آزمون نرم‌افزار روشی برای بهبود کیفیت و قابلیت اطمینان نرم‌افزار

علی کریمی
استادیار دانشگاه جامع امام حسین (ع)
فرهاد کریمی
دانشجوی دکتری دانشگاه جامع امام حسین (ع)

چکیده

توسعه سامانه‌های نرم‌افزاری، به دلیل افزایش مداوم توقعات مشتریان و پیچیدگی روزافزون نیازمندی‌های آن‌ها، فرآیندی بسیار چالش‌برانگیز است. هدف این مقاله ارائه راه‌کارهایی برای بهبود کیفیت نرم‌افزار است. این مقاله به بررسی عوامل کیفی مؤثر بر نرم‌افزار از جنبه‌های مختلف مانند عملکرد، امنیت و قابلیت اطمینان می‌پردازد و استانداردهای شناخته‌شده و بین‌المللی از قبیل ISO و IEEE را معرفی می‌کند. پیاده‌سازی این استانداردها نقش کلیدی در برنامه‌ریزی، طراحی و توسعه نرم‌افزار کیفی دارد و به‌طور مؤثری در تضمین و ارتقاء کیفیت نرم‌افزار تأثیرگذار است. همچنین، انواع مختلف آزمون‌های نرم‌افزاری، از جمله آزمون واحد، آزمون یکپارچگی، و آزمون سیستم، به‌طور دقیق بررسی شده‌اند. در نهایت، یکی از روش‌های پیشرفته توسعه نرم‌افزار که پشتیبانی بیشتری از چرخه حیات آزمون ارائه می‌دهد و امکان انطباق سریع‌تر با تغییرات نیازمندی‌ها را فراهم می‌سازد، به‌طور مختصر معرفی و تشریح می‌شود. این مقاله با ارائه مجموعه‌ای از راهنمایی‌ها و بهترین شیوه‌ها، تلاش دارد تا به توسعه‌دهندگان و تیم‌های نرم‌افزاری در تولید محصولاتی با کیفیت بالاتر، جلب رضایتمندی مشتریان و کسب مزیت رقابتی در بازار کمک کند.

واژگان کلیدی: آزمون نرم‌افزار، قابلیت اطمینان نرم‌افزار، کیفیت نرم‌افزار، استانداردهای آزمون نرم‌افزار

۱- مقدمه

امروزه، سامانه‌های نرم‌افزاری به‌طور فزاینده‌ای پیچیده شده‌اند و توسعه آن‌ها چالش‌برانگیزتر از قبل شده است. درک و پیاده‌سازی کامل نیازمندی‌های مشتریان نیز روزبه‌روز دشوارتر می‌شود. رقابت در عرصه توسعه نرم‌افزار رو به افزایش است. شرکت‌های توسعه‌دهنده، تمام تلاش خود را به‌کار می‌گیرند تا مشتریان بیشتری را جذب و سود بیشتری کسب کنند. بنابراین، محصولات نرم‌افزاری باید علاوه‌بر کیفیت، قابل‌اعتماد نیز باشند تا بتوانند انتظارات و نیازهای مشتریان را برآورده سازند. ویژگی قابلیت‌اطمینانِ یک نرم‌افزار را می‌توان به ویژگی‌های فرعی متعددی تقسیم کرد که برای بهبود کیفیت کلی نرم‌افزار باید به آن‌ها توجه نمود. از آنجایی که شرکت‌های مختلف، از روش‌شناسی‌های متفاوتی استفاده می‌کنند، آزمون نرم‌افزار یکی از مهم‌ترین مراحل در هر روش توسعه محسوب می‌شود. آزمون نرم‌افزار، با شناسایی خطاها و نقص‌ها در نرم‌افزار، نقش بسزایی در بهبود کیفیت آن ایفا می‌کند. با این حال، آزمون آنقدرها هم که به نظر می‌رسد آسان نیست؛ زیرا انواع مختلفی از آزمون وجود دارد و هر کدام از آن‌ها در موقعیت خاصی کاربرد دارند. بنابراین، این مرحله حیاتی به‌منظور بهبود قابلیت‌اطمینان نرم‌افزار، باید مورد توجه ویژه‌تری قرار گیرد.

۲- قابلیت اطمینان نرم‌افزار

قابلیت اطمینان نرم‌افزار را می‌توان به‌صورت احتمال عملکرد بدون خرابی یک نرم‌افزار در یک محیط مشخص و در یک بازه زمانی معین تعریف کرد (N. Pavlov, G. Spasov, A. Rahnev, and N. Kyurkchiev, 2018).

