Subject :
Year . Month . Date . ( )

مهندسی نرم افزار

۲

سنه استاد مربوط : دکتر لیلان محمدی نسب

Subject:

Year. Month. Date. ( )

Subject:
Year. 1 Month. Date. ( )

مراجع:

- R. pressman, SE, of practioner's approach, 7th edition 2010
- I. Sommerville, SE, 8th edition

روش ارزیابی:

- ۲۰٪ امتحان میان ترم
- ۴۰٪ امتحان پایان ترم
- ۲۰٪ پروژه
- ۵٪ تمرین
- ۵٪ فعالیت کلاسی
- ۱۰٪ نمره اضافه

فهرست مطالب:

بخش اول: مهندسی نیازمندی‌ها
- تعریف نیازمندی
- طبقه‌بندی نیازمندی
- مراحل مهندسی نیازمندی
- تکنیک مهندسی نیازمندی

بخش دوم: مدیریت پروژه‌های نرم‌افزاری
- مفاهیم مدیریت پروژه
- اندازه‌گیری در نرم‌افزار
- تخمین پروژه‌های نرم‌افزاری
- زمانبندی پروژه‌های نرم‌افزاری
- مدیریت ریسک

PqPCO

Subject:
Year. Month. Date. ( )

بخش سوم: مدل های فرایندی توسعه نرم افزار
- بررسی انواع مدل های فرایندی
- ذهنیت Agile
- RUP

بخش چهارم: تحلیل و طراحی نرم افزار
- تحلیل نیازمندی ها
- مفاهیم طراحی شی گرا
- معماری نرم افزار
- زبان UML

بخش پنجم: مدیریت کیفیت
- مفاهیم کیفیت نرم افزار
- تکنیک های بازبینی
- کنترل و تضمین کیفیت
- تست نرم افزار
- مدیریت پیکربندی نرم افزار (SCM)

Subject:
Year. Month. Date. ( )

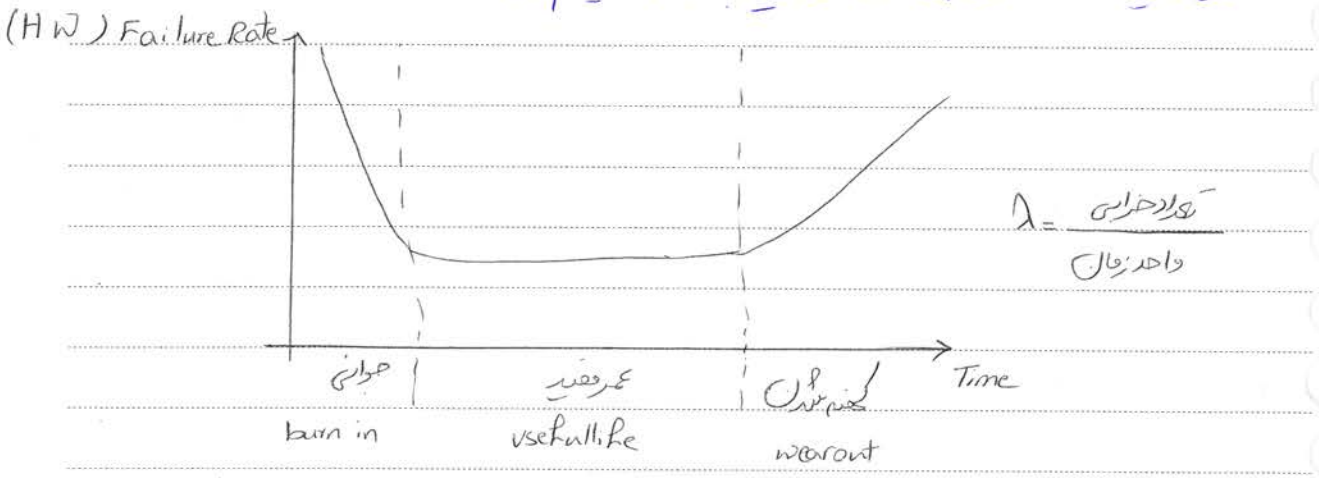
مهندسی نرم افزار؟

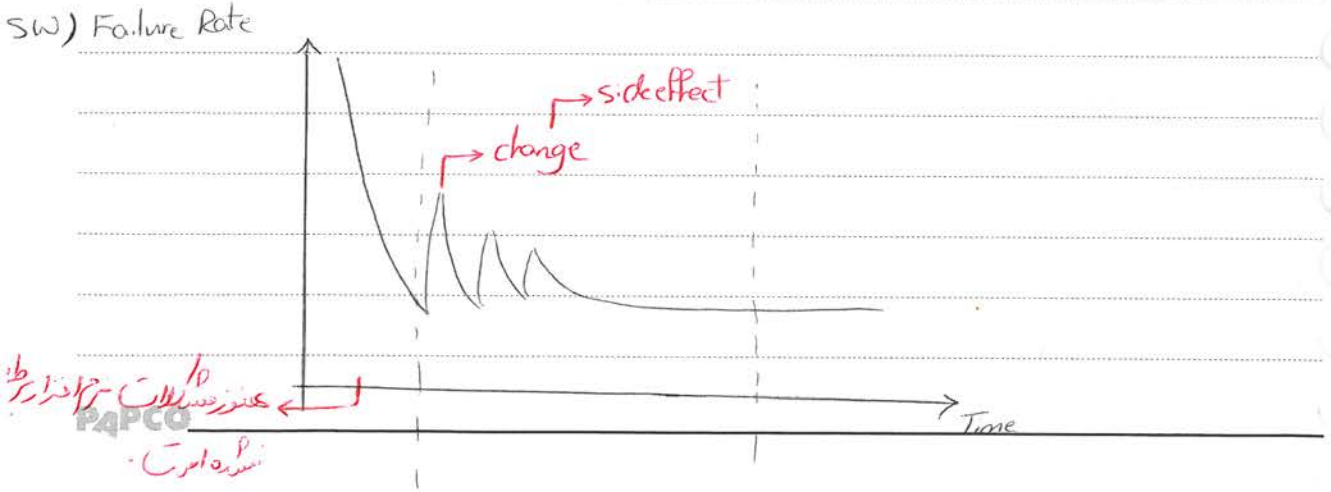
مهندسی ← اندازه گیری

نرم افزار ← کد اجرایی
- مستندات
- داده ها

بحث مهندسی نرم افزار در سال 1968 در کنفرانس NATO تحت عنوان software crisis مطرح شده است. در آن زمان مهندسی نرم افزار به صورت code & fix سریع اما با کیفیت پایین انجام می شد.

مشاهیم سخت افزار با نرم افزار؟
نمودار یک سخت افزار طی چند شغل زیر به سه مرحله تقسیم می شود





3

Subject:
Year. Month. Date. ( )

پیش از اینکه دوره استفاده از نرم افزار شروع شود به علت تغییرات مطرح شده باید دوره عوض شود که خود این تغییرات باعث بوجود آمدن side effect می شود و این امر دوباره هزینه های آن را بر طرف می شود.

* نرم افزار کهنه نمی شود. در شکل طبق شکل در مرحله سوم هزینه ها بالا است، در واقع با نرم افزار تنها به علت تغییر نیازمندی ها کنار گذاشته می شود.

* قطعه دیگری که در صنعت افزار وجود دارد این است که برای هر قطعه سخت افزاری یک spare وجود دارد که در صورتی که یکی از قطعات خراب شود جایگزین آن وجود دارد.

* Failure: از نظر کاربر برنامه زمانی انتظار آن را آورده نشود. در واقع هر وقت سیستم نتواند مورد انتظار آن را آورده کند Failure اتفاق افتاده است. specification

انواع نرم افزار:

۱- نرم افزار سیستمی: نرم افزاری که هدف آن ارائه سرویس به سایر نرم افزارهاست. مانند OS، DBMS، ....

۲- نرم افزار کاربردی (APP): هدف آن پاسخ گویی به نیازهای خاص کسب و کار است. این نرم افزارها Data oriented هستند

۳- نرم افزارهای مهندسی: در کارهای علمی استفاده می شود: CAD، MATLAB، .... این نرم افزارها process oriented هستند

۴- نرم افزارهای نهفته: (embedded) کارکرد سیستم با نرم افزار درونی آن یکسان است و عملکرد نرم افزار هسته اصلی سیستم است. هدف سیستم چیز دیگری است و این نرم افزار فقط به تحقق آن هدف کمک می کند.

