ویژه کنترل کیفیت و بررسی نرم افزار

بررسی کیفیت نرم‌افزار و سیستم‌های مدیریت آن

۱۶

چکیده

ضرورت تهیه و تدوین استاندارد لازم برای تولید هر محصول، بر همگان روشن است و نرم‌افزار رایانه‌ای که به‌عنوان مقوله‌ای فراگیر و بسیار مهم در جهان صنعتی و مبتنی بر فناوری اطلاعات و ارتباطات شناخته می‌شود، از این قاعده مستثنا نبوده و نیست. در طی چند دهه گذشته و در مقابله با بحران موجود در تولید محصولات نرم‌افزاری، متخصصان و صاحب‌نظران، با به وجود آوردن علوم مهندسی نرم‌افزار، تلاش‌های ارزنده بسیاری را در راستای قوام بخشیدن، برقراری و بهبود نظام تولید نرم‌افزار و برنامه‌های رایانه‌ای به‌انجام رسانیده‌اند؛ اما ویژگی‌های خاص این صنعت که در اثر رشد فناوری، با جهش سریع و تغییر زودهنگام نسل‌ها، و نیز با رُشد فزاینده و همه‌جانبه کاربردهای آن همراه بوده است، زمینه‌ساز تداوم برخی ابهام‌ها و پیچیدگی‌ها و همچنین، به وجود آمدن مسائل جدید در مبانی و امور مرتبط با آن می‌باشد. با وجود فعالیت‌های گوناگون توسط مؤسسات استانداردسازی، هنوز استانداردهای جامع و مورد پذیرش جهانی در این زمینه موجود نیست و استانداردها و رهنمودهای حاصل از این فعالیت‌ها، همچنان سیر تکاملی خود را سپری می‌نمایند.

کلیدواژگان: سیستم مدیریت، سیستم کیفیت، کیفیت نرم‌افزار.

مقدمه

کیفیت، به‌عنوان یکی از مهم‌ترین مفاهیم در مهندسی نرم‌افزار، نظر بسیاری از کارشناسان را به خود جلب نموده و تلاش‌های بسیاری را به خود اختصاص داده است. در این مقوله، همواره دو دیدگاه «کنترل کیفیت محصول نهایی» و «تضمین کیفیت تولید»، به‌عنوان راهکارهای مستقل و یا مکمل یکدیگر، در نظام کیفیت نرم‌افزارهای رایانه‌ای مطرح بوده‌اند. نتایج فعالیت‌های مذکور، به صورت مجموعه‌ای از کتب، استانداردها و رهنمودهای کلی تحت عناوین: تعریف کیفیت، فرایند تضمین کیفیت، ارزیابی و کنترل کیفیت، و یا مهندسی کیفیت، به جامعه کاربران برنامه‌های رایانه‌ای عرضه شده است. در بسیاری از متون فنی، این‌چنین عنوان شده که کیفیت بر مبنای خواسته‌های مشتری تعریف می‌شود. اگرچه این نظریه به‌طورکلی صحیح است، اما به همین دلیل (کلی بودن)، فاقد ارزش عملیاتی و غیرقابل اتکاست؛ زیرا خواسته‌ها، خود مبتنی و تحت‌الشعاع پارامترهای مختلف مانند: زمان، مکان، رشد فناوری، توان مالی، ادراک، سلیقه، قدرت پذیرش و دیگر محدودیت‌های موجود در محیط تولید و استفاده هستند و بنابراین، با تکیه بر این نگرش، کیفیت به مقوله‌ای نسبی، پارامتری، ناهمگون و تقریباً غیرقابل ارزیابی مبدل می‌شود. روشن است که گام اوّل در حل این مسئله، تخصیص مقادیر مشخص به پارامترهای فوق، و یا محدود کردن آنها، در حدّ ضرورت یا امکان می‌باشد؛ به عبارت دیگر، برای به وجود آوردن امکان اظهار نظر درباره کیفیت یک محصول، لازم است که تعریف کاربردی و غیرپارامتری برای آن ارائه شود که قابلیت استفاده در فرایندهای تولید، تضمین و ارزیابی کیفیت را دارا باشد.

