مقاله پژوهشی

مروری بر ابزارها و روش‌های داده‌کاوی موثر بر مخازن‌کاوی نرم‌افزار

Doi: 10.30508/kdip.2024.453254.1102

حدیث شفائی[1]

۱-گروه مهندسی نرم افزار، دانشگاه علوم و تحقیقات، تهران، ایران

تاریخ دریافت: ۱۴۰۳/۰۲/۰۱
تاریخ پذیرش: ۱۴۰۳/۰۳/۲۱
صفحه: ۷۵ - ۵۴

چکیده

مخازن‌کاوی نرم‌افزار، عملیات استخراج و تحلیل داده ارزشمند از مخازن نرم‌افزاری است. از این رو استفاده از ابزارها و روش‌های داده‌کاوی مناسب برای کاوش درست مخازن نرم افزاری بسیار حائز اهمیت است. داده‌هایی که از مخازن‌کاوی نرم‌افزار بدست می‌آید، برای توسعه نرم‌افزار مفید می‌باشد و توسعه نرم‌افزار نیز هدف نهایی مخازن‌کاوی نرم‌افزار است. در این تحقیق کمیت و کیفیت پژوهش‌ها در استفاده از ابزارها و روش‌های داده‌کاوی با استفاده از پایگاه گوگل اسکولار مورد تحلیل و ارزیابی قرار گرفته است. نتایج نشان می‌دهد که ابزارهایی چون GNUPlot ، RapidMiner، Weka، Eclips، برنامه‌هایی چون Perl و R توسط محققین به منظور مخازن‌کاوی نرم‌افزار بسیار مورد کاربرد است که در بهینه‌سازی کاوش مخازن نرم‌افزاری و تکامل کد و کنترل نسخه و ردیابی اشکال و تغییرات صحیح کد بسیار موثر می‌باشد. تعداد پژوهش‌های صورت گرفته با ابزار مورد نظر کمتر از نیمی از تعداد پژوهش‌ها انجام شده با سایر ابزارها را در بر می‌گیرد، که به نسبت نشان دهنده استقبال خوب محققین از ابزارهای مورد بررسی در این پژوهش می‌باشد.

کلمات کلیدی: مخازن کاوی نرم‌افزار، توسعه نرم افزار، گوگل اسکولار، ابزارها و روش‌های داده‌کاوی.

۱- مقدمه

کاوش مخازن نرم‌افزاری[1] (MSR) یک حوزه تحقیقی در حال رشد از مهندسی نرم‌افزار (SE) می‌باشد. بنابر ضرورت آن از سال ۲۰۰۴، تعداد زیادی از پژوهش‌ها جنبه‌های متفاوتی از این مطالعات را مورد تحلیل قرار داده‌اند. مخازن‌کاوی نرم‌افزاری به عنوان یک جهت تحقیقاتی در دهه گذشته ضرورت یافته است و در دو زمینه تحقیقاتی و عملیاتی برای پشتیبانی وظایف نگهداری نرم‌افزار موفقیت قابل‌توجهی به دست آورده است. مخازن نرم‌افزاری شامل مخازن اشکال[2]، آرشیوهای ارتباطات، و مخازن کنترل منبع، ... می‌باشد. مخازن مورد استفاده برای پشتیبانی نگهداری نرم‌افزار، دربرگیرنده اطلاعات غیرمرتبط در هر مخزن هستند که منجر به کاهش کارایی یا حتی نتایج اشتباه می‌شود (ویدونی[3]، ۲۰۲۲). فرآیند نظام‌مند برای مخازن‌کاوی نرم‌افزار، مرور ادبیات نظام‌مند هدایت‌شده (SLRs) را به منظور تحلیل مواد و تضمین ارتباط مطالعات موجود برای پروژه‌های تحقیقی شامل می‌شود. این فرآیند به شناسایی چگونگی هدایت مطالعات مخازن‌کاوی نرم‌افزار بویژه از نظر انتخاب مخزن و استخراج داده کمک می‌کند. تعداد زیادی مطالعات MSR رویکرد نظام‌مندی را در انتخاب مخزن یا انتخاب گزارش و پروتکل‌های استخراج داده دنبال نمی‌کنند. در نتیجه به دستورالعمل انجام مطالعات نظام‌مند به طور موثر نیاز می‌باشد. مخازن‌کاوی نرم‌افزار، در تحلیل داده درون مخازن نرم‌افزاری به منظور استخراج دید ارزشی در مورد سیستم‌های نرم‌افزاری و پروژه‌ها کاوش می‌کنند. محققین در این حوزه سوالات مهندسی نرم‌افزاری متنوعی را با استفاده از داده مخزن همانند ارزیابی نرم‌افزار، مدل‌های فرآیند توسعه، توصیف توسعه‌دهنده، پیش‌بینی کیفیت‌های نرم‌افزار، تکنیک‌های یادگیری ماشین روی داده پروژه، پیش‌بینی خطا، تحلیل الگوی تغییر کد، و شبیه‌سازی کد استخراج می‌کنند. اهمیت MSR با توجه به داده محور شدن جامعه و تجارت افزایش یافته است. علاقه صنعت به تبدیل داده مخازن نرم افزار به بینش‌های عملی برای شیوه‌های توسعه برتر در حال رشد است. تحلیل‌های به کار برده شده برای توسعه نرم‌افزار مشابه استفاده از آن در بازاریابی برای درک مشتریان محبوب‌تر می‌شود (سان، لی، لیونگ، لی و لی[4]، ۲۰۱۵).

۲- مبانی نظری

فرآیند نرم‌افزار مجموعه‌ای از فعالیت‌های مرتبط که در تولید یک بسته نرم‌افزاری به اوج خود می‌رسد: خصیصه‌سازی، طراحی، پیاده‌سازی، تست، تکامل در نسخه‌های جدید، و نگهداری، همچنین فعالیت‌های پشتیبانی دیگری همچون پیکربندی و مدیریت تغییر، تضمین کیفیت، مدیریت پروژه، ارزیابی تجربه کاربری، غیره. مخازن نرم‌افزاری، زیرساخت‌هایی برای پشتیبانی تمامی این فعالیت‌ها می‌باشد. آن‌ها می‌توانند از چندین سیستم شامل؛ مدیریت تغییر کد، ردیابی اشکال، بررسی کد، سیستم ساخت، انتشار باینری، ویکی‌ها، انجمن‌ها و غیره است (یو[5]، ۲۰۱۲). مخازن‌کاوی نرم‌افزار (MSR) یک زمینه تحقیقات مهندسی نرم‌افزار است که به تحلیل و درک مخازن داده‌ای مرتبط با توسعه نرم‌افزار کمک می‌کند.

1- Mining Software Repository
2- Bugs
3- Vidoni
4- Sun, Li, Leung, Li, & Li
5- Yu

هدف اصلی MSR استفاده هوشمندانه از این مخازن نرم‌افزاری برای پشتیبانی فرآیند تصمیم‌گیری توسعه نرم‌افزار می‌باشد (شانگ، آدامز، و حسان[1]، ۲۰۱۲). حوزه مخازن‌کاوی نرم‌افزار داده در دسترس را برای افزایش بهره‌وری توسعه‌دهندگان و کاهش هزینه‌های پروژه استفاده می‌کند. استفاده از تمام داده‌های موجود، ساختاریافته و غیرساختاریافته فواید را به حداکثر می‌رساند (شانگ و همکاران، ۲۰۱۲). اصطلاح مخازن کاوی نرم‌افزار برای توصیف دسته وسیعی از تحقیقات در بررسی مخازن نرم‌افزاری است. مخازن نرم‌افزاری به مصنوعاتی اطلاق می‌شود که در طول تکامل نرم‌افزار تولید و بایگانی می‌شود. آن‌ها شامل منابعی همانند؛ اطلاعات ذخیره شده در سیستم‌های کنترل نسخه کد منبع (مانند سیستم نسخه‌های همزمان (CVS))، سیستم‌های نیازها/ سیستم های ردیابی اشکال (مانند باگ زیلا)، و بایگانی ارتباطات (مانند ایمیل) می‌باشد. این مخازن اطلاعات ارزشمندی را در خود جای داده است و دید منحصر به فردی از مسیر تکامل واقعی برای تحقق یک سیستم نرم‌افزاری ارائه می‌دهد (وندکرویس و همکاران[2]، ۲۰۰۸).

در تحقیق ویدونی (۲۰۲۲) یک مرور ادبی نظام‌مند از مطالعات MSR به منظور پیروی از راهنما و الگوی پیشنهاد شده توسط Mian و همکاران، انجام شده است. این راهنما به منظور فراهم‌نمودن چارچوب برای مطالعات نظام‌مند MSR بسط داده شده است و مورد تجدید نظر قرار گرفته است. از آنجاکه مطالعات MSR، تحقیق مبتنی بر شواهد می‌باشند، به ندرت فرآیند نظام‌مندی را دنبال می‌کنند. بنابراین ضرورت ایجاد راهنما برای انجام نظام‌مند مطالعات MSR بیش از پیش احساس می‌شود. راهنماهای جدید و یک الگو پیشنهاد شده است که مطالعات مرتبط در حوزه MSR و راهبردها برای مرور ادبی نظام‌مند را یکپارچه می‌کند. برای یک وظیفه نگهداری در حال اقدام، نگهدارنده نیاز به پیاده‌سازی درخواست نگهداری روی سیستم جاری دارد. در مقاله سان و همکاران (۲۰۱۵)، رویکرد MSR4SM برای استخراج اطلاعات غیرمرتبط از هر مخزن نرم‌افزاری مبتنی بر درخواست نگهداری و سیستم جاری پیشنهاد می‌شود. اگر اطلاعات در یک مخزن نرم‌افزاری مرتبط با درخواست نگهداری یا سیستم جاری باشد، این اطلاعات می‌بایست دربردارنده انجام وظیفه جاری نگهداری باشد. MSR4SM مدل موضوعی را برای استخراج موضوع از این مخازن نرم‌افزاری استفاده می‌کند. سپس اطلاعات مرتبط در هر مخزن نرم‌افزاری مبتنی بر موضوع استخراج می‌شود. MSR4SM برای دو وظیفه نگهداری، تحلیل اثر تغییر و ویژگی مکانی که مبتنی بر چهار سیستم موضوعی به نام‌های Rhino، ArgoUML، jEdit،و KOffice هستند. نتایج تجربی نشان می‌دهد کارایی مخازن نرم افزاری سنتی مبتنی بر وظایف نگهداری بوسیله MSR4SM می‌تواند به میزان زیادی بهبود یابد. مقدار زیادی اطلاعات نامرتبط در مخازن نرم‌افزاری وجود دارد. قبل از استفاده از آن‌ها برای پیاده‌سازی یک وظیفه نگهداری در دست اقدام، نیاز به پیش پردازش آن ها وجود دارد. که در نتیجه انجام آن، بهبود کارایی وظایف نگهداری حاصل خواهد شد.

در تحقیق یوو (۲۰۱۲) رویکرد جدید در زمینه برنامه‌نویسی تکاملی مبتنی بر روش یادگیری جمعی ماشین بردار پشتیبان حداقل مربعات وزنی نامتقارن (LSSVM) برای مخازن‌کاوی پیشنهاد می‌شود. در این روش ابتدا مدل وزنی نامتقارن LSSVM پیشنهاد می‌شود و سپس جزییات فرآیند ساخت روش یادگیری جمعی LSSVM وزنی نامتقارن مبتنی بر EP شرح داده می‌شود. مجموعه‌های داده ناقص دو نرم‌افزار در دسترس عموم در نهایت برای تجسم و سنجش کارایی روش یادگیری جمعی LSSVM وزنی نامتقارن مبتنی بر EP استفاده شدند. نتایج آزمایش نشان می‌دهد که روش یادگیری جمعی LSSVM وزنی نامتقارن مبتنی بر EP می‌تواند صحت دسته‌بندی بهتری را در مخزن‌کاوی نرم‌افزار نسبت به روش‌های دسته‌بندی لیست شده در این تحقیق بدست دهد. حوزه کاوش مخازن نرم‌افزاری (MSR) داده مخزن نرم‌افزار را برای کشف دانش و ارزیابی توسعه داده برای هر سیستم پیچیده و در حال رشد تحلیل می‌کند. به هر حال در زمان مطالعه MSR با مقیاس بزرگ، رویکردها و چارچوب‌های موجود برای تحلیل MSR با چالش‌های زیادی مواجه هستند.

