دوره پانزدهم، شماره پاییز و زمستان ۱۴۰۱

مجله فناوری اطلاعات در طراحی مهندسی

**Information Technology in Engineering Design**

**http://ited.sinaweb.net**

# مروری بر نگهداری نرم‌افزارهای کاربردی

**حدیث شفائی*(۱)  علی هارون آبادی (۲)**

**(۱) واحد علوم و تحقیقات، دانشگاه آزاد اسلامی، تهران، ایران***

**(۲) گروه مهندسی کامپیوتر و فناوری اطلاعات، واحد تهران مرکزی، دانشگاه آزاد اسلامی، تهران، ایران**

**(تاریخ دریافت :۱۴۰۲/۰۱/۳۰  تاریخ پذیرش: ۱۴۰۲/۰۵/۱۶)**

**چکیده**

نرم افزار کاربردی نوعی نرم افزار است که وظایف و عملکرد خاصی را برای کاربر نهایی انجام می‌دهد. نرم افزار کاربردی طیف وسیعی از برنامه‌ها را در بر می‌گیرد که بر روی کامپیوتر شخصی، لپ تاپ، گوشی هوشمند و تبلت مورد استفاده قرار می‌گیرد. نمونه‌هایی از نرم‌افزار کاربردی عبارتند از واژه پردازها، نرم افزارهای آموزشی، نرم‌افزارهای حسابداری، نرم‌افزارهای گرافیکی و طراحی. یکی از شاخص‌های ارزیابی کیفیت نرم افزار کاربردی، قابلیت نگهداری آن می‌باشد. نگهداری نرم افزار به تغییرات اعمال شده روی نرم‌افزار پس از تحویل آن به منظور تصحیح خطاها، بهبود کارایی و تطبیق نرم افزار گفته می‌شود. این تحقیق به تحلیل رویکردهای نرم‌افزار کاربردی و نگهداری آن می‌پردازد. به منظور بررسی و مقایسه این رویکردها، شاخص‌های بازه زمانی، نوع مقاله و محدوده موضوعی در نظر گرفته شده است. آنچه که از نتایج مشخص می‌شود، علی رغم کلیدی بودن موضوع نگهداری نرم افزار کاربردی، در سال‌های اخیر کمتر  در پژوهش‌ها ارائه شده است. بنابراین ضرورت پژوهش در حوزه نگهداری نرم‌افزار کاربردی به عنوان گامی مهم برای آغاز و تداوم توسعه بیش از پیش احساس می‌شود.

کلمات کلیدی: نرم‌افزار کاربردی، تصحیح خطاها، بهبود کارایی، تطبیق نرم‌افزار، نگهداری

*عهده‌دار مکاتبات:

حدیث شفائی

نشانی: واحد علوم و تحقیقات، دانشگاه آزاد اسلامی، تهران، ایران

پست الکترونیکی: Shafaei_h@ymail.com

## ۱- مقدمه

نرم‌افزار یک مجموعه از اعضای پردازش‌کننده/محاسبه‌کننده می‌باشد که روی یک معماری سخت افزار اجرا می‌شود و سرویس-های مرتبط با تجهیزات را می‌تواند ارائه دهد. نرم‌افزار برنامه‌های کاربردی در انواع متفاوتی ایجاد می‌شوند:

نرم‌افزار کاربردی [۱](یا عملیاتی): درون یک وسیله تعبیه شده‌است، بخشی از سیستم نهایی می‌باشد و به مشتری به عنوان بخشی از یک پروژه و یا محصول تحویل داده می‌شود.

دو زیرنوع از نرم‌افزار کاربردی وجود دارند:

- برنامه‌های کاربردی قابل پیکربندی: در این نوع برنامه کاربردی، یک مجموعه داده[۲]، ویژگی‌های نرم‌افزار را برای یک کاربرد خاص فعال می‌سازد.
- برنامه‌های کاربردی غیرقابل پیکربندی (کمتر رایج هستند): برنامه‌های یا نرم‌افزارهایی شامل مجموعه‌ای ثابت از قابلیت‌ها و تنظیمات، که توسط کاربر قابل تغییر نیستند.

ابزار توسعه: مرتبط به نرم‌افزار کاربردی داخلی شرکت هستند و به مشتری تحویل داده نمی‌شوند. هدف این ابزارها کمک به تحقق (ویرایشگر، محیط یکپارچه توسعه[۳]، تولیدکننده‌ی کد، کامپایلر، شارژر)، و در یک مفهوم گسترده، شامل تست (محیط تستی، تست محک داخلی[۴]) می‌باشند.

تست محک داخلی: مجموعه تست‌هایی که روی یک بخش اصلی حذف شده از مجموعه انجام می‌شود.

ابزار برون خط[۵]: این دسته از نرم‌افزارها امکان آماده‌سازی عناصری مانند داده‌ها برای یکپارچه‌سازی در نرم‌افزار نهایی را فراهم می‌کنند، بخشی از نرم افزار کاربردی نهایی هستند اما روی تجهیزات اجرا نمی‌شوند و تحویل آن‌ها به مشتری اختیاری است.

تحقق یک نرم‌افزار کاربردی شامل فعالیت‌های توسعه همچنین تصدیق، اعتبارسنجی، تولید، توسعه و نگهداری می‌باشد. فرآیند تحقق یک نرم‌افزار کاربردی به پیاده سازی مراحل، فعالیت‌ها و منابع متفاوتی نیاز دارد. بدین دلیل می‌بایست آن را در یک چرخه حیات شکل دهیم:

یک نرم‌افزار کاربردی یک عضو از یک مجموعه پیچیده‌تر (سیستم، زیرسیستم، تجهیزات) می‌باشد. که در شکل ۱ نشان داده شده است. یادآوری می‌شود که یک نرم‌افزار کاربردی به طور مستقیم به تجهیزات مرتبط است و در صورت عدم وجود معماری سخت افزاری نمی‌توان نرم‌افزار کاربردی داشت. تحقق نرم‌افزار کاربردی به فرآیند تبدیل یک مفهوم یا ایده به یک نرم‌افزار کاربردی عملی می‌پردازد. فرآیند تحقق نرم‌افزار کاربردی می‌تواند در چندین فاکتور به حساب آورده شود: نوع نرم‌افزار کاربردی، اندازه نرم‌افزار کاربردی، بحرانی بودن، مهلت زمانی. نرم‌افزار کاربردی به دو موقعیت مرتبط می‌شود : مورد اول، تحقق بخشیدن به نرم‌افزار کاربردی از نوع غیریکپارچه (نرم‌افزار کاربردی دسکتاپ، کامپایلر، شبیه ساز، محیط تست)، و مورد دوم، به تحقق یک سیستم، زیرسیستم یا تجهیزات مرتبط می‌شود و بنابراین نیاز به تحقق یک نرم‌افزار کاربردی تعبیه شده وجود دارد. شکل ۱ رابطه تحقق یک نرم افزار را نشان می‌دهد [۱].

---

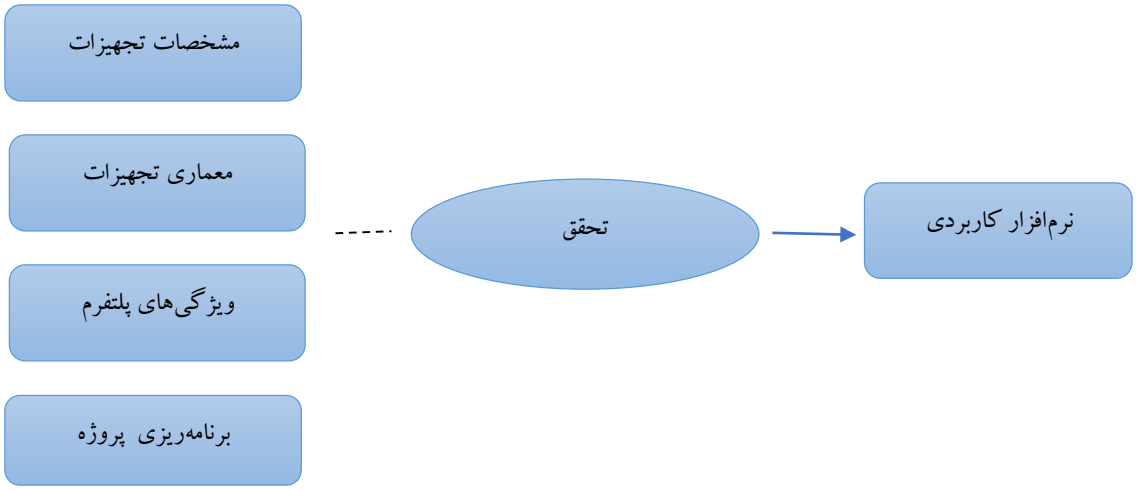
[۱] Application Software

[۲] Dataset

[۳] Integrated development environment

[۴] Internal bench testing

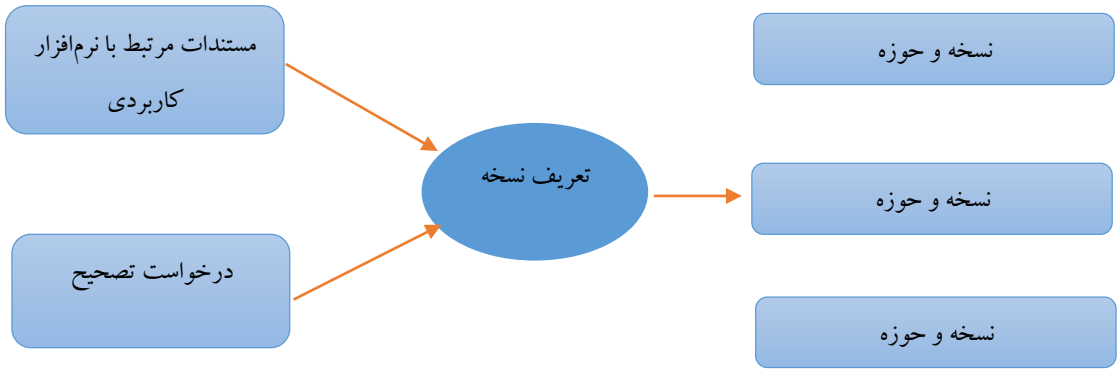
[۵] Offline




شکل ۱- رابطه تحقق یک نرم افزار کاربردی

### ۱-۱- فرآیند تعریف نسخه‌های نرم افزار

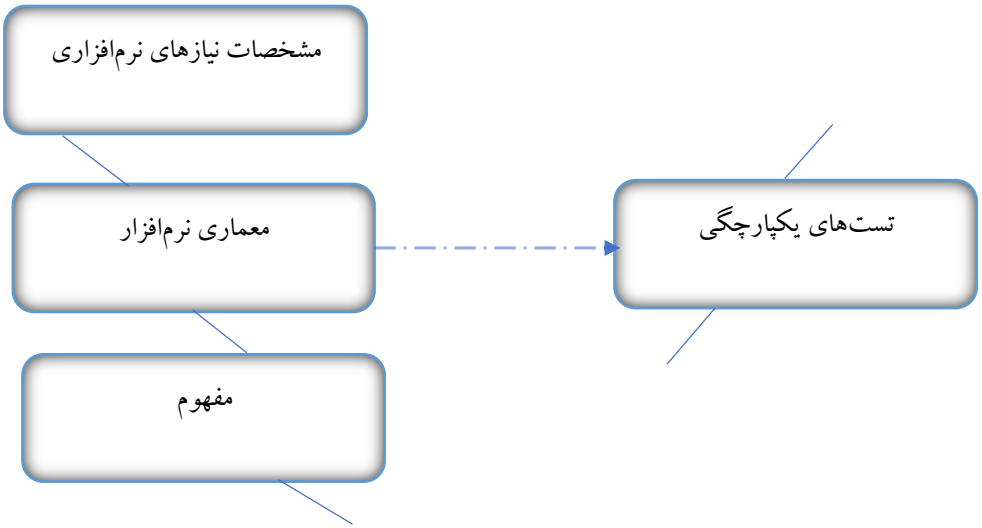
در طول راه‌اندازی تحقق اولین نسخه از یک نرم‌افزار کاربردی یا در طول عملیات نگهداری، حوزه نسخه‌ها باید تعریف شود. اسناد تعریف نسخه، منابع مهمی در توسعه نرم‌افزارهای کاربردی هستند و شرح جامعی از نسخه‌های نرم‌افزار کاربردی ارائه می‌دهند که شامل ویژگی‌ها، بهبودها، رفع اشکال و سایر تغییرات آن‌ها می‌باشد. درخواست تغییر در فرآیند تعریف نسخه به معنای درخواست رسمی برای اصلاح یا تغییر نسخه فعلی یک فرآیند ثبت می‌باشد (شکل ۲).