مدل‌های متعددی از ابتدای پیدایش تاکنون برای ارزیابی قابلیت اطمینان نرم‌افزار و بهبود کیفیت کلی آن، از جمله در حین توسعه، ایجاد شده‌اند (Y. Tamura and S. Yamada, 2016).

به‌علاوه، از مدل‌های پیش‌بینی قابلیت اطمینان نرم‌افزار قبل از آزمون نرم‌افزار و با استفاده از مجموعه‌ای از داده‌ها، نظیر پیچیدگی پروژه، نوع زبان برنامه‌نویسی، معماری و غیره، استفاده می‌شود. پس از آزمون نرم‌افزار، از یک مدل قابلیت اطمینان، که یک فرمول ریاضی است، برای ارزیابی قابلیت اطمینان نرم‌افزار از طریق بررسی نرخ خرابی و اطلاعات مربوط به خطاها استفاده می‌شود (K. Y. Song, I. H. Chang, and H. Pham, 2019).

آزمون مداوم یکی از مهم‌ترین مراحل در توسعه نرم‌افزار است، زیرا می‌تواند یک بررسی کلی از کیفیت نرم‌افزار توسعه‌یافته در اختیار تیم تضمین کیفیت قرار دهد. همچنین می‌توان از آن برای بهبود قابلیت اطمینان نرم‌افزار و برآورده کردن انتظارات مشتریان استفاده کرد (R. Pietrantuono, A. Bertolino, G. De Angelis, B. Miranda, and S. Russo, 2019).

قابلیت آزمون نرم‌افزار یکی از عوامل مهم در توسعه پروژه‌های نرم‌افزاری در نظر گرفته می‌شود. بسیاری از استانداردهای بین‌المللی نظیر ISTQB[1]، IEEE، ISO، IEC و MIL-STD قابلیت آزمون نرم‌افزار را به‌این‌صورت تعریف کرده‌اند: درجه‌ای که یک نرم‌افزار یا یک مولفه‌ی نرم‌افزاری می‌تواند مطابق با معیارهای آزمون تحمیل شده، توسط شرکت توسعه‌دهنده، آزمون و

[1] International Software Testing Qualifications Board

اعتبارسنجی شود. آزمون نرم‌افزار، قابلیت اطمینان و کیفیت کد پیاده‌سازی‌شده‌ی و در کل مهارت‌های برنامه‌نویسی را دو برابر بهبود داده است (O. A. Lazzarini Lemos, F. Fagundes Silveira, F. Cutigi Ferrari, and A. Garcia, 2015).

۳- ویژگی‌های قابلیت اطمینان

قابلیت اطمینان نرم‌افزار می‌تواند دارای ویژگی‌های بسیاری از جمله الزامات غیرعملکردی باشد. اگر این ویژگی‌ها به‌خوبی در نظر گرفته شوند، می‌توانند کیفیت نرم‌افزار را بهبود بخشند. این ویژگی‌ها را به ترتیب زیر فهرست شده‌اند: قابلیت دسترسی، کارایی، قابلیت نصب، یکپارچگی، قابلیت همکاری، قابلیت اصلاح، عملکرد، قابلیت حمل، قابلیت اطمینان، قابلیت استفاده مجدد، استحکام، ایمنی، مقیاس پذیری، امنیت، قابلیت استفاده و قابلیت تأیید. همچنین این ویژگی‌ها باید به‌خوبی باهم ترکیب شوند زیرا برخی از آنها در صورت ترکیب می‌توانند تاثیر بدی بر سیستم داشته باشند (F. Pinciroli, 2016).

قابلیت دسترسی را می‌توان به‌عنوان زمان یا دوره‌ای تعریف کرد که کاربران می‌توانند به یک سیستم یا برنامه دسترسی داشته باشند. کارایی نرم‌افزار را می‌توان به‌عنوان کمیت یا نرخ تعداد ورودی‌های موردنیاز برای دستیابی به مقدار بیشتر خروجی تحت هر بار کاری تعریف کرد. قابلیت نصب نرم‌افزار را می‌توان از "نصب آسان" تا "نصب دشوار" یا گاهی اوقات "عدم موفقیت در نصب" به دلیل خطاهای پیاده‌سازی طبقه‌بندی کرد. این دو دسته‌ی آسان و دشوار با استفاده از عامل زمان نصب ارزیابی می‌شوند (M. Nabi, M. Toeroe, and F. Khendek, 2016).

