

**دانشکده فنی و مهندسی**

**گروه مهندسی کامپیوتر و فناوری اطلاعات**

**گزارش سمینار کارشناسي ارشد رشته مهندسی کامپیوتر نرم‌افزار(M.Sc)**

**عنوان سمینار:**

**سیستم‌های توصیه شده بازبینی کننده گذشته و کد فعلی**

**استاد راهنما:**

**دکتر سید علی رضوی**

**نگارنده:**

**زهرا کولیوند**

شهریور 1400

BISM2

**فهرست مطالب**

[چکیده: 1](#_Toc83029973)

[1.مقدمه………………………………………………………………………………………………………………………………………..2](#_Toc83029974)

[1-1تعريف مسئله و بيان سؤال‌های اصلي تحقيق: 3](#_Toc83029975)

[2-1 هدف: 4](#_Toc83029976)

[3-1 ضرورت تحقیق: 4](#_Toc83029977)

[4-1 روش ها و مراحل انجام تحقیق: 5](#_Toc83029978)

[5-1 سازمان پایان‌نامه مورد بررسی: 5](#_Toc83029979)

[1-6 ساختار گزارش تحقیق: 6](#_Toc83029980)

[2. ادبیات تحقیق و پیش زمینه 7](#_Toc83029981)

[1-2 مقدمه: 7](#_Toc83029982)

[2-2 مروری بر ادبیات سیستم ها 7](#_Toc83029983)

[3-2 بررسی ادبیات در مهندسی نرم افزار 9](#_Toc83029984)

[4-2  بررسی ادبیات در استخراج داده ها 9](#_Toc83029985)

[5-2 بررسی ادبیات در داده کاوی 10](#_Toc83029986)

[6-2 خلاصه 11](#_Toc83029987)

[3. پیشینه تحقیق 12](#_Toc83029988)

[1-3 مقدمه: 12](#_Toc83029989)

[2-3 پیشینه: 12](#_Toc83029990)

[CRRS 3-3 مبتنی بر مشخصات 14](#_Toc83029991)

[4-3 روش پژوهشی 15](#_Toc83029992)

[5-3 نتایج 16](#_Toc83029993)

[6-3 سیستم های پیدا شده 17](#_Toc83029994)

[1-6-3 بازنگر(REVFINDER ) 17](#_Toc83029995)

[cHRev 2-6-3 19](#_Toc83029996)

[CoRReCT 3-6**-3** 20](#_Toc83029997)

[TIE 4-6-3 20](#_Toc83029998)

[CodeFlow 5-6-3 22](#_Toc83029999)

[7-3 خلاصه 25](#_Toc83030000)

[4. تجزیه وتحلیل موضوع انتخابی 26](#_Toc83030001)

[1-4 مقدمه 26](#_Toc83030002)

[2-4 روش پروهش تحقیق: 26](#_Toc83030003)

[1-2-4 بررسی غربالگری 26](#_Toc83030004)

[3-4 برخی از روندها و الگوهای مشاهده شده 34](#_Toc83030005)

[5. جمع بندی و پیشنهادات 39](#_Toc83030006)

[1-5 مقدمه: 39](#_Toc83030007)

[2-5 نتایج حاصل از تحقیق 39](#_Toc83030008)

[1-2-5 پاسخ به سوالات تحقیق 39](#_Toc83030009)

[3-5 پیشنهادات 42](#_Toc83030010)

[4-5 کارهای آینده 43](#_Toc83030011)

[**مراجع** 45](#_Toc83030012)

[Abstract 47](#_Toc83030013)

**فهرست اشکال**

شکل 3.1: مثال محاسبه الگوریتم رتبه بندی Code-Reviewers.................................................................18

شکل 3.2: معماری TIE....................................................................................................................................21

شکل 3.3: روش تحقیق سه‌مرحله‌ای................................................................................................................23

شکل 3.4: مدل درخت تصمیم‌گیری برای طبقه‌بندی نظرات مفید ..................................................................24

شکل 1-4 : معیارهای انتخاب مرورگر کد........................................................................................................31

شکل 4-2 انواع ويژگي هايCRRS**.........................................................................................................**32

شکل 3-4 : اولویت زمانی که توصیه بازبینی کد وجود دارد**.......................................................................**33

شکل 4-4 : نوع مرور کد..................................................................................................................................33

شکل 5-4 : نوع CRRS و نقش شغلی...........................................................................................................35

شکل 6-4 : ویژگی های CRRS و نقش شغلی................................................................................................37

**فهرست جداول**

جدول 3.1: رشته‌های جستجو.........................................................................................................................16

جدول 3.2: مقاله‌های استخراج شده و داده‌های مورداستفاده........................................................................17

جدول 4.1: نقش‌های شغلی شرکت‌کنندگان..................................................................................................28

جدول 4.2: موقعیت جغرافیایی شرکت‌کنندگان...............................................................................................28

جدول 4.3: اندازه تیم پروژه.............................................................................................................................29

جدول 4.4: توزیع تیم......................................................................................................................................29

جدول 4.5: آشنایی با CRRS در بین شرکت‌کنندگان.....................................................................................30

# چکیده:

زمینه: انتخاب مرورگر کد یکی از جنبه‌های مهم توسعه نرم‌افزار است و به عوامل مختلفی بستگی دارد.

اهداف: هدف، درک راه حل­های موجود برای سیستم­های توصیه بازبینی کد(CRRS)، عواملی است که هنگام ساخت آنها باید در نظر گرفته شود و ابعاد مختلفی که بر اساس آنها می‌توان آنها را طبقه­بندی کرد. هدف ما درک ویژگی­های مهم CRRS و آنچه می­توان در CRRSهای موجود بهبود بخشید است.

روش­ها­:مطالعه مروری بر ادبیات برای درک CRRS های موجود انجام شد. نظرسنجی از اعضای پروژه توسعه نرم­افزار برای درک ویژگی­های مهم و مفقود شده در CRRS انجام شد.

نتایج: ما مقالات انتخاب شده را به دو دسته طبقه­بندی کردی: بر اساس نوع داده مورد استفاده برای ارائه توصیه­ها و نوع پروژه مورد استفاده برای ارزیابی. این نظرسنجی به ما کمک کرد ویژگی­های موجود در CRRS را درک کرده و برخی از روندها و الگوها را مشاهده کنیم.

**کلمات کلیدی:** بازبینی کد، برنامه‌نویسی جفتی، بررسی سیستماتیک

فصل اول

# مقدمه

مرور کد یک بررسی سیستماتیک از کد منبع رایانه است و اغلب به‌عنوان یک بررسی همکار انجام می‌شود. هدف بازبینی کد شناسایی و اصلاح اشتباهات در کد منبع و همچنین بهبود کیفیت کد و مهارت توسعه دهندگان نرم‌افزار است. همچنین، این هدف فقط بهبود کیفیت کد یا یافتن نقص در کد منبع نیست. همچنین باعث افزایش آگاهی تیم و همچنین کمک به توزیع دانش می‌شود. همچنین مالکیت کد مشترک را تشویق می‌کند.

چهار نوع بررسی کد وجود دارد:

1. **برنامه‌نویسی جفت:** در این نوع بازبینی کد، دو توسعه‌دهنده به طور هم‌زمان کد منبع را تولید می‌کنند و به طور هم‌زمان مرور می‌کنند.
2. **مرور کد به کمک ابزار**: برای این نوع مرور کد، نویسندگان و توسعه دهندگان از ابزارهای بررسی کد همتا استفاده می‌کنند.
3. **بازبینی کد مرور:** در اینجا ، توسعه دهنده مرورگر را از طریق مجموعه ای از تغییرات کد راهنمایی می کند.
4. **بازبینی رسمی کد**: این نوع بررسی کد شامل یک دقت و بازرسی دقیق کد با مشارکت تعدادی از شرکت‌کنندگان و در چند مرحله. این یک روش سنتی برای مرور کد است که شامل شرکت در تعدادی از جلسات و مرور خط به خط است.

مرور کد را می توان به عنوان بازرسی دستی تغییرات در کد منبع دانست. تعدادی ابزار و سیستم توصیه ای وجود دارد که به منظور بررسی کد توسط تعدادی از سازمان‌های مختلف توسعه یافته است .

زمینه‌های متعددی وجود دارد که در آنها مشارکت سیستم‌های توصیه شده برای اعضای پروژه توسعه نرم‌افزار مفید بوده است. برای کمک به وظیفه بازبینی کد، تحقیقات قابل‌توجهی در مورد سیستم‌های توصیه‌ای انجام شده است که هدف آنها ارائه توصیه‌های بازبینان کد بر اساس جنبه های مختلف است. دلایل مختلفی وجود دارد که چرا علاوه بر یافتن عیوب کد، به مرورگر کد نیز نیاز است به این امر به این دلیل است که مرورگران کد همچنین بر بهبود کد، یافتن راه‌حل‌های جایگزین برای یک مشکل، انتقال دانش، بداهه­پردازی در فرایند توسعه، اجتناب از وقفه‌های ایجاد شده، اشتراک مالکیت کد و همچنین ارزیابی تیم تمرکز می‌کنند.

سابقه و همچنین تخصص یک بازبین کد در یک زمینه خاص بر اساس درخواست­های کشش آنها.

نمونه­هایی از سیستم های توصیه­ای کلی در مهندسی نرم­افزار

1. سیستم‌های توصیه­گر کد گرافیکی
2. سیستم هیپی کات

Lee و kang مطالعه ای بر روی "سیستم‌های توصیه گر کد گرافیکی" انجام دادند تا بفهمند ابزارهای تجسم نرم افزار تا چه اندازه به توسعه­دهندگان در درک کد کمک کرده است. نویسندگان دریافتند که توسعه­دهندگان زمان قابل توجهی را صرف درک مبانی کد می‌کنند. برای سهولت این کار، تعدادی از توصیه­کنندگان کد گرافیکی برای آنها ایجاد شد. این سیستم‌های توصیه­گر از دو چکیده استفاده کردند:

1. طراحی و توضیح کد و مستندات سیستم نرم‌افزاری
2. تجزیه‌وتحلیل کد [[1]](#footnote-1)

یک سیستم توصیه‌گر به نام توسعه داده شد که دسترسی توسعه‌دهندگان را به حافظه گروهی که شامل آثار مرتبط با پروژه است که در طول توسعه پروژه ایجاد شده است، فراهم می‌کند D. Cubranic, G. C. Murphy] j.Singerوهمکاران2005[.

 این به توسعه­دهندگان کمک می‌کند تا در غلبه بر مشکلات فنی و جامعه‌شناختی وقت صرفه‌جویی کنند. این ابزار با تغییرات بسیار اندک یا بدون تغییر در شیوه‌های کاری موجود، حافظه گروه را به طور خودکار ایجاد می‌کند. این سیستم پیشنهادی در به‌اشتراک‌گذاری اطلاعات مربوط به یک پروژه ازهرجهت به همه اعضای تیم توسعه کمک کرده و در نتیجه در توضیح مفاهیم برای اعضای موجود و جدید تیم توسعه صرفه‌جویی می‌کند.

## 1-1 تعریف مسئله و بیان سؤال‌های اصلی تحقیق:

مرور کد یک بررسی سیستماتیک از کد منبع رایانه است و اغلب به‌عنوان یک بررسی همکار انجام می‌شود. هدف بازبینی کد شناسایی و اصلاح اشتباهات در کد منبع و همچنین بهبود کیفیت کد و مهارت توسعه­دهندگان نرم‌افزار است. همچنین، این هدف فقط بهبود کیفیت کد یا یافتن نقص در کد منبع نیست. همچنین باعث افزایش آگاهی تیم و همچنین کمک به توزیع دانش می‌شود. همچنین مالکیت کد مشترک را تشویق می‌کند.

1. راهکارهای موجود برای سیستم‌های توصیه برای مرورگران کد چیست؟
2. هنگام ایجاد یک سیستم توصیه برای مرورگران کد، چه عواملی باید در نظر گرفته شوند؟
3. چگونه می‌توان سیستم‌های توصیه‌ای موجود برای مرورگران کد در ادبیات را دسته‌بندی کرد؟
4. ویژگی‌های مهم سیستم توصیه برای مرورگران کد چیست؟
5. چگونه می‌توان سیستم‌های پیشنهادی موجود برای مرورگران کد را بهبود بخشید؟ به‌عبارت‌دیگر، چه ویژگی‌هایی در پیاده‌سازی‌های موجود برای سیستم‌های توصیه بازبینی کننده کد وجود ندارد؟

سؤال 1 و 2 و 3 با استفاده از مرور ادبیات سیستماتیک پاسخ داده می‌شوند درحالی‌که سؤال 4 و 5 با بررسی طیف وسیعی از اعضای پروژه نرم‌افزاری پاسخ داده می‌شوند.

## 2-1 هدف:

هدف تحقیق ما دوگانه است: این کار بر روی کد پشتیبانی شده از ابزار تمرکز می‌کند و یافتن پاسخ سؤالات "گذشته" با انجام "مرور ادبیات سیستماتیک" و دوم یافتن پاسخ سؤالات "حال" با انجام نظرسنجی از اعضای پروژه نرم‌افزاری. "مرور ادبیات سیستماتیک" در یافتن جزئیات در مورد سیستم‌های توصیه بازبینی کننده کد موجود کمک می‌کند، درحالی‌که این نظرسنجی به یافتن تغییراتی که مهندسان نرم‌افزار فکر می‌کنند نیاز دارند یا آنچه در سیستم‌های توصیه بازبینی کد موجود وجود ندارد، کمک می‌کند.

