دانشکده فنی و مهندسی واحد شهرری

گروه مهندسی کامپیوتر و فناوری اطلاعات

گزارش سمینار کارشناسی ارشد رشته مهندسی کامپیوتر

نرم‌افزار (M.Sc)

عنوان سمینار:

بررسی سیستم‌های پیشنهادی بازبینی کننده کدهای گذشته و حال

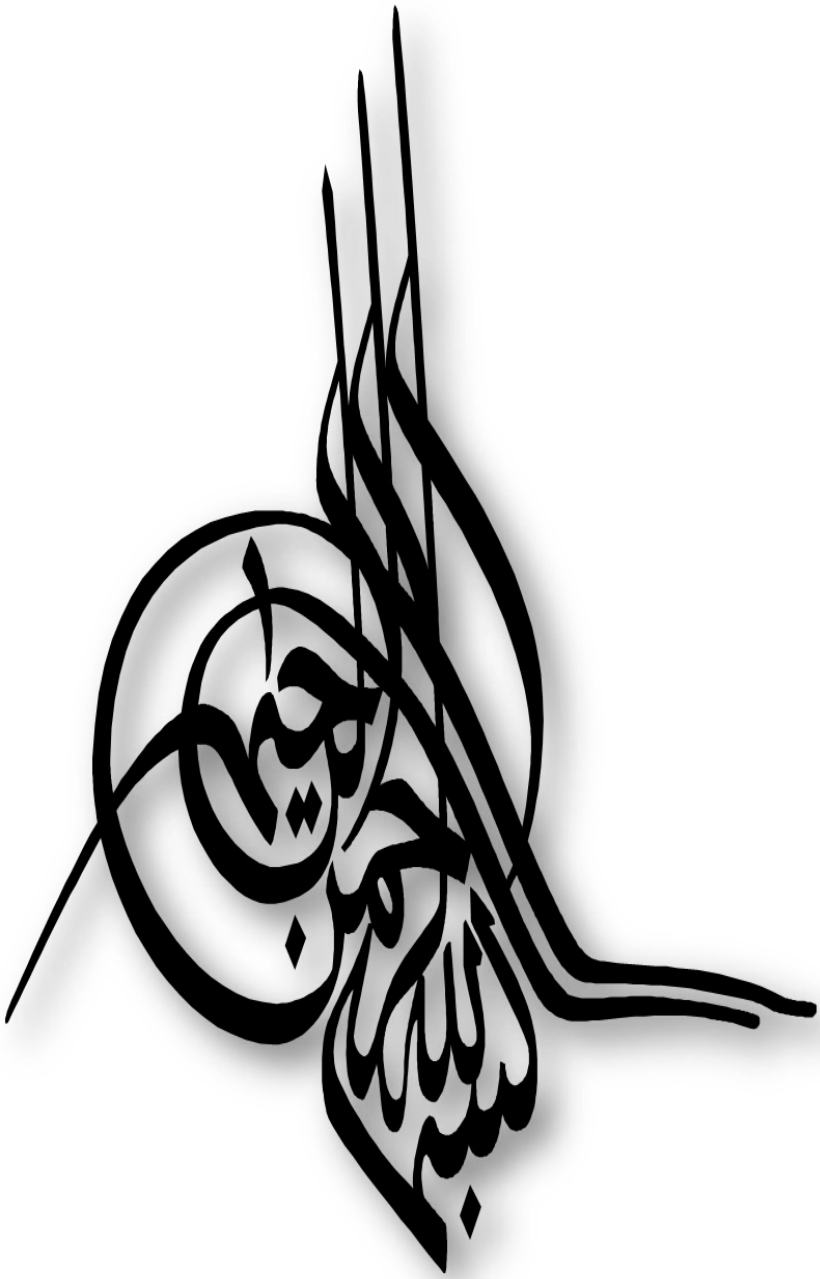
استاد راهنما:

دکتر سید علی رضوی

نگارنده:

زهرا کولیوند

شهریور ۱۴۰۰


بسم الله الرحمن الرحيم

فهرست مطالب

فهرست اشکال

چکیده:

این مقاله باهدف درک راه‌حل‌های موجود برای سیستم‌های توصیه بازنگری کد (CRRS)، عواملی که باید در هنگام ساخت آن‌ها در نظر گرفته شوند و ابعاد مختلفی که بر اساس آن می‌توان آنها را طبقه‌بندی کرد، است بعلاوه هدف ما درک ویژگی‌های مهم CRRS و آنچه می‌توان درCRRS های موجود بهبود داد، نیز می‌باشد. روش مطالعه در این مقاله با روش مروری ادبیات برای درک CRRSهای موجود انجام شد. یک نظرسنجی از اعضای پروژه توسعه نرم‌افزار برای درک ویژگی‌های مهم و گمشده در CRRS انجام شد.

بررسی کد یک بررسی سیستماتیک از کد منبع کامپیوتر است و اغلب به‌عنوان یک بررسی همتا انجام می‌شود. هدف بررسی کد شناسایی و اصلاح اشتباهات در کد منبع و همچنین بهبود کیفیت کد و مهارت‌های توسعه‌دهنده نرم‌افزار است. همچنین، هدف آن تنها 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 می کند. یافته‌های مقالات انتخاب شده را به دودسته دسته‌بندی کردیم: بر اساس نوع داده‌های مورداستفاده برای ارائه توصیه‌ها و نوع پروژه مورداستفاده برای ارزیابی. این نظرسنجی به ما کمک کرد تا ویژگی‌های گمشده در CRRS را درک کنیم و برخی روندها و الگوها را مشاهده کنیم.

کلمات کلیدی: بازبینی کد، برنامه‌نویسی جفتی، بررسی سیستماتیک

فصل اول

کلیات پژوهش

۱–۱ مقدمه

این پژوهش به بررسی سیستم‌های پیشنهادی بازبینی کننده کدهای گذشته و حال می‌پردازد همچنین در این سمینار ابتدا به درک راه‌حل‌های موجود برای سیستم‌های توصیه بازنگری کد (CRRS)، عواملی که باید در هنگام ساخت آن‌ها در نظر گرفته شوند و ابعاد مختلفی که بر اساس آن می‌توان آنها را طبقه‌بندی کرد، است پرداخته شد و همچنین هدف ما از این پژوهش درک ویژگی‌های مهم CRRS و آنچه می‌توان در CRRSهای موجود بهبود داد، نیز می‌باشد. روش مطالعه در این مقاله با روش مروری ادبیات برای درک CRRSهای موجود انجام شد. یک نظرسنجی از اعضای پروژه توسعه نرم‌افزار برای درک ویژگی‌های مهم و گمشده در CRRS انجام شد. بررسی کد یک بررسی سیستماتیک از کد منبع کامپیوتر است و اغلب به‌عنوان یک بررسی همتا انجام می‌شود.

