



دانشکده فنی و مهندسی
گروه مهندسی کامپیوتر و فناوری اطلاعات

گزارش سمینار کارشناسی ارشد رشته مهندسی کامپیوتر نرم افزار (M.Sc)

عنوان سمینار:

سیستم‌های توصیه شده بازبینی کننده گذشته و کد فعلی

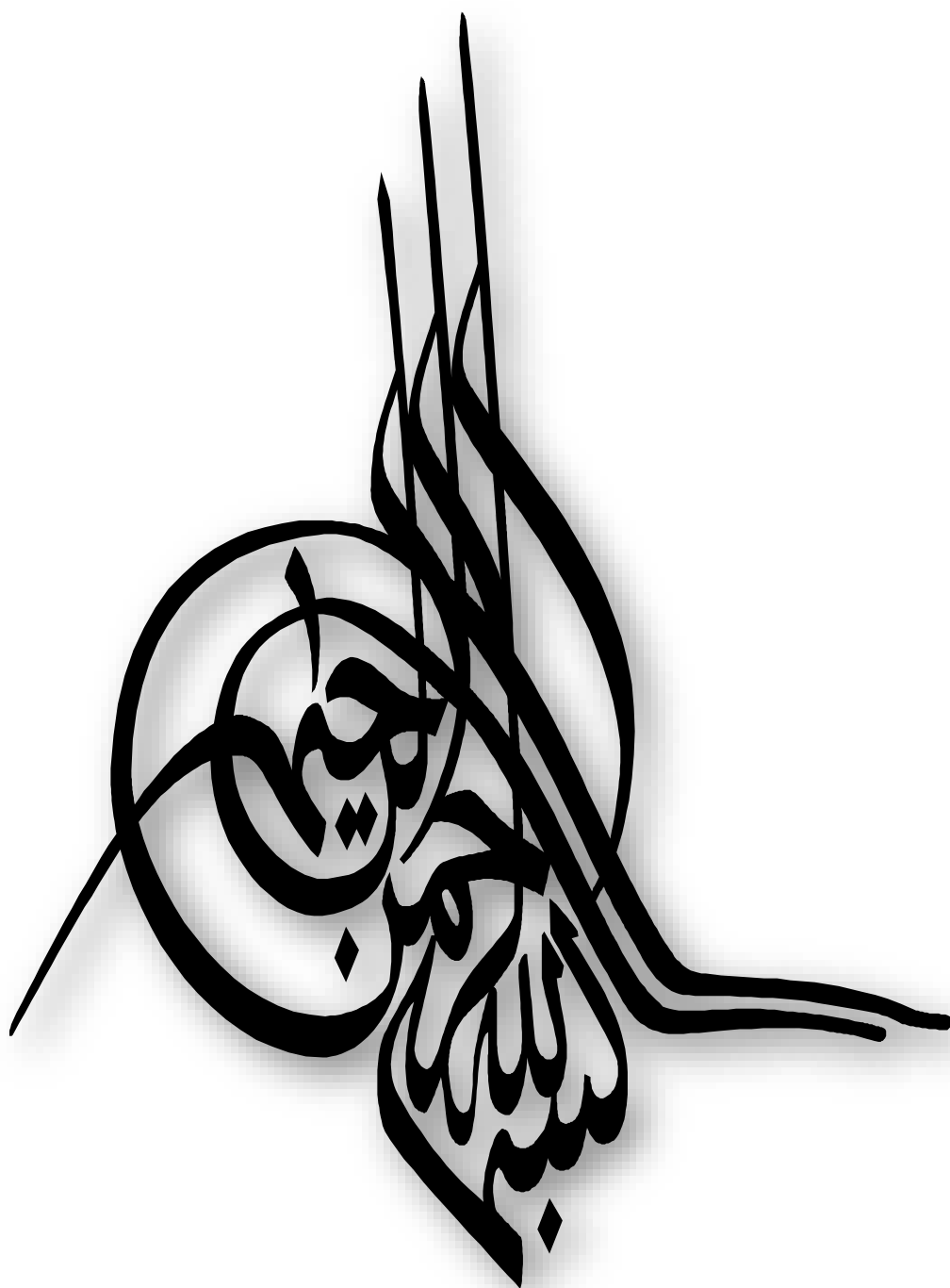
استاد راهنما:

دکتر سید علی رضوی

نگارنده:

زهرا کولیوند

شهریور 1400



فهرست مطالب

چکیده:	1
1. مقدمه	2
1-1 تعریف مسئله و بیان سؤال‌های اصلی تحقیق:	3
2-1 هدف:	4
3-1 ضرورت تحقیق:	4
4-1 روش‌ها و مراحل انجام تحقیق:	5
5-1 سازمان پایان‌نامه مورد بررسی:	5
1-6 ساختار گزارش تحقیق:	6
2. ادبیات تحقیق و پیش زمینه	7
1-2 مقدمه:	7
2-2 مروری بر ادبیات سیستم‌ها	7
3-2 بررسی ادبیات در مهندسی نرم افزار	9
4-2 بررسی ادبیات در استخراج داده‌ها	9
5-2 بررسی ادبیات در داده کاوی	10
6-2 خلاصه	11
3. پیشینه تحقیق	12
1-3 مقدمه:	12
2-3 پیشینه:	12
3-3 CRRS مبتنی بر مشخصات	14

15	4-3 روش پژوهشی
16	5-3 نتایج
17	6-3 سیستم های پیدا شده
17	1-6-3 بازنگر (REVFINDER)
19	2-6-3 cHRev
20	3-6-3 CoRReCT
20	4-6-3 TIE
22	5-6-3 CodeFlow
25	7-3 خلاصه
26	4. تجزیه و تحلیل موضوع انتخابی
26	1-4 مقدمه
26	2-4 روش پژوهش تحقیق:
26	1-2-4 بررسی غربالگری
34	3-4 برخی از روندها و الگوهای مشاهده شده
39	5. جمع بندی و پیشنهادات
39	1-5 مقدمه:
39	2-5 نتایج حاصل از تحقیق
39	1-2-5 پاسخ به سوالات تحقیق
42	3-5 پیشنهادات
43	4-5 کارهای آینده
45	مراجع
47	Abstract

فهرست اشکال

- شکل 3.1: مثال محاسبه الگوریتم رتبه بندی Code-Reviewers 18
- شکل 3.2: معماری TIE 21
- شکل 3.3: روش تحقیق سه مرحله‌ای 23
- شکل 3.4: مدل درخت تصمیم‌گیری برای طبقه‌بندی نظرات مفید 24
- شکل 4-1: معیارهای انتخاب مرورگر کد 31
- شکل 4-2: انواع ویژگی‌های CRRS 32
- شکل 4-3: اولویت زمانی که توصیه‌های بازبینی کد وجود دارد 33
- شکل 4-4: نوع مرور کد 33
- شکل 4-5: نوع CRRS و نقش شغلی 35
- شکل 4-6: ویژگی‌های CRRS و نقش شغلی 37

فهرست جداول

جدول 3.1: رشته‌های جستجو.....	16
جدول 3.2: مقاله‌های استخراج شده و داده‌های مورد استفاده.....	17
جدول 4.1: نقش‌های شغلی شرکت کنندگان.....	28
جدول 4.2: موقعیت جغرافیایی شرکت کنندگان.....	28
جدول 4.3: اندازه تیم پروژه.....	29
جدول 4.4: توزیع تیم.....	29
جدول 4.5: آشنایی با CRRS در بین شرکت کنندگان.....	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 توصیه‌بازبینی 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 CRRS‌های 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

1. مقدمه

مرور کد یک بررسی سیستماتیک از کد منبع رایانه است و اغلب به عنوان یک بررسی همکار انجام می شود. هدف بازبینی کد شناسایی و اصلاح اشتباهات در کد منبع و همچنین بهبود کیفیت کد و مهارت توسعه دهندگان نرم افزار 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 می شود. همچنین مالکیت کد مشترک را تشویق می کند. چهار نوع بررسی کد وجود دارد:

1. **برنامه نویسی جفت:** در این نوع بازبینی کد، دو توسعه دهنده به طور هم زمان کد منبع را تولید می کنند و به طور هم زمان مرور می کنند.

