فصل یازدهم

بازرسی و کنترل درستی انجام کار

هدف‌های رفتاری

۱- روش‌های Trace و Debug کردن برنامه را انجام دهد.

۲- ورود داده‌های فرضی برای بررسی درستی انجام کار را انجام دهد.

۳- روش‌های بررسی سندهای تجزیه و تحلیل را توضیح دهد.

۴- روش‌های بررسی مستندسازی سیستم اطلاعاتی را توضیح دهد.

۱-۱۱ مقدمه

چرا در برخی موارد سیستم‌های اطلاعاتی نیاز متقاضیان را برطرف نمی‌کند؟ چرا در برخی موارد سیستم‌های اطلاعاتی قابل اصلاح یا تغییر یا ارتقاء نمی‌باشند؟ راه‌حل چیست؟ ما از متخصصین سیستم‌های اطلاعتی سخنانی مانند این را زیاد شنیده‌ایم که : «ما نمی‌توانیم برنامه را تغییر دهیم زیرا که نمی‌دانیم چگونه آن را انجام دادیم یا ما نمی‌توانیم برنامه پایگاه اطلاعاتی را تعیین کنیم زیرا روشی که آن را طراحی کرده برای ما مشخص نیست یا .....» این شرایط وقتی رخ می‌دهد که سیستمی در حال عمل است و نیاز به تغییرات یا ارتقاء دارد ولی به دلیل ناآشنایی با روش‌های طراحی و برنامه‌نویسی آنلاین کارها ممکن نمی‌باشد. به اصطلاح در اکثر موارد اصلاح یک سیستم به مراتب سخت‌تر از ایجاد یک سیستم جدید است.

اما دلیل رخ دارد این مشکلات چیست؟ یک پاسخ عدم وجود مستندسازی برای سیستم اطلاعاتی و یا نقص در مستندسازی سیستم اطلاعاتی می‌باشد. بنابراین عدم مستندسازی سیستم اطلاعاتی می‌تواند به‌عنوان یک عامل در شکست پروژه در نظر گرفته شود. مستندسازی اغلب به عنوان نوع جدیدی از اطلاعات اضافی است که در دسترس سازمان و نیز تیم طراح سیستم قرار دارد.

۲-۱۱

تحلیل سیستم های اطلاعاتی

تحلیل سیستم‌های اطلاعاتی یکی از مراحل پیش‌نیاز برای توسعه سیستم است. در این مرحله انتظارات و نیازمندی‌های ذی‌نفعان سیستم شناسایی شده و گردش کار سیستم در قالب مدل‌ها و نشانه‌های استاندارد مستند می‌گردد تا از این رهگذر توسعه‌دهندگان سیستم بتوانند به یک شناخت جامع نسبت به سیستم دست یافته و آنرا منطبق بر نیازهای کاربران و ذی‌نفعان نهایی، طراحی و ایجاد نمایند. به منظور استاندارد نمودن ابزارها و روش‌های مدلسازی و مستندسازی سیستم‌های اطلاعاتی، مجموعه‌ای از مدل‌ها و متدولوژی‌ها ارائه شده‌اند که این مدل‌ها و متدولوژی‌ها به‌عنوان زبان مشترک تحلیل‌گران سیستم و برنامه‌نویسان به‌شمار می‌آیند. در قالب این مدل‌ها و متدولوژی‌ها مجموعه استانداردی از مستندات تهیه شده تا برنامه‌نویسان بتوانند با بهره‌گیری از این مستندات نیازهای کاربران را در قالب نرم‌افزارهای کاربردی پاسخگو باشند. متدولوژی RUP یکی از متدولوژی‌های تحلیل و مدلسازی سیستم‌های اطلاعاتی است که با رویکرد شیءگرا ارائه شده است. این متدولوژی از نشانه‌ها، علائم و مدل‌های زبان مدلسازی UML برای مدلسازی سیستم‌های اطلاعاتی استفاده می‌نماید. این متدولوژی در