تعریف کنترل کیفیت (Quality control)

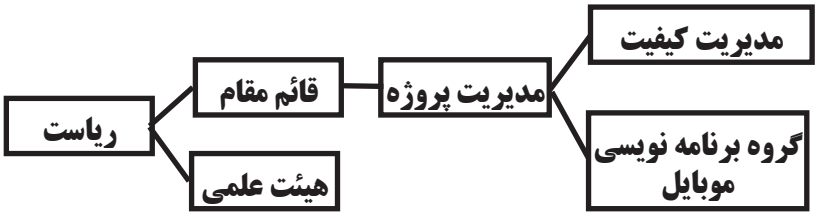
مجموعه‌ای از فعالیت‌ها برای ارزیابی و اطمینان از اینکه استانداردها و رویه‌های کیفی توسط تیم تولید و توسعه پروژه درست انجام گرفته باشد، کنترل کیفیت نام دارد.(۱)؛

در حقیقت، کنترل کیفیت به استانداردها آشناست و به آنها آگاهی دارد و اجرای آن را بررسی و نتیجه را مشاهده می‌کند. مجموعه ویژگی‌های یک کالا و عرضه‌کننده آن، از لحاظ کیفی، کمّی، بها و غیره، باعث می‌شوند آن کالا مورد تقاضا قرار گیرد و فروش رود. کیفیت، در مقابل کمیت نیز قرار دارد. چگونگی و ماهیت یک پدیده یا شئ را کیفیت شئ یا پدیده می‌نامند و همچنین، محتوا نیز شامل کیفیت می‌باشد.

کیفیت یک محصول یا خدمت، همان چیزی است که مشتری طلب می‌کند. اگر محصول یا خدمت ارائه‌شده در جنبه‌هایی خاص، کیفیت بسیار بالایی داشته باشد، اما مشتری بدان نیاز نداشته باشد، تنها هزینه‌ای است که پرداخته شده؛ ولی ارزش افزوده‌ای ایجاد نکرده است. برای کنترل یک فرآیند و دستیابی به یک فرآیند با کیفیت، مدل‌ها و تئوری‌های فراوانی ارائه شده و برای هرکدام از این مدل‌ها، ابزارهای کنترلی ایجاد شده که به وسیله آنها می‌توانیم کیفیت فرآیند را زیر نظر بگیریم. کیفیت محصول، برآورده کردن نیاز امروز و فردای مشتری و همچنین، شایستگی آن جهت استفاده و تطابق آن با نیازهای کاربران است. سازمان جهانی استاندارد نیز کیفیت را این‌گونه تعریف کرده: تمامی ویژگی‌های یک محصول که در توانایی آن برای برآورده نمودن نیازهای تصریح‌شده یا تلویحی مؤثر، مثل راحت بودن کار با نرم‌افزار است. کیفیت یک فرآیند و یک محصول را می‌توان از زوایای متعددی بررسی کرد؛ اما همان‌گونه که در تعاریف سنّتی مشاهده کردیم، کیفیت در واقع برآورده کردن نیازهای مشتریان است. از این رو، شناخت و تحلیل نیازهای مشتریان، از جایگاه ویژه‌ای برخوردار است. در این راستا، مدل‌ها و روش‌های مختلفی برای تولید و توسعه نرم‌افزار وجود دارد؛ مانند: کانو، اجایل، pmbok و نایبلر که در جای مناسب به آنها خواهیم پرداخت.