2. **مرور کد به کمک ابزار:** برای این نوع مرور کد، نویسندگان و توسعه دهندگان از ابزارهای بررسی کد همتا استفاده می کنند.

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 شرکت کنندگان 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 سازمان های مختلف توسعه یافته است.

زمینه های متعددی وجود دارد که در آنها مشارکت سیستم های توصیه شده برای اعضای پروژه توسعه نرم افزار مفید بوده است. برای کمک به وظیفه بازبینی کد، تحقیقات قابل توجهی در مورد سیستم های توصیه ای انجام شده است که هدف آنها ارائه توصیه های 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 بازیکن کد در یک زمینه خاص بر اساس درخواست‌های کشش آنها.

نمونه‌هایی از سیستم‌های توصیه‌ای کلی در مهندسی نرم‌افزار

1. سیستم‌های توصیه‌گر کد گرافیکی

2. سیستم هیپی کات

Lee و kang مطالعه‌ای بر روی "سیستم‌های توصیه‌گر 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. تجزیه و تحلیل کد¹

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, Singer, 2005].

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 به اشتراک‌گذاری اطلاعات مربوط به یک پروژه از هر جهت 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 به عنوان 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.

¹ [D. Cubranic, G. C. Murphy, J. Singer, and K. S. Booth, "Hipikat]

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؟ به عبارت دیگر، چه ویژگی‌هایی در پیاده‌سازی‌های موجود برای سیستم‌های توصیه‌بازبینی 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 توصیه‌بازبینی کننده کد موجود کمک می‌کند، درحالی‌که این نظرسنجی به یافتن تغییراتی که مهندسان نرم‌افزار فکر می‌کنند نیاز دارند یا آنچه در سیستم‌های توصیه‌بازبینی 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 توصیه‌کد 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:

- تجزیه و تحلیل راه حل های ارائه شده توسط "مرورگر کد موجود سیستم های توصیه "
- برای درک راه حل ها/ویژگی های موجود در "سیستم های توصیه مرورگر کد" موجود.
- تجزیه و تحلیل وسایل مختلف انشعاب سیستم های پیشنهادی مرورگر کد بر اساس نحوه پیاده سازی آنها.

1-4 روش ها و مراحل انجام تحقیق:

1. تعدادی از ویژگی های موجود در سیستم های پیشنهادی بازبینی کد (CRRS) را شناسایی کرده و آن ویژگی ها را بر اساس مفید بودن آنها.
2. CRRS های موجود را بر اساس ابعاد مختلف طبقه بندی کردیم.
3. ویژگی هایی را که می توان هنگام انتخاب مرورگر کد مهم تلقی کرد، شناسایی کردیم.
4. بهبودهای احتمالی CRRS های موجود را برای تسهیل یافتن مرورگران کد مناسب شناسایی کردیم.

1-5 سازمان پایان نامه مورد بررسی:

فصل های این پایان نامه به صورت ذیل مرتب شده است:

- **فصل 2:** کارهای مرتبط و این فصل خلاصه ای از مرور ادبیات قبلی انجام شده در مهندسی نرم افزار، داده کاوی و سیستم های توصیه گر را ارائه می دهد .
- **فصل 3:** نتایج مطالعه مرور ادبیات و شیوه ما و ابزارهای بازبینی کد ارائه شده است.
- **فصل 4:** بررسی نتایج به منظور انجام نظرسنجی از مهندسان نرم افزار 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 نتیجه گیری و کارهای آینده است.

6-1 ساختار گزارش تحقیق:

فصل 1: به تعریف و مقدمه و دلایل نیاز به طرح ارائه شده پرداخته می شود.

فصل 2: به پیش زمینه و کارهای وابسته پرداخته می شود.

فصل 3: مروری است بر کارهای انجام شده طرح پیشنهادی پایان نامه.

فصل 4: به کاربردها و مزایا و معایب روش های مطرح شده پرداخته می شود.

فصل 5: نیز به جمع بندی و نتیجه گیری پرداخته می شود.

فصل دوم

2. ادبیات تحقیق و پیش زمینه

2-1 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 در گذشته انجام شده و برای راهنمایی روش بررسی ادبیات ما مورد استفاده 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 در داده‌کاوی دو مدل پرکاربرد برای داده‌کاوی در CRM (مشتري مدیریت روابط) را کشف کرد.

2-2 مروری بر ادبیات سیستم‌ها

مرور ادبیات توسط SUHENDROYONO ،Ismail ، HARUNA و همکاران انجام شد در مورد سیستم‌های توصیه‌کننده آگاه به زمینه (CARS) که هدف آنها استخراج اطلاعات مربوطه از حجم زیادی از دانش است. این سیستم‌های توصیه‌گر، ارائه اطلاعات زمینه‌ای و 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. برجسته کردن 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 [M. Haruna, 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, Janes انجام شد، که نتایج عملکردی که RSSE‌های موجود (سیستم توصیه برای مهندسی نرم‌افزار) ارائه می‌دهند، شکاف‌های تحقیقی و همچنین 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 RSSE‌های موجود
 - مزایایی که این RSSE‌ها برای مهندسان نرم‌افزار فراهم می‌کند
 - انواع ورودی موردنیاز این RSSE‌ها و چه تلاش‌هایی مهندس نرم‌افزار نیاز به استفاده از این RSSE‌ها دارد
 - برخی از خروجی‌های RSSE‌های موجود شامل فایل‌های کد منبع باینری
 - تغییرات در محیط استقرار
 - الگوهای طراحی و اسناد دیجیتالی است که ممکن است برای مهندس نرم‌افزار جالب باشد
- RSSE‌های موجود عمدتاً از چه چیزهای پشتیبانی می‌کنند:
- از استفاده مجدد، اشکال‌زدایی، پیاده‌سازی، مراحل/فعالیت‌های نگهداری و پشتیبانی از بهبود کیفیت سیستم به نفع مهندسان نرم‌افزار.
- ورودی‌هایی که این RSSE‌های فعلی از آنها استفاده می‌کنند:

شامل فایل های گزارش، ارتباط بین مهندسين نرم افزار، کد منبع، ورودی کاربر (به عنوان مثال، عبارت های جستجو، پرس و جو، تنظیمات، ترجیحات)، مصنوعات آزمایشی و فرایند توسعه نرم افزار است همچنین، تلاش هایی که یک مهندس نرم افزار باید برای استفاده از RSSE های موجود انجام دهد به عنوان تلاش های گسترده، تلاش های کم و بدون تلاش طبقه بندی می شود. [M. Gasparic and A. Janes و همکاران 2016]

2-3 بررسی ادبیات در مهندسی نرم افزار

پیش بینی دقیق خطاهای کد می تواند 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 مقاله ها و مطالعات مربوطه و حذف مطالعات مکرر است. هنگام حذف و شامل مقاله ها، موارد مختلفی در نظر گرفته می شود، مانند مقاله هایی که از منابع مختلف مانند مجلات، کنفرانس ها و پایگاه های داده استخراج شده اند و بر اساس محتوای عنوان و چکیده آنها مرتب شده اند. یافته ها نشان می دهد که اکثر مطالعات اطلاعات زمینه ای و روش شناختی کافی را برای درک کامل یک مدل گزارش نمی دهند. نویسندگان مجموعه ای از معیارها را ارائه می دهند.

