پنجشنبه 03 مرداد 1392 - 12:10

روان‌شناسی نرم‌افزار

آزمایش نرم‌افزار یک کار فنّی است امّا مستلزم ملاحظات مهم اقتصادی و روان‌شناسی نیز می‌باشد. برنامه‌نویسان در بهترین حالت مایلند که تمام ترکیبات ممکن یک برنامه را مورد آزمایش قرار دهند. امّا در اغلب موارد این امر امکان‌پذیر نیست. حتی یک برنامه ظاهراً ساده هم می‌تواند صدها یا شاید هزاران ترکیب ممکن ورودی و خروجی داشته باشد. ایجاد آزمایه ۱ برای تمام این احتمالات، کاری غیر عملی است. آزمایش کامل یک نرم‌افزار کاربردی پیچیده، به زمان و نیروی انسانی بسیار زیادی نیاز دارد که این امر معمولاً از لحاظ اقتصادی به صرفه نیست.

روان شناسی نرم افزار

به علاوه آزمونگر ۲ نرم افزار باید نگرش و دید مناسبی داشته باشد تا بتواند آزمایش یک نرم افزار کاربردی را با موفقیت انجام دهد. در بعضی موارد، نگرش آزمونگر ممکن است حتی از خود فرایند واقعی آزمایش نیز مهمتر باشد. به این جهت، باید پیش از پرداختن به مسائل فنی آزمایش نرم افزار، جنبه های روان شناسانه آن را مورد بررسی قرار داد.

یکی از دلایل اصلی آزمایش ضعیف برنامه ها این واقعیت است که اغلب برنامه نویسان با تعریف اشتباهی از این عبارت کار خود را آغاز می کنند. آن ها پیش خود فکر می کنند: «آزمایش عبارت است از فرایند نشان دادن عدم وجود خطا». و «یا هدف از آزمایش این است که نشان داده شود برنامه، عملکرد مورد نظر را به درستی انجام می دهد». و یا آزمایش عبارت است از فرایند اطمینان یافتن از این که برنامه، کاری که باید انجام دهد را انجام می دهد».

این تعاریف در واقع وارونه هستند!

هنگامی که شما برنامه ای را آزمایش می کنید، می خواهید که ارزشی را به آن بیافزائید. منظور از افزودن ارزش از طریق آزمایش، بالا بردن کیفیت یا اطمینان پذیری ۳ برنامه است. و بالا بردن اطمینان پذیری برنامه هم به معنی یافتن و برطرف کردن خطاهاست.

بدین خاطر، نباید برنامه را آزمایش کرد تا نشان داد درست کار می کند، بلکه باید با این فرض که برنامه حاوی خطاست شروع کرد (فرضی که در مورد اغلب برنامه ها صادق است) و آنگاه برنامه را برای یافتن خطاهای هرچه بیشتر، مورد آزمایش قرار داد.

بنابراین، تعریف دقیقتری از آزمایش نرم افزار چنین است:

آزمایش عبارت است از فرایند اجرای برنامه با هدف یافتن خطاها. هر چند ممکن است این بازی با کلمات به نظر آید امّا وقعاً وجه تمایز مهمی وجود دارد. درک تعریف درست آزمایش نرم افزار می تواند تفاوت چشمگیری در موفقیت تلاش های شما در این زمینه به وجود آورد.

انسان تمایل شدید به هدف گرایی دارد و قرار دادن هدف مناسب، دارای تأثیر روانی مهمی است. اگر هدف ما این باشد که نشان دهیم برنامه خطا ندارد، به طور ناخودآگاه به سمت این هدف هدایت می شویم. یعنی داده های آزمایشی ای را انتخاب می کنیم که احتمال کمتری دارد تا باعث عملکرد اشتباه برنامه گردند. امّا از سوی دیگر، چنانچه هدف ما نشان دادن این باشد که برنامه دارای خطاست، آنگاه داده های آزمایشی ما نیز احتمال بیشتری دارد که باعث کشف خطاها گردند. این رویکرد دوم نسبت به اولی، ارزش بیشتری را به برنامه می افزاید.

این تعریف از آزمایش نرم افزار دارای تأثیرات و معانی ضمنی بسیاری است. برای مثال، بیانگر این است که آزمایش، فرایند مخرب و حتی آزاردهنده ای می باشد و به همین دلیل است که برای اغلب مردم دشوار است. دیدگاه و نگرش اغلب ما نسبت به زندگی، سازنده است نه مخرّب. در حالی که با این تعریف، نتیجه یک فرایند آزمایش موفق، کاملاً برخلاف تمایل قلبی ما خواهد بود. این تعریف همچنین برروی چگونگی طراحی آزمایه ها و داده های آزمایشی و نیز این که چه کسی باید یک برنامه را آزمایش کند و چه کسی نباید، تأثیر گذار است.

راه دیگری که برای تحکیم تعریف مناسبی از آزمایش نرم افزار وجود دارد، تحلیل استفاده از واژه های «موفق» و «ناموفق»، به ویژه استفاده از آن ها توسط مدیران پروژه در رده بندی نتایج آزمایه ها، می باشد. اغلب مدیران پروژه، آزمایه ای که خطایی نیافته است را «موفق» و برعکس آن را «ناموفق» می خوانند.

یکبار دیگر باید بگوئیم که این تعبیر کاملاً وارونه است! واژه «ناموفق» نشانگر یک چیز نامطلوب یا ناراحت کننده است. در حالی که بنا به تعریفی که از آزمایش نرم افزار ارائه شد، یک آزمایش خوب سازماندهی شده و خوب اجرا شده هنگامی موفق است که به یافتن خطاهایی که قابل اصلاح باشند بیانجامد. چنانچه همین آزمایش نهایتاً به اثبات رساند که دیگر خطایی برای یافتن وجود ندارد آنگاه باید آن را آزمایش موفقی نامید. امّا آزمایش ناموفق، آزمایشی است که نرم افزار را به طور مناسب مورد بررسی قرار ندهد. در اغلب موارد، آزمایشی که خطایی نیابد را باید ناموفق خواند زیرا مفهوم برنامه بدون خطا اساساً مفهومی غیر واقعی است.

اگر این تعبیر را به طراحی آزمایه ها تعمیم دهیم باید بگوئیم آزمایه ای که منجر به یافتن یک خطای جدید شده است را به هیچ وجه نباید ناموفق نامید، بلکه برعکس باید ارزش زیادی برای آن قائل شد. یک آزمایه ناموفق آزمایه ای است که باعث شود برنامه، بدون یافتن هیچ خطایی، جواب درست تولید کند.

در مقام مقایسه، فردی را در نظر بگیرید که به پزشک مراجعه می کند و به او می گوید که حالش خوب نیست و نوعی حس کسالت عمومی دارد. اگر پزشک چند نوع آزمایش برای آن فرد بنویسد و نتایج آزمایش ها نشان دهنده مشکلی نباشند، ما آن آزمایش ها را «موفقیت آمیز» نمی نامیم زیرا او متحمل مقداری هزینه شده و کسالتش نیز همچنان پابرجاست. امّا اگر نتیجه آزمایش نشان دهد که فرد زخم معده دارد، آن آزمایش موفقیت آمیز بوده است زیرا اکنون پزشک می تواند به درمان مناسب بپردازد. به همین خاطر است که پزشکان معمولاً واژه های «موفق» و «ناموفق» (یا مثبت و منفی) را در این مورد درست به کار می برند. مقایسه ای که در اینجا وجود دارد این است که ما نیز برنامه ای که می خواهیم شروع به آزمایشش کنیم را باید به مثابه یک فرد بیمار در نظر بگیریم.

مشکل دومی که با تعریف آزمایش به عنوان «فرایند نشان دادن عدم وجود خطا» پیش می آید این است که دستیابی به چنین هدفی، تقریباً برای تمام برنامه ها حتی برنامه های کوچک و ساده، غیرممکن است.

باز هم مطالعات روان شناسی به ما می گوید که انسان ها هنگامی که به انجام دادن کاری گمارده می شوند که می دانند غیرممکن و دست نیافتنی است، عملکرد ضعیفی از خود بروز می دهند. برای مثال، اگر از فردی خواسته شود که یک جدول سخت را ظرف ۱۵ دقیقه حل کند، معمولاً بعد از گذشت ۱۰ دقیقه یا هیچ پیشرفتی در حل جدول دیده نمی شود و یا پیشرفت بسیار کمی دیده می شود زیرا اغلب افراد در مقابل کاری که غیرممکن به نظر می رسد تسلیم می شوند و دست از تلاش بر می دارند. امّا اگر از آن فرد خواسته شود

که همان جدول را در چهار ساعت حل کند، بعد از گذشت ۱۰ دقیقه پیشرفت بیشتری نسبت به حالت اوّل دیده خواهد شد. بنابراین درنظر گرفتن آزمایش نرم افزار به عنوان فرایند کشف خطاهای برنامه، آن را به صورت کاری امکانپذیر جلوه می سازد و این مشکل روانی را نیز از میان برمی دارد.

سومین مشکلی که با تعریف آزمایش نرم افزار به صورت «فرایند نشان دادن این که برنامه کاری که باید انجام دهد را انجام می دهد» وجود دارد این است که برنامه هایی که این چنین هستند، یعنی کاری که باید انجام دهند را انجام می دهند، نیز هنوز ممکن است دارای خطا باشند. به عبارت دیگر، اگر برنامه ای کاری که باید انجام دهد را انجام ندهد، واضح است که خطایی در آن وجود دارد امّا اگر برنامه ای کاری که نباید انجام دهد را انجام دهد نیز دارای خطاست. اگر ما به آزمایش برنامه به صورت فرایند کشف خطاها فکر کنیم، احتمال بیشتری دارد که این نوع خطاها را بیابیم تا آن که آن را به صورت فرایند نشان دادن این که برنامه کاری که باید انجام دهد را انجام می دهد، در نظر بگیریم.

خلاصه آن که آزمایش برنامه مناسب تر است به صورت فرایند تلاش برای کشف خطاهای برنامه (که فرض می شود وجود دارند) در نظر گرفته شوند. یک آزمایه موفق، آزمایه ای است که در این جهت باعث پیشرفت گردد، یعنی باعث عملکرد اشتباه برنامه شود. البته نهایتاً ما می خواهیم که از طریق آزمایش نرم افزار به درجه قابل قبولی از اطمینان به این که برنامه کاری که باید انجام دهد را انجام می دهد و کاری که نباید انجام دهد را انجام نمی دهد، برسیم امّا این هدف تنها از طریق جستجوی پیگیرانه خطاها دست یافتنی است.

فرض کنید کسی نزد شما بیاید و ادعا کند که «برنامه اش کامل و بی نقص است». بهترین راه برای اطمینان یافتن از صحت ادعای او این است که سعی کنیم ادعایش را رد کنیم، یعنی سعی کنیم خطایی در آن بیابیم، نه این که صرفاً به خاطر این که برنامه اش با چند داده آزمایشی درست کار می کند ادعایش را بپذیریم.

اصول آزمایش نرم افزار

با یادآوری مجدّد این نکته که مهمترین ملاحظه ای که در آزمایش نرم افزار باید در نظر گرفته شود جنبه های روان شناسی آن است، می توان مجموعه ای از اصول یا رهنمودهای مهم و ضروری برای آزمایش را به شرح زیر ارائه کرد. هرچند اغلب آن ها ممکن است بدیهی به نظر آیند امّا غالباً نادیده گرفته می شوند.

اصل ۱: یک بخش ضروری در هر آزمایه، تعریف خروجی یا نتایج مورد انتظار است.

این اصل بدیهی، یکی از شایعترین اشتباهات در آزمایش برنامه ها است. این اصل هم بر پایه روان شناسی انسان قرار دارد. اگر نتیجه مورد انتظار یک آزمایه از پیش تعریف نشده باشد، احتمال دارد که یک جواب ظاهر فریب ولی اشتباه، به عنوان یک جواب صحیح و معتبر تلقی گردد زیرا در روان شناسی اصلی وجود دارد با این عنوان که «چشم چیزی را می بیند که می خواهد». به عبارت دیگر، تمایل ناخودآگاه ما، دیدن جواب صحیح است. یک راه برای مقابله با این تمایل، تعریف دقیق و با جزئیات کامل خروجی مورد انتظار برنامه است. بنابراین، هر آزمایه باید شامل دو بخش باشد:

۱) توصیف داده های ورودی برنامه.

۲) تعریف دقیق خروجی صحیح برنامه برای آن مجموعه از داده های ورودی.

اگر انتظاراتی وجود نداشته باشد، هیچ چیز ما را شگفت زده نخواهد کرد.

اصل ۲: یک برنامه نویس باید از آزمایش برنامه ای که خودش نوشته است اجتناب کند.

هر نویسنده ای می داند (یا باید بداند) که سعی در ویرایش یا غلط گیری متنی که خودش نوشته، ایده درستی نیست. شما خودتان می دانید که آن متن چه چیزی قرار است بگوید و ممکن است اگر واقعاً چیز دیگری را القاء می کند نتوانید آن را تشخیص دهید. به علاوه، هیچکس نمی خواهد در کارهای خود خطایی را کشف کند. چنین موضوعی در مورد نویسندگان نرم افزار نیز صادق است.

مشکل دیگری که پیش می آید این است که پس از آن که برنامه نویسی با دید «مثبت» و سازنده به طراحی و کد کردن برنامه اش پرداخت، بسیار مشکل است که ناگهان تغییر دیدگاه داده و با چشم «منفی» و تخریبی به آن برنامه بنگرد.

بسیاری از صاحب خانه ها می دانند که کندن کاغذ دیواری (فرایند تخریبی) کار ساده ای نیست امّا همین کار چنانچه کاغذ دیواری ها را قبلاً با دستان خودتان چسبانده باشید، بسیار سخت تر و ناراحت کننده تر جلوه می کند. به طریق مشابه، بسیاری از برنامه نویسان نیز نمی توانند برنامه های خود را به نحو مؤثری آزمایش کنند زیرا نمی توانند با تغییر ذهنیت لازم به منظور یافتن و نشان دادن خطاهای کار خودشان، کنار آیند. به علاوه، گاهی اوقات یک برنامه نویس به طور ناخودآگاه از ترس تحقیر در مقابل همکاران یا مدیر پروژه یا مشتریان، از کشف خطاها اجتناب می کند. علاوه بر این جنبه های روان شناسانه، مشکل مهم دیگری نیز وجود دارد و آن این که خطاهای برنامه ممکن است به علّت درک نادرست مشخصات ۴ مسأله از سوی برنامه نویس باشد. در این صورت، برنامه نویس همان درک نادرست خود را به آزمایش برنامه خودش نیز منتقل خواهدکرد.

البته این بدان معنی نیست که آزمایش برنامه توسط خود برنامه نویس امکانپذیر نیست. بلکه اگر کس دیگری آزمایش را انجام دهد، مؤثرتر و موفقیت آمیزتر خواهد بود.

توجه داشته باشید که این مسأله در مورد اشکالزدایی ۵ (تصحیح خطاهای یافته شده) صادق نیست. اشکالزدایی توسط برنامه نویس اولیه بسیار مؤثرتر است.

اصل ۳: یک نهاد برنامه نویسی نباید برنامه های خود را آزمایش کند.

استدلالی که در این مورد وجود دارد مشابه اصل قبلی است. سازمان پروژه، از بسیاری جهات، همانند یک موجود زنده است با همان مشکلات روانی که برای یک برنامه نویس منفرد وجود دارد. به علاوه، در اغلب محیط ها کار یک نهاد برنامه نویسی یا یک مدیر پروژه عمدتاً بر اساس توانایی تولید نرم افزار در یک زمان مشخص و با یک هزینه خاص، مورد ارزیابی قرار می گیرد. یک دلیلش این است که اندازه گیری زمان وهزینه مصرف شده آسان است امّا اندازه گیری قابلیت اطمینان یک برنامه و کمّی کردن آن بسیار دشوار می باشد. به این جهت، برای یک نهاد برنامه نویسی، مشکل است که به آزمایش واقعگرایانه و جدّی برنامه هایی که خود تولید کرده بپردازد زیرا فرایند آزمایش اگر بخواهد با تعریف درست انجام شود، احتمال اتمام پروژه در موعد مقرّر و با هزینه معیّن را کاهش خواهد داد.

باز هم تکرار می کنیم که برای یک نهاد برنامه نویسی غیر ممکن نیست که خطاهای برنامه های خود را کشف کند امّا انجام این کار توسط یک نهاد مستقل، بهتر و اقتصادی تر است.

اصل ۴: نتایج هر آزمایش را به دقت مورد بررسی قرار دهید.

این احتمالاً واضح ترین و بدیهی ترین اصل است امّا این اصل هم معمولاً نادیده گرفته می شود. ما بارها شاهد بوده ایم که بسیاری از خطاها، حتی با وجودی که رد پای آن ها به وضوح در لیست خروجی ها قابل مشاهده بوده، یافته نشده اند. به عبارت دیگر، خطاهایی که در آزمایش های بعدی یافت می شوند غالباً آن هایی هستند که در بررسی نتایج آزمایش های قبلی نادیده گرفته شده بودند.

اصل ۵: آزمایه ها باید هم برای ورودی های معتبر و مورد انتظار و هم برای ورودی های نامعتبر و غیر منتظره نوشته شوند.

تمایل طبیعی به هنگام آزمایش برنامه، تمرکز بر وضعیت های معتبر و مورد انتظار ورودی ها است و این گاهی باعث غفلت از وضعیت های غیر منتظره و نامعتبر می شود. معمولاً آزمایه هایی که بیانگر این گونه وضعیت ها هستند، خطاهای بیشتری را نسبت به آزمایه های شرایط معتبر ورودی کشف می کنند.

اصل ۶: آزمایش برنامه به منظور بررسی این که برنامه کاری که باید انجام دهد را انجام می دهد یا نه، تنها یک بخش از کار است. بخش دیگر، باید بررسی این باشد که برنامه کاری که نباید انجام دهد را انجام می دهد یا نه.

این اصل نتیجه منطقی اصل قبلی است. برنامه ها باید برای اثرات جانبی ناخواسته، مورد بررسی و آزمایش قرار بگیرند. برای مثال، یک برنامه حقوق و دستمزد که برگه های حقوق را به صورت صحیح تولید می کند هنوز ممکن است دارای خطا باشد. مثلاً برگه های حقوق اضافی برای کارمندانی که وجود ندارند تولید کند و یا روی نخستین رکورد پرونده کارکنان، اطلاعاتی را به اشتباه بنویسد.

اصل ۷: آزمایه ها را هرگز دور نیاندازید مگر آن که خود برنامه، دور انداختنی باشد.

این مسأله معمولاً در مورد سیستم های تعاملی ۶ که برای آزمایش برنامه ها به کار می روند پیش می آید. یک روش متداول این است که فرد در پشت پایانه می نشیند و آزمایه ها را فی البداهه می سازد و سپس برنامه ها را با آن ها مورد آزمایش قرار می دهد. مشکل اصلی این است که آزمایه ها که سرمایه باارزشی هستند، در این محیط پس از تکمیل شدن آزمایش، از بین می روند. هرگاه لازم باشد که برنامه دوباره مورد آزمایش قرار گیرد (مثلاً پس از تصحیح یک خطا یا افزودن یک قابلیت تازه)، آزمایه ها باید دوباره از نو ساخته شوند. و چون این کار معمولاً زمان قابل ملاحظه ای می گیرد، مردم از آن اجتناب می کنند. به این جهت، آزمایش مجدّد برنامه، ندرتاً به دقت و کاملی آزمایش اولیه صورت می گیرد و این بدان معنی است که چنانچه اصلاحات باعث شود که بخشی از برنامه که قبلاً عملیاتی بود دچار مشکل و خطا گردد، این خطا غالباً کشف نشده باقی می ماند. نگهداری آزمایه ها و اجرای مجدّد آن ها پس از اعمال تغییرات در مؤلفه های دیگر برنامه را آزمایش پسنمائی ۷ می نامند.

اصل ۸: آزمایش نرم افزار را با این فرض اشتباه که خطایی یافته نخواهد شد، برنامه ریزی نکنید.

این اشتباهی است که غالباً مدیران پروژه انجام می دهند و نشانه ای است از کاربرد تعریف نادرست آزمایش. یعنی این فرض که آزمایش عبارت است از فرایند نشان دادن این که برنامه درست کار می کند. یکبار دیگر تأکید می کنیم که تعریف آزمایش، فرایند اجرای برنامه با نیّت یافتن خطاها است.

اصل ۹: احتمال وجود خطاهای بیشتر در یک بخش از برنامه، متناسب است با تعداد خطاهایی که تا کنون در آن بخش یافته شده است.

طریقه دیگری برای بیان این اصل این است که بگوئیم خطاها معمولاً به صورت خوشه ای پدید می آیند و این که در برنامه های معمولی، بعضی از بخش ها نسبت به سایر بخش ها بیشتر مستعد خطا هستند. هرچند، هیچکس تا کنون توضیح مناسب و جالبی درباره این که چرا این گونه است، ارائه نکرده است. آگاهی از این پدیده، اطلاعات با ارزشی درباره فرایند آزمایش در اختیار ما می گذارد. اگر در بخشی از برنامه، خطاهای بیشتری نسبت به سایر بخش ها یافته شد، آنگاه این پدیده به ما می گوید که بهتر است تلاش جداگانه و ویژه ای بر روی آزمایش آن بخش متمرکز گردد.

اصل ۱۰: آزمایش نرم افزار، کار بسیار خلاقانه و از نظر ذهنی چالش برانگیزی است.

این جمله احتمالاً درست است که خلاقیت مورد نیاز برای آزمایش یک برنامه بزرگ، بیشتر از خلاقیت مورد نیاز برای طراحی آن برنامه است. ما تا کنون متوجه شده ایم که تضمین عدم وجود خطا از طریق آزمایش امکانپذیر نیست. طراحی آزمایه های مناسب برای پوشش دادن بیشتر وضعیت های امکانپذیر برای یک برنامه، به مقدار زیادی خلاقیت نیاز دارد.

نتیجه گیری

به هنگام آزمایش نرم افزار، همیشه این سه اصل مهم را در نظر داشته باشید:

۱) آزمایش عبارت است از فرایند اجرای یک برنامه با هدف یافتن خطاها.

۲) یک آزمایه خوب آزمایه ای است که احتمال زیادی داشته باشد که خطاهایی که تا کنون یافته نشده اند را کشف کند.

۳) یک آزمایه موفق آزمایه ای است که یک خطای تاکنون کشف نشده را بیابد.

منبع : IRAN HELPDESK

تهیه کننده : کیان محمدی