در کتاب مدیریت پروژه، کنترل کیفیت این‌طور تعریف شده است که: نظارت بر نتایج مشخص پروژه، به منظور تعیین انطباق آنها با استانداردهای کیفیت مرتبط و شناسایی

نمودار اوّل



راه‌هایی برای حذف علل عملکرد ناخوشایند. این فرایندها با یکدیگر و همچنین با فرایندهای سایر حوزه‌های دانش، تعامل دارند. ممکن است، هر فرایند بر مبنای نیازهای پروژه تلاش یک یا تعداد بیشتری از افراد یا گروه‌هایی از آنان را در برداشته باشد. هر فرایند به طور معمول، حداقل یک بار در هر مرحله پروژه به وقوع می‌پیوندد.

دو نمودار در رابطه با جایگاه کنترل کیفیت وجود دارد که در حال حاضر، اجرایی است :

نمودار اوّل، مربوط به کتاب مدیریت پیکره دانش می‌باشد و این همان روش pmbok می‌باشد. در این روش، مدیریت پروژه، مسئولیت هماهنگی‌ها را بر عهده دارد و پس از اطمینان از تولید محصول، آن را به مدیریت کیفیت می‌دهد و پس از بررسی و تأیید، گزارش آن را به مدیریت پروژه بدهد تا تصمیم نهایی برای تولید و عرضه محصول در بازار گرفته شود.

اما نوع دیگری از جایگاه کنترل کیفیت وجود دارد (نمودار دوم) که در بسیاری از شرکت‌های خارجی، به‌خصوص مایکروسافت و گوگل اجرا می‌شود. در این روش، مدیریت کیفیت، بخش مستقلی است که پس از تعامل با مدیریت پروژه، گزارش فعالیت خود را در اختیار قائم مقام یا هیئت مدیره برای تصمیم نهایی محصول تقدیم می‌کند.

در جای خود، به توضیح بیشتری درباره جایگاه کنترل کیفیت خواهیم پرداخت.

تعریف بررسی نرم‌افزار (پروژه) (Software Review)

بررسی نرم‌افزار، مجموعه‌ای از فعالیت‌هاست که امکانات و پیش‌بینی‌هایی را که در نرم‌افزار انجام گرفته تا کار با نرم‌افزار را آسان کند، بررسی می‌نماید.(۲) در واقع، در بررسی نرم‌افزار، اعمال اصول و شیوه‌های استاندارد در یک پروژه انجام می‌شود تا سیستم با مشکل مواجه نشود. به نظر می‌رسد، زمانی که کنترل کیفیت بخواهد برای کیفیت نرم‌افزار تضمین دهد، ناگزیر است که برخی تست‌های نرم‌افزار نهایی، نسخه آلفا و بتای محصول را به صورت دستی و توسط شخصی که خود را در جایگاه کاربر نهایی قرار می‌دهد، انجام دهد.

در متولوژی اجایل، برای بررسی نرم‌افزار، جایگاه خاصی در نظرگرفته شده است. به صورت واقعی، کاربر نهایی از اوّلین نسخه‌هایی که نرم‌افزار بیرون می‌آید یا در پاره‌ای از موارد، بعد از تشکیل زیرسیستم‌ها در اختیار کاربر قرار داده می‌شود و توسعه بر اساس بازخوردی که از کاربران گرفته می‌شود، انجام می‌گردد و این، برتری متدولوژی اجایل نسبت به دیگر متولوژی‌های مورد استفاده خواهد بود.

اجرای یک برنامه با هدف پیداکردن خطا

– تست خوب: احتمال پیدا کردن خطاهای کشف‌نشده توسط ارزیابی زیاد است.