2-4 بررسی ادبیات در استخراج داده ها

مجموعه ای از جزئیات اساسی زمینه ای و روش شناختی را که مطالعات پیش بینی خطا باید گزارش کنند، شناسایی می کند.

روش توسعه نرم افزار چابک یک متدولوژی توسعه نرم افزاری متداول است که توسط بسیاری از پروژه های توسعه نرم افزار استفاده می شود. هدف این روش اطمینان از تحویل خوب محصول مطابق نیاز کاربر و تجربه کاربری مناسب (UX) است. به منظور ارائه یک محصول باکیفیت، مشارکت ذی نفعان و کاربران، همراه با حلقه های بازخورد از هر دو طرف ضروری است. به منظور ارائه یک محصول باکیفیت، مشارکت ذی نفعان و کاربران، همراه با حلقه های 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 پیشرفت های احتمالی آینده برای

بررسی جنبه‌هایی که در وضعیت فعلی وجود ندارد، انجام شد [E. Schon, J. Thomaschewski, and M. J Escalona 2017]. 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 تجزیه و تحلیل داده‌ها انجام شد.
3. **گزارش:** با هدف استخراج و بحث درباره نتایج به دست آمده از مرحله قبل و سپس نوشتن، ارزیابی و قالب بندی گزارش نهایی برای مطالعه انجام شد.

2-5 بررسی ادبیات در داده‌کاوی

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

تکنیک‌های داده‌کاوی در مدیریت ارتباط با مشتری (CRM) اعمال می‌شود و Chau و XIU، Ngai به کمک مرور ادبیاتی که انجام داده‌اند، بینش کاملی در این زمینه ارائه می‌دهد. نویسندگان حدود 87 مقاله تحقیقاتی مرتبط را برای این منظور جمع‌آوری 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 داده‌کاوی 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.

2-6 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

3-1 مقدمه:

در گذشته تعدادی سیستم/ابزار توصیه بازبینی کد وجود داشته است. به عنوان مثال، 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 می کند. به هر نویسنده ای که روی تغییر کد در کد منبع کار کرده است امتیاز تعلق می گیرد اما نویسندگان با تغییرات اخیر نسبت به نویسندگان تغییرات قدیمی امتیاز بیشتری کسب می کنند. در پایان، از جمع بندی هر نویسنده جداگانه برای تصمیم گیری نویسندگان برتر k استفاده می شود و سپس به آنها توصیه می شود 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 روش شناسی خود را قبل از ارائه نتایج مطالعه خود تعریف می کنیم.

3-2 پیشینه:

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

مطالعه اکتشافی آنها بر 3 جنبه بررسی کد متمرکز بود:

1. انگیزه های بررسی کد
2. شیوه های فعلی
3. تفسیر توسعه دهندگان بررسی کد.

به منظور ایجاد ساختار بیشتر در بازبینی کد، چندین ابزار در نرم افزار 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، Gerrit، استفاده شده توسط Google Chromium، ReviewBoard توسعه یافته توسط VMware nmngbbb و Phabricator مورد استفاده فیس بوک است. در زیر مروری کوتاه بر هر یک از این سیستم های توصیه مرورگر کد است.

1. **CodeFlow**: CodeFlow وضعیت هر شخص (توسعه دهنده یا بازبینی کننده) و موقعیت آنها در این مرحله (یعنی انتظار، بررسی، امضا) را ردیابی کرد. CodeFlow نویسندگان را از ارسال هیچ گونه تغییری بدون تأیید منع نمی کند و همچنین از گپ ها در موضوعات نظر پشتیبانی می کند.

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

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 تجزیه و تحلیل می کنند، و در نتیجه کیفیت مرور کد بهبود یافته است.

Phabricator: Phabricator، که توسط فیس بوک استفاده می شود، به یک بازبین اجازه می دهد 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 می کند.

به منظور درک روند بررسی کد در Google، نویسندگان بر دو جنبه اصلی تمرکز کردند: فرایند بازبینی که توسعه دهندگان در طول بررسی های خاص تجربه می کنند و اینکه آیا توسعه دهندگان با وجود چالش ها از بررسی های 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 می تواند تفاوت برجسته تغییرات پیشنهادی را ببیند و همچنین یک بحث موضوعی در مورد خطوط کد با توسعه دهندگان یا سایر بازبینان آغاز کند. CRITIQUE همچنین نمایی از همه عملکردهای ورود به سیستم یک توسعه دهنده، و همچنین تعامل آن با ابزار را شامل بازکردن ابزار، ایجاد تغییرات، مشاهده تفاوت و تأیید تغییرات ارائه می دهد. به منظور درک انگیزه توسعه دهندگان برای بررسی کد در Google با استفاده از CRITIQUE و درک توسعه دهندگان در مورد همین، نویسندگان از مصاحبه به عنوان ابزاری برای جمع آوری داده ها استفاده کردند.

بر اساس داده‌های جمع‌آوری شده از مصاحبه‌های انجام شده، یافته‌های زیر به دست آمده است.

یافتن 1: بررسی کدهای انجام شده در Google نه تنها باهدف تصحیح خطاها یا مشکلات است، بلکه همچنین برای اطمینان از خوانایی و قابلیت نگهداری کد که به عنوان 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.

[2018, Sadowski, Soderberg]

3-3 CRRS مبتنی بر مشخصات

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

در مدل توصیه بازیابی 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 بازیابی‌ها مقایسه می‌شود شباهت بین نمایندگی چند مجموعه‌ای از تعهدات و نمایه‌های بازیابی‌کنندگان محاسبه می‌شود و n بازنگری برتر انتخاب می‌شوند. در اینجا، به روز رسانی مشخصات بازیابی‌کننده یکی از مهم‌ترین و مکررترین عملیات است. هرگاه نظر جدیدی توسط یک بازیاب 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 به ازای 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 تک نفره 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 بررسی ها طی سه روز برای پروژه های LibreOffice و OpenStack، حداکثر دو روز برای Qt و حداکثر شش روز برای پروژه های Android تکمیل شد [2018, M. Fejzer, P. Przymus].

3-4 روش پژوهشی

مطالعه مروری بر ادبیات سیستماتیک روشی است که در آن ادبیات موجود مربوط به تحقیق تعیین می شود، سپس ارزیابی می شود و در نهایت درک می شود. برای تحقیقات خود، ما رویکردی را که Kitchenham و Charters اتخاذ کرده است دنبال می کنیم 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، سیستم ها، ابزارها و توصیه کننده. این کلمات کلیدی بر اساس موضوع تحقیق ما مشخص شد.