## 3-1 ضرورت تحقیق:

هدف از این تحقیق مستندسازی آموخته‌های به‌دست‌آمده از"مطالعه مروری بر ادبیات سیستماتیک" و همچنین "بررسی" انجام شده بر روی طیف وسیعی از اعضای پروژه نرم‌افزار. این کار به این دلیل انجام شده است که تحقیقات بسیار کمی در مورد سیستم‌های توصیه کد مرورگر انجام شده است (CRRS)و تحقیقات بیشتری در مورد شیوه‌ها و رویه‌های بازبینی کد انجام شده است. اهدافی که در یافتن پاسخ به نتیجه موردنیاز به ما کمک می‌کند در زیر ذکر شده است:

• تجزیه‌وتحلیل راه‌حل‌های ارائه شده توسط "مرورگر کد موجود سیستم‌های توصیه "

• برای درک راه حل­ها/ویژگی­های موجود در "سیستم­های توصیه مرورگر کد" موجود.

**•** تجزیه­وتحلیل وسایل مختلف انشعاب سیستم‌های پیشنهادی مرورگر کد بر اساس نحوه پیاده‌سازی آنها.

## 4-1 روش‌ها و مراحل انجام تحقیق:

1. تعدادی از ویژگی‌های موجود در سیستم‌های پیشنهادی بازبینی کد (CRRS) را شناسایی کرده و آن ویژگی‌ها را بر اساس مفید بودن آنها.
2. CRRS های موجود را بر اساس ابعاد مختلف طبقه­بندی کردیم.
3. ویژگی‌هایی را که می‌توان هنگام انتخاب مرورگر کد مهم تلقی کرد، شناسایی کردیم.
4. بهبودهای احتمالی CRRS های موجود را برای تسهیل یافتن مرورگران کد مناسب شناسایی کردیم.

## 5-1 سازمان پایان‌نامه مورد بررسی:

فصل‌های این پایان‌نامه به‌صورت ذیل مرتب شده است:

* **فصل 2:** کارهای مرتبط و این فصل خلاصه‌ای از مرور ادبیات قبلی انجام شده در مهندسی نرم‌افزار، داده‌کاوی و سیستم‌های توصیه‌گر را ارائه می‌دهد.
* **فصل 3**: نتایج مطالعه مرور ادبیات و شیوه ما و ابزارهای بازبینی کد ارائه شده است.
* **فصل 4:** بررسی نتایج به‌منظور انجام نظرسنجی از مهندسان نرم‌افزار برای تعیین نیازهای اطلاعاتی برای مرورگران کد ارائه شده است.
* **فصل 5:**به سؤالات تحقیق خود پاسخ داده و بحث شده است.
* **فصل 6:**درباره نتیجه‌گیری و کارهای آینده است.

## 1-6 ساختار گزارش تحقیق:

**فصل 1:** به تعریف و مقدمه و دلایل نیاز به طرح ارائه شده پرداخته می‌شود.

**فصل 2:** به پیش‌زمینه و کارهای وابسته پرداخته می‌شود.

**فصل 3:** مروری است بر کارهای انجام شده طرح پیشنهادی پایان‌نامه.

**فصل 4 :** به کاربردها و مزایا و معایب روش‌های مطرح شده پرداخته می‌شود.

**فصل 5 :**نیز به جمع‌بندی و نتیجه‌گیری پرداخته می‌شود.

فصل دوم

# 2. ادبیات تحقیق و پیش‌زمینه

## 1-2 مقدمه:

این فصل خلاصه‌ای از مرور ادبیات قبلی انجام شده در مهندسی نرم‌افزار، داده‌کاوی و سیستم‌های توصیه‌گر را ارائه می‌دهد. این مطالعات ارائه شده است تا نشان دهد که چگونه مطالعات مرور ادبیات درگذشته انجام شده و برای راهنمایی روش بررسی ادبیات ما مورداستفاده قرار گرفته است.

بررسی ادبیات انجام شده در مورد سیستم‌های توصیه­گر شامل سیستم‌های توصیه‌گر است که هدف آنها استخراج اطلاعات مربوطه از حجم زیادی از دانش و سیستم‌های توصیه‌ای برای مهندسی نرم‌افزار است که ویژگی‌های سیستم‌های موجود، شکاف‌های تحقیقاتی و کارهای احتمالی آینده را ارائه می‌دهد. به طور مشابه، یک مطالعه مرور ادبیات در زمینه مهندسی نرم‌افزار در مورد مطالعات پیش‌بینی خطا و روش توسعه نرم‌افزار چابک انجام شد. بررسی ادبیات انجام شده درداده کاوی دو مدل پرکاربرد برای داده کاوی در CRM (مشتری مدیریت روابط) را کشف کرد.

## 2-2 مروری بر ادبیات سیستم‌ها

## 

مرور ادبیات توسط HARUNA ،Ismail ، SUHENDROYONO و همکاران انجام شد در مورد سیستم‌های توصیه­کننده آگاه­به زمینه (CARS) که هدف آنها استخراج اطلاعات مربوطه از حجم زیادی از دانش است. این سیستم‌های توصیه‌گر، ارائه اطلاعات زمینه‌ای­و مرتبط بر اساس "جستجوهای کاربران" و ارائه توصیه‌های شخصی‌تر کاربر است شامل سه مرحله اصلی است:

1. بررسی عمیق و طبقه‌بندی ادبیات براساس حوزه‌های مختلف مدل‌های کاربردی، فیلترینگ، استخراج و همچنین رویکردهای ارزیابی است.
2. ارائه نتایج بررسی با مزایا و معایب مرور است.
3. برجسته‌کردن چالش‌ها/فرصت‌های احتمالی یا کار یا تحقیقات آینده است که می‌توان انجام داد. این شامل کمک به مبتدیان و محققان جدید برای درک پیش نیازهای توسعه CARS و همچنین ارائه این بررسی به عنوان معیاری برای توسعه CARS برای کاربران متخصص است K. Haruna] M. AIsmail, S. Suhendroyono, 2017[.

سیستم توصیه نوعی نرم‌افزار کاربردی است که هدف آن ارائه/توصیه اطلاعات مناسب به کاربران بر اساس الزامات کاربر است.

برای این حوزه، یک مطالعه مروری ادبیات سیستماتیک مشابه GASPARIC, Janesانجام شد، که نتایج عملکردی کهRSSE های موجود(سیستم توصیه برای مهندسی نرم­افزار) ارائه می‌دهند،­ شکاف‌های تحقیق­و­همچنین جهت‌های احتمالی تحقیق را بررسی می‌کند. به آنها از رویکرد روش شناختی پیروی کردند که شامل فیلتر کردن مقالات تحقیقاتی مرتبط و جمع­آوری شده بر اساس معیارهای مختلف بود. معیارهای خروج آنها شامل مقاله‌هایی بود که به حوزه تحقیق بی­ربط بود، مقالاتی که راه حل­های اجرایی نشده را توصیف می‌کرد یا مقاله‌هایی که کاملاً در دسترس نبودند. برای استخراج مقالات مربوطه، مقاله‌ها بر اساس محتوای توصیف شده در چکیده مقاله یا گاهی عنوان، فیلتر و تقسیم می‌شوند. نویسندگان پس از پیروی از رویکرد روش شناختی خود، به چهار سال تحقیقاتی خود پاسخ دادند که عبارتند از:

* خروجی ارائه شده توسط RSSE های موجود
* مزایایی که این RSSE ها برای مهندسان نرم افزار فراهم می کند
* انواع ورودی موردنیاز این RSSE هاو چه تلاش‌هایی مهندس نرم‌افزار نیاز به استفاده از اینRSSEها دارد
* برخی از خروجی های RSSE های موجود شامل فایل های کد منبع باینری
* تغییرات در محیط استقرار
* الگوهای طراحی و اسناد دیجیتالی است که ممکن است برای مهندس نرم‌افزار جالب باشد

RSSE های موجود عمدتا ازچه چیزهای پشتیبانی می کنند:

از استفاده مجدد، اشکال‌زدایی، پیاده‌سازی، مراحل/فعالیت‌های نگهداری و پشتیبانی از بهبود کیفیت سیستم به نفع مهندسان نرم‌افزار.

ورودی هایی که این RSSE های فعلی از آنها استفاده می کنند:

شامل فایل‌های گزارش، ارتباط بین مهندسین نرم افزار، کد منبع، ورودی کاربر (به عنوان مثال، عبارت‌های جستجو، پرس و جو، تنظیمات، ترجیحات)، مصنوعات آزمایشی و فرایند توسعه نرم افزار است همچنین، تلاش‌هایی که یک مهندس نرم افزار باید برای استفاده از RSSEهای موجود انجام دهد به عنوان تلاش‌های گسترده، تلاش‌های کم و بدون تلاش طبقه بندی می شود. M. Gasparic and A. Janes]وهمکاران 2016[

## 3-2 بررسی ادبیات در مهندسی نرم‌افزار

      پیش بینی دقیق خطاهای کد می‌تواند هزینه آزمایش را تا حد زیادی کاهش دهد و همچنین کیفیت محصول نرم افزاری را افزایش دهد. به همین منظور، یک مطالعه مروری بر ادبیات، انجام شد که بر مطالعات پیش بینی خطا متمرکز شده است.

مراحل اولیه شامل مقاله‌هاو مطالعات مربوطه و حذف مطالعات مکرر است. هنگام حذف و شامل مقاله‌ها، موارد مختلفی در نظر گرفته می‌شود، مانند مقاله‌هایی که از منابع مختلف مانند مجلات، کنفرانس‌ها و پایگاه‌های داده استخراج شده‌اند و بر اساس محتوای عنوان و چکیده آنها مرتب شده‌اند. یافته‌ها نشان می‌دهد که اکثر مطالعات اطلاعات زمینه‌ای و روش‌شناختی کافی را برای درک کامل یک مدل گزارش نمی‌دهند. نویسندگان مجموعه‌ای از معیارها را ارائه می‌دهند.

## 4-2  بررسی ادبیات در استخراج داده‌ها

مجموعه‌ای از جزئیات اساسی زمینه‌ای و روش‌شناختی را که مطالعات پیش‌بینی خطا باید گزارش کنند، شناسایی می‌کند.

روش توسعه نرم­افزار چابک یک متدولوژی توسعه نرم‌افزاری متداول است که توسط بسیاری از پروژه‌های توسعه نرم‌افزار استفاده می‌شود. هدف این روش اطمینان از تحویل خوب محصول مطابق نیاز کاربر و تجربه کاربری مناسب(Ux)است. به‌منظور ارائه یک محصول باکیفیت، مشارکت ذی‌نفعان و کاربران، همراه با حلقه‌های بازخورد از هر دو طرف ضروری است. به‌منظور ارائه یک محصول باکیفیت، مشارکت ذی‌نفعان و کاربران، همراه با حلقه‌های بازخورد از هر دو طرف ضروری است. یک مطالعه مروری بر روی متدولوژی چابک توسط Schon ,Thomaschewski, Escalona منظور بررسی وضعیت فعلی کار در زمینه و پیشرفت‌های احتمالی آینده برای بررسی جنبه‌هایی که دروضعیت فعلی وجود ندارد، انجام­ شد[2017, E. Schon, J. Thomaschewski, and M. J Escalona]

آنها مطالعه را در سه مرحله اصلی انجام دادند:

1. **برنامه‌ریزی**: یافتن نیاز شناسایی برای بررسی, چارچوب‌بندی سؤالات تحقیق و توسعه و ارزیابی پروتکل بازبینی بود.
2. **انجام:** باهدف جستجوی مقالات تحقیق، انتخاب مقالات مربوط به مطالعه، ارزیابی کیفی و استخراج و تجزیه‌وتحلیل داده ها انجام شد.
3. **گزارش:** با هدف استخراج و بحث درباره نتایج به­د­ست­آمده از مرحله قبل و سپس نوشتن، ارزیابی و قالب بندی گزارش نهایی برای مطالعه انجام شد.

## 5-2 بررسی ادبیات در داده‌کاوی

مرور ادبیات انجام شده توسط Ngai ، XIU وChau نمونه دیگری از روش‌شناسی برای مرور ادبیات سیستماتیک در زمینه داده­کاوی را ارائه می‌دهد.

تکنیک های داده­کاوی در مدیریت ارتباط با مشتری(CRM) اعمال می­شود و Ngai ، XIU و Chau به کمک مرور ادبیاتی که انجام داده­اند، بینش کاملی در این زمینه ارائه می­دهد. نویسندگان حدود 87 مقاله تحقیقاتی مرتبط را برای این منظور جمع آوری کردند که بر اساس چهار بعد CRM تقسیم شده­اند:

شامل توسعه مشتری، شناسایی مشتری، جذب مشتری و حفظ مشتری

هفت تکنیک داده‌کاوی:

ارتباط، طبقه‌بندی، خوشه‌بندی، پیش‌بینی، رگرسیون، کشف دنباله و تجسم.