این تحقیق مستندسازی آموخته‌های به‌دست‌آمده از «مطالعه مروری نظام‌مند ادبیات» و همچنین «نظرسنجی» انجام‌شده بر روی طیف وسیعی از اعضای پروژه نرم‌افزاری است. این کار به این دلیل انجام شد که تحقیقات بسیار کمتری در مورد سیستم‌های توصیه بازبین کد(CRRS) و تحقیقات بیشتری در مورد رویه‌ها و رویه‌های بررسی کد انجام شده علاوه بر کمک به کار بازبینی کد، تحقیقات قابل‌توجهی در مورد سیستم‌های توصیه‌ای انجام شده است که هدف آن ارائه توصیه‌های بازبینان کد بر اساس جنبه‌های مختلف است. دلایل مختلفی وجود دارد که چرا علاوه بر یافتن عیوب کد، به بازبینی کد نیاز است. دلیل این امر این است که بازبینی کنندگان کد نیز بر بهبود کد، یافتن راه‌حل‌های جایگزین برای یک مشکل، انتقال دانش، بداهه‌سازی در فرایند توسعه، اجتناب از شکست‌های ساخت،[1] اشتراک‌گذاری مالکیت کد و همچنین ارزیابی تیم تمرکز می‌کنند.

۱.هنگامی که یک توسعه‌دهنده تغییراتی را به مخزن کد منبع اضافه می‌کند که منجر به شکست فرآیند ساخت بعدی می‌شود، توسعه‌دهنده «ساخت را شکسته است»

۱-۳ تعریف مسئله و بیان سؤال‌های اصلی تحقیق:

بررسی کد یک بررسی سیستماتیک از کد منبع کامپیوتر است و اغلب به‌عنوان یک بررسی همتا انجام می‌شود. هدف بررسی کد شناسایی و اصلاح اشتباهات در کد منبع و همچنین بهبود کیفیت کد و مهارت‌های توسعه‌دهنده نرم‌افزار است. همچنین، هدف آن تنها 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.

سؤالات تحقیق برای این کار به شرح زیر است:

۱) راه‌حل‌های موجود برای سیستم‌های پیشنهادی برای بازبینی کنندگان کد چیست؟

۲) چه عواملی باید در هنگام ایجاد یک سیستم توصیه برای بازبینان کد در نظر گرفته شوند؟

۳) چگونه می‌توان سیستم‌های پیشنهادی موجود برای بازبینی کنندگان کد در ادبیات را دسته‌بندی کرد؟

٤) ویژگی‌های مهم یک سیستم توصیه برای بازبینان کد چیست؟

۵) چگونه می‌توان سیستم‌های پیشنهادی موجود برای بازبینان کد را بهبود بخشید؟ به‌عبارت‌دیگر، چه ویژگی‌هایی در پیاده‌سازی‌های موجود برای سیستم‌های توصیه بازبین کنندگان کد وجود ندارد؟

۱ و ۲ و ۳ با استفاده از مرور ادبیات سیستماتیک پاسخ داده می‌شوند، درحالی‌که ٤ و ۵ با بررسی طیف گسترده‌ای از اعضای پروژه نرم‌افزار پاسخ داده می‌شوند.

۱-۳ اهداف تحقیق:

۱-۳-۱ – هدف اصلی تحقیق:

بررسی سیستم‌های پیشنهادی بازبینی کننده کدهای گذشته و حال

۱-۳-۲ اهداف فرعی

هدف تحقیق ما دو مورد است: اول، یافتن پاسخ به سؤالات «گذشته» با انجام «مرور ادبیات سیستماتیک» و دوم یافتن پاسخ سؤالات «حال» با انجام یک نظرسنجی از اعضای پروژه نرم‌افزاری. یک «مرور ادبیات سیستماتیک» به یافتن جزئیات در مورد سیستم‌های پیشنهادی بازبین کد موجود کمک می‌کند، درحالی‌که این نظرسنجی به یافتن تغییراتی که مهندسان نرم‌افزار فکر می‌کنند موردنیاز است یا آنچه در سیستم‌های پیشنهادی بازبینی کد موجود وجود ندارد، کمک می‌کند.

۱-٤ ضرورت تحقیق:

هدف این تحقیق مستندسازی آموخته‌های به‌دست‌آمده از «مطالعه مروری نظام‌مند ادبیات» و همچنین «نظرسنجی» انجام‌شده بر روی طیف وسیعی از اعضای پروژه نرم‌افزاری است می‌باشد. این کار به این دلیل انجام شد که تحقیقات بسیار کمتری در مورد سیستم‌های توصیه بازبین کد (CRRS) و تحقیقات بیشتری در مورد رویه‌ها و رویه‌های بررسی کد انجام شده است. اهدافی که به ما در یافتن پاسخ برای نتیجه موردنیاز کمک می‌کند در زیر ذکر شده است:

۱.تجزیه‌وتحلیل راه‌حل‌های ارائه شده توسط "سیستم‌های پیشنهادی بازنگری کد" موجود در حال حاضر.

۲.برای درک راه‌حل‌ها/ویژگی‌هایی که در «سیستم‌های پیشنهادی بازنگری کد» موجود وجود ندارد.

۳.تجزیه‌وتحلیل ابزارهای مختلف انشعاب "سیستم‌های پیشنهادی بازنگری کد" موجود بر اساس روش اجرای آنها.

۱-۵ روش‌ها و مراحل انجام تحقیق:

۱.ما تعدادی از ویژگی‌های موجود در سیستم‌های پیشنهادی بازبینی کد موجود (CRRS)را شناسایی کردیم و آن ویژگی‌ها را بر اساس مفید بودنشان رتبه‌بندی کردیم.

۲. ما CRRSهای موجود را بر اساس ابعاد مختلف دسته‌بندی کردیم.

۳.ما ویژگی‌هایی را شناسایی کردیم که می‌توان هنگام انتخاب یک بازبین کد مهم در نظر گرفت.

٤.ما بهبودهای احتمالی را در CRRSهای موجود شناسایی کردیم تا یافتن بازبینی کننده‌های کد مناسب را تسهیل کنیم.