P4PCO

4

Subject:
Year. Month. Date. ( )

۵- product_line software : برای هدف general تولید شده است و قابلیت آن با نرم افزار کاربردی این است که برای کاربردی برای یک bussines خاص ایجاد شده است درحالی که هدف از این نوع نرم افزار ارائه یک قابلیت مشخص به کاربران متفاوت است ← word

۶- Web Application : زیر مجموعه ای از software Application ها است.

۷- نرم افزارهای هوش مصنوعی: مانند Image processing ، speach_to_tent
در واقع از دیگر هوش مصنوعی برای تولید نرم افزار استفاده می کنند.

نوترندهای جدید:

- open_world computing (pervasive computing) : نرم افزار با استفاده هوش مصنوعی و به صورت اتوماتیک کار می کنند.

- Net sourcing : استفاده از سرویس های دیگران (SOA)

open source:

کد منبع را در اختیار دیگران قرار می دهد. در مقابل commercial (closed) قرار دارد

۱) copyleft strict (باید در مقابل استفاده از کد دیگران، کارهایی که خود انجام داده ای را در اختیار دیگران قرار دهید)

۲) without copyleft (می تواند کد خود را در اختیار دیگران قرار ندهد) برای open source برای نیاز commercial استفاده می شود)

۳) limited copyleft (برای قسمت هایی که دیگران تایپ کرده اند نمی تواند محدودیتی ایجاد کند)

license های موجود عبارتند از:

- GNU general license (GPL)
- Apache software license ← دیگر فقط نامی از من برده شود.
- GNU lesser general public license (LGPL)

PqPCO

Subject:
Year. Month. Date. ( )

Legacy software: نرم‌افزار میراثی

این نرم‌افزارها باید دو ویژگی اصلی داشته باشند:

۱- قدیمی بودن: زمان طولانی از توسعه آن گذشته است - تکنولوژی قدیمی ... ← طول عمر زیاد

۲- برای کسب و کار کلیدی هستند یعنی امکان حذف یا توقف عملکرد آن وجود ندارد.

زمانی که یک سیستم legacy مواجه می‌شویم باید با آن به یکی از روش‌های زیر برخورد شود:

- اگر سیستم کار می‌کند و نیازی را پاسخ می‌دهد هیچ کاری انجام نشود.
- در مورد این نیازها تصمیم است:
  - پاسخگویی به نیازهای تکنولوژی جدید
  - پاسخگویی به نیازهای کسب و کار جدید
  - پاسخگویی نیازهای ارتباط با نرم‌افزارهای جدید

برای استفاده و همکاری نرم‌افزار legacy با نرم‌افزار جدید دو روش Reverse Eng یا Re Eng وجود دارد که از طریق مهندسی معکوس ابتدا نیازهای سیستم را شناسایی کرده و متناسب با آن نرم‌افزار را تغییر می‌دهد.

اسرائیل حمله Dos و web APP

Subject:
Year. Month. Date. ( )

بخش اول :

مهندسی نیازمندی ها: (Requirment Engineer)

مسئله : مواردی که برای آن مهندسی نیازمندی ها و ایجاد یک نرم افزار جدید می باشیم عبارت اند از:

۱- مشکلات (problem) : شرایط نامطلوب که مانع دستیابی سازمان به اهداف در برنامه هایش می شود

۲- فرصت ها (opertunity) : از طریق فرصت های موجود در وضعیت می توان سیستم را ترقی داد

۳- رهنمود (Dirctives) : مانند پورتال آموزشی که از طرف وزارت علوم دستور داده می شود که هر دانشگاه ها از طریق پورتال آموزشی انجام شود

دامنه مسئله (problem Domain)	دامنه راهکار (solution Domain)
مشکل خالی بودن کارت ها از پول	- پارتیشن خط تولید
	- چک کردن درب
	- Xray
	- ...

برای بررسی یک مسئله بهترین راه این است که دامنه مسئله را از دامنه راهکار جدا کنیم و سعی کنیم برای حل یک مسئله راهکارهای مختلفی ارائه داده و بهترین انتخاب شود زیرا راهکارها متناسب با وضعیت فعلی حل کننده است

مفهوم نیازمندی : یک ویژگی یا قابلیت که باید با توسعه سیستم برآورده شود

دسته بندی ذی نفع (stack holder) :

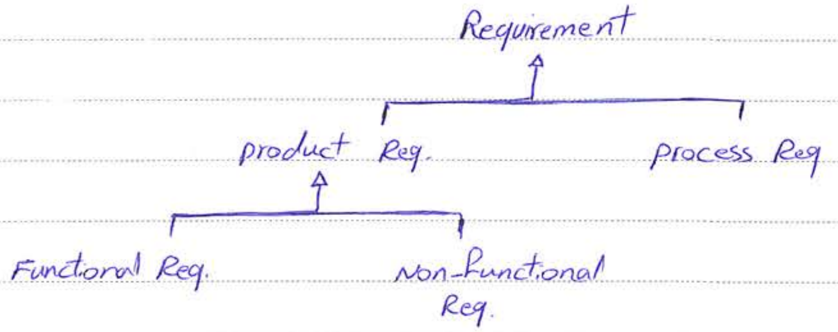
- صاحب سیستم (sponsor) ⟵ وزارت علوم
- مشتری (customer) ⟵ دانشگاه

PAPCO

Subject:
Year. Month. Date. ( )

- کاربران سیستم (users) ← مثلاً دانشجو
- کاربران انتهای سیستم (endusers) ← مثلاً اساتید دانشگاه
- تیم توسعه نرم افزار (Developers)

انواع نیازمندی ها :



در process Req تعریف مسئله به صورت تعیین محدودیت ها و تکنولوژی های مورد استفاده در هنگام ایجاد نرم افزار است . (فرایندهای تولید نرم افزار مشخص می شود)
ولی در product Req. نیازمندی های مستقیم سیستم و آن چیزی که در نهایت نیاز داریم مطرح می شود.

Functional Req نیازمندی هاست که رفتار مشخصی دارد و ۰-۱ هستند ولی Non-Functional ها قیودی هستند که برای تعریف نرم افزار به کار می رود و نیازمندی های کیفی هستند

نیازمندی های غیرکارکردی (NFR) : قیودی را بر طراحی اعمال می کنند و گزینه های طراحی را محدود می کنند.
- مربوط به محصول : ویژگی های خود محصول را تعیین می کنند.
- سیاست های سازمان (مربوط به سازمان) ← opensource بودن سیاست سازمان است.
- نیازهای خارجی (قیود اقتصادی، قانونی و همکاری با سایر سیستم ها) ← نرم افزار قیمت کمتر از 0 داشته باشد.

PAPCO

Subject:
Year. Month. Date. ( )

مدل FURPS+ : یک مدل برای طبقه‌بندی نیازمندی‌های سیستم است
(F — NFR)

F → Functionality (Functional Req)

U → Usability
- سهولت استفاده
- سهولت یادگیری
- قابلیت درک
- راهنمای کاربر

R → Reliability (قابلیت اطمینان) → Availability (دسترس‌پذیری)
↓
احتمال در دسترس بودن سیستم هنگام درخواست کاربر

P → performance

S → supportability

- Reliability : یک سیستم در زمان t که با R(t) نمایش داده می‌شود، احتمال آن است که سیستم در بازهٔ [0, t] بدون خرابی کار کرده باشد.

[0, t]
0 —————— t

- Availability :

متوسط زمان تا زمانی که Failure بوجود آید

$$A_\infty = \frac{MTTF}{MTTF + MTTR}$$

۱ > عدد

مدت زمان از لحظهٔ خرابی تا زمان repair و درست شدن آن

مهم‌ترین ویژگی سرویس‌های معروف مانند google این است که همیشه در دسترس است و Availability بالایی دارد.

Availability	Down time
90%	36.5 day/year
99%	3.65 day/year
99.9%	8.76 hour/year
99.99%	52 minute/year
99.999%	5 minute/year
99.9999%	31 second/year

Subject:
Year. Month. Date. ( )

- کارایی (performance) :
- Response time : در حقیقت این عدد در پردازش اهمیتی مدت زمانی طول بکشد. دو جزء هم در اینجا وجود دارد یکی تأخیر خود سیستم و دیگری latency یعنی مدت زمانی که سرویس مثل اینترنت تأخیر ایجاد می‌کند.
- Throuput (توان عملیاتی) : تعداد تراکنش‌های قابل انجام توسط سیستم در واحد زمان
- Resource : منابع مورد نیاز و نحوه استفاده کمتر از آن
- Supportability :
  - changability ← قابلیت تغییر (سیستم باید بتواند با حداقل تغییر بدون تغییر در کد برنامه تغییر کند)
  - Ease of installation
  - compatibility

در واقع یعنی تغییر سیستم عوارضی و پشتیبانی سیستم.