یکپارچگی یکی از جنبه‌های مهم یک نرم‌افزار است، زیرا یکپارچگی به‌معنای حفاظت از کد منبع و داده‌های نرم‌افزار است. هرگونه تغییر در پیاده‌سازی داخلی نرم‌افزار می‌تواند رفتار مورد نظر آن‌را تغییر دهد. یکپارچگی می‌تواند یک زیرمجموعه از ویژگی امنیتی باشد، زیرا امنیت قوی در سیستم‌های نرم‌افزاری به‌معنای توانایی جلوگیری از هرگونه حملات مخرب ورودی صرف‌نظر از نوع حمله است. قابلیت همکاری، توانایی نرم‌افزار یا سخت افزار برای کارکرد و تعامل روان با سایر سیستم‌ها تعریف می‌شود. با تکامل و رشد فناوری، نیازهای مشتریان نیز دائما در حال تغییر است، که این تکامل توسعه‌دهندگان را مجبور می‌کند تا اغلب نرم‌افزار را برای انطباق با این تغییرات اصلاح کنند (A. Al-Said Ahmad and P. Andras, 2019).

یکی دیگر از موارد قابلیت اصلاح است. قابلیت اصلاح، توانایی ارتقا و بهبود نرم‌افزار برای برآورده‌کردن نیازهای کاربران است. عملکرد یکی از مهم‌ترین جنبه‌های یک نرم‌افزار است. عملکرد توانایی نرم‌افزار برای پاسخگویی سریع و تعامل با کاربر است. همچنین توان عملیاتی بخشی از ویژگی‌های عملکردی نرم‌افزار است. همانطور که قبلا ذکر شد، محیط‌های رایانه‌ای به‌دلیل تحول سریع فناوری اغلب در حال تغییر هستند. بنابراین، قابلیت حمل برای سیستم‌های نرم‌افزاری برای مهاجرت از محیط‌های محاسباتی قدیمی به جدید مهم است .استحکام به‌عنوان قابلیت یک نرم‌افزار برای پایداری در محیط‌های غیرمنتظره یا استرس‌زا و عملکرد صحیح در حین دریافت ورودی‌های نامعتبر تعریف می‌شود (C. Hutchison et al, 2018).

از آنجایی که از سیستم‌های نرم‌افزاری می‌توان در شرایط بحرانی مانند کنترلرها استفاده کرد، ویژگی ایمنی، پتانسیل یک سیستم نرم‌افزاری یا کنترلر برای عملکرد صحیح در طول عملیات و محیط‌های بحرانی است (J. Zhang and J. Li, 2020).