۱-٦ متغیرها:

کدنویسی، بازنگر

۱-۷ تعریف نظری و عملیاتی متغیرها

۱-۱-۷ تعریف نظری بررسی کد:

بررسی کد را می‌توان به‌عنوان یک بازرسی دستی از تغییرات در کد منبع در نظر گرفت. تعدادی ابزار و سیستم‌های پیشنهادی برای بررسی کد توسط تعدادی از سازمان‌های مختلف توسعه‌یافته‌اند.

۱-۲-۷- تعاریف عملیاتی بررسی کد:

تعریفی از مبحث بررسی کد در نتیجه پژوهش به دست می‌آید .

۱-۳-۷ - تعریف نظری بازنگر

تعدادی CRRS بر اساس سیستم‌های مسیر فایل(FPS) رویکرد مبتنی بر مکان فایل ارائه شده است. REVFINDER پیشنهاد شده است که از رویکرد بازبینی کد مبتنی بر مکان فایل پیروی می‌کند.

۱-٤-۷ تعریف عملیاتی بازنگر:

تعریفی از کلمه بازنگر در نتیجه پژوهش به دست می‌آید .

۱-۸ سازمان پایان‌نامه مورد بررسی:

فصل‌های این پایان‌نامه به‌صورت ذیل مرتب شده است:

- **فصل ۲:** کارهای مرتبط و این فصل خلاصه‌ای از مرور ادبیات قبلی انجام شده در مهندسی نرم‌افزار، داده‌کاوی و سیستم‌های توصیه‌گر را ارائه می‌دهد .
- **فصل ۳**: نتایج مطالعه مرور ادبیات و شیوه ما و ابزارهای بازبینی کد ارائه شده است.
- **فصل ٤:** بررسی نتایج به‌منظور انجام نظرسنجی از مهندسان نرم‌افزار برای تعیین نیازهای اطلاعاتی برای مرورگران کد ارائه شده است.
- **فصل ٥:**به سؤالات تحقیق خود پاسخ داده و بحث شده است.
- **فصل ٦:**درباره نتیجه‌گیری و کارهای آینده است.

۱-۹-ساختار گزارش تحقیق:

فصل ۱: به تعریف و مقدمه و دلایل نیاز به طرح ارائه شده پرداخته می‌شود.

فصل ۲: به‌مرور و پیش‌زمینه و کارهای وابسته پرداخته می‌شود.

فصل ۳ : به کاربردها و مزایا و معایب روش‌های مطرح شده پرداخته می‌شود.

فصل ٤ : نیز به جمع‌بندی و نتیجه‌گیری پرداخته می‌شود.

فصل دوم

۲. ادبیات و پیشینه تحقیق

۲-۱ مقدمه:

این فصل خلاصه‌ای از بررسی‌های ادبیات قبلی انجام شده در مهندسی نرم‌افزار، داده‌کاوی و سیستم‌های توصیه‌کننده را ارائه می‌کند. این مطالعات برای نشان‌دادن اینکه چگونه مطالعات مرور ادبیات درگذشته انجام شده و برای هدایت روش‌شناسی مرور ادبیات ما استفاده شده است، ارائه شده است.

بررسی ادبیات انجام شده در مورد سیستم‌های توصیه‌گر شامل سیستم‌های توصیه‌کننده است که هدف آن استخراج اطلاعات مرتبط از حجم عظیمی از دانش و سیستم‌های توصیه‌ای برای مهندسی نرم‌افزار است که ویژگی‌های موجود در سیستم‌های موجود، شکاف‌های تحقیقاتی و کارهای احتمالی آینده را ارائه می‌دهد. به طور مشابه، مطالعه مروری بر ادبیات در زمینه مهندسی نرم‌افزار دررابطه‌با مطالعات پیش‌بینی خطا و روش توسعه نرم‌افزار چابک انجام شد. بررسی ادبیات انجام شده در داده‌کاوی دو مدل پرکاربرد برای داده‌کاوی در CRM(مدیریت ارتباط با مشتری) را کشف کرد.

۲-۲ بررسی‌های ادبیات سیستم‌های توصیه‌کننده

مرور ادبیات توسط SUHENDROYONO ،Ismail ، HARUNA انجام شد در مورد سیستم‌های توصیه‌کننده آگاه از زمینه (CARS) که هدف‌آن استخراج اطلاعات مربوطه موردنیاز از حجم عظیمی از دانش است. هدف این نوع سیستم‌های توصیه‌گر ارائه اطلاعات متنی و مرتبط بر اساس «جستجوهای کاربر» و ارائه توصیه‌های شخصی شده‌تر برای کاربران است. از سه مرحله اصلی تشکیل شده است:

۱.مرحله اول شامل بررسی عمیق و طبقه‌بندی ادبیات بر اساس حوزه‌های مختلف مدل‌های کاربردی، فیلترینگ، استخراج و همچنین رویکردهای ارزیابی است.

۲.شامل ارائه نتایج بررسی با مزایا و معایب بررسی است.

۳.مرحله نهایی شامل برجسته‌کردن چالش‌ها/فرصت‌های احتمالی یا کار یا تحقیقات آینده است که می‌توان انجام داد. این شامل کمک به محققان تازه‌کار و جدید در درک پیش‌نیازهای توسعه CARS و همچنین ارائه این بررسی به‌عنوان 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 مرتبط به کاربران بر اساس نیازهای کاربر است.

برای این حوزه، یک مطالعه مروری نظام‌مند ادبیات مشابه با Gasparic و Janes انجام شد که نتایج عملکردی را که RSSEهای موجود (سیستم توصیه‌ای برای مهندسی نرم‌افزار) ارائه می‌دهند، شکاف‌های تحقیقاتی و همچنین جهت‌های تحقیقاتی ممکن را بررسی می‌کند. آنها یک رویکرد روش‌شناختی را دنبال کردند که شامل فیلترکردن مقالات پژوهشی جمع‌آوری‌شده و مرتبط بر اساس معیارهای مختلف بود. معیارهای خروج آنها شامل مقالاتی بود که به زمینه تحقیق بی‌ربط بودند، مقالاتی که راه‌حل‌های اجرا نشده را توصیف می‌کردند یا مقالاتی که به طور کامل در دسترس نبودند. برای استخراج مقالات مربوطه، مقالات بر اساس محتوای شرح داده شده در چکیده مقاله یا گاهی عنوان، فیلتر و تقسیم شدند. نویسندگان پس از پیروی از رویکرد روش‌شناختی خود، به چهار سؤال تحقیقاتی خود پاسخ دادند که عبارتنداز:

- خروجی ارائه شده توسط RSSEهای موجود.
- مزایایی که این RSSEها برای مهندسان نرم‌افزار ارائه می‌دهند.
- انواع ورودی‌هایی که این RSSEها نیاز دارند و تلاش‌هایی که یک RSSE انجام می‌دهد. مهندس نرم‌افزار باید برای استفاده از این RSSEها قرار دهد. مشاهده شد که برخی از خروجی‌های ارائه شده توسط 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 استفاده مجدد، اشکال‌زدایی، پیاده‌سازی، مراحل/فعالیت‌های نگهداری و پشتیبانی از بهبود کیفیت سیستم به نفع مهندسان نرم‌افزار. برخی از ورودی‌هایی که این RSSEهای موجود استفاده می‌کنند شامل فایل‌های گزارش، ارتباط بین مهندسان نرم‌افزار، کد منبع، ورودی کاربر (به‌عنوان‌مثال، عبارت‌های جستجو، پرس‌وجو، تنظیمات، اولویت‌ها)، مصنوعات آزمایشی و فرایند توسعه نرم‌افزار است. همچنین تلاش‌هایی که یک مهندس نرم‌افزار برای استفاده از RSSEهای موجود می‌خواهد به‌عنوان تلاش گسترده، تلاش کم و بدون تلاش دسته‌بندی می‌شود. [M. Gasparic and A. Janesو همکاران ۲۰۱۶]

۲-۳ بررسی ادبیات در مهندسی نرم‌افزار

پیش‌بینی دقیق عیوب در کد می‌تواند هزینه تست را تا حد زیادی کاهش دهد و همچنین کیفیت محصول نرم‌افزار را افزایش دهد. برای این منظور، مطالعه مروری بر ادبیات توسط Hall, Beecham, Bowes انجام شد که بر مطالعات پیش‌بینی خطا متمرکز بود. نویسندگان رویکرد مرور سیستماتیک ادبیات را همان‌طور که توسط Charters پیشنهاد شده بود، دنبال کردند که در آن مراحل اولیه شامل مقالات و مطالعات مربوطه و حذف مطالعات مکرر است. در حذف مقالات به جنبه‌های مختلفی توجه می‌شود، مانند مقالاتی که از منابع مختلف مانند مجلات، کنفرانس‌ها و پایگاه‌های اطلاعاتی استخراج شده و بر اساس محتوای عنوان و چکیده مرتب شده‌اند. نویسندگان در نهایت حدود ۲۰۸ مقاله داشتند. یافته‌ها نشان می‌دهد که اکثر مطالعات اطلاعات زمینه‌ای و روش‌شناختی کافی برای درک کامل یک مدل را گزارش نمی‌کنند. نویسندگان مجموعه‌ای از معیارها را ارائه می‌کنند که مجموعه‌ای از جزئیات زمینه‌ای و روش‌شناختی ضروری را که مطالعات پیش‌بینی خطا باید گزارش کنند، شناسایی می‌کنند.

آنها مطالعه را در سه مرحله اصلی انجام دادند: برنامه‌ریزی، انجام و گزارش.

۱. **"برنامه‌ریزی"** شامل یافتن نیاز شناسایی برای بازبینی، چارچوب‌بندی سؤالات تحقیق و توسعه و ارزیابی پروتکل بازبینی بود.
۲. **"انجام"** باهدف جستجوی مقالات پژوهشی، انتخاب مقالات مربوطه برای مطالعه، ارزیابی کیفی و استخراج و تجزیه‌وتحلیل داده‌ها است.
۳. **"گزارش"** باهدف استخراج و بحث در مورد نتایج به‌دست‌آمده از مرحله قبل و سپس نوشتن، ارزیابی و قالب‌بندی گزارش نهایی برای مطالعه است.

۲-٤ بررسی ادبیات در داده‌کاوی

بررسی ادبیات انجام شده توسط Ngai، Xiu,Chau نمونه دیگری از روش‌شناسی برای مرور ادبیات سیستماتیک در حوزه داده‌کاوی را ارائه می‌دهد.
تکنیک‌های داده‌کاوی برای مدیریت ارتباط با مشتری (CRM) اعمال می‌شود وNgai ، Xiu با کمک بررسی ادبیاتی که انجام داده‌اند، بینش کاملی در این مورد ارائه می‌دهند. نویسندگان حدود ۸۷ مقاله تحقیقاتی مرتبط را برای این منظور جمع‌آوری کردند که بر اساس چهار بعد CRM تقسیم شدند که شامل توسعه مشتری، شناسایی مشتری، جذب مشتری و حفظ مشتری و هفت تکنیک داده‌کاوی می‌شود. ارتباط، طبقه‌بندی، خوشه‌بندی، پیش‌بینی، رگرسیون، کشف توالی و تجسم. جدای از این، برای وضوح بیشتر، ابعاد CRM بیشتر به ۹ زیرشاخه از عناصر CRM که تحت تکنیک‌های داده‌کاوی قرار می‌گیرند، طبقه‌بندی شدند. بر اساس مطالعه، مشخص شد که طبقه‌بندی و انجمن دو مدل پرکاربرد برای داده‌کاوی در CRM هستند. همچنین، از چهار بعد CRM، حفظ مشتری بیشترین تحقیق را دارد، اگرچه بیشتر آنها مربوط به برنامه‌های بازاریابی و وفاداری یک‌به‌یک بودند.

۲-۵ خلاصه

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

فصـــل سوم

اقدامات انجام شده و یافته ها

۳–۱ مقدمه:

این مقاله باهدف بررسی سیستم‌های پیشنهادی بازبینی کننده کدهای گذشته و حال انجام‌گرفته است، در این سمینار ابتدا به درک راه‌حل‌های موجود برای سیستم‌های توصیه بازنگری کد(CRRS)، عواملی که باید در هنگام ساخت آن‌ها در نظر گرفته شوند و ابعاد مختلفی که بر اساس آن می‌توان آنها را طبقه‌بندی کرد، است پرداخته شد و همچنین هدف ما از این پژوهش درک ویژگی‌های مهم CRRS و آنچه می‌توان در CRRSهای موجود بهبود داد، نیز می‌باشد. روش مطالعه در این مقاله با روش مروری ادبیات برای درک CRRSهای موجود انجام شد. یک نظرسنجی از اعضای پروژه توسعه نرم‌افزار برای درک ویژگی‌های مهم و گمشده در CRRS انجام شد. این تحقیق مستندسازی آموخته‌های به‌دست‌آمده از «مطالعه مروری نظام‌مند ادبیات» و همچنین «نظرسنجی» انجام‌شده بر روی طیف وسیعی از اعضای پروژه نرم‌افزاری است. این کار به این دلیل انجام شد که تحقیقات بسیار کمتری در مورد سیستم‌های توصیه بازبین کد (CRRS)و تحقیقات بیشتری در مورد رویه‌ها و رویه‌های بررسی کد انجام شده است همچنین این مقاله باهدف درک راه‌حل‌های موجود برای سیستم‌های توصیه بازنگری کد (CRRS)، عواملی که باید در هنگام ساخت آن‌ها در نظر گرفته شوند و ابعاد مختلفی که بر اساس آن می‌توان آنها را طبقه‌بندی کرد، است بعلاوه هدف ما درک ویژگی‌های مهم CRRSو آنچه می‌توان در CRRSهای موجود بهبود داد، نیز می‌باشد. روش مطالعه در این مقاله با روش مروری ادبیات برای درک CRRSهای موجود انجام شد. یک نظرسنجی از اعضای پروژه توسعه نرم‌افزار برای درک ویژگی‌های مهم و گمشده در CRRS انجام شد. ابزاری به نام Review Bot یکی دیگر از ابزارهای توسعه‌یافته توسط Balachandran است‌که درپروژه VMware مورداستفاده قرار گرفت. Review Bot از الگوریتمی تشکیل شده بود که تغییرات کد انجام شده در یک خط کد را به روشی کاملاً شبیه به دستور git blame بررسی

می‌کند. به هر نویسنده‌ای که روی تغییر کد در کد منبع کارکرده باشد، امتیاز تعلق می‌گیرد، اما نویسندگانی که تغییرات اخیر دارند امتیاز بیشتری نسبت به نویسندگانی که تغییرات قدیمی‌تر دارند، کسب می‌کنند. در پایان، از جمع‌بندی هر نویسنده برای تعیین نویسندگان برتر استفاده می‌شود و سپس به آن‌ها توصیه می‌شود تا مرورگر کد شوند.

۳-۲ بازنگر (REVFINDER)

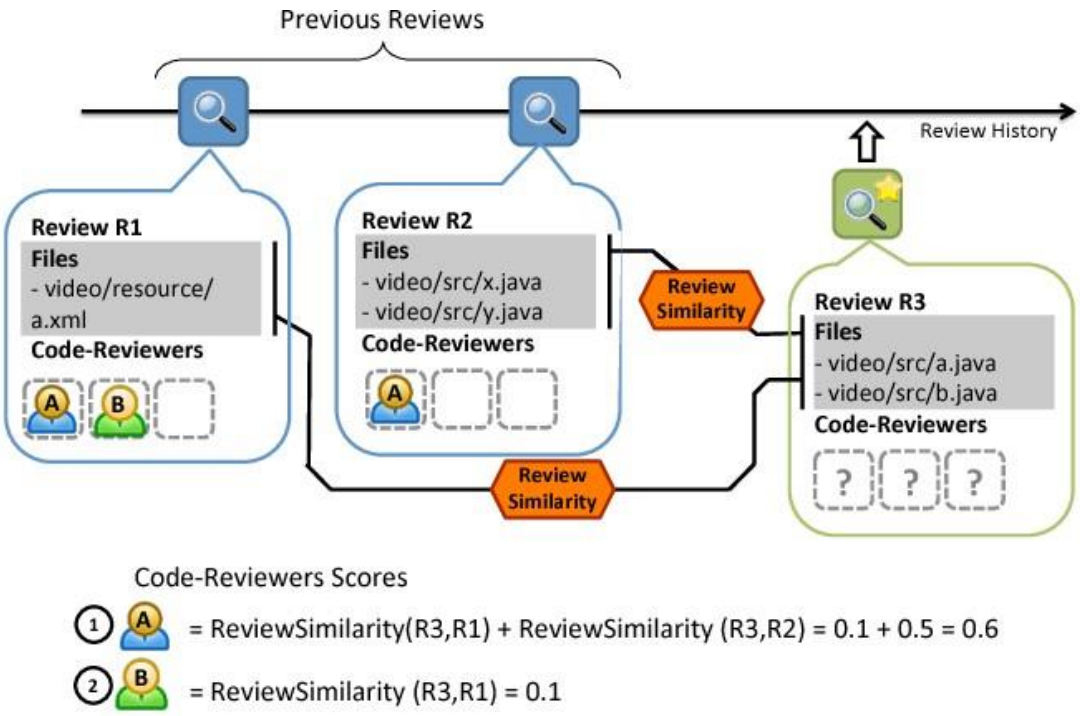
تعدادی CRRS بر اساس سیستم‌های مسیر فایل(FPS) رویکرد مبتنی بر مکان فایل ارائه شده است. REVFINDER پیشنهاد شده است که از رویکرد بازبینی کد مبتنی بر مکان فایل پیروی می‌کند. شهود پشت این رویکرد این است که چندین فایل با یک مکان/مسیر فایل مشابه توسط بازبینی‌کنندگان کد باتجربه مشابه بررسی و مدیریت می‌شوند.

Thongtanunam، Tantithamthavorn، Kula،همچنین یک مطالعه اکتشافی در مورد اینکه چگونه تخصیص مرورگر کد بر زمان بازبینی تأثیر می‌گذارد، انجام دادند. مطالعه اکتشافی نشان داد که حدود ۳۰-٤ درصد از بررسی‌های کد با مشکل تعیین بازنگری کد صحیح روبرو هستند و حدود ۱۲ روز بیشتر طول می‌کشد تا تغییر کد تأیید شود. بر اساس نتایج این مطالعه، نویسندگان REVFINDER را پیشنهاد کردند.

REVFINDER شامل دو بخش است:

- الگوریتم رتبه‌بندی بازبینان کد: نویسندگان از الگوریتم رتبه‌بندی بازبینان کد استفاده کردند تا نمرات بازبینی کنندگان کد را بر اساس شباهت مسیرهای فایل‌هایی که قبلاً بررسی شده‌اند ارزیابی کنند.
- تکنیک ترکیبی

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



شکل ۳-۱: مثال محاسبه الگوریتم رتبه‌بندی Code-Reviewers

۳-۳ فاز ساخت مدل

فاز ساخت مدل شامل یک مدل ترکیبی به نامTIECOMPOSER که با استفاده از بررسی‌های تاریخی بازبینان شناخته شده ساخته شده است. در این مرحله، سیستم TIE ابتدا بررسی‌های آموزشی بازبینان شناخته شده را از محتوای متنی بررسی‌های گذشته و مسیرهای فایل و همچنین زمان آپلود جمع‌آوری می‌کند. در مرحله بعد، TIE یک مدل متن‌کاوی را بر اساس داده‌های متنی پردازش شده با استفاده از تکنیک طبقه‌بندی متن می‌سازد. شهود پشت حالت داده‌کاوی این است که همان بازبینان احتمال بیشتری دارد که تغییرات را با اصطلاحات یا کلمات مشابه بررسی کنند.