+ (فیتور) :

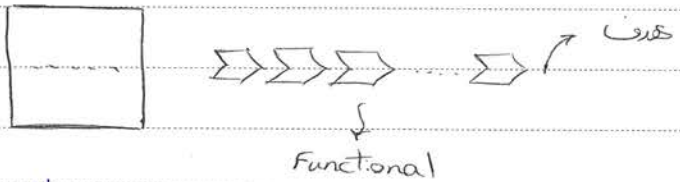
- نیازمندی‌های طراحی (Design Req) : استفاده از پایگاه داده رابطه‌ای
- نیازمندی‌های پیاده‌سازی (Implementation Req) : استفاده از زبانی خاص
- نیازمندی‌های رابط interface req : منظور واسطین سیستم‌ها و قابل برقراری ارتباط با سایر سیستم‌ها را داشته باشد
- نیازمندی‌های فیزیکی (physical Req) : قید برای سخت‌افزار ← توان محاسباتی

stakeholder : کسی که به صورت مستقیم و غیرمستقیم از سیستم مورد توسعه سود می برد.

Subject:
Year. Month. Date. ( )

نکته: معمولاً security به عنوان نیازمندی های Functional مطرح می شود، زیرا قواعدی کاملاً مشخص است.

Business process: در هنگام انجام یک process مراحلی که در workflow مورد نیاز است تا سرویس مورد نظر فراهم شود را business process می گویند که یک نیازمندی Functional است.



در صورتی که هر یک از مراحل پشت سر هم اتفاق بیفتد در نهایت value به بار می آورند و در صورتی که این مراحل کامل اجرا نشود، ارزش مورد نظر بدست نمی رسیم.

نکته: مفهوم Engineering با value آمیخته شده است یعنی خود به عنوان مهندس دانش یا ایده ای داریم و آن را به کار می بندیم تا ارزش را برای جامعه ایجاد کنیم.

کاربردهای Web App:

یک web App مانند پورتال آموزشی، یا سایت بانکی است. ویژگی های این نوع App ها عبارتند از:

۱- تکنیک شبکه (وب، اینترنت) ⟵ TCP / IP , HTTP, ...

۲- همروندی: تعداد افرادی که به صورت همزمان از سیستم استفاده می کنند زیاد است و باید قابلیت همزمان کار کردن را داشته باشد.

۳- بار غیرقابل پیش بینی: در یک زمان ممکن است ۱ کاربر و در یک زمان دیگر 500 کاربر وارد شوند.

۴- کارایی (performance): باید با هر شرایط قبلی کارایی مورد نیاز سیستم قابل تنظیم باشد.

PAPCO

Subject:
Year. Month. Date. ( )

۵- دسترسی پذیری : 24*7*365
یعنی در تمام اوقات باید در دسترس باشد

6- مبتنی بر داده : مانند email، پورتال آموزشی و ... مجموعه‌ای داده هستند.

۷- حساس به محتوا : در صورتی که محتوا تغییر کند web App تغییر می‌کند.

۸- تکامل پیوسته : زمانی که از server استفاده می‌شود برنامه‌های بروی client نیست و به راحتی می‌توان با تغییر برنامه، قابلیت‌های سیستم را تغییر داد.

9- فوریت : بحث Time-to-market ← زمان زیادی برای توسعه نداریم.

۱۰- امنیت (خیلی مهم) : برای یک نرم‌افزار Desktop امنیت سیستم به صورت فیزیکی است ولی در web App چون تحت وب و شبکه عمل می‌کند امنیت حساسیت بیشتری دارد.

۱۱- زیبا شناسی (UI)

هسته آن چیزی که باعث انجام یک پروژه می‌شود این است که یک وضعیت As-Is نامطلوب وجود دارد و می‌خواهیم سیستم را به وضعیت مطلوب ببریم.

دامنه مسئله : (problem Domain)
- هدف توجه به نیازهای ذی‌نفعان است.
- دامنه‌ای است که سیستم در آن استفاده می‌شود.
- نیازمندی‌ها بدون توجه به مسائل فنی و راه‌حل‌ها کاملاً بررسی می‌شود.
- یک سری ایده و نظرات داریم.

PAPCO

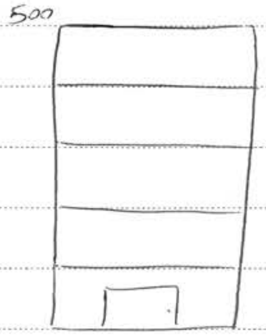
Subject:
Year. Month. Date. ( )

در نتیجه در ابتدا بدون توجه به مسائل فنی فقط نیازهای سیستم را مطرح می‌کنیم زیرا در صورتی که مسائل فنی نیز در نظر گرفته شوند راهکارها محدود به همان‌ها می‌شود.

دامنه راهکار (solution Domain):

- دامنه‌ای که مهندسان در آن از خلاقیت خود برای حل مسائل استفاده می‌کنند
- در دامنه راهکار یک مجموعه نیازمندی مشخص داریم

مثال: در یک ساختمان پنج طبقه ازدحام جمعیت در هنگام خروج افراد در پایان ساعت کاری ایجاد می‌شود بهترین راهکار چیست؟



حل مسئله ازدحام جمعیت:
- اضافه کردن درب جدید
- تغییر ساعت خروج
- استفاده از نردبان
- نصب آینه در دو طرف

تضاد در نیازمندی‌ها (conflict):

عموماً از نیازهای سیستم تضاد زیادی بین نیازهاست مثلاً می‌خواهیم سیستم کاملاً secure باشد ولی در کنار آن performance بالایی داشته باشد این دو نیاز با هم برآورده نمی‌شود و باید یک trade off داشته باشیم.

توهمات در نرم‌افزار (myth):

* مدیران ← هنگامی که از برنامه زمان‌بندی عقب هستیم با افزودن نیروی جدید مسئله حل می‌شود
* مشتری ← تنها یک توصیف کلی می‌توان نرم‌افزار را توصیف کرد

PAPCO

Subject :
Year . Month . Date . ( )

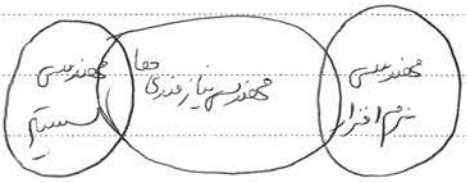
* تا وقتی برنامه اجرا نشود نمی‌توان درمورد کیفیت آن نظر داد.

* مهندسی نرم‌افزار صرفاً باعث تولید مستندات حجیم و بی‌فایده می‌شود.

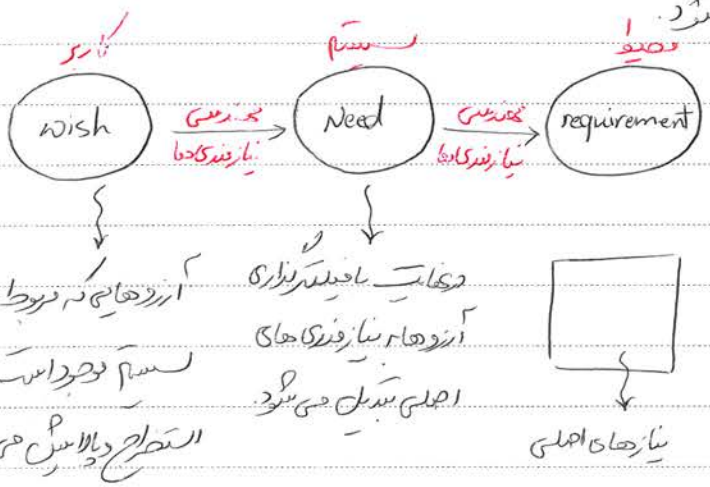
↳ هدف مهندسی نرم‌افزار تولید مستندات نیست، هدف تولید نرم‌افزار باکیفیت است.

مهندسی نیازمندی‌ها (Requirements Eng.) :

یک رویکرد نظام‌مند برای استخراج، مستندسازی، تحلیل، اعتبارسنجی و مدیریت نیازمندی‌ها است.



