تست نرم افزار

Azimi.marjan@gmail.com

مفاهیم پایه در تست نرم‌افزار

- Failure: ناتوانی یک سیستم و یا یک جز سیستم در اجرای کاری که مطابق با خصوصیات آن باشد و یا به عنوانی عملکرد نرم‌افزار مطابق توضیحات داده شده و مورد انتظار نباشد.

- Fault: شرایطی که باعث شود نرم‌افزار در اجرای دستور مورد نیازش دچار شکست شود. به عبارتی نقص استاتیک در نرم افزار را Fault گویند.

- Mistake(error): عبارت است از کارهایی که منجر به نتیجه اشتباه یا نادرست می‌شوند.

مفاهیم پایه در تست نرم‌افزار

- Defect: ناشی از ابزار است و معمولا به مشکلاتی اشاره می‌کند که در محصولات نرم‌افزاری است و ضرورتا به این معنی نیست که در کد bug وجود دارد بلکه می‌تواند تابعی باشد که اجرا نشده است اما در ابزار نرم‌افزار تعریف شده است. تست نقص، موفقیت آن در نمایان سازی خطاهایی است که موجب کارکرد نادرست سیستم می‌شود.

مفاهیم پایه در تست نرم‌افزار

- Bug: اشکال نوعی خطا یا اشتباه در اجرای نرم‌افزار است که موجب نتایج اشتباه یا اجرا نشدن نرم افزار می‌شود. علت این اشکالات می‌تواند اشتباه در هنگام برنامه نویسی و خطای سهوی و عمدی باشد. شرکت‌های سازنده نرم‌افزارها برای حل این مشکل قبل از ارایه نسخه نهایی، نسخه‌های تحت عنوان آلفا یا بتا انتشار می‌دهند تا افرادی آن‌ها را بررسی کنند و این باگ‌ها را گزارش دهند.
- وجود باگ ممکن است سهوا یا عمدا باشد.
- گاهی شرکت‌های سازنده نرم‌افزار به عمد یک یا چند اشکال در نسخه‌های قابل استفاده قبل از انتشار که رایگان پخش می‌کنند، می‌گذارند تا از کپی غیر مجاز آن‌ها جلوگیری نمایند.

مفاهیم پایه در تست نرم‌افزار

- Risk: یک اتفاق غیر قطعی است که احتمال وقوع و آسیب رسانی دارد. تعریف مدیریت ریسک یک مساله مهم و نگران کننده در هر پروژه نرم‌افزاری است و انواع مختلفی دارد:
- ریسک در زمانبند: زمانبندی پروژه زمانی دچار لغزش می‌شود که وظایف پروژه به خوبی مشخص نشده باشد. ریسک در زمانبند اساسا در پروژه و در نهایت بر اقتصاد شرکت اثر می‌گذارد و منجر به شکست پروژه می‌شود.
- ریسک بودجه: عبارت است از تخمین اشتباه بودجه، افزایش هزینه و توسعه پروژه
- ریسک عملیاتی: این ریسک منجر به پیاده سازی پروسه نامناسب، شکست سیستم با ریسک‌های خارجی میشود.
- ریسک تکنیکی: این ریسک منجر به شکست کارایی و عملکرد می‌شود.

اهداف تست

- تست یک فرآیند اجرای برنامه به منظور یافتن خطاهاست.
- یک تست خوب آن است که با احتمال زیادی منجر به یافتن خطایی شود که تا کنون کشف نشده است.
- کشف خطاها در عملکرد، منطق یا پیاده سازی هر نمایشی از نرم‌افزار است.
- تصدیق این که نرم افزار مورد بازبینی قادر به نیازهای عملکردی است.
- میزان قابلیت اعتماد و درجه کیفیت در نرم افزار مشخص می‌شود.

اهداف تست

- تست معمولا برای رسیدن به اهداف زیر مورد استفاده قرار می‌گیرد:
- برای بهبود کیفیت
- برای وارسی و اعتبار سنجی
- برای تخمین قابلیت اعتماد

اصول آزمایش نرم‌افزار

- همه تست‌ها باید تا حد خواسته‌های مشتری قابل ردیابی باشند.
- تست باید مدت‌ها قبل از شروع تست، طرح ریزی شود. طرح ریزی تست می‌تواند به محض کامل شدن مدل خواسته‌ها آغاز شود.
- اصل پارتو در تست نرم‌افزار صدق می‌کند.
- باید در ابعاد کوچک آغاز شود و به ابعاد بزرگتر گسترش یابد.
- تست کامل امکان پذیر نیست.
- برای این که تست بیشترین بازدهی را داشته باشد، باید توسط یک شخص ثالث بی طرف انجام شود. منظور از بیشترین بازدهی آن است که خطاها را با احتمال بیشتری بیابد.

آزمون پذیری نرم‌افزارها

- آزمون پذیری نرم افزار میزانی از سهولت آزمودن یک برنامه کامپیوتری است. چون آزمودن دشوار است، بهتر است بدانیم چه چیزی آن را به جریان می‌اندازد.

ویژگی‌های نرم‌افزار آزمون پذیر

- قابلیت کار: هر چه بهتر کار کند، آزمون آن کارامدتر است.

- سادگی: هر چه مورد آزمون کوچکتر باشد، سریعتر می‌توان آن را آزمود.
  - سادگی عملیاتی: مثلا مجموعه ویژگی‌ها، حداقل مجموعه لازم برای برآوردن خواسته‌هاست.
  - سادگی ساختاری: مثلا معماری به صورت پیمانه‌هایی در آمده تا انتشار خطاها را به حداقل برساند.
  - سادگی کد: مثلا یک استاندارد کد نویسی رعایت می‌شود که وارسی و نگهداری آن آسان باشد.

ویژگی‌های نرم‌افزار آزمون پذیر

- پایداری: هر چه تعداد تغییرات کمتر باشد، آزمون کمتر با مانع مواجه می‌شود.
- درک پذیری: هر چه اطلاعات بیشتری داشته باشیم، آزمون هوشمندانه تر اجرا می‌شود.

صفات یک آزمون خوب

- آزمون خوب با احتمال زیادی خطاها را می‌یابد.
- آزمون خوب دارای زواید نیست.
- یک آزمون خوب باید بهترین باشد.
- آزمون خوب نه باید بیش از حد ساده و نه بیش از حد پیچیده باشد.

چرخه حیات آزمایش نرم‌افزار

- چرخه حیات تست نرم‌افزار بیان می‌کند که کدام فعالیت‌های تست و در چه زمانی باید انجام شوند. ولو این که فعالیت‌های تست بین سازمان‌های مختلف متفاوت است ولی یک چرخه حیات برای تست وجود دارد. چرخه حیات تست نرم افزار از شش فاز کلی تشکیل شده است.

طرح‌ریزی تست

- این فازی است که مدیر پروژه باید تصمیم بگیرد که چه مواردی باید تست شوند و اینکه آیا بودجه لازم و کافی در دسترس هست یا خیر. طبیعتا طرح ریزی کافی در این مرحله به میزان زیادی ریسک نرم افزارهای بدون کیفیت را کاهش می‌دهد. این طرح ریزی یک فرآیند در حال پیشرفت بدون هیچ انتهایی است.

تحلیل تست

- زمانی که طرح ریزی تست انجام شد و روی آن تصمیم گیری شد، مرحله بعدی کمی کاوش پیرامون پروژه و تصمیم راجع به نوع تستی است که باید انجام شود.
- چه ملاقات‌های مناسب و منظم بین تیم‌های تست، مدیران پروژه و تیم‌های برنامه نویسی و تحلیل گر شرکت باید صورت گیرد تا پیشرفت مواردی که تا حد زیادی پیشرفت پروژه و تکمیل طرح تست ایجاد شده در فاز طرح ریزی را تحت تاثیر قرار می‌دهند، کمک بیشتری به بهبود استراتژی صحیح تست کردن کند.

طراحی تست

- طرح‌ها و موارد تست که در فاز تحلیل توسعه داده می‌شوند باید بازبینی شوند و ماتریس ارزیابی عملکرد نیز بازبینی و نهایی می‌شود. در این مرحله معیار ارزیابی تست نیز ایجاد می‌شود و اگر به اتوماسیون فکر می‌کنید باید موارد تستی را که می‌خواهید اتومات کنید را انتخاب کنید و شروع به نوشتن اسکریپت برای آن‌ها کنید. داده‌های تست نیز باید جمع‌آوری شوند. استانداردهای تست واحد و معیارهای پذیرش و شکست در اینجا تعریف شده‌اند. اگر نیاز باشد برنامه تست کردن دوباره بازبینی و نهایی می‌شود و محیط تست نیز آماده می‌شود.

ساخت و ارزیابی

- در این فاز باید چرخه‌های تست تکمیل شوند تا موارد تست بدون هیچ خطا یا شرایط از قبل تعیین شده‌ای انجام شوند.
  - موارد تست را اجرا کنید.
  - باگها را گزارش دهید.
  - اگر نیاز است موارد تست را بازبینی کنید.
  - اگر نیاز است موارد تست جدید را اضافه کنید.
  - تصحیح باگها

تست کردن نهایی و پیاده سازی

- در این مرحله باید موارد تست استرس و کارهای باقیمانده را اجرا کنیم و مستندات تست باید تکمیل و به هنگام شوند و معیارهای متفاوتی برای تست ارایه و تکمیل شوند. تست پذیرش و بار و ترمیم اجرا می‌شوند و برنامه کاربردی نیز باید تحت شرایط کاری بازبینی شود.

تست پس از پیاده‌سازی

- در این فاز، فرآیند تست ارزیابی می‌شود و درس‌های گرفته شده از فرآیند تست، ثبت می‌شوند. خطوط مرزی برای پیشگیری از سایر مسایل مشابه در پروژه‌های آینده مشخص و شناسایی و ثبت می‌شوند. طرح‌هایی برای بهبود فرآیند عرضه می‌شوند. ثبت خطاهای جدید و ویژگی‌های اضافه شده یک فرآیند ادامه دار است.

تست عملکردی در برابر تست غیر عملکردی

- تست عملکردی: به فعالیت‌هایی که یک عمل خاص و یا تابعی از کد را وارسی می‌کند، اشاره دارد. تست عملکردی تمایل به پاسخ به این سوال دارد که آیا کاربر می‌تواند این کار را انجام دهد و یا آیا این کار از ویژگی‌های خاصی برخوردار است.

- تست غیر عملکردی: به جنبه‌های نرم افزار که ممکن است به یک تابع خاص و یا عمل کاربر مرتبط نباشد، مانند مقیاس پذیری و یا کارایی دیگر، رفتار تحت محدودیت‌های خاص و یا امنیت اشاره دارد.

تست پویا در برابر تست ایستا

- در تست ایستا نرم افزار اجرا نمی‌شود ولی در روش تست پویا نرم افزار مورد تست با مقادیر واقعی ورودی اجرا می‌شود.

مزایای تست ایستا

- از زمانی که تست در چرخه حیات شروع می‌شود به همان سرعت می‌توان خروجی و بازخوردهای آن را در مسایل کیفیت مراحل بعد لحاظ کرد.
- با تشخیص هنگام ورود خطا در یک مرحله، می‌توان به طور چشمگیری از هزینه‌های اضافی افزایش کیفیت کاسته و توسعه نرم افزاری مقرون به صرفه ای داشته باشیم.
- تست ایستا یک روش افزایش کیفیت و بهره وری هوشمند است.
- در نتیجه روش تست ایستا روش بسیار مناسب جهت رشد کیفیت و کار محصول نرم افزاری می‌باشد. بازخورد تست ایستا در مرحله توسعه، اجازه بهبود فرآیند را می‌دهد و باعث ممانعت از تکرار این خطاها در آینده می‌شود.
- انواع مرور از غیر رسمی تا رسمی می‌تواند بسیار متنوع باشد. روش ایستا در میان روش‌های آزمون یکی از روش‌های مستند و رسمی جهت بازبینی است.
- روش‌های تست ایستا انحصارا برای قطعه برنامه‌هایی به کار می‌رود که به حد کافی کوچک هستند و می‌توانند امتحان شوند.

تست پویا

- با استفاده از روش تست پویا، نرم افزار با یک سری ورودی اجرا می‌شود و خروجی‌های آن با مقادیر قابل انتظار مقایسه می‌شود.
- تست پویا هم در محصول و هم در تست واحد به کار برده می‌شود. دو نوع مختلف از تست‌های پویا را می‌توان به کار گرفت: منطقی و تابعی.
- تست تابعی نشان می‌دهد که برنامه تحت شرایط عملیاتی نمونه خوب عمل می‌کند و داده تست از روی نیازمندی‌های تابعی مخصوص بدون توجه به ساختار منحصر به فرد برنامه گرفته می‌شود. برنامه باید مقدار خروجی صوری صحیح را برای مقدار ورودی صوری مشخص محاسبه کند. همان طور که در مشخصه نشان داده می‌شود. تست تابعی به بررسی رفتار خارجی برنامه می‌پردازد به عبارت دیگر از دیدگاه جعبه سیاه.
- تست منطقی در رابطه با چگونگی اجرای اعمال محاسباتی برنامه است. تست منطقی همچنین می‌تواند برای بررسی کردن داده و کنترل واسطه‌های بین واحدهای برنامه به کار رود.

انواع تست

- تست عملکرد
- تست استرس
- تست بار
- تست رگرسیون
- تست قابلیت استفاده
- تست امنیت
- تست پوشش

تست عملکرد

- در این نوع تست، نرم افزار تست می‌شود تا از لحاظ درستی عملکرد بررسی شود. تستها نوشته می‌شوند تا ببینند که آیا نرم افزار همان گونه که انتظار می‌رود عمل می‌کند. اگر چه تست عملکرد معمولا در انتهای فرآیند توسعه انجام می‌شود ولی می‌تواند و در واقع باید سریع‌تر آغاز شود.

تست استرس

- برنامه در مقابل بار سنگین مثل مقادیر عددی پیچیده، مقادیر زیاد ورودی و مقادیر زیاد پرس‌وجو تست می‌شود تا میزان باری که برنامه می‌تواند آن را تحمل کند بررسی شود. هدف طراحی محیطی است که مخرب‌تر از محیطی است که برنامه در دنیای واقعی و در شرایط نرمال با آن روبه‌رو می‌شود. این مساله سخت‌ترین و پیچیده‌ترین مقوله از تست است که باید انجام شود و احتیاج به تلاش توام همه تیم‌های برنامه‌نویسی دارد.

تست استرس

- شرایط رقابت و رخنه‌های حافظه در تست استرس پیدا می‌شوند. شرایط رقابت تعارضی بین حداقل دو ایستگاه تست است و هر تست زمانی که به تنهایی کار کند به درستی کار می‌کند، اما زمانی که دو تست به صورت موازی کار کنند. یکی یا هر دو تست شکست می‌خورند و این معمولا به خاطر یک قفل است که به درستی مدیریت نشده است.
- یک رخنه حافظه زمانی رخ می‌دهد که یک تست، حافظه تخصیص یافته را رها می‌کند و به درستی حافظه را بر نمی‌گرداند. به نظر تست درست کار می‌کند اما بعد از مدتی اجرای تست حافظه در دسترس کاهش پیدا می‌کند و سیستم شکست می‌خورد.

تست بار

- برنامه در مقابل بار زیاد یا ورودی ها تست می‌شود مثل تست وب سایت‌ها برای یافتن نقاطی که در آن نقاط وب سایت یا برنامه شکست می‌خورد و یا نقاطی که در آن ها کارایی وب سایت کاهش پیدا می‌کند.
- تست بار در سطح بار از پیش تعیین شده‌ای انجام می‌شود معمولا بالاترین باری که سیستم میتواند بپذیرد در حالیکه هنوز به درستی کار می‌کند. تست بار نمی‌خواهد سیستم را با فشار آوردن از پا درآورد. اما می‌خواهد سیستم را به طور پیوسته زیر بار نگه دارد. در تست بار باید مجموعه داده‌های زیادی برای تست در دسترس داشته باشیم.

تست رگرسیون

- بعد از تغییر نرم افزار، چه برای تغییر در عملکرد یا برای تصحیح یک خطا، یک تست رگرسیون تمام تست‌هایی که قبلا نرم افزار آن‌ها را با موفقیت انجام داده است را دوباره اجرا می‌کند تا اطمینان حاصل کند که نرم افزار تصادفا در عملکردهای قبلی دچار خطا نشده است. تست رگرسیون می‌تواند در هیچ یا همه سطوح قبلی صورت بگیرد. این تست‌ها اکثرا به صورت اتومات انجام می‌شوند.
- تست رگرسیون می‌خواهد دو ریسک را کاهش دهد:
  - تغییری که می‌بایست یک خطا را از بین می‌برد، شکست می‌خورد.
  - بعضی تغییرات اثر جانبی دارند، اگر تصحیح نکنید خطای قبلی باقی می‌ماند و اگر تصحیح کنید خطای جدید ایجاد می‌شود.

انواع تست رگرسیون

- رگرسیون خطا: یک باگ خاص که قبلا تعمیر شده، تست می‌شود.
- تست بازگشت تعمیرات قبلی: چندین خطای قدیمی را که تعمیر شده بودند دوباره تست می‌کنیم تا بررسی کنیم که آن ها دوباره ایجاد نشده باشند.
- بازگشت عملکرد معمول: محصول را تست می‌کنیم تا ببینیم تغییرات اخیر، کدهایی را که اجرا می‌شدند را ناپایدار نکرده باشد.
- تست پیکربندی: برنامه با یک ابزار جدید یا یک سیستم عامل جدید اجرا می‌شود تا مشخص شود آیا انتقال موفقیت آمیز بوده یا خیر. فقط کامپوننت‌های خارجی که سیستم باید با آن‌ها تعامل کند تغییر کرده‌اند.

تست قابلیت استفاده

- این نوع تست تنها در مواردی انجام می‌شود که رابط کاربر برای برنامه خیلی مهم باشد و باید برای هر کاربر، خاص باشد. تست قابلیت استفاده، فرآیند کارکردن مستقیم یا غیر مستقیم با کاربران نهایی است و برای شناخت این است که چگونه هر کاربر با یک بسته نرم افزاری تعامل می‌کند. این فرآیند نقاط قوت و ضعف برنامه را برای کاربران کشف می‌کند.
- هدف این تست، شناسایی مشکلات مربوط به طرز استفاده از سیستم از دید کاربران می‌باشد.

تست امنیت

- آزمون امنیت ویژگی‌هایی از سیستم که به در دسترس بودن، یکپارچگی و محرمانه بودن داده‌ها و خدمات سیستم مرتبط هستند را ارزیابی می‌کند.
- هدف آزمون امنیت بررسی کارایی مکانیزم‌های دفاعی سیستم وب در مقابل دسترسی‌های نامطلوب کاربران بدون مجوز و حفظ منابع سیستم در مقابل کاربران ناشایست و همچنین دادن دسترسی به کاربرانی که مجوز دارند، می‌باشد.
- آسیب پذیری‌های سیستم که بر روی امنیت سیستم تاثیر می‌گذارد ممکن است منشا در کد برنامه یا در کامپوننت‌های سخت افزاری، نرم افزاری یا میان افزاری سیستم داشته باشد.

مفاهیم اساسی امنیت

- محرمانه بودن: یک معیار امنیت که از افشای اطلاعات به افرادی که مطمن نیستند محافظت می‌کند.
- جامعیت: معیاری که به گیرنده اجازه می‌دهد که مشخص کند اطلاعاتی که دریافت کرده است در طول انتقال یا توسط صادر کننده اطلاعات، تغییر داده نشده است.
- تصدیق هویت: معیاری است که برای تصدیق مجاز بودن انتقال، پیام یا صادر کننده طراحی شده است. به گیرنده اجازه می‌دهد تا اطمینان حاصل کند که اطلاعات از یک منبع شناخته شده‌ی مشخص دریافت شده است.
- مجوز دادن: پردازشی که طی آن مشخص می‌شود یک درخواست کننده اجازه دارد که سرویسی را دریافت کند یا عملی را انجام دهد. Access Control مثالی از مجوز دادن می‌باشد.

مفاهیم اساسی امنیت

- در دسترس بودن: اطمینان از این که سرویس‌های ارتباطی در زمانی که نیاز است، اطلاعات را برای استفاده آماده می‌کنند. اطلاعات باید در دسترس نگهداشته شوند تا با استفاده از آن‌ها به افراد مجوز داده شود.
- عدم انکار: معیاری است که از انکار عملی که اتفاق افتاده یا ارتباطی که برقرار شده و غیره، جلوگیری می‌کند. در مورد ارتباطات، عدم انکار شامل تبادل اطلاعات تصدیق هویت همراه با شکلی از برچسب زمان که قابل اثبات کردن است، می‌شود.

مفاهیم اساسی امنیت

- تاثیر نفوذهای امنیتی می‌توانند گسترده باشد و باعث موارد زیر شوند:
  - از دست دادن اطلاعات
  - خرابی اطلاعات
  - اطلاعات نادرست
  - تجاوز به حریم خصوصی
  - رد خدمات

تست پوشش

- تست پوشش به دو دسته کلی پوشش عبارات و پوشش انشعابات می‌توان تقسیم نمود. در تست پوشش عبارات، کد طوری اجرا می‌شود که هر عبارتی از برنامه حداقل یکبار اجرا شود و این باعث می‌شود بفهمیم همه جملات بدون اثر جانبی اجرا می‌شوند.
- از آنجاییکه هیچ برنامه نرم‌افزاری نمی‌تواند در مد پیوسته‌ای از کد اجرا شود، در بعضی از نقاط نیاز است که برای یک عملکرد خاص به نقطه‌ای خارج از کد انشعاب کنیم. تست پوشاندن انشعابات کمک می‌کند که همه انشعابات در کد را ارزیابی کنیم و اطمینان حاصل کنیم که هیچ انشعابی در کد منجر به رفتار غیر نرمال در برنامه کاربردی نمی‌شود.