0	67,6	65,2	61,5
24	*O_commit_committed*	60,0	56,0	54,8	51,3	50,1
25	*O_issue_assigned*	55,0	54,7	51,6	50,1	47,7
26	*O_issue_closed*	50,0	44,0	43,0	41,3	40,7
27	*O_issue_closedPR*	60,0	56,3	56,6	57,2	55,8
28	*O_issue_commented*	58,0	58,7	59,2	59,1	58,6
29	*O_issue_opened*	55,0	59,0	58,4	61,2	60,3
30	*O_pr_assigned*	36,0	31,3	30,6	30,0	28,9
31	*O_pr_commented*	62,0	65,3	64,2	61,8	59,0
32	*O_pr_merged*	60,0	57,7	57,8	57,9	55,5
33	*O_pr_opened*	55,0	54,7	53,0	54,6	55,8
34	*pr_2_comment*	58,0	58,0	56,4	54,6	53,2
35	*pr_assigned*	36,0	30,7	30,0	29,8	29,5
36	*pr_commented*	60,0	62,7	61,4	60,1	58,0
37	*pr_merged*	65,0	64,3	62,2	60,3	57,6
38	*pr_opened*	70,0	71,3	69,2	68,3	65,9
39	*pr_related*	69,0	71,0	71,2	67,6	67,2
40	*Sun_metric*	68,0	71,3	71,4	72,0	69,8

B
ANKET SORULARI

A. İş hakkında sorular

(1) Çalıştığınız firmadaki pozisyonunuz nedir?

a. Yazılım Geliştirici
b. Takım Lideri
c. Proje Yöneticisi
d. Diğer

(2) Yazılım sektöründeki deneyiminiz ne kadardır?

a. 1 yıldan daha az
b. 1-5 yıl arası
c. 6-10 yıl arası
d. 10 yıldan daha fazla

(3) Çalıştığınız firmadaki toplam yazılımcı sayısı nedir?

a. 1-10 kişi
b. 11-50 kişi
c. 51-100 kişi
d. 100 kişiden daha fazla

(4) OSS geliştirme platformlarından hangisini/hangilerini kullanıyorsunuz? (GitHub, GitLab, Bitbucket, vs.)

a. Cevaplar doğrudan şık olarak verilmedi. (Sonradan gruplandırıldı.)

(5) OSS geliştirme topluluklarında yer aldınız mı?

a. Evet
b. Hayır

(6) Açık kaynak bir proje/yazılım kullandınız mı?

a. Evet
b. Hayır

(7) Açık kaynak bir proje/yazılım geliştirdiniz mi?

a. Evet
b. Hayır

B. Kod geliştirme özellikleri hakkında sorular

(1) Bir geliştiricinin aşağıdaki aktivitelerden hangisini yapıyor olması onun aktiviteyi yaptığı proje ile **ilgisini** daha çok gösterir?

   a. *Issue*
   b. *Commit*
   c. *Pull Request-PR*

(2) Bir geliştiricinin aşağıdaki aktivitelerden hangisini yapıyor olması onun aktiviteyi yaptığı projede **yetkin** (projeye hakim) olduğunu daha çok gösterir?

   a. *Issue*
   b. *Commit*
   c. *Pull Request-PR*

(3) *Issue* ile ilgili aşağıdakilerden hangisi daha önemli bir aktivitedir?

   a. *Issue* Açma Sayısı
   b. *Issue* Kapatma Sayısı
   c. *Issue* Yorumları Sayısı
   d. *Issue* İnceleme Sayısı
   e. *PR* bağlı *Issue* Sayısı

(4) *Commit* ile ilgili aşağıdakilerden hangisi daha önemli bir aktivitedir?

   a. *Commit* Etme Sayısı
   b. *Commit* Yorumları Sayısı
   c. *Commit* Etiketleme Sayısı

(5) *PR* ile ilgili aşağıdakilerden hangisi daha önemli bir aktivitedir?

   a. *PR* Açma Sayısı
   b. *PR* Birleştirme Sayısı
   c. *PR* Yorumları Sayısı
   d. *PR* İnceleme Sayısı

(6) Yukarıdaki özellikler bağlamında bir projede yorum ile katkı yapmanın, o projede kod geliştirmek kadar önemli olduğunu düşünüyor musunuz? Neden?

TEZDEN ÜRETİLMİŞ YAYINLAR

Makale

A. A. Şeker, B. Diri, H. Arslan "New Developer Metrics for Open Source Software Development Challenges: An Empirical Study of Project Recommendation Systems," *Applied Sciences* vol.11, no:3, 2021. DOI:10.3390/app11030920

B. A. Şeker, B. Diri, H. Arslan, M. F. Amasyalı "Open Source Software Development Challenges: A Systematic Literature Review on GitHub," *International Journal of Open Source Software and Processes (IJOSSP)* vol.11, no:4, 2020. DOI:10.4018/IJOSSP.2020100101

C. A. Şeker, B. Diri, H. Arslan, M. F. Amasyalı "Summarising big data: public GitHub dataset for software engineering challenges," *Cumhuriyet Science Journal* vol.41, no:3, 2020. DOI:10.17776/csj.728932

Konferans Bildirisi

A. A. Şeker, B. Diri, H. Arslan "Using Open Source Distributed Code Development Features on GitHub: A Real-World Example," *2. International Eurasian Conference on Science, Engineering and Technology (EurasianSciEnTech)* 2020.