روش‌ها و موانع آزمون در نرم‌افزار

علی کریمی

استادیار دانشگاه جامع امام حسین (ع)

فرهاد کریمی

دانشجوی دکتری دانشگاه جامع امام حسین (ع)

چکیده

آزمون نرم‌افزار فرآیندی است که بررسی می‌کند آیا نرم‌افزار توسعه‌یافته مطابق با نیازهای واقعی مشتری و استانداردهای مشخص شده طراحی شده است یا خیر. این فرآیند به شناسایی مشکلات، خطاها و نواقص نرم‌افزار کمک می‌کند تا قبل از استفاده نهایی، اصلاح شوند. در طول دهه‌های گذشته، روش‌های توسعه نرم‌افزار دستخوش تغییرات زیادی شده‌اند. این تغییرات ناشی از پیشرفت فناوری، نیازهای جدید کاربران و افزایش پیچیدگی نرم‌افزارها بوده است. در نتیجه، روش‌های آزمون نرم‌افزار نیز برای همگام شدن با این تغییرات تکامل یافته‌اند. با رشد سریع فناوری در حوزه‌های مختلف مانند پزشکی، رسانه، ایمنی، کشاورزی، رباتیک، بازی‌های رایانه‌ای، هوانوردی و دفاع ملی، آزمون نرم‌افزار به یک ضرورت تبدیل شده است.خطاهای نرم‌افزاری در این حوزه‌ها می‌توانند منجر به مشکلات جدی، از جمله آسیب‌های مالی، جانی و امنیتی شوند. در این مقاله، روندهای جدید، فنون مدرن و چالش‌های موجود و در حال ظهور در زمینه آزمون نرم‌افزار بررسی می‌شوند. هدف این بررسی، شناسایی راهکارهای مؤثر برای بهبود کیفیت نرم‌افزار و اطمینان از عملکرد صحیح آن در شرایط مختلف است.

واژه‌های کلیدی: آخرین فناوری‌ها، فنون آزمون، آزمون خودکارسازی، چالش‌های آزمون و چرخه عمر آزمون نرم‌افزار.

۱- مقدمه

آزمون نرم‌افزار فرآیندی است که سامانه را براساس نیازهای مشخص‌شده توسط مشتری ارزیابی می‌کند. این فرآیند به‌عنوان بخشی از تضمین کیفیت، به شناسایی اشکالات نرم‌افزار کمک می‌کند و به توسعه‌دهندگان این امکان را می‌دهد که کد را بدون خطا ارائه دهند. شکل ۱، مراحل چرخه عمر آزمون نرم‌افزار را نمایش می‌دهد. در گذشته، این مراحل به‌صورت دستی انجام می‌شدند، اما با پیشرفت فناوری، ابزارهای مختلفی توسعه یافته‌اند که به آزمون‌گران کمک می‌کنند تا این آزمون‌ها را به‌صورت خودکار انجام دهند. این ابزارها باعث می‌شوند آزمون‌گران بیشتر بر تحلیل و تفکر منطقی تمرکز کنند و از کارهای تکراری و وقت‌گیر دوری کنند. استفاده از ابزارهای خودکارسازی آزمون، به‌ویژه در پروژه‌های بزرگ و پیچیده، می‌تواند به افزایش دقت و سرعت آزمون‌ها منجر شود. این ابزارها معمولاً شامل قابلیت‌هایی مانند ثبت و گزارش‌گیری اتوماتیک، شبیه‌سازی سناریوهای مختلف و تحلیل نتایج هستند که به تسهیل فرآیند آزمون کمک می‌کنند. در نهایت، این پیشرفت‌ها نه‌تنها کیفیت نرم‌افزار را بهبود می‌بخشند، بلکه زمان و منابع لازم برای آزمون را نیز کاهش می‌دهند، که در نهایت به بهبود کارایی کلی فرآیند توسعه نرم‌افزار منجر می‌شود.

