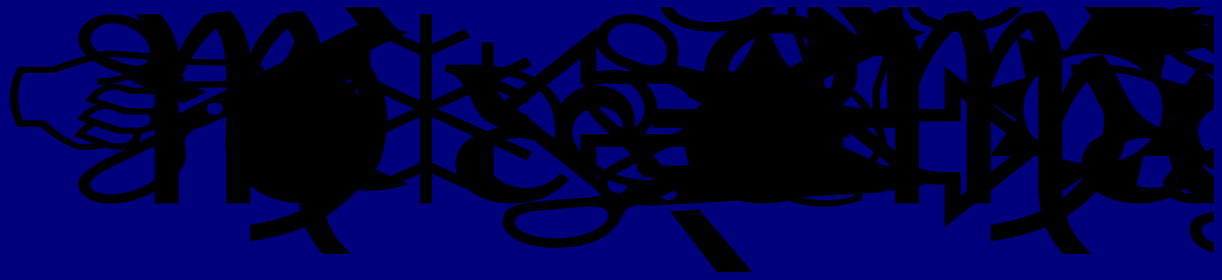


مهندسی نرم افزار

فرآیند تست نرم افزار

قبل از پرداختن به مبحث تست نرم افزار ، فرآیند های نرم افزاری را یادآوری و مرور می کنیم .

گرچه فرآیند های نرم افزاری مختلفی وجود دارند ، فعالیتهای اساسی وجود دارند که در تمام فرآیند های نرم افزاری مشترک اند :

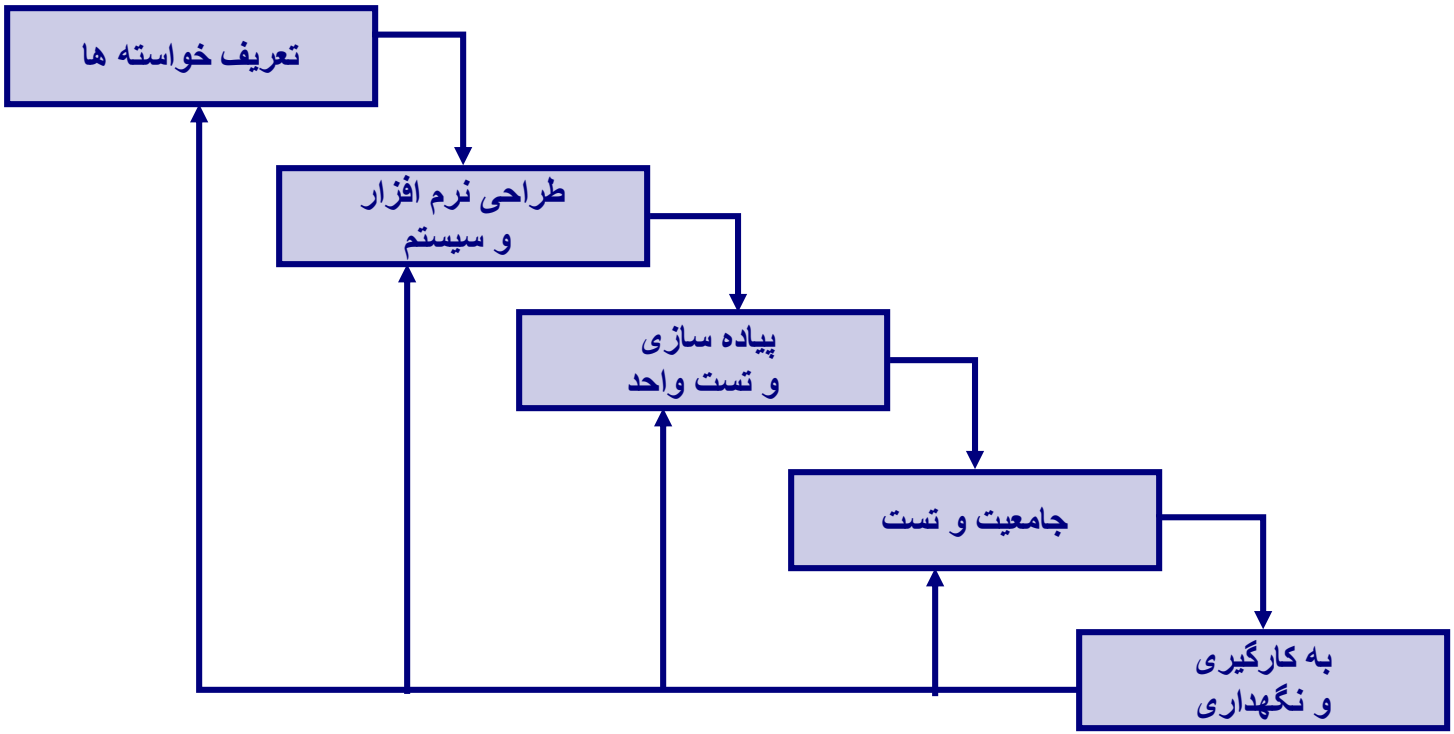
1- تعیین مشخصات نرم افزار : عملکرد نرم افزار و محدودیت های عملیاتی آن باید تعریف شود

2- طراحی و پیاده سازی نرم افزار : نرم افزار باید مطابق با مشخصات تعیین شده تولید شود

3- اعتبار سنجی نرم افزار : نرم افزار باید اعتبار سنجی شود تا تضمین گردد که خواسته های کاربر را انجام می دهد .

4- تکامل نرم افزار : نرم افزار باید تکامل یابد تا تغییر نیازهای کاربران را برآورده کند.

چرخه حیات نرم افزار



تست برنامه متداول ترین روشی است که بررسی می کند آیا برنامه مشخصاتش را برآورده می کند یا خیر و آیا خواسته های مشتری برآورده می شود یا خیر .

در ادامه به موارد زیر خواهیم پرداخت

1- تست سیستم

2-تست قطعه

3- طراحی مورد تست

4 - خودکار سازی تست

دو فعالیت اصلی تست عبارتند از تست قطعه (تست بخش هایی از سیستم) و تست سیستم (تست کل سیستم)

هدف مرحله تست قطعه ، کشف نقص ها از طریق تست هریک از قطعات است که این قطعات ممکن است توابع ، اشیا یا قطعات قابل استفاده مجدد باشند .

در اثنای تست سیستم ، این قطعات مجتمع می شوند تا زیر سیستم ها یا سیستم کامل را ایجاد کنند ، در این مرحله ، تست سیستم باید مشخص کند که سیستم خواسته های عملکردی و غیر عملکردی را برآورده می کند و رفتار غیر منتظره ای ندارد.

بطور کلی فرآیند تست نرم افزار دو هدف دارد :

1- به توسعه دهنده و مشتری نشان می دهد که نرم افزار خواسته هایش را برآورده می کند .

برای نرم افزار سفارشی ، معنایش این است که باید برای هر خواسته در اسناد خواسته های کاربر و سیستم ، حداقل یک تست وجود داشته باشد ، برای محصولات نرم افزاری کلی ، معنایش این است که برای تمام ویژگی هایی از سیستم که در نسخه محصول وجود دارند باید تست هایی وجود داشته باشد.

2- برای کشف عیب ها و نقص ها در نرم افزار ، که رفتار نرم افزار ، نادرست و نا مطلوب است ویا از مشخصاتش پیروی نمی کند.

تست نقص با ریشه یابی تمام انواع رفتارهای نا مطلوب سیستم ، مثل فروپاشی سیستم ، تعامل های نا خواسته با سیستم های دیگر ، محاسبات نادرست و تخریب داده ها سروکار دارد.

هدف اول منجر به تست اعتبارسنجی می شود که در آن انتظار دارید سیستم به درستی عمل کند.
این کار با استفاده از مجموعه ای از موارد تست انجام می شود که کاربر مورد انتظار سیستم را منعکس می کند .

هدف دوم منجر به تست نقص می شود که در آن موارد تست برای کشف نقص ها طراحی می شوند ، موارد تست الزاما مبهم هستند و لازم نیست مشخص کنند که سیستم به طور عادی چگونه کار می کند .

برای تست اعتبار سنجی ، تست موفق آن است که سیستم به درستی کار کند اما برای تست نقص ، تست موفق آن است که نقصی را نشان دهد که موجب می شود سیستم به درستی کار نکند .

تست نمی تواند نشان دهد که نرم افزار خالی از نقص است یا در هر شرایطی بر اساس انتظار کار می کند ، همواره ممکن است هرچه تست بیشتری انجام شود ، مشکلات بیشتری از سیستم پیدا شوند ، همان طور که دیکسترا می گوید : " تست فقط می تواند وجود خطا ها را نشان دهد نه عدم وجود آن ها را " .

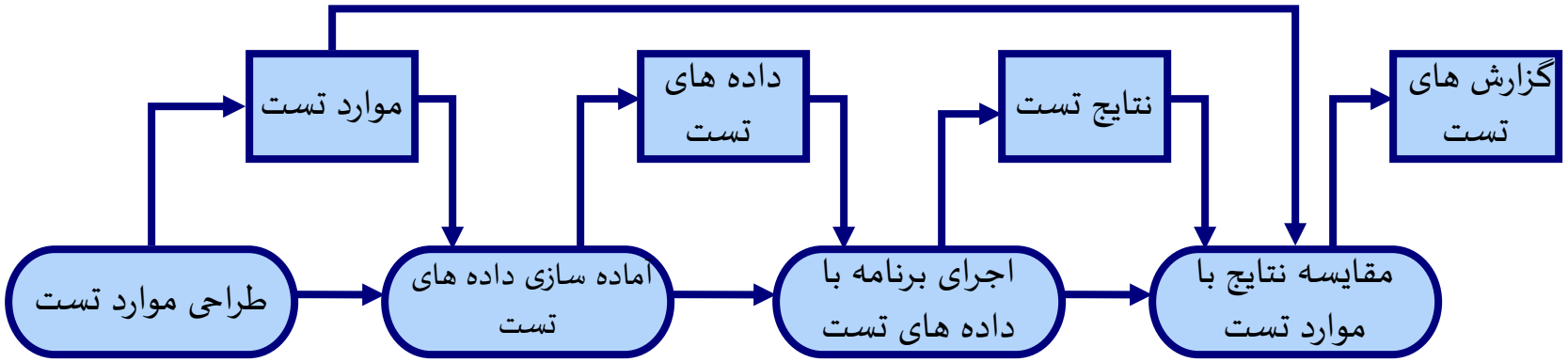
بنابراین هدف نهایی تست نرم افزار این است که توسعه دهندگان و مشتریان قانع شوند که نرم افزار برای به کارگیری مناسب است .

مراحل تست

توسط توسعه دهنده نرم افزار

توسط تیم تست مستقل

فرآیند تست عیب



1 – تست سیستم

تست سیستم شامل جامعیت دو قطعه است که ویژگی ها و عملکردهای سیستم را پیاده سازی می کنند و سپس این سیستم مجتمع تست می شود.

در فرآیند توسعه تکراری ، تست سیستم با تست گامی که باید به مشتری تحویل شود ، سروکار دارد و در فرآیند آبشاری ، تست سیستم با تست کل سیستم سروکار دارد.

برای اغلب سیستم های پیچیده ، دو فاز جداگانه برای تست سیستم وجود دارد.

1- تست جامعیت

2- تست نسخه ها

3- تست کارایی

1-1 تست جامعیت

که در آن تیم تست به کد اصلی (منبع) سیستم دسترسی دارند ، وقتی که مشکلی پیدا شد ، تیم جامعیت سعی می کند منبع مشکل را بیابد و قطعاتی را شناسایی کند که باید اشکال زدایی شوند ، تیم جامعیت اغلب با یافتن عیب ها سروکار دارد.

تست جامعیت بررسی می کند که قطعاتی که در سیستم استفاده شده اند ، واقعا با هم کار می کنند ، به درستی فراخوانی می شوند و داده های درستی را در زمان مناسبی از طریق واسط های آن ها عبور می دهد .

مشکل عمده در تست جامعیت ، یاقتن محل خطاهایی است که در اثنای این فرآیند مشخص می شوند تعامل های پیچیده ای بین قطعات سیستم وجود دارد و وقتی خروجی ناهنجاری تولید می شود، یافتن منبع خطا دشوار است .

برای یافتن خطا ها در جامعیت و تست سیستم باید از روش تدریجی استفاده کنید ، بدین گونه که در آغاز باید حداقل پیکربندی سیستم را جامعیت ببخشید و آن را امتحان کنید سپس قطعات دیگر را به این پیکربندی بیافزایید و تست را تکرار کنید . اگر مشکلاتی در این تست ها پیدا شود احتمالا به معنای آن است که مشکل ناشی از تعامل قطعه جدید است و منبع مشکل پیدا می شود و در نتیجه مکان یابی و ترمیم نقص آسان می شود .

1-2 تست نسخه ها

تست نسخه ، فرآیند تست نسخه ای از سیستم است که به مشتریان تحویل داده خواهد شد ، هدف اصلی این فرآیند ، افزایش اعتماد عرضه کننده نسبت به این نکته است که سیستم خواسته هایش را برآورده می کند و در این صورت می تواند به عنوان یک محصول به مشتری ارائه شود و برای این که نشان داده شود سیستم خواسته هایش را برآورده می کند باید نشان دهید که عملکرد ، کارایی و قابلیت اتکای مورد نظر را ارائه می کند و در حین کار عادی با شکست مواجه نمی شود .

تست نسخه معمولا یک فرآیند تست جعبه سیاه است که در آن ، تست ها از مشخصات سیستم تعیین می شوند ، سیستم به عنوان یک جعبه سیاه در نظر گرفته می شود که رفتار آن می تواند با مطالعه ورودی ها و خروجی ها تعیین شود . نام دیگر آن تست عملکردی است زیرا فقط با عملکرد نرم افزار سروکار دارد نه پیاده سازی آن .

برای اعتبار سنجی این نکته که سیستم خواسته هایش را برآورده می کند ، بهترین روش استفاده از تست بر اساس سناریو است ، که در آن تعدادی سناریو ایجاد کرده و موارد تستی را از این سناریو ها استخراج می کنید .

مولفینی مثل ویتاکر تجربه های تست خود را به صورت مجموعه ای از رهنمود ها ارائه کردند که منجر به موفقیت تست نقص می شوند ، بعضی از این رهنمود ها عبارتند از :

1. ورودی هایی را انتخاب کنید که سیستم را وادار به تولید تمام پیام های خطا می کنند.
2. ورودی هایی را طراحی کنید که منجر به سرریز میانگیر ورودی می شوند.
3. یک ورودی یا مجموعه ای از ورودی ها را چندین با تکرار کنید.
4. سیستم را وادار به تولید خروجی های نا معتبر کنید .
5. کاری کنید که نتایج محاسباتی خیلی بزرگ یا خیلی کوچک باشند.

1-3 تست کارایی

وقتی سیستم کاملا مجتمع شد ، می توان آن را برای ویژگی های جدید تست کرد ، مثل کارایی و قابلیت اعتماد . تست های کارایی باید طراحی شوند تا تضمین کنند سیستم می تواند بار مربوط به خود را پردازش کند ، این کار شامل برنامه ریزی مجموعه ای از تست ها است که در آن ها ، بار به تدریج افزایش می یابد تا کارایی سیستم غیر قابل قبول شود .

همانند سایر تست ها ، تست کارایی دو نکته را مشخص می کند :

نشان می دهد که سیستم خواسته هایش را برآورده می کند و مشکلات و نقص های موجود در سیستم را می یابد و برای اینکه تست شود آیا خواسته های کارایی برآورده می شوند یا خیر

تست فشار دو عملکرد دارد :

1. رفتار شکست سیستم را تست می کند .
2. فشار ممکن است خطاهایی را نشان دهد که در حالت عادی پیدا نمی شوند .

2- تست قطعه

تست قطعه (که گاهی تست واحد نام دارد) فرآیند تست هریک از قطعات در سیستم است . این تست ، یک تست نقص است و هدفش نشان داده عیب های موجود در این قطعات می باشد .

برای اغلب سیستم ها ، توسعه دهندگان قطعات ، مسئول تست قطعه اند .

انواع مختلفی از قطعات وجود دارند که در این مرحله باید تست شوند :

1. توابع یا متدهای موجود در یک شی .
2. کلاس های شی ای که صفات و متد های زیادی دارند.
3. قطعات مرکبی که از اشیا و توابع مختلفی تشکیل شده اند ، این قطعات مرکب واسط تعریف شده ای دارند که برای دستیابی به عملکرد آن ها به کار می روند .

توابع و متد ها ، ساده ترین نوع قطعه اند و تست مجموعه ای از فراخوانی های این روال ها با پارامترهای مختلف است .

با استفاده از روش های طراحی تست که در بخش بعدی مطرح می شود ، تست هایی را برای توابع طراحی کنید.

هنگام تست کلاس های شی تست ها را باید طوری طراحی کنید که تمام ویژگی های شی را دربرگیرد ، لذا تست کلاس شی باید شامل موارد زیر باشد .

1. تست ، مستقل از عملیات مربوط به هر شی است.
2. تنظیم و جامعیت تمام صفات مربوط به شی .
3. امتحان کردن شی در تمام حالت های ممکن ، یعنی تمام رویداد هایی که منجر به تغییر حالت در شی می شوند ، باید شبیه سازی گردد.

اگر از وراثت استفاده شود ، طراحی تست های کلاس شی دشوار است ، وقتی کلاس پایه عملیاتی دارد که تعدادی از زیر کلاس ها آن ها را به ارث می برند ، تمام این زیر کلاس ها باید با تمام عملیات ارثی تست شوند ، علتش هم این است که عملیات ارثی ممکن است فرض هایی راجع به سایر عملیات و صفات کرده باشند این فرض ها در هنگام به ارث برده شدن ، تغییر کنند .

2-1 تست واسط

قطعات بسیاری در سیستم وجود دارند که توابع یا اشیا نیستند بلکه قطعات مرکبی اند که از چند شی متعامل تشکیل شده اند . دستیابی به عملکرد های این اشیا از طریق واسط تعریف شده انجام می شود ، این قطعات مرکب تست می شوند تا مشخص شود رفتار واسط های آن ها بر اساس مشخصات انجام می شود .

تست واسط برای توسعه شی گرا و قطعه گرا مهم است ، اشیا و قطعات توسط واسط خود تعریف می شوند و ممکن است در ترکیب با قطعات دیگر در سیستم های مختلف ، مجددا مورد استفاده قرار گیرند .

خطاهای واسط در قطعه مرکب نمی تواند از طریق تست هر یک از اشیا یا قطعات مشخص شود چون خطاها در قطعه مرکب ممکن است به دلیل تعامل بین آن ها ایجاد شود.

انواع مختلفی از واسط ها بین قطعات برنامه وجود دارد و در نتیجه انواع مختلفی از خطاهای واسط ممکن است رخ دهد :

1. واسط های پارامتر : واسط هایی وجود دارند که در آن ها ، ارجاع به داده ها یا توابع ، از قطعه ای به قطعه دیگر ارسال می شود .
2. واسط های حافظه مشترک : این ها واسط هایی هستند که در آن ها بلوکی از حافظه بین زیر سیستم ها مشترک است ، داده ها توسط زیر سیستم ها در حافظه قرار می گیرند و توسط زیر سیستم های دیگری بازیابی می شوند .
3. واسط های ارسال پیام : واسط هایی هستند که در آن ها ، زیر سیستم ها از طریق ارسال پیام ، خدماتی را از زیر سیستم های دیگر درخواست می کنند ، پیام برگشتی حاوی اجرای خدمات است.
4. واسط های رویه ای : واسط هایی هستند که در آن ها ، یک زیر سیستم مجموعه ای از رویه هایی را بسته بندی می کند که توسط زیر سیستم های دیگر فراخوانی می شوند ، اشیا و انواع داده این واسط را دارند .

خطا های واسط ها ، یکی از متداول ترین شکل های خطا در سیستم های پیچیده اند ، این خطا ها سه دسته اند :

1. استفاده نادرست از واسط : قطعه فراخوان ، قطعه دیگری را فراخوانی می کند و در استفاده از واسط دچار خطا می شود ، این نوع خطا ها در واسط های پارامتر مرسوم است که در آن پارامتر ها ممکن است از نوع نادرستی باشند.
2. درک نادرست از واسط : قطعه فراخوان ، مشخصات واسط قطعه ی فراخوانی شده را به خوبی درک نمی کند و فرضیاتی را در مورد آن قطعه در نظر می گیرد ، قطعه فراخوانی شده آن طوری که باید عمل کند ، عمل نمی کند و در قطعه فراخوان منجر به رفتار غیر منتظره ای می شود . به عنوان مثال جستجوی باینری ممکن است ممکن است در آرایه نا مرتبی اجرا شود که حتما با شکست مواجه می شود .
3. خطا های زمانی : این ها در سیستم های بی درنگ رخ می دهد که از حافظه مشترک استفاده می کنند یا در قطعه ارسال پیام رخ می دهند ، ممکن است تولید کننده داده و مصرف کننده آن با سرعت های متفاوتی عمل کنند ، گرچه در طراحی واسط ها باید دقت کافی به خرج داد ، مصرف کننده می تواند به اطلاعات کهنه دسترسی داشته باشد زیرا تولید کننده اطلاعات واسط مشترک را نوسازی نکرده .

نکاتی در مورد تست واسط عبارتند از :

1. کدی را که باید تست شود ، امتحان کنید و فراخوانی قطعه خارجی را یادداشت کنید ، مجموعه ای از تست ها را طراحی کنید که در آن ها مقادیر مقادیر پارامتر هایی که به قطعات خارجی ارسال می شوند در حد نهایی بازه های خود هستند ، این مقادیر نهایی احتمالا ناسازگاری های واسط را نشان می دهند .
2. جایی که اشاره گر ها از طریق واسط ها ارسال می شوند ، واسط را با اشاره گر تهی تست کنید.
3. جایی که قطعه ای از طریق واسط های رویه ای فراخوانی می شود ، تست هایی را طراحی کنید که قطعه به شکست بیانجامد.
4. در سیستم های ارسال پیام از تست فشار استفاده کنید و تست هایی را طراحی کنید مه پیام هایی بیش از حد تولید کنند ، مشکلات زمانی احتمالا در این شرایط بروز می کنند.
5. جایی که چندین قطعه از طریق حافظه مشترک تعامل دارند ، تست هایی را طراحی کنید که ترتیب فعال شدن این قطعه را تغییر دهد . این تست ها ممکن است فرض های ضمنی برنامه نویسان را در مورد ترتیب قطعات روشن سازد .

3- طراحی موارد تست

طراحی موارد تست ، بخشی از تست سیستم و قطعه است که در آن ، موارد تستی مشخص می شوند که سیستم را تست می کنند (ورودی و خروجی های پیش بینی شده) و هدف فرآیند طراحی موارد تست ، ایجاد مجموعه ای از موارد تست است که نقص های برنامه را کشف می کنند و نشان می دهند که سیستم خواسته هایش را برآورده می کند .

برای طراحی مورد تست ، ویژگی ای از سیستم یا قطعه مورد آزمون را انتخاب کنید ، سپس مجموعه ای از ورودی ها را انتخاب کنید که آن ویژگی اجرا می کند ، خروجی های مورد انتظار را مستند سازی کنید و در صورت امکان ، یک بررسی خودکار طراحی کنید که که مشخص کند خروجی های واقعی و مورد انتظار یکسان هستند .

روش های مختلفی برای طراحی موارد تست وجود دارد :

1. تست بر اساس خواسته ها
2. تست اِفراز
3. تست ساخت یافته

3-1 تست بر اساس خواسته ها

در این روش موارد تستی طراحی می شوند که تا خواسته های سیستم را تست کنند ، این تست در مرحله تست سیستم به کار می رود که خواسته های سیستم معمولا توسط چندین قطعه پیاده سازی می شوند ، برای هر خواسته ، موارد تستی را مشخص خواهید کرد که می تواند نشان دهد سیستم خواسته هایش را برآورده می کند .

اصول کلی مهندسی خواسته ها این است که خواسته ها باید قابل تست باشند ، یعنی خواسته ها باید طوری نوشته شوند که تستی بتواند طراحی شود که یک بیننده بتواند بررسی کند خواسته برآورده شده است ، لذا تست بر اساس خواسته ها ، یک روش سیستماتیک برای طراحی مورد تست است که در آن ، هر خواسته در نظر گرفته می شود و مجموعه ای از تست ها برای آن مشخص می شود .

تست بر اساس خواسته ها ، یک تست اعتبار سنجی است و نه تست نقص و در این تست سعی می شود که نشان داده شود که سیستم خواسته ها را به طور مناسب پیاده سازی کرده است .

3-2 تست اِفراز

افراز ها گروهی از داده ها هستند که ویژگی های یکسانی دارند ، مثل اعداد منفی ، تمام اسامی با کمتر از 30 حرف ، تمام رویدادهایی که با انتخاب گزینه ای از منو به وجود می آیند و ...

تست افراز ، افرازهایی از ورودی ها و خروجی ها شناسایی و تست هایی طراحی می شوند ، به طوری که سیستم ورودی هایی را از تمام افراز ها می گیرد و خروجی ها را در تمام افراز ها تولید می کند .

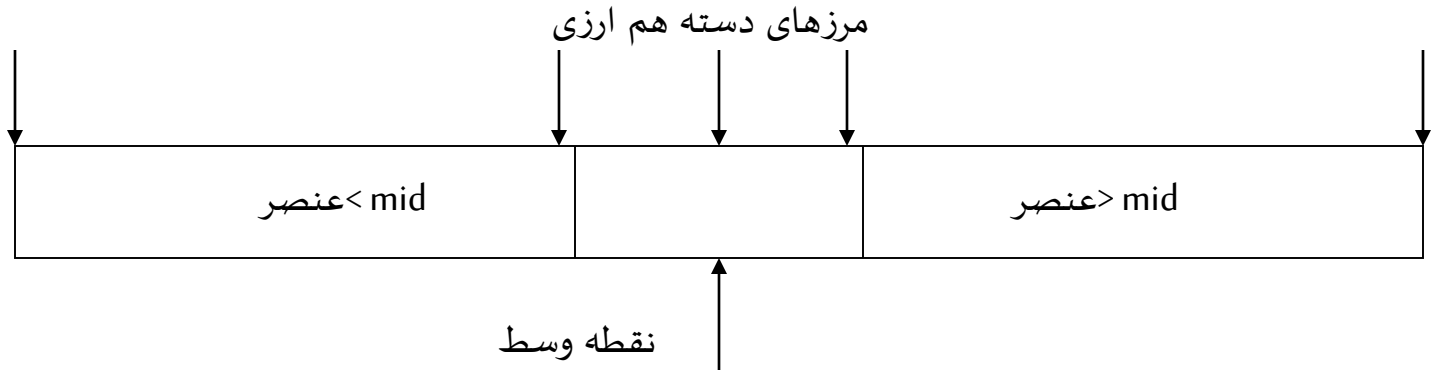
داده های ورودی و نتایج خروجی برنامه معمولا به چند دسته تقسیم می شوند که ویژگی های مشترکی دارند مثل اعداد مثبت و انتخاب منو ها ، برنامه ها برای هر عضو یک دسته ، به روش مشابهی عمل می کنند ، یعنی اگر برنامه ای را تست کنید که محاسباتی ندارد و به دو عدد مثبت نیاز دارد آن گاه انتظار دارید برنامه برای تمام اعداد مثبت به یک روش عمل کند به دلیل این رفتار معادل این دسته ها را افرازهای هم ارزی و یا دامنه گویند .

یک روش سیستماتیک برای طراحی مورد تست ، مبتنی بر شناسایی تمام افراز های مربوط به یک سیستم ویا قطعه است ، موارد تست طوری طراحی می شوند که ورودی ها یا خروجی ها در این افراز ها قرار می گیرند.

3-3 تست ساخت یافته

تست ساخت یافته روشی است برای تست که در آن با توجه به ساختار به پیاده سازی نرم افزار انجام می شود .

اطلاع از الگوریتمی که برای پیاده سازی بعضی از عملکرد ها استفاده شده است ، می تواند برای شناسایی افرازهای هم ارزی دیگر به کار گرفته شود ، برای تشریح این موضوع ، مشخصات یک روال جستجو به صورت جستجوی دودویی نمونه سازی شده است .



با امتحان کردن کد روال جستجو ، می بینیم که جستجوی دودویی فضای جستجو را به سه بخش تقسیم می کند . هر یک از این بخش ها یک افراز هم ارزی را می سازد و برای آزمایش کد موارد تستی را باید انتخاب کرد که در آن ها ، کلید در مرزهای این افراز قرار دارد.

3-4 تست مسیر

تست مسیر یک راهبرد تست ساختاری است که هدف آن ، امتحان کردن هر مسیر اجرای مستقل در قطعه یا برنامه است ، اگر هر مسیر مستقل اجرا شود ، آن گاه تمام دستورات قطعه حداقل یک بار اجرا می شوند ، علاوه بر این تمام دستورات شرطی برای مسیرهای درست و نادرست تست می شوند .

در فرآیند شی گرا تست مسیر وقتی استفاده می شود که متد های اشیا تست می شوند .

تعداد مسیر های برنامه معمولا متناسب با اندازه آن است ، وقتی پیمانه ها در سیستم ها مجتمع می شوند ، استفاده از تست ساختاری ممکن نیست . تکنیک های تست مسیر معمولا در تست واحد و مراحل تست پیمانه ها در فرآیند تست به کار می روند .

نقطه شروع تست مسیر ، یک گراف جریان برنامه است ، این گراف یک مدل اسکلتی از تمام مسیرهای جریان شامل گره های تصمیم گیری و یال هایی هستند که جریان کنترل را نشان می دهند .

4 – خودکار سازی تست

تست یک مرحله گران و پرکار در فرآیند نرم افزار است ، در نتیجه باید ابزارهای تست ایجاد شوند که امکاناتی را فراهم می کنند و استفاده از آن ها ، از هزینه فرآیند تست می کاهد .

یک از این ابزارها JUnit است که مجموعه ای از ابزارها برای پشتیبانی از فرآیند تست است .

پایان