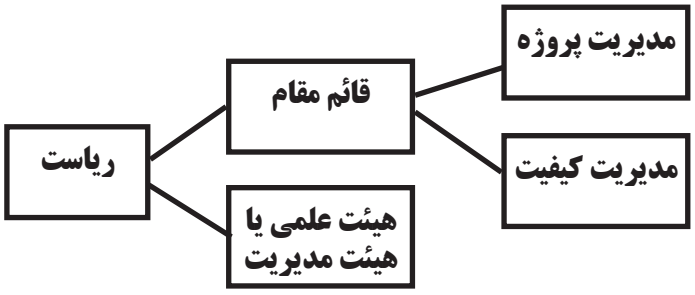
–تست موفق: حداقل یک خطای کشف‌نشده را بیابد. تست فقط وجود خطا را نشان می‌دهد و نه عدم وجود آن را. پیدا نشدن خطا در تست، به معنای بدون خطا بودن برنامه نیست.

تست از اجزای کوچک شروع می‌شود؛ ضمن اینکه تست کامل برای مؤثربودن، باید توسط شخص ثالث بی‌طرف انجام شود.

هدف از به‌کارگیری محک‌های کیفیت نرم‌افزار

کیفیت نرم‌افزار، درجه‌ای است که به اندازه آن، محصول نرم‌افزاری، دارای ترکیب دلخواهی از ویژگی‌هاست. این ترکیب دلخواه، باید به‌روشنی تعریف شود؛ در غیر این صورت، برآورد آن به صورت منطقی امکان‌پذیر نخواهد بود. انتخاب مجموعه‌ای مناسب از محک‌ها که با عوامل و خصوصیات انتخابی مرتبط هستند، گام بعدی استفاده‌کننده از این مجموعه استانداردها می‌باشد.

هدف از به‌کارگیری محک‌های نرم‌افزاری، ایجاد امکان برآورد در چرخه حیات است؛ به گونه‌ای که میزان برآورده شدن نیازمندی‌های کیفی قابل تشخیص گردد. اگرچه



نمودار دوم

محک‌های نرم‌افزاری، با فراهم آوردن یک پایه کمّی (مقداری)، امکان تصمیم‌گیری نسبی در مورد کیفیت نرم‌افزار را فراهم می‌آورند، اما استفاده از آنها، قضاوت انسانی را در ارزیابی کیفیت نرم‌افزار حذف نمی‌نماید. انتظار می‌رود، با توجه به آشکار نمودن جزئیات پنهان و ویژگی‌های خاص محصولات نرم‌افزاری، استفاده از محک‌های نرم‌افزاری در یک سازمان یا یک پروژه، کوششی مؤثر و مفید باشد.

طبق نظر پروفسور جوران(۳)، «کیفیت» دارای دو مفهوم مکمل است:

اوّل: هدف، کیفیت بالاتر، ایجاد رضایت بیشتری در مشتری و انتظار درآمد افزون‌تر است. ایجاد ویژگی‌های بیشتر و بهتر کیفیت، مستلزم سرمایه‌گذاری و افزایش هزینه است. از این زاویه، کیفیت بالاتر هزینه‌بردار است؛ اما برای درآمد مطلوب‌تر، لازم می‌نُماید.

دوم: کیفیت، یعنی عدم عیب و خطا. عیب و خطا موجب ضایعات، تعمیرات، دوباره‌کاری، محصول مرجوع، نارضایتی مشتری، شکایت، جریمه و از دست دادن مشتری می‌گردد و همه اینها هزینه‌آور هستند. بدین ترتیب، کیفیت بالاتر حاوی عیب و خطای کمتر و در نتیجه، هزینه کمتر است.

موضوع کیفیت و بهبود آن، موضوع مهم و حیاتی در هر سازمان و شرکت است. برخی کیفیت را تنها انطباق با ضوابط از پیش‌تعیین‌شده می‌دانند؛ به عبارت دیگر، کیفیت مترادف با استاندارد تلقی می‌شود؛ غافل از آنکه امروزه کیفیت مفهومی وسیع‌تر داشته و به معنای پاسخگویی به درخواست‌های منطقی و حق‌مدارانه مشتریان است. اگرچه انطباق با استاندارد نیز در سطوح ملی و بین‌المللی می‌تواند درجه‌ای از کیفیت محسوب شود، اما همه ابعاد کیفیت نیست. بر این اساس، در شرایط کنونی، هدف از کیفیت، علاوه بر اجرای استانداردها، کسب رضایت بیشتر مشتری در همه ابعاد محصول را نیز شامل می‌شود.

