بسمه تعالی

مهندسی نرم افزار 1

خلاصه درس

تالیف پرسمن

ویراست هفتم

دانشگاه پیام نور مرکز ملایر

مدرس : مهندس زهرا رضایی

نیمسال اول 92 - 91

فصل اول

نگاهی گذرا به نرم افزار

محصول کار از دیدگاه مهندس نرم افزار محصول کار ، برنامه ها ، مستندات ، داده هاست ولی از دیدگاه کاربر محصول کار اطلاعاتی است که به نحوی به درد کاربر میخورند.

ماهیت نرم افزار

امروزه نرم افزار نقشی دو گانه دارد: نرم افزار نوعی محصول است و در عین حال وسیله نقلیه ای برای تحویل یک محصول است.

تعریف نرم افزار(1 - 1 - 1)

نرم افزار عبارت است از(1)دستورالعملها که هنگام اجرا ، ویژگی، عملکرد و کارایی مطلوب را فراهم می سازند(2) ساختمان های دادهایی که برنامه ها را قادر به پردازش مناسب داده ها کنند و(3)اطلاعات توصیفی در هر دو قالب کپی سخت و مجازی که راه اندازی و استفاده از برنامه ها را شرح دهند.

1. نرم افزار بیشتر یک عنصر منطقی است تا یک عنصر سیستمی فیزیکی بنابراین نرم افزار با سخت افزار تفاوت چشمگیری دارد:
2. نرم افزار مهندسی بسط داده می شود و چیزی نیست که به معنای کلاسیک کلمه ساخته شود.
3. در هر دو عمل ، کیفیت بالا از طریق طراحی خوب بدست می آید ولی فاز ساخت برای سخت افزار حاوی مشکلات کیفیتی میشود که برای نرم افزار وجود ندارند (یا به راحتی قابل رفع هستند). هر دو عمل وابسته به انسان هستند ، هر دو عمل مستلزم ساخت یک محصول هستند ولی روش ها متفاوت است.
4. نرم افزار فرسوده نمی شود.

"منحنی وانی" نمودار آهنگ شکست را به صورت تابعی از زمان برای سخت افزار نشان میدهد نشان می دهد.

نرم افزار نسبت به ناملایمات محیطی که باعث فرسایش آن می شود نفوذ پذیر نیست. بنابراین در تئوری منحنی شکست برای نرم افزار باید شکل منحنی ایده آل را به خود بگیرد. منحنی ایده آل نسبت به منحنی واقعی مدل های شکست نرم افزار، بسیار ساده تر است. ولی، معنای آن واضح است ، نرم افزار هیچ گاه دچار فرسایش نمیشود بلکه زوال می یابد!

5. گرچه صنعت در حال حرکت به سوی مونتاژ قطعات است، اکثر نرم افزارها همچنان به صورت سفارشی ساخته می شوند.

در جهان سخت افزار ، استفاده ی مجدد از قطعات بخشی طبیعی از فرایند مهندسی است. در مهندسی نرم افزلر این امر به تازگی مورد توجه قرار گرفته است.

دامنه های کاربرد نرم افزار

هفت گروه وسیع از نرم افزارهای کامپیوتری که امروزه باعث چالش برای مهندسان نرم افزار میشوند:

- نرم افزارهای سیستمی : مجموعه ای از برنامه هاست که باعث که برای سرویس دهی به برنامه های دیگر نوشته شده اند.
- نرم افزارهای کاربردی : برنامه های مستقلی که یک نیاز تجاری مشخص را بر طرف می کند.
- نرم افزارهای مهندسی/علمی : نرم افزارهای علمی توسط الگوریتم هایی مشخص میشوند که اعداد و ارقام را پردازش می کنند. کاربردهای نوین در حیطه ی مهندسی و علمی از الگوریتم های عددی مرسوم فراتر رفته اند.
- نرم افزارهای تعبیه شده : در حافظه فقط خواندنی جای دارند و برای کنترل محصولات و سیستم های مربوط به بازارهای صنعتی و مصرفی به کار می رود.
- نرم افزارهای خط تولید : برای فراهم آوردن یک قابلیت خاص جهت استفاده توسط بسیاری از مشتریان مختلف طراحی میشوند.
- برنامه های کاربردی تحت وب : این گروه از نرم افزارهای شبکه ای شامل مجموعه ی گسترده ای از برنامه های کاربردی می باشد.

- نرم افزارهای هوش مصنوعی : نرم افزارهای هوش مصنوعی (AI) برای حل مسائل پیچیده ای که به روش های عددی قابل حل نیستند، از الگوریتم های غیر عددی استفاده می کنند.

- کار با کامپیتر در جهانی باز : چالشی که مهندسان فرا روی خود خواهند داشت، توسعه ی سیستم ها و برنامه های کاربردی است که با برقراری ارتباط میان کامپیوترهای شخصی دستگاه های همراه و سیستم های اداری را از طریقه شبکه های گسترده میسر می سازند.

- تامین منابع از طریق شبکه : شبکه جهانی وب به سرعت در حال تبدیل به یک موتور کامپیوتری و نیز منبعی برای ارائه اطلاعات است. چالش برای مهندسان نرم افزار، معمازی برنامه های کاربردی ساده (مانند برنامه های مالی شخصی) و پیچیده ای است که بازار های کاربر نهایی هدف را در سرتا سر جهان منتفع سازند.

- کد منبع باز : تمایل رو به رشدی است که منجر به توزیع کدهای منبع سیستم ها و برنامه های کابردی شده است به طوری که افراد بسیاری بتوانند در توسعه آن سهیم شوند.

نرم افزارهای قدیمی

از دهه ی 1960 کانون توجه بودند. دیانی- فاراد و همکاران نرم افزارهای قدیمی را چنین توصیف می کنند:

"سیستم های نرم افزاری قدیمی چند دهه قبل ساخته شده اند و پیوسته اصلاح شده اند تا تغییرات به عمل آمده در خواست های تجاری و سکوهای محاسباتی را پاسخ گو باشند. ازدیاد این گونه سیستم ها باعث درد سر برای سازمان های بزرگی می شود که نگه داری از آن ها را پر هزینه و تکامل بخشیدن به آن ها را خطرناک میدانند."

با گذشت زمان سیستم های قدیمی به یک یا چند دلیل از دلایل زیر تکامل می یابند:

- نرم افزار باید برای برآورده ساختن نیازهای محیط جدید کامپیوتری اصلاح گردد.
- نرم افزار باید بهبود یابد تا خواسته های تجاری جدید را پیاده سازی کند.
- نر م افزار باید گسترش داده شود
- نرم افزار باید دوباره معماری شود تا در یک محیط شبکه نیز قادر به ادامه حیات باشد.

ماهیت منحصر به فرد برنامه های کاربردی تحت وب

امروزه برنامه های تحت وب به ابزارهای کامپیوتری پیچیده ای تکامل یافته اند که نه تنها عملکردی مستقل را در اختیار کاربر نهایی قرار میدهند بلکه با بانک های اطلاعاتی و برنامه های کاربردی تجاری یکی شده اند.

در اکثریت وسیع برنامه های تحت وب صفت های زیر مشاهده می شود:

- میزان تمرکز شبکه : برنامه های تحت وب روی یک شبکه قرار دارند و باید نیازهای جامعه ای متنوع از کلاینت ها را براورده سازند.
- همروندی : ممکن است یک باره تعداد بسیاری از کاربران به برنامه های تحت وب دستیابی داشته باشند.
- بار غیر قابل پیش بینی : تعداد کاربران برنامه های تحت وب از روزی به روز دیگر ده یا صد برابر شوند.
- کارایی : اگر کاربر یک برنامه تحت وب باید یک مدت طولانی منتظر بماند، ممکن است تصمیم بگیرد به جای دیگری برود.
- قابلیت دسترسی : گرچه انتظار 100 در صد قابلیت دسترسی ، غیر منطقی است، کاربران برنامه های تحت وب پر طرفدار غالبا تقاضای دسترسی 24 ساعته در 7 روز هفته و 12 ماه سال را دارند.
- داده محوری : عملکرد اصلی بسیاری از برنامه های تحت وب استفاده از ابر رسانه ها برای ارائه متون به کاربران نهایی است.
- حساس به محتویات : کیفیت و ماهیت زیبا شناختی محتویات از جمله مهمترین عوامل تعیین کننده کیفیت در برنامه های تحت وب است.
- تکامل پیوسته : برخلاف نرم افزارهای کاربردی سنتی برنامه های تحت وب پیوسته در حال تکامل هستند.
- بی واسطگی : نیاز اجباری برای رساندن سریع نرم افزار به بازار است.

- امنیت : برای محافظت از محتویات حساس معیار های امنیتی قوی ای باید پیاده سازی شود.

- زیبایی شناسی

فرایند نرم افزار

فرایند مجموعه ای از فعالیت و کوششی ست در جهت رسیدن به هدفی گسترده

یک کنش شامل مجموعه ای از وظایف است که یک محصول کاری عمده را تولید می کند

در حیطه مهندسی نرم افزار فرایند یک روش انطباق پذیر است که تیم نرم افزار به کمک آن می توانند مجموعه ای مناسب از کنش ها و وظایف کاری را برگزینند

چارچوب فرایند کلی برا مهندسی نرم افزار شامل پنج فعالیت می شود

1. ارتباطات
2. برنامه ریزی
3. مدل سازی
4. ساخت
5. استقرار

فعالیتهای چتری در یک پروژه ی نرم افزاری به تیم نرم افزاری کمک می کند تا پیشرفت ، کیفیت ، تغییر و ریسک را کنترل کند.

فعالیت های چتری عبارتند از :

1. کنترل و پیگیری پروژه های نرم افزاری
2. مدیریت ریسک
3. تضمین کیفیت نرم افزار
4. بازبینی های فنی
5. اندازه گیری
6. مدیریت پیکربندی نرم افزار
7. مدیریت قابلیت استفاده مجدد
8. تهیه و تولید محصول کاری

فصل دوم : مدل های فرایند

فرایند نرم افزاری : کلیه یمراحل قابل پیش بینی که یک نقشه ی کلی برای تولید یک سیستم با کیفیت بالا باید طی شود.

در واقع فرایند نرم افزاری از دیدگاه فنی چارچوبی برای اعمال مورد نیاز جهت ساختن نرم افزار است. فرایند نرم افزار روش مهندسی را مشخص می کند.

چهارچوب کلی مهندسی نرم افزار:1-ارتباطات 2-برنامه ریزی 3:مدل سازی 4:ساخت 5:استقرار

جریان فرایند:شرح میدهد که فعالیت های چتری ، کنش ها و وظایفی که داخل هر فعالیت چارچوبی رخ میدهد از نظر زمانی چگونه سازماندهی شده است.

جریان فرایند خطی: هر کدام از فعالیت های چارچوبی به ترتیب انجام میشود به طوری که با ارطبات اغاز وبه استقرار ختم می شود.

جرایان فرایند مبتنی بر تکرار: پیش از رفتن به تکرار دور بعدی یک یا چند فعالیت تکرار میشود.

جریان فرایند تکاملی: فعالیت ها به شیوه ی حلقوی اجرا می شوند هر مدار از پنج فعالیت عبور می کند که به نسخه ی کاملتری از نر م افزار می انجامد.

جریان فرایند موازی: یک یا چند فرایند به موازات سایر فغعالیت ها انجام می شود.

تعریف یک فعالیت چارچوبی

در تعریف یک فعالیت چارچوبی اگر پروژه یک ذی نفع داشته باشد به مراتب از پیچیدگی کمتری برخوردار است تا چندین ذی نفع داشته باشد.

فعالیت ارتباطات ممکن است شش کنش متمایز داشته باشد:1-شروع 2-استخراج 3:شناخت 4:مذاکره 5:تعیین مشخصات 6:اعتبار سنجی هر کدام از این کنش های مهندسی دارای چندین وظیفه ی کاری و تعدادی محصولات کاری متمایزند.

تعیین مجموعه وظایف:

هر کنش مهندسی نرم افزار را می توان باتعدادی از مجموعه وظایف متفاوت نشان که هر کدام مجموعه ای از وظایف کاری در مهندسی نرم افزار مرتبط با محصولات کاری را انجام می دهد که نقاط تضمین کیفیت و عطف پروژه اند.

الگوی فرایند:

محیطی که درآن مشکل مشاهده شده مشخص می شود.

ارزیابی فرایند بهبودی:

- روش ارزیابی استانداردCMMI برای بهسازی (SCAMPI):

یک مدل ارزیابی فرایند پنج مرحله ای فراهم می اورد که شامل پنج فاز مشود:شروع،عیب یابی،ساخت،عملیات ویادگیری.(سنبل ارزیابی SEL CMMI است)

- ارزیابی مبتنی بر CMM برای بهبود بخشیدن به فرایند های داخلی(CBA3P3):یک تکنیک عیب یابی استکه برای ارزیابی بلوغ نسبی یک سازمان نرم افزاری فراهم مکی اورد.(SEI CMM مبنای ارزیابی است)
- SP/CE(ISO/IEC1550HC) : استانداردی کهمجموعه هی از خواسته ها را برای ارزیابی فرایند نرم افزار تعریف میکند.

مدل های فرایند تجویزی:

مدل های فرایند تجویزی برای نظم بخشیدن به اشوب موجود در توسعه نرم افزار پیشنهاد شدند.

این مدل ها را تجویزی کویند زیرا مجموعه اب از عناصر فرایند-فعالیت های چارچوبی،عملیات مهندسی نرم افزار،وظایف،محصولات کاری،تضمین کیفیت و سازوکارهای کنتل تغییرات را بر پروژه تجویز می کند.

هر مدل فرایند یک جریان فرایند که شیوهی ارتباط میان عناصر فرایند را تعریف کند بیان می کند.

مدل ابشاری:

هر گاغه خواسته های مربوط به یک مساله به خوبی شناخته شده باشند(کار به طریق خطی ازبرقراری ارتباط تا استقرار جریان پیدا کند) این مدل را چرخه ی حیاتی کلاسیک گویند و روشی سینماتیک و ترتیبی برای توسعه ینرم افزار است.

معایب مدل ابشاری:

1:چون تکرار در این مدل غیر مستقیم انجام می شود با پیشرفت پروژه تغییرات باعث سردرگمی می شود

2-همه ی نیاز های مشتری باید به وضوح بیان شود و این کار برای مشتری دشوار است.

3-تا اتمام پروژه مشتری باید حوصله داشته باشد این کار باعث می شود تا عیوب کار تا پایان پنهان بماند.

مدل فرایند افزایشی:

خواسته های اولیه ی نرم افزاربه خوبی تعریف شده اند،ولی حوزه ی کلی تلاش های به عمل امده در توسعه ی نرم افزار مانع از یک فرایند خطی محض می شود. مدل افزایشی، عناصر مدل ترتیبی خطی را با جریان فرایند خطی و موازی تلفیق می کند.

مدل فرایند تکاملی:

غالبا یک مشتری نیاز های کلی خود را بیان میکند و به جزعیات اشاره نمی کند در این موارد ساخت نمونه ی اولیه می تواند بسیار ماثر باشد.

معایب نمونه ی اولیه:

1: افراد ذی نفع ظاهرا یک نمونه ی کاری از نرم افزار را می بینند در حالی کهخ این نمونه با موم سرهم شده است.

2: مهندس نرم افزار برای تولید هر چه سریعتر نمونه ی اولیه از برنامه یا الگوریتم یا...مناسب استفاده نکند واین کار بهعث شود تا در تولید نرم افزار اصلی دچار مشکل شود.

مدل مارپیچی(حلزونی):

یکمدل فرایند نرم افزاری تکاملی است که ماهیت تکراری مدل سشاخت نمونه ی اولیه را با جنبه های کنترولی و سینماتیک مدل ترتیبی خطی تلفیق می کندو بر خلاف سایر مدل های فرایند کلاسیک که با تحویل نرم افزار پایان می یابد،مدل مارپیجی را متوان طوری تطبیق داد که در سراسر عمر نرم افزار قابل به کارگیری باشد.

مدل توسعه ی همروند:

مدل توسعه ی همروند یا مهندسی همروند به تیم نرم افزاری این امکان را می دهد عناصر تکراری و هخمروند هر کدام از مدل های فرایند توصیف شده را ارایه نماید.

سخن اخر درباره ی فرایند های تکاملی:

مشخصه ی اصلی نرم افزار های مدرن ،تغییر پیوسته،در فواصل زمانی فشرده و با تاکید بسیار بر رضایت مشتری است و تا حد زیادی این کار را انجام داده است.

بهخ رغم مزایای غیر قابل انکار نرم افزار های تکاملی،دغدغه هایی نیز وجود دارد:

نخست این که تهییه ی نمونه ی اولیه به دلیله قطعی نبودن تعداد چرخه های ساخت پروژه ایجاد مشکل می کند.

دومفرایند های تکاملی حداکثر سرعت تکامل را تعیین نمی کنند اگر تکامل بیش از حد سرعت داشته باشد، بدون اینکه زمان اسایشی داشته باشد، فرایند به اشوب کشید خواهد شد.

سوم،انعطاف پذیری و بسط پذیری باید بیش از کیفیت مورد توجه قرار گیرد، این تاکید قدری ضریب خطا را بالا می برد.

مدل های فرایند تخصیص یافته:

شامل بسیاری از ویژگی های یک یا چند مدل سنتی بخش های پیش می شود.

تو سعه مبتنی بر مولفه:

فعالیت های مدل سازی و ساخت با شنا سایی مولفه های کاندیدا اغاز می شود. این مولفه ها مکیتوان به صورت پیمانمه های نرم افزاری شی گرا طراحی کرد.

مکدل روش های رسمی:

مجموعه ای از فعالیت ها که به مشخص کردن ریاضی و رسمی نرم افزار کامپیوتری منجر می شود.

مدل روش های رسمی گرچه عمومیت ندارد اما نوید نرم افزاری عاری از نقص را می دهد.

- توسعه ی این مدل ها در حال حاضر وقت گیر است
- به اموزش گسترده نیاز مند است
- ارتباط با مشتری که دید فنی ندارد دشوار است

توسعه ی نرم افزار به روش جنبه گرا

توسعه ی نرم افزار به روش جنبه گرا یا برنامه نویسی جنبه گرا یک الگوی مهندسی نبتا جدید است که رویکردی فرایندی و روش شناختی برای تعریف ،مشخص سازی،طراحی و ساخت جنبه ها ارایه میکند.

فرایند های یک پارچه:

یک فرایند نرم افزاری مبتنی بر معماری، تکرار ، و افزایش است.

مدل های فرایند تیمی و شخصی:

بهترین فرایند نرم افزاری ، فرایندی است که به کسانی که کار می کنند نزدیک باشد.در یک شرایط ایده ال باید شرایطی را ایجاد کرد که به بهترین وجه بر نیازهای شما تطبیق یابد.

فرایند های نرم افزار شخصی:

در مدل فرایند های نرم افزار شخصی پنج فعالیت چارچوبی تعریف می شود:

- برنامه ریزی:خواسته ها شناسایی می شود و براورد منابع و تعیین اندازه ی پروژه انجام می شود.
- طراحی سطح بالا: مشخصات خارجی برای هر کدام از مولفه هایی که قرار است تعیین شود و طراحی مولفه انجام میشود.
- مرور طراحی سطح بالا:روش های وارسی رسمی برای یافتن خطاهایطراحی،اعمال میشوند.
- پایان کار:با استفاده ازمعیار ها وموازین جمع اوری شده،اثر بخشی فرایند تعیین میشود.

فرایند نرم افزار تیمی:

برای انجام پروژه های نرم افزاری در پایه ی صنعتی را تیمی از دست اندر کاران انجام می دهند. هدف فرایند نرم افزار تیمی تشکیل تیم پروزه ی ((خود هدایت گر)) است.

فعالیت های زیر را برای فرایند نرم افزار تیمی تعریف می کنند.

- تشکیل تیم خود هدایت گری که کار خود را برنامه ریزی و پیگیری می کند،اهداف را تعیین می کند و خود به تعیین فرایند و طرح ها اقدام می کنند.
- نشان دادن شیوه ی راهبردی و ایجاد انگیزه در تیم ها به مدیران و چگونگی کمک به انها در حفظ حداکثر کارایی.
- شتاب بخشیدن به بهبود فرایند نرم افزاری.
- فراهم ساختن دستور بهسازی برای سازمان های بالغ.
- تسهیل اموزش دانشگاهی مهارت های تیمی در سطح صنعتی.

فن اوری فرایند :

فناوری فرایند، جهت کمک به سازمان های نرم افزاری در تحلیل فرایند جاری خود،سازماندهی وظایف کاری،کنترل فرایند و نظارت بر ان و مدیریت کیفیت فنی به وحود امده اند.

محصول فرایند:

اگر فرایندی ضعیف باشد بدون شک محصول نهایی نیز ضعیف خواهد بود اما اتکای بیش از حد نیز به فرایند نیز خطرناک است

فصل سوم توسعه چابک

مهندسی نرم افزار چابک تلفیقی از یک فلسفه و مجموعه ای از دستورالعمل های توسعه است. دستورالعمل های توسعه بر تحویل نرم افزار بر اساس تحلیل و طراحی و برقراری ارتباط فعال و پیوسته میان توسعه دهندگاه نرم افزار و مشتریان آن تاکید دارد.

مهندسان نرم افزار و سایر طرف های ذی نفع در پروژه مدیران و مشتریان تیم چابک را تشکیل میدهند.محیط های کاری جدیدی که سیستم های کامپیوتری و محصولات نرم افزاری را تشکیل میدهند پیوسته در حال تغییر است و مهندسی نرم افزار چابک یک جایگزین منطقی برای نرم افزار ها و پروژه های نرم افزاری ارائه میدهد. تنها محصول کاری مهم واقعی یک نرم افزار عملیاتی است که در تاریخ مناسب به مشتری تحویل شود . تغییر هزینه بردار است به ویژه اگر کنترل نشده باشد یا مدیریت ضعیفی برآن اعمال شود. کاهش هزینه ناشی از تغییر از وظایف توسعه چابک است .

چون سازگاری در کنش از نقاط ضعف انسان است روش هایی با انظباط بالا شکننده اند برای کسانی که در کار نرم افزار هستند روش هایی با تحمل راحت تر قابل پذیرش است . ولی ممکن است بهره وری آنها کمتر شود .

برنامه ریزی باید به شیوه ای صورت گیرد که متغیر بودن یک روش توسعه چابک در آن دیده شده باشد همه محصولات کاری به جز موارد ضروری باید حذف شوند.

پاسخگویی به تغییر هنگامی که تیم نرم افزاری مشغول جمع آوری خواسته هاست نسبتا آسان است یک فرآیند چابک با طرحی خوب منحنی هزینه تغییر را تسطیح میکند فرآیند چابک شامل تحویل افزایشی محصول میشود فرآیند چابک باید به گونه ای باشد که مواردی را پیش بینی کند از جمله اینکه کدام خواسته های نرم افزار تغییر میکند و کدام یک ثابت اند. همچنین در بسیاری از نرم افزارها طراحی و ساخت به صورت موازی انجام میشود

فرآیند چابک باید دارای انطباق پذیری باشد و روند انطباق باید به طور افزایشی انجام پذیرد برای دستیابی به انطباق تیم نرم افزاری چابک نیاز به بازخورد از مشتری دارد، نمونه اولیه ای از سیستم عملیاتی عامل شتاب دهنده به بازخورد مشتریان است .

12 اصل در رابطه اصول چابکی وجود دارد:

1: جلب رضایت مشتری به خاطر تحویل زود هنگام و پیوسته نرم افزارهای ارزشمند

2: پذیرا بودن تغییرات درخواسته ها حتی دراواخر فرآیند توسعه

3: تحویل پیوسته نرم افزارهای کاری از دو هفته گرفته تا دوماه

4: دست اندرکاران و افراد تجاری باید در سرتاسر پروژه هرروز باهم کارکنند

5: سپردن پروژه به افراد با انگیزه فراهم سازی محیط و پشتیبانی مورد نیاز آنها و اطمینان کردن به آنها در انجام کارها

6: اثربخش ترین و موثرترین روش انتقال اطلاعات به درون و بیرون تیم توسعه گفتگویی رو در روست

7: نرم افزار کاری میزان اصلی در سنجش پیشرفت است

8: فرآیندهای چابک توسعه پایدار را ارتقاء میبخشد حامیان و سازندگاه و کاربران باید قادر به حفظ سرعت ثابت در پیشرفت کار باشند

9: توجه پیوسته به اعتلای فنی و طراحی خوب باعث بهبود افزایش چابکی میشود

10: سادگی هنر به انجام رساندن کارهایی که انجام نمیشوند ضروری است

11: بهترین معماری های خواسته ها و طراحی ها از تیم های خود سازمان دهی شده ظهور میکند

12: تیم در بازه های منظم بازخوردی از میزان بهبود اثر بخش خود را ارائه میدهد و سپس رفتار خود را مطابق این بازخورد تنظیم میکند

مهندسی نرم افزار عبارتست از ارزیابی و تحلیل و به کارگیری اطلاعاتی که با تیم نرم افزاری تبادل میشود و اعضای تیم باید با یکدیگر همکاری داشته باشند .

توانایی تصمیم گیری: تیم باید اجازه تصمیم گیری برای مسائل فنی و پروژه را داشته باشد.

توانایی حل مساله با منطق فازی .

احترام و اطمینان متقابل:تیم چابک باید به تیم قوام یافته تبدیل شود همانطور که دومارکو و لیستر میگویند.

خود سازمان دهی: به سه معناست :1: تیم چابک خودش را سازمان دهی کند تا کارها به انجام رسد

تیم فرآیند را سازماندهی میکند تا به بهترین تحو در محیط محلی اسکان یابد.

تیم زمانبندی کاری را سازماندهای میکند تا به بهترین نحو تحویل یک نسخه از نرم افزار را امکان پذیر سازد مهمترین مزیت خود سازماندهای بهبود همکاری و تقویت روحیه تیمی است . کن شوابر میگوید تیم است که تصمیم میگیرد چه مقدار کار میتواند در هر دور از تکرار انجام پذیرد و تیم متعهد این کار است

هیچ چیز به این اندازه انگیزه تیم را از بین نمیبرد که آدم دیگری برایش تعیین تکلیف کند

هیچ چیز به اندازه پذیرش مسئولیت برای تعیین وظایف و تعهدات در تیم ایجاد انگیزه نمی کند برنامه نویسی حدی (ایکس پی) پرکاربردترین رویکرد در توسعه ی نرم افزار به روش توسعه چابک است . آی ایکس پی شکل دیگری از ایکس پی است که فرآیند چابک را مشخصا برای استفاده در سازمان های بزرگ هدف قرار داده است

بک مجموعه ای از 5 ارزش را تعریف میکند که بنایی برای کارهای انجام شده در ایکس پی تشکیل میدهند : ارتباطات- سادگی – بازخورد- جرات – احترام

ایکس پی به منظور دستیابی به ارتباطات اثربخش میان مهندسان نرم فزار و سایر طرف های ذی نفع برای به مشورت گذاشتن مفاهیم مهم بازخورد پیوسته و پرهیز از مستندات پر حجم به عنوان واسطه ی ارتباطی تاکید دارد

ایکس پی سازندگان را محدود میکند تا تنها برای نیازهای فوری کار طراحی را انجام دهند نه اینکه همه نیازهای آینده را در نظر بگیرند

بازخورد از سه منبع قابل حصول است : خود نرم افزار مشتری و سایر اعضای تیم نرم افزاری میزان پیاده سازی خروجی عملکرد در رفتار ذکر شده در یک یوز کیس شکلی از بازخورد است با بدست آمدن خواسته های جدید به عنوان بخشی از برنامه ریزی تکراری تیم یک بازخورد سریع از تاثیر زمانبندی و هزینه در اختیار مشتری قرار میدهد.

اکثر تیم های نرم افزاری اقرار میکنند که طراحی برای آینده در دراز مدت به صرفه جویی در زمان و تلاش کمک میکند.دربرنامه نویسی حدی یک روش شی گرا به عنوان الگوی توسعه استفاده میشود . و این فرآیندشامل مجموعه ای از قواعد و اصول میشود که در حیطه 4 فعالیت چارچوبی رخ میدهند: برنامه ریزی و طراحی و کدنویسی و آزمون.

برنامه ریزی : باگوش دادن آغاز میشود و فعالیتی است برای جمع آوری خواسته ها

طراحی : قویا از اصل(کا آی اس) (حفظ سادگی) پیروی میکند در ایکس پی استفاده از کارت های(سی آر سی) سازو کاری موثر برای تفکر درباره نرم افزارهایی شی گرا محسوب میشود کارتهای(سی آر سی) (کلاس – مسئولیت – همکار) کلاس های شی گرایی را شناسایی میکنند که به نسخه فعلی نرم فزار مربوط میشود (سی آر سی) تنها اصول طراحی هستند که به نوان بخشی از فرآیند ایکس پی تولید میشود

نمونه اولیه طراحی راهکار خیزش نامیده میشود هدف آن کاهش خطر هنگام پیاده سازی واقعی و نیز اعتبار سنجی برآوردهای اولیه است

فاولبر بازآرایی را چنین میگوید: تغییر یک سیستم نرم افزاری به شیوه ای که رفتار خارجی را تغییر ندهد و در عین حال ساختار داخلی را بهبود بخشد

مفهوم محوری ایکس پی اینست که طراحی هم قبل و هم بعد از کدنویسی رخ میدهد مفهوم کلیدی ایکس پی که در فعالیت کد نویسی وجود دارد برنامه نویسی جفتی است که دونفر با هم روی یک ایستگاه کاری کار کنند در این روش انجام بخشی پیوسته به پرهیز از مشکلات سازگاری و ایجاد واسط کمک میکند و یک محیط دود آزمون فراهم می آورد برای کشف سریع خطاها،با سازمان دهی آزمون واحد های آزمون انسجام و عتبار سنجی را به صورت روزانه میتوان انجام داد و به این ترتیب تیم ایکس پی در جریان پیشرفت کار قرار خواهد گرفت

امکان سنجی خواسته های تجاری پایه را تعیین میکند .

مطالعه تجاری خواسته های عملیاتی و تجاری را تعیین میکند .

تکرار مدل عملیاتی مجموعه ای از نمونه های اولیه افزایش که عملکرد را برای مشتری به نمایش میگذارند .

تکرار طراحی در ساخت بازبینی نمونه های اولیه ساخته شده است و پیاده سازی قرار دادن آخرین نسخه نرم افزار در محیط کاری است .

پیاده سازی : مدل ها به کد منبع ترجمه میشوند .

آزمون :

استقرار : بر تحویل یک نسخه از نرم افزار و گرفتن بازخورد از کاربران نهایی توجه دارد.

مدیریت پیکربندی و پروژه : مدیریت تغییرات و مدیریت ریسک .

مدیریت محیطی وظیفه ی هماهنگی زیر ساخت فرآیند را بر عهده دارد .

فصل چهارم اصول راهنما در مهندسی نرم افزار

دانش مهندسی نرم افزار

بسیاری از نرم افزار نویسان ،دانش مهندسی نرم افزار را تقریباً به طور انحصاری، اطلاع دشتن از فناوری های خاص می دانند جاوا ، پرل ، windows nt, ,linu html, c++,وغیره

نیمه عمر توسه نرم افزار سه سال است یعنی نیمی از انچه می دانید سه سال دیگر به کارتان نخواهد امد .در دامنه دانش های مرطبط با فناوری، این احتمال درست است، ولی یک نوع دانش در توسعه نرم افزار وجود دارد _نوعی که ما ان را به عنوان «اصول مهندسی نرم افزار »در نظر می گیریم _که نیمه عمر ان سه سال نیست،این اصول مهندسی نرم افزار احتمالاً در سرا سر زندگی کاری یک برنامه نویس حرفه ای به او کمک می کند.

اصول هسته ای

در سطح فرایند، اصول فرایند ، یک بنیاد فلسفی ایجاد می کنندکه تیم نرم افزاری را به هنگام فعالیتهای چارچوبی وچتری هدایت می کندجریان فرایند را مورد کاوش قرار می دهد ومجموعه ای از محصولات کاری مهندسی نرم افزار تولید می کند. در سطح کاری ، اصول مهندسی نرم افزار ، مجموعه ای از ارزشها وقواعد تعیین می کند که شما را در تحلیل یک مساله، طراحی یک راهکارپیاده سازی و ازمون ان راهکار و سرانجام استقرار نرم افزار در جامه کاربری راهنمایی می کند

اصول راهنمای فرایند مهندسی

اصل1.چابک باشید .

اصل2.در هر مرحله کیفیت را در کانون توجه قرار دهید .

اصل3.امادگی انطباق را دشته باشید .

اصل4.تیمی اثر بخش طراحی کنید.

اصل5.ساز وکار هایی برای بر قراری ارطباط و هماهنگی ایجاد کنید .

اصل6.مدیریت تغیرات .

اصل7.ارزیابی ریسک به موازات توسعه نرم افزار، اشتباهات زیادی ممکن است رخ دهد.ارائه طرح اینده نگری ضروری است

اصل8.ایجاد محصولات کاری که برای دیگران ارزش فرهم می کند

اصول راهنمای کار مهندسی نرم افزار

اصل1.تقسیم وحل. یک مساله بزرگ را اگر به مجموعه ای از عناصر تقسیم کنیم حل ان راحت تر است

اصل2.درک بکار گیری انتزاع ها در کار مهندسی نرم افزار از چندین سطح انتزاع استفاده خواهیم کردکه هر کدام معنایی دارد که باید انتقال داده شود، در کار تحلیل و طراحی ، تیم نرم افزاری معمولاً با مدل هایی شروع می کند که نشانگر سطوح بالایی از انتزاع هستند و به اهستگی این مدل ها را به سطوح پایین تری از انتزاع پالایش می کنند

اصل3.تلاش برای سازگاری، سازگاری بیان می کند که یک حیطه اشنا، استفاده از نرم افزار را اسان می کند.

اصل4.توجه ویژه به انتقال اطلاعات ، کار نرم افزار انتقال اطلاعات است. از بانک اطلاعاتی به کاربر نهایی ، از یک سیستم قدیمی به یک برنامه کاربردی تحت وب، از سیستم عامل به یک برنامه واین فهرست تغریباً پایانی ندارد. در هر مورد، اطلاعات از طریق یک واسط جریان پیدا می کند و در نتیجه، فرصت هایی برای خطا، جا افتادگی یا ابهام پیش می اید . بنا به این اصل باید توجه ویژه ای به تحلیل، طراحی، ساخت و ازمون واسط ها مبذول داشت.

اصل5.توسعه نرم افزاری که ساختار پیمانه ای اثر بخش داشته باشد. جدا سازی دغدغه ها (اصل 1) فلسفه ای برای نرم افزار پایه گذاری می کند . ساختار پیمانه ای، ساز و کاری برای تحقق بخشیدن به ای فلسفه فراهم می سازد . هر سیستم پیچیده را می توان به چند پیمانه تقسیم کرد، ساختار پیمانه ای باید اثر بخش هم باشد. یعنی، هر پیمانه باید انحصاراً جنبه ای از سیستم را کانون توجه قرار دهد که قید و بند های ان به خوبی مشخص است _ یعنی از نظر عمکرد باید یکپارچه باشد ویا در حیطه ای که ارائه می دهد ابعادی مشخص داشته باشد به علاوه ارتباط میان پیمانه ها باید به شیوه ای نسبتاً ساده بر قرار شود _ هر پیمانه باید ارتباط اندکی با سایر پیمانه ها، منابع داده ها و سایر جنبه های محیطی داشته باشد.

اصل6.جستجو به دنبال الگوها .هدف الگو ها در جامعه نرم افزاری تهیه نوشتاری است که سازندکان را در حل مشکلات تکراری در سرتاسر فرایند توسعه نرم افزار یاری دهد

اصل7.هر گاه که امکان دارد ، مساله و راهکار ان را از چند دیدگاه متفاوت به نمایش بگذارید. این احتمال که دید بهتری از ان بدست می اید و خطاها و جا افتادگی ها کشف شوند بیشتر خواهد شد.

اصل8. به خاطر دشته باشیدکه نرم افزار را نگهداری خواهید کرد. در دراز مدت نرو افزار با کشف شدن نقایص بهبود خواهد یافت ، خودش را با تغییرات محیط منطبق خواهد ساخت ویا از سوی طرفداران ذی نفع قابلیت های ان ارتقاع خواهد یافت.

این اصول همه ی ان چیزی نیست که برای ساخت نرم افزار های با کیفیت بالا لازم است.

اصول راهنمای فعالیت های چار چوبی

اصول ارتباطی

اصل1. گوش سپردن .تلاش کنید به سخنان گوینده گوش فرا دهید، نه انکه در ان اثنا به فکر اماده کردن پاسخ خود باشید، اگر چیزی برایتان واضح نیست از گوینده بخواهید تا منظور خود را به وضوح بیان کند

اصل2.خود راقبل از برقراری ارتباط اماده کنید. پیش از ملاقات با دیگران، قدری وقت برای دانستن مساله صرف کنید.

اصل3.یک نفر باید این فعالیت را تسهیل کند. هر جلسه ارتباطی باید دارای رهبر (تسهیل گری)باشد که (1) مکالمه را در جهتی پیش ببرد که بهره وری داشته باشد. (2) هر گونه تقابلی را که رخ می دهد میانجی گری کند. (3) اطمینان حاصل کند که اصول دیگر رعایت می شود.

اصل4.بهترین راه، ارتباط رودرروی است.

اصل5.یاد داشت بردارید و تصمیم گیری را مستند کنید . احتمال اینکه چیز ها فراموش شوند یا نادیده انگاشته شوند ، زیاد است. یکی از حاضران در جلسه باید به عنوان «منشی» عمل کند و نکات و تصمیم گیری های مهم را یاد داشت کند.

اصل6.تلاش برای همکاری.هر همکاری کوچک، به بالا بردن اطمینان در اعضای تیم و ایجاد هدفی مشترک برای تیم کمک می کند.

اصل7.توجه خود را معطوف کنید بحث خود را پیمانه ای کنید.هر چه تعداد افراد حاضر در یک ارتباط بیشتر باشد، احتمال اینکه بحث از شاخهای به شاخه ی دیگر برود ، بیشتر است .تسهیل گر باید مکالمه را پیمانه ای کند و تنها زمانی یک مبحث را ترک کند که برای ان تصمیمی گرفته شده باشد.

اصل8.اگر چیزی واضح نبود، یک تصویر بکشید.

اصل9.الف)هنگامی که بر سر موضوعی به توافق رسیدید، به مبحث دیگر بپردازید .ب)اگر به توافق نرسیدیدبه مبحث دیگر بپردارییدپ) اگر ویژگی یا قابلیتی واضح نیست و نمی توان در حال حاضر ان را واضح کرد ، باز هم به مبحث دیگر بپردازید .

اصل10.مذاکره یک مسابقه یا بازی نیست . وقتی بهترین نتیجه راا می دهد که هر دو طرف برنده باشند . اگر اعضای تیم همکاری خوبی داشته باشند ، همه طرف ها هدفی مشترک خواهند داشت. هنوز هم مذاکره، مستلزم مصالحه از تمام طرف ها است.

اصول برنامه ریزی

اصل1.شناخت حوزه پروژه. اگر ندانید مقصد کجاست ، استفاده از نقشه راه غیر ممکن است. حوزه ی پروژه ، مقصدی برای تیم نرم افزاری ترسیم می کند.

اصل2.طرفهای ذی نفع را در فعالیت برنامه ریزی دخالت دهید. طرفهای ذی نفع اولویت وقید وبند های پروژه راتعیین می کنند. برای پاسخ گویی به ای واقعیت ها، مهندسان نرم افزار باید غالباً بر سر تاریخ تحویل، زمان بندی و سایر مسائل مرتبط با پروژه، با هم مذاکره کنند.

اصل3.این را بدانید که برنامه ریزی ماهیتی مبتنی بر تکرار دارد. برنامه ریزی باید طوری تنظیم شود که این تغییرات را پاسخ گو باشد . بعلاوه، در مدلهای فرایند افزایشی و مبتنی بر تکرار ، برنامه ریزی دو باره ، پس از تحویل هر نسخه از نرم افزاربر اساس باز خورد های گرفته شده از کاربران، حکمی قطعی است.

اصل4.براورد خود را بر اساس اچه که می دانید ،انجام دهید. اگر اطلاعات، مبهم یا غیر قابل اطمینان باشد، براورد ها نیز به همان میزان غیر قابل اطمینان خواهد بود.

اصل5.هم زمان با برنامه ریزی، ریسک را هم در نظر بگیرید. اگر ریسک هایی تعیین کرده اید که تاثیر واحتمال ان بالا است، برنامه ریزی برای حوادث محتمل ضروری است.

اصل6.واقع بین باشید ، مردم هر روز صد در صد کار نمی کنند. حتی بهترین مهندسان نرم افزار هم مرتکب اشتباه می شوند.

اصل7.هنگام تعریف برنامه ریزی ، گرانولیته را تعیین کنید. منظور از گرانولیته، سطحی از جزئیات است که در برنامه ریزی پروژه به ان پرداخته می شود.در برنامه ریزی یا گرانولیته بالا،جزئیات کاری چشم گیری ارائه می شود که روی بازه های زمانی نسبتاً کوتاهی برنامه ریزی می شود.

اصل8.تعیین کنید که چگونه می خواهید که ازکیفیت اطمینان یابید.

اصل9.چگونگی انجام دادن تغییرات را شرح دهید. حتی بهترین برنامه ریزی هم ممکن است با تغییرات کنترل نشده، اعتبار خود را از دست بدهد. باید مشخص کنید که تغییرات را چگونه می توان با پیشرفت کار مهندسی نرم افزار انجام داد. برای مثال ،ایا مشتری هر زمان در خواست تغییر دارد ؟اگر تغییری در خواست شود، ایا تیم ناگزیر از پیاده سازی فوری ان است؟ تاثیر وهزینه ی تغییر را چگونه باید ارزیابی کرد؟

اصل10.برنامه ریزی را به وفور پی گیری کنید و در صورت نیاز، تنظیماتی را به عمل اورید.پرو ژه نرم افزاری گاهی از زمانبندی عقب می افتد در صورت هر گونه لغزش، طرح را باید به فراخور، تنظیم کرد.

اصول مدل سازی

ما مدل را برای درک بهتر یک موجودیت واقعی که قرار است ساخته شود، ایجاد می کنیم

اصل1.هدف اصلی تیم نرم افزاری، ساخت نرم افزار است نه ایجاد مدل.چابکی به معنای رساندن نرم افزار به مشتری در سریع ترین زمان ممکن استمدل هایی که به رخ دادن ای اتفاق کمک می کند ، ارزش ایجاد را دارند ولی از مدل هایی که فرایند را کند کنند، یا سود چندانی نداشته باشند ، باید پرهیز کرد.

اصل2.سبک بار سفر کنید _ مدل هایی بیش ازنیاز ایجاد نکنید.هر مدلی که ایجاد می شود باید بارخ دادن تغییرات بهنگام سازی شود

اصل3.بکوشید ساده ترین مدل را بسازید که مساله یا نرم افزار را تعریف کند.نرم افزار را بزرگتر از حد لازم نسازید. با ساده نگه داشتن مدل ها ، نرم افزار حاصل نیز ساده خواهد بود.

اصل4.مدل را طوری بسازید که قابل تغییر باشد.

اصل5.توانایی بیان صریح هدف هر مدل ایجاد شده را داشته باشید.هر بار که مدلی را ایجاد می کنید از خود بپرسید که چرا چنین می کنید . اگر نمی توانید توجیه قانع کننده ای برای وجود مدل ارائه کنید ، وقتی صرف ان نکنید.

اصل6.مدل هایی را که توسعه می دهید، بر سیستم مورد نظر مطابقت دهید. ممکن است برای مطابقت دادن مدل بر برنامه کاربردی به نماد گذاری یا قواعدی نیاز باشد ،برای مثال ، یک بازی کامپیوتری ممکن است به تکنیک مدل سازی متفاوت با یک نرم افزار تعبیه شده بی درنگ (که موتور خودرو ها را کنترل می کند) نیاز داشته باشد.

اصل7.سعی کنید مدل مفید بسازید ، ولی ساخت مدل های کامل را فراموش کنید

اصل8.در مورد قالب و نحوه مدل ،تعصب به خرج ندهید . اگر در انتقال مفاهیم موفق است ،نمایش در مرحله دوم اهمیت قرار دارد

اصل9.اگر غریزه به شما می گوید مدلی درست نیست، هر چند روی کاغذ درست به نظر می رسد احتمالاً دلیلی برای این نگرانی دارید.

اصل10.به محض اینکه توانستید بازخورد بگیرید. هر مدلی باید مورد باز بینی اعضای تیمنرم افزاری قرار گیرد

اصول مدل سازی ساخته ها

اصل1.دامنه اطلاعاتی یک مساله باید نمایش داده ودرک شوند .

اصل2.عملکرد نرم افزار باید تعریف شود

اصل3.رفتار نرم افزار (به عنوان نتیجه ای از رویداد های خارجی) باید نمایش داده شود .

اصل4.مدل هایی که اطلاعات ، قابلیت ها و رفتار ها را تصویر می کنند باید به شیوه ای تقسیم شوند که جزئیات را به گونه ای لایه ای (یا سلسله مراتبی) نمایش دهند.

اصل5.وظیفه تحلیل باید از اطلاعات ضروری به سمت جزئیات پیاده سازی حرکت کند.

اصول مدل سازی طراحی

اصل1.طراحی باید تا مدل خواسته ها قابل رد گیری باشد.

اصل2.همواره معماری سیستمی را که قرار است ساخته شود ، در نظر داشته باشید.

اصل3.طراحی داده ها به اندازه طراحی عملکرد ها اهمیت دارد

اصل4.واسط ها (چه درونی چه بیرونی) باید با احتیاط طراحی شوند

اصل5.طراحی واسط کاربر باید مطابق کاربر نهایی تنظیم شود .

اصل6.طراحی در سطح مولفه ها باید مستقل از عملکرد باشد

اصل7.مولفه ها باید با یکدیگر وبا محیط خارجی ارتباطی سست داشته باشند

اصل8.نمایش ها (مدل های) طراحی باید به اسانی قابل درک باشند

اصل9.طراحی باید به صورت تکراری توسعه یابند . در هر دور از تکرار ، طراحی باید بکوشد تا سادکی بیشتر شود.

اصول ساخت

فعالت ساخت شامل مجموعه ای از وظایف کد نویسسی وازمایش می شود که نتیجه ی ان ،نرم افزاری عملیاتی و اماده تحویل به مشتری یا کاربر نهایی است.

اصول کد نویسی . اصول راه گشا در وظایف کد نویسی ، بستگی تنگاتنگی با سبک برنامه نویسی ، زبان برنامه نویسی و شسیوه ی برنامه نویسی مورد نظر دارند. اما چند اصل بنیادی در این زمینه قابل بیان است:

اصول اماده سازی: پیش از انکه حتی یک خط برنامه بنویسید ،اطمینا نحاصل کنید که:

- **می دانید چه مساله ای را قرار است حل کنید.**
- **اصول و مفاهیم طراحی پایه را می دانید.**
- **زبانی برای برنامه نویسی انتخاب کنید که نیاز های نرم افزار و محیطی را که قرار است در ان کار کند بر اورده سازد.**
- **محیطی برای برنامه نویسی انتخاب کنید که ابزار های لازم برای اسان تر کردن کار را در اختیار تان قرار دهد.**
- **مجموعه ای از ازمون های پایه را ایجاد کنید که با کامل شدن کد نویسی مولفه بتوانید ان را به کار ببرید.**

اصول برنامه نویسی: باشروع به کد نوسیسی ، اطمینان حاصل کنید که:

- **الگوریتم هایتان را با دنباله روی از برنامه سازی ساخت یافته ،مقید کنید**
- **استفاده از برنامه نویسی جفتی را در نظر داشته باشید**
- **ساختمان داده هایی را انتخاب کنید که نیاز های طراحی را براورده کند.**
- **معماری نرم افزار را بشناسید و واسط های سازگار با ان را بسازید.**
- **منطق شرطی را تا خد امکان ساده نگه دارید.**
- **حلقه های تو در تو را به شیوه ای بنویسید که به اسانی قابل ازمون باشد.**
- **نام های با معنی برای متغیرها انتخاب کنید و از سایر استاندارد های کد نویسی محلی پی روی کنید.**
- **کد هایی بنویسید که خود مستند سازی شده باشد.**
- **یک چیدمان بصری ایجاد کنید (مثلاً با تو رفتگی و خطوط خالی) که به فهم کد های شما کمک کند.**

اصول اعتبار سنجس : پس از به پایان رساندن اولین دور کد نوسیسی، حتماً:

- **در صورت امکان کشتی در میان کد ها بزنید.**
- **ازمون واحد ها را اجرا کنید و خطاهایی را که کشف می شوند ،تصحیح نمایید .**
- **کد ها را باز ارایی کنید.**

اصول ازمون

- **ازمون ، فرایند اجرای برنامه به قصد یافتن خطاهاست.**
- **یک مورد ازمون خوب باید خطاهای کشف نشده را با احتمال زایادی کشف کند.**
- **ازمون موفق ، ازمونی است که خطای کشف نشده تا کنون را کشف کند.**

دیویس[dav95b] مجموعه ای از اصول ازمون را پیشنهاد می کند:

اصل1.همه ی ازمون ها تا خواسته های مشتری قابل رد گیری باشند

اصل2.ازمون هارا باید مدتها قبل از شروع ازمون برنامه ریزی کرد

اصل3.اصل پارتو در ازمون نرم افزار کاربرد دارد

اصل4.ازمون باید در مقیاس کوچک انجام شود و به سمت مقیاس بزرگ پیش برود.

اصل5.ازمون کامل امکان پذیر نیست.

اصول استقرار

فعالیت استقرار شامل سه کنش می شود تحویل ،پشتیبانی وباز خورد.در همان حال که تیم اماده تحویل یک نسخه جدید می شود ، چند اصل کلیدی را باید رعایت کند:

اصل1.انتظارات مشتری برای نرم افزار باید مدیریت شود. به وفر پیش می اید که انتظارات مشتری بیش از ان چیزی است که تیم قول ان را داده است و بلا فاصله نا رضایتی شروع می شود ، این امر منجر به باز خوردی می شود که فاقد بهره است و با عث دل سردی نتیم میشود.

اصل2.پکیج تحویل کامل بای مونتاژ و ازمایش شود .

اصل3.پیش از تحویل نرم افزار، یک روال پشتیبانی باید مشخص کرد.پشتیبانی باید برنامه ریزی شود مواد پشتیبانی باید اماده شود و ساز و کارهای مناسب برای حفظ سوابق باید وضع شود تا تیم نرم افزاری بتواند انواع پشتیبانی را مورد ارزیابی قرار دهد.

اصل4.مواد اموزشی مناسب باید برای کاربران نهایی تهیه شود

اصل5. نرم افزار مشکل دار ابتدا باید اصلاح و بعداً تحویل داده شود.

فصل پنجم مهندسی خواسته ها

هدف اصلی: مشخص کردن خواسته هاست.

ما می دانیم که مراحل ساختن یک نرم افزار چیست : 1- برقراری ارتباط یا همان شناخت خواسته ها 2- برنامه ریزی 3- مدل سازی 4- ساخت 5- پشتیبانی.

پس اولین مرحله ساخت نرم افزار یا محصول شناخت خواسته هاست، مهمترین و اساسی ترین مرحله برای یک محصول است،و از همه مهمتر دشوارترین مرحله است زیرا درک نیازهای مشتری دشوار است شاید بصورت شفاهی بتوانیم بیان کنیم ولی اگر بخواهیم خواسته ها را به صورت گزارش تهیه کنیم کمی مشکل میشود.

در ابتدا خواسته ها را تعریف میکنیم:

خواسته ها:

در کل برای خواسته ها میتوانیم یک مجموعه وظایف مهندسی تعیین کنیم که عبارتند از:

1- شروع 2- استخراج 3- شناخت 4- مذاکره 5- تعیین مشخصات 6- اعتبار سنجی 7- مدیریت

محصول کار چیست؟یک گزارش کتبی از خواسته ها را فراهم میکنیم که این گزارش میتواند سناریو کاربردی،مستند،مدل و موارد دیگر باشد.

مهندسی خواسته ها:همان طور که مرحله تعیین خواسته ها کمی مشکل است ولی کاری خلاقانه و جذاب است در عین حال اگر نتوانیم خواسته ها را درست تعیین کنیم ضربه خیلی شدیدی به سیستم وارد میشود.

برخی سازندگان یا همان مهندسان فکر می کنند اصلا نیازی به مشخص کردن خواسته ها نیست و قبل از مشخص کردن خواسته ها شروع به ساخت نرم افزار می کنند و معتقدند که در طول ساخت،خواسته ها معلوم میشوند و این کار کاملا اشتباه است و باعث میشود که عناصر نرم افزار اگرچه به ظاهر درست به نظر میرسد ولی هر کدام عیبی داشته باشند و باعث شکست پروژه شوند پس اینکه بدون مشخص کردن خواسته ها دست به ساخت نرم افزار بزنیم کاری بیهوده است.

پس در مهندسی خواسته ها یک سری وظایف و فنون را برای شناخت خواسته ها انجام می دهیم،نکته مهم اینست که مهندسی خواسته ها از مرحله برقراری ارتباط تا مدلسازی مطرح است و نقطه ی شروع مهندسی نرم افزار آنجاست که نیازها تعیین می شود یا جایی که نرم افزار چیزی جز مولفه ای از یک سیستم بزرگتر نیست.

مهندسی خواسته ها همانند پلی به سوی طراحی و ساخت عمل میکند.

مراحل کار برای بدست آوردن گزارشی از خواسته ها:

7 مرحله دارد:

1- شروع 2- استخراج 3- شناخت 4- مذاکره 5- تعیین مشخصات 6- اعتبار سنجی 7- مدیریت

شروع: در این مرحله ماهیت مسئله ای که میخواهیم حل کنیم را مشخص میکنیم

استخراج: در این مرحله آن چه مورد نیاز است تعریف میشود.

تعریف میشود که سیستم یا محصول قرار است چگونه استفاده شود

در این مرحله ممکن است مشکلاتی به وجود آید:

مشکل اول مربوط به حوزه ی پروژه: مشتری جزئیات غیر ضروری را بیان کند و باعث مبهم شدن هدف سیستم شود

مشکل دوم مربوط به درک پروژه: در این نوع مشکل مشتری/کاربر مطمئن نیستند چه چیزی مورد نیاز است تا مطرح کنند یا مشتری ممکن است خواسته های مبهم را بیان کند

مشکل سوم مربوط به تغییر پذیری:یعنی ممکن است خواسته ها در طول پروژه تغییر کند.

شناخت: اطلاعاتی که از مشتری بدست می آید اصلاح میشوند، خواسته های کلی بدست می آید و خواسته های مشتری پالایش میشوند.

مذاکره: در این مرحله ممکن است مهندس یا مشتری خواسته هایی داشته باشند که با توجه به محدودیت،بر آورده شدن آنها امکان پذیر نباشد یا خواسته های متضاد داشته باشند بخاطر همین مذاکره میکنند تا تضادها را بر طرف کنند و به توافق برسند.

تعیین مشخصات: حال که اطلاعات جمع آوری شد در این مرحله اطلاعات را بصورت گزارش که میتواند مستند یا سناریو یا مدل باشد تهیه میکنیم،نکته در این مرحله اینست که بر اساس عقیده برخی برای سیستم های بزرگ مستندات و مدل و برای سیستم های کوچک سناریوهای کاربردی مناسب است.

اعتبار سنجی: حال این اطلاعات یا خواسته ها مورد ارزیابی قرار میگیرند تا مطمئن شویم که خواسته ها بدون ابهام بیان شده اند،ناسازگاری و خطاها شناسایی و تصحیح میشوند و همه ی خواسته ها از استاندارد های وضع شده پیروی میکنند.

مدیریت: در این مرحله یک مجموعه، فعالیت مهندس نرم افزار را در مقابل تغییرات،شناسایی،کنترل و پیگیری خواسته ها در طول پروژه کمک میکند.

چند مرحله برای تدارک شناخت خواسته های نرم افزار وجود دارد که درمورد آنها بحث میکنیم:

شناسایی طرفهای ذی نفع:

ابتدا طرفهای ذی نفع را تعریف میکنیم: هر کس که بطور مستقیم یا غیر مستقیم از سیستم بهرمند خواهد شد. برخی از این افراد میتوانند بازاریاب ها، مشتریان داخلی و خارجی، کاربران نهایی، مدیران تجاری، مدیران تولید و سایر افراد باشند.

شناخت دیدگاههای چندگانه:

در این مرحله وقتی افراد ذی نفع را شناختیم باید دیدگاههای آنها را نیز برای بررسی سیستم بدانیم، مثلا کاربران نهایی به ویژگی هایی از سیستم توجه دارند که با آنها آشنایی دارند و یادگیری و استفاده از آنها آسان است. همچنین بازاریاب ها به قابلیتهایی علاقه نشان میدهند که بازار بالقوه را برانگیخته و فروش سیستم جدید را آسانتر مینماید.

تلاش برای همکاری:

لازم است این نکته را بیان کنیم که این سه مرحله همگی نوعی آمادگی قبل از شروع کار برای شناخت خواسته ها میباشد. در این مرحله باید افراد یا طرفهای ذی نفع همکاری لازم را بعد از فراهم آوردن مقدمات کار با مسئولان ساخت نرم افزار، داشته باشند.

جمع آوری خواسته ها:

جلسات همکاری به منظور جمع آوری خواسته ها تشکیل می دهیم که این جلسات باید بر طبق یک سری دستورالعمل باشد و نتیجه آنها یک متن یک یا دو صفحه ای به عنوان درخواست محصول می باشد:

1. همه مهندسین نرم افزار طرف های ذی نفع جلسه حضور داشته باشند.
2. قواعدی که برای مشارکت وجود دارد وضع شود.
3. دستور کاری که قرار است روی آن بحث شود هم جنبه رسمی داشته باشد و هم جنبه غیررسمی.

4. یکی از افراد که میتواند از مشتریان یا سازندگان باشد کنترل جلسه را بدست بگیرد.
5. یک ساز و کار تعریف شده بکار گرفته شود.

مبحث بعدی استقرار عملکرد کیفیت(QFD) یک تکنیک کیفیتی است که نیازهای مشتری را به خواسته های فنی برای نرم افزار ترجمه می کند و بر نیازهای با ارزش مشتری تاکید دارد.

در اینجا سه نوع خواسته داریم که عبارتند از:

1- خواسته های عادی: اهدافی که طی جلسه به مشتری برای محصول بیان میشود.
2- خواسته های مورد انتظار: این خواسته ها برای سیستم ضروری است ولی در طول جلسه ذکر نمیشود و اگر برآورده نشوند نارضایتی مشتری را بدنبال دارد.
3- خواسته های مهیج: یک سری ویژگی های بالاتر از انتظار مشتری هستند و در صورت برآورده شدن باعث رضایتمندی مشتری میشوند.

بعد از اتمام جلسات گزارشی از خواسته ها تهیه میشود که این گزارش میتواند به مدلهای گوناگون باشد که یکی از آنها سناریوی کاربردی است.

سناریوی کاربردی: سناریوی کاربرد یک رشته کاربرد را برای سیستم تعیین میکند که به آن use case هم می گویند و همچنین چگونگی استفاده از سیستم را توضیح میدهد.

محصولات کاری:

محصولات کاری تولید شده پس از استخراج خواسته ها که این محصولات بسته به اندازه سیستم یا محصولی که قرار است ساخته شود متغیر است. برخی از این محصولات عبارتند از: 1- فهرستی از مشتریان 2- بیان نیازها 3- توصیفی از محیط فنی سیستم و سایر چیزها

توسعه سناریوی کاربردی:

گام اول تعریف کنش گرها میباشد که عبارتند از افراد یا دستگاههای متفاوتی که از سیستم یا محصول در حیطه عملکرد یا رفتاری که قرار است توصیف شود استفاده میکند.

هرکدام از این کنش گرها نقشی را بر عهده میگیرند که با کاربران نهایی یکسان نیستند، چون کاربران میتوانند در طول پروژه چند نقش بر عهده داشته باشند ولی کنش گرها فقط یک نماینده هستند.

انواع کنش گر:

کنش گرهای نوع اول که بطور مستقیم با نرم افزار کار می کنند و کنش گرهای نوع دوم که از نرم افزار پشتیبانی میکنند.

با شناسایی این کنش گرها use case توسعه می یابد.

مدل خواسته ها:

این مدل که همان سناریوی کاربردی است، میتواند عناصری هم داشته باشد:

1- عناصر مبتنی بر سناریو: که سیستم از دیدگاه کاربر توصیف میشود.
2- عناصر مبتنی بر کلاس: هر سناریوی کاربردی مجموعه ای از اشیا است که با تعامل یک کنش گر با سیستم دستکاری میشود.
3- عناصر مبتنی بر رفتار: رفتار یک سیستم کامپیوتری میتواند اثری عمیق بر روی انتخاب طراحی و رویکرد مورد استفاده در پیاده سازی داشته باشد.

4- عناصر جریان گرا: انتقال اطلاعات با جریان یافتن در یک سیستم صورت میپذیرد، یعنی سیستم شکل های گوناگون ورودی را میگیرد، پردازش میکند و خروجی را به شکل های متنوع تولید میکند.

مذاکره

وقتی با مشتری مذاکره میکنیم باید یک سری اصول را رعایت کنیم که عبارتند از:

1- بدانیم رقابتی در کار نیست.
2- در مورد خواسته و هدف خود تصمیم بگیریم.
3- عادلانه به حرف مشتری گوش دهیم.
4- به علایق مشتری توجه کنیم.
5- خلاق باشیم.
6- اجازه ندهیم بحث جنبه شخصی به خود بگیرد.
7- در صورت توافق به قول خود عمل نماییم.

فصل 6 : مدل سازی خواسته ها

مدل سازی خواسته ها شامل سناریو ها ، اطلاعات و کلاس های تحلیل می باشد.

مدل سازی خواسته ها تلفیقی از شکل های متنی و نموداری برای به تصویر کشیدن خواسته ها است به شیوه ای که درک آن راحت تر باشد و مهمتر از آن به آسانی بتوان آن را برای تصحیح ، تکمیل و سازگاری مورد دید قرار دارد.

این مدل را مهندس نرم افزار با به کارگیری خواسته های استخراج شده از مستری می سازد که به آن **تحلیل گر** نیز می گویند.

سه دیدگاه متفاوت مدل سازی خواسته ها:

1. مدل های مبتنی بر سناریو : سیستم را از دیدگاه کاربر نشان میدهد.
2. مدل های داده ای (اطلاعاتی) : فضای اطلاعاتی را به نمایش در می آوردند.
3. مدل های مبتنی بر کلاس : اشیاء ، صفات و روابط را تعریف می کنند.

پس از اینکه این سه مدل تهیه شدند ، تحلیل می شوند تا به کمال و سازگاری لازم برسند

محصول : آرایه ای از فرم های متنی و نموداری را میسازد که هر کدام دیدگاهی از یک یا چند عنصر مدل فراهم می کنند.

Ã نکته: مدل خواسته ها نخستین نمایش فنی از یک سیستم است که مجموعه ای ازمدل هاست.

تحلیل خواسته ها

به تعیین مشخصات خصوصیات عملیاتی نرم افزار منجر می شود ، واسط نرم افزار با سایر عناصر سیستم را مشخص می کند و کارهایی را که نرم افزار باید رعایت کند ، تعیین می کند.

انواع کنش های مدل سازی خواسته

1. مدل مبتنی بر سناریو
2. مدل های داده ای
3. مدل های مبتنی بر کلاس
4. مدل های جریان گرا : عناصر عملیاتی سیستم و چگونگی تبدیل داده ها را نشان می دهد
5. مدل های رفتاری : چگونگی رفتار نرم افزار را نشان میدهد

این مدل ها ابزاری برای ارزیابی کیفیت پس از ساخته شدن نرم افزار فراهم می آورند

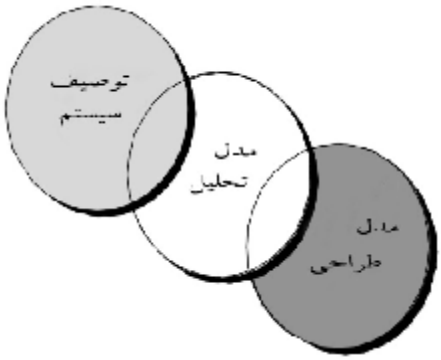
فلسفه و اهداف کلی

در سراسر مدل سازی خواسته آنچه که در مرحله اول مورد توجه قرار می گیرد چیستی است نه چگونگی.

سه هدف مدل سازی خواسته ها عبارتند از :

1. توصیف آنچه مشتری نیاز دارد
2. ایجاد مبنایی برای تهیه طراحی نرم افزار
3. تعریف مجموعه ای از خواسته ها که پس از ساخته شدن نرم افزر بتوان آنها را اعتبار سنجی کرد.

مدل تحلیل پلی است میان توصیف در سطح سیستم و یک طراحی نرم افزار :



شکل 1-6 : مدل خواسته ها به عنوان پلی میان توصیف سیستم و مدل طراحی

قواعد ساده تحلیل

1. مدل باید خواسته هایی را کانون قرار دهد که در دامنه تجاری قابل مشاهده باشند.
2. هر عنصر از مدل خواسته ها باید چیزی به درک ما اضافه کند و از دامنه اطلاعاتی و رفتار سیستم دیدی فراهم سازد.
3. ملاحظات زیر ساختی و سایر مدل های غیر عملیاتی را تا طراحی به آخر اندازد.
4. ارتباط را در سراسر سیستم به حداقل برساند.
5. مدل خواسته ها باید رضایت همه طرف های معامله را جلب کند.
6. سادگی مدل را تا حد امکان حفظ کند.

تحلیل دامنه

تحلیل دامنه نرم افزار عبارت است از شناسایی، تحلیل و تعیین مشخصات خواسته های رایج از یک دامنه کاربردی خاص.

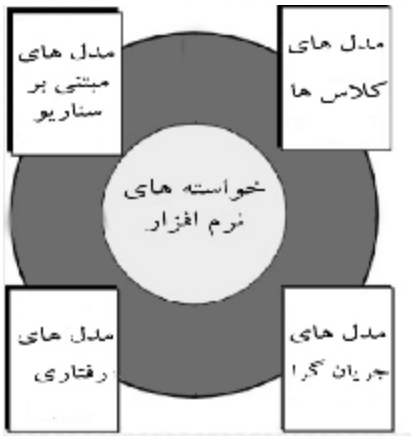
تحلیل دامنه، به یک برنامه کاربردی خاص توجه نمی کند بلکه دامنه ای را مورد توجه قرار می دهد که برنامه کاربردی در آن قرار دارد هدف آن، شناسایی عناصر مشترک حل مسأله است که می توان در همه برنامه های کاربردی دیگر از آن استفاده کرد.

روش های مدل سازی خواسته ها

1. در رویکرد اول داده ها و فرآیند هایی که این داده ها را تبدیل می کنند در نظر گرفته می شود.

اشیای داده به شیوه ای مدل سازی می شوند که صفات و روابط میان آنها را تعریف کند. فرآیندهایی که این اشیای داده را دستکاری می کنند، به شیوه ای مدل سازی می شوند که چگونگی تبدیل داده ها را به هنگام جریان یافتن اشیای داده در سیستم نشان دهد.

2. رویکرد دوم برای مدل سازی تحلیلی ، که تحلیل **شی گرا** نامیده می شود بر تعریف کلاس ها و روش همکاری آنها با یکدیگر برای برآورده کردن خواسته های مشتری تأکید دارد.



شکل 3-6 عناصر مدل تحلیل

هر عنصر از مدل خواسته ها مسأله را از دیدگاهی متفاوت نشان می دهد :

1. عناصر مبتنی بر سناریو: چگونگی تعامل کاربر با سیستم و فعالیت های خاص را به تصویر می شد.
2. عناصر مبتنی بر کلاس ها: اشیایی را که سیستم دستکاری می کند، عملیات را که روی اشیا انجام می شود تا این دستکاری ها اثر کند، روابط میان این اشیا و همکاری هایی را که بین کلاس ها تعریف می شود، مدل سازی می کند.
3. عناصر رفتاری: به تصویر کشیدن چگونگی تغییر یافتن حالت سیستم توسط رویدادهای خارجی.
4. عناصر جریان گرا: نمایش سیستم به عنوان یک تبدیل اطلاعات و به تصویر کشیدن چگونگی تبدیل اشیای داده به هنگام جریان یافتن در تیم.

نتیجه مدل سازی تحلیل ، به دست آمدن هر کدام از این عناصر مدل سازی است.

مدل سازی مبتنی بر سناریو

هر چه موفقیت یک سیستم از طریق راه های گوناگون انجام شود رضایت مشتری در صدر قرار می گیرد.

اگر بدانیم که کاربران چگونه می خواهند با یک سیستم رابطه برقرار کنند تیم مهندسی نرم افزار ما بهتر قادر به مشخص کردن خواسته ها و مدل های طراحی خواهد بود که UML با ایجاد سناریو هایی به شکل use case نمودارهایی را آغاز می کند.

ایجاد یک use case مقدماتی

use case ها به تعریف آنچه که در بیرون سیستم قرار دارد (کنش گر) و آنچه که باید توسط سیستم انجام شود (use case) کمک می کند.

- دو وظیفه مهم مهندسی خواسته ها، **دریافت** و **استخراج** اطلاعات مورد نیاز برای شروع به نوشتن است.
- برای شروع به توسعه یک use case فعالیت هایی را که یک کنش گر انجام می دهد فهرست کنیم. با پیشرفت گفتگو با طرف معامله تیم جمع آوری خواسته برای هر کدام ار قابلیت ها use case تهیه می کند.
- use case ها ابتدا به صورت روایی و غیر رسمی نوشته می شوند و با یک قالب ساخت یافته بازنویسی می شوند.

پالایش یک use case مقدماتی

شرحی از تعامل های متفاوت برای درک کامل قابلیت توصیف شده در یک use case لازم است. بنابراین هر مرحله از سناریوی اولیه با پرسیدن سوالاتی مانند آیا کنش گر در این نقطه کنش دیگری انجام می دهد ؟ ارزیابی می شود. پاسخ به این پرسش ها باعث ایجاد سناریوی ثانویه می شود.

هر یک از وضعیت ها یک استثنا برای use case می باشد.

استثنا وضعیتی است که باعث می شود سیستم ، رفتاری متفاوت از خود نشان دهد.

استثنا را در صورتی باید در use case توصیف کرد که نرم افزار قادر به تشخیص شرایط توصیف شده و سپس انجام اقدام مناسب در صورت تشخیص آن باشد.

نوشتن یک use case رسمی

use case های غیر رسمی برای مدل سازی خواسته ها کفایت می کنند ولی هنگامی که یک use case فعالیتی مهم را شامل می شود روشی رسمی تر مطلوب است.

use case با محدودیتی همراه است. توصیف use case باید روشن و واضح باشد و در غیر ایصورن باعث گمراهی می شود.

use case بر خواسته های رفتاری و عملیاتی تاکید دارد و اگر خوب نوشته شود ، مزایای اساسی دارد.

مدل های UML که use case را تکمیل می کنند

وضعیت های زیادی در مدل سازی خواسته ها وجود دارد که در آنها مدل های مبتنی بر متن ممکن است اطلاعات را به طور واضح ارائه ندهد، در چنین مواردی مدل های گرافیکی UML را داریم.

توسعه نمودار فعالیت ها

نمودار فعالیت های UML ، کنش ها و تصمیم گیری هایی را که در اجرای یک عملکرد رخ می دهند به نمایش می گذارد.

نمودار های بخش بندی (Swimlane)

نمودار بخش بندی UML ، جریان کنش ها و تصمیم گیری ها را به نمایش می گذارد و نشان می دهد کدام کنش گران کدام کنش را انجام می دهند

مفاهیم مدل سازی داده ها

اگر خواسته های نرم افزار شامل ایجاد یا برقراری ارتباط با یک بانک اطلاعاتی شود یا فایل پیچیده ای داشته باشد سیستم مهندسی نرم افزار ممکن است تصمیم به ایجاد مدل داده ای بگیرد. تحلیل گر همه اشیای داده و روابط میان آنها را تعیین می نماید که نمودار موجودیت ـ ارتباط (ERD) این مسائل را به نمایش می گذارد.

اشیای داده یک شی داده نمایشی است از هرگونه اطلاعات مرکب که توسط نرم افزار پردازش می شود. منظور از اطلاعات مرکب چیزی است که دارای چند صفت متفاوت باشد.

صفات داده ها صفات داده ها ، خواص یک شی داده را تعریف می کنند و یکی از 3 خصوصیات مهم زیر را به خود می گیرند. از آنها میتوان برای نامگذاری نمونه ای از شی داده ای ، توصیف نمونه یا ارجاع به نمونه ای دیگر در جدولی دیگر استفاده نمود. به عبارت دیگر صفات، یک شی داده را نام می برند، خصوصیات آن را توصیف می کنند و در برخی موارد به شی دیگر ارجاع می دهند.

شی داده یک موجودیت داده ای مرکب را تعریف می کند و به صورت منفرد هستند.

کلاس شی گرا صفات داده ها را پنهان سازی می کند. کلاس ها از طریق پیام ها با هم ارتباط برقرار می کنند.

ارتباطات رابطه ها نشان گر شیوه اتصال اشیای داده به یکدیگرند.

ابزار های نرم افزاری : توانایی نمایش اشیای داده ، خصوصیات آنها و روابط میان آنها را نشان می هند.

مکانیک: ابزارهای این گروه کاربر را در توصیف اشیای داده و روابط میان آنها یاری می دهند و از نمادگذاری ERD استفاده می کنند.

مدل سازی مبتنی بر کلاس ها

مدل سازی مبتنی بر کلاسها ، اشیایی را که سیستم دستکاری می کند ، عملیاتی که در مورد اشیا برای دستکاری به کار می رود ، روابط میان اشیا و همکاری هایی را که بین کلاس ها رخ می دهند، نمایش می دهد.

عناصر یک مدل مبتنی بر کلاس ها شامل کلاس ها و اشیا ، صفات ، عملیات ، مدل های همکاری ـ مسئولیت کلاس ها (CRC) ، نمودار های همکاری و پکیج ها می شود.

شناسایی کلاس های تحلیل

1. موجودیت های خارجی (مانند سایر سیستم ها ، دستگاه ها ، افراد) که اطلاعات مورد استفاده یک سیستم کامپیوتری را تولید یا مصرف می کنند.
2. چیزهایی (مانند گزارش ها ، صفحه نمایش ها ، نامه ها ، سیگنال ها) که بخشی از دامنه اطلاعاتی مسأله اند.
3. رخدادها یا رویدادهایی (مانند انتقال یک خاصیت یا کامل شدن یک سری حرکات روبات) که در حیطه عملیاتی سیستم به وقوع می پیوندند.
4. نقش هایی (مانند مدیر ، مهندس ، فروشنده) که توسط افراد در حال تعامل با سیستم ایفا می شود.
5. واحد های سازمانی (مانند بخش ، گروه ، تیم) که به کاربردی خاص مربوط می شوند.
6. مکان هایی (مانند قسمت تولید یا بارانداز) که حیطه مسأله و عملکرد کلی سیستم را تعیین می کند.
7. ساختارهایی (مانند حس گر ها ، وسایل چهارچرخ یا کامپیوتر ها) که کلاسی از اشیا یا کلاس های مرتبطی از اشیا را تعریف می کنند.

مشخص کردن صفات

صفات، کلاس انتخاب شده برای گنجاندن در مدل خواسته را توصیف می کنند. در اصل همین صفات هستند که کلاس را تعریف می کنند که مشخص می کنند منظور از کلاس در حیطه فضای مسأله چیشت.

برای توسعه مجموعه ای با معنی از صفات برای یک کلای تحلیل باید هر کدام از use case ها را مطالعه نموده و آن چیزهایی را انتخاب کرد که به طور منطقی به کلاس تعلق دارند. به علاوه برای هر کلاس باید به این پرسش پاسخ داد که کدام اقلام داده ای به طور کامل این کلاس را در حیطه مسئله مورد نظر تعریف می کنند؟

تعریف عملیات ها

عملیات ها رفتار شی را تعریف می کنند که معمولا آنها را در چهار گروه گسترده تقسیم می کنند:

1. عملیاتی که داده ها را به طریقی دستکاری می کنند (مثل اضافه کردن ، حذف کردن ، فرمت بندی دوباره و انتخاب کردن).
2. عملیاتی هایی که محاسبه انجام می دهند.
3. عملیات هایی که درباره حالت یک شی تحقیق می کنند.
4. عملیات هایی که یک شی را برای وقوع یک رویداد کنترل شده پایش می کنند.

نکته : عملیات ها باید از ماهیت همبستگی ها و صفات کلاس آگاهی داشته باشند.

مدل سازی همکار مسؤولیت کلاس ها (CRC)

این مدل ابزاری ساده برای شناسایی و سازماندهی کلاس های مرتبط با خواسته های سیستم یا محصول فراهم می سازد.
مدل CRC در واقع مجموعه ای از کارت های شاخص استاندارد است که کلاس ها را به نمایش می گذارند. یکی از اهداف کارت های CRC ، شکست زود هنگام شکست زیاد و شکست کم هزینه است. پاره کردن چند تکه کاغذ به مراتب کم خرج تر از سازمان دهی دوباره به مقادیر فراوانی از کد منبع است.

کلاس ها کلاس های ارائه شده را میتوان به صورت زیر طبقه بندی کرد :

1. کلاس های موجودیت، که کلاس های مدل یا تجاری نیز نامیده می شوند، مستقیماً از بیان مسأله استخراج می شوند.
2. کلاس های مرزی در ایجاد واسط به کار می روند که کاربر به هنگام استفاده از نرم افزار و تعامل با آن مشاهده می کند.
3. کلاس های کنترل گر که یک واحد کار را از ابتدا تا انتها مدیریت می کند.

مسئولیت ها پنج دستورالعمل برای تخصیص مسؤولیت ها به کلاس ها پیشنهاد شده است :

1. هوش مندی سیستم باید طوری میان کلاس ها توزیع شود که به بهترین وجه پاسخ کوی نیازهای مسأله باشد.
2. هر مسؤولیتی تا حد امکان بصورت کلی بیان شود.
3. اطلاعات و رفتار مرتبط با آن باید در یک کلاس قرار داده شوند.
4. اطلاعات نربوط به یک چیز باید تنها در یک کلاس قرار داده شوند و نباید در میان چند کلاس توطیع شوند.
5. مسؤولیت ها را در صورت امکان باید در میان کلاس های مرتبط به اشتراک گذاشت.

همکاری ها کلاس ها به یکی از این دو شیوه مسئولیت های خود را به انجام می رسانند :

1. کلاس می تواند از عملیات خودش برای دستکاری صفات خودش استفاده کند.
2. کلاس می تواند با سایر کلاس ها همکاری کند.

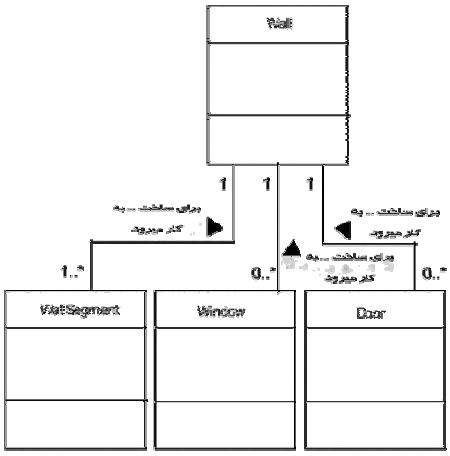
همکاری نشان گر درخواست های یک کلاینت از سرور برای به انجام رساندن مسؤولیت یک کلاینت هستند

برای کمک به شناسایی همکارها می توان سه رابطه کلی میان کلاسها را بررسی نمود :

1. رابطه شمول
2. رابطه آگاهی داشتن از
3. رابطه بستگی داشتن به

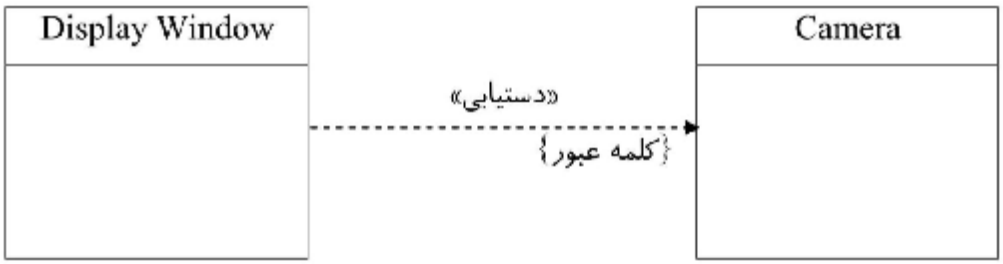
اجتماع و وابستگی

در بسیاری از موارد ، دو کلاس تحلیل ، به شیوه ای بسیار مشابه با ارتباط دو شی داده ای با هم ارتباط دارند ، در UML این روابط را اجتماع می نامند.
در شکل زیر کلاس Wall با سه کلاس همبستگی دارد که به دیوار امکان ساخته شدن را میدهند و عبارتند از Wall Segment و Window و Door .



شکل 13-6 چند گانگی

در برخی موارد ، یک همبستگی ممکن است با ذکر چندگانگی (multiplidity) بهتر قابل تعریف باشد.
اجتماع، رابطه میان کلاس ها را مشخص می کند. چندگانگی مشخص می کند که چند کلاس با چند کلاس دیگر ارتباط دارند.



شکل 14-6 وابستگی ها

پکیج های تحلیل

بخش مهمی از مدل سازی تحلیل ، گروه بندی است.
یعنی عناصر گوناگون مدل تحلیلی (مثل use case ها و کلاس های تحلیل) طوری گروه بندی می شوند که یک پکیج از آنها تشکیل شود و به آنها **پکیج تحلیل** گفته می شود؛ به هر کدام از این پکیج ها یک نام مشخص داده می شود.
نکته : پکیج برای بسته بندی مجموعه از کلاس های مرتبط به کار می روند.
علامت مثبت قبل از نام کلاس تحلیل در هر پکیج نشان گر آن است که این کلاس در معرض دید عموم هستند و بنابراین از طریق سایر پکیج ها قابل دستیابی اند.
علامت منفی نشان می دهد که عنصر از همه پکیج های دیگر پنهان است و علامت # نشان می دهد که یک عنصر تنها در دسترس پکیج های موجود در داخل پکیجی مشخص قرار دارد.

فصل هفتم : مدل سازی خواسته ها

مدل سازی خواسته ها چند بعد دارد . مدل های جریان گرا ، رفتاری ، ملاحظات خاص (مبتنی بر الگو) و برنامه های تحت وب که هر کدام از این ها مکمل use case هستند . این مدلسازی را مهندس نرم افزار یا تحلیل گر با بکارگیری خواسته ی مشتری یا طرف ذینفع می سازد .

مراحل کار :

جریان گرا : چگونگی تبدیل اشیای داده ای توسط قابلیت پردازش

رفتاری : حالت های سیتم و کلاس های آن و تأثیر رویدادها بر این حالت

مبتنی بر الگو : دانش موجود درباره ی دامنه ، برای تسهیل در امر خواسته ها

برنامه های تحت وب : برای نمایش خواسته های مرتبط با محتوا ، تعامل ، عملکرد و ...

محصول کار : فرم های متنی و نموداری

راهبردهای مدل سازی خواسته ها

دو دیدگاه داریم :

1- ساخت یافته : اشیای داده ای و فرآیندهایی که این داده ها را تبدیل می کنند ، موجودیتی مجزا در نظر گرفته می شوند . فرآیندهایی که داده ها را دستکاری می کنند ، به شیوه ای مدلسازی می شوند که چگونگی تبدیل اشیای داده ای را بهنگام جریان یافتن آنها در سیستم نشان دهند .

2- تحلیل شی گرا : بر تعریف کلاس ها و شیوه ی همکاری آنها با همدیگر برای برآورده ساختن خواسته های مشتری تأکید دارد .

مهم این نیست که کدام مدلسازی بهتر است . مسأله این است که کدام یک از نمایش ها ، بهترین مدل خواسته های نرم افزار را در اختیار مشتری قرار می دهد .(تیم نرم افزار غالباً یک روش را انتخاب می کند .)

مدلسازی جریان گرا (جریان داده ای)

اکثر مهندسین نرم افزار این مدلسازی را خارج از رده می شمارند امّا همچنان یکی از پرکاربرد ترین نماد گذاری های تحلیل خواسته ها تا کنون بوده است . نمودار جریان داده ها (DFD) و اطلاعات و نمودارهای مرتبط با آن بخش رسمی از UML به شمار نمی روند ولی می توان از آن ها برای تکمیل نمودارهای UML استفاده کرد .

DFDنمایی از سیستم بر اساس ورودی - فرآیند - خروجی به دست می دهد . یعنی داده ها درون نرم افزار جریان پیدا کرده ، توسط عناصر پردازشی تبدیل شده و داده حاصل به بیرون نرم افزار جریان پیدا می کند .

اشیای داده با پیکان برچسب دار و تبدیلات با دایره که **حباب** هم نامیده می شود ، مشخص می شوند .

DFD شیوه ای سلسله مراتبی است ؛ یعنی اولین مدل در جریان داده ها (که DFD سطح صفر یا نمودار حیطه ای نامیده می شود) کل سیستم را نمایش می دهد . نمودارهای بعدی پالایش قبلی است و جزئیات بیشتر .

*** ایجاد مدل جریان داده**

با پالایش DFD در سطوح بالاتر می توانیم سیستم را از نظر عملیاتی تجزیه کنیم .

چند دستور العمل در بدست آوردن نمودار جریان داده ها :

1- نمودار در سطح صفر باید نرم افزار / سیستم را به عنوان یک حباب منفرد نشان دهد .

2- ورودی و خروجی اولیه باید به دقت ذکر شود .

3- پالایش باید با جداسازی فرآیندهای کاندیدا ، اشیای داده ای و مخزن های داده ای که قرار است در سطح بعد نشان داده شوند ، آغاز گردد .

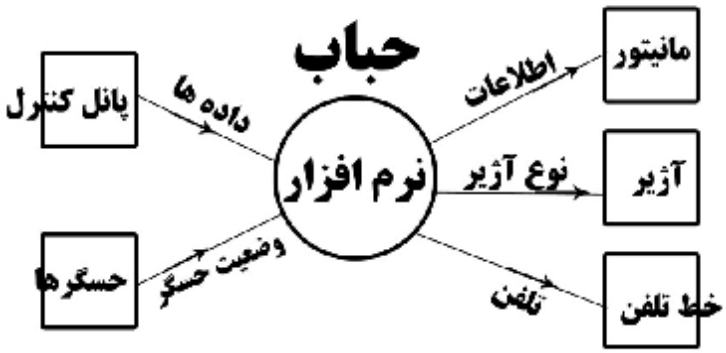
4- همه ی پیکان ها و حباب ها باید با نام مناسب نشانه گذاری شوند .

5- پیوستگی جریان اطلاعات باید از سطحی به سطح دیگر حفظ گردد .

6- هر بار تنها باید یک حباب پالایش شود .

اصولاً مهندسین تمایل دارند که نمودار جریان داده ها را بیش از حد پیچیده کنند و این هنگامی است که سعی می کنند جزئیات بیش از حد را زود تراز موعد نشان دهند .

شکل صفحه 203



ایجاد مدل جریان کنترل

در برخی برنامه ها مدل داده ها و نمودار جریان داده ها صرفاً برای به دست آوردن دید مناسب از خواسته نرم افزار لازم است . اما اگر بیشتر گفته شد ، گروهی بزرگی از برنامه های کار بردی توسط « رویدادها» اداره می شوند تا توسط داده ها ، بیشتر اطلاعات کنترلی تولید کنند تا گزارش و اطلاعات را پردازش کنند ؛ که این برنامه کاربردی علاوه بر مدلسازی جریان داده به مدلسازی جریان کنترلی هم نیاز دارد .

یک آیتم کنترلی یا رویداد به صورت مقدار بولی (0 و 1 ، True, False ، خاموش و روشن) با شرط ها (خالی ، پر ، گیر کرده) پیاده سازی می شود .

عوامل مؤثر در انتخاب رویداد

همه ی حس گرها ، شرایط وقفه ، شرایط داده و همه ی کلیدهایی که توسط اپراتور فعال می شوند ، فهرست کنید .

همه ی آیتم های کنترلی را به عنوان ورودی / خروجی برای تعیین مشخصات کنترلی مرور کنید .

رفتار سیستم را با شناسایی حالات توصیف کنید .

جا افتادگی ها و خطاهای رایج را بررسی کنیم ؛ مثلاً از خود بپرسیم : « آیا از این راه می توانم به این حالت برسم یا از آن خارج شوم ؟ » .

مشخصات کنترل

رفتار سیستم را به دو شیوه نشان می دهد :

1- CSPEC:حاوی یک نمودار حالت است که رفتار را به صورت ترتیبی مشخص می کند و این مشخصات حاوی یک جدول فعال سازی برنامه ها (مشخصات ترکیبی رفتار) است . CSPEC رفتار سیستم را توصیف می کند ولی درباره ی کار کرد داخلی فرآیندهایی که در نتیجه ی این رفتار فعال می شوند هیچ اطلاعاتی نمی دهد .

2- جدول فعال سازی فرآیند ها (PAT) : که این جدول اطلاعات موجود در نمودار حالت را در حیطه فرآیند ها و نه حالت ها نشان می دهد . یعنی نشان می دهد کدام فرآیند ها (حباب ها) در مدل جریان ، هنگامی فراخوانی می شود که رویدادی رخ دهد . از PAT می توان به عنوان دستور العملی برای نشان دادن فرآیندهایی که طراح باید فایل اجرایی آن را ایجاد کند ، استفاده کرد .

ایجاد مدل رفتاری

برای ایجاد این مدل باید مراحل زیر اجرا شود :

1- ارزیابی همه ی موارد برای درک کامل تعامل های داخل سیستم
2- شناسایی رویدادهایی که این تعامل را اداره می کنند
3- ایجاد یک دنباله توالی برای هر Use case
4- ساخت یک نمودار حالت برای سیستم .
5- مرور مدل رفتاری برای نشان دادن درستی و سازگاری .

شناسایی رویداد به کمک Use case

هر گاه سیستم و کنش گری به تبادل اطلاعات بپردازند ، یک رویداد رخ می دهد . رویداد ، اطلاعات تبادل شده نیست بلکه حقیقت این است که اطلاعات ، تبادل شده است .

برای مثال : کاربر یک کلمه 4 رقمی را با صفحه کلید وارد می کند . این کلمه با کلمه ی عبور معتبر نگهداری شده در سیستم مقایسه می شود . اگر عبور وارده صحیح باشد ، منتظر کنش بعدی می ماند ولی اگر صحیح نباشد قاب کنترل به صدا در می آید و reset می شود .

نوع دوم نمایش مدل رفتار ، نمودار ترتیب است که نسخه ای خلاصه شده از Use case است . نشان می دهد که رویدادها چگونه باعث گذار یک شی ء به شیء دیگر می شوند . پس از شناسایی رویداد با بررسی Use case ، مدلساز یک نمودار ترتیب ایجاد می کند . این نمودار کلاس های کلیدی و رویدادهایی را نشان می دهد که باعث جریان یافتن رفتار از کلاسی به کلاس دیگر می شوند .

الگویی برای مدلسازی خواسته ها

الگوهای نرم افزاری ، ساز و کارهایی هستند برای آگاهی از دامنه به گونه ای که بتوان هنگام مواجهه با مسأله ای جدید آن ها را دوباره به کار برد . نویسنده ی الگوی تحلیل ، الگو را ایجاد نمی کند ، بلکه آنرا به موازات اجرای کار کشف می کند .

Use case

اصلی ترین عنصر در توصیف مدل خواسته هاست . مجموعه ای یکپارچه از Use case به عنوان مبنا و اساس برای کشف الگوی تحلیل است .

SAP(الگوی تحلیل معنا شناختی)

الگویی است که مجموعه ی کوچکی از Use case یکپارچه را توصیف می کند که به همراه یکدیگر یک کاربرد کلی و پایه ای را توصیف می کند .

برنامه های تحت وب

تحلیل خواسته زمان می برد اما حل مسئلی اشتباهی از آن هم بیشتر وقت می برد . اما برای هر برنامه نویس تحت وب این سوال پیش می آید که آیا مطمئنی خواسته های مسئله را می دانی ؟ اگر مثبت باشد مدل سازی امکان پذیر است و اگر منفی مدل سازی باید انجام شود ؛ که به عوامل زیر بستگی دارد :

1- اندازه و پیچیدگی نسخه های برنامه های تحت وب
2- تعداد مشتریان و طرف های ذینفع
3- اندازه تیم برنامه های تحت وب
4- میزان همکاری قبلی اعضای تیم برنامه های تحت وب
5- میزان بستگی مستقیم موفقیت سازمان به موفقیت برنامه های تحت وب

هر چقدر پروژه کمتر شود تعداد طرفهای ذینفع ، یکپارچکی و تحلیل کمتر می شود .

ورودی در مدل سازی خواسته ها

اطلاعاتی هستند که طی فعالیت برقراری ارتباط به دست می آیند و شامل هر چیزی از نامه های الکترونیکی غیر رسمی گرفته تا یک خلاصه پروژه مشروح با سناریوهای کاربرد جامع و مشخصات کامل محصول .

خروجی در مدل سازی خواسته ها

در برنامه های تحت وب به پنج گروه تقسیم می شوند .

مدل محتوا : طیف کاملی از محتوایی است که قرار است برنامه ی تحت وب فراهم آورد که این محتوا عبارت است از داده های متنی گرافیکی ، تصاویر ، ویدئو و داده های صوتی .

مدل تعامل ها : شیوه تعامل کاربران با برنامه های تحت وب

مدل عملیاتی : عملیاتی را تعریف می کند روی محتوای برنامه های تحت وب انجام می شوند و قابلیت های پردازشی و مستقل از محتوا و در عین حال ضروری برای کاربر را توصیف می کند .

مدل گشت و گذار : راهبرد کلی گشت و گذار را برای برنامه های تحت وب تعریف می کند

مدل پیکر بندی : محیط و زیر ساختی را توصیف می کند که برنامه های تحت وب در آن قرار داده می شود.

مدل محتوا

این مدل حاوی عناصر ساختاری است و شامل اشیاء محتوایی و همه ی کلاس های تحلیل است . موجودیت قابل مشاهده از دید کاربر که در تعامل کاربر با برنامه های تحت وب ایجاد یا دستکاری می شود . محتوا را قبل از پیاده سازی برنامه های تحت وب یا حین ساخته شدن یا پس از عملیاتی شدن برنامه می توان گسترش داد . این محتوا از طریق مرجع گشت و گذار در ساختار کلی برنامه های تحت وب گنجانده می شود . یک شیء می تواند توضیح رویداد خبری ، عکسی از یک رویداد ورزشی ، پاسخی به یک کاربر در یک میز گرد ، نمایش پویا نمایی شده از لوگوی یک شرکت ، یک قطعه ویدئو یا صداگذاری باشد (متن ،عکس ،صدا ، تصویر).

برای هر محتوایی که از چند شیء محتوایی و آیتم داده ای تشکیل می شود ، می توان یک درخت داده ایجاد کرد . درخت داده ها برای تعریف روابط سلسله مراتبی میان اشیاء محتوایی و فراهم ساختن ابزاری برای مرور محتوا توسعه می یابد . بطوریکه ناساگاری ها و جاافتادگی ها قبل از شروع طراحی کشف شوند .

مدل تعامل برای برنامه های تحت وب

اکثر برنامه های تحت وب گفتگو میان کاربر نهایی و قابلیت های عملیاتی ، محتوا یا رفتار برنامه را میسر می سازد . این گفتگو را با بکارگیری یک مدل تعامل میتوان توصیف کرد که از عناصر زیر تشکیل شده :

1) use case 2) نمودار ترتیب 3) نمودارهای حالت 4) نمونه های اولیه واسط کاربری

در بسیاری از نمونه ها از use caseبرای توصیف تعامل در سطح تحلیلی استفاده می شود (پالایش جزئیات بیشتر در طی مرحله طراحی وارد خواهد شد.) . اگر تعامل پیچیده و شامل کلاس چندگانه باشد ، می توان از نمودار استفاده کرد .

ایجاد نمونه اولیه ایده ی خوبی است که طی آن هر چه زودتر بتوانیم نمایش فیزیکی واسط کاربری را بازبینی کنیم ، احتمال رسیدن کاربران نهایی به آنچه می خواهند بیشتر می شود . در نمونه اولیه باید پیوندهای اصلی مربوط به گشت و گذار دربرنامه پیاده سازی شود و چیدمان کلی صفحه به همان صورت که قرار است ساخته شود ، به نمایش درآید .

مدل عملیاتی برای برنامه های تحت وب

برنامه های تحت وب قابلیت های محاسباتی و دستکاری داده را ارائه می دهند که می تواند بطور مستقیم با محتوا همراه باشد و هدف اصلی ، تعامل بین کاربر و برنامه است . این مدل عملیاتی 2 عنصر پردازشی دارد : 1- قابلیت عملیاتی که توسط تحت وب به کار بر نهایی ارائه می شود . 2- عملیاتی که در داخل کلاس های تحلیل قرار دارند و رفتارهای مرتبط با هر کلاس را پیاده سازی می کنند .

قابلیت هایی که کاربر قادر به دیدن آنهاست ، شامل کلیه ی قابلیت های پردازشی است که مستقیماً با کاربر آغاز می شود و ممکن است واقعاً با استفاده از عملیات داخل کلاس های تحلیل پردازش شود و از دید کاربر مخفی است و یعنی پیامد قابل مشاهده است .

در هر سطحی از انتزاع که باشیم میتوان از نمودار UML برای نمایش جزئیات استفاده کرد . اما در سطح تحلیل ،زمانی ، از نمودار فعالیت ، می توانیم استفاده کنیم که عملکرد پیچیده باشد. پیچیدگی برنامه نه در قابلیتِ قراهم شده ، بلکه در ماهیت اطلاعاتی است که قابل دستیابی اند و شیوه ای دستکاری آنهاست .

مدل پیکر بندی برای برنامه های وب

در برخی موارد مدل پیکر بندی مربوط به صفحات کلاینت سرور است . اما در برنامه های پیچیده ، انواع پیچیدگی پیکر بندی (مثلاً توزیع بار در میان چند سرور و قرار دادن معماری ها در نهان گاه بانک اطلاعاتی دور دست ، سرورهای چند گانه که به اشیای گوناگون موجود در یک صفحه وب سرویس می دهند) ممکن است بر تحلیل و طراحی تأثیر بگذارد . در وضعیت هایی که معماری پیچیده است از نمودار استقرار UML استفاده می شود .

مدلسازی گشت و گذار

در این مدلسازی چگونگی سیاحت گروه های کاربری از یک عنصر برنامه به عنصر دیگر در نظر گرفته می شود . که مکانیک این گشت و گذار بخشی از طراحی است . باید خواسته های کلی گشت و گذار را مورد توجه قرار دهیم . در همین راستا باید سوالاتی از خود بپرسیم :

1- اولویت ارائه عناصر چگونه است (یعنی نحوه ی دسترسی به یک سری عناصر آسانتر باشد).
2- آیا باید بر عناصر خاصی تأکید ورزید تا کاربران گشت و گذار خود را به آن متمایل کنند ؟
3- با خطاهای گشت و گذار باید چه کرد ؟
4- آیا باید گشت و گذار مرتبط به گروهی از عناصر مرتبط ، بر گشت و گذار بر یک عنصر اولویت داشته باشد ؟
5- گشت و گذار باید از طریق پیوندها قابل انجام باشد ، از طریق جستجو یا از طریق دیگر ؟
6- ....

این پرسش ها بخشی از تحلیل گشت و گذار است . ما و سایر مشتریان باید خواسته های کلی را برای گشت و گذار تعیین کنیم .

فصل8: مفاهیم طراحی

اهمیت طراحی در ان است که طراحی جایی است که در ان کیفیت نرم افزار تثبیت می شود.

طراحی در حیطه مهندسی نرم افزار

طراحی نرم افزارهسته اصلی نرم افزاررا تشکیل میدهد وبه کارگیری ان مستقل از نوع مدل فرایند نرم افزار مورد استفاده است طراحی و روشهای طراحی عبارتند از:

1 - طراحی داده های کلاسها 2-طراحی معماری 3-طراحی واسط 4-طراحی مولفه ها

فرایند طراحی

طراحی نرم افزار فرایندی مبتنی بر تکرار است که از طریق ان خواسته ها به (نقشه ای) برای ساخت نرم افزار ترجمه میشوند.

دستور العمل ها وصفات کیفیت نرم افزار

مک گلافین سه خصوصیت پیشنهاد میکند که به عنوان راهنمایی برای تکامل طراحی خوب به شمار می روند :

1-طراحی باید همه ی خواسته های صریح موجود در مدل خواسته ها را پیاده سازی کند وباید همه ی خواسته های ضمنی مطلوب طرفهای ذی نفع را پاسخ گوید.

2-طراحی باید ین راهنمای خوانا و قابل فهم (1)برای کسانی باشد که کدها را تولید میکنندو(2) برای کسانی که نرم افزار را ازمایش وبعدا پشتیبانی میکنند .

3-طراحی باید تصویر کاملی از نرم افزار باشد که دامنه های داده ی عملیاتی ورفتاری را از دیدگاه پیاده سازی به نمایش بگذارد.

صفات کیفیتی:

لت –باکارد یک مجموعه صفات کیفیتی برای نرم افزار توسعه داده است که به اختصار به عنوان(1)قابلیت عملیاتی (2)قابلیت کاربرد(3)قابلیت اطمینان-کارایی(4)قابلیت پشتیبانی.

شایان ذکر است که این صفات کیفیتی را باید همان زمان که طراحی شروع میشود مورد توجه قرار گیردنه پس از کامل شدن طراحی وشروع ساخت.

تکامل طراحی نرم افزار

تکامل طراحی نرم افزار فرایندی پیوسته است .کار طراحی اولیه بر ملاکهایی برای توسعه ی برنامه های پیمانه بندی شده و روشهایی برای پالایش ساختارهای نرم افزاربه شیوهای از بالا به پایین متمرکز است.همه روشها به یک سری خصوصیات مشترک دارند :

(1)سازوکاری برای ترجمه مدل خواسته ها به نمایش طراحی (2)یک نماد گذاری برای نمایش مولفه های عملیاتی و واسط های انها (3)ابتکاراتی برای پالایش وافرازو(4)دستورالعمل هایی برای ارزیابی کیفیت.

مفاهیم طراحی

ام. ای.جکسون گفته است :شروع فرد برای مهندسی نرم افزارزمانی است که اختلاف میان به کار انداختن یک برنامه ودرست انجام دادن ان را تشخیص دهد.

انتزاع (Abstraction)

هنگامی که راهکاری پیمانه ای را برای مساله در نظر می گیرید سطوح انتزاع متعددی ممکن است پیش آید: در بالاترین سطح انتزاع راه کار در قالب عبارت هایی کلی وبا استفاده از زبان محیط مساله بیان میشود در پایین ترین سطح ازانتزاع راهکاربه شیوه

ای بیان میشود که به طورمستقیم قابل اجرا وپیاده سازی است باتوسعه یافتن سطوح متفاوت انتزاع روی ایجادهردونوع انتزاع فرایندی وداده ای کارمیکنید. انتزاع فرایندی دستور العمل هایی است که وظیفه ای مشخص ومحدود دارند. مثال وازه بازکردن برای درها. انتزاع داده ای مجموعه ای از داده ها به نام مشخص است که شی داده ای را توصیف می کند.

معماری (Architecture)

معماری نرم افزار به ((ساختارکلی نرم افزار وشیوه هایی مربوط می شود که این ساختار باعث یکپارچگی مفهومی در سیستم میگردد.)) یکی از اهداف مهندسی نرم افزار بدست آوردن یک نمای معماری از سیستم است شاوگارلان مجموعه ای از خواص را توصیف می کنند که خوب است به عنوان بخشی از طراحی معماری در نظر گرفته شوند:(1) خواص ساختاری (2) خواص عملیاتی اضافی (3)خانواده های سیستم های مرتبط .

الگوها (patterns):

الگوی طراحی توصیفی است ازیک ساختار طراحی که یک مساله طراحی را درحیطه ای خاص حل میکند.

هدف هرالگوی طراحی فراهم ساختن توصیفی است که طراح به کمک آن بتواند تعیین کند که 1)آیااین الگوها ی کارفعلی قابل استفاده هست2)آیا این الگو قابل استفاده مجدد هست 3)آیا این الگو می تواند به عنوان راهنما یی برای توسعه یک الگوی مشابه ولی باساختارو عملکردمتفاوت عمل کند.

جداسازی دغدغه ها(Separation of concerns)

جدا سازی دغدغه ها یک مفهوم طراحی است که پیشنهاد میکند هرمساله پیچیده ای را میتوان بهتر حل کرداگر به قطعاتی تقسیم گردد که هر یک را بتوان به طور مستقل حل ویا بهینه سازی کرد.

پیمانه بندی (Modularity):

پیمانه بندی متداول ترین نمود جداسازی دغدغه هاست .نرم افزار به مولفه های جداگانه و دارای نام تقسیم میشود که گاه از انها با عنوان پیمانه یاد میشوداز انسجام این پیمانه ها خواسته- های مساله براورده میشود.

شکل 2-8نشان میدهد که تلاش (هزینه)لازم برای توسعه ی یک پیمانه نرم افزاربافزایش تعداد پیمانه هاکاهش می یابد. اگرمجموعه یکسانی ازخواسته ها را درنظربگیریم بیشتربودن تعدادپیمانه ها به معنای کوچکتر شدن اندازه ی هر پیمانه خواهد بود.ولی با رشد تعدادپیمانه ها تلاش(هزینه)مرتبط با انجام بخشیدن به این پیمانه ها نیزرشد میکند.

پنهان سازی اطلاعات(Information Hiding)

پیمانه هابایدطوری مشخص وطراحی شوندکه اطلاعات موجوددریک پیمانه نتوانددردسترس پیمانه های دیگردیگری قرارگیرد که به این اطلاعات نیاز ندارند. پنهان کردن به این معناست که پیمانه بندی اثربخش ازطریق تعریف یک مجموعه پیمانه های مستقل قابل انجام است. استفاده ازپنهان کردن اطلاعات به عنوان ملاک طراحی برای سیستمهای پیمانه ای هنگامی بیشترین مزا یا را به همراه خواهد داشت که اصلاحاتی طی ازمایش و سپس طی نگهداری نرم افزارلازم باشد .

استقلال عملیاتی(Functional Independence)

مفهوم استقلال عملیاتی نتیجه مستقیم جداسازی دغدغه ها پیمانه بندی ومفاهیم پنهان سازی اطلاعات و انتزاع است.استقلال عملیاتی با توسعه ی پیمانه هایی با عملکرد(یگانه)و(دوری جستن)از تعامل بیش ازحد با سایر پیمانه ها بدست می اید.به بیان دیگر باید نرم افزارراطوری طراحی کنید که هر پیمانه به زیرمجموعه مشخصی ازخواسته ها بپردازدواز نگاه سایربخشهای ساختار برنامه دارای واسطی ساده باشد.استقلال با استفاده ازدوملاک کیفیتی قابل ارزیابی است: یکپارچگی و اتصال .یکپارچگی نشان ازقدرت عملیاتی نسبی یک پیمانه دارد.اتصال نشان از استقلال در میان پیمانه دارد.

پالایش(Refinement)

پالایش مرحله ای یک راهبردطراحی ازبالا به پایین است که اولین بار توسط نیکولاس ویرث پیشنهادشد.پالایش وانتزاع مفاهیمی مکمل یکدیگرند.به کمک انتزاع میتوانید روال وداده ها را ازنظرداخلی مشخص کنیدولی نیاز (افراد متفرقه)به داشتن اگاهی از جزئیات سطح پایین را بر طرف میسازد.پالایش به شما کمک میکندتا با پیشرفت طراحی جزئیات سح پایین را اشکار کنید هردومفهوم شما رادر ایجادیک مدل طراحی کامل یاری میدهند.

جنبه ها(Aspects)

جنبه نمایشی از یک دغدغه ی پیش نیاز است .شناسایی جنبه ها به طوری که طراحی بتواند به صورت مناسب انها را ضمن انجام پالایش و پیمانه بندی دربرگیرداهمیت دارد.درحالت ایده ال هرجنبه بصورت پیمانه ای جداگانه (مولفه)پیاده سازی میشودنه به صورت تکه هایی ازنرم افزار که درسرتاسرچندین مولفه(پراکنده) و(درهم وبرهم) شده باشند.برای دستیبابی به این مقصود معماری طراحی باید ازسازوکاری برای تعریف یک جنبه پشتیبانی کند.یعنی پیمانه ای که پیاده سازی یک دغدغه رادرسرتاسرهمه ی دغدغه هایی دیگری پیش نیاز ان باشد میسرسازد.

باز ارایی(Refactoring)

یک فعالیت مهم طراحی که برای بسیاری از روشها ی چابک پیشنهاد شده است بازارایی است که تکنیکی برای سازماندهی مجدد به شمار میرود وطراحی (باکد) یک مولفه را ساده میکندبدون اینکه قابلیت عملیاتی رفتار ان را تغییر دهد. فاولر بازآرایی رابه این صورت تعریف میکند:((بازآرایی عبارتست از فرایند تغیر دادن سیستم نرم افزاری به گونه ای که رفتار خارجی کد(طراحی) تغیرنکندودرعین حال ساختار درونی آن بهبود یابد.))

کلاس های طراحی (Design classes)

پنج نوع متفاوت ازکلاس های طراحی می توان توسعه دادکه هرکدام لایه متفاوتی ازمعماری طراحی را نمایش می دهند:1)کلاس های واسط کار بری 2)کلاس های دامنه تجاری 3)کلاس های پردازش 4) کلاس های ماندگار 5)کلاس های سیستمی آرلو ونویشتات برای کلاس طراحی ((خوش فرم)) چهارمشخصه را برمیشمرد: 1) کامل وکافی 2) سادگی 3)یکپارچگی بالا 4) اتصال پایین

مدل طراحی (Design model)

مدل طراحی را از دوبعد متفاوت می توان در نظرگرفت . بعد فرآیندی تکامل مدل طراحی را به موازات اجرای وظایف طراحی به عنوان بخشی از فرآیند نرم افزار نشان می دهد . بعد انتزاعی سطح جزئیات را به موازات تبدیل هرعنصرازمدل تحلیل به یک مدل طراحی وسپس پالایش تکراری نمایش می دهد.

عناصر طراحی داده ها:

طراحی داده هاهمانند فعالیت های دیگر مهندسی نرم افزار یک مدل از داده ها ویا اطلاعات ایجاد میکند که در سطح بالایی از انتزاع نمایش داده میشود. در سطح مولفه های برنامه طراحی ساختمان های داده ها والگوریتم های مورد نیازبرای دست کاری آنها در ایجاد برنامه های کاربردی باکیفیت بالا اهمیت اساسی دارد.

عناصر طراحی معماری:

عناصر طراحی معماری هم دیدکلی از نرم افزاررا به دست میدهد. مدل معماری از سه منبع بدست می آید: 1)اطلاعات مربوط به دامنه ی کاربرد برای نرم افزاری که قرار است ساخته شود. 2)عناصرخاصی از مدل خواسته ها از قبیل نمودارهای جریان داده ها یا کلاس های تحلیل روابط وهمکاری های میان آن ها برای مساله موردنظر. 3) قابلیت دسترسی به سبک های معماری والگوهای معماری.

عنصرطراحی واسط ها

عناصرطراحی واسط ها برای نرم افزارجریانهای اطلاعات به درون وبیرون سیستم وچگونگی برقراری ارتباط میان مولفه های تعریف شده به عنوان بخشی از معماری رابه تصویرمیکشد. سه عنصر مهم درطراحی واسط ها وجود دارد:

1)واسط کاربری

2)واسط های خارجی باسایرسیستم ها دستگاها شبکه ها یا سایرتولیدکنندگان یامصرف کنندگان اطلاعات

3)واسط های داخلی میان مولفه های طراحی گوناگون. واسط به مجموعه ای ازعملیات گفته میشود که بخشی ازرفتاریک کلاس راتوصیف میکند ودستیابی به این اطلاعات را فراهم می اورد.

عناصرطراحی در سطح مولفه ها

طراحی در سطح مولفه ها برای نرم افزار توصیف کاملی است ازجزئیات داخلی هر مولفه ی نرم افزار برای نیل به این مقصود طراحی در سطح مولفه ها ساختمان داده ها برای کلیه اشیای داده ای محلی وجزئیات الگوریتمی برای کلیه پردازش هایی که در داخل مولفه رخ می دهد و واسطی که در دستیابی به همه ی عملیات ها (رفتارهای)مولفه را امکان پذیر می سازدتعریف می کند

عناصر طراحی در سطح استقرار

عناصر طراحی درسطح استقرار چگونگی تخصیص یافتن زیر سیستم ها وقابلیت های عملیاتی نرم افزار را در داخل محیط کامپیوتری را نشان میدهند که نرم افزار را پشتیبانی میکند.

فصل 9 : طراحی معماری

طراحی معماری نشانگر ساختمان داده ها و مولفه های برنامه ای است که برای ساخت یک سیستم کامپیوتری مورد نیازند.

سبک معماری که سیستم به خود می گیرد ساختار و خواص مولفه های تشکیل دهنده سیستم و روابط میان همه مولفه های معماری سیستم ، در این طراحی معماری در نظر گرفته می شود .

طراحی معماری با طراحی داده ها شروع می شود وسپس با به دست آوردن یک یا چند نمایش از ساختار معماری سیستم ادامه می یابد . سبک ها یا الگوهای معماری متفاوت تحلیل میشوند تا ساختاری به دست آید که بیشترین مناسبت را با صفات کیفیتی و خواسته های مشتری داشته باشد . پس از این که این سبک معماری انتخاب شد ، جزئیات این معماری با استفاده از یک روش طراحی معماری تعیین خواهد شد.

معماری نرم افزار

معماری نرم افزار عملیاتی نیست بلکه نمایشی است که به کمک آن می توانید 1- میزان اثربخشی طراحی را در براورده ساختن خواسته های بیان شده تحلیل کنید ، 2- در مرحله ای که اعمال تغییرات طراحی هنوز آسان است ، آلترناتیوهایی برای معماری در نظر بگیرید و 3- خطرات مرتبط با ساخت نرم افزار را کاهش دهید.

نکته کلیدی: معماری نرم افزار باید ساختار یک سیستم و شیوه همکاری داده ها و مولفه های عملیاتی با یکدیگر را مدلسازی کند.

اهمیت معماری در چیست ؟

- نمایش معماری نرم افزار ، برقراری ارتباط بین طرفهای ذینفع در توسعه یک سیستم کامپیوتری را میسر می سازد .
- معماری ، تصمیم گیریهای زودهنگامی را که بر همه ی کارهای بعدی مهندسی نرم افزار تاثیر عمیق می گذارند ، برجسته می سازد. و با همان میزان از اهمیت ، موفقیت نهایی سیستم را به عنوان یک موجودیت عملیاتی نمایان می سازد.
- معماری « یک مدل نسبتا کوچک و قابل درک از چگونگی ساخت یافتگی و چگونگی همکاری مولفه های آن تشکیل می دهد

توصیف های معماری

توصیف معماری در واقع مجموعه ای از محصولات کاری است که نماهای متفاوتی از سیستم را ارائه می دهد . توصیف معماری یک سیستم نرم افزاری است که باید خصوصیاتی را از خود نشان دهد .

نکته کلیدی: مدل معماری نمایی کلی از سیستم فراهم می آورد بطوریکه که مهندس نرم افزار می تواند آنرا به عنوان یک کل بررسی کند .

ژانرهای معماری

اگرچه اصول بنیادی طراحی معماری در تمامی انواع معماری کاربرد دارند ژانر معماری غالبا رویکرد معماری مشخصی را در ساختاری که قرار است ساخته شود دیکته می کند .

ژانرهای معماری برای سیستم های کامپیوتری شامل :

هوش مصنوعی ،تجاری و غیرانتفاعی،ارتباطی ،پردازش محتویات، دستگاه ها ، ورزش وتفریح ، مالی ، بازیها، دولتی ، صنعتی ، حقوقی ، پزشکی و .... می باشد.

سبک های معماری

نرم افزاری که برای سیستم های کامپیوتری ساخته می شود ، یکی از چند سبک معماری را از خود نشان می دهد هر سبک گروهی از سیستم ها را توصیف می کند که شامل موارد زیر می باشد:

1- مجموعه ای از مولفه ها که وظیفه ای برای سیستم به انجام می رساند .

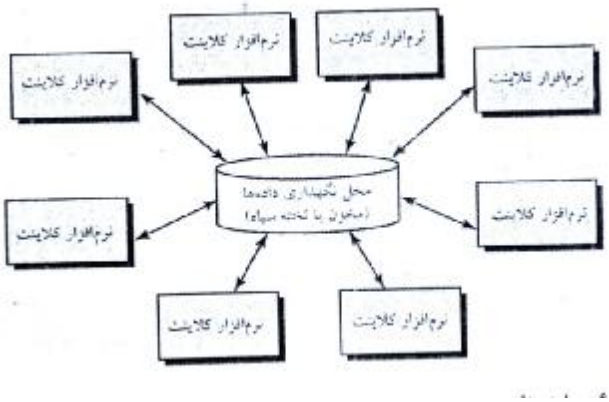
2- مجموعه ای از کانکتورها که «برقراری ارتباط، هماهنگ سازی و همکاری» میان مولفه ها را امکان پذیر می سازند .

3- قید و بند هایی که تعیین می کنند مولفه ها را چگونه می توان با هم منسجم ساخت و سیستم را ایجاد کرد .

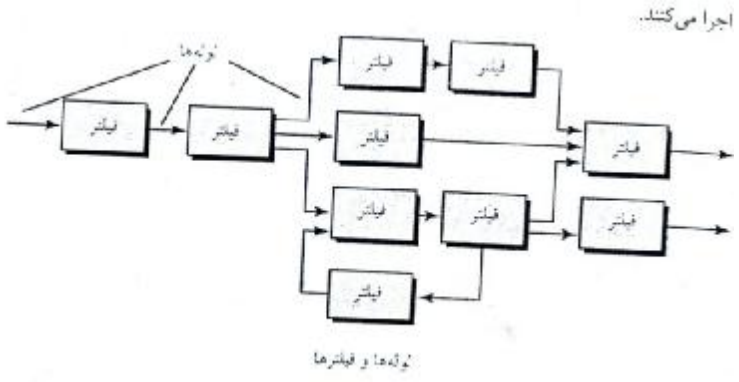
4- مدلهای معنا شناسی که طراح به کمک آنها می تواند خواص کلی سیستم را با تحلیل خواص بخش های سازنده آن درک کند.

طبقه بندی مختصر سبک های معماری

معماری های داده محور : محور این نوع معماری را یک انبار داده ها تشکیل می دهد ، که دستیابی به آن غالبا توسط مولفه های دیگری صورت می گیرد که داده های موجود در این انبار را به هنگام اضافه ، حذف یا به طریقی دیگر اصلاح می کند . شکل زیر نمونه ای از سبک معماری داده محور است که نرم افزار کالینت به یک مخزن مرکزی دستیابی دارد در برخی موارد این مخزن داده ها منفعل است به این معنی که نرم افزار کلاینت مستقل از هرگونه تغییراتی در داده ها یا کنش های سایر نرم افزارهای کلاینت به این داده ها دستیابی دارد با تغییر این رویکرد مخزن به یک تخته سیاه تغییر ماهیت می دهد که هرگاه داده های مورد نظر کلاینت تغییر کند، کلاینت را از آن آگاه می سازد.



معماری داده محور ، یکپارچگی و انسجام را ارتقاء می بخشد یعنی مولفه های موجود را می توان تغییر داد و مولفه های کلاینت جدیدی را به معماری افزود بی آنکه نیازی باشد نگران کلاینت های دیگر باشیم . بعلاوه داده ها را می توان با استفاده از ساز و کار تخته سیاه میان کلاینت ها تبادیل کذد «یعنی مولفه ی تخته سیاه به هماهنگ کردن انتقال اطلاعات میان کلاینت ها کمک می کند » . مولفه های کلاینت، فرایندها را مستقل از هم اجرا می کنند.

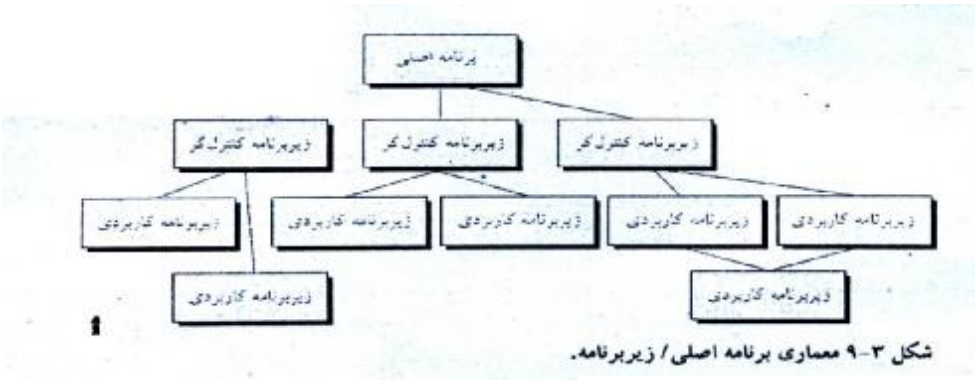


معماری جریان داده این معماری هنگامی به کار برده می شود که قرار باشد داده های ورودی از طریق یک سری مولفه های محاسباتی و دستکاری ، به داده های خروجی تبدیل شوند . الگوی لوله و فیلتر شامل مجموعه ای از مولفه ها موسوم به فیلتر ، می شود که توسط یک سری لوله به هم متصل می شوند ، این لوله ها داده ها را از مولفه ای به مولفه بعدی ارسال می کنند . هر فیلتر مستقل از مولفه های فوقانی و زیرین خود عمل میکند ، طوری طراحی می شود که داده های ورودی را در شکلی معین پذیرا باشد و داده های خروجی را با شکل خاص تولید می کند .

معماری های فراخوانی

به کمک این سبک معماری می توانید به ساختاری برای برنامه دست پیدا کنید که اصلاح و تغییر دادن ابعاد آن نسبتا آهسته باشد . در این گروه چند سبک فرعی وجود دارد:

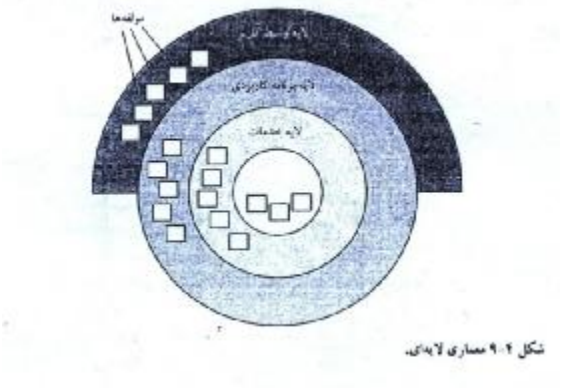
معماری های برنامه اصلی/ زیر برنامه . در این ساختار کلاسیک برنامه ، تابع به یک سلسله مراتب کنترلی تجزیه می شود که در آن یک برنامه «اصلی» چند مولفه از برنامه را فرا می خواند که هر یک به نوبه خود ممکن است مولفه های دیگری را فراخوانی کند.



شکل ۳-۹ معماری برنامه اصلی/ زیربرنامه.

معماری های فراخوانی روالهای راه دور. مولفه های معماری «برنامه اصلی/ زیربرنامه» در میان چندین کامپیوتر روی یک شبکه توزیع می شوند .

معماری لایه ای . ساختار اصلی یک معماری لایه ای در شکل زیر نشان داده شده است . تعدادی لایه های متفاوت تعریف می شود که هر یک عملیاتی را انجام می دهند و به طور تدریجی به دستورات ماشین نزدیک تر می شود . در لایه خارجی ، مولفه ها به عملیات واسط کاربر سرویس می دهند . در لایه داخلی ، مولفه ها ، ارتباط با سیستم عامل را برقرار می کنند . لایه های میانی خدمات و عملکرد های اصلی نرم افزار را فراهم می آورند.

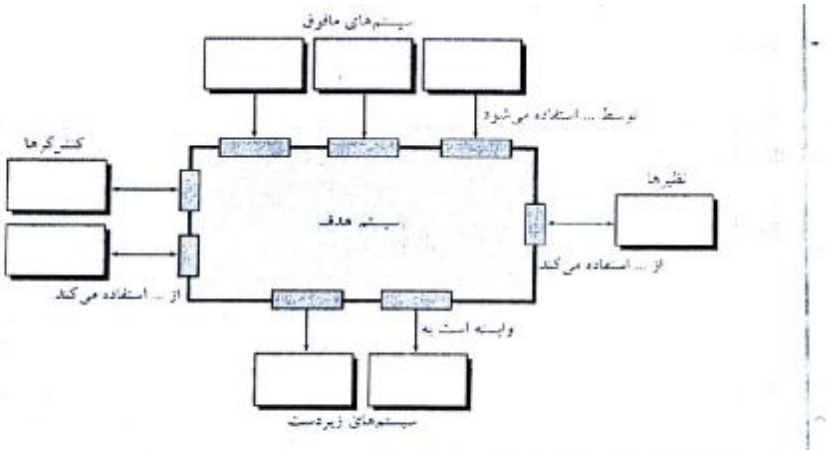


شکل ۹-۴ معماری لایه‌ای.

الگوهای معماری:

به مساله ای با کاربرد خاص . تحت مجموعه ای از محدودیت ها و قید و بند ها می پردازند ، و می تواند به عنوان مبنایی برای طراحی معماری عمل دند.

طراحی معماری: به موازاتی که طراحی آغاز می شود به نرم افزاری که قرار است توسعه یابد باید در حیطه ی کاری قرار داده شود ، یعنی طراحی باید موجودیت های خارجی را که نرم افزار با آنها تعامل دارد و نیز ماهیت تعامل را تعریف کند.



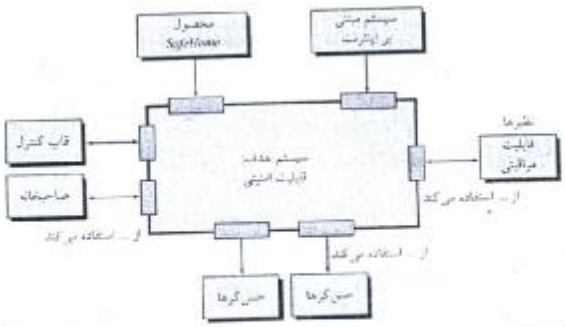
شکل ۹-۵ نمودار حیطه‌ی معماری.

نمایش سیستم در حیطه کاری

در سطح طراحی معماری، معماری نرم افزار برای مدل سازی شیوه ی تعامل نرم افزار با موجودیت های خارج از مرزهای خود از نمودار حیطه معماری(ACD) استفاده می کند. سیستم های مافوق: سیستم هایی که از سیستم هدف به عنوان بخشی از الگوی پردازشی سطح بالاتر استفاده می کنند . سیستم های زیردست : سیستم هایی که توسط سیستم هدف به کار گرفته می شوند و داده ها یا پردازش مورد نیاز برای کامل کردن قابلیت های عملیاتی سیستم هدف را می سازند .

سیستم های سطح نظیر: سیستم هایی که در سطح نظیربه نظیر تعامل دارد.

کنش گرها: موجودیت ها که با تولید یا مصرف اطلاعات مورد نیاز برای پردازش های ضروری با سیستم هدف تعامل دارند .
هر کدام از این موجودیت های خارجی از طریق یک واسطه با سیستم هدف ارتباط برقرار می کند .



شکل ۶-۹ نمودار جعبه‌ی معماری برای قابلیت امنیتی منزل در محصول SafeHome

تعریف نمونه های اولیه :

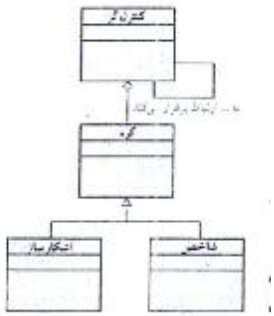
نمونه های اولیه کلاس یا الگویی است که یک انتزاع هسته ای نشان می دهد که در طراحی معماری برای یک هدف اهمیت حیاتی دارد که شامل :

گره : مجموعه یکپارچه از عناصر ورودی و خروجی که قابلیت امنیتی منزل را نشان می دهد برای مثال یک گره ممکن است از 1- حسگرهای گوناگون 2- انواع شاخصهای آژیر تشکیل شده باشد.

شاخص: انتزاعی که همه سازوکارها را نشان می دهد تا مشخص کند که شرایط کشتار رخ داده است .

آشکار ساز: انتزاعی که شامل همه تجهیزات حسگر می شوند و اطلاعات را به درون سیستم هدف تغذیه می کند .

کنترل گر: انتزاعی که نشان دهنده ی سازوکاری برای مسلح کردن یا غیر مسلح کردن یک گره است .



شکل ۷-۹ روابط UML برای نمونه‌های اولیه قابلیت امنیتی منزل.

پالایش معماری به مولفه ها :

- مدیریت ارتباطات خارجی
- پردازش پانل کنترلی
- مدیریت آشکارسازها
- پردازش هشدارها

روشهای تحلیل توازن های معماری :

موسسه مهندسی نرم افزار SEI یک روش تحلیل توازن های معماری توسعه داده است که برای معماری های نرم افزار ، یک فرایند ارزیابی تکراری تدارک می بیند . فعالیت های تحلیل طراحی زیر به صورت تکراری اجرا می شود :

1- جمع آوری سناریوها

2- روشن کردن خواسته ها

3- توصیف سبک ها

4- ارزیابی صفات کیفیتی با در نظر گرفتن هر صفت به طور مجزا

5- شناسایی حساسیت صفات کیفیتی

6- نقد معماری های کاندیدا

پیچیدگی معماری

یک تکنیک سودمند برای ارزیابی پیچیدگی کلی یک معماری عبارتند از در نظر گرفتن وابستگی های میان مولفه های موجود در معماری ، این وابستگی ها توسط جریان اطلاعات /کنترل موجود در سیستم به دست می آید . ژائو سه نوع وابستگی را پیشنهاد میکند :

1- وابستگی اشتراکی : نشانگر روابط میان مصرف کننده هایی که از یک منبع مشترک استفاده می کنند یا تولید کننده هایی که برای مصرف کننده ای مشترک تولید می کنند .

2- وابستگی جریانی : نشان دهنده ی روابط میان تولید کننده ها و مصرف کننده های منابع .

3- وابستگی مفید: نشانگر قید و بندهایی حاکم بر جریان نسبی کنترل در میان مجموعه ای از فعالیت ها .

زبان های توصیف معماری

زبان توصیف معماری ADL ابزار معنایی و نحوی را برای توصیف معماری نرم افزار فراهم می آورد.

نگاشت معماری ها با بکارگیری جریان داده ها :

یکی از روشهای مربوط به نگاشت معماری ، روش فراخوانی و بازگشت است . گذار از جریان اطلاعات به ساختار برنامه به عنوان بخشی از فرایند شش مرحله ای انجام می شود :

1- نوع جریان اطلاعات تعیین می شود .

2- مرز های جریان اطلاعات مشخص می شود .

3- DFD به یک ساختار برنامه ای نگاشت می شود .

4- سلسله مراتب کنترلی تعریف می شود .

5- ساختار حاصل با استفاده از موازین و اصول طراحی مورد پالایش قرار می گیرد.

6- توصیف معماری پالایش شده تعیین می شود .

نگاشت تبدیل

به مجموعه مراحلی گفته می شود که نگاشت از DFD با خصوصیات جریان تبدیل به یک سبک معماری مشخص را امکان پذیر می کند برای نگاشت طراحی جریان داده به یک معماری باید مراحل زیر طی کنیم:

مرحله 1- بازبینی مدل سیستم بنیادی

مرحله 2- بازبینی و پالایش نمودار جریان داده ها برای نرم افزار

مرحله 3- تعیین اینکه آیا DFD دارای ویژگیهای جریان تراکنشی است یا جریان تبدیلی .

مرحله 4- جداسازی مرکز تبدیل با مشخص کردن مرزهای جریان ورودی و خروجی .

مرحله 5- اجرای فاکتور بندی سطح اول یک کنترل گر اصلی

مرحله 6- اجرای فاکتور بندی سطح دوم

مرحله 7- پالایش نخستین تکرار معماری با استفاده از اصول طراحی برای بهبود بخشیدن به کیفیت نرم افزار.

هدف 7 مرحله فوق توسعه یک نمایش معماری از نرم افزار است .

همچنین یک عنصر مدل تحلیل ، مجموعه ای از نمودار های جریان داده هاست که جریان اطلاعات را داخل قابلیت امنیتی منزل توصیف می کند .

پالایش طراحی معماری

پالایش معماری باید به دنبال کوچکترین تعداد مولفه هایی باشد که با پیمانه بندی اثربخش سازگار باشد. و نیز به دنبال ساختمان های داده ای با کمترین پیچیدگی باشد که خواسته های اطلاعاتی را به طرزی مناسب پاسخگو باشد.

فصل 10 طراحی در سطح مولفه ها:

در هنگام طراحی معماری مجموعه کاملی از مولفه های نرم افزاری تعریف می شوند ولی ساختار های داخلی داده ها وجزئیات پردازش هر مولفه در سطحی از انتزاع نشان داده نمی شود که به کد نزدیک باشد در طراحی در سطح مولفه ها ساختمان داده ها الگوریتمها ساز وکارهای ارتباطی وخصوصیات واسط های هرمولفه از نرم افزار تعریف می شود. طراحی برای هر مولفه که در قالب های گرافیکی جدول یا نمادهای متنی ارائه میشودمحصول کاری تولید شده طی طراحی در سطح مولفه هاست.

مؤلفه:

مولفه به قطعات سازنده ی پیمانه ای نرم افزار کامپیوتر گفته میشود در زبان مدلسازی یکپارچه(OMG) به این صورت تعریف میشود،بخش پیمانه ای قابل استقرار وقابل تعویض از یک سیستم که جزئیات پیاده سازی را در خود دارد ومجموعه ی از واسط ها را ارائه میدهد سه دیدگاه مهم در باره ی ماهیت مولفه ها وجود دارد.

1)-دیدگاه شی گرا: دراین دیدگاه مولفه حاوی مجموعه ای از کلاس هاست که با یکدیگر همکاری میکنند هر کلاس در داخل یک مولفه دارای جزئیات کامل است به طوری که شامل کلیه صفحات و عملیات مرتبط با پیاده سازی کلاس میشود بنا براین باید اول از مدل خواسته ها شروع کنیم وبر جزئیات کلاس های تحلیلی وزیر ساختی اضافه نماییم .کلاسهای تحلیلی را برای مولفه هایی که بادامنه مساله مرتبطند وکلاس های زیر ساختی برای مولفه هایی که سرویس ها را برای دامنه مساله فراهم میسازند به کارمی بریم .

2)-دیدگاه سنتی: مولفه در این دیدگاه یک عنصر عملیاتی از برنامه است که منطق پردازش, ساختمان داده های داخلی و واسطی را در بر می گیرند که فرا خوانی مولفه ها وتحویل داده ها به آن را میسر می سازد مولفه سنتی(پیمانه) در داخل معماری نرم افزار جای دارند وبه عنوان یکی از سه نقش عمل می کند. 1- مولفه کنترلی 2-مولفه دامنه مساله 3-مولفه زیر ساختی .

مولفه های سنتی ومولفه های شی گرا از مدل تحلیل بدست می آیند عنصر مدل تحلیل به عنوان مبنایی برای بدست آوردن مولفه ها عمل میکند.

دیدگاه فرایندی :

درطی دودهه اخیر بر ساخت سیستمهایی تاکید شده که از مولفه های نرم افزاری یا الگوهای طراحی استفاده می کنند در اصل کاتالوکی از مولفه ها در سطح طراحی یا کد در اختیار شما قرار داده می شود. با توسعه معماری نرم افزار مولفه یا الگوهای طراحی را از کاتالوگ انتخاب می کنید ودر معماری خود جای می دهید این مولفه ها با در نظر داشتن قابلیت استفاده مجدد توصیف کاملی از واسط ها ووظایف و ارتباطات وهمکاری های مورد نیاز در اختیار شما قرار میدهد.

اصول پایه طراحی:

1- اصل باز- بسته (OCP): یک پیمانه باید برای عمل بسط باز وبرای عمل اصلاح بسته باشد.

2- اصل جایگزینی لیسکوف (LSP): زیرکلاس ها باید با کلاس های پایه خود جایگزین پذیر باشند یعنی مولفه ای که از یک کلاس پایه استفاده میکند اگر کلاس مشتق آن را به مولفه ارسال کندمولفه باید به درستی عمل کند.LSP حکم میکند که هر کلاس مشتق باید به قرار دادهای میان کلاس پایه ومولفه ای که از آن استفاده می کند بها دهد. قرار داد در این بحث پیش شرطی است که باید درست باشد تا مولفه از یک کلاس پایه استفاده کند وپس شرطی است که باید پس از اینکه مولفه از یک کلاس پایه استفاده کند درست باشد.پیش شرطها وپس شرطها با هم همخوانی دارند.

3- اصل وارونگی وابستگی(DIP) : ((به انتزاع متکی باشید نه به عینیت ها)) هرچه وابستگی یک مولفه به سایر مولفه ها ی عینیت یافته بیشتر باشد بسط وگسترش آن دشوارتر است.

4- اصل جداسازی واسط ها (ISP) :داشتن واسط های خاص کلاینت بسیار بهتر از یک واسط چند منظوره است.

اصول پکیج سازی دیگری نیز وجود دارد که برای طراحی در سطح مولفه ها قابل استفاده اندعبارتند از

1- اصل هم ارزی استفاده مجدد از نسخه هادر استفاده مجدد سنگ بنای ارائه نسخه هایجدید است.

2-اصل بستار مشترک(CPP) :کلاسها یی که با هم تغییر میکنند به هم تعلق دارند.

3-اصل استفاده مجددمشترک(CPP) :کلاسهایی که با هم دوباره استفاتده نمی شوند نباید در یک پکیج باشند.

یکپارچگی : به این معنی است که یک مولفه یا کلاس فقط صفات وعملیات هایی رادر خود پنهان سازی کند که رابطه تنگاتنگ با یکدیگر و با خود کلاس یا مولفه دارند چند نوع یکپارچگی وجود دارد:

1-عملیاتی : بوسیله عملیات ها نشان داده می شودوهنگامی رخ می دهد که مولفه ای یک محاسبه هدف دار انجام دهد وسپس نتیجه ای را بر گرداند.

2-لایه ای: بوسیله ی پکیج ها ,مولفه ها وکلاس ها نمایش داده میشود وهنگامی رخ میدهد که یک لایه ی بالایی به سرویسهای لایه پایینی دستیابی دارد ولی لایه پایینی به بالایی دسترسی ندارد

3-ارتباطی : همه عملیات هایی که به داد ه یکسان دستیابی دارند تنها در کلاس تعریف میشوند وفقط داده های مورد نظر ودستیابی به آنها وذخیره سازی آنها کانون توجه است.

اتصال: میزان کیفی از درجه اتصال کلاس ها به یکدیگر است هرچه وابستگی بیشتر باشد اتصال هم افزایش می یابد

انواع اتصال: 1- اتصال مشترک 2- اتصال کنترل 3-اتصال مهری 4-اتصال داده ای 5- اتصال فراخوانی روال ها 6- اتصال استفاده از نوع داده 7- اتصال واردات یا شمول 8- اتصال خارجی

اجرای طراحی در سطح مولفه ها:

مرحله 1: شناخت کلاس های متنا ظر با دامنه ی مساله

مرحله 2-همه کلاس های طراحی متناظر بادامنه زیر ساختار را شناسایی کنید

مرحله 3-کلاس هایی که به عنوان مولفه های قابل استفاده مجدد به دست نمی آیند تعیین کنید.الف)جزییات پیامها را مشخص کنید ب)برای هر مولفه واسط هایی مناسب مشخص کنیدپ)جزئیات صفت ها وانواع داده ها وساختمان داده های مورد نیاز را تعریف کنید ت)توصیف مشروح جریان پردازش در هر عملیات

مرحله4-توصیف منابع داده ای پایدار وکلاس های لازم برای مدیریت آنها

مرحله 5-نمایش های رفتاری مربوط به یک کلاس یا مولفه را بسط وتوسعه دهید.

طراحی در سطح مولفه برای برنامه های تحت وب:

1)توابع یکپارچه ای که محتوا را دستکاری میکنند یا پردازش محاسباتی یا دادهای را برای کاربر نهایی فراهم می سازند

2)پکیج های از محتوا وتوابع هستند که قدری از توانایی لازم را در اختیار کاربر نهایی قرار میدهند

طراحی در سطح مولفه برای برنامه های تحت وب :

1) طراحی در سطح مولفه ها: در محتوا در سطح مولفه ها آن چه کانون توجه قرار میگیرد اشیای داده ای وشیوه بسته بندی آنها برای ارائه به کاربر نهایی برنامه تحت وب است.
2) طراحی عملیاتی در سطح مولفه ها : برنامه های تحت وب نوین حاوی قابلیتهای پردازشی ای هستند که پیوسته بر پیچیدگی آنها افزوده میشود که عبارتند از 1-اجرای پردازش محلی برای تولید محتوا وقابلیت گشت وگذار به شیوه ای پویا 2-فراهم ساختن توانایی پردازش یا محاسبه داده ها که برای دامنه تجاری برنامه ی تحت وب مناسب باشد 3- فراهم ساختن امکان مراجعه به بانکهای اطلاعاتی پیچیده ودستیابی به آنها 4-برقراری واسط های دادهای با سیستمهای خارجی

طراحی مؤلفه ها های سنتی :

این ساختارها بر نگهداری از دامنه عملیاتی تاکید داشتند یعنی هر ساختار داده ای پیکر بندی منطقی قابل پیش بینی بود که ورود به آن از بالا وخروج از آن از پایین رخ میداد به طوری که خواننده با سهولت بیشتری میتوانست جریان رویه ای را دنبال کند این ساختار ها عبارتند از: ترتیب ,شرط وتکرار. ترتیب مراحلی از پردازش را پیاده سازی می کند که در تعیین مشخصات برنامه ضروری است و شرط تسهیلات مربوط به پردازش انتخاب شده را براساس یک رخداد منطقی فراهم می آورد و تکرار ایجاد حلقه را میسر می سازد .

1) طراحی با ابزار های گرافیکی: به کمک نمودارفعالیت می توانید ترتیب شرط وتکرار را به نمایش در آوید نمودا گردش همانند نمودار فعالیت از نظر تصویری کاملا ساده است .

2)نماد گذاری طراحی با روش جدولی: هنگامی که به مجموعه ای پیچیده از شرایط وکنش ها در یک مولفه برخوردید از جدول تصمیم گیری استفاده نمایید

3)زبان طراحی برنامه : زبان طراحی برنامه(PDL) که به آن انگلیسی ساخت یافته یا شبه کد نیز گفته میشود ساختار منطقی زبان برنامه نویسی را با توانایی بیانی یک زبان طبیعی در هم می آمیزد نحو اصلی در(PDL) باید شامل ساختارهایی برای تعریف زیر برنامه ها, توصیف واسط ها, اعلان داده ها ,تکنیک هایی برای سازماندهی بلوکها, ساختارهای شرطی ,ساختارهای تکرار وساختارها ی i/oباشد .PDL را می توان بسط داد تا واژه های کلیدی وپردازش همروند را در بر گیرد PDL شکل نهایی زبان را دیکته میکند مثال صفحه 320

توسعه مبتنی بر مولفه ها:

مهندسی نرم افزاری مبتنی بر مولفه ها فر آیندی است که بر طراحی وساخت سیستمهای کامپیوتری با به کارگیری مولفه های نرم افزار با قابلیت استفاده مجدد تاکید دارد .

1- مهندسی دامنه: هدف شناسایی, پیاده سازی ,کاتالوگ بندی وتوزیع مجموعه ای از مولفه های نرم افزاری است که در یک دامنه در نرم افزار فعلی ونرما فزارهای آینده کاربر دارد هدف کلی برقراری ساز وکارهایی است که مهندس نرم افزار به کمک آنها بتواند این مولفه ها را طی کار روی سیستم جدید وسیستمهای موجود به اشتراک بگذارد تا از آنها استفاده ی مجدد به عمل آید وشامل سه فعالتی اصلی می باشد تحلیل- ساخت وتوزیع

2- صلاحیت ,تطبیق وترکیب : مهندسی دامنه ,کتابخانه ای از مولفه های قابل استفاده ی مجدد را فراهم میکندکه برای مهندسی نرم افزار مبتنی مولفه ها مورد نیاز است برخی از این مولفه ها قابل استفاده مجدد به صورت داخلی وبرخی دیگر از برنامه های کاربردی موجود قابل استخراج هستند وتعدادی را نیز میتوان از یک شرکت سازنده تامین کرد .صلاحیت مولفه ها تضمین میکند که مولفه مورد نظر وظیفه ی مورد نیاز را انجام دهد ودر سبک معماری مشخص شده در سیستم می گنجد و ویژگی کیفی مورد نیاز برای برنامه ی کاربردی را فراهم می سازد .

عوامل صلاحیت مولفه ها عبارتند از :

-واسط برنامه کاربردی

–ابزارهای توسعه وانسجام بخشی مورد نیاز مولفه

-خواسته های زمان اجرا از جمله استفاده از منابع (مثل حافظه یا فضای ذخیره سازی)زمان بندی یا سرعت پروتکل شبکه

- خواسته های سرویس از جمله واسط های سیستم عامل و پشتیبانی از سایر مولفه ها

- ویژگی های امنیتی از جملهکنترل های دستیابی وپروتکل تایید

-فرض های پذیرفته شده در طراحی از جمله استفاده از الگوریتم های عددی یا غیر عددی

-مدیریت استثناها .

وظیفه ترکیب مولفه ها : عبارت است از مونتاژمولفه های تایید صلاحیت شده,تطبیق یافته و مهندسی شده جهت تشکیل معماری تعیین شده برای یک برنامه ی کاربردی .

3 -تحلیل و طراحی برای استفاده مجدد :

مفاهیم طراحی از قبیل انتزاع, پنهان سازی ,استقلال عملیاتیوپالایش و برنامه نویسی .ساخت یافته همراه با روش های شی گرا,آزمون ,تضمین کیفیت نرم افزار و روش های وارسی همگی در ایجاد مولفه های نرم افزاری با قابلیت استفاده ی مجدد سهم دارند . چند مساله کلیدی جهت طراحی برای استفاده مجدد :

داده های استاندارد :دامنه کاربرد باید برسی شود و ساختمان داده ها تعیین گردندپس همه ی مولفه های طراحی را می توان طوری مشخص کرد که از ساختمان داده ها ی استاندارد استفاده کنند

.پروتکل های واسطه استاندارد: سه سطح از پروتکل واسطه را باید وضع کرد .ماهیت واسطه های بین پیمانه ها ,طراحی واسطه های فنی خارجی و واسطه های میان انسان و کامپیوتر .

قالب های برنامه : یک سبک معماری انتخاب میشود و می تواند به عنوان قالبی برای طراحی معماری نرم افزار جدید عمل کند .

4-طبقه بندی و بازار یابی مولفه ها :

مفهوم مولفه های نرم افزار توصیفی است از آنچه که مولفه انجام می دهد .واسط مولفه ها به طور کامل توصیف می شود و معنا شناسی تعیین می شود مفهوم مولفه باید با هدف و مقصد مولفه ارتباط بر قرار کند .محتوای مولفه چگونگی تحقق یافتن مفهوم آن را توصیف می کند در اصل محتوا اطلاعاتی است که از دید کار بران اتفاقی پنهان می ماند و تنها کسانی که قصد اصلاحیا آزمون مولفه را دارند باید از آن مطلع باشند .حیطه جایگاه مولفه های قابل استفاده ی مجدد را در دامنه ی قابلیت کار برد آن تعیین می

کند .یعنی حیطه با مشخص کردن ویژگی مفهومی ، عملیاتی و پیاده سازی به مهندس نرم افزار این امکان را می دهد تا مولفه ی مناسبی را بیابد که خواسته های او را بر آورده سازد.

فصل یازدهم: طراحی واسط

واسط مورد نظر برای هر دستگاهی که باشد آنچه اهمیت دارد قابلیت استفاده است. در نتیجه مطالعاتی که کارشناسان فن آوری روی تعامل های انسانی به عمل آوردند دو مسأله غالب برملا شد:

1) مجموعه ای از قواعد طلایی

2) مجموعه ای از سازکارهای تعاملی تعریف شده. طراح نرم افزار به کمک 2- می تواند سیستم هایی بسازد که 1- را به طرزی شایسته پیاده سازی کند.

سپردن کنترل به کاربر: مندل [mang7] چند اصل طراحی را تعریف می کند که کنترل را در اختیار کاربر قرار می دهد. تعریف شیوه های تعامل به طریق که کاربر را وادارنه انجام عملیات غیر ضروری یا نامطلوب نکند. شیوه تعامل، حالت فعلی واسط است 2-انعطاف پذیری در تعامل 3- امکان توقف تعامل و خنثی کردن عملیات توسط کاربر 4- تعامل روان با پیشرفت سطح مهارت و امکان سفارشی کردن نوع تعامل5- پنهان کردن جزئیات فنی از دید کاربران موردی 6- طراحی برای تعامل مستقیم با اشیایی که روی صفحه ظاهر می شوند.

کاستن از بار حافظه: مندل اصول طراحی ای را معین می کند که واسط را قادر به کاهش دادن بار حافظه می کنند. 1- کاهش تقاضا برای حافظه کوتاه مدت 2- برقراری پیش فرض های با معنی 3- تعریف میانبرهای هوشمند 4- چیدمان بصری و دیداری واسط باید مبتنی بر استعاره جهان واقعی باشد: مثلاً در سیستم پرداخت حقوق باید از استعاره دسته چک و ثبت چک استفاده کرد تا کاربر فرایند پرداخت چک راهنمایی شود. 5- فاش کردن اطلاعات به شیوه ای تدریجی

سازگار ساختن واسط: 1- واسط باید همه ی اطلاعات بصری را سازماندهی که در تمام صفحات نمایش رعایت شود

2- راهکارهای ورودی به طور سازگار در سرتاسر برنامه استفاده شود3- راهکارهایی برای رفتن از وظیفه ای به وظیفه دیگر مندل مجموعه ای از اصول طراحی که به سازگار کردن واسط کمک می کند تعریف می کند.3- به کاربر اجازه دهید تا وظیفه کنونی را در زمینه معنی دار قرار دهد. 2- در میان خانواده ای از برنامه های کاربردی سازگاری را حفظ کنید. 3- اگر مدل های تعامل گذشته انتظارات کاربر را برآورده کرده است تغییری اعمال نکند مگر آنکه دلیل قانع کننده ای بر آن داشته باشید (مثلاً استفاده از Alt+s برای ضبط فایل) اگر در سیستم باشد از آن برای مثلاً تغییر مقیاس استفاده نشود باعث سردرگمی می شود.

تحلیل و طراحی واسط کاربر:

چهار مدل متفاوت هنگام طراحی واسط در نظر گرفته می شود. مهندس نرم افزار یک مدل طراحی ویک مدل کاربر می سازد. کاربر نهایی یک مدل ذهنی یا ادراک سیستم و پیاده کنندگان سیستم یک مدل پیاده سازی ایجاد می کنند.

کاربران را به سه گروه تازه کار، مطلع و متوسط و مطلع و دائمی گروه بندی می کنندکه دانش معنای متفاوتی (کم یا زیاد) نسبت به برنامه کاربردی و سیستم کامپیوتر دارند همه ی این مدل ها طراح واسطه را قادر می سازد که یک عنصر کلیدی از مهم ترین اصل طراحی واسطه کاربرد براورد سازد (شناخت کاربر، شناخت وظایف) فرآیند: فرآیندطراحی و تعلیل واسطه های کاربر، طبیعتی تکراری دارد که از مدل مارپیچی استفاده می کنیم که شامل چهار فعالیت : فرایند: فرایند طراحی و تحلیل واسط های کاربر، طبیعتی تکراری دارد که از مدل مارپیچی استفاده می کنیم که شامل چهار فعالیت 1- تحلیل و مدل سازی واسط 2- طراحی واسط 3- پیاده سازی واسط 4- اعتبار سنجی واسط. که هریک از این وظایف بیش از یکبار رخ می دهند وبا هر بار دور زدن

مارپیچ خواسته های بیشتری آشکار می شود. هدف از طراحی واسط تعریف مجموعه از اشیاء و عملیات واسط است که کاربر را قادر به انجام کلیه وظایف تعریف شده می سازد به طوریکه همه اهداف قابلیت استفاده سیستم برآورده شود. اعتبار سنجی واسط بر این موارد تأکید دارد. 1- توانایی واسط در پیاده سازی کلیه وظایف انجام شان و دستیابی به کلیه خواسته های عمومی کار 2- میزان سهولت استفاده و فراگیری واسط 3- تلقی کاربران از واسط به عنوان یک ابزار مفید در کارهای آنها.

یک اصل کلیدی در تمامی مدل های فرآیند مهندسی نرم افزار: درک مسأله قبل از اینکه اقدام به طراحی راهکار کنید.
تحلیل کاربران: برای همگرا کردن تصویر ذهنی کاربر و مدل تنها راه این است که خود کاربران و شیوه ی استفاده آنها از سیستم را درک کنید که اطلاعاتی از منابع گوناگون برای نیل به مقصود را میتوان به کاربرد و مصاحبه با کاربران، ورودی فروش، ورودی **بازاریابی، ورودی پشتیبانی**.
مدلسازی و تحلیل وظایف: use case شیوه های تعامل انسان با سیستم را توصیف می کند که به عنوان بخشی از تحلیل وظایف طوری توسعه داده می شود که چگونگی انجام وظایف خاص توسط کاربر نهایی را نشان می دهد که در اکثر موارد به سبک غیر رسمی و از زبان اول شخص نوشته می شود.
1- پرداختن به جزئیات وظایف: مهندس نیروی انسانی باید وظایفی را بشناسد که انسان ها در حال حاضر انجام می دهند و سپس آنها را در یک مجموعه مشابه از وظایف که در زمینه واسط کاربر پیاده سازی می شود نگاشت کند. روش کلی تحلیل وظایف هرچه باشد مهندس نیروی انسانی باید ابتدا وظایف را تعریف و طبقه بندی کند.
2- پرداختن به جزئیات اشیاء به جای توجه به وظایف استفاده از use case و سایر اطلاعات بدست آمده از کاربر را بررسی می کند و اشیای فیزیک مورد استفاده در طراحی داخلی را استخراج می کند. 3- تحلیل جریان کاری: به کمک این تکنیک می توانید چگونگی به انجام رسیدن یک فرایند کاری را در صورت وجود چند نفر درک کنید. مثال صفحه 345 کتاب ما را قادر به شناسایی خصوصیت مهم واسط قادر می سازد:
1- هرکاربر وظایف متفاوت را از طریق واسط پیاده سازی می کند.
2- طراحی واسط باید دستیابی به اطلاعات از منابع ثانویه و به نمایش درآوردن این اطلاعات امکان پذیر باشد.
3- بسیاری از فعالیت ها را میتوان با استفاده از تشریح اشیاء و تحلیل وظایف ریزتر که دو جزئیات بیشتریبه آنها افزود.
4- نمایش سلسله مراتبی.
تحلیل محتوای صفحه نمایش: طی این مرحله از تحلیل واسط فرصت بندی و زیبا شناسی محتوا مدنظر قرار خواهد گرفت.
تحلیل محیط کار: اینکه محیط کار کاربر پسندیده باشد یا خیر فقط در حد بهینه باشد.
مراحل طراحی واسط فرایندی مبتنی بر تکرار است در هر دو تکرار اطلاعات توسعه یافته در مرحله ی قبل پالایش و بر جزئیات آن افزوده می شود. مدل های فراوانی برای طراحی واسط پیشنهاد می شود ولی ترتیب وظایف طراحی هرچه باشد باید:
1) همواره قواعد طلایی را رعایت کند
2) چگونگی پیاده سازی واسط را مدل سازی کرد
3) محیطی را که واسط در آن به کار گرفته می شود مدنظر گرفت
هنگامی که اشیاء و عملیات تعیین شدند که یک مرحله مهم در طراحی واسط است ودر چند دور تکرار مورد شرح و تفسیر قرار گرفته. از نظر نوع، گروه بندی می شوند. هدف منبع و اشیای کاربردی مورد شناسایی قرار میگیرد یک شیء منبع کشیده می شود و در یک شیء مقصد رها می شود. بر خلاف فعالیت های دیگر طراحی چیدمان واسط یک فرایند تعاملی است. 3-4-11 مسائل طراحی، همواره چهار مسأله طراحی متداول مطرح می شود:

1- زمان پاسخ دهی سیستم، امکانات کاربر راهنما، مدیریت اطلاعات خطا، نشانه گذاری منوها و فرمان ها که این مسائل را باید در ابتدای طراحی به آنها پرداخت که ما را دچار مشکل نکند. توضیحاتشان:

1) مشکل اصلی بسیاری از برنامه کاربردی تعامل است دو ویژگی مهم طول و تغییر پذیر داردکه دومی مهم تر است

2) در بیشتر موارد راهنمای آنلاینی فراهم می سازند که به کاربر این امکان را می دهدتا بدون ترک واسط پاسخ پرسش خود را بگیرد و مشکل را حل کند

3) پیام خطا باید به زبان ساده شرح داده شود باید حاوی یک توصیه برای رهایی خطا باشد همه ی تبعات منفی خطا را خاطر نشان کند تا کاربر چک کند، همراه با صدای بوق یا رنگ مشخص خطا باشد. کاربر را مورد شماتت قرار ندهد.

دسترسی پذیری در برنامه های کاربردی: دسترسی پذیری برای کاربرانی که دچار مشکلات بدنی هستند به دلایل اخلاقی، قانونی و تجاری الزامی است مانند(w3 co3)، سایر دستورالعمل هایی برای فن آوری کمک رسان برای افرادی با نارسایی بینایی، شنوایی، حرکتی، گفتاری، یادگیری وجود دارد.

جهانی سازی: واسط کاربر باید طوری طراحی شود که فقط مناسب آن محل نباشد در سرتاسر جهان امکان استفاده از آن وجود داشته باشد و به آنها تحویل داده شود. استاندارد یونیکه [unio3] برای پرداختن به چالش مدیریت ده ها زبان طبیعی با صدها کاراکتر و نماد توسعه یافته است. 5-11 طراحی واسط برنامه کتاب: همانند طراحی واسط برای نرم افزارهای سنتی، طراحی واسط برای برنامه های تحت وب نیز ساختار و سازمان دهی واسط کاربر را توصیف می کند و شامل نمایشی از چیدمان صفحه نمایش، تعریف شیوه های تعامل و توصیف سازوکارهای گشت و گذار می شود. مجموعه ای از اصول طراحی ویک جریان کاری طراحی، طراح برنامه ی تحت وب را در طراحی می سازد. کارهای کنترلی واسط و چیدمان آن راهنمایی خواهند کرد. تونیروتزی به منظور طراحی واسط هایی برای برنامه تحت وب یک مجموعه اصول طراحی مطرح می کند که بر سایر اصول برتری دارد: 1) پیش بینی: برنامه ی تحت وب باید طوری طراحی شود که حرکت بندی کاربر پیشنهاد کند. 2) ارتباطات واسط باید وضعیت هرکدام از فعالیت های آغاز شده توسط کاربر را انتقال دهد یا آشکار یا با ظرافت 3) سازگاری هر ویژگی واسط باید به شیوه ای پاسخ دهد که انتظارات کاربر سازگار باشد. 4) خود مختاری کنترل شده: واسط باید حرکت کاربر را در سرتاسر برنامه تحت وب تسهیل کند و قراردادهای گشت و گذار وضع شده را تقویت کند. 5) بازدهی. 6) انعطاف پذیری: واسط باید به قدر کافی انعطاف پذیر باشد تا کاربران به انجام وظایف خود چه مستقیم وچه کاوشی در برنامه قرار دهد. 7) کانون توجه: واسط باید وظایفی را کانون توجه قرار دهد که کاربر در دست دارد. 8) قانون فیت: زمان لازم برای رسیدن به یک هدف تابعی است از فاصله تا آن هدف و اندازه آن قانون فیت: روش مؤثری برای مدل سازی حرکت های سریع و هدفمند است که در آن یک دستگاه فرعی از سکون در موقعیت شروع حرکت خود را آغاز و در ناحیه هدف دوباره به سکون میرسد. 9) اشیای واسط انسانی: منظور کتابخانه ای گسترده از اشیای واسط انسانی هر شیء که کاربر نهائی بتواند بشنود ببیند، لمس کند یا درک کند در این کتابخانه تأمین می شود. 10) کاهش تأخیر. 11) قابلیت یادگیری. 12) استعاره ها: واسطی که از یک استعاره تعاملی استفاده می کند راحت تر قابل فراگیری و قابل به یادگیری است. استعاره باید از تصاویر و مفاهیمی از تجربیات کاربر را به کمک گیرد.13) حفظ انسجام محصول کاری: یک محصول کاری باید به طور خودکار ضبط شود تا اگر خطایی رخ داد محتوای آن از بین نرود 14) خوانایی. 15) وضعیت پیگیری: وضعیت تعامل کاربر باید پیگیری و ضبط شود تا کاربر بتواند کار را از جایی که رها کرده رفته و برگشته ادامه دهد 16) گشت و گزار مرئی

وظایف زیر نشانگر یک جریان کاری مقدماتی برای طراحی واسط برنامه ی تحت وب هستند :

1- مرور بر اطلاعات موجود در مدل خواسته ها و پالایش آن در صورت نیاز.

2- تهیه ی اتودی اولیه از چیدمان واسط برنامه ی تحت وب .

3- کاشت اهداف کار بر درکنش های ویژه واسط

4- تعریف مجموعه ای از وظایف کار بر که به هرکنش مربوط می شود.

5- تهیه استوری بورد برای هر کنش واسط

6- پالایش چیدمان واسط و استوری بوردها با استفاده از ورودی حاصل از طراحی زیبایی شناختی

7- شناسایی اشیایی از واسط کاربر که برای پیاده سازی واسط ضرورینند.

8- توسعه ی یک نمایش رویه ای از تعامل کاربر با واسط

9-توسعه ی یک نمایش رفتاری از واسط

10- توصیف چیدمان واسط برای هر حالت

11-پالایش و مرور مدل واسط در مرور واسط «قابلیت استفاده » باید کانون توجه قرار گیرد.

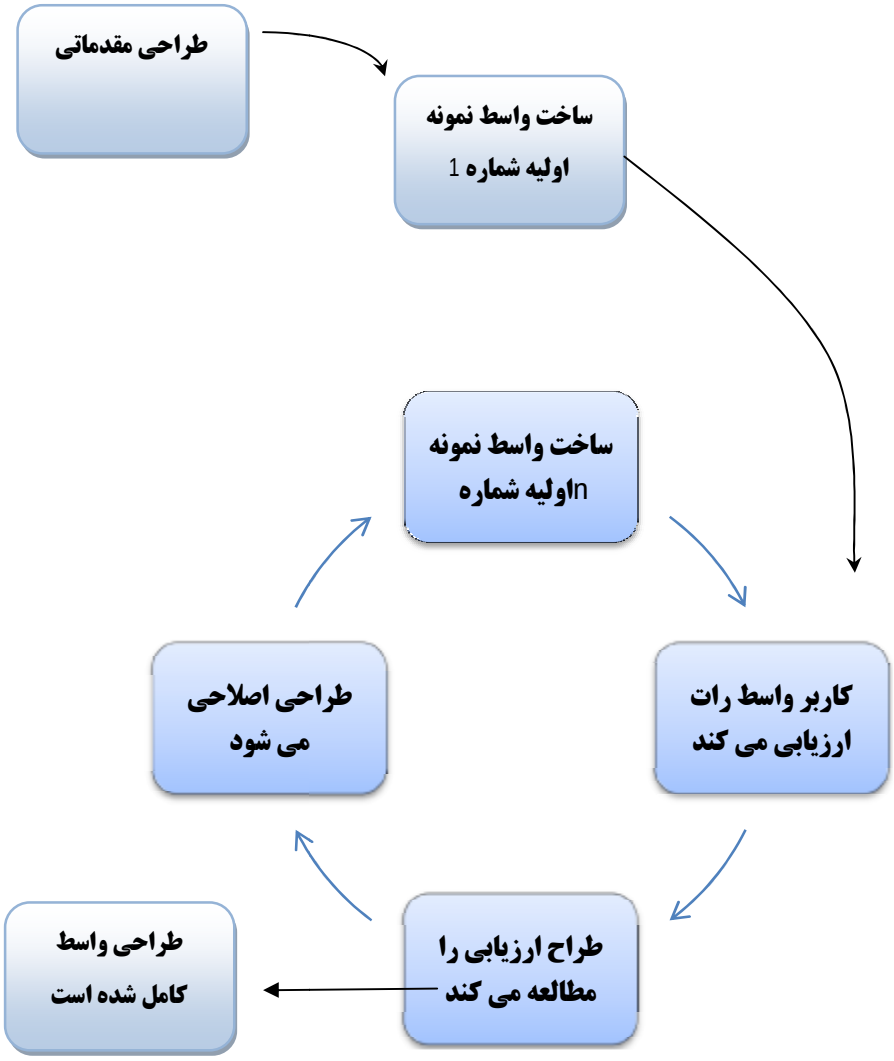
ارزیابی طراحی : هنگامی که نمونه ی اولیه ی عملیاتی واسط کاربر تهیه شده باید آن را ارزیابی کنید تا معلوم شود که آیا نیازهای کاربر را برآورده می سازد یا خیر/ ارزیابی شامل طیفی از رسمیت است که از غیر رسمی تا رسمی می کتواند در تغییر باشد . چرخه ی ارزیابی تا زمانی ادامه می یابد که دیگر نیازی به اصلاحات بیشتر در طراحی واسط نباشد. اگر مسائل بالقوه را زود هنگام شناسایی و تصحیح کنیم تعداد چرخه های ارزیابی کاهش می یابد و زمان توسعه کوتاه می شود و ملاک هایی نیز برای ارزیابی می توان در اولین مرورهای طراحی به کاربرد.

در آخر

واسط کاربر ، پنچره ای فرا روی نرم افزار است . واسط در بسیاری موارد، ادراک کاربر از کیفیت سیستم را شکل می دهد اگر این پنجره غبار گرفته شود، موج دار شود یا شکسته شود، کاربر ممکن است سیستمی پر قدرت را پس بزند.

شکل 5-11

چرخه ارزیابی طراحی واسط


طراحی مقدماتی
ساخت واسط نمونه
اولیه شماره 1
ساخت واسط نمونه
اولیه شماره n
کاربر واسط رات
ارزیابی می کند
طراح ارزیابی را
مطالعه می کند
طراحی اصلاحی
می شود
طراحی واسط
کامل شده است