مراحل چندگانه خود مستندات و خروجی‌های مختلفی را ارائه می‌نماید که می‌تواند توسط برنامه‌نویسان و توسعه‌دهندگان سیستم مورد استفاده قرار گیرد. در قالب این خدمت، قبل از اقدام برای توسعه یک سیستم اطلاعاتی و یا یک برنامه کاربردی، نیازمندی‌های کارکردی و غیرکارکردی سیستم با بهره‌گیری از متدولوژی RUP تعیین‌شده و الزامات توسعه آن در قالب مستند RFP ارائه می‌شود. در این حالت سازمان می‌تواند با به مناقصه‌گذاردن موضوع پروژه براساس مستندات RFQ و RFP نسبت به انتخاب پیمانکار مناسب برای تولید و استقرار سیستم اقدام نماید.

مستندسازی سیستم اطلاعاتی: مستندسازی یک پروژه ابزاری است برای ثبت و اصلاح و توسعه تصمیمات و فعالیت‌های مختلف که در چرخه حیاتی یک پروژه نقش دارند. مستندسازی می‌تواند در هدایت فعالیت‌ها ورسیدن به اهداف از پیش تعیین شده و نیز ارتقاء مسئولیت پاسخگوئی به مشتریان کمک نماید. مستندسازی در هر مرحله از اجرای پروژه باید به‌صورت جداگانه تهیه شود به‌صورتی که بتواند نقطه شروعی برای مرحله بعدی خود قرار گیرد.

طراح پروژه : یکی از مهمترین مراحل در ایجاد یک سیستم اطلاعاتی طرح پروژه می‌باشدکه سندی رسمی‌است که برای اداره و کنترل و اجرای پروژه مورد استفاده قرار می‌گیرد و رئوس اهداف سیستم را نشان می‌دهد طرح پروژه ممکن است در طول زمان طراحی و ایجاد پروژه تعدیل گردد.

مرحله اجرای پروژه:در مرحله اجرای پروژه چندین گزارش تولید خواهد شد که بر چگونگی روش انجام فعالیت‌ها و چگونگی استانداردهای کیفیت مورد نیاز و چگونگی برنامه فعالیت و بودجه پروژه تأکید می‌نماید.

مرحله کنترل و تغییر: در این مرحله طرح پروژه به روز یا تعدیل شده و آینده پروژه پیش‌بینی می‌گردد.

زبان استفاده‌کنندگان: مستندسازی سیستم‌های اطلاعاتی می‌بایست برحسب نیاز و درک هر گروه از استفاده‌کنندگان به‌صورت جداگانه تهیه شود. انواع استفاده کنندگان می‌تواند شامل: کاربران، پشتیبان سیستم، طراحان سیستم و مشتریان بالقوه و تیم توسعه فنی و ... باشد.

نقش مستندسازی سیستم‌های اطلاعاتی

❊مستندسازی یک سیستم اطلاعاتی به عنوان یکی از فعالیت‌های پر هزینه و زمانبر در اجرای پروژه درنظر گرفته می‌شود و لذا با وجود عقاید مختلف متخصصان در مورد نقش مستندسازی سیستم‌های اطلاعاتی، در اینجا لیست مزایای این جزء از ساختار سیستم را بیان می‌کنیم :

ـ وسیله روشن کردن جنبه‌های خاص سیستم در حال ارتقاء و یافتن پاسخ‌هایی برای مسائل مختلف که ممکن است در طول پروژه اتفاق بیفتد.

*نوعی قرارداد بین اعضای تیم پروژه و ذی‌نفعان سیستم و یا بین تیم پروژه و مدیریت سازمان در مورد نوع احتیاجاتی که سیستم بایدبرآورده کند یا می‌تواند برآورده کند.

ـ وسیله ارتباطی بین اعضای تیم پروژه که اجزائی ضروری برای انجام فعالیت‌های مختلف خاص هر مرحله از چرخه توسعه سیستم هستند. در این خصوص می‌توان گفت از طریق یک مستندسازی.