شرکت‌ها و مؤسساتی که در تولید نرم‌افزار در سطح جهان فعالیت می‌کنند، شیوه‌ها و روش‌های مختلفی برای خود دارند که در چرخه طول عمر نرم‌افزار از این روش‌ها و شیوه‌ها استفاده می‌کنند تا آنکه نرم‌افزار طول عمرش تمام بشود و بایگانی یا نسخه تازه‌تری از آن ارائه شود.

استانداردهای بررسی نرم‌افزار

در توسعه یک نرم‌افزار مجموعه فعالیت‌هایی داریم که بروز اشتباهات انسانی در آنها غیرقابل اجتناب است. پس از آنکه کد منبع تولید شد، نرم‌افزار باید برای خطاهای احتمالی و موجود مورد ارزیابی قرار گیرد. بنابراین، هدف، طراحی حالاتی است که نرم‌افزار تحت آنها تست گردد و احتمال یافتن خطا توسط آنها زیاد باشد. تحت این تکنیک‌ها منطق درونی قطعات نرم‌افزار و همچنین، دامنه‌های ورودی و خروجی به کل نرم‌افزار مورد بررسی قرار می‌گیرد. در مراحل اوّلیه، مهندسان نرم‌افزار این کار را انجام می‌دهند و با توسعه بیشتر، متخصصان تست.

معیارهای تست‌پذیر بودن نرم‌افزار(٤)

این معیارها عبارت‌اند از:

۱. قابلیت اجرا (Operability): هرچه نرم‌افزار بهتر کار کند و در محیط‌های بیشتری قابل اجرا باشد، n بهتر قابل ارزیابی است.

۲. مشاهده‌پذیری (Observability): قابلیت مشاهده نتایج ارزیابی.

۳. کنترل‌پذیری (Controlability): قابلیت اجرای تست‌های خودکار؛ مثل امکان اجرای خودکار تست‌های واحد توسط jUnit برای زبان جاوا.

۴. تجزیه‌پذیری (Decomposability): ارزیابی می‌تواند هدفمندتر شود.

۵. سادگی (Simplicity): کاهش پیچیدگی معماری و منطق برنامه.

۶. پایداری (Stability): برای ارزیابی، تغییرات کمی بخواهد.

۷. درک‌پذیری (Understandability): قابلیت درک طراحی و وابستگی‌های بین اجزا.

سطوح مختلف تست

این سطوح عبارت‌اند از:

۱. تست واحد (Unit testing)؛

۲. تست یکپارچه‌سازی (Integration testing)؛

۳. تست سیستم (System testing)؛

۴. تست پذیرش (Acceptance testing):

– تست آلفا (Alpha test)؛

– تست بتا (beta test).

*** تست واحد**

تست واحد یا micro level، پایین‌ترین سطح تست است. هر کد تست واحد، یک قطعه کد یا یک تابع (متد) خاص را تست می‌کند. این تست، نیاز به دانش در مورد

طراحی و شیوه عملکرد داخلی تابع یا قطعه کد دارد. این نوع تست، توسط برنامه‌نویس انجام می‌شود؛ نه تست‌کننده.

*** تست یکپارچه‌سازی افزایشی**

تست یکپارچه‌سازی افزایشی با افزوده شدن قابلیت جدید به نرم‌افزار، اجرا می‌شود. هدف این تست، بررسی درستی نرم‌افزار پس از افزوده شدن امکان جدید است. امکانات نرم‌افزار باید از هم استقلال داشته باشند تا بتوان پیش از تکمیل کل نرم‌افزار و به صورت افزایشی، نرم‌افزار را تست کرد. این فرایند، توسط برنامه‌نویس یا تیم تست انجام می‌شود.