برای وضوح بیشتر، ابعاد CRM بیشتر به 9 زیر گروه از عناصر CRM طبقه­بندی می­شود که تحت تکنیک­های داده­کاوی قرار می­گیرند. بر اساس مطالعه، مشخص شد که طبقه­بندی و ارتباط دو مدل پرکاربرد برای داده کاوی در CRM هستند. همچنین، از چهار بعد CRM، مشتری حفظ شده­ترین مورد تحقیق است، اگرچه اکثر آنها مربوط به برنامه­های بازاریابی وفاداری یک به یک بود.

## 6-2 خلاصه

مطالعات مرور ادبیات برای سیستم‌های توصیه‌گر و از زمینه‌های مهندسی نرم‌افزار و داده‌کاوی ارائه شد. در مورد سیستم‌های توصیه، بررسی ادبیات بر سیستم‌های توصیه‌گر و سیستم‌های توصیه در مهندسی نرم‌افزار متمرکز است. در زمینه مهندسی نرم‌افزار، مطالعه ارائه شده در زمینه پیش‌بینی خطا و همچنین روش Agile انجام شده است. در زمینه داده‌کاوی، یک مطالعه مروری بر ادبیات انجام شد که در آن مفهوم داده‌کاوی برای مدیریت ارتباط با مشتری (CRM) مورد هدف قرار گرفت.

**فصل سوم**

# **3. پیشینه تحقیق**

## 1-3 مقدمه:

درگذشته تعدادی سیستم/ابزار توصیه بازبینی کد وجود داشته است. به‌عنوان‌مثال، ابزاری به نام Review Bot یکی دیگر از ابزارهای توسعه یافته توسط Balachandran است. که در پروژه VMware استفاده شد. Review Bot متشکل از یک الگوریتم بود که تغییرات کد انجام شده در یک خط کد را به شکلی که کاملاً شبیه به دستور Git fault است، بررسی می­کند. به هر نویسنده­ای که روی تغییر کد در کد منبع کار کرده است امتیاز تعلق می­گیرد اما نویسندگان با تغییرات اخیر نسبت به نویسندگان تغییرات قدیمی امتیاز بیشتری کسب می­کنند. در پایان، از جمع­بندی هر نویسنده جداگانه برای تصمیم­گیری نویسندگان برتر k استفاده می­شود و سپس به آنها توصیه می­شود که مرورگر کد شوند.

ما یک مطالعه سیستماتیک مرور ادبیات را انجام دادیم تا به سه سال تحقیقاتی اولیه خود پاسخ پیدا کنیم. ابتدا ما روش‌شناسی خود را قبل از ارائه نتایج مطالعه خود تعریف می‌کنیم.

## 2-3 پیشینه:

یک مطالعه موردی انجام داد که در آن آنها یک مطالعه اکتشافی از شیوه‌های مرور کد مدرن در Google انجام دادند.

مطالعه اکتشافی آنها بر 3 جنبه بررسی کد متمرکز بود:

1. انگیزه‌های بررسی کد
2. شیوه‌های فعلی
3. تفسیر توسعه‌دهندگان بررسی کد.

به منظور ایجاد ساختار بیشتر در بازبینی کد، چندین ابزار در نرم­افزار منبع باز(OSS) و تنظیمات صنعتی ظاهر شد. برای این منظور، نویسندگان برخی از رویکردهای مرور مبتنی بر ابزار را مطالعه کردند. این ابزارها شامل CodeFlow مورد استفاده مایکروسافت،­Gerritاستفاده شده توسطGoogle Chromium ، ReviewBoard توسعه­یافته توسط VMwareb nmnngbbb و Phabricator مورد استفاده فیس­بوک است. در زیر مروری کوتاه بر هر یک از این سیستم‌های توصیه مرورگر کد است.

1 CodeFlow :CodeFlow .وضعیت هر شخص (توسعه­دهنده یا بازبینی کننده) و موقعیت آنها در این مرحله (یعنی انتظار، بررسی، امضا) را ردیابی کرد. CodeFlow نویسنده را از ارسال هیچ گونه تغییری بدون تأیید منع نمی­کند و همچنین از گپ ها در موضوعات نظر پشتیبانی می­کند.

2 Google Chromium **:Gerrit.** از سیستم توصیه بازبینی کد موجود به نام Gerrit استفاده می‌کند که در آن تغییرات تنها پس از تأیید داوران و تأیید خودکار مبنی بر اینکه تغییر ساختار را خراب نمی‌کند، در شاخه اصلی ادغام می‌شوند.

3 ReviewBoard **:ReviewBoard**. توسط VMware توسعه یافته است و هدف آن این است ادغام تجزیه و تحلیل استاتیک در فرایند بررسی این ادغام متکی است، در مورد تغییراتی که نویسندگان به صورت دستی درخواست تجزیه­و­تحلیل می‌کنند، و در نتیجه کیفیت مرور کد بهبود یافته است.

Phabricator **:Phabricator** ، که توسط فیس بوک استفاده می‌شود، به یک بازبین اجازه می‌دهد تا تغییر را "به عهده بگیرد" و خود آن را انجام دهد. همچنین، این سیستم برای تجزیه و تحلیل استاتیک خودکار یا خطاهای ادغام مداوم رفع می‌کند.

به منظور درک روند بررسی کد در Google، نویسندگان بر دو جنبه اصلی تمرکز کردند: فرایند بازبینی که توسعه­دهندگان در طول بررسی‌های خاص تجربه می‌کنند­و اینکه آیا توسعه­دهندگان با وجود چالش‌ها از بررسی‌های ارائه شده راضی هستند یا خیر. برای بررسی کد در Google، آنها از CRITIQUE استفاده کردند، یک ابزار مرور کد داخلی مبتنی بر وب، توسعه یافته داخلی. در این ابزار، یک بازبین می‌تواند تفاوت برجسته تغییرات پیشنهادی را ببیند و همچنین یک بحث موضوعی در مورد خطوط کد با توسعه­دهندگان یا سایر بازبینان آغاز کند. CRITIQUE همچنین نمایی از همه عملکردهای ورود به سیستم یک توسعه­دهنده، و همچنین تعامل آن با ابزار را شامل بازکردن ابزار، ایجاد تغییرات، مشاهده تفاوت و تأیید تغییرات ارائه می‌دهد. به منظور درک انگیزه توسعه­دهندگان برای بررسی کد در Google با استفاده از CRITIQUE و درک توسعه­دهندگان در مورد همین، نویسندگان از مصاحبه به عنوان ابزاری برای جمع آوری داده‌ها استفاده کردند.

بر اساس داده‌های جمع‌آوری‌شده از مصاحبه‌های انجام شده، یافته‌های زیر به‌دست‌آمده است.

**یافتن 1**: بررسی کدهای انجام شده در Google نه‌تنها باهدف تصحیح خطاها یا مشکلات است، بلکه همچنین برای اطمینان از خوانایی و قابلیت نگهداری کد که به‌عنوان جنبه آموزشی در نظر گرفته شد، انجام می‌شود.

**یافته 2**: انتظارات در مورد بررسی کد خاص بستگی به رابطه مشترک توسعه‌دهنده/نویسنده و مرورگر کد دارد.

Sadowski, Soderberg], 2018[

## CRRS 3-3 مبتنی بر مشخصات

Fejzer ، Przymus، یک سیستم توصیه بازبینی کد مبتنی بر مشخصات را پیشنهاد کرد. در مدل پیشنهادی داور پیشنهادی، مشخصات بازبینی شامل سابقه بازبینی و تعهدات یک داور بالقوه است.

در مدل توصیه بازبینی آنها، هنگامی که یک درخواست تعهد جدید به مخزن می‌رسد، با نمایش چند مجموعه‌ای از تعهدات (مجموعه‌های متعددی از دنباله کلمات موجود در یک مسیر فایل اصلاح شده در یک تعهد) و همچنین پروفایل بازبین‌ها مقایسه می‌شود شباهت بین نمایندگی چند مجموعه‌ای از تعهدات و نمایه‌های بازبینی­کنندگان محاسبه می­شود و n بازنگری برتر انتخاب می‌شوند. در اینجا، به روز رسانی مشخصات بازبینی­کننده یکی از مهم‌ترین و مکررترین عملیات است. هرگاه نظر جدیدی توسط یک بازبین انجام شود، تعهد به نمایه وی اضافه می‌شود. همچنین، وقتی صحبت از مشخصات بالقوه یک داور می‌شود، زمان یکی از عوامل مهمی است که باید مورد توجه قرار گیرد. کاندیدایی که بررسی‌های اخیر یا تعهداتی در نمایه خود داشته باشد، بعنوان کاندیدای محتمل تری برای بررسی درخواست تعهد در نظر گرفته می‌شود.

نویسندگان با استفاده از روش پیشنهادی خود ارزیابی تجربی انجام دادند Android ، LibreOffice ، OpenStack نتایج تجربی به صورت زیر بود به شرح زیر است:

1. **تعداد نظرات به‌ازای هر داور**: اکثر داوران کمتر از 20 نظر برای Android و LibreOffice و کمتر از 60 نظر برای OpenStack و Qt ایجاد کردند.

2. **مدت فعالیت تک‌نفره داوران:** در مورد Android و LibreOffice، داوران در مقایسه با نظر دهندگان Qt و OpenStack زمان بیشتری صرف کردند. علت احتمالی این نتیجه، نگهدارندگان تعیین شده­ای است که برای شرکت‌های مشارکت­کننده در این پروژه­ها کار می‌کنند.

3. **مدت بررسی های فردی:** اکثر بررسی­ها طی سه روز برای پروژه­های LibreOffice و OpenStack، حداکثر دو روز برای Qt و حداکثر شش روز برای پروژه­های Android تکمیل شد[2018, M. Fejzer, P. Przymus].

## 4-3 روش پژوهشی

مطالعه مروری بر ادبیات سیستماتیک روشی است که در آن ادبیات موجود مربوط به تحقیق تعیین می‌شود، سپس ارزیابی می‌شود و در نهایت درک می‌شود. برای تحقیقات خود، ما رویکردی را که Kitchenham وCharters اتخاذ کرده است دنبال می‌کنیم که شامل مراحل زیر است:

1. تمام کلمات کلیدی احتمالی مربوط به تحقیق مشخص شده است. ما کلمات زیر را شناسایی کردیم:

کد، مرورگر، توصیه، سیستم‌ها، ابزارها و توصیه‌کننده. این کلمات کلیدی بر اساس موضوع تحقیق ما مشخص شد.

2. از کلمات کلیدی مشخص شده برای تشکیل رشته‌های جستجو استفاده کنید. رشته‌های جستجو برای به‌دست‌آوردن مقالات تحقیق از پایگاه‌های داده آنلاین استفاده می‌شود. ما از یک‌رشته جستجو برای کلمات کلیدی احتمالی و مترادف آنها استفاده کردیم. ما دو رشته جستجو ایجاد کردیم. برای یافتن رشته‌های جستجو به جدول 3.1 مراجعه کنید.

3. مقالات تحقیقاتی به‌دست‌آمده سپس بر اساس معیارهای مختلف حذف و ورود فیلتر می‌شوند. مقالات ابتدا با خواندن عنوان‌ها و چکیده مقالات تحقیق فیلتر می‌شوند.

فیلترکردن نتایج جستجو در سه مرحله اصلی انجام شد:

* فیلتر کردن مقالات تحقیق بر اساس خواندن عنوان مقاله
* فیلترکردن مقالات تحقیق بر اساس چکیده مقاله
* فیلترکردن مقالات تحقیق بر اساس خواندن متن کامل

مقالات فیلتر شده به طور کامل خوانده می‌شوند، ارزیابی و تفسیر می‌شوند تا اطلاعات مربوطه را به دست آورند. [B. Kitchenham and S. Charters,2007]

|  |  |
| --- | --- |
| 1 | (کد) و (داوران) و  (توصیه‌کننده) یا  (توصیه)  و)سیستم‌ها) یا (ابزارها) |
| 2 | (توصیه(و  (توصیه‌کننده) و ((سیستم‌ها)) یا  ابزارها و (کد) و  (بازدیدها) یا (داوران) |

جدول 3.1: رشته‌های جستجو

در هر مرحله از فیلتراسیون، تعدادی مقاله تحقیقاتی فیلتر شد. پس از خواندن عنوان مقاله، 19 مقاله فیلتر شد. از بین این مقالات به‌دست‌آمده، 21 مقاله پس از مطالعه چکیده مقالات فیلتر شد. در نهایت، 7 مقاله پس از خواندن متن کامل مقالات تحقیقی تصفیه شد. در پایان 14 مقاله به دست آمد.