1- Shang, Adams, & Hassan
2- Vandecruys & etal

چنین رویکردها و چارچوب‌هایی به ندرت مقیاس خارج از چارچوب دارند. در عوض، آن ها اغلب به ترفند ها و طراحی مقیاس بندی سفارشی نیاز دارند، که این موضوع برای نگهداری هزینه بر است و برای سایر انواع تحلیل قابلیت استفاده مجدد ندارد. در مقاله شانگ و همکاران (۲۰۱۲)، باور بر این است که جامعه وب پیش از این با تعداد زیادی چالش‌های مقیاس‌بندی مهندسی نرم‌افزار همانند این که مجبورند با رشد فزاینده داده وب مقابله کنند. در این تحقیق استفاده از چارچوب مقیاس‌بندی وب به عنوان یک زبان آماده‌سازی داده به هدف مطالعه MSR مقیاس بزرگ گزارش می‌شود. همچنین از طریق سه مطالعه موردی به دقت کاربرد این چارچوب وب برای آماده‌سازی داده (به عبارت دیگر استخراج، انتقال، و بارگذاری، ETL)، به منظور تحلیل بیشتر اعتبارسنجی می‌شود. علی‌رغم محدودیت‌هایی که وجود دارد، هنوز محققین MSR شیوه pig را در مطالعات مقیاس بزرگ توصیه می‌کنند و این به دلیل انعطاف‌پذیری و مقیاس‌پذیری pig می‌باشد. گزارش تجربی در این مقاله به محققین کمک خواهد کرد که مقیاس‌بندی را در تحلیل خود داشته باشند. مدیران نرم‌افزار به طور معمول با پروژه های نرم‌افزاری مواجه هستند، که اشکال‌ها یا ناسازگاری‌هایی را شامل می‌شوند که از حدود هزینه و بودجه تخطی می‌کنند. بوسیله مخزن‌کاوی نرم‌افزار روش‌های جامع داده‌کاوی، مدل‌های پیش‌بینی‌کننده می‌توانند بینشی را به مدیران نرم‌افزاری ارائه دهند که برای رویارویی با مشکلات کیفیت و بودجه از طریق یک روش کارا به آن نیاز دارند.

تحقیق وندکرویس و همکاران (۲۰۰۸) مرتبط با نقش روش دسته‌بندی مبتنی بر بهینه‌سازی کلونی مورچه‌ها (ACO) می‌باشد که در آن Ant Miner++می‌تواند به عنوان یک روش داده‌کاوی جامع برای پیش‌بینی اشکالی ماژول‌های نرم‌افزاری بازی می‌کند. در یک مقایسه تجربی روی سه مجموعه داده عمومی جهان مدل‌های مبتنی بر قانون تولید شده بوسیله Ant Miner++ صحت پیش بینی را نشان می‌دهد که قابل رقابت با سایر روش‌های دسته‌بندی همانند C۴٫۵ ، رگرسیون لجستیک و ماشین بردار پشتیبان می‌باشد. بعلاوه، در این تحقیق در نظر گرفتن جامعیت و شهودی بودن مدل Ant Miner++ به عنوان مدل برتر نسبت به مدل‌های اخیر بحث می‌شود. در سیستم‌های کنترل نسخه همانند Subversion مخازن نرم‌افزاری کل تغییرات نگهداری انجام شده برای سازه کد منبع را در طول تکامل سیستم نرم‌افزاری حفظ می‌کند. داده مستند شده از تاریخچه نسخه به عنوان تثبیت۱ سازمان‌دهی می‌شود. به هر حال، این تثبیت‌ها، برچسبی را برای شناسایی تغییر انجام شده مرتبط با تثبیت حفظ نمی‌کنند.

بنابراین جزییات کمی برای هدایت توسعه دهندگان برای تغییرات مرتبط با نوع خاصی از نگهداری وجود دارد. هدف از تحقیق مقداد، آلهیندواوی، اسکاران، سایفان، و میقدادی۲ (۲۰۱۹) آزمایش تاریخچه نسخه از یک سیستم منبع باز برای دسته‌بندی خودکار تثبیت نسخه در یکی از دو طبقه‌بندی به نام تثبیت‌های تطبیقی و غیرتطبیقی می باشد. در این مقاله تثبیت‌های تاریخچه نسخه از سه سیستم منبع باز، جمع‌آوری می‌شود، سپس ۸ معیار تغییر کد متفاوت مرتبط بدست آورده می‌شود. برای مثال تعداد جملات تغییر داده شده، روش‌ها، و hunkها۳، و فایل‌ها. بر اساس این معیارهای تغییر، در تحقیق، یک رویکرد یادگیری ماشین برای دسته‌بندی تثبیت تطبیقی ایجاد شده است. در نتایج مشاهده می‌شود که معیارهای تغییر می‌تواند فعالیت‌های نگهداری تطبیقی را مشخص کند. همچنین یافته‌های دسته‌بندی نشان می‌دهد که دسته‌بند یادگیری ماشین توسعه داده شده به طور تقریبی ۷۵ درصد صحت پیش‌بینی در تاریخچه‌های تغییر برچسب دارد.

روش پیشنهادی فرآیند آزمایش تاریخچه نسخه از یک سیستم نرم‌افزاری خودکارسازی می‌کند و تثبیت‌ها را برای سیستم‌هایی که مرتبط با یک وظیفه نگهداری تطبیقی هستند، شناسایی می‌کند. ارزیابی روش کارایی و کاربردی بودن آن را پشتیبانی می‌کند. اگر چه ارزیابی دسته‌بند پیشنهادی تاریخچه تغییر برچسب نشده نشان می‌دهد که بهتر از حدس تصادفی از نظر معیار

1- Commit
2- Meqdadi, Alhindawi, Alsakran, Saifan, & Migdadi
3- Hunks : blocks of changes typically found in unified diff patch files, or, more commonly today, found in Git patches

۱F نیست. به نظر می‌رسد که دسته‌بند پیشنهادی به عنوان پایه برتر برای توسعه دسته‌بندی پیشرفته به کار گرفته خواهد شد که توان پیش‌بینی تثبیت‌های تطبیقی را بدون نیاز به تلاش دستی دارا می‌باشد. در تحقیق ووینی، و تلا۲ (۲۰۰۷) تلاش‌های صورت‌گرفته برای تجمیع روش‌های بصری‌سازی۳ اطلاعات در مدیریت پیکربندی برای سیستم‌های نرم‌افزاری توصیف می‌شود و تمرکز برای کمک به مهندس نرم‌افزار در مدیریت ارزیابی سیستم‌های بزرگ و پیچیده بوسیله پیشنهاد روش‌های کارا و موثر برای پرس و جو و ارزیابی ویژگی‌های سیستم با استفاده از روش‌های بصری است. برای رسیدن به این مهم، در این تحقیق چندین روش از حوزه‌های مختلف ترکیب می‌شوند. ابتدا، یک دستورالعمل ساخته می‌شود، که امکان پرس و جوی کلی و داده‌کاوی از انواع مختلف مخازن نرم‌افزاری همانند CVS و Subversion را فراهم می‌کند. با استفاده از این زیرساخت۴، چندین مدل از تکامل کد منبع نرم‌افزار در سطوح متفاوت از جزییات اعم از پروژه و بسته گرفته تا تابع و خط کد ساخته شده است.

دوم، یک مجموعه از دیدها توصیف می‌شود که امکان آزمایش مدل‌های تکامل کد در سطوح متفاوت از جزییات و از جنبه‌های مختلف توصیف می‌شود. سه دید به صورت جزئی بررسی شده است: دید فایل تغییرات در سراسر سطح خطی تعداد زیادی نسخه از یک یا چند فایل را نشان می‌دهد. دید پروژه، تغییرات در سطح فایل را در کل پروژه‌های نرم‌افزاری نشان می‌دهد. دید تجزیه، تغییر در زیرسیستم کل پروژه‌ها را نشان می‌دهد. چگونگی روش‌های پیشنهادی تجسم می‌شود که یک مجموعه ابزار کاملا عملیاتی پیاده سازی می‌شود که برای جواب به سوالات غیربدیهی در چندین پروژه نرم‌افزاری جهان واقعی و اندازه صنعت استفاده می‌شود. کار تحقیق در تقاطع مهندسی نرم‌افزار (SE) کاربردی و بصری‌سازی اطلاعات می‌باشد که هدف از مجموعه ابزار تحقیق مذکور، تجمیع دقیق روش‌های ارتقا بوسیله حوزه InfoVis در زمینه ۵SE می‌باشد. مولفه زبان نقش مهمی را در بازیابی داده/اطلاعات بازی می‌کند. بازیابی داده اغلب برای دریافت داده از نرم‌افزار تجاری با مشکل مواجه می‌شود. پیدایش مخزن منبع باز در جمع‌آوری داده نرم‌افزاری سهم فوق العاده داشته است. نکات برجسته تحقیق روش بازیابی داده برای نرم‌افزارکاوی از مخزن نرم‌افزار منبع باز گسترده، SourceForge می‌باشد. به منظور خودکارسازی بازیابی داده از مخزن، یک تجزیه‌کننده۶ با استفاده از زبان برنامه‌نویسی پایتون نوشته شده که مبتنی بر الگوریتم تطبیق الگو است. داده بازیابی شده بعدا برای تخمین کیفیت نرم‌افزار منبع باز استفاده می‌شود (باکار۷، ۲۰۱۱). در تحقیق یک مدل مبتنی بر گراف ناهمگن جدید برای ضبط و رسیدگی به همه اطلاعات پیچیده و با همبستگی قوی از یک شبکه اجتماعی توسعه‌دهنده نرم‌افزار (DSN) برای پشتیبانی چندین وظیفه تحلیلی پیشنهاد شده است. بویژه، مسئله چالشی استخراج خودکار جوامعی از توسعه‌دهندگان نرم‌افزار که با تکیه بر یافته‌های تحلیل شبکه اجتماعی (SNA)، علایقی را برای پروژه‌های مشابه به اشتراک می‌گذارند.

برای غلبه بر موضوع حجم انبوه گراف، روش‌های مختلف تعبیه گراف استفاده شده است. در تحقیق رویکرد پیشنهادی با جنبه نوآوری از نقطه نظر کارایی و موثر بودن با انجام آزمایش‌ها و با استفاده از مجموعه داده GitHub با سایر رویکردها مقایسه شده است (دلوکا و همکاران۸، ۲۰۲۳). بیشترین تمرکز محققین مخازن نرم‌افزاری، روی تحلیل معایب و فرآیندهای توسعه در رابطه با برنامه‌های منبع باز می‌باشد. بنابراین یک فضای خالی بین پروژه‌های نرم‌افزاری صنعتی و منبع باز وجود دارد که بویژه مرتبط با طرح‌های۹ متفاوتی برای ایجاد مخازن نرم‌افزاری و

1- F-measure
2- Voinea, & Telea
3- Visualization
4- Infrastructure
5- Software Engineering
6- Parser
7- Bakar
8- De Luca, Fasolino, Ferraro, Moscato, Sperlí, & Tramontana
9- Schemes

توسعه طرح ها می‌باشد و این مسئله به طور خاص برای سیستم‌های تعبیه شده، صحت دارد، به طوری‌که بازارهای بزرگ را بدست می‌آورند و پیچیده‌تر می‌شوند، هدف از تحقیق پلازک، و ساسنوسکی[1] (۲۰۲۱) کشف مخازن نرم‌افزاری سیستم‌های تعبیه شده صنعتی و حصول ویژگی‌های مشخصه به منظور ارزیابی کیفیت و شناسایی مشکلات مربوط به کار با فرآیندهای توسعه می‌باشد. روش کار در این تحقیق رویکردی جدید برای تحلیل مخازن نرم‌افزاری مبتنی بر پنج دستاورد جدید از موضوع مخازن کنترل کد و ردیابی می‌باشد. در این پروژه فعالیت‌های متنوعی همانند ایجاد توابع جدید، رفع اشکال، بهبود کارایی، اصلاح تست‌ها) و تحلیل آن‌ها در متن، نه فقط در تک پروژه، بلکه در یک مجموعه از پروژه‌های مرتبط توسعه داده شده در شرکت توسط عوامل پروژه انجام می‌شود.

این موضوعات در تحقیق‌های قبلی نادیده گرفته شده است. این تحلیل‌ها به رویکرد کلی‌نگر جدید برای مخزن‌کاوی شامل معیارهای آماری، متن کاوی، و تکنیک‌های یادگیری ماشین نیاز دارد. در جستجوی پروژه‌های صنعتی منتخب ۴۰-۷۵٪ از موضوعات مربوط به معایب است؛ گزارش‌های موضوعی و شرح تعهدات شامل مقدار زیادی داده می‌باشد که در تحقیق‌های قبلی نادیده گرفته شده‌اند. این داده‌ها یافتن انواع گوناگونی از تغییر کد را تسهیل می‌کند و به شناسایی نقایص در مخازن نرم‌افزاری کمک می‌کند. در نتیجه تحلیل مخزن در این تحقیق یک دید وسیع و کامل از پروژه بدست می‌آید که به توسعه پروژه منتهی می‌شود. معماری ریزسرویس‌ها یک الگوی سرویس‌گرای ضروری است که به طور وسیعی در صنعت برای توسعه و استقرار سیستم‌های نرم‌افزاری مقیاس‌پذیر استفاده شده است. ایده‌ی برجسته‌ی تحقیق آسانکائو، کراگر، ماسر، وسلائونی ۲ (۲۰۲۳) طراحی سرویس های خیلی مستقل است که واحدهای کوچکی از عملکرد هستند و می توانند با همدیگر از طریق واسط‌های سبک‌وزن تعامل کنند. هدف از این تحقیق بررسی چالش‌های طراحی و نگهداری ریزسرویس‌ها و یافتن جواب برای وجود استقلال عملی در ریز سرویس‌ها است. بنابراین درک چگونگی و چرایی چرخه عمر ریزسرویس‌ها، برای نمونه در حوزه بازسازی و بهبود معماری یک سیستم یا برای توسعه ابزار پشتیبانی، بسیار مهم می‌باشد. به منظور بهبود دید در چگونگی ریزسرویس‌ها، یک مطالعه تجربی در مقیاس بزرگ روی فرگشتی[3] ریزسرویس‌های در ۱۱ سیستم منبع باز شامل ۷۳۱۹ تحلیل کمی و کیفی، انجام شده است. نتایج کمی نشان می‌دهد که الگوهای بازگشتی از فرگشتی تمام سیستم‌ها، برای نمونه تثبیت‌های «تشریح انفجاری[4]» و ریزسرویس‌ها که به میزان زیادی مستقل هستند، در تاپل‌ها یا تقریبا تمامی تغییرات تکامل می‌یابند. نتایج به وسیله تحلیل تثبیت‌های تکامل سرویس به صورت کیفی برای استخراج مستقل ریزسرویس‌ها معرفی می‌شود و تکامل خاص آن‌ها آن ها را به دست می‌دهد. انجام این تحقیق درک متخصصین و محققین در مورد چگونگی و بهبود ریزسرویس ها و تکامل می‌بخشد. تحقیق مخزن‌کاوی نرم‌افزار و تحلیل داده از چندین مخزن نرم‌افزار برای فهم تاریخچه توسعه از سیستم‌های نرم‌افزاری و برای پیشنهاد روش‌های برتر برای تکامل چنین سیستم‌هایی در آینده انجام می‌شود. گومین و مینس[5] (۲۰۱۳) در مطالعه‌ای به بررسی فعالیت‌ها و تعامل‌های بین افراد مشارکت‌کننده در فرآیند توسعه نرم‌افزار پرداختند. چالش اصلی در چنین موقعیت‌هایی قابلیت تعیین و احراز هویت (حساب‌های ورودی[6] یا ایمیل) در مخازن نرم‌افزاری می‌باشد که فرد فیزیکی یکسانی را نمایش می‌دهد.