*** تست یکپارچه‌سازی**

این تست، حاصل از کنار هم قرار گرفتن قطعات مختلف نرم‌افزار به منظور بررسی درستی عملکرد آن است و این نرم‌افزار یکپارچه‌شده، شامل قطعه کدها، یعنی ماژول‌هایی از کد برنامه‌های مجزاست که در کنار هم، برنامه اصلی را تشکیل می‌دهند. برنامه‌های مشتری ـ کارگزارِ عمل‌کننده در یک شبکه، پس از تست واحد انجام می‌شود.

*** تست سیستم**

این تست به منظور بررسی عملکرد نرم‌افزار روی پلتفرم‌های مختلف و نرم‌افزارهای پلتفرم OS انجام می‌شود. سخت‌افزار و نرم‌افزار، یعنی موارد کاربردی مورد نیاز، به منظور اطمینان از این امر است که برنامه با مؤلفه‌های دیگر محیط اجرایش به‌خوبی کار می‌کند و نیز جهت اطمینان از اینکه نرم‌افزار ارائه‌شده در محیط مورد نظر قابل استفاده است. نمونه مشکل حاصل از انجام ندادن تست سیستم نرم‌افزار، بازی شیر شاه دیزنی (Disney's Lion King Game) در پاییز سال ۱۹۹۴ است. شرکت دیزنی اوّلین CD بازی خود تحت عنوان شیر شاه Lion King را که بر اساس کارتونی به همین نام ساخته شده بود، وارد بازار کرد. بسیاری از شرکت‌های دیگر تا آن زمان اقدام به ساخت بازی‌های رایانه‌ای کرده بودند؛ اما این اوّلین‌بار بود که شرکت دیزنی وارد این تجارت شده بود. دیزنی برای فروش این بازی، دست به تبلیغات گسترده‌ای زد و در نتیجه، این محصول با فروش بسیار بالایی مواجه شد؛ اما اتفاقات پس از آن، تبدیل به کابوسی برای این شرکت شد. در ۲۶ دسامبر، روز پس از کریسمس، تلفن‌های بخش پشتیبانی مشتریان شرکت دیزنی شروع کرد به زنگ زدن، زنگ زدن و زنگ زدن! متصدیان پاسخگویی به تماس‌ها با خیل عظیمی از والدین عصبانی با بچه‌های گریان مواجه شدند که ادعا می‌کرند نرم‌افزار مزبور کار نمی‌کند. این خبر به‌سرعت در مطبوعات و تلویزیون نیز پخش شد و کریسمس آن سال را برای بسیاری از پرسنل دیزنی تلخ کرد. علت چه بود؟ پس از بررسی مشخص شد که دیزنی نرم‌افزار خود را روی بسیاری از مدل‌های PC تست نکرده بود و در نتیجه، تنها روی سیستم‌هایی کار می‌کرد که برنامه‌نویسان دیزنی روی آن سیستم‌ها نرم‌افزار خود را توسعه داده بودند و نه دستگاه‌های متداولی که عموم مردم از آن استفاده می‌کردند.

*** تست پذیرش**

تست پذیرش به منظور بررسی اینکه نرم‌افزار نیازهای مشتری را برآورده می‌کند، انجام می‌شود و بعد از تست سیستم انجام می‌شود. این نوع تست، شامل موارد ذیل است:

ـ تست آلفا: در پایگاه توسعه‌دهنده نرم‌افزار و در اغلب موارد، توسط n کارمندان داخلی و گاهی نیز، توسط مشتری تعدادی از کاربرانش که به محل دعوت می‌شوند، انجام می‌گیرد.

ـ تست بتا: در این تست، نسخه‌هایی از نرم‌افزار در اختیار تعدادی از کاربران قرار می‌گیرد تا در بازه‌ای با آن کار کنند و خطاها را گزارش دهند.