2. از کلمات کلیدی مشخص شده برای تشکیل رشته های جستجو استفاده کنید. رشته های جستجو برای به دست آوردن مقالات تحقیق از پایگاه های داده آنلاین استفاده می شود. ما از یک رشته جستجو برای کلمات کلیدی احتمالی و مترادف آنها استفاده کردیم. ما دو رشته جستجو ایجاد کردیم. برای یافتن رشته های جستجو به جدول 3.1 مراجعه کنید.

3. مقالات تحقیقاتی به دست آمده سپس بر اساس معیارهای مختلف حذف و ورود فیلتر می شوند. مقالات ابتدا با خواندن عنوان ها و چکیده مقالات تحقیق فیلتر می شوند.

فیلتر کردن نتایج جستجو در سه مرحله اصلی انجام شد:

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

مقالات فیلتر شده به طور کامل خوانده می شوند، ارزیابی و تفسیر می شوند تا اطلاعات مربوطه را به

دست آورند. [B. Kitchenham and S. Charters, 2007]

1	(کد) و (داوران) و (توصیه کننده) یا (توصیه) (سیستم‌ها) یا (ابزارها)
2	(توصیه) و (توصیه کننده) 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 به دست آمده، 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.

3-5 نتایج

پس از خواندن متن کامل مقاله‌های تحقیقاتی فیلتر شده، نه سیستم توصیه کننده مرور کد شناسایی شد. جدول 3.2 این سیستم‌ها و داده‌های مورد استفاده 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 می شود	مرور کد مدرن: مطالعه موردی در گوگل
وضعیت هر منتقد یا نویسنده را دنبال می کند	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	بر اساس تجربه پروژه GitHub توصیه مرورگر کد در correct: 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: مقاله های استخراج شده و داده های مورد استفاده

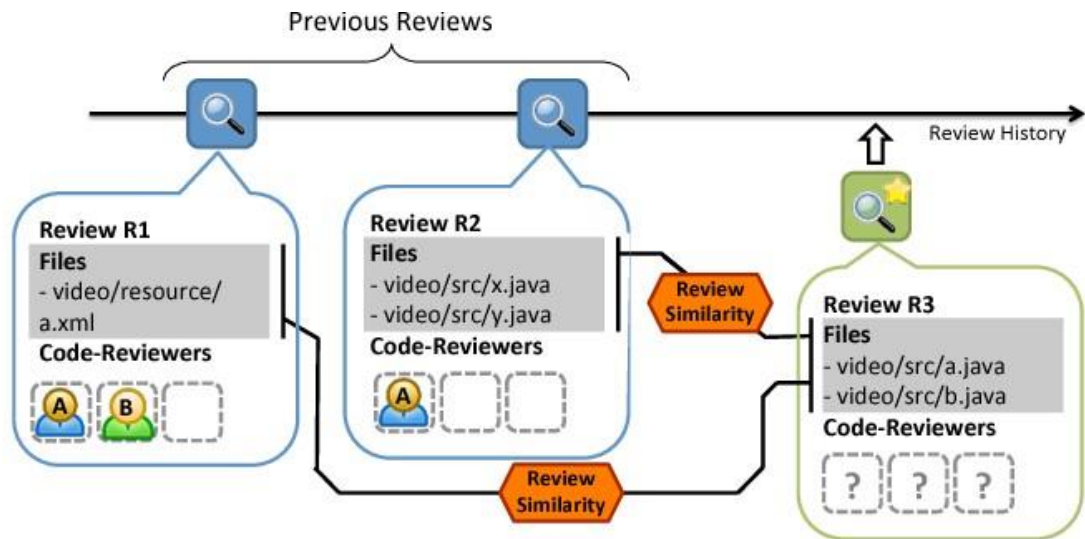
3-6 سیستم های پیدا شده

3-6-1 بازنگر (REVFINDER)

تعدادی CRRS براساس سیستم های 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:



Code-Reviewers Scores

- 1 = $\text{ReviewSimilarity}(R3, R1) + \text{ReviewSimilarity}(R3, R2) = 0.1 + 0.5 = 0.6$
- 2 = $\text{ReviewSimilarity}(R3, R1) = 0.1$

شکل 3.1: مثال محاسبه الگوریتم رتبه‌بندی Code-Reviewers

1. طولانی‌ترین پیشوند مشترک (LCP)
 2. طولانی‌ترین پسوند مشترک (LCS)
 3. طولانی‌ترین زیر رشته مشترک (LCSubstr)
- LCP اجزای مسیر فایل رایج را که در هر دو مسیر فایل از ابتدا تا انتها ظاهر می‌شود، محاسبه می‌کند.
- LCS اجزای مسیر فایل مشترکی را که در هر دو مسیر فایل از انتهای هر دو مسیر فایل ظاهر می‌شوند، محاسبه می‌کند.

LCSubstr اجزای مسیر پرونده رایج را که در هر دو مسیر پرونده به صورت متوالی ظاهر می شوند، اما در هر موقعیتی در مسیرهای فایل ظاهر می شوند، محاسبه می کند.

4. طولانی ترین متعاقب مشترک (LCSubseq)

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

cHRev 2-6-3

تعدادی CRRS بر اساس بررسی های گذشته ساخته شده اند و Bird و Zanjani, Kagdi یک چنین سیستم توصیه ای به نام cHRev ایجاد کرده اند. cHRev به طور خودکار مرورگران کد را براساس مشارکت های قبلی خود در بررسی های قبلی خود توصیه می کند. 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. بررسی کنندگان کد توصیه شده توسط cHRev ممکن است لزوماً در توسعه بخشی از کد منبع که در حال بررسی آن هستند مشارکت نداشته باشند، اما ممکن است بر روی کد منبع کار کرده باشند که به طور غیرمستقیم به منبع، منبع مورد بررسی بستگی دارد.

2. تخصص در طول زمان تغییر می کند و بنابراین تکرار و تکرار باید هنگام جستجوی مناسب ترین 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 رتبه بندی از داوران داوطلب بر اساس فایل های 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 به دست آمده توصیه کنید.

به منظور آزمایش اثربخشی رویکرد آنها، زنجان، کادی روش آنها را با REVfinder، xFinder و RevCom مقایسه کرد. مشخص شد که cHRev از نظر دقت و فراخوانی توصیه های دقیق تری را ارائه می دهد. 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]

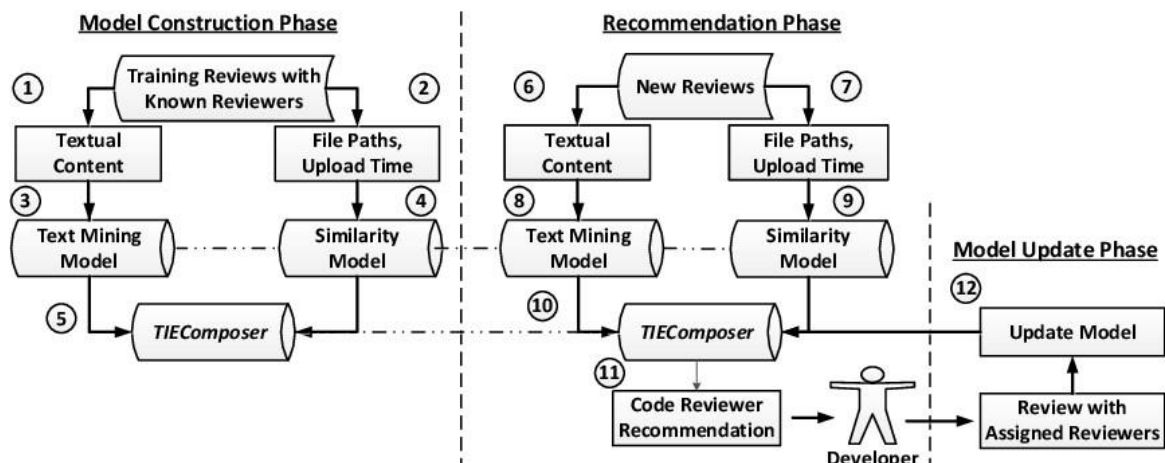
CoRReCT 3-6-3

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 درخواست های pull فعلی در نظر گرفته شوند. باتوجه به 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 تجزیه و تحلیل 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 تغییر یافته 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 تغییر یافته 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، داده‌های 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 به‌روزرسانی 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