برای دستیابی به این مهم، الگوریتم‌های ادغام احراز هویت مختلفی در گذشته پیشنهاد شده است. این تحقیق مقایسه موضوعی از الگوریتم‌های ادغام احراز هویت شامل برخی از بهبودها روی الگوریتم‌های موجود فراهم می‌کند. نتایج روی یک انتخاب بزرگ از پروژه‌های نرم‌افزاری منبع باز در حال انجام، اعتبارسنجی می‌شود.

1- Polaczek, & Sosnowski
2- Assunção, Krüger, Mosser, & Selaoui
3- Co-evolution
4- shotgun surgery
5- Goeminne, & Mens
6- Login

اشکال‌های نرم‌افزاری اجتناب ناپذیر هستند. پروژه‌های نرم‌افزاری بزرگ اغلب سیستم‌های ردیابی اشکال را به منظور جمع‌آوری، سازمان‌دهی، و نگهداری مسیر اشکال‌های گزارش‌شده، استفاده می‌کنند. موضوع اصلی کاربرد سیستم‌های ردیابی اشکال برای سرعت بخشیدن به فرآیند رفع اشکال می‌باشد که به میزان زیادی کیفیت کلی محصول نرم افزاری را بهبود می‌بخشد. به هر حال مدیریت سیستم‌های ردیابی اشکال هزینه و تلاش‌های مضاعف را متحمل می‌شود. از این رو نظارت بر کارایی فرآیند رفع اشکال مورد پیگیری در سیستم‌های ردیابی اشکال حائز اهمیت می‌باشد. هدف تحقیق گویال و ساردانا[1] (۲۰۲۰) شناسایی مشخصه‌های کارایی فرآیند رفع اشکال در مخازن اشکالی منبع باز می‌باشد. معیارهای کارایی مختلف برای برآورد معیار کیفیت رفع اشکال به کار برده شده‌اند. برای ارزیابی آزمایش، گزارش‌های اشکال از مخزن Bugzila از مرورگر فایرفاکس، NetBeans، Eclipse و پروژه‌های باز آفیس جمع‌آوری می‌شود. طبق گزارش‌های دریافتی ۴۵٫۶۷٪ اشکال‌ها مربوط به وضوح ثابت[2]، می‌باشد.

مابقی ۵۴٫۳۳٪ گزارش‌های اشکال‌ها مربوط به مخازن اشکال‌های رفع نشده می‌باشد که منجر به جمع‌شدن کارهای اضافی برای توسعه‌دهنده‌های نرم‌افزاری می‌شود. بعلاوه، علل ریشه‌ای مشخص می‌کند، که نگهداری بخش بزرگی از گزارش‌های حجیم با تکیه بر مخازن اشکالی باز شده است و اشکال‌های غیرقابل تکرار بیشترین فاکتور تعیین‌کننده در فرآیند رفع اشکال می‌باشند و روی کارایی رفع اشکال تاثیر منفی دارد. تحلیل علل ریشه‌ای یک روش داده محور برای بهبود فرآیند توسعه نرم افزار در سازمان‌های نرم‌افزاری بالغ است. جستجوی فرآیندهای منفرد با شدت بالای نقایص که تحلیل تفصیلی درج معایب (DICA) نامیده می‌شود و بطور بالقوه یک رویکرد موثر و کم هزینه هنگام اقدام برای تحلیل علل ریشه‌ای نقایص کلی استفاده می‌شود. در DICA، داده از مخازن موجود (بایگانی نسخه، ردیاب اشکال) به میزان زیادی به طور خودکار به منظور تعیین شرایط (همانند افراد، نقش‌ها، مولفه‌ها یا دوره زمانی) که به نقایص فوق عادی درج مکرر وابسته است. با این وجود، هیچ گزارشی از صنعت مبنی بر استفاده از DICA منتشر نشده است. هدف علت‌یابی عدم استفاده از DICA توسط متخصصین در اغلب موارد می‌باشد. روش استفاده از مطالعه موردی، تک‌موردی، مطالعه موارد شایع، مطالعه مکاشفه‌ای برای ارزیابی شش علت محتمل به طور موازی (R1 تا R6) می‌باشد. مورد مبتنی بر داده ۱۱ ساله مخزن از یک شرکت نرم افزاری بالغ اما کوچک می‌باشد.

این شرکت محصول را در دامنه سیستم مدیریت محتوای سطح بالا می‌سازد و تلاش ۴ ماه - نفر را برای استفاده از این داده‌ها، توصیف می‌کند. در حالی که DICA تلاش غیرقابل اغماض (R3) و درجه‌ای از خلاقیت (R2) نیاز داشت، مهم‌ترین مانع عدم اطمینان کافی به نتایج (R6) همراه با دشواری ارزیابی از این قابلیت (R5) می‌باشد. در تحقیق سه مشکل شناسایی می‌شود که به این نتیجه منتهی می‌شود و نتیجه عبارتست از این که روش‌های مخزن‌کاوی رایج برای موفقیت DICA خیلی نابالغ است. بهبود تدریجی برای کمک بعید به نظر می‌رسد. اصول متفاوتی از عملیات نیاز خواهد شد. حتی با چنین روش‌های متفاوتی، مسئله کیفیت داده ورودی همچنان ادامه دارد و دستیابی به نتایج مطلوب را دشوار می‌سازد (پریچلت و پیپر[3]، ۲۰۱۴) . الگوهای طراحی راه‌حل‌های شناخته شده هستند.، با توجه به کیفیت نرم‌افزار مزایای فراوانی دارند. به هر حال، تلاش‌های علمی برای جمع‌آوری اطلاعات در پروژه‌های نرم‌افزاری به منظور استفاده از الگوهای طراحی وجود ندارد. در تحقیقی یک مخزن وب از نمونه‌های الگوی طراحی معرفی می‌کند که در پروژه‌های منبع باز به کار برده می‌شوند. سودمندی چنین مخزنی بر فراهم نمودن یک دانش پایه متکی است، به طوری که توسعه دهنده‌ها می‌توانند مولفه‌های قابل استفاده مجدد را شناسایی کنند و محققین می‌توانند یک مجموعه داده کاوش شده را بیابند.

در حال حاضر، ۱۴۱ پروژه منبع باز در نظر گرفته شده‌اند و بیشتر از ۴۵۰۰ نمونه الگوی طراحی یافته شده و در پایگاه

1- Goyal, & Sardana
2- Fixed resolution
3- Prechelt, & Pepper

داده مخزن ثبت شده‌اند. ارزیابی مخزن از نقطه نظر علمی و عملی انجام شده است، که نتایج پیشنهاد می‌کند که مخزن می‌تواند برای کاربران با تجربه و بی تجربه مفید باشد. به هر حال، فواید استفاده از مخزن برای کاربران بی تجربه بیشتر قابل توجه است (امپادزوگلو، میچو، و استاملس[1]، ۲۰۱۳). معرفی اینترنت نه تنها برای جامعه، بلکه برای صنعت نرم‌افزار نیز از طریق اشتراک اطلاعات و دانش تحول به همراه داشته است به طوری که بخش مرکزی از فرآیند توسعه نرم افزار به شمار می‌رود. در نتیجه جهانی‌شدن صنعت نرم‌افزار نه تنها بر قابلیت استفاده مجدد نرم‌افزار افزوده می‌شود، بلکه همچنین چالش‌های جدید معرفی می‌شوند. در میان چالش‌ها، مهمترین آن برخاسته از اشتراک‌گذاری دانش، یعنی امنیت اطلاعات می‌باشد. امنیت یک رفتار اصولی برای جامعه توسعه نرم‌افزار می‌باشد از این رو، نه تنها منبع کد بلکه آسیب‌پذیری‌ها نیز از طریق مرزهای پروژه به اشتراک گذاشته می‌شوند. توسعه‌دهنده‌ها به چنین آسیب‌پذیری‌ها در پروژه‌هایشان، اغلب تا زمانی که این آسیب‌پذیری‌ها توسط هکرها مورد بهره‌برداری قرار گیرند یا توسط پایگاه‌داده‌های مستقل مشاوران امنیتی در دسترس عموم قرار گیرند، آگاهی نمی‌یابند. در تحقیقی دیگر یک رویکرد مدلسازی ارائه می‌شود که مزایای فناوری‌های وب معنایی به منظور ایجاد لینک‌های قابل مسیریابی بین مخازن مشاوران امنیتی و سایر مخازن نرم‌افزاری در بر می‌گیرد.

بویژه، در تحقیق یک بازنمایی هستان‌شناسی یکپارچه ایجاد می‌کند که لینک‌های مسیریابی دو جهته بین دانش جمع‌آوری شده در مخازن ساخت نرم‌افزار و پایگاه داده تخصصی آسیب‌پذیری را پشتیبانی می‌کند. این مخازن می‌تواند به عنوان مخازن اطلاعات در نظر گرفته شوند که به طور معمول به سایر منابع همانند مخازن منبع کد شامل نمونه‌های گزارش شده از مسائل گفته شده، لینک مستقیمی ندارد. نوآوری آن، غلبه بر مسئله مخازن قدیمی اطلاعات و انتقال آن‌ها به مرکز اطلاعات قدیمی می‌باشد، به طوری که اشتراک دانش از طریق مرزهای مخازن را توسعه می‌دهد. در این تحقیق چندین آزمایش قابلیت رویکرد پیشنهادی بوسیله مسیریابی آسیب‌پذیری‌های موجود برای پروژه را نشان می‌دهد که ممکن است به طور مستقیم یا غیرمستقیم بوسیله آسیب‌پذیری‌های ارث‌بری شده از سایر پروژه ها و کتابخانه ها تحت تاثیر قرارگیرد (القحطانی، اقان، وریلینگ[2]، ۲۰۱۶). مخازن کنترل منبع، مخازن اشکال، ارتباطات بایگانی‌شده، توسعه ثبت‌های وقایع، و مخازن کد مثال‌هایی از مخازن نرم‌افزاری هستند که معمولا برای بیشتر پروژه‌های نرم‌افزاری موجود هستند.

زمینه مخزن‌کاوی نرم‌افزار (MSR) داده غنی موجود در این مخازن را برای اطلاعات کاربردی و جالب در مورد سیستم‌های نرم‌افزاری بدست می‌دهد. بوسیله تبدیل این مخازن از مخازن ثبت - نگهداری به مخازن فعال تحلیل و کشف می‌کند. تحقیق فرآیندهای تصمیم را در پروژه‌های نرم‌افزاری مدرن هدایت می‌کند. مخازن نرم‌افزاری شامل اطلاعات ارزشمند و تاریخی در مورد توسعه کلی سیستم‌های نرم افزاری می‌باشد. بوسیله انتقال این مخازن از یک مخزن ثبت ایستا به مخازن فعال می‌توان فرآیند تصمیم را به پروژه‌های نرم‌افزار مدرن هدایت کرد. داده در مخازن کنترل منبع به طور سنتی برای بایگانی کد استفاده می‌شود و می‌تواند با داده در مخزن اشکال برای کمک به متخصصین پیوند داده شود و بدین وسیله تغییرات پیچیده منتشر می‌شود و به آن‌ها در مورد مخاطرات کد مبتنی بر اشکال‌ها و تغییرات قبلی هشدار می‌دهد. در مطالعه‌ای یک تاریخچه‌ی خلاصه از زمینه MSR ارائه می‌شود و چندین بهبود و نتایج اخیر با استفاده از روش‌های MSR برای پشتیبانی تحقیق و کاربرد نرم افزار بحث می‌شود. سپس در مورد فرصت‌ها و چالش‌های متنوع پیش روی این حوزه مهم و نوظهور بحث می‌شود (اسپادین، آنچی، و باچیل[3]، ۲۰۱۸). مخزن‌کاوی نرم‌افزار (MSR) امروزه به عنوان یکی از جذاب‌ترین زمینه‌های رشد در مهندسی نرم‌افزار در نظر گرفته می‌شود. MSR روی استخراج و تحلیل داده موجود در مخازن نرم‌افزاری به منظور دسترسی به اطلاعات عملی، و مفید و جذاب در

1- Ampatzoglou, Michou, & Stamelos
2- Alqahtani, Eghan, & Rilling
3- Spadini, Aniche, & Bacchelli

مورد سیستم متمرکز است. اگر چه MSR نقش مهمی را در مهندسی نرم‌افزار بازی می‌کند، تعداد کمی ابزار به طور عموم برای پشتیبانی توسعه دهنده ها برای استخراج اطلاعات از مخازن Git ساخته و پرداخته شده است. در این تحقیق یک چارچوب پایتون به عنوان PYDRILLER ارائه می‌شود که فرآیند کاوش Git را تسهیل می‌کند.