ـ اعضای هر مرحله مشکلات و نیازهای اعضای مراحل قبلی و بعدی رامطلع می‌گردند.

ـ حمایت در زمان تصمیم‌گیری در گزینش بین راه‌حل‌های متفاوت که از طریق تحلیل‌هایی که مستندسازی سیستم‌های اطلاعاتی در اختیار تیم پروژه و مدیریت قرار می‌دهد. انجام می‌گیرد.

ـ تجزیه تحلیل و طراحی مستندسازی سیستم یک نقطه شروع برای کاربری سیستم و تغییر و کاربردی کردن آن محسوب می‌شود. مستندسازی نهائی، عنصر اصلی برای اطمینان از درستی کار و حفظ سیستم می‌باشد.

ـ مستندسازی سیستم‌های اطلاعاتی باید هر زمانی‌که اجزای سیستم اولیه تغییر می‌کند، تعدیل شود.

تمام جنبه‌هایی که در بالا ذکر شد به چرخه‌های ارتقاء سیستم برمی‌گردد. اما هر فرد باید در نظر داشته باشد که مستندسازی سیستم، مدیریت کافی برای توسعه پروژه را فراهم می‌کند. و براساس جزییات و محتویات آن پروژه کنترل و نظارت می‌شود و نیز نیاز استفاده کنندگان که از طریق ملاقات‌های اسپانسرها با کاربران مشخص گردیده است در آن درج می‌شود.

اثرات نواقص مستندسازی سیستم‌های اطلاعاتی

نقص در جدول زمانی اولیه پیشرفت پروژه : به این صورت که ما نمی‌توانیم زمان مورد نیاز برای طراحی سیستم را در نظر بگیریم ونیز با در دست داشتن کل زمان موجود به تخصیص زمان برای مراحل مهم‌تر بپردازیم.

نقص در مشاهده بودجه عملیاتی طراحی سیستم: این امر باعث می‌گردد که هزینه‌های طراحی و آزمون و بازسازی و..... به طور صحیح قابل برآورد و تخصیص بین مراحل مختلف نباشد. ایجاد یک مستندسازی تکی برای تمام افرادی که نیاز به استفاده از آن را دارند، به مفهوم استفاده از زبانی است که تمامی‌استفاده‌کنندگان قابلیت فهم آن را ندارند.

نبود سندی روشن و قطعی از اجزای سیستم اطلاعاتی

*فقدان سیاست‌ها و استانداردها و فرآیندهای مورد نیاز که می‌بایست در طول طراحی و اجرای سیستم مورد نظر قرارگیرند.

*عدم امکان پاسخگوئی پشتیبان‌های سیستم اطلاعاتی در مواقع مورد نیاز به اصلاح و تغییرات.

ـ امکان وقوع اشتباه و خطا در بازسازی یا اصلاح یا تغییر سیستم اطلاعاتی.

ـ دوره زمانی طولانی‌تر برای تحلیل سیستم موجود: با توجه به عدم وجود شرحی از قواعد سیستم بین کاربران و پشتیبان برنامه نیاز به وقت بیشتر دارد.

ـ از قلم افتادن برخی نیازها و تقاضاهای کاربر.

مشکلات در آموزش سیستم

ارتقاء سیستم اطلاعاتی یک موضوع پیچیده است که می‌بایست هم از بعد اجزائی که باید شامل باشد و هم از بعد مسئولیت پاسخگوئی در مقابل استفاده‌کنندگان سیستم باید به آن توجه شود:

تیم پروژه باید مقداری از زمان خود را صرف مستندسازی سیستم بنمایند، زیرا از بین عوامل کیفیت سیستم، عامل انسانی مهمترین عامل در طراحی و اجرای یک سیستم است. زیرا پشتیبان برنامه باید از انگیزه‌های متقاضی و طراحان سیستم آگاه باشد تا بتواند برنامه را اصلاح کند.

مستندسازی سیستم اطلاعاتی باید برای هر گروه ازمتقاضیان به صورت جداگانه تهیه و نوشته شود.

مدیریت هر پروژه ای مسئول گردهم‌آوری و ترکیب مستندهای سیستم باشد.

مستندسازی سیستم همواره باید در هر مرحله از تغییرات در سیستم، تعدیل شود.

۱۱-۳

آشنایی با روش‌های debug و trace کردن برنامه

Debugger

Debugger برنامه‌ای است که به توسعه‌دهنده (Developer) اجازه می‌دهد برنامه را در حال اجرا مشاهده نماید. دو ویژگی مهم آنها قرار دادن Breakpoint و همچنین Trace کردن برنامه‌ها می‌باشد. این ویژگی‌ها به توسعه‌دهنده اجازه می‌دهد خطاهای برنامه را یافته و در جهت اصلاح آنها اقدام کند. Debugger یکی از مهمترین ابزارهای مهندسی معکوس بوده که از یک Disassembler برای برگرداندن کدها به زبان اسمبلی استفاده می‌نماید.

هیچ برنامه‌نویسی نمی‌تواند ادعا کند کدی که می‌نویسد، در اولین بار اجرا، بی‌شک درست کار خواهد کرد. حتی ماهرترین برنامه‌نویسان هم کد بی‌نقص نمی‌نویسند و همواره نیاز است که کدهای خود را رفع ایراد کنند تا از مشکلات آن باخبر شود، اما این اشکال‌زدایی یعنی چه و به چه صورت می‌توان آن را رفع کرد؟ شما یک برنامه نوشته‌اید

که خروجی آن مطابق با نتیجه دلخواه شما نیست و کدهای شما هم از نظر منطقی درست به‌نظر می‌آیند، ولی خروجی آن درست نیست. در این زمان باید خط به خط برنامه را گشت و مشکل را یافت. در غیر این‌صورت با افزایش حجم و تعداد خطوط برنامه، تصحیح هر کد به دشواری انجام خواهد شد. یکی از راه‌های یافتن اشکال این است که در هر مرحله مهم از برنامه، متغیرهای کد خود را مشاهده کنید و مطمئن شوید که درست در کدام خط و کجای برنامه داده‌های شما اشتباها تغییر می‌کنند. بعد از گذرانیدن این مرحله، باید به بررسی قطعه کدی که داده را خراب می‌کند بپردازید. یکی دیگر از روش‌های رفع خطا، استفاده از آزمایش واحد است. در این روش شما توابعی که نوشته‌اید را می‌آزمایید تا از درستی خروجی آن‌ها مطمئن شوید. برای این کار بسته‌های کاربردی مختلفی تولید شده است که کار را برای تست هر تابع شما راحت تر می‌کند. روش دیگر، استفاده از ابزارهای اشکال‌زدایی (debugger) است. این ابزارها تمامی‌این امور را خودشان انجام می‌دهند و در هر لحظه خروجی نرم‌افزار را به شما نشان می‌دهند. به این ترتیب متوجه خواهید شد که دقیقاً در کجای برنامه خود داده‌های خراب تولید می‌کنید. برخی از این ابزارها به همراه محیط توسعه سیستم (ide) عرضه می‌شوند و برخی به‌صورت افزونه قابل دریافت هستند. چون امروز بیشتر برنامه‌نویسان قدرت محیط ویژوال استودیو را به خوبی درک کرده‌اند، سراغ یکی از ابزارهای دیباگ ویژوال استودیو می‌رویم.

visual studio debugger این برنامه به همراه تمامی‌نسخه‌های ویژوال استودیو منتشر شده است و امکانات زیادی دارد که می‌توان از میان آنها به موارد زیر اشاره کرد: ۱- یکسان کردن سورس و سمبل کدها به طور کامل۲- اضافه شدن به پردازش‌های در حال اجرای روی سیستم برای اشکال‌زدایی (از این روش به منظور اشکال زدایی سرویس‌های ویندوزی نوشته شده در ویژوال استودیو استفاده می‌شود.)۳- امکان اشکال زدایی برنامه‌های نوشته شده در دات نت و برنامه‌های محلی نوشته شده در c++ ۴- امکان اشکال‌زدایی به صورت از راه دور ۵- قابلیت‌های ویژه و حرفه ای برای گذاشتن نقطه توقف (breakpoint) ۶- نمایش داده‌ها و وضعیت آنها.حال که با برخی از ویژگی‌های دیباگر ویژوال استودیو آشنا شدیم، نحوه استفاده از آن را در محیط ویژوال استودیو با هم مرور خواهیم کرد: در بخش منوها، با انتخاب گزینه debug، می‌توانید برنامه خود را در مود اشکال‌زدا یا بدون اشکال‌زدا اجرا کنید. اما تفاوت این دو حالت در چیست؟ در حالت اشکال‌زدا یک فایل شامل سیمبل‌های برنامه در کنار فایل کامپایل شده ایجاد می‌شود. با استفاده از این فایل می‌توان دوباره برنامه را اشکال‌زدایی کرد و حتی امکان سوءاستفاده از آن را به هکرها خواهید داد. ولی در حالت بدون اشکال‌زدا یا عرضه (release) فایل شامل سیمبل‌ها فعال نخواهد شد و گزینه‌های بهینه‌سازی کامپایلر فعال می‌شوند و از نظر حجم فایل ایجاد شده کوچک‌تر از فایل اصلی خواهد بود و سرعت اجرا شدن در این دو حالت در بعضی از الگوریتم‌ها تفاوت زیادی خواهندداشت.

جلوگیری از Debug کردن نرم‌افزار

CRC چیست؟

معمول‌ترین آن CRC۳۲ بوده که یک عدد ۳۲ بیتی است و برای هر داده‌ای قابل محاسبه می‌باشد از فایل گرفته تا یک مقدار رشته‌ای یا حتی یک قسمت از حافظه. همانطور که می‌دانید داده‌ها به‌صورت رشته‌ای از بایت‌ها قابل نمایش بوده که مقدار CRC هر بایت قابل محاسبه می‌باشد. الگوریتم‌های مختلفی برای محاسبه CRC وجود دارد ولی نکته قابل توجه این است که تمام آنها برای یک داده ثابت مقدار یکسانی را تولید می‌کنند. توجه داشته باشید که اگر برای یک داده مقدار CRC چند بار محاسبه شود مقادیر به دست آمده یکسان هستند و هر داده‌ای مقدار CRC مختص به خود را دارد به عبارت دیگر مقدار CRC هیچ دو داده‌ای با هم یکسان نیست. به‌طور معمول برای استفاده از قفل‌های سخت‌افزاری یا نرم‌افزاری برای ارتباط با قفل باید از DLL یا ActiveX استفاده نمود. به دلیل ماهیت خاص این نوع Objectها، آنها قابل جایگزین شدن بوده و با این کار می‌توان اداره نرم‌افزار را در دست گرفت. پیشنهاد می‌شود در صورت امکان با استفاده از روش‌های مختلف محاسبه CRC قبل از استفاده از متدهای یک DLL یا ActiveX مشخص نمایید که آیا این Object، Object اصلی است یا خیر؟

```
// Before using object
If (CalculateCRC object) = (MainCRC)
}
// you can use methods of object
{
```

استفاده از کدهای نامعلوم و نامشخص: Debug نرم‌افزار را مشکل ساخته و زمان بیشتری برای Crack کردن نرم‌افزار باید صرف شود. در واقع این کدها در نرم‌افزار عمل خاصی را انجام نداده و فقط در بخش‌هایی از نرم‌افزار که قرار است کلمه عبور یا شماره سریال وارد شود قرار گرفته و کار Debug را مشکل‌تر می‌کند. برای مثال دستوراتی به زبان اسمبلی وجود دارد (Macro) که با قرار دادن آنها در بین دستورات برنامه باعث می‌شود بعضی از Debuggerها را با مشکل مواجه شوند.

غیر فعال نکردن منوها و دکمه‌ها در نرم‌افزارهای Trial: اگر قرار است یک نرم‌افزار را به صورت Trial منتشر نمایید، منوها و دکمه‌ها را غیر فعال نکرده و کد اصلی را به‌طور مستقیم در نرم‌افزار قرار ندهید بعضی از ابزارهای نرم‌افزارسازی امکاناتی را در اختیار قرار داده تا بتوانید فقط کدهای مورد نظر را Compile نمایید :

```
{DEFINE Trial$}
{IFDEF Trial$}
// No Action
{ELSE$}
// No Compile Operation
{ENDIF$}
```

استفاده نکردن از رویدادها به طور مستقیم در Borland Delphi: به علت ماهیت خاص VCL(Visual Component Library) در Delphi یافتن Eventها از طریق Decompile کردن فایل EXE کار ساده ای می‌باشد بنابراین پیشنهاد می‌شود که برنامه‌نویسان دلفی از رویدادهای آن به‌طور مستقیم استفاده نکرده و به روش زیر عمل کنند :

```
Method1:=Button1.OnClick
Code Checksum  – CRC Calculate
```

محاسبه CRC برای بخشی از کد یا تمام فایل در زمان اجرا، راه مناسبی برای جلوگیری از Debug می‌باشد. زیرا Debuggerها به منظور قرار دادن Breakpoint در برنامه باید در کد تغییر ایجاد کنند. در این لحظه با محاسبه مجدد CRC برای متدهای درون برنامه می‌توان مشخص نمود که کد تغییر کرده است یا خیر. این روش نه‌تنها در مقابل Debuggerها مؤثر است بلکه می‌توان در مقابل Code Patching نیز از آن استفاده نمود.

```
// Calculate CRC in memory
if (Func۱CRC)!= MainCRC
{
// Do an action to stop your progra
}
```

Trace کردن برنامه: تست نرم‌افزار عموماً در چهار سطح مختلف صورت می‌گیرد که این چهار مرحله به‌صورت ترتیبی انجام می‌پذیرند و عبارتند از :

تست واحد (Unit testing)

تست مجتمع‌سازی (Integration Testing)

تست سیستم (System Testing)

تست پذیرش (Acceptance Testing)

تست واحد (Unit testing) : یک واحد کوچک‌ترین قسمت قابل تست یک نرم‌افزار می‌باشد. که این واحد در برنامه‌نویسی شیءگرا می‌تواند یک متد باشد و در برنامه نویسی رویه‌ای می‌تواند کل برنامه (در زبانی مانند کوبول) یا یک تابع و ... باشد. هدف در این سطح از تست این است که آیا واحد مورد نظر به تنهایی کاری را که باید انجام بدهد می‌دهد یا نه.

تست مجتمع‌سازی (Integration Testing): تست واحد را برای هر کدام از واحدها به صورت جداگانه انجام دادید و از صحت عملکرد آنها مطمئن شدید. همه واحدها به‌صورت منفرد به طور صحیح وظایف خود را انجام می‌دهند، آیا نیازی به تست اینکه وقتی واحدها کنار هم قرار گرفتند و ارتباط برقرار کردند وظایفشان را به شکل صحیح انجام می‌دهند هست یا نیست. فرض کنید ۲ نفر مشغول کاری هستند هنگامی‌که موارد مورد نیاز برای انجام کار به‌طور کامل مهیا باشد هر کدام از آن ۲ فرد می‌توانند کارشان را به شکل کامل انجام بدهند. اما اگر موارد مورد نیاز برای یکی از آنها توسط دیگری تأمین شود ممکن است موارد تهیه شده دقیقاً چیزی نباشد که فرد دوم نیاز دارد، یا زمانی زیاد برای تحویل آن موارد مورد نیاز باشد که عملکرد فرد دوم را با مشکل روبرو کند. پس ما نیاز داریم تا مطمئن شویم که آیا واحدها در کنار هم کار می‌کنند، به درستی فراخوانی می‌شوند، و داده‌های درستی را در زمان مناسبی از طریق واسط‌های آنها عبور می‌دهند. (تست مجتمع‌سازی یکی از مهمترین و شاید مهمترین سطح از تست باشد، به‌خصوص زمانی‌که سیستم تغییرات زیادی دارد بعد از انجام تغییرات هرگز این مرحله را فراموش نکنید. پس روش‌های مختلف تست مجتمع‌سازی را بررسی و مطالعه کنید.)

تست سیستم (System Testing): تست مجتمع‌سازی را برای نرم‌افزار مورد نظر انجام دادید و از این مطمئن شدید که تمام قطعات در کنار هم می‌توانند قرار بگیرند و بدون هیچ مشکلی وظایفشان را انجام دهند. قطعات در کنار هم مجتمع شده‌اند و پیکره اصلی نرم‌افزار تشکیل شد، ولی نرم‌افزار خود جزئی از یک سیستم است و نیاز است که با عناصر دیگر این سیستم مانند سخت افزارها ارتباط برقرار کند و با آنها مجتمع شود. پس نیاز داریم تا مطمئن شویم که سیستم به‌عنوان یک واحد به‌طور کامل عمل خواهد کرد و نیازمندی‌های سیستم را برآورده می‌کند. این سطح از تست آخرین سطحی است که توسط توسعه‌دهندگان صورت می‌گیرد تا قبل از تحویل نرم‌افزار به کاربر نهایی برای تست از عملکرد آن مطمئن شویم. برای نمونه موارد زیر در تست سیستم مورد بررسی قرار می‌گیرد:

تست امنیت (Security Testing): فرض کنید سیستم باید اطلاعات حساس و حیاتی را پردازش و مدیریت کند و افرادی هستند که به‌دنبال دسترسی غیرمجاز به این اطلاعات و سوءاستفاده از آن هستند. برای اطمینان از عملکرد سیستم در برابر نفوذگران ما باید مکانیزم امنیتی ایجاد شده در سیستم را بررسی کنیم تا مطمئن شویم که سیستم

می‌تواند نفوذهای غیر قانونی را تشخیص دهد و در برابر آنها عکس‌العمل نشان دهد.

تست بازیابی (Recovery Testing) : در این نوع آزمایش باعث ایجاد مشکل و از کار افتادن سیستم به روش‌های مختلف می‌شویم و بررسی می‌کنیم که آیا سیستم می‌تواند خود را به‌طور خودکار بازیابی کند و به فعالیت خود ادامه دهد.

تست پذیرش (Acceptance Testing): نرم‌افزار به‌طور کامل توسط توسعه‌دهندگان در تمام سطوح تست، با موفقیت تست شد، اما آیا نرم‌افزار واقعاً به طور کامل (آن‌گونه که کاربر نهایی می‌خواهد) کار می‌کند. آیا تمام نیازهای فعلی کاربر نهایی را برآورده می‌کند. پس ما به آزمایشی نیاز داریم که توسط کاربران نهایی، مشتریان و نه توسعه‌دهندگان صورت می‌گیرد و هدف آن است که کاربر مشخص کند عملیاتی که برنامه انجام می‌دهد نیازمندی‌های آنها را برآورده می‌کند یا نه. تست پذیرش دارای انواع مختلفی است که می‌توان به موارد زیر اشاره کرد:

تست آلفا: تست آلفا در سایت توسعه‌دهنده نرم‌افزار و در اغلب موارد توسط کارمندان داخلی و در بعضی از موارد توسط مشتری انجام می‌گیرد.

تست بتا: تست بتا در سایت مشتریان و توسط مشتریان که از سیستم استفاده خواهند کرد صورت می‌گیرد و مشکلات مشاهده شده را به توسعه‌دهندگان گزارش می‌کنند.

خلاصه فصل

تحلیل سیستم‌های اطلاعاتی یکی از مراحل پیش‌نیاز برای توسعه سیستم است. در این مرحله انتظارات و نیازمندی‌های ذی‌نفعان سیستم شناسایی شده و گردش کار سیستم در قالب مدل‌ها و نشانه‌های استاندارد مستند می‌گردد تا از این رهگذر توسعه‌دهندگان سیستم بتوانند به یک شناخت جامع نسبت به سیستم دست یافته و آن‌را منطبق بر نیازهای کاربران و ذی‌نفعان نهایی، طراحی و ایجاد نمایند.

مستندسازی یک پروژه ابزاری است برای ثبت و اصلاح و توسعه تصمیمات و فعالیت‌های مختلف که در چرخه حیاتی یک پروژه نقش دارند. مستندسازی می‌تواند در هدایت فعالیت‌ها و رسیدن به اهداف از پیش تعیین شده و نیز ارتقا مسئولیت پاسخگوئی به مشتریان کمک نماید.

Debugger برنامه‌ای است که به توسعه‌دهنده (Developer) اجازه می‌دهد برنامه را در حال اجرا مشاهده نماید. دو ویژگی مهم آنها قرار دادن Breakpoint و همچنین Trace کردن برنامه‌ها می‌باشد. این ویژگی‌ها به توسعه‌دهنده اجازه می‌دهد خطاهای برنامه را یافته و در جهت اصلاح آنها اقدام کند. Debugger یکی از مهم‌ترین ابزارهای مهندسی معکوس بوده که از یک Disassembler برای برگرداندن کدها به زبان اسمبلی استفاده می‌نماید.

تست نرم‌افزار عموماً در چهار سطح مختلف صورت می‌گیرد که این چهار مرحله به صورت ترتیبی انجام می‌پذیرند و عبارتند از :

تست واحد (Unit testing)

تست مجتمع‌سازی (Integration Testing)

تست سیستم (System Testing)

تست پذیرش (Acceptance Testing)

تست واحد (Unit testing) :یک واحد کوچک‌ترین قسمت قابل تست یک نرم‌افزار می‌باشد که این واحد در برنامه نویسی شیءگرا می‌تواند یک متد باشد و در برنامه‌نویسی رویه‌ای می‌تواند کل برنامه (در زبانی مانند کوبول) یا یک تابع و ... باشد. هدف در این سطح از تست این است که آیا واحد مورد نظر به تنهایی کاری را که باید انجام بدهد می‌دهد یا نه.

در تست مجتمع‌سازی(Integration Testing) ما نیاز داریم تا مطمئن شویم که آیا واحدها در کنار هم کار می‌کنند، به درستی فراخوانی می‌شوند، و داده‌های درستی را در زمان مناسبی از طریق واسط‌های آنها عبور می‌دهند.

نرم‌افزار خود جزئی از یک سیستم است و نیاز است که با عناصر دیگر این سیستم مانند سخت افزارها ارتباط برقرار کند و با آنها مجتمع شود. پس نیاز داریم تا مطمئن شویم که سیستم به‌عنوان یک واحد به‌طور کامل عمل خواهد کرد

و نیازمندی‌های سیستم را برآورده می‌کند. در این سطح تست سیستم انجام می‌شود.

پس ما به آزمایشی نیاز داریم که توسط کاربران نهایی، مشتریان و نه توسعه‌دهندگان صورت می‌گیرد و هدف آن است که کاربر مشخص کند عملیاتی که برنامه انجام می‌دهد نیازمندی‌های آنها را برآورده می‌کند یا نه. این تست، تست پذیرش نام دارد.

خودآزمایی

۱- با استفاده از نرم‌افزار Visio پروژه سیستم آموزشی هنرستان (خودآزمایی فصل قبل) را مدل‌سازی نمایید.

۲- با استفاده از امکانات زبان ویژوال بیسیک، پروژه تمرین بالا را Debug و Trace نمایید.

۳- پروژه تمرین یک را از طریق تست‌های چهارگانه ارزیابی کنید.