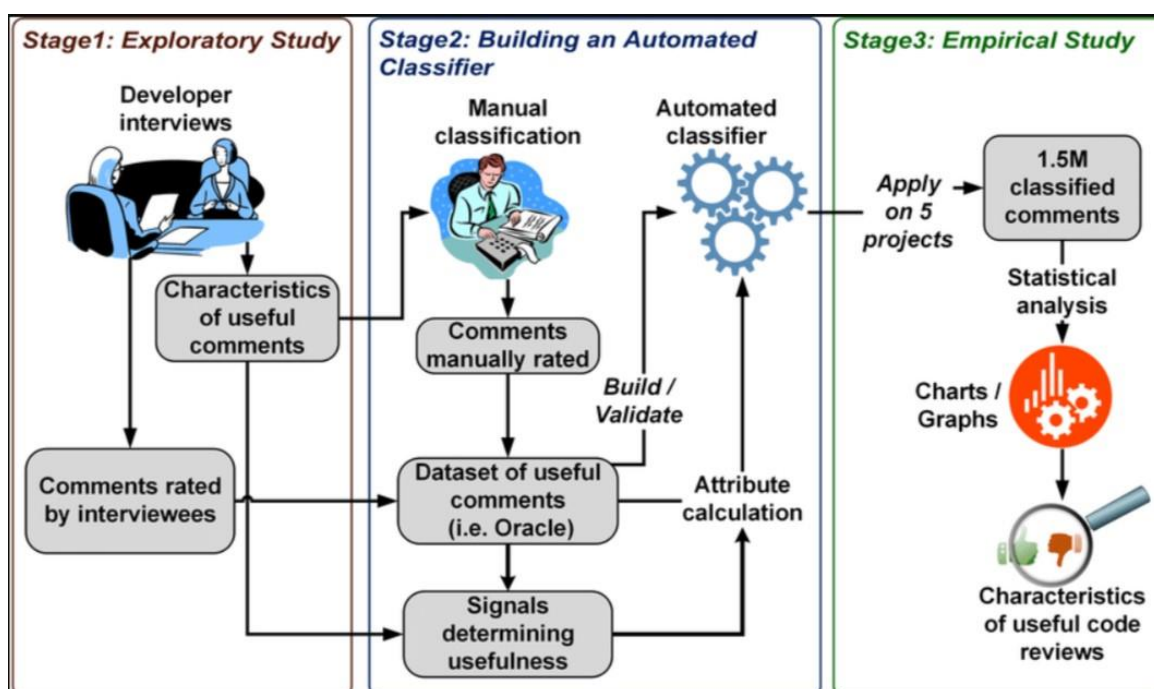
Greiler, Bosu با انجام مصاحبه با توسعه‌دهندگان و همچنین تجزیه و تحلیل نظرات بازبینی پنج پروژه مایکروسافت که با استفاده از CodeFlow CRRS انجام شده بود، در مایکروسافت ویژگی تجزیه و تحلیل کدهای مفید را در مایکروسافت انجام دادند. مطالعه در سه مرحله انجام شد. ابتدا، آنها با انجام مصاحبه‌ای با توسعه‌دهندگان، 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. [M. Greiler, Bose, and C. Bird, 2015]

همچنین تجزیه و تحلیل نظرات بازبینی پنج پروژه مایکروسافت که با استفاده از CodeFlow CRRS انجام شده بود، در مایکروسافت ویژگی تجزیه و تحلیل 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. [2015, Bosu M. Greiler, Bird]

روش تحقیق سه مرحله‌ای:

گردش کار CodeFlow نسبتاً ساده است.

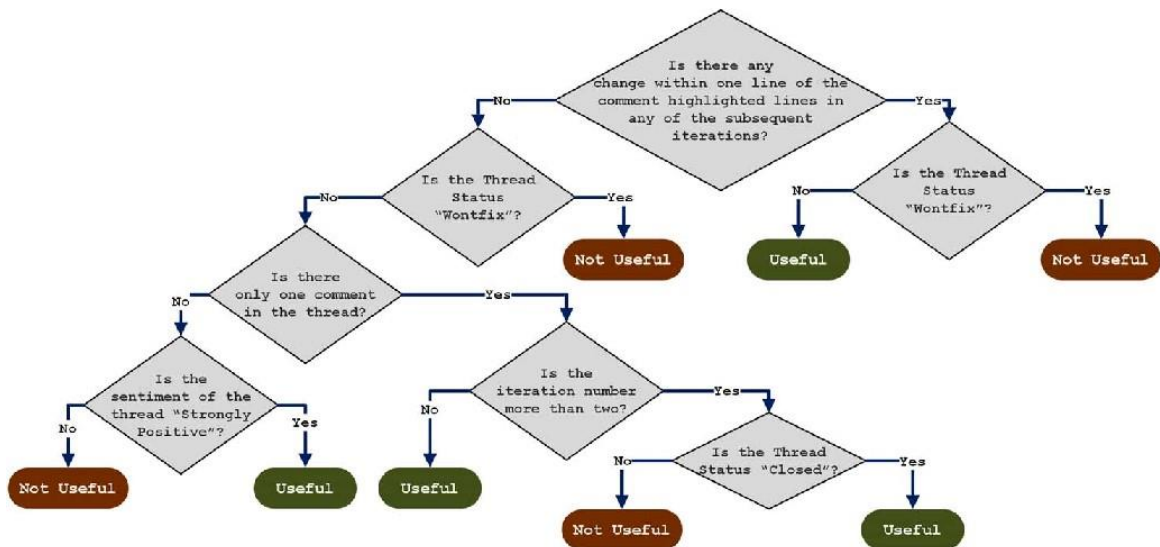
1. ابتدا، به تشخیص نظرات بازبینی کد مفید و غیرمفید بر اساس مصاحبه با توسعه‌دهندگان کمک کرد. نویسنده تغییر نظر را ارسال می‌کند و درخواستی از طریق ایمیل به داور اطلاع داده می‌شود.
2. نویسندگان یک طبقه‌بندی خودکار با استفاده از یافته‌های به‌دست‌آمده از مرحله اول ایجاد کردن به‌منظور ایجاد کلاس بهتر، نویسندگان نظرات بازبینی را به‌صورت دستی به دودسته مفید و غیرمفید طبقه‌بندی 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 با گذشت 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 Visual Studio، Bing، Azure، Office و Exchange هستند، استفاده کردند. بر اساس نتایج، نویسندگان موارد زیر را نتیجه گرفتند:

1. توسعه‌دهندگانی که در گذشته 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 بازیگران 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 افزایش یافته 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، CoRRcCT، CRRS مبتنی بر مشخصات، RevFinder، CodeFlow، TIE و CRITIQUE این سیستم‌ها بر اساس دو بعد تقسیم می‌شوند:

- منبع داده مورد استفاده برای ساخت سیستم
- نوع پروژه مورد استفاده برای ارزیابی سیستم.