TIE همچنین از یک رویکرد مبتنی بر مکان فایل آگاه به زمان استفاده می‌کند که هدف آن محاسبه شباهت بین بررسی‌های جدید و تاریخی است. این شباهت بین مسیرهای فایل تغییریافته (مسیرهای فایل‌هایی که در درخواست بررسی جدید تغییر یا اصلاح شده‌اند)و مسیرهای فایل بررسی شده (مسیرهای فایل‌هایی که در بررسی‌های تاریخی بررسی شده‌اند) محاسبه می‌شود. شهود پشت رویکرد مبتنی بر مکان فایل این است که همان بازبینان تمایل دارند فایل‌ها یا فایل‌هایی را با مسیرهای مشابه بررسی کنند. این دو مدل برای ساخت مدل TIECOMPOSER با هم ترکیب شده‌اند.

۳-٤ مرحله توصیه

برای این مرحله، از TIE برای توصیه بازبینی کنندگان کد برای درخواست بازبینی اختصاص‌نشده جدید استفاده می‌شود. TIE ابتدا توضیحات تغییر مسیرهای فایل و زمان آپلود را همان‌طور که برای بررسی‌های تاریخی در «مرحله ساخت مدل» انجام شد، استخراج می‌کند. برای مرحله بعدی، داده‌های متنی از توضیحات استخراج شده و به‌عنوان ورودی در مدل داده‌کاوی ساخته شده در "مرحله ساخت مدل" استفاده می‌شود. به طور مشابه، سیستم همچنین مسیرهای فایل و زمان آپلود را در مدل مشابه ساخته شده در "مدل ساخت فاز" وارد می‌کند.

سپس این دو مدل فهرستی از بازبینی کننده‌های کد را تولید می‌کنند و سپس این دو فهرست با استفاده از مدل TIECOMPOSER ساخته‌شده در «مرحله ساخت مدل» ترکیب می‌شوند.

۳-۵ مرحله به‌روزرسانی مدل

در مرحله به‌روزرسانی مدل، سیستم TIE با استفاده از بازبینی کننده‌های کد جدید اختصاص‌داده‌شده به‌روز می‌شود. در عمل، توسعه دهندگان به طور معمول لیست

بازبینان بالقوه را بررسی می‌کنند و سپس یک درخواست کشش جدید را به گروهی از بازبینان اختصاص می‌دهند.

CodeFlow ٦-٣

Bosu، یک مطالعه تجربی در مایکروسافت در مورد ویژگی‌های بررسی کد مفید با انجام مصاحبه با توسعه‌دهندگان و همچنین تجزیه‌وتحلیل نظرات بررسی پنج پروژه مایکروسافت با استفاده از CodeFlow CRRS انجام دادند. مطالعه در سه مرحله انجام شد. ابتدا، آنها یک مطالعه اکتشافی را با انجام مصاحبه با توسعه دهندگان انجام دادند تا تفسیر آنها از "مفید" را در زمینه بررسی کدها درک کنند. ثانیاً، آنها یک طبقه‌بندی برای تفکیک نظرات «مفید» و «غیرمفید» با استفاده از داده‌های مصاحبه می‌سازند. در نهایت، آنها طبقه‌بندی‌کننده خود را در پنج پروژه مایکروسافت اعمال کردند تا نظرات «مفید» و «غیرمفید» را تشخیص دهند.[M. Greiler,Bose, and C. Bird,2015]

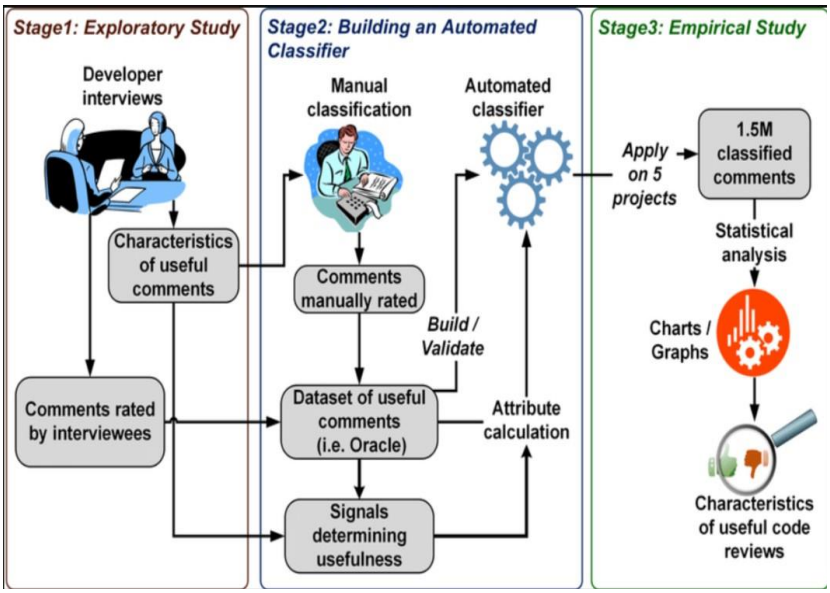
روش تحقیق سه‌مرحله‌ای:

گردش‌کار CodeFlow نسبتاً ساده است.

١. 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.

٢. 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 مفید و غیرمفید طبقه‌بندی کردند. بازبین می‌تواند تغییر در خود ابزار را مرور کند.