همچنین ابزار ارائه شده با نوآوری پایتون به عنوان چارچوب GitPython مقایسه می‌شود. نتایج نشان می‌دهد که PYDRILLER همان خروجی را به طور متوسط با ۵۰ درصد LOC کمتر و به طور معناداری پیچیدگی کمتری ارائه می‌دهد. در تحقیق دیگری، مقاله‌های دربردارنده تحلیل آزمایشی از پروژه‌های نرم‌افزاری مرتبط با داده کاوی در مهندسی نرم‌افزار انجام می‌شود. مجموعه‌های داده، تکنیک‌ها و ابزار به کار گرفته شده/توسعه داده شده/پیشنهاد شده در این مقالات شناسایی می‌شود. بیشتر از نیمی از مقالات دربردارنده کارهای انجام شده در زمینه ایجاد یا استفاده از ابزار داده کاوی برای مخزن‌کاوی نرم‌افزار می‌باشد. نتایج تحلیل مقالات مشخص می‌کند که نویسندگان MSR به طور کلی داده‌های خام در دسترس عموم را پردازش می‌کنند. در این تحقیق، ابزارهای متنوع استفاده شده در MSR مبتنی بر ابزارهای داده‌کاوی قدیمی، جدید توسعه داده شده، نمونه اولیه توسعه داده شده و اسکریپت‌ها دسته‌بندی می‌شود. در تحقیق نوع کاوش و مجموعه داده‌های استفاده شده، نشان داده می‌شود (چاتوید، سینگ و سینگه[1]، ۲۰۱۳). محققین مخزن‌کاوی نرم‌افزار (MSR) را برای مطالعه مهندسی نرم‌افزار به صورت تجربی و بوسیله تحلیل سازه همانند منبع کد، سیستم‌های کنترل نسخه، فراداده و غیره استفاده می‌کنند. به هر حال، برای ایجاد یک مطالعه با استفاده از این تکنیک‌ها، محققین معمولا زمانی را برای جمع‌آوری داده و توسعه یک زیرساخت پیچیده صرف می‌کنند که مستلزم فضای دیسک و زمان پردازش می‌باشد. در تحقیقی برنامه‌کاربردی تحت وب[2] به نام MetricMiner ارائه می‌شود که هدف آن کمک به پشتیبانی محققین در برخی مراحل مخزن‌کاوی نرم‌افزار همانند محاسبه معیار، استخراج داده، استنتاج آماری می‌باشد. این ابزار همچنین دربردارنده آماده‌سازی داده برای تحلیل، ذخیره زمان، و منابع محاسباتی می‌باشد (سوکل، آنیچ، و جروسا[3]، ۲۰۱۳). در تحقیق دیگری یک رویکرد برای بازیابی/کشف لینک‌های قابل ردیابی بین سازه نرم‌افزاری از طریق آزمایش تاریخچه نسخه سیستم نرم‌افزاری ارائه می‌شود. یک رویکرد مبتنی بر اکتشاف که الگوکاوی ترتیبی را به منظور تثبیت در مخزن‌کاوی نرم افزار برای کشف مجموعه‌هایی از سازه‌ها مکرر هم تغییر (یعنی منبع کد و مستندات) استفاده می‌کند. اگر انواع مختلفی از فایل‌ها به کرار با هم دیگر تثبیت شوند، احتمال وجود لینک قابل مسیریابی بین آن‌ها وجود خواهد داشت. رویکرد با تعدادی نسخه منبع باز سیستم KDE ارزیابی می‌شود.

به عنوان یک مرحله اعتبارسنجی، لینک‌های کشف شده برای پیش‌بینی تغییرات مشابه در نسخه‌های جدید سیستم یکسان استفاده می‌شوند. نتایج دقت پیش بینی بالا انواع اصلی لینک‌های مسیریابی را نشان می‌دهد (کاجی، مالتیک، و شریف[4]، ۲۰۰۷). مخازن نرم‌افزاری همانند منبع کد، بایگانی ایمیل، و پایگاه داده‌های اشکال، شامل متن برچسب نشده، و بدون ساختار می‌باشند که تحلیل آن‌ها به روش سنتی مشکل می‌باشد. پیشنهاد یکی از تحقیق‌ها، استفاده از مدل موضوعی برای کشف خودکار ساختار در مخازن متنی می‌باشد. ساختار کشف شده پتانسیل کاربرد در وظایف مهندسی نرم‌افزار همانند پیش‌بینی اشکال و کشف لینک مسیریابی را دارا می‌باشد. هدف تحقیق آدرس‌دهی چالش‌های کاربرد مدل‌های موضوعی برای مخازن نرم‌افزاری می‌باشد (توماس[5]، ۲۰۱۱). علی‌رغم تعداد زیادی از مخازن نرم‌افزاری در حال استفاده، دسته‌بندی ابزارهای متنوع و مقایسه آن‌ها حوزه تحقیقی نسبتا جدیدی است. در تحقیق یک تحلیل مقایسه‌ای از ابزارهای مختلف برای MSR مبتنی بر برخی معیارهای جدید و در دسترس ارائه می‌شود. این مقایسه به تعیین ابزاری که بهترین عملیات را برای نوعی از برنامه کاربردی داده

1- Chaturvedi, Sing, & Singh
2- Web application
3- Sokol, Aniche, & Gerosa
4- Kagdi, Maletic, & Sharif
5- Thomas

شده و استفاده آن به طور مستقیم و بدون نیاز به رویکرد آزمایش و خطا کمک خواهد کرد. تحقیق چندین هدف را دنبال می‌کند: الف) به عنوان یک مکانیسم تکوینی برای طراح ابزار عمل می‌کند (ادراک و مقایسه ابزارهای متفاوت) عمل می‌کند. ب) به عنوان یک ابزار ارزیابی برای کاربران بالقوه ابزار (برای مرور ساده در نمودار تحلیل مقایسه‌ای را به منظور فهم در یک نگاه و به منظور گنجاندن مولفه‌های ضروری در ابزار مورد نظر) و به عنوان یک نقطه عطف در مقایسه می‌باشد به طوری که محققین ابزار MSR به سادگی می‌توانند آن را از میان سایر ابزارها تمیز دهند، بنابراین مسیر جدیدی را برای محققین ارائه می‌دهد و نیز با ارائه جدولی یک شاخص سریع برای خواننده و تحلیل سریعی برای ابزار مورد نظر فراهم می‌کند (اولاتینگ، ادریس، القمدی، و القمدی۱، ۲۰۱۰). یکی از تصمیم‌های محققین در مواجهه با تحلیل ارزیابی سیستم‌های نرم‌افزاری انتخاب یک فرمت از تحلیل/ تعویض داده می‌باشد.

بیشتر فرمت های تعویض ناگزیر با برنامه‌های ویژه نوشته شده به طور خاص برای این هدف پردازش می‌شوند و به طور سادگی قابل گسترش نیستند. بنابراین اکثر دانشمندان پایگاه داده‌های خویش را استفاده می‌کنند که تکرار نوشتن برنامه‌های import/export را برای فرمت موردنظر به دنبال دارد. در یک تحقیق، EvoOnt را به عنوان یک مخزن نرم‌افزاری تعویض فرمت داده مبتنی بر زبان هستان‌شناسی وب (OWL) ارائه می‌دهد. از این رو EvoOnt، معنایی از داده را بدین صورت توصیف می‌کند، ۱) EvoOnt گسترش‌پذیری آسان، ۲) همراه با تعداد زیادی ابزار در دسترس ۳) اجازه استخراج ادعاها را از طریق قابلیت‌های استدلال منطقی ذاتی آن می‌دهد. همچنین iSPARQL را ارائه می‌دهد که SPARQL مبتنی بر موتور پرس و جوی وب معنایی شامل الحاق شباهت را نشان می‌دهد. EvoOnt و SPARQL با یکدیگر می‌توانند تعدادی وظایف قابل توجه در پروژه های مخزن‌کاوی نرم‌افزار ، همانند ارزیابی مقادیر تغییر میان نسخه‌های تشخیص بوهای کد بد به دنبال دارد. برای به تصویر کشیدن فواید EvoOn (و iSPARQL) یک سری آزمایش با پروژه‌های جهان واقعی جاوا انجام می‌شود. این مسئله نشان می‌دهد که یک تعداد از تحلیل‌های نرم‌افزار می‌توانند برای پرس و جوهای iSPARQL روی یک پایگاه داده EvoOnt کاهش داده شوند (کیفر، بیمستین، وتاپلت۲، ۲۰۰۷). مخازن نرم‌افزاری مقادیر زیادی داده دربردارنده تغییرات نرم‌افزاری در طی تکامل آن فراهم می‌کنند. این مخازن می‌توانند به طور موثری برای استخراج و تحلیل اطلاعات مرتبط استفاده شود و نتایج مربوط به تاریخچه نرم‌افزار یا تصویر جاری آن را استخراج کند. هدف از تحقیق برآورد مطالعات اخیر روی رویکردهای مخازن‌کاوی نرم‌افزار (MSR) جمع‌آوری شواهد در مورد اهداف تحلیل نرم افزار می‌باشد (هدف، تمرکز، موضوع تحلیل)، منابع داده، روش‌های ارزیابی، ابزار، حوزه‌ی دربردارنده. در تحقیق یک مطالعه نگاشت سیستمی برای شناسایی و تحلیل تحقیق در مورد مخازن‌کاوی نرم‌افزار با استفاده از تحلیل پنج ویرایش[3] کنفرانس‌کاری حوزه مخازن‌کاوی نرم‌افزار - کنفرانس اصلی در این زمینه، انجام شده است. رویکردهای مخازن‌کاوی نرم‌افزار برای کاربردهای مختلفی و به طور اصلی برای فهم معایب، تحلیل مشارکت و رفتار توسعه‌دهنده و فهم تکامل نرم‌افزار به کار برده می‌شوند. از جانب دیگر، مقداری شکاف با توجه به اهداف، تمرکز و نوع منبع داده (به عبارت دیگر عدم کاربرد مولفه برای شناسایی بوها، بازسازی، و موضوعات کیفیت نرم افزار) آن‌ها شناسایی شدند.

در مورد روش ارزیابی تحلیل به ارزشمند برخی انواع ارزیابی تجربی اشاره می‌کند. مطالعات MSR بر اهداف مختلفی تمرکز دارد و تعداد زیادی فرصت تحقیقاتی برای کشف وجود دارد و موضوعات مرتبط با MSR باید در نظر گرفته شود (فاریاس و همکاران۴، ۲۰۱۶). در تحقیقی دیگر، رویکرد جدیدی برای کاوش فرآیند داده‌ی (فرآیند کاوش) مخازن نرم‌افزاری از بایگانی داده تولیدی به عنوان نتیجه‌ی ساخت نرم‌افزار توسط تیم دانشجویی در یک محیط آموزشی و نیز یک برنامه کاربردی از کاوش سه مخزن نرم افزاری : تیم wiki (مورد استفاده در طی

1- Olatunji, Idrees, Al-Ghamdi, & Al-Ghamdi
2- Kiefer, Bernstein, & Tappolet
3- Edition
4- Farias & etal

مهندسی نیازمندی‌ها)، سیستم کنترل نسخه (توسعه و نگهداری)، و سیستم ردیابی اشکال (نگهداری اصلاحی و تطبیقی) در چارچوب یک دوره کارشناسی مهندسی نرم‌افزار ارائه می‌شود. در این مقاله تجسم، معیارها، و الگوریتم‌ها برای ارائه یک دید در رویه‌ها و شیوه‌ها در طول فازهای مختلف از چرخه عمر توسعه نرم‌افزار دنبال می‌شوند. تجسم و معیارهای پیشنهادی (تحلیل‌های یادگیری) یک دید چندوجهی برای سازنده ابزار کاربردی بازخورد دانشجویان، در فرآیند توسعه و کیفیت می‌باشد. همچنین رویدادهای ثبت وقایع تولیدی بوسیله مخازن نرم‌افزاری کاوش می‌شود و دیدها همانند درجه مشارکت فردی در یک تیم، کیفیت تثبیت پیام‌ها، شدت و سازگاری تثبیت فعالیت‌ها، فرآیند رفع اشکال و کیفیت، مولفه، آنتروپی توسعه‌دهنده و تطابق فرآیند و صحت سنجی به دست می‌آید. علاوه بر آن، یک تحلیل تجربی روی مجموعه داده مخازن نرم افزاری شامل ۱۹ تیم، هر کدام از تیم‌ها شامل ۵ عضو و بحث چالش‌ها، محدودیت‌ها و توصیه‌ها ارائه می‌شود (میتال و سورکا[1]، ۲۰۱۴). تحلیل موثر شناسایی محصولات کار تحت تاثیر بوسیله درخواست تغییر پیشنهادی، رفع اشکال یا درخواست ویژگی جدید می‌باشد. در تعداد زیادی از پروژه‌های منبع باز همانند KDE، گنوم، موزیلا، openoffice، درخواست‌های تغییر، و داده‌ی مرتبط، و ذخیره شده در یک سیستم ردیابی رفع اشکال همانند باگ‌زیلا[2] ذخیره می‌شوند. این داده‌ها به همراه سایر داده‌ها در یک سیستم نسخه‌سازی همانند CVS منبع ارزشمندی از اطلاعات روی تحلیل‌های مفید می‌باشد. در تحقیق روشی برای استخراج مجموعه‌ای از فایل‌های منبع تحت تاثیر بوسیله درخواست تغییر پیشنهادی می‌باشد.