## 5-3 نتایج

پس از خواندن متن کامل مقاله‌های تحقیقاتی فیلتر شده، نه سیستم توصیه‌کننده مرور کد شناسایی شد. جدول 3.2 این سیستم‌ها و داده‌های مورداستفاده سیستم را برای ارائه توصیه می‌کند. بقیه این بخش شرح هر یک از این سیستم‌ها را ارائه می‌دهد.

|  |  |
| --- | --- |
| نوع داده | عنوان مقاله پژوهشی |
| تاریخچه مرور کد | توصیه خودکار داوران در بررسی کد مدرن |
| متعهد می­شود | مرور کد مدرن: مطالعه موردی در گوگل |
| وضعیت هر منتقد یا نویسنده را دنبال می‌کند | : An ویژگی های بررسی کد مفید  مطالعه تجربی در مایکروسافت |
| شباهت مسیر پرونده  (FPS) | یک مطالعه در مقیاس وسیع در مورد توصیه مرورگر کد منبع |
| پروژه متقابل مرتبط و تجربه فناوری | correct: توصیه مرورگر کد در GitHub بر اساس تجربه پروژه و فناوری |
| استخراج متن و محل فایل | چه کسی باید این تغییر را مرور کند؟ |
| بر اساس مشخصات | توصیه مبتنی بر مشخصات مرورگران کد |

جدول 3.2: مقاله‌های استخراج شده و داده‌های مورداستفاده

## 6-3 سیستم‌های پیدا شده

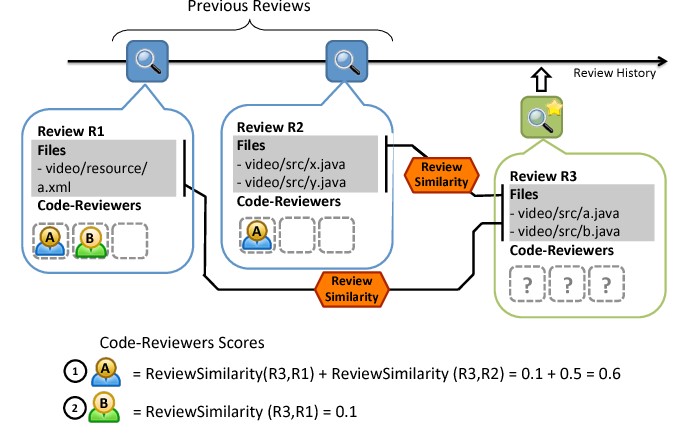
### 1-6-3 بازنگر(REVFINDER )

تعدادی CRRS براساس سیستم­های مسیر فایل (FPS)رویکرد مبتنی بر مکان فایل ارائه شده است. پیشنهاد REVFINDER که از روش توصیه مرورگر مبتنی بر مکان فایل پیروی می کند. شهود پشت این رویکرد این است که چندین فایل با یک مکان/مسیر فایل مشابه توسط مرورگران کد مجرب مشابه بررسی و مدیریت می شوند.

REVFINDER شامل دو بخش است:

* الگوریتم رتبه‌بندی مرورگران کد: از الگوریتم رتبه‌بندی مرورگران کد برای ارزیابی نمرات مرورگران کد بر اساس شباهت مسیرهای فایل‌هایی که قبلاً بررسی شده بودند، استفاده کردند.
* تکنیک ترکیبی

به‌منظور محاسبه شباهت مسیر فایل، نویسندگان از چهار تکنیک مقایسه رشته‌ای پیشرفته استفاده کردند:



شکل 3.1: مثال محاسبه الگوریتم رتبه بندی Code-Reviewers

1. طولانی­ترین پیشوند مشترک (LCP)

LCP اجزای مسیر فایل رایج را که در هر دو مسیر فایل از ابتدا تا انتها ظاهر می‌شود، محاسبه می‌کند.

2. طولانی­ترین پسوند مشترک (LCS)

LCS اجزای مسیر فایل مشترکی را که در هر دو مسیر فایل از انتهای هر دو مسیر فایل ظاهر می‌شوند، محاسبه می‌کند.

3. طولانی­ترین زیر رشته مشترک (LCSubstr)

LCSubstr اجزای مسیر پرونده رایج را که در هر دو مسیر پرونده به‌صورت متوالی ظاهر می‌شوند، اما در هر موقعیتی در مسیرهای فایل ظاهر می‌شوند، محاسبه می‌کند.

4. طولانی­ترین متعاقب مشترک (LCSubseq)

LCSubseq اجزای مسیر فایل مشترکی را که در هر دو مسیر پرونده‌ها به ترتیب نسبی ظاهر می‌شوند، محاسبه می کند[D. Gusfield,2017]

### cHRev 2-6-3

تعدادی CRRS بر اساس بررسی‌های گذشته ساخته شده‌اند و Zanjani, Kagdiو Birdیک چنین سیستم توصیه­ای به نام cHRev ایجاد کرده­اند. cHRev به طور خودکار مرورگران کد را براساس مشارکت‌های قبلی خود در بررسی‌های قبلی خود توصیه می‌کند. cHRev مخفف عبارت review code Histories over other types of previous information برای توصیه به Reviewers است.

این سیستم توصیه دارای دو ویژگی اصلی است:

1. بررسی‌کنندگان کد توصیه شده توسط cHRev ممکن است لزوماً در توسعه‌بخشی از کد منبع که در حال بررسی آن هستند مشارکت نداشته باشند، اما ممکن است بر روی کد منبع کارکرده باشند که به طور غیرمستقیم به منبع، منبع مورد بررسی بستگی دارد.

2. تخصص در طول زمان تغییر می‌کند و بنابراین تکرار و تکرار باید هنگام جستجوی مناسب‌ترین مرورگر کد در نظر گرفته شود.

فرایند مورد استفاده cHRev شامل سه مرحله است:

1. کد منبع را که باید بازبینی شود استخراج کنید.

2. تخصص بازبینی را براساس جزئیات مختلف مانند اینکه چه کسی، چه تعداد و چه زمانی در گذشته انجام شده است، تدوین کنید.

3. یک لیست رتبه‌بندی از داوران داوطلب بر اساس فایل‌های کد منبع در مرحله 1 و مشارکت تجمعی داوران در مرحله 2 به دست آورید و سپس با استفاده از یک پارامتر تعریف شده توسط کاربر، تعداد مترهای برتر کاندیداها را از لیست به‌دست‌آمده توصیه کنید.

به‌منظور آزمایش اثربخشی رویکرد آنها، زنجانی، کادی روش آنها را با REVFINDER ، xFinder و RevCom مقایسه کرد. مشخص شد که cHRev از نظر دقت و فراخوانی توصیه‌های دقیق‌تری را ارائه می‌دهد. همچنین، مشاهده شد که cHRev عملکرد بهتری نسبت به REVFINDER دارد، از نظر مرورگرها بر اساس پرونده‌های دارای نام و مسیر مشابه و xFinder که به داده‌های مخزن کد منبع بستگی دارد. مشخص شد که cHRev از نظر آماری معادل RevCom است که نیاز به بررسی و تعهدات قبلی دارد. [M. B. Zanjani, H. H. Kagdi,2016]

### CoRReCT 3-6**-3**

Rahman, Roy, Collins پیشنهاد یک سیستم توصیه بازبینی کد به نام CoRReCT (توصیه بازبینی­کننده کد بر اساس تجربه پروژه و پروژه) که هدف آن توصیه بازبینی­کننده‌های کدنویسی بر اساس سابقه کاری مرتبط با پروژه و همچنین تجربه توسعه­دهندگان در یک فناوری تخصصی خاص مرتبط با درخواست کشش این دو منبع اطلاعاتی برای تعیین تجربه برنامه‌نویس برای بررسی کد استفاده شد. ایده اساسی پشت CRRS پیشنهادی آنها این است که اگر درخواست های pull قبلی دارای کتابخانه‌ها یا فناوری‌های تخصصی مشابه با درخواست های pull فعلی باشند، پس مرورکنندگان کد که آن درخواست های pull را بررسی کرده­اند می‌توانند به­عنوان مرورگرهای احتمالی کد برای درخواست­های pull فعلی در نظر گرفته شوند.

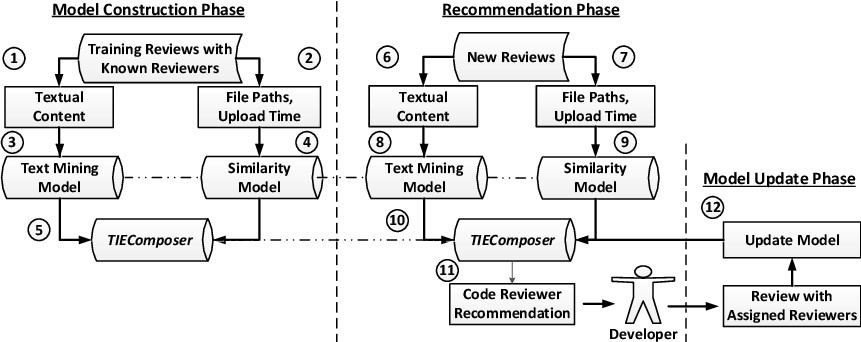
باتوجه‌به ایده پیشنهادی نویسندگان، توسعه‌دهندگان باتجربه بیشتر در کتابخانه‌های خارجی و فناوری‌های تخصصی پذیرفته شده در پرونده‌های تغییر در مجموعه نشانه‌های درخواست‌های فعلی کشش، نسبت به مواردی که تجربه کمتری دارند، انتخاب مناسب‌تری برای انجام بازبینی کد محسوب می‌شوند.

### TIE 4-6-3

ایده پشت این رویکرد تجزیه‌وتحلیل محتوای متنی در یک درخواست بازبینی با استفاده از یک مدل کاوی متنی افزایشی و محاسبه شباهت بین مسیرهای فایل بررسی شده و مسیرهای فایل تغییریافته با استفاده از یک مدل شباهت است که از مزایای استخراج متن و روش مبتنی بر مکان فایل برای توصیه مرورگر کد استفاده می‌کند.

معماری کلی TIE به سه مرحله تقسیم می‌شود:

* ساخت مدل
* توصیه
* به‌روزرسانی مدل



شکل 3.2: معماری TIE

1. **فاز ساخت مدل**

مرحله ساخت مدل شامل یک مدل ترکیبی به نام TIECOMPOSER که با استفاده از بررسی‌های تاریخی داوران شناخته شده ساخته شده است. در این مرحله، سیستم TIE ابتدا مرورهای آموزشی داوران شناخته شده را از محتوای متنی مرورها و مسیرهای فایل گذشته و همچنین زمان بارگذاری جمع‌آوری می‌کند. در مرحله بعد، TIE یک مدل استخراج متن را بر اساس داده‌های متنی پردازش شده با استفاده از تکنیک طبقه‌بندی متن ایجاد می‌کند. شهود پشت حالت داده‌کاوی این است که احتمالاً همان بازبین‌ها تغییرات را با اصطلاحات یا کلمات مشابه مرور می‌کنند.

TIE همچنین از یک رویکرد مبتنی بر مکان برای آگاهی از زمان استفاده می‌کند که هدف آن محاسبه شباهت بین بررسی‌های جدید و تاریخی است. این شباهت بین مسیرهای تغییریافته فایل (یعنی مسیرهایی از پرونده‌هایی که در درخواست بازبینی جدید تغییر کرده یا اصلاح شده‌اند) و مسیرهای فایل مرور شده (یعنی مسیرهای فایل‌هایی که در بررسی‌های تاریخی بررسی شده‌اند) محاسبه می‌شود. شهودی که در پشت رویکرد مبتنی بر مکان قرار دارد این است که همان مرورگران تمایل دارند فایل‌ها یا پرونده‌های مشابه را با مسیرهای مشابه مرور کنند.این دو مدل برای ساختن مدل TIECOMPOSER با هم ترکیب شده‌اند.

**2. مرحله توصیه**

در این مرحله،TIE برای توصیه بازبینی­کنندگان کد برای درخواست بازبینی جدید تعیین نشده استفاده می‌شود. TIE ابتدا توضیحات تغییر، مسیرهای فایل و زمان بارگذاری را برای بررسی­های تاریخی در "مرحله ساخت مدل" انجام می­دهد. برای مرحله بعدی، داده های متنی از توضیحات استخراج شده و به عنوان ورودی در مدل داده­کاوی ساخته شده در "مرحله ساخت مدل" استفاده می­شود. به طور مشابه، سیستم مسیرهای فایل و زمان بارگذاری را در مدل تشابه ایجاد شده در "مرحله ساخت مدل" وارد می­کند. سپس این دو مدل لیستی از مرورگران کد را ارائه می‌دهند و این دو لیست سپس با استفاده از مدل TIECOMPOSER ساخته شده در "مرحله ساخت مدل" ترکیب می‌شوند.

**3. مرحله به‌روزرسانی مدل**

در مرحله به روزرسانی مدل، سیستم TIE با استفاده از مرورگران کد اختصاص داده شده به روز می­شود. در عمل، توسعه­دهندگان به طور معمول لیست بازبین­های احتمالی را بررسی می­کنند و سپس یک درخواست کشش جدید را به گروهی از داوران اختصاص می­دهند.

### CodeFlow 5-6-3

Bosu، Greilerبا انجام مصاحبه با توسعه­دهندگان و همچنین تجزیه‌وتحلیل نظرات بازبینی پنج پروژه مایکروسافت که با استفاده از CodeFlow CRRS انجام شده بود، در مایکروسافت ویژگی تجزیه‌وتحلیل کدهای مفید را در مایکروسافت انجام دادند. مطالعه در سه مرحله انجام شد. ابتدا، آنها با انجام مصاحبه‌ای با توسعه دهندگان، یک مطالعه اکتشافی انجام دادند تا تفسیر آنها از "مفید" را در زمینه بررسی کد درک کنند. ثانیاً، آنها طبقه‌بندی‌کننده‌ای برای تفکیک نظرات "مفید" و "بی‌فایده" با استفاده از داده‌های مصاحبه‌ها ایجاد نمی‌کنند. در نهایت، آنها طبقه‌بندی خود را برای پنج پروژه مایکروسافت اعمال کردند تا نظرات "مفید" و "بی‌فایده" را از هم متمایز کنند.