شکل ۱.چرخه عمر آزمون نرم افزار

آزمون نرم‌افزار مبتنی بر مخاطره است، زیرا هزینه شناسایی یک اشکال در مراحل بعدی توسعه بسیار بالا است. همان‌طور که گفته شده، زمانی‌که یک سامانه تحت آزمون قرار می‌گیرد، مستعد خطا می‌شود و در این حال آزمون بیش از حد می‌تواند سامانه را در برابر مشکلات مصون کند. به عبارت دیگر، هرچه یک نرم‌افزار بیشتر آزمون شود، از نظر نسبت به خطاها ایمن‌تر می‌شود. بنابراین، این مسئولیت بر عهده آزمون‌گر است که به طور فعال فکر کند و فرآیند آزمون را به طور ماهرانه انجام دهد تا بیشترین نقص و اشکال ممکن را شناسایی کند. در این مقاله، مروری بر روند آزمون نرم‌افزار، فنون و چالش‌های پیش روی آزمون‌گران ارائه می‌شود.

شکل ۱ که نشان‌دهنده‌ی چرخه آزمون نرم‌افزار است شامل مراحل مختلفی است که به‌صورت زیر انجام می‌شود: ابتدا نیازمندی‌ها جمع‌آوری و تجزیه و تحلیل می‌شوند، سپس طرح آزمون برای کل فرآیند آزمون تهیه می‌گردد. در ادامه، موارد آزمون نوشته می‌شوند و پس از آن، این موارد آزمون اجرا می‌شوند. در نهایت نتایج به‌دست‌آمده از اجرای آزمون‌ها باهم مقایسه می‌شوند و در آخر با بسته‌شدن آزمون، تمام مستندات آزمون ارائه می‌شود.

۲- چرخه آزمون نرم‌افزار، فنون، مشکلات و چالش‌ها

روند آزمون نرم‌افزار در دو دهه گذشته به سرعت تکامل یافته است. فرآیند آزمون نرم‌افزار یک نهاد مستقل نیست و به طور مستمر با توسعه همکاری می‌کند. با ظهور نوآوری‌های مختلف در روش‌های جدید توسعه نرم‌افزار و فناوری، فرآیند آزمون نیز دستخوش تحول شده است. بر اساس روندهای تکنولوژی، در این مقاله روندها و فنون آزمون نرم‌افزار پوشش داده شده است.