شکل ۲- فرآیند تعریف نسخه‌ها

فرآیند تعریف نسخه به شناسایی نسخه‌ها برای قرار گرفتن در یک مکان و تعریف محتوای آن نسخه کمک می‌کند. به عنوان بخشی از توسعه یک نرم افزار کاربردی پیچیده، تولید چندین نسخه و پیاده سازی یک تکرار شونده یا رویکرد افزایشی ضروری است. نیاز به تولید چندین نسخه بوسیله تمایل به انتشار یک رویکرد تکرارشونده/افزایشی استنتاج می‌شود [۱].

معماری یک نرم افزار کاربردی متشکل از واسطهایی با سخت افزار و نرم‌افزار مختلف و ارائه یک سازمان مبتنی بر سرویس‌ها (توابع، عملیات و ...) می‌باشد که نرم‌افزار کاربردی می‌بایست آن‌ها را فراهم کند. سازمان باید استقلال بین مولفه‌ها، قابلیت تست مولفه‌ها، مدیریت مولفه (استفاده مجدد از مولفه‌های توسعه داده شده‌ی قبلی، شامل سامانه تجاری در دسترس[۱]) و نیز قابلیت نگهداری نرم‌افزار کاربردی در زمان را تضمین نماید. پیاده‌سازی یک معماری نرم‌افزار کاربردی یک مرحله مهم است چرا که می‌بایست عناصر توسعه داده شده را شناسایی نماید. این عناصر عبارتند از: عناصر مجدد استفاده شده، عناصر تجاری (COTS)، واسط بین اعضا و واسط منابع سخت افزاری (مانند حافظه، آدرس خاص، پورت‌های ورودی/خروجی، CAN[۲]، آی پی و غیره) . هدف فاز معماری نرم‌افزار کاربردی تجزیه یک نرم افزار کاربردی در یک مجموعه از مولفه‌ها می‌باشد، در حالی‌که چندین معیار همچون استقلال، اتصال، قابلیت تست، و یا نگهداری را دنبال می‌کند. این مولفه‌های نرم‌افزاری یک مجموعه از واسطها را شامل خواهد بود و یک سرویس مجموعه را فراهم خواهد کرد. مولفه‌ها می‌توانند به طور خاص مولفه‌های پیاده‌سازی شده، مولفه‌های استاندارد یا مولفه‌های مجدد استفاده شده باشند. در رابطه با این اهداف اصلی، به مفاهیم اصلی همچون معماری و مولفه‌ها خواهیم پرداخت [۱].




شکل ۳- موضع‌گیری فاز معماری نرم‌افزار

### ۱-۲- فرآیند نگهداری

نگهداری نرم‌افزار مهمترین بخش توسعه محصول نرم‌افزاری و نیز مولفه‌ی اصلی همه مدل‌های چرخه عمر توسعه نرم‌افزار می‌باشد[۲]. فرآیند نگهداری خوب برای نگهداری کیفیت نرم‌افزار خیلی ضروری است. چندین نویسنده مدل‌های فرآیند متنوعی را برای نگهداری نرم افزار ارائه داده‌اند. این مدل‌ها به نگهداری نظام‌مند در رشته‌ای از فعالیت‌های مرتبط، یا فازها، و تعریف آرایه‌هایی که در این فازها اجرا می‌شوند، می‌پردازد. اساسا هفت فاز اصلی در فرآیند نگهداری وجود دارد که در زیر بیان می‌شود[ ۳]:

[۱] Controller Area Network (CAN)
[۲] Controller Area Network (CAN)

الف) مدیریت تغییر

فازی که در آن درخواست کاربر برای اصلاح، به یک مشتری، یک برنامه نویس، یا یک مدیر به طبقه‌بندی مدیریت، اولویت و شناسه انحصاری تخصیص داده می‌شود. این فاز همچنین فعالیت‌هایی را برای پذیرش یا رد درخواست برقرار می‌کند و یک مجموعه اصلاحات زمانبندی شده را برای پیاده‌سازی نگهداری جامع مشخصات اختصاص می‌دهد.

ب) تحلیل

این فاز اهداف پایه‌ای را برای طراحی، تست اجرا، و تحویل ترتیب می‌دهد. هدف اصلی تحلیل برای استنتاج امکان تغییر درخواست و تنظیم و پیاده‌سازی تغییر می‌باشد. تحلیل در دو سطح تنظیم شده است: امکان‌سنجی تحلیل و تحلیل تفصیلی. تحلیل تفصیلی راه حل‌های جایگزین و ارزیابی تاثیرات و هزینه‌ها را تعیین می‌کند، تحلیل تفصیلی ضروریات را برای اصلاح، طرح‌ریزی یک خط مشی آزمایشی و توسعه یک هدف پیاده‌سازی تعریف می‌کند.

پ) طراحی

تغییر در سیستم به طور واقعی در این فاز طراحی شده‌است. این امر تمامی سیستم‌های حاضر و مستندات پروژه‌ها، پایگاه داده و نرم افزار موجود و خروجی فاز تحلیل را به همراه دارد.  این فاز به گسترش یک منطق تجدیدپذیر و طراحی فیزیکی برای انجام و طراحی تغییرات در تمام طبقه بندی‌های نگهداری کمک می‌کند.

ت) پیاده سازی

این فاز شامل فعالیت کدنویسی و واحد آزمایش، ادغام کد شخصی سازی شده، یکپارچه سازی و تحلیل، آزمون رگرسیون، و خطرپذیری می‌باشد. همچنین یک مرور آمادگی- آزمایش را برای ارزیابی هوشیاری برای سیستم و آزمون رگرسیون شامل می شود.

ث) آزمایش سیستم / رگرسیون

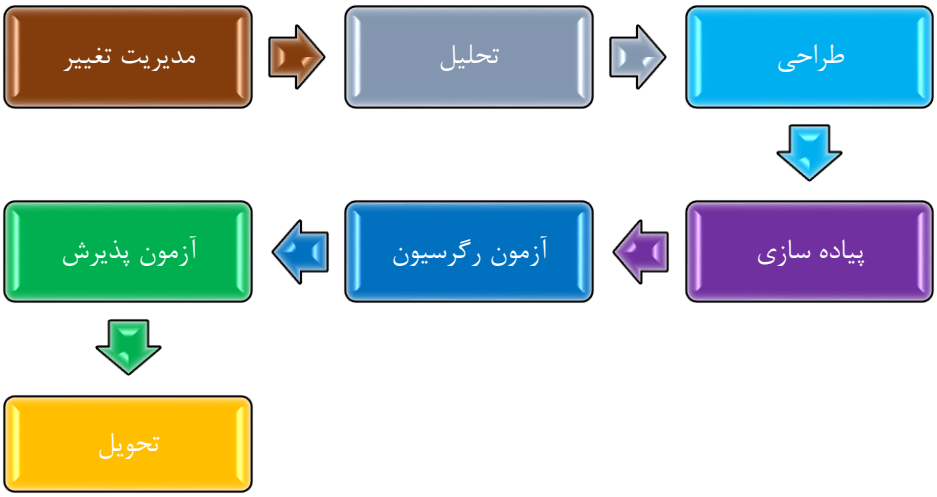
در این فاز سیستم کامل برای ساختن انطباق اصلی ضروریات جدید به علاوه تغییرات آزمایش می‌شود. به علاوه این فاز شامل آزمون رگرسیون برای آزمایش عملکرد و واسط می‌شود، که تصدیق می‌کند هیچ خطای جدیدی به سیستم اضافه نشده است. سرانجام، این فاز برای تایید هوشیاری برای پذیرش آزمایش پاسخگو می‌باشد.

ج) آزمون پذیرش

در این فاز از آزمایش با کاملا ثبت شده درک می‌شود  و کاربران، مشتریان، یا یک سوم شخص منتخب توسط مشتری را شامل می‌شود.

چ) تحویل

فازی که سیستم شخصی‌سازی‌شده برای دو مورد راه‌اندازی و عملیات بی‌قید می‌باشد. آن فعالیت مطلع کردن جامعه کاربر، انجام راه‌اندازی و آموزش را شامل می‌شود.




شکل ۴- فرآیند فاز نگهداری

### ۱-۳- انواع نگهداری نرم‌افزار

نگهداری نرم‌افزار به چهار نوع با درجه اهمیت یکسان تقسیم بندی می‌شود [۳]:

نگهداری اصلاحی[۱]: رایج‌ترین نوع نگهداری نرم افزار می‌باشد. تصحیح هر نوع خطا یا عیب درون نرم افزار که کارایی آن را چه در منطق، کد یا طراحی تحت تاثیر قرار می‌دهد. معمولا این نوع نگهداری در به روزرسانی‌های کوچک و مکرر برای ممانعت از بدترشدن مشکلات در آینده اجرا می‌شود.

نگهداری تطبیقی[۲]: به فرآیند انجام تغییرات در سیستم نرم‌افزاری و به روزرسانی‌ها برای سازگاری با این تغییرات گفته می‌شود. نگهداری تطبیقی سیستم را برای ادامه‌ی کار با چارچوب خاص هدایت می‌کند. معمولا با سیستم عامل، پایگاه‌داده، واسط کاربر، پروتکل‌های شبکه ای و چارچوب‌های سخت افزاری گوناگون در محاوره هستند.

نگهداری پیشگیری‌کننده[۳]: بررسی‌های منظم و به طور معمول برای ممانعت از خرابی‌های غیرمنتظره، پرهزینه و برنامه‌ریزی نشده می‌باشد. هدف آن کمک به ادامه‌ی کار نرم افزار تا زمان ممکن می‌باشد. این نوع، نرم افزار را در برابر تهدیدات جدید و نو ظهور ایمن نگه می‌دارد.

نگهداری تکمیلی[۴]: این نوع نگهداری بر بهبود نرم‌افزار با توجه به تجربه کاربر در کار با نرم افزار و در پاسخ به بازخورد مشتری می‌باشد. افزودن ویژگی‌ها یا ایجاد تغییرات در نرم‌افزار همانطور که در آینده کسب و کار تغییر یا تکامل پیدا می‌کند.

---

۱ Corrective maintenance
۲ Adaptive maintenance
۳ Preventive maintenance
۴ Perfective maintenance


شکل ۵– انواع نگهداری نرم‌افزار

**۱-۴- موضوعات و مشکلات**

بیشترین مشکلات که با نگهداری نرم افزار مرتبط هستند می‌تواند در کمبودهای فرآیند توسعه نرم افزار ردیابی شود. چندین مشکل مدیریتی و فنی در نگهداری نرم افزار وجود دارند، در زیر برخی از این مشکلات بیان شده است[۳]:

الف ) هزینه‌ها

مطالعات تحقیقی متنوعی نشان می‌دهد که هزینه نگهداری نرم افزار ۶۰ درصد تا ۸۰ درصد کل چرخه عمر توسعه را مصرف می‌کند. نتایج مطالعات بیان شده در [۳] نشان می‌دهد که هزینه‌های نگهداری به طور اصلی به بهبودها نسبت به اصلاحات منجر می‌شود.

ب) تحلیل تاثیرات

یکی از مهمترین چالش‌ها در نگهداری نرم افزار یافتن آثار یک تصحیح پیشنهادی روی باقی سیستم است. تحلیل تاثیرات عمل ارزیابی تاثیرات احتمالی یک تغییر با هدف کاهش اثر جانبی ناگهانی می‌باشد. وظیفه ای که ارزیابی درستی یک اصلاح برنامه ریزی شده و ارزیابی خطرهای مرتبط با تکمیل آن، به اضافه‌ی تخمین تاثیر روی ویژگی‌ها، انرژی و توسعه می‌باشد.

تغییرات تصحیح‌کننده

یکی از موضوعات کلیدی تغییرات تصحیح‌کننده است چرا که یافتن مکان صحیح برای انجام تغییرات سخت است. تعیین پایه کد می‌تواند مشکل باشد. اگر طراحی اولیه به تغییر دقیقه‌ای کاهش یابد، ممکن است تغییرات معماری پایه ای زمان زیادی ببرد. اگر یک راه‌حل کامل از یک مشکل وجود داشته باشد، با این حال رفع مشکل بعدی برای شکستن[۱] سخت‌تر است. طراحی خطاها برای تعمیر دشوار می‌باشد چرا که زمان زیاد و فهم پایه کد ورودی با مخاطرات مرتبط هستند.

تغییرات تطبیقی

[۱] crack

تغییرات تطبیقی اغلب به دلیل کمبود اطلاعات در مورد آنچه نرم اصلاح می‌شود، آسان نیست. حقایق متنوعی از فناوری جدید برای تطبیق نگهداری مشکل می‌باشد. همچنین تحلیل تاثیر و کشف واسط برای چیزهای جدید مشکل است. مشکلات طراحی مقدماتی نامتعادل موضوع مورد دغدغه می‌باشند.

فهم برنامه

موضوع کلیدی دیگر فهم برنامه می‌باشد که مقادیر بزرگی از زمان توسط مهندسین نگهداری، برای خواندن و فهم کد صرف می‌شود، مستندات مربوطه برای داشتن یک دیدگاه بهتر روی منطق نگهداری، هدف و ساختار نگهداری بخشی از نرم افزار و برای بهبود کیفیت نرم‌افزار است [۳].

در تحقیق حاضر به بررسی نرم‌افزارهای کاربردی و نگهداری آن‌ها پرداخته شده است و پژوهش‌های انجام شده در این حوزه مورد بررسی قرار گرفته است و در پایان این پژوهش ها بوسیله شاخص های بازه زمانی، نوع مقاله و محدوده موضوعی از لحاظ آماری تحلیل شده‌اند.

## ۲- مروری بر کارهای مرتبط

در مطالعات تجربی روی فرآیندها، تمرین‌ها و تکنیک‌های مهندسی نرم‌افزار، یادگیری ماشین و اتوماسیون محبوبترین هستند. به منظور استخراج دانش از پروژه‌های نرم افزاری، اغلب یک دنباله از تحلیل شبیه سازی با استفاده از متدولوژی های متفاوت، داده و فراداده انجام می‌شود، این تحقیق نظام‌مند ادبیات روی رویکردهای موجود از شبیه‌سازی، طبقه‌بندی و تحلیل مبتنی بر ارتباط روی نرم‌افزار کاربردی تمرکز دارد. به طور کلی، ۱۳۶ مقاله منتشر شده‌ی مرتبط و اختراع بین سال های ۲۰۰۲ تا ۲۰۱۹ بر طبق معیار شمول و حذف نشر، شناسایی شده‌اند، که به طور کلی یک محاسبه از شبیه‌سازی نرم‌افزار یا پشتیبانی از فازهای اصلی مهندسی نرم‌افزار انجام می‌دهد. بر این اساس یک استراتژی جستجوی مبتنی بر کامپیوتر به کار برده شده است که یک جستجوی خودکار در پایگاه‌داده های الکترونیکی را شامل می‌شود. یک تحقیق ادبی اولیه تحت مرتبط‌ترین منابع ادبی برای تحقیق ادبی نظام‌مند[۱] انجام شده است. مرتبط‌ترین کلمات کلیدی از سوالات تحقیق استخراج شده‌اند. تمام کلمات کلیدی موجود از کارهای پیشین در هر نشریه‌ای از جستجوی ادبی اولیه نیز استفاده نموده است. لیست کلمات کلیدی پس از تلفیق و تلخیص با اولویت‌سازی متناظر منطبق بر تعداد دفعات نتیجه می‌شوند. کلمات کلیدی مشابه و مترادف ادغام شدند. لیست نتایج کلمات کلیدی مورد جستجو بوسیله یک ابر[۲] کلمه ارائه می‌شوند. عناوین جستجو با استفاده از عملگرهای بولین AND، OR کلمات کلیدی موردنظر توصیف شده‌اند. نشریه‌ها بوسیله به کار بردن عناوین جستجوی توصیفی معین برای منابع ادبی منتخب بر طبق معیارهای خاص فیلتر شده‌اند. درحالی که معیار شمول دربردارنده کار مبتنی بر سوال تحقیق و ویژگی‌های مورد نیاز از نشریه می‌باشد، چندین معیار حذف برای فیلتر منطبق بر موضوعات ویژه و ویژگی‌هایی از کار تعیین شده‌اند. همانطور که در شکل ۶ مشخص است، در فرآیند SLR، در هر مرحله از فرآیند ابتدا معیار شمول به کار برده شده است، سپس در مرحله بعدی معیار حذف انجام می‌شود. در مرحله بعد فرآیند SLR بوسیله الگوریتم گلوله برفی رو به جلو و رو به عقب بسط داده شده است. گلوله برفی برای استفاده به لیست مرجع یا نقل‌قول ها برای کار موردنظر به‌منظور شناسایی سایر کارهای مرتبط اشاره می‌کند، روی نتیجه نهایی ۵۸ مقاله، روش گلوله برفی برنامه‌ریزی شده، انجام شده است. در سطح ۱، ۳۰۶۶ مقاله مرجع و استنادی بررسی شده‌اند و به منظور شمول و حذف، در سطح دوم تکمیل شده از روش گلوله برفی ۲۱۶۹ مقاله و در سطوح بعدی ۱۱۴۶

[۱] SLR(Systematic literature review)

[۲] Cloud

و ۴۸۶ مقاله مورد بررسی قرار گرفته‌اند. اولین جستجو با استفاده از عنوان جستجوی تعریف شده به ۳۳۸۷ قلم[۱] منتج و در چندین مرحله به ۵۸ مقاله مرتبط فیلتر شده است [۴].


شکل ۶- پردازش مراحل در فرآیند SLR

در شکل ۶ به منظور فیلترکردن، ابتدا موارد تکراری به طور خودکار براساس شماره‌های DOI موجود حذف می‌شوند. این امر بوسیله یک بررسی دستی از روی محتوای هر عنوان دنبال می‌شود. برای مقاله‌هایی که به طور واضح قابل استثنا نبودند، بررسی بیشتری با استفاده از چکیده ها انجام شده است و سپس به دلیل این‌که اغلب چکیده‌ها مبهم بودند، باقی مقاله‌ها به طور کامل بررسی شدند. این مقاله‌ها به طور کلی کیفیت متغیری را در زمینه مهندسی نرم‌افزار دارند. یکی از مشخصه‌های اصلی نرم افزار کاربردی دوره عمر طولانی آن از نگهداری فعال می‌باشد. تعداد کمی تحقیق مهندسی نرم افزار در مشخصه‌های توسعه نرم افزار علمی و در فاکتورهایی که تکامل طولانی موفقیت آمیز آن را پشتیبانی می‌کنند، وجود دارند [۴]. در تحقیق [۵].یک مدل جدید برای آزمودن ذات تغییر معرفی می‌شود که یک نمونه از نرم افزار علمی صنعت روی طول عمر آن تاثیر می‌گذارد. تحقیق مدل را برای فراهم کردن یک تحلیل موضوعی از فاکتورها فراهم می‌آورد که در تکامل طولانی مدت نرم‌افزار سیستمی مشارکت می‌کند. نتیجه طراحی معماری نرم‌افزار را پیشنهاد می‌دهد و مشخصه‌های گروه توسعه نرم‌افزار در تکامل موفقیت آمیز نرم‌افزار نقش اصلی را بازی می‌کند. مدل جدید تغییر و روش تحقیقی مستقل از نوع نرم افزار تحت بررسی برای این تحقیق توسعه داده شده است. نرم‌افزار استفاده شده در این تحقیق به طور موفقیت آمیز و فعالانه در دوره حدود ۲۰ ساله نگهداری می‌شود.

[۱] Item

مشخصه‌های نرم‌افزار و محیط آن شامل گروه‌های توسعه آزموده و مقیم طولانی‌مدت، رویکردهای مدیریتی محافظه‌کارانه برای توسعه نرم‌افزار، تعداد کمی مستندات همانند تغییر رکوردها، رشد نرم افزار پایدار و ثابت و نوعی سبک معماری نرم افزار علمی می‌باشد. در این تحقیق مشخصه‌های مذکور بسط داده شده‌اند [۵]. نرم افزار پیش‌بینی نقص (SDP) یادگیری ماشین را برای تعیین خطاها[۱] در کد منبع استفاده می‌کند. مجموعه داده استفاده شده برای SDP نوعا توسط موضوع عدم تعادل کلاسی تحت تاثیر قرار می‌گیرد. الگوریتم‌های یادگیری سنتی روی مجموعه داده‌های عدم تعادل کلاسی به خوبی اجرا نمی‌شوند. یادگیری حساس به هزینه در SDP برای کاهش هزینه‌های درآمد بوسیله پیش بینی‌ها ایجاد شده است. در این تحقیق یک چارچوب مولد پیش بینی‌های حساس به هزینه و نیز کاهش عدم تعادل کلاسی ارائه شده است. از این رو الگوریتم مربوطه یک دسته‌بند جنگل تصمیم را می‌سازد. دانش می‌تواند بوسیله بررسی خودکار درخت‌های تصمیم منفرد استخراج شود. برای بهبود این فرآیند کشف دانش، یک الگوریتم برای استخراج الگوهای علاقه‌ی بیشتر از یک جنگل تصمیم ارائه شده است. الگوریتم علاقه‌مندی را به عنوان سود مالی بالقوه از شناخت الگو محاسبه می‌کند. فرآیند ارائه شده، روش‌های مذکور را در پردازش کشف دانش حساس به هزینه end to end ترکیب می‌کند. این فرآیند بوسیله استخراج دانش از چهار پروژه نرم افزاری، توسط سازمان ملی هوانوردی و فضای آمریکا (NASA) ارائه شده است. مشارکت اصلی این تحقیق در جدول ۱ بیان شده است [۶].

جدول ۱- مشارکت تحقیق

| هدف | شرح |
|---|---|
| ساختن جنگل تصمیم | BCF: یک روش دسته بندی که یک کلاس متعادل شده و دسته بند حساس به هزینه را بدون تغییر مجموعه داده اصلی تولید می‌کند. |
| کشف دانش | SibCost: معیار علاقه‌مندی مجموعه قوانین حساس به هزینه. برای تعیین کمیت علاقه-مندی یک مجموعه قوانین با حساسیت به هزینه استفاده شده است. |
| کشف دانش | یک الگوریتم برای استخراج خودکار مجموعه قوانین علاقه‌مندی بیشتر. |
| کشف دانش | یک فرآیند برای کشف دانش حساس به هزینه در درخت تصمیم/جنگل‌ها. |
| ارائه کشف دانش | دانش حساس به هزینه: یافته‌های حاصل از روش این تحقیق روی پروژه‌های نرم‌افزاری ناسا |

، مرورگر وب به عنوان چارچوب هدف برای نرم افزار کاربردی در نظر گرفته شده است. برنامه‌های کاربردی سبک دسکتاپ همانند واژه پرداز ورد، صفحه گسترده، تقویم‌ها، بازی‌ها، و سیستم‌های پیام‌رسان فوری که پیشتر برای سیستم‌های عامل خاص نوشته شده بودند، معماری CPU یا دستگاه‌ها برای وب جهانی برای استفاده در مرورگر وب هم اکنون نوشته شده‌اند. در این تحقیق خلاصه ای از آزمایش استفاده مرورگر وب به عنوان چارچوب هدف برای نرم‌افزار کاربردی واقعی آورده شده است. به عنوان یک مثال واقعی SUNTM Labs Lively Kernel در این تحقیق استفاده شده است که سیستمی برای پیاده سازی یک محیط برنامه نویسی وب تعاملی مستثنای اجرایی در یک مرورگر وب بدون هر گونه مولفه پلاگین می‌باشد. بر پایه این کار، محدودیت ها، چالش‌ها، و فرصت‌ها مرتبط با مرورگر وب به عنوان یک چارچوب نرم افزار کاربردی تحلیل می‌شود [۷]. نگهداری و بهبود نرم افزارکاربردی سهم عمده ای از کل هزینه چرخه عمر یک سیستم را مصرف می‌کند. آمار تقریبی حدود

[۱] bug

مصرفی منابع برنامه‌نویسی و کل سیستم ها به میزان بالای ۷۵-۸۰ درصد در هر طبقه‌بندی می‌باشد. برای تحلیل مسئله در این حوزه یک نظرسنجی بسط داده و آزمایش شده است. سپس برای ۱۲۰ سازمان ارسال شده‌است. کل پاسخ ها ۶۹ می‌باشد. پاسخ ها با استفاده از بسته آماری SPSS تجزیه و تحلیل شده‌اند. تحلیل نتایج موارد زیر را مشخص نمود:

- نگهداری و بهبود مصرف تعداد زیادی از کل منابع سیستم و گروه های برنامه نویسی را در پی دارد.
- نگهداری و بهبود می‌بایست توسط مدیریت به عنوان یکی از حداقل موارد مهم نسبت به توسعه نرم افزار کاربردی جدید دیده شود.
- در نگهداری و بهبود، مسئله جهت‌گیری مدیریت نسبت به جهت‌گیری فنی ارجحیت دارد.
- مطالبات کاربر برای بهبود و گسترش(افزونه‌ها) مهمترین حوزه مسائل مدیریت را تشکیل می‌دهد [۸].

مشکلات نگهداری نرم افزار کاربردی در ۴۸۷ سازمان پردازش‌داده بررسی شده‌اند. تحلیل فاکتورها به شناسایی ۶ فاکتور مشکل منتج می‌شود: دانش کاربر، اثربخشی برنامه نویس، کیفیت محصول، دسترس پذیری زمان برنامه نویس، نیازهای ماشین و قابلیت اطمینان سیستم. دانش‌کاربر برای حدود ۶۰ درصد واریانس مسائل رایج را تشکیل می‌دهد. شواهد جدید اهمیت رابطه کاربر را برای موفقیت یا شکست سیستم ارائه می‌کند. مسائل اثربخشی برنامه‌نویس و کیفیت محصول برای سیستم‌های بزرگتر و قدیمی‌تر بیشتر بودند و نیز در جایی که تلاش‌های بیشتری برای نگهداری صحیح صرف شده بود. محیط پردازش داده مقیاس بزرگتر به طور معناداری با بیشتر مسائل اثربخشی برنامه‌نویس مرتبط بودند، اما نه با فاکتور مسائل دیگر. زمانی که تکنیک های بهره‌وری در توسعه استفاده شدند، کیفیت محصول به عنوان مسائل کوچکتری دیده شدند. مسائل عمده در نرم‌افزار کاربردی چه مواردی هستند و چگونگی انجام این مسائل متفاوت بر طبق سیستم نرم‌افزار کاربردی نگهداری شده و محیط پردازش داده و این که نگهداری چه جایگاهی دارد؟ این تحقیق نتایجی را گزارش می‌دهد که برخی از این سوالات پاسخ داده است. نتایج مبتنی بر بررسی ۲۰۰۰ نمونه مدیریت سازمان پردازش داده می‌باشد. هر مدیر به طور تصادفی از اعضای انجمن مدیریت پردازش داده انتخاب شده‌اند (DPMA) و یک پرسشنامه روی نگهداری نرم‌افزار کاربردی ارسال شده است. تعداد کل ۴۸۷ پاسخ‌ها دریافت شدند. نرخ بازگشت ۲۴ درصد با توجه به ماهیت درخواستی پرسشنامه خوب در نظر گرفته شده‌اند. هدف کلی تحقیق استخراج پنج مجموعه از موضوعات مرتبط با نگهداری می‌باشد: الف. موضوعات مفهومی (معنی نگهداری) ب. موضوعات مقیاس تلاش (سطح منابع و تخصیص) پ. موضوعات سازمانی (تعریف وظایف و انتساب) ت. موضوعات کمک به بهره‌وری (تاثیر بر نگهداری) و ث. موضوعات مربوط به حوزه‌ی مسئله (بهبود نسبی در نگهداری). این موضوعات از یک مرور ادبیات نگهداری توسعه داده شده‌اند[۹]. روابط میان تعداد زیادی معیارهای پیچیدگی نرم‌افزار، این معیارها را به عنوان ابزار بدون جایگزین مدیریت پروژه ساخته است. در این تحقیق، مفهومی از تک معیار توسعه داده شده است که پیچیدگی نسبی نامیده می‌شود. بدین صورت که به هر مسئله‌ای در مجموعه ای از مسائل بوسیله پیچیدگی آن‌ها، تک مقدار نسبت داده می‌شود. برای یک مجموعه داده تستی، پیچیدگی نسبی برنامه برای ۲۷ برنامه ایجاد شده است. این داده پیچیدگی نسبی در رابطه با زمان هر برنامه در فاز اشکال زدایی تست شده است. یک رابطه معنادار بین پیچیدگی نسبی و زمان اشکال زدایی برنامه‌نویسی به عنوان یک معیار از تلاش درک شد. واضح است که معیار پیچیدگی نسبی در کل فرآیند طراحی، ثابت است و می‌تواند به عنوان یک شاخص اصلی برای مجموعه‌ای از برنامه‌ها به کار گرفته شود که به مقادیر بزرگی از منابع سیستم در طول فازهای نگهداری و توسعه نیاز خواهد داشت. هدف اصلی تحلیل فاکتور برای توصیف، امکان روابط کوواریانس در میان معیارهای پیچیدگی از نظر چند مقدار

اساسی اما قابل فهم، مقادیر تصادفی فاکتور نامیده می‌شود. اساسا فاکتور مدل بوسیله آرگومان بعدی برانگیخته می‌شود. معیارهای پیچیدگی فرضی بوسیله همبستگی آن‌ها گروهبندی می‌شود. تمام معیارها درون یک گروه خاص قرار دارند که به میزان بالای همبستگی را درون خودشان دارند اما به طور نسبی با معیارهای گروه‌های متفاوت نیز همبستگی دارند. قابل تصور است که هر گروه از متغیرها نمایانگر تک‌ساختار زیربنایی یا فاکتور مسئول همبستگی‌های مشاهده شده است. رویه تحلیل فاکتور به عنوان پایه برای ساخت معیار پیچیدگی نسبی به کار گرفته خواهد شد. تحلیل فاکتور به عنوان یک افزونه از تحلیل مولفه اصلی در نظر گرفته می‌شود، اما به عنوان تلاشی برای تخمین ماتریس کوواریانس دیده می‌شود. نکته اساسی از پیچیدگی نسبی نمایانگر تک معیار واحد روی ساختار یک برنامه است و معیار نسبی که برای دسته‌بندی یک مجموعه از برنامه به منظور افزایش پیچیدگی در رابطه با یکدیگر به کار گرفته می‌شود. از نقطه نظر مدیریت پروژه برنامه‌های افزایش پیچیدگی به طور اصلی آن‌هایی هستند که منابع بیشتری را در طول چرخه عمر توسعه مصرف خواهند کرد. در واقع روابط بین پیچیدگی نسبی یک برنامه و تعداد خطاهایی یافته شده در طول فاز تست تخمین زده شده‌اند. همچنین این معیارهای یکسان از پیچیدگی به طور نزدیکی با معیار کیفیت نرم افزار در رابطه است. تشخیص زودهنگام ماژول‌های پیچیدگی نسبی برنامه می‌تواند به عنوان ابزاری برای ایجاد برنامه‌های ساده شده و مجدد طراحی شده مورد استفاده قرار گیرد. در صورت ایجاد به طور مثال، یک مقدار نامتناسب از زمان مصرفی در طول فاز اشکال زدایی روی تعداد کمی پیچیدگی نسبی برنامه‌ها، معیارهای پیچیدگی نسبی به عنوان شاخص اصلی خوبی از برنامه‌ها به کار گرفته خواهد شد که ممکن است منابع پرسنلی بزرگ غیرضروری را مصرف کند[۸]. کشف دانش از داده معیار مهندسی نرم افزار در اخذ نتایج درست از آزمایش‌ها ضروری می‌باشد. روش‌های تحلیل داده متنوع می‌تواند تحلیل های داده ای را با دید تکمیلی پدیده ها را مطالعه و بررسی نماید. در این تحقیق، دو روش تحلیل داده رگرسیون لجستیک و مجموعه های ناهموار از دو نقطه نظر تئوری و آزمایشی مورد مقایسه قرار می‌گیرند. به ویژه، مطالعه تجربی به عنوان بخشی از پروژه ESPRIT/ESSI (CEMP) روی یک پروژه نگهداری چرخه عمر واقعی انجام می‌شود [۱۰]. به عنوان یک هدف در نگهداری مهندسی نرم افزار، ردیابی تکرار در سیستم های نرم افزاری بزرگ می‌تواند به صورت کارایی مفید باشد. تکرار در کد اغلب شکلی از بخش‌های کد برای یک تغییر سیستماتیک از پارامترها همانند شناسه‌ها و ثابت‌ها یکسان می‌باشند. برای مدلسازی چنین تکرارهای پارامتری در کد، این تحقیق اهمیت رشته‌های پارامتری و تطبیق پارامتری از رشته های پارامتری معرفی می‌کند. یک ساختار داده یک درخت پسوندی پارامتری تعریف شده برای کمک در جستجوی تطبیق پارامتری را فراخوانی می‌کند. برای الفبای ثابت الگوریتم‌ها برای ساختن یک درخت پسوندی پارامتری در زمان خطی و برای یافتن بیشترین تطبیق‌های پارامتری روی یک طول آستانه در یک p رشته پارامتری در زمان خطی در اندازه ورودی به علاوه تعداد تطبیق های گزارش شده، داده می‌شوند. نتایج پیاده سازی شده‌ی الگوریتم‌ها نشان می‌دهند که به خوبی روی کد c کار می‌کنند [۱۱]. این تحقیق یک مطالعه تجربی در رابطه با نگهداری نرم‌افزار کاربردی جاوا مبتنی بر وب و به هدف فهم و پیش بینی طبقه‌بندی نگهداری نرم افزار و تلاش‌های مرتبط به آن ارائه می‌دهد. نرم افزار کاربردی ویژه در نرم افزار کاربردی مدیریتی جاوا مبتنی بر وب در عرصه دولت الکترونیک توصیف می‌شود. نرم افزار کاربردی مبتنی بر طراحی یک چارچوب model-controller-view منبع باز می‌باشد. فاکتورهای دامنه، که همچنین منابع متنی را فراهم می‌کنند روی هستی‌شناسی نگهداری نرم‌افزار Kitchenham و دیگران توصیف می‌شوند. این تحقیق تعداد گزارش‌های خطا و تلاش های نگهداری را برای هر طبقه‌بندی نگهداری را مشخص می‌کند. نتایج تحقیق نشان می‌دهد که توزیع تلاش نگهداری نرم افزار در نرم افزار کاربردی جاوا مبتنی بر وب مشاهده شده مشابه به مطالعات نگهداری نرم افزار قبلی برای توزیع است، بدین صورت که غیر شی‌گرا و غیر مبتنی بر وب تحلیل می‌شود

[۱۲]. در این تحقیق، از الگوریتم‌های دسته‌بندی انجمنی برای تخمین کارایی دسته‌بندهای متفاوت به منظور دسته‌بندی بازبینی-های در چندین وظیفه نگهداری استفاده می‌شود. همچنین یک رویکرد الگوریتم دسته‌بند جدید برای بازبینی کاوی (ARCM) پیشنهاد می‌شود. دسته‌بندی بازبینی به مراحل پیش‌پردازش برای به کار بردن پیش‌پردازش زبان طبیعی و تحلیل متن نیاز دارد. همچنین تاثیر روش های انتخاب دو ویژگی (بدست آوردن اطلاعات و توزیع مربع کای[۱]) روی دسته‌بندها مطالعه می‌شود. قوانین انجمنی فهم بهتری از علاقه کاربران می‌دهند از این رو آن ها الگوهای پنهان در کلمات را استخراج می‌کنند و ویژگی‌هایی که با یکی از وظایف نگهداری مرتبط هستند، و آن را به عنوان کلاس قوانین انجمنی (CARs) ارائه می‌دهد.. نتایج نشان می‌دهد که بالاترین صحت بوسیله الگوریتم های AC برای دو مجموعه داده بدست آورده می‌شود. ACRM بالاترین دقت، فراخوانی، امتیاز F و صحت را دارا می‌باشد. انتخاب ویژگی به طور معناداری کارایی دسته بند را بهبود می‌بخشد [ ۱۳].

در [۱۴] چندین متغیر به عنوان واسطهایی برای نگهداری سیستم نرم‌افزاری در کل توسعه نرم افزار استفاده می‌شوند. هدف این مطالعه بررسی سازگاری این متغیرها با یکدیگر و چگونگی تلاش‌های نگهداری در مرحله سیستم می‌باشد. نگهداری محصولات نرم افزاری شامل بررسی خطا، عملکرد مناسب سیستم، بهبود کارایی، تطبیق پذیری سیستم و پیگیری در خواست تغییری که از سمت مشتری می‌آید. نگهداری نرم افزار مرحله غیرقابل جایگزین در چرخه عمر توسعه نرم‌افزار [۲] است و طولانی‌ترین، پیچیده‌ترین و پرهزینه‌ترین مرحله است. فاز نگهداری زمانی شروع می‌شود که نرم‌افزار به کاربر نهایی تحویل داده می‌شود و هنگامی که نرم افزار از سرویس خارج می‌شود، عمر آن پایان می‌یابد. در تحقیق [۱۵] مدل‌های نگهداری نرم افزار بررسی می‌شود و مدلی برای نگهداری نرم‌افزار در محیط محاسبات ابری پیشنهاد می‌شود، ویژگی‌های ابر فرآیند نگهداری را تسهیل می‌کند. بنابر تحقیق[۱۶] نگهداری نرم‌افزار یک حوزه پژوهشی گسترده در مهندسی نرم‌افزار که از اهمیت ویژه‌ای در برنامه‌های کاربردی بحران – ماموریت و بحران – ایمنی برخوردار می‌باشد. در این گونه برنامه‌های کاربردی نقایص می‌توانند تاثیر زیادی داشته باشند و کد باید به دقت از طریق تحلیل داده بررسی شود. این داده با استفاده از تعدادی ابزارهای توسعه داده شده به منظور برآورد جنبه‌های خاص جمع‌آوری می‌شوند. به هر حال چنین ابزارهایی اغلب متخصصین را از به کار بردن رویکردهای پیشرفته و جدید برای پروژه های صنعتی باز می‌دارد. تحقیق [۱۶] یک بررسی اولیه درباره ابزار تحلیل کد مورد استفاده برای انجام مطالعات تحقیقی روی پیش‌بینی نگهداری نرم‌افزار ارائه می‌دهد. تمرکز این تحقیق بر شناسایی ابزار در دسترس و قابل استفاده توسط متخصص برای به‌کار بردن رویکردهای نگهداری یکسان توصیف‌شده در تحقیق‌های دانشگاهی منتشر شده می‌باشد. عناوین به کار برده شده برای شناسایی ابزار در جدول ۲ ارائه شده است [۱۶].

جدول ۲- عناوین جستجو

| | |
|---|---|
| P: نگهداشت‌پذیری نرم‌افزار[۳] | عناوین P1 : "نرم‌افزار", "نگهداشت‌پذیری", "نگهداری", "پیمانه‌ای بودن" , "قابلیت استفاده مجدد" ,"قابلیت تجزیه و تحلیل" ,"اصلاح‌پذیری " ,"آزمون پذیری" ,"قابلیت اطمینان" |
| I: ابزار | عناوین I1 : "ابزار*" ,"تحلیل ایستا" ,"تحلیل پویا" |

[۱] کای دو

[۲] SDLC

[۳] Maintainability

هر چند در بخش‌هایی از دانش مهندسی نرم‌افزار در زمینه خودکارسازی فرآیندها، ابزارها و روش‌ها پیشرفت‌های بزرگی صورت گرفته است، اما بخش‌هایی همچون نگهداری نرم افزار بوسیله صنعت و دانشگاه کمتر مورد توجه قرار گرفته است. بنابراین حل و فصل وظایف فنی یا سرمایه انسانی به صورت دستی یا نیمه‌خودکار انجام می‌شود. در این زمینه تکنیک‌های یادگیری ماشین[۱] نقش مهمی در بهبود فرآیندهای نگهداری و شیوه‌های خودکارسازی[۲] ایفا می‌کند، این امر می‌تواند مراحل واگذاری وظایف را تسریع ببخشد. هدف تحقیق [۱۷] حصول درک عمومی از وضعیت نگهداری نرم‌افزار مبتنی بر یادگیری ماشین با استفاده از ترجمه، دسته‌بندی و تحلیل یک مجموعه از مطالعات مرتبط با موضوع می‌باشد. این مطالعه با به‌کاربردن پروتکل مطالعه نگاشت نظام‌مند صورت گرفت، که بوسیله استفاده یک مجموعه از مراحل تقویت تکثیر، مشخص می‌شود. این بررسی تعداد کل ۳۷۷۶ مقاله تحقیقی را شناسایی نموده است که در چهار مرحله فیلتر و سرانجام ۸۱ مقاله برای تحلیل انتخاب شده است. نتایج فراوانی کاربرد شبکه‌های عصبی را برای نگهداری پیشگیرانه و نیز مطالعات موردی نقش یادگیری ماشین را در موضوع مدیریت نگهداری و مدیریت افرادی که این وظایف را انجام می‌دهند[۱۷]. روش‌های [۳] چابک[۴] به دلیل ویژگی‌های نرم‌افزاری اخیرا محبوبیت زیادی بدست آورده‌اند. علی‌رغم موفقیت روش‌های چابک در فرآیند نگهداری نرم‌افزار، چالش‌های متعددی در این روش وجود دارد. در این تحقیق به بررسی این چالش‌ها و تاثیر روش‌های چابک در نگهداری نرم‌افزار از لحاظ فاکتورهای کیفیت پرداخته شده است و سپس یک نظرسنجی از اجراکنندگان روش چابک به منظورجمع‌آوری داده و انتشار نظرات آن‌ها در مورد چالش‌های موجود صورت گرفت. با توجه به تجزیه و تحلیل آماری داده‌های نظرسنجی، چالش‌هایی با اثربخشی متوسط در مدیریت‌پذیری، مقیاس‌پذیری، ارتباطات، همکاری و شفافیت[۵] وجود دارند[۱۸]، چالش‌ها در محیط محلی در جدول ۳ و چالش‌ها در محیط عمومی در جدول ۴ ذکر شده‌اند:

جدول ۳– چالش‌ها در محیط محلی [۱۸]

| چالش‌های[۶] نگهداری روش چابک | دسته‌بندی چالش‌ها بر طبق فاکتورهای کیفیت |
|---|---|
| توسعه تکراری | شفافیت: عدم شفافیت در یک پروژه منجر به دید مبهم راجع به پروژه می‌شود. |
| | مدیریت‌پذیری: عدم مدیریت‌پذیری منجر به اسپرینت[۷] می‌شود. |
| | زیرساخت نرم‌افزاری: عدم زیرساخت، بر فرآیند نگهداری و توسعه به طور مکرر تاثیر می‌گذارد. |
| تمرکز بر اهداف کاری | شفافیت: عدم شفافیت در پروژه منجر به دید نامشخص راجع به پروژه می‌شود. |
| همکاری نزدیک تیمی | شفافیت: عدم شفافیت در پروژه می‌تواند همکاری تیمی را تحت‌تاثیر قرار دهد. |
| | همکاری: عدم همکاری ناشی از کار تیمی ضعیف |
| مشارکت نزدیک مشتریان | همکاری و ارتباطات: عدم همکاری و ارتباطات بین مشتریان و تیم‌ها می‌تواند کیفیت پروژه را تحت‌تاثیر قرار دهد. |

[۱] Machine Learning (ML)
[۲] Automation
[۳] Methods
[۴] Agile
[۵] Transparency
[۶] Challenges
[۷] Sprint

| | |
|---|---|
| ارتباطات حضوری<br>وضوح مستندات | مدیریت‌پذیری: عدم مدیریت‌پذیری می‌تواند منجر به تضاد بین نیازهای مشتریان و تیم نگهداری شود. |
| | همکاری و ارتباطات: عدم همکاری و ارتباطات بین مشتریان و تیم‌ها می‌تواند کیفیت پروژه را تحت‌تاثیر قرار دهد. |
| | مدیریت‌پذیری: عدم مدیریت‌پذیری می‌تواند منجر به تعارض بین نیازهای مشتریان و تیم نگهداری شود. |
| آزمایش‌های مکرر | مقیاس‌پذیری: عدم مقیاس‌پذیری ممکن است فرآیند آزمایش‌های مکرر را که فاکتور اساسی در فرآیند نگهداری در نظر گرفته می‌شود، تحت‌تاثیر قرار دهد. |
| | زیرساخت نرم‌افزاری: عدم ارائه‌ی زیرساخت آزمایش‌های مکرر را تحت‌تاثیر قرار می‌دهد. |
| انگیزش از طریق مالکیت جمعی[1] | ارتباطات و همکاری: عدم همکاری و ارتباطات بین تیم‌ها مالکیت جمعی را باز می‌دارد. |
| انتقال دانش از طریق باز بودن[2] | همکاری و ارتباطات: عدم همکاری و ارتباطات بین تیم‌ها انتقال دانش را باز می‌دارد. |
| | شفافیت: عدم شفافیت در یک پروژه احتمالا بر انتقال دانش در میان اعضای تیم نگهداری تاثیر می‌گذارد. |

جدول ۴- چالش‌ها در یک محیط عمومی [۱۸]

| **چالش‌های نگهداری روش چابک** | **دسته‌بندی چالش‌ها بر طبق فاکتورهای کیفیت** |
|---|---|
| چالش های مرتبط با ارتباطات | ارتباطات: عدم ارتباطات بین تیم‌ها منجر به ایجاد مشکل در همکاری درون یک محیط عمومی خواهد شد. |
| | مدیریت‌پذیری: عدم مدیریت‌پذیری منجر به تعارض در انجام وظایف خواهد شد. |
| تمرکز بر اهداف کاری | مدیریت‌پذیری: عدم مدیریت‌پذیری منجر به کنترل ضعیف پروژه خواهد شد. |
| | شفافیت: عدم شفافیت در یک پروژه احتمالا درجه و کیفیت کنترل روی پروژه را تحت تاثیر قرار می‌دهد. |
| همکاری نزدیک تیمی | ارتباطات: عدم ارتباطات منجر به عدم اعتماد خواهد شد. |
| | همکاری: عدم همکاری بین تیم ها منجر به عدم اعتماد خواهد شد. |
| | مدیریت پذیری: نظارت بیش از حد[3] می‌تواند به منجر به کاهش اعتماد شود. |

تحقیق در مورد تحلیل بصری[4] برای نگهداری نرم‌افزار در سال‌های اخیر رشد چشمگیری داشته‌است. برای بسیاری از پروژه های نرم‌افزاری، نگهداری نرم‌افزار به مسیر موثر و کارآمد از داده تا تصمیم نیاز دارد. تحلیل بصری (VA) ایجاد چنین مسیری،

[1] Collective ownership
[2] Openness
[3] Excessive monitoring
[4] Visual analytics

کاربر را قادر می‌سازد، دیدها را بوسیله تعامل با اطلاعات مرتبط استخراج کند. تحقیق [۱۹] روی VA در نگهداری نرم‌افزار تمرکز دارد و اهداف زیر را دنبال می‌کند:

- بررسی تطبیق
- پیشنهاد مفاهیم برای عملکرد
- شناسایی روندهای جستجوی جاری
- مسائل باز
- زمینه‌های بهبود

به منظور حصول نتایج یک بررسی سازمان یافته با سه سوال پژوهشی و ۵۱ مطالعه منتشرشده در دو دهه گذشته انجام شده است. نتایج عدم همکاری بین محققین دانشگاهی و متخصصین صنعتی را مشخص می‌کند. به دلیل عدم اطمینان در پذیرش صنعت، اعتبار ابزارها و روش‌های پیشنهادی سلب می‌شود. بعلاوه، در این مطالعه نیاز به گسترش VA برای زبان‌های برنامه-نویسی دیگر و وظایف نگهداری نرم‌افزار شناسایی می‌شود[۱۹]. سیستم‌های نرم‌افزاری در جامعه گسترش یافته و نقش حیاتی در زندگی روزمره دارند و نیز به طور روزافزون پیچیده‌تر می‌شوند. نگهداری نرم افزار یکی از مهمترین فعالیت‌های توسعه نرم افزار می‌باشد. طبق برآوردها بیش از ۷۰ درصد کل هزینه یک سیستم نرم‌افزاری صرف فعالیت های نگهداری می‌شود. در کل این هزینه می‌تواند با استفاده از یادگیری ماشین(ML) و هوش مصنوعی (AI) کاهش داده شود. هوش مصنوعی می‌تواند در مراحل متنوعی از نگهداری نرم‌افزار استفاده شود، به عنوان مثال برای شناسایی عیوب طراحی در مراحل اولیه یا کد و به منظور استفاده مجدد. در تحقیق[۲۰] مقادیر بزرگی از داده تحلیل می‌شود و الگوها با روشی کارآمد بوسیله الگوریتم‌های یادگیری ماشین شناسایی می‌شود و نیز دو مطالعه یادگیری عمیق برای تشخیص بوی‌کد[۱] و یادگیری ماشین برای استفاده مجدد کد در طول نگهداری نرم‌افزار گزارش داده می‌شود. تشخیص بوی کد به روش‌های شناسایی عیوب موجود در کد اطلاق می‌شود. هدف اصلی تشخیص بوی کد، بهبود خوانایی، قابلیت تغییرپذیری، قابلیت توسعه و کیفیت کد می‌باشد. نگهداری نرم افزار، نام کلی برای مجموعه ای از فعالیت‌هایی است که پس از انتشار نرم افزار برای استفاده عملیاتی آن انجام می‌شود. عدم توانایی در تحقق نگهداری نرم‌افزار منجر به انبوهی تغییرات در برنامه کاربردی می‌شود. حفظ دوام یک سیستم نرم‌افزاری بزرگ در سطح کیفیت قابل قبول، یک چالش بزرگ است. در تحقیق [۲۱] مدیریت فرآیند نگهداری نرم‌افزار با تاکید بر زمینه سازماندهی و با هدف پیاده‌سازی مدیریت از نظر روش‌های موجود و ابزار پشتیبانی برای این روش‌ها بحث می‌شود. چهار مرحله پایه در مدیریت فاز نگهداری عبارتند از:

۱. فهم نرم‌افزار
۲. تعیین علت نیاز به تغییر و روشی که استفاده خواهد شد.
۳. پیاده‌سازی تغییر
۴. اعتبارسنجی تغییر

تحقیق [۲۲] راجع به توسعه نگهداری الکترونیکی[۲] برنامه کاربردی موبایل می‌باشد و این برنامه کاربردی برای شرکتی که به تعداد زیادی کارهای زمانبندی شده نیاز دارد، ساخته شده است. اهداف اصلی، طراحی و توسعه سیستم نگهداری الکترونیکی

[۱] Code Smell

[۲] E-Maintenance

هستند که شامل اندروید و وب‌سایت می‌باشند. حوزه اصلی برای این سیستم، شرکت Mega Auto & Electrical Engineering می‌باشد که برنامه کاربردی طراحی شده را استفاده می‌کند، این برنامه به مدیریت اجازه می‌دهد سفارش کار را برای کاربر به منظور پیگیری ایجاد کند. روش اصلی در این تحقیق مدل آبشاری است که در آن مصاحبه‌ها[۱]، نمودارهای زبان مدل‌سازی یکپارچه[۲] رسم و نمونه‌سازی[۳] انجام می‌شود. مرحله بعد از متدولوژی، تحلیل و طراحی می‌باشد به طوری که واسط نگهداری الکترونیکی ساخته می‌شود. بعد از ساخت نمونه اولیه، کار با پیاده سازی ادامه می‌یابد. نیازهای عملکرد در نگهداری الکترونیکی در جدول ۵ ذکر شده است:

جدول ۵- نیازهای عملکرد در نگهداری الکترونیکی [۲۲]

| ماژول[۴] | عملکردها[۵] |
|---|---|
| ورود به سیستم[۶] | کاربر با وارد نمودن نام کاربری و رمز عبور صحیح قادر به ورود به سیستم است. در صورت ورودی نامعتبر پیغام خطا نمایش داده می‌شود. |
| اعلان[۷] | یک روز قبل از تاریخ نگهداری، اعلان نمایش داده می‌شود. |
| دستور کار[۸] | دستور کار را می‌توانید از طریق ایجاد، ویرایش، و حذف مدیریت کنید. کاربر فقط می‌تواند بازدید داشته باشد و تاریخ و زمان و وضعیت را ویرایش نماید. |
| کاربر | مدیریت می‌تواند کاربران را از طریق ایجاد، ویرایش و دید و حذف مدیریت کند. |
| تکنسین[۹] | مدیریت می‌تواند تکنسین ها را از طریق ایجاد، ویرایش، دید و حذف مدیریت کند. |
| وسایل نگهداری[۱۰] | مدیریت وسایل نگهداری را از طریق ایجاد، ویرایش، دید و حذف میسر می سازد. |

ابزارهای نگهداری نرم افزار به منظور افزایش نرخ موفقیت سیستم های نرم افزاری در قرن گذشته توسعه یافته است. بهبود مدل‌های کیفیت ابزارهای نرم افزاری و دیگر عناصر نرم‌افزاری جلب رضایت مشتری و دستیابی به پایداری مشتری را در پی خواهد داشت. چندین مدل کیفیت و متغیرها به منظور کاهش خطا و پیچیدگی سیستم نرم‌افزاری و نیز مدل‌های کیفیت نرم‌افزار برای ارزیابی انواع عمومی و خصوصی محصولات نرم‌افزاری پیشنهاد شده است. این مدل‌ها برای تعیین حوزه‌های عمومی و

[۱] Interview
[۲] Unified Model Language (UML)
[۳] Wireframe
[۴] Module
[۵] Functionalities
[۶] Login
[۷] Notification
[۸] Work Order
[۹] Technician
[۱۰] Maintenance Devices

خصوصی محصولات نرم‌افزاری پیشنهاد می‌شوند. هیچ یک از این مدل‌های کیفیت، مربوط به کیفیت ابزارهای نگهداری نرم‌افزار نمی‌شود. مدل‌های نگهداری کیفیت نرم‌افزار مبتنی بر فاکتور ابزارهای نگهداری و مقایسه بین مدل‌های کیفیت رایج توسعه داده شده است. روش پیشنهاد شده در تحقیق [۲۳] برای نگهداری ابزارهای نگهداری نرم‌افزار به کار برده می‌شود. خروجی روش پیشنهاد شده نشان می‌دهد که می‌بایست دوازده فاکتور برای افزایش کیفیت ابزارهای نگهداری نرم‌افزار در نظر گرفته شود. این فاکتورها عبارتند از: آموزش[۱]، مشاوره‌ای[۲]، ارزیابی[۳]، بازسازی[۴]، به روزرسانی[۵]، ترتیبی[۶]، پیشگیرانه[۷]، کارایی[۸]، تطبیقی[۹]، تقلیل دهنده[۱۰]، بازگرداننده[۱۱]، تقویت کننده[۱۲] [۲۳]. افزایش سیستم‌های محاسباتی، دغدغه‌ها را درمورد ویژگی‌هایی همچون کارایی، در دسترس بودن، قابلیت اطمینان، و ظرفیت نگهداری برجسته کرده است. این ویژگی‌ها می‌تواند بر کیفیت سرویس تاثیر گذارد و شکست در فرآیند توسعه نرم‌افزار ممکن است این ویژگی‌ها را تحت تاثیر قرار دهد. کد معیوب و طراحی نادرست کلی نرم افزار، خطاهای داخلی و نقص عملکرد سیستم را به همراه دارد. برخی خطاها در طول فرآیند تست شناسایی و تصحیح می‌شوند. سایر خطاها ممکن است فقط در مرحله تولید ظاهر شوند. این مورد پدیده پیری نرم افزار[۱۳] است که منجر به تخریب پیشرونده کارایی یا قابلیت اطمینان نرم‌افزار در طول عمر عملیاتی آن می‌شود. تحقیق [۲۴] روشی را برای نگهداری نرم افزار به منظور شناسایی، تصحیح، و کاهش اثرات پیری نرم‌افزار ارائه می‌دهد. اگر کد منبع قابل اصلاح باشد و یک نسخه جدید با کمترین تاثیر توسعه داده شود، در این صورت داده تشخیص پیری برای نگهداری اصلاحی استفاده شود. اگر نرم افزار قابل تعمیر نباشد یا نسخه به روزرسانی شده‌ی آن، بدون وقفه طولانی سیستم باشد یا سایر پیامدهای بد، در این صورت رویکرد پیشنهادی [۲۴] می‌تواند در یک نگهداری پیشگیرانه برای اجتناب از قطعی سرویس، اثرات پیری را کاهش دهد. متدولوژی پیشنهادی [۲۴] از طریق مدل‌های شبکه پتری[۱۴] و آزمایش‌ها در یک محیط کنترل شده، اعتبارسنجی می‌شود. ارزیابی مدل یک روال نگهداری ترکیبی (پیشگیرانه و اصلاحی) با دسترس‌پذیری ۹۹,۸۲٪، زمان توقف سالیانه ۱۵,۹ ساعت در نظر می‌گیرد. در مقابل، سناریوی پایه تنها شامل نگهداری واکنشی (یا به عبارت دیگر تعمیر فقط بعد از شکست) بیشتر از ۱۳۴۲ ساعت زمان توقف سالیانه داشت – یعنی ۸۰ برابر بیشتر از رویکرد پیشنهادی در [۲۴]. هدف سازمان‌های نرم‌افزاری، توسعه و نگهداری سیستم‌های نرم‌افزاری با کیفیت بالا است. با توجه به مقدار زیاد داده‌های رفتاری در دسترس، سازمان‌های نرم‌افزاری می‌توانند نگهداری نرم‌افزار داده‌محور را انجام دهند. در واقع، برنامه‌های بهبود و تضمین کیفیت توجه تعداد زیادی از محققین را به خود جلب کرده است.

---

[۱] Training
[۲] Consultive
[۳] Evaluative
[۴] Reformative
[۵] Updative
[۶] Groomative
[۷] Preventive
[۸] Performance
[۹] Adaptive
[۱۰] Reductive
[۱۱] Corrective
[۱۲] Enhancive
[۱۳] Software Aging Phenomenon
[۱۴] Petri Net (SPN)

در تحقیق[۲۵] شبکه‌های بیزین[۱] (BNs) به عنوان یک تکنیک تحلیل ثبت وقایع[۲] برای کشف شاخص‌های عملکرد ضعیف در یک سیستم پیشنهاد می‌شوند و برای بررسی الگوهای کاربردی که معمولا به تحلیل زمانی نیاز دارند. بدین منظور، یک عمل مطالعه تحقیقی برای بهبود کیفیت نرم افزار و تجربه کاربری از یک برنامه کاربردی وب با استفاده از BNs به عنوان یک روش برای تحلیل ثبت وقایع نرم‌افزار طراحی و اجرا می‌شود. برای تحقق این هدف، سه مدل با BNs ایجاد می‌شوند. در نتیجه، چندین نقطه بهبود در برنامه از موضوع کارایی و خطاها تا الگوهای بازگشتی کاربرد کاربر شناسایی می‌شود. این نقاط بهبود، ایجاد کارت‌ها را در فرآیند اسکرام[۳] از برنامه کاربردی وب میسر می‌سازد و به نگهداری نرم‌افزار داده محور کمک می‌کند. در نهایت در تحقیق[۲۵] شبکه‌های بیزین در نگهداری نرم‌افزار داده محور و حساس به کیفیت در نظر گرفته می‌شود که به عنوان تکنیک تحلیل ثبت وقایع نرم افزاری پتانسیل بزرگی دارند و جامعه را به بررسی عمیق برنامه‌های کاربردی آن تشویق می‌کنند. بدین منظور متدولوژی به کاربرده شده و یک بسته تکثیر[۴] به اشتراک گذاشته می‌شوند.

## ۳- بحث و نتیجه‌گیری

در این بخش، برای مطالعه و بررسی پژوهش‌های صورت گرفته، شاخص‌های بازه زمانی، نوع مقاله و محدوده موضوعی در نظر گرفته شده و نتایج بررسی این شاخص‌ها در نمودارهای ذیل آمده است:

بازه زمانی:

منظور از بازه زمانی، تاریخ انتشار پژوهش‌ها بوده است. پژوهش‌ها در فاصله زمانی سال‌های ۲۰۰۰ تا ۲۰۲۲ بررسی شده‌اند. شکل ۷ توزیع پژوهش‌های انجام شده در حوزه نرم افزار کاربردی را نشان می‌دهد. همانطور که مشاهده می‌شود، پژوهش‌های منتشر شده در محتوای نرم افزار کاربردی طی این ۲۲ سال رشد نسبی داشته است.

نوع پژوهش‌های صورت گرفته:

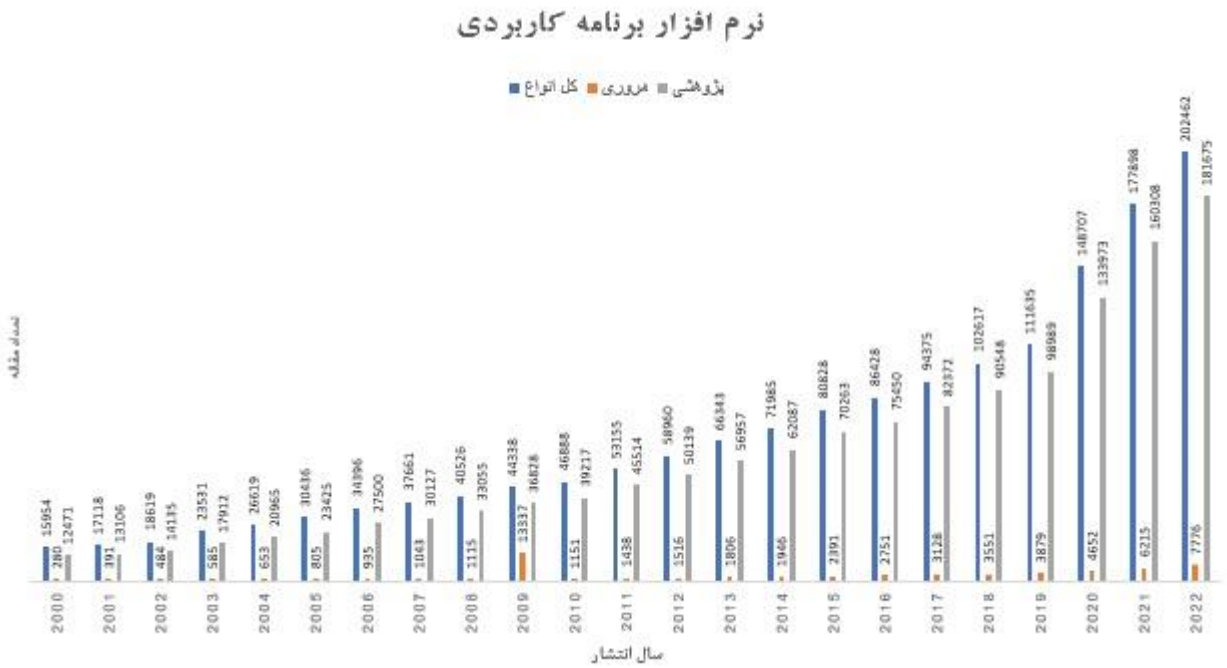
منظور از نوع پژوهش‌ها، نحوه انتشار آن‌ها در انواع پژوهشی، مروری، بخش‌هایی از کتاب و ارتباطات کوتاه می‌باشد. توزیع فراوانی انواع مذکور در شکل ۹ نشان داده شده است.

محدوده موضوعی:

منظور از محدوده‌ی موضوعی، موضوعات مشمول در بررسی نگهداری نرم افزار کاربردی می‌باشد. موضوعات بررسی شده برای پژوهش‌ها شامل علوم‌کامپیوتر، مهندسی، تجارت، مدیریت و حسابداری، انرژی، مهندسی شیمی، علوم تصمیم‌گیری، علوم زمین و سیارات، علوم مواد و علوم اجتماعی می‌باشد.

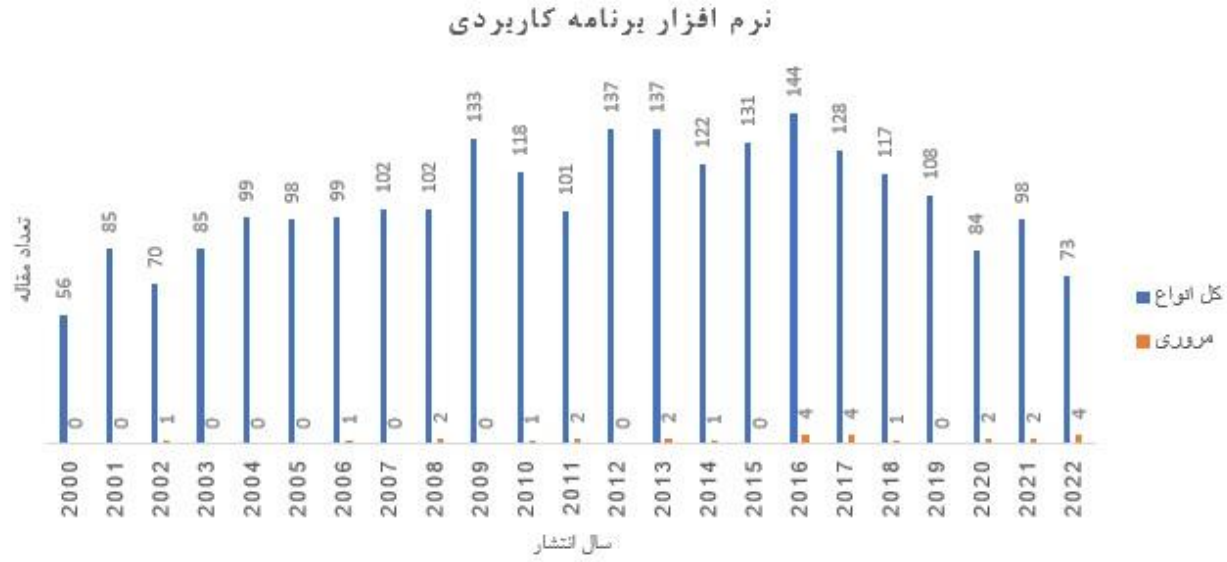
همانطور که در شکل ۷ نشان داده شده‌است تعداد مقالات پژوهشی در ۲۰ سال گذشته به میزان قابل توجهی نسبت به مقالات مروری رشد داشته است و مقالات پژوهشی بیشترین مقدار کل انواع مقالات منتشر شده را تشکیل می‌دهد.

[۱] Bayesian
[۲] Logs
[۳] Scrum
[۴] Replication



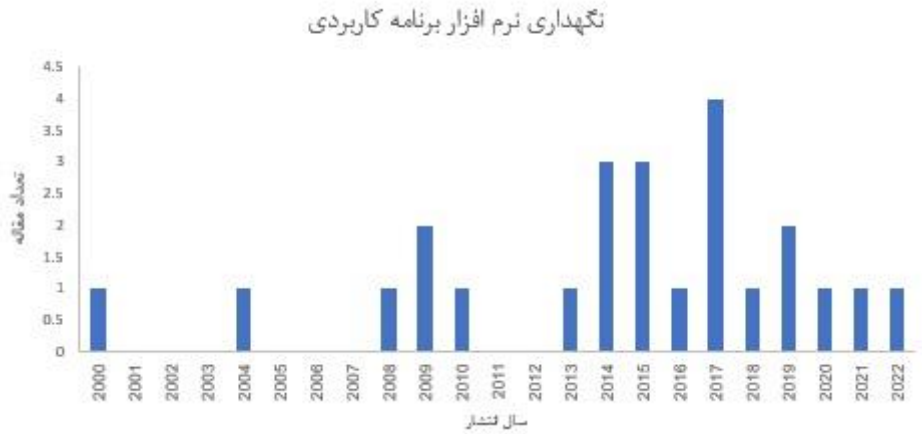

شکل ۷- سری زمانی مقاله‌های منتشر شده در حوزه نرم‌افزار کاربردی (ساینس دایرکت)

شکل ۸ سری زمانی مقاله‌های منتشر شده در حوزه نرم‌افزار کاربردی را نمایش می‌دهد که توسط موتور جست و جوی گوگل اسکولار بدست آمده است. همانطور که در شکل نشان داده شده است، با گذشت زمان تعداد مقاله‌ها رشد نسبی داشته است و بیشترین تعداد مقاله مربوط به سال ۲۰۱۶ می‌باشد.




شکل ۸- سری زمانی مقاله‌های منتشر شده در حوزه نرم‌افزار کاربردی (گوگل اسکولار)

شکل ۹، سری زمانی مقاله‌های منتشر شده در حوزه نگهداری نرم افزار کاربردی را برای پایگاه ساینس دایرکت نشان می‌دهد. نتایج بیانگر بیشترین تعداد مقاله منتشر شده در حوزه مذکور در سال ۲۰۱۷ می‌باشد.

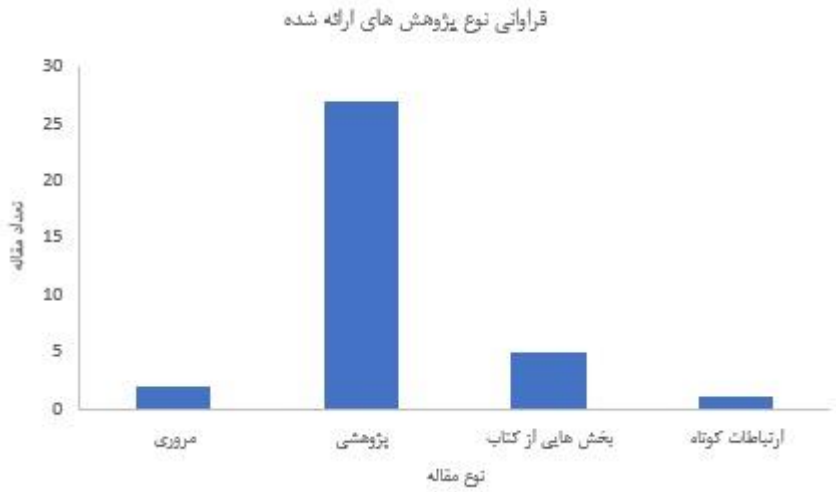



شکل ۹- سری زمانی مقاله‌های منتشر شده در حوزه نگهداری نرم‌افزار کاربردی (ساینس دایرکت)

شکل ۱۰، سری زمانی مقاله‌های منتشرشده در حوزه نگهداری نرم‌افزارکاربردی را در موتور جست و جوی گوگل اسکولار نشان می‌دهد. نتایج بیانگر بیشترین تعداد مقاله منتشر شده در سال ۲۰۰۸ و کمترین تعداد انتشار در سال‌های ۲۰۱۸ تا ۲۰۲۲ در حوزه مذکور می‌باشد.

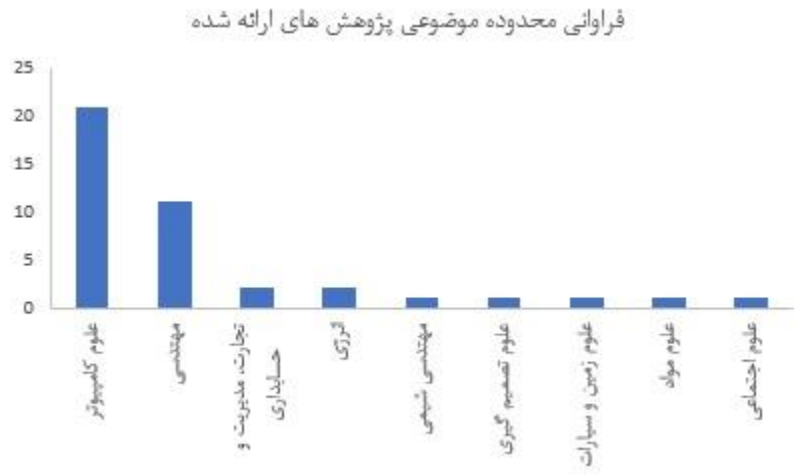

شکل ۱۰- سری زمانی مقاله‌های منتشر شده در حوزه نگهداری نرم‌افزار کاربردی (گوگل اسکولار)

شکل ۱۱، فراوانی نوع مقاله‌های ارائه شده در حوزه نگهداری نرم افزار کاربردی را در پایگاه ساینس دایرکت نشان می‌دهد. نتایج بیانگر بیشترین تعداد مقاله منتشر شده از نوع پژوهشی و کمترین تعداد از نوع ارتباطات کوتاه در حوزه مذکور می‌باشد.




شکل ۱۱- فراوانی نوع مقاله‌های ارائه شده در حوزه نگهداری نرم‌افزار کاربردی (ساینس دایرکت)

شکل ۱۲، فراوانی محدوده موضوعی مقاله‌های ارائه شده در حوزه نگهداری نرم افزارکاربردی را در پایگاه ساینس دایرکت نشان می‌دهد. نتایج بیانگر بیشترین تعداد مقاله منتشر شده در علوم کامپیوتر حوزه مذکور می‌باشد.




شکل ۱۲- فراوانی محدوده موضوعی مقاله‌های منتشر شده در حوزه نگهداری نرم‌افزار برنامه کاربردی (ساینس دایرکت)

### ۴- بحث و نتیجه گیری

در این تحقیق با بررسی پژوهش‌های انجام‌شده در حوزه نگهداری نرم‌افزار کاربردی، تجزیه و تحلیل آماری انجام شد. پژوهش‌ها بر اساس فاکتورهای زمان، نوع، محدوده موضوعی مورد تحلیل قرار گرفتند. آنچه که از نتایج مشخص شد، این است که مقاله‌های پژوهشی در این حوزه نسبت به انواع دیگر با استقبال بیشتری توسط محققین رو به رو شده است، همچنین در علوم کامپیوتر نسبت به محدوده موضوعات دیگر، مقالات بیشتری در حوزه نگهداری نرم افزار کاربردی کار شده است. نتایج بدست آمده از پایگاه ساینس دایرکت و موتور جست و جوی گوگل اسکولار نشان می‌دهد که در سال‌های ۲۰۲۰ تا ۲۰۲۲ در زمینه نگهداری نرم‌افزار کاربردی مقالات کمتری منتشر شده است. بنابراین با توجه به این که قابلیت نگهداری یکی از فاکتورهای سنجش کیفیت نرم‌افزار کاربردی می‌باشد، انجام کارهای پژوهشی بیشتر در حوزه نگهداری نرم افزار کاربردی ضروری است.

## مراجع


[1] J-L.Boulanger, "Certifiable Software Applications 2," pp.1-270, 2017. https://doi.org/10.1016/C2015-0-01249-7.

[2] Sh.Aziz Butt, A.C.Melisa, "Software Product Maintenance: A Case Study," Computer information systems and industrial management, pp.81-92, 2022.

[3] A.Gupta, Sh.Sharma, "Software Maintenance: Challenges and Issues," International Journal of Computer Science Engineering (IJCSE), vol.4, No.01, 2015.

[4] M.Auch, M.Weber, P.Mandl, and Ch.Wolff, "Similarity-based analyses on software applicatio جعns: A systematic literature review," The Journal of Systems & Software, 168, 110669, 2020.

[5] D.Kelly, "Determining factors that affect long-term evolution in scientific application software," The Journal of Systems and Software, vol.82, pp.851-861, 2008.

[6] M.J.Siers, M.Z.Islam, "Novel Algorithms for Cost-Sensitive Classification and Knowledge Discovery in Class Imbalanced Datasets with an Application to NASA Software Defects," INFORMATION SCIENCES, Informatics and Computer Science Intelligent Systems Applications, pp.1-25, 2018.

[7] A.Taivalsaari, T.Mikkonen, "Web Browser as an Application Platform", 34th Euromicro Conference Software Engineering and Advanced Applications, IEEE Computer Society, 2008.

[8] B.P.Lientz, E.B.Swanson, G.E.Tompkins, "Characteristics of Application Software Maintenance," Management Applications, Communications of ACM, pp.466-471, vol.21, 1978.

[9] B.P.Lientz, E.B.Swanson, "Problems in Application Software Maintenance," Applications: Operations and Management, Communications of ACM, pp.763-769, vol.24, 1981.

[10] J.C.Munson, T.M.Khoshgoftaar, "Applications of a Relative Complexity Metric for Software Project Management," SYSTEMS SOFTWARE, vol.12, pp.283-291, 1990.

[11] S.Morasca, G.Ruhe, "A hybrid approach to analyze empirical software engineering data and its application to predict module fault-proneness in maintenance," The Journal of Systems and Software, vol.53, pp.225-237, 2000.

[12] B.S.Baker, "Parameterized duplication in strings: Algorithms and an application to software maintenance," 25th Annual ACM Symposium on Theory of Computing, ACM, Society for Industrial and Applied Mathematics, vol.26, No.5, pp.1343-1362, 1997.

[13] M.G.Lee, Th.L.Jefferson, "An Empirical Study of Software Maintenance of a Web-based Java Application," 21st IEEE International Conference on Software Maintenance (ICSM'05), 2005.

[14] A.AI.Hawari, H.Najadat, R.Shatnawi, "Classification of application reviews into software maintenance tasks using data mining techniques," Software Quality Journal, 2020.

[15] M.A, A.Y, "Software Maintenance Process Towards Cloud Environment: A Review Study," International Journal for Multidisciplinary Research (IJFMR), vol.4, 2022.

[16] V.Lenarduzzi, A.Sillitti, D.Taibi, "A Survey on Code Analysis Tools for Software Maintenance Prediction," Springer Nature Switzerland AG, SEDA, AISC, pp.165-175, 2020.

[17] O.A.Bastias, J.Diaz, J.L.Fenner, "Exploring the Intersection between Software Maintenance and Machine Learning-A Systematic Mapping Study," Applied sciences,2023.

[18] M.Almashhadani, et al, "Challenges in Agile Software Maintenance for Local and Global Development: An Empirical Assessment," Information, vol.14, 261,2023.

[19] K.Liu, S.Reddivari, "Visual Analytics in Software Maintenance: A Systematic Literature Review", ACMSE, Proceedings of the ACM Southeast Conference, pp.70-77, 2023.

[20] F.Kh, M.M.Rahman, A.Barbez, "Intelligent Software Maintenance", Natural Computing Series, bookes (NCS), 2023.

[21] K.H.Bennett, "The Software Maintenance of Large Software Systems: Management, Methods and Tools," Reliability Engineering and System Safety, vol.32, pp.135-154, 2003.

[22] L.H.Yew, Hairulnizam Mahdin, "E-Maintenance Application: System Development," Applied Information Technology And Computer Science, vol.4, no.1, pp.365-387, 2023.
[23] H.Fawareh, "Software Quality Model for Maintenance Software Purposes," International Journal of Engineering Research and Technology, vol.13, pp.158-162, 2020.
[24] J.Araujo, C.Meto, F.Oliveira, P.Pereira, R.Matos, "A Software Maintenance Methodology: An Approach Applied to Software Aging," IEEE International Systems (SysCon), 2021.
[25] S.d.Rey, S.M-Fernandez, A.Salmeron, "Bayesian Network analysis of software logs for data-driven software maintenance," TET Software, The Institution of Engineering and Technology, vol.17, pp.268-286 2023.