این روش الگوریتم‌های بازیابی اطلاعات را به شرح تغییر درخواست ارتباط می‌دهد و مجموعه‌ای از بازدیدهای فایل منبع تاریخچه بوسیله درخواست های تغییر مشابه قبلی تحت تاثیر قرار داده می‌شود. روش بوسیله کاربرد آن در پروژه‌های منبع باز ارزیابی می‌شود (کانفورا و کرولا[3]، ۲۰۰۵). اطلاعات برنامه‌نویس به منظور تحلیل کیفیت نرم‌افزار، انجام جرم‌شناسی نرم‌افزار[4] و بهبود نگهداری نرم‌افزار می‌باشد. به هر حال ابزارهای موجود، آخرین توسعه‌دهنده که یک خط از کد را تغییر می‌دهد، بدون توجه به تغییرات قبلی، برنامه‌نویس آن در نظر می‌گیرد. این تقریب سبب از دست دادن اطلاعات مهم می‌شود. در تحقیق دو مدل مولف سطح - خط جدید برای غلبه به این محدودیت ارائه می‌دهد. در تحقیق ابتدا گراف مخزن به عنوان یک گراف انتزاعی برای مخزن کد توصیف می‌شود، بدین صورت که گره‌ها ثابت هستند و لبه‌ها وابستگی‌ها را نمایش می‌دهند. سپس برای هر خط از کد، مولف ساختاری به عنوان یک زیرگراف از گراف مخزن تعریف می‌شود و همه تثبیت‌های مربوط به خط تغییر داده شده و وابستگی‌های توسعه بین تثبیت‌ها را ثبت می‌کند. مولفه‌ی وزن‌دار به عنوان یک بردار از وزن‌های مشارکت مولف استخراج شده از مولف ساختاری خط و مبتنی بر یک معیار تغییر کد بین تثبیت‌ها، برای مثال بهترین فاصله ویرایش، تعریف می‌شود. در تحقیق دو مدل مولف به عنوان یک ابزار گیت-مولف[5] درونی گیت[6] پیاده‌سازی شده است. گیت - مولف در یک مطالعه تجربی و یک مطالعه مقایسه‌ای ارزیابی می‌شود. در مطالعه تجربی، گیت-مولف روی پنج پروژه منبع باز اجرا می‌شود و در نتیجه گیت-مولف می‌تواند بیشترین اطلاعات را از یک ابزار موجود (git-blame) برای حدود ۱۰ درصد خطوط بازیابی کند. در مطالعه مقایسه‌ای گیت - مولف برای ساخت یک مدل خط - سطح با یک مدل نمایشی فایل - سطح مقایسه می‌شود. نتایج نشان می‌دهد که مدل خط - سطح سازگاری بهتری نسبت به مدل فایل- سطح در ارزیابی مجموعه داده استفاده شده از پروژه سرور HTTP آپاچی نشان می‌دهد (منگ، میللر، ویلیامز، و برنات[7]، ۲۰۱۳).

1- Mittal, & Sureka
2- Bogzila
3- Canfora, & Cerulo
4- Software Forensics
5- git-author
6- git
7- Meng, Miller, Williams, & Bernat

کاوش مخازن نرم‌افزاری استخراج دو اطلاعات پایه و ارزش‌افزوده از مخازن نرم‌افزاری موجود را شامل می‌شود.

مخازن برای استخراج حقایق توسط ذینفعان مختلف (همانند محققین، مدیران) و برای اهداف متنوع کاوش خواهد شد. برای اجتناب از پیش‌پردازش غیرضروری و مراحل تحلیل، اشتراک و یکپارچه‌سازی دو واقعیت پایه و ارزش افزوده نیاز هستند. در تحقیقی، SeCold یک چارچوب باز و مشارکتی برای به اشتراک‌گذاری مجموعه داده‌های نرم‌افزار معرفی می‌شود. SeCold ابتدا یک چارچوب برخط داده پیوندی زیست بوم نرم‌افزار فراهم می‌کند، که استخراج داده و یکپارچه‌سازی درون مجموعه داده‌ای را از سیستم‌های کنترل نسخه، ارزیابی کیفیت، ردیابی اشکال در لحظه پشتیبانی می‌کند. در اولین انتشار آن، مجموعه داده دو بیلیون واقعیت، همانند عبارات منبع کد، گواهینامه[1] نرم‌افزار، و شبیه‌سازی[2] از ۱۸۰۰۰ پروژه نرم‌افزاری را شامل می‌شود. در دومین انتشار آن، پروژه SeCold واقعیت‌های افزوده کاوش‌شده از ردیاب اشکال و سیستم نسخه‌بندی را در بر می‌گیرد. رویکرد اصلی تحقیق مبتنی بر مفاهیم پایه همانند ویکی پدیا می‌باشد: محققین و توسعه‌دهنده‌های ابزار نتایج تحلیل بدست آمده از ابزارهای آن‌ها را بوسیله انتشار آن‌ها به عنوان بخشی از پرتال SeCold به اشتراک می‌گذارند و بدین ترتیب آن‌ها را بخش جدایی‌ناپذیر از دامنه دانش عمومی می‌سازند. پروژه SeCold یک عضو رسمی از ابر مجموعه داده پیوندی است و در حال حاضر هشتمین مجموعه داده بزرگ برخط در دسترس وب می‌باشد (کیوانلو و همکاران[3]، ۲۰۱۲). تحقیقی دیگر در مخزن‌کاوی نرم‌افزار (MSR)، دربردارنده موضوعات بشری است، به طوری که معمولا مخازن داده‌های مربوط به توسعه‌دهندگان و کاربران در تعامل با مخازن و با یکدیگر را شامل می‌شوند. موضوعات اخلاقی بوسیله چنین تحقیقی برجسته می‌شود، بنابراین قبل از شروع در مخازن‌کاوی نرم‌افزار باید در نظر گرفته شود. تحقیق در مورد موضوعات اخلاقی می‌باشد و می‌تواند در تحقیق MSR در نظر گرفته شود. با استفاده از چالش‌های کاوش از سال ۲۰۰۶ تا ۲۰۲۱ به عنوان یک مطالعه موردی برای شناسایی انواع داده به کار برده می‌شود. بر اساس تحقیقات کنونی چارچوب‌های اخلاقی می‌تواند چالش مورد بحث در ایجاد و کاربرد مخازن و مجموعه داده‌های مرتبط باشد. همچنین گزارش برخی نتایج نظرسنجی جامعه از رویکردها برای اخلاق در تحقیق MSR گزارش می‌شود. بعلاوه، تحقیق چهار مورد مطالعه از موضوعات اخلاقی عادی که فرد در پروژه‌ها با آن مواجه می‌شود، را ارائه می‌دهد و چگونگی شکل‌گیری پروژه قبل از شروع را بوسیله در نظر گرفتن اخلاق نشان می‌دهد. مبتنی بر تجربه برخی دستورالعمل‌های حاضر و عمل‌های انجام شده، می‌تواند به طور بالقوه موضوعات اخلاقی را در نظر گیرد و میزان خطرات را کاهش دهد (گلد و کنیک[4]، ۲۰۲۲). مخازن نرم‌افزاری اطلاعات فراوان ارزشمندی در مورد پروژه‌های منبع باز تولید می‌کنند. با افزایش حجم داده‌ی نگهداری شده بوسیله مخازن، استخراج خودکار چنین داده‌هایی از مخازن مجزا، به اندازه اطلاعات پیوسته از طریق مخازن ضروری می‌شود. در تحقیق دیگری یک چارچوب توصیف می‌شود که خراشیدن وب[5] را به طور خودکار برای کاوش مخازن و اطلاعات پیوندی از طریق مخازن توصیف می‌کند. در این تحقیق دو پیاده سازی از چهارچوب مورد بحث قرار می‌گیرد. در پیاده‌سازی اول، به طور خودکار گزارش‌های مسائل امنیتی از مخازن پروژه‌ها شناسایی و جمع‌آوری می‌شود که ردیاب اشکال باگ‌زیلا را با استفاده از اطلاعات آسیب‌پذیری مرتبط از پایگاه داده ملی آسیب‌پذیری برپا می‌کند، در پیاده‌سازی دوم، گزارش‌های مسائل امنیتی برای پروژه‌ها جمع‌آوری می‌شود که ردیاب اشکال لانچ پد[6] با اطلاعات آسیب‌پذیری آن از پایگاه داده ملی آسیب‌پذیری مرتبط شده است.

ابزارهای موردنظر تحقیق روی نسخه‌های مختلف از پروژه های RedHat، Suse، Ubuntu، Fedora، و Firefox ارزیابی شده است. درصد اشکال‌های امنیتی شناسایی

1- Licenses
2- Clone
3- Keivanloo & etal
4- Gold, & Krinke
5- Web Scraping
6- Launchpad

شده با استفاده از ابزار تحقیق با اشکال‌های گزارش‌شده بوسیله سایر محققین سازگار می‌باشد (انباگان و ووک[1]، ۲۰۰۹). تعداد زیادی از مخازن نرم‌افزاری روی بستر اینترنت به طور اساسی تضادهای سنتی از نگهداری نرم‌افزار را تغییر می‌دهند. طبقه‌بندی کارا از پروژه‌های انبوه برای بازیابی نرم‌افزار مرتبط در این مخازن از اهمیت حیاتی برای وظایف نگهداری مبتنی بر اینترنت همانند جستجوی راه حل، یادگیری بهترین شیوه برخوردار می‌باشد. تعداد زیادی از کارهای پیشین روی طبقه‌بندی نرم‌افزار بوسیله کاوش کد منبع یا کد بایت انجام شده است که فقط به طور نسبی مجموعه کوچکی از پروژه‌ها با طبقه‌بندی درشت‌دانه یا خوشه‌ها تایید می‌شوند.