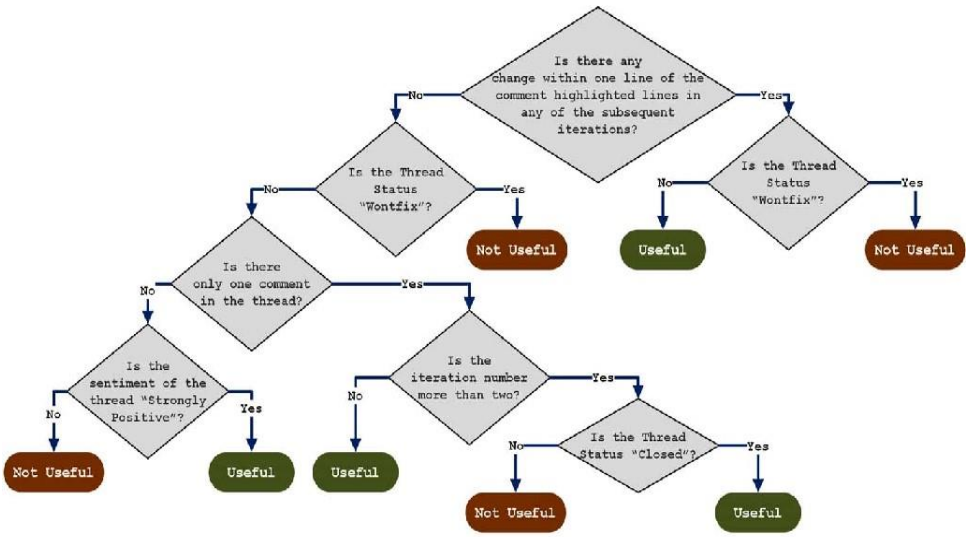
٣. هنگامی که یک داور می‌خواهد در مورد یک خط یا بلوک کد نظر دهد، منتقد آن قسمت از کد را برجسته و اضافه می‌کند. این نظرات به‌عنوان موضوعاتی که در آن بحث شروع می‌شود و همچنین نقاط تعامل برای افرادی که در بررسی مشارکت دارند، ظاهر می‌شود.



شکل ۳-۲: روش تحقیق سه‌مرحله‌ای

هر یک از این موضوعات دارای وضعیتی هستند که شرکت‌کنندگان می‌توانند در طول دوره بررسی آن را تغییر دهند. این وضعیت در ابتدا "فعال" است و باگذشت زمان می‌توان آن را به "در انتظار"، "حل شده" تغییر داد، "رفع نمی‌شود" و "بسته است". در CodeFlow، هر به‌روزرسانی یک «تکرار» نامیده می‌شود و چرخه بررسی دیگری را تشکیل می‌دهد؛ بنابراین، ممکن است قبل از اینکه تغییر در کد در نهایت در مخزن کد منبع ادغام شود، چندین بار تکرار شود.

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



شکل ۳-۳ : مدل درخت تصمیم‌گیری برای طبقه‌بندی نظرات مفید

بر اساس گره‌های تصمیم، نظرات به‌عنوان مفید یا غیرمفید طبقه‌بندی می‌شوند. به‌منظور ارزیابی روش پیشنهادی، نویسندگان از نظرات پنج پروژه بزرگ مایکروسافت که شامل Azure، Bing، Visual Studio، Exchange و Office بودند، استفاده کردند. بر اساس نتایج، نویسندگان به این نتیجه رسیدند:

۱. توسعه‌دهندگانی که درگذشته تغییراتی ایجاد کرده‌اند یا یک قطعه کد یا یک مصنوع را بررسی کرده‌اند نظرات مفیدتری ارائه می‌دهند.

۲. تفاوت قابل‌توجهی در مفید بودن نظرات وجود دارد (یعنی آن دسته از نظراتی که دارای کلماتی مانند "اصلاح"، "اشکال" یا "حذف" هستند به‌عنوان نظرات "مفید" در نظر گرفته می‌شوند) که توسط بازبینان در همان تیم ارائه می‌شود و نظرات ارائه شده توسط نویسنده و داور از تیم‌های مختلف.

۳. تعداد نظرات مفید در طول زمان برای چهار پروژه از پنج پروژه افزایش یافت و دلیل این امر افزایش تجربه بازبینان باگذشت زمان در نظر گرفته شد.

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

۱. این مطالعه نشان داد که تعداد سودمندی نظرات مرور کد باتجربه توسعه‌دهنده یک کد افزایش‌یافته است.

۲. این مطالعه همچنین پیشنهاد کرد که اثربخشی بررسی‌ها با افزایش تعداد پرونده‌ها کاهش می‌یابد. پیشنهاد شد که توسعه دهندگان باید تغییرات کوچک‌تر را با تعداد فایل‌های بیشتر برای بررسی ارسال کنند.

۳. تراکم سودمندی نظرات می‌تواند توسط تیمی از توسعه دهندگان برای شناسایی مناطقی که بررسی کد در آنها مؤثرتر است استفاده شود.

[A. Bosu, M. Greiler, and C. Bird,2015]

۳-۷ خلاصه

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

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

فصل چهارم

۴. جمع‌بندی و پیشنهادها

۴-۱ مقدمه:

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

٤-٢ نتایج حاصل از تحقیق

ما نتایج به‌دست‌آمده با بررسی طیف گسترده‌ای از اعضای پروژه نرم‌افزاری را در اینجا ارائه کردیم که در آن متوجه شدیم کدام ویژگی‌های CRRS مفیدتر هستند، کدام ویژگی‌ها در سیستم موجود گم شده‌اند و چه عواملی هنگام انتخاب یک بازنگری کد مربوطه مهم هستند. ما همچنین ترجیحات توسعه‌دهنده/بازبین را نسبت به اینکه چه نوع بررسی کد باید انجام شود (بررسی طولانی یا کوتاه) و در چه مرحله‌ای از گردش‌کار باید انجام شود، به دست آوردیم. جدای از این، ما همچنین گرایش‌ها و الگوهایی رابین استفاده از سیستم CRRS و ارتباط آن با اطلاعات جمعیت‌شناختی بازبین/توسعه‌دهنده پیدا کردیم.

٤-٢-١ پاسخ به سؤالات تحقیق

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

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

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

۳. چگونه می‌توان سیستم‌های توصیه‌ای موجود برای مرورگران کد در ادبیات را دسته‌بندی کرد؟

٤. ویژگی‌های مهم سیستم توصیه برای مرورگران کد چیست؟
۵. 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 ۱:

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 ۲:

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 ۳:

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، بر اساس 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 ٤:

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:

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

٢ .ادغام با یک نرم‌افزار ردیابی مسئله‌مان Trello ، JIRA و غیره.

٣ .بررسی کد را از قبل مرتکب شوید.

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

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

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

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

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

٩ .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.

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

۱۱. گزینه‌ای برای انتخاب یک شاخه یا پرونده خاص در یک پروژه برای حفظ گردشکار سیستماتیک و روش بازبینی کد سازمان‌یافته.
۱۲. تشخیص کد را با استفاده از یک طرح کدگذاری رنگی با نام توسعه‌دهنده منتقل کرد.