فصل چهارم

4. تجزیه و تحلیل موضوع انتخابی

4-1 مقدمه

در این گزارش راه حل های برای سیستم بازبینی کد (CRRS) و عوامل هنگام ساخت آنها در نظر گرفته می شود. بر اساس این می توان آنها را طبقه بندی کرد که به درک بیشتر ما از ویژگی های مهم CRRS و بهبود CRRS موجود است. در این گزارش CRRS و تحقیقات بیشتری در مورد شیوه ها و رویه های بازبینی کد انجام شده است با هدف تجزیه و تحلیل راه حل های ارائه شده توسط "مرورگر کد" که موجود است.

4-2 روش پژوهش تحقیق:

در پایان نامه مورد بررسی به منظور انجام نظرسنجی از مهندسان نرم افزار برای تعیین نیازهای اطلاعاتی برای مرورگران کد، از مراحل زیر برای اطمینان از نتایج صحیح، غیر مغرضانه و دقیق استفاده کرده است.

4-2-1 بررسی غربالگری

نظرسنجی انجام شده برای مهندسان نرم افزار به دو قسمت تقسیم می شود:

1. یک بررسی غربالگری است. ما از نظرسنجی غربالگری استفاده کردیم تا مطمئن شویم از اعضای توسعه محصول نرم افزاری که در سیستم های توصیه بازبینی کد تجربه دارند و بنابراین می توانند اطلاعات دقیق و بی طرفانه ارائه دهند، پاسخ می دهیم.
2. یک نظرسنجی جمعیت شناختی و تجربه CRRS به افرادی که مرحله غربالگری را پشت سر گذاشته اند داده شد، یعنی شرکت کنندگان حداقل دو سال سابقه کار و تجربه استفاده از CRRS را داشتند. سوالات پایان نامه 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 (به‌عنوان مثال خط خالی یا دندان‌دار)، ارگان ization از کد (به‌عنوان مثال چگونه قابلیت به روش تقسیم) و رویکرد راه‌حل بود به‌عنوان در نظر گرفته تا حدودی مفید. همه نظراتی که یا مثبت کاذب بودند (به‌عنوان مثال 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 طبقه‌بندی 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 طبقه‌بندی شده‌اند، در این مرحله دوم در دسته مفید قرار گرفتند. بر اساس مصاحبه و تجزیه و تحلیل 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.

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

جدول 4.1: نقش‌های شغلی شرکت‌کنندگان

2. موقعیت جغرافیایی شرکت‌کنندگان

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

درصد	رشته
9.09	آمریکای جنوبی
9.09	شمالی عامری CA
81.82	آسیا

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.

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

جدول 4.3: اندازه تیم پروژه

4. توزیع تیم

همچنین در پایان نامه مورد بررسی اطلاعاتی در مورد نحوه توزیع تیم ها جمع آوری کردند، به این معنی که آنها در تیم هایی کار می کنند که در محل مشترک یا توزیع شده اند. مشاهده شد که 45.45 of از تیم ها در محل مشترک، 18.18 were توزیع شده و 36.36 were ترکیبی از هر دو مکان مشترک و توزیع شده بود.

درصد	رشته
45.45	در محل مشترک
18.18%	توزیع شده است
36.36	هر دو

جدول 4.4: توزیع تیم

5. آشنایی با CRRS در بین شرکت کنندگان

اکثر شرکت کنندگان 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 76.4 of از شرکت کنندگان با Phabricator آشنا بودند.

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

جدول 4.5: آشنایی با CRRS در بین شرکت کنندگان

6. مفید بودن ویژگی های CRRS

در پایان نامه مورد بررسی تعدادی از ویژگی های CRRS را برای شرکت کنندگان خود لیست کرده اند و از آنها خواسته همه آن ویژگی هایی را که برای آنها مفید بوده است انتخاب کنند. موارد زیر ویژگی هایی 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 مورد توجه قرار گیرد:

- تعدادی سال سابقه کار
- تخصص بازیابی کنندگان کد در زبان برنامه نویسی
- تخصص مرورگر کد در یک حوزه (به عنوان مثال: مهندس نرم افزار هوش مصنوعی و غیره)
- زبان ارتباط بین مرورگر کد و توسعه دهنده نرم افزار
- نقش مرورگر کد
- تعداد پروژه هایی که روی آنها کار شده است
- تعداد بررسی کد انجام شده

#	Field	Extremely likely	Somewhat likely	Neither likely nor unlikely	Somewhat unlikely	Extremely unlikely	Total
1	Number of years of work experience	9.09%	72.73%	9.09%	9.09%	0.00%	11
2	Code reviewers expertise in programming language	54.55%	36.36%	9.09%	0.00%	0.00%	11
3	Code reviewer's expertise in a domain (eg> software engineering, artificial intelligence etc.)	45.45%	45.45%	9.09%	0.00%	0.00%	11
4	Language of communication between the code reviewer and software developer	54.55%	18.18%	27.27%	0.00%	0.00%	11
5	Role of the code reviewer	18.18%	45.45%	36.36%	0.00%	0.00%	11
6	Count of projects worked on	9.09%	63.64%	0.00%	27.27%	0.00%	11
7	Count of code reviews done	27.27%	45.45%	9.09%	18.18%	0.00%	11

Showing rows 1 - 7 of 7

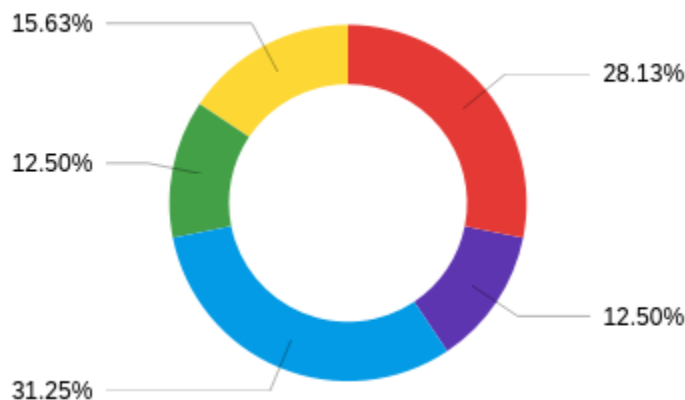
شکل 1-4: معیارهای انتخاب مرورگر کد

8. رابط کاربری CRRS (UI):

به غیر از ویژگی های CRRS، همچنین در مورد ویژگی های مختلف که می تواند برای CRRS مفید واقع شود، با استفاده بیشتر از تجربه کاربری، قابل دسترسی و راحت نشان داد. درصد افرادی که ویژگی های زیر را مفید دانسته اند به شرح زیر با توزیع نمودار پای زیر نشان داده شده است:

- ارائه خط لوله ای که نشان می دهد پروژه در چه مرحله ای است یعنی ساخت، آزمایش، بازنویسی کد، استقرار و غیره 31.25٪: از شرکت کنندگان این ویژگی رابط کاربری را مفید و راحت تر می دانند.
- وجود پنل کاربری برای هر فرد که داده های آماری کلیه اقدامات انجام شده را نشان می دهد (مانند تعداد تعهدات، تعداد بررسی کد انجام شده، تعداد خطاها/هشدارهای کد در پروژه فعلی و غیره) 28 درصد از شرکت کنندگان 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 شرکت کنندگان این ویژگی را مفید می دانند تا روند بازبینی کد را برای تازه کار و توسعه دهنده آسان تر کند.
- انتقال کد با استفاده از طرح رنگی با نام برنامه نویس 12.5 درصد از شرکت کنندگان این ویژگی را مفید دانستند.