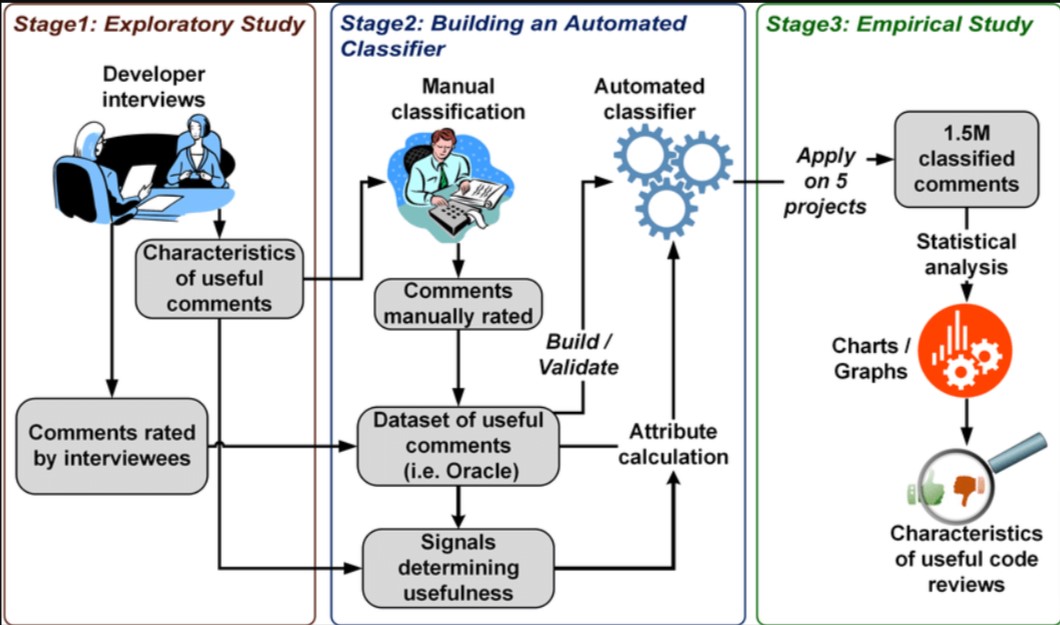
M. Greiler,Bose, and C. Bird,2015]].

همچنین تجزیه‌وتحلیل نظرات بازبینی پنج پروژه مایکروسافت که با استفاده از CodeFlow CRRS انجام شده بود، در مایکروسافت ویژگی تجزیه‌وتحلیل کدهای مفید را در مایکروسافت انجام دادند. مطالعه در سه مرحله انجام شد. ابتدا، آنها با انجام مصاحبه‌ای با توسعه‌دهندگان، یک مطالعه اکتشافی انجام دادند تا تفسیر آنها از "مفید" را در زمینه بررسی کد درک کنند. ثانیاً، آنها طبقه‌بندی‌کننده‌ای برای تفکیک نظرات "مفید" و "بی‌فایده" با استفاده از داده‌های مصاحبه‌ها ایجاد نمی‌کنند. در نهایت، آنها طبقه‌بندی خود را برای پنج پروژه مایکروسافت اعمال کردند تا نظرات "مفید" و "بی‌فایده" را از هم متمایز کنند. [ Bosu M. Greiler, Bird,2015[

**روش تحقیق سه مرحله ای:**

گردش کار CodeFlow نسبتاً ساده است.

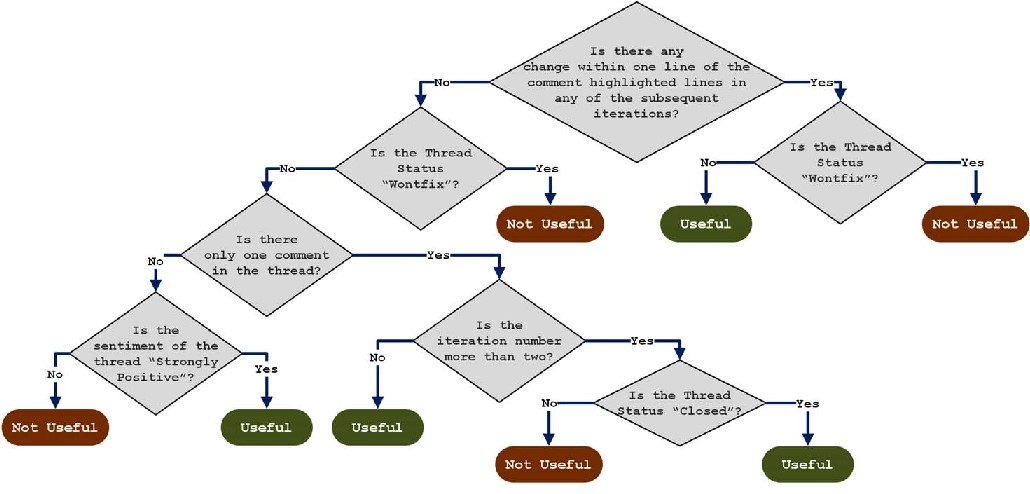
1. ابتدا، به تشخیص نظرات بازبینی کد مفید و غیرمفید بر اساس مصاحبه با توسعه‌دهندگان کمک کرد. نویسنده تغییر نظر را ارسال می‌کند و درخواستی از طریق ایمیل به داور اطلاع داده می‌شود.
2. نویسندگان یک طبقه‌بندی خودکار با استفاده از یافته‌های به‌دست‌آمده از مرحله اول ایجادکردن به‌منظور ایجاد کلاس بهتر، نویسندگان نظرات بازبینی را به‌صورت دستی به دودسته مفیدو غیرمفیدطبقه‌بندی کردند. بازبین می‌تواند تغییر در خود ابزار را مرور کند.
3. هنگامی که یک داور می‌خواهد در مورد یک خط یا بلوک کد نظر دهد، منتقد آن قسمت از کد را برجسته و اضافه می‌کند. این نظرات به‌عنوان موضوعاتی که در آن بحث شروع می‌شود و همچنین نقاط تعامل برای افرادی که در بررسی مشارکت دارند، ظاهر می‌شود.



شکل 3.3: روش تحقیق سه‌مرحله‌ای

هر یک از این موضوعات دارای وضعیتی هستند که شرکت‌کنندگان می‌توانند در طول دوره بررسی تغییر دهند. این وضعیت در ابتدا "فعال" است و باگذشت زمان می‌تواند به "در انتظار" ، "حل شده" ،"برطرف نمی‌شود" و "بسته" است. در CodeFlow، هر به‌روزرسانی "تکرار" نامیده می‌شود و چرخه بازبینی دیگری را تشکیل می‌دهد؛ بنابراین، قبل از ادغام تغییر کد در مخزن کد منبع، ممکن است تکرارهای متعددی وجود داشته باشد.

بر اساس این ویژگی‌ها و دسته‌ها، "مدل درخت تصمیم برای طبقه‌بندی نظرات مفید" مطابق شکل زیر ساخته شد.



شکل 3.4: مدل درخت تصمیم‌گیری برای طبقه‌بندی نظرات مفید

بر اساس گره‌های تصمیم‌گیری، نظرات به‌عنوان مفید یا بی‌فایده طبقه‌بندی می‌شوند. به‌منظور ارزیابی روش پیشنهادی، نویسندگان از نظرات پنج پروژه بزرگ مایکروسافت که شامل Azure ،Bing ،Visual Studio ، Exchange و Office هستند، استفاده کردند. بر اساس نتایج، نویسندگان موارد زیر را نتیجه گرفتند:

1. توسعه‌دهندگانی که درگذشته تغییراتی را انجام داده یا یک قطعه کد یا یک مصنوع را بررسی کرده‌اند، نظرات مفیدتری ارائه می‌دهند.
2. تفاوت قابل‌توجهی در سودمندی بین نظرات وجود دارد (یعنی آن نظراتی که کلماتی مانند "ثابت"، "اشکال" یا "حذف" به‌عنوان نظرات "مفید" در نظر گرفته شده‌اند) که توسط بازبینان در همان تیم ساخته شده است و نظرات نویسنده و منتقد از تیم‌های مختلف است.
3. تعداد نظرات مفید در طول زمان برای چهار مورد از پنج پروژه افزایش یافت و دلیل این امر افزایش تجربه مرورگران باگذشت زمان در نظر گرفته شد.

در زیر پیامدهای نتایج برای شرکت‌کنندگان در مرور کد و همچنین برای محققان آمده است:

1. این مطالعه نشان داد که تعداد مفید بودن نظرات مرور کد باتجربه توسعه‌دهنده کد افزایش‌یافته است پایه.

2. این مطالعه همچنین نشان داد که با افزایش تعداد فایل‌ها، اثربخشی بررسی‌ها کاهش می‌یابد. پیشنهاد شد که توسعه­دهندگان باید تغییرات کوچک‌تر را با تعداد بیشتری فایل برای بررسی ارسال کنند.

3. تراکم مفید بودن نظر می‌تواند توسط تیمی از توسعه‌دهندگان برای شناسایی مناطقی که بررسی کد در آنها کمتر مؤثر است استفاده شود. [A. Bosu, M. Greiler, and C. Bird,2015]

## 7-3 خلاصه

بر اساس مطالعه مروری بر ادبیات انجام شده، ما هفت سیستم توصیه بازبینی کدگذار را یافتیم: cHREv، CoRReCT ، CRRS مبتنی بر مشخصات،RevFinder ، CodeFlow ، TIE و CRITIQUEاین سیستم‌ها بر اساس دو بعد تقسیم می‌شوند:

* منبع داده مورداستفاده برای ساخت سیستم
* نوع پروژه مورداستفاده برای ارزیابی سیستم.

# فصل چهارم

# 4. تجزیه‌وتحلیل موضوع انتخابی

## 1-4 مقدمه

در این گزارش راه‌حل‌های برای سیستم بازبینی کد (CRRS) وعوامل هنگام ساخت آنها در نظر گرفته می‌شود. بر اساس این می‌توان آنها را طبقه­بندی کرد که به درک بیشتر ما از ویژگی­های مهم CRRS و بهبود CRRSموجود است. در این گزارشCRRS و تحقیقات بیشتری در مورد شیوه‌ها و رویه‌های بازبینی کد انجام شده است با هدف تجزیه و تحلیل راه حل­های ارائه شده توسط "مرورگر کد" که موجود است.

## 2-4 روش پژوهش تحقیق:

در پایان‌نامه مورد بررسی به‌منظور انجام نظرسنجی از مهندسان نرم‌افزار برای تعیین نیازهای اطلاعاتی برای مرورگران کد، از مراحل زیر برای اطمینان از نتایج صحیح، غیر مغرضانه و دقیق استفاده کرده است.

### 1-2-4 بررسی غربالگری

نظرسنجی انجام شده برای مهندسان نرم‌افزار به دو قسمت تقسیم می‌شود:

1. یک بررسی غربالگری است. ما از نظرسنجی غربالگری استفاده کردیم تا مطمئن شویم از اعضای توسعه محصول نرم‌افزاری که در سیستم‌های توصیه بازبینی کد تجربه دارند و بنابراین می‌توانند اطلاعات دقیق و بی‌طرفانه ارائه دهند، پاسخ می‌دهیم.
2. یک نظرسنجی جمعیت شناختی و تجربه CRRS به افرادی که مرحله غربالگری را پشت سر گذاشته­اند داده شد، یعنی شرکت­کنندگان حداقل دو سال سابقه کار و تجربه استفاده از CRRSها را داشتند. سوالات پایانامه مورد بررسی به ما کمک کرد تا اطلاعات مورد نیاز مرورگران کد را درک کنیم، آنها چه ویژگی‌هایی را در سیستم‌های توصیه­کننده مرورگر کد مهم می‌دانند و چه ویژگی‌هایی را در سیستم‌های موجود مفقود می‌دانند.

مطالعه تحقیقاتی در سه مرحله انجام شد که در آن اولین گام به تشخیص نظرات بازبینی کد مفید و غیرمفید بر اساس مصاحبه با توسعه­دهندگان کمک کرد. مصاحبه فردی نیمه‌ساختاریافته از توسعه­دهندگان داشتن سطوح مختلف تجربه در بررسی کد و توسعه کد از چهار روش مختلف مایکروسافت طی پروژه انجام شد. از مصاحبه‌شوندگان خواسته شد تا نظرات را از مقیاس 1- 3 (1- مفید، 2- تا حدودی مفید و 3- مفید) امتیاز دهند. نتایج مصاحبه نشان داد که 69 comments از نظرات مرور "مفید" یا "تا حدودی مفید" بودند. توریم و نظرات بررسی است که نشان نقص عملکردی به‌عنوان در نظر گرفته شد مفید نظر. از سوی دیگر، به نظر که به دسته‌بندی تعلق در کد مستندات، نمایش تصویری از کد (به‌عنوان‌مثال خط خالی یا دندانه‌دار)، ارگان ization از کد (به‌عنوان‌مثال چگونه قابلیت به روش تقسیم) و رویکرد راه‌حل بود به‌عنوان در نظر گرفته تا حدودی مفید. همه نظراتی که یا مثبت کاذب بودند (به‌عنوان‌مثال به دلیل عدم تخصص هنگامی که داور مشکلی را در کد نشان می‌دهد) یا در هیچ طبقه‌ای قرار نگرفت همان‌طور که قبلاً ذکر شد به‌عنوان نظرات غیرمفید طبقه بندی شدند.