روش‌های ارزیابی

۱. روش جعبه سفید: دانستن شیوه کار داخلی برنامه، امکان تأیید چگونگی عمل هر تکه کد و مسیر اجرا مراحل اولیه ارزیابی.

۲. روش جعبه سیاه: دانستن عمل مورد انتظار و مطلوب، و امکان تأیید کاری که سیستم باید انجام دهد. مراحل انتهایی ارزیابی استراتژی تست نرم‌افزار، توصیفی رسمی است از اینکه نرم‌افزار چگونه تست خواهد شد. هدف استراتژی تست، تعریف همه مراحل برای فرایند تست نرم‌افزار است که شامل برنامه‌ریزی آزمایش، طراحی ابزار آزمایش، اجرای آزمایش و جمع‌آوری و ارزیابی داده‌های به‌دست‌آمده باشد.

استراتژی جعبه سیاه

این آزمایش، جایگزین آزمایش جعبه سفید نمی‌باشد؛ بلکه مکمل آن است و خطاهایی متفاوت با آن را تست می‌کند. شما نرم‌افزاری را که به آن نیاز داشتید، تهیه می‌کنید و روی سیستم خود نصب می‌کنید. در بیشتر موارد بعد از نصب برنامه، فقط یک نسخه اجرایی آن را در سیستم خود خواهید داشت و هیچ دسترسی به سوریس کد و منابع دیگر برنامه ندارید. سیستم نرم‌افزاری موجود، برای شما مانند یک جعبه سیاه است که

نمی‌توانید دورن آن‌را مشاهده کنید و به آن دسترسی داشته باشید. استراتژی جعبه سیاه، دقیقاً از این دیدگاه برنامه را مورد آزمایش قرار می‌دهد؛ یعنی با این پیش فرض که شما هیچ اطلاعاتی از کد و طراحی داخلی برنامه ندارید. حال که هیچ اطلاعاتی از کد و طراحی برنامه در اختیار ما نیست، چگونه می‌توان به صحت کارکرد برنامه پی بُرد؟ جواب، خیلی ساده است؛ با تمرکز بر ورودی‌ها و خروجی‌ها، برای اینکار آزمایش‌کننده نرم‌افزار به مستندات نرم‌افزار مراجعه می‌کند تا مشخص کند که سیستم در مقابل یک عمل خاص، چه پاسخی باید بدهد. سپس، داده‌ها را برای هرکدام از عملیات انتخاب می‌کند و رفتار سیستم را در مقابل آن داده‌ها با رفتار واقعی سیستم که در مستندات وجود دارد، مقایسه و بررسی می‌کند.

در یک استراتژی آزمایش جعبه سیاه، عموماً موارد زیر را مورد بررسی و آزمایش قرار می‌دهیم:

ـ بررسی مقادیر مرزی برای متغیرها: به یک متغیر، مقداری کمتر از حداقل مقداری که می‌تواند قبول کند یا بیشتر از حداکثر مقداری که می‌تواند قبول کند، می‌دهیم و سیستم را در این شرایط تست می‌کنیم.

ـ بررسی خروجی‌های سیستم: مجموعه‌ای از ورودی‌های صحیح با خروجی‌های مربوط به آن را تهیه می‌کنیم و سپس، ورودی‌ها را به سیستم وارد می‌کنیم و خروجی‌هایی را که توسط سیستم داده می‌شود، با خروجی‌های واقعی مقایسه می‌کنیم و رفتار سیستم را در برابر پردازش ورودها و پرس و جوهای بزرگ و سنگین بررسی نماییم.