۴- کیفیت نرم‌افزار

کیفیت نرم‌افزار، طبق استاندارد IEEE 730-2014 سازمانIEEE ، به معنای توانایی نرم‌افزار توسعه‌یافته در برآورده‌کردن تمامی الزامات کاربران و ذی‌نفعان است. این مفهوم به‌ویژه در دنیای امروز که نیازها و انتظارات مشتریان به سرعت در حال تغییر است، اهمیت بیشتری پیدا کرده است. کیفیت نرم‌افزار نه تنها به عملکرد درست نرم‌افزار، بلکه به قابلیت اطمینان، امنیت، قابلیت استفاده و مقیاس‌پذیری آن نیز اشاره دارد. این ویژگی‌های کیفیت نرم‌افزار شامل قابلیت اطمینان و سایر ویژگی‌های کیفی است که قبلاً ذکر شده‌اند و به‌عنوان معیاری برای سنجش موفقیت یک نرم‌افزار عمل می‌کنند. برای تأمین این ویژگی‌های کیفی، تضمین کیفیت نرم‌افزار به‌عنوان یک فرآیند کلیدی در چرخه توسعه نرم‌افزار مطرح می‌شود. تضمین کیفیت نرم‌افزار شامل فعالیت‌های مختلفی است که به‌منظور اطمینان از کیفیت نهایی نرم‌افزار انجام می‌شود. این فعالیت‌ها شامل بازرسی‌های منظم، آزمون‌های مختلف، و ارزیابی پیوسته نرم‌افزار در طول مراحل مختلف توسعه است. هدف اصلی تضمین کیفیت نرم‌افزار این است که اطمینان حاصل کند نرم‌افزار تولید شده از نظر خدمات نرم‌افزاری مناسب است و تمامی الزامات توصیف‌شده توسط مشتری را برآورده می‌کند. کیفیت نرم‌افزار می‌تواند از طریق رویکردهای مختلف در طول تمامی مراحل چرخه حیات توسعه نرم‌افزار به دست آید. این چرخه شامل مراحل تحلیل نیازمندی‌ها، طراحی، پیاده‌سازی، آزمون و نگهداری است. در هر یک از این مراحل، توجه به کیفیت و استفاده از ابزارها و فنون مناسب می‌تواند به بهبود نتایج نهایی کمک کند. به‌ویژه در مرحله آزمون، انجام آزمون‌های جامع و دقیق می‌تواند به شناسایی و رفع مشکلات قبل از عرضه نرم‌افزار به بازار کمک کند. یک سیستم نرم‌افزاری که دارای خطاهای متعددی است، ممکن است کیفیت پایینی داشته باشد و این موضوع می‌تواند پیامدهای جدی برای سازمان‌ها و کاربران نهایی داشته باشد. این خطاها می‌توانند ناشی از عوامل مختلفی باشند، از جمله الزامات اشتباه، عدم ارتباط مؤثر بین توسعه‌دهندگان و مشتریان، انحراف از الزامات مشتری، طراحی ضعیف نرم‌افزار و خطاهای پیاده‌سازی. همچنین، عدم انطباق بین اسناد و کد، فرآیندهای آزمون ناکارآمد، رابط‌های کاربری ناپسند و در نهایت اسناد نادرست نیز می‌توانند به کاهش کیفیت نرم‌افزار منجر شوند. به این ترتیب، توجه به کیفیت نرم‌افزار در تمامی مراحل توسعه و استفاده از فرآیندهای مناسب برای مدیریت و بهبود آن، از اهمیت بالایی برخوردار است. این توجه می‌تواند به کاهش هزینه‌ها و زمان‌های صرف‌شده برای اصلاح مشکلات در مراحل بعدی منجر شود و همچنین به افزایش رضایت مشتریان کمک کند. در دنیای کنونی که رقابت در بازار نرم‌افزار بسیار شدید است، سازمان‌ها باید به‌طور مستمر کیفیت نرم‌افزارهای خود را ارزیابی و بهبود دهند تا بتوانند به نیازهای متغیر مشتریان پاسخ دهند و در نتیجه در بازار موفق باشند. بنابراین، برای دستیابی به کیفیت مطلوب نرم‌افزار، علاوه بر راهبردهای آزمون مناسب، استفاده از فناوری‌های نوین و روش‌های توسعه چابک نیز می‌تواند به سازمان‌ها کمک کند تا به سرعت به تغییرات نیاز مشتریان پاسخ دهند. در نهایت، سرمایه‌گذاری در کیفیت نرم‌افزار نه تنها به بهبود محصولات کمک می‌کند، بلکه به ساخت یک برند معتبر و قابل اعتماد در بازار نرم‌افزار نیز منجر خواهد شد. این اقدامات، می‌توانند نهایتاً به ایجاد یک چرخه مثبت منجر شوند که در آن کیفیت بالا، رضایت مشتری و موفقیت تجاری به یکدیگر وابسته‌اند (D. Galin, 2018).

۵- استانداردهای کیفیت

دستیابی به کیفیت نرم‌افزار، ضرورتی انکارناپذیر در عصر دیجیتال، مستلزم اتخاذ رویکردی چندوجهی و تعهد به پیاده‌سازی استانداردهای بین‌المللی معتبر مانند ISO/IEC و IEEE است. این استانداردها، به‌عنوان سنگ‌بنای توسعه نرم‌افزار قابل اعتماد و کارآمد، فراتر از ارائه چارچوبی برای اندازه‌گیری کیفیت، راهنمایی‌های عملی برای ایجاد فرآیندهایی را ارائه می‌دهند که بهبود مستمر و یکپارچه کیفیت در تمام مراحل چرخه حیات نرم‌افزار را تضمین می‌کنند. سازمان سازنده نرم‌افزار، در مسیر دستیابی به یک سیستم مدیریت کیفیت نرم‌افزار بهینه، می‌تواند با به‌کارگیری خانواده استاندارد ISO/IEC 9000، نقشه‌ای جامع برای تحقق این هدف ترسیم کند. این خانواده از استانداردها، با ارائه رهنمودهایی برای دستیابی به سطح مطلوبی از کیفیت، به سازمان‌ها کمک می‌کند تا فرآیندهای مستند، رویه‌های قابل تکرار و مسئولیت‌های مشخص را ایجاد کنند. برای مثال، ISO/IEC 9001، به‌عنوان پرکاربردترین استاندارد در این خانواده، بر درک دقیق الزامات مشتری، تدوین سیاست‌های کیفیت و اطمینان از کنترل کیفیت در تمامی مراحل چرخه حیات توسعه نرم‌افزار تاکید دارد. این کنترل کیفیت می‌تواند شامل بازرسی کد، آزمون واحد، آزمون یکپارچه‌سازی و آزمون سیستم باشد. فراتر از این‌ها، ممیزی‌های داخلی منظم و بررسی‌های مدیریتی دوره‌ای، ابزارهایی ارزشمند برای ارزیابی مداوم اثربخشی سیستم مدیریت کیفیت و شناسایی فرصت‌های بهبود هستند. فرض کنید یک شرکت توسعه‌دهنده نرم‌افزار، به‌طور دوره‌ای جلسات بررسی کد برگزار می‌کند که در آن همکاران کد یکدیگر را بررسی کرده و بازخورد ارائه می‌دهند. این امر نه تنها به شناسایی و رفع زودهنگام باگ‌ها کمک می‌کند، بلکه به انتقال دانش و بهبود مهارت‌های برنامه‌نویسی در تیم نیز منجر می‌شود.