در مرحله دوم، نویسندگان یک طبقه‌بندی خودکار با استفاده از یافته‌های به‌دست‌آمده از مرحله اول ایجاد کردند. به‌منظور ایجاد کلاس بهتر، نویسندگان نظرات بازبینی را به‌صورت دستی به دودسته مفید و غیرمفید طبقه‌بندی کردند. نظراتی که در مطالعه اکتشافی تا حدی مفید طبقه‌بندی‌شده‌اند، در این مرحله دوم در دسته مفید قرار گرفتند. بر اساس مصاحبه و تجزیه‌وتحلیل دستی، 8 ویژگی بعدی نظرات مشخص شد.

ما 30 پرس‌وجو در مورد نظرسنجی خود به داشتیم، اما تنها 18 مورد از آنها به جلو حرکت کردند و به بررسی غربالگری ما پاسخ دادند. از بین این 15 پاسخ، ما 4 پاسخ را فیلتر کردیم که با استفاده از نظرسنجی غربالگری حداقل معیارها را نداشتند. در زیر نتایج به‌دست‌آمده برای نظرسنجی سیستم‌ها و ابزارهای توصیف مرورگر جمعیت‌شناختی و کد ارائه شده است.

1. **نقش شغلی شرکت‌کنندگان**

بر اساس نتایج به‌دست‌آمده، مشخص شد که اکثر شرکت‌کنندگان یا توسعه‌دهندگان، یا مهندسان نرم‌افزار یا برنامه‌نویس بودند.

|  |  |
| --- | --- |
| رشته | درصد |
| توسعه دهنده/برنامه نویس/مهندس نرم افزار | 72.73 |
| سرپرستی تیم | 9.09 |
| مهندس DevOps/مهندس زیرساخت | 9.09 |
| مالک محصول | 9.09 |

جدول 4.1: نقش‌های شغلی شرکت‌کنندگان

1. **موقعیت جغرافیایی شرکت‌کنندگان**

اکثر شرکت کنندگان از آسیا بودند و درصد معادل باقیمانده افراد از آمریکای شمالی و آمریکای جنوبی بودند.

|  |  |
| --- | --- |
| رشته | درصد |
| آمریکای جنوبی | 9.09 |
| شمالی عامری CA | 9.09 |
| آسیا | 81.82 |

جدول 4.2: موقعیت جغرافیایی شرکت‌کنندگان

**3. اندازه تیم پروژه**

از همه شرکت‌کنندگانی که در این مطالعه شرکت کردند، 45.45٪ آنها به ترتیب در تیمی از 2-7 نفر و 8-12 نفر کارکرده‌اند/در حال کار هستند درحالی‌که 9.09٪ از افراد کارکرده‌اند/در گروهی بیش از 40 نفر.

|  |  |
| --- | --- |
| رشته | درصد |
| 2-7 نفر | 45.45 |
| 8-12 نفر | 45.45 |
| بیش از 40 نفر | 9.09 |

جدول 4.3: اندازه تیم پروژه

**4.توزیع تیم**

همچنین در پایان‌نامه مورد بررسی اطلاعاتی در مورد نحوه توزیع تیم‌ها جمع‌آوری کردند، به این معنی که آنها در تیم‌هایی کار می‌کنند که در محل مشترک یا توزیع شده‌اند. مشاهده شد که 45.45 of از تیم‌ها در محل مشترک، 18.18 were توزیع شده و 36.36 were ترکیبی از هر دو مکان مشترک و توزیع شده بود.

|  |  |
| --- | --- |
| رشته | درصد |
| در محل مشترک | 45.45 |
| توزیع شده است | 18.18٪ |
| هر دو | 36.36 |

جدول 4.4: توزیع تیم

5. **آشنایی با** CRRS **در بین شرکت‌کنندگان**

اکثر شرکت‌کنندگان در نظرسنجی پایان نامه مورد بررسی با GitHub/GitLab آشنا بودند. از سوی دیگر، هیچ کس با Gerrit Code Review System (Chromium) و ReviewBoard آشنا نبود. مشاهده شد که 47.62٪ از شرکت کنندگان با GitHub/GitLab ، 23.81٪ از شرکت کنندگان با BitBucket ، 14.29٪ از شرکت کنندگان با Code Flow Review Tool (مایکروسافت)، 9.52٪ با سایر ابزارهای بازبین(که شامل (SVN است آشنا بودند. و 4.76 of از شرکت کنندگان با Phabricator آشنا بودند.

|  |  |
| --- | --- |
| CRRS | درصد انتخاب ها |
| سیستم بازبینی کد Gerrit (Chromium) | 0٪ |
| GitHub/GitLab | 47.62 |
| ابزار مرور کد جریان (مایکروسافت) | 14.29 |
| تابلوی بررسی (VMware) | 0٪ |
| سازنده | 4.76 |
| سطل بیت | 23.81 |
| دیگر | 9.52 |

جدول 4.5: آشنایی با CRRS در بین شرکت‌کنندگان

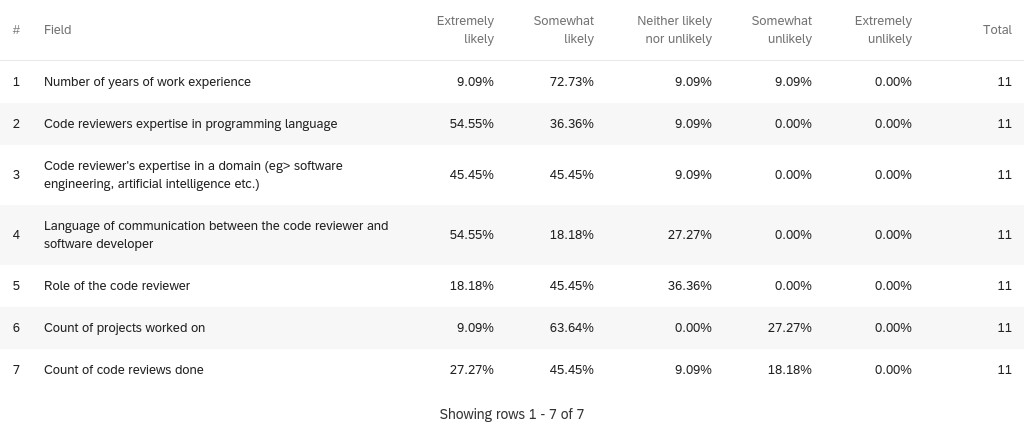
6. **مفید بودن ویژگی های CRRS**

در پایان نامه مورد بررسی تعدادی از ویژگی‌های CRRS را برای شرکت کنندگان خود لیست کرده­اند و از آنها خواسته همه آن ویژگی‌هایی را که برای آنها مفید بوده است انتخاب کنند. موارد زیر ویژگی‌هایی است که برای شرکت کنندگان در نظر گرفته و درصد مشارکت کنندگان آن را مفید دانستند.

7. **معیارهای انتخاب مرورگر کد**

هنگام انتخاب مرورگر کد عوامل متعددی باید موردتوجه قرار گیرد:

* تعدادی سال سابقه کار
* تخصص بازبینی کنندگان کد در زبان برنامه‌نویسی
* تخصص مرورگر کد در یک حوزه (به‌عنوان‌مثال: مهندس نرم‌افزار هوش مصنوعی و غیره)
* زبان ارتباط بین مرورگر کد و توسعه‌دهنده نرم‌افزار
* نقش مرورگر کد
* تعداد پروژه‌هایی که روی آنها کار شده است
* تعداد بررسی کد انجام شده



شکل 1-4 : معیارهای انتخاب مرورگر کد

**8. رابط کاربری** **(UI) CRRS:**

به‌غیراز ویژگی‌هایCRRS ، همچنین در مورد ویژگی‌های مختلف که می­تواند برای CRRS مفید واقع شود، با استفاده بیشتر از تجربه کاربری، قابل‌دسترسی و راحت نشان داد. درصد افرادی که ویژگی­های زیر را مفید دانسته­اند به شرح زیر با توزیع نمودار پای زیر نشان‌داده‌شده است:

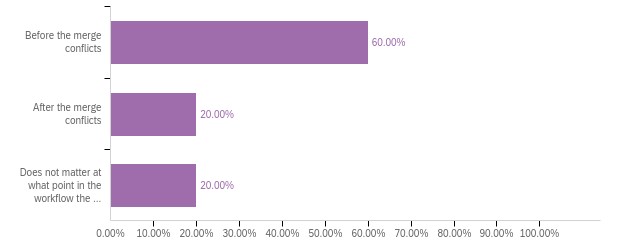
* ارائه خط لوله­ای که نشان می‌دهد پروژه در چه مرحله­ای است یعنی ساخت، آزمایش، بازنویسی کد، استقرار و غیره : 31.25٪ از شرکت کنندگان این ویژگی رابط کاربری را مفید و راحت‌تر می‌دانند.
* وجود پنل کاربری برای هر فرد که داده‌های آماری کلیه اقدامات انجام شده را نشان می‌دهد (مانند تعداد تعهدات، تعداد بررسی کد انجام شده، تعداد خطاها/هشدارهای کد در پروژه فعلی و غیره) 28 درصد از شرکت‌کنندگان این دیدگاه برای جستجوی جزئیات و اقدامات انجام شده توسط هر فرد مفید بود.
* هنگام تغییر کد، بحث کدگذاری جدید و قدیمی با کد 15 درصد از شرکت‌کنندگان این ویژگی را مفید می‌دانند تا روند بازبینی کد را برای تازه‌کار و توسعه‌دهنده آسان‌تر کند.
* انتقال کد با استفاده از طرح رنگی با نام برنامه‌نویس 12.5 درصد از شرکت‌کنندگان این ویژگی را مفید دانستند.
* گزینه ای برای انتخاب یک "شاخه/پرونده" خاص در یک پروژه برای حفظ گردش­کار سیستماتیک و روش بازبینی کد سازماندهی شده  12.5 درصد از شرکت­کنندگان با سازماندهی بیشتر فرآیند بازبینی کد، این ویژگی را مفید دانستند. نتایج در شکل 1 نشان داده شده است.



شکل 4-2 انواع ويژگي­هايCRRS

**9**. **ترجیح زمان توصیه برای بررسی کد:**

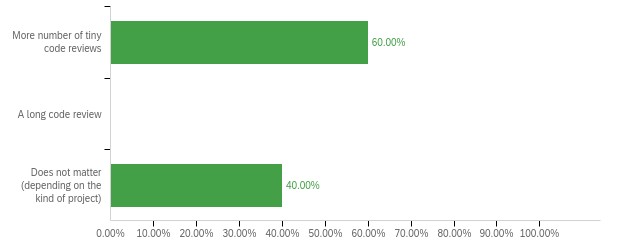
توصیه بازبینی کد می‌تواند در مراحل مختلف فرایند توسعه نرم‌افزار انجام شود که شامل قبل از درگیری ادغام، پس از ناسازگاری ادغام یا در هر نقطه از گردشکار می‌شود. مشاهده شد که 60 of از شرکت‌کنندگان معتقد بودند که توصیه کد باید قبل از درگیری ادغام انجام شود، 20.00 them از آنها معتقد بودند که باید پس از درگیری ادغام انجام شود درحالی­که تا 20.00 them از آنها مهم نیست که در کدام نقطه از گردشکار بررسی کد انجام شده است.



شکل 3-4 : اولویت زمانی که توصیه بازبینی کد وجود دارد

**10. نوع مرور کد**

از شرکت‌کنندگان خواسته شد که چه نوع بازبینی کد را ترجیح دهند که شامل بسیاری از بررسی‌های کد کوچک‌تر یا مرور طولانی کد یا بستگی به نوع پروژه دارد. مشاهده شد که 60.00 of از شرکت‌کنندگان ترجیح می‌دهند بسیاری از بررسی‌های کد کوچک‌تر را انجام دهند درحالی‌که بقیه (40.00٪) از شرکت‌کنندگان احساس می‌کردند که مهم نیست. هیچ یک از شرکت‌کنندگان ترجیح می‌دهند که بررسی طولانی کد را انجام دهند.