برای ارتباط نرم‌افزار ایجاد شده و سیستم مربوطه نیاز است نه مهندسی نیازمندی‌ها



چرا مهندسی نیازمندی‌ها :

اگر نیازمندی‌ها مدیریت نشود در اغلب موارد به یک راه‌حل درست برای مسئله نادرست می‌رسیم.
یعنی مسئله و نیازمندی‌ها به درستی شناخته نمی‌شوند یعنی راه‌حل درست است ولی نیازهای سیستم را برطرف نمی‌کند

Subject:
Year. Month. Date. ( )

مراحلی که برای مهندسی نیازمندی ها باید انجام شود عبارت اند از:

1) استخراج (Elicitation): هدف استخراج نیازمندی ها از طریق تعامل با ذینفعان و کاربران است. در بسیاری از مواقع نیازمندی ها مخفی است و نیاز است که آنها را استخراج کنیم

2) تحلیل و مدلسازی نیازمندی ها: با تحلیل نیازمندی ها conflict ها و یا نیازهای تکراری که قابل استخراج است

3) توصیف نیازمندی ها (Requirement specification): در یک قالب مشخص نیازمندی ها توصیف شود

SRS → software requirement specification

4) صحت سنجی و اعتبارسنجی نیازمندی ها (verification & validation):
[V & V]

verification ← آیا سیستم به درستی تهیه شده است؟ ← خود سیستم درست است

validation ← آیا سیستم درستی تهیه شده است؟ ← آیا همان سیستمی است که ما می خواهیم

5) مدیریت نیازمندی ها (Requirement management):

- تغییر (change): نیازمندی به صورت مستقیم با سایر در ارتباط است و باید دقت بررسی شود که آیا این نیاز تغییر کرده است یا نه. در بعضی از حالات ها یک تغییر بر روی سایر نیازها نیز تأثیر می گذارد در نتیجه از چه کدهایی کرد، مدیریت نیازمندی انجام می شود عبارت است از:

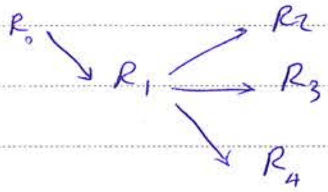
- change management
- configuration management
- Tracability

set of req { $R_1, R_2, \ldots, R_n$ } (change)

Subject:
Year. Month. Date. ( )

قابلیت ردیابی (Tracability):

پیگیری نیازمندی‌های مختلف به دو روش قابل انجام است:



- Backward Tracability: این نیاز از کجا آمده است.

- Forward Tracability: این نیاز چه نیازهای دیگری را به دنبال خود می‌آورد.

تکنیک‌های جمع‌آوری نیازمندی‌ها:

متناسب با پروژه در هر مرحله می‌توان از تکنیک‌های موجود استفاده کرد:

۱) استخراج نیازمندی‌ها:

۱-۱- مطالعه مستندات (Doc study)

۱-۲- مشاهده (observation): با مشاهده فرایندهای موجود در بررسی BP کمک می‌کند تا اجزای سازمان کشف شود.

۱-۳- مصاحبه (interview): روش‌های مختلفی وجود دارد در روش close سوالات به صورت غیرمستقیم مطرح می‌شوند اطلاعات مورد نظر کشف می‌شود و در روش open مجموعه‌ای از سوالات تهیه شده و یکی یکی سوال می‌شود

۱-۴- تهیه پرسشنامه (Questionaries): در مواقعی که سیستم تعداد و تنوع ذی‌نفعان زیاد است به جای مصاحبه از این روش استفاده می‌شود

۱-۵- نمونه‌سازی (prototyping): نمونه‌های آماده می‌شوند به سادگی در دسترس هستند

۱-۵-۱- نمونه‌سازی تکاملی (Evalutonary):

در ادامه پروژه همین نمونه تکمیل شده و پروژه اصلی را می‌سازد.

PAPCO

Subject:
Year. Month. Date. ( )

۱-۵-۲- نمونه سازی دورانداختنی (Throw-away): تعاملات کاربا سیستم نیازمندی‌ها به کار می‌رود و پس از آن دیگر از آن استفاده نمی‌شود.

۱-۶- JRP (joint requirment planning): زیرمجموعه JAD است. جلسه‌ای نشستن که در هنگام توسعه نرم‌افزار با مشتری‌ها و کاربران دارند به منظور درک نیازمندی‌ها استفاده می‌شود. اولین مرحله JAD (joint App develpment) همان JRP است.

۱-۷- یورش فکری (Brainstorming): برای پروژه‌هایی که مستندات ساختار و شبیه یا نشانه‌هایی است می‌توان با برگزاری جلسه‌هایی جهت گرفتن همه نظرات بدست آورد.

۱-۸- مورد کاربرد (Use case): این دیاگرام در مهندسی نیازمندی‌ها در مرحله اول کاربرد زیادی دارد.

۱-۹- ایفای نقش (Role playing): خود به شرکت رفته و با انجام آن کارها نیاز سیستم را پیدا کرده تا نحوه عملکرد سیستم را درک کنیم.

12 مدل سازی و تحلیل:

انواع مدل‌ها:

- Iconic: مدل شبیه سیستم و مقیاس کوچک
- Analog: مدل از نظر رفتاری شبیه سیستم مثل چارت سازمانی
- ریاضی (کمی): بیان سیستم از طریق روابط ریاضی.

PAPCO

Subject :
Year . Month . Date . ( )

13 V&V: استفاده از checklist که چه چیزهایی باید چک شود

14 توصیف نیازمندی‌ها:

- روش استفاده از قالب (Boiler Plate): متدهای هان template است و برای یافتن نیازمندی‌ها به صورت زیر طبیعی عمل می‌کنیم که باید قالب مشخص داشته باشد:

* حوزه مربوط به دامنه فنی:

- (نوع نیازمندی) باید امکان (قابلیت) را داشته باشد

- (نوع نیازمندی) باید امکان انجام (قابلیت) را با (قیود) داشته باشد

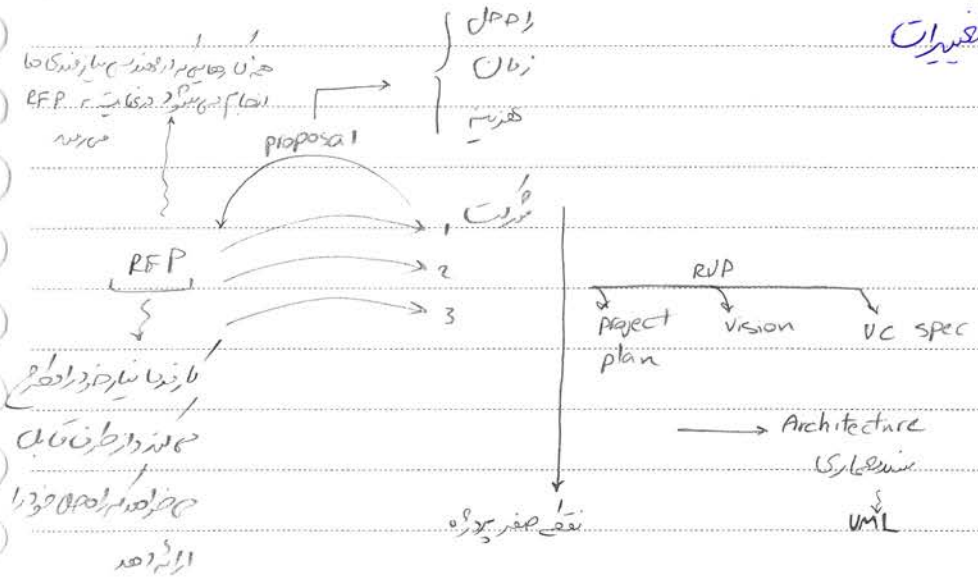
* حوزه مربوط به دامنه اجرا:

- (سیستم) باید امکان انجام (عملکرد) (مفعول) در (کارایی) (دامنه اندازه‌گیری) را داشته باشد

15 مدیریت نیازمندی‌ها:

- مدیریت تغییرات

- ردیابی



P4PCO

Subject:
Year. Month. Date. ( )

ویژگی های مطلوب برای سند توصیف نیازمندی ها:

۱- صحیح: نیازمندی ها به درستی بیان شوند.