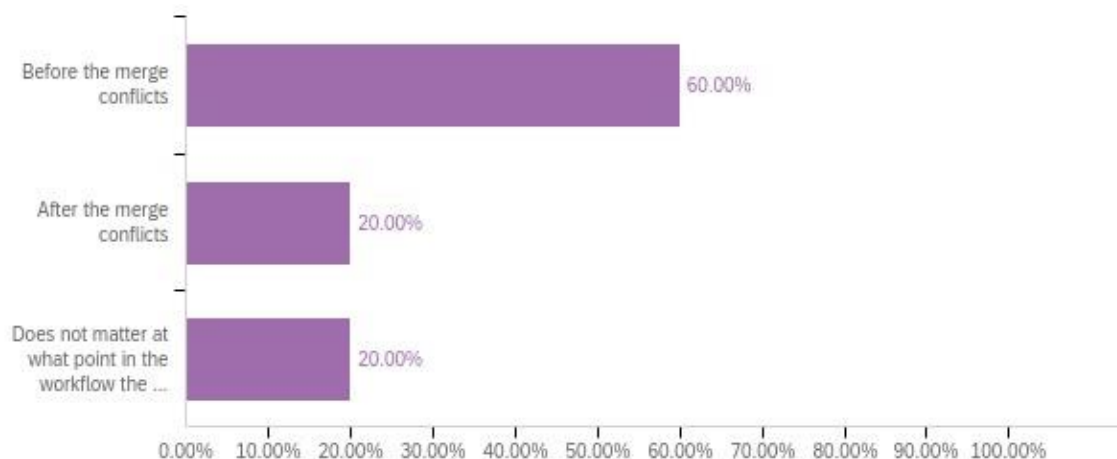
- گزینه ای برای انتخاب یک "شاخه/پرونده" خاص در یک پروژه برای حفظ گردش کار سیستماتیک و روش بازیابی کد سازماندهی شده 12.5 درصد از شرکت کنندگان با سازماندهی بیشتر فرآیند بازیابی کد، این ویژگی را مفید دانستند. نتایج در شکل 1 نشان داده شده است.



شکل 2-4 انواع ویژگی های CRRS

9. ترجیح زمان توصیه برای بررسی کد:

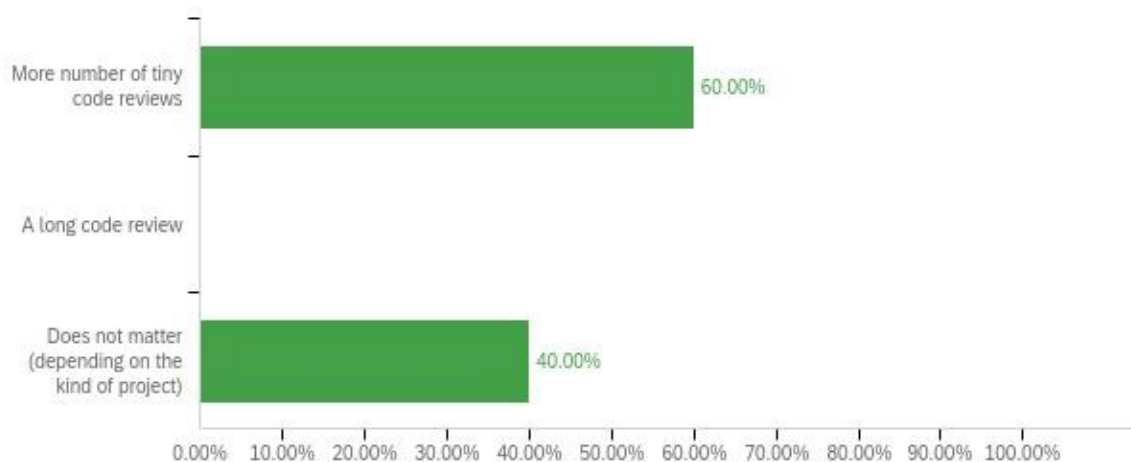
توصیه بازیابی کد می تواند در مراحل مختلف فرایند توسعه نرم افزار انجام شود که شامل قبل از درگیری ادغام، پس از ناسازگاری ادغام یا در هر نقطه از گردش کار می شود. مشاهده شد که 60 of از شرکت کنندگان معتقد بودند که توصیه کد باید قبل از درگیری ادغام انجام شود، 20.00 them از آنها معتقد بودند که باید پس از درگیری ادغام انجام شود درحالی که تا 20.00 them از آنها مهم نیست که در کدام نقطه از گردش کار بررسی کد انجام شده است.



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

10. نوع مرور کد

از شرکت کنندگان خواسته شد که چه نوع بازبینی کد را ترجیح دهند که شامل بسیاری از بررسی های کد کوچک تر یا مرور طولانی کد یا بستگی به نوع پروژه دارد. مشاهده شد که 60.00 of از شرکت کنندگان ترجیح می دهند بسیاری از بررسی های کد کوچک تر را انجام دهند درحالی که بقیه (40.00%) از شرکت کنندگان احساس می کردند که مهم نیست. هیچ یک از شرکت کنندگان ترجیح می دهند 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 به دست آمده، برخی از الگوها و روندها بین نتایج دو یا چند سؤال نظرسنجی یافت شد. برخی از روندهایی که مشاهده کردیم به شرح زیر است:

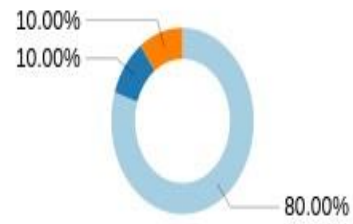
1. نوع CRRS مورد استفاده و نقش شغلی

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

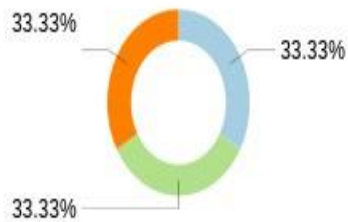
Gerrit Code Review System (Chromium)

NO DATA

GitHub/ GitLab



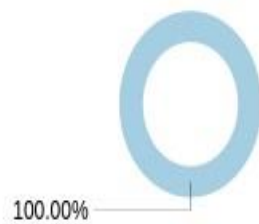
Code Flow Review Tool (Microsoft)



Review Board (VMware)

NO DATA

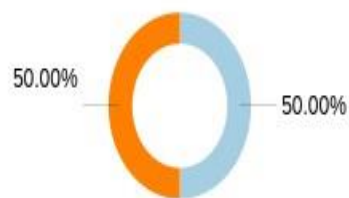
Phabricator



Bitbucket



Other



Developer/ Programmer/ Software Engineer Team Lead

DevOps Engineer/ Infrastructure Developer Architect UI/UX developer Technical Support