جواب سؤال ۵:

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:

۱ .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.

۲. 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 از تجربه و تخصص مفید است.

٤-۳ پیشنهادها

یک سیستم توصیه بازنگری کد بهبودیافته را پیشنهاد می‌کنیم که تمام ویژگی‌های لازم را داشته باشد که در همه سیستم‌ها یا ویژگی‌هایی که در سیستم‌های موجود وجود ندارند وجود ندارد. سیستم توصیه‌گر پیشنهادی دارای ویژگی‌های زیر خواهد بود:

۱. شفاف‌تر در شرایطی که تمام جزئیات مربوط به بازبینی کد روی داشبورد قابل‌مشاهده باشد. این جزئیات شامل تعداد پروژه‌هایی است که روی آن‌ها کارکرده‌اند (تجربه کاری آنها)، زمینه کاربردی خاص که در آن تخصص دارند، تعداد بررسی‌های کد انجام‌شده توسط آنها و حجم کاری آنها (یعنی تعداد بررسی‌هایی که در حال

حاضر بازبین انجام می‌شود. بررسی) برای اطمینان از اینکه بازبینی کننده با بررسی کدهای جدید بیش از حد سنگین نیست. اعتقاد بر این است که ارائه این جزئیات در نتیجه روند بررسی کد را سرعت می‌بخشد.

۲. ترکیب گسترده‌تری از داده‌ها برای آموزش توصیه‌کننده. این مجموعه‌داده شامل نظرات مرور گذشته و پیام‌های تعهد و تجربه بین پروژه‌ای و فناوری مرتبط خواهد بود. این داده‌ها به دانستن اینکه آیا کدی که باید بازبینی شود مطابقت نزدیکی باتجربه پروژه که یک بازبین دارد، کمک می‌کند. به طور مشابه، تاریخچه گذشته نظرات و تعهدات بررسی به انتخاب یک بازبین مربوطه بر اساس تعداد تعهدات و بررسی‌هایی که قبلاً توسط آنها انجام شده و اینکه بررسی‌های کد گذشته چقدر برای توسعه دهندگان مفید بوده است، کمک خواهد کرد. این کمک می‌کند تا اطمینان حاصل شود که نظرات بررسی آینده آنها برای بررسی کد مفید خواهد بود.

۳. بررسی کد قبل از ادغام کد و تداخل کد بالقوه انجام می‌شود؛ بنابراین، سیستم پیشنهادی، قبل از وقوع تضادهای ادغام، بازبینان کد را توصیه می‌کند. بااین‌حال، این سیستم همچنین اجازه می‌دهد تا پس از تضادهای ادغام، انتخاب بازبینی کد انجام شود تا در صورت نیاز از تأخیر جلوگیری شود و درعین‌حال از ایجاد یک محصول نرم‌افزاری باکیفیت خوب اطمینان حاصل شود.

٤-٤ کارهای آینده

جهت‌گیری‌های احتمالی آینده بر اساس این کار عبارت‌اند از:

مروری بر ادبیات سیستماتیک گسترده‌تر: یک مطالعه مروری سیستماتیک گسترده‌تر می‌تواند انجام شود که نه‌تنها به سیستم‌های توصیه بازبین کد، بلکه به شیوه‌ها و رویه‌های بررسی کد نیز می‌پردازد. این می‌تواند به ارائه تصویر بهتری در مورد نیازهای اعضای پروژه نرم‌افزاری در مورد بررسی کد همراه با استفاده از CRRS کمک کند.

ساختن یک سیستم توصیه بازبینی کد: برای کارهای آینده، هدف ما ساختن سیستمی است که تمام جزئیات بازبین را در داشبورد سیستم قابل‌مشاهده باشد (یعنی تجربه کاری، تجربه فناوری، تعداد بررسی‌های کد انجام شده و غیره). همچنین، سیستم داده‌های بیشتری برای آموزش سیستم توصیه‌گر خواهد داشت که شامل نظرات مرور گذشته و پیام‌های تعهد، پروژه متقابل مرتبط و تجربه فناوری است. بر اساس بازخورد به‌دست‌آمده از نظرسنجی، بررسی‌ها قبل از وقوع تضادهای ادغام انجام می‌شود.

٤-٥ 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.

٤-٦ جمع‌بندی و پیشنهادها

در این تحقیق ما تعدادی از سیستم‌های پیشنهادی بازنگری کد (CRRS)را که در ادبیات یافت می‌شوند، راه‌های مختلف طبقه‌بندی این سیستم‌ها، چه ویژگی‌هایی برای یک سیستم توصیه‌ای برای بازبینی کنندگان کد مهم هستند و سیستم‌های موجود را چگونه می‌توان شناسایی کرد. بهبودیافته یا اینکه کدام ویژگی در CRRSهای موجود وجود ندارد. یک مطالعه مرور ادبیات سیستماتیک برای شناسایی سیستم‌های پیشنهادی بازبینی کد موجود و درک جزئیات مربوط به این سیستم‌ها که شامل ویژگی‌ها و عواملی است که برای یک CRRS مهم هستند، انجام شد. سپس ما یک

نظرسنجی برای درک نیازهای اعضای پروژه نرم‌افزار در مورد سیستم‌های توصیه بازنگری کد انجام دادیم که مشخص می‌کند کدام ویژگی‌ها در یک CRRS مهم هستند و چه چیزی می‌تواند در CRRSهای موجود بهبود یابد.

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 توصیه برای مرورگران کد را شناسایی کردیم، عواملی که هنگام ایجاد CRRS و دسته‌بندی 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.
  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

This article aims to understand the solutions available for Code Review (CRRS) systems, the factors to consider when building them, and the various dimensions by which they can be classified. In addition, we aim to understand the important features of CRRS and what It can also be improved in existing CRRS. Methods The study in this paper was performed with the literature review method to understand the existing CRRS. A survey of members of the software development project was conducted to understand important and missing features in CR.RS Code review is a systematic review of computer source code and is often performed as a peer review. The purpose of the code is to identify and correct errors in the source code, as well as to improve the quality of the code and the skills of the software developer. Also, its purpose is not only to improve the quality of the code or to find defects in the source code. It also increases team awareness as well as helps distribute knowledge. It also encourages shared code ownership. We categorized the findings of the selected articles into two categories: based on the type of data used to provide recommendations and the type of project used for evaluation. This survey helped us to understand the missing features in CRRS and to 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 2

۳۵