۲- غیرمبهم: تا جایی که امکان دارد ابهامات را کم کنیم. یکی از روش هایی که برای رفع آن کمک می کند درست کردن glossary است.

۳- کامل: همه نیازمندی های سیستم را شامل شود.

۴- قابل تست: حداقل یک روش مشاهده با هزینه مناسب برای بررسی قابل پذیرش بودن نیازمندی وجود داشته باشد.

۵- سازگار بودن:
• استفاده از واژگان یکسان برای یک شیء
• خصوصیات اشیاء: گزارشات شماره دارند → عدد صحیح / → عدد + حروف } باید در همه جا یکسان باشد.
• خطاهای موقتی یا منطقی: "الف بعد دنبال ب انجام می شود."

۶- مرتبط باشد: حتما باید دقت شود تا در سند نیازمندی چیزهایی نوشته شود که مربوط می باشند. Req. درباره what صحبت می کند ولی هیچ وقت درمورد How حرفی نمی زنند.

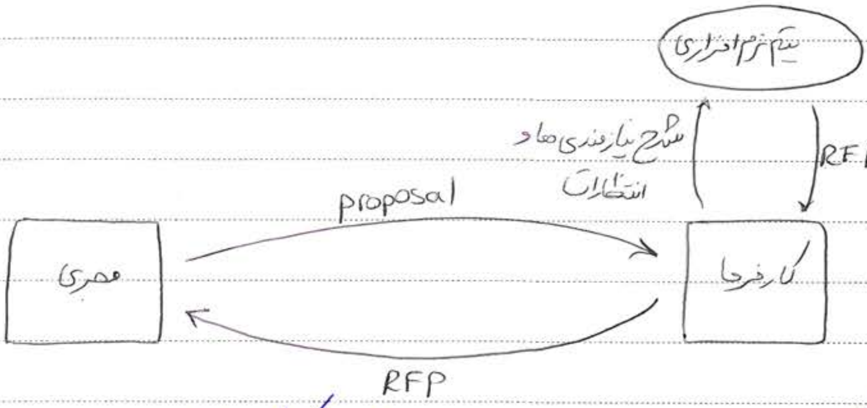
spec → program code
spec ↓ what را مطرح می کند
program code ↓ How را مشخص می کند

۷- قابل تغییر و ردیابی باشد: باید سند گونه ای آماده شود که برای تغییر یک قسمت همه قسمت ها تغییر نکنند. روش های موجود عبارتند از:
• ۱- برچسب گذاری نیازمندی ها: از طریق کدها نیازمندی ها و ارتباط آنها راحت تر می شود.

PAPCO

Subject:
Year. Month. Date. ( )

RFP (سند درخواست ارائه پیشنهاد):



برای انجام یک پروژه قدم اول مطالعه سند RFP است که تولیدکنندگان تقاضای انجام پروژه را دارند. در واقع تولیدکنندگان اصلی آن مشتریان نرم افزار هستند که به سفارش صاحبان اصلی RFP را تولید می کنند.

تیم نرم افزاری با استفاده از روش های مهندسی نیازمندی ها اقدام به استخراج و ساماندهی نیازمندی های کارفرما می کنند.

محتوای RFP به طور کلی ۲ قسمت اصلی دارد:
- بخش فنی
- بخش شروط، قوانین و مقررات و محدودیت های قراردادی

(هر یک از بخش ها و محتویات آن به تفصیل در اسلایدها بیان شده است.)

مناقصه

فرایندی است رقابتی برای تأمین کیفیت مورد نظر که در آن تعهدات موضوع معامله به مناقصه گری که کمترین پیشنهاد با بهترین قیمت مناسب را ارائه کرده باشد واگذار می شود. سند RFP جزو اسناد مناقصه محسوب می شود.

مناقصه از نظر نوع بررسی:
- یک مرحله ای: نیازی به ارزیابی فنی نیست و پاکت های پیشنهاد مناقصه گران در یک جلسه باز شده و برنده مشخص می شود.
- دو مرحله ای: ابتدا بررسی فنی و ارزیابی پیشنهادها انجام شده و نتیجه ارزیابی گزارش می شود.

مناقصه از نظر روش دعوت:
- عمومی: فراخوان مناقصه از طریق آگهی عمومی به اطلاع مناقصه گران می رسد.
- محدود: تنها برای افرادی که صلاحیت دارند دعوتنامه ارسال می شود.

(جزئیات در اسلایدها)

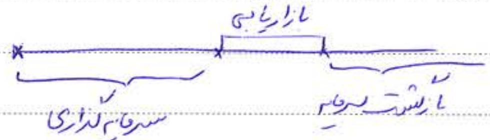
Subject:
Year. Month. Date. ( )

طرح تجاری (Bussiness plan) :

چه محصولی را با چه قیمتی، به چه مشتری، در چه دوره بازگشت سرمایه‌ای می‌توان ایجاد کرد.
چرا مشتری محصول شما را خریداری می‌کند؟
باید قبل از اینکه وارد یک پروژه شویم مدیریت رقابتی خود را بسنجیم، محصولات موجود در بازار را بدست آورده و ابعاد مشکل را مشخص کنیم

ROI (Return of Investment) :

برای رسیدن به نقطه‌ای که محصول را ارائه دهیم از چه زمانی محصولات به فروش می‌رسد تا اصل سرمایه برمی‌گردد



امکان سنجی (Feasibility) :

آیا این امکان وجود دارد که این پروژه انجام شود

۱- امکان سنجی فنی: آیا دانش لازم برای انجام این پروژه وجود دارد یا نه.

۲- امکان سنجی زمانی: اگر محصول دیرتر از یک زمان به بازار ارائه شود دیگر مفید نیست.

۳- امکان سنجی اقتصادی: بر اساس میزان سرمایه‌گذاری لازم برای این پروژه آیا امکان دارد در یک زمان برای بازگشت سرمایه صبر کرد

۴- امکان سنجی قانونی (legal): مجاز بودن نرم‌افزار، مجوز برای ارائه نرم‌افزار داده می‌شود.

Subject:
Year. Month. Date. ( )

بخش دوم

Subject :
Year . Month . Date . ( )

ابزارهای RE :

- IBM Rational Requisitepro

- IBM Rational DOORS

از ابتدای مهندسی نیازمندی ها را می توان وسیله آن بازه سازی کرد.

بخش دوم : مدیریت پروژه های نرم افزاری

پس از تعریف یک پروژه نرم افزاری نیاز به راهنمایی هایی برای مدیریت پروژه در طول اجرای آن است.

مدیریت پروژه های نرم افزاری شامل 4P است :

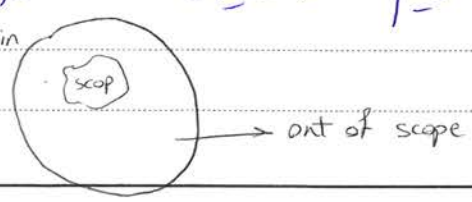
- people
- product
- process
- project

ترتیب این 4 مورد مهم است. در یک نرم افزار همه اجزا هستند و براساس اولویت آنها پروژه انجام می شود. در واقع نرم افزار people oriented است.

product : محصول مد نظر scope باید مشخص باشد.

scope : در محدوده تعیین شده برای Domain فعالیتی که قصد Domain داریم انجام می دهیم.

Domain : فضایی که می خواهیم در آن کار کنیم مثلاً می خواهیم در Domain نرم افزارهای صنعتی کار کنیم



P4PCO

23

Subject:
Year. Month. Date. ( )

process: وظایف و فعالیت‌ها و نحوه چیدمان آنها

project: پروژه تلاشی است موقتی برای ایجاد یک محصول یا ارائه خدمتی یکتا
unique — temporal — endevor

شروع پروژه ←—→ خاتمه پروژه

شروع پروژه: از زمانی که قرارداد بسته می‌شود.

خاتمه پروژه:
→ اهداف پروژه محقق شود.
→ اهداف قابل دستیابی نباشد.

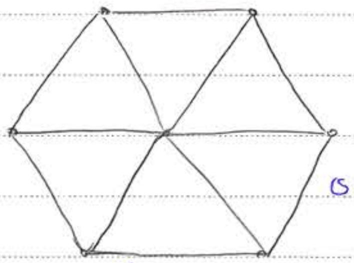
یکتا بودن پروژه به این معناست که ویژگی‌های خاص خود را دارد و حتی اگر یک پروژه مشابه هم ولی در instance متفاوت هستند و لزومی ندارد نتیجه‌های یکسانی داشته باشند.