Data Analyst/ Data Scientist/ Data Engineer Other

شکل 4-5: نوع CRRS و نقش شغلی

- ما نتایج زیر را برای هر یک از CRRS های زیر به همراه توزیع نمودار پای در شکل 4.4 به دست آوردیم.
- **Github/GitLab:** از شرکت کنندگانی که با این CRRS آشنا بودند، 10٪ از شرکت کنندگان سرپرست تیم بودند، 10٪ دیگر صاحبان محصولات و بقیه (80٪) توسعه دهندگان/برنامه نویسان/مهندسان نرم افزار بودند.
 - **دوم ابزار بازیابی کد جریان (مایکروسافت):** از شرکت کنندگان که با این CRRS آشنا بودند، 33.33٪ توسعه دهندگان/برنامه نویسان، مهندسان نرم افزار، DevOps Engineer/Infrastructure Developer و دیگران بودند
 - **Phabricator:** توسعه دهندگان/برنامه نویسان/مهندسان نرم افزار تنها افرادی بودند که با Phabricator آشنا بودند
 - **BitBucket:** از شرکت کنندگانی که با این CRRS آشنا بودند، 60.00٪ آنها توسعه دهندگان/برنامه نویسان/مهندسان نرم افزار بودند، 20.00٪ آنها رهبران تیم و دیگران بودند.
 - **دیگران:** تعدادی CRRS دیگر در بازار وجود دارد و از شرکت کنندگان خواسته شد در صورت اطلاع از CRRS های دیگری که در نظرسنجی ذکر نشده اند، آن گزینه را انتخاب کنند. زیرا، ما 50.00٪ پاسخ از توسعه دهندگان و دیگران دریافت کردیم.
- از بین شرکت کنندگانی که در نظرسنجی ما شرکت کردند، هیچ یک از آنها با دو مورد از سیستم ها/ابزارهای توصیه بازبینی کد آشنا نبودند (Chromium Gerrit Code System Review (VMware) ReviewBoard).

2. ویژگی های CRRS و نقش شغلی

ما تعدادی ویژگی CRRS را برای شرکت کنندگان مطرح کردیم که آنها معتقد بودند استفاده از CRRS ها راحت تر و آسان تر است. شکل 4.7 نشان می دهد که نقش شغلی کدام ویژگی مفید است.

#	Field	Pre commit code review	Code discussion with old and new versions being highlighted to show the change in code	Code improvement suggestion by the code reviewer (Other than just pointing out the code errors)	Prioritizing code changes based on its level of importance and its effect on the functionality of the software	Integration of project tracking software (such as Trello, Jira etc.)	Integration of source-code editor (such as Visual Studio, Atom etc.)	Integration of business communication platform (such as Slack)	Total
1	Developer/ Programmer/ Software Engineer	12.50%	25.00%	8.33%	0.00%	20.83%	16.67%	16.67%	24
2	Team Lead	0.00%	33.33%	33.33%	0.00%	33.33%	0.00%	0.00%	3
3	DevOps Engineer/ Infrastructure Developer	33.33%	33.33%	33.33%	0.00%	0.00%	0.00%	0.00%	3
4	Architect	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0
5	UI/UX developer	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0
6	Technical Support	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0
7	Data Analyst/ Data Scientist/ Data Engineer	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0
8	Other	25.00%	0.00%	25.00%	25.00%	25.00%	0.00%	0.00%	4

Showing rows 1 - 8 of 8

شکل 4-6: 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).
- (2) بحث کد با نسخه‌های قدیمی و جدید که برای نشان‌دادن تغییر کد مشخص می‌شوند:
از بین همه شرکت‌کنندگان، 75٪ آنها توسعه‌دهنده، 12.50٪ آنها رهبران تیم و مهندس/توسعه‌دهنده زیرساخت DevOps بودند که این ویژگی CRRS را مهم می‌دانستند.
- (3) 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).

- 4) 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.
- 5) ادغام نرم‌افزارهای ردیابی پروژه (مانند Jira، Trello و غیره): برای این ویژگی، 71.43٪ توسعه دهندگان، 14.29٪ رهبران تیم و مدیر پروژه مفید دانستند.
- 6) ادغام ویرایشگر کد منبع (مانند Atom، Visual Studio و غیره): فقط توسعه‌دهندگان از همه شرکت‌کنندگان بودند که این ویژگی را به‌عنوان یک ویژگی مهم در نظر گرفتند.
- 7) ادغام بستر ارتباطات تجاری مان (Slack): این ویژگی فقط توسط توسعه‌دهندگان همه شرکت‌کنندگان که در نظرسنجی ما شرکت کردند مفید واقع شد.

فصل پنجم

5. جمع‌بندی و پیشنهادها

5-1 مقدمه:

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

5-2 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 مرحله‌ای 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.

5-2-1 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؟ به عبارت دیگر، 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، Gerrit، Phabricator، 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/مجموعه داده‌های 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 می‌کرد. REVFINDER CRRS دیگری بود که مرورگران کد را بر اساس شباهت مسیر فایل توصیه می‌کرد. به طور مشابه، براساس بررسی‌های 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 بازیابی 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 بازیابی 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.

3-5-3 پیشنهادها

پیشنهاد برای بهبود سیستم توصیه‌کننده مرور کد بر اساس یافته‌های حاصل از تحقیقات انجام شده در پایان‌نامه مورد بررسی یک سیستم توصیه‌بازبینی‌کننده 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 قابل مشاهده باشد. این جزئیات شامل تعداد پروژه‌هایی است که آنها روی آن کار کرده‌اند (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 وسیع‌تر از داده‌ها برای آموزش توصیه‌گر این مجموعه داده شامل نظرات مرور قبلی و پیام‌های متعهد و تجربه متقابل پروژه و فناوری خواهد بود. این داده‌ها به شما کمک می‌کند تا بدانید آیا کدی که باید بازبینی شود با تجربه پروژه‌ای 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 می‌دهد 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. با این حال، سیستم همچنین اجازه می‌دهد تا انتخاب کد برای بررسی پس از ادغام درگیری‌ها انجام شود تا در صورت لزوم از تأخیر جلوگیری شود و درعین حال 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 قابل مشاهده 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 به دست آمده 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 به صورت لینک به منابع بود و شخصاً کار مشکلی در درک مفاهیم آن داشتیم.
- همچنین در پایان‌نامه مورد بررسی، قسمتی تحت عنوان کلمات اختصاری و معرفی آنها وجود نداشت.
- همچنین هیچ توضیحی به عنوان پاورقی وجود نداشت.

5-6 جمع بندی و پیشنهادات

در کل و به عنوان نتیجه تحقیق، تعدادی از سیستم‌های 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 در CRRS‌های 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 توصیه‌بازبینی کد انجام دادیم که مشخص می‌کند آنها در CRRS چه ویژگی‌هایی مهم هستند و چه چیزی را می‌توان در CRRS های موجود بهبود بخشید.

در پایان‌نامه مورد بررسی 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 راه‌حل‌های موجود برای سیستم‌های توصیه‌برای مرورگران کد را شناسایی کردیم، عواملی که هنگام ایجاد CRRS و دسته‌بندی CRRS های موجود باید در نظر گرفته شوند. از سوی دیگر، با انجام نظرسنجی، ویژگی‌هایی را که برای یک سیستم توصیه‌برای مرورگران کد مهم هستند و چه پیشرفت‌هایی می‌توان در CRRS های موجود انجام داد، دریافتیم.

- 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.
- 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.
- 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/>.
- 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