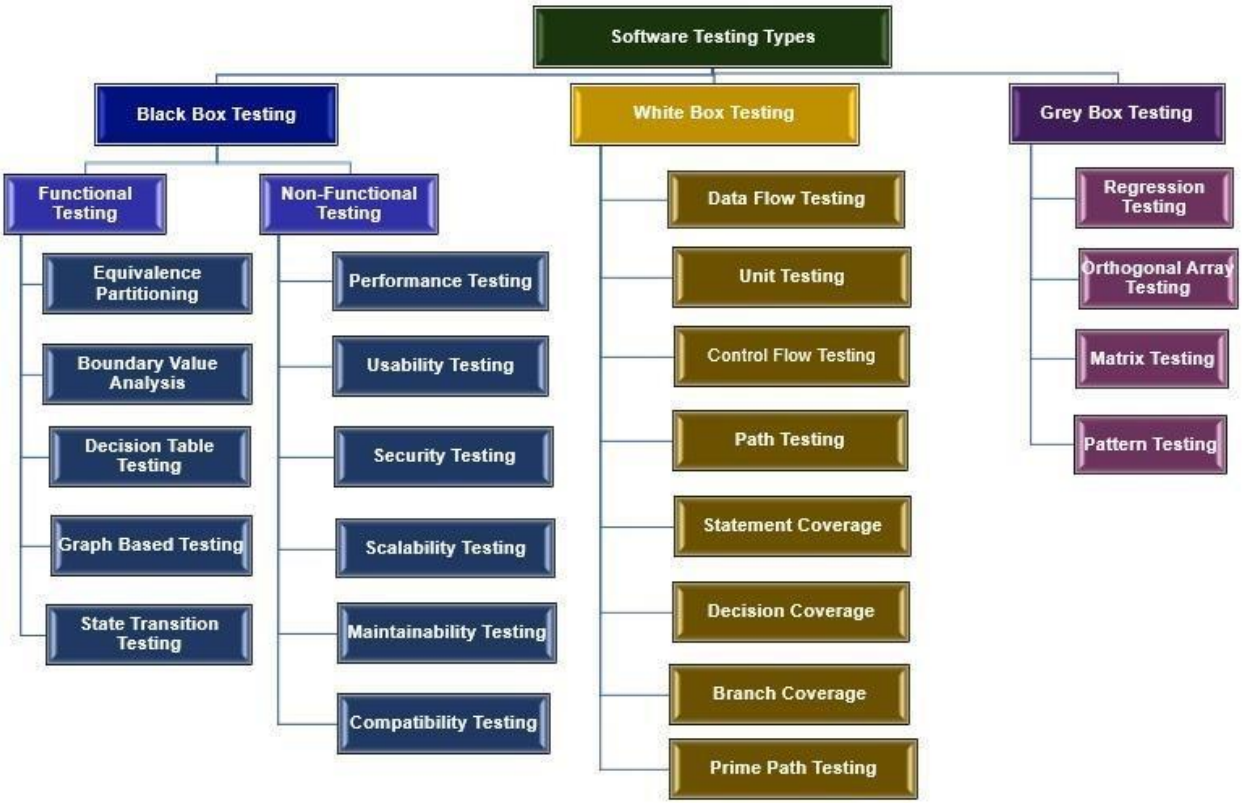
شکل ۲.چرخه آزمون نرم افزار

شکل ۲ نشان‌دهنده تأثیر آخرین فناوری‌ها بر نرم‌افزار و فرآیندهای آزمون آن است. این پیشرفت‌ها باعث شده‌اند که آزمون نرم‌افزار به یک فرآیند هدایت‌شده و کارآمد تبدیل شود.

در حالی که این فرآیند همچنان چالش‌هایی دارد، دیگر مانند گذشته خسته‌کننده و کاملاً دستی نیست. امروزه، ابزارها و فناوری‌های نوین به آزمون‌گران کمک می‌کنند تا نرم‌افزارها را سریع‌تر و مؤثرتر ارزیابی کرده و کیفیت آن‌ها را بهبود بخشند.

۲-۱- مدل نرم‌افزار سنتی

مدل سنتی توسعه نرم‌افزار، که به عنوان مدل آبشاری[۱] شناخته می‌شود، به این شکل عمل می‌کند که فرآیند آزمون تنها پس از اتمام کامل مراحل توسعه انجام می‌شود. این مدل در بسیاری از متدولوژی‌های نرم‌افزاری محبوب مورد استفاده قرار می‌گیرد، اما تفاوت‌هایی در زمان استفاده از فنون آزمون نرم‌افزار سنتی و نحوه اصلاح و به‌کارگیری آن‌ها وجود دارد. شکل ۳ انواع آزمون نرم‌افزار و دسته‌بندی‌های اساسی آن‌ها را نشان می‌دهد و به ما کمک می‌کند تا با انواع مختلف آزمون و کاربردهای آن‌ها آشنا شویم. این دسته‌بندی‌ها می‌تواند در انتخاب روش‌های مناسب برای آزمون نرم‌افزار در مراحل مختلف توسعه مؤثر باشد.



شکل ۳.انواع آزمون نرم‌افزار

شکل ۳ انواع مختلف آزمون نرم‌افزار را نشان می‌دهد که از مدل آبشار سنتی استفاده می‌شود. اگر پیاده‌سازی داخلی سیستم مشخص باشد و آزمون بر اساس آن انجام شود، آزمون جعبه سفید است. اگر آزمون‌کننده اطلاعی در مورد پیاده‌سازی نداشته باشد و آزمون را انجام دهد، آنگاه آزمون جعبه سیاه است. آزمون جعبه خاکستری ترکیبی از هر دو نوع است.

[۱] waterfall

۲-۲- خودکارسازی

در ابتدا، فرآیندهای توسعه نرم‌افزار همگی به صورت دستی و زمان‌بر انجام می‌شدند. نخستین گام در بهبود آزمون نرم‌افزار، خودکارسازی بود که امکان اجرای مکرر آزمون‌ها را بدون دخالت دستی و مقایسه خودکار نتایج فراهم کرد. امروزه با خودکار شدن تولید موارد آزمون، داده‌های آزمون و اجرای آن‌ها، فرآیند آزمون به مراتب تسهیل شده است.