N people
M task
t team
⇒
- N=M → هر فرد یک task می‌گیرد
- M<N → یعنی داریم که می‌خواهند روی پروژه کار کنند.

ساختار تیم‌های نرم‌افزاری:

① Democratic Decentralized (DD):

ویژگی‌ها:



1) فاقد رهبر دائمی
2) ارتباطات به صورت افقی
3) مناسب برای پروژه‌های کوچک (حدود ۵ الی ۷ نفر) و پروژه‌های تحقیقاتی و پیچیده.

افراد با یکدیگر در ارتباط هستند و بین هر دو نفر حداقل یک مسیر وجود دارد ولی به عنوان رهبر وجود ندارد. این نوع پروژه‌ها در دانشگاه‌ها وجود دارد.

PAPCO

Subject:
Year. Month. Date. ( )

مزایا:

• در مقاطع زمانی مختلف رهبر تیم قابل تغییر است (هر یک از اعضای تیم در مقطعی می‌توانند رهبر تیم باشند)

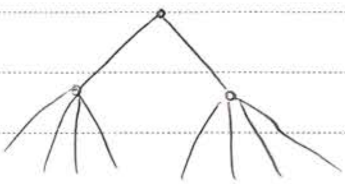
• احساس رضایت به خاطر خودمختار بودن و ارائه ایده‌ها

(۲) Controlled centralized (CC):

1) تیم دارای رهبر مشخص است

2) حل مسئله، ارتباطات و مدیریت توسط رهبر تیم انجام می‌شود.

3) ارتباطات عمودی است



ساختار درخت واره دارد و هر گروه از افراد مدیر مختص به خود دارند.

مزایا:

• کار تقسیم وظایف و نیازمندی‌ها و دستاوردها به صورت هماهنگ توسط رهبر تیم انجام می‌شود

- مقایسه معایب دو روش قبلی:

معایب روش DD:

• امکان اتلاف وقت توسط اعضای تیم به دلیل ارتباطات زیاد و عدم وجود حاکم بالاتر

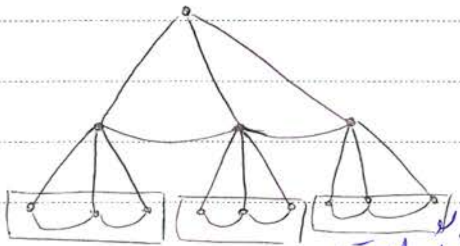
معایب روش CC:

• حجم زیادی از مسئولیت‌ها متوجه رهبر تیم است.

• وجود نقطه‌ی شکست تکی (single point of failure) گره: یعنی در صورتی که برای مدیر مشکلی پیش بیاید، نیروها نیز درست تلف می‌شوند.

Subject:
Year. Month. Date. ( )

③ controlled Decentralized (CD)



1. ترکیبی از دو روش قبل است
2. رهبر مشخص برای هر تیم
3. ارتباطات افقی
4. حل مسئله به صورت گروهی
5. مناسب برای سازمان‌ها و پروژه‌های بزرگ است

در این حالت در صورتی که مدیر باشد، چون تعداد بسیار افرادی که در یک گروه قرار دارند با همکاری یکدیگر می‌توانند کار را پیش ببرند. مزایای این روش شامل مزایای دو روش قبلی است.

راهنما برای تعیین نوع پروژه:

- برای پروژه‌های دشوار ← DD
  ← به معنای این است که ارتباط زیادی نیاز دارد و تحقیقاتی است.
- برای پروژه‌های بزرگ ← CC یا CD
- برای پروژه‌ها با محدودیت زمانی ← CC
- برای پروژه‌هایی که نیاز به تعامل بین اعضای تیم است ← DD
- برای وظایف و پروژه‌های حساس (Reliable) ← CD
  ← در صورتی که به مشکل بخورد ضرر زیادی به سیستم وارد می‌شود.

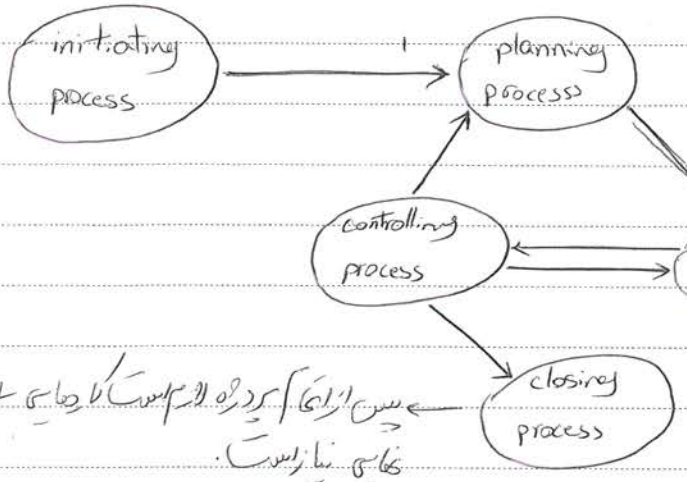
مفاهیم مدیریت پروژه:

گروهی به نام PMBOK ایجاد شده‌اند که هر یک از آنها تجارب در پروژه‌های مختلف را جمع‌آوری کرده و guide line هایی را برای مدیریت پروژه نرم‌افزاری ارائه کرده‌اند.

PAPCO Project Management Body of knowledging

Subject:
Year. Month. Date. ( )

- فرایندهای لازم برای پروژه:



این روند و آورده‌ها اغلب انجام می‌شود تا به پایان پروژه برسد یا پایان پروژه رد می‌شود.

پس از اتمام پروژه لازم است کارهایی برای تحلیل نهایی نیاز است.

براساس PMBOK مدیریت پروژه شامل ۹ وظیفه اصلی است:

۱- مدیریت یکپارچگی (integration Management):
اطمینان از اینکه هماهنگی مناسب بین عناصر مختلف پروژه وجود دارد. دارای ۳ شامل می‌شود عبارتند از

- توسعه طرح پروژه (project plan
- اجرای طرح پروژه (
- کنترل تغییرات (overall change

۲- مدیریت حوزه و عملکرد (scope Mangement)
اطمینان از اینکه تمام کارهای مورد نیاز در پروژه انجام می‌شود و کار اضافی نداریم. هدف تعیین محدوده پروژه است.

در صورتی که محدوده پروژه مشخص نباشد: - پروژه از نظر زمان و هزینه افزایش می‌یابد.
- کیفیت کار پایین خواهد بود.

- اقدامات آغازین و برنامه‌ریزی حوزه عملکرد (scope)
- تعریف حوزه عملکرد
- صحت سنجی (verification) و کنترل حوزه عملکرد.

Subject :
Year . Month . Date . ( )

۳- مدیریت زمان (Time) :
اطمینان از اینکه پروژه به موقع انجام می شود.

- تعریف فعالیت ها
- ترتیب فعالیت ها
- تخمین مدت زمان فعالیت ها
- کنترل و پایبندی پروژه

۴- مدیریت هزینه (cost Management) :
اطمینان از اینکه پروژه در چارچوب بودجه تعیین شده و صحیح انجام شود.

- برنامه ریزی منابع
- تخمین هزینه
- بودجه ریزی و کنترل بودجه

۵- مدیریت منابع انسانی (Human Resource) :
اطمینان از اینکه بهترین پرسنل از منابع انسانی استفاده می شود.

- برنامه ریزی سازمانی
- جذب نیرو
- تشکیل تیم پروژه

۶- مدیریت ریسک (Risk) :

- شناسایی ریسک
- کم کردن ریسک
- اتخاذ واکنش مناسب
- کنترل پاسخ به ریسک

PqPCO

28

Subject:
Year. Month. Date. ( )

۷- مدیریت کیفیت (quality):

- تضمین کیفیت محصول
- برنامه ریزی کیفیت
- کنترل و تضمین کیفیت

۸- مدیریت ارتباطات (Communication):

اطمینان از تولید - جمع آوری - تهیه و انتشار و توزیع مناسب و به موقع اطلاعات پروژه با استفاده از روش های ارتباطی

۹- مدیریت تدارکات (procurement):

تامین کالا و خدمات مورد نیاز پروژه از خارج از سازمان مجری. می خواهیم بررسی کنیم آیا برای پروژه باید چه چیز انجام شود و سایل مورد نیاز را هم تعیین کرده و در ادامه ی پروژه به تعهدات آن پرداخته می شود

- برنامه ریزی تدارکات
- برنامه ریزی تقاضا (solicitation)
- تقاضا
- انتخاب منبع
- مدیریت قراردادها و خاتمه آنها

مهندسی یعنی اندازه گیری که از دهه ۶۰ مطرح شده است. حال می خواهیم سیستم اندازه گیری مهندسی نرم افزار چگونه انجام می شود

تعریف اندازه گیری: فرایندی است که در آن اعداد و علائم به خصوصیات اشیا حال براساس قوانین تعریف شده مشخص شده انتصاب می یابد

attribute ← Entity

صندلی وزن 2kg
مقدار یا اندازه

در اندازه‌گیری دو مفهوم Reliability و validity وجود دارد. بسته به این که در دفعات مختلف اندازه‌گیری آیا مقادیر نزدیک به هم هستند یا نه Reliable گفته می‌شود و validity میزان نزدیکی مقادیر به مقدار واقعی است.