در عرصه خطیر تضمین کیفیت، استاندارد IEEE Std. 730-2014، به‌عنوان چراغ راهنمایی، سازمان‌ها را در پیاده‌سازی اثربخش فرآیندهای تضمین کیفیت یاری می‌رساند. این استاندارد، با ارائه راهنمایی‌هایی برای ایجاد یک برنامه تضمین کیفیت نرم‌افزار، فعالیت‌هایی نظیر بررسی الزامات، طراحی، کد و آزمون را در بر می‌گیرد. استاندارد IEEE Std. 730-2014، فراتر از شناسایی ریسک‌های بالقوه، بر اجرای اقداماتی برای کاهش آنها، مانند استفاده از ابزارهای تحلیل استاتیک کد برای شناسایی آسیب‌پذیری‌های امنیتی یا پیاده‌سازی فرآیندهای مدیریت پیکربندی دقیق برای جلوگیری از ناسازگاری‌های نرم‌افزاری، تاکید دارد. تصور کنید تیمی از توسعه‌دهندگان، قبل از شروع کدنویسی، جلسات بررسی الزامات را برگزار می‌کنند تا از درک مشترک و کامل نیازمندی‌ها اطمینان حاصل شود. این اقدام پیشگیرانه می‌تواند از بروز بسیاری از مشکلات در مراحل بعدی توسعه جلوگیری کند. همچنین، ایجاد معیارهایی برای اندازه‌گیری کیفیت نرم‌افزار، مانند تعداد باگ‌های گزارش‌شده در واحد زمان، میزان پوشش کد توسط آزمون‌ها یا امتیاز قابلیت استفاده نرم‌افزار، و ردیابی پیشرفت در طول چرخه حیات توسعه نرم‌افزار، امکان ارزیابی عینی و تصمیم‌گیری آگاهانه را فراهم می‌سازد.

برای صیقل دادن فرآیندهای چرخه حیات توسعه نرم‌افزار و افزایش بهره‌وری و کیفیت، استاندارد ISO/IEC/IEEE 12207:2008، به‌عنوان یک راهنمای جامع، مسیر را روشن می‌سازد. این استاندارد، با ارائه چارچوبی نظام‌مند برای فرآیندهای نرم‌افزاری، فعالیت‌ها و وظایف مورد نیاز در هر مرحله از چرخه حیات نرم‌افزار، از مرحله برنامه‌ریزی و تحلیل نیازمندی‌ها گرفته

تا پیاده‌سازی، آزمون، استقرار و نگهداری را به تفصیل شرح می‌دهد. پیروی از استاندارد ISO/IEC/IEEE 12207:2008، به سازمان‌ها این امکان را می‌دهد که از انجام فرآیندهای خود به‌طور مداوم و کارآمد هزینه‌ها را کاهش داده و کیفیت محصولات نرم‌افزاری را افزایش دهند. مستندسازی کامل فرآیندها، مدیریت پیکربندی دقیق، و اطمینان از رعایت الزامات قانونی و نظارتی، از دیگر مزایای این استاندارد هستند. فرض کنید یک سازمان، با استفاده از ابزارهای مدیریت پیکربندی، نسخه‌های مختلف کد منبع، اسناد و تنظیمات سیستم را به‌طور دقیق مدیریت می‌کند. این امر به آنها کمک می‌کند تا در صورت بروز مشکل، به نسخه‌های قبلی برگردند و از از بین رفتن داده‌ها جلوگیری کنند.