به هر حال نگهداری نرم‌افزار مبتنی بر اینترنت به رویکردهای دانه‌بندی ریز، مقیاس‌پذیری بیشتر و طبقه‌بندی مستقل از زبان نیاز دارد. در تحقیق یک رویکرد جدید برای طبقه‌بندی سلسله مراتبی پروژه‌های نرم افزاری مبتنی بر نمایه‌های برخط آن‌ها از طریق چندین مخازن ارائه می‌شود و نیز یک چارچوب طبقه‌بندی مبتنی بر SVM برای دسته‌بندی تعداد انبوهی از نرم‌افزارها به طور سلسله مراتبی طراحی شده است. انواع مختلفی از ویژگی‌های نمایه از چندین مخازن جمع‌آوری و یک راهبرد[2] ترکیبی وزن‌دار طراحی شده است که وزن‌های بیشتری را برای ویژگی‌های مهم‌تر اختصاص می‌دهد. آزمایش‌های گسترده‌ای روی بیش از ۱۸۰۰۰ پروژه با استفاده از سه مخزن انجام شده است. نتایج نشان می‌دهد که رویکرد پیشنهادی تحقیق بهبود معناداری را با استفاده از ترکیب وزن‌دار می‌تواند دقت کلی و فراخوانی و معیار-F را به ترتیب به مقادیر ۷۱٫۴۱٪، ۶۵٫۶۰٪، و ۶۸٫۳۸٪ در تنظیمات مناسب بدست آورد. در مقایسه با کارهای پیشین رویکرد پیشنهادی تحقیق، نتایج رقابتی با ۱۲۳ طبقه‌بندی دانه ریز و چند لایه ارائه دهد. در تضاد با این موضوع، با استفاده از کد منبع یا کد بایت رویکرد پیشنهادی برای طبقه‌بندی نرم‌افزار در مقیاس بزرگ و مستقل از زبان بسیار کارا می‌باشد (وانگ و همکاران[3]، ۲۰۱۳). سیستم‌های مهندسی نرم‌افزار یک فعالیت چندرشته‌ای است به طوری که تعدادی مصنوعات باید به طور همزمان ایجاد و نگهداری گردد. در تحقیق کد تولیدی و تکامل آزمون‌های همروند بوسیله استخراج سیستم نسخه‌بندی پروژه، گزارش پوشش کد، و معیارهای اندازه برآورد می‌شود. هدف اصلی برای مطالعه این هم‌تکاملی ایجاد آگاهانه توسط توسعه دهنده‌ها و مدیران در مورد فرآیند آزمایشی است که دنبال می‌شود. امکان‌سنجی روش پیشنهادی از طریق دو مطالعه موردی منبع باز استخراج می‌شود و یک تعداد متنوعی از سناریوهای هم‌تکاملی مشاهده می‌شود. نتایج تحقیق با کمک دو مورد ثبت وقایع - پیام و توسعه‌دهنده‌های اصلی از سیستم نرم‌افزاری ارزیابی می‌شود. در تحقیق، هم‌تکاملی بین کد تولیدی و کد تستی مطالعه می‌شود. سه دید به صورت الف) تاریخچه تغییر ب) تاریخچه رشد ج) دید ارزیابی کیفیت معرفی می‌شود. استفاده از این دیدها روی دو مطالعه موردی پدیدار می‌شود و بیشترین هم‌تکاملی همزمان (Checkstyle) از یک رویکرد آزمایشی مرحله‌ای (ArgoUML) مشخص می‌شود (زادمن، ون رومپی، دیمیر، و ون دروسن[4]، ۲۰۰۸). در دنیای نرم‌افزار محور امروز، مخازن نرم‌افزاری مقیاس فوق‌العاده بزرگ[5] همانند SourceForge، Github، و کد گوگل کتابخانه‌های جدید اسکندریه هستند. این مخازن مجموعه عظیم از نرم‌افزار و اطلاعات مرتبط را شامل می‌شوند. دانشمندان و محققین به طور مشابه به تحلیل این ثروت اطلاعاتی علاقمند هستند. به هر حال استخراج نظام‌مند و تحلیل داده مربوطه از این مخازن برای تست فرضیه‌ها دشوار است و بهترین کار برای کارشناسان مخازن‌کاوی نرم‌افزار است. به طور ویژه، کاوش کد منبع، دید معناداری در مصنوعات و فرآیندهای توسعه نرم‌افزار بدست می‌دهد. متاسفانه، کاوش کد منبع در مقیاس بزرگ یک وظیفه دشوار را بجای می‌گذارد. رویکردهای قبلی بیشتر در حوزه دانه‌بندی درشت است و یا تحت شعاع تاریخچه کد قرار می‌گیرد.

در تحقیق کاوش منبع کد مورد بحث قرار می‌گیرد. الف) در یک مقیاس خیلی بزرگ ب) در یک سطح دانه‌بندی ریز

1- Anbalagan, & Vouk
2- Strategy
3- Wang, Wang, Yin, Ling, Li, & Zou
4- Zaidman, Van Rompaey, Demeyer, & Van Deursen
5- Ultra

از جزییات. ج) به همراه اطلاعات کامل تاریخچه. به منظور آدرس‌دهی این چالش‌ها، ویژگی‌های زبان اختصاصی دامنه برای کاوش منبع کد در زبان و ساختار به نام Boa ارائه می‌دهد. هدف Boa تست آسان فرضیه‌های مربوط به MSR می‌باشد. ارزیابی تحقیق به طور قابل‌توجهی زحمت‌های برنامه‌نویسی را کاهش می‌دهد. بنابراین موانع ورود را کاهش می‌دهد. همچنین بهبود چشمگیری در مقیاس‌پذیری را نشان می‌دهد (دییر، نگین، راجان، و نگین[1]، ۲۰۱۵). الگوریتم‌های زیرمجموعه یادگیری عمیق که به طور خودکار بازنمایی ترکیبی را یاد می‌گیرند. توانایی این مدل‌ها برای تعمیم، پیشرفت‌های زیادی را در بسیاری از زمینه‌ها همانند پردازش زبان طبیعی (NLP) سبب شده است. تحقیقات اخیر در زمینه مهندسی نرم افزار (SE) به طور جامع فواید کاربرد روش NLP را برای بدنه نرم‌افزار آشکار می‌کند. از این رو، در تحقیق یادگیری عمیق برای مدل‌سازی زبان نرم افزار و تفاوت‌های اساسی برجسته بین نوآوری مدل‌های زبان نرم‌افزار و مدل‌های ارتباط‌گرا مشخص شده است. مدل‌های یادگیری تحقیق برای فایل‌های کد منبع (از این رو فقط نیاز به تحلیل لغوی کد منبع نوشته شده در هر زبان برنامه‌نویسی نیاز می‌شود) و دیگر انواع مصنوعات کاربردی می‌باشند. در تحقیق نشان داده می‌شود، چگونه مدل می‌تواند به طور موثری حالت آن را برای مدل‌سازی داده ترتیبی به یاد آورد. به عبارت دیگر جریان توکن‌های نرم‌افزاری و حالت آن حاکی از توکن‌های مجزا در یک پیشوند می‌باشد. سپس نمونه‌ای از مدل‌های یادگیری عمیق ارائه می‌شود و نشان داده می‌شود که یادگیری عمیق مدل‌های با کیفیت بالا را در مقایسه با n-gram ها و n-gram های مبتنی بر حافظه نهان روی بدنه‌ای از پروژه‌های جاوا کاهش می‌دهد. تحقیقبا دو فراپارامتر مدل آزمایش می‌شود که به منظور اطلاع پیش‌بینی‌ها قبل از ساختن چندین مدل زبان نرم‌افزاری به هدف عمومی‌سازی، ظرفیت و مقادیر متن را کنترل می‌کند.

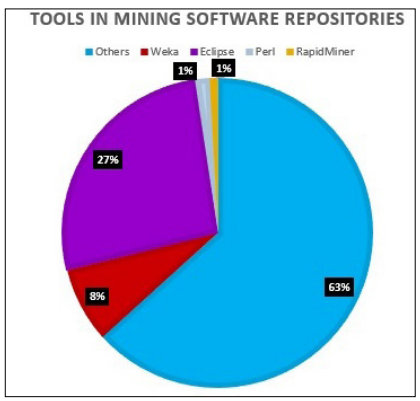
در تحقیق مدل‌های یادگیری عمیق برای پیشنهاد کد و اثربخشی آن در یک وظیفه مهندسی نرم‌افزار واقعی در مقایسه با مدل‌های عملی به کار برده می‌شود. برای کارهای آینده یادگیری عمیق می تواند تست مبتنی بر مدل را پشتیبانی کند و نرم‌افزار را از لحاظ لغوی بهبود بخشد و مفهوم مصنوعات نرم افزار را بیان کند. کار تحقیقی به عنوان اولین مرحله از مخازن‌کاوی نرم‌افزار در نظر گرفته شود (وایت، وندوم، لینارس وسکوئیز، وپوشینک[2]، ۲۰۱۵). اگر چه نوآوری در زمینه مخازن‌کاوی نرم‌افزار مبتنی بر اطلاعات سیستم نسخه‌بندی برای ارزیابی به تحول یک سیستم نرم‌افزاری مفید می‌باشد، اطلاعات آن‌ها به چندین روش محدود می‌باشد. سیستم نسخه‌بندی همانند CVS یا انباره زیرنسخه فقط متن فایل‌ها را ذخیره می‌کند که به از دست دادن اطلاعات منجر می‌شود: دنباله‌ای دقیق از تغییرات بین دو نسخه برای بازیابی دشوار می‌باشد. در تحقیق یک مخزن اطلاعات انتخابی ارائه می‌شود که تغییرات افزایشی را برای سیستم تحت مطالعه ذخیره می‌کند، از IDE بازیابی و برای ساخت نرم‌افزار استفاده می‌شود. سپس این مدل مبتنی بر تغییر از تحول سیستم برای ارزیابی به هنگام بازسازی در دو مطالعه موردی استفاده می‌کند و یافته‌های تحقیق با بازسازی تشخیص رویکردها روی مخازن سیستم نسخه‌بندی کلاسیک مقایسه می‌شود (رابس[3]، ۲۰۰۷). برنامه‌نویسی ذاتا چندوجهی است و شامل چندین مهارت می‌باشد، با استفاده از چارچوبی مثل گیت‌هاب، توسعه‌دهنده‌ها فرصت مشارکت در پروژه‌های از سازمان‌های مختلف و همکاری با سایر توسعه دهنده‌های جهان را دارند. از طریق داده گیت‌هاب، فرصت‌های جدیدی برای شناسایی قابلیت‌های توسعه‌دهنده‌ها امکان‌پذیر است. همچنین از طریق گیت‌هاب، چندین مهارت از جمله نقش کاربر یک کتابخانه را از توسعه‌دهنده استنتاج کنیم. در تحقیقی، یک روش برای شناسایی خبرگان کتابخانه مبتنی بر دانش محصول آن‌ها روی گیت‌هاب پیشنهاد می‌گردد. روش پیشنهادی روی یک آزمایش برای شناسایی خبرگان احتمالی در سه کتابخانه جاوا انجام می‌شود. بدین صورت که روش پیشنهادی ۱۰۰ توسعه‌دهنده برتر را برای هر فناوری شناسایی می‌کند. سپس نمایه‌های

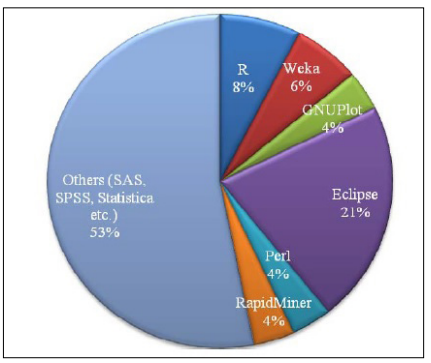
1- Dyer, Nguyen, Rajan, & Nguyen
2- White,Vendome, Linares-Vásquez, & Poshyvanyk
3- Robbes

این توسعه‌دهنده‌ها روی شبکه اجتماعی لینکدین به منظور تطبیق آنچه روی لینکدین گزارش می‌دهند و آنچه که در گیت‌هاب تولید می‌کنند، مقایسه می‌گردد. همچنین به منظور مقایسه نتایج تحقیق برای تحلیل خودکار، دانش‌آموزان مورد بررسی قرار گرفته‌اند. نتایج نشان می‌دهد ۸۹ درصد توسعه‌دهندگان منتخب گیت‌هاب مهارت‌های خود را در سایت‌های شبکه اجتماعی لینکدین منطبق بر رتبه‌بندی ساخته شده در روش تحقیق گزارش داده‌اند و رتبه‌بندی تولید شده در روش تحقیق بوسیله روش پیشنهادی به دسته‌بندی ایجاد شده توسط مشارکت‌کنندگان نظرسنجی مرتبط می‌شود (سانتوس، سوزا، اولیورا، و فیگردوا، ۲۰۱۸).

۳- یافته‌های تحقیق

مقالات متعدد ارائه شده در زمینه مخازن‌کاوی نرم افزار مورد مطالعه و بررسی و نتایج بدست آورده شده با نتایج تحقیق چاتودی و همکاران (۲۰۱۳) مورد مقایسه قرار داده شد (شکل ۱). با توجه به این که تحقیق مذکو، مقالات منتشر شده با موضوع ابزار مورد استفاده در حوزه مخازن‌کاوی نرم‌افزار در کنفرانس ICSE در سال‌های ۲۰۰۷ تا ۲۰۱۲ را مورد مطالعه و بررسی قرار داده است، در تحقیق حاضر مقالات منتشر شده با موضوع ابزار مورد استفاده در حوزه مخازن‌کاوی نرم‌افزار و نیز برخی از روش‌های داده‌کاوی در کنفرانس ICSE در سایر کنفرانس ها و مجلات در سال های ۲۰۰۷ تا ۲۰۲۴ مورد مطالعه و بررسی قرار گرفته است، پایگاه گوگل اسکولار به منظور استخراج داده‌های پژوهش مورد استفاده قرار گرفته است.