در خودکارسازی، استفاده از ابزارهای مختلف به‌طور خودکار انجام می‌شود و سایر فنون آزمون نرم‌افزار خودکارسازی محبوب عبارتند از:

- آزمون خودکارسازی
- ضبط/بازپخش
- اسکریپت‌نویسی
- آزمون مبتنی بر داده
- آزمون مبتنی بر کلمه کلیدی
- آزمون موازی
- آزمونAPI
- آزمون برنامه‌نویسی سوکت
- آزمون وب‌سرویس

با رشد سیستم‌های مقیاس بزرگ که کل فرآیند آزمون را به‌صورت خودکار انجام می‌دهند، انجام کامل آزمون می‌تواند خسته‌کننده، زمان‌بر و پیچیدگی فرآیند را افزایش دهد. انتخاب مناسب روش آزمون و مدیریت خطرات مالی بالقوه ناشی از نتایج نادرست خودکارسازی، چالش‌های رایج هستند (Li et al., ۲۰۱۹).

۲-۳- چابک

در روش‌های چابک، آزمون همزمان با توسعه نرم‌افزار شروع می‌شود. این یک تغییر بزرگ در نحوه آزمون نرم‌افزار است، چون آزمون‌ها از همان ابتدا انجام می‌شوند و برخلاف روش‌های قدیمی، تکراری و قابل تغییر هستند. با توجه به اینکه استراتژی و تاکتیک‌های آزمون[۲] (TSTT) در چابک به صورت مرحله‌ای و قابل تطبیق هستند، روش‌های مختلفی مثل توسعه آزمون محور، آزمون پذیرش، توسعه بر مبنای رفتار، آزمون اکتشافی و تکنیک‌های مبتنی بر جلسه استفاده می‌شوند (Para et al., ۲۰۱۷).

آزمون رگرسیون در چابک خیلی مهم است، چون نرم‌افزار نسخه‌های کوتاهی را پشت سر می‌گذارد. یکی از اصول چابک این است که از تغییرات استقبال می‌کند. اما اگر نیازمندی‌ها خیلی زیاد تغییر کنند، آزمون زمان‌بر می‌شود، چون آزمون رگرسیون زمان زیادی می‌خواهد. اگر نیازمندی‌ها درست تعریف نشوند، کل محصول مشکل پیدا می‌کند و تغییرات زیاد می‌توانند کیفیت را پایین بیاورند. همچنین، به خاطر کمبود وقت، ممکن است آزمون‌های امنیتی یا غیرعملکردی انجام نشوند یا محدود باشند. تکنیک‌های

[۲] Test Strategy and Test Tactics

مختلفی برای آزمون رگرسیون وجود دارد و ممکن است آزمون‌کننده‌ها با مشکل آزمون‌های تکراری روبرو شوند که با چند بار اجرا روی یک کد، هم ممکن است قبول شوند و هم رد شوند (Lam et al., ۲۰۲۰).

۲-٤- اپلیکیشن‌های موبایل

فرآیند آزمون در برنامه‌های تلفن همراه نیز اجرا می‌شود، جایی که پلتفرم‌های آزمون با یکسان بودن عملکرد آزمون نصب، TSTT اصلی متفاوت است. علاوه بر آزمون خودکار، آزمون رابط، آزمون نشت حافظه[۳]، آزمون وقفه و آزمون گواهینامه [۴]نیز برای برنامه‌های کاربردی تلفن همراه استفاده می‌شود. مشکلات متعددی در آزمون برنامه‌های تلفن همراه وجود دارد، از جمله تقسیم‌بندی سیستم‌عامل، آزمون بر روی اندازه‌های مختلف صفحه نمایش تلفن همراه، پاسخ‌دهی، شبکه‌های مختلف تلفن همراه، رابط‌ها و قابلیت استفاده. اگرچه شبیه‌سازهایی برای چنین آزمون‌ی وجود دارد، اما این شبیه‌سازها نمی‌توانند سناریوهای بلادرنگ را به‌خوبی منعکس کنند و دامنه آزمون محدود است.

