بسمه تعالی

مهندسی نرم افزار 2

خلاصه درس

تالیف پرسمن

ویراست هفتم

دانشگاه پیام نور مرکز ملایر

مدرس : مهندس زهرا رضایی

نیمسال اول 92 - 91

فصل 12 : طراحی مبتنی بر الگوها

در طراحی مبتنی بر الگوها برنامه کاربردی جدید، با یافتن مجموعه ای از راهکارهای اثبات شده در یک مجموعه مسائل کاملاً مشخص ایجاد می شود ، هر مساله و الگوی آن به وسیله یک الگوی طراحی توصیف می شود که توسطسایر مهندسان نرم افزار بررسی و فهرست بندی شده است که هنگام طراحی برنامه های دیگر با این مساله مواجه شده اند و راهکاری برای آن پیاده سازی کرده اند . هر الگوی طراحی برای بخشی از مساله که قرار است حل شود ، یک رویکرد و روش اثبات شده در اختیار شما می گذارد.

الگوهای طراحی:

الگوهای طراحی را میتوان یک قاعده سه بخشی دانست که واسط میان یک حیطه معین ، یک مساله و یک راهکار را بیان می کند . برای طراحی نرم افزار حیطه به خواننده این امکان را میدهد تا محیطی را که مساله در آن جای دارد درک کند و در یابد چه راهکارهای ممکن است در این محیط مناسب باشد . مجموعه ای از خواسته ها ،از جمله محدودیت ها و قید و بندها ، بعنوان سیستم نیروهای تاثیرگذار بر شیوه تفسیر مساله در حیطه اش و چگونگی بکارگیری موثر آن راهکار عمل می کند. نیروها آن مجوعه از خواص مساله صفات راهکار هستند که بر شیوه طراحی قیدو بند اعمال می کنند.

کوپلن مشخصات طراحی الگوهای اثربخش را چنین بر می شمارد.

مفهوم اثبات شده است : الگوها راهکارهایی را به دست می آورند که دارای سابقه باشند.

- راهکار، واضح نیست
- یک رابطه را توصیف میکند
-الگو دارای یک مولفه انسانی چشمگیر است.

الگوهای طراحی ، اگر به طور موثر استفاده شود از شما طراحی نرم افزار بهتر خواهدساخت.

انواع الگوها:

یکی از دلایل که مهندسان به الگوهای طراحی علاقه دارند آن است که انسان ها ذاتاً در تشخیص الگوها مهارت دارند . در جهان واقعی الگوهایی را می شناسیم که با گذر زمان و از طریقه تجربه بدست می آیند ما این الگوها را بلافاصله تشخیص می دهیم و ذاتاً می فهمیم که چه معنایی دارند و چگونه از آنها می توان استفاده کرد.

هنگام پرداختن به الگوهای طراحی در جستجوی شناسایی و مستند سازی الگوهای مولد هستیم ، یعنی الگویی را شناسایی می کنیم که جنبه ای مهم و تکرارپذیر از سیستم را توصیف می کند و شیوه ساخت آن جنبه را در سیستمی از نیروها که در یک حیطه ای مفروض منحصر به فرد هستند در اختیارمان قرار می دهد.

الگوهای طراحی : شامل طیف گسترده ای از اختراع ها وکاربردها می شوند .

الگوهای معماری: مسائل طراحی گسترده ای را توصیف می کنند که با به کارگیری یک رویکرد ساختاری حل می شوند .

الگوهای داده ای: مسائل داده گرای تکراری و مسائل مدل سازی داده را توصیف می کنند که در حل این مسائل قابل استفاده اند.

الگوهای مولفه ای : به مسائل مرتبط با توسعه زیر سیستم ها و مولفه ها ، شیوه برقراری ارتباط آنها با یکدیگر و تعیین مکان آنها در معماری بزرگتر می پردازند.

الگوی های طراحی واسط: مسائل واسط کاربری متداول و راهکار آنها را با سیستمی از نیروها توصیف می کنند که شامل خصوصیات کاربران نهایی می شود.

الگوهای تحت وب: به مسائلی اختصاص دارند که هنگام ساخت این نوع برنامه ها مشاهده می شوند و غالباً خود شامل بسیاری از الگوها می شوند .

* گاما و همکارانش سه نوع الگوها را کانون توجه قرار داده اند: الگو ایجادی ، الگوساختاری ، الگو رفتاری

الگو ایجادی: آنچه کانون توجه قرار می گیرد «ایجاد، ترکیب و نمایش» اشیاست.

الگو ساختاری: مسائل وراهکارهای مرتبط با چگونگی سازمان دهی و انسجام بخشیدن به اشیا برای ایجاد ساختاری بزرگتر کانون توجه قرار می گیرد.
الگو رفتاری: مسائل مرتبط با تقسیم مسئولیت ها میان اشیا و شیوه تاثیر گرفتن ارتباط میان اشیاست.

چارچوب ها :

در برخی موارد ممکن است برای کار طراحی ، نیاز به فراهم آوردن زیرساختار مختص یک پیاده سازی باشد که به آن چارچوب گفته می شود . چارچوب الگوی معماری نیست ، بلکه اسکلتی است با مجموعه ای از نقاط اتصال (یا قلاب ها) که به کمک آنها می توان این اسکلت را بر یک دامنه مساله خاص تطبیق داد. این نقاط اتصال به شما امکان می دهند تا کلاس و قابلیت های عملیاتی خاص مساله را در این اسکلت انسجام بخشید.
گاما و همکارانش اختلاف میان الگوهای طراحی و چارچوب ها را چنین شرح می دهند.
1. الگوهای طراحی، انتزاعی از چارچوب ها هستند.
2. الگوهای طراحی،عناصر معماری کوچکتر از چارچوب ها هستند.
3. تخصص یافتگی الگوهای طراحی کمتر از چارچوب ها هستند.

توصیف الگوها:

طراحی مبتنی بر الگو با شناسایی الگوها آغاز با جستجو برای تعیین این که آیا دیگران به این الگو پرداخته اند ادامه می یابد و با بکارگیری الگوی مناسب برای مساله پایان می یابد.
نام الگوها باید با احتیاط انتخاب گردد، یکی از مسائل فنی در طراحی مبتنی بر الگوها ناتوانی یافتن الگوها در میان صدها یا هزاران الگوست.
قالب الگو، ابزاری استاندارد برای توصیف الگوی طراحی فراهم می آورد . هر کدام از مدخل ها ، خصوصیاتی از الگوهای طراحی را فراهم می کند که می توان آنرا جستجو کرد . بطوری که الگوی مناسب را بتوان بدست آورد.

مخازن و زبان های الگو:

زبان الگوها شامل مجموعه ای از الگوها می شود که هر یک با به کارگیری یک قالب استاندارد شده توصیف می شود و با سایر الگوهای مربوطه ارتباط داده می شود تا مسائل موجود در یک دامنه کاربرد را با همکاری یکدیگر حل کنند . زبان الگوها مشابه با یک جزوه راهنمای ابرمتنی است که برای حل مساله در یک دامنه خاص بکار می رود .
ده ها زبان الگو برای طراحی نرم افزار پیشنهاد شده است در اکثر موارد ، الگوهای طراحی که بخشی از زبان الگوها هستند در مخزنی قرار داده می شوند که از طریق وب قابل دستیابی است و این مخزن نمایه ای از تمامی الگوهای طراحی فراهم می آورد.

طراحی مبتنی بر الگوها در حیطه (can text)

طراحی مبتنی بر الگو در خلا به کاربرده نمی شود ، مفاهیم و تکنیک های بحث شده برای طراحی معماری، طراحی در سطح مولفه ها و طراحی واسط همگی در ارتباط با رویکرد مبتنی بر الگو به کار برده می شوند.
هنگامی که کار خود را بعنوان طراح شروع کردید، همواره باید صفات کیفیتی را مدنظر داشته باشید ، این صفات راهی برای ارزیابی کیفیت نرم افزار تعیین می کنند ولی در دستیابی به آن کمک چندانی ارائه نمی دهند.

وظایف طراحی:

وظایف طراحی زیر هنگامی بکار برده می شود که از فلسفه طراحی مبتنی بر الگو استفاده شود:
1- مدل خواسته ها و توسعه سلسله مراتبی از مسائل را بررسی کنید.
2- تعیین کنید آیا زبان الگوی مناسبی برای مساله وجود دارد یا خیر.
3- با شروع از یک مساله گسترده تعیین کنید آیا یک یا چند الگوی معماری بر آن در دسترس است یا خیر

و ...

ساخت جداول سازمان دهی الگو:

مایکروسافت برای کمک به سازمان دهی الگوها یک جدول سازماندهی پیشنهاد می کند به نام جدول سازماندهی الگوها که این جدول را می توان بصورت یک مدل صفحه گسترده و با بکارگیری نمونه شکل زیر پیاده سازی کرد.

زیرساخت ها	پیاده سازی	کاربرد	بانک اطلاعاتی	
				داده ها(محتویات)
			نام الگوها	صورت مساله
	نام الگوها			صورت مساله
				معماری
		نام الگوها		صورت مساله
				در سطح مولفه
	نام الگوها	نام الگوها		صورت مساله
				واسط کاربر
	نام الگوها	نام الگوها		صورت مساله

اشتباهات متداول در طراحی:

طراحی مبتنی بر الگو می تواند از شما یک طراح نرم افزار بسازد ولی نیست. همانند همه روش های طراحی ، باید با اصل اول شروع کنید ، تاکید ورزیدن به مبانی کیفیت نرم افزار و حصول اطمینان از این که طراحی ، واقعاً نیازهای بیان شده در مدل خواسته ها را پوشش می دهد.

* یک الگو را به زور به کار نبرید حتی اگر مناسب مساله مدنظر باشد.

الگوهای معماری

هر معماری نرم افزار ممکن است چند الگو معماری داشته باشد که به مسائل گوناگون از قبیل همروندی، ماندگاری و توزیع مربوط می شوند.

بوش و بوچ چند دامنه الگو معماری تعریف کرده اند که چند مثال آن به شکل زیر است:

- کنترل دستیابی
- همروندی
- الگوی operating system process management
- توزیع (Distribution)
- ماندگاری (Persistence)

الگوهای طراحی در سطح مولفه ها:

الگوهای طراحی در سطح مولفه ها راهکارهای اثبات شده در اختیار شما قرار می دهند که به یک یا چند مساله فرضی استخراج شده از مدل خواسته های می پردازند . در بسیاری موارد، الگوهای طراحی از این نوع یک عنصر از سیستم را کانون توجه قرار می دهند.

الگوهای طراحی واسط کاربر:

صدها الگو برای واسط کاربر (UI) طی سالهای اخیر پیشنهاد شده اند که اکثر آنها در یکی از 10 گروه زیر قرار می گیرند.

واسط کاربر کامل: راهنمایی برای طراحی ساختار سطح بالا و گشت و گذار در سرتاسر واسط فراهنم می آورد.

- الگو Top Level Navigation : هنگامی استفاده می شود که یک سایت یا برنامه کاربردی چند قابلیت عملیاتی عمده را پیاده سازی کند . یک منوی سطح بالا فراهم می آورد که با لوگو یا آرمی همراه است که گشت و گذار مستقیم در هر کدام از قابلیت های عمده را فراهم می سازد.

- الگو Card Stack: هنگامی استفاده می شود که چند گروه از محتویات با قابلیت های عملیاتی خاص مرتبط بایک ویژگی یا قابلیت عملیاتی ، باید بطور تصادفی انتخاب شوند . ظاهری شبیه یک پشته از کارتهای برگه دار را ایجاد می کند که با کلیک کردن روی برگه های هر صفحه محتویات آن صفحه نمایش داده می شود.

-الگوی Fill-in-the-blanks : امکان وارد کردن همه داده ها ی حرفی- عددی در یک کادر مبتنی را فراهم می آورد.

- الگو sortable table : فهرستی بلند بالا از رکوردها را نشان می دهد که می توان با انتخاب عنوان هر ستون از جدول ،آن را مرتب کرد.

- الگو Bread crambs : هنگامی که کاربر با سلسله مراتب پیچیده ای از صفحات وب یا صفحات نمایش کار می کند یک مسیر کامل برای گشت و گذار ترسیم می کند.

-الگوی Edit in place : قابلیت ویرایش ساده متون را برای انواع معینی از محتویات در مکان نمایش آنها فراهم می آورد.

- الگو simplesearch : توانایی جستجو به دنبال یک وب سایت یا منبع داده های ماندگار برای یک آیتم داده ای ساده که توسط رشته ای حرفی – عددی توصیف می شود.

- الگو wizard : کاربر را در یک کار پیچیده گام به گام پیش می برد و برای کامل شدن این کار از طریق یک سری صفحات نمایش پنجره ای ساده ، راهنمایی لازم را فراهم می آورد.

- الگو shopping cart : فهرستی از آیتم های انتخاب شده برای خرید را فراهم می آورد.

- الگو Progress In dicator : هنگامی که عملیاتی بیش از nثانیه به طول انجامد، پیشرفت کار را نشان می دهد.

الگوهای طراحی برای برنامه های تحت وب:

هنگام پرداختن به مسائل طراحی مرتبط با ساخت برنامه های تحت وب، در نظر گرفتن گروه های الگو با در نظر گرفتن دو بعد می تواند مفید واقع گردد: کانون طراحی الگو و سطح دانبه بندی

کانون طراحی مشخص می کند که کدام جنبه از مدل طراحی مد نظر است.

دانبه بندی، سطح انتزاعی را مشخص می کند در نظر گرفته می شود.

کانون طراحی:

در سطح بالایی از انتزاع آغاز می شود ، و به تدریج به جزئیات و مشخصات آنها افزوده می شود، به بیان دیگر ، کانون طراحی با نزدیکتر شدن به طراحی باریک تر می شود . مسائلی که هنگام طراحی معماری اطلاعاتی برای برنامه تحت وب به آنها بر می خورید با مسائل که هنگام اجرای طراحی واسط مشاهده می کنید، تفاوت دارد. الگوهای برنامه تحت وب را میتوان بر اساس سطح کانون طراحی زیر گروه بندی کرد:

- الگوی معماری اطلاعاتی
- الگوهای گشت و گذار
- الگوهای ارائه
- الگوهای عملیاتی

دانیه بندی طراحی

از نظر سطح دانیه بندی الگوها را می توان در سطوح زیر تعریف کرد:

- الگوهای معماری

-الگوهای طراحی

- الگوهای مولفه ها

فصل 13 : طراحی برنامه های تحت وب چیست؟

طراحی برنامه های تحت وب شامل فعالیت های فنی و غیر فنی می شود که عبارتند از:

تعیین ظاهر برنامه های تحت وب-ایجاد چیدمان زیبا شناختی واسط کاربر-تعریف ساختار معماری کلی-توسعه محتوا و قابلیت عملیاتی که در معماری جای داده می شود و طراحی گشت و گذاری که در داخل برنامه ی تحت وب رخ می دهد.

طراحی: یک فعالیت مهندسی است که به ایجاد محصول با کیفیت بالا می انجامد.

اولسینا و همکارانش یک درخت خواسته های کیفیتی تهیه کرده اند که شامل صفات کیفیتی هستند.

کیفیت برنامه های تحت وب: قابلیت استفاده ، قابلیت اطمینان، بازدهی، قابلیت نگهداری، قابلیت عملیاتی

قابلیت استفاده : قابلیت درگ سایت در تمام جهان، ویژگی های زیبایی شناختی و واسط ، ویژگی های راهنما و بازخورد برخط ، ویژگی های خاص

قابلیت های عملیاتی : قابلیت جست وجو و بازیابی، ویژگی های گشت و گذار و مرورگری، ویژگی های مرتبط با دامنه کاربرد

قابلیت اطمینان : پردازش پیوندهای صحیح، رهایی از وضعیت خطا، اعتبارسنجی وبازیابی ورود کاربر

بازدهی : کارایی زمان پاسخ، سرعت ایجاد صفات، سرعت ایجاد گرافیک ها

قابلیت نگهداری : سهولت تصحیح ، انطباق پذیری، بسط پذیری

در میان این صفات افوت چهار صفت کیفیتی دیگر را ذکر کرده است:

امنیت-دسترس پذیری-گسترش پذیری-زمان عرضه به بازار

امنیت:

برنامه های تحت وب همکاری وهمبستگی سنگینی با بانک های اطلاعاتی شرکتی و دولتی مهم پیدا کرده اند.برنامه های کاربردی تجارت الکترونیک اطلاعات حساس مشتریان را استخراج و سپس ذخیره می کنند به این دلایل امنیت برنامه های تحت وب اهمیت بنیادی دارد.راهکار کلیدی توانایی برنامه و محیط سرور آن در رد دستیابی غیر مجاز و یا جلوگیری از حمله نفوذگران است.

دسترس پذیری:

از دیدگاه فنی دسترس پزیری میزانی از درصد زمان در دسترس بودن برنامه های تحت وب برای استفاده است.استفاده از ویژگی های در دسترس تنها روی یک مرورگر یا یک سکو برنامه تحت وب را از دسترس کسانی که مرورگر/سکوی دیگری در اختیار دارند،خارج می سازد.

گسترش پذیری:

ساخت یک برنامه تحت وب موفق کافی نیست،ساخت برنامه تحت وب که بار موفقیت راتحمل می کند و موفق تر شود نیز به همان اندازه اهمیت دارد.

اهداف طراحی جین کیسر در ستون نظم خود اهدافی را برای طراحی وب تعریف می کند:

سادگی-سازگاری-هویت-استحکام-قابلیت گشت و گذار-جاذبه ی بصری-همسازمندی

سادگی:

در برنامه های تحت وب همه چیز باید متعادل باشد-محتوا باید حاوی اطلاعات مفید و موجز باشد-از شیوه ی تحویلی استفاده کند-ظاهر برنامه دلپذیر باشد و آزار دهنده نباشد-معماری باید ساده ترین شیوه ی ممکن باشد-اهداف برنامه ی تحت وب را دست یافتنی کند-گشت گذار باید صریح باشد و سازوکارهای گشت و گذار باید به طور حسی برای کاربر نهایی باشد و استفاده از قابلیت ها باید آسان و درک آنها آسانتر باشد.

سازگاری:

محتوا باید به صورت سازگار ساخته شود-طراحی گرافیکی باید ظاهری سازکار در میان همه ی بخش های برنامه ی تحت وب ارایه دهند-طراحی معماری باید قالب هایی را وضع کنند که به یک ساختار ابررسانه ای منجر شود-طراحی ماسط باید شیوه های سازگار تعامل,گشت و گذار و نمایش محتوا تعریف شود, سازو کارهای گشت گذار را باید به طور سازگار در میان همه ی عناصر برنامه تحت وب به کاربرد.

هویت:

طراحی زیباشناختی,واسط و گشت و گذار در یک برنامه تحت وب باید دامنه کاربردی که برای آن ساخته می شود سازگار باشد و در سرتاسر مسیر طراحی,هویتی برای برنامه برقرار کند.

استحکام:

بر اساس هویتی که برقرار است برنامه تحت وب غالبا <<نویدی>> ضمنی به کاربر می دهد و کاربر انتظار محتوا و قابلیتهایی مستحکم را دارد که با نیازهای او در ارتباط باشد.

قابلیت های گشت و گذار:

گشت و گذار باید ساده و سازگار باشد،طراحی باید مبتنی بر حس و قابل پیش بینی باشد،قرار دادن پیوندهایی به محتوا و قابلیتهای اصلی برنامه در مکانی قابل پیش بینی نیز اهمیت دارد.

جاذبه ی بصری:

از میان همهی گروه های نرم افزار ، برنامه های کاربردی تحت وب بی تردید بصری ترین،پویاترین و...نوع نرم افزار است.

همسازمندی:

برنامه ها در محیط متنوع به کار گرفته خواهند شد و باید طوری طراحی شوند که با همه ی آنها همساز باشند.

طراحی واسط برنامه ی تحت وب:

یکی از چالش های طراحی واسط برای برنامه های تحت وب،ماهیت نامعین نقطه ورود کاربر است.یعنی کاربر ممکن است از صفحه ی اصلی وارد برنامه ی تحت وب شود یا ممکن است پیوندی او را به یکی از سطوح پایین تر در معماری برنامه ی تحت وب هدایت کرده باشد.طراحی برنامه تحت وب باید ویژگی هایی برای گشت و گذار در واسط فراهم بیاورد که شامل همه ی اشیای محتوایی شوند و فارغ از چگونگی ورود کاربر به سیستم،در دسترس باشد.

اهداف واسط:

1- ساخت پنجره ای سازگار فراروی محتوا و قابلیت های فراهم شده به وسیله واسط
2- راهنمایی کاربر از طریق یکسری تعامل با برنامه ی تحت وب
3- سازماندهی گزینه های گشت وگذار و محتوای در دسترس کاربر به منظور دستیابی به واسطی سازگار،باید از طراحی زیبایی شناسانه برای رسیدن به سیمایی یکپارچه استفاده کنید.

طراحی زیباشناسانه:

طراحی زیباشناسانه،که گاه طراحی گرافیکی نیز خوانده می شود،تلاشی هنرمندانه است که جنبه های طراحی برنامه های تحت وب را تکمیل می کند.برنامه ها ممکن است بدون طراحی گرافیکی کار کند ولی جاذبه ندارند.

مسایل مربوط به چیدمان: هر صفحه ی وب دارای مقدار محدودی منابع است که می توان از آن برای گرافیک های غیر عملیاتی،ویژگی های گشت گذار ،محتوای اطلاعاتی و قابلیت های عملیاتی اداره شده توسط کاربران استفاده کرد.

چند دستورالعمل کلی برای چیدمان:

از فضای خالی نترسید-بر محتوا تاکید کنید-عناصر را از چپ به راست و از بالا به پایین سازماندهی کنید-گشت گذار،محتوا و قابیلت عملیاتی را به لحاظ جغرافیایی گروه بندی گنید-«منابع»خود را با نوارهای جابه جایی توسعه ندهید-هنگام طراحی چیدمان،تفکیک و اندازه ی صفحه ی پنجره را مدنظر داشته باشید.

طراحی محتوا:

در طراحی محتوا، دو وظیفه طراحی متفاوت کانون توجه قرار می گیرد که هر کدام را افرادی با مجموعه مهارت های خاص اداره می کند.نخست،یک نمایش طراحی اشیاء محتوایی و سازکارهای لازم برای ایجاد رابطه میان آنها توسعه می یابد.به علاوه،اطلاعات درون هر شیء محتوایی خاص ایجاد می شود.وظیفه دوم ممکن است توسط نویسندگان مطالب،طراحان گرافیکی،سایرین اجرا شوند.

اشیای محتوایی:

رابطه میان اشیای محتوایی اشیای طراحی که محتوا را نشان می دهدمشابه رابطه میان کلاس های تحلیل و مؤلفه های طراحی است.شیء محتوایی صفاتی دارد که شامل اطلاعات خاص محتوا و صفات خاص پاده سازی است.

مسایل طراحی محتوا: هنگامی که همه ی اشیای محتوایی مدل سازی شدند،اطلاعاتی که قرار است هر شیء تحویل دهد،باید مدیریت شوند و سپس طوری فرمت بندی شوند که با نیازهای مشتری همخوانی داشته باشد.

طراحی معماری:

طراحی معماری،ارتباطی تنگاتنگ با اهداف تعیین شده برای برنامه تحت وب،محتوایی که قرار است ارائه شود،کاربران بازدیدکننده از برنامه و ... هر کس به عنوان طراح معماری باید معماری محتوا و معماری تحت وب را تعیین کند.در معماری محتوا،شیوه ساختاردهی اشیای محتوایی برای عرضه و گشت گذار کانون توجه قرار می گیرد.معماری برنامه تحت وب به شیوه ی ساختاردهی به برنامه کاربردی برای مدیریت تعامل با کاربر،اداره وظایف پردازش درونی،گشت و گذار مؤثر ارائه محتوا می پردازد.

معماری محتوا: در طراحی معماری محتوا،آنچه کانون توجه قرار می گیرد،تعریف ساختار ابررسانه ای کلی برنامه تحت وب است.

انواع ساختار محتوایی:

1 - **ساختار خطی**:هنگامی مشاهده می شوند که دنباله ای قابل پیش بینی از تعامل ها متداول باشند. یک مثال کلاسیک می تواند نمایش ترتیب وارد کردن سفارش محولات است.به موازاتی که محتوا و پردازش پیچیده تر می شوند جریا خطی خالص راه را برای ساختارهای خطی پیچیده تری باز می گند.

2 - **ساختار مشبک**: هنگامی مشاهده می شوند که می توان هنگام سازماندهی محتوای برنامه ی تحت وب در دو یا چند به کار برد.این برنامه هنگامی مفید واقع می شوند که محتوای کاملا منظم در آن ارائه گردد.

3 - **ساختار سلسله مراتبی**:بدون تردید متداول ترین معماری برنامه های تحت وب به شمار می روند.ساختار سلسله مراتبی برنامه را می توان طوری طراحی کرد که جریان افقی از میان شاخه های عمودی ساختار را امکان پذیر می سازد.گرچه با اینگونه انشعابهاگشت گذار در میان محتوای برنامه سرعت می گیرد،می تواند برای برخی کاربران به سردرگمی بینجامد.

4 - **ساختارهای شبکه ای**:

مشابه با بسیاری از شیوه های معماری است که برای سیستم های شئ گرا تکامل پیدا می کند.مؤلفه های معماری طوری طراحی میشوند که ممکن است کنترل را به هر مؤلفه دیگر سیستم تحویل دهد با این روش،انعطاف پذیری در گشت گذار به طور چشمگیری امکان پذیر می شود.

معماری برنامه تحت وب:

معماری برنامه های تحت وب زیرساختاری را توصیف می کندکه سیستم یا برنامه ی کاربردی مبتنی بر وب را قادر می سازد تا به اهداف تجاری خود دست پیدا کنند.

جدا نگه داشتن واسط،برنامه ی کاربردی و گشت و گذار پیاده سازی را تسهیل و استفاده مجدد را بهبود می بخشد.

معماری مدل-نما-کنترلگر(MVC) :یکی از چند مدل زیرساختی برای برنامه های تحت وب است که واسط کاربر را از قابلیت عملیاتی و محتوای اطلاعاتی آن حفظ میسازد.

مدل حاوی همه ی محتوای خاص برنامه ی کاربردی و منطق پردازش،کلیه اشیای محتوایی،دستیابی به منابع داده ای/اطلاعاتی خارجی و کلیه ی قابلیت های عملیاتی پردازشی می شود که کاربر نهایی به آن نیاز دارد.

کنترلگر،دستیابی به مدل و نما را مدیریت می کند و جریان داده ها را میان آنها هماهنگ می سازد.یک برنامه تحت وب ،«نما توسط کنترگر با داده های بدست آمده از مدل،بر اساس ورودی کاربر،به هنگام می شود».

طراحی گشت و گزار:

در این طراحی باید دو اصل را مشخص کرد:

1-معناشناسی گشت و گذار را برلی کاربران متفاوت سایت مشخص کنید.

2-مکانیک(نحوه)دستیابی به گشت و گذار را تعریف کنید.

معناشناسی گشت و گذار:

به یکسری واحدهای معناشناختی گشت و گذار (NSU)برمی خورند.«مجموعه ای ازاطلاعات و ساختارهای گشت وگذار مرتبط که با همکاری یکدیگر،زبرمجموعه ای از خواسته های مرتبط با کاربر را برآورده می سازند.»

همه یNSU از مجموعه ای عناصر گشت و گذار موسوم به راه های گشت وگذار (WON)تشکیل میشوند.یک WONنشانگر بهترین مسیر گشت و گذار برای دستیابی به هدف گشت و گذار برای دستیابی به هدف گشت و گذار برای نوع خاصی کاربر است.هرWON به صورت مجموعه ای از گره های گشت و گذار (NN)سازماندهی می شود که از طریق پیوندهای گشت و گذار باهم در ارتباط هستند.در برخی موارد،یک پیوند گشت و گذار ممکن است خود یک NSUدیگر باشد.پس یک WONگره های گشت و گذار و سپس پیوندهایی را مشخص می کند که گشت و گذار میان این گره ها را میسر می کند.

طراحی در سطح مؤلفه ها:

برنامه های تحت وب مدرن قابلیت هایی را ارائه می دهند که پیوسته بر پیچیدگی آنها افزوده می شود و 1-پردازش های محلی را برای ایجاد محتوا و قابلیت های گشت و گذاری را به شیوه ای پویا انجام می دهند.2-توانایی پردازش و انجام محاسبات مناسب را برای دامنه تجاری برنامه فراهم می سازد.3-درخواست از بانک های اطلاعاتی پیچیده و دستیابی به آنها را فراهم می سازد.4-واسط های داده ای را میان سیستم های شرکتی خارجی برقرار می کنند.

طراحی ابررسانه ها به روش شی گرا (OOHDM)

این طراحی از چهار فعالیت طراحی متفاوت تشکیل می شود:طراحی مفهومی-طراحی گشت وگذار-طراحی واسط انتزاعی وپیاده سازی.

طراحی مفهومی برای (OOHDM):طراحی مفهومی در OOHDM،نمایش از زیرسیستم ها،کلاسهاو روابط را ایجاد می کند که دامنه کاربرد رابرای برنامه تحت وب تعریف می کند.برای ایجاد کلاس مناسب،نمایشهای کلاسهای مرکب،نمودارهای همکاری و سایر اطلاعات می توان از UMLاستفاده کرد.

نمودارهای کلاس و کلاس های مرکب واطلاعات وابسته که به عنوان بخشی از تحلیل برنامه تهیه می شوند طی طراحی مفهومی مورد استفاده مجدد قرار می گیرد تا روابط میان کلاسها را به نمایش در آورند.

طراحی امکلنات گشت وگذار برای OOHDM:

در این طراحی مجموعه ای از اشیاع تعریف می شوند که از کلاسهای تعریف شده در طراحی مفهومی به دست می آیند.برای ایجاد پرونده های کاربرد مناسب،نمودارهای حالت ونمودارهای ترتیب که در فهم بهتر خواسته های گشت و گذار کمک می کند از UMLمی توان استفاده کرد.

OOHDMازیک مجموعه کلاسهای گشت وگذار از پیش تعریف شده،گره ها،پیوندها،لنگرهاوساختارهای دست یابی استفاده می کنند.هنگامی که کلاس های گشت وگذار تعریف شدند،OOHDM باگروه بندی اشیاءگشت وگذاردرمجموعه هایی به نام «حیطه»به فضای گشت وگذار ساختار می دهد.هر حیطه شامل توصیفی از ساختار گشت و گذار محلی-محدودیتهای ناشی از دستیابی اشیای محتوایی-متدهای مورد نیاز برای دستیابی به اشیای محتوایی می شود.

طراحی وپیاده سازی واسط انتزاعی:

در کنش طراحی واسط انتزاعی،اشیای واسطی مشخص می شود که کاربر در رخ دادن تعامل با برنامه تحت وب می بیند.مدل رسمی از اشیای واسط،که نمای داده های انتزاعی(ADV)خوانده می شود.برای به نمایش در آوردن ارتباط میان اشیای واسط واشیای گشت و گذار و خصوصیات رفتاری اشیای واسط به کار گرفته می شود.مدل ADVیک چیدمان ایستا تعریف می کند به علاوه مدل ADVحاوی یک مولفه رفتاری است که نشان می دهد رویدادهای خارجی چگونه شروع گشت وگذار را رقم می زند وهنگامی که کاربر با برنامه تعامل دارد کدام تبدیلات واسط رخ می دهد.

فعالیت پیاده سازی OOHDMنشانگر یک تعامل طراحی است که خاص محیطی است که برنامه تحت وب در آن پیاده سازی می شود.

فصل 14 : کیفیت نرم افزار

رابرت گلاس استدلال می کندکه یک رابطه ی مستقیم تر وجود دارد:

تحویل درزمانبندی وبودجه تعیین شده کیفیت خوب + محصول مطابق بااستاندارد= رضایت کاربر

یک فرایندنرم افزاری موثرکه به شیوه ای به کاربرده می شودکه محصولی مفیدایجادمی کندتاارزشی قابل سنجش برای سازندگان این محصول واستفاده کنندگان ازآن ایجادکند.

این تعریف به تاکیدبرسه نکته مهم کمک می کند:

1-فرایندنرم افزاراثربخش زیرساختی رابنامی کندکه هرگونه تلاش برای ساخت یک محصول نرم افزاری باکیفیت بالاراپشتیبانی می کند.جنبه های مدیریتی فرایندموازنه هاونقاطی برای بررسی ایجادمی کنندکه به پروژه کمک می کندتاازآشوب –یک عامل کلیدی درضعف کیفیت درامان بمانند.

یک نرم افزارباکیفیت بالاباافزودن ارزش برای تولیدکننده وکاربراین محصول نرم افزاری هم برای سازمان نرم افزاری وهم برای جامعه کاربران نهایی مزیت فراهم می کند.سازمان نرم افزاری ازآن روارزش افزوده کسب می کندکه نرم افزارباکیفیت بالابه تلاش کمتربرای نگهداری اشکال زدایی کمتروپشتیبانی کمتربرای مشتری نیازدارد.این به مهندسان نرم افزارامکان می دهدتاوقت بیشتری راصرف ایجادبرنامه های کاربردی جدیدکنندوکمتربه دوباره کاری بپردازند.جامعه کاربران ازآن روارزش افزوده کسب می کنندکه برنامه یک قابلیت مفیدفراهم می سازد.به طوری که یک فرایندتجاری خاص باسرعت بیشتری انجام شودنتیجه نهایی درآمدبیشتربرای محصول نرم افزاری منفعت بهترهنگام پشتیبانی یک برنامه کاربردی ازیک فرایندتجاری ویابهبوددسترسی به اطلاعات حیاتی برای شرکت تجاری خواهدبود.

ابعادکیفیتی گاروین :

1-کیفیت کارایی

2-قابلیت اطمینان

3-قابلیت سرویس دهی

4-زیبایی شناسی

عوامل کیفیتی مک کال

عوامل کیفیتی نرم افزاربرسه ویژگی های عملیاتی توانایی تحمل تغییرات وتطبیق پذیری یامحیط های جدید درستی حدبرآورده شدن مشخصه های یک برنامه توسط آن برنامه ورسیدن به اهداف مشتری .

قابلیت اطمینان حدی که می توان ازبرنامه انتظارداشت تاعملکردهای موردنظررابادقت لازم ارائه دهد.

بازدهی مقدارمنابع کامپیوتری وکدلازم برای آنکه برنامه قادربه اجرای عملکردهای خودباشد.

انسجام حدکنترل دستیابی افرادغیرمجازبه نرم افزاریاداده ها

قابلیت استفاده کارلازم برای فراگیری راه اندازی آماده کردن ورودی وتفسیرخروجی برنامه

قابلیت نگهداری کارلازم برای یافتن وتصحیح خطاهای برنامه (این تعریف بسیارمحدوداست)

انعطاف پذیری کارلازم برای اصلاح برنامه کامل شده

آزمون پذیری کارلازم برای آزمودن برنامه برای اطمینان یافتن ازاینکه عملکردموردنظررابه خوبی اجرامیکند.

حمل پذیری کارلازم برای انتقال دادن نرم افزارازیک سخت افزارویامحیط سیستم نرم افزاری به دیگری

قابلیت استفاده ی مجددحدی که می توان یک برنامه (یابخش هایی ازبرنامه)رادوباره درکاربردهای دیگرمرتبط باپکیج سازی ودامنه عملیاتی که برنامه اجرامی کنداستفاده کرد

قابلیت کارمتقابل کارلازم برای جفت کردن یک سیستم به سیستم دیگر.

عوامل کیفیتیI so 9126

استانداردI so 9126 به منظورتعیین صفات کیفیتی مهم برای نرم افزارهای کامپیوتری تدوین شده است. این استانداردشش صفت کلیدی رابرای کیفیت درنظرمی گیرد: قابلیت عملیاتی :حدی که نرم افزارنیازهای ذکرشده براساس این صفات رابرآورده می کند.

مناسب بودن صحیح بودن قابلیت کارمتقابل تطابق وامنیت

قابلیت استفاده : حدسهولت استفاده ازنرم افزاربراساس این صفات قابلیت درک قابلیت فراگیری قابلیت کاربآن

بازدهی :حدی که نرم افزاربراساس این صفات ازمنابع سیستم استفاده بهینه به عمل می آورد:

قابلیت نگهداری :

سهولت ترمیم نرم افزاربراساس این صفات تحلیل پذیری تغییرپذیری پاسداری وآزمون پذیر

حمل پذیری :سهولت انتقال نرم افزارازمحیطی به محیط دیگربراساس این صفات :تطبیق پذیری ناپایداری –مطابقت –قابلیت جایگزینی

عو**امل کیفیتی هدفمند**:

بصریت گرایی (Intuitiveness)

میزان پیروی واسط ازالگوهای کاربردموردانتظاربه طوری که حتی یک کاربرتازه کاربتواندبدون نیازبه آموزش زیادازآن استفاده کند.

بازدهی (Efficiency)میزانی ازامکان یافتن عملیات هاواطلاعات یااستفاده ازآنها

استحکام (Robustness)میزان اداره کردن داده های ورودی یدیاتعامل نامناسب کاربرتوسط نرم افزار

غنا (Richness)میزان ارائه مجموعه ای ازویژگی هابه وسیله ی واسط

تعیین کیفیت یک عامل کلیدی درردیادهای روزمره است

موضوعی بودن وتخصصی بودن درتعیین کیفیت نرم افزارنیزصادق است.

هزینه کیفیت شامل همه ی هزینه هایی می شودکه درجستجوی کیفیت یااجرای فعالیت های مرتبط باکیفیت وهزینه های ناشی ازفقدان کیفیت تحصیل می شوندبرای شناخت این هزینه هاسازمان بایدمعیارهایی جمع آوری کندکه بستری برای هزینه جاری کیفیت شناسایی فرصت هابرای کاهش دادن این هزینه هاوفراهم ساختن مبنایی بهنجارجهت مقایسه به دست دهد.هزینه ی کیفیت رامی توان به هزینه های مرتبط باپیش گیری ارزیابی وشکست تقسیم کرد.

هزینه های پیش گیری عبارتندازفعالیت های مدیریتی موردنیازبرای برنامه ریزی وهماهنگ کردن کلیه فعالیت های تضمین کیفیت (2) هزینه فعالیت های فنی برای توسعه وتکمیل مدل خواسته هاومدل طراحی (3)هزینه های برنامه ریزی آزمون و(4)هزینه همه ی آموزش های مرتبط بااین فعالیت ها

هزینه های ارزیابی شامل فعالیت های انجام شده برای به دست آوردن دیدی ازوضعیت محصول درنخستین گذرازهرفرایندمی شودمثال هایی ازهزینه های ارزیابی عبارتنداز:

هزینه اجرای بازبینی های فنی (فصل 15)برای محصولات کاری مهندسی نرم افزار

هزینه جمع آوری داده هاوارزیابی عبارتنداز:

هزینه جمع آوری داده هاوارزیابی معیارها(فصل 23)

هزینه آزمون واشکال زدایی (فصل های 18 تا21)

هزینه های شکست به آن دسته ازهزینه هایی گفته می شودکه درصورت عدم بروزخطاقبل یابعدازرسیدن محصول به دست مشتری ناپدیدمی شوندهزینه های شکست رامی توان به هزینه های شکست داخلی وهزینه های شکست خارجی تقسیم کرد.هزینه های شکست داخلی هنگامی تحصیل می شوندکه درمحصول وقبل ازرسیدن ان به مشتری کشف می شوناین هزینه هاعبارتنداز:

هزینه لازم برای اجرای دوباره کاری (ترمیم)برای تصحیح خطا

هزینه ناشی ازاثرات جانبی که دراثردوباره کاری هاایجادمی شود.

هزینه های مربوط به جمع آوری معیارهای کیفیتی که به سازمان این امکان رامی دهندتابه حالت های شکست دست پیداکند

هزینه های شکست خارجی به نقایصی مربوط می شودکه پس ازرسیدن محصول به دست مشتری کشف می شوند

همانطورکه انتظارمی رودهزینه های نسبی برای یافتن وترمیم خطاهایانقایص بارفتن ازپیش گیری به سوی کشف خطاوسپس شکست داخلی وسرانجام شکست خارجی به شدت افزایش می یابد.

منظوراین است که نرم افزارهای باکیفیت پایین هم برای سازنده وهم برای کاربرنهایی ایجادریسک می کنند

کیفیت ضعیف به ریسک هایی منجرمی شودکه برخی ازآنهابسیارجدی اند.

به بیان ساده نفوذدرنرم افزاری که کیفیت بالایی ازخودنشان ندهدراحت تراست ودرنتیجه نرم افزارهای باکیفیت پایین می توانندبه طورغیرمستقیم ریسک امنیتی راباتمام هزینه هاومشکلات مربوط به آن افزایش دهند.

تاثیرکنش های مدیریتی

کیفیت نرم افزارغالبابه همان اندازه که ازتصمیم گیری های فن آوری تاثیرمی پذیرداز‌تصمیم گیری های مدیریتی نیزتاثیرپذیراست.حتی بهترین کارهای مهندسی نرم افزارممکن است باتصمیم گیری های تجاری ضعیف وکنش های مدیریت پروژه ضعیف به بیراهه کشیده شود.

تصمیم گیری برآوردی

تصمیم گیری های زمان بندی

تصمیم گیری های مربوط به ریسک

دستیابی به کیفیت نرم افزار

کیفیت نرم افزارچیزی نیست که یک باره ظاهرشودنتیجه ی مدیریت خوب پروژه وکارمهندسی نرم افزارمستحکم است مدیریت وکاردرحیطه ی چهارفعالیت گسترده است که تیم نرم افزاری رادردستیابی به کیفیت بالای نرم افزاری یاری می دهد.روش های نرم افزارتکنیک های مدیریت پروژه کنش های کنترل کیفیت وتضمین کیفیت نرم افزار.

تکنیک های مدیریت پروژه

1-مدیرپروژه ازبرآوردهااستفاده کندتابیندآیاتاریخ های تحویل قابل تحقق هستند

2- وابستگی های زمان بندی درک شده وتیم دربرابروسوسه ی استفاده ازمیان برهامقاومت کند

3-برنامه ریزی برای ریسک انجام شده باشدبه طوری که مسائل تولیدآشوب نکنندکیفیت نرم افزاررامی توان به نحوی مثبت تحت تاثیرقرارداد

کنترل کیفیت

کنترل کیفیت شامل مجموعه ای ازکنش های مدیریت نرم افزارمی شودکه به کمک آنهامی توان اطمینان حاصل کردکه هرمحصول کاری اهداف کیفیتی اش رابرآورده ساخته است

تضمین کیفیت

تضمین کیفیت زیرساختی راتعیین می کندکه روش های مهندسی نرم افزارمدیریت پروژه موجه وکنش های کنترل کیفیت راکه همگی درساخت نرم افزارهای باکیفیت بالااهمیت محوردارند-پشتیبانی می کند.به علاوه تضمین کیفیت شامل یک مجموعه وظایف ممیزی وگزارش دهی می شودکه اثربخش بودن وکامل بودن کنش های کنترل را ارزیابی می کنند.

فصل 15 : تکنیک های مرور نرم افزار

مرورهای فنی ، اثربخش ترین سازوکار برای یافتن زودهنگام خطاها در فرایند نرم افزار به شمار می آیند.

مهندسان نرم افزار همراه با همکاران خود ، مرورهای فنی را انجام می دهند که از آنها به عنوان مرورهای فنی یا مرورهای نظیر یاد می شود.

گر خطای موجود در فرایند را زود هنگام بیابید ،تصحیح آن ، هزینه کمتری در بر خواهد داشت. به علاوه، طبیعت خطاها به گونه ای است که با پیشرفت فرایند ، قوت می گیرند. بنابراین ، یک خطای نسبتا جزئی که در اوایل فرایند برطرف نشده باشد،

بعدا در پروژه می تواند به مجموعه ای از خطاها منجر گردد. سر انجام ، این مرورها با کاستن از مقدار دوباره کاری هایی که ممکن است در آینده ضرورت پیدا کنند ، باعث صرفه جویی در زمان خواهند شد.

رویکرد شما در قبال مرورها بسته به درجه رسمیتی که برمیگزینید ، متغیر است. به طور کلی ، شش مرحله بکار می رود:

1-برنامه ریزی 2-آماده سازی 3-سازماندهی به جلسات 4-ذکر خطاها 5- انجام تصحیحات 6- وارسی درستی انجام تصحیحات

خروجی مرور، فهرستی از مسائل و یا خطاهاست که کشف نشده اند. به علاوه وضعیت فنی محصول کاری ذکر می شود چگونه اطمینان حاصل کنم که درست از انجام کار ها بر آمده ام؟

مرورهای نرم افزار به مثابه فیلترهایی برای فرایند مهندسی نرم افزار عمل می کنند یعنی در نقاط گوناگونی از توسعه نرم افزار اعمال می شوند و به کشف خطاها و نقایصی که قابل رفع باشند ، کمک می کنند.مرورهای نرم افزار به ((خالص سازی)) فعالیت های مهندسی نرم افزار که آنها را تحلیل ، طراحی و کد نویسی نامیدیم ، کمک می کنند.

تاثیر نقایص نرم افزار بر هزینه ها:

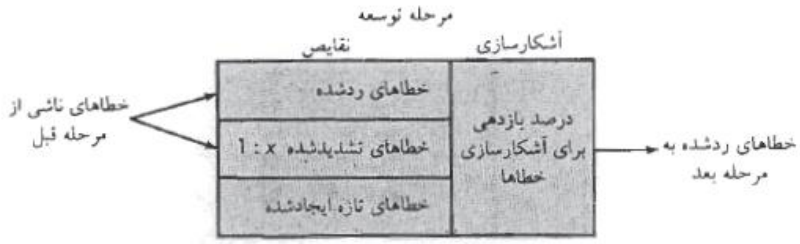
در حیطه فرایند نرم افزار ، واژه های نقص و عیب مترادف است. هر دو تداعی گر مشکلی هستند که پیش از ارائه نرم افزار به کاربر نهایی (یا فعالیت دیگری در فرایند نرم افزار) کشف می شوند. در فصول اولیه ، از واژه خطا برای مشکلات کیفیتی استفاده کردیم که توسط مهندسان نرم افزار (یا دیگران) پیش از ارائه نرم افزار به کاربر نهایی (یا فعالیت دیگری در فرایند نرم افزار) کشف می شوند.

هدف اصلی مرور های فنی رسمی ، یافتن خطاها در اثنای فرایند است به طوری که پس از ارائه نرم افزار به نقص تبدیل نشود. مزیت اشکار مرورهای فنی رسمی ، کشف زود هنگام خطاهاست ، به طوری که به مرحله بعدی فرایند نرم افزار انتشار پیدا نکنند.

چند مطالعه صنعتی نشان می دهد که فعالیت های طراحی بین 50 تا 65 % خطاها (و نهایتا همه نقایص) را در اثنای فرایند نرم افزار باعث می شوند. ولی ثابت شده است که تکنیک های مرور رسمی تا 75 % در کشف معایب طراحی موثر واقع می شوند.

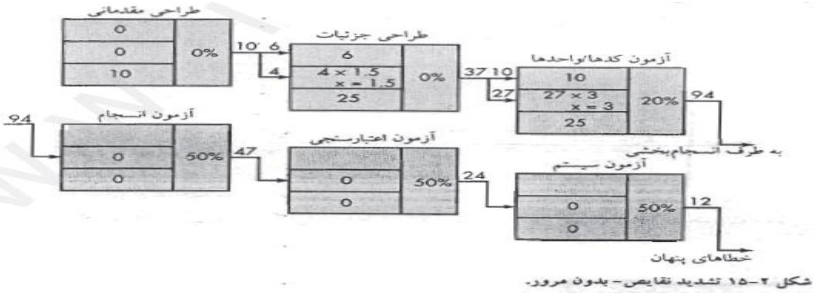
تشدید نقایص و حذف آنها

از مدل تشدید نقص می توان برای نمایش تولید و یافتن خطاها طی طراحی مقدماتی ، طراحی مشروح و مراحل کد نویسی در فرایند مهندسی نرم افزار استفاده کرد. این مدل به طور شماتیک در شکل 15-1 نشان داده شده است . چهارگوش خاکستری نشانه گر یک مرحله توسعه نرم افزار است . طی این مرحله، خطاها ممکن است به طور ناخواسته تولید شوند. مرور ممکن است از کشف خطاهای تازه تولید شده ، و خطاهای مراحل پیشین باز مانده و در نتیجه چند خطا به مرحله بعدی راه پیدا کنند.در برخی موارد ، خطاهایی که از مرحله قبلی عبور می کنند ، توسط کار فعلی تشدید می شوند (با ضریب تشدید X) . در تقسیمات فرعی این چهارگوش ، هریک از این ویژگی ها ودرصد بازدهی برای یافتن خطا که تابعی از کامل بودن مرور است ، نشان داده شده است .

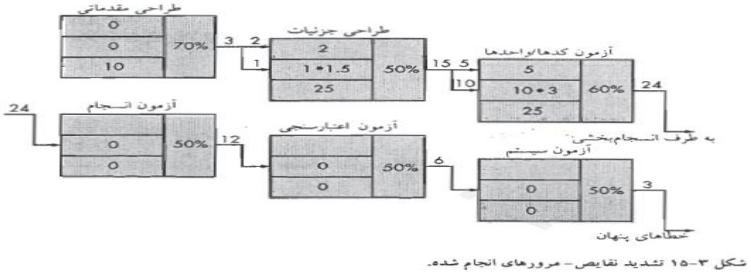


شکل ۱-۱۵ مدل تشدید نقایص.

در شکل 15-2 یک مثال فرضی از تشدید نقص برای فرایند توسعه نرم افزار نشان داده شده است که در آن هیچ مروری صورت نمی پذیرد . در این شکل فرض شده است که در هر مرحله 50 % از کلیه ی خطاهای وارد شده کشف می شود ، بدون اینکه خطای جدیدی وارد شود (یک فرض بهینه) . ده نقص طراحی مقدماتی پیش از شروع آزمون تا 94 خطا تشدید می شود 12 خطای نهفته وارد میدان می شود . در شکل 15-3 همین شرایط فرض می شود ، با این تفاوت که مرورهای طراحی و کدها به عنوان بخشی از هر مرحله ی توسعه انجام می شود . در این مورد ، ده خطای طراحی اولیه پیش از شروع آزمون تا 24 خطا تشدید می شود ، فقط سه خطای نهفته وجود دارد . با به خاطر آوردن هزینه های نسبی کشف و تصحیح خطاها ، هزینه کل (با مرور مثال فرضی ما و بدون مرور آن) را می توان تعیین کرد. تعداد خطاهای کشف شده طی هر یک از مراحل ذکر شده در شکل های 15-2 و 15-3 ، در هزینه لازم برای حذف یک خطا ضرب می شود (1/5 واحد برای طراحی ، 6/5 واحد پیش از آزمون و 15 واحد حین آزمون و 67 واحد پس از آزمون) . با استفاده از این داده ها هزینه کل توسعه و نگه داری در هنگام اجرای مرورها ، 783 واحد می شود. هنگامی که هیچ مروری صورت نپذیرد ، هزینه کل 2177 واحد می شود که تقریبا سه برابر هزینه بر می دارد .



شکل ۲-۱۵ تشدید نقایص- بدون مرور.



شکل ۳-۱۵ تشدید نقایص- مرورهای انجام شده.

مهندس نرم افزار برای اجرای مرورها باید انرژی و زمان صرف کند و سازمان توسعه باید پول خرج کند . ولی ، نتایج مثال قبلی ، دیگر کوچکترین تردیدی را بر جای نمی گذارد ، می توان حالا پرداخت کرد یا بعدا خیلی بیشتر پرداخت.

معیارهای مرور و کاربرد آن ها

مرورهای فنی از جمله چندین کنشی هستند که به عنوان بخشی از کار مهندسی نرم افزار خود به شمار می روند. هر کنش نیاز به تلاش انسانی دارد. چون منابع پروژه متناهی است ، مهم است که سازمان مهندسی نرم افزار با تعریف یک مجموعه مجموعه معیار (فصل 23) که در ارزیابی به کار می روند ، اثربخشی و موثر بودن هر کنش را بداند.

معیارهای مرور زیر را می توان برای هرکدام از مرورهای اجرا شده جمع آوری کرد :

*تلاش آماده سازی، E_p- تلاش و کار لازم (برحسب نفر-ساعت) برای مرور یک محصول کاری قبل از جلسه مرور واقعی.

* تلاش ارزیابی ، E_a- تلاش صرف شده (بر حسب نفر – ساعت) طی مرور واقعی.

* تلاش دوباره کاری ، E_r- تلاش اختصاص داده شده (برحسب نفر- ساعت) به تصحیح آن دسته از خطاهایی که طی مرور کشف می شوند.

* اندازه محصول کاری ، WPS- میزانی از اندازه محصول کاری که مرور شده است (مثلا تعداد مدل های UML، یا تعداد صفحات مستندات یا تعداد خطوط کد) .

* خطاهای جزئی یافته شده ، Err_{minor}- تعداد خطاهای یافته شده که می توان آنها را در زمره ی خطاهای جزئی دسته بندی کرد (تلاش لازم برای تصحیح آن ها از یک مقدار تعیین شده کوچک تر است) .

* خطاهای عمده یافته شده Err_{major}- تعداد خطاهای یافته شده که می توان ان ها را در زمره ی خطاهای عمده دسته بندی کرد (تلاش لازم برای تصحیح آن ها از یک مقدار تعیین شده، بزرگ تر است).

تحلیل معیارها

پیش از شروع تحلیل به چند محاسبه ساده نیاز است:

$$E_{review} = E_p + E_a + E_r$$

$$Err_{tot} = Err_{minor} + Err_{major}$$

چگالی خطا نشانگر خطاهای یافته شده به ازای واحد محصول کاری مرور شده است :

$$\text{چگالی خطا} = \frac{Err_{tot}}{WPS}$$

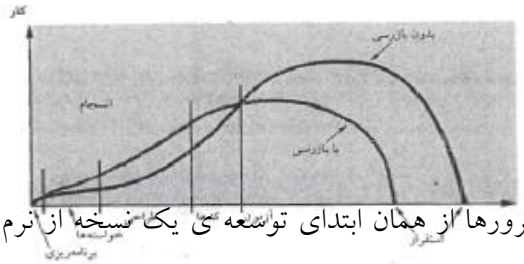
اثربخشی هزینه مرور ها

اندازه گیری اثربخشی هزینه ی هر مرور فنی به صورت همزمان ، دشوار است . یک سازمان مهندسی نرم افزار می تواند اثر بخشی مرور ها و عایدی حاصل از صرف هزینه در این بخش را تنها پس از پایان مرورها ، جمع آوری معیارها ، محاسبه ی داده های میانگین و سپس اندازه گیری کیفیت پایین دستی (ازطریق انجام آزمون) ارزیابی کند.

خطاهای مرتبط با خواسته ها که طی آزمون برملا می شوند ، برای یافتن و تصحیح به طور میانگین به 45 نفر –ساعت تلاش نیاز دارند (درباره ی شدت نسبی خطاها هیچ گونه داده ای در اختیار نداریم) . با به کارگیری میانگین های ذکرشده داریم :

$$\text{تلاش صرفه جویی شده به ازای هر خطا} = E_{testing} - E_{reviews}$$

45-6=30=(نفر ساعت به ازای هر خطا)



همانطور که در شکل می بینید ، تلاش صرف شده هنگام بکارگیری مرورها از همان ابتدای توسعه ی یک نسخه از نرم افزار افزایش می یابد ، ولی این سرمایه گذاری زود هنگام برای مرورها ده ها برابرد سود خواهد داشت ، زیرا تلاش های مورد نیاز برای آزمون و تصحیح را کاهش می دهد. علاوه بر آن ، تاریخ استقرار فرایندهایی که شامل مرور می شوند ، از تاریخ استقرار بدون مرور زودتر خواهد بود. مرورها زمانبر نیستند. بلکه باعث صرفه جویی در زمان می شوند.

مرورها : یک طیف رسمیت

مرورهای فنی را باید با سطحی از رسمیت به کار برد که با محصولی که قرار است ساخته شود ، خط زمانی پروژه و کسانی که کار را انجام می دهند ، تناسب داشته باشد. در شکل 5-15 یک مدل مرجع برای مرورهای فنی ارائه شده است که چهار خصوصیت سهیم در تعیین سطح رسمیت اجرای مرور را مشخص می کند.

رسمیت مرور هنگامی افزایش می یابد که (1) نقش های متمایزی به صراحت برای افراد تیم مرور تعریف می شود ، (2) مقدار کافی برنامه ریزی و آماده سازی برای مرور وجود داشته باشد ، (3) یک ساختار متمایز برای مرور (ازجمله وظایف و محصولات کاری درونی) تعریف شود و (4) هرگونه تصحیحاتی که قرار است انجام شود ، توسط افراد تیم مرور پیگیری شود.

ولی اگر واسط در موفقیت کل پروژه نقش محوری داشته باشد ، چطور ، اگر جان انسان ها به واسطی وابسته باشد که از نظر ارگونومی مناسب است، چطور ؟ ممکن است به این نتیجه برسید که به یک رویکرد شدیدتر نیاز دارید . یک تیم مرور تشکیل می شود . هر یک از اعضای تیم باید نقشی بر عهده بگیرند – رهبری تیم ، ثبت یافته ها ، ارائه مطالب و غیره . به هرکدام از افراد تیم مرور اجازه داده می شود به یک محصول کاری (که در این مورد ، نمونه اولیه ی واسط است) دستیابی داشته باشد و او زمانی را صرف یافتن خطاها ، ناسازگاری ها و جا افتادگی ها می کند.

مرور های فنی رسمی.در هر یک از این گروه های عمده ، چند رویکرد متفاوت می تواند انتخاب کرد.

مرورهای غیر رسمی

مرورهای غیر رسمی شامل بررسی ساده رو میزی (desk check) یک محصول کاری مهندسی نرم افزار با یکی از همکاران ، یک جلسه اتفاقی (با بیش از دو نفر) به هدف مرور یک محصول کاری یا جنبه های مرور گرایی در برنامه نویسی جفتی (فصل 3) می شود.

بررسی رومیزی ساده یا نشست اتفاقی با یک همکار ، یک مرور است . ولی از انجا که هیچ برنامه ریزی یا آماده سازی قبلی وجود ندارد ، دستور کار یا ساختاری برای نشست تنظیم نمی شود و برای خطاهای کشف شده ، هیچ پیگیری در کار نیست ، اثر بخشی اینگونه مرورها بطور چشمگیری کوچکتر از رویکردهای رسمی تر است. ولی یک برسی رومیزی ساده می تواند به کشف خطاهایی منجر شود که درغیر اینصورت ممکن است در فرایند نرم افزار منتشر گردد.

هرگونه خطا یا مشکل ذکر شده توسط افراد تیم مرور ، توسط طراح ثبت می شود تا بعدا برطرف گردد. بررسی های رومیزی را می توان به شیوه ای منظم زمان بندی کرد یا به عنوان بخشی از کار مهندسی نرم افزار خوب ، اجباری کرد . بطور کلی ، مقدار مطالبی که باید مرور شود ، نسبتا کوچک بوده زمان کل صرف شده در بررسی رومیزی از دو ساعت فراتر نمی رود.

برنامه نویسی جفتی را می توان یک بررسی رومیزی پیوسته دانست . برنامه نویسی جفتی به جای زمان بندی یک مرور در نقاط زمانی مشخص ، مرور پیوسته به موازات ایجاد محصول کاری (طراحی یا کدها) را ترغیب می کند. مزیت آن کشف بلافاصله ی خطاها و درنتیجه ، کیفیت بهتر محصول کاری است .

مرورهای فنی رسمی

مرورهای فنی رسمی (FTR) یکی از فعالیت های SQA است که مهندسان نرم افزار (و دیگران) انجام می دهند . اهداف FTR عبارتند از : (1) کشف خطاها در عملکرد ، منطق یا پیاده سازی هر نمایشی از نرم افزار ، (2) تصدیق اینکه تصدیق اینکه نرم افزار مورد مرور ، خواسته های خود را برآورده می سازد ، (3) حصول اطمینان از اینکه نرم افزار طبق استانداردهای از پیش تعیین شده ارائه شده است ، (4) رسیدن به نرم افز اری که به شیوه ای یکنواخت توسعه یافته است و (5) قابل اداره کردن پروژه ها . به علاوه ، FTR به عنوان یک پایه آموزشی عمل کرده مهندسان رده پایین را قادر به مشاهده روش های متفاوت برای تحلیل ، طراحی و پیاده سازی نرم افزار می سازد . FTR همچنین به ارتقای پیوستگی و پشتیبانی کمک می کند ، زیرا چند نفر با بخش هایی از نرم افزار آشنا می شوند که ممکن است در غیر اینصورت امکان دیدن آنها برایشان فراهم نشود.

FTRدر واقع طبقه ای از مرورهاست که شامل بررسی مقدماتی ، بازرسی ها ، مرورهای نوبت چرخشی و ارزیابی فنی دیگر نرم افزاری است. هر FTR به صورت یک ملاقات به اجرا در می آید و فقط درصورتی موفق خواهد بود که به طور مناسب برنامه ریزی ، کنترل و توجه شود.

نشست مرور

هر قالب FTR که انتخاب شود ، در کلیه نشست های مرور باید شرایط حدی زیر را رعایت کرد :

* بین سه تا پنج نفر (معمولا) باید در نشست حضور یابند
* آمادگی قبلی لازم است ، ولی نباید بیش از دو ساعت از وقت هر نفر را بگیرد
* مدت زمان جلسه باید کمتر از دو ساعت باشد.

با توجه به شرایط حدی فوق پیداست که :

- FTR بر بخش خاص (و کوچکی) از کل نرم افزار تاکید دارد.
- FTR با تمرکز بیشتر ، احتمال کشف خطا ها را افزایش می دهد.
- FTR محصول کاری را کانون توجه خود قرار می دهد.

کسانی که در نشست شرکت می کنند عبارتند از رهبر مرور ، همه ی مسوولان مرور و تولید کننده.یکی از مسوولان مرور،وظیفه ثبت موارد مهم را برعهده می گیرد.FTRباذکر دستور کار و معرفی مختصر تولید کننده آغاز می شود. سپس تولید کننده به تفصیل محصول کاری راتوضیح می دهد ومسوولان مرور نیز مسائلی را عنوان می کنند که از قبل آماده کرده اند.هنگامی که مشکلات وخطاهای معتبر کشف شد،مسوول ثبت آنها را ثبت می کند. در پایان، همه ی حاضران FTR باید تصمیم بگیرند که آیا(1) محصول کاری رابدون هررگونه اصلاح اضافی بپذیرند،(2) محصول را به خاطر خطاهای جدی رد کنند(پس از تصحیح به یک مرور دیگر نیاز است)،آیا(3)محصول را بپذیرند مشروط برآنکه خطاهای جزئی آن تصحیح شود (ولی دیگرنیازی به مرور نخواهد بود).پس ازاتخاذ تصمیم ،حضارFTR،برگه ای راامضا می کنند تا حضور خود را در نشست مرور خاطرنشان سازند.

دستورالعمل های مرور

1.مرور محصول،نه تولید کننده محصول ،هرFTR شامل مجموعه ای از افراد و برداشت های آنها است. اگر FTR درست اجراشود، باید پس از خاتمه ، احساسی گرم موفقیت را بر جای بگذارد.اگرFTR درست اجرا نشود،می تواند حال و هوای جلسه ی تفتیش عقاید رابه خود بگیرد.

2.تهیه یک دستور کار ورعایت آن.یکی از مشکلات همه ی نشست ها، انحراف است.FTR باید طبق زمان بندی انجام شود. رهبرمرور،مسوول تعیین زمان نشست بوده نباید به هنگام انحراف از موضوع ،از بر حذر داشتن افراد واهمه داشته باشد.

3. محدود کردن بحث و مشاجره. هنگامی که یکی از مسوولان مرور ، مسئله ای را مطرح می کند ، ممکن است همه با آن موافق نباشند . بجای صرف وقت برای مشاجره در باره ی آن ، باید مسئله را برای بحث بیشتر ثبت نمود.

4. بیان واضح بخش های مشکل دار و نه کوشش برای حل همه مشکلات ذکر شده . مرور ، یک جلسه حل مشکل نیست . حل مشکل غالبا به دست خود تولید کننده یا به کمک یک نفر دیگر صورت می پذیرد . حل مشکل را باید به بعد از نشست مرور موکول کرد.

5. یادداشت برداری. گاهی بد نیست که مسوول ثبت ، نکاتی را روی یک تابلوی دیواری یادداشت کند تا مسوولان دیگر مرور بتوانند توضیحات و اولویت بندی ها را ارزیابی کنند.

6. محدود کردن تعداد شرکت کنندگان و اصرار بر آمادگی قبلی . دو نفر از یک نفر بهتر است، ولی 14 نفر الزاما از 4 نفر بهتر نیست . تعداد افراد دخیل در نشست را در حداقل نگه دارید. ولی همه ی اعضای تیم مرور باید آمادگی قبلی داشته باشند.

7. تهیه چک لیستی برای هر محصولی که احتمال مرور آن می رود . این چک لیست به رهبر مرور کمک می کند تا نشست FTR سازمان دهی کند و به هریک از مسوولان مرور کمک می کند تا بر مسائل مهم تاکید کنند.

8. تخصیص منابع و زمان بندی برای FTRها . برای آنکه مرورها موثر واقع شوند ، باید آنها را به عنوان یکی از وظایف در اثنای فرایند مهندسی نرم افزار زمان بندی کرد. به علاوه ، زمان را باید برای اطلاعات اجتناب ناپذیر برنامه ریزی کرد که به عنوان نتیجه ای از FTR رخ می هد.

9. اجرای آموزش معنی د ار برای همه ی مسوولان مرور . همه ی مسوولان مرور ، برای بالا رفتن بازدهی باید آموزش رسمی ببینند .

10. مرور مرورهای قبلی . مرور خود فرایند مرور در کشف مشکلات مفید واقع می شود. نخستین محصولی که باید مرور شود ، ممکن است خود دستورالعمل ها باشد.

مرور های نمونه محور

تلین و همکاران یک فرایند مرور نمونه – محور را پیشنهاد می کنند که در آن ، نمونه هایی از محصولات کاری مهندسی نرم افزار وارسی می شوند تا تعیین شود که کدام محصولات کاری بیش از همه مستعد خطا هستند. سپس منابع FTR کامل فقط روی آن دسته از محصولات کاری متمرکز می شوند که احتمال بروز خطا در آنها بیشتر است (بر اساس داده های جمع آوری شده طی نمونه برداری) . در فرایند مرور نمونه محور برای اثربخشی باید تلاش به عمل آید که محصولات کاری هدف اولیه برای FTRهای کامل باشد . برای نیل به این مقصود ، مراحل زیر پیشنهاد می شود.

کسر a_i را برای هر کدام از محصولات کاری Iوارسی کنید. تعداد خطاهای f_i یافته شده در a_i را ثبت کنید.

تعداد خطاهای موجود در محصول کاری I را با ضرب کردن f_i در $1/a_i$ برآورده کنید .

محصولات کاری را به ترتیب نزولی برحسب برآورد حدودی تعداد خطاهای موجود در هر کدام از آنها مرتب کنید .

منابع در دسترس برای مرور را روی آن دسته از محصولات کاری متمرکز کنید که دارای بالا ترین تعداد خطاهای برآورده شده اند.

کسری از محصولات کاری که نمونه برداری شده اند باید نماینده کل محصولات کاری باشند و به قدر کافی بزرگ باشد تا برای افراد تیم مرور که از آن نمونه برداری می کنند ، معنا داشته باشند . با افزایش a_i ، احتمال اینکه نمونه ، نماینده ی معتبری از محصولات کاری باشد نیز افزایش می یابد . تیم مهندسی نرم افزار باید بهترین مقدار a_i را برای انواع محصولات کاری تولید شده تعیین کند.

فصل شانزدهم : تضمین کیفیت نرم افزار

تضمین کیفیت نرم افزار چه اهمیتی دارد؟از آن جهت اهمیت دارد که اگر یک تیم نرم افزاری در همه ی فعالیت های مهندسی نرم افزار بر کیفیت تاکید کند از مقدار دوباره کاری ها خواهد کاست و این منجر به کاهش هزینه ها و مهمتر از آن تسریع در زمان تحویل به بازار می شود.

پیش از آنکه بتوان فعالیت های تضمین کیفیت را آغاز کرد تعریف کیفیت نرم افزار در چند سطح متفاوت از انتزاع اهمیت دارد. هنگامی که دانستیبد کیفیت چیست تیم نرم افزار باید یک مجموعه فعالیت SQA را شناسایی کند که خطاها را پیش از تحویل محصولات کاری از آنها جدا کند.

طی تحلیل،طراحی،و تولید کد،محصول کاری اولیه SQA،یک گزارش خلاصه از بازبینی فنی رسمی است.در اثنای آزمون،روال ها،و طرح های آزمون تولید می شود. محصولات کاری دیگر مرتبط با بهسازی فرایند نیز مکن است تولید شوند.

هدف واحد مهندسی نرم افزار تولید نرم افزاری با کیفیت بالاست.

.کیفیت در واقع مفهومی چالش برانگیز است. تضمین کیفیت نرم افزار یک فعالیت چتری است.

تضمین کیفیت نرم افزار (SQA) شامل موارد زیر است:

1-یک فرایند SQA

2- وظایف خاص تضمین کیفیت و کنترل کیفیت

3- کار مهندسی نرم افزار اثر بخش

4- کنترل همه ی محصولات کاری نرم افزاری و تغییرات اعمال شده در آنها

5- رویه ای برای حصول اطمینان از مطابقت با استانداردهای توسعه نرم افزار

6- سازوکارهای اندازه گیری و گزارش دهی.

مسائل پس زمینه:کنترل کیفیت و تضمین کیفیت،فعالیت هایی ضروری برای هر شرکتی هستند که کار آن تولید محصولاتی برای استفاده توسط دیگران است.

نخستین وظیفه ی کنترل کیفیت و تضمین کیفیت در بل لبز در سال 1916 معرفی شد و به سرعت در سرتاسر جهان تولید شایع شد. طی دهه ی 1940، رویکرد های رسمی تری در قبال کنترل کیفیت پیشنهاد شد. این رویکرد ها بر اندازه گیری و بهبود مستمر فرایند تکیه دارند که عناصر کلیدی مدیریت کیفیت هستند.

سابقه ی تضمین کیفیت در توسعه ی نرم افزار،موازی سابقه کیفیت در تولید سخت افزار بود. طی سالهای اولیه علم کامپیوتر، کیفیت، مسوولیت اصلی برنامه نویس بود.

با گسترش دادن تعریفی که در بالا ارائه شد می توان گفت تضمین کیفیت یک الگوی برنامه ریزی شده و سیستماتیک از عملیات است که برای حصول اطمینان از کیفیت نرم افزار مورد نیاز هستند.

گروه SQA به عنوان نماینده خانگی مشتری عمل می کند. یعنی کسانی که SQA را انجام میدهند باید از دیدگاه مشتری به نرم افزار بنگرند.

عناصر تضمین کیفیت نرم افزار

تضمین کیفیت نرم افزار شامل گسترده وسیعی از دغدغه ها و فعالیت ها می شود که مدیریت کیفیت نرم افزار را کانون توجه قرار میدهد.آنها را می توان به شیوه ی زیر خلاصه کرد:

1- استانداردها: سازمان های IEEE و ISO و سایر سازمان ها،آرایه وسیعی از استانداردهای مهندسی نرم افزار و مستندات وابسته به آن را تدوین کرده اند.استانداردها را ممکن است سازمان مهندسی نرم افزار،داوطلبانه بپذیرد یا مشتری یا یک طرف ذینفع دیگر الزامی کند. وظیفه ی SQA حصول اطمینان از رعایت استانداردهای پذیرفته شده و مطابقت تمامی محصولات کاری با این استانداردهاست.

2- مرورها ومیزی ها :مرورهای فنی یک نوع فعالیت کنترل کیفیت هستند که توسط مهندسان نرم افزار برای مهندسی نرم افزار اجرا می شوند هدف از انجام آنها کشف خطاهاست.میزی ها نوعی از مروری هستند که توسط پرسنل SQA وبه قصد حصول اطمینان از رعایت دستورالعمل های کیفیتی برای کار مهندسی نرم افزار دنبال می شوند.

3- آزمون:آزمون/ نرم افزار یکی از وظایف کنترل کیفیت است که یک هدف اصلی را دنبال می کند. یافتن خطاها وظیفه ی SQA حصول اطمینان از طرح ریزی درست ومناسب آزمون ها و اجرای اثربخشی آنهاست به طوری که احتمال دستیابی به هدف اصلی آن به حداکثر برسد.

4- جمع آوری و تحلیل خطاها/نقایص:تنها راه بهبود بخشیدن،سنجیدن عملکرد است SQA داده های مربوط به خطاها و نقایص را جمع آوری و تحلیل می کند تا بهتر معلوم شود که خطاها چگونه وارد می شوند و کدام فعالیت های مهندسی نرم افزار برای حذف آنها از همه مناسب ترند.

5- مدیریت و تغییرات: تغییر، یکی از مخرب ترین جنبه های هر پروژه نرم افزار است و اگر خوب مدیریت نشود می تواند به سردرگمی منجر شود و سردرگمی همیشه به کیفیت ضعیف می انجامد.با SQA می توان اطمینان حاصل کرد که اقدامات مناسبی برای مدیریت نهادینه شده است.

6- آموزش :هر سازمان نرم افزاری مایل است کارهای مهندسی نرم افزار خود را بهبود بخشد. یک عامل کلیدی سهیم در این بهبود بخشی،آموزش مهندسان نرم افزار، مدیران آنها و سایر طرف های ذی نفع است.سازمان SQA در بهبود فرایند نرم افزار جلو دار است ودر حمایت از برنامه های آموزشی نقش کلیدی دارد.

7- مدیریت منابع خرید: سه گروه از نرم افزارها از منابع خارجی تامین نرم افزار خریداری می شوند 1- پکیج های بسته بندی شده(Microsoft office)2- قطعات نیمه آماده که ساختار اسکلتی پایه را فراهم می آورند و می توان آن را مطابق با میل و سلیقه ی خریدار تکمیل کرد و نرم افزارهای قراردادی که از روی مشخصات ارائه شده توسط سازمان مشتری،طراحی و ساخته می شود.وظیفه ی سازمان SQAاین است که اطمینان حاصل کند با پیشنهاد اقدامات کیفیتی خاص که منبع خرید بایدرعایت کند. نرم افزار با کیفیت بالا نتیجه می شود . الزامات کیفیتی خاصی را در هر قرارداد منعقد شده با منبع خرید یگنجاند.

8- مدیریت امنیت: با افزایش جرم های سایبری و مقررات دولتی در حیطه ی حفظ حریم خصوصی، هر سازمان نرم افزار باید خط مشی ها و سیاست هایی را نهادینه سازد که داده ها را در تمامی سطوح محافظت کنند، محافظت فایروالی برای برنامه های تحت وب فراهم کند و اطمینان حاصل کند که نرم افزار از درون دست کاری نشده باشد. با SQA می توان مطمئن شد که فرایندو فناوری مناسب در دست یابی کیفیت نرم افزار به کار رفته است.

9- ایمنی : از آنجا که نرم افزارها تقریبا همیشه یک جزء محوری در سیستم های در خدمت انسان هستند، تاثیر نقایص پنهان می تواند مصیبت بارباشد. SQA ممکن است مسئول ارزیابی تاثیر شکست نرم افزار و شروع مراحل لازم برای کاهش ریسک باشد.

10- مدیریت ریسک: گرچه تحلیل ریسک و کاستن از میزان آن دغدغه مهندسان نرم افزار است، سازمان SQA است که باید اطمینان حاصل کند فعالیت های مدیریت ریسک به طور مناسب اجرا می شوندو طرح های مربوط به احتمال بروز ریسک تدوین شده اند.

علاوه بر هر کدام از این دغدغه ها و فعالیت ها، SQA تلاش می کند تا اطمینان حاصل کند که فعالیت های پشتیبانی نرم افزار انجام شده اند، در حالی که دغدغه اصلی در انجام آن ها، کیفیت بوده است.

وظایف، اهداف و معیار های SQA

1- تضمین کیفیت نرم افزار از چند وظیفه مرتبط با دو گروه متفاوت تشکیل می شوند که عبارتند از:

مهندسان نرم افزار که کارهای فنی را انجام می دهند.

2- یک گروه SQA که مسوولیت برنامه ریزی برای تضمین کیفیت،نظارت،ثبت وقایع،تحلیل و گزارش دهی بر عهده آنان است.

مهندسان نرم افزار با اعمال روشهای فنی و موازین منسجم،اجرای بازبینی های فنی رسمی و اجرای آزمون های نرم افزاری برنامه ریزی شده،کیفیت را کنترل می کنند.

وظیفه گروه SQA : کمک به تیم نرم افزاری، جهت دستیابی به یک محصول نهایی با کیفیت بالاست.

موسسه مهندسی نرم افزار،مجموعه ای از کنش های SQA را توصیه می کند که برنامه ریزی تضمین کیفیت،نظارت، ثبت وقایع، و گزارش دهی را مشخص می کندو این کنش ها توسط یک گروه SQA مستقل اجرا می شوند.

نقش های گروه SQA:

1- تهیه یک طرح SQA برای پروژه: این طرح طی برنامه ریزی پروژه توسعه می یابد و توسط همه ی طرفهای علاقه مند مورد بازبینی قرار می گیرد. فعالیت های تضمین کیفیت که توسط تیم مهندسی نرم افزار و گروهSQA اجرا می شوند ،از این طرح دستور می گیرند. در این طرح موارد زیر مشخص می شود:

ارزیابی هایی که باید انجام شوند، بازرسی ها و بازبینی هایی که باید اجرا شوند، استانداردهایی که در پروژه لازم الاجرا هستند، روال هایی برای گزارش و پیگیری خطا، مستنداتی که باید توسط گروه SQA تولید شود، مقدار بازخوردی که برای تیم پروژه نرم افزاری فراهم می آید.

2- شرکت در توسعه توصیف فرایند نرم افزاری پروژه: تیبم نرم افزاری فرایندی برای انجام کار انتخاب می کند. گروه SQA توصیف فر ایند را برای مطابقت با سیاست سازمانی، استانداردهای داخلی، استانداردهای تحمیل شده از خارج سازمان و بخش های دیگر برنامه پروژه نرم افزار، مورد بازبینی قرار می دهد.

3- بازبینی فعلیت های مهندسی نرم افزار برای وارسی مطابقت با فرایند نرم افزاری مشخص: گروه SQA انحرافات از فرایند را شناسایی،مستند سازی و پیگیری کرده انجام تصحیحات را مورد وارسی قرار می دهد.

4- بازرسی محصولات کاری برای وارسی مطابقت با محصولات تعیین شده به عنوان بخشی از فرایند نرم افزار: همانند مرحله قبل است فقط SQA نتایج کاری خود را به مدیر پروژه گزارش می کند.

5- حصول اطمینان از مستند سازی انحرافات در کار نرم افزار و محصولات کاری و مقابله با آنها بر اساس یک رویه مستند سازی شده: ممکن است انحرافاتی در برنامه پروژه ، توصیف فرایند ، استانداردهای لازم الاجرا یا محصولات کاری به چشم بخورد.

6- ثبت هر گونه عدم مطابقت و گزارش به مدیریت ارشد: موارد عدم مطابقت آنقدر پیگیری می شوند تا برطرف شوند. علاوه بر این فعالیت ها، گروه SQA کنترل و مدیریت تغییر را هماهنگ کرده و به جمع آوری و تحلیل معیارهای نرم افزاری کمک می کند.

اهداف، صفات و معیارها:

کنش های SQA برای دستیابی به اهداف عملی زیر اجرا می شوند:

1- کیفیت خواسته ها:صحت،کامل بودن و سازگاری مدل خواسته ها تاثیری قوی بر کیفیت همه ی محصولات کاری بعدی خواهد گذاشت. SQA باید مطمئن شود که تیم نرم افزاری به طور مناسب مدل خواسته ها را مرور کرده است تا به سطح بالایی از کیفیت دست پیدا کند.

2- کیفیت طراحی :هر عنصر از مدل طراحی باید توسط تیم نرم افزاری ارزیابی شود تا اطمینان حاصل گردد که کیفیت بالایی از خود نشان دهد و خود طراحی با خواسته ها مطابقت دارد. SQA به دنبال صفاتی از طراحی است که نشانگر کیفیت هستند.

3- کیفیت کد ها:کد منبع و محصولات کاری مرتبط باآن بایدبا استانداردهای محلی کدنویسی مطابقت داشته باشدو خصوصیاتی از خود به نمایش بگذارند که قابلیت نگهداری را بهبود بخشند SQA باید این صفات را که تحلیل منطقی کیفیت کدها را میسر می سازند جدا کند.

4- اثر بخشی کنترل کیفیت: تیم نرم افزاری باید منابع محدود را به گونه ای به کار گیرد که احتمال دستیابی به نتیجه ای با کیفیت بالاتری، بسیار زیاد باشد. SQA، تخصیص منابع به مرورها و آزمون ها را تحلیل می کند تا ارزیابی کند که آیا به بهترین نحو ممکن تخصیص داده شده اند یا خیر.

رویکردهای رسمی در SQA

در بخش قبل استدلال کردیم که کیفیت نرم افزار وظیفه ی همه ی افراد است و از طریق کار مهندسی نرم افزار رقابتی و نیز از طریق به کارگیری بازبینی های فنی، راهبرد آزمون چند لایه ، کنترل بهتر محصولات کاری نرم افزاری و تغییرات به عمل آمده در آنها و سرانجام به کارگیری استانداردهای پذیرفته شده در مهندسی نرم افزار قابل حصول است . به علاوه کیفیت را میتوان بر حسب انواع صفات کیفیتی تعریف کرد و با استفاده از انواع شاخص ها و معیارها سنجید .

طی سه دهه گذشته، بخشی کوچک ولی تاثیر گذاری از جامعه مهندسی نرم افزار استدلال کرده است که رویکرد رسمی تری برای تضمین کیفیت نرم افزار مورد نیاز است . میتوان استدلال کرد که یک برنامه کامپیوتری ، شی ءای ریاضی است .

تلاشهای به عمل آمده برای تصحیح برنامه ها ، جدید نیستند دیکسترا و لینگر ، میلز و ویت و بسیاری دیگر از اثبات درستی برنامه ها پشتیبانی کرده اند و آن را به مفاهیم برنامه نویسی ساخت یافته ربط داده اند .

تضمین کیفیت آماری نرم افزار

تضمین کیفیت آماری نرم افزار رفته رفته در سر تا سر صنعت گسترش می یابد ، تا کیفیت ، ویژگی کمی بیشتری بگیرد که تضمین کیفیت آماری شامل مراحل زیر می باشد:

اطلاعات مربوط به نقایص نرم افزار جمع آوری و گروه بندی می شود.

کوشش می شود رد هر نقص تا علت اصلی آن پیگیری شود.

با استفاده از اصل پارتو (ریشه 80% نقایص را می توان در 20% از همه علل ممکن یافت) آن 20% علل جدا می شود.

هنگامی که چند علت حیاتی شناسایی شدند ، حرکت برای تصحیح مشکلاتی که باعث این نقایص شده اند ، آغاز می شود.

این مفهوم نسبتاً ساده، گام مهمی در راستای ایجاد یک فرآیند نرم افزار تطبیقی است که درآن تغییراتی صورت می پذیرد تا عناصری از فرآیند را بهبود بخشد که تولید خطا می کند .

شش سیگما برای مهندسی نرم افزار

شش سیگما پرکاربرترین راهبرد برای تضمین کیفیت آماری است که توسط موتورولا در دهه 1980 به عموم شناسانده شد . « یک روش شناسی شدید ومنضبط است که از داده ها وتحلیل آماری برای اندازه گیری عملکرد شرکت وبهبود بخشیدن به آن از طریق شناسایی و حذف نقایص در فرآیندهای تولیدی و خدماتی بهره می برد ».

اصطلاح شش سیگما از شش برابر مقدار « انحراف معیار » به دست آمده است که معادل با 3/4 نمونه به ازای هر یک میلیون نمونه است – و این حد اعلای استانداردهای کیفیتی است در روش شناسی شش سیگما سه مرحله اصلی وجود دارد :

تعریف خواسته های مشتری، محصولات قابل تحویل واهداف پروژه ازطریق روشهای کاملاً مشخص برای برقراری ارتباط با مشتری .

اندازه گیری فرآیند موجود و خروجی آن برای تعیین کیفیت فعلی (جمع آوری معیارهای نقص).

تحلیل معیارهای نقص و تعیین چند علت حیاتی .

اگر یک فرآیند نرم افزار موجود ، در جای خود باشد ، ولی به بهبود نیازداشته باشد ، شش سیگما دو مرحله اضافی برای آن پیشنهاد میکند:

بهبود بخشیدن به فرآیند با حذف علل ریشه ای نقایص .

کنترل فرآیند برای حصول اطمینان از این که کارهای آینده باعث ورود دوباره علل این نقایص نخواهد شد.

این مراحل اصلی و اضافی را گاهی روش DMAIC می نامند .

اگر سازمانی در حال توسعه یک فرآیند نرم افزار باشد مراحل اصلی به صورت زیر بسط پیدا میکند :

طراحی فرآیند برای 1- پرهیز از علل ریشه ای نقایص و 2- برآورده ساختن خواسته های مشتریان.

وارسی اینکه مدل فرآیندی واقعاً از نقایص پرهیز میکند وخواسته های مشتری را برآورده می سازد این شکل اصلاح شده را گاهی روش DMADV می نامند .

قابلیت اطمینان نرم افزار

قابلیت اطمینان یک برنامه کامپیوتری ، عنصر مهمی از کیفیت کلی آن به شما می رود . اگر برنامه ای به کرات از اجرا باز بماند ، عوامل کیفیتی دیگر نرم افزار، اهمیت خود را از دست خواهد داد .

قابلیت اطمینان بر خلاف عوامل کیفیتی دیگر ، با استفاده از داده های تاریخی و توسعه ای ، مستقیما ض قابل برآورد و اندازه گیری است.قابلیت اطمینان به زبان آماری، به عنوان«احتمال عملکرد بدون شکست یک برنامه کامپیوتری در محیطی مشخص برای یک زمان معیین » تعریف می شود .

برای مثال ، برآورد می شود که برنامه X در عرض 8 ساعت پردازش دارای قابلیت اطمینان 0/999 است به عبارت دیگر اگر قرار باشد در عرض 8 ساعت پردازش این برنامه 1000 بار اجرا شود ، احتمالاً 999 بار از 1000 بار را درست کار میکند .

در حیطه بحث کیفیت نرم افزار و قابلیت اطمینان ، شکست عبارت از عدم همخوانی با خواسته های نرم افزار است . و با این وجود حتی در همین حیطه نیز درجات مختلفی از شکست وجود دارد. شکست ها ممکن است ناراحت کننده یا فاجعه بار باشند . یک شکست را شاید در عرض چند ثانیه بتوان تصحیح کرد ، حال آنکه تصحیح شکست دیگر شاید به هفته ها یا

حتی ماه ها زمان نیاز داشته باشد. اگر بخواهیم مسئله را پیچیده تر کنیم تصحیح یک شکست ممکن است منجر به وارد شدن خطاهای دیگر شود که آنها نیز منجر به شکست های دیگر میشوند .

موازین مرتبط با قابلیت اطمینان و دسترس پذیری

اکثر مدل های قابلیت اطمینان مرتبط با سخت افزار مبتنی بر شکست های ناشی از فرسودگی هستند نه نقایص طراحی. در سخت افزار احتمال شکست های ناشی از فرسودگی فیزیکی نسبت به شکست های طراحی بیشتر است. متاسفانه در مورد نرم افزارعکس این مورد صحت دارد در واقع ریشه همه ی شکست های نرم افزاری را می توان در مشکلات طراحی یا پیاده سازی جستجو کرد، فرسایش در اینجا نقشی ندارد.

اگر یک سیستم کامپیوتری را در نظر بگیریم، میزان ساده ای از قابلیت اطمینان، زمان میانگین بین شکست(MTBF) است که در آن:

$$MTBF=MTTF+MTTR$$

که MTTFوMTTR به ترتیب زمان میانگین شکست و زمان میانگین ترمیم هستند.

بسیاری از پژوهشگران معتقدند MTBF به مراتب مفید تر از سایر معیار های مرتبط با کیفیت نرم افزارند به بیان ساده تر، کاربر نهایی با شکست ها سرو کار دارد نه با تعداد کل خطاها. از آنجا که همه ی خطاهای موجود در یک برنامه، نسبت شکست یکسانی ندارند، تعداد کل خطاها، شاخص ضعیفی از قابلیت اطمینان سیستم را به دست می دهد.

ولی MTBF به دو دلیل می تواند مشکل آفرین باشد:

1- یک بازه زمانی زا میان دو شکست تصویر می کند ولی میزانی از شکست به دست نمی دهد.

2- MTBF را ممکن است به غلط به عنوان بازه زمانی میانگین تعبیر کنند، هر چند که معنای آن این نیست.

یک میزان دیگر برای قابلیت اطمینان، شکست در زمان (FIT) است- میزان آماری از تعداد شکست هایی است که یک مولفه طی یک میلیارد ساعت کار از خود ممکن است نشان دهد. بنابراین، یک FIT معادل با یک شکست در هر یک میلیارد ساعت عملکرد است.

دسترس پذیری نرمافزار عبارت است از احتمال کار کردن برنامه طبق خواسته ها در یک نقطه ی مفروض از زمان،که به صورت زیر تعریف می شود:

$$\text{دسترس پذیری} = \frac{MTTF}{MTTF+MTTR} \times 100\%$$

میزان قابلیت اطمینان MTBF به یک اندازه نسبت به MTTF و MTTR حساس است. میزان دسترس پذیری تا حدی نسبت به MTTR که میزان غیر مستقیم از قابلیت نگهداری نرم افزار است حساس است.

ایمنی نرم افزار (software safety)

ایمنی نرم افزار یکی از فعالیت های تضمین کیفیت است که بر شناسایی و سنجش ریسک های باالقوه ای تاکید دارد که ممکن است تاثیری منفی بر نرم افزار داشته منجر به شکست کل سیستم شود. اگر بتوان ریسک ها را در همان ابتدای فرایند نرم افزار شناسایی کرد، می توان ویژگی هایی را در طراحی نرم افزار مشخص کرد که ریسک های باالقوه را حذف یا کنترل کند.

به عنوان بخشی از ایمنی نرم افزار، یک فرایند مدل سازی و تحلیل به اجرا در می آید. در آغاز ریسک ها شناسایی و بر اساس اهمیت و احتمال وقوعی که دارند گروه بندی می شوند برای مثال برخی ریسک های مربوط به کنترل گشت کامپیوتری با

یک خودرو عبارتند از: 1- باعث شتاب کنترل نشده ای می شود کهغیر قابل توقف است. 2- با فشار دادن پدال ترمز پاسخ نمی دهد. 3- وقتی سوئیچ فعال می شود سیستم کار نمی کند.4- سرعت به آهستگی زیاد و کم می شود.

هنگامیکه ریسک های سیستمی شناسایی شدند، برای تعیین شدت و احتمال وقوع هر یک، از تکنیک های تحلیل استفاده می شود. برای آنکه نرم افزار موثر واقع شود، باید در کل سیستم تحلیل شود.

هنگامیکه ریسک ها شناسایی و تحلیل شد، خواسته های مرتبط با ایمنی را می توان برای نرم افزار مشخص کرد. یعنی مشخصه می تواند حاوی لیستی از رویدادهای نامطلوب و پاسخ های سیستمی مطلوب به این رویدادها باشد. سپس نقش نرم افزار در مدیریت رویدادهای نامطلوب مشخص می شود.

قابلیت اطمینان نرم افزار و ایمنی نرم افزار ارتباط تنگاتنگی با هم دارند. در قابلیت اطمینان نرم افزار، از تحلیل آماری برای تعیین احتمال وقوع شکست نرم افزار استفاده می شود.به هر حال وقوع یک شکست الزاما به ریسک با اتفاق سوء نمی انجامد. یعنی شکست ها در خلاء در نظر گرفته نمی شوند،بلکه در کلیت سیستم کامپیوتری ارزیابی می شوند.

استانداردهای کیفیتی ISO 9001

سیستم تضمین کیفیت را می توان به عنوان ساختار سازمانی، مسئولیت ها، روال ها، فرایند ها و منابع پیاده سازی مدیریت کیفیت تعریف کرد. سیستم های تضمین کیفیت به این منظور ایجاد می شوند که به سازمان کمک کنند تا اطمینان حاصل کنند رضایت مشتریان حاصل شده است و آنچه مشخص کرده اند،برآورده شده اند. این سیستم ها گستره ی وسیعی از فعالیت ها را پوشش می دهند که کل چرخه حیات محصول را از طرح ریزی، کنترل، اندازه گیری، آزمون و گزارش دهی و بهبود سطوح کیفیت در سرتاسر فرایند توسعه و تولید در بر می گیرد. ISO 9001 عناصر کیفیتی را به زبان کلی توصیف می کند که برای هر تجارتی با هر تجارتی با هر محصول یا خدماتی که ارائه می دهد،قابل اعمال هستند.

برای اینکه نام شرکتی در یکی از مدل های تضمین کیفیت ISO 9000 ثبت گردد،سیستم و عملیات های کیفیت شرکت باید توسط یک شرکت مجاز ممیزی شود تا پیروی آن از این استاندارد به تایید برسد. پس از ثبت موفق، یک گواهی از سوی شرکت ممیزی برای این شرکت صادر می شود.هر شش ماه یک بار شرکت دوباره مورد ممیزی قرار می گیرد تا از پیروی مستمر از استاندارد اطمینان حاصل شود.

الزامات ترسیم شده توسط ISO 9000:2000 به مباحثی همچون مسئولیت پذیری مدیران، سیستم کیفیت، بازبینی قراردادها، کنترل طراحی، کنترل داده ها و مستندات، اقدامات تصحیحی و پیش گیرانه، کنترل سوابق کیفیتی، ممیزی های کیفیتی درونی ، آموزش، خدمات رسانی و تکنیک های آماری می پردازد. برای آن که یک شرکت نرم افزاری موفق به اخذ گواهینامه ISO 9000:2000 شود، باید برای پرداختن به هر کدام از الزامات ذکر شده در بالا،خط مشی ها و روال هایی را تعیین کند وسپس بتواند نشان دهد که این خط مشی ها و روال ها دنبال می شوند.

طرحSQA

طرحSQAراهنمایی برای نهادینه کردن تضمین کیفیت نرم افزار فراهم می آورد.این طرح که توسط گروه SQA تهیه می شود، به عنوان الگویی برای فعالیت های SQA عمل می کند که برای هر پروژه نرم افزاری نهادینه می شوند.

IEEE استانداردی برای طرح های SQA پیشنهاد کرده است. این استاندارد ساختاری را پیشنهاد میکند که در آن موارد زیر باید مشخص گردد: 1- هدف و دامنه کاربرد طرح. 2- توصیفی از همه ی محصولات کاری مهندسی نرم افزار.3- همه ی استانداردهای قابل استفاده در طول فرایند نرم افزار. 4- وظایف و کنش های SQA و محل قرار گرفتن آنها در سرتاسر فرایند نرم افزار. 5 - ابزارها و روشهایی که از وظایف و کنش های SQA پشتیبانی می کنند. 6- رویه های مدیریت پیکربندی نرم

افزار. 7- روش های مونتاژ، ایمن سازی و نگهداری کلیه سوابق با SQA. 8- نقش ها و مسئولیت های سازمانی در قبال کیفیت محصول شرح داده شده است.

فصل 17 : راهبرد های آزمون نرم افزار

نرم افزار ،مورد آزمون قرار میگیرد تا خطاهایی که سهوا در طراحی و پیاده سازی وارد شدند بر ملا شوند

مدیر پروژه توسعه ، مهندسان نرم افزار ، متخصصان آزمون توسعه

ویژگی های راهبردهای آزمون نرم افزار:

1.اجرای بازبینی های فنی اثربخش که باعث می شود بسیاری از خطاها از آغاز آزمون حذف شود

2.آزمون در سطح مولفه ها شروع میشود و رفته رفته کل سیستم را در بر میگیرد

3.در رویکردهای متفاوت مهندسی نرم افزار و در زمان های متفاوت باید از تکنیک های آزمون متفاوتی بهره برد

4.آزمون توسط سازنده نرم افزار وبرای پروژه های بزرگتر توسط یک گروه آزمون گر مستقل انجام میشود

5. آزمون و اشکال زدایی دو فعالیت جدا از هم اند اما اشکال زدایی باید در هر راهبرد آزمون جایی داشته باشد

سطوح راهبرد آزمون:

درسطح پایین: وارسی و آزمون بخش کوچکی از آزمون در سطح بالا:عملکرد اصلی سیستم در مقابل خواسته های مشتری اعتبار می بخشد آزمون نرم افزار یکی از عناصر وارسی و اعتبار سنجی است

وارسی و اعتبار سنجی(verification & validation):

وارسی عبارت است از مجموعه فعالیت هایی که پیاده سازی صحیح یک عملکرد خاص توسط نرم افزار را تضمین می کند اعتبار سنجی مجموعه ای از فعالیت هاست که تضمین می کندنرم افزارساخته شده با خواسته های مشتری مطابقت دارد

تعریف بوهم :

وارسی : آیا محصول را درست ساخته ایم

اعتبارسنجی:آیامحصول درست را ساخته ایم

وارسی واعتبار سنجی (V&V):

بسیاری از فعالیت ها را در بر میگیرد که از آنها به عنوان تضمین کیفیت نرم افزار SQA یاد کنیم شامل :

1.بازبینی های فنی رسمی

2.ممیزی های کیفیتی و پیکربندی

3.نظارت بر کارایی

4.شبیه سازی

5.آزمون توسعه

6.مطالعه ی امکان سنجی

7.بازبینی مستندات

و ...

سازنده نرم افزار. که این خود زیانبار است

با اینکه سازنده نرم افزار بهتر از هر کسی به نرم افزار شناخت دارد اما همه انها علاقه زیادی دارند که نشان دهند که نرم افزار عاری از هر گونه خطاست

ITG

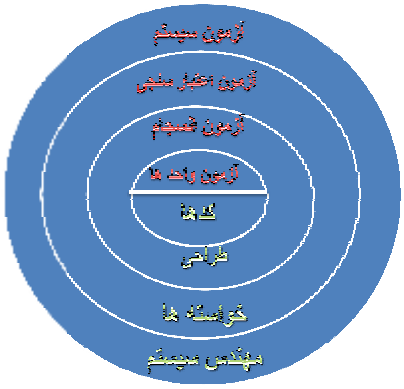
1.نقش آن برطرف کردن مشکل ذاتی است که در واگذاری آزمون به سازنده وجود دارد

2.اختلاف سلیقه ها را برطرف می کند

3.سازنده و ITGدر سراسر پروژه نرم افزاری رابطه کاری تنگاتنگی دارند

4. در حالی که آزمون انجام میشود سازنده باید در دسترس باشد تا خطاهای پیدا شده را برطرف سازد

مارپیچ فرایند مهندسی نرم افزار



1.مهندسی سیستم نقش نرم افزار را تعیین می کند

2.خواسته های نرم افزار تحلیل میشود

3.طراحی می کنیم

4.و در آخر کد نویسی

به طرف داخل مارپیچ حرکت می کنیم تا در هر دور از سطح انتزاع کاسته شود .دامنه آزمون در هر دور حرکت به بیرون مارپیچ وسعت پیدا می کند.

راهبرد های آزمون نرم افزار :

1.آزمون واحد ها

2.آزمون انسجام

3.آزمون اعتبار سنجی

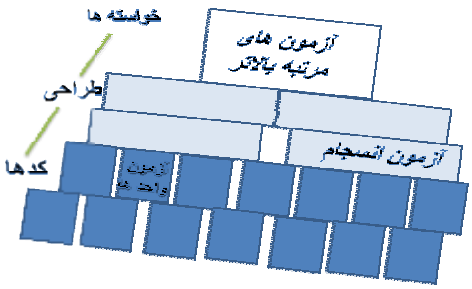
4. آزمون سیستم

آزمون واحد ها : در مرکز مارپیچ آغاز میشود .بر هر واحد یا مولفه ی نرم افزار که در کد منبع پیاده سازی می شود تمرکز دارد

آزمون انسجام: به طراحی و ایجاد معماری نرم افزار تاکید دارد

آزمون اعتبار سنجی: تحلیل مطابقت خواسته های نرم افزار با نرم افزار ساخته شده

آزمون سیستم: نرم افزار و دیگر عناصر سیستمی به صورت یک کل آزمون میشوند



1.**آزمون واحد ها** : در آغاز آزمون ها بر تک تک مولفه ها تاکید دارند تا اطمینان حاصل شود که هر یک به طور منفرد درست عمل می کند.استفاده زیاد از تکنیک های جعبه سفید

2.**آزمون انسجام** : باید مولفه ها را مونتاژ و مجتمع کرد تا بسته نرم افزاری کامل تشکیل شود استفاده زیاد از تکنیک های جعبه سیاه و استفاده کم از تکنیک جعبه سفید

3.**آزمون اعتبار سنجی** : برای حصول اطمینان نهایی از رعایت همه ی خواسته های رفتاری عملیاتی و کارایی

4. **آزمون سیستم** : هنگامی که نرم افزار منسجم شد باید با سایر عناصر سیستمی دیگر تلفیق شود

ملاک هایی برای کامل کردن آزمون.

آزمون هیچ گاه تمام نمی شود فقط بار مسئولیت از شما به مشتری محول می شود

از طرفی آزمون وقتی پایان می یابد که زمان و پول شما تمام شود

نظریه قابلیت اطمینان برای پیش بینی کامل بودن آزمون

اتاق تمیز و تکنیک های کاربرد آماری پیشنهاد می شود

آیا راهبرد آزمون مناسب برای موفقیت کافی است ؟

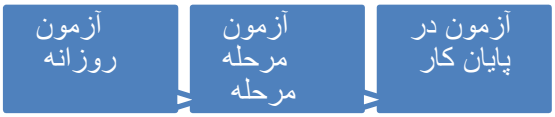
حتی اگر راهبرد مناسبی برای آزمون نرم افزار در دسترس باشد اگر یک سری مسائل جانبی رعایت نشود حتی بهترین راهبردها نیز محکوم به شکست خواهد بود

نرم افزار

سنتی

شی گرا و تحت وب

راهبرد های آزمون برای نرم افزار های سنتی



پر کاربرد ترین آزمون آزمون مرحله به مرحله است آزمون روزانه خسته کننده و آزمون پایان کار احتمال دوباره کاری و خطای بیشتر رو داره

آزمون واحد ها در نرم افزار سنتی

آزمون واحد ها تلاش های وارسی کوچکترین واحد طراحی نرم افزار(مولفه و پیمانه) راکانون توجه قرار میدهد با بکار گیری طراحی در سطح مولفه ها به عنوان راهنما مسیر های کنترلی مهم آزمون می شوند تا خطا های موجود در مرز پیمانه بر ملا شوند پیچیدگی نسبی این آزمون ها و خطا ها یی که این آزمون ها بر ملا می سازند توسط قیود بندی محدود می شوند این نوع آزمون را می توان به طور موازی برای چند مولفه انجام داد

آزمون واحد ها معمولا به عنوان الحاقی برای مرحله کد نویسی در نظر گرفته می شود طراحی آزمون واحدها ممکن است قبل از شروع کد نویسی یا بعد از تولید کد های منبع رخ دهد آزمون واحد ها وقتی ساده می شود که مولفه ای با انسجام بالا طراحی شود وقتی که فقط یک عملکرد بر عهده مولفه ای قرار داده می شود تعداد موارد آزمون کاهش می یابد و کشف خطا راحت تر میشود.

1.**آزمودن واسط پیمانه ها**

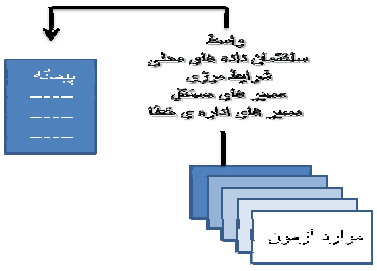
جریان داده هایی که از واسط یک پیمانه عبور میکند باید پیش از شروع هر آزمون دیگری آزموده شود 2.**بررسی ساختمان داده های محلی** حصول اطمینان از مناسب عمل کردن پیمانه ها در مرز ها و ارزیابی تاثیر محلی بر داده های سراسری آزمون واحد ها

3.**مسیر های مستقل**

مسیر ها امتحان می شوند تا اطمینان حاصل شودکه همه ی دستورات موجود در یک پیمانه حداقل یک بار اجرا شده است

4.آزمون تمام مسیر های کنترل خطا

آزمون مرز ها مهمترین و آخرین مرحله است،به خاطر اینکه شکست ها معمولا در مرز ها رخ می دهدخطا ها غالبا زمانی رخ می دهند که nامین عنصر از یک آرایه nبعدی پردازش می شود



شدت شکست به عنوان تابعی از زمان اجرا

ضد اشکال زدایی

طراحی خوب پیش بینی می کند که شرایط خطا پیش بینی شود و مسیر هایی برای اداره ی خطا مشخص شود که در صورت بروز خطا پردازش را دوباره جهت دهی کند یا به آن پایان دهد . یوردون این روش را ضد اشکال زدایی می نامد

ذکر برخی خطاهای متداول محاسباتی

1.تقدم محاسباتی نادرست

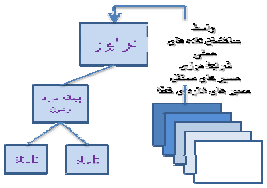
2.گوناگونی عملیات

3. مقدار دهی اولیه نادرست

4. دقت نادرست

5.ارائه نماد های نادرست برای یک رابطه یا عبارت

دلیل وجود درایور و stubدر آزمون واحدها



محیط آزمون واحد

چون مولفه ی یک برنامه مستقل و قائم به ذات نیست برای آزمون هر واحد باید یک نرم افزار درایور و یا stubتوسعه یابد در اکثر کاربرد ها درایور همان برنامه اصلی است که داده های مورد آزمون را پذیرفته این داده ها را به مولفه ای که باید آزمون شود تحویل داده نتایج مربوطه را چاپ می کند stubها جایگزین پیمانه هایی می شو ندکه توسط مولفه مورد آزمون فراخوانی می شوند stubاز واسط پیمانه فرا خوانده شده استفاده می کند حداقل دستکاری داده را انجام می دهد و کنترل را به پیمانه ای که در حال آزمون است باز می گرداند

آزمون انسجام در نرم افزار سنتی
(integeration testing)

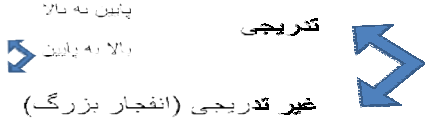
دلایل الزام آزمون انسجام

ممکن است داده ها در گذر از یک واسط از بین بروند یک پیمانه می تواند اثری وارونه بر دیگری داشته باشد عملکرد های فرعی پس از ترکیب ممکن است نتیجه ی مطلوب را نداشته باشد ساختمان داده های سراسری ممکن است باعث ایجاد مشکلات شوند

تعریف آزمون انسجام :

تکنیک سیستماتیک برای ایجاد ساختار برنامه ودر عین حال اجرای آزمون هایی جهت کشف خطاهادر ایجاد واسط ها

انواع روش های انسجام



غالبا انسجام مولفه ها به شیوه غیر تدریجی انجام می شودانسجام تدریجی آنتی تز انفجار بزرگ است

غیر تدریجی

1.همه مولفه ها از قبل ترکیب می شوند و کل برنامه یک جا آزمون می شوند

2. معمولا بی نظمی ایجاد می شود

3.مجموعه ای از خطاها مشاهده می شود

4. تصحیح دشوار است

تدریجی

1.برنامه ها تدریجا ایجاد می شوند و مورد آزمون قرارمی گیرد

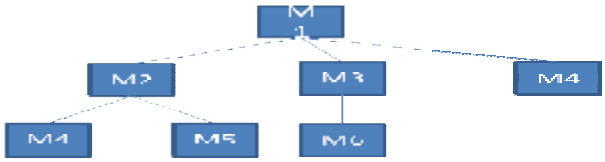
2.جدا کردن خطا ها و تصحیح آن ها آسانتر است

3.احتمال اینکه واسط ها به طور کامل مورد آزمون قرارگیرد زیاد است

انسجام تدریجی :انسجام بالا به پایین ،انسجام پایین به بالا

پیمانه ها با حرکت به طرف پایین در سلسله مراتب کنترلی و با شروع از پیمانه کنترل اصلی مجتمع می شوند

پیمانه های زیر دست هم به روش عمقی یا عرضی در ساختار قرار داده می شود



اگرانسجام عمقی باشد نمایش اولیه قابلیت عملیاتی به سازنده ومشتری قوت قلب میدهد

اگر انسجام عرضی باشد تا انتهای کار خروجی قابل رویت نیست

فرایند انسجام در مراحل 5 گانه:

1. پیمانه کنترل اصلی به عنوان درایور آزمون به کار می رود

stubها جایگزین کلیه مولفه ها می شود که مستقیما زیر دست پیمانه کنترل اصلی است

2.بسته به روش انتخابی (عرضی،عمقی) stubها ی زیر دست به نوبت،جای خود را به مولفه های واقعی می دهند

3.هر زمان با انسجام هر مولفه ، آزمون صورت می گیرد

4.با کامل شدن هر آزمون یک stubدیگر جای خود را به مولفه واقعی می دهد

5.آزمون رگرسیون برای اطمینان از عدم ورود خطا ها ی جدید انجام می شود

تکرار مراحل 2تا 5تا زمانیکه کل ساختار برنامه ساخته شود

عیب انسجام بالا به پایین :

مشکلات لوجیستیکی

متداول ترین این مشکلات زمانی به وجود می آید که برای آزمون سطوح بالاتر به پردازش در سطوح پایین تروجود داشته باشد در چنین شرایطی stubها جایگزین پیمانه ها ی سطح پایین می شوند از این رو داده ها در ساختار برنامه چندان به طرف بالا جریان پیدا نمی کنند

راه کار پیش روی آزمون گر:

1.به تاخیر انداختن بسیاری از آزمون ها تا اینکه stubها جای خود را به پیمانه های واقعی بدهند

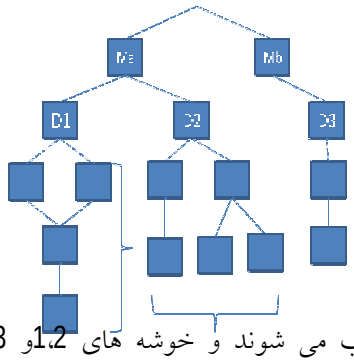
ارتباط بین بعضی از آزمون ها و بعضی پیمانه ها تحت کنترل نیست و این تعیین خطا را دشوار می کند

2. توسعه stubهایی که با اجرای عملیاتی محدود ،پیمانه واقعی را شبیه سازی می کند

سربار چشمگیری دارد اما عملی است

3. انسجام پایین به بالا

انسجام پایین به بالا:



مولفه ها با هم ترکیب می شوند و خوشه های 1،2و 3را به وجود می آورند هر خوشه با یک درایور مورد آزمون قرارا میگیرد درایور ها حذف می شوند و بین خوشه های آن واسط بر قرار می شود با حرکت به طرف بالا نیاز به درایور های آزمون جداگانه کمتر می شود

انواع آزمون های انسجام :

آزمون رگرسیون آزمون دود

آزمون رگرسیون :

1.عبارت است از اجرای دوباره ی زیر مجموعه ای از ازمون ها که قبلا اجرا شده اند تا اطمینان حاصل شود که تغییرات باعث انتشار اثرات جانبی ناخواسته نشده اند

2.آزمون رگرسیون فعالیتی است که به کمک آن می توان اطمینان یافت که تغییرات ، خطاها ی رفتاری ناخواسته ای را ایجاد نمی کند

3.آزمون رگرسیون را می توان به صورت دستی یا با استفاده از ابزار عقبگرد انجام داد

دلیل نیاز به آزمون رگرسیون :

هر بار که یک پیمانه جدید به عنوان بخشی از آزمون انسجام افزوده می شود 1.نرم افزار تغییر می کند

2. I/Oهای جدید رخ می دهد3.مسیر های جریان داده ی جدید برقرار می شود

4.هر بار به منطق کنترلی جدید روی آورده می شود5.مشکلاتی در عملکرد هایی که قبلا به درستی کار می کردند به وجود می آید

آزمون دود:

1.مورد کاربرد در هنگام توسعه ی محصولات نرم افزاری «بسته بندی شده »

2. یک ازمون گام به گام مورد کاربرد در پروژه هایی که زمان در آن اهمیت فراوان دارد

روش آزمون دود:

1. تشکیل سازه ها
2. آزمون ها یی برای کشف خطا های خطرناک
3. الحاق سازه ها به هم به یکی از روش های انسجام
4. آزمون روزانه محصول

تعریف مک کانل ازآزمون دود:

آزمون دود باید سیستم کامل را از ابتدا تا انتها امتحان کند لازم نیست این آزمون جامع باشد بلکه باید قادر به آشکار کردن مشکلات اصلی باشد .این آزمون باید به اندازه ای کامل باشد که اگر سازه در آن قبول شد ، بتوان فرض کرد که پایداری ان به حدی است که بتوان آزمون های کامل تری روی آن انجام داد

مزایای آزمون دود:

1. خطر انسجام به حداقل می رسد
2. کیفیت محصول نهایی بهبود مییابد
3. تشخیص و تصحیح خطا راحت تر می شود
4. ارزیابی پیشرفت آسان است

بررسی روش های آزمون انسجام

عیب انسجام بالا به پایین نیاز به stubها و مشکلات مرتبط با آن

عیب انسجام پایین به بالا تا زمان افزوده شدن آخرین پیمانه برنامه به عنوان یک موجودیت وجود ندارد

آزمون ساندویچی:

گزینش راهبرد انسجام به ویژگی نرم افزار و زمانبندی پروژه بستگی داردر آزمون ساندویچی یک روند ترکیبی استفاده می شودکه برای سطوح بالاتر از روش بالا به پایین و برای سطوح زیرین از روش پایین به بالا استفاده می شود

آزمون اعتبار سنجی :

پایان آزمون انسجام آغاز ازمون اعتبارسنجی است

در سطح اعتبار سنجی و سیستمی تمایز میان نرم افزار سنتی و شی گراو تحت وب رنگ می بازد

اعتبار سنجی زمانی موفق است که نرم افزار بر اساس انتظار مشتری عمل کند

بعد از آزمون اعتبار سنجی با دو نتیجه روبرو هستیم:

1.مشتری راضی است

2.وجود انحراف از خواسته مشتری که برای رفع کاستی به مشاوره با مشتری نیاز است

عناصر مهم اعتبار سنجی :

1.بازبینی پیکربندی

2.آزمون آلفا و بتا

بازبینی پیکربندی: گاه ممیزی خوانده می شود

هدف:

اطمینان یافتن از توسعه ی مناسب کلیه عناصر واطمینان از اینکه عناصر پیکر بندی دارای شناسنامه اند و دارای جزئیات لازم برای تقویت کردن فاز پشتیبانی چرخه زندگی نرم افزار است

سفارشی سازی محصول :

برای یک مشتری راهکار آزمون پذیرش

برای مشتریان زیاد آزمون آلفا و بتا

آزمون پذیرش:

مشتری را قادر به اعتبار سنجی کلیه خواسته ها می کند

به جای مهندس نرم افزار توسط کاربر نهایی انجام می شود

می تواند یک آزمون غیر رسمی ،رسمی و یا برنامه ریزی شده و سیستماتیک باشد

آزمون پذیرش را می توان در عرض یک هفته یا یک ماه انجام داد

آزمون آلفا:

در مکان سازنده نرم افزار ،توسط مشتری انجام میشود

نرم افزار در شرایط طبیعی اجرا می شود

سازنده ناظر بر اجرای آزمون بوده و خطا ها را ثبت می کند

آزمون آلفا در محیطی کنترل شده اجرا می شوند

آزمون بتا:

در یک یا چند مکان متعلق به کاربر نهایی انجام میشود

بر خلاف آزمون آلفا سازنده معمولا حضور ندارد

آزمون بتا یک کاربرد زنده از نرم افزار در محیطی است که سازنده قادر به کنترل آن نیست

مشتری کلیه خطا های ممکن را ثبت می کند و در دوره های زمانی متفاوت گزارش می دهد

مهندسان با توجه به مشکلات گزارش شده اصلاحات را انجام می دهند

شکل دیگر آزمون بتا(آزمون پذیرش مشتری):

یک نرم افزار سفارشی تحت قرار داد به مشتری تحویل داده می شود. مشتری یک سری ازمون جهت کشف خطا ها انجام می دهد تا قبل از پذیرش نرم افزار خطا تعیین شود . آزمون گاهی کاملا رسمی است و ممکن است روزها حتی هفته ها بیانجامد

آزمون سیستم :

هدف اصلی آزمون کل سیستم است .مشکل آزمون سیستم متهم کردن دیگران است

وظیفه مهندس نرم افزار در آزمون سیستم :

1.طراحی مسیر هایی برای کنترل خطا

2.اجرای آزمون هایی برای یافتن خطا ها ی بالقوه

3.ثبت نتایج آزمون به عنوان مدرک

4. شرکت در برنامه ریزی و آزمون های کل سیستم

انواع آزمون های سیستم :

آزمون ترمیم (recovery testing)

آزمون امنیت (security testing)

آزمون فشار (stress testing)

آزمون کارایی (performance testing)

آزمون استقرار (deployment testing)

آزمون ترمیم:

خودکار ، دستی

نرم افزار رابه طرق گوناگون وادار به شکست می کندو سپس در مورد اجرای مناسب ترمیم نرم افزار تحقیق می کند

آزمون فشار:

آزمون فشار برای مقابله با شرایط غیر عادی طراحی می شود آزمون گر ی خواهد بداند تا قبل از شکست نرم افزار را تا کجا می تواند تحت فشار قرار دهد در برخی شرایط به ویژه در الگوریتم های ریاضی ممکن است گستره بسیارکوچکی از داده های موجود در مرز داده ها ی معتبر برای یک برنامه باعث پردازش زیاد یا حتی نادرستی در کارایی شود

شکل دیگر آزمون فشار (تکنیک آزمون حساسیت :)

آزمون حساسیت می کوشد تا ترکیباتی از داده ها را در انواع معتبر ورودی کشف کند که ممکن است باعث ناپایداری یا پردازش نا مناسب شود

آزمون کارایی:

1.این آزمون در سراسر مراحل فرایند آزمون رخ می دهد حتی در سطح مولفه هاغالبا به همراه آزمون فشار انجام می شود

2.این آزمون به تجهیزات سخت افزاری و نرم افزاری نیاز دارد

آزمون استقرار:

1. این آزمون نرم افزار را در سیستم عامل ها و محیط ها ی متفاوت تمرین می دهد
2. گاهی آزمون پیکر بندی نامیده می شود
3. در آزمون استقرار همه ی روال های نصب و نرم افزار های تخصص یافته نصب و همه ی مستندات مورد استفاده در معرفی نرم افزار به کاربران نهایی بررسی می شوند

راهبرد ها ی آزمون برای نرم افزار های شی گرا :

آزمون واحد ها در حیطهOO شی گرا:

در نرم افزار شی گرا مفهوم واحد فرق می کند واحد قابل آزمون ، کلاس یا شی بسته بندی شده است .یک عمل به عنوان جزئی از یک کلاس آزمون می شود .

نرم افزار سنتی	نرم افزار شی گرا
آزمون واحدها	آزمون کلاس ها
آزمون واحد ها بیشتر بر جزئیات الگوریتمی یک پیمانه و داده هایی که در میان واسط پیمانه در جریان است تأکید دارد	آزمون کلاس ها به وسیله عملیات بسته بندی شده توسط کلاس و رفتار های کلاس انجام می شود

راهبر آزمون برای برنامه های تحت وب :

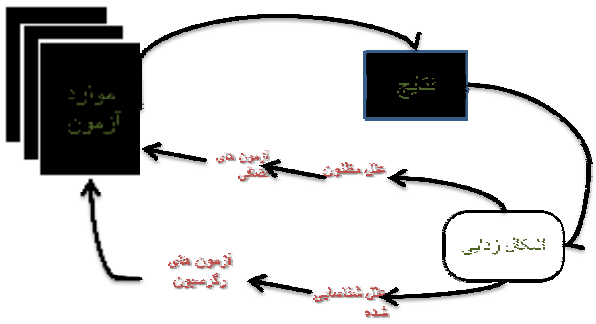
شامل اصول پایه ای مربوط به همه ی آزمون های نرم افزار می شود و تاکتیک هایی در آن به کار می رود که برای سیستم های شی گرا بکار می رود . پیوسته تعداد بی شماری از برنامه های تحت وب در حال تکامل است پس فرایند آزمون فعالیتی مداوم است و توسط کارمندان پشتیبانی اجرا می شود آن ها از آزمون رگرسیون استفاده می کنند

هنر اشکال زدایی (debuging)

فرایند ذهنی که نشانگان و نمود خارجی را به علت آنها ربط می دهد اشکال زدایی در نتیجه ی آزمون موفق رخ می دهد

اشکال زدایی می تواند و باید فرایند منظمی باشد

پیامد های اشکال زدایی:



نکاتی که دانستن آن اشکال زدایی را راحت تر می کند:

اگر نشانگان و علت از نظر جغرافیایی دور باشند مؤلفه های با چسبندگی بالا این مشکل را بیشتر می کنند نشانگان ممکن است با تصحیح خطا های دیگر رفع شود نشانگان ممکن است واقعا ناشی از خطا های دیگر باشد نشانگان ممکن است به خاطر خطا ها ی انسانی باشد که به سادگی قابل رد گیری نیست نشانگان ممکن است به خاطر خطا های زمانبندی باشد و نه پردازشی

انواع اشکال زدایی:

دستی: جستجوی جامع عقبگرد حذف علت

نیمه خودکار: اشکال زدایی را می توان با ابزار های اشکال زدایی تکمیل کرد و اشکال زدایی را نیمه خودکار کرد

جستجوی جامع : وقتی همه ی روش های دیگر به شکست می انجامد به کار گرفته می شود. متداول ترین و کم اثر بخش ترین روش است

عقبگرد : نسبتا متداول .موفق در برنامه های کوچک. با شروع از علت خطا و حرکت به سمت عقب کد منبع بررسی و مشکل پیدا می شود .

حذف علت : توسط استقرا یا استنتاج بیان می شود

مفهوم افراز دودویی وارد می شود

داده های مرتبط با خطا برای یافتن علت های بالقوه سازماندهی می شود یک فرضیه «علت » بیان می شود با استفاده از داده ها علت انکار یا اثبات می شود اگر علت درست باشد تصحیح می شود

تصحیح خطا

اگر اشکالی باشد باید رفع شود اما ممکن است این اشکال خود خطا های دیگری را وارد کند

فان فلک :

سه سوال را برای مهندس نرم افزار قبل از رفع خطا کمک می کند :

1. آیا خطا ی ایجاد شده در بحش دیگری از برنامه قرار دارد
2. با رفع اشکال فعلی چه اشکال دیگری ممکن است بروز کند
3. برای جلوگیری از این اشکال چه می توانسته ایم انجام دهیم

فصل18(آزمون برنامه های کاربردی سنتی)

آزمون پذیری نرم افزارصرفاعبارت است از این که یک برنامه کامپیوتری را چقدر آسان میتوان آزمایش کرد. به یک نرم افزار آزمون پذیر خصوصیات زیر را میتوان نسبت داد:

1-قابلیت کار کردن:هرچه بهترکارکند،آزمون آن اثربخش تر است.

2-قابلیت مشاهده:آنچه می بینید،همان است که آزمایش می کنید.

3-کنترل پذیری:هرچه بهتر بتوان نرم افزار را کنترل کرد،آزمون را بیشترو بهترمیتوان خودکاروبهینه کرد.

4-تجزیه پذیری:باکنترل دامنه کاربرد آزمون میتوان مسائل را سریع تر جداسازی کردوباهوشمندی بیشترانجام داد.

5-سادگی:هرچه مورد آزمون کوچکترباشد،سریع ترمیتوان آن راآزمود.

6-پایداری:هرچه تعدادتغییرات کمترباشد آزمون کمترباموانع مواجه میشود.

7-درک پذیری:هرچه اطلاعات بیشتری داشته باشیم آزمون هوشمندانه تری انجام میشود.

آزمون خوب با احتمال زیادی خطاهارامیابد=»آزمونگر بایدنرم افزار رابشناسدوکوشش کندتایک تصویرذهنی ازچگونگی شکست احتمالی نرم افزاربسازد.

مثال:یک دسته از شکست های بالقوه درروابط گرافیکی کاربر:شکست درتشخیص موقعیت ماوس است.

خصوصیات آزمون خوب:

آزمون خوب دارای زوایدنیست،زمان ومنابع آن محدود است.آزمون خوب نباید بیش از حد ساده ویابیش ازحد پیچیده باشد.دراجرای آزمونی که هدف آن با هدف یک آزمون دیگریکی است هیچ نکته ای وجود ندارد.هریک از آزمون ها باید دارای هدفی متناوب باشد.

دیدگاه های درونی و بیرونی نسبت به آزمون:

هرمحصول مهندسی شده را میتوان به یکی از دو شیوه زیر آزمود:

1-بادانستن قابلیت عملیاتی مشخصی که یک محصول برای ارائه آن طراحی شده است.

2-با دانستن کارکرد درونی محصول.

نکته:در رویکرداول درآزمون،دیدگاهی بیرونی مدنظراست که آزمون جعبه سیاه نامیده میشود.درورویکرددوم نیازبه دیدگاهی درونی است که آزمون جعبه سفید نامیده میشود.

هرآزمون جعبه سیاه جنبه ای عملیاتی ازسیستم رابررسی میکند.درحالی که ساختارمنطقی کمترموردتوجه قرار میگیرد.

درآزمون جعبه سفیدنرم افزار ازنظرجزئیات روالی مورد بررسی دقیق قرار میگیرد.

نکته کلیدی:آزمون های جعبه سفیدراتنها پس از ایجادطراحی درسطح مؤلفه هایا(کدمنبع)میتوان طراحی نمودزیراجزئیات منطقی برنامه بایددردسترس باشد.

آزمون جعبه سفید چیست؟به آن آزمون جعبه شیشه ای نیز میگویند ،یک روش طراحی برای موارد آزمون است که برای به دست آوردن موارد آزمون از ساختار کنترلی در برنامه استفاده میشود.

مواردآزمونی که مهندس نرم افزارباستفاده از متدهای آزمون جعبه سفید به دست می آوردبرچهارقسم است:

1-تضمین میکندکه همه ی مسیرهای مستقل دریک پیمانه حداقل یکبارامتحان شده اند.

2-همه ی تصمیم گیری های منطقی رادردوبخش درست وغلط امتحان میکند.

3-همه ی حلقه هارادرمرزهاودرداخل مرزهای عملیاتی آنهااجرا میکند.

4-ساختمان داده های داخلی راامتحان کندتااعتبارآنهاثابت شود.

نکته:آزمون مسیرهای پایه یک تکنیک آزمون جعبه سفیداست که نخستین بار توسط تام مک کیب پیشنهاد شدواین آزمون، طراح موارد آزمون را قادر میسازدتا میزانی منطقی از پیچیدگی رویه ای به دست آوردواز این میزان به عنوان راهنمایی جهت تعریف یک مجموعه ی پایه از مسیرهای اجرا استفاده کند.

*****از نماد موسوم به گراف جریان برای نمایش دادن جریان کنترلی استفاده میشود.*****

در گراف جریان هر گره بایک دایره نشان داده میشود،هرکدام ازاین دایره ها یک یاچند دستور رویه ای را نشان میدهد .ترتیبی ازمستطیل های پردازشی ویک لوزی تصمیم گیری رامیتوان دریک گره منفرد خلاصه نمود.پیکان های روی گراف جریان که یال یا پیوند خوانده میشوند،نشانگرجریان کنترل بوده ومشابه پیکان های نمودار گردشی هستند.

*****هررویه باید در گره پایان یابد حتی اگر گره هیچ دستور رویه ای رانشان ندهد.*****

تذکر:مساحت های محصور شده توسط یال هاوگره هاراناحیه(region)مینامند.

ساختار شرط مرکب در گراف جریان:

شرط مرکب زمانی رخ میدهد که یک یا چند عملگر بولی (NAND,NOR,AND,OR) منطقی در یک دستور شرطی موجود باشند.

نکته:هرگره که حاوی یک شرط باشد،گره گزاره ای نامیده میشود.

مسیرهای مستقل برنامه:

هرمسیری از برنامه است که حداقل یک مجموعه جدیدازدستورهای پردازش یایک دستور شرطی رامعرفی کند.اگر مسیرمستقل برحسب گراف جریان بیان شودحداقل بایددرراستای یک یال حرکت کندکه پیش ازتعریف مسیراز آن عبور نکرده باشد.

پیچیدگی سیکلوماتیک ریشه در نظریه گراف ها داردویک معیار نرم افزاری سودمند را فراهم می آورد.

پیچیدگی به یکی ازسه روش زیر محاسبه میشود:

1-تعداد نواحی گراف جریان متناظربا سیکلوماتیک.

2-پیچیدگی سیکلوماتیکV(G)برای یک گراف جریانGبه صورت V(G)=E-N+2 تعریف میشود که در آنEتعداد یال های گراف جریان و Nتعداد گره های آن است.

3-پیچیدگی سیکلوماتیکV(G)برای یک گراف جریانGبه صورت V(G)=P+1 نیز تعریف میشود که Pتعداد گره های گزاره ای موجود در گراف جریانGاست.

نکته کلیدی:پیچیدگی سیکلوماتیک مرز بالایی تعدادموارد آزمونی را مشخص میکندکه لازم هستند تاتضمین شود که هردستور از برنامه دست کم یکبار اجرا شده است.

برای به دست آوردن موارد آزمونمسیرهای پایه ای را میتوان در یک طراحی رویه ای یا کد منبع به کار برد.

برای به دست آردن مجموعه ی پایه باید چهار مرحله ی زیر را انجام داد:

1-استفاده از طراحی یا کد به عنوان یک بسترورسم گراف جریان مربوط.

2-پیچیدگی سیکلوماتیک گراف جریان حاصل را تعیین کنید.

3-تعیین مجموعه پایه برای مسیرهای مستطیل خطی.

4-مواردآزمونی را تهیه کنید که اجرای همه ی مسیرها را در مجموعه ی پایه الزامی کنند.

ماتریس گراف چیست؟

ماتریس گراف یک ماتریس مربعی است که اندازه آن (یعنی تعدادسطرهاو ستون های آن)برابرتعدادگره های موجود درگراف جریان است.هرسطروستون متناظر بایکی از گره هاست ومدخل های ماتریس بااتصالات یعنی (یال ها)میان گره ها متناظرند.

میتوان با افزودن وزن پیوندبه هریک از مدخل های ماتریس، آن را به ابزاری پر قدرت برای ارزیابی ساختار کنترلی برنامه در اثنای آزمون تبدیل کرد.

وزن پیوند اطلاعاتی درباره جریان کنترل فراهم می آورد. وزن پیوند در ساده ترین شکل خود برابر1(وجودارتباط)یا0(نبودارتباط)است .

خواص جالب دیگری را نیز میتوان به اوزان پیوند نسبت داد:

1-احتمال آنکه یک پیوند(یال)اجراشود.

2-زمان پردازش صرف شده برای طی کردن یک پیوند.

3-حافظه لازم برای یک پیوند.

4-منابع لازم برای طی کردن یک پیوند.

آزمون ساختارکنترلی:

1-آزمون شرط ها

2-آزمون جریان داده ها

3-آزمون حلقه ها

4-آزمون جعبه سیاه

چهار دسته ی متفاوتاز حلقه هارا میتوان تعریف کرد:

1-حلقه های ساده

2-حلقه های تسلسلی

3-حلقه های تودرتو

4-حلقه های غیرساخت یافته

آزمون جعبه سیاه:

آزمون جعبه سیاه که آزمون رفتاری نیز خوانده میشود،برخواسته های عملیاتی نرم افزارتکیه دارد.یعنی،آزمون جعبه سیاه مهندس نرم افزاررا قادر میسازدتا مجموعه هایی ازشرط های ورودی رابه دست آوردکه همه ی خواسته های عملیاتی برنامه را به طور کامل امتحان کند.

جایگزینی برای تکنیک های جعبه سفید به شمار نمی رود، بلکه یک روش مکمل است که احتمال پیدا کردن دسته دیگری از خطا هارا فراهم می آورد.

آزمون جعبه سیاه سعی میکندخطاهای موجود در این گروه ها را بیابد:

1-عملکرد نادرست یاجاافتاده

2-خطاهای واسط

3-خطاهای موجوددرساختمان داده ها یادستیابی به یانک اطلاعاتی خارجی

4-خطاهای رفتاری یاکارایی

5-خطاهای مقداردهی اولیه یا خاتمه برنامه

روش های آزمون مبتنی بر گراف:

نخستین گام در آزمون جعبه سیاه شناخت اشیایی است که در نرم افزار مدلسازی میشوندونیز روابط میان این اشیاست.هنگامی که این منظور برآورده شد،مرحله بعدی تعیین تعدادی آزمون است که ثابت کنند همه ی اشیاء،رابطه ی مورد انتظار را بایکدیگر دارند.

برای انجام این مراحل ،مهندس نرم افزار باایجاد یک گراف شروع میکند:مجموعه ای از گره ها که اشیاء رانشان میدهند،پیوندها که روابط میان اشیاء رانشان میدهند،وزن گره که خواص گره را تعیین میکنندو وزن پیوند که خواص پیوند راتوصیف میکنند.

پیوندها اشکال متفاوتی به خود میگیرند:

1-جهت دار:بایک پیکان نشان داده میشودونشان میدهدکه رابطه تنها دریک جهت برقرار است.

2-دوجهته:پیوند متقارن نیز خوانده میشود،بدان معناست که رابطه در دو جهت برقرار است.

نکته کلیدی:گراف،نشانگرروابط میان اشیای داده واشیای برنامه ای است و به کمک آن میتوانید موارد آزمونی به دست آوریدکه خطاهای مرتبط با این روابط را جستجو میکنند.

پیوندجهت دار ، پیوندهای موازی، یوندبدون جهت

بیرزچند روش آزمون رفتاری را توصیه میکند که در آن ها از گراف استفاده میشود:

1-مدلسزی جریان تراکنش

2-مدلسازی حالت متناهی

3-مدلسازی جریان داده ها

4-مدلسازی زمان بندی

افراز هم ارزی

افراز هم ارزییکی از روش های آزمون جعبه سیاه است که دامنه ورودی یک برنامه را با دسته هایی از داده ها تقسیم میکندوموردآزمون را میتوان از روی آن به دست آورد.

مورد آزمون برای افراز هم ارزی،مبتنی بر تعیین دسته های هم ارزی برای یک شرط ورودی است.

تحلیل مقادیر مرزی:

تعداد خطاهای موجود در مرزهای دامنه ورودی نسبت به مقادیر مرکزی دامنه بیشتر است.به همین دلیل تکنیکی موسوم به تحلیل مقادیر مرزی(BVA)توسعه یافته است.تحلیل مقادیر مرزی،به موارد آزمونی منجر میشودکه مقادیر مرزی را امتحان میکنند.

نکته کلیدی: BVAافرازهمارزی راباتأکیدبر"لبه های"یک دسته ی هم ارزی بسط میدهد.

آزمون آرایه های متعامد:

ورودی بسیاری از برنامه های کاربردی محدود است.یعنی تعداد پارامترهای ورودی،کوچک ومقادیری که هریک از پارامترها به خود میگیرنددارای مرز مشخصی است.هنگامی که این اعدادبسیارکوچک باشند میتوان تمام حالت های ورودی را در نظر گرفت و پردازش دامنه ورودی را به طور جامع مورد آزمایش قرار داد.ولی با رشد تعداد مقادیر ورودی وتعداد مقادیر مجزا جهت هر عنصر داده ای آزمون جامع غیرعملی وامکانپذیرمیشود.

آزمون آرایه های متعامد را میتوان در مورد مسائلی به کاربرد که در آن ها دامنه ورودی نسبتاًکوچک است،ولی اجرای آزمون جامعیت بیش از حد بزرگ است.

نکته کلیدی: به کمک آزمون آرایه های متعامد میتوانید موارد آزمونی طراحی کنید که حداکثر پوشش آزمون را باتعدادی منطقی از موارد آزمون فراهم سازند.

آزمون مبتنی بر مدل:

یک آزمون مبتنی بر مدل(MBT)یک تکنیکاًآزمون جعبه سیاه است که از اطلاعات موجود در مدل خواسته ها به عنوان مبنایی برای تولید موارد آزمون بهره میبرد.

تکنیک MBTبه پنج مرحله نیاز دارد:

1-تحلیل یک مدل رفتاری موجودبرای نرم افزار با ایجاد آن

2-مرور مدل رفتاری و مشخص کردن ورودی هایی که نرم افزار راوادار میسازند تا از حالتی به حالت دیگر گذار کند

3-مرور بر مدل رفتاری و توجه به خروجی قابل انتظار از نرم افزار ،هنگامی که از حالتی به حالت دیگر گذار میکند

4-اجرای موارد آزمون

5-نتایج واقعی و مورد انتظار را مقایسه کنید ودر صورت نیاز ،اقدامات تصحیحی را به عمل آورید.

آزمون واسط گرافیکی کاربر:

واسط گرافیکی کاربر(GUI)مشکات جالبی سر راه مهندسی نرم افزار قرار میدهد . به دلیل وجود مؤلفه های قابل استفاده مجدد در محیط های توسعه ی GUI، ایجاد واسط کاربر ،با زمانی کمتر وبا دقت بیشتر امکانپذیر شده است. ولی در عین حال ،پیچیدگی GUIنیز فزونی یافته است.

آزمون مربوط به سیستم های بی درنگ:

ماهیت وابسته به زمان در بسیاری از کاربردهای بی درنگ ،یک عنصر دشوار"زمان "را به آزمون می افزاید .طراح موارد آزمون ،نه تنها آزمون جعبه سیاهو جعبه سفیدرا در نظر بگیرید؛بلکه باید کنترل رویدادها،زمان بندی داده ها وموازی بودن وظایفی را که با داده ها کار میکنند در نظر بگیرید.

الگوهای مربوط به آزمون نرم افزار:

مسائل رایج در آزمون و راهکارهایی را توصیف میکنند که ممکن است شما را در کار با آن ها یاری دهند.

نکته کلیدی:الگوهای آزمون میتوانند تیم نرم افزار را در برقراری ارتباط اثربخش تر ودرباره ی آزمون ها و درک بهتر نیروهای منجر به یک روش آزمون خاص یاری دهند.

فصل 19 : تحویل داده نشد.

فصل 20 : آزمون برنامه های کاربردی تحت وب

مفهوم آزمون برای برنامه های تحت وب

آزمون : فرایند تمرین دادن نرم افزار با هدف یافتن خطاهاست . برای درک اهداف آزمون در حیطه مهندسی وب باید ابعاد بسیاری از کیفیت برنامه های تحت وب را در نظر گرفت آزمون در حیطه مهندسی وب ، باید ابعاد بسیاری از کیفیت برنامه های تحت وب را در نظر گرفت.

کیفیت در یک برنامه تحت وب، نتیجه طراحی خوب است. محتوا در هر دو سطح نحوی و معنایی ارزیابی می شود 1- در سطح لغوی ، املای واژه علامت گذاری و گرامر و ...

2- در سطح معنایی ، درستی ، سازگاری ، و فقدان اجسام بررسی می شود.

خطاهای موجود در یک محیط برنامه ی تحت وب:

1- چون بسیاری از آزمون ها ی برنامه تحت وب مشکلاتی را آشکار می کنند که نخستین بار به چشم کلانیت می آید ، غالباً نشانه ای از خطا را می بینند، نه خود خطارا.
2- گرچه برخی خطاها نتیجه ی طراحی نادرست هستند ولی بسیاری از خطاها را می توان تا پیکر بندی برنامه تحت وب دنبال کرد.
3- برخی خطاها از محیط عملیاتی ایستا ناشی می شود در حالی که خطاها را می توان تا پیکر بندی برنامه تحت وب دنبال کرد.
4- برخی خطاها از محیط عملیاتی ایستا ناشی می شود در حالی که خطاهای دیگر را می توان به محیط عملیاتی ایستا نسبت داد و...

راهبرد آزمون: همان اصول پایه ای مربوط به آزمون همه ی نرم افزار ها مد نظر است . که به طور خلاصه : مدل محتوای برنامه تحت وب مرور می شود تا خطاها برملا گردد.

مدل واسط مرور می شود تا اطمینان حاصل شود که همه ی پرونده ها ی کاربرد قابل پاسخگویی است.

گشت واگذار در سراسر معماری آزمایش می شود. آزمون کارایی اجرا می شود و ...

برنامه ریزی آزمون :برخی سازندگان برنامه ریزی نمی کنند آنها فقط شروع می کنند به این امید که شاید یک شاهکار ایجاد کنند. ولی برنامه ریزی بسیار ضروری است.

فرآیند آزمون : با آزمون هایی آغاز می شود که محتوای عملیات را که به چشم کاربرنهایی می آید، تمرین می دهد.

آزمون محتوا: در آزمون محتوا دو بخش مرور و تولید موارد آزمون قابل اجرا باهم ترکیب می شود.

اهداف آزمون محتوا: سه هدف مهم 1- کشف خطاهای نحوی 2- کشف خطاهای معنایی 3- خطاهای موجود در سازماندهی

آزمون بانک اطلاعاتی: در این آزمون اشیای محتوایی پویاست.

آزمون وسط کاربر: در سه نقطه متمایز رخ می دهد. 1- طی مرحله تحلیل خواسته ها 2- طی طراحی و 3- طی آزمون

راهبرد آزمون واسط: سازوکارهای تعامل را تمرین می دهد. و جنبه های زیبایی شناختی واسط کاربر را اعتبار سنجی می کند. راهبرد کلی عبارت است از : 1- کشف خطای مرتبط با سازو کارهای خاص واسط 2- کشف خطاهای موجود در روش پیاده سازی معنا شناسی گشت و گذار

آزمون سازو کارهای واسط: هنگامی که یک کاربری با یک برنامه تحت وب تعامل می کند ، این تعامل از طریق یک یا چند ساز و کار واسط رخ می دهد.

آزمون معنا شناختی واسط: وقتی که هر کلام از سازوکارهای واسط، مورد آزمون واحد قرار گرفت، کانون توجه آزمون به معنا شناسی واسط تغییر می کند . و این را تعیین می کند که طراحی تا چه حد ، کاربران را در نظر دارد، راهنماییی های روشنی را ارائه می دهد. با زخورد ها را تحویل می دهد.

آزمون های قابلیت استفاده : آزمون های قابلیت استفاده مشابه با آزمایش معنا شناختی واسط است. از این لحاظ که این آزمون نیز میزان اثر بخش تعامل کاربران با برنامه تحت وب ارزیابی می شود . مرور ها و آزمون ها ی قابلیت استفاده به این منظور طراحی شوند که تعیین کنند واسط برنامه تحت وب تا چه میزان کارها را برای کاربر آسان می کند.

آزمون های سازگاری: کامپیوترها، دستگاه نمایش، سیستم عامل، مرورگرها می توانند تاثیر چشمگیری برعملکرد برنامه تحت وب بگذارند. در برخی موارد مسائل سازگاری کوچک هیچ مشکل چشمگیری به بار نمی آورد. ولی در سایر موارد خطاهای جدی ممکن است مشاهده شود. مثلا سرعت دانلود ممکن است غیر قابل قبول باشد . و این آزمون سازگاری ***** دارد.این مشکلات را قبل از آنلاین شدن برنامه تحت وب کشف کند. هدف این آزمون ، کشف خطاها یا مشکلاتی در اجراست تا اختلاف های پیکر بندی قابل رد گیری باشند.

آزمون در سطح مولفه ها : که گاهی آزمون توابع نامیده می شود: مجموعه ای از آزمون ها را کانون توجه ار می دهد که سعی در کشف خطاهای موجود در توابع برنامه های تحت وب دارند. موارد آزمون در سطح مولفه ها به وسیله ورودی در سطح فرم ها طراحی می شود . هنگامی که داده های فرم ها تعیین شدند، کاربر یک دکمه یا سازوکار کنترلی دیگری را برای شروع اجرا انتخاب می کند. هر مورد آزمون در سطح مولفه ها کلیه مقادیر ورودی و خروجی مورد انتظار از مولفه ها را مشخص می کند، خروجی واقعی که به عنوان نتیجه ای از آزمون تولید می شود برای ارجاهای بعدی در اثنای پشتیبانی و نگهداری ثبت می شود.

آزمون گشت و گذار و نخستین مرحله ازآزمون گشت و گذار ، در واقع طی آزمون واسط آغاز می شود. سازوکارهای گشت و گذار ، آزمایش می شوند تا اطمینان حاصل شود که هر کدام وظیفه خاص خود را اجرا می کند.

آزمون معنا شناختی گشت و گذار:واحد معنا شناختی گشت و گذار به عنوان مجموعه ای از اطلاعات و ساختارهای گشت و گذاری مرتبط دانستیم که با همکاری یکدیگر، زیر مجموعه ای از خواسته های مرتبط کاربر را برآورده می سازند. آزمون گشت و گذار، همانند آزمون واسط و قابلیت استفاده ، باید توسط هر تعداد ممکن از هر هیئت ها اجرا شوند.

آزمون پیکربندی: تغییر پذیری و ناپایداری ، عوامل مهمی هستند که آزمون برنامه تحت وب را به چالش می کشد. وظیفه آزمون پیکر بندی، تمرین دادن هر پیکر بندی ممکن در طرف کلا نیت نیست. در عوض باید مجموعه ای از پیکر بندی های محتمل در طرف کلانیت و در طرف سرور را بیازماید تا اطمینان حاصل شود که تجربه ی کاربر، روی همیه آنها یکسان خواهد بود و خطاهایی راکه ممکن است خاص یک پیکربندی ویژه باشند، جدا کند.

مسائل طرف سرور: موارد آزمون پیکربندی در طرف سهرور طوری طراحی می شوند که وارد م@@@ کنند آیا اطلاعات پیش بینی شده برای سرور می توانند بدون خطا برنامه تحت وب را پشتیبانی کنند. ازجمه پرسش هایی که باید طی آزمون پیکر بندی طرف سرور پرسید عبارتند از:

آیا برنامه تحت وب به طور کام باسیستم عامل سرور سازگار است؟

آیا برنامه تحت وب از انسجام مناسب با نرم ازار بانک اطلاعاتی برخوردار هستند؟

آیا برنامه تحت وب به تفاوت نسخه در نرم افزار بانک اطلاعاتی حساس است؟

مسائل طرف کلانیت : در طرف کلانیت آزمون های پیکر بندی ، بر سازگاری با پیکربندی های حاوی یک یا چند ترکیب چایگشتی از مولفه های زیر تاکید دارند:

سخت افزار CPU ، حافظه، دیسک ها و دستگاه چاپ

سیستم عامل linux ، windows، یک سیستم عامل تلفن همراه

نرم افزار مرورگر، Fire fox ، chrome، و ...

آزمون امنیت : امنیت برنامه تحت وب موضوعی پیچیده است که باید پیش از دستیابی به آزمون امنیتی اثر بخش آن را به طور کامل درک کرد.

آزمون های امنیتی طوری طراحی می شوند که آسیب پذیری های محیط طرف کلانیت ارتباطات شبکه ای که در تبادل اطلاعات میان سرور و کلانیت رخ می دهند و محیط سرور را بر ملاسازند

به هر کدام از دامنه ها می توان حمله کرد و کشف نقاط ضعفی که توسط افراد سودجو ممکن است مورد سوء استفاده قرار گیرد. وظیفه ی آزمون گرامینی است.

آزمون کارایی: هیچ چیز ناراحت کننده تر از این که برنامه ی تحت وب نیست که چند دقیقه وقت صرف دانلود محتوا کند، در حالی که سایت های رقیب همان محتوا را در عرض چند ثانیه دانلود می کنند. از آزمون کارایی برای کشف مسائل کارایی استفاده می شود که ممکن است در اثر فقدان منابع طرف سرور، پهنای باند نامناسب برای شبکه و سایر مسائل نرم افزار که ممکن است به کاهش کارایی کلانیت – سرور بینجامد، رخ می دهد.

هدف دوگانه دنبال می شود:

- درک چگونگی پاسخگویی سیستم با افزایش بار تحمیل شده
- جمع آوری معیارهایی که به اصطلاحاتی در طراحی برای بهبود بخشیدن به کارایی می انجامد

اهداف آزمون کارایی: آزمون کارایی برای شبیه سازی شرایط ازدحام بار در جهان واقعی طراحی می شوند . آزمون کارایی به پرسش های زیر پاسخ می دهد.

1- کدام مولفه سیستم مسئول تنزل کارایی اند.
2- ایا تنزل کارایی بر امنیت سیستم تاثیر دارد
3- آیاتنزل کارایی بر درآمد شرکت تاثیر داردو..

آزمون ازدحام بار: هدف آزمون ازدحام بار ، تعیین چگونگی پاسخگویی برنامه های تحت وب و محیط سرور به انواع شرایط ازدحام بار است.

آزمون فشار : آزمون فشار ادامه آزمون ازدحام بار است. و هدف از این آزمون ها پاسخ دادن به هر کدام از پرسش های زیر است:

1- آیا با فراتر رفتن از ظرفیت ها، سیستم «به ملایمت» تنزل می یابد یا سرور از کار می افتد؟
2- ایا با فرا رفتن از حد ظرفیت، تراکنش ها از بین می روند؟
3- آیا با فرا رفتن از حد ظرفیت، انسجام داده ها تاثیر می پذیرد؟

فصل 21 :مدل سازی و وارسی رسمی

دو روش مهندسی نرم افزار پیشرفته-مهندسی نرم افزار اتاق تمیز و روش های رسمی-با فراهم ساختن یک رویکرد ریاضی-محور برای برنامه ریزی مدل سازی وتوانایی وارسی مدل حاصل،به تیم نرم افزاری کمک می کنند تا«کار را در همان بار نخست درست انجام دهد.» مهندسی نرم افزار اتاق تمیز بر وارسی ریاضی پیش از شروع ساخت برنامه و بر تایید قابلیت اطمینان به عنوان بخشی از فعالیت آزمون تاکید دارد.در روش های رسمی از نظریه مجموعه ها و نماد گذاری منطقی برای ایجاد بیان واضحی از حقایق (خواسته ها)استفاده می شود که می توان آنها را برای بهبود بخشیدن به صحت و درستی(و حتی اثبات آن)تحلیل کرد.خط مبنای هر دو روش،ایجاد نرم افزاری با مقادیر بسیار پایین خطاست.

اشتباهات باعث دوباره کاری می شوند.دوباره کاری زمان می برد و هزینه ها را افزایش می دهد.بهتر نیست اگر بتوانیم تعداد اشتباهات(خطاها)را در زمان طراحی و ساخت نرم افزار کاهش دهیم؟

مدل های خواسته ها و طراحی با به کارگیری یک نمادگذاری تخصصی ایجاد می شوند که قابلیت وارسی ریاضی را داشته باشند.مهندسی نرم افزار اتاق تمیز،از نمایش ساختارهای چهارگوش استفاده می کند که سیستم(یا جنبه ای از سیستم)را در سطح مشخصی از انتزاع کپسوله می کنند.وارسی،هنگامی به کار برده می شود که طراحی ساختارهای چهارگوش کامل باشد.هنگامی که صحت برای هر ساختار چهارگوش به تایید رسید،آزمون کاربرد آماری آغاز می شود.روش های رسمی،با به کارگیری نمادها و مفاهیم نظریه مجموعه ها برای

تعریف داده های ثابت،حالت ها و عملیات های مربوط به یک وظیفه سیستمی،خواسته های نرم افزار را به نمایشی رسمی تر ترجمه می کنند.

بر خلاف مرورر و آزمون که پس از توسعه ی مدل ها و کدها آغاز می شوند،مدل سازی و وارسی رسمی شامل روش های مدل سازی تخصص یافته ای می شود که در کنار رویکردهای وارسی تجویزی به کار برده می شوند.بدون رویکردهای مدل سازی مناسب،وارسی را نمی توان به انجام رساند.

مهندسی نرم افزار اتاق تمیز از یک نسخه ی تخصص یافته از مدل نرم افزار افزایشی است که در فصل 2 معرفی شد.چند تیم نرم افزاری کوچک و مستقل،«لوله ای از نسخه های نرم افزار» [b94Lin] را توسعه می دهند.هر نسخه پس از این که به تایید رسید به کل سیستم افزوده می شود.از این رو،قابلیت عملیاتی سیستم با گذر زمان رشد می کند.

در داخل لوله ی نسخه های اتاق تمیز،وظایف زیر باید به انجام برسد:

برنامه ریزی برای نسخه ها ، جمع آوری خواسته ها، مشخصه ی ساختار چهارگوش، طراحی رسمی، وارسی، تولید،بازرسی و وارسی کدها.

برنامه ریزی برای آزمون های آماری

آزمون کاربرد آماری

صدور گواهی

مشخصات عملیاتی

در رویکرد مدل سازی در مهندسی نرم افزار اتاق تمیز،از روش موسوم به تعیین مشخصات ساختارهای چهارگوش استفاده می شود.یک«چهارگوش»سیستم(یا جنبه ای از سیستم)را در سطحی از جزئیات پنهان سازی می کند.از طریق فرایند پالایش مرحله به مرحله و پرداختن به جزئیات،چهارگوش ها به صورت یک سلسله مراتب پالایش می شوند که در آن هر چهار گوش دارای شفافیت ارجاعی است.یعنی،«محتوای اطلاعاتی هر مشخصه برای تعریف پالایش آن کفایت می کند،بدون این که به پیاده سازی هسچ جعبه دیگری وابسته باشد«[Lin94b].به این ترتیب ،تحلیل گر می تواند سیستم را به صورت سلسله مراتبی با حرکت از نمایش اساسی در بالا تا جزئیات خاص پیاده سازی در پایین،افزاز کند.

از سه نوع چهار گوش استفاده می شود:

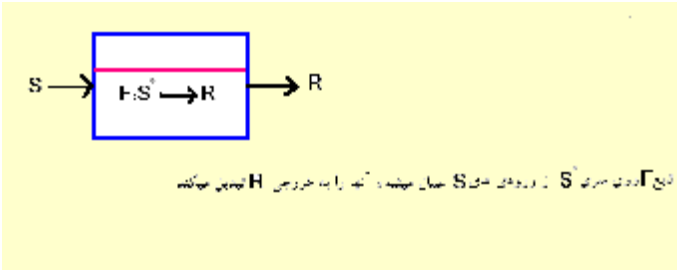
چهار گوش سیاه(Black Bok)چهار گوش سیاه،رفتار یک سیستم یا بخشی از یک سیستم را مشخص می کند.

چهار گوش حالت(State Box).چهار گوش حالت،سرویس ها(عملیات ها) و داده های حالت را به شیوه ای مشابه با اشیا کپسوله می کنند.

چهار گوش شفاف(Clear Box).توابع گذاری(transition functions) که توسط چهار گوش حالت مشخص می شوند،در چهارگوش شفاف تعریف می شوند.

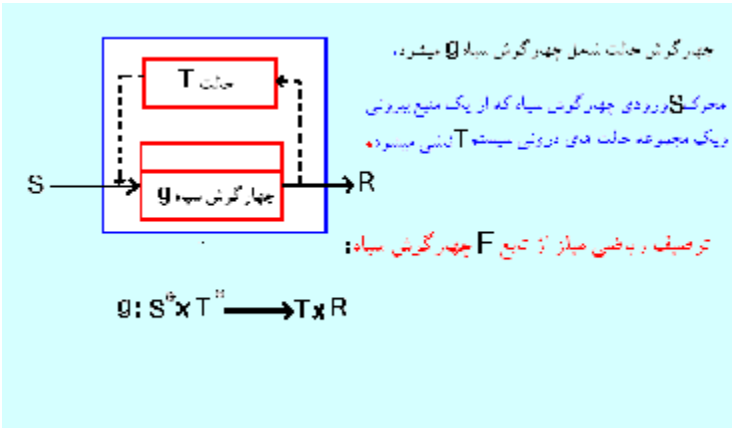
مشخصات چهار گوش سیاه

مشخصات چهار گوش سیاه،توصیف گر انتزاع،محرک ها و پاسخ با استفاده از نمادگذاری نشان داده در شکل3-21 است[Mil88].تابعf روی سری*s از ورودی ها(محرک های) sاعمال می شود وآن ها را به خروجی(پاسخ)Rتبدیل می کند.برای مولفه های یک نرم افزار ساده،fممکن است یک تابع ریاضی باشد،ولی در کل،f با بکارگیری زبان طبیعی(یک زبان رسمی برای تعیین مشخصات)توصیف می شود.



مشخصات چهار گوش حالت

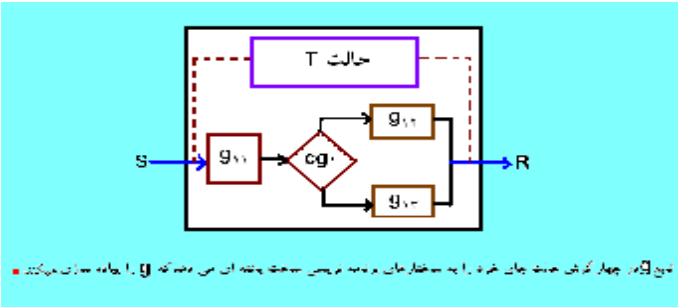
چهار گوش حالت«تعمیم ساده ای از یک ماشین حالت است»[Mil88].با به خاطر آوردن بحث مدل سازی رفتاری و نمودارهای حالت در فصل7،هر حالت یک شیوه ی مشاهده پذیر از رفتار سیستم است.با رخ دادن پردازش،سیستم به رویدادها(محرک ها)با انجام گذار از حالت فعلی به حالت جدید پاسخ می دهد.با انجام این گذار،ممکن استکنشی رخ دهد.چهار گوش حالت از یک انتزاع داده ای برای تعیین گذار به حالت بعدی واکنش(پاسخی)استفاده می کند که در نتیجه ی گذار رخ می دهد.



مشخصه ی چهار گوش شفاف

مشخصات چهار گوش شفاف،همسویی تنگاتنگی با طراحی روالی و برنامه نویسی ساخت یافته دارد.در اصل،زیر تابعgرا پیاده سازی می کند.

به عنوان مثال،چهار گوش شفاف نشان داده شده در شکل5-21 را در نظر بگیرید.چهار گوش سیاهg،که در شکل 3-21 نشان داده شده،جای خود را به یک ساختار ترتیبی می دهد که شامل یک ساختار شرطی می شود.این ها نیز به نوبه خود و در ادامه ی پالایش مرحله ای به چهارگوش هایی با سطح پایین تر قابل پالایش هستند.



طراحی اتاق تمیز

در مهندسی نرم افزار اتاق تمیز،از فلسفه ی برنامه نویسی ساخت یافته،استفاده ی گسترده به عمل می آید ولی در این مورد،برنامه نویسی ساخت یافته بسیار شدیدتر استفاده می شود.

توابع پردازشی پایه(که طی پالایش های اولیه مشخصات توصیف می شوند)با استفاده از یک «روش بسط مرحله ای توابع ریاضی به ساختارهای زیر تابع ها و ارتباط های منطقی[مانندif-then-else]،پالایش می شوند که در آن،این بسط،آن قدر ادامه می یابد تا همه ی زیر تابع های شناسایی شده را بتوان مستقیماً به زبان برنامه نویسی به کار رفته برای پیاده سازی بیان کرد.»[Dye92]

از رویکرد برنامه نویسی ساخت یافته می توان به طور اثر بخش برای پالایش تابع استفاده کرد.

پالایش طراحی

هر مشخصه ی چهار گوش شفاف،نشان گر طراحی یک روال (زیر تابع)لازم برای محقق ساختن یک گذار چهار گوش حالت است.در داخل یک چهار گوش شفاف ،از ساختارهای برنامه نویسی ساخت یافته و پالایش مرحله ای برای پایش جزئیات روالی استفاده می شود.برای مثال ،تابعfبه یک سری زیر تابع های gوhپالایش می شود.این زیر تابع ها نیز به نوبه ی خود ساختارهای شرطی(مانند-do) whileوif-then-elseپالایش می شوند.پالایش باز هم ادامه می یابد تا این که جزئیات روالی کافی برای ایجاد مولفه ی مورد نظر موجود باشد.

تیم اتاق تمیز در هر سطح از پالایش،یک وارسی صحت رسمی انجام می دهد.برای دستیابی به این منظور،مجموعه ای از شرایط عمومی صحت،به ساختارهای برنامه نویسی ساخت یافته متصل می شوند.اگر تابعh به یک سریgوh بسط داده شود،شرط صحت برای همه ی ورودی های hعبارت است از:

آیاgو پس از آنf،h را انجام می دهند؟

هنگامی که pبه یک ساختار شرطی به شکلif<c>then q،else r پالایش شود،شرط صحت برای کلیه ورودی های pعبارت است از:

هر گاه<c>درست باشدآیا p،qرا انجام می دهد؛و هر گاه<c>نادرست باشد،آیا q,r را انجام می دهد؟

هنگامی که تابعm به صورت یک حلقه پالایش شود،شرط های درستی برای همه ی ورودی های mعبارتند از:

آیا پایان یافتن حلقه تضمین شده است؟

هر گاه <c> درست باشد آیا nکه پس از آنmرا انجام می دهد؛ و هر گاه <c>نادرست باشد،آیا با جا انداختن حلقه،هنوز mانجام می شود؟

هر بار که یک چهارگوش شفاف به سطح بعدی جزئیات پالایش شود،این شرط های درستی اعمال می شوند.

وارسی طراحی

باید توجه داشته باشید که استفاده از ساختارهای برنامه نویسی ساخت یافته،تعداد آزمون های درستی را که باید اجرا شود،محدود می کند.برای ساختارهای ترتیبی یک شرط؛دو شرط برای if-then-elseو سه شرط برای حلقه ها چک می شود.

برای نشان دادن وارسی برای یک طراحی روالی از مثال ساده ای استفاده می کنیم که نخستین بار توسط لینگر،میلز و ویت[Lin79] معرفی شده است.هدف،طراحی و وارسی برنامه کوچکی است که Y،جزء صحیح جذر یک عدد صحیحxرا می دهد.

برای اثبات صحت طراحی ،لازم است شرط های init،loop،cont،yesوexit که در تمامی حالت ها اثبات شوند.این اثبات ها را گاهی اثبات فرعی می نامند.

آزمون اتاق تمیز

راهبرد و تاکتیک های آزمون اتاق تمیز با رویکردهای سنتی آزمون تفاوت بنیادی دارند.در روش های سنتی یک مجموعه موارد آزمون به دست می آید که خطاهای طراحی و کد نویسی را کشف می کنند.هدف آزمون اتاق تمیز،اعتبار سنجی خواسته های نرم افزار است و برای این منظور نشان می دهد که یک نمونه آماری از موارد آزمون با موفقیت اجرا شده است.

آزمون کاربرد آماری(Statistical Use Testing)

کاربر یک برنامه کامپیوتری،به ندرت نیاز پیدا می کند تا جزئیات فنی طراحی را بداند.رفتار برنامه که به چشم کاربر می آید،به وسیله ی ورودی ها و رویدادهایی تعیین می شوند که غالباً توسط کاربر تولید می شوند.ولی در سیستم های پیچیده،طیف ورودی ها و رویدادهای

ممکن(یعنیuse caseها)می تواند بی اندازه گسترده باشد.چه زیر مجموعه ای ازuse caseها به طور مناسب،رفتار برنامه را وارسی می کند؟

این نخستین پرسشی است که آزمون کاربرد آماری باید به آن بپردازد.

آزمون کاربرد آماری«در کل عبارت است از آزمودن نرم افزار به شیوه ای که کاربران تمایل به استفاده از آن دارند»[Lin94b].تیم های آزمون اتاق تمیز(که تیم های تایید نیز نامیده می شوند)برای نیل به این مقصود،باید یک توزیع احتمال کاربرد برای نرم افزار تعیین کنند.مشخصات(چهارگوش سیاه)برای هر نسخه از نرم افزار تحلیل می شود تا مجموعه ای از محرک ها(رویدادها یا ورودی ها)که باعث تغییر رفتار نرم افزار می شوند،تعریف شوند.

صدور گواهی(Certification)

در حیطه ی مهندسی نرم افزار اتاق تمیز،تایید به این معناست که قابلیت اطمینان را [که توسط میانگین شکست(MTTF)سنجیده می شود]برای هر یک از مولفه ها می توان مشخص کرد.

رویکرد تایید شامل پنج مرحله می شود[Woh94](1)سناریوهای کاربرد باید ایجاد شوند،(2)پروفایل کاربرد مشخص می شود،(3)موارد آزمون از روی پروفایل ایجاد می شوند،(4)آزمون ها اجرا و داده های مربوط به شکست ها،ثبت و تحلیل می شوند.(5)قابلیت اطمینان،محاسبه و تایید می شود.مراحل 1تا4 را در بخش قبل مورد بحث قرار دادیم.تایید برای مهندسی نرم افزار اتاق تمیز به ایجاد سه مدل نیاز دارد[Poo93]:

مدل نمونه برداری.آزمون نرم افزار،mمورد آزمون تصادفی را اجرا می کند و اگر هیچ شکستی مشاهده نشود یا تعداد مشخصی از خطا رخ دهد،تایید انجام می شود.مقدار mبه روش ریاضی به دست می آید تا اطمینان حاصل شود که قابلیت اطمینان لازم وجود دارد.

مدل مولفه ها.سیستمی متشکل ازnمولفه باید تایید شود.مدل مولفه ها به تحلیل گر این امکان را می دهد تا احتمال شکست مولفه ی iقبل از کامل شدن سیستم را تعیین کند.

مدل تایید.قابلیت اطمینان کل سیستم،پیش بینی می شود و به تایید می رسد.

مفاهیم روش های رسمی

در فرهنگ بزرگ مهندسی نرم افزار [Mar01]،روش های رسمی به شیوه زیر تعریف می شوند:

روش های رسمی به کار رفته در توسعه ی سیستم های کامپیوتری،تکنیک هایی با اساس و پایه ریاضی برای توصیف خواص سیستم هستند.این روش های رسمی،چارچوب هایی فراهم می آورند که از طریق آن ها می توان سیستم ها را به شیوه ای سیستماتیک و پیش بینی شده،مشخص کرد،توسعه داد و وارسی کرد.

خواص مطلوب یک مشخصه ی رسمی-سازگاری،کامل بودن و فقدان ابهام-اهداف همه ی روش های تعیین مشخصات هستند.به هر حال،زبان به کار رفته برای تعیین مشخصات در روش های رسمی که پایه ای ریاضی دارند،احتمال دستیابی به این خواص را به مراتب افزایش می دهد.قالب نحوی رسمی یک زبان تعیین مشخصات (بخش7-21)تفسیر خواسته ها یا طراحی را تنها در یک جهت میسر می سازد یعنی در جهت حذف ابهامی که غالباً هنگام تفسیر یک زبان طبیعی (مثلاً فارسی)یا نماد گذاری گرافیکی(مثلاًUML)توسط خواننده رخ می دهد.به کمک امکانات توصیفی نظریه مجموعه ها و نماد گذاری منطقی،می توان خواسته ها را به وضوح و روشنی بیان کرد.برای سازگاری،خواسته های بیان شده در یک نقطه از مشخصات نباید با خواسته هایی در جای دیگر در تناقض باشند.سازگاری با اثبات ریاضی این نکته محقق می شود که حقایق اولیه را می توان به طور رسمی (با استفاده از قواعد استنباط)در گزاره های بعدی موجود در مشخصات،تصویر کرد.برای معرفی مفاهیم روش های رسمی،به بررسی چند مثال ساده برای روشن شدن کاربرد مشخصه ی ریاضی می پردازیم بی آنکه خود را بیش از حد غرق جزئیات ریاضی کنیم.

استفاده از نمادگذاری ریاضی برای مشخصه ی رسمی

برای نمایش کاربرد نماد گذاری ریاضی در مشخصه ی رسمی یک مولفه ی نرم افزار،به همان مثال مدیریت بلوک ها خواهیم پرداخت که در بخش5-21 بحث شد.گفتیم که مولفه ی مهمی از سیستم عامل،فایل های ایجاد شده توسط کاربران را نگهداری می کند.مدیریت بلوک ها،مخزنی از بلوک های استفاده نشده را نگهداری می کند و در عین حال حساب بلوک هایی را هم دارد که در حال حاضر مورد استفاده هستند.هنگامی که بلوک ها از یک فایل حذف شده آزاد شدند،معمولاً به صف بلوک هایی افزوده می شوند که در انتظارند تا به مخزن بلوک های استفاده نشده اضافه شوند.طرحی از این فرایند در شکل8-21ارائه شده است.

مجموعه ای با نامBLOCKSشامل شماره ی تمامی بلوک ها می شود.ALLBlocks مجموعه ای از بلوک هاست که بین 1تاMaxBlocksقرار می گیرد.حالت توسط دو مجموعه و یک دنباله مدل سازی می شود.این دو مجموعه عبارتند ازusedوfreeهر دو مجموعه حاوی بلوک هستند-مجموعه ی usedحاوی بلوک هایی است که هم اکنون در فایل استفاده شده اند و مجموعه ی free حاوی بلوک هایی است که برای فایل های جدید در دسترس اند.دنباله حاوی مجموعه ای از بلوک هایی خواهد بود که آماده ی آزادشدن از فایل های حذف شده اند.حالت را می توان به صورت زیر توصیف کرد:

Used,free: P BLOCKS
BlockQueue:seq P BLOCKS

این توصیف،شباهت بسیار به اعلان متغیرهای برنامه دارد.و بیان می کند کهusedوfreeمجموعه ی بلوک ها خواهند بود وBlockQueueیک دنباله است که هر عنصر از آن،مجموعه ای از بلوک هاست.ثابت داده ای را می توان به صورت زیر نوشت:

used∩free=∅
used∩free=AllBlocks
∀i:dom BlockQueue0BlockQueuei used
∀I,j:dom BlockQueue0j≠j= BlockQueue i∩ BlockQueue j=∅

زبان های تعیین مشخصات رسمی

زبان های تعیین مشخصات رسمی معمولاً از سه مولفه اصلی تشکیل می شود:(1)یک قالب نحوی که تعیین می کند مشخصات با چه نماد گذاری خاصی باید ارائه شود،(2)معنا شناختی برای کمک به تعریف«مجموعه اشیاء مرجع»[Win90]که برای توصیف سیستم به کار گرفته خواهد شد و (3)مجموعه ای از روابط که قواعدی را تعریف می کنند که نشان می دهند کدام اشیا به طور مناسب در مشخصه صدق می کنند.

مجموعه متنوعی از زبان های تعیین مشخصات رسمی هم اکنون در حال استفاده است.

OCL[OMG03b]،Z[ISO02]،LARCH[Gu93]وVDM[Jon91]چند نمونه از زبان های تعیین مشخصات رسمی اند.

زبان قید و بند اشیا(OCL)

زبان قید و بند اشیا(OCL)یک نماد گذاری رسمی است که طوری توسعه یافته است که کاربران UMLبتوانند دقت بیشتری به مشخصات خود بیفزایند.همه ی قدرت منطق و ریاضیات گسسسته،در این زبان در دسترس قرار دارد.ولی،طراحان،(OCL) تصمیم گرفته اند که در گزاره های (OCL)فقط از کاراکترهایASCII(به جای نماد گذاری سنتی ریاضی)استفاده شود.این باعث می شود که زبان مذکور نزد افرادی که میانه ی چندان خوبی با ریاضیات ندارند،ظاهری دوست داشتنی تر بگیرد و کامپیوتر راحت تر بتواند آن را پردازش کند.ولی این باعث می شود که (OCL) گاهی ظاهر کلامی به خود بگیرد.

خلاصه ای از نمادگذاری (OCL)

OCLعملیات های توکاری را فراهم می سازد که عملگرهای منطقی و مجموعه ها،مشخصات سازنده و ریاضیات وابسته به این مباحث را پیاده سازی می کنند.

عبارت های OLC که نمودار کلاس ها را تکمیل می کنند،متناظر با شش بخش از ثابت ها هستند.

1-هیچ بلوکی هم به عنوان استفاده شده و هم به عنوان استفاده نشده علامت زده نمی شود.

Context BlockHandler inv:

(self.used->intersection(self.free))->isEmpty()

توجه دارید که هر عبارت با واژه کلیدی context آغاز می شود.این واژه نشان گر عنصری از نمودارUMLاست که عبارت بر آن قید و بند می گذارد.به طریق دیگر،می توانید قید و بند را مستقیماً روی نمودارUML بگذارید و آن را با آکلاد{}محصور کنید.واژه کلیدی self در اینجا به نمونه ای از BlockHandlerاشاره دارد؛در مورد بعدی،آن گونه که درOCLرواست،selfرا حذف خواهیم کرد.

2-همه ی مجموعه بلوک های موجود در صف،زیر مجموعه هایی هستند از مجموعه بلوک هایی که در حال حاضر مورد استفاده اند.

Context BlockHandler inv:
blockQueue->forAll(aBlockset|used->includesAll(aBlockset))

3-هیچ عنصری از صف حاوی تعداد بلوک های یکسان نیست.

Context BlockHandler inv:
blockQueue->forAll(aBlockset1, Blockset2|
Blockset1<>Blockset2 implies
Blockset1.elements.number->excludesAll(Blockset1, Blockset2))

عبارت قبل از implies لازم است تا اطمینان حاصل شود که از جفت های حاوی دو بلوک یکسان چشم پوشی می شود.

4-مجموعه بلوک های استفاده شده و بلوک هایی که استفاده نشده اند برابر با مجموعه کل بلوک های تشکیل دهنده فایل ها خواهد بود.

Context BlockHandler inv:
allBlock=used->union(free)

5-مجموعه بلوک های استفاده نشده فاقد بلوک های تکراری خواهد بود.

Context BlockHandler inv:
Free->isUnique(aBlocks| aBlocks.number)

6-مجموعه بلوک های استفاده شده فاقد بلوک های تکراری خواهد بود.

Context BlockHandler inv:
Used->isUnique(aBlock|aBlocks.number)

زبان تعیین مشخصات Z

Z(با تلفظ زِد)یک زبان تعیین مشخصات است که در جامعه ی روش های رسمی،کاربردی گسترده دارد.در زبانZ مجموعه های نوع دار،رابطه ها و وظایف در حیطه ی منطق گزاره ای مرتبه ی اول به کار برده می شوند تاشِماها(schema)را بسازند-شِما ابزاری برای ساختاردهی به مشخصه ی رسمی است.مشصخاتZ به صورت مجموعه ای از شِماها سازماندهی می شوند-زبانی ساختاری که متغیرها را معرفی کرده واسط میان این متغیرها را مشخص می سازد.شِما اساساً همان مولفه ی زبان برنامه نویسی در مشخصه ی رسمی است.از شِماها در ساختاردهی به مشخصه رسمی استفاده می شود،درست همان گونه که مولفه ها در ساختاردهی به سیستم به کار می روند.

یک شِما داده های انبار شده ای را توصیف می کند که سیستم به آن ها دستیابی دارد و آن ها را تغییر می دهد.در حیطه یZ ،این را«حالت»می نامند.این کاربرد واژه ی حالت در Zقدری با کاربرد آن در بقیه ی کتاب تفاوت دارد.به علاوه،شِما،عملیات هایی را توصیف می کند که برای تغییر دادن حالت و روابطی به کار می روند که در داخل سیستم رخ می دهند.

که اعلان ها، متغیر های تشکیل دهنده حالت سیستم را معین می کنند و ثابت ها،قید و بندهای حاکم بر شیوه ی تکامل حالت را تعیین می کنند.

جدول خلاصه نماد گذاریZ

نمادگذاریZ مبتنی بر نظریه ی مجموعه های نوع دار و منطق گزاره ای مرتبه ی اول است.Z ،ساختاری به نام شِما رائه می دهند که از آن برای توصیف فضای حالت و عملیات های مشخصه استفاده می شود.در زبان Z،شِمای X به شکل زیر مشخص می شود.

فصل 22 : مدیریت پیکر بندی نرم افزار

زمانی که یک نرم افزار کامپیوتری ساخته می شود تغییراتی در آن رخ داده می شود که در جهت کنترل این تغییرات از مدیریت پیکر بندی نرم افزار یا (SCM) که یک فعالیت چتری است، استفاده می شود. که SCM وظیفه ی شناسایی ، کنترل ، ممیزی و گزارش اصلاحات است.

مدیریت پیکر بندی نرم افزار

خروجی فرآیند نرم افزار ، اطلاعاتی است که به سه گروه عمده قابل تقسیم است.

که به 1) برنامه های کامپیوتری 2) مستندات 3) داده ها

که در مجموع پیکربندی نرم افزار نام دارند

باپیشرفت فرآیند نرم افزار، تعداد آیتم های پیکر بندی نرم افزار (SCIها) به سرعت رشد می کند . اگر هر SCI فقط شامل SCI های دیگر می شد مشکلی وجود نداشت . عامل دیگری به عنان تغییر در فرآیند دخالت کند. SCI (مجموعه ای از اشیای مرتبط به هم را تشکیل می دهد.)

چهار منبع اصلی برای این تغییرات ذکر شده است. 1- شرایط بازاری ، تجاری جدید

2- نیازهای جدید ذی نفع که اصلاحاتی را در داده تولید شده توسط سیستم های اطلاعاتی طلب می کنند.

3- سازماندهی مجدد یا رشد 4- قید و بندهای بودجه ای یا زمان بندی که باعث تعریف مجدد محصول می شود SCM یک فعالیت تضمین کیفیت نرم افزار است.

سناریوی SCM

یک سناریوی CM شامل اجزا زیر است.

1- مدیر پروژه : که مسئولیت گروه نرم افزار رو به عهده دارد. هدف مدیر حصول اطمینان از اینکه محصول در یک چارچوب زمانی معین توسعه می یابد. از این رو مدیر مشکلات را تشخیص می دهد. این کار با ایجاد و تحلیل گزارش در خصوص وضعیت سیستم نرم افزار انجام می شود.

2- مدیر پیکربندی: که مسئولیت روال ها و خط مشی های CM رو بر عهده دارد.
که این روال ها حتماً رعایت شود این مدیر ساز و کارهایی را برای مجاز ساختن تغییرات معرفی می کند .

3- مهندسان نرم افزار: که مسئول تولید و نگهداری نرم افزار هستند که هدف آنان کار کردن اثر بخش است. مهندسان برای ایجاد تغییر، آزمون و منسجم ساختن کدها از فضای کاری خاصی خو را خواهند داشت در نقطه ای معین از کدها یک خط مبنا ساخته می شود که توسعه بیش تر بر اساس آن خط مبنا است.

4- مشتری: که از محصول استفاده می کن. مشتی روال های سمی مربوط به تغییرات در محصول دنبال می کنند.

عناصر سیستم مدیریت پیکر بندی

4 عنصر مهم وجود دارد.

1- عناصر مولفه ای: مجموعه ای از ابزار های نهاد شده در داخل یک سیستم مدیریت فایل

2- عناصر پردازشی: مجموعه ای از کنش ها و وظایف که رویکردی اثر بخش برای تغییر دادن فعالیت ها دارد.

3- عناصر ساختمانی: مجموعه ای از ابزار ها که ساخت خود کار نرم افزار را امکان پذیر می نماید.

4- عناصر انسانی : مجموعه ای از ابزار ها و ویژگی های پردازشی که مهندس نرم افزار به کارمی گیرد.

خط مبنا (Baseline)

هنگامی که یک شی پیکر بندی توسعه یافت و مرور شد به خط مبنا تبدیل می شود. تغییراتی که در یک خط مبنا اعمال می شود، منجر به ایجاد نسخه جدیدی از آن شی می شود.

خط مبنا یک نقطه عطف در توسعه نرم افزار است.

پیشرفت رویدادها که منجر به تشکیل یک خط مبنا می شود در شکل زیر آمده است.

وظیفه ی مهندس نرم افزار این است که یک یا چند SCI را تولید کرده و در بانک اطلاعاتی نگهداری کند. هنگامی که یک عضو از تیم یکی از SCI های خط مبنا را اصلاح کند آن را از بانک اطلاعاتی به فضای کاری خود اختصاص داده است.

4-1-22 آیتم های پیکر بندی نرم افزار

آیتم های پیکر بندی نرم افزار را پیش از این به عنوان اطلاعاتی تعریف کردیم که به عنوان بخشی از فرآیند مهندسی نرم افزار ایجاد می شود.

در حالتی واقع بینانه تر ، SCI یک سند، مجموعه کاملی از موارد آزمون یا یک قطعه برنامه است.

بسیاری از سازمان های مهندسی نرم افزار ، علاوه بر SCI های به دست آمده از محصولات کاری نرم افزار، ابزار های نرم افزاری تحت کنترل پیکر بندی را قار می دهند .

ابزار ها در تولید مستندات ، داده ها و کدهای منبع به کار می روند از این رو می توانند به عنوان خط مبنا برای بخشی از یک فرآیند جامع پیکر بندی عمل کند.

در واقع SCI ها برای تشکیل اشیای پیکربندی سازماندهی می شوند.

در نخستین روزهای مهندسی نرم افزار آیتم های پیکر بندی نرم افزار به صورت مستندا کاغذی در کابینت های فلزی نگهداری می شد که به 4 دلیل مشکل آفرین بود 1- یافتن یک آیتم دشوار بود 2- تعیین اینکه کدام آیتم چه هنگام و توسط چه کسی تغییر کرده ، ایجاد چالش می کرد. 3- ایجاد نسخه ی جدید از یک برنامه وقت گیر و در معرض خطا بود 4- توصیف روابط غیر ممکن بود ولی امروزه ، SCI ها در بانک اطلاعاتی یا مخزن نگهداری می شوند.

مخزن ←« محلی برای انباشتن یا ذخیره سازی » تعریف می شود.

در روزهای اولیه از برنامه نویس استفاده می شد که برای به خاطر سپردن و باز سازی اطلاعات کار آمد نبود و بعد از یک چیز به نام مخزن که بک بانک اطلاعات است استفاده کردند.

نقش مخزن

مخزن SCM ، مجموعه ای از سازو کارها و ساختمان داده هاست که تیم نرم افزاری رو به صورت اثر بخش مدیریت می کند. مخزن SCM به عنوان مرکزی برای انسجام بخشیدن ، به ابزار های نرم افزاری عمل می کند. که سبب یکنواختی و فرمت بندی محصولات نرم ازاری می شود.

برای دستیابی به این قابلیت ها مخزن بر حسب یک شبه مدل عمل می کند.

که چگونگی دسترسی، نگهداری و مشاهده و حفظ امنیت به همراه پاسخ گویی به نیازهای جدید را تعیین می کند.

محتوا و ویژگی های عمومی مخزن

ویژگی و محتویات مخزن رو از دو دیدگاه می توان شناخت.

1- آن چه که قرار است در مخزن انباشته شود.

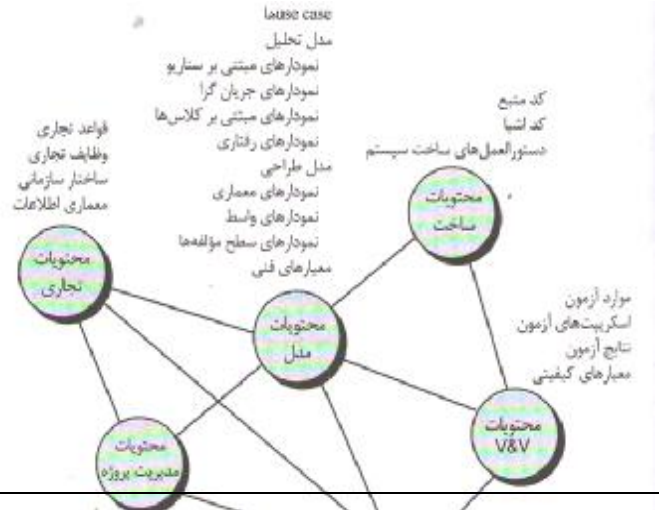
2- خدمات ویژه ای که توسط مخزن فراهم خواهد آمد.

یک مخزن پر قدرت دو دسته خدمات ارائه می دهد.

1- انواع خدمات مورد انتظار از هر سیستم

2- خدمات مختص

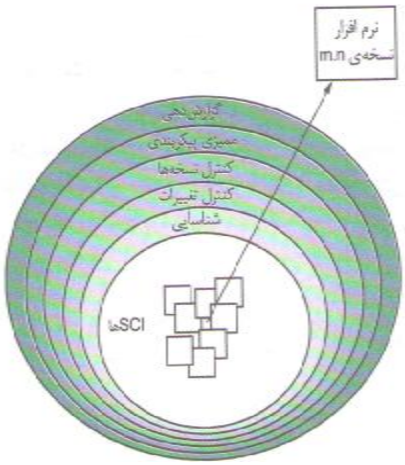
مخزن که به تیم مهندسی نرم افزار خدمات می دهد هم چنین باید:



1- قابلیت مدیریت منسجم یا مستقیماً پشتیبانی کند.
2- قواعد ویژه حاکم بر مخزن رو پشتیبانی کند.
3- فراهم ساختن واسطی با سایر ابزار های مهندسی نرم افزار.
4- ذخیره سازی اشیاء داده ای پیچیده شکل 3-22 محتوای مخزن

ویژگی های SCM

برای پشتیبانی SCM ، مخزن باید مجموعه ابزارهایی داشته باشد که ویژگی ها ی زیر را پشتیبانی کند.
ایجاد نسخه ها : به موازات پیش رفتن پروژه و وجود نسخه های فراوان ، مخزن توانایی ذخیره نسخه ها رو داشته باشد.
مدیریت تغییرات و رد گیری وابستگی ها : مخزن گستره ی وسیعی از روابط در خود مدیریت می کند . این روابط میان فرآیند ها ، موجودت ها مولفه های طراحی و غیره است. که این روابط از دو نوع وابستگی و اجباری هستند. توانایی رد گیری روابط و ذخیره ی اطلاعات در مخزن و محصولات قابل تحویل ، اهمیت حیاتی دارد. روگیری خواسته ها: این وظیفه ی خاص به مدیریت پیوندها مربوط می شود. و توانایی رد گیری مولفه های طاحی ، ساخت و محصولات قابل تحویل را فراهم می سازد.
مدیریت پیکر بندی : یک تسهیلات مدیریت پیکر بندی، اطلاعات پیکر بندی را نگهداری می کند.
جلسات ممیزی : در جلسه ممیزی ، اطلاعات اضافی در بازه ی زمان ، علت و عامل انجام دهنده ی تغییرات فراهم می آید.
اطلاعات مربوط به مخزن را می توان به عنوان صفت وارد مخزن کرد.



فرآیند SCM

فرآیند SCM 4 هدف اصلی را دنبال می کند . 1- شناسایی همه ی آیتم ها ، که پیکر بندی نرم افزار را تعریف می کند. 2- مدیریت تغییرات به عمل آمده در آیتم ها 3- تسهیل در ایجاد نسخه های متفاوتی از یک برنامه کاربردی 4- حصول اطمینان از اینکه کیفیت نرم افزار با تکامل پیکر بندی به مرور زمان حفظ می شود.
می توان وظایف SCM ا به صورت لایه ای در نظر گفت که لایه ها هم مرکز بوده و در سرتاسر عمر خود مفید هستند.
هرچه لایه ها به سمت بیرون می آید به یک یا چند نسخه از یک سیستم یا برنامه کاربردی تبدیل می شود. شکل 4-22 لایه های فرآیند SCM

شناسایی اشیاء در پیکر بندی نرم افزار

برای کنترل و اداره ی آیتم های پیکر بندی نرم افزار باید هر کدام از آنها را جداگانه نام گذاری و با استفاده از روشی شیء گرا سازماندهی کرد.
دو نوع اشیاء قابل شناسایی است.
1- شی ء پایه : که یک واحد متنی است. در راستای طراحی تحلیل و کد نویسی و آزمون

2- شی ء مرکب: مجموعه ای از اشیاء پایه و مرکب است. برای مثال Designpecifiction شی مرکب است. که آن سو به عنوان فهرست در نظر گرفت .

Data Modelو ARCHitectaral Mode اشیاء مرکب هستند.

Componentnو UML classDiagramn اشیاء پایه هستند.

هر شیء دارای مجموعه ای از ویژگی هاست:

نام شیء یک رشته کاراکتری که شیء رو بدون هیچ ابهامی مشخص می کند.

توصیف شیء فهرستی از آیتمهای داده ای است.

شناسه پروژه اطلاعات مربوط به تغییر

صنایع موجودیت هایی که توسط شیء فراهم، پردازش و ارجاع داده می شوند که مورد نیاز شیء است

کنترل نسخه ها (Version control)

سیستم کنترل نسخه ها 4 قابلیت اصلی دارد.

1- بانک اطلاعاتی 2- قابلیت مدیریت نسخه ها 3- تسهیات ساخت و ساز که به کمک آن می توانید همه ی اشیاء پیکر بندی را جمع آوری و نسخه ء خاص از نرم افزار را ایجاد کند.

4 - قابلیت رد گیری مسائل

چند سیستم کنترل نسخ یک مجموعه ی تغییرات را تشکیل می دهند

برای هر سیستم یا برنامه ی کاربردی ، ممکن است چند مجموعه تغییرات با نام ها مشخص شناسایی شود. ه این ترتیب می توان نسخه ای از نرم افزار را با مشخص کردن چند مجموعه تغییرات ایجاد کرد. که باید در پیکر بندی خط مبنا اعمال کرد. برای دستیابی به این هدف از رویکرد «مدلسازی سیستمی » استفاده می شود که حاوی موارد زیر است.

1) قالبی که شامل یک سلسله مراتب از مولفه ها و یک (سفارش ساخت) برای مولفه هایی می شود که شرح می دهد ، سلسله مراتب باید چگونه ایجاد شود.
2) قواع ساخت
3) قواعد وارسی.

کنترل تغییرات

واقعیت کنترل تغییرات را جیمز بک به طور خلاصه بیان می کند که :

«باید متعادل رفتار کنیم، اگر بیش از حد به کنترل تغییرات بپردازیم، مشکل ایجاد می کند و اگر کم باشد مشکلات دیگری ایجاد می کند.»

در پروژه های بزرگ مهندسی ، رویه های بشری و ابزا های خودکار را باهم ترکیب کرده ساز و کاری برای کنترل تغییرات فراهم می اورند.

درخواست تغییر به یک گارش تبدیل می شود و سپس مورد ارزیابی قرار گرفته که مورد استفاده ی مسئول کنترل تغییر (CCA) قرار می گیرد – یعنی شخصی که درباره ی تغییرات تصمیمات نهایی می گیرد- برای هر تغییر یک سفارش تغییر مهندسی (ECO) تولید می شود. تغییرات اعمال شده – شرایط جدی که باید رعایت شود – ملاک ها مرور و ممیزی را توصیف می کند.

اشیاء تغییر داده شده در دایرکتوری قرار داده می شود که فقط توسط مهندسی نرم افزار قابل کنترل است.

سیستم کنترل نسخه ، پس از اعمال تغییر ،فایل را بهنگام سازی می کند.

به عنوان یک راه حل شی ء هایی که باید تغییر داده شوند از بانک (مخزن) خارج شده و فعالیت های SQA مناسب اعمال شده و دوباره وارد مخزن می شود.

ساز و کارهای کنترل نسخه دو عنصر مهم کنترل تغییرات را پیاده سازی می کند.

1. کنترل دستیابی: کدام مهندسان اجازه دستیابی به یک شیء و اصلاح آن را دارند.

2. کنترل همگام سازی : اطمینان حاصل کرد که تغییرات موازی و انجام شده دو نفر متفاوت، روی یکدیگر نوشته نمی شوند. شکل 22-5 فرآیند کنترل تغییرات

هنگامی که شیء دستخوش تغییرات شد و تصویب شد به خط مبنا تبدیل می شود. هنگامی که SCI به خط مبنا تبدیل شد کنترل تغییر سطح پروژه به اجرا در می آید.

پس از اعمال تغییرات باید به تصویب نرم ازا نویس یا CCA برسد.

CCA در لایهء دوم و سوم کنترل نقش مهمی دارد.

CCA ممکن است از یک یا چند نفر تشکیل شود.

هدف CCA ممکن است از یک یا چند نفر تشکیل شود.

هدف CCA به دست آوردن یک دیدکلی و سرتاسری است. یعنی ارزیابی تاثیر تغییرات در ورای SCI مورد نظر.

ممیزی پیکر بندی (Conflguration Audit)

شناسایی، کنترل نسخه و تغییرات به نظم امور کمک می کند

موثر ترین کنترل ها یک تغییر را تا جایی پیکر بندی می کنند که یک Eco تولید شود.

در جهت اینکه تغییر به طور مناسب پیاده سازی شده است دو وجه دارد.

1. مرور های فنی رسمی
2. ممیزی پیکر بندی نرم افزار .

1- بردرستی فنی شیء پیکر بندی اصاح شده تاکید دارد . مسئولان SCI را مورد سنجش قرار می دهند همه ی تغییرات باید مرور فنی رسمی شوند.

2- ممیزی پیکر بندی کسری از سوالات است. در جهت اینکه آیا تغییر SCI برجسته است؟

روال ها SCM دنبال شده است؟ SCI مربوط به هنگام سازی و نگهداری شده است؟ گاهی اوقات این پرسش ها به عنوان بخشی از مرور فنی و رسمی است اما اگر فعالیت رسمی باشد به طور جداگانه بررسی می شود.

گزارش وضعیت

یک فعالیت SCM است 1- چه زمانی اتفاق افتاده 2 - چه کسی سبب آن است؟ 3-چه اتفاقی رخ داده 4-چه چیزهایی تحت تاثیر قرار گرفته است

هر بار که یک SCI بازسازی می شود یک مدخل CSR ساخته می شود که تغییری به تصویب CCA می رسد یک پیمانه CSR ساخته می شود.

هر بار که یک ممیزی انجام می شود نتایج به عنوان بخشی از وظیفه ی CSR گزارش می شوند خروجی یک CSR را می توان در یک بانک اطلاعاتی آنلاین نهداری کرد که با واژه ها کلیدی می توان به آن دسترسی داشت.

22-4 مدیریت پیکر بندی برای برنامه های تحت وب

خصوصیاتی که برنامه های تحت وب را از نرم افزار های سنتی متمایز می سازند، ماهیت همه گیری تغییر است.

سازندگان این برنامه ها از یک مدل فرآیندافزایشی استفاده می کنند که نرم افزار چابک را به کار بسته و با این رویکرد این نرم افزار در دوره زمانی بسیار کوتاه و مشتری گرا توسعه می یابد . در نسخه های بعدی قابلیت و محتوا اضافه می شود.

برنامه تحت وب پذیرای تغییرات است.

مسائل غالب

هنگام توسعه ی تاکتیک ها برای مدیریت پیکربندی نرم افزار ، برنامه ها تحت وب، 4- مساله را باید در نظر داشت

1. محتوا: یک برنامه تحت وب آرایه ای گسترده از متون – تصاویر – جداول – فرم ها و بسیاری از موارد دیگر است که چالش پیش را سازماندهی این محتوا درقالب مجموعه ای از اشیای پیکر بندی و سپس ایجاد ساز و کارهای کنترلی پیکر بندی برای

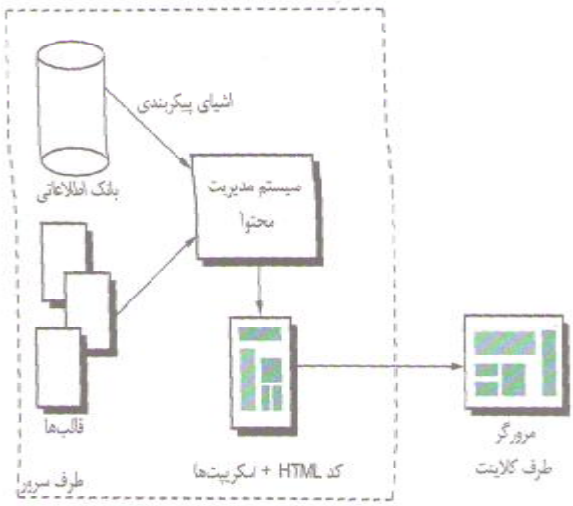
این اشیاست. یک رویکرد برای این منظور ، مدل سازی برنامه تحت وب ، به کارگری تکنیک ها سنتی مدل سازی داده ای و متصل کردن مجموعه ای از خواص تخصصی به هر شی است.

2. افراد: از آن جایی که درصد چشمگیری از برنامه ی تحت وب به شوه ی برنامه ریزی نشده است. افراد در ایجاد محتوا می توانند مبادرت ورزند.
3. گسترش پذیری : تکنیک های به کار رفته در برنامه ها کوچک در برنامه ها بزرگ قابل تعمیم و توسعه نیست. بنابراین دشواری ساز و کارهای کنترل پیکر بندی باید با ابعاد برنامه متناسب باشد.
4. سیاست : چه کسی «مالک» برنامه ی تحت وب است؟ که پاسخ آن تاثیر چشمگیری بر فعالیت های مدیریتی و کنترلی دارد.
5. که چه کسی مسئولیت صحت اطلاعات را بپذیرد؟ چه کسی فرآیند ها ی کنترل را دنبال می کنند. چه کسی مسئولیت اعمال تغییرات و هزنه ها را دارد؟
6. مدیریت پیکر بندی برای برنامه ی تحت وب در حال تکامل است. فرآیند SCM سنتی دشوار است اما ابزار های مدیریت محتوا که برای مهندسی وب طراحی شده اند – اطلاعات را می گیرند تغییرات را مدیریت و سازماندهی می کنند و برای کاربرد نهایی قابل ارائه و قابل ارائه و قابل نمایش در محیط باشد.

اشای پیکربندی برنامه ی تحت وب

برنامه ها تحت وب به گستره وسیعی از اشیای پیکر بندی 1. اشیاء محتوایی (متون – تصاویر و غیره)
2 . مولفه های عملیاتی (اسکریپت ها و اپلت ها) 3. اشیاء واسط (CORBA یا COM) تقسیم می شوند. که برای حصول اطمینان قرار دادهای زیر توصیه می شود.

نام فایل به طول 32 کاراکتر ، از ترکیب درون کوچک و بزرگ ، و همه حروف بزرگ و زیر خط باید پرهیز کرد. در مراجع ، URL (پیوندها) در داخل یک شی پیکربندی باید همواره از مسیرهای نسبی استفاده کرد.



مدیریت محتوا: مدیریت محتوا از این حیث با
مدیریت پیکر بندی در ارتباط است که فرآیندی را وضع
می کند محتوای موجود را می گیرد آنها را سازماندهی و برای کاربر نهایی قابل ارائه می کند و آنها را برای نمایش در اختیار محیط طرف کلانیت قرار می دهد.

رایج ترین کاربرد سیستم مدیریت ، در برنامه ها تحت وب پویا است.

برنامه ها تحت وب اطلاعات خاصی را درخواست می کند آنها را فرمت بندی و برای کاربر عرضه نماید.

CMS محتوا را برای کاربر نهایی با فرا خواندن سه زیر سیستم منسجم «پیکر بندی » می کند. زیر سیستم جمع آوی شامل کلیه کنش ها که برای به دست آوردن محتوا و قابلیت ها ی فنی که برای موارد زیر لازم است.

1. تبدیل محتوا به شکلی قابل ارائه (HTML و XML)
2. سازماندهی محتوا به طور اثر بخش در طرف کلانیت قابل نمایش باشند. این فعالیت خلاقیت و پژوهش را در هم می آمیزد و با ابزار های پشتیبانی می شود که برای برنامه تحت وب قابل استاندار سازی باشد.

محتوای خام باید از اطلاعات بیهوده مبرا باشد.

2- زیر سیستم مدیریت : محتوای موجود باید در مخزن نگهداری شود تا در استفاده ها ی بعدی فهرست بندی و کاتالوگ شده باشد. به طوری که موارد زیر در آن تعریف شده باشد.

1. وضعیت فعلی
2. نسخه ی مناسب شی محتوایی
3. اشیای محتوایی مرتبط

بنابراین زیر سیستم مدیریت مخزنی را پیاده سازی می کند که شامل عناصر زیر است.

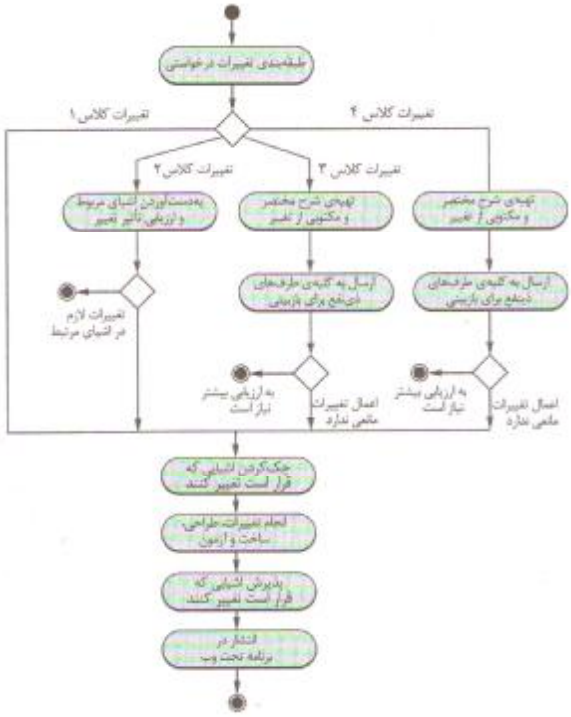
بانک اطلاعات محتوا: ساختار اطلاعاتی برای نگهداری کلمه ی اشیاء

قابلیت های بانک اطلاعاتی: تا اشیاء محتوایی خاص را جستجو، نگهداری و بازیابی کند و ساختار فایل را مدیریت کند.

عملیاتی مدیریت پیکر بندی: عناصر عملیاتی و جریان کاری مرتبط که شناسایی شیء محتوایی – ککنترل نسخه و تغییرات و گزارش دهی را پشتیبانی می کند.

زیر سیتم مدیریت شامل یک قابلیت بهنام شبه داده و قواعدی برای کنترل و پشتیبانی است.

3- زیر سیستم انتشار : محتوای خارج شده از مخزن باید به شکلی مناسب برای انتشا تبدیل شوند که این فرمت ها به مرورگر ها نصب شود در طرف کلانیت قابل انتقال باشد.



زیر سیستم انتشار با به کارگیری یکسری قالب موفق می شود.

که هر قالب با یکی از 3 مولفه متفاوتزیر ساخته می شود.

عناصر انتشار: ستون – گرافیک – رسانه و اسکریپت ها که نیاز به پردازش ندارند مستقیم به کلانیت داده می شود

سرویس های انتشار : خدمات ویژه بازیابی و فرمت بندی را مشخص می کنند و تبدیل داده ها و پیوند های مناسب را می سازند.

سرویس های خارجی : دستیابی به زیر ساخت اطلاعاتی خارجی ، تغییر داده های شرکتی یا برنامه «اتاق پشتی»

یک زیر سیستم مدیریت محتوا که شامل هر 3 زیر سیستم فوق است برا ی پروژه های بزرگ قابل استفاده است.

مدیریت تغییرات

جریان کاری مرتبط با کنترل تغییرات برای نرم افزار های سنتی عموماً برای توسعه برنامه های تحت وب سنگین است.

که هر تغییر باید در قالب یکی از چهار کلاس زیر دسته بندی شود:

کلاس 1: تغییری در محتوا یا عملکرد هایی که با یک خطا در ارتباط است یا محتوا و عملکرد محلی را بهبود می بخشد.

کلاس 2: تغییری در محتواياعملکرد هایی که با سایر مولفه ها عملیاتی یا اشیای محتوایی تاثیر می گذارد

کلاس 3: تغییری در محتوا یاعملکردهایی که تاثیری گسترده ی برنامه ی تحت وب دارند.

کلاس 4: تغییر عمده بر طراحی که بلافاصله نزد یک یا چند گروه از کاربران قابل توجه است.

شکل 7-22 مدیریت تغییرات برای برنامه های تحت وب

در کلاس 1 باید تغییرات را سنجید و نیازی به مرور مستند سازی نیست.

کلاس 2 تاثیر تغییر بر اشیاء مرتبط را مرور می کند.

تغییرات کلاس 3 و 4 به شیوه ای چابک پرداخته و مستند سازی و مرور تقریباً رسمی تر است. برای تغییرات کلاس 3 «شرح تغییر» که توصیی از تغییرات است برگزیده می شود. این شرح در تمام گروه توزیع می شود. در کلاس 4 شرح تغییر تهیه می شود اما توسط افراد ذی نفع مرور انجام می گیرد.

کنترل نسخه ها

به موازاتی که برنامه های تحت وب از طری یکسری نسخه تکامل پیدا کرد چند نسخه ی متفاوت همزمان موجود است . که یکی در حال توسعه دیگری در حال آزمون است. که باید اشیاء پیکر بندی به طور واضح تعریف کرد که هر کدام را با نسخه مناسب مرتبط کرد. در ایلینگ اهمیت کنترل نسخه را بیان کرده است.

فرآیند کنترل نسخه ها

1- ایجاد مخزن مرکزی، که نسخه های خطی و برنامه های تحت وب را در خود نگهداری می کند.
2- ایجاد پوشه کاری توسط مهندس، حاوی ان دسته از اشیاء که ایجاد یا تغییر داده شده است.
3- ساعت روی همه ی ایستگاه های کاری باید با هم تنظیم شده باشد.
4- اشیاء پیکر بندی جدید یا تغییر یافته وارد مخزن مرکزی می شوند.
5- با واردات یا صادرات اشیاء مخزن یک پیام خودکار تهیه می شود که وقایع در ان ثبت می شود. اطلاعات مفیدی برای ممیزی فراهم می آورد و به یک الگوی گزارشی دهی اثر بخش تبدیل می شود.

6-4-22 ممیزی و گزارش دهی

همه ی اشیای که وارد مخزن شده یا از آن خارج می شوند در یک فایل کارنامه ثبت می شوند. که هر زمان می توان آن را کرد. می توان یک گزارش کامل را طوری ایجاد کرد که اعضای تیم به ترتیب زمان تغییرات آن سو در اختیار داشته باشند. می توان یک تذکر خودکار در قالب نامه ی الکترونیکی را می توان با هر بار ورود یک شیء به مخزن یا خروج از آن ارسال کرد.

فصل 23 : تحویل داده نشد.