در نهایت، برای تأیید اینکه محصول نهایی مطابق با مشخصات خود ساخته شده و نیازهای کاربر را برآورده می‌کند، استاندارد IEEE Std. 1012-2012، دستورالعمل‌هایی روشن و دقیق برای تأیید و اعتبارسنجی نرم‌افزار ارائه می‌دهد. این استاندارد، با ارائه یک رویکرد نظام‌مند برای تأیید و اعتبارسنجی نرم‌افزار، فعالیت‌هایی مانند بررسی کد، آزمون واحد، آزمون یکپارچه‌سازی، آزمون سیستم و آزمون پذیرش را در بر می‌گیرد. به‌عنوان مثال، در فرآیند تأیید، بررسی می‌شود که آیا کد منبع با استانداردهای کدنویسی مطابقت دارد و آیا الگوریتم‌ها به درستی پیاده‌سازی شده‌اند. در مقابل، فرآیند اعتبارسنجی، به این سوال پاسخ می‌دهد که آیا نرم‌افزار نیازهای کاربر را برآورده می‌کند و آیا می‌تواند به‌طور مؤثر در محیط واقعی استفاده شود. پیروی از دستورالعمل‌های این استاندارد به سازمان‌ها کمک می‌کند تا خطاها و نقص‌های نرم‌افزاری را به‌طور زودهنگام در چرخه حیات توسعه نرم‌افزار شناسایی و اصلاح کنند، از بروز مشکلات بزرگتر در مراحل بعدی جلوگیری کنند و رضایت مشتریان را تضمین نمایند. تلفیق این استانداردها با فرآیندهای توسعه نرم‌افزار، سازمان‌ها را قادر خواهد کرد که سطح کیفیت محصولات خود را به‌طور چشمگیری ارتقا دهند، هزینه‌ها را کاهش دهند، رضایت مشتریان را افزایش دهند و در عرصه رقابتی فناوری اطلاعات، جایگاه خود را مستحکم‌تر سازند (M. Philip, N. Singhal, R. Ravi, and V. B, 2020).

۶-آزمون نرم‌افزار

آزمون نرم‌افزار، که به‌عنوان ارزیابی نرم‌افزار تولید شده تعریف می‌شود، گامی بسیار مهم برای اطمینان از کیفیت مطلوب نرم‌افزار است. برای وضوح بیشتر، آزمون نرم‌افزار را می‌توان به دو دسته اصلی تقسیم کرد: آزمون ایستا و آزمون پویا. نوع اول، یعنی آزمون ایستا، برای ارزیابی سند نیازمندی‌ها، طراحی نرم‌افزار و کد منبع از طریق بازرسی‌ها، بازبینی‌ها و بررسی‌های دقیق انجام می‌شود. از طرف دیگر، آزمون پویا به بررسی نرم‌افزار تولید شده از طریق اجرای آن با استفاده از ورودی‌های مختلف اشاره دارد؛ این نوع آزمون‌ها به توسعه‌دهندگان اجازه می‌دهند تا عملکرد و رفتار سیستم را مشاهده کنند. دستیابی به کیفیت مطلوب نرم‌افزار با استفاده از تنها یک نوع آزمون پویا امکان‌پذیر نیست. از این رو، انواع مختلفی از آزمون پویا مانند آزمون واحد، آزمون یکپارچگی، آزمون سیستم و آزمون پذیرش در دسترس است. این آزمون‌ها بدون برنامه‌ریزی انجام نمی‌شوند؛ تولیدکننده نرم‌افزار راهبردهای آزمون را تعیین می‌کند و سپس برای انجام آن‌ها بر روی سیستم نرم‌افزاری، برنامه‌ریزی و طراحی آزمون را انجام می‌دهد. دو نوع راهبرد آزمون مهم دیگر نیز وجود دارد: راهبرد آزمون جعبه سیاه و راهبرد آزمون جعبه سفید (یا جعبه شیشه‌ای). آزمون جعبه سیاه شامل آزمون سیستم بدون آگاهی از ساختار کد آن است و