۲-٥- داده‌های حجیم

همان‌طور که سیستم‌ها رشد می‌کنند، داده‌های آن‌ها نیز با افزایش پیچیدگی سیستم‌های داده‌ای که راه را برای داده‌های بزرگ هموار می‌کند، افزایش می‌یابند. داده حجیم یا داده‌های حجیم کاملاً با رویکردهای آزمون‌ی سنتی متفاوت است، زیرا آزمون داده‌های حجیم مستلزم بررسی حجم زیادی از داده‌های ساختاریافته و بدون ساختار است. الگوریتم‌های خودکارسازی به دریافت ارزش از داده‌های بزرگ کمک می‌کنند. در اینجا، مجموعه داده‌های بزرگ برای پنج مورد آزمون قرار می‌گیرند، نه فقط سیستم به تنهایی. این موارد شامل تجزیه و تحلیل داده‌ها، آزمون حجم داده‌ها، آزمون صحت داده‌ها، آزمون سرعت داده‌ها و آزمون تنوع داده‌ها است. فنونی که برای آزمون داده حجیم استفاده می‌شوند خاص هستند و چالش‌های اصلی آن شامل آزمون داده‌های ناهمگن و گسترده، درک روابط داده‌ها، امنیت و مقیاس‌پذیری است.

۲-٦- رایانش ابری

رایانش ابری و داده‌های حجیم به هم مرتبط هستند. در رایانش ابری، اطلاعات از مکان‌های دور قابل دسترس و پردازش می‌شوند. آزمون در محیط ابری معمولاً با شبیه‌سازی داده‌های ترافیک وب و داده‌های واقعی انجام می‌شود تا برنامه‌های کاربردی وب را در شرایط مبتنی بر ابر آزمون کنیم (Wang et al., ۲۰۱۷).

در آزمون ابری، تکنیک‌های خاصی وجود دارد، از جمله:

- **آزمون بازیابی فاجعه[۵]:** برای بررسی توانایی سیستم در بازگرداندن داده‌ها پس از یک فاجعه.
- **آزمون چند اجاره‌ای [۶]:** برای ارزیابی عملکرد سیستم در شرایطی که چندین کاربر به طور همزمان از آن استفاده می‌کنند.
- **آزمون امنیتی:** برای شناسایی و رفع مشکلات امنیتی.

---

[۳] Memory Leak Testing
[۴] Certification Testing
[۵] Disaster Recovery Testing
[۶] Multi-tenancy Testing

- **متعادل‌سازی بار[۷]:** برای توزیع بهینه بار کاری [۸]بین سرورها.
- **تکنیک‌های خوشه‌بندی:** برای مدیریت و سازمان‌دهی داده‌ها.

داده‌ها از طریق پلتفرم‌های مختلف و سرورهای متنوع جمع‌آوری می‌شوند. اما آزمون داده‌ها چالش‌هایی دارد، مانند تخمین دقت و سازگاری داده‌ها، زیرا این داده‌ها از چندین کانال عبور کرده‌اند. یکپارچگی داده‌ها نیز مسئله‌ای مهم است. علاوه بر این، مشکلاتی در زمینه مقیاس‌پذیری، ذخیره‌سازی و انتقال داده‌ها بین ابرها وجود دارد (Wang et al., ۲۰۱۷).

۲-۷- DevOPS و QAOps

روند آزمون چابک و خودکارسازی به سادگی در دواپس پیاده‌سازی می‌شود و کل چرخه عمر نرم‌افزار را با یکپارچگی مداوم و تحویل ترکیب می‌کند. این شامل کواپس [۹]مستمر و چارچوب تمام فنون آزمون چابک است که عمدتاً شامل آزمون خودکار، آزمون رگرسیون، آزمون یکپارچه‌سازی، آزمون موازی و آزمون اکتشافی می‌شود. همچنین فنون آزمون مقیاس‌پذیری نیز در این روند مورد استفاده قرار می‌گیرند. با وجود کدهای بزرگ، دستیابی به پوشش ۱۰۰ درصدی کد در آزمون دشوار است. تمایل به یکپارچه‌سازی و استفاده از فناوری‌های DevOps مانند تحویل مداوم وجود دارد. در این شرایط، زمان به عنوان یک عامل حیاتی در نظر گرفته می‌شود و پوشش کامل کد در چنین بازه‌های کوتاه زمانی قابل تحویل، واقعاً چالش‌برانگیز است (Faber, ۲۰۲۰).