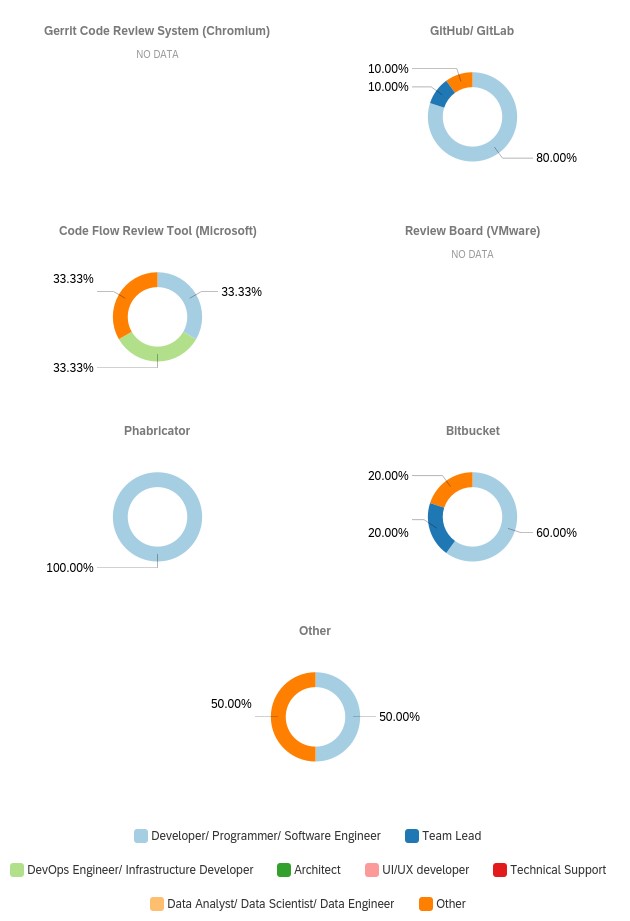
شکل 4-4 : نوع مرور کد

## 3-4 برخی از روندها و الگوهای مشاهده شده

بر اساس نتایج به‌دست‌آمده، برخی از الگوها و روندها بین نتایج دو یا چند سؤال نظرسنجی یافت شد. برخی از روندهایی که مشاهده کردیم به شرح زیر است:

1. **نوع CRRS مورد استفاده و نقش شغلی**

بر اساس مشاهدات ما، مشخص شد که توسعه‌دهنده تقریباً از تمام سیستم­های CRRS که در سوال ذکر کرده بودیم آگاه بود در­حالی­که DevOps Engineer فقط از یک CRRS مطلع بود، سرپرست تیم از 2 CRRS و محصول مطلع بود. مالک در مورد 4 CRRS می‌دانست.



شکل 5-4 : نوع CRRS و نقش شغلی

ما نتایج زیر را برای هر یک ازCRRSهای زیر به همراه توزیع نمودار پای در شکل4.4 به دست آوردیم.

* **Github/GitLab** :از شرکت­کنندگانی که با این CRRS آشنا بودند، 10٪ از شرکت‌کنندگان سرپرست تیم بودند، 10٪ دیگر صاحبان محصولات و بقیه (80٪) توسعه­دهندگان/برنامه‌نویسان/مهندسان نرم‌افزار بودند.
* **دوم ابزار بازبینی کد جریان (مایکروسافت):** از شرکت­کنندگان که با این CRRS آشنا بودند، 33.33٪ توسعه­دهندگان/برنامه‌نویسان، مهندسان نرم­افزار،DevOps Engineer/Infrastructure Developer ودیگران بودند
* **Phabricator:**توسعه­دهندگان/برنامه‌نویسان/مهندسان نرم‌افزار تنها افرادی بودند که با Phabricator آشنا بودند
* **:BitBucket** از شرکت کنندگانی که با این CRRS آشنا بودند,60.00٪ آنها توسعه­دهندگان/برنامه نویسان/مهندسان نرم­افزار بودند، 20.00٪ آنها رهبران تیم و دیگران بودند.
* **دیگران**: تعدادی CRRS دیگر در بازار وجود دارد و از شرکت‌کنندگان خواسته شد در صورت اطلاع از CRRSهای دیگری که در نظرسنجی ذکر نشده‌اند، آن گزینه را انتخاب کنند. زیرا، ما 50.00 respons پاسخ از توسعه­دهندگان و دیگران دریافت کردیم.

از بین شرکت‌کنندگانی که در نظرسنجی ما شرکت کردند، هیچ یک از آنها با دو مورد از سیستم‌ها/ابزارهای توصیه بازبینی کد آشنا نبودند. ReviewBoard (VMware) Gerrit Code System Review (Chromium)

**2. ویژگی های CRRS و نقش شغلی**

ما تعدادی ویژگی CRRS را برای شرکت‌کنندگان مطرح کردیم که آنها معتقد بودند استفاده از CRRSها راحت‌تر و آسان‌تر است. شکل 4.7 نشان می‌دهد که نقش شغلی کدام ویژگی مفید است.



شکل 6-4 : ویژگی­های CRRS و نقش شغلی

1. بررسی کد قبل از انجام:

مشاهده شد که از همه شرکت‌کنندگانی که موافقت کردند که این ویژگی را مهم بدانند 60٪ آنها توسعه‌دهنده بودند، 20٪ مهندس DevOps/توسعه­دهنده زیرساخت و دیگران (یعنی صاحب محصول).

1. بحث کد با نسخه‌های قدیمی و جدید که برای نشان‌دادن تغییر کد مشخص می‌شوند:

از بین همه شرکت­کنندگان، 75 them از آنها توسعه­دهنده، 12.50 them از آنها رهبران تیم و مهندس/توسعه­دهنده زیرساخت DevOps بودند که این ویژگی CRRS را مهم می­دانستند.

1. پیشنهاد بهبود کد توسط مرورگر کد (به جز اشاره به خطاهای کد):

از همه شرکت‌کنندگانی که این ویژگی را مفید دانستند، 40.00٪ آنها توسعه‌دهنده، 20.00٪ آنها رهبران تیم، مهندس DevOps/توسعه­دهنده زیرساخت و دیگران (صاحب محصول).

1. اولویت‌بندی تغییرات کد بر اساس میزان اهمیت آن و تأثیر آن بر عملکرد نرم‌افزار: از بین همه شرکت‌کنندگان، فقط صاحب محصول این ویژگی را مفید دانست.
2. ادغام نرم‌افزارهای ردیابی پروژه (مانند Trello ، Jira و غیره): برای این ویژگی، 71.43٪ توسعه دهندگان، 14.29٪ رهبران تیم و مدیر پروژه مفید دانستند.
3. ادغام ویرایشگر کد منبع (مانند Visual Studio ، Atom و غیره): فقط توسعه­دهندگان از همه شرکت‌کنندگان بودند که این ویژگی را به‌عنوان یک ویژگی مهم در نظر گرفتند.
4. ادغام بستر ارتباطات تجاری مان :(Slack) این­ویژگی فقط توسط­توسعه­دهندگان همه شرکت‌کنندگان که در نظرسنجی ما شرکت کردند مفید واقع شد.

فصل پنجم

# 5. جمع‌بندی و پیشنهادها

## 1-5 مقدمه:

در این بخش نتایج حاصل از تحقیق باتوجه‌به مباحث پژوهش پرداخته شده پیشنهادات ارائه می‌شود مزایا ومعایب نکات اصلاحی از دیدگاه شخصی و باتوجه‌به تجربه‌ام در زمینه تحقیقاتی در خصوص بهبود طرح مطرح می‌شود و به سؤالات تحقیق ما پاسخ می‌دهد.

## 2-5 نتایج حاصل از تحقیق

ما نتایج حاصل از بررسی طیف وسیعی از اعضای پروژه نرم‌افزار را در اینجا ارائه کردیم که در آن متوجه شدیم کدام ویژگی CRRS مفیدتر است، چه ویژگی­هایی در سیستم موجود وجود ندارد و چه عواملی هنگام انتخاب یک مرورگر کد مربوط مهم است. ما همچنین ترجیح توسعه­دهندگان/داوران را نسبت به نوع بررسی کد (بررسی طولانی یا کوتاه) و در چه مرحله ای از گردش کار باید بدست آوریم. به غیر از این، ما همچنین برخی از گرایش­ها و الگوها را بین استفاده از سیستم CRRS و ارتباط آن با اطلاعات جمعیتی مرورگر/توسعه­دهنده پیدا کردیم.

### 1-2-5 پاسخ به سؤالات تحقیق

برای درک تأثیر معنایی برنامه، باید پاسخ‌هایی به سؤالات مطرح شده در فصل اول داشته باشیم:

1. راهکارهای موجود برای سیستم‌های توصیه برای مرورگران کد چیست؟

2. هنگام ایجاد یک سیستم توصیه برای مرورگران کد، چه عواملی باید در نظر گرفته شوند؟

3. چگونه می‌توان سیستم‌های توصیه ای موجود برای مرورگران کد در ادبیات را دسته بندی کرد؟

4. ویژگی­های مهم سیستم توصیه برای مرورگران کد چیست؟

5. چگونه می‌توان سیستم‌های پیشنهادی موجود برای مرورگران کد را بهبود بخشید؟ به‌عبارت‌دیگر، چه ویژگی‌هایی در پیاده‌سازی‌های موجود برای سیستم‌های توصیه بازبینی کننده کد وجود ندارد؟

**جواب سؤال 1:**

در پایان‌نامه مورد بررسی تعدادی مقاله را، بررسی کرده و تعدادی از سیستم‌های توصیه بازبینی کد موجود را پیدا شدند. این سیستم‌ها/ابزارها شامل cHRev ، REVFINDER ، CoRReCT ، TIE ، CodeFlow ، ReviewBoard، Phabricator ،Gerrit ، rDevX و CRRS مبتنی بر پروفایل است که بر اساس عوامل متعددی/انواع داده توصیه‌هایی را ارائه می‌دهند. این نوع داده‌ها شامل سابقه مرور کد، شباهت مسیر پرونده، تجربه بین پروژه و فناوری مرتبط، استخراج متن و مکان‌یابی فایل و وضعیت ردیابی هر داور یا نویسنده است.

**جواب سؤال 2:**

فاکتور اصلی که هنگام ایجاد یک سیستم توصیه برای مرورگران کد باید موردتوجه قرار گیرد، معیارهای ارزیابی منبع داده یا نوع پروژه‌ای است که سیستم روی آن آزمایش شده است (یعنی منبع‌باز یا تجاری یا هر دو) وقتی نوبت به توصیه بازبینی کننده کد بر اساس نمایه بازبینی کننده می‌رسد، لازم است مشخصات مرورگر که شامل مرور قبلی و سابقه تعهد است به‌روز شود. به طور مشابه، هنگامی که صحبت از سابقه مرور گذشته می‌شود، مهم است که مخزن/مجموعه‌داده‌های بررسی‌های گذشته را به‌روز کنید تا در آینده بر اساس بررسی‌های گذشته، مرورگران کد مربوطه را توصیه کنید.

**جواب سؤال 3:**

سیستم‌های توصیه‌ای موجود بر اساس نوع داده طبقه‌بندی شده‌اند که شامل سابقه مرور کد، شباهت مسیر پرونده، تجربه پروژه و فناوری مرتبط، استخراج متن است و موقعیت فایل وضعیت ردیابی هر داور یا نویسنده.

cHRev یک سیستم توصیه مرورگر کد بود که مرورگران کد را بر اساس سابقه مرور کد توصیه می کرد. REVFINDER CRRS دیگری بود که مرورگران کد را بر اساس شباهت مسیر فایل توصیه می کرد. به طور مشابه ، براساس بررسی های انجام شده یک CRRS به نام CoRReCT پیدا کردیم که هدف آن توصیه بازبینی کنندگان کد بر اساس پروژه و فناوری مرتبط بود. TIE (Text mIning and a filE) همانطور که از نامش مشخص است، مرورگرهای کد را با کمک متن داده­کاوی و مکان فایل توصیه می­کند.

**جواب سؤال 4:**

بر اساس نظرسنجی اعضای پروژه نرم‌افزاری که انجام شد، تعدادی ویژگی پیدا کردیم که برای یک سیستم توصیه برای مرورگران کد مهم تلقی می‌شد که شامل موارد زیر است:

1. بحث کد با نسخه‌های قدیمی و جدید که برای نشان‌دادن تغییر کد مشخص می‌شوند.

2. ادغام با یک نرم افزار ردیابی مسئله­مان Trello ، JIRA و غیره.

3. بررسی کد را از قبل مرتکب شوید.

4. پیشنهادات بهبود کد توسط مرورگر کد، فراتر از اشاره به خطاهای کد.

5. ادغام با ویرایشگر کد منبع، مانند Visual Studio یا Atom.

6. ادغام با یک بستر ارتباطی تجاری، مانند Slack یا MS Teams.

7. اولویت‌بندی تغییرات بر اساس میزان اهمیت و تأثیر آن بر عملکرد نرم‌افزار.

8. ارائه خط لوله‌ای که نشان می‌دهد پروژه در کدام مرحله توسعه شامل ساخت، آزمایش، بررسی کد و استقرار است.

9. وجود داشبورد برای همه اعضای پروژه که داده‌های آماری کلیه اقدامات انجام شده را نشان می‌دهد، مانند تعداد تعهدات، تعداد بررسی کد انجام شده و تعداد خطاها/هشدارهای کد در پروژه فعلی.

10. بحث کد جدید و قدیمی در صورت تغییر در کد.

11. گزینه‌ای برای انتخاب یک شاخه یا پرونده خاص در یک پروژه برای حفظ گردشکار سیستماتیک و روش بازبینی کد سازمان‌یافته.

12. تشخیص کد را با استفاده از یک طرح کدگذاری رنگی با نام توسعه‌دهنده منتقل کرد.

**جواب سؤال 5:**

بر اساس نتایج نظرسنجی، برخی از ویژگی‌هایی که می‌توان آنها را بهبود بخشید یا در سیستم‌های توصیه‌ای مرورگر کد یا هنگام جستجوی یک بازبین کد مربوطه وجود ندارد، عبارت‌اند از:

1. وقتی صحبت از انتخاب مرورگر کد می‌شود، شرکت‌کنندگان معتقد بودند که تخصص بازبینی کنندگان در زبان برنامه‌نویسی پروژه، زمینه تخصص، سال‌ها تجربه کاری، تخصص کیفیت کد و درک معماری پروژه عوامل مهمی هستند.