عمدتاً برای شناسایی اشکالات و عملکردهای نامناسب کاربرد دارد. از سوی دیگر، آزمون جعبه سفید آزمونی است که در آن کد داخلی برای یافتن اشکالات و خطاها بررسی می‌شود. به‌طور کلی، آزمون جعبه سیاه می‌تواند بیشتر از آزمون جعبه سفید انجام شود، زیرا به بررسی صحت، دسترس‌پذیری، قابلیت اطمینان، امنیت، قابلیت استفاده، قابلیت نگهداری، انعطاف‌پذیری، قابلیت آزمون، قابلیت حمل، قابلیت استفاده مجدد، قابلیت همکاری و سایر ویژگی‌ها می‌پردازد. در حالی که آزمون جعبه سفید تنها می‌تواند به صحت، قابلیت نگهداری و قابلیت استفاده مجدد بپردازد. همه آزمون‌های ذکر شده، چه خودکار و چه دستی، می‌توانند به بهبود قابلیت اطمینان و کیفیت کلی نرم‌افزار تولیدشده کمک کنند. همچنین، روش‌های توسعه نیز نقش مهمی در بهبود کیفیت نرم‌افزار دارند، زیرا یک روش مناسب می‌تواند کارایی بیشتری را به تمام مراحل چرخه حیات، از جمله مرحله آزمون، ارائه دهد. به‌عنوان مثال، روش چابک یا اسکرام به‌عنوان یک راه خوب برای بهبود کیفیت نرم‌افزار شناخته می‌شود، زیرا الزامات روشن و بدون ابهام هستند. از این‌رو، آزمون‌های انجام‌شده می‌توانند این الزامات را ارزیابی کنند تا بررسی کنند که آیا با نیازهای کاربر مطابقت دارد یا خیر (M. D. Haiderzai and M. I. Khattab,M. D. Haiderzai and M. I. Khattab, 2019).

۷-نتیجه‌گیری

کیفیت و قابلیت اطمینان نرم‌افزار پس از تحولات فناوری و تغییرات سریع در نیازمندی‌های بازار، توجه بیشتری را از سوی توسعه‌دهندگان و سازمان‌ها به خود جلب کرده است. با پیشرفت‌های بیشتر و پیچیده‌تر شدن نیازمندی‌های مشتریان، نرم‌افزارها باید نه تنها عملکردهای اساسی را به‌خوبی انجام دهند، بلکه باید قابلیت اطمینان، امنیت و کارایی بالایی نیز داشته باشند. به همین دلیل، بسیاری از سازمان‌های بین‌المللی استانداردهایی را برای سیستم‌های مدیریت کیفیت و تضمین کیفیت معرفی کرده‌اند. این استانداردها به‌منظور حفظ کیفیت نرم‌افزار و جلب رضایت مشتریان طراحی شده‌اند و شامل رهنمودهایی برای فرآیندهای توسعه، آزمون و ارزیابی نرم‌افزار هستند. این رهنمودها به سازمان‌ها کمک می‌کنند تا محصولات خود را مطابق با نیازهای مشتریان و الزامات بازار تولید کنند و در نتیجه توانایی رقابتی خود را افزایش دهند. آزمون نرم‌افزار نقش حیاتی در مدیریت کیفیت و قابلیت اطمینان یک نرم‌افزار ایفا می‌کند. با استفاده از راهبردهای آزمون متعددی که برای پوشش و ارزیابی تمامی عملکردهای نرم‌افزار در دسترس است، توسعه‌دهندگان می‌توانند مشکلات و اشکالات نرم‌افزار را شناسایی کرده و پیش از عرضه به بازار، آن‌ها را برطرف کنند. انواع مختلف آزمون‌ها، از جمله آزمون واحد، آزمون یکپارچگی، آزمون سیستم و آزمون پذیرش، هر یک به نحوی خاص به ارزیابی جنبه‌های مختلف نرم‌افزار کمک می‌کنند. با این‌حال، با وجود روش‌شناسی‌های توسعه مختلف برای پروژه‌های نرم‌افزاری، مدیریت زمان و هزینه یک پروژه همچنان می‌تواند چالش‌برانگیز باشد. فرآیند آزمون نرم‌افزار معمولاً زمان‌بر و پرهزینه است و این موضوع می‌تواند بر راهبردهای آزمون تأثیر بگذارد. به‌عنوان مثال، اگر زمان و منابع کافی برای انجام آزمون‌ها در دسترس نباشد، ممکن است برخی از آزمون‌ها نادیده گرفته شوند و این موضوع می‌تواند منجر به کاهش کیفیت نرم‌افزار شود. از این‌رو، کار آینده می‌تواند بر این موضوع متمرکز شود که چگونه انتخاب روش‌شناسی‌های مناسب می‌تواند به بهبود کارایی در هنگام انجام آزمون‌های مختلف منجر شود. به‌عنوان مثال، استفاده از روش‌های توسعه چابک می‌تواند به تیم‌ها کمک کند تا به سرعت به تغییرات نیاز مشتریان پاسخ دهند و در عین حال کیفیت نرم‌افزار را حفظ کنند. توجه به این