۲-۸- هوش مصنوعی

با ظهور هوش مصنوعی و یادگیری ماشین، تحول بزرگی در فرآیند آزمون نرم‌افزار ایجاد شده است. از تولید موارد آزمون گرفته تا جمع‌آوری داده‌های آزمون، اجرا و مقایسه نتایج، همه این مراحل توسط الگوریتم‌های هوش مصنوعی مدیریت می‌شوند. هوش مصنوعی با استفاده از الگوریتم‌های یادگیری ماشین، کارایی آزمون نرم‌افزار را بهبود می‌بخشد و نقش کلیدی در این فرآیند ایفا می‌کند. فنون مختلفی مانند آزمون بلاکچین، آزمون سیستم‌های قرارداد هوشمند[۱۰]، آزمون همتا/گره[۱۱]، خودکارسازی پیشرفته، آزمون شبکه امنیت سایبری، آزمون مبتنی بر قانون، آزمون طبقه‌بندی، آزمون مبتنی بر مدل داده و آزمون دگرگونی (غیر جعبه‌ای) [۱۲]در این زمینه استفاده می‌شوند (Gao et al., ۲۰۱۹). همچنین، فنون آزمون شامل نظارت شده، بدون نظارت و تقویتی نیز به کار می‌روند.

آزمون نرم‌افزار اکنون شامل آزمون داده‌های آزمون واکنشی پویا است. در اینجا مشکل تعیین خروجی صحیح برای ورودی مشخص به عنوان "Test Oracle" مطرح می‌شود که یکی از چالش‌های رایج در زمینه هوش مصنوعی و یادگیری ماشین است. این مشکل به دلیل غیرقطعی بودن الگوریتم‌ها به وجود می‌آید، جایی که نمی‌توان خروجی مشابهی را برای ورودی یکسان انتظار داشت. الگوریتم‌های یادگیری ماشین برای آزمون پیچیده هستند زیرا به ورودی‌های انسانی متکی هستند و برای پیش‌بینی‌ها به

---

[۷] Load Balancing
[۸] Workload
[۹] QAOps
[۱۰] Smart Contract Systems Testing
[۱۱] Peer/Node Testing
[۱۲] Metamorphic Testing (Non-Box)

حجم داده‌های آموزشی نیاز دارند. معیارهای ارزیابی آزمون، تعیین معیارهای آزمون و تولید آزمون نرم‌افزار مبتنی بر جستجو از دیگر چالش‌ها در این زمینه به شمار می‌روند.

۲-۹- اینترنت اشیا

اینترنت اشیا در چند سال گذشته شتاب قابل توجهی گرفته است. جایی که آزمون دستگاه‌های اینترنت اشیا شامل آزمون سخت‌افزار، نرم‌افزار و لوازم جانبی مربوطه و همچنین ارتباط بین آن‌ها است. این فرآیند مستلزم مجموعه‌ای از مهارت‌های تخصصی در آزمون است که شامل دانش عمیق از تمام اجزای اینترنت اشیا می‌شود. دستگاه‌های اینترنت اشیا معمولاً از چهار جزء اصلی تشکیل شده‌اند: حسگر، اپلیکیشن، شبکه و همگام‌سازی. آزمون قابلیت همکاری، آزمون دستگاه به دستگاه و آزمون وقفه از جمله آزمون‌های خاصی هستند که بیشتر به اینترنت اشیا اختصاص دارند. با ظهور چشمگیر اینترنت اشیا در سال‌های اخیر، آزمون امنیتی به یک چالش بزرگ برای آزمون‌کنندگان نرم‌افزار تبدیل شده است. نقاط آسیب‌پذیر زیادی در دستگاه‌های اینترنت اشیا وجود دارد که در سیستم‌های ادغام شده‌اند، زیرا تعداد پیوندها برای نقاط دستگاه متصل به سرعت در حال افزایش است. آزمون سازگاری متقابل دستگاه نیز یکی دیگر از نگرانی‌ها و چالش‌ها است که باید به آن توجه زیادی داشت. فنون آزمون استاندارد برای اینترنت اشیا هنوز به‌طور کامل تعریف نشده‌اند. در اینجا، جدا از مسائل امنیتی، آزمون‌کننده باید دانش عمیقی در زمینه نرم‌افزار، سخت‌افزار، رابط‌ها و به‌ویژه شبکه‌ها داشته باشد. نیاز به وجود افراد باکیفیت و با دانش عمیق در این حوزه، یک نیاز جدی خواهد بود (Bures et al, ۲۰۱۸).