استراتژی جعبه سفید

حال تصور کنید که خود یک توسعه‌دهنده نرم‌افزار هستید. شما می‌توانید به سورس، طراحی و منابع دیگر نرم‌افزار دسترسی داشته باشید. در این حالت، سیستم را می‌توان به یک جعبه شیشه‌ای (جعبه سفید) تشبیه کرد که می‌توانید به‌راحتی محتویات داخل و شیوه عملکرد آن را مشاهده کنید. آزمایش جعبه سفید نیز دقیقاً از دیدگاه توسعه‌دهنده، نرم‌افزار را مورد آزمایش قرار می‌دهد؛ یعنی با این فرض که شما به منطق داخلی و ساختار کد برنامه دسترسی و احاطه دارید و می‌دانید که سیستم چگونه پیاده‌سازی شده است. با دانستن این موارد، می‌توانید مشخص کنید که آیا اعمال داخلی بر طبق مشخصه‌ها نجام می‌شود و یا نه.

در یک استراتژی آزمایش جعبه سفید، عموماً موارد زیر را مورد توجه و بررسی قرار می‌دهیم:

ـ بررسی سطربه‌سطر کد (Code coverage): در این حالت، باید سیستم را به گونه‌ای اجرا و بررسی کنیم که مطمئن شویم سطربه‌سطر کد برنامه حداقل یکبار اجرا شده است.

ـ بررسی همه انشعاب‌ها در کد برنامه (branch): در کد برنامه باید تمام عبارت‌های شرطی (if elseها و Switch caseها) را تک‌به‌تک مورد بررسی قرار داد؛ به این صورت که در یک عبارت if else، هم قسمت if و هم قسمت else هرکدام به صورت مجزا یکبار اجرا شوند.

ـ بررسی همه حلقه‌ها در برنامه: حلقه‌ها، در نرم‌افزار نقش اساسی دارند؛ چون می‌توانند با اشتباه جزئی، مقدار بسیاری از منابع را مصرف کنند؛ برای مثال، شرط خروج از حلقه، به‌اشتباه هیچ وقت True نشود؛ برای نمونه، حلقه‌ها را با ورودی بزرگ‌تر از شرط خروج حلقه چک کنید؛ یعنی حلقه اصلاً اجر نشود. تستی طراحی کنید که حلقه دقیقاً یکبار اجرا شود و یا حلقه در یک بازه خاص اجرا گردد.

ـ مدیریت خطای مطلوب: بررسی اینکه اگر به یک روش یک ورودی نامعتبر وارد شود، شیوه آگاه‌سازی و نمایش مطلوب خطا برای کاربر چگونه باشد.

ـ بررسی امنیت: سیستم را در برابر دسترسی‌های غیرمجاز، هک، کرک و هر چیز دیگر که می‌تواند به آن آسیب برساند، مورد بررسی قرار می‌دهد. در اینجا ما باید مکان‌هایی از کد را که داده‌ها را اعتبارسنجی و مدیریت می‌کنند و یا دسترسی به منابع و عملیات مهم و حیاتی را انجام می‌دهند، بررسی نماییم.

پی‌نوشت‌ها:

۱. جوران، **خروج از بحران**.
۲. پارسا، سعید، **مهندسی نرم‌افزار**.
۳. پایگاه اینترنتی ویکی پدیا، ذیل عنوان «Software Review».
۴. ـــــ . ذیل عنوان «تست نرم‌افزار».

منابع:

۱. حسنی، فرنود. «چالش‌های مدیریتی تولید نرم‌افزار درایران»، نشریه کتاب ماه کلیات، مهر و آبان ۸۴، ش ۹۴ و ۹۵.
۲. کاظمی، علی. «تأثیر رهبری معنوی اسلامی بر عملکرد سازمانی مورد مطالعه شرکت گاز استان لرستان»، نشریه مدیریت اسلامی، بهار و تابستان ۹۲، ش ۵.
۳. پارسا، سعید. **مهندسی نرم‌افزار**. انتشارات علم و صنعت، اسفند ۱۳۹۱ش.
۴. پایگاه اینترنتی ویکی‌پدیا. (دسترسی در: بهار ۱۳۹۵). نشانی:
■www.wikipedia.org