Subject:
Year. Month. Date. ( )

اندازه (measure): یک شاخص کمی از مقدار، میزان، بعد، ظرفیت یا اندازه یک خصوصیت از یک محصول یا فرایند.

... خط برنامه ← ۲ تا باگ وجود دارد.

معیار (metric): یک رابطه کمی بین اندازه‌هاست که با توجه به نیاز و شرایط سنجه تعریف می‌شود.

معیاری برای کیفیت یا انجام شده خواهد بود ← تعداد متوسط Bug در ۱۰۰۰ خط / یک ماه

اندازه‌ها به تنهایی معنایی ندارند بلکه وقتی که براساس معیار در قالب خاصی مطرح می‌شوند معنا پیدا می‌کنند.

شاخص (Indicator): یک معیار یا ترکیبی از معیارهاست که بینشی را نسبت به پروژه، فرایند یا محصول ایجاد می‌کند.

برنامه‌نویس خوب → 5000 line / month

بوسیله این شاخص می‌توان تشخیص داد اندازه‌ای که متناسب با معیار بدست آمده است چگونه ارزیابی شود.

معیار بررسی کیفیت توصیف نیازمندی‌ها:

تعداد نیازمندی‌هایی که reviewer های مختلف نظر یکسانی در مورد آنها داشته‌اند ← $Q = \frac{N_{ui}}{N_r}$ $N_r = N_f + N_{nf}$

N_{ui} مشخص می‌کند که توصیف از نیازمندی‌ها آن قدر خوب بوده که افراد مختلف از آن نیازمندی دید یکسانی را ایجاد کرده. این کسر حداکثر ۱ است و بهترین حالت زمانی است که $Q \to 1$.

تعیین اندازه نرم‌افزار:

- روش مستقیم: مثلاً تعداد خطوط برنامه (line of code ← measure) را به عنوان معیار در نظر بگیریم.

PAPCO

Subject:
Year. Month. Date. ( )

$$Q_1 = \frac{Loc}{month}$$

یعنی تعداد خطوط برنامه در یک ماه به عنوان معیار انتخاب می‌شود ولی این معیار مناسبی نیست و مشکلاتی را همراه دارد

1) وابسته به برنامه‌نویس ← برنامه‌نویس می‌تواند برای گرفتن پاداش کد را طولانی کند.
2) وابسته به زبان برنامه‌نویسی ← این کد از اسمبلی استفاده شود با جاوا متفاوت است
3) پیچیدگی الگوریتم‌ها را در نظر نمی‌گیرد ← این کد کم است ولی پیچیده است در نظر نمی‌گیرد
4) وجود code generator ها ← محیط‌های visual خود کد را تولید می‌کند.

در صورتی که بخواهیم از Loc استفاده کنیم باید Logical Line را در نظر بگیریم مثلاً تعداد خطوط برابر با تعداد ;ها است ولی در physical line تعداد خطوط فاصله داده شود یا comment ها نیز شمرده می‌شود

زمانی که کد برنامه زیاد می‌شود می‌توان بر اساس 1000 kilo بیان کرد killo LOC ⟵ KLOC

برای برطرف کردن مشکلات Loc می‌توان یکی از مکانیزم‌های زیر را اعمال کرد:

- Language gearing Factor (I)

میزان productivity زبان برنامه‌نویسی است یعنی مشخص می‌شود چه خط از یک زبان معادل تعداد خطوط دیگر در زبان دیگر است و به این وسیله تا حدودی مشکل زبان Loc برطرف می‌شود

Language	gearing Factor
Assembly	320
C	128
C++	55
Java	53

Subject:
Year. Month. Date. ( )

در این جا هیچ توجهی به ماهیت نرم افزاری سروکار ندارد و در حالت کلی بحث می کند.

(II) Function-point (FP):

با توجه به ماهیت مسئله اندازه نرم افزار تعیین می شود. بر اساس یک سری پارامتر اندازه ها را مقایسه می کند

External Input External output Logical internal file external interface file External Inquiry	مقدار هر یک از این پارامترها [illegible] را در وزن آن متناسب با پروژه ضرب می کنیم و VFP به دست می آید

ابتدا به وسیله ورودی، خروجی و query های موجود VFP به دست می آید که تنظیم نشده است سپس از طریق پارامترهای Nonfunctional آن را تنظیم می کنیم. باید پارامترهای Nonfunctional را نیز متناسب با پروژه ارزش دهی می کنیم و از رابطه زیر VAF به دست می آید:

value adjustment Factor ←

$$VAF = 0.65 + 0.01 * \sum_{i=1}^{14} C_i$$

C_i ← ارزش های 14 پارامتر Nonfunctional

(0.65 و 0.01): مقادیر استانداردی به صورت تجربی به دست آورده است.

$$FC = VAF * VFP$$

عدد به دست آمده ملموس نیست و باید به یک ارزش معنی دار تبدیل شود مثلاً می گوییم:

1 FP ⟶ میلیون درآمد

1 FP ⟶ 10 KLoC

در این طریق می توان FP را به صورت های متفاوتی معنا و تبدیل کرد.

PAPCO

در صورتی که بخواهیم LOC را قبل از نوشتن کد تخمین بزنیم از چندین متخصص نرم افزار سوال کرده و بر اساس رابطه زیر بدست می‌آید:

Subject:
Year. Month. Date. ( )

(کمترین ، بیشترین)

$$S = \frac{S_1 + S_2 + 4S_M}{6} \rightarrow \text{most likely}$$

8 Use Case Point (III)

بر اساس یک سری پارامتر پیچیدگی هایی را تعریف می‌کنیم عددی بدست می‌آوریم که بر اساس آن را adjust کنیم

	Factor
Simple	5
medium	10
Complex	15

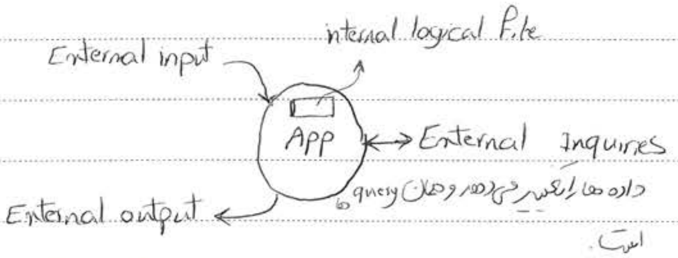
$$TCF = 0.6 + (0.01 * (\text{جمع Factor}))$$

در دو جدول متناسب با پارامترها پیچیدگی ها را انعکاس کرده و به پیچیدگی وزن می‌دهیم از جمع وزن های هر Factor ها در رابطه بالا گذاشته و TCF را بدست می‌آورد.

نکته: در Function point با در نظر گرفتن Requirement specification عمل می‌کند که موارد آن پارامترهای

پنج دسته FP عبارتند از:

1) Internal Logical File
2) External Interface File
3) External Input
4) External output
5) External Inquiries



نکته: logical file مجموعه ای که برای ذخیره کردن اطلاعات نیاز دارد ولی الزاماً نباید یک پارچه باشد بلکه می‌تواند مجموعه ای از داده های مرتبط می‌شود (a group of logical related file)

حال برای بدست آوردن UFP این پنج پارامتر را در ضریب هایی ضرب کرده و حاصل را جمع می‌کنیم

حال می‌توان FP را به تکنیک های دیگری تبدیل کرد و بر اساس تجربه می‌توان رابطه FP و LOC را بدست آورد.