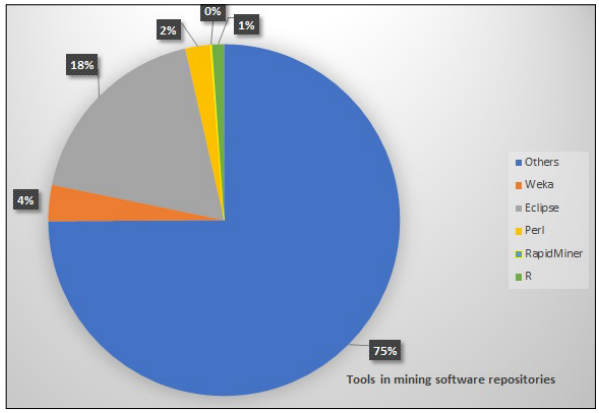




شکل (۱): ابزارهای مورد استفاده در مخازن کاوی نرم افزار، شکل سمت راست نمودار مرجع (چاتودی و همکاران، ۲۰۱۳) و سمت چپ نمودار نتایج تحقیق جاری

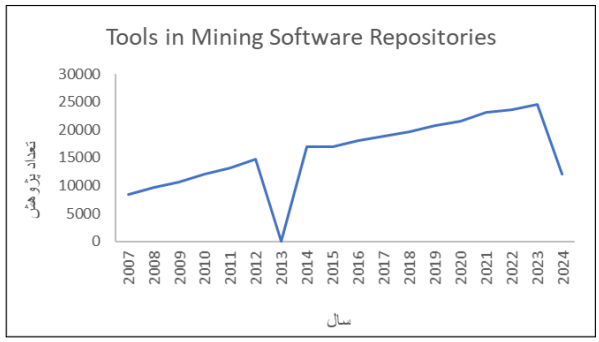
همانطور که در شکل شماره (۱) مشخص است، درصد استفاده از Eclips در مخازن‌کاوی نرم‌افزار بیشترین مقدار پژوهش‌های کنفرانس ICSE را به خود اختصاص داده است و تحقیق حاضر نشان می‌دهد، تعداد پژوهش‌های مربوطه در پایگاه اسکولار مقدار بیشتری را نسبت به پژوهش]۱۸[نشان می‌دهد. این مقدار در مورد کاربرد زبان برنامه‌نویسی perl و چارچوب RapidMiner در حوزه مخازن‌کاوی نرم افزار با داشتن مقدار یکسان در تحقیق چاتودی و همکاران (۲۰۱۳) مقدار بیشتری را به خود اختصاص داده است. در مورد کاربرد ابزار Weka، تعداد پژوهش‌های استخراج شده در تحقیق حاضر که از این ابزار استفاده کرده‌اند، نسبت به پژوهش فوق مقدار بیشتری دارد. مقدار بدست آمده برای زبان برنامه‌نویسی R و نیز برنامه GNUPlot در تحقیق حاضر، صفر می‌باشد. به طور کلی نتایج نشان می‌دهد، ابزارها و روش‌های مذکور کمتر از نیمی از کل پژوهش‌ها را به خود اختصاص داده‌اند.

1- Santos, Souza, Oliveira, & Figueiredo



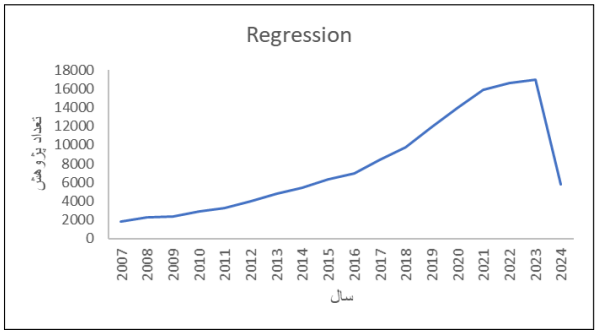
شکل (۲): درصد استفاده از ابزارهای مخازن کاوی نرم‌افزار بین سال‌های ۲۰۰۷-۲۰۲۴

شکل شماره (۲) ، برآوردها و نتایج استفاده از ابزار و روش‌های داده‌کاوی در مخازن‌کاوی نرم‌افزار را از سال ۲۰۰۷ تا ۲۰۲۴ نشان می‌دهد، که بروزرسانی پژوهش‌های پیشین را نشان می‌دهد و نسبت به پژوهش‌های پیشین تعداد پژوهش‌های بیشتری مورد تجزیه و تحلیل قرار گرفته است. همچنان که نتایج تحقیق حاضر نشان می‌دهد، محیط توسعه یکپارچه (IDLE) به عبارت دیگر نرم‌افزار Eclips بیشترین کاربرد را نسبت به سایر ابزارها و روش‌ها در مخازن‌کاوی نرم‌افزار دارا می‌باشد.



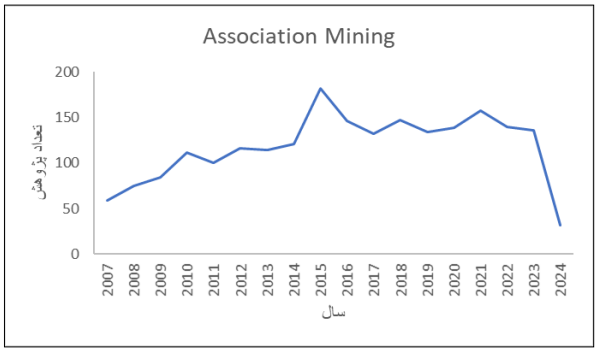
شکل (۳): تعداد پژوهش‌ها با حوزه موضوعی ابزارها در مخازن کاوی نرم‌افزار

شکل شماره (۳)، نشان‌دهنده تعداد پژوهش‌های انجام شده در سال های ۲۰۰۷ تا ۲۰۲۴ با موضوع ابزارها در مخازن‌کاوی نرم‌افزار می‌باشد، چنان که از شکل بر می‌آید، در سال ۲۰۲۳ بیشترین و در سال ۲۰۱۳ کمترین تعداد پژوهش در زمینه مخازن‌کاوی نرم‌افزار و ابزارهای مورد استفاده در این زمینه انجام شده است.



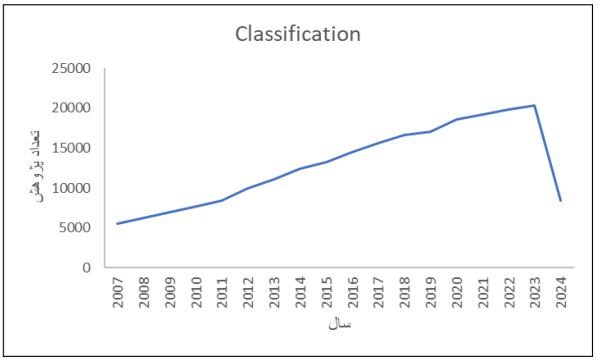
شکل (۴): تعداد پژوهش‌های منتشر شده در حوزه مخازن کاوی نرم‌افزار با استفاده از روش رگرسیون

شکل شماره (۴)، میزان کاربرد رگراسیون را در پژوهش‌های مخازن‌کاوی نرم‌افزار نشان می‌دهد، که کمترین میزان آن در سال ۲۰۰۷ و بیشترین مقدار کاربرد آن در سال ۲۰۲۳ بوده است و به طور کلی سیر صعودی داشته است.



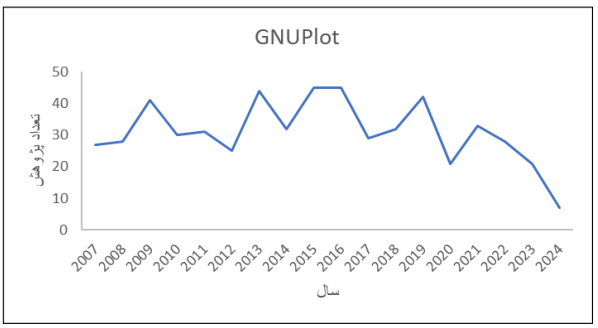
شکل (۵): تعداد پژوهش‌های انجام شده در حوزه مخازن کاوی نرم‌افزار با استفاده از روش کاوش قوانین انجمنی

شکل شماره (۵) میزان کاربرد قوانین انجمنی را به عنوان یکی از روش‌های داده‌کاوی در پژوهش‌های مخازن‌کاوی نرم‌افزار نشان می‌دهد، که کمترین میزان آن مربوط به سال ۲۰۰۷ و بیشترین مقدار کاربرد آن مربوط به سال ۲۰۱۵ بوده است. همانطور که در شکل مشخص است، روند تغییر نمودار غیریکنواخت است.



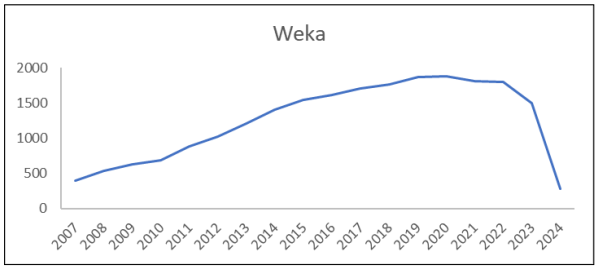
شکل (۶): تعداد پژوهش‌های انجام شده در حوزه مخازن کاوی نرم‌افزار با استفاده از روش دسته‌بندی

طبق شکل شماره (۶)، تعداد پژوهش‌هایی که از روش دسته‌بندی در مخازن‌کاوی نرم‌افزار استفاده نموده‌اند، در سال ۲۰۲۳ بیشترین مقدار و در سال ۲۰۰۷ کمترین مقدار را داشته است. در سال های ۲۰۱۱، ۲۰۱۵، ۲۰۱۹ کاهش اندک تعداد پژوهش مربوطه در سیر صعودی نمودار مشاهده می‌شود.



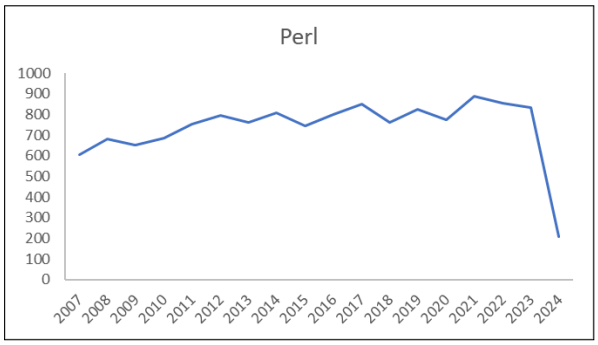
شکل ۷: تعداد پژوهش‌های انجام شده در حوزه مخازن کاوی نرم‌افزار با استفاده از GNUPlot

شکل شماره (۷) تغییر غیریکنواخت تعداد پژوهش‌ها در زمینه مخازن‌کاوی نرم‌افزار در محیط خط فرمان و برنامه GUI به عبارت دیگر GNUPlot را نشان می‌دهد. در سال های ۲۰۱۵ و ۲۰۱۶ بیشترین تعداد پژوهش با استفاده از GNUPlot انجام شده است. در سال های ۲۰۲۱ و ۲۰۲۳ کمترین تعداد پژوهش در این زمینه انجام شده است.



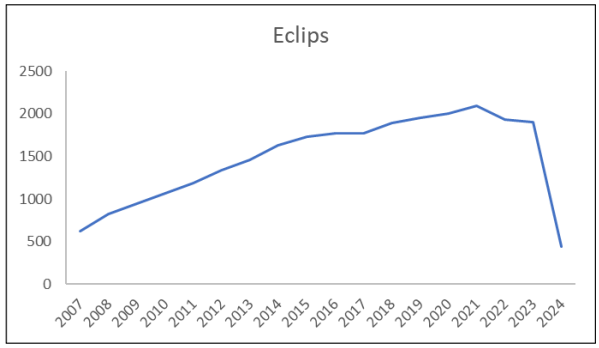
شکل ۸: تعداد پژوهش‌های انجام شده در حوزه مخازن کاوی نرم‌افزار با استفاده از ابزار Weka

شکل شماره (۸)، تعداد پژوهش‌هایی که از Weka به عنوان یکی از ابزارهای پرکاربرد داده‌کاوی استفاده نموده‌اند، را نمایش می‌دهد، که در سال‌های ۲۰۱۹ و ۲۰۲۰ بیشترین مقدار و در سال ۲۰۰۷ کمترین را دارد. بنابراین می توان نتیجه گرفت که در سال‌های اخیر با توجه به ظهور ابزارهای جدید داده‌کاوی، استقبال از Weka کاهش یافته است.



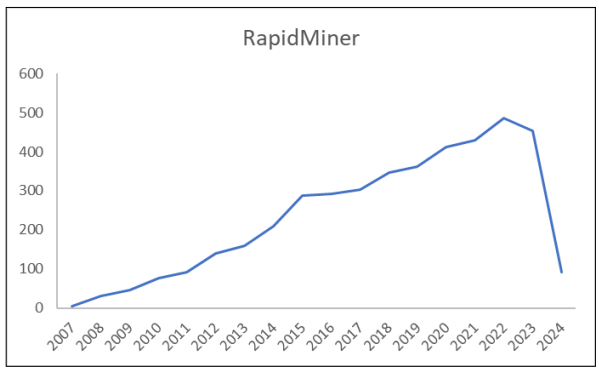
شکل (۴): تعداد پژوهش‌های انجام شده در حوزه مخازن کاوی نرم‌افزار با استفاده از زبان برنامه‌نویسی پرل

شکل شماره (۹) کاربرد زبان برنامه‌نویسی پرل را در حوزه مخازن‌کاوی نرم‌افزار نشان می‌دهد، اگر چه سیر تقریبی نمودار صعودی است، اما اختلاف عددی پژوهش‌ها کم است، در سال ۲۰۲۲ تعداد پژوهش‌ها در این زمینه بیشترین مقدار و در سال ۲۰۰۷ کمترین مقدار را داشته است.



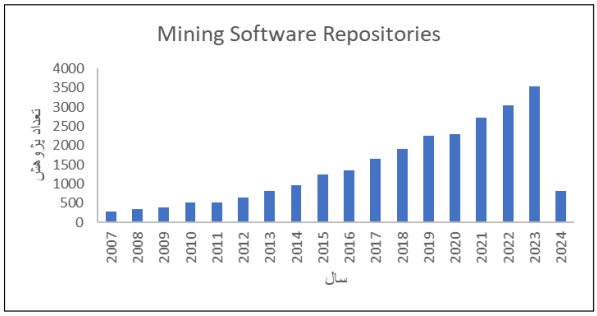
شکل (۱۰): تعداد پژوهش‌های انجام شده در حوزه مخازن کاوی نرم‌افزار با استفاده از نرم‌افزار Eclips

شکل شماره (۱۰)، میزان کاربرد نرم افزار Eclips را در پژوهش‌های مخازن‌کاوی نرم‌افزار نشان می‌دهد، که کمترین میزان آن در سال ۲۰۰۷ و بیشترین مقدار کاربرد آن در سال ۲۰۲۱ بوده است و به استثنای سال ۲۰۱۷ و ۲۰۲۲، نمودار به طور کلی روند رو به رشد داشته است.



شکل ۱۱: تعداد پژوهش‌های انجام شده در حوزه مخازن کاوی نرم‌افزار با استفاده از چارچوب RapidMiner

همانطور که در شکل شماره (۱۱) نمایان است، کاربرد چارچوب RapidMiner در سال‌های اخیر با اندکی نوسان روند افزایشی داشته است و در سال ۲۰۲۲ بیشترین مقدار و در سال ۲۰۰۷ کمترین مقدار را داشته است.



شکل (۱۲): تعداد پژوهش‌های انجام شده با حوزه موضوعی مخازن کاوی نرم‌افزار

در جستجوی پژوهش‌های مربوط به مخازن‌کاوی نرم‌افزار از موتور جستجوی گوگل‌اسکولار، نتایج شکل شماره (۱۲) بدست آمد. در این جستجو منحصرا و به طور دقیق مخازن‌کاوی نرم‌افزار مدنظر بوده است، به طوری که در سال ۲۰۰۷ کمترین پژوهش با موضوع مخازن‌کاوی نرم‌افزار و در سال ۲۰۲۳ بیشترین تعداد پژوهش مخازن‌کاوی نرم‌افزار انجام شده است. در نمودارهای فوق توجه به این نکته که این پژوهش در ماه‌های ابتدایی سال ۲۰۲۴ انجام شده است، به درک کم بودن تعداد پژوهش‌ها در این سال کمک می‌کند. با این حال از نمودارها کاملا مشخص است که روش دسته‌بندی همچنان به عنوان یکی از روش های داده‌کاوی مورد استقبال قرار می‌گیرد.

۴- نتیجه‌گیری

یکی از جنبه‌های کلیدی اهمیت مخازن‌کاوی در پژوهش‌های مربوط به مهندسی نرم‌افزار این است که محققین را قادر می‌سازد به تحلیل تکامل کد، بهره‌وری محصول، و جمع‌آوری عکس‌های فوری[1] از وضعیت‌های کد بپردازند (مهادی، ارنست، و تانگی[2]، ۲۰۲۲). مرور نظام‌مند ادبیات، مطالعات مربوط به مخازن‌کاوی نرم‌افزار (MSR) را شناسایی می‌کند، و بر ارزیابی مطالعات MSR از لحاظ فرآیندها و خروجی آن‌ها تاکید دارد. مخازن نرم‌افزاری به ارائه بینش در مورد آخرین پیشرفت‌ها در توسعه نرم‌افزار و ارائه درک عمیق‌تر از تکامل نرم‌افزار کمک می‌کند. چالش‌های مخازن‌کاوی نرم‌افزار شامل ملاحظات اخلاقی، کاوش مقیاس بزرگ، و استفاده از منابع داده‌ای مختلف است.

1- Snapshots
2- Mahadi, Ernst, & Tongay

چالش‌های اخلاقی ناشی از ماهیت انسانی مخازن نرم‌افزاری است که حاوی داده‌هایی درباره تعامل توسعه‌دهندگان با مخازن و با یکدیگر است. کاوش مقیاس بزرگ برای نتایج تحقیقاتی قابل تعمیم حائز اهمیت است و نیاز به استفاده از مجموعه بزرگی از مصنوعات نرم‌افزاری دارد. منابع داده‌ای مختلفی بوسیله چالش‌های کاوش استفاده یا فراهم می گردد، که دربردارنده داده کنترل نسخه، داده ردیابی اشکال، بایگانی‌های ایمیل، ایجاد ثبت‌های وقایع، سرریز پشته، و رویدادهای IDE است. هر منبع داده‌ای چالش‌های خاص خود را دارد، همانند تضمین حریم خصوصی، مدیریت حجم زیادی از داده، و مواجهه با پیچیدگی فرمت‌های داده‌ای مختلف. در تحقیق حاضر، ابزارها و روش‌های مختلف مورد استفاده در مخازن‌کاوی نرم افزار مورد بررسی قرار گرفت و نتایج با برخی از پژوهش‌ها، در بازه زمانی یکسان مقایسه گردید، علاوه بر این نتایج در بازه زمانی ۲۰۱۲ تا ۲۰۲۴ نیز مورد مطالعه و بررسی قرار داده شد. همانطور که در نتایج مشخص است، نرم افزار Eclips به نسبت بیشترین کاربرد را در مخازن‌کاوی نرم‌افزار دارد، و ابزار GNUPlot و برنامه R کمترین تعداد پژوهش‌ها را در بر می‌گیرد، تعداد پژوهش‌ها در سال ۲۰۰۷ کمترین مقدار و به طور نسبی در سال‌های ۲۰۱۹ تا ۲۰۲۳ بیشترین مقدار را دارا می باشد (با در نظر گرفتن کامل نبودن سال ۲۰۲۴). تعداد پژوهش‌های صورت گرفته با ابزار مورد نظر کمتر از نیمی از تعداد پژوهش‌ها انجام شده با سایر ابزارها را در بر می‌گیرد، که به نسبت نشان دهنده استقبال خوب محققین از ابزارهای مورد بررسی در این پژوهش می‌باشد.

منابع:

1-Vidoni, M. (2022). A systematic process for Mining Software Repositories: Results from a systematic literature review. *Information and Software Technology*, *144*, 106791.
2-Sun, X., Li, B., Leung, H., Li, B., & Li, Y. (2015). MSR4SM: Using topic models to effectively mining software repositories for software maintenance tasks. *Information and Software Technology*, *66*, 1-12.
3-Yu, L. (2012). An evolutionary programming based asymmetric weighted least squares support vector machine ensemble learning methodology for software repository mining. *Information Sciences*, *191*, 31-46.
4-Shang, W., Adams, B., & Hassan, A. E. (2012). Using Pig as a data preparation language for large-scale mining software repositories studies: An experience report. *Journal of Systems and Software*, *85*(10), 2195-2204.
5-Vandecruys, O., Martens, D., Baesens, B., Mues, C., De Backer, M., & Haesen, R. (2008). Mining software repositories for comprehensible software fault prediction models. *Journal of Systems and software*, *81*(5), 823-839.
6-Meqdadi, O., Alhindawi, N., Alsakran, J., Saifan, A., & Migdadi, H. (2019). Mining software repositories for adaptive change commits using machine learning techniques. *Information and Software Technology*, *109*, 80-91.
7-Voinea, L., & Telea, A. (2007). Visual data mining and analysis of software repositories. *Computers & Graphics*, *31*(3), 410-428.
8-Bakar, N. S. A. A. (2011). Using language-based search in mining large software repositories. *Procedia-Social and Behavioral Sciences*, *27*, 160-168.
9-De Luca, M., Fasolino, A. R., Ferraro, A., Moscato, V., Sperlí, G., & Tramontana, P. (2023). A community detection approach based on network representation learning for repository mining. *Expert Systems with Applications*, *231*, 120597.
10-Polaczek, J., & Sosnowski, J. (2021). Exploring the software repositories of embedded systems: An industrial experience. *Information and Software Technology*, *131*, 106489.
11-Assunção, W. K., Krüger, J., Mosser, S., & Selaoui, S. (2023). How do microservices evolve? An empirical analysis of changes in open-source microservice repositories. *Journal of Systems and Software*, *204*, 111788.
12-Goeminne, M., & Mens, T. (2013). A comparison of identity merge algorithms for software repositories. *Science of Computer Programming*, *78*(8), 971-986.
13-Goyal, A., & Sardana, N. (2020). Performance assessment of bug fixing process in open source

repositories. *Procedia Computer Science*, *167*, 2070-2079.
14-Prechelt, L., & Pepper, A. (2014). Why software repositories are not used for defect-insertion circumstance analysis more often: A case study. *Information and Software Technology*, *56*(10), 1377-1389.
15-Ampatzoglou, A., Michou, O., & Stamelos, I. (2013). Building and mining a repository of design pattern instances: Practical and research benefits. *Entertainment Computing*, *4*(2), 131-142.
16-Alqahtani, S. S., Eghan, E. E., & Rilling, J. (2016). Tracing known security vulnerabilities in software repositories–A Semantic Web enabled modeling approach. *Science of Computer Programming*, *121*, 153-175.
17-Spadini, D., Aniche, M., & Bacchelli, A. (2018, October). PyDriller: Python framework for mining software repositories. In *Proceedings of the 2018 26th ACM Joint meeting on european software engineering conference and symposium on the foundations of software engineering* (pp. 908-911).
18-Chaturvedi, K. K., Sing, V. B., & Singh, P. (2013, June). Tools in mining software repositories. In *2013 13th International Conference on Computational Science and Its Applications* (pp. 89-98). IEEE.
19-Sokol, F. Z., Aniche, M. F., & Gerosa, M. A. (2013, September). MetricMiner: Supporting researchers in mining software repositories. In *2013 IEEE 13th International Working Conference on Source Code Analysis and Manipulation (SCAM)* (pp. 142-146). IEEE.
20-Kagdi, H., Maletic, J. I., & Sharif, B. (2007, June). Mining software repositories for traceability links. In *15th IEEE International Conference on Program Comprehension (ICPC'07)* (pp. 145-154). IEEE.
21-Thomas, S. W. (2011, May). Mining software repositories using topic models. In *Proceedings of the 33rd International Conference on Software Engineering* (pp. 1138-1139).
22-Olatunji, S. O., Idrees, S. U., Al-Ghamdi, Y. S., & Al-Ghamdi, J. S. A. (2010). Mining software repositories–a comparative analysis. *International Journal of Computer Science and Network Security*, *10*(8), 161-174.
23-Kiefer, C., Bernstein, A., & Tappolet, J. (2007, May). Mining software repositories with isparol and a software evolution ontology. In *Fourth International Workshop on Mining Software Repositories (MSR'07: ICSE Workshops 2007)* (pp. 10-10). IEEE.
24-de F. Farias, M. A., Novais, R., Júnior, M. C., da Silva Carvalho, L. P., Mendonça, M., & Spínola, R. O. (2016, April). A systematic mapping study on mining software repositories. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing* (pp. 1472-1479).
25-Mittal, M., & Sureka, A. (2014, May). Process mining software repositories from student projects in an undergraduate software engineering course. In *Companion proceedings of the 36th international conference on software engineering* (pp. 344-353).
26-Canfora, G., & Cerulo, L. (2005, September). Impact analysis by mining software and change request repositories. In *11th IEEE International Software Metrics Symposium (METRICS'05)* (pp. 9-pp). IEEE.
27-Meng, X., Miller, B. P., Williams, W. R., & Bernat, A. R. (2013, September). Mining software repositories for accurate authorship. In *2013 IEEE international conference on software maintenance* (pp. 250-259). IEEE.
28-Keivanloo, I., Forbes, C., Hmood, A., Erfani, M., Neal, C., Peristerakis, G., & Rilling, J. (2012, June). A linked data platform for mining software repositories. In *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)* (pp. 32-35). IEEE.
29-Gold, N. E., & Krinke, J. (2022). Ethics in the mining of software repositories. *Empirical Software Engineering*, *27*(1), 17.
30-Anbalagan, P., & Vouk, M. (2009, May). On mining data across software repositories. In *2009 6th IEEE International Working Conference on Mining Software Repositories* (pp. 171-174). IEEE.
31-Wang, T., Wang, H., Yin, G., Ling, C. X., Li, X., & Zou, P. (2013, September). Mining software profile across multiple repositories for hierarchical categorization. In *2013 IEEE International Conference on Software Maintenance* (pp. 240-249). IEEE.

32-Zaidman, A., Van Rompaey, B., Demeyer, S., & Van Deursen, A. (2008, April). Mining software repositories to study co-evolution of production & test code. In *2008 1st international conference on software testing, verification, and validation* (pp. 220-229). IEEE.
33-Dyer, R., Nguyen, H. A., Rajan, H., & Nguyen, T. N. (2015). Boa: Ultra-large-scale software repository and source-code mining. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *25*(1), 1-34.
34-White, M., Vendome, C., Linares-Vásquez, M., & Poshyvanyk, D. (2015, May). Toward deep learning software repositories. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories* (pp. 334-345). IEEE.
35-Robbes, R. (2007, May). Mining a change-based software repository. In *Fourth International Workshop on Mining Software Repositories (MSR'07: ICSE Workshops 2007)* (pp. 15-15). IEEE.
36-Santos, A., Souza, M., Oliveira, J., & Figueiredo, E. (2018, September). Mining software repositories to identify library experts. In *Proceedings of the VII Brazilian Symposium on Software Components, Architectures, and Reuse* (pp. 83-91).
37-Mahadi, A., Ernst, N. A., & Tongay, K. (2022). Conclusion stability for natural language based mining of design discussions. *Empirical Software Engineering*, *27*, 1-42.