۳- نتیجه‌گیری

روند بازار به سمت فناوری‌های مرتبط با هوش مصنوعی، اینترنت اشیا، چابک، ابر، بلاکچین، واقعیت مجازی، اپلیکیشن‌های موبایل، محاسبات لبه، روباتیک و دیگر حوزه‌های نوین در حال تغییر است. در این شرایط، آزمون‌کننده‌ها باید در زبان‌های برنامه‌نویسی تخصص کامل داشته باشند تا بتوانند تمامی فناوری‌های پرطرفداری را که در آن کدنویسی و خودکارسازی ضروری است، آزمون کنند. آزمون نرم‌افزار برای اطمینان از کیفیت محصول بسیار اهمیت دارد. فنون آزمون نرم‌افزار را می‌توان از ابعاد مختلفی مانند فناوری مورد استفاده، سهامداران، عملکرد، روش‌های توسعه یا خود مرحله توسعه مشاهده کرد. این مقاله به بررسی فنون و چالش‌های رایج در عرصه آزمون نرم‌افزار می‌پردازد. برای غلبه بر چالش‌های موجود در آزمون پلتفرم‌های تکه‌تکه[۱۳]، می‌توان مدل‌های خاص آزمون نرم‌افزار تعمیم‌یافته‌تری را توسعه داد. این مدل‌ها می‌توانند به آزمون‌گران کمک کنند تا با پیچیدگی‌های فناوری‌های نوین و نیازهای متنوع بازار بهتر سازگار شوند و در نتیجه کیفیت محصولات نرم‌افزاری را بهبود بخشند.

[۱۳] Fragmented Platforms

٤- مراجع

J.Jenny Li, Andreas Ulrich, Xiaoying Bai & Antonia Bertolino, ٢٠١٩. Advances in test automation for software with special focus on artificial intelligence and machine learning. Software Quality Journal.

Pulasthi Perera, Roshali Silva & Indika Perera, ٢٠١٧. Improve Software Quality through Practicing DevOps. International Conference on Advances in ICT for Emerging Regions.

Wing Lam,August Shi,Reed Oei,Sai Zhang,Michael D. Ernst & Tao Xie, ٢٠٢٠. Dependent-Test-Aware Regression Testing Techniques. ISSTA '٢٠, Association for Computing Machinery.

Lizhe Wang, Rajiv Ranjan, Jinjun Chen and Boualem Benatalla, ٢٠١١. Cloud Computing: Methodology, Systems, and Applications. CRC Press, Taylor & Francis group.

Frank Faber, ٢٠٢٠. Testing in DevOps. In: Goericke S. (eds) The Future of Software Quality Assurance.

Jerry Gao, Chuanqi Tao, Dou Jie & Shenqiang, ٢٠١٩. What is AI Software Testing? and Why? ٢٠١٩ IEEE International Conference on Service-Oriented System Engineering (SOSE).

Miroslav Bures, Tomas Cerny & Bestoun S. Ahmed, ٢٠١٨. Internet of Things: Current Challenges in the Quality Assurance and Testing Methods. ٩th iCatse Conference on Information Science and Applications ٢٠١٨.