بعد از اینکه چندین پروژه را بدست آوردن FP و LOC می‌توان به این رابطه منطقی رسید:

$$LOC = a * FP + b$$

(a و b بر اساس مشخصات گروه و تجارب تیم)

Subject:
Year. Month. Date. ( )

یکی از ضعف های FP این است که پیچیدگی الگوریتمی را در نظر نمی‌گیرد برای برطرف کردن این مشکل در روش Feature point استفاده می‌کنیم. در این روش الگوریتم را به ۳ قسمت تقسیم می‌کنیم:

1) simple
2) medium
3) sophisticated

$FP_{\alpha} = F_e + FP$

F_e: Feature point
FP: Function point

هر الگوریتم باید در یکی از این سه دسته قرار گیرد و ورودی های مربوط به آن ضرب می‌کنیم F_e بدست می‌آید. از جمع F_e و FP، Function point نهایی بدست می‌آید.

در Use case point مراحل زیر باید اجرا شود:

1) تعیین نوع و تعداد UC ها: ← UUCP

2) Technical complexity Factor (مسائل فنی)

$$TCF = 0.6 + \left(0.01 * \sum_{i=1}^{14} C_i\right)$$

3) Environment Factor (EF) (عوامل محیطی)

$$EF = 1.4 + (-0.03 * EFactor)$$ → Sample

4) $UCP = UUCP * TCF * EF$

تخمین نرم افزار:
شامل موارد زیر می‌باشد:
- زمان
- هزینه

PAPCO

Subject:
Year. Month. Date. ( )

تخمین هزینه:

I) روش قیاسی (analogy):
براساس تجارب پروژه‌های مشابه برای پروژه کنونی تخمین می‌زنیم. ولی این روش مناسبی نیست زیرا شرایط همیشه ثابت نیستند.

II) روش‌های تجربی:
با استفاده از نظرات افراد خبره هزینه‌ها پیش‌بینی می‌شود و با روش‌های مختلف آن‌ها را جمع‌بندی می‌کنیم.

III) روش بالا به پایین (Top-down):
برای هر App یک مقیاس عددی مانند FP را بدست آورده و براساس جداول موجود مقادیر اندازه بدست آمده را درنظر می‌گیریم. (اندازه نرم‌افزار تخمین زده می‌شود.)

IV) روش پایین به بالا (Bottom-up):
نرم‌افزار را شکسته و زیرفعالیت‌هایی را بدست می‌آوریم و برای آن‌ها هزینه را تخمین می‌زنیم.
این روش مشکلاتی را دارد:
1) همه فعالیت‌ها را ابتدا مشخص نیست.
2) فعالیت‌های چتری (Umbrella activity) دیده نمی‌شوند یعنی فعالیت‌هایی که در کل طول دسته باید دیده شود.
3) دقت در تخمین هر task

نکته: Estimation جزء فعالیت‌های چتری است و باید در تمام طول مسیر اندازه‌گیری شود.

هزینه‌ها:
1- هزینه تولید ← افراد، تجهیزات و ...
2- هزینه موادمصرفی ← موادی که مصرف می‌شود و در سایر پروژه‌ها قابل استفاده نیست
3- هزینه سرمایه‌ای (تجهیزات لازم) ← هزینه شرکت، هزینه کامپیوترها و ...

Subject:
Year. Month. Date. ( )

man/month : مقدار کاری که یک نفر در طول یک ماه انجام می دهد. (نفر-ماه)

ابتدا از روی FP ، نفر-ماه را تخمین می زنیم سپس باید از روی آن مدل لازم را بدست آوریم

روش های تجربی مطرح شده برای تخمین Non-algorithmic بودند بدون داشتن الگوریتم و تنها با در دست داشتن FP یا LOC در نظر گرفته می شد.

روش Algorithmic :

هدف تخمین نرم افزار بر مبنای یک مدل ریاضی است.

$Effort = f(x_1, x_2, \dots, x_n)$

$x_1, x_2 \dots x_n \longrightarrow$ cost factor

Cost factor: مجموعه متغیرهای x_1 تا x_n جزو یکی از این گروه ها قرار می گیرد

- product factor
- computer factor
- personal factor
- project factor

تابع f به صورت های مختلفی مدل شده اند که برخی از آن ها در ادامه بررسی می شود.

power function model :

a و b توابع ثابتی هستند که از طریق داده های مدل بدست می آید

$Effort = a \times S^b$

S ↓ (code size اندازه کد)

بر اساس بررسی ها از روی مدل های بالا فرمولی ارائه شد به اسم COCOMO (constructive cost model) که توسط آقای Bohem ارائه شده است.

PAPCO

Subject:
Year. Month. Date. ( )

$$MM = a\,(KDSI)^{b} * EAF$$

man month ← MM
EAF → effort adjustment factor
KDSI ↓ kilo Derived source Instruction

$$TDEV = c\,(MM)^{d}$$

time to develop ← TDEV

$$\text{people required} = \frac{\text{Effort}}{\text{Development time}}$$

برای استفاده از این فرمول، برای هر نوع پروژه ای که در وضعیت خود داریم a، b، c و d را بدست آوریم و مقیاس بندی را انجام داده و کار کنیم یا از مقادیر تعریف شده توسط Bohem استفاده کنیم. این اعداد از بررسی چندین پروژه بدست آمده است.

Baseline: قوانین اساسی موجود در شرکت یعنی جدولی که History شرکت در آن وجود دارد.

COCOMO II:
یک ورژن پیشرفته تر از روش قبلی است. کتاب هایی که این روش ها را توضیح داده اند عبارت اند از:

Software Engineering Economics
software cost estimation with COCOMO II –
در این حالت تعداد فاکتورها افزایش پیدا کرد.

proper collaboration:
باید در هنگام محاسبه مقادیر مقیاس ها و اندازه ها را معین کنیم که براساس یک دیتا تنظیم شده باشد.

software equation:

$$E = \left[loc \times B^{0.333} / P \right]^{3} \times \left(\frac{1}{t^{4}} \right)$$

Effort ← E
B ↑ ضریب مهارت های ویژه
P ↓ پارامتر بهره وری تولید
t → زمان

P4PCO

37

Subject:
Year. Month. Date. ( )

بررسی کارایی مدل ها:

از مکانیزم های می توان استفاده کرد تا کیفیت مدل های مورد استفاده را بررسی کنیم

Mean Absolute Relative Error (MARE)

$$MARE = \sum_{i=1}^{n} \left(\left| (estimate_i - actual_i)/actual_i \right| \right)/n$$

$p = 1, \dots, n$ n پروژه

Mean Relative Error (MRE):

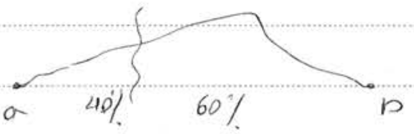
$$MRE = \sum_{i=1}^{n} \left((estimate_i - actual_i)/actual_i \right)/n$$

تفاوت در نداشتن قدر مطلق است

MRE → + (بزرگ) → overestimate

MRE → - → underestimate

این تخمین خاص شروع پروژه نمی باشد و یکبار در ابتدا و در اسم نمی گردد. باید در طول انجام پروژه میزان تخمین ها بررسی شود:



ETC = 15000 × 1/60 = 900 $ هزینه کل: 15000 $

↓
Estimate to completion

مقدار مورد نیاز برای ادامه از وسط پروژه

Subject:
Year. Month. Date. ( )

برون سپاری (outsourcing) :

سپردن انجام کاری (مسئولیت) به واحدهای بیرون ، تنها اجرای پروژه به بیرون سپرده می‌شود و مدیریت و نظارت بر عهده خود کارفرما است و حسابات مذاکره با پیمانکار بر عهده اوست.

از معایب این روش عبارت است:

1) سود پروژه کامل به خود مال نمی رسد.
2) دانش فنی از شرکت خارج می‌شود.
3) کیفیت پروژه درست مانیست.
4) در صورت ناکامل شدن پروژه تعهدات و مسئولیت بر عهده کارفرما می‌شود.
5) استرس

رابطه بین people ، effort :

مثال: در یک تیم فعالیت تولید هر یک از اعضای تیم برابر 500 loc/math است.

به صورت مستقل است. در یک پروژه یکساله، 4 نفر را درگیر می‌کنیم.

$4 \times 500 \times 12 = 24000$ loc/year

این عدد درست نمی‌باشد زیرا این افراد مستقل کار می‌کنند ولی با یکدیگر در ارتباