Subject:
Year. Month. Date. ( )

مهندسی نرم افزار ۱

مهندس پوروطن

نگارنده: الهه سلطان آبادی

Subject:
Year. Month. Date. ( )

Subject:
Year. Month. Date. ( )

۱۲۴ ۱۱۲
۱۲۱ ۲۰۶
۱۹۶ ۱۰۵

مهندسی نرم افزار:

برای انجام هر کاری یک عنوان وجود دارد که موضوع کار را مشخص می کند. از زمان مشخص شدن عنوان تا رسیدن به نتیجه مطلوب مراحلی به صورت تدریجی و منظم طی می شود که حاصل یک کار جدید و نوین خواهد بود. پس رسیدن به نتیجه عملکرد انفجاری نیست. شیوه و مراحل رسیدن به نتیجه همان مهندسی نرم افزار است.

دو نوع مدیریت در مهندسی نرم افزار وجود دارد:
۱- process management ← مهندسی نرم افزار (نرم ۱)
۲- project management ← مهندسی IT (نرم ۲)

برای رسیدن به مقصد یک vision (دیدگاه) انتخاب می کنیم که مراحل بعدی بر اساس آن انتخاب می شود. در اینجا دو نوع دیدگاه وجود دارد: ۱- structure Analysis ۲- object_oriented Analysis

نرم افزار: یک سری نقل و انتقالات حاکم بر سیستم ها که ماهیت فیزیکی ندارند ولی output های مختلفی دارد که شامل ۱- برنامه ۲- مستندات ۳- Data می شود. اگر این نقل و انتقالات قابل مشاهده باشند برنامه ایجاد می شود.

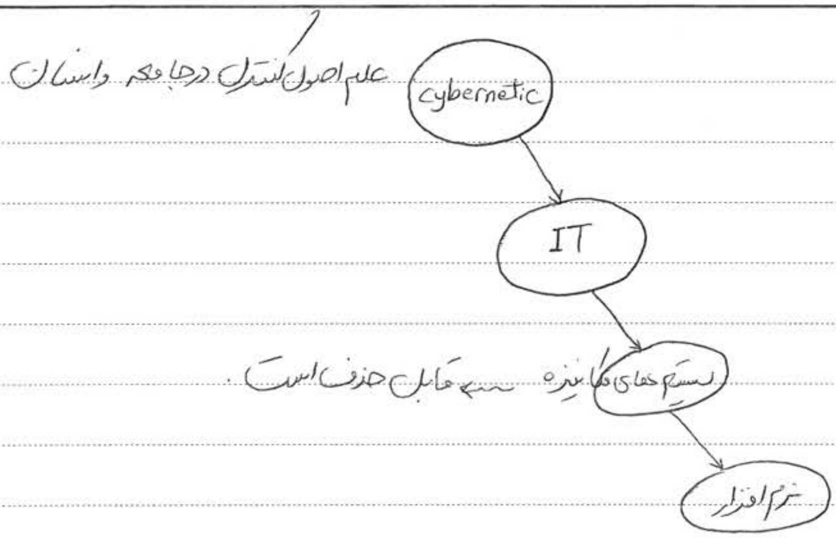
مستندات: مطالعات هایی که منجر به تولید برنامه نرم افزاری می شود مثل شرکت وزارت نفت که می خواهد متناسب با نیاز خود کامپیوتر بخرد اینکه ساختار سخت افزاری آن چه ویژگی هایی داشته باشد تا پاسخگوی این نیازها باشد یک document ایجاد می کند در نتیجه خروجی برنامه نیست.

داده: مانند بورس کالا که اقدام به فروش بعضی کالا می کنند. تبدیل کالا به سند کالایی که قابل معامله باشد از جمله کارهای بورس است. روش های معامله مختلفی از جمله ۱- پول نقد و انتقال کالا ۲- پیش فروش ۳- نسیه وجود دارد. برای هر یک از این روش ها آیین نامه هایی نوشته می شود که همان خروجی از نوع داده است.

تمامی روش ها در ساختار داده یک پیمایش در آن ساختار است یعنی هر مسئله را به یک ساختار داده تبدیل می کنیم و از طریق پیمایش به جواب می رسیم مثلاً در یک ساختار درختی هر یک از گره ها، node یا برگ می توانند یک جواب باشند. همه علوم نیاز به یک جهان بینی دارند تا راهی برای رسیدن به جواب پیدا کنند. جهان بینی علمی در حالت کلی همان cybernetic یا علم اصول کنترل در ماشین و انسان است.

PAPCO

Subject:
Year. Month. Date. ( )



براساس علم IT سیستم های اطلاعاتی شامل عناصر زیر هستند:

۱- تکنولوژی: میزان خودکاری و سطوح آن از اهمیت بالایی برخوردار است.

۲- نرم افزارها: همان برنامه ها هستند

۳- نیروی انسانی: افرادی که درگیر است با آن ها باید ببینیم آیا سطح علمی خوبی دارند یا نه؟ آموزش پذیر باشند

٤- رویه ها (procedure): همان فرهنگ است در واقع عمل و افعال ها بعد از مدتی که صورت گرفت شود به فرهنگ تبدیل می شود

۵- سازماندهی

این عناصر در موفقیت یک سیستم اطلاعاتی موثر است.

در programming ۳ حوزه مختلف وجود دارد که هر یک جهان بینی خاص خود را دارد.

1- Functional

2- procudure

3- object_oriented: برای اجزای موجود obj هایی را تعریف می کنیم همراه با ویژگی و ارتباط آنها. process وجود ندارد ولی حرکت از obj به obj دیگر یک زنجیره obj oriented ایجاد می کند.

PAPCO

Subject:
Year. Month. Date. ( )

4- database

5- logic : قوانینی وجود دارند که براساس مفروضات discovery knowledge می کنیم.

میزان اهمیت نرم افزار چقدر است؟

در همه فعالیت های اقتصادی و اجتماعی کاربرد دارد. همه رفتارها با این نرم افزارها پیوند خورده است.

تاریخچه نرم افزار:

در ابتدا هر کس کامپیوتر را می ساخت از آن نیز استفاده می کرد ولی با پیشرفت و افزایش استفاده از کامپیوتر باید افراد زیادی برای استفاده آموزش می دیدند که باید با ساختار و معماری آن آشنا باشند تا بتوانند با آن برنامه بنویسند. البته با پیشرفت نرم افزارها دادن امکانات نیازی به دانستن ساختار و معماری سخت افزار نبود.

مشکلات این پیشرفت ها عبارتند از:

۱- زمان گیر: تولید نرم افزار زمان گیرتر از چیزی است که ما پیش بینی می کنیم در پروژه ها نیز همین طور است.

۲- هزینه: هزینه در انتها بیشتر از انتظار ما می شود.

۳- همه خطاها و مشکلات نرم افزار قبل از تحویل قابل بررسی نیست.

٤- اندازه گیری: در اندازه گیری ها دچار مشکل می شویم و مقیاس ثابتی برای تعیین شاخص هایی مانند accessibility یا reliability یا درصد کار انجام شده و میزان کیفیت وجود نداشت.

ویژگی های نرم افزار:

۱- اولین چیزی که در نرم افزار مهم است مهندسی بودن آن است یعنی تغییر ماهیت وجود ندارد و تنها یک مسئله به زبان ریاضی بیان می شود در نتیجه نرم افزار وابسته به هیچ رشته خاصی نیست و در همه جا دارد می شود.

روی کاغذ بندی می شود ← document

همان حرف یادداشت می شود ← program

* توسعه داده می شود و به صورت معمول ساخته نمی شود.

PAPCO

نکته: محتوای اطلاعات و مشخص بودن، فاکتورهای مهمی در تعیین ماهیت کاربرد نرم افزار می باشند.
محتوا اشاره به محتوی و شکل اطلاعات مبادله شده دارد. مشخص بودن اطلاعات به قابل پیش بینی بودن ترتیب و زمانبندی اطلاعات اشاره دارد.

Subject: Year. Month. Date. ( )

۲- مستهلک نمی شود اما ممکن است در طول زمان کارایی اش را از دست بدهد و مشتری نیازش عوض شود ولی کهنه و فرسوده نمی شود. * نرم افزار در ابزار خاصی نیست.

۳- همه فرایندهای مهندسی component base است یعنی همه اجزا وجود دارد و فقط استفاده می شود و نیازی به ساخت و تولید اجزا نیست ولی در نرم افزار سفارشی است و براساس نیاز باید ویژگی و اجزای هر بخش مشخص شود.

* اگر صنعت سمت نصب قطعات و پیش ساخته می رود اکثر نرم افزارها بصورت مستقل ایجاد می شوند.

نرم افزارها به 7 حوزه تقسیم می شوند:

1. Application software: توجه به دستورهای انتقال، انشعاب و I/O دارد (control unit را دارد)

2. Sientific-engineering: بیشتر توجه به دستورالعمل های مستقل و رابطه ای دارد (ALU و cont واحد اصلی یا دسته عملی در Input/output است.

3. embeded software: نرم افزارهایی که داخل یک hardware قرار می گیرند تا کار خاصی انجام دهند.

4. product-line soft: به صورت general است و به منظور گروه خاص یا مشتری معلومی تولید شده است (مانند office)

5. web Application sof.

6. AI

ubiquitors:

software هایی گفته می شود که از نظر سازمانی، برنامه های مستقلی هستند ولی در پردازش ها با یکدیگر در ارتباط اند. منظور از ایجاد استفاده برای هدف خاصی است ولی در سیستم های توزیع شده یک نرم افزار در جاهای مختلفی استفاده می شود و در پردازش ها مرتب با یکدیگر در ارتباط اند.

open Source:

Application مناسب برنامه ای است که قابلیت ~~customise~~ Customise کردن براساس نیاز را داشته باشد.

PAPCO

Subject:
Year. Month. Date. ( )

این تغییرات می‌توانند شامل Data یا Functionality باشد که در صورتی که برنامه برای source تغییر Function ها در اختیار قرار گیرد opensource می‌نویسیم.

Net sources:

نرم‌افزارهای شبکه‌ای یا هر تکنولوژی جدیدی که به علت پراکندگی دستگاه‌ها از ارتباطات بیشتر استفاده می‌کند.

در ابتدا وقتی کامپیوتری‌ها برنامه می‌نوشتند خودشان استفاده از آن را بر عهده داشتند و تمام کامپیوترها متمرکز در محل آنها بود. فعالیت با این برنامه‌ها زیر نظر کامپیوتری‌ها و توسط آنها انجام می‌شد ولی از آن جایی که اطلاعات ممکن است محرمانه باشد باید کامپیوترها توانایی پراکنده بودن پیدا می‌کردند. اگر قرار بود همه‌ی کامپیوترها محرم باشند گزینش سخت می‌شود. با پخش شدن کامپیوترها شرایط نرم‌افزاری متفاوت می‌شود و باید قابلیت کارکردن در شبکه را داشته باشد.

نرم‌افزارهایی که از قبل وجود دارد امکان هماهنگی و تغییر متناسب با نیاز را ندارد. به طور مثال در شیوه‌ی ذخیره‌ی منابع پزشکی متناسب با داشتگی آنها در کشورهای اروپایی و آمریکایی متفاوت است در نتیجه امکان هماهنگ‌سازی و ترکیب این منابع وجود ندارد. از طرفی تغییر نرم‌افزار نیز امکان‌پذیر نیست زیرا محلی برای نگهداری همه‌ی داده‌های پیشین وجود ندارد. بودجه‌ی کافی برای ذخیره‌ی اطلاعات و تغییر و update آنها در دست نیست. مثلاً سایت وزارت خانه توسط کامپیوتری‌ها نوشته شد ولی کسی نبود که آن را به روز رسانی کند و برای این کار نیاز به تشکیل و سازمان داشتند.

یکی دیگر از مشکلات نرم‌افزار انتقال اطلاعات از نرم‌افزار قدیم به جدید است.

سه گروه با یک سیستم اطلاعاتی درگیرند: ۱- مدیران ۲- کاربران ۳- مهندسین و متخصصان

مدیران:

کم‌تخصص‌ترین افراد در مجموعه اند. وظیفه‌ی مشاوران و کارشناسان است که مدیر را از تصمیمات غیرعاقلانه باز دارد. به عنوان مثال در افزایش حمل و نقل راه آهن ایده‌های مختلفی را اعمال کردند از جمله دو طبقه کردن قطارها که به علت کوتاه بودن سقف تونل‌ها امکان‌پذیر نبود. - خرید لکوموتیو جدید با سرعت ۱۲۰k به جای ۶۰k که

PAPCO

Subject: Year. Month. Date.

* **توضیحات جزئیات:** در تولید نرم افزار جزئیات را به تعویق می اندازند گاهی در جهان ابتدا پروژه نمی توان پیش بینی تاریکی کرد یا جزییات ظاهری قابل تعیین نیست. گاهی این جزئیات فراموش شده و آنقدر به کمیات تبدیل می شود که وقایع نمی شود نباید نادیده گرفت. برای این جزئیات بی اندازه تعویق است و مشکلات در همین زمینه است. برای مثال را از خطوط بردی یک ترمیند ولی دقتی به ذرات نیرو در طرح کردند هدیه حرف مورد نیاز برابر با هزینه خرید تمام کوتوپتوها بود. این نشان دهنده ایده های بد کارشناسانه است.

* **توهمات مدیران:**

۱- **استانداردها:** خیال می کنند با گرفتن استانداردهای موجود پیشرفت سازمان محقق می شود ولی با وجود داشتن استانداردها دستگاه خراب وارد بازار می شود. سیاست در همه کشورها این است که پول گرفته شود ولی بازگردانده نشود (مانند شرکت ایران خودرو - بهمن)

۲- **پشتوانه به کتاب و مستندات:** پشتوانه مدیران به کتاب ها و مستندات موجود است درحالیکه خادمین شکل تخصص از اهمیت بالایی برخوردار است یعنی درآوردن تخصص از ذهن یک فرد و پخش آن در سازمان. ولی این کار انجام نمی شود. مراکز علمی موجود تبلیغاتی می کنند و دروغ خادمین گذاره است در نتیجه صرف مطالعات و بررسی ها دلیلی بر درست بودن آنها نیست و اعتباری به اطلاعات نیست.

۳- **پول در اختیار مدیران:** بسیاری از کارها با خرج کردن پول درست نمی شود. مثلاً در بازار بورس اگر شبکه را گسترده کنیم زمان ارسال حجم از زمان دریافت است یعنی هرکار زودتر در معامله شرکت کند برنده است به علت تاخیر در شبکه چنین مکانیزمی امکان پذیر نیست در نتیجه با گسترده کردن سرانه بورس و خرج کردن پول مکانیزم بورس درست نمی شود. البته کالاهای استراتژیک انگیزه عمل نمی شود و بیشتر بورس صوری است.

خودش توهم از مدیران است.

برای حل مشکلات موجود مدیران ۱- **outsource کردن منابع:** قراردادی با بیرون می بندند ولی آیا این کار نیازمند کنترل برای تحویل به موقع نرم افزار ندارد؟ شاید بتوان با پیگردهای قانونی شکایت کرد ولی درنهایت نتیجه مطلوب که تولید نرم افزار است محقق نمی شود.

* جزئیات را به تعویق می اندازند که این کار می تواند باعث Fail شدن این سیستم شوند (توضیحات بالا)

توهم مشتری

۴- **هر تغییر امکان پذیر است:** یکی از مشکلات نرم افزار این است که تغییر می دهند، چون نرم افزار است و می توان هر تغییری را کنترل کرد. بر اساس این ایده مدیران تغییرات غیر معقول را نیز در نظر می گیرند این توهم باعث سخت شدن کار می شود.

توهم مجری

۵- اگر در اجرای اولیه bug ندارد پس کار تمام است ولی در حقیقت کوچکترین و کم اهمیت ترین بخش coding است. میزان کیفی از همان شروع باید مشخص شود و نیازی به رسیدن پایان پروژه نیست.

PAPCO

داستان های مدیریتی: ۱- تأمین بودجه ۲- استاندارد سازی ۳- تولید هرگونه برنامه ۴- افزایش کارمندان ۵- دانلودی چرخه

Subject: داستان مشتری: ۱- انتظارات نادرست ۲- به تعویق انداختن جزئیات ۳- انعطاف پذیری دائمی نرم افزار کمک بودن

Year. Month. Date. ( ) داستان مجری: ۱- عدم دستیابی به کیفیت تا زمان اجرا ۲- عدم مستندسازی

این مسئله باعث دوباره کاری می‌شود. محصولی که مهندس نرم افزار می دهد برنامه ای است که در حال اجراست و تنها نوشتن برنامه مهم نیست.

۶- ایجاد مستندات را در نظر نمی گیرند در حالی که مستندات عامل نگهداری شدن تخصص است.

توهم مجری ← ولی با کیفیت مستندات در مرحله تحلیل می توان ادامه پروژه در مراحل بعدی را کنترل کرد.

با همه این اوصاف هدف ما این است که این مشکلات را برطرف کنیم.

فصل دوم:

زمانی که یک برنامه نوشته می شود ابتدا باید موضوعی به عنوان مشخص شود که در مهندسی نرم افزار آن را process می گویند یعنی انجام مراحلی که از نیاز به محصول برسیم.

4 بعد عمل یک پروژه عبارت است از:

project ← عوامل محیطی برای انجام درست مسئله

process ← برای انجام مسئله چه قدم هایی برداریم که به نتیجه برسیم.

product / problem ← نوع مسئله، میزان درگیری و کم و زیادی ها را مشخص می کند.

people ← نیروی انسانی نقش مهمی دارد و تعارضها نمی توانند جلوی آن را بگیرند (؟) پیش بروند.

برای رسیدن به محصول مطلوب process اهمیت زیادی دارد. تنها انجام عمل (Functionality) مهم نیست بلکه کیفیت نیز اهمیت دارد. process تضمین می کند که علاوه بر محصول، کیفیت خوب نیز ارائه می دهیم. یکی از مدل های موجود Agile است که در حقیقت process model نیست و تنها یک صفت بر روی process model است.

process model از لایه هایی تشکیل شده است تا به محصول مطلوب برسد.

tools
method
process
کیفیت

communication → planning → modeling → construction → deployment

نحوه جمع آوری اطلاعات خود یک method است.

هر روش مهندسی باید برروی یک ساختار سازمانی بر مبنای کیفیت قرار گیرد.

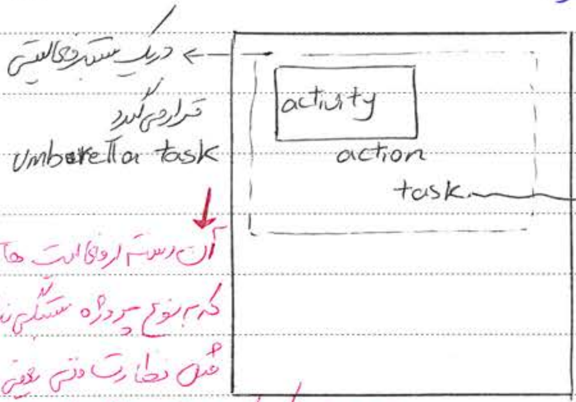
method: برای شروع یک process ابتدا باید به جمع آوری اطلاعات بپردازیم پس با جمع آوری اطلاعات method ها نیز تعریف می شوند. روش ها شامل مجموعه گسترده ای از کارهایی هستند که هر یک شامل امکان سنجی، تحلیل، طراحی، ایجاد برنامه، آزمایش و پشتیبانی می باشد.

مهندسی نرم افزار: به کارگیری روش سیستماتیک، اصولی، و قابل بررسی برای توسعه، اجرا و پشتیبانی نرم افزار می باشد
یعنی به کارگیری مهندسی در نرم افزار – مطالعه گرایش های بخش

Subject:
Year. Month. Date. ( )

tools : ابزارهایی که در جمع آوری اطلاعات استفاده می شود

تولید محصول خوب از همان ابتدا مشخص می شود هر process این مراحل را طی می کند و می توان با یکسری method های مطلوب در انتها به نتیجه دلخواه رسید. یک process یک framework دارد که تمام این فعالیت ها را مشخص می کند.

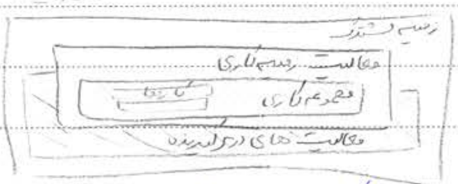


یک کار کوچک در communication است جمع آوری اطلاعات بر گروه های مختلف خود یک task محسوب می شود هر وظیفه یک هدف دارد.

آن در بستر از فعالیت ها که به نوع پروژه بستگی ندارد شامل نظارت دقیق، مدیریت اینکه کار به درستی انجام می شود یا خیر
(بدون توجه به اندازه و پیچیدگی)

کارهایی که در این بستر انجام می شود عبارت اند از:

۱. کنترل خطر
۲. مدیریت خطا ← خطاهایی که باعث تغییر می شود.
۳. جلسات رسمی
۴. اندازه گیری ها
۵. مدیریت تغییر ← هر کس قسمتی از آن را انجام می دهد تغییر یک قسمت روی کارهای قسمت ها اثر می گذارد. مدیریت سطح ها باید به گونه ای باشد که تغییرات سر روی کارگیری تاثیر نگذارد. این از بخش های مهم نرم افزار است.



در بستر پروسه ها همان روند جلو می رود ولی بعضی از قسمت های آن بر اساس نیاز یا نوع پروسه از اهمیت کم تری برخوردار است: (نوع محصول، نوع کاربرد)

شیوه برخورد با پروسه ها طبقه بندی های مختلفی دارد: (سطح بلوغ افراد در استفاده از process ها)

Level 0 incomplement

مربوط به افرادی است که هر حوزه ها را انجام می دهند و کارهای انجام شده به هدف اهداف نمی رسند هر practice ها را انجام می دهد. در حقیقت به فعالیت افرادی بستگی دارد.

Subject:
Year. Month. Date. ( )

Level 1 performed

توی project planning حوزه کاری است که در همه مراحل process وجود دارد. یعنی همه سطوح را می‌پوشاند ولی الزاما به تحویل درست نیست. هدف تنها تحویل کار است. برنامه‌نویس این توی وقتی به صورت تکی کار می‌کنند این گونه‌اند زیرا وابستگی به خودشان زیاد است و باید روی همه حوزه‌ها تسلط داشته باشند. این سطح اولین اَعلا برای شروع کار جدی است. در این سطح همه کارها را انجام می‌دهیم ولی فقط رفع تکلیف می‌کنیم مثلاً جلسه همیشه برگزار می‌شود ولی به اهداف جلسه نمی‌رسیم. فقط به این توجه دارد که کار انجام شود و الزامی به سازماندهی ندارد.

Level2 managed

در این سطح یک تیم برای کار داریم که وظیفه هر کس در آن مشخص است. از یک سری ساختار استاندارد باید تبعیت شود و علاوه بر آن همه کارها انجام می‌شود. در بعضی از پروژه‌ها ساختار را ترسیم و تعریف می‌شود که در آن پروژه مسئولیت افراد را مشخص می‌کند و در واقع نقش هر شخص بر اساس پروژه تعیین می‌شود.

Level 3 defined

در این مرحله تعاملات هدف در اهمیت ندارد بلکه می‌خواهیم استانداردهای tool هایی که بر اساس آن شخص کارش را انجام می‌دهد مشخص کنیم.

Level 4 quantified

برای مدیریت افراد quantified manage استفاده می‌کنیم یعنی با اعداد و ارقام کار افراد را مشخص. مثلاً چند درصد از کار در زمان انجام شده که در این حالت سطح مدیریتی بالایی دارد و مشخص می‌شود آیا به اهداف روزانه خود رسیده‌ایم یا نه. ولی هنوز درباره مشخص نیست. مدیریت یعنی تعیین و مشخص خطر

Level 5 optimized

در این سطح چون اندازه‌گیری‌ها دقیق شده به بهتر کردن برنامه می‌پردازیم مثلاً با افزایش سطح علمی و گذاشتن دوره‌های آموزشی، ایجاد امکانات لازم به بهبود وضعیت بپردازیم.

بهترین شرکت‌های ایرانی در سطح 2 قرار دارند. البته در تمام سطوح estimation وجود دارد و با پیش‌بینی‌ها می‌توان گفت به سطوح بیشتر رفته‌ایم. این شرکت‌ها قصد دارند با گرفتن استانداردهای مختلف به سطوح بیشتر بروند.

Subject:
Year. Month. Date. ( )

نحوه‌ی انتخاب یک سیستم برای کار کردن بر اساس این سطح بندی انجام می‌شود زیرا مدیریت باید به سیستم اعتماد کند هرچه در سطوح بیشتری قرار گیرد این اعتماد بیشتر خواهد بود. در واقع برای رسیدن به اهداف یک سری practice انجام می‌شود این level بندی براساس مهم است که آیا همه‌ی این practice ها انجام می‌شود یا خیر؟
در فرایند مهندسی، شکل مهندسی آن مهم است نه نتیجه‌ای که می‌گیریم.
شرکت هایی که در سطح incomplement قرار دارند با مقایسه خود با سایر شرکت ها هزینه ها را پیش بینی می کنند در حین کار یعنی به نتیجه‌ی درست نخواهند رسید.
(عملکرد هر یک از سطوح در کتاب خوانده شود.)

با وجود الگوهای مطرح شده ولی software ها براساس توانایی افراد است و هر کس براساس توانایی و علاقه‌ی خود کاری را انجام می‌دهد بین الگوهای کاری را تعریف می‌کنیم:
1- team software process
2- personal software process
(تعریف این دو از کتاب خوانده شود.)

هدف از process تولید محصول است اینکه محصول چه باشد مدل‌های مختلف process ایجاد می‌شود:
1- waterfall : (Sequential Model)
مراحل مختلف یک process به صورت زیر است.
communication → planning → modeling → construction → deploment
یعنی ابتدا پروژه را تعریف می‌کنیم نیازها را مشخص می‌کنیم دلیل شکل‌گیری را می‌شناسیم سپس بر روی آن planning می‌کنیم (estimate، scheduling، tracking) بعد از آن نیازها را اصلاح کرده طراحی، آنالیز و سپس تولید می‌کنیم. یعنی همه‌ی این مراحل باید به صورت سلسله مراتبی و به ترتیب انجام شود.
اشکال این روش این است که 1- ممکن است شاخص‌هایی در مرحله‌ی communication آماده نباشد در نتیجه پروژه متوقف می‌شود. یعنی طبیعت بعضی از پروژه‌ها این است که همه چیز از ابتدا مشخص نیست.
2- چون برای مشتری مبهم است در انتقال به کارفرما مشکل ایجاد می‌شود و نمی‌تواند خوب بیان کند. پس از قرارداد در مرحله‌ی communication انجام می‌شود و بعد از آن کارفرما غیب می‌شود در نتیجه اعتماد مشتری نسبت به ... کم شده و در نتیجه‌ی نهایی تاثیر می‌گذارد. در نتیجه جدا شدن از مشتری خطاهایی را ایجاد می‌کند امروزه کمتر از این روش استفاده می‌کنند. هدف از آن این است که با یک بار طی کردن این مراحل به نتیجه برسیم

در این روش همه گام ها باید به ترتیب برداشته شود اگر یک بار طی کرد باید به نتیجه برسد . در صورتی برگشت اتفاق می افتد که خطایی وجود داشته باشد در نتیجه به ابتدا برمی گردیم (برگشت به واسطه خطا است)

Subject:
Year. Month. Date. ( )

در صورت رسیدن به بن بست باید از ابتدا شروع کنیم . (مفهوم waterfall)

۲- Rappid Application develop (RAD) : (Sequential Model)

در روش قبلی زمان زیادی صرف می شود ولی در این روش همان مسیر را با حضور و تعامل مشتری طی می کنند یعنی هر زمانی که خواستیم در اولین فرصت برای آن وقت بگذاریم . از این روش نمی توان برای هر پروژه استفاده کرد روند کار به صورت زیر است :

- bussiness modeling : صورت مسئله و جابجایی آن را در می آوریم

- Data modeling : داده ها را شناسایی می کنیم . سازمان باید یک بانک اطلاعاتی داشته باشد.

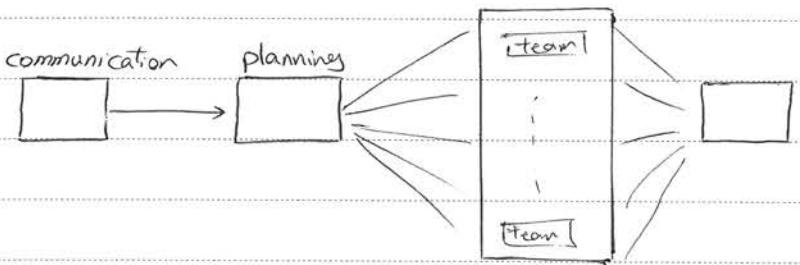
- process modeling : روند هایی که از آن استفاده می شود از tool های سطح بالا باشد (database / interface) باید برنامه نویسی قوی داشته باشیم . تمام platform ها باید در سطح dB باشد

- Application generator : تولید برنامه

- Test & turn over : تست و بازیابی برنامه

در واقع پروژه بنا به داده هایی دارد با این روش قابل انجام باشد معمولاً به صورت ماشینی بالا Functional است کار کردن با این گونه مدل ها نیاز به متخصص دارد چون مقدار چنین افرادی کم است پس پروژه ها کوچک می شوند ابزارهایی که وجود دارد تعیین کننده محدودیت پروژه است . در این روش ریسک تکنیکی بالا است و ممکن است به موقع تمام نشود.

روش ۱ و ۲ به صورت sequential انجام می شود.



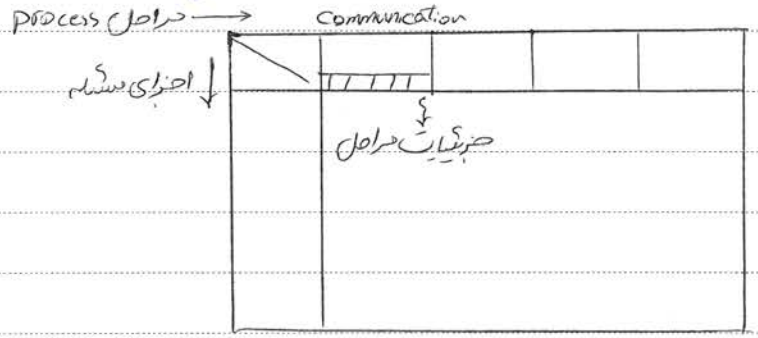
تیم های مختلف
این مدل دارای بن بست ... هر تیم نمی تواند با هم انجام شود

PAPCO

Subject:
Year. Month. Date. ( )

۳- Incremental Model (روش تدافعی):

در این روش محصول را خرد کرده و در هر قسمت بخشی از محصول تولید می‌شود و در پایان از محصول در هر قسمت به صورت sequential انجام می‌شود. در نتیجه اگر یک قسمت در فرجه‌ای گیر کند بسایر قسمت‌ها را انجام می‌دهیم. این روش برای جایی که نیاز کم است نیز مفید است. می‌توان منابع را به قسمت‌های کوچک‌تر اختصاص داد. pipeline ۱. در این روش نرم‌افزارها باید slot-base باشند یعنی قسمت‌ها مجزا از یکدیگر باشند و توان قسمت‌ها را بروی یکدیگر پیاده کرد. ارتباط بین این قسمت‌ها درمیانی است. در نتیجه می‌توان هزینه‌ها را براساس اجزا مشخص کرد. و این افتان وجود می‌آید که با حذف یک قسمت هزینه و هزینه کم‌تر شود. همه پروژه‌ها قابل تقسیم هستند. بر این اساس داحاترین به صورت زیر است:



این داحاترین برای مدل incremental مناسب است و ایجاد آن نیازمند تجربه زیاد است.
از خصوصیات خوب این روش این است که: ۱- برای نرم‌افزارهایی که App های درونی هستند و بروی انسانی و منابع زیادی دارند می‌توان به صورت pipeline کارها را انجام داد.
۲- براساس نیازمندی می‌توان قسمت‌هایی را آن را دوباره آماده کرد.

۴- Spinal (روش تدافعی):

در این روش حل مسئله از طریق خرد کردن آن صورت نمی‌گیرد بلکه پروژه را در چندین فرجه به صورت کامل انجام می‌دهیم. یعنی در هر فرجه با شروع از سطوح بالا در هر فرجه communication نیاز افراد در آن سطح را بررسی کرده و فرجه بعدی را متناسب با آن طی می‌کنیم و سپس به سطوح پایین‌تر و جزئی‌تر می‌رسیم تا نرم‌افزار نیاز همه افرادی که با آن سروکار دارند را برطرف کند. در پایان هر فرجه نیز برنامه متناسب با نیاز آن سطح کامل می‌شود. از آن جایی که هر فرجه به صورت هرم است با حرکت از سطوح بالا به پایین تقاطعی بوجود نمی‌آید. هر چه بالاتر باشد تصمیم‌گیری است و در حالت‌های پایینی اجرایی است.

PqPCO

12

تفاوت میان prototyping ، spiral ، incremental را مشخص کنید

Subject:
Year. Month. Date. ( )

یک بار پروژه را در بالاترین سطح به صورت کامل تعریف می‌کنیم. هر سطحی در چهارچوب سطح بالاتر کار می‌کند در سطح پایین ترقسمت به ماشکلات در مورد یک سری متدها و موارد پیاده سازی می‌شود. این برنامه Maintainable است. در هر مقطعی کار کامل است و فقط فرم متفاوت است. این چرخه‌ها هر یک در مرحله process را شامل می‌شود.

مثال: هیئت دولت به وزیر آموزش عالی برای ارتقا سطح علمی دانشگاه‌ها دستور می‌دهد که سیستم‌های خود را کامپیوتری کنند. ممکن است افرادی که قانون را تصویب کرده اند از جزئیات شبکه اطلاعی نداشتند ولی این نیاز را در چهارچوب کلان مملکت دیده اند. و توجهی به راحتی دانشجویان یا کارمندان ندارند. پس از طرح مسئله متناسب با نیاز دولت پاسخگویی شده به سطوح پایین ترفته در سطح دانشگاه‌ها نیاز به گزارشات دارند که امکانات لازم را فراهم می‌کند. سپس در سطح آموزشی و دانشجویان هم نیاز را شناسایی کرده و برنامه را متناسب با آن انتخاب می‌کنیم.

نکته: ترکیب دو روش spiral و incremental مورد علاقه تعداد بیشتری از افراد است.

ه- Rapid prototyping : (روش تکاملی)

همان چرخه وجود دارد و هر چرخه براساس دی نمونه‌های مختلف انجام می‌شود و نه در این روش یک نفر هست و ذره ذره حرف را می‌گیریم در پایه درست فهمیدن برنامه را می‌نویسیم.

در این مواقع مشتری چیزی می‌خواهد ولی نمی‌داند چیست. در این مواقع که موضوع کامل مشخص نیست آنقدر چرخه می‌چرخد که کار انجام شود.

از جمله مشکلات آن عبارتند:

۱- معلوم نیست چرخه کی تمام می‌شود. توانایی فردی مدیر پروژه برای اتمام خیلی مهم است.

۲- کسی که در این زمینه تحقیق می‌کند پیاده سازی هم می‌کند و چون بلکی می‌شود برای سیستم‌های کوچک مناسب است. این روش این حس را به مشتری می‌دهد که چرخش برنامه را دوست است.

۳- زمانی که توضیحات مشتری به پایان رسید بگوئی که پروژه تمام شده از نظر functionality نشان داده‌ام ولی از نظر non-functionality امکان پذیر نیست.

۴- رعایت برنامه روی platform های سطح پایین و همراه با ادلیف خواهد بود. یعنی از نظر برنامه‌ریزی کیفیت خوبی نخواهد داشت.

PAPCO

Subject:
Year. Month. Date. ( )

و این روش را به عنوان یک سیستم نمی پذیریم بلکه برای جمع آوری اطلاعات زمانی که خود مشتری هم نمی تواند استفاده می کنیم.

برای اطلاعاتی که از مشتری می گیریم هر چه را یکبار به صورت کامل انجام می دهیم. کسی می تواند این پروژه را انجام دهد که حداقل یک بار مهارت و تجربه اجرای آن را داشته باشد.

win

۶- Win win Spiral :

این روش یک process model نیست و همان spiral است با این تفاوت که باید stackholder را مشخص کنیم. در این روش هدف این است که هر دو طرف موفقیت داشته باشند یا به هر چیزی که مشتری می خواهد فراهم شود یعنی در مرحله communication باید stackholder ها را شناسایی کرد و متناسب با خواسته آنها کاری را انجام داد که احساس رضایت کنند.
در واقع ساختار یکی است فقط توصیه ها متفاوت است.

نکته: می توانیم مسئله را خرد کنیم و هر جزء را به صورت spiral حل کنیم.

۷- Component-base Model :

در روش های قبل تمام توجه به نیازهاست ولی در این روش تمام توجه برای حل مسئله، موجودی هاست که در پایان saas (software as a service) به وجود می آید. در واقع به این نکته توجه می کنیم که چه چیزهایی را در محیط موجود می توانیم سرهم کنیم تا به نیازهایی که می خواهیم برسیم. این ایده در هر یک از روش های قبلی می تواند باشد و به صورت process در نظر نمی گیریم.

Aspect :

کلاسی است که در آن obj ها براساس موجودیت واقعی ایجاد نمی شوند بلکه به واسطه رسیدن به کارایی و سرعت اجرایی مناسب کنار یکدیگر قرار می گیرند. در روش object oriented در هر زمان فقط یکی از obj ها کار می کند و کارایی پایینی دارد ولی در این روش obj ها را طوری انتخاب می کنیم که کارایی و سرعت سیستم بالاترین حالت باشد.
در object-base هر obj یک core کار می کند ولی در Aspect طوری کلاس ها را انتخاب می کنیم

P4PCO

Subject:
Year. Month. Date. ( )

کدوم یک از سیستم Multicore ، core ها میتوانند با یکدیگر کار کنند. Aspect هایی دارند (صفات یکی بر روی یک پدیده که میتواند هر چیزی را در برگیرد.)

۱- استفاده از ریاضیات در کار (Formal method):

برای کار با Aspect ها نیاز به ریاضیات است روش های Formal الگوی خاصی ندارند و می توانند از spiral و ... استفاده کرد. فقط مشخص می کنیم که مدل باید به صورت ریاضی باشد تا بتوانیم با حساب و سنج تحلیل خوبی داشته باشیم.
قدم های انجام این کار عبارتند از:

1) Reduction:

در وضعیت واقعی $۲^۲$ ، ۲+۲ ، ۲+۲ می توانیم با ۳ روش مختلف ۴ برسیم. هر سه عبارت مختلف نتیجه یکسانی دارد. در واقع به خاطر overlap عبارات یک مفهوم را می توان به صورت مختلف نشان داد. برای اینکه گذر این دولت های مختلف نشویم یک فرمت عددی خاص می توان هم عبارات را فقط با یک عمل جمع انجام داد.

برای مشخص کردن برای دو فرمول در روی های مختلفی به هر دو می دهیم (مانند Data time stream) که برای هر D_1 ، t_1 متعلق به ماشین اول یک D_2 ، t_2 وجود داشته باشد که متعلق به ماشین دوم باشد در این حالت می توان گفت Functionality یکدیگر را دارا است.

$$\{\forall (D_1,t_1)\in M_1 \ , \exists (D_2,t_2)\in M_2 \Longrightarrow M_1 \subseteq M_2\}$$

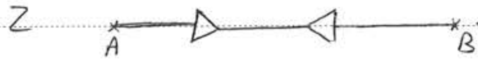
در این روش نیاز به روش های ریاضی نشان می دهیم و سعی می کنیم عبارات overlap نداشته باشند تا به یک آن بتوانیم حکم نهایی دهیم. برای اطمینان از اینکه خطا وجود نداشته باشد از همان الگوهای قبلی ولی با مدل ریاضی استفاده می کنیم تا بتوانیم آن را کنترل کنیم.
برنامه هم نوعی از دستورات است که نمی توان آنها را فرموله کرد زیرا حتی ترتیب دستورها نیز بر اساس قانون نیست و برنامه ای در ذهن داشت. اگر بتوانیم یک رابطه ریاضی معادل برنامه بدست آوریم به راحتی می توان با مدل چند جمله ای و عبارات بروی آن به اطلاعات مورد نظر رسید.
با فرموله کردن می توانیم رفتار را نیز با یکدیگر آنالیز کنیم.

Subject:
Year. Month. Date. ( )

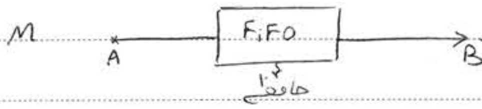
به عنوان مثال در روش channel base حالت‌های مختلف زیر را تعریف می‌کنیم که تمام پردازنده‌های کامپیوتری از طریق این channel ها قابل مدل‌سازی هستند. (در واقع دستورهای حافظه‌ای و غیرحافظه‌ای می‌توانند از نوع channel باشند) دستور یک پردازنده به صورت وارد می‌شود و می‌توان آنها را مدل کرد.

اگر در A، fire شود در B نیز دیده می‌شود.

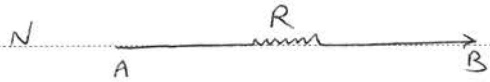
داده‌ای در A، fire می‌شود در B می‌تواند دیده شود یا گم شود.



باید هر دو با یکدیگر داده بدهند تا منتقل شود و در غیر اینصورت داده گم می‌شود.

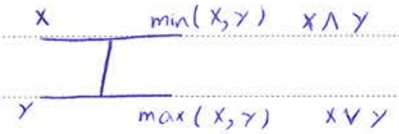


اگر حافظه خالی باشد A، fire می‌شود و اگر B، fire شود در صورت وجود داده در حافظه منتقل می‌شود.

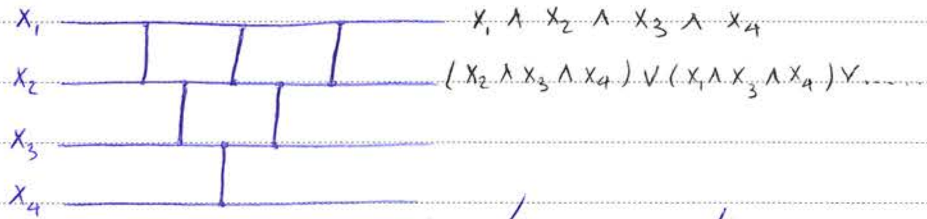


اگر value برابر R باشد منتقل می‌شود.

یکی دیگر از گیت‌های مدل‌سازی به صورت زیر است:



با توجه به این مدل می‌توان الگوریتم bobble sort را به صورت زیر نشان داد:



این روش مشکلاتی دارد از جمله اینکه روش سخت و غیر قابل فهم است. افراد کامپیوتری که با این روش آشنا باشند کم است و آموزش آنها هزینه زیادی دارد.

PAPCO

Subject:
Year. Month. Date. ( )

رفتاری

ممکن است نرم افزار بعد از استفاده کاربر در شرایط ویژه و استثنایی به درستی کار بکند (validation) درستی همیشگی نرم افزار را verification می گویند. با ریاضی کردن مدل و تبدیل آن به formal به راحتی Aspect ها قابل تشخیص هستند در نتیجه در برنامه توجهی به design و construction ندارد و از روش ها نیز نیستند و به requirment توجه می کنند که چگونه آن را توصیف کنند.

هدف روش ها می توانند یک برنامه را از حالت شهودی و تجربی خارج کنند و برنامه به صورت یک فرمول ریاضی تبدیل شود.

خود formal یک process نیست و می توان از روش های spiral و ... برای پیاده سازی استفاده کرد.

روش حالت موازی concurrency:

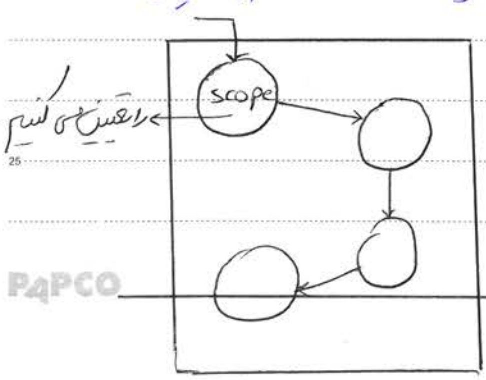
یک حالت کنترل فرایند مدیریتی است برای تیم هایی که همزمان چندین پروژه را انجام می دهند. در هنگام انجام چندین پروژه مدیریت باید آن را کنترل کند زیرا خطرها باید شناسایی شود و از آنها دوری کنیم

یکی از وظایف مدیریت پیش بینی خطرهاست:

monitor
↑
RMMM → management
↓ ↓
risk metigation

یعنی ریسک را در نظر گرفته و از آن اجتناب می کند، monitor می کند و اگر اتفاق افتاد مدیریت می کند. بعد از monitoring یکی از کارهای مدیریت برطرف کردن خطرها با تخصیص منابع است (افزایش حقوق فردی که می خواهد از شرکت برود.)

هر پروژه در یک activity قرار می گیرد. این activity ها می توانند حالات مختلفی داشته باشند که یک Automata درست می کنند که state ها را نشان می دهد. هر کاری انجام می دهیم وضعیت تعیین می کند که به صورت دیاگرام نشان داده می شود یعنی هر پروژه در هر لحظه در چه وضعیتی است و با هر رفتار به چه وضعیتی می رود.



PAPCO

Subject:
Year. Month. Date. ( )

این روش یک process model نیست و برای کنترل و monitoring پروژه های مختلف به کار می رود. یعنی مدیریت منابع پویا.

در activity ها به طور مثال در مرحله communication یک pattern برای دقتی که با مشتری ارتباط برقرار می کنیم وجود دارد تا نیازهایش را بفهمیم. برای جمع آوری اطلاعات نمی توانیم به روشات از آن شرکت بازدید کنیم و سروی چروه صرف کنیم. در هر فاز چیزی برای دیدن وجود دارد و تعامل می شود برای برطرف کردن این مشکل از ابزارهایی استفاده می کنیم:
بخشی از modeling مربوط به Analyse است که یک جنبه سنجش و ابزار مورد نیاز را می دهد که در صورتی که برنامه کارهای مشخص شده انجام شود کافی است.

دو نوع Analyse وجود دارد: 1- Structure Analyse 2- object orient Analyse
و هر یک ابزارهای خاص خود را دارد.
در مدل structure تابعی در ۲ مرحله و تابعی در ۱ مرحله انجام می شود. روش تحلیل object oriented همان تعاریف قبلی است که یک پارچه و یک سان شده و از تکنولوژی های خاص خود استفاده کرده اند و لغات آن را تکرار دارند نه اینکه با روشهایی جدید راست باشد.
در واقع ذات فانی تعریف شونده ای است که می توان هر کسی برای آن اسمی تعیین کرد.

یک نرم افزار وجود دارد برای این کار RUP است که اصل و اساس آن برای آنالیز است و مربوط به modeling است. از این نرم افزار به نوع مختلف process، روش تحلیل و مهندسی نرم افزار یاد کرده اند که ارتباط ما را هی برای حل مشکل است. در واقع تعیین می کند که از سیستم به چه هایی مورد نظر اطلاعات جمع آوری شرکت است.

در یک process، Activity های مختلف، مهارت های مختلفی نیاز دارد. پس از افراد خاص خود استفاده می کنیم در یک تیم افراد در سطوح و دانش های مختلفی هستند پس باید تخصص غالب شود یکی از روش ها برای انکار مستند سازی است.
در این صورت اگر فرد دیگری بخواهد کار او را ادامه دهد با استفاده از مستندات مشکلی نخواهد داشت. از طرفی بسیاری از افرادی که انجام می دهند تخصص ندارند در نتیجه وقتی یک طراح می خواهد ارتباط برقرار کند

P4PCO

؟؟ در Agile خودمان یک سری اصول داشتیم و دو تا تفکر می‌شود یک [illegible] که به این روش انجام می‌شود یعنی چی؟؟

Subject:
Year. Month. Date. ( )

... بارد باید یک کپی از همه اطلاعات مورد نیاز که او به آن آگاهی ندارد در اختیارش قرار دهد. حتی ممکن است خود designer به علت کار زیاد از جزئیات اطلاع نداشته باشد. این مستندات به حد افراطی رسیده بود تا اینکه مهندسان دانستند وارد می‌شوند و دیگر نیازی به مستندسازی نبود. همه اطلاعات را خودشان می‌دانستند ولی مدیران نمی‌خواستند این موضوع را قبول کنند.

با پیشرفت کار، designer ها و constructor ها با یکدیگر ادغام شدند و ابزارها در سطح بالاتری قرار گرفتند. در نتیجه لازم نیست چیزی را منتقل کنیم. document ها کمتر و سرعت بیشتر و هزینه پایین‌تر می‌شود. سرعت کار بیشتر در نتیجه زمان بیشتری خواهیم داشت.

توجه به اینکه انجام بعضی از کارها ضایع است و آن‌ها را دور بریزیم باعث بوجود آمدن پدیده‌ای به نام Agile شد. شخصی معتقدند خود یک process model است درحالی که هیچ activity ندارد و توصیه است. در این روش کارها را تقسیم می‌کنیم و document تولید می‌کنیم. عده‌ای معتقدند Agile بودن یک صفت است و یک مدل می‌تواند از روش‌های spiral و ... برای آن استفاده کند. عده‌ای می‌پندارند در هم کار کردن و documentless بودن یعنی Agile. در واقع یک سبد به نام schoom ایجاد کرده و همه ابزارها و ویژگی‌های مورد نیازش را در آن چیدمان می‌کنیم و توصیه‌ها را انتخاب می‌کنیم.

نکته: مستندات در هر مرحله دیده می‌شود و افزایش دانش عمومی می‌تواند جایگزین مستندسازی باشد.

با این وجود مدیران هنوز به دنبال جمع آوری هر چه بیشتر اطلاعات هستند.

از ویژگی‌های Agile عبارتند از:

۱ـ tools های مورد نیاز برای طراحی و تولید کد موجود است.

۲ـ Agile ها فقط بر روی مستندات لازم تکیه دارند.

۳ـ همکاری تنگاتنگ با مشتری

۴ـ پاسخگویی به تغییراتی که می‌توانند بر روی سیستم ایجاد شود. مثلاً نیازها عوض شود و اهداف نیز تغییر پیدا

Subject:
Year. Month. Date. ( )

تغییرات در سطح تکنولوژی باشد. باید بتوان این تغییرات را در سیستم اعمال کرد (یا تغییرات در ساختار جا بجایی نیرو).

Agile شامل یک سری اصول است که عبارتند از:

۱- برای وجود ارتباط با مشتری و همواره در تماس بودن با او در بسیاری از موارد معمول همان الگوی تکاملی یا prototype می‌شود.

۲- تغییرات در هر زمانی بوجود آید باید بتوانیم handle کنیم. در واقع هم این تغییرات که جزء اصول مهندسی نرم افزار است برای رضایت مشتری می‌باشد.

۳- از مشتری غافل نشویم و با او در تماس باشیم. مقصد همیشه غایب است و با ارتباط مستقیم با مشتری ملت می‌شود که کار درستی انجام شود.

برای انتقال و جمع آوری اطلاعات مؤثرترین راه ارتباط Face to Face است. در شرکت های بزرگ سعی بر این است که روابط Face to Face کنترل شود زیرا مهم ترین بخش است.

۴- توجه مستمر به تکنیک های خوب از اصول مهم در Agile است.

۵- سادگی: که جزو فاکتورهای مهم کنترل نیست در واقع سرخود و نگهداری (maintanable) ساده باشد و قابل درک باشد. بنابراین الگوریتم ساده‌ای به کار برده شود.

۶- عملی بودن کارها و سازمان ها، معتقد بودند وقتی سازماندهی درست باشد هرکسی در جای خود کار خود را انجام دهد. بعدها توجه شد که هر چه سیستم خوب باشد good stuff یعنی افراد خوب و قابل اعتماد اهمیت بیشتری دارد چون یکی از کارها در Agile این است که هر چند وقت یک بار خود را ارزیابی کنند.

Agile توصیف برای process است که با قرارداد یک توصیف و به کارگیری یک سری process model Agile process بوجود آورده است.

Subject:
Year. Month. Date. ( )

چند مشکل مهم در Agile عبارت اند از: (مشکلات موجود)

۱- خیلی سخت است بگوییم تغییر نداشته باشیم. مشتری مشخص در بسیاری از موارد وجود ندارد. در نتیجه با در نظر گرفتن یک جامعهٔ آماری نرم افزار تولید می شود ولی در Agile مشتری مشخص است و باید بتوانیم با نیاز او نرم افزار تغییر کند.

۲- کارهای انجام شده به صورت مرتب به مرتب یا interleaving هستند یعنی باید بلافاصله انجام شوند. ممکن است مراحلی از design در جلوتر کنار رود و آنقدر پارامترهای جانبی کم اهمیت شود که prototype به عنوان محصول کار قرار گیرد.

۳- در برنامه ریزی مشخص می کنیم که در مرحله پیاده سازی هر کار در زمان مشخص انجام شود ولی معمولاً مطابق برنامه پیش نمی رود و زمان بیشتری نیاز دارد.

در نتیجه interleaving و هیچ یک از مشکلات بالا قابل پیش بینی نیست. Agile از روشی جلو می رود که خود را با تغییرات جلو می برد. مدل های process که با روش Agile مطابقت داشته باشد از جمله روش های تکاملی است.

در پروژه Agile به دو ویژگی توجه داریم:

۱- از هر نوع ایده ای استقبال می کند. مشکلات یک محل را افرادی که برای مدت طولانی در آن جا کار می کنند درک نمی کنند برای همین به طور مرتب مدیران عوض می شوند تا نقطه نظرات جدید مورد نظر قرار گیرد. البته باید تجربه قبلی فدا نشود. (ایده در همه جا هست)

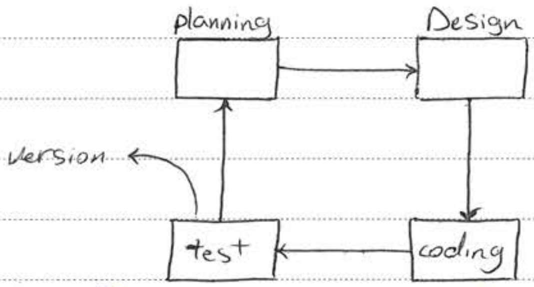
۲- عوامل process بودن Agile : فاکتور انسانی از جمله این دلیل هایی باشد که شامل توصیه هایی است

1) افراد باید صلاحیت داشته باشند.
2) داشتن نقطه نظر مشترک: در یک پروژه دیدگاه و انگیزه در پروژه یک فرم باشد.
3) همکاری: حس همکاری در سطح بالا وجود داشته باشد.
4) قابلیت تصمیم گیری: مدیر باید شجاع باشد و بتواند تصمیم بگیرد
5) راه حل ها یک راه حل قطعی برای هر زمان نیست (probability)
6) اعتماد دوطرفه: مطمئن باشند نظرات تابع هیچ تفکر منفی نیست.
7) خودسازماندهی

Subject:
Year. Month. Date. ( )

نمونه‌هایی از process model ها برای Agile عبارت اند از:

I) extreme programming :(XP)



در هر مرحله بین دو هفته محصول را تولید می‌کنیم چیزی که تولید می‌شود می‌تواند از طریق روش spiral با افزایش تست در هر مرحله یا بوسیله روش incremental در هر مرحله تکه‌ای از محصول تولید می‌شود.

کارهایی که در هر مرحله انجام می‌شود عبارت است از:

1- planning:

یک سری user story ایجاد کرده و نتایج صحبت با مشتری در آن نوشته می‌شود. user story ها را تفکیک کرده و از نقطه نظر مشتری اولویت بندی می‌کنند. ریسک برای هر story مشخص شده و با توجه به ریسک و اولویت بندی sort می‌کنیم که در هر مرحله چیزی تولید شود.

2- design:

یک راه حل قطعی و مشخص برای story ارائه می‌دهد. به این روش مشخص برای design، spark solution می‌گویند. در واقع با توجه به سوالات و ... می‌دانیم به چه چیزهایی پاسخ می‌دهد. یک سری test case ایجاد کرده و متناسب با آن coding را انجام می‌دهیم.

در طراحی‌های دیگر به دنبال این هستیم که طوری طراحی کنیم که به فکر آینده باشیم و چندین مدل در نظر گرفته که در صورت انجام شدن یکی از آنها استفاده کنیم ولی در Agile طراح از حال نمی‌رود و این طور می‌پندارد که آینده یک پروژه جدید است. لازم نیست چیز جدیدی در نظر گرفت مثلاً یک نگرش به آینده جزو story ها باشد.

PAPCO

Subject:
Year. Month. Date. ( )

3- coding:

برای پیاده‌سازی از روش pair program استفاده می‌کنند که در آن دو نفر برای برنامه‌نویسی وجود دارد یک نفر توجه به syntax دارد و دیگری توجه به مسئله‌ای دارد که حل می‌شود. یک نفر وسطی دیگر نظارت می‌کند در نتیجه error کمتر می‌شود.

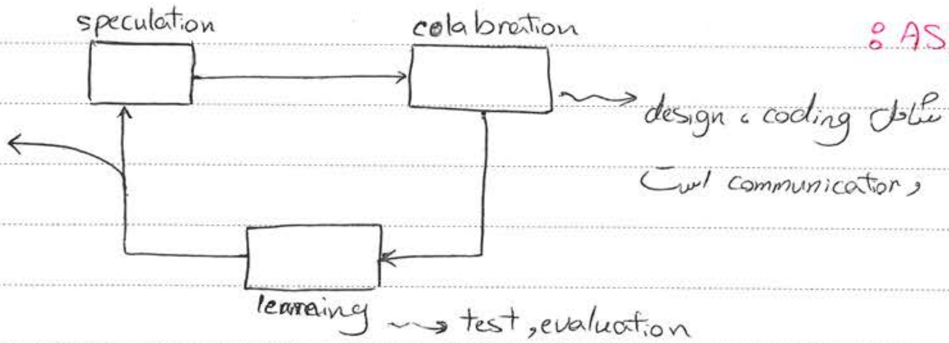
4- test:

coding را تست می‌کنیم و مرحله ارزیابی با حضور مشتری در سیستم محصول خارج شده و دارد چرخه بعدی می‌رود تا story ها تمام شود. این روش برای پروژه‌های کوچک مناسب است که لایه‌بندی ندارد. تیم کوچک است و محصولات آینده‌ای وجود ندارد.

در این روش تضمینی برای expantion و adaption نیست و توصیه این است که هدف ایجاد محصولی برای استفاده طولانی نمی‌باشد. اگر تکنولوژی، نیاز و ... تغییر می‌کند که درصد بالایی را خود را انعکاس می‌دهد تغییر ۵۰٪ به معنی بازنویسی است پس نیازی به صرف هزینه بیشتر برای کارایی بالا نداریم.

در واقع ایده‌های سایر process model ها استفاده شده است فقط در هر مرحله توصیه‌هایی را اضافه می‌کنیم.

ASD (III):



1- speculation:

وظایف، نیازها، زمان‌بندی و کارهای ابتدایی مشخص می‌شود در واقع ترکیب دو مرحله communication و planning در دیگر process model است.

PAPCO

Subject:
Year. Month. Date. ( )

scrum

2- celaberation :

جمع آوری اطلاعات، نیازها، مشخصات سیستم

توصیه ها در این مرحله عبارت اند از:

۱- تیم باید شکل گرفته باشد و هم بستگی داشته باشد یکپارچه و انعطاف پذیر بوده و بتواند با شرایط تغییر کند.

۲- اعتماد باید در تیم وجود داشته باشد.

۳- کمک افراد به یکدیگر بدون اینکه مستقیم داشته باشند به کمک داشته باشند

۴- سخت کوش بوده

۵- همه مهارت ها در دسترس باشد خوب درک کنیم و نظر بدهیم

3- learning :

تیم درک بهتر از نیازها و مشتری است در واقع تیم و مشتری با یکدیگرند. داشتن جلسات دوره ای برای مرور کارها - ارزیابی و evaluation مرحله در هر بار که اجرا گردد یک محصول تولید می شود

در اصل بین این روش و روش قبلی هیچ تفاوتی جز تغییر اسم وجود ندارد

(III) scrum :

برای تیم های خیلی کوچک مناسب است در واقع از process model های موجود استفاده کرده و چون عامل انسانی مهم است توصیه اضافه شده است.

در scram و XP یک سری ابزار خاص ارائه شده است.

PAPCO

Subject:
Year. Month. Date. ( )

فصل پنجم:

فرهنگ و اصول در زمینه مهندسی نرم افزار:

برای حل هر مسئله خارج از مهندسی نرم افزار ۴ گام اصلی وجود دارد.

۱- درک مسئله: (communication, analysis)
چه کسی مسئله را می‌خواهد؟ چه چیزهایی ناشناخته است؟ آیا می‌توان مسئله را خرد کرد؟ آیا می‌شود مسئله را به صورت گرافیکی نمایش داد؟ چه داده‌هایی برای حل مسئله مورد نیاز است؟

۲- برنامه‌ریزی برای حل: (software design, modeling)
آیا قبلاً چنین مسئله‌ای حل شده است؟ آیا زیر مسائل آن تعریف شده و مشخص هستند؟ روش حل را چگونه نمایش دهیم که کارهای عمومی وجود داشته باشد؟

۳- پیاده‌سازی: (code generator)
آیا پیاده‌سازی با کاری که انجام می‌دهیم مطابقت دارد؟ آیا همه‌ی اجزای حل درست است؟ آیا دارای الگوریتم و کد درست؟

۴- آزمون و بررسی: (quality assurance, testing)
آیا می‌توان همه‌ی اجزای راه حل را ارزیابی کرد؟ آیا محصول دارای نتیجه مورد نیاز می‌باشد؟ آیا همه نیازهای مشتری را برطرف می‌کند؟
ادبیات کد دهنده کامپیوتری شامل اصولی برای نرم افزار هست که این اصول تا activity ها در جزئیات پیش می‌رود.

اصول نرم افزار:

اصل اول- همیشه حق با مشتری است. هر چیزی که ارزش افزوده برای مشتری دارد مشخص می‌شود.
(همه اصول دیگر از این اصل حمایت می‌کنند)

اصل دوم- KISS: خیلی ساده برخورد می‌کنیم تا افراد دیگر بتوانند درک کنند. در واقع مبتنی بر سادگی است پس باید ساده باشد. این سادگی به معنای سریع و کثیف بودن نیست، بلکه بدان معناست زمان زیادی باید صرف شود تا ساده شود. نتیجه سادگی maintanable و less error-prone
(keep it simple, stupid)

اصل سوم- حفظ دیدگاه: اهداف باید مد نظر قرار بگیرد. باید در طول انجام هدف مدام مورد نظر قرار می‌گیرد و متناسب با آن قضاوت می‌کند. بدون یکپارچگی مفهومی سیستم به یک طراحی ناسازگار و وصله پینه تبدیل شده و در یک سطح خوردگی غلط یکدیگر متصل می‌شوند.

اصل چهارم- فعالیتی که انجام می‌دهیم برای این است که دیگران از آن استفاده کنند نه خودمان. در سیستم باید interface آن انعطاف‌پذیر باشد. با توجه به استفاده کنندگان باید specify کرد. طراحی با توجه به پیاده‌سازی کنندگان، کد نویسی با توجه به کسانی که در آینده سیستم را گسترش دهند. ممکن است کسی سیستم شما را debug کند.
این کارها اضافه کردن ارزش‌ها و مشخص‌ها را راحت‌تر می‌سازد.

if you do think about sth and still do it wrong, it becomes valuable experience.

Subject:
Year. Month. Date. ( )

اصل پنجم - همیشه به فکر آینده باشیم. از آن جایی که آینده دیده نمی‌شود هر اتفاقی می‌تواند بیفتد و قابل پیش‌بینی نیست. چون نمی‌دانیم چه اتفاقی می‌افتد باید طوری برخورد کنیم که تغییرات امکان‌پذیر باشد.
این کار هزینه‌ها را کاهش می‌دهد و ارزش قطعات های reusable در سیستم بالا می‌رود.

اصل ششم - کاری را انجام می‌دهیم که دیگران می‌خواهند دریافت کنند reusability دارد.
با reuse زمان و کار را می‌توان بدست آورد. reuse of design در قالب ایجاد الگوی نرم‌افزاری بوجود می‌آید.

اصل هفتم - همیشه با فکر همراه باشد (think) جمع‌آوری اطلاعات و درآوردن عوامل موثر در تصمیم‌گیری. با نادیده گرفتن تصمیم مواردی از think هستند. در واقع باید تصمیم را وقتی بگیریم که بیشترین اطلاعات را داشته باشیم. با فکر کردن می‌توان تصمیم درست تری گرفت و از اشتباهات جانبی آن یادگیری این است که جلوی دوباره چیزی را نمی‌گیریم در مورد آن تحقیق کرده و به جواب خوب برسیم.

هر activity (فعالیت) نیز برای خود اصولی دارد:

ص ۱۰۳ ۱- communication (requirement elicitation):

۱) listen گوش کردن در درآوردن اطلاعات: برای این کار باید بیشتر گوش دهید و کمتر حرف بزنید تا مشتری فرصت حرف زدن داشته باشد.

۲) prepare befor com قبل از ارتباط برقرار کردن خود را آماده کنیم. (زمانی را صرف فهمیدن مسئله کنید قبل از ملاقات)

۳) facilitate یک نفر مسئول باشد که به سمت هدف ببرد و اجازه صحبت به دیگران بدهد.

۴) face to face بهترین نوع ارتباط است face to face در برقراری ارتباط. take note نوت برداری کنید. حداقل این است که مخاطب فکر می‌کند به حرف های او گوش می‌دهید و چیزی فراموش نمی‌شود.

۵ - strive for collab همکاری داشته باشید عناد و جود نداشته باشد.

۶) draw picture به صورت تصویری به او بفهمانید.

۷) move on ادامه دهید. وقتی چیزی را قبول کردید یا رد شد از آن رد شوید و بیهوده تکرار نکنید.

۸) stay focus اجازه ندهید مسئله حل نشده و پس به سراغ موضوع جدید بروید.

ص 106 ۲- planning:

۱- مشخص کردن scope: محصولی که می‌خواهیم تولید کنیم آماده است (on touch) اینکه چه عواملی در کار بوده چه سرویس هایی باید داده شود در چه حوزه‌ای عملیاتی می‌کند ورودی و خروجی آن چیست همه این موارد بودن توجه به اینکه چه کاری انجام می‌دهد. هدف نرم‌افزار تعیین می‌شود.

non-scoping وجود ندارد یعنی هیچگاه نمی‌گوییم چه چیزهایی وجود ندارد بلکه می‌گوییم چه چیزهایی هست.

PAPCO

انواع تئوری در مورد planning: ① minimal.st: تعیین نیاز ← برنامه‌ریزی جزئی را رد می‌کند.
② traditional.st: برنامه‌ریزی یک road map دقیق ایجاد می‌کند و تعیین جزئیات باعث می‌شود چیزی از دست برود
③ agilist: یک planning game سریع نیاز است ولی road map به عنوان کار اصلی برای ... در نظر گرفته می‌شود

۲- درگیر کردن مشتری در planning: تا قبل از این فکر می‌کردند اولویت‌بندی‌ها قابل تعریف است. باید هر جزئی می‌بینیم آن اولویت آن را بدانیم تا مشتری درگیر شود و درک کند و بتواند کمک کند. برنامه‌ریزی یک موجود زنده است که می‌تواند تغییر کند. اولویت‌ها را مشتری تعیین می‌کند.

۶- همیشه تغییرات وجود دارد و باید بتوان پاسخگو باشیم در روش‌های increment‌ای بعد از تحویل هر increment باید replanning شود

۳- تخمین‌ها بر اساس اطلاعات شما باید انجام شود و بنا بر دانسته‌های دیگران astimate کنیم ⑧ باید مشخص شود چه تغییراتی می‌تواند وجود داشته باشد و سیستم آن را استقبال می‌کند. ⑨ باید مشخص شود چه چیزی از کیفیت را دنبالش هستیم ⑩ این برنامه پیگیری شود و در صورت عقب افتادن آن علت کشف شود

۴- ریسک در برنامه‌ریزی را در نظر بگیرید ⑦ واقع‌بین باشید ← اگر ریسک وجود دارد، تغییرات هست؛ امکان دارد حتی بهترین جدیدترین نرم‌افزار اشتباه تولید شود

۵- میزان جزئیات باید بالا و مناسب باشد (دوره‌بندی) میزان جزئیات در هر دوره باید ...
fine granularity: جزئیات در حد increment‌هایی که در زمان کوتاه انجام می‌شود. coarse grar: در زمان بیشتر انجام می‌شود
coarse ← fine

3- modeling:
به دو بخش عمده ۱- Analysis (تحلیل) ۲- design (طراحی) تقسیم می‌شود

Analysis: آیا اطلاعاتی که جمع‌آوری کردیم کافی است یا نه؟ آیا دقیق و با جزئیات هستند یا نه؟ آیا ابهاماتی وجود دارد یا نه؟ چطور مشکل را بیان کنیم تا مهندسانی که پیاده‌سازی می‌کنند کارشان را به درستی بتوانند انجام دهند.

هر سیستم نرم‌افزاری را با سه بعد می‌توان شناسایی کرد:
I) information: داده‌هایی که برای پردازش‌ها ذخیره می‌کنیم نه متغیرهای موجود (آنهایی که برای اجرای برنامه خودمان اختیار می‌کنیم مد نظر نیست)

II) Function: (عملیات) عملی که بر روی داده‌ها انجام شده و منجر به تغییر آن می‌شود

III) behavior: (رفتار) تصمیم بگیرد که در وضعیت موجود چه کاری انجام دهد.

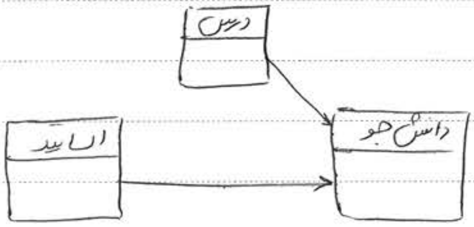
نکته: Analysis model برای نمایش نیازهای مشتری است و Design model ویژگی‌های مورد نیاز برای ساخت نرم‌افزار را مشخص می‌کند

نکته: Analysis modeling اولین گام در حل مسائل software engineer است

Subject:
Year. Month. Date. ( )

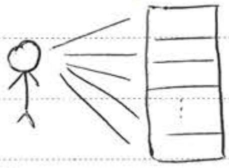
اصول موجود در این مرحله عبارتند از:

۱- در طراحی هر سیستم‌ها به این سه مورد توجه می‌شود. چیزی که اهمیت دارد این است که اطلاعات را به درستی درک کنیم و یک data model (آن هم تمام اطلاعات یک سیستم) برای آن تعریف کنیم. یعنی اطلاعات را چگونه ذخیره کنیم و چگونه پاسخ داده شود.

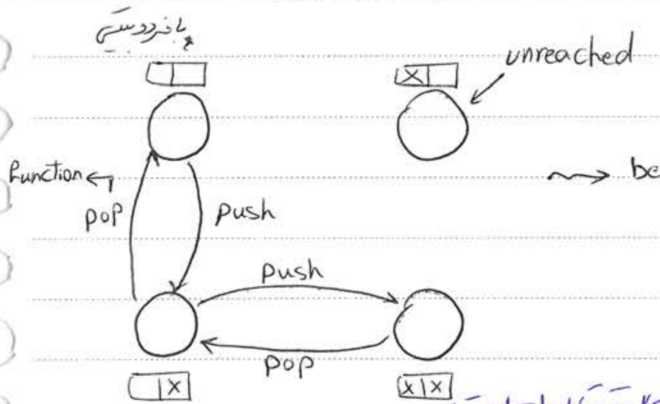


← چه ارتباطی بین پایگاه داده‌های مختلف وجود دارد. آنها را درک کرده و بدانیم چگونه از آن استفاده کنیم.

۲- بدانیم چگونه function ها را تعریف کنیم مانند user story در Agile



۳- رفتار سیستم را بتوانیم توضیح دهیم



← behavior model کل آن یک است

behavior model می‌گوید در هر زمانی چه حالتی قابل اجرا است.

۴- function ها باید لایه‌بندی شود. می‌توان function هایی که قرار است اجرا شوند را به لایه‌های مختلف برده و در سطوح مختلف پردازش کنیم و پاسخگویی کنیم. در پروژه‌های پیچیده اینروش تقسیم و حل استفاده کرده و در زیر بخش‌هایی ایجاد می‌کنیم که به این عمل partitioning می‌گویند.

۵- برای تحلیل از کجا شروع کنیم. نقطه شروع به محصول مورد نظر بستگی دارد و باید از information مورد نیاز آن شروع کنیم یعنی از نیاز شروع کرده، جلو می‌رویم تا به محصول برسیم.

باید از داده‌ها به پیاده‌سازی برسیم. جزئیات پیاده‌سازی که مربوط به قسمت مدل design است نشان می‌دهد

هر چه درست جلو برویم پیاده‌سازی می‌شود.

Subject: component-level detail (۳) Architecture (۲) user interface (۱): ویژگی هایی که در این بخش تعیین می‌شود
Year. Month. Date. ( )

data - driven: از طریق ساختار داده‌ای معماری تعیین می‌شود
pattern driven: از اطلاعات مرحله analysis استفاده می‌کنند و معماری تعیین
object-oriented: از زوج‌های موجود استفاده می‌کند
→ متدهای design

design: طراحی

مهندسین تحلیل را جدای از طراحی می‌دانند و برای طراحی از ابتدا شروع می‌کنند حتی در پیاده‌سازی هم همین طور است. ابتدا چیزی را طراحی می‌کنند ولی زمان پیاده‌سازی الگوریتم متفاوتی پیاده‌سازی می‌شود.

اصول مربوط به این مرحله عبارت است از:

۱- باید بتوانیم از تحلیل به طراحی برسیم و آن را ردیابی کنیم

۲- بی‌نهایت Application وجود دارد. بی‌نهایت مسئله می‌تواند بی‌نهایت راه‌حل داشته باشد. یک سری ویژگی‌های کلان مشترک بین مسائل وجود دارد که می‌توان بوسیله آن تعدادی از مسائل را یک طور حل کنیم در نتیجه یک معماری کلی ایجاد کرده که به ما کمک می‌کند مسئله را چگونه حل می‌کنیم

۳- بین برنامه و محیط خارجی کاربر باید تعریف شود interface نمی‌تواند یک ایده ذهنی باشد
(۹) user interface باید به گونه‌ای تعریف شود که نیاز مشتری را برطرف کند حتی اگر در داخل خود با نگرش interface ...

۴- component هایی که function ها را تعیین می‌کند باید مشخص شود. هر component فقط و فقط باید یک function کار داشته باشد

۵- واحدهایی که توصیف می‌کنیم باید مستقل باشند در نتیجه به extendebility و reuse bility می‌رسیم یعنی ارتباطات واحدها و component های مختلف باید سست و ضعیف باشد.
(۸) توابعی تعیین می‌شود که برای آینده نیاز به شناخت مشخص نباشد در ارتباط برقرار کردن component ها هم همین طور است و به جای call کردن پیام ارسال می‌شود

۶- طراحی ساده و قابل درک باشد مدل‌سازی آنالیز برای مشتری ولی مدل‌سازی طراحی برای constructor است. در سادگی انعطاف‌پذیری وجود دارد این طراحی در ارتباط با کسانی است که می‌خواهند کد تبدیل کنند و آنهایی که نرم‌افزار را test می‌کنند و آنهایی که برنامه را در آینده نگه می‌دارند.

۷- طراحی یک مرحله تکراری است و لزوماً یک بار انجام نمی‌شود و می‌توان هشت طراحی انجام داد.
در مرحله اول سعی بر این است که خطاها را از بین ببریم و در تکرارهای بعدی می‌خواهیم طراحی را ساده کنیم.

Analysis model: information domain of the problem ، user visible functions ، system behavior ، Analysis class
Design model: اطلاعات بالا را به معماری تبدیل می‌کند / subsystem ، component level design
implement mayer function

PAPCO

د) کدهای تولیدی شامل سه دسته است: ۱- ایجاد مستقیم کد ۲- تولید اتوماتیک کد از طریق design مثل ایجاد component
Subject:
Year. Month. Date. ( )
۳- تولید اتوماتیک کد که از یک زبان برنامه‌نویسی نسل چهارم استفاده می‌کنیم.

٤- constructing:

اصول به دو دسته: coding و test تقسیم می‌شود.

coding: سه گام در آن وجود دارد: ۱- قبل از کدگذاری
۲- زمان کدگذاری
۳- اعتبارسنجی کدگذاری

اصول مربوط به گام اول عبارت اند از:
۱- scope مسئله را باید بدانیم. تمام عوامل کار باید این را بدانند.

۲- تمام مفاهیم طراحی باید درک شود. قبل از کدگذاری باید طراحی بتوانند را درک کند باید premitive یا سواد وجود داشته باشد (گرنه روش های Agile وجود ندارد.)

۳- انتخاب زبان برنامه‌سازی: اگر requirement وجود داشته باشد بر اساس آن تعیین می‌شود یا اینکه در این مرحله تعیین می‌شود. هم نیاز از طرف کد و هم با محیط سازگار باشد

٤- محیطی که می‌خواهیم کار در آن انجام شود باید تعیین شود و tools) محیط برنامه‌نویسی

۵- test case برای تست کردن آن که باز نوشتن یک component با unit test برای آن اجرا کرد

اصول مربوط به گام دوم عبارت اند از:
۱- کد باید تا حد امکان ساخت یافته باشد less programming

۲- باید ساختمان داده مناسب انتخاب شود.

۳- معماری را باید خوب درک کنیم.

PAPCO

Subject:
Year. Month. Date. ( )

۴- شرط ها را خیلی ساده باید بنویسیم تا side effect بوجود نیاید

```
For I:=1 to 4 do
    I = I+2;
```

if ها را تو در تو بنویسیم ولی ادغام نکنیم

۵- حلقه های تو در تو را طوری بنویسید که قابل تست باشد.

```
while ( I < length(s) ) do
{
    while ( I < length(s) && s[i] = 0 )  I++;
    while (                              )
    while (                              )
}
```

باید طوری نوشته شود که قابل درک باشد

۶- اسامی با معنا انتخاب کنید.

۷- برنامه خوانا باشد (space , ...)

۸- self document باشد. instruction ها را طوری بنویسیم که حرف می زنیم یا سیستم بدانیم که برنامه چرا گم کردن تعداد خطوط باشد

اصول مربوط به گام سوم عبارتند از:

۱- ابتدا از اول تا آخر برنامه را بررسی کرده که اشکالی نداشته باشد سپس آن را اجرا می کنیم (walk through)

۲- test case ها را برای آن اجرا کرده و error ها را برطرف می کنیم

unit test را انجام می دهیم

۳- آیا می شود الگوریتم را بهتر کرد یا نه (خلاصه کردن برنامه)

PAPCO

31

Subject:
Year. Month. Date. ( )

test:

تست را برای این انجام می‌دهیم که خطا را پیدا کنیم. یک test case خوب این است که احتمال پیدا کردن خطا در آن زیاد باشد. هرچه شناخت فرد از مسئله بیشتر باشد test case بهتری می‌نویسد. بهتر است تیم دیگری تست را انجام دهد زیرا کسی که برنامه را نوشته سعی می‌کند مواردی که در نظر گرفته را چک کند که درست است یا نه.

اصول مربوط به این بخش عبارت است از:

۱- تست از نیازها بدست می‌آید. در نتیجه در همان اولین جلسه‌ای که برگزار می‌شود test case مشخص است. باید بتواند برای این نیازها برنامه داشته باشد ... analysis ... design ...

۲- ۸۰٪ خطاهای برنامه مربوط به ۲۰٪ برنامه است پس هیچ وقت دست نکشید.

۳- برنامه از پایین به بالا تست می‌شود. در خود بخش اول تست می‌کنیم بعد از ترکیب قسمت‌های مختلف، سیستم ما از اول تست می‌شود...

۴- تمامی حالات ممکن برای تست امکان‌پذیر نیست. در نتیجه نمی‌توانیم از تست به یک چیز درست برسیم. البته می‌توان ... component-level ... طراحی شد.

deployment:

delivery
support
feedback

اصول این بخش عبارت است از:

۱- باید به مشتری package کامل داده شود که مستقل از ما کار کند. باید ... مشخص شود: hardware ، OS ، peripheral device ، networking

۲- حداقل مشکلات برای trouble shooting

۳- ممکن است اشکالاتی ایجاد شود باید نظامی برای پشتیبانی تعیین کرده باشد.

۴- بالا رفتن دانش مشتری باعث بالا رفتن انتظارات مشتری می‌شود در نتیجه نباید زیاد قول دهیم و به مهندسین اجازه قول دادن داده شود. انتظارات باید کنترل شود.

نقش sys. engeer این است که این عناصر را برای یک سیستم مبتنی بر کامپیوتر در همهٔ سطوح سلسله مراتبی آن (macro element) تعریف کند (?)

Subject:
Year. Month. Date. ( ).

ه- بعضی شرکت‌ها که بازار انحصاری دارند کیفیت را برای آزاد کردن منابع پایین می‌آورند. بنابراین خطاها (118)
نگذاریم ابتدا باید خطاها را از بین ببریم و به مشتری تحویل دهیم

system engineer: PP
BPE
PP
engineer process model

فصل ششم

تولید محصول برای یک problem چگونه انجام می‌شود؟

دو گروه برای حل وجود دارد: (برای ترکیب سیستم compute base دو راه حل وجود دارد) (۱۲۲)

۱- product Eng: از یک راه حل قطعی استفاده می‌کند و راه حل را به صورت یک بار طرح می‌کرد. داشتن ابزار مهم نیست ولی این روش کاربرد چندانی ندارد و مشکل به این گونه حل نمی‌شود زیرا ابزار وجود دارد ولی متخصص برای استفاده از آن نیست

۲- مشکل مطرح می‌شود و محصول مشخص نیست. به این روش Bussiness process Engin. (BPE) می‌گویند. در این روش از مشکل شروع کرده مراحل مختلف را گذرانده و به محصول نرم افزاری می‌رسیم. یعنی برای پیدا کردن راه حل و محصول باید از مشکل شروع کنیم و مرحله مرحله جلو برویم تا به محصول برسیم که به این قسمت کار System Eng می‌گویند.

یک سیستم مبتنی بر کامپیوتر شامل عناصر زیر است:

- software: مهندسی نرم افزار در تولید software برنامه‌های کامپیوتری، ساختمان داده‌ها
کارهای مرتبط که برای تأثیرگذاری عملیات منطقی، رویه‌ها و کنترل‌هایی که مورد نیاز است استفاده می‌شوند
- hardware: تعیین hardware و شیوهٔ بکارگیری آن توسط مهندس نرم افزار مشخص می‌شوند زیرا آنها کار کردن با آن را می‌دانند. (ابزارهای الکترونیکی، ارتباطی و الکترومکانیکی ...)
- people: نیروی انسانی، فرهنگ، آموزش و استفاده از سیستم افراد در این سؤال بدون در نظر گرفتن افراد، software و hardware را تعیین کرد و باید فرهنگ در نظر گرفته شود.
زیرا افراد همان استفاده کنندگان از سخت افزار و نرم افزار هستند
- Database: تابع قوانین کشوری است. چطور می‌توان داده‌ها را نگه داشت
که از طریق نرم افزار قابل دسترسی است
- procedure: قوانین رویه و سیستم موجود زیرا سیستم paperless وجود ندارد
مراحل استفاده از عناصر سیستم

PAPCO

در سیستم های پیچیده یک سلسله مراتب از macro element ها وجود دارد که خود یک سیستم فنی کامپیوتری

Subject: یک سیستم فنی کامپیوتری که اجزای خودش را دارد که macro element ها را تشکیل داده که هر دوی اینها مثل اجزای اصلی یک manufacturing cell

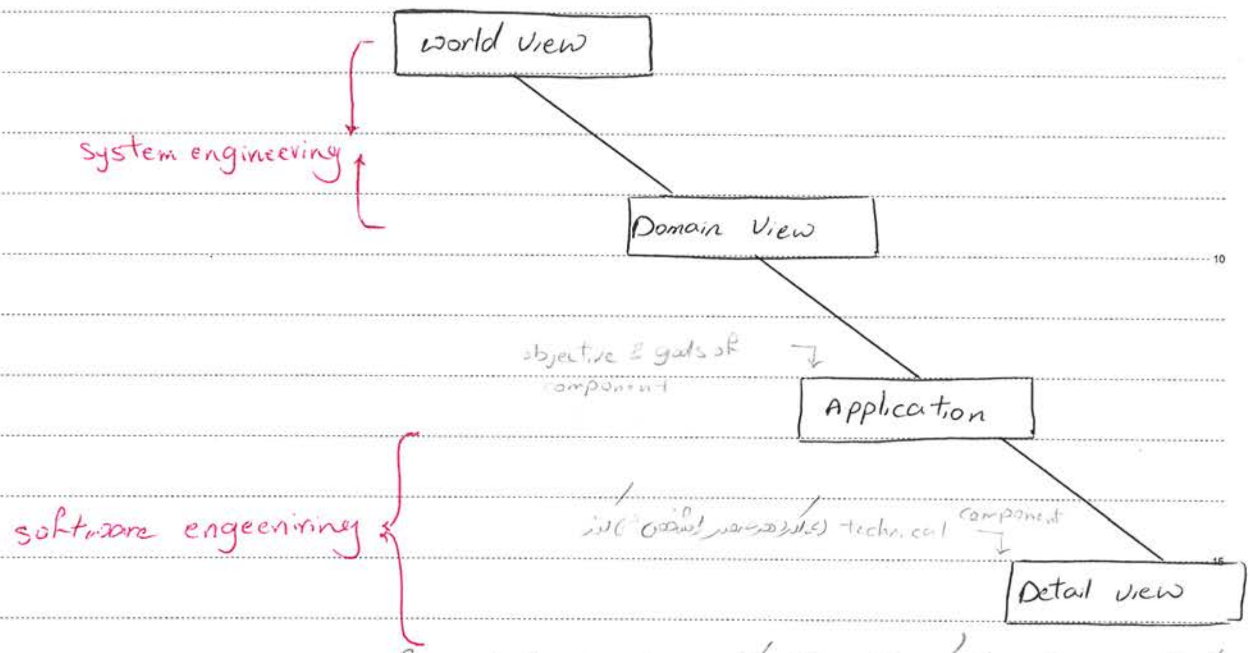
Year. Month. Date. ( )

سیستم فنی کامپیوتری هستند.

paperless (این اولین نیازی است که وجود ندارد زیرا e-signiture نیست در کنار هر سیستم یک دیتای کاغذی نیز وجود دارد.)

: document ← اطلاعاتی در مورد شیوه ارتباط های این سیستم (مثل ......)

برای اینکه بتواند در system engine نیاز به یک دید جهانی دارد



یک component می تواند برنامه کامپیوتری، ماژول، روتین، داده یا زبان برنامه نویسی باشد.

world view : اهداف کلی سیستم شیوه های کنترل و ترکیب و مشکلات در بعضی کارها آن را vision (هدف) در نظر می گیرند. هیچ کاری بدون داشتن آن نمی تواند به جایی برسد. در واقع با توجه به دیدی که از دنیا دارد قوانینی را تنظیم می کند که در ظاهر مرتبط نیست ولی با حسن انداز هایی که داریم به آن می رسیم.

با بررسی اهداف برای رسیدن به آنها یک سری سیاست ها را تنظیم می کنیم که به آن domain می گویند

Domain View : هر یک از domain ها بررسی می شود.

کارهای اقتصادی ← بورس ایران یا بانک — domain → در حقیقت یک domain view می رسیم.

بیمه — سیاستگذاری ترخیص

در این هر Domain نیاز به عناصر سیستم خود هستیم که این عناصر عبارت (data ، software ، hardware ...)

Subject:
Year. Month. Date. ( )

قسمتی از BPE است.

در این وجله از سندهایی بنام ISP) information strategy planning) استفاده می‌کنیم
این سند دارای سه جزء زیر است:

1) معماری داده‌ها ← زمانی که می‌خواهیم یک vision را در نظر قرار دهیم داده‌ها نیز باید در نظر گرفته شود زیرا جزئی است که سیستم را توصیف می‌کند. براساس داده‌ها
2) معماری Application ها
3) زیرساخت‌های تکنولوژی موجود در سیستم زیرساخت‌های تکنولوژی تعیین می‌شود

IT	Software
knowledge	Data
process	function
communicate	Behaivior

در واقع براساس اهداف یک سری domain بوجود می‌آید و سندی باید وجود داشته باشد که تعیین کند چه افرادی در سیستم وجود دارند، چه کارهایی را در domain view (vision) می‌خواهیم انجام دهیم. چه ساختار سیگنالی لازم است و وظایف هرکدام چیست. همه این موارد در سند ISP مشخص می‌شود، که علت ایجاد هر سازمان را تعیین می‌کند.
سپس برای شناسایی محیط این سند باید خوانده شود. اگر ISP در معماری وجود نداشته باشد باید ابتدا آن ایجاد شود و سپس سیستم تحلیل شود.

در سند Domain view ایجاد می‌شود BAA (Bussiness Analysis Area) می‌گوید که دیدگاه انجام کارهای هر domain در آن مشخص می‌شود و هریک از کارهایی که انجام می‌شود یک Application view است.

همین world view در product engineering وجود دارد ولی چون مراحل مشخص است در world view ابزارهای خودیاز آن را تعیین می‌کند و مراحل براساس آن طی می‌شود. عده‌ای دیگر معتقدند که براساس راه‌های مختلف لایه‌بندی stack holder تقسیم‌بندی در دیدگاه world view مشخص می‌شود.

مدل ترم اولیه عمومی و کلی از world view است یا detailed باید اجزای مورد نظر توجه شود.

Subject:
Year. Month. Date. ( )

System modeling: یک جزء اصلی در system engineering است

کاری که در این مراحل انجام می‌شود منجر به ساخت مدلی می‌گردد که در این مدل موارد زیر باید در نظر گرفته شود

۱- عناصر سیستم باید مشخص شود. دارندهای که نیازهای view را برآورده می‌کنند

۲- خروجی و output باید مشخص شود

۳- ورودی‌ها باید مشخص شود.

۴- پیوندهای بین عناصر که باعث می‌شود سیستم را خوب درک کنیم.

۵- اتفاقات و عناصری که نیازهای ما را می‌توانند تعیین کنند به خوبی درک شود.

چندین فاکتور نیز برای این مدل وجود دارد:

۱- فرضیات **Assumption:**

مدل باعث می‌شود که بتوانیم وقایع را به صورت منطقی ببینیم و این وقایع به صورت abstract در نظر گرفته می‌شوند. در واقع جزئیات را کم می‌کند و در حالت کلی در نظر می‌گیرد.

تعداد حالات ممکن را کاهش می‌دهد. مهندس سیستم با در نظر گرفتن جزئیات دقیق در مورد مجموعه‌ای از ...

۲- **Simplification:** ورودی و پردازش‌ها را کاهش می‌دهد

کلی نگری باعث می‌شود همیشه یک مدل ساده داشته باشیم و هم چنین قابل دیدن باشد

در واقع جزئیات حذف می‌شود.

یک سری variation (تغییرات) وجود دارد که می‌توان شرط‌های ذخیره موجود که مشابه هم هستند را در یک گروه قرار داد که به صورت یک پارچه آنها نگاه کنیم. در نتیجه این فرض‌ها به صورت قانون در می‌آید.

PAPCO

یک مهندس سیستم باید تأثیر عناصر سیستمی مختلف را برای انتخاب مدل در نظر بگیرد.

Subject:
Year. Month. Date. ( )

۳- limitation:

عواملی هستند که به ما از مسئله تحمیل می‌شود مثلاً در خرید یک cpu اولین چیزی که ما را محدود می‌کند انتخاب cpu ای است که با بقیه اجزا compatible باشد. یا زمان اتمام پروژه توسط کارفرما

۴- constraint:

در واقع limitation از طبیعت مسئله ناشی می‌شود ولی constraint از تصمیمات ناشی می‌شود. مثلاً در یک Air taxi در حوزه خلیج فارس. انتخاب هواپیمایی که بتواند روی سطح ۷۰۰ متری فرود آید limitation مسئله است ولی استفاده از یک GSP constrain است

۵- performance: (preference)

همیشه در مدل یکی از فاکتورها راه حل های ترجیحی است یعنی که هر راه حلی هدف است ولی ترجیح می‌دهیم کار خاصی را انجام دهیم. زیرا رضایت مشتری در اولویت است

از ص ۱۲۱ باید خوانده شود

در اینجا مدل ساخته نمی‌شود بلکه فقط ویژگی‌ها و شرایط آن تعیین می‌شود تا بعداً ابزار مدلسازی مشخص شود.

فصل ۷:

Request engineering:

درک بهتر مسئله از نقطه شروع در یک مهندس نرم‌افزار به یک سیستم است.

از جمله فعالیت هایی که در این بخش انجام می‌شود:

این امکان را به مهندس نرم افزار می‌دهد تا مسئله را بهتر درک کند

- جمع آوری نیازها
- مدل سازی — ۱- inception: کارهایی که scope مسئله و ماهیت آن را مشخص می‌کند.
- تحلیل — ۲- elicitation: به مشتری این امکان را می‌دهد تا چیزهایی که مورد نیاز است را تعریف کند
- مدیریت — ۳- elaboration: نیازهای پایه‌ای اصلاح و دوباره بازنویسی می‌شوند.
- spicify — ۴- negotiation: اولویت نیازها تعیین می‌شود.
- validation — ۵- specify:

۶- validate: آیا محصول درست تهیه شده است یا نه.

نکته: بعضی معتقدند در ISP می‌توان مدلسازی کرد ولی در آن ها هنوز ورودی و خروجی مشخص نشده است

یکی از اقدامات مهندسی نرم‌افزار است که در طول مراحل communication و modeling رخ می‌دهد

ادامه دارد.

در requirement engineering علاوه بر مرحله design و construction هم می‌سازد. بدون این اقدام ریسک در مورد عدم رضایت است

Subject:
Year. Month. Date. ( )

بالا رفتن هزینه بالا می‌رود.

نکته: در validation ابزاری است که از روش‌ها و ابزار استفاده می‌شود تا مطمئن شویم در حالت پایداری قرار دارد. اولین مرحله Formal technical review است

task هایی که در این مرحله وجود دارد عبارت اند از:

۱- Inception

۲- Elicitation

۳- ~~Negotiation~~ Elaboration

۴- Negotiation ← تعیین اولویت‌ها، محدودیت منابع و اهداف متفاوت مشتریان

۵- specification ← function و performance یک سیستم و محدودیت‌هایی که با پیاده‌سازی آن درگیر است مشخص می‌شود

۶- validation ← specification را ارزیابی می‌کند تا مطمئن شود هیچ حالت مبهمی وجود ندارد.

۷- Requirment management ← ابزارهایی برای مشخص کردن نیازهایی که در اجرای تکمیل تغییر یابند. identifier است

سؤالات context-free ← پرسیده می‌شود. هدف هم پایه‌ای شدن اطلاعات به بیشترین حالت

۱- Inception: ماهیت و ... مشخص می‌شود

نقطه شروع یک پروژه کجاست. برخورد ابتدایی با پروژه چندین هدف را دنبال می‌کند:

۱- stackholder ها را بشناسیم. چه کسانی پشت آن هستند و افرادی که با آن کار می‌کنند چه کسانی هستند
کسی که علاقه مستقیم نسبت به سیستم دارد یا از آن سود می‌برد.

۲- نیازهای مختلف شناسایی شود. view point همه افراد و دیدگاه‌های آنها جمع‌آوری شود.
از آن جایی که stakholder ها متفاوت هستند، نیازها و نیازهایشان متفاوت است. اطلاعات از view point های مختلف جمع‌آوری شود

۳- روحیه همکاری و قابلیت‌های خود را به نمایش بگذاریم. مشکلات و ریسک‌ها را توضیح دهیم که یک نوع present کردن خودمان است (collaboration)

برای همین در Inception باید یک سری مطالعه انجام شود. برای مرحله اول برخورد با سیستم در این مرحله است.

۴- سؤالاتی که می‌توان در این مرحله بیان کرد عبارت است از: در گروه اول سؤالات به صورت زیر است

۱- چه کسی پشت مسئله است؟ و انگیزه اصلی او چیست؟ ← توجه آن بر روی customer و stakeholder است.

۲- چه کسانی از آن استفاده می‌کنند؟

۳- صرفه اقتصادی این مسئله چیست؟

۴- راه حل دیگری برای صرفه اقتصادی وجود دارد؟

PAPCO

Subject:
Year. Month. Date. ( )

گروه دوم: این گروه به تیم کمک می‌کند تا ابزاری را که در اختیار دارند بهتر درک کند

۱- خوب بودن یعنی چه؟ چه جلوه‌ای برای آن دارد؟

۲- چه مشکلاتی ممکن است در اینجا اتفاق بیفتد؟ می‌توان خود را با آدرس دادن ریسک‌ها و تحلیل‌ها present کرد که این مستلزم تجربه و مطالعه شخصی است.

۳- آیا این محصول باید ویژگی خاصی داشته باشد؟ (این ویژگی می‌تواند (مثلاً) از خوب بودن نیز مشخص شود) یک ویژگی روی عمل قرار می‌دهیم و عملکردی نیست

گروه سوم: این گروه از سؤالات نباید پرسیده شود و از رفتار شخص باید فهمیده شود (meta question

۱- آیا کسی که با او صحبت کردیم فرد درستی بوده یا نه (صلاحیت صحبت کردن را دارد یا نه؟)

۲- سؤالات نباید نامربوط باشد و این حالت classic نیست می‌توان به صورت شوخی و طنز بیان شود

۳- آیا کس دیگری در سیستم هست که بتواند راهنمایی کند؟ (افرادی که بالاتر هستند لزوماً کامل‌ترین اطلاعات را ندارند)

۴- آیا سؤال دیگری هست که بخواهد بگوید؟

در نتیجه با این سؤالات یک سری ایده کلی در مورد سیستم بدست می‌آید ولی نمی‌توان سیستم را به تمامی شناخت انگیزه، قراردادی و درخواست آنها باید مشخص شود ولی به fizible نمی‌رسیم. به این مرحله Elicitation یا discovery knowledge یا requierment gathering می‌گویند.

این سؤالات نوعاً در این جلسه باید پرسیده شود در مراحل بعدی برای فهم مسئله باید از elicitation استفاده شود.

۲- Elicitation: (استخراج)

نکاتی که در این مرحله باید در نظر گرفته شود عبارت است از:

۱- برای جمع‌آوری اطلاعات باید یک همکاری دوجانبه بین مشتری و متخصص باشد. جمع‌آوری اطلاعات آنی نیست.

الف) باید جلسات مستمر با مشتری داشته باشیم.

مثلاً کار دچار مشکل است اگر مشکل فنی باشد قابل حل است ولی اگر مسئله فنی به شکل اجتماعی و اخلاقی کشیده شود قابل حل نیست که شرط آن جلسات مستمر است.

ب) همیشه قواعدی برای مشارکت داشته باشیم: نقش‌ها را کنار بگذاریم و آن را مشخص کنیم

PAPCO

quality Function deployment (QFD) تکنیکی است برای تبدیل نیاز مشتری به نیازهای تکنیکی برای نرم افزار. تاکید آن
Subject:
Year. Month. Date
بر روی فهم چیزهایی است که برای مشتری ارزشمند است و سپس این ارزش را از طریق فرایند مهندسی به سایر سازمان ها گسترش می دهد
پس از دیاگرام و ماتریس های مختلف استفاده می شود تا نیازهای expected ایشان را پیدا کرده و به سمت نیازهای exciting / رو به جلو پیش برویم.

۲- ذره ذره اطلاعات را به حالت Formal ببریم که هم قابل فهم شود و هم در سطح عمیق دقیق باشد ولی مشتری هم آن را بفهمد.
باید گونه ای با text ها آمیخته باشد تا مواردی که Formal پذیر نیست اعمال شود (نمودار، فلوچارت، تصاویر)

۳- کار سیاسی انجام نمی دهیم. یک قدم از شناسایی و به دنبال راه حل های های قدیمی تر در سیستم های جدید چیزی است که می بینیم و اجازه ورود به اطلاعات شخصی را نداریم.

✓ اطلاعات جمع آوری شده به یک سری نیازها و خواسته های مشتری به دو گروه تقسیم می شوند:
- Functional:
خواسته های عملی که مشتری دارد. مثل برنامه مکانیزه برای ثبت نام دانشجویان ← یک عمل انجام می شود

- nonFunctional:
ویژگی های یک عمل ← کارنامه ها در یک روز صادر می شود.

✓ از نظر مشتری نیازها به سه گروه تقسیم می شوند:
- normal:
مشتری یک چیزی را می خواهد و باید براساس آن انجام دهیم. اگر این نیازها برطرف شود مشتری راضی می شود

- expected:
مشتری یک سری نیازها را جزء بدیهیات می داند و بیان نمی کند.

- excited:
با انجام آن جلب رضایت مشتری می شود. ولی انتظار ندارد که برای آن انجام شود

هدف فاز Elicitation تعیین requirement ها است.

PAPCO

usecase : باید گونه استفاده مشتری از function و feature ها را بدانیم تا بتوان به سمت مهندسی نرم افزار تدریجی حرکت کرد
سناریو های مختلف که گونه تعامل مشتری با سیستم را نشان می دهد usecase می گویند
و ۱۵۹ بار سازی usecase

Subject:
Year. Month. Date. ( )

در نتیجه محصولات Elicitation عبارت اند از:

۱- نیازها

۲- امکان سنجی ها

۳- scope مسئله

۴- لیستی از همه مشتری ها و آنهایی که با آن درگیرند

۵- ورودی، خروجی و interface های آن ها تعیین می شود.

۶- توصیف محیط تکنیکی و استفاده از تکنولوژی

۷- محدودیت هایی که وجود دارد مشخص می شود.

۸- prototype تعیین می شود.

وقتی لیستی از نیازها داریم این نیازها باید با ۵ شاخص مشخص شود:

۱- urgency : چقدر ضروری است و اهمیت دارد

۲- visibility

۳- benefit

۴- priority : اولویت برای انجام

۵- possible solution

۶- PIECES

performance ← کارایی سیستم

information ← ورودی و خروجی نیاز

economition ← جنبه اقتصادی

control privacy

service ← میزان

efficiency ← تغییرپذیری

مقایسه با چیزهایی که موجود هستند

document ها را کم کنیم

۱- در نظر گرفتن سود که چگونه این کار را بکنیم
۲- اگر کار انجام شود چه محاسن اقتصادی دارد

Subject:
Year. Month. Date. ( )

نکته: اصولاً هر چیزی که ارزشمند است اولویت بالایی ندارد. در کار حسابداری تعیین سود و زیان ارزشمند است ولی در اولویت نیست و اول باید هزینه‌ها تعیین بشود

برای همه باید این شاخص‌ها لحاظ شود تا چیزی از قلم نیفتد

برای اینکه در این مرحله کاری باید تعیین شود scope است که باید آن را بصورت individual قرار دهیم یعنی چه کسانی از آن استفاده می‌کنند؟ تا کاربران چه نقشی ایفا می‌کنند؟ از چه کسی چه اطلاعاتی گرفته می‌شود
هدف این است که بدانیم bussiness event ها چه هستند عناصر خارجی (external entity) چه هستند و با چه کسانی ارتباط برقرار می‌کنند چه knowledge هایی وجود دارد. هر کدام اینها یک base اولیه ایجاد می‌کند که به آن base line می‌گویند.
در این مرحله سه چیز وجود دارد که روی آن کار می‌کنیم:
۱- عملیات
۲- مشکلات موجود
۳- فرصت‌ها برای بهره‌برداری

برای اینکه elicitation بتواند این کارها را انجام دهد ۷ activity وجود دارد:

۱- نمونه‌برداری
۲- مشاهده
۳- مطالعه تحقیق
۴- مطالعه سوابق
۵- پرسش‌نامه
۶- مصاحبه
۷- نمونه‌سازی
۸- سمینار

PAPCO

Subject:
Year. Month. Date. ( )

۱- پویش و برداری:
مستندات زیادی در سیستم وجود دارد یکی از روش‌ها برای آگاهی مراجعه و استفاده از آن‌ها شامل مکاتبات غیر رسمی و نگاه کردن است. اطلاعاتی که می‌توان از روی یادداشت‌ها، نامه‌ها، صندوق پیشنهادها، شکایت‌ها، گزارش کار روزانه، اندازه‌گیری‌های مختلف، تقاضاهایی که قدیم داشتند را دارند.
فعالیت‌هایی که در این مرحله انجام می‌شود عبارت‌اند از:
۱- آگاه‌سازی (برنامه استراتژیک) strategic plan: آگاه‌سازان برنامه استراتژیکی وجود داشته باشد باید به سطوح بالاتر رفت و دید که در کشور چه برنامه‌ای برای آن وجود دارد.
۲- اهداف سازمانی
۳- سیاست‌گذاری‌ها
۴- (SOP) standard operation procuder
۵- فرم‌های کاغذی که وجود دارد (manual، DB، پرینت‌ها، output‌ها)
۶- گزارش سیستم‌های قبلی (فلوچارت‌ها، مستندات، حروف آموزشی)
۷- برآورد اندازه‌ها: از آمارها استفاده کرده و از این اطلاعات جمع‌آوری شده چیزهایی را خواهد داشت تا اطلاعات کافی بدست آید.

۲- مشاهده:
بازدیدهایی است که از محل کار باید انجام دهیم. به صحبت‌ها نباید اطمینان کرد. از فرصت‌های این روش عبارت است از:
۱- قابلیت اطمینان: به چیزی که دیده‌اید می‌توانید اعتماد کنید.
۲- اطلاعاتی که هیچ‌وقت نمی‌دهد را می‌توان دید.
۳- می‌توان اندازه‌گیری کرد: کارهایی که مردم دیگر در زمان اندازه‌گیری می‌کنند متفاوت است و همه underestimate و overestimate دارد.
۴- خیلی ارزان است.

معایب آن عبارت است از:
۱- طرف مقابل و خودتان راحت نیستید. رفتارهایی نشان دهند که از نظر اطلاعاتی ارزنده نباشد.

PAPCO

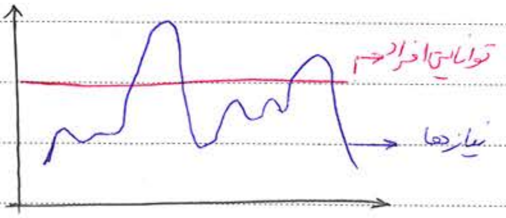
13

The God speaks in
the silence of heart;
listening is the beginning
of the prayer ...

Mother Tresa

Subject:
Year. Month. Date. ( )

۲- در دوری بازدید ممکن است مشکلی بوجود بیاید. هر انسان برای کار کردن در روز توانایی دارد که دارای توزیع یکنواخت است ولی نیازها تابع توزیع یکنواخت در روز ندارد.



برای استفاده بهینه از توانایی ها می‌توان دو کار کرد:

۱- یک مدیر کاری می‌تواند آنقدر منابع را زیاد کند که pick را بپوشاند ولی در ساعات دیگر این منابع بلا استفاده می‌ماند

۲- فرهنگ صرفه‌جویی را به مردم یاد داده و بخواهیم که pick را کاهش دهد و در ساعاتی که استفاده از آن کمتر است کارها را در آنجا انجام دهیم

ممکن است نیروی انسانی در یک شرکت طوری انجام شده باشد که در ساعاتی که مراجعه خیلی زیاد است نتوانند پاسخگو باشند برای همین خیلی مهم است که در چه زمانی بازدید انجام شود.

راهنمایی های موجود برای این مرحله عبارت است از:

۱- قبل از مشاهده چه کسی، کجا، چرا و چگونه برای بازدید می‌رود و چه چیزی را بازدید می‌کند.

۲- باید اجازه بگیریم و قبل از آن هماهنگی صورت بگیرد.

۳- هدف از مشاهده باید توضیح داده شود.

۴- profile ی از مشاهده تهیه کنیم. در آن مدل چیزی ننویسید و دقت کنید یادداشت بردارید.

۵- روی یک کار خاص focus نکنید. نباید کاری که برایش جذابیت است احساسات نشان دهید.

۶- فرض نکنید و هر چیز را سوال کنید.

۷- در کار وقفه ایجاد نکنید.

Subject:
Year. Month. Date. ( )

۳- پرسش نامه:

از مشکلات این روش عبارت است از:

۱- الزامی برای جواب دادن به پرسش ها وجود ندارد و عده ای به همه سؤال ها جواب نمی دهند.

۲- ممکن است درست جواب ندهند، برای همین در آمارگیری های کشور فردی می آید و خود پرسش ها را پر می کند.

۳- اگر فردی سؤالی را نفهمد فرصتی برای فهمیدن و درست جواب دادن ندارد.

۴- طراحی پرسش نامه بسیار دشوار است.

در هیچ جای دنیا پرسش نامه روش مناسبی نبوده است. پرسش نامه ها به دو دسته تقسیم می شوند:

۱- Free format: جدولی نیستند و سؤالی پرسیده شده و هر آنچه که بخواهیم می توان توضیح داد.

۲- Fim format: به صورت جدولی است که می تواند چند گزینه ای باشد یا ranking داشته باشد.

سؤالات پرسیده شده به صورت دوم برای اندازه گیری ها امکان پذیر است حتی بعضی از اندازه گیری ها نیز با این روش جواب نمی دهد.

ارکان مطالعه:

۴.۱- مطالعه تطبیقی:

کارهایی که دیگران انجام داده اند را مورد مطالعه قرار می دهیم این کار کمک می کند تا اطلاعاتی که مشتری در مورد کار خود نمی داند را بفهمیم و ایده های جدیدی به او پیشنهاد کنیم و می توان از الگوهای دیگران نیز استفاده کرد. و حالات و ویژگی های آن مشخص می شود.

۵.۲- مطالعه سوابق:

فعالیت های مکانیزه و کامپیوتری که قبلاً انجام شده را مورد مطالعه قرار می دهیم. چه ویژگی های درشتی دیده که برای انجام پروژه انتخاب کرده است؟ چرا با شرکت قبلی قطع همکاری کردند؟ در واقع سوابق شرکت باید استخراج شود. دستیم با اشکار زمان Elicitation کمتر شده و می توان در رقابت پیروز شد.

PAPCO

Subject:
Year. Month. Date. ( )

۶- مصاحبه (interview) :

در پرسش نامه مهارت فردی نقشی ندارد ولی در اینجا چرا! پس ویژگی یک طراح سیستم با ویژگی های یک مصاحبه کننده برابر است.

از محاسن این روش عبارت است از:

۱- سؤالات می تواند open باشد ممکن است در جواب سؤال چیزی به ذهنش برسد بپرسد

۲- می توان از طرف مقابل feedback بگیریم

۳- در صورتی که به سؤال متفاوت جواب بدهند نشان بدهد که نفهمیده می توان سؤال را به گونه ای دیگر بیان کرد

۴- nonverbal communication داشته باشد و اطلاعاتی را از شیوه رفتار آن بفهمد.

از معایب آن عبارت است از:

۱- زمان گیر است.

۲- به مهارت فردی بستگی دارد و شاید نتوانیم با او صحبت کنیم (اجازه صحبت کردن نمی دهد)

دو نوع مصاحبه وجود دارد:

۱) unstructure : هدف کلی داریم و نمی دانیم چه چیزی از آن نتیجه می شود scope را می دانیم ولی سؤالات مشخص نیست

۲) structure : کلیه سؤالات ها مشخص checklist است

مصاحبه شونده
تاریخ
ساعت
محل

زمان	سؤال	جواب
1-2	انگیزه کار	

یک plan برای انجام مصاحبه ← ایجاد کرده که ماکسیمم زمان 30 دقیقه جواب می دهد.

PAPCO

Subject:
Year. Month. Date. ( )

فرد مصاحبه کننده چه ویژگی های فردی باید داشته باشد:

۱- قبل از مصاحبه خود را آماده کنید

۲- با چه کسی قرار است صحبت کنم و متناسب با او خود را آماده کنم

۳- شرط سنی را در نظر بگیریم باید فرد در مقطعی باشد که مورد احترام بوده و در back ground او تجربه را مشاهده کنیم. زیرا باید تناسب سنی وجود داشته باشد.)

۴- تکان نروید. تعادل از خیلی مهم است.

۵- مشکلاتی که مانع از صحبت می‌شود یا نقص عضو معمولاً در ارتباط برقرار کردن اثر خوبی نمی‌گذارد

۶- تناسب اندام باید وجود داشته باشد.

۷- تناسب پوشش با فردی که صحبت می‌کنید

۸- آرایش صورت خیلی اهمیت دارد ← زیبایی یک ارزش است و در برخورد اول تأثیرگذار است.

۹- نجوا نکنید (در گوشی حرف نزنید)

۱۰- میوه هایی را برای خوردن انتخاب کنید که راحت باشد. بعد از مدیران شروع کنید

۱۱- فضولی موقوف (به حریم افراد نمی‌توانید دخالت کنید)

۱۲- مصاحبه را در محیط خارج از کار برگزار کنید در نتیجه می‌توان در مورد مسائل غیرکاری نیز صحبت کرد

۱۳- شوخی زننده نکنید

۱۴- خنده از حد تبسم بیشتر نشود.

۱۵- بحث الکی نکنید

۱۶- موضع گیری اقلیتی و سیاسی ممنوع

۱۷- غیبت دیگران را نکنید.

۱۸- از طرف تعریف کنید

۱۹- ادبیات زیبا باشد.

۲۰- طرز ادا کردن جملات، زیبا باشد.

۲۱- به همه احترام بگذارید.

۲۲- موقع حرف زدن به صورتش نگاه کنید.

۲۳- حفظ فاصله بین خود و فرد مقابل (proxemic)

۲۴- خیلی حرف نزنید

۲۵- ضبط نکنید (اگر شنونده خوبی نیست)

PAPCO

Subject:
Year. Month. Date. ( )

۲۶- اگر چند نفر هستید هر با هم سؤال نکنید
۲۷- دلت نوشتید یعنی هر لغت ننویسید.
۲۸- کسی که سؤال می‌کند یادداشت بر ندارد و همراه او این کار را انجام دهد
۲۹- سؤال‌ها از نوع loaded نباشد مثلاً استاد تمام تمرینات را باید حل کند.
۳۰- loading question نباشد یعنی نباید در سؤالات راهنمایی کنید.
۳۱- سؤالاتی که شامل پیشنهادات هست نباید مطرح شود
۳۲- سؤالات واضح، کوتاه و زبان خوب ادبیات زننده نباشد.
۳۳- از سؤالاتی که تهدید آمیز باشد اجتناب کنید.
۳۴- زمانی که منظور، جامعه مشتری است برخورد از کلمه "شما ها" نباید استفاده شود این شروع جدا کردن است
۳۵- همیشه در ابتدا یک تقدیر چیزی باید وجود داشته باشد.
۳۶- پیشنهادات را بگیرید.
۳۷- خوب به هر چه هست گوش دهید.
۳۸- تشکر و قدردانی کنید.
۳۹- به همدیگر نگاه نکنید (بار اطلاعاتی زیادی دارد و حس بدی می‌دهد)
۴۰- یک دقیقه آرام بگیرید.
۴۱- از ادامه سؤال‌ها در بحث‌های اصلی اجتناب کنید.
۴۲- تا جواب یک سؤال تمام نشده، سؤال نکنید.
۴۳- هیچ چیزی را فرض نکنید
۴۴- با سرعت وارد جلسات شوید.
۴۵- طوری رفتار کنید که طرف مقابل راحت باشد.

۷- prototype : (نمونه)
I) یک base درنظر می‌گیرد و تغییرات را اعمال می‌کند در واقع با یک نمونه از قبل داشته باشیم و فقط تغییراتی که لازم است در آن ایجاد می‌کنیم.
II) ممکن است یک مشتری مشکل را بداند و فقط تصمیم می‌گیرد که کاری را انجام دهد. این نوع نمونه‌ها زمان بر و سخت است. با مشتری پیش می‌رویم و مشکل درک می‌شود تا طراحی به درستی انجام شود.

PAPCO

Subject:
Year. Month. Date. ( )

می‌خواهیم ببینیم physibt و usable هست یا نه و حکم می‌کنیم آیا می‌تواند ویژگی مشتری را داشته باشد یا نه

از معایب این روش عبارت است از:
۱- زمان کمی را برای کشف حقایق لازم داریم
۲- هزینه برداری یک مکانیزم آموزشی برای مشتری کم هزینه است.
۳- با یکدیگر مشکلی را حل می‌کنیم. همراه بودن با افراد رابطه خوبی را ایجاد می‌کند.

در واقع نمونه یک sample ایجاد می‌کنیم و هدف این است که آیا این کار قابل visible است یا نه

معایب آن نیز عبارت است از:
۱- زمان زیادی می‌خواهد
۲- Functionality را نشان می‌دهیم ولی nonfunctionality (performance و ...) در آن لحاظ نمی‌شود
۳- برای استفاده از ابزارهای سطح بالا باید آموزش بدهیم و هزینه زیادی نیاز دارد.

۸- نشست‌های بسیارها (GRP):
افرادی که ذی‌نفع هستند را در یک جا جمع می‌کنیم و از صحبت‌های آنها اطلاعاتی کسب می‌شود. یکی از تکنیک‌های به کار رفته brain storm است. یعنی هر چیزی که به ذهن فرد می‌رسد یادداشت می‌شود و بعد نظرات گروه‌بندی می‌شوند. این روش به ندرت به کار می‌رود.
عقیده است بسیار دور از محل کار باشند تا interrupt ای به وجود نیاید

استراتژی استفاده از این activity ها به صورت زیر است:
۱- مطالعه: مطالعه نکرده هیچ کاری را شروع نکنید. اطلاعات می‌توانند کلی باشند.
۲- اولین مصاحبه (inception) با بالاترین مقام انجام می‌شود
۳- مشاهده
۴- مطالعه تطبیقی
۵- مطالعه سوابق
۶- پرسش‌نامه را هیچ وقت روی اول حساب نکنید

PAPCO

Subject:
Year. Month. Date. ( )

۷_ ترکیبی از مصاحبه های دهد } فرایند دوباره تکرار می شود.
۸_ مطالعه ← مطالعه سوابق جزء به جزء
۹_ عکس برداری

بیشترین وزن را مصاحبه ها دارند.

اطلاعات و نیازهای جمع آوری شده در مرحله elicitation باید ویژگی هایی داشته باشند که عبارت است از:

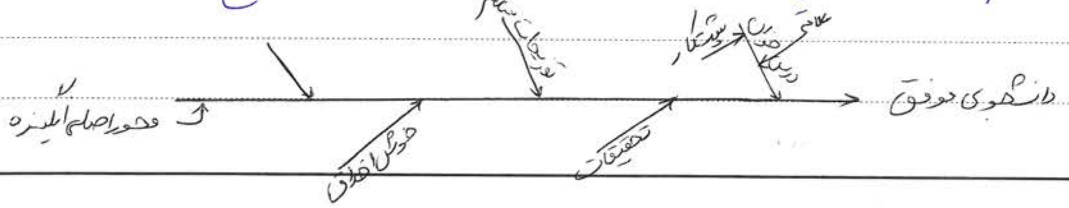
۱_ نیاز جمع آوری شده کامل باشد: complete
۲_ consistent
۳_ feasible امکان پذیر
۴_ Accurate دقیق
۵_ trackable قابل ردیابی
۶_ requiered مورد نیاز
۷_ verifiable

پس باید تحلیل کنیم که آیا نیازی از قلم افتاده یا نه؟ در تضاد با یکدیگرند؟ آیا نیازها می توانند امکان پذیر نباشند (infeasible)؟ ابهامی در درک آنها وجود دارد یا نه؟ (Ambique)

برای این تحلیل باید بتوانیم نیازها را مدل سازی کنیم و به صورتی نمایش دهیم. شکل های نمایشی در elicitation عبارت اند از:

۱) Ishikawa / fishbone:

محور اصلی انگیزه را در نظر گرفته و تمام عامل هایی که بر روی موجودیت می توانند تاثیر بگذارند با دلیل مشخص می کنیم. در سطح بعدی عوامل دوم بر روی عامل های دیگر شده را مشخص می کنیم در نتیجه همه عامل ها را می توان رسم کرد. مشتری نیز به راحتی می فهمد و اگر نقصی وجود داشته باشد در رفع آن کمک می کند.



Subject:
Year. Month. Date. ( )

2: decision table

همونطوری که صورت زیر ایجاد کرده که شروط مختلف در سطر قرار می‌گیرند. در این جدول می‌توان راه‌های قرار گیری از قانون را مشخص کرد. برای ماشین قواعد و آیین نامه‌هایی که وجود دارند به کار می‌رود.

condition																
جنسیت	M								F							
درجه تحصیلی	باسواد				بی سواد				باسواد				بی سواد			
شاغل	O		N		O		N		O		N		O		N	
سن 0,1,2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2
action1																									X					
action2																														

پس action ها را وارد می‌کنیم اگر فردی در گزینه‌های بیان شده را در حالت خاصی داشته باشد باید action مربوطه را انجام دهد. ممکن است برای یک گزینه چندین کار وجود داشته باشد که با شماره 0 و 1 و 2 گزینه مربوطه وارد می‌کنیم در نتیجه توالی انجام کارها نیز مشخص می‌شود.
بوسیله این جدول تعیین می‌شود که برای کدام بخش‌ها قانونی وجود ندارد. کجا مغایرت وجود دارد و خلأهای اطلاعاتی را تعیین می‌کند.

3: تعیین فرم:

فرمی طراحی می‌کنیم که شامل موارد زیر باشد و با پر کردن فرم شروط برای هر نیاز قابل جمع بندی هستند.

شماره نیاز	تاریخ
عنوان نیاز	ارزش critical نیاز باید انجام شود یا نه.
توصیف	خوب است که انجام شود یا اختیاری
نوع نیاز	الگوی PIECES
حریمیات	

P4PCO

Feature traceability table : هرنیاز به کدام دسته مربوط می‌شود

Subject: subsystem : نیازها را براساس subsystem هایی که آنها را انجام می‌دهند دسته‌بندی می‌کنیم

Year. Month. Date. ( )

---

۴) جدول نیز وجود دارد که می‌توان از آن در تحلیل‌ها استفاده کرد که شامل موارد زیر است: ← مربوط به درس requirement management است.

I) source traceability : یک جدول است که ستون نیازها و سطرها نشان دهنده منبع ورود نیاز است و با ضرب در زدن table می‌توان مشخص کرد که منشأ هر نیاز چه بوده است.

II) interface traceablity table : باید user و stackholder هر نیاز مشخص شود رابط نیاز با کسانی که با آنها در ارتباط هستند را نشان می‌دهد.

III) Dependancy traceability table : سطر و ستون نیازهستند و نیازهایی که با هم مرتبط هستند را ضرب در می‌زنیم. برای این کار می‌توان subsystem ها را مشخص کرد.

۵) use case : scenario-Based model ←

برای نمایش انجام یک وظیفه به صورت اتوماتیک یا دستی انجام می‌شود. برای ساخت usecase ها actor ها را مشخص می‌کنیم. یک function را انتخاب می‌کنیم و actor آن را مشخص می‌کنیم.

actor : موجودیتی است که یک وظیفه (عمل) را initiate می‌کند. یک مفهوم Abstract است و هیچ پدیده مشخصی قابل اطلاق نیست و به عنوان یک trigger کننده عمل می‌کند. actor می‌تواند روان نیز باشد که usecase آن "نسبت حضور افراد در یک حمل" است. actor یک پدیده انتزاعی است و در هر حمل نقشی را بازی می‌کند و وظیفه اش بررسی نقش‌هاست نه فرد با موجودیت خاص.

حسن حسنی ← دانشجو، کارمند، مدرس

یک فرد ممکن است نقش‌های مختلف داشته باشد و به کدام این نقش‌ها هم است.

این نمادها به صورت زیر تبدیل به یک document می‌شود.

usecase بیان یک وظیفه یا فعالیت است و actor یک trigger برای use-case است.

یک actor می‌تواند یک رویداد نیز باشد که پیش‌بینی شده و هم پیش‌بینی نشده

PqPCO

Subject :

Year . Month . Date . ( )

swimlane Diagram

Actor	دوری	Use-case نام	توصیف
دانشجو	نام درس	حذف درس	

Usecase-name

actor

توصیف

مرجع

انواع رویدادهای سیستم	انواع رخ دادهای actor
سیستم usecase است	تقاضای حذف درس

post condition

Actor

system

association

Elaboration یکی از مراحل مفید است ولی باید دید که آن را متوقف کنید. منظور از این است که شرایط و الگوهای تکراری کنید یک پایه ثابت برای طراحی بوجود آید. نتیجه کار این مرحله تولید مدل آنالیز است

Subject:
Year. Month. Date. ( )

انواع ارتباط‌ها عبارت‌اند از:

۱- Association: یک ارتباط بین یک actor و یک usecase است. (نشانه: خط ساده ———)

۲- generalization: معمولاً یک سری usecase داریم که می‌توان آن را general کرد. ———▷

خرید کالا < خرید نقدی / خرید اعتباری

در واقع یک رابطه بین usecase هاست. (انتقال جزئیات به یک کلی می‌رود)

۳- include: یک سری عملیات که usecase های مختلف را باهم به کار می‌گیرد. include——>
همواره همه آنها اجرا می‌شوند (حذف درس مثلاً شامل تعداد واحدها و تقدم و تأخر اخذ درس است)

۴- extended:
زمانی که برای حالت‌های خاص یک کار باید انجام شود. و برای شرایط خاص به کار می‌رود. extend- - - ->

به عنوان مثال در دریافت پاسپورت مهر کردن include است ولی انگشت‌نگاری برای گروه خاصی انجام می‌شود پس از نوع extend است.

این دیاگرام یک آینه تمام‌نمای سیستم را ایجاد می‌کند که مشخص می‌کند در واقع با چه کسانی هستند به آن usecase diagram می‌گویند.

برای تحلیل احتیاج به یک مدل داریم که دارد Elaboration می‌شود.

فصل هشتم:

3- Elaboration: ← نیازهای جمع‌آوری شده در مرحله قبل اولویت‌های شبیه‌سازی نشان داده می‌شود

تا اینجا نیازها را بدست آوردیم حال می‌خواهیم حلاجی نیازها و سیستم که باید طراحی شود را ترسیم مدل باید به گونه‌ای باشد که بتوان مشکلات و نارسایی‌ها را از آن معلوم کرد. elicitation تا اینجا را در تحلیل سیستم انجام می‌گویند. همیشه یک وضعیت موجود داریم که می‌خواهیم آن را به وضعیت مطلوب تبدیل کنیم

وضعیت موجود | وضعیت مطلوب

phisical ——> logical ——————> logical ——> phisical

- phisical: فیزیک جاهایی که دارد انجام می‌شود / به سراغ چیزهایی می‌رویم که فیزیک ماهیت می‌دهد
- logical: رسیدن به سراغ ارتباطها و منطق / تصمیم می‌گیریم
- logical: منطق را در جهتی که می‌خواهیم تغییر دهیم در جهت سیستم
- phisical: بعد از رسیدن به وضعیت مطلوب منطقی به سراغ حل مسئله می‌رویم / فیزیک می‌دهیم

Analaysis model ترکیبی از متن و فرم های دیاگرامی است که نیازهای مربوط به data، Function و behavior را شامل شده تشخیص آسانی باشد و به راحتی بتوان correctness، completeness و consistency را اعمال کرد. ص ۱۷۸

Subject: Year. Month. Date.

حال آنالیز و requirement specification یک راهی است که از نیازها ... [illegible] است و دقیق یک نرم افزار درست می شود.

اگر سیستم را بتوان به خوبی طراحی کرد می توان آن را به وضعیت مطلوب رساند مثل ATM که در logic آن ها دست عده ای باید کار کنند. در واقع logical را سمتی که می خواهیم بهبود دهیم بخش می شوند و سپس به وسیله تکنولوژی آن را توسعه می دهیم.

برای تحلیل از دو روش ۱- Structure Analysis ۲- object-oriented Analysis استفاده می کنیم که این در مرحله بعد بر اساس این ها این دو تحلیل جلو می رود. یک مهندس نرم افزار از object oriented استفاده می کند ولی در بازار صنایع از structure Analysis استفاده می شود.

قبل از شروع تحلیل به دو روش بالا مخصوصی به نام Feasibility study آشنا می شویم:

Feasibility study:

یک سری محدودیت هایی دارد که به 4 گروه تقسیم می شود.

۱- schedule: زمان بندی

۲- cost: هزینه

۳- technology: با توجه به امکانات و یک بستر از تکنولوژی می خواهیم استفاده کنیم

۴- policy: سیاست گذاری، قوانین و آیین نامه ها

در این مطالعه با در نظر گرفتن constraint ها می خواهیم تشخیص دهیم آیا از نظر تکنولوژی Feasible هست یا نه. آیا می توان با تکنولوژی این کار را انجام داد ←

1) tecknology Feasible

2) operation Feasible

3) ecconomic Feasible → به اهداف اقتصادی می رسد یا نه

4) schedule Feasible

از جمله شاخص هایی که برای Feasibility باید در نظر گرفته شود عبارت است از:

- نگه داری ارزان 2

- آموزش کم 1

- تخصص ویژه نیاز نداشته باشد 3

- نیروی انسانی زیاد نشود. 4

PAPCO

انواع مدل ها عبارتند از: scenario-based M. .11 : سیستم را از نقطه نظر استفاده کننده ها و user نشان می دهد.

Subject:

Flow-oriented M. .12 : چگونه Data obj ها و بعد عملیات پردازش تغییر می کنند.

Year. Month. Date. ( )

Class-based M. .13 : obj ، Attr و روابط را تعریف می کنند

behavioral Model .14 : state های سیستم و class ها را مشخص کرده و تاثیر event ها بر روی آنان مشخص می شود

پس این شاخص ها اولویت بندی کرده و درجه اهمیت تعیین می کنیم. این شاخص ها از همان جلسه اول که می گوییم سیستم چون به عنوان مشخص می شود ممکن است در تناقض باشد.

پس برای هر طرح بیان شده به شاخص ها نمره می دهیم و در نهایت از نمره های هر طرح میانگین می گیریم و طرح بهتر انتخاب می شود ولی تصمیم نهایی با مشتری می باشد و اگر چیز دیگری را انتخاب کرد مسئولیت خودش است.

شاخص ها	اولویت	طرح ۱	طرح ۲
	3	۱۸	14
	4	14	12
	5	:	:
میانگین	.	14	12

درواقع در Feasible طرح بهتر مشخص می شود. یعنی شاخص ها functionality دارند و عوامل جانبی برای آنها وجود دارد که این عوامل نیز در تعیین اولویت موثر است و ضریب گذاری می شود که به آن task selector number می گویند.

- چند تا فرم خروجی ← دو فرم خروجی با یکدیگر برابر نیستند

- فرم ورودی

Structure Analysis :

← Data و process را به عنوان موجودیت های جدا از هم برای تعمیر در نظر می گیرد.

Data object به صورتی مدل می شود که می تواند جزئی و روابط را نشان دهد - process ها نشان می دهد چگونه Data به عنوان یک obj Data

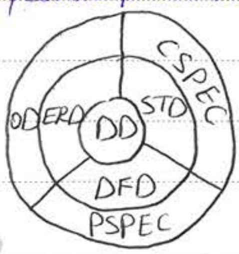
یک سیستم از سه عنصر تشکیل شده است: ۱- Data از طریق سیستم تغییر می کند (دیکشنری DD)

۲- عملیات: اعمالی که به کار می روند

۳- رفتار سیستم: حالاتی که یک سیستم می تواند از آن نتیجه بشود.

elicitation

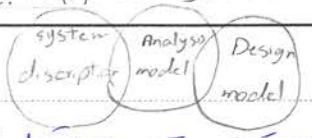
الگوی وجود دارد که مشخص می کند اگر اطلاعات به صورت ارزش های مطرح شده در inception از آنها منظور برای اینکه اطلاعات درست و کاربردی باشد باید ۷ عنصر را رعایت کنید:



PAPCO

56

Subject:
Year. Month. Date.



۱- Data dictionary:

یک عنصر مهم است ولی شاخصی برای جمع آوری اطلاعات نیست. برای ایجاد برقرار کردن یادآوری باید با ادبیات او صحبت کنیم. Data dictionary ادبیات خود را برای ما تعریف می‌کند که برای هر چیز چه واژه‌ای باید به کار رود. این DD به صورت یک کتابچه وجود دارد. هرلغتی که به کار می‌رود داده کرده و توصیف آن را می‌نویسیم.

۲- ERD (entity relational diagram):

داده‌هایی که در یک برنامه نمایی با آن سروکار داریم به سه دسته تقسیم می‌شوند:

۱- داده‌هایی که از صفحه کلید وارد برنامه می‌شود.

۲- داده‌های کنترلی که خود آن را اضافه می‌کنیم تا عملیات به درستی انجام شود و این درستی را چک کنیم.

۳- متغیرهای داخلی برای رویه‌های برنامه از آن استفاده می‌کنیم.

از جمله عوامل موثر در ERD داده‌هایی است که بخشی از آن را وارد می‌کند اسامی که در document و سیستم به وضوح مشخص است به عنوان entity تعریف می‌شود هریک از این entity ها یک سری ویژگی دارند که در obj discriptor مشخص می‌شود. در DD نظم وجود ندارد و سطحی عام‌تر از OD است.

۳- DFD (data flow diagram):

عملیات در این قسمت تعریف می‌شوند که داده‌های کنترلی را اینجا خودنمایی می‌کنند. چگونگی انجام این عملیات در PSPEC تعریف می‌شود.

۴- STD (state transition diagram):

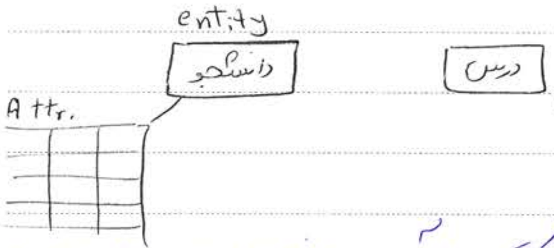
فقط مهندسی نرم‌افزار با این مقوله آشنا هستند که ویژگی‌ها و چگونگی state, transition در CSPEC تعریف می‌شود.

Subject:
Year. Month. Date. ( )

: Data model - ERD

منظور از data model نمایش داده‌ها و روابط بین آنهاست. همهٔ کلمه‌ها و داده‌های موجود در یک سیستم به عنوان صورت داده و Attribute وجود دارد. مشخص‌کردن عناصر اصلی اهمیت دارد تا بتوان زمان ... را از یک Attr. تشخیص داد که این کار از روی همبستگی عناصر قابل تشخیص است. به عنوان مثال در برنامه‌نویسی ماژول‌های متفاوتی تعریف می‌شود که بعضی از آنها به یکدیگر وصل و مرتبط هستند. مثلاً یک دانشجو شامل چندین صفت است که بعضی از آنها در عمل ثبت نام مورد استفاده قرار می‌گیرد و بعضی به کار نمی‌آید.
تعریف یک موجودیت با تمام ویژگی‌هایش یک obj را به وجود می‌آورد. ولی تعریف یک موجودیت با ویژگی‌های مورد استفاده در یک کاربرد یک aspect به وجود می‌آورد.
در obj براساس دیدگاه واقعی ویژگی‌ها انتخاب می‌شود و کاری با ویژگی‌های کاربردی و عملیاتی ندارد. پس دیدگاه در انتخاب ویژگی‌ها تأثیر می‌گذارد.

عناصر مدل عبارت‌اند از:



در Data object فقط داده نگهداشته می‌شود و operation هایی که بر روی Data اعمال می‌شود دارد نمی‌شود.

۱- entity می‌تواند افراد، محل‌ها، obj (کتاب، فرآورده، لوازم منزل)، event (رویداد) یا مفاهیم ذهنی که پدیدهٔ واقعی نیستند (اعتبار، حساب بانکی، ...) باشد.
یک entity مثل اسم عام است که اگر به صورت خاص برای یک چیز به کار رود یک instance ایجاد می‌شود. هر entity شامل یک سری Attr. است.

۲- Attr. ها می‌توانند مرکب یا غیرمرکب باشند (ویژگی‌هایی که همیشه همراه هم به کار می‌روند به ارزش متفاوت کنار هم قرار می‌گیرد و یک واحد را نشان می‌دهد). اگر عملی وجود داشته باشد که روی یکی از اجزای خاص صفت مرکب سؤال کند باید صفت مرکب را به ساده تبدیل کنیم.
در این مرحله باید data type صفت‌ها تعیین شود و سپس domain آن مشخص شود. آیا default value دارد یا نه؟

PAPCO

Subject:
Year. Month. Date. ( )

Attr.	DT	Domain	Default value

: جدول بالا را عملیات اشیاء object discriptor می‌گویند. به صورت syntax یک موجودیت به همراه ویژگی‌هایش به صورت زیر تعریف می‌شود:

دانشجو = شماره دانشجویی ⊕ جنسیت + {دروس‌های گذرانده شده} + (optional Attr.)
[مرد، زن]

یعنی فقط یکی از این دو انتخاب می‌شود. — تکرار شونده و چندمقداره — ممکن است برای بعضی از رکوردها مقدار داشته باشد و بعضی نه

نکته: در Data dictionary می‌توان از هر اسمی استفاده کرد مانند "شماره دانشجویی" ولی در obj disc سعی می‌کنیم اسم را به سمت برنامه‌نویسی و اسم منفرد (STN) به کار ببریم و برای استفاده از یک صفت در قسمت‌های بعدی از همین نام موجود در obj discriptor استفاده می‌کنیم. اگر در مراحل بعدی به یک Attr. احتیاج داشته باشیم و در OD تعریف نشده باشد نشان می‌دهد که entity ناقص است و این خود شاخصی برای تولید است. هر چیزی که در DD است در OD وجود ندارد ولی هر چه در OD باشد در DD نیز موجود است. در واقع حالت formal و سیستماتیک یعنی اقلام DD در OD به کار می‌رود و تعریف می‌شود.

entity چیزی است که ذخیره می‌شود و در صورت نیاز بازیابی می‌کنیم. در structure Analysis برای داده‌هایی مناسب است که Database ی وجود دارد و اطلاعات قرار است ذخیره شود.

هدف از این ساختار برای تحلیل این است که اطلاعاتی وجود نداشته باشد و چیزی miss نشود. اینجا برای ارتباط با مشتری است برای همین مشتری باید صفت‌های مورد نیاز خود را در آن ببیند. در مراحل بعدی این اشیاء را ولی تبدیل می‌کنیم که برنامه‌نویس بفهمد.

PAPCO

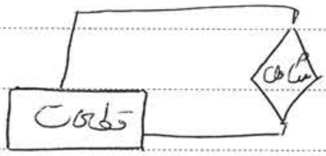
رابطه نشان می دهد چگونه data object ها به یکدیگر متصل شده اند.

3- بعد از آن مهم است که بدانیم کدام entity ها با هم رابطه دارند. هر entity می تواند رابطه ای با دیگر entity ها داشته باشد.

رابطه چند به چند از درجه ۲

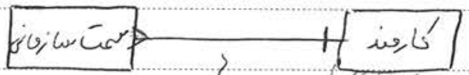
entity با obj در برنامه نویسی متفاوت است. درجه یک رابطه عبارت است از entity هایی که در رابطه شرکت دارند.

نکته: در دیاگرام زیر این رابطه درست نیست و معمولاً به کار نمی رود. به عنوان مثال چندین cpu وجود دارد و یک PC که بالا نیز وجود دارد است و cpu های مربوط به PC را نمی شود در واقع Atomic نیست.



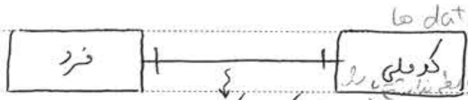
سپس چندی روابط را بین آنها مشخص می کنیم که به صورت زیر است: (cardinality)

تعداد entity هایی که با یکدیگر در ارتباط هستند را نشان می دهد (چندی)



یک فرد چندین سمت سازمانی داشته باشد

modality ← مشخص می کند که آیا data obj ها در روابط شرکت می کنند ← می توانند که در رابطه اصلاً شرکت نکنند یا همه رکوردها باید در رابطه شرکت کنند.



یک به یک

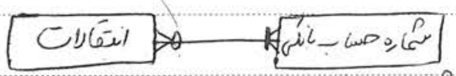
سپس modality را تعیین می کنیم که آیا باید حتماً رابطه داشته باشد یا خیر (total / patial)

اولین مرحله در Datamodel، content model است که کشیدن این entity ها و روابط آنهاست. مرحله بعد تعیین کلیدهاست. هر entity باید یک شناسه منحصر به فرد داشته باشد که همان primary key است.

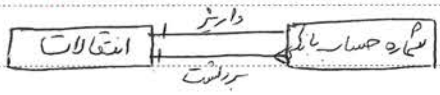
Subject:
Year. Month. Date. ( )

چندین راه برای ازبین بردن رابطه چند به چند وجود دارد.

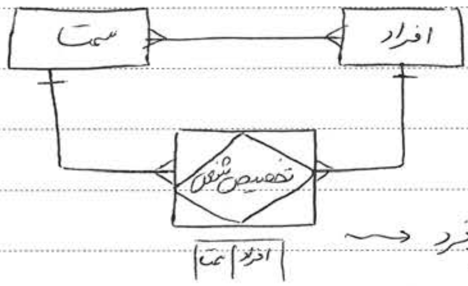
۱- می‌توان با خرد کردن و تقسیم روابط، آنها را به یک به یک تبدیل کنیم



یک انتقال با چند حساب بانکی می‌تواند در رابطه باشد می‌تواند صفر باشد ← حسابی باشد که انتقالی نداشته باشد ولی برای هر انتقال حتماً یک شماره حساب وجود دارد.



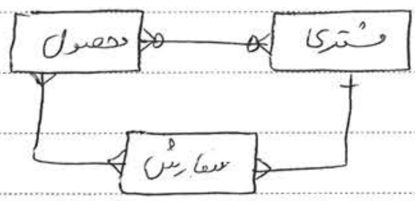
۲- خود یک entity جدید قرار می‌دهیم به چیزی نیست که مشتری ببیند و سعی می‌کنیم از یک کلید اصلی درون دانش خود در entity جدید قرار دهیم و یک رابطه یک به چند ایجاد می‌شود



هر فرد چند رکورد دارد ولی هر رکورد خاص یک فرد است

۳- در سیستم دوم وارد یک entity اضافه می‌کنیم که سیستم قرار داده و با مشتری در ارتباط است

برای سفارش



کلید اصلی خاص خود را دارد

در این حالت رابطه چند به چند و یک رابطه یک به چند ایجاد می‌شود که رابطه چند به چند به روش دوم ازبین برده می‌شود.

conten model → key model → Attr. model → normal model

کلید خارجی بیرون کشیده می‌شود اجزای غیر کلیدی نباید با اجزای غیر کلید دیگری در ارتباط باشد.

Subject:
Year. Month. Date. ( )

نرمال سازی (NF):

سطوح مختلف نرمال سازی به صورت زیر است که ویژگی سطوح بالاتر در سطوح پایین تر وجود دارد.

1) صفت تکرار شونده نداشته باشیم.

2) هم ویژگی های اولی را دارد و هم اینکه صفت های غیرکلید اصلی رابطه ای با کلید اصلی همان وجود نداشته باشد.

3) هیچ رابطه ای بین صفات غیرکلیدی نباید وجود داشته باشد.

Data model اولین مدلی است که برای structure Analysis به سراغ آن می رویم.

Data model:

از ویژگی های این مدل عبارت است از:

1- vocabulary مشتری را می فهمیم

2- خیلی زود ساخته می شود

3- هیچ ابهامی وجود ندارد (قوانین دانشگاه ها مربوط policy سیستم است) ولی Data model به خاطر ثابت بودن سریع ایجاد می شود.

4- بروزی sheet قابل نمایش است.

5- خیلی بیشتر شبیه به دنیای واقعی هستند که این بیشتر یک مزیت است.

PAPCO

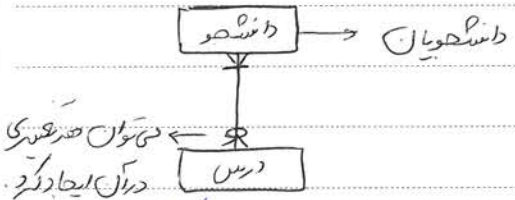
Subject:
Year. Month. Date. ( )

یک Data model خوب دارای ویژگی های زیر است :

1) داده ای که لازم داریم فقط در آن دیده می شود و داده بدون نیاز وجود دارد. افزونگی در داده های موجود وجود نداشته باشد منظور از افزونگی اتریبیوت زیادی است نه اتریبیوتی چون در یک فیلد در یک پایگاه داده اگر بخواهیم تا کمتر کار را تغییر دهیم باید یک یک رکوردها را عوض کنیم ولی می توانیم سمت ها را کلیدواری کنیم و برای هر رکورد کدآن را بدهیم در این حالت فشردگی کم شده و جدول کلی را نمی بینند.

رسیدن به استقلال داده ای و integrity با افزونگی در دیتا جنبه متناقض دارد برای همین این ویژگی ها را در دیتا فشردگی می کنیم.

2) انعطاف پذیر باشد : بتوان تغییرات را بر روی آن به راحتی اعمال کرد. مثلاً اگر یک Data model بر روی یک پوشه بزرگ باشد چون نمی توان تغییرات را بر روی آن اعمال کرد انعطاف پذیر نیست.



3) قابل تعمیم برای آینده : به این معناست که یک موجودیت های خالی با attr. اضافه می کنیم تا بتوان در آینده تغییرات را اعمال کرد. یعنی به این معناست که ابزارهایی را استفاده می کنیم که در آینده بتوان تغییراتی را بر روی آن اعمال کرد.

Data to Location :

می تواند به صورت یک جدول باشد که ارتباط هر Data با آن در ربط مشخص می شود که در هر مکان چه کاری می تواند و چه فعالیتی بر روی داده ها انجام می دهد.

Data	دانشگاه	
دانشجو	R	C R
Attr.		

Subject:
Year. Month. Date. ( )

از جمله کارهایی که می‌توان کرد روی داده‌ها در جدول‌های گوناگون انجام داد عبارت اند از:

CRUD → delete
read
create
update

یک query که ایجاد می‌شود می‌تواند نتیجه را به صورت یکی از اشکال زیر نمایش دهد:

SS → subset

SR → Single record

All → همه

که بر اساس ماتریس می‌توان تعیین کرد که چه چیزهایی نمایش داده شود.
در صورتی که در یک سازمان تشکیلات مختلفی وجود داشته باشد برای هر موجودیت درون تشکیلات جدول را آماده می‌کنیم و بعد روابط بین موجودیت‌ها را تعیین می‌کنیم.

process model:

دنیای مدل را باید با activity ها شناخت که مربوط به این لایه است. بخش‌های منطقی برای انجام یک عمل مدل منطقی را ایجاد می‌کنند ولی بخش‌های logic که در مورد شیوه عملکرد در سطوح بالاتر صحبت می‌کند با Data سر و کار دارد و کاری با منابع فیزیکی نداریم.
ویژگی‌های منحصر به فرد هر سیستم و چگونگی پیاده‌سازی آن در این مرحله انجام می‌شود.
عناصر مختلف یک process model عبارت اند از:

yordon SSMDN

process نمایش ←

۱- process
۲- external entity
۳- Data set
۴- Data flow

شیوه‌های مختلفی برای نمایش این اجزا وجود دارد. که در این درس از yordon استفاده می‌کنیم.

PAPCO

level 0 DFD : سیستم را به عنوان یک bubble در نظر می‌گیریم و باید ورودی و خروجی های اصلی آن مشخص شود.

Subject: Year. Month. Date.

level 1 DFD : process های مختلف برای آن را جدا کرده و تفصیلات را انجام می‌دهیم همه فلش ها و دو اسم باید با آن ارتباط داشته باشند.

تقسیم بندی ها را با جای اسم و فعل انجام می‌دهیم به هر کدام bubble یک function ساده را نشان می‌دهد
bubble ← verb
flow ← noun

تفاوت process Model و فلوچارت ؟

۱- همه فرایند‌ها می‌توانند با هم به اجرا در بیایند ولی در فلوچارت هر لحظه یک Data اجرا می‌شود

۲- در processM انتقال داده‌ای است ولی در فلوچارت انتقال کنترلی است (control flow) یعنی داده جابجا نمی‌شود فقط می‌گوییم چه کاری باید انجام شود.

۳- در فلوچارت یک سری شاخص ها را نمی‌توان انجام داد یا action های undemand قابل نمایش نیستند منظور از undemand این است که وقتی در یک state هستیم متناسب با شرایط موجود همزمان به state های دیگری برود درحالی که در فلوچارت چنین امکانی وجود ندارد و در هر حالت صبر می‌کند تا ورودی بگیرد به state بعدی برود

I) یک process چیست؟

می‌توان کل سیستم و عملیاتی که می‌خواهد انجام شود را به عنوان یک process تعریف کنیم. که در این حالت به دیاگرام حاصل content diagram می‌گوییم (DFD)
↓
data flow

در این حالت کاری نداریم که سیستم در جزئیات آن چیست ولی در مورد اسم دیم وصطح و چه کسانی در ارتباط است و ورودی خروجی کلی و feedback ها قابل نمایش است. در واقع نمایش به صورت کلی انجام می‌شود که به آن software scope می‌گویند

یک process می‌تواند یکی از اجزای زیر باشد:

۱- محاسباتی
۲- تصمیم گیری (main decision)
۳- sort / filter یا هر خلاصه سازی
۴- manipulate داده‌ها مثل گزارش یا اطلاعات آماری نوشته برای حالت خاص است

PAPCO

۵- می‌تواند سیستم یک process، trigger دیگری دارد
۶- برای استفاده از Database ها باشد

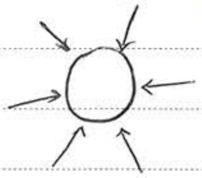
هر process می‌تواند یک bussiness event باشد که باید به صورت کامل انجام شود. نمی‌توان وسط آن متوقفش کرد. مثلاً در ATM در وسط پول گرفتن نمی‌توان آن را قطع کرد و اگر پول نداشته باشد یا به شبکه وصل نباشد به این معنا نیست که فرایند قطع شده بلکه خود event جدایی است.

* pass وارد می‌شود ← پول ندارد ← transaction

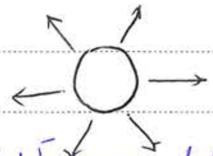
* pass وارد می‌شود ← قیمت ← به شبکه وصل است ← پول دارد ← پول را می‌گیرد

هر یک از این اتفاق‌ها یک transaction است. شباهت اینها: می‌توانیم همه این‌ها را در یک فرایند نوشت و فقط شرط را تغییر داد. این معنی نیست که همه آنها یکی هستند.

هر process یک سری ورودی و خروجی دارد که باید سه ویژگی زیر را نداشته باشد:
۱۱ فرایندها نمی‌توانند black hole باشند یعنی فقط ورودی داشته باشد و خروجی نداشته باشد.



۱۲ معجزه نمی‌توانند بکنند (No miracle): نمی‌تواند فقط خروجی داشته باشد.



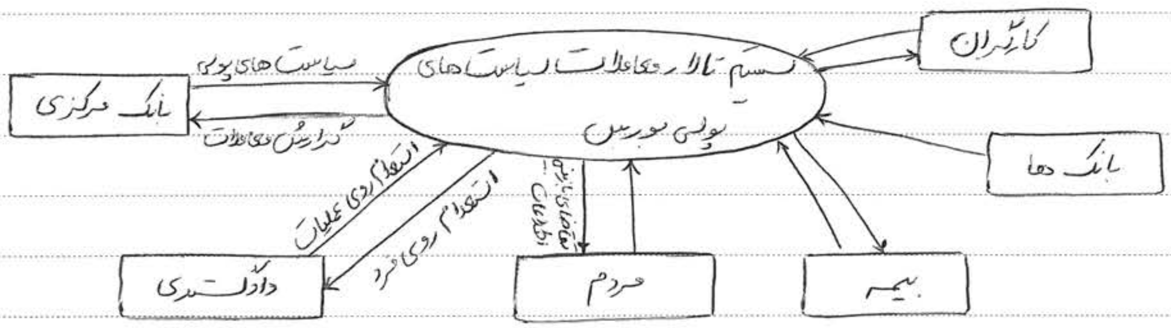
۱۳ نباید gray hole باشد که همان قانون بقای اطلاعات است همه خروجی‌ها باید از ورودی‌ها قابل تحصیل باشد. نمی‌توان فرض کرد داده‌ای در دل process‌ها هست بلکه process‌ها فقط یک فعالیت خاص انجام می‌دهد و داده‌ها باید به عنوان ورودی تعیین شود.

PSPEC: برای نمایش اینکه یک فرایند چه کاری انجام می‌دهد و چگونه باید specification برای آن بنویسیم چون مشتری باید آن را بفهمد از یک زبان قابل فهم استفاده می‌کنیم به علت مشکلاتی که زبان طبیعی دارد (تعمیم کرده‌اند) یک structure english ایجاد کردند که PSPEC را تشکیل می‌دهد. مانند برنامه ساختمان داده دارای حلقه‌های تودرتو if و ... است (PDL)

ویژگی های PSPEC عبارت است از:

۱. فقط از جملات امری و مشخص استفاده می‌کنیم
۲. فقط از اسامی می‌توان استفاده کرد که در Data dic وجود داشته باشد
۳. فرمول ریاضی باشد.
۴. صفت های مهم در Data dic باید تعریف شود
۵. دستورات به صورت black box و گروه‌بندی باشد
۶. پایان دستورات مشخص باشد.
۷. اگر در توضیح دادن دچار شک شدید باید نقطه نظر میل لذا به سمت خوانایی برای مشتری نه برنامه‌نویس‌ها

(II) external entity: در یک مستطیل نشان می‌دهیم: شخص، سازمان یا هر چیز خارج از محصول که با سیستم می‌خواهد در ارتباط باشد.
افرادی که اطلاعات خود را به سیستم را تولید می‌کنند یا از اطلاعات سیستم استفاده می‌کنند



اگر external entity نباشد سیستم هویت ندارد

: Data Flow

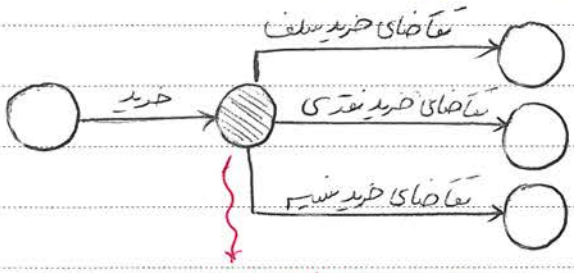
جریان داده‌ای را نشان می‌دهد که با یک فلش مشخص می‌شود:

داده باید روی فلش نوشته شود → user می‌بیند و می‌فهمد
داده هایی است که در Data model وجود دارد

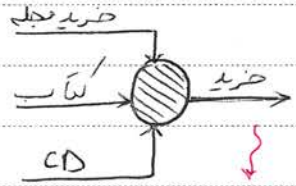
Subject :
Year . Month . Date . ( )

اگر از Data هایی استفاده کنیم که وجود ندارد (Data های انتزاعی) از علامت زیر استفاده می کنیم :

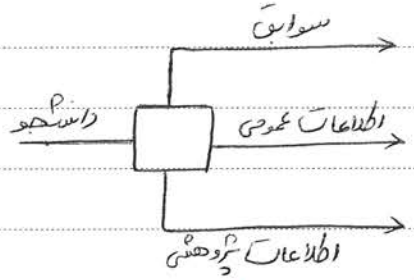
باید حداقل یک سرویس process باشد و راجع نمی تواند بین دو external entity باشد



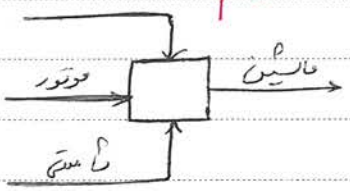
conjunction → وقتی به اینجا می رسد فقط یکی از اینها اجرا می شود
یا ارتباطی



می خواهد در حالتی که ... نوعش مهم نیست.



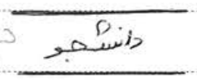
همزمان باید رخ دهند



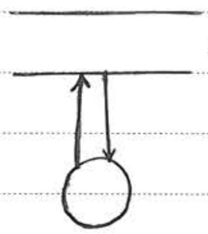
Subject:
Year. Month. Date. ( )

Data Set :

نماز نمایش آن به صورت زیر است که entity های Data model هستند که process ها فقط می‌توانند با آن سروکار داشته باشند.



در شکل زیر اگر یک طرف Data set باشد می‌تواند چیزی بر روی آن نوشته شود.



اگر روی Dataset چیزی نوشته شود یعنی تمام قالب ← رکورد را می‌خواند و می‌نویسد.

تولید process model :

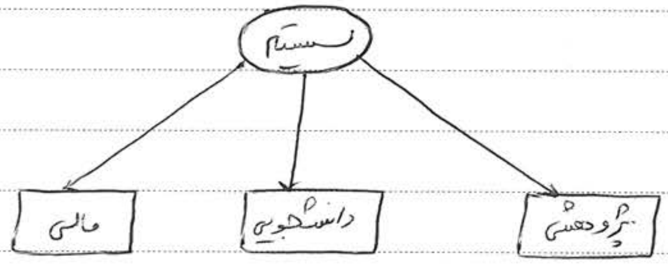
برای تولید یک process model مراحل زیر را باید انجام داد:

۱- content diagram :

ابتدا باید یک content diagram ایجاد شود در این دیاگرام نمی‌توان Data set داشت چون Data set ها داخل خود سیستم یا external entity است.

۲- function decomposition :

از اینجا به بعد تجزیه را شروع می‌کنیم و بر اساس function های مختلفی که یک سیستم می‌تواند داشته باشد تقسیم‌بندی می‌کنیم. برای سیستم هایی که کوچک هستند فقط یک App را انجام می‌دهد و این قسمت را ندارد.



Subject:
Year. Month. Date. ( )

۳- use case List :

برای یک سری رویدادها usecase را مشخص می‌کنیم

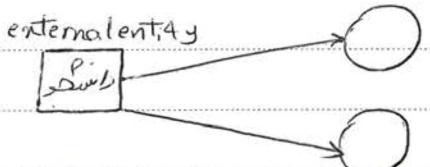
← temporal event ← زمان مشخص - در زمان یک واحد زمانی چیزی را انجام می‌شود

← state event ← شرایط و ویژگی‌های درونی خاصی بوجود می‌آید

در این مرحله event های usecase ها را در می‌آوریم.

۴- برای هر function ، event ها را مشخص می‌کنیم

۵- برای هر usecase یک content diagram می‌کشیم یک سری ورودی دارد و در این مرحله می‌توانند Dataset باشند



۶- به یکدیگر مرتبط می‌کنیم و سیستم را یکپارچه می‌کنیم

۷- هر usecase را یک سری task های کوچکتر تقسیم می‌کنیم و تا جایی پیش می‌رویم که pspec آنها مشخص و واضح باشد و تا در سه level کافی است

8 Component Diagram

نشان دهنده سرویس های مستقل ماست. component کوچکترین واحد reuse است.

service : مجموعه‌ای از component ها که یک task را انجام می‌دهند

برای توزیع کار برای تیم و گروه‌های اجرایی مناسب است

PAPCO

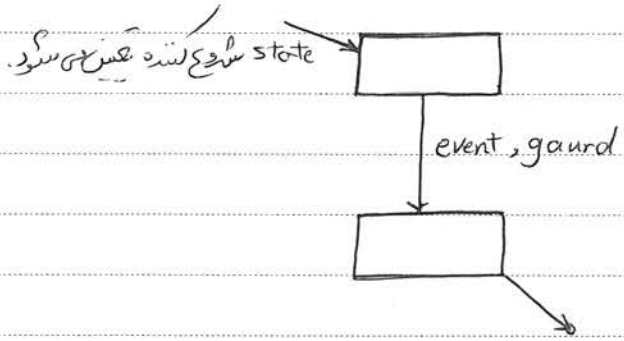
Subject:
Year. Month. Date. ( )

State machine diagram :

هرکلاسی که در مدل تحلیل object oriented در نظر می‌گیریم سه بعد دارد :

۱- بعد Data یا کلاس های دیتا مانند ERD

۲- هرکلاس پس از اینکه یک متد آن اجرا می‌شود تغییر حالت می‌دهد حالت‌های آنها state machin diagram انجام می‌دهد. state مشخص می‌کند کدام حالت هستیم و با چه شرطی به حالت دیگری برود. هرکلاس یک initial state دارد ولی می‌تواند final state های متعددی داشته باشد.



انواع کلاس ها :

۱- passive object : یک کلاس اگری ساخت obj است و فقط در obj passive داریم. instance ها را آن ساخته می‌شود. در حافظه هستند تا پیامی به آنها داده شود و هر وقت به آن احتیاج داریم متدی از آن را اجرا می‌کنیم.

۲- active object : هر obj فعال است (رده بندی) classification دارد و در واقع یک component را ایجاد می‌کند خودش متدش را اجرا می‌کند و اگر قرار است تغییر حالت دهد پیامی به آن ارسال می‌شود. مگر نیست پیام واقعاً ارسال شود و منتظر جواب نمی‌ماند ولی در passive فرق می‌کند تا جواب پیام را دریافت کند.

۳- slot_base object : یک متد دارند و هر متد با توجه به obj هایی که به آن کلاس هستند می‌تواند فرق داشته باشد. مثلاً عملاً خوردن متناسب با نوع خوردن متفاوت است که حالت خوردن درون خود obj تعریف شده ولی در passive می‌توان از طریق پیام متد را ایجاد کنیم ولی با پارامتر خوردنی از خارج تعیین می‌کنیم این روش در کتاب به channel programming می‌رود.

P4PCO

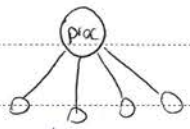
Subject :
Year . Month . Date . ( )

۴- Functional object :

برای هر Function یک کلاس ایجاد می‌کند فقط یک CLOS ایجاد شد برای اینکه بتوانند برای Function ها الگوی ساخت ایجاد کنند.

process model :

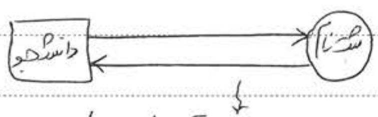
1) content diagram
2) Function decomposition
3) use case list —— a) throughput: یکی از بزرگترین معضلات نظام بانکی این است که در هر ثانیه throughput 700 PS را باید بتواند handle کند. b) Responce time
4) categorized in Functions → می‌توانیم use case ها را بین Function های مختلف تقسیم کنیم
5) content diagram For event → Dataset ها از این مرحله نمایش داده می‌شود.
6) system diagram → content diagram موجود برای هر event رجوع کرده و مرتبط با یکدیگر نمایش داده می‌شود.
7) event → task ها طرح می‌کنند.



قانون: 1) هر process باید بیش از یک زیر process تقسیم شود
2) هر process نمی‌تواند فرزند یک proc باشد ولی در بعضی از سیستم ها هر proc می‌تواند فرزند سیستم و proc های مختلف باشد.

8) pspec و cspec → مثل pspec فقط مباحث کنترلی را لحاظ می‌کنیم
بین proc هایی که ما اضافه می‌کنیم و user نمی‌بیند تفکیک شده است و این رو جدا می‌شوند

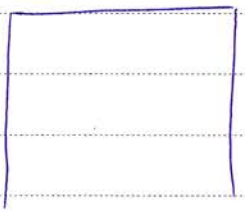
9) Data structure → به ازای هر ورودی ها یک فلش قرار می‌دهیم. تمام اطلاعات را روی یک فلش قرار داده و نتیجه بین آن دو یک ورودی و یک خروجی وجود



برای هر قلم اطلاعاتی که می‌فرستند فلش قرار می‌دهیم
قلم ها را جدا جدا قرار داده و روی هر فلش فقط یک قلم اطلاعاتی قرار می‌دهیم

PAPCO

Subject:
Year. Month. Date. ( )



```
begin                                   ۰
  readln (N);                           ۱
  I:=1;                                 ۲
  P:=1;                                 ۳
  while (I<N)                           ۴
    begin                               ۵
      P:=P*I                            ۶
      I:=I+1                            ۷
    end
  writeln(P);
```

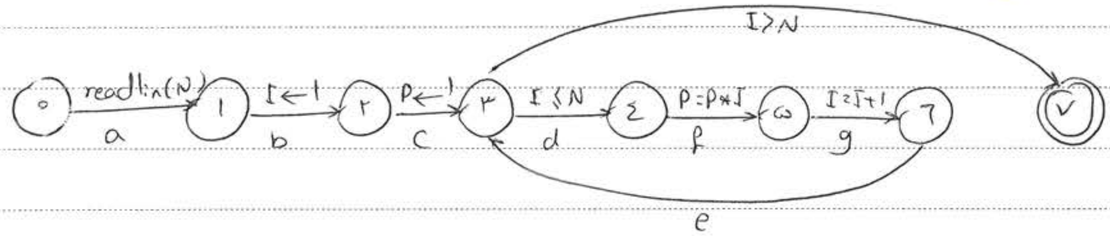
Subject:
Year. Month. Date. ( )

برای تبدیل این برنامه به یک گراف، یک سری پدیده های دقیق را رأس در نظر گرفته و یال ها هر عملی هستند که آن را تغییر می دهد

No operation ← program counter را تغییر می دهد

چک کردن یک شرط ← flag ها را تغییر می دهد.

در نتیجه با نقش symbolic همه پدیده ها می توان یک حالت کلی بدست آورد.



یک مسیر گراف که از نود اولیه شروع می شود یک run است. مسیری که به رأس حالت نهایی برساند جواب است

در نتیجه از روی گراف می توان unreached ها را تشخیص داد

دو پدیده در گراف در برنامه تعریف می شود:

1) Live Lock:

اگر یک حلقه را چندین بار طی کند و هیچ وقت از روی آن خارج نشود چون state عوض می شود پس زنده است.

2) dead lock:

اگر وارد state شود که نتوانیم از آن خارج شویم

از روی گراف می توان deadlock و livelock ها را تشخیص داد. هر رویدادی که اتفاق می افتد می تواند یک condition نیز بر روی آن وجود داشته باشد. موفقیت یک transaction به پایان رسیدن آن است. به پارامترهای این مشخصات یک state transition diagram ایجاد می کند و از روی آن علاوه بر کنترل اتفاق ها و transaction های مختلف می توان به کد مربوط رسید. این std را می توان به صورت یک Automata نشان داد:

$A(Q, Names, \longrightarrow, Q_0, F)$

- Q: State
- Names: عمل ها (a, b, c)
- $\longrightarrow$: transition — $Q \times A \times g$ ← شرط
- Q_0: حالت اولیه
- F: حالت نهایی

PAPCO

Subject:
Year. Month. Date. ( )

$A_1(Q_1, N_1, \rightarrow_1, Q_{01}, F_1)$

$A_2(Q_2, N_2, \rightarrow_2, Q_{02}, F_2)$

رفتار دو تا رفتار باید بتوانند ترکیب شوند به مرحله‌ای برسند به این نتیجه برسیم که رسیدن به رضایت قسمتی غیر ممکن است.

$A_1 \bowtie A_2 = (Q, N, \rightarrow, Q_0, F)$

$Q = Q_1 \times Q_2 \rightarrow$ خروجی یک دوتایی است

$N = N_1 \cup N_2$

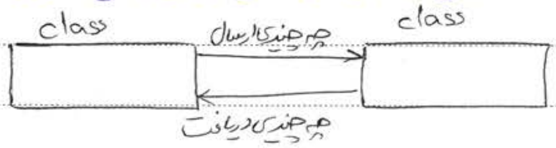
$\rightarrow : P_1 \xrightarrow{A_1, g_1} q_1 \;,\; P_2 \xrightarrow{A_2, g_2} q_2 \;,\; n_1 \cap N_2 = n_2 \cap N_1$

$\Longrightarrow \langle P_1, P_2 \rangle \xrightarrow{n_1 \cup n_2,\; g_1 \wedge g_2} \langle q_1, q_2 \rangle$

برای این حالت تا قفل ایجاد نمی‌شود و قابل ترکیب نیستند می‌خواهیم نقص n در دیگری وجود نداشته باشد.

: Colaboration diagram

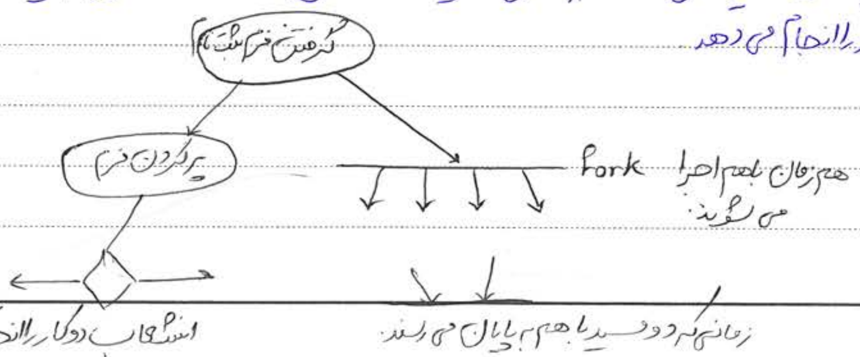
بعد سوم کلاس‌ها در colab. diagram نشان داده می‌شود. یک کلاس با یک کلاس دیگر چه ارتباطی پیامی می‌تواند داشته باشد.



scope برای هر کدام مشخص می‌شود. در نتیجه برای کارهای تیمی مناسب است.

: activity diagram

هر کلاس چه نقشی دارند چه نقش هایی دارند Functionality آن باید نشان داده شود که activity d. این کار را انجام می‌دهد

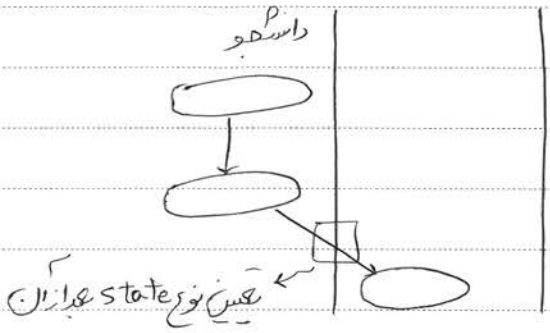


PAPCO

Subject:
Year. Month. Date. ( )

؟ swimlam diagram

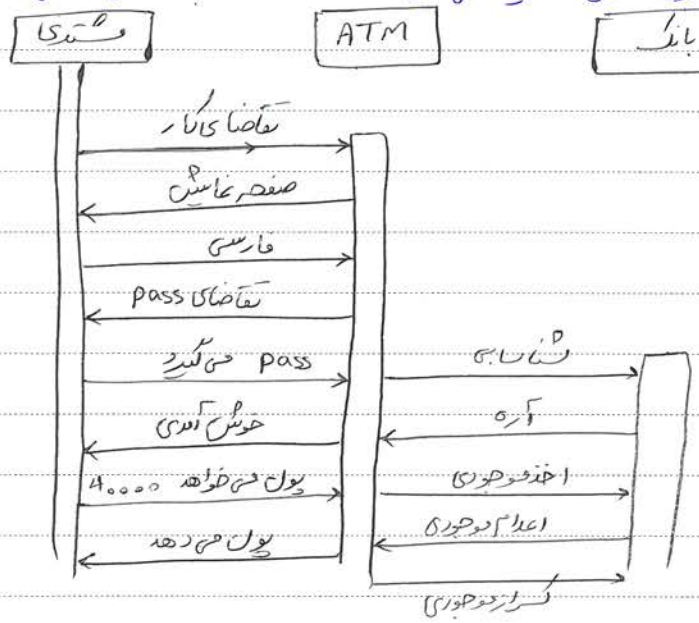
اگر برای هر activity مشخص شود که مربوط به کدام obj است و تغییر حالت در آن نشان داده شود یک swimlan diagram ایجاد می‌شود.



process های تعیین شده در activity diagram به صورت سری اجرا می‌شوند مگر اینکه fork و join باشد. در process model هر process ها جدای انجام می‌شوند.
state diagram قبل از activity diagram ایجاد می‌شود.

؟ Sequence diagram

یک task پایان نیز را با sequence d. نشان می‌دهیم. هر obj در هر لحظه دقیقاً مشخص می‌شود. حیات به صورت یک خط برای هر obj کشیده می‌شود. مثلاً ATM وجود دارد ولی زمانی اهمیت دارد که تقاضایی از آن داده شود.



Subject:
Year. Month. Date. ( )

این دیاگرام روابط یک عمل دوتوق را نشان می‌دهد برای اینکه مثلاً اگر لیستی باشد باید یک diagram دیگر بکشیم است در refactory می‌توان اینها را ترکیب کرد.

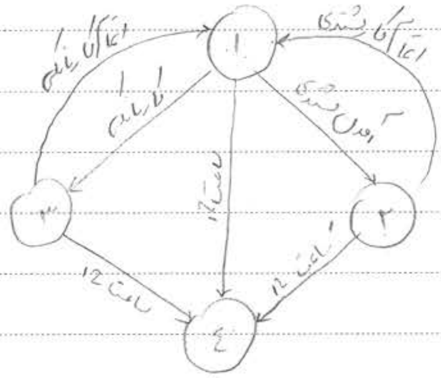
: CRC

یکی از فعالیت‌های جانبی OO تولید CRC (class Responsibility colaboration) است. هر کلاسی که مسئول می‌رسد را روی یک کارت می‌نویسند.
بعد از usecase هنوز کلاس‌ها تعیین نمی‌شود برای همین CRC تولید می‌کنیم تا برای مراحل بعدی از آن استفاده کنیم.

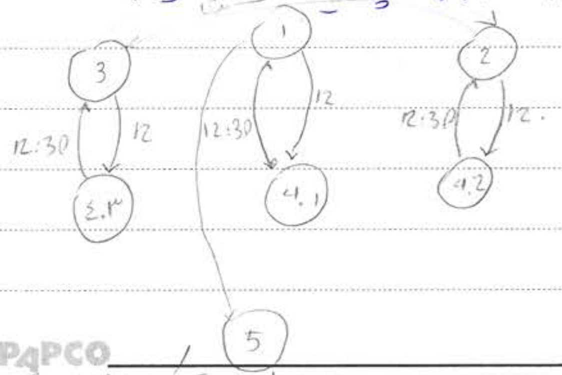
اسم کلاس	
مسئولیت	colaboration

: State diagram

ابتدا همه state ها را مشخص می‌کنیم:
1) آماده به کار
2) در حال سرویس دهی به مشتری
3) در حال سرویس دهی به خدمات بانکی
4) در حال استراحت
5) اتمام کار



از آن جایی که باید ساعت 12:30 به حالت کاری برگردد که قبلاً انجام می‌داد پس این state باید جدا شود.



Subject:
Year. Month. Date. ( )

اولویت در هر حالت با مشتری است پس اگر در وسط کار بانکی مشتری بیاید اول کار او را انجام می‌دهد. ۱، ۲، ۳

قسمت های پیچیده بعداً اضافه شود.

PAPCO

Subject:

Year. Month. Date. ( )

Subject:
Year. Month. Date. ( )

طراحی:

باید دقیق بوده و همش بگوییم اطلاعاتی که داریم کامل هست یا نه. هیچ چیزی نباید پیش خودمان در نظر بگیریم. در انجام کارهای مختلف با مدل شدن می‌توان سیستم کنترلی یکسانی را برای آنها نشان داد. مدل طراحی شده باید قابل فهم برای مشتری باشد. در نتیجه در مدل طراحی هدف این است که با مشتری در تماس باشیم اگر نقصی وجود دارد آن را کامل می‌کنیم. در واقع می‌خواهیم فعالیت مهم انبوهی از اطلاعات را به مدل تحلیلی تبدیل کنیم. برنامه‌نویس با اطلاعات کاری ندارد و از روی مدل این کار را انجام می‌دهد.

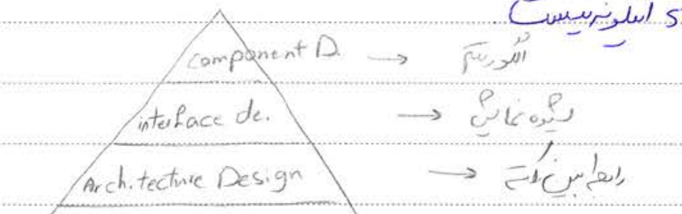
تعیین الگوریتم Data definition:

چگونه داده‌ها توصیف می‌شوند. در Data design عناصر برنامه باید تعریف بشوند. بعد از تعیین Data ها تکه‌های مسئله را مشخص می‌کنیم. سپس باید مشخص بشود رابطه بین تکه‌ها چگونه است (Architecture design) که به آن معماری می‌گویند. رابطه تکه در آن تعیین می‌شود. در معماری باید قطعات تعیین شود و سپس ارتباط بین تکه‌ها مشخص می‌شود.

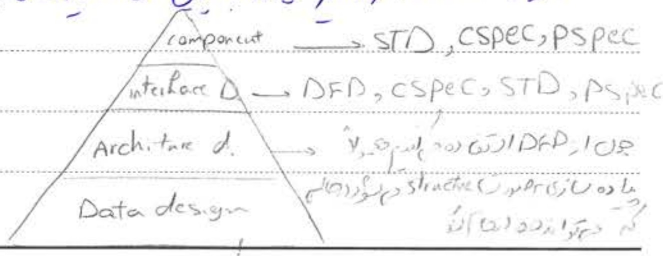
interface باید ابتدا شیوه نمایش برنامه تعیین شود سپس متناسب با آن الگوریتم‌ها مشخص می‌شود که الگوریتم‌ها در component design مشخص می‌شود.

بعد از انجام این کارها باید مسئله به صورت مدل سازی بیان شود تا برنامه‌نویس آن را بفهمد. در development یک مدل می‌سازیم که فعالیت‌ها بر روی آن انجام شود و برنامه را تولید کند.

در نزدیکی بین realworld و machine word باعث شده که در OO با تغییر جزئی بتوان از مدل قبلی دوباره استفاده کرد ولی در structureal اینگونه نیست.



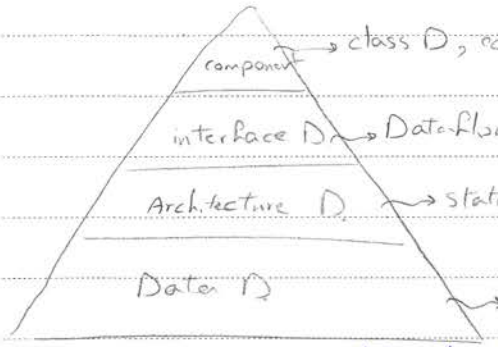
اگر در OO انجام دهیم نیازی به تعیین مدل نداریم ولی برای structure باید 4 جزء تعیین شود.



DD، OD و ERD از اینها برای ایجاد آن استفاده می‌کنیم

P4PCO

Subject:
Year. Month. Date. ( )



این مراحل جدای از هم هستند و فقط از اطلاعات استفاده می کنیم و فعل بلدیدینیستند ولی لایه ای نیستند در لایه ای یک مفهوم است و ماهیتش را عرض می کنیم.
دورویش تولید وجود دارد که با این اشوهخوانی ندارد یکی از آنها RAD است. زیرا کارهایی که در آن انجام می دهد از طریق manipulate کردن اطلاعات هستند یعنی داده هایی وجود دارد و با استفاده از ابزارهای دستکاری بروی این داده ها manipulate می کنیم. درنتیجه در همه پروژه هایی که از این روش استفاده می کند Architecture ثابت و یکسان است. Data design داریم و component هایی از قبل پیاده سازی شده.
یکی از روش ها prototyping است زیرا در این روش اصلا قسد مشخص نیست که بخواهیم معماری داشته باشیم فقط سریال تولید محصول با کیفیت خوب نیستیم بلکه سریال پیدا کردن یک راه حل مناسب است، داده ها را نمی شناسیم و یک interface وجود دارد و تنها توجه به interface است. از آن جایی که به لایه ها توجهی ندارد کیفیت خوب نیست.

طراحی باید سه ویژگی داشته باشد:
۱- وظایف طراحی پیاده سازی است نه مشتری ولی طراحی معماری، یکی از طراحی هاست که مشتری باید ببیند. و این تنها document از طراحی است که مشتری می بیند زیرا خیلی کلی تر از جزئیات است تا تصمیم های شما در مورد مدل و دل خود را بخوبی بفهمد.

۲- طراحی باید تمام خواسته ها را در بر بگیرد و نشان دهد.

۳- طراحی باید روی رفتار، Data و Function کار کند.
(behavior)

P4PCO

Subject:
Year. Month. Date. ( )

برای اینکه طراحی از کیفیت خوبی برخوردار باشد یک سری راهنمایی وجود دارد:

۱ـ باید ماژوله و پیمانه‌ای باشد. باید یک سری کلی تقسیم کنیم و این تکه‌ها وظایفشان با یکدیگر را نمایش دهیم.

۲ـ طراحی باید Data، معماری و واسط و Component را نشان دهد.

۳ـ از واحدهای شناخته شده استفاده کرد و از چیزهای موجود استفاده کنیم و چیزی را ابداع نکنیم. به دلیل اینکه صرف از یک روش شناخته شده استفاده می‌شود که همه آن را درک می‌کنند و نیازی به توضیح نیست.

۴ـ در طراحی حتماً باید فرمول‌هایی وجود داشته باشد که قابل درک و تکرار کردن باشد.

۵ـ ارتباط بین عناصر خیلی کم باشد. اگر دو ماژول خیلی با هم ارتباط داشته باشند ترکیب شده و ادغام می‌شوند. میزان ارتباطشان نشان می‌دهد که آیا باید ترکیب شوند یا نه.

FURPS الگویی برای صفت صنعتی طراحی است.

← functionality چه میزان functionality دارد.

← usability ـ فاکتورهای انسانی شکلی دارد، چقدر افراد متخصص دارند و چقدر نیاز به آموزش دارند، مستندات چگونه است.

← Reliability سیستم چه قدر وقت به چقدر وقت خراب می‌شود MTTF (main time to finish) چقدر طول می‌کشد که خرابی درست شود (MTTR) که اعتبار محصول را مشخص می‌کند.

← performance زمان پاسخ، سرعت سیستم، throughput، چه میزان پردازش و دیسک و... به کار می‌رود.

← supportability نگهداری سیستم چگونه است maintainable هست یا نه.

P4PCO

Subject:
Year. Month. Date. ( )

اصول طراحی:

۱- تکرار فعالیت انجام بدهیم. از چیزهایی که هست استفاده کنیم. برای انتخاب واحدها هزینه وجود دارد.

① COTS: (component of the shelve)

هر چیزی که در قفسه و در دسترس هست و می‌توان استفاده کرد. دیگران تولید کردند ست شده و قابل استفاده است (برای عدم نوشتن شده)

②

از محصولی که قبلاً خودمان تولید کردیم همان را با تغییرات جزئی مورد استفاده قرار دهیم (برای یک App خاص ولی خودمان نوشتیم)

③

یک نرم‌افزاری است که دیگران برای App خاص نوشته و ما استفاده می‌کنیم.

④ New product

یک محصول جدید تولید می‌شود

نمی‌توان گفت هزینه‌ها از بالا به پایین زیادتر می‌شود زیرا فعالیت component هایی از قبل تولید شده، هزینه‌ی زیادی دارند دقیقاً...

۲- حتماً از فعالیت تحلیل شروع کنید

۳- یک بدی نباشد. یک tunnel vision بعضی وقت‌ها بد نیست. اگر فعالیت از قبل‌ها می‌توانیم از روش‌های قبلی استفاده کنیم درست است ولی نباید به خاطر تخصص و مهارت خود، خود را به آن سمت ببریم.
باید ریسک محصول نهایی با محصول واقعی یکی باشد زیرا باید هر تغییری که در محصول واقعی صورت می‌گیرد در برنامه قابل اضافه کردن باشد.

PqPCO

Subject:
Year. Month. Date. ( )

۴- طراحی باید یکنواخت و منسجم باشد این طور به نظر بیاید که یک نفر انجام داده. این یکپارچگی از طریق دین بدست می‌آید که همه از یک base استفاده می‌کنند

۵- تغییرپذیر باشد ۱- مهم رفتن ۲- استفاده از ابزارهای خودکار و کنابینده

۶- انفجاری نباشد مرحله به مرحله طراحی و دستاوردها باید از مراحل قبل قابل استنتاج باشد

۷- کیفیت را در راستای طراحی ببینیم نه پس از آن. طراحی و کدنویسی با یکدیگر فرق دارند. توصیف باید در سطحی باشد که کدنویس بتواند از آن استفاده کند نه اینکه هر چند آن توضیح داده شود چون خودمان نمی‌خواهیم که بنویسیم و باید با توجه به تخصص کدنویس باشد

۸- باید بازنویسی شود

فصل نهم 8

Abstraction: پنهان کردن جزئیات و چگونگی‌ها. پدیده‌ای داریم و جزئیات را به گونه‌ای مطرح می‌کنیم که فقط خواسته‌ها از آن مشخص باشد

سه نوع abstraction وجود دارد:

1) Data Abstract: بسته‌هایی وجود دارد که تغییرات به وسیله آن مشخص می‌شود. مانند class که برای انجام یک کار، این که چگونه انجام می‌شود مهم نیست

2) Program Abstract: از یک برنامه می‌خواهیم استفاده کنیم کاری نداریم آن برنامه چطور انجام می‌دهد. فقط مهم این است که چه می‌کند

3) Control Abstract: از نظر پیاده‌سازی زبان با ① یکی است ولی از نظر طراحی سیستم و App متفاوت است. داده‌هایی که برای کنترل و حرکت کردن آن نوشته می‌شود و مشتری آن را نمی‌بیند ولی از نظر یک برنامه‌نویس این دو یکی هستند و فقط از دید مشتری فرق می‌کند.

PAPCO

Subject:
Year. Month. Date. ( )

encapsulation / Information hiding:

پنهان کردن جزئیات است. درون هر تابع یک سری متغیر وجود دارد که در حین استفاده از تابع نمی‌توانیم آن متغیرها را تغییر دهیم. در واقع نباید سیستم وابستگی وجود ندارد Domain مختلف را کاهش می‌دهیم تا تصمیم گیری در مورد چیزهای مهم باشد. private کردن و scope بندی برای این سرویس است.

Refinement (پالایش):

هر صورت مسئله که بتوانیم دستورات را به دستورات کوچکتر تبدیل کنیم. وقتی می‌خواهیم پالایش کنیم و یک دستور را به چندین دستور دیگر تبدیل کنیم چندین روش وجود دارد.

A(x, y) ← دقیق‌تر ← اندازه
F(N) ← دقت ← اندازه

باید حلقه‌ها را در برنامه مشخص کرد.

Refactoring:

فاکتورهای یکسان یا تکراری در جاهای مختلف داریم که در دستور به فاکتور گرفتن می‌کنیم

$ax+ay=2$ $a(x+y)=2$

refactoring و refining هم در طراحی و هم در پیاده‌سازی است.

Architecture:

هدف از معماری چهار ویژگی زیر است: 1) عناصر اصلی سیستم 2) ارتباط بین آنها 3) ارتباطات بین نوع‌ها 4) تقسیمات هم معماری نقش افراد مختلف در سیستم و محل‌های ارتباطی شان را نشان می‌دهد. Functionality سیستم را نیز توسط یک معماری می‌توان نشان داد و برای یک سیستم تنها یک معماری وجود ندارد.

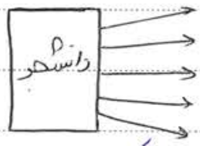
1) structure model: عناصر و ارتباط افراد را نشان می‌دهد
2) framework model: ویژگی‌ها و کارهای مشترک را نشان می‌دهد
3) process model: فرایند تجاری و مراحل انجام کار (task) را نشان می‌دهد برای هر task جدا یک pm رسم می‌شود

PAPCO

Subject:
Year. Month. Date. ( )

14 Dynamic model: رفتار سیستم event های خارجی رفتاری نشان می دهد حالت های مختلفی که یک دانشجو می تواند داشته باشد.

15 Functional model: عملکرد کلی سیستم را نشان می دهد. سلسله مراتب عملیاتی سیستم را نشان می دهد. بار وظایفی برای کار کردن functional را می است. در اینجا سلسله مراتب بین task ها را داریم.



تقدم و تاخر در اینجا مطرح می شود task های آن را نشان می دهد.

تبدیل این مدل ها به یکدیگر امکان پذیر نیست.

ویژگی های واحدبندی ۵

- composability: یکی از ویژگی های برنامه نویسی است. برای اینکه کارهای تکراری انجام ندهیم semantic مانع آن می شود ولی به syntax اهمیت می دهیم.

- Decomposition: از واحدبندی به این خاصیت می رسیم. روش های ذهنی و تحصیلی حاکم است هیچ قانونی برای آن وجود ندارد. Decomposition، reusability را تقویت می کند. عمل های قبل join و ضرب برای composition تعریف شده اند و می توان از عملگرهایی قبل تقسیم برای Decomposition استفاده کرد.

- understandability: قابلیت درک ما را بالا می برد.

- protection: اگر بتوانیم خطا را کنترل کنیم می توان راه حل را پیدا کرد. واحدبندی این امکان را می دهد که خطاها را پیدا کنیم.

- continuaty: سیستم وقتی ماژولار باشد می توان آن را به راحتی تغییر داد بدون اینکه نیازی باشد به همه توجه کنیم.

PAPCO

Subject:
Year. Month. Date. ( )

یک واحد می‌تواند کلاس، actor، process، Function و ... باشد. اینکه واحدها چیست و چه ارتباطی با یکدیگر دارد برای ما اهمیت دارد.

1- cohesion: واحدها چه هستند. پایین‌ترین سطح آن این است که به صورت تصادفی فقط تقسیم کنیم

2- coupling: ارتباط بین واحدها

1-1 Logic cohision: رفتارهای منطقی را در یک جا قرار دهیم: پردازش و محاسبات در یک جا - رفتارهای منطقی مرتبط نیز در یک جا وجود دارد بوسیله دستور ~~...~~ union در C می‌توان به این سطح برسیم. مثلاً یک بند خواندن تعریف می‌کنیم که هر نوع خواندن را باید بتوان انجام دهد

1-2 produderal cohision: برای اینکه منابع حافظه خوب به کار گرفته شود، تیم عملیاتی که به صورت سریال می‌خواهند با هم اجرا شوند را در یک جا قرار می‌دهد. Function ها و proc ها را بوجود آورد.

1-3 communication coh: Data Abstraction است. برای هر نوع انتقال داده انجام می‌دهیم.

1-4 content cohision: تمام توجه روی عناصر با سیو تری است ولی توجه به عناصر محیط واحد است. هر کدام از انتقال داده‌های مختلف که می‌توانند یک تفسیر واقعی را تعریف کنند. (communication)

Subject:
Year. Month. Date. ( )

2- :coupling

با زیاد شدن سطح ها همبستگی ارتباط بیشتر می‌شود.

- پایین ترین سطح ارزش یک داده را ارسال می‌کنیم مانند call by value در توابع یعنی یک داده ای که در اختیار داریم را ارزش آن را ارسال می‌کنیم

- stamp coupling: call by reference یک متغیر دیگر ارزش دارد و به ماژول دیگر آدرس آن را می‌دهیم در نتیجه هر تغییری در محتوای آن آدرس بدهد در آن اعمال می‌شود.

- control coupling: با یک flag ماهیت ماژول را مشخص می‌کنیم یعنی یک ماژول است و رفتارهای متفاوت می‌تواند داشته باشد.

- external coupling: واحد همبستگی زیادی به دستگاه خروجی خاص دارد و فقط متناسب با آن ایجاد می‌شود.

- common coupling: global varieble ها هستند.

تمام این متغیرها برای پیاده سازی است اما در طراحی هستیم و در روش Data coupling اجازه فراخوانی توابع را ندارید طرح های جدید Data coupling این است که سعی می‌کند اجرای کاملا مجزایی را ایجاد کند که هیچ ارتباطی با یکدیگر نداشته باشند.

:structure hierarchy

نشان می‌دهد که واحد چگونه با دیگران در ارتباط است. در سلسله مراتب از واحدها یک قانون وجود دارد. هرچه به بالا می‌رویم تصمیم می‌گیرد هرچه به پایین سلسله می‌رویم اجرایی می‌شود.

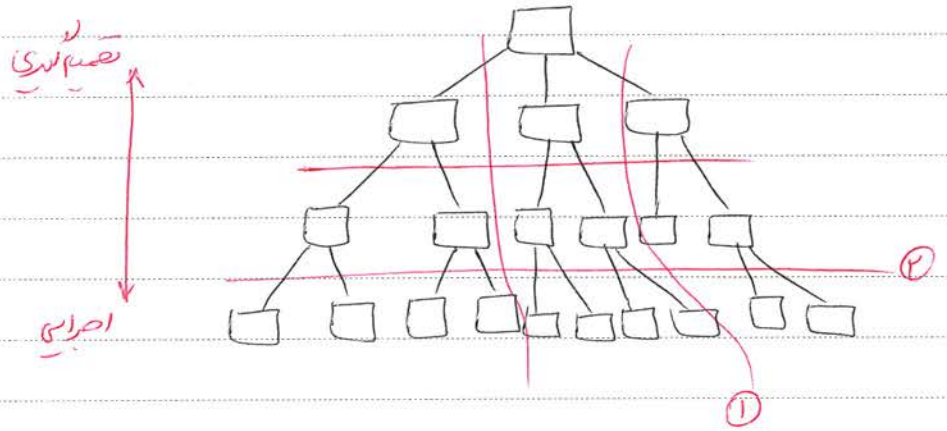
PAPCO

Subject:
Year. Month. Date. ( )

دونوع decomposition وجود دارد :

horizental decom : یعنی خط افق را به صورت دو به دو جدا می کند و task های مختلف تشکیل می دهد.

vertical decom : سطح عمودی را تقسیم می کند.



pattern :

یک سری pattern های استاندارد برای استفاده وجود دارد. درواقع یک لغت معمولی نیست و بار معنایی زیادی را باخود همراه دارد (مانند خانه ویلایی)

Concurrency :

یک pattern است یعنی عملیات به صورت موازی با هم اجرا می شوند و رابطه و وابستگی با یکدیگر ندارند.

persistance : داده هایی که می خواهیم استفاده کنیم را درون یک pattern قرار می دهیم

PdPCO

Subject:
Year. Month. Date. ( )

Data design :

تعدادی از اصطلاحات مورد استفاده در اینجا عبارت اند از:

master file : فایل هایی که اطلاعاتی که مشتری وارد می کند درون آن ذخیره می شود.

transaction file : یک عمل bussiness می خواهد انجام شود اطلاعات مانع مانده درون این فایل ها ذخیره می شود.

Document file : فایل هایی اطلاق می شود که اطلاعات خلاصه ای از master file را به ما می دهد. یک سیستم مستقل نیست و هویتش را از master می گیرد. کسی اطلاعات درون آن را تغییر نمی دهد و فقط استفاده می کند (همزمان عمل ذخیره می شود)

Archical file : رفتارهای افرادی که با فایل کار کرده ذخیره می شود کسی که update کرد کی بوده و چکاره؟ اگر خراب شد با این وسیله می توانیم ببینیم به ترتیب چه اتفاق هایی افتاده. روباتی توانایی عملیات می تواند کمک کند که خطا را شخص دهیم.

table lookup file : فایل هایی اطلاق می شود که همه اطلاعات که مورد نیاز است از آن load می شود. اطلاعات عمومی که فرم ها از آن استفاده می کنند. فایل ثابتی که با اول همه اطلاعات مورد نیاز دارد. مثل فایل بلوک در tetris

Audit file : یک سری عملیات که نا کید به صورت individual باید انجام شود فایلی که اطلاعات آخرین عملیاتی که داریم انجام می دهیم در آن قرار می گیرد.

update های بردی master file انجام می شود تا هر یک transaction فایل انجام می شود یک transaction که دستیم بردی master انجام می شود اطلاعات را:
trigger می کند ← یک proc ذخیره شده در سرور database است و بردی cond خاص برداشت
بصورت اتوماتیک اجرا می شود. هیچ کس proc بردی سرور را صدا نمی زند.
stored procudure ← برنامه هایی که برای کار کردن با آن است و توسط App قابل صدا زدن است.

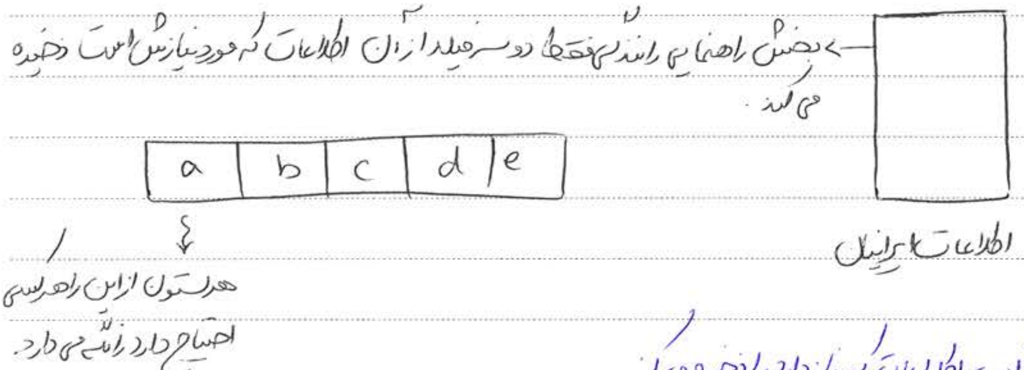
Subject:
Year. Month. Date. ( )

centrelized : تمام اطلاعات روی یک سیستم مرکزی وجود دارد و برای دسترسی مجبوریم به آن ارتباط باشد.

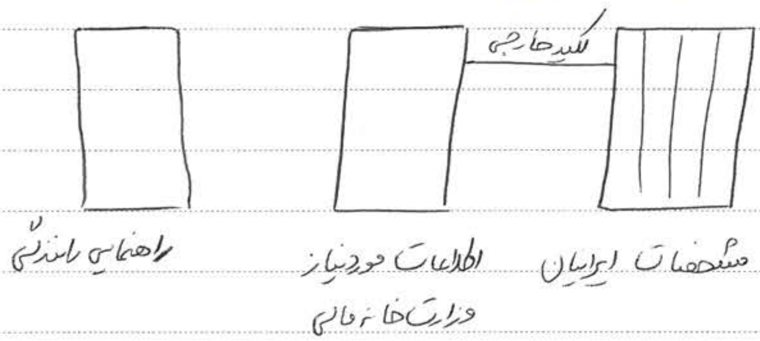
distributed : در سیستم های توزیع شده به دو روش می توان اطلاعات را در سیستم های دیگر پخش کرد.

1- پخش از طریق horizental distribu. : اگر همه اطلاعاتی که در اختیار داریم پاره ای از آن را در یک جا قرار می دهیم و پاره ای دیگر را در جای دیگر قرار می دهیم ← اطلاعات مالی هر شهر را در سایت اطلاعاتی همان شهر نگه داریم.
اطلاعات دانشجویان هر دانشگاه درون همان دانشگاه ذخیره می شود و می توانند آن را update کنند

2- vertical distribution : از هر پدیده ها هر جایی داریم. سوابق مالیاتی در تهران است و بازرسی مالیاتی را در جاهای دیگر پخش می کنیم.



یعنی هر کس اطلاعاتی که نیاز دارد را ذخیره می کند.
که ولی موجود همان کلید خارجی پس پایگاه داده های مختلف است.



PAPCO

Subject:
Year. Month. Date. ( )

Replication :

هر کسی اطلاعات خود را تغییر می دهد دلیل است افزونگی وجود داشته باشد. اطلاعات والی تهران در پایگاه ها نیز وجود دارد تا در صورت از دست دادن اطلاعات در یک محل در جای دیگر هم باشد. پس اگر اطلاعات یک محل عوض شود باید در پایگاه های محل ها نیز تغییر کند.

Data warehouse :

ویژگی اصلی آن عبارت است از:

1) subject-oriented : براساس موضوع تقسیم بندی می شود

2) Integration : ارتباط بین اطلاعات که براساس اطلاعات موجود است. دانشی از آن استخراج می شود که یک حس است و درست بودن آن حکمی می شود. عاملی بین بانک های اطلاعاتی است که بتوانیم با یکدیگر ربط بدهیم.

3) time : اطلاعات برای یک برهه زمانی خاص باشد

4) غیرقابل تغییر بودن

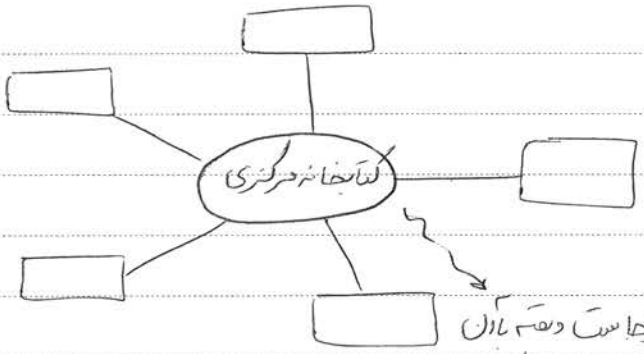
مهم ترین DNA هر کشور مرکز آمار آن کشور است. وزارت خانه ها نیز دارندولی اعتبار بین المللی ندارد.
از روی اطلاعات می خواهیم تصمیم گیری کنیم، برای این کار اطلاعات را از روی نمونه گیری بدست می آوریم معمول است درصدی خطا هم وجود داشته باشد ولی اصیت دارد. در پیشامدی هیچ اطلاعاتی برای تصمیم گیری ندارد. یک تابع توزیع از افراد را مشخص می کند تا بدانیم از کجا باید نمونه برداری کنیم. برآورد تابع توزیع را آمار می گوید.

طراحی معماری:

انواع معماری ها به صورت زیر است:

centralized : معماری مرکزی به چند چیز اطلاق می شود: ۱- پردازش ها ۲- تصمیم ها و داده ها ۳- کنترل و Appl. و process در یک جا متمرکز است.

Subject:
Year. Month. Date. ( )



دارای بلط هستند.

باید همه معماری های کار رفته در یک سیستم از یک نوع باشد:

محاسن:

۱- مدیران دوست دارند همه چیز دست خودشان باشد.

۲- کنترل داده ها بهتر است.

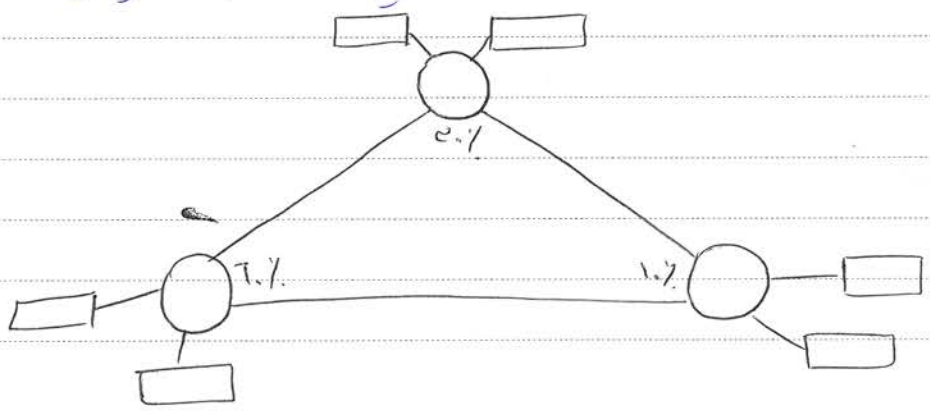
معایب:

۱- pps باید بالا باشد تا بتواند به همه پاسخ دهد.

۲- تجهیزات گران قیمت نیاز دارد.

distributed: با یک سیستم ارزان قیمت، سیستم را ایجاد می کنیم و ترافیک ها را در یک محل پاسخ می دهیم

به عنوان مثال بلیط فروشی است ولی اگر سیستم مرکزی خراب شود دیگر کسی نمی تواند بلیط بخرد.

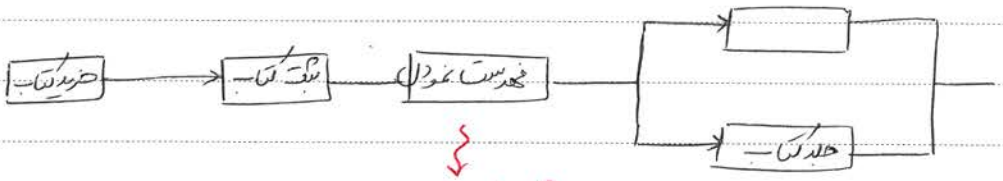


PAPCO

Subject:
Year. Month. Date. ( )

می‌تواند اندازه ۱۰٪ بلیط فروش را از سیستم‌ها قطع شود و مشکلی ایجاد نمی‌کند. درواقع تصمیم‌گیری می‌کند زمانی را که آنها مهم است کدام event بیاید و تصمیم خودش را اتخاذ کند در نتیجه می‌تواند اگر بعضی node ها مهم خالی دارند از آنها استفاده کند. شکل زمانی پیش می‌آید که کسی بلیط خود را cancel کند.
هر App را می‌توان به صورت توزیع شده داشته باشیم.

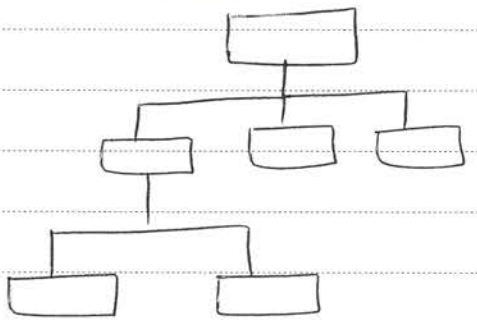
Dataflow : به دنبال نشان دادن کانون تصمیم‌گیری و کنترل و ... نیست بلکه به دنبال نشان دادن process هاست.



pipe and filter

دراین دیاگرام فقط مراحل انجام را نشان می‌دهد ولی در Dataflow های قبلی قوانین تصادفی داشتند و Data دوبدل می‌شد. در هر تقسیل فعالیت هایی که در یک process انجام می‌شود را نشان می‌دهد.
از این ساختارها بیشتر برای structure Analysis استفاده می‌کنیم. طراحی معماری با شدی مطابقت دارد در حالی که طراحی برای برنامه‌سازی انجام می‌شود.

call & return : یعنی OO نیست. یک برنامه اصلی وجود دارد که زیر برنامه‌هایی را صدا می‌زند. نحوه صدا زدن این برنامه‌ساختار، call & return را تشکیل می‌دهد.

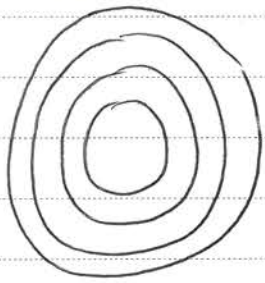


از dataflow در می‌آید

Subject:
Year. Month. Date. ( )

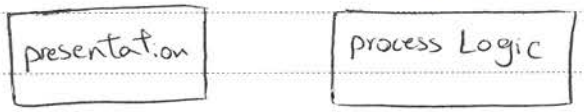
Layer:

لایه بندی مربوط به سطوح مختلف یک application با توصیه به دید طراح دارد. دهها لایه همان فعالیت های قبلی را ارائه می دهد اگر لایه های پایین تر وجود نداشته باشد فایده ندارد ۱- برای جهان تغییر تکنولوژی بالایه بندی تنها لازم است همان لایه را عوض کنیم. ۲- reusability دارد. سیستم را در هدهای دیگر هم می توان استفاده کرد. ۳- می توان فازهای incremental را به این شیوه بالا برد تا هدهایی که بودجه دارید بالا می روید در هر سطح که باشیم سیستم قابل استفاده است.



tier:

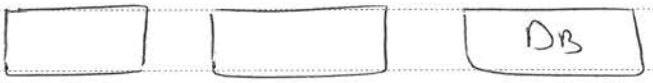
سیستم خرد کردن است ولی پردازش ها خرد می کنیم.



2-tier processing

اگر یکی از آنها را انجام دهیم فایده ندارد ← و باید هر دو پیاده سازی شده باشند.

تغییرات تکنولوژی یکی در دیگری منتقل می شود با این تفاوت که برای کار کردن هر دو باید وجود داشته باشد.



tree-tier processing

می توان DB را تغییر داد ولی تغییری در دیگری ایجاد نشود.

درایو ها سه نوع اطلاعات وجود دارد:

۱) File system: منبع مشترکی فراهم می کند هم باید از آن استفاده کند اگر این منبع یک فایل باشد باید هر کس که می خواهد از آن استفاده کند یک برنامه داشته باشد هم client ها باید FAT client باشد

Subject:
Year. Month. Date. ( )

یعنی همه اطلاعات یک کامپیوتر را داشته باشد.

۲_ client server : منبع مشترک یک برنامه است client ها فقط Data می‌دهند و آن نتیجه را می‌دهد داده‌ها را رد و بدل گروه در client برنامه‌ای وجود ندارد فقط داده می‌دهند و می‌گیرند. client ها می‌توانند thin client باشند.

۳_ Internet base : در منبع می‌توان Aplicaton/data داشته باشیم و در client ها یک browser داریم که عمل پردازش درون آن وجود دارد که فقط بتوانند اطلاعات را نمایش دهد، که می‌توان با تعداد pc هایی که می‌خواهند از آن استفاده کنند. App / data بگذاریم و n_tier را بوجود آوریم.

هر یک از این معماری‌ها پیچیدگی خاص خود را دارند یکی از این پیچیدگی‌ها فقدان inheritene است.

نکته: هسته اصلی در design ↓coupling و ↑cohision است.
اختلاف cohision باعث می‌شود که ما coupling های مختلفی را انتخاب کنیم. روش‌هایی که برای طراحی در نظر می‌گیریم عبارت اند از:
۱_ عناصر و محتوا مسئله را در نظر می‌گیریم. اگر می‌توانیم اصلاً OO در نظر بگیریم برای همین از module استفاده می‌کنیم
۲_ مشخص واحدها در زیر ساخت‌ها
۳_ مشخصات و جزئیات ماژول را تعیین می‌کنیم. ماژول‌ها از طریق GOS / component of the shell قابل استفاده است.
۴_ interface برای هر ارتباط را مشخص می‌کنیم. interface در کجا قرار می‌گیرد و چه کسی استفاده می‌کند.
۵_ type و نوع داده را مشخص می‌کنیم. ساختمان داده‌ای که قرار است رد و بدل شود.
۶_ چه زیر عملیاتی انجام می‌گیرد. زیر عملیات را به صورت procedure تعیین می‌کند. در کجا چگونه فعال شود
۷_ چه فایل‌ها و DB هایی باید وجود داشته باشند داده DB یا فایلی داریم چه عملیاتی بر روی آن انجام می‌شود و یک self_strategy می‌دهیم و مدیریتی که بر روی آن باید انجام شود را تعیین می‌کنیم
۸_ دنبال گروه بندی می‌گردیم که همه را در یک سیستم قرار دهیم.
۹_ در یک روش دیگر با راه حل های دیگر با آن برخورد می‌کنیم.

PAPCO

Subject:
Year. Month. Date. ( )

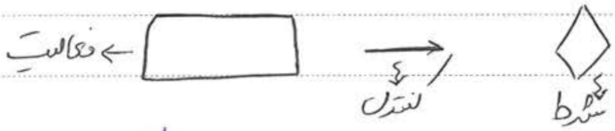
decomposition ← اول package مشخص می‌شود سپس کلاس بندی می‌شود ← ابزارها هم شهودی و ذهنی است

composition ← کلاس ها مشخص شده دیده package تعیین می‌کند ← ابزاری برای آن وجود دارد.

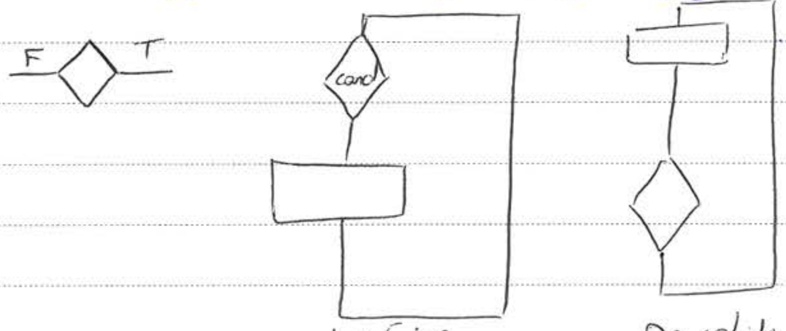
روتین ها را چگونه بیان کنیم؟ چگونه نمایش دهیم؟

می‌توان این کار را با چند روش انجام داد:

۱- فلوچارت (flow chart): نمایش گرافیکی دارد و خوب است و سعی می‌کند کنترل اجرایی یک سیستم را نشان دهد.



فلوچارت control flow فقط سپس را نشان می‌دهد ولی در Data flow نمایش Data که بین آنها تبادل می‌یابد را نشان می‌دهد



حداقل یکبار اجرا می‌شود.

۱- چیزهای پیچیده را نمی‌توان با آن نشان داد.

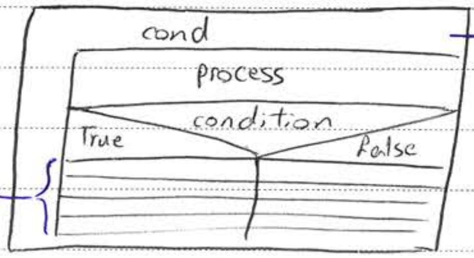
۲- استفاده از آنجا ممکن است بد باشد. مثل استفاده از Goto که مناسب نیست و باعث می‌شود برنامه قابل درک کردن از هم گسسته باشد. بعد از آن if و while استفاده کردند که از آن زبان ساخت یافته است. از یک جا وارد می‌شود و از جای دیگر خارج می‌شود.

فلوچارت برای این نوع از زبان ها مناسب بود برای همین از یک ساختار دیگر استفاده شد.

PAPCO

Subject:
Year. Month. Date. ( )

۲- nass_schinideman:



مورد استفاده قرار نگرفت زیرا در شرط های تودرتو پیچیده می شود.

۳- Decision table:

ولی برای تصمیم ها و حلقه ها قابل استفاده نبودند.

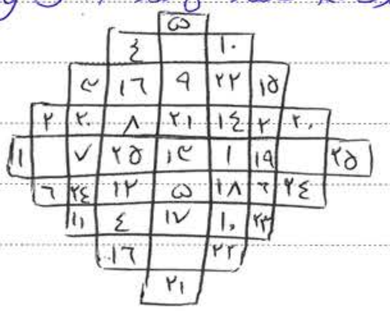
۴- PDL:

همان زبان سودوکد است. از زبان های نزدیک برنامه نویسی استفاده کرد و syntax مهم نیست و تا جایی که دارد جزئیات می شویم که طرف مقابل نمی فهمد در صورتی که مفهوم را درک می کند لازم به پیاده سازی نیست.

ممکن است اصلاً وجود نداشته باشد ولی مفهوم را مشخص می کند. ⟵ swap(a,b)

refactoring:

یکی از عوامل مهم در Design است. هر برنامه ای می نویسیم زمانی که دستورالعمل هایی که در اسمبلی وجود دارد با یک برنامه ده میلیون خط ایجاد می شود. بسیاری از قسمت ها شبیه یکدیگرند که می توان قسمت های مشابه را جدا کرده و به صورت مشترک استفاده می کنند که به آن refactoring می گویند.



(3,4) = 1
(2,5) = 2
(1,1) = 3
(5,2) = 4
(4,3) = 5
(4,5) = 6

Subject:
Year. Month. Date. ( )

سعی می‌کنیم گروه بندی می‌کنیم همهٔ آنهایی که در بخش پذیر هستند و هر نوع دسته بندی یک نظم پیدا می‌شود.
پس همواره می‌توان با این نظم برخورد کرد و می‌توان دسته بندی‌هایی ایجاد کرد و نظم برقرار کرد.

8 Interface Design

اگر prototype ایجاد می‌کنیم اولین مرحله Design است ولی وقتی یک چیز واقعی است آخرین مرحله است. در آن با مفاهیم زیر برخورد می‌کنیم:

1- رابطهٔ انسان با برنامه

1a خروجی‌ها: از طریق خروجی‌ها ارتباط برقرار می‌شود خروجی‌ها متفاوت هستند در دسته‌های زیر تقسیم می‌شوند:

internal output: میان همراهان سیستم رد و بدل می‌شود

- Detail report کامل پردازه‌ها را فرا... در این بخش‌ها جا داده می‌شود.
- Summery rep خلاصه‌هایی که به مدیران داده می‌شود.
- exception نقاط و شروط‌ها

external output: به enduser داده می‌شود.

time around out: معمولاً به عنوان دردی استفاده می‌شود.

output ها می‌توانند جدولی یا ماتریسی باشند:

هر پدیده‌ها در سطر است.

سن	نام خانوادگی	نام

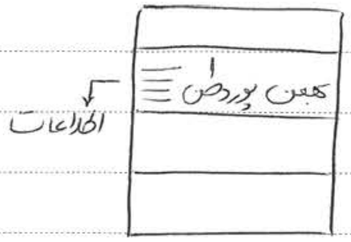
جدولی

اصفهان				
تهران				
⋮				

ماتریسی

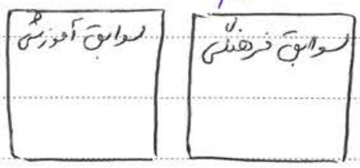
PAPCO

Subject:
Year. Month. Date. ( )



⟵ اطلاعات یادداشت جدول نمی‌شود.

output zone : محدوده‌ای اختصاص می‌دهد که همه اطلاعات را در آن وارد می‌کنیم



راهنمای تولید خروجی‌ها:

۱- از قدیم از شیت (sheet) های آماده برای خروجی‌ها استفاده می‌کردند طول حروف و اندازه را ثابت در نظر می‌گرفتند ولی در پرینترهای جدید امکان‌پذیر نیست.

۲- خروجی‌ها باید ساده و قابل خواندن باشند

۳- هر خروجی باید دارای title باشد

۴- تاریخ و ساعت باید داشته باشد

۵- section بندی باید شده باشد و هر section، segment خاص خود را داشته باشد

۶- هر فیلد ها دارای lable باشند و حوزه آن مشخص شود. دارای level و قابل تشخیص است

۷- اگر ستونی یا سطری است سطر و ستون مورد استفاده قرار گیرد

۸- تنها اطلاعاتی که لازم داریم را نشان دهیم.

Subject:
Year. Month. Date. ( )

۹ـ خروجی باید editable باشد برای همین قوه‌ی قضاییه نمی‌پذیرد Add * Name
مستندات اینترنتی را قبول ندارد. نام باید یا عددی باشد و scan شده باشد.
خروجی خودش را نشان می‌دهد
۱۰ـ خروجی باید پالایش شود. خروجی‌ها در یک جا تراکم اطلاعاتی زیاد دارد ولی در جای دیگر خالی باشد.
۱۱ـ به راحتی هر چند بار بتوانیم
۱۲ـ زمان پاسخ خیلی مهم است. خروجی‌ها باید بر اساس آن تعیین شود. عکس‌العمل خروجی متناسب با سرعت انسان باشد.
۱۳ـ همه کاربران را باید در نظر بگیرد. هر ذی‌نفع که در سیستم وجود دارد باید output ی برای آن مشخص شود.
۱۴ـ خروجی باید به درد کاربر بخورد و خروجی را کسی باید بپسندد که استفاده می‌کند ولی خوب نوشتن برنامه باید باید الگو ای باشد که کسی که پول می‌دهد بپسندد.

چیزهایی که در خروجی بخصوص کاغذی در نظر گرفته می‌شود:
۱ـ heading
۲ـ Agriggation (خلاصه‌ها)
۳ـ heading برای هر ستون
۴ـ Alignment بین heading
۵ـ page size
۶ـ heading برای سطرها
۷ـ فونت‌ها برای تعیین و...
۸ـ کنترل control break مثل سرجمع است که اطلاعات عمومی را جمع‌بندی کنیم
۹ـ پایان گزارش باید مشخص شود

در screen چه چیزهایی اهمیت دارد؟
navigation - partitioning - information hiding - highlight size

PAPCO

Subject:
Year. Month. Date. ( )

ورودی :

ورودی ها می توانند به صورت Data entry و Data capture باشد
Data entry: اطلاعات در اختیار است می خواهیم وارد سیستم کنیم
Data capture: جمع آوری اطلاعات

keyboard ، elec magnetic ، smart card و ... می تواند ورودی باشد

توصیه :

۱ـ فقط متغیرها را بگیریم و ثابت ها را لازم نیست.
۲ـ قابل محاسبه را نگیریم و به جای آن اطلاعات ثابت را نگه داریم.
۳ـ برای ورود اطلاعات کدگذاری کنیم برای اینکه سرعت خوبی داشته باشیم. انواع کدینگ ها را باید بدانیم یعنی کدینگ ها را باید یکی یکی بلد باشیم (؟) و حجم ورود اطلاعات را کم می کند.
۴ـ Data capturing توضیح برای پرکردن آن
۵ـ تا جایی که می شود فرم ها و کنترل ها دستنویس نباشد از check box و یا ضرب در کردن این کار را انجام دهد. کسی پرکردن را نمی خواهد
۶ـ فرمی برای ورود اطلاعات در نظر گرفته ترتیب خواندن آن هم به همین فرم باشد.
۷ـ Data entry کنترل برای اینکه درست وارد شده باشد.
emistance check ← آیا چیزی زده شده یا نه
Data type ← عددی باشد یا غیر عددی
Domain chect ← رنج منطقی باشد
combinational ← فیلدهای به یکدیگر مرتبط اند.

اصول موجود عبارت اند از:

۱ـ باید بتواند با همه آیتم هایی که روی یک صفحه است تعامل داشته باشد. باید برای هر کار در صفحه خاص خودش را قرار دهیم. نباید گزینه های disable داشته باشیم هر چیزی که روی صفحه است باید با آن تعامل داشته باشد.

PAPCO

Subject:
Year. Month. Date. ( )

۲- بنابراین کاربر این اجازه را بدهد چیزی را حفظ کند. بنابراین عملیات طوری باشد که بخواهد چیزی را در جایی ذخیره کند تا بتواند در صورت توقف یادآوری کند. باید راهنمایی هایی در interface وجود داشته باشد. باید کدها و اسامی با معنی باشد تا این حالت برقرار شود.

۳- فرم های Data entry باید شبیه دنیای واقعی باشد و با فرم های کاغذی یکی باشد.

۴- سازگاری: interface ها باید سازگاری داشته باشند.
help ← مربوط به F_1
ESC ← خارج شدن است
همه اینها را یاد گرفته‌اند و باید با اینها سازگار باشد
برای داشتن سازگاری باید استاندارد هایی تعریف شود تا همه با یکدیگر هماهنگ باشند.

کاربران ناشی: به کاربرانی که نه کامپیوتر بلد است نه کار خودشان را بلد است.

کاربران حرفه‌ای: نه تنها با کامپیوتر آشنا هستند تخصص کافی در کار خود دارند ← چون کار زیادی با PC انجام می‌دهند interface را برای کار خود customize می‌کنند. مهارت با آموزش به دست نمی‌آید.

هر گروه باید بر اساس اهدافی که دارند تقسیم بندی شده و interface خاص خودشان را داشته باشد.
عواملی که تعیین کننده interface است:
۱- کاربر: user چه کسی است؟ چه سابقه‌ای دارد؟ شغل افراد نیز تعیین کننده است؟ طراحی واسط در طول مدت spiral استفاده می‌شود. یکی از روش ها برای تعیین interface این است که چیزی ایجاد کرده به ایشان نشان می‌دهیم در روش spiral تکمیلش می‌کنیم.
۲- Work Flow نیز تعیین می‌شود که دوست دارد چطور کار انجام شود و بر اساس آن کاری انجام می‌دهیم.

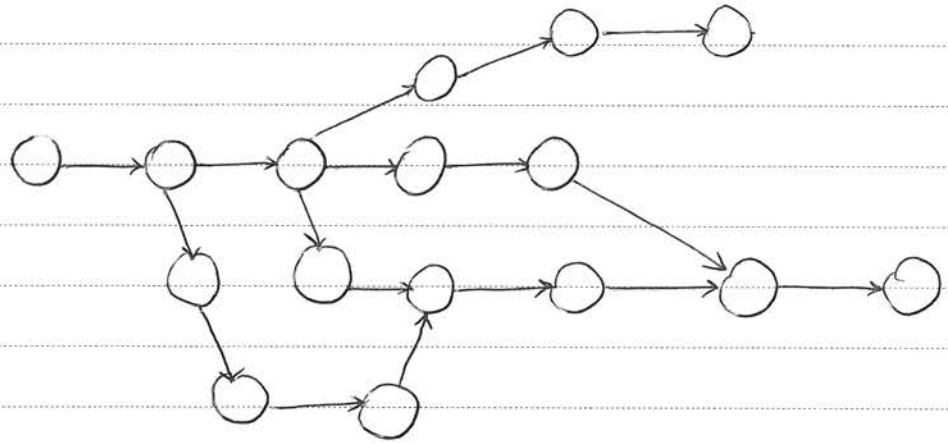
یکی از مشکلات اصلی response time است. اوقات مختلف زمان پاسخ متفاوتی دارند.

توصیه:
۱- پیام را طوری شرح دهد که بفهمد چه شده
۲- باید به گونه‌ای باشد که بتواند خطا را تصحیح کند
۳- ترکیبات منفی خطا را گوشزد می‌کند
۴- لحن پیام تحقیر کننده نباشد.

PAPCO

Subject :
Year . Month . Date . ( )

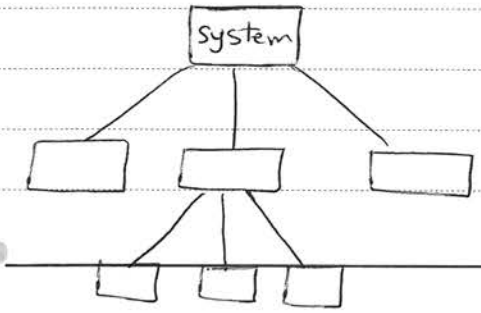
یکی از عناصر Structure Analysis استفاده از Data flow diagram است در pipe فلش‌ها فقط ترتیب انجام فعالیت ها را مشخص می کند ولی در DFD همراه فلش ها داده ها منتقل می شود.



برای تقسیم برنامه سه فازول های کوچکتر می توانیم بر روی DFD آن برش بزنیم این برش ها به هر صورتی می توانند باشند. معیار برای cut کردن عبارت است از:

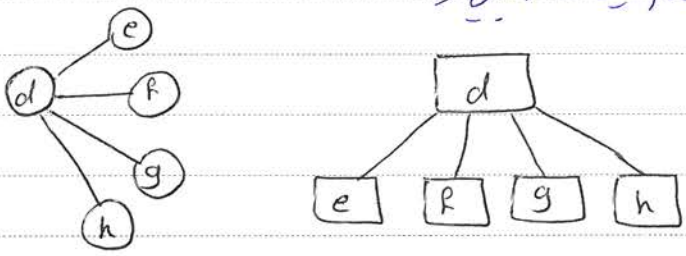
۱- توازن و تعادل بین بخش های موجود.
۲- جنس IPD : همه ورودی هایی که وارد شده اند ورودی هستند ، پردازش های منطقی ، آماده سازی برای خروجی . هر یک node ها را اساس این سه دسته تقسیم بندی می شوند.
۳- ترکیبی : اگر فقط یک ورودی و یک خروجی وجود داشته باشد آن ها را جدا کرده و سپس در صورت میانی از روش درختی استفاده می کنیم

به خاطر IPD معمولاً تقسیم بندی اولیه به صورت سه تایی انجام می شود. از طرفی نمی توان به صورت pancake یک دفعه تقسیم بندی را انجام داد و باید به صورت سلسله مراتبی آن را گسترش دهیم.



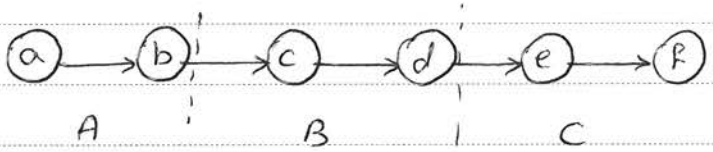
Subject:
Year. Month. Date. ( )

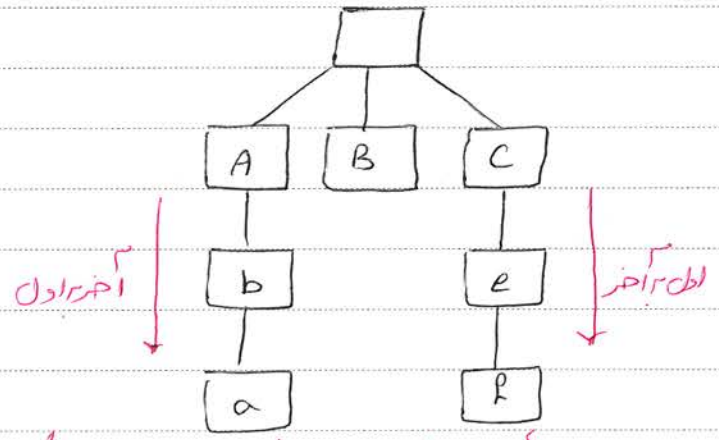
یک سیستم با الف کردن این سه قسمت ازهم جدا می‌شود. اگر با ازای یک cut تعدادی آغازکننده داشته باشیم می‌توان به تعداد آغازکننده‌ها شاخه ایجاد می‌کنیم
یکی از عوامل دیگر تقسیم‌بندی محدوده transaction ها است. اگر در یک حالت چندین transaction وجود داشته باشد می‌توان برای هرکدام یک شاخه تعیین کرد



درنتیجه سه عامل برای تقسیم‌بندی داده‌ها وجود دارد است:
۱- درونی ۲- cut ۳- transaction

Sequential : در یک DFD که به‌صورت sequential باشد سیستم را به سه زیرشاخه تقسیم می‌کنیم که درون هریک از این بخش‌ها به صورت زیر تقسیم‌بندی می‌شود:





ابتدا b را صدا می‌زنیم که داده خودش را صدا می‌زند و سپس داده‌های خود را صدا می‌کند. چون b برای قسمت بعدی Visible است.

چون e داده قسمت قبلی می‌باشد سپس باید داده e صدا زده شود و داده‌های آن نیز را صدا می‌زنیم.

Subject:
Year. Month. Date. ( )

رابطه بین قسمت های مختلف A و B و C بوسیله ساختار خود درخت انجام می شود.

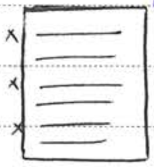
یکی از دلایل انتخاب cut به گونه ای است که درخت را فرم داشته باشیم و ساختار درختی حفظ شود پس باید cut را به گونه ای دهیم که هر node فقط یک parent داشته باشد.

طراحی واحدهای برنامه:

از جمله واحدهای مختلف برنامه عبارتند از:

- function \ procedure: در حالت کلی به این دو subroutine می گوییم. دهی به دستورالعمل هایی است که به صورت instruction یا جزئی از یک عبارت خواهد بود که یک value دارد و به آن function می گوییم.

- Coroutine: نقطه‌ی ورود می تواند در جاهای مختلفی وجود داشته باشد و لزوماً از یک جا شروع نمی شود مهم این است که قبلاً کجا بوده است و از هر جایی transfare داشته باشد به همان جا می گردد.
برای برنامه های موازی نمی توان از procudure استفاده کرد و باید بتوان در حین اجرا عملیات را متوقف کرده و سپس برگشته و ادامه دهیم.



- object: entity هایی هستند که دارای یک رفتار خاص و discriptor هستند که تغییرات discriptor توسط رفتارها انجام می شود.

int ← discriptor: دارای 4 بایت است که یک بیت علامت دارد که برای هر این discriptor ها یک دایره‌ی int به کار می بریم.
از جمله عملیاتی که بر روی آن انجام می شود SHL ، SHR و جمع، تفریق و ضرب است.

فقط براساس همین علائم هایی توان این کار را انجام داد، که در اینجا به آن type می گوییم. وقتی این مفاهیم

Subject:
Year. Month. Date. ( )

وارد محیط غیر کامپیوتری شود به آن زده obj می گوییم.

برای اینکه یک زده obj استقلال داشته باشد نباید هرکسی به آن دسترسی داشته باشد. در صورتی که استقلال وجود داشته باشد می توان reuse کرد. زمانی که یک زده obj می خواهد با اشیای بیرون در تماس باشد ولی نیازی به شناخت محیط نداشته باشیم باید در صورتی که کسی می خواهد با آن در تماس باشد از طریق یک interface خاص این کار را انجام دهیم. در نتیجه به گونه ای باشد که اگر متغیری درون آن به کار رفته است کسی نتواند از آن استفاده کند. از طرفی کسانی که می خواهند با متدها کار کنند باید آنها را بشناسند، پس اجازه فراخوانی متدها را به آنها می دهیم. پس برای ارتباط آنها با یکدیگر از massage passing استفاده می شود و تنها با فرستادن پیام در ارتباط با یکدیگر قرار می گیرند. پس هر کدام باید یک متد فعال داشته باشند که به آن component می گویند.

- component : یک زده obj هستیم فعال است در java کلاس های مربوط به GUI یک component است ولی چون ماهیت برای زده obj oriented است دسترسی به متدها وجود دارد. می توان بوسیله آن comp ایجاد کرد ولی component programing برای آن نیست. زده obj یک موجود مرده است و با اجرای متدهای آن فعال می شود (زده passive obj) ولی در component چندین زده obj که مرده هستند در کنار یکدیگر قرار داده و یک برنامه بر روی آن ها اجرا می شود پس از اجرای برنامه هم با یکدیگر کار می کنند.

- سرویس (Software as a service - SAAS) : سرویس ها در زمان انتخاب و اتصال بین اجزای مختلف هم زنده هستند و در حال اجرا و سرویس دهی به دیگران می باشند و ما تنها یک کانال از چندین سرویس گرفته و کار خود را انجام می دهیم. تفاوت service و component در سطح cohisiont است.

در سرویس هویت زنده آن منحصر به فرد است و اگر چندین نفر هم بخواهند از آن استفاده کنند بین هم مشترک است ولی در component هر کسی به آن احتیاج داشته باشد یک کپی از آن می گیرد.

PAPCO

Subject:
Year. Month. Date. ( )

اصول طراحی یک واحد:

اصولی که برای طراحی یک واحد در نظر گرفته می شود عبارت اند از:

OCP: واحدی که طراحی می شود به تغییراتی که از آن استفاده می کنند اجازهٔ modify کردن داده می شود یا نه. در واقع برنامه باید open source باشد یا نه. زمانی که یک برنامه را open source می کنیم علاوه بر اینکه هزینهٔ زیادی باید صرف شود تا افراد روی آن برنامه را فهمند و فایل ها را با حزئیات بررسی کنند، نیاز افراد متخصص زیادی دارد. در objectoriented می توان از طریق inheritence این کار را در سطح بالاتری انجام داد. به گونه ای که برنامه نویس از برنامهٔ دیگر می تواند تغییراتی را که می خواهد روی آن بدهد. در این حالت نیازی به دانستن جزئیات نوشتهٔ عامل در فایل ها نیست.

LSP: یک پدیدهٔ کلی را در یک کلاس داشته و از آن استفاده می کنیم علاوه در زمان اجرا با استفاده از superclass ها جزئیات و کلیات تعیین می شود. در واقع جایی که می توانیم جزئیات را دور ریخته و کارها را در سطح کلی انجام داده و سپس در جزئیات می شویم و ذره ذره جزئیات را اضافه می کنیم.

DIP: توصیه Abstraction به ما کمک می کند که واحدی را ببینیم که دقیقاً برای ماندست و در طراحی واحدها هم به abstraction در نظر گرفته شده است باید به گونه ای طراحی شده باشد تا در جاهای دیگر نیز بتوان از آن استفاده کرد.

ISP (interface segrigation principle): در طراحی سعی بکنید یک فایل کلی پیدا کرده و برای هم سازگار باشد. این شیوه از این لحاظ خوب است که هم چند را در نظر می گیرد ولی از این لحاظ که سخت و پیچیده است مناسب نیست. از نظر کارایی پایین می آید و روی کیفیت تأثیر می گذارد.

CCP (common clasure principle): در اصل یک سری کلاس تعریف شده و آنها را گروه بندی کرده تا یک package ایجاد شود. در این اصل همهٔ آن هایی که یک کار را انجام می دهند باید در یک جا قرار گیرند و باید کلاس هایی که یک کار می کنند در یک جا باشند.

CRP: آن هایی که هیچ ارتباطی ندارند در یک جا قرار ندهیم.

Subject:
Year. Month. Date. ( )

(reuse, equivalencing principle)

REP : چیزی که قرار است reuse شود باید جدا و مستقل از دیگری باشد بدون اینکه چیزی وصل باشد

نیازمندی که قرار است reuse شود دقیق باشد.

Subject:
Year. Month. Date. ( )

8. Test

دو عامل مهم وجود دارد که باعث می شود با test رابطه برقرار کنیم:

۱- verification ـ اشکال نداشته باشد. چیزی که تولید کرده ایم bug ندارد و همین حالت بررسی می شود

۲- validation ـ کارکرد درست دارد یا نه. برنامه را برای چی توضیح می دهیم و همان است همیشه درست کار کند یا نه

اینکه برنامه فاقد هرگونه اشکال باشد را verification می گویند. ممکن است برنامه اشکال نداشته باشد ولی اشتباه برود و خروجی درست یا نه خروجی validation می شود. وقتی یک برنامه نوشته می شود دو کار برای آن باید انجام دهیم:
۱- نمایش درستی برنامه که با ورودی های مختلف چک می کنیم. ۲- اثبات درستی ← بهترین راه مدل های ریاضی

هدف از تست چیست؟ انگیزه اصلی عمل چیست؟
فقط دنبال خطا می گردیم. هدف درستی برنامه نیست. به این علت است که وقتی خودمان برنامه را نوشتیم داده هایی می دهیم که جواب درست را بدهیم. پس فقط تست برای پیدا کردن خطاست.

اولین جلسه پروژه تست و برنامه ریزی آن شروع شده. در اصل درمان ها انجام شده در اسم review انجام می شود.

از زمان construction به بعد این فعالیت ها را تحت عنوان test آشنایی داریم. به بازبینی های قبلی review می گوییم.

بعضی از خطاها ممکن است پنهان باشد و در یک روز یا مجموعه داده خاص نمایان شود. در validation نیز گاهی از ابتدا مشخص نیست و بعد از چند سال قابل تشخیص است. خیلی از جواب های قشنگ برای است که نمی دانیم درست کار می کند یا نه و نمی توان با usecase ها همین چیزی را فهمید.
بهترین روش برای این دو، روش های فرمال و ریاضی است که به آن model checking می گویند.

یکی از پرکاربردترین مدل فرمال این است که برنامه را از قالب ماشین خارج کرده پس می خواند logic و semantic را حلی کند ولی اشکالات برنامه نویسی مثل type ها و امکان بررسی آن، برداشت ها که از دستورات می توان کرد قابل تشخیص نیست.

Subject:
Year. Month. Date. ( )

مثلاً در if ترجای == از = استفاده کنیم error می دهد.

```
int A[10], I ;
For(I=0; I<=10 ; I++)
```

A[I]=0 ← هیچ چیز چاپ نمی‌شود زیرا وقتی I=11 است خانه 11 را می‌خواهد عوض کند که هان I است پس تا بی‌نهایت cout<<I ; حلقه می‌رسد دوباره صفر شده و از حلقه بیرون نمی‌آید در C دقیقاً همین طور است.

این مدت از نرم افزار هزینه بسیار زیادی دارد. شرکت‌هایی هستند که کارشان چک کردن برنامه‌ها و زدن تست به certification است. در یک برنامه چیزی که اهمیت دارد Data است نه روی‌ها پس دادن تست به شرکت‌های دیگر مشکلی ایجاد نمی‌کند.

استراتژی‌های به کار رفته عبارت است از:

1) Formal :

استراتژی تست از پایین به بالاست. هر ماژولی که تولید می‌شود شروع به تست کردن می‌کنیم. پس ترکیب آنها بررسی می‌شود. روش‌های تست باید مختلف باشد. روش‌های تست مختلف می‌تواند آشکار کننده error های ترکیب آنها باشد. تست واشکال زدایی دو مقوله کاملاً جدا است. تست باید توسط یک گروه مستقل انجام شود. برنامه‌نویس دقیقاً کدی را تولید می‌کند تست case را به گونه‌ای می‌نویسد که چیزی که نوشته شده راست کند. آن گروه نیز فقط خطا پیدا می‌کند واشکالات را برطرف نمی‌کنند.

test case ها باید از requirement درست بیایند در همان مرحله inception می‌توان تست را ایجاد کرد. آن کسی می‌تواند خوب تست کند که اشراف کامل بر روی برنامه داشته باشد. یعنی کسی که تست می‌کند باید تکنیک‌های پیاده سازی و requirement را بداند.

تست از کوچک به بزرگ است. اولین چیزی که تست می‌شود unit test است که code ها را بررسی می‌کند که درست است یا نه

P&PCO

Subject:
Year. Month. Date. ( )

unit test	code
integration	Design → ترکیب همه با هم
validation	test requirement
System test	System engineering → طراحی کلی سیستم مشکل دارد

اصل pareto: 80٪ خطاهایی که در یک برنامه بوجود می‌آید مربوط به ۲۰٪ برنامه است. یک بخش کوچکی از برنامه است. اگر درست باشد حل می‌شود. دیدن خطای زیاد به این معنا نیست که برنامه را کنار بگذاریم.

Rebustance: برنامه‌ها باید Rebust باشد. همان تست‌های Anti bugging است. یعنی خودمان خطاها را چک کنیم. در برنامه همه عملیات باید به صورت دام چک شود و این سطح کد نویسی را deffence programming می‌نویسند.

چه برنامه‌ای testable است و می‌توان آن را چک کرد؟ (ویژگی‌ها)

observability: می‌دانیم چه ورودی و چه خروجی دارد.

operability: برنامه را می‌توان run کرد. سهل است و صفو عملیات خاصی نیاز داشته باشد که در اختیار نیست. به هاردی نیاز دارد که در دسترس نیست.

controlability: هرچه بتوان برنامه را کنترل کرد برنامه بهتر خواهد بود. هرچه خروجی که می‌توانیم آنرا از ترکیبات ورودی بدست می‌آید.

decomposability: برنامه را بتوان به تکه‌های مستقل تبدیل کرد.

تست برای object oriented هم همین است.

P&PCO

Subject:
Year. Month. Date. ( )

Unit Test :

برای مشخص کردن یک سری خطاها unit test را انجام می‌دهیم در unit test به دنبال semantic نیستیم و می‌خواهیم system را چک کنیم و استفاده از min mode یکی از عواملی است که خطا ایجاد می‌کند.

- مقداردهی اولیه
- دقت محاسبات

Float i = 1E 34 بی‌نهایت بار اجرا می‌شود
while (i>0) i = i - 1;

زیرا { 1 E 34
1.........0 E 34
7 رقم را فقط می‌گیرد

- استفاده از symbol های اشتباه

if (3 & 4) p = 0 → and می‌کند و صفر می‌شود
else p = 1 → نتیجه p = 1

3 && 4 → غیر صفر → p = 0
(3 ← غیر صفر، 4 ← غیر صفر)

- اولویت‌ها

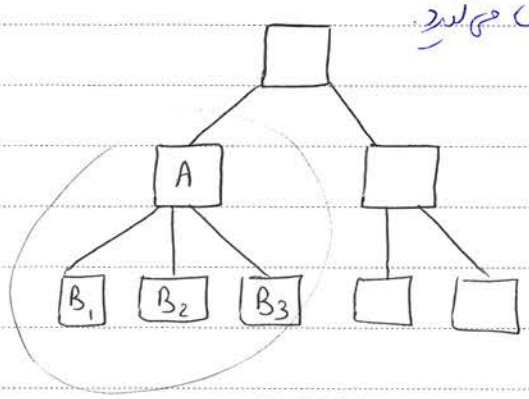
این جور error ها را می‌خواهیم کشف کنیم. بقیه‌ی های آن در ادامه بیان می‌شود. برای اینکه یک unit را run کنیم یک داده می‌تواند توسط کسی صدا زده شود و کسی آن را صدا نزند.

PAPCO

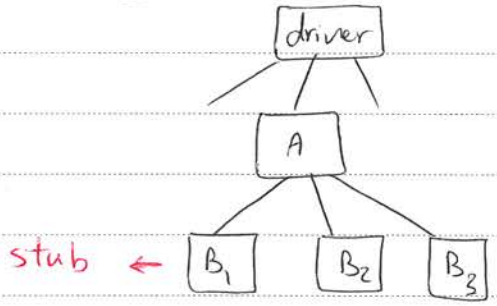
Subject:
Year. Month. Date. ( )

Integration Test

design درست است یا نه به چند روش صورت می‌گیرد.



برای تست کردن A یک برنامه نوشت که A را صدا ← بزند و بتوانیم آن را تست کنیم به این برنامه driver می‌گوییم.



با خروجی های دقیق تست می‌کنیم و ← / / ماهیتی ندارد

در این مرحله واحدها را از پایین به بالا می‌بینیم و تست می‌کنیم ولی در OO نمی‌توان قسمتی با عنوان Bottom-up داشته باشیم تست integrity داریم ولی نه به این روش.

① Regrasion test : هر تست هایی را باید دوباره تکرار کرد و فقط فایل‌هایی که تغییر کرده است را به عنوان یک شاخه تست می‌کنیم. یک شاخه یک task است و در واقع داریم task ها را تست می‌کنیم.

② sandwitch test : ترکیب هم با بالا و پایین را تست می‌کنیم.

③ Smoke test : نیاز به بودن همه برنامه‌ها نیست که تست را انجام دهیم و فقط همان قسمت برنامه که در دسترس است را ترکیبی تست می‌کنیم.

P4PCO

Subject:
Year. Month. Date. ( )

Validation Test:

بدترین حالت این است که وقتی تحویل می‌دهیم خطا داشته باشد و ایراد بگیرند. اطمینان مشتری را کم می‌کند. خودمان سیستم را روی plattform خود حلقه می‌کنیم و تست کرده آیا همه ویژگی‌هایی که آن می‌خواهد را دارد یا نه. اگر این تست را توسط افراد خود انجام دهیم α می‌گویند ولی این به این معنا نیست که محصول خوبی باشد و بعد از مشتری می‌خواهیم تست کند.

این که مشتری جای محیط واقعی سعی می‌کند آن را تست کند تست β می‌گویند. در این حالت در تست‌هایی که معلم گفتیم درست کار می‌کند در برنامه قرار نمی‌دهیم. در نتیجه در صورتی که تست آن خوب بود نرم‌افزار خریداری می‌شود.

System Test:

محصول را به محیط واقعی می‌بریم. آیا درست کار می‌کند یا نه. چندتا باید انجام داد:

11 recovery test: عمداً سیستم را در مرحله‌ای قرار می‌دهیم که خراب شود تست کند چقدر طول می‌کشد که درست کند.

12 security test: وقتی سیستم روی سکه است چطور عمل می‌کند؟ چقدر امنیت در آن وجود دارد که داده‌های محرمانه از دست نرود.

13 stress test: افراد ناخودآگاه در یک کاری stress قرار می‌گیرند و ممکن است کاری ناشایسته انجام دهند. مشتری در شرایط استرس را چطور عمل می‌کند. چه خطاهایی ممکن است داشته باشد و چطور کنترل کنیم.

14 performance test: تعداد زیادی درودی می‌دهیم که آیا عملکرد سیستم به درستی کار می‌کند و در ترکیب همه تست‌ها با یکدیگر درست کار می‌کند.

هندسال زدایی: روز موفقیت آمیز بودن یک خطا نگر از خطاست. باید بتوان خطا را گزارش کرد تا بتوانند

Subject:
Year. Month. Date. ( )

عامل بوجود آمدن خطا را پیدا کنید.

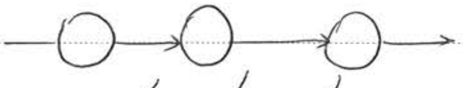
روش های تست:

۱- black box : مربوط به تستی می‌شود که از جزئیات ساخت آن آگاهی نداریم. مثلاً تست لامپ به برق زده و روشن شدن آن را چک می‌کنیم

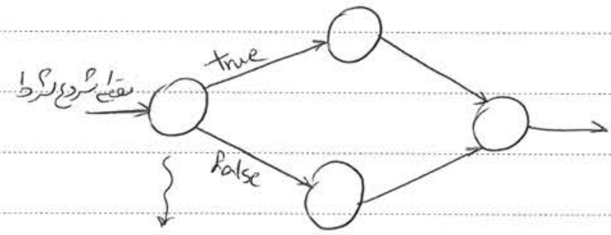
۲- white box : قسمتی از جزئیات مشخص است. معمولاً هدف این است زیرا می‌توانیم در این تحلیل هایی بکنیم که گستره اطراف آن بیشتر از خطا باشد.

تست black box تست سناریو است. کاری که حجم سفید نمی‌تواند انجام دهد این است که نمی‌تواند سناریو چگونه رفتار کند

basis path test : یک برنامه به یک گراف تبدیل می‌شود و هر مسیرها را بتوان از آن رد شویم. برای تبدیل آن از flow graph notation استفاده می‌کنیم هر دستورالعمل با دایره نشان داده می‌شود.

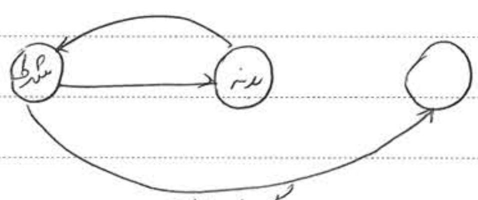


می‌توان همه را با یکی تبدیل کرد زیرا در کل انتها در نهایت یک خروجی داریم و به دنبال پیدا کردن مسیرها هستیم.



دستور شرطی را این گونه نشان می‌دهیم

while :

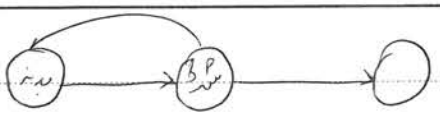


اگر نشود انتها می‌رود.

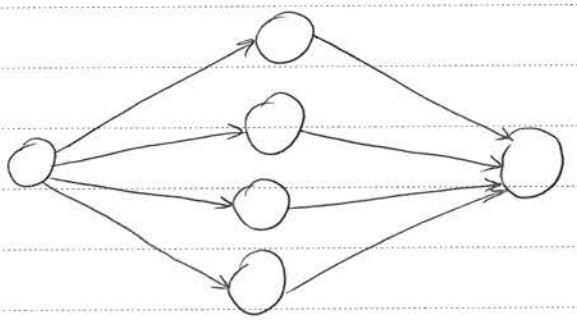
هر دستور باید یک نقطه انتهایی داشته باشد.

Subject:
Year. Month. Date. ( )

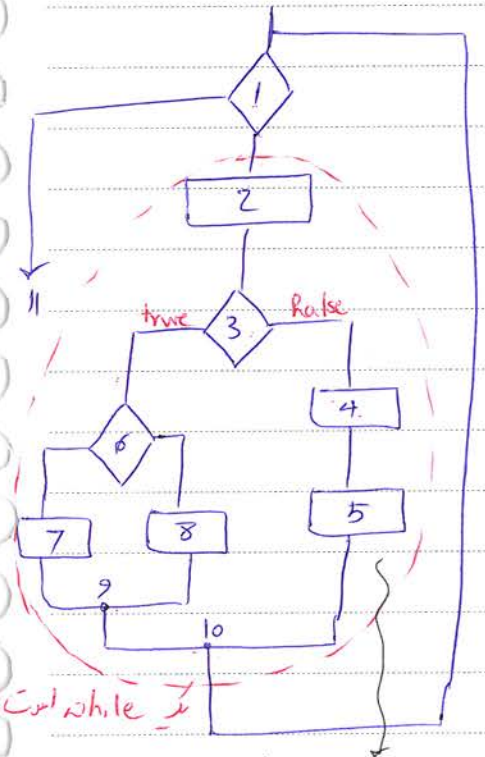
until:



switch case:

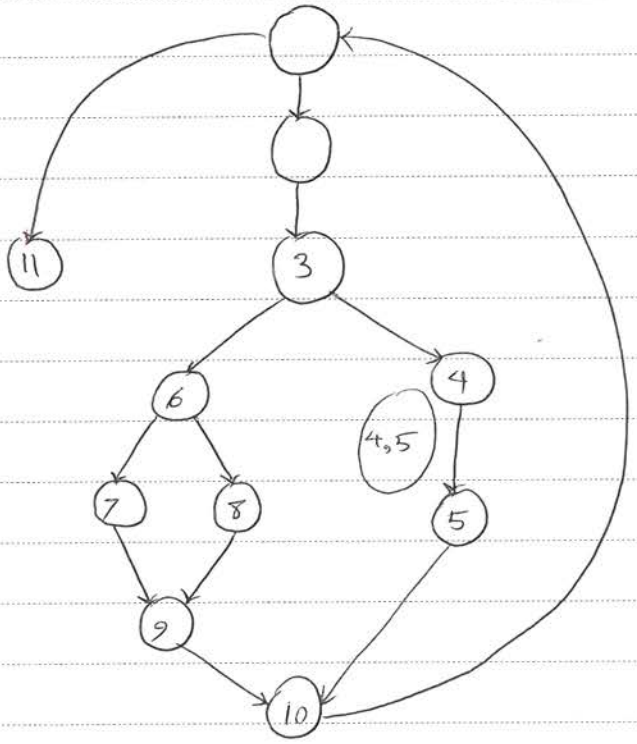


می خواهیم فلوچارت را به یک Flow graph تبدیل کنیم. صرفاً برای هر component این کار را unit test در انجام می دهیم. برای هر حالت دیگر به صورت همزمان در نظر گرفته می شود.



یک while است.

اگر 4 و 5 نباشد فقط یک edge در Flow graph از 3 به 10 می کشیم

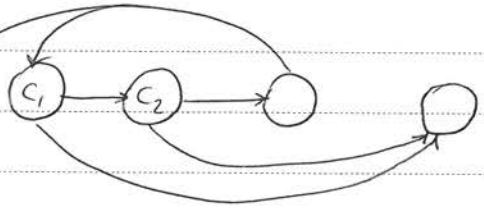


در این حالت فقط انشعابات را می بینیم. در صورتی که یک سورس کد داشته باشیم Tevel گذاری کرده و آن را کشیده شرط های مرکب که and دارد condition می شود.

P4PCO

Subject:
Year. Month. Date. ( )

while (C_1 & C_2)



می‌خواهیم تعداد مسیرهای پایه‌ای را پیدا کنیم. برای این تعداد cyclomatic complexity می‌گوییم.

تعداد شرط‌ها + 1 : $3+1=4$

تعداد حوزه‌هایی که ایجاد می‌شود ← face ← 4

$E-N+2 \quad : \quad 11-9+2=4$

از طریق فاکتورین‌ها می‌توان cyclomatic complexity را بدست آورد.

بدست آوردن path ها:

1) 1 - 11
2) 1 - 2,3 - 4,5 - 10 - 1 - 11
3) 1 - 2,3 - 6 - 7 - 9 - 10 - 1 - 11
4) 1 - 2,3 - 6 - 8 - 9 - 10 - 1 - 11

4 تا مسیر

حال باید test case درآوریم که همه این مسیرها را طی کند.

برای 4 face تا مسیر است.

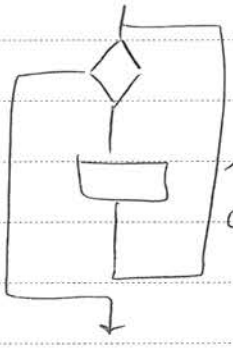
- C_1 = false
- C_1 = true , C_3 = false
- C_1 = true , C_3 = true C_6 = false
- C_1 = true , C_3 = true C_4 = true

Subject:
Year. Month. Date. ( )

اینجا ترکیب همه شرط ها نیست. زیرا برای c_3 = false دیگر true نخواهیم داشت. ست جامع امکان پذیر نیست. می توان برای بال ها هزینه تعیین کنیم و مشخص کنیم هزینه برای هر ست حقیر می تواند باشد و زمان پاسخ را بدست آورد.

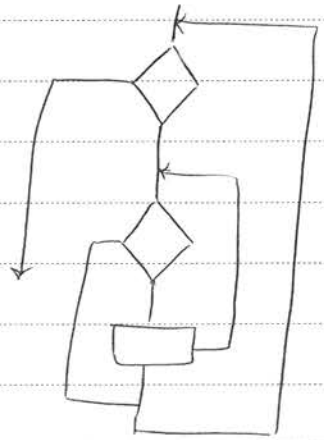
8 control structure test

حلقه ها و شرط ها را ست می کنیم و



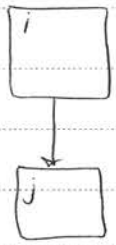
- جوری ست می کنیم که صفر تا n بار اجرا شود و حلقه ها را برای همه حالت ها و تقاطع ها بررسی می کنیم.
- حلقه ها می توانند تودرتو باشند.

حلقه ها می توانند در کنار یکدیگر قرار می گیرند.



- حلقه ها می توانند تودرتو باشند

از حلقه داخلی شروع می کنیم و سپس با حلقه بالایی ترکیب می کنیم.



- یکبار این زنگ می زنیم و داخلی را اجرا می کنیم و سپس به ازای مقادیر دیگر i بررسی می شود.

j در حلقه بالایی مقداردهی می شود.

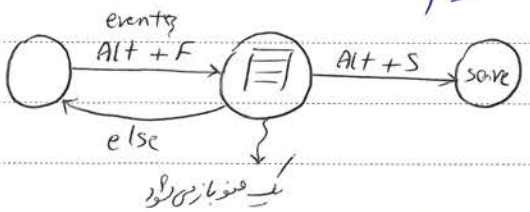
شرط های goto را با استاندارد structure A. تبدیل می کنیم.

P4PCO

Subject:
Year. Month. Date. ( )

? black box

گراف حالات یک Automata برای آن می‌کشیم.



کسی تست می‌کند سعی می‌کند مطابق با آن جلو رود. اگر نتواند event دیگری تولید کند پس error دارد و اگر تولید نمی‌کند آیا همین event ها با این صورت باید اجرا شود.

Subject:
Year. Month. Date. ( )