نکته نیز مهم است که راهبردهای مناسب آزمون، می‌توانند به بهینه‌سازی منابع و زمان کمک کنند و در نهایت منجر به تولید نرم‌افزاری با کیفیت بالاتر و رضایت بیشتر مشتریان شوند. با اتخاذ رویکردهای مناسب در برنامه‌ریزی و اجرای آزمون‌ها، سازمان‌ها می‌توانند در چرخه حیات توسعه نرم‌افزار، اطمینان بیشتری از کیفیت و قابلیت اطمینان محصولات خود داشته باشند و در نهایت به موفقیت بیشتری در بازار دست یابند. با تمرکز بر کیفیت و قابلیت اطمینان نرم‌افزار، سازمان‌ها می‌توانند روابط بهتری با مشتریان خود برقرار کنند و به ایجاد نرم‌افزارهایی بپردازند که نه تنها نیازهای فعلی، بلکه نیازهای آینده مشتریان را نیز برآورده کنند. این رویکرد منجر به وفاداری مشتریان و ایجاد یک نشان تجاری معتبر در بازار می‌شود که در نهایت به رشد و توسعه پایدار سازمان کمک خواهد کرد. با توجه به اهمیت روزافزون کیفیت نرم‌افزار در دنیای فناوری اطلاعات، سازمان‌ها باید به‌طور مداوم به ارزیابی و بهبود فرآیندهای خود بپردازند تا بتوانند به نیازهای متغیر مشتریان پاسخ دهند و در محیط رقابتی امروزی موفق عمل کنند.

۸- منابع

N. Pavlov, G. Spasov, A. Rahnev, and N. Kyurkchiev, “A new class of Gompertz–type software reliability models,” Int. Electron. J. Pure Appl. Math., vol. 12, no. 1, pp. 43–57, 2018.

Y. Tamura and S. Yamada, “Software Reliability Model Selection Based on Deep Learning with Application to the Optimal Release Problem,” J. Ind. Eng. Manag. Sci., vol. 2016, no. 1, pp. 43–58, 2016.

K. Y. Song, I. H. Chang, and H. Pham, “NHPP software reliability model with inflection factor of the fault detection rate considering the uncertainty of software operating environments and predictive analysis,” Symmetry (Basel)., vol. 11, no. 4, 2019.

R. Pietrantuono, A. Bertolino, G. De Angelis, B. Miranda, and S. Russo, “Towards continuous software reliability testing in DevOPs,” in Proceedings - 2019 IEEE/ACM 14th International Workshop on Automation of Software Test, AST, 2019, pp. 21–27

V. Garousi, M. Felderer, and F. N. Kılıçaslan, “A survey on software testability,” Inf. Softw. Technol., vol. 108, pp. 35–64, 2019.

O. A. Lazzarini Lemos, F. Fagundes Silveira, F. Cutigi Ferrari, and A. Garcia, “The impact of Software Testing education on code reliability: An empirical assessment,” J. Syst. Softw., vol. 137, no. Issre 2015, pp . Journal of Applied Technology and Innovation (e -ISSN: 2600-7304) vol. 5, no. 2, (2021) 46 497–511, 2018.

F. Pinciroli, “Improving Software Applications Quality by Considering the Contribution Relationship among Quality Attributes,” Procedia Comput. Sci., vol. 83, no. Antifragile, pp. 970–975, 2016

M. Nabi, M. Toeroe, and F. Khendek, “Availability in the cloud: State of the art,” J. Netw. Comput. Appl., vol. 60, pp. 54–67, 2016.

A. Al-Said Ahmad and P. Andras, “Scalability analysis comparisons of cloud-based software services,” J. Cloud Comput., vol. 8, no. 1, 2019.

M. Philip, N. Singhal, R. Ravi, and V. B., “A Quantitative Approach to Analyze Modifiability in Software Architectural Design of Agile Application Systems,” Inf. Technol. Control, vol. 49, no. 2, pp. 249 –259 2020.

C. Hutchison et al., “Robustness testing of autonomy software,” Proc - .Int. Conf. Softw. Eng., pp. 276–285, 2018, doi :10.1145/31835

J. Zhang and J. Li, “Testing and verification of neural-network-based safety-critical control software: A systematic literature review,” Inf .Softw. Technol., vol. 123, no. April 2019, 2020.

D. Galin, Software Quality Assurance: Concepts and Practice. WileyIEEE Press, 2018 [۱۹] .N. Honest, “Role of Testing in Software Development Life Cycle,” Int .J. Comput. Sci. Eng., vol. 7, no. 5, pp. 886–889, May 2019, doi :10.26438/ijcse/v7i5.886889.

M. D. Haiderzai and M. I. Khattab, “How software testing impact the quality of software systems ?,” Int. J. Eng. Comput. Sci., vol. 1, no. 2 ,pp. 5–9, 2019.