2. برخی از شرکت‌کنندگان معتقد بودند که چندین سال سابقه کار و همچنین زمینه تخصص، هر دو مهم هستند. استدلال این بود که این عوامل هنگام یافتن رویکرد بهینه شده برای یک مشکل و ارائه پیشنهادات برای LLD)طراحی سطح پایین) می‌توانند مفید واقع شوند. همچنین، این عوامل در نوشتن یک استاندارد استاندارد از تجربه و تخصص مفید است.

## 3-5 پیشنهادها

پیشنهاد برای بهبود سیستم توصیه‌کننده مرور کد بر اساس یافته‌های حاصل از تحقیقات انجام شده در پایان‌نامه مورد بررسی یک سیستم توصیه بازبینی­کننده کد را پیشنهاد می‌کنید که دارای تمام ویژگی‌های لازم باشد که در همه سیستم‌ها وجود ندارد یا ویژگی‌هایی که در سیستم‌های موجود مفقود شده است. سیستم پیشنهادی دارای ویژگی‌های زیر است:

1. شفاف‌تر از نظر شرایطی که همه جزئیات مربوط به مرورگر کد در داشبورد قابل‌مشاهده باشد. این جزئیات شامل تعداد پروژه‌هایی است که آنها روی آن کارکرده‌اند (یعنی تجربه کاری آنها)، زمینه کاربردی خاصی که در آن تخصص دارند، تعداد بررسی کد انجام شده توسط آنها و حجم کار آنها (یعنی تعداد بررسی‌هایی که داور در حال حاضر دارد) مرور برای اطمینان از بارگیری بیش از حد داور با بررسی کد جدید اعتقاد بر این است که ارائه این جزئیات در نتیجه روند بررسی کد را تسریع می‌کند.

2. ترکیبی وسیع تر از داده ها برای آموزش توصیه­گر این مجموعه داده شامل نظرات مرور قبلی و پیام­های متعهد و تجربه متقابل پروژه و فناوری خواهد بود. این داده ها به شما کمک می­کند تا بدانید آیا کدی که باید بازبینی شود با تجربه پروژه ای که یک بازبین دارد مطابقت دارد یا خیر. به طور مشابه، تاریخچه گذشته نظرات و تعهدات بازبینی بر اساس تعداد مرتکبین و بررسی­هایی که قبلاً توسط آنها انجام شده بود و اینکه چگونه مرورهای کد قبلی برای توسعه­دهندگان مفید بوده است، در انتخاب یک مرورگر مربوطه کمک خواهد کرد. این امر به شما اطمینان می دهد که نظرات مرور آینده آنها برای بررسی کد مفید خواهد بود.

3. بررسی کد قبل از ادغام کد و درگیری احتمالی کد اتفاق می‌افتد؛ بنابراین سیستم پیشنهادی قبل از وقوع درگیری ادغام، مرورگران کد را توصیه می‌کند. بااین‌حال، سیستم همچنین اجازه می‌دهد تا انتخاب کد برای بررسی پس از ادغام درگیری‌ها انجام شود تا در صورت لزوم از تأخیر جلوگیری شود و درعین‌حال از ایجاد یک محصول نرم‌افزاری باکیفیت خوب اطمینان حاصل شود.

## 4-5 کارهای آینده

جهت‌های احتمالی آینده بر اساس این کار عبارت‌اند از:

* **مروری گسترده تر بر ادبیات سیستماتیک:** می­توان مطالعه ادبیات سیستماتیک گسترده­ای را انجام داد که نه تنها سیستم­های توصیه مرورگر کد، بلکه شیوه­ها و رویه­های بازبینی کد را نیز بررسی می­کند. این می­تواند به ارائه تصویر بهتر در مورد نیازهای اعضای پروژه نرم­افزار در مورد بررسی کد در ارتباط با استفاده از CRRS کمک کند.
* **ایجاد سیستم توصیه بازبینی کد:** برای کارهای آینده، هدف ما ایجاد سیستمی است که تمام جزئیات مرورگر در داشبورد سیستم قابل‌مشاهده باشد (یعنی تجربه کار، تجربه فناوری، تعداد بررسی کد انجام شده و غیره). همچنین، این سیستم داده‌های بیشتری برای آموزش سیستم توصیه­گر دارد که شامل نظرات مرور قبلی و پیام‌های متعهد، پروژه‌های مرتبط و تجربه فناوری است. بر اساس بازخورد به‌دست‌آمده از نظرسنجی، بررسی‌ها قبل از وقوع درگیری‌های ادغام انجام می‌شود.

**5-5 مشکلات موجود در ساختار پایان‌نامه موجود**

* مشکل که مربوط به نگار ارجاع بهتر است مطابق راهنمای نگارش پایان‌نامه معاونت آموزشی و تحصیلات تکمیلی دانشگاه پیام‌نور باشد. شیوه ارجاع در این پایان‌نامه به‌صورت لینک به منابع بود و شخصاً کار مشکلی در درک مفاهیم آن داشتم.
* همچنین در پایان‌نامه مورد بررسی، قسمتی تحت عنوان کلمات اختصاری و معرفی آنها وجود نداشت.
* همچنین هیچ توضیحی به‌عنوان پاورقی وجود نداشت.

**6-5 جمع بندی و پیشنهادات**

در کل و به‌عنوان نتیجه تحقیق، تعدادی از سیستم های توصیه بازبینی کد (CRRS)موجود در ادبیات، روش­های مختلف طبقه­بندی این سیستم­ها، ویژگی­های مهم یک سیستم توصیه برای مرورگران کد و چگونگی سیستم­های موجود را شناسایی کردیم. بهبود یافته یا کدام ویژگی­ها درCRRS های موجود وجود ندارد. یک مطالعه مروری سیستماتیک برای شناسایی سیستم­های پیشنهادی مرورگر کد و درک جزئیات مربوط به این سیستم­ها، که شامل ویژگی­ها و عوامل مهم برای CRRS است، انجام شد. سپس ما یک نظرسنجی برای درک نیازهای اعضای پروژه نرم­افزار در مورد سیستم­های توصیه بازبینی کد انجام دادیم که مشخص می­کند آنها درCRRS چه ویژگی­هایی مهم هستند و چه چیزی را می­توان درCRRS های موجود بهبود بخشید.

در پایان‌نامه مورد برسی با انجام مرور ادبیات سیستماتیک، پاسخ به سه سؤال اول تحقیق و با استفاده از نظرسنجی، پاسخ دو سؤال آخر را پیدا کردم.

با انجام یک مرور ادبیات سیستماتیک (SLR) برخی از راه حل­های موجود برای سیستم‌های توصیه برای مرورگران کد را شناسایی کردیم، عواملی که هنگام ایجاد CRRS و دسته بندی CRRSهای موجود باید در نظر گرفته شوند. از سوی دیگر، با انجام نظرسنجی، ویژگی‌هایی را که برای یک سیستم توصیه برای مرورگران کد مهم هستند و چه پیشرفت‌هایی می‌توان در CRRSهای موجود انجام داد، دریافتیم.

# **مراجع**

* D. Gusfield, Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology. 1997.
* D. Cubranic, G. C. Murphy, J. Singer, and K. S. Booth, “Hipikat: A project memory for software development,” IEEE Trans. Software Eng., vol. 31, no. 6, pp. 446–465, 2005. DOI: 10.1109/TSE.2005.71.
* VMware, Reviewboard, 2006. [Online]. Available: https://www.reviewboard.org/.
* B. Kitchenham and S. Charters, “Guidelines for performing systematic literature reviews in software engineering,” Journal of Software Engineering and Applications, 2007.
* H. H. Kagdi, M. Hammad, and J. I. Maletic, “Who can help me with this source code change?,” pp. 157–166, 2008. DOI: 10.1109/ICSM.2008.4658064. [Online]. Available: https://doi.org/10.1109/ICSM.2008.4658064.
* E. W. T. Ngai, L. Xiu, and D. C. K. Chau, “Application of data mining techniques in customer relationship management: A literature review and classification,” Expert Syst. Appl., vol. 36, no. 2, pp. 2592–2602, 2009. DOI: 10.1016/j.eswa.2008.02.021.

[Online]. Available: https://doi.org/10.1016/j.eswa.2008.02.021

* T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, “A systematic literature review on fault prediction performance in software engineering,” IEEE Trans. Software Eng., vol. 38, no. 6, pp. 1276–1304, 2012. DOI: 10.1109/TSE.2011.103. [Online]. Available: https://doi.org/10.1109/TSE.2011.103.
* D. H. Park, H. K. Kim, I. Y. Choi, and J. K. Kim, “A literature review and classification of recommender systems research,” Expert Syst. Appl., vol. 39, no. 11, pp. 10059– 10072, 2012.
* A. Bacchelli and C. Bird, “Expectations, outcomes, and challenges of modern code review,” D. Notkin, B. H. C. Cheng, and K. Pohl, Eds., pp. 712–721, 2013.
* V. Balachandran, “Reducing human effort and improving quality in peer code reviews using automatic static analysis and reviewer recommendation,” D. Notkin, B. H. C. Cheng, and K. Pohl, Eds., pp. 931–940, 2013.
* Bosu, M. Greiler, and C. Bird, “Characteristics of useful code reviews: An empirical study at microsoft,” M. D. Penta, M. Pinzger, and R. Robbes, Eds., pp. 146–156, 2015. DOI: 10.1109/MSR.2015.21.
* P. Thongtanunam, C. Tantithamthavorn, R. G. Kula, N. Yoshida, H. Iida, and K. Matsumoto, “Who should review my code? A file location-based code-reviewer recommendation approach for modern code review,” Y. Gueh´ eneuc, B. Adams, and´ A. Serebrenik, Eds., pp. 141–150, 2015.
* X. Xia, D. Lo, X. Wang, and X. Yang, “Who should review this change?: Putting text and file location analyses together for more accurate recommendations,” R. Koschke, J. Krinke, and M. P. Robillard, Eds., pp. 261–270, 2015.
* M. Gasparic and A. Janes, “What recommendation systems for software engineering recommend: A systematic literature review,” *J. Syst. Softw.*, vol. 113, pp. 101–113, 2016.
* S. Lee and S. Kang, “What situational information would help developers when using a graphical code recommender?” *J. Syst. Softw.*, vol. 117, pp. 199–217, 2016. DOI: [10.1016/j.jss.2016.02.050.](https://doi.org/10.1016/j.jss.2016.02.050)
* M. M. Rahman, C. K. Roy, and J. A. Collins, “Correct: Code reviewer recommendation in github based on cross-project and technology experience,” L. K. Dillon, W. Visser, and L. Williams, Eds., pp. 222–231, 2016.
* M. B. Zanjani, H. H. Kagdi, and C. Bird, “Automatically recommending peer reviewers in modern code review,” *IEEE Trans. Software Eng.*, vol. 42, no. 6, pp. 530–543, 2016.
* K. Haruna, M. A. Ismail, S. Suhendroyono, D. Damiasih, A. C. Pierewan, H. Chiroma, and T. Herawan, “Context-aware recommender system: A review of recent developmental process and future research direction,” 2017.
* E. Schon, J. Thomaschewski, and M. J. Escalona, “Agile requirements engineering:¨

A systematic literature review,” *Comput. Stand. Interfaces*, vol. 49, pp. 79–91, 2017.

* M. Fejzer, P. Przymus, and K. Stencel, “Profile based recommendation of code reviewers,” *J. Intell. Inf. Syst.*, vol. 50, no. 3, pp. 597–619, 2018[.](https://doi.org/10.1007/s10844-017-0484-1)
* V. Kovalenko, N. Tintarev, E. Pasynkov, C. Bird, and A. Bacchelli, “Does reviewer recommendation help developers?,” 2018.
* C. Sadowski, E. Soderberg, L. Church, M. Sipko, and A. Bacchelli, “Modern code¨ review: A case study at google,” F. Paulisch and J. Bosch, Eds., pp. 181–190,
* Gerrit, *Gerrit*, 2021. [Online]. Available: [https://www.gerritcodereview.com/.](https://www.gerritcodereview.com/)
* phacility, *Phabricator*, 2021.

## Abstract

Context: Selecting a code reviewer is an important aspect of software development and depends on several factors.

Objectives: The aim is to understand existing solutions for code reviewer recommendation systems (CRRSs), factors to be considered when building them and various dimensions based on which they can be categorised. Our goal is to understand important features of CRRSs and what can be improved in existing CRRSs.

Methods: A literature review study was conducted to understand the existing CRRSs. A survey of software development project members was conducted to understand the important and missing features in CRRSs.

Results: We categorized the selected papers into two categories: based on the data type used to make recommendations and the kind of project used for evaluation. The survey helped us understand the features missing in CRRSs and observe some trends and patterns.

**Keywords:** code reviewer, Pair programming, systematic examination

****

**Payam Noor University**

**Department of Computer Engineering and Information Technology**

**Seminar Report (M.Sc)**

Title:

**INVESTIGATING PAST AND PRESENT CODE REVIEWER**

**RECOMMENDATION SYSTEMS**

**Supervisor:**

**Dr. Ali Razavi**

**By:**

**Zahra Kolivand**

September 2021

1. [D. Cubranic, G. C. Murphy, J. Singer, and K. S. Booth, “Hipikat] [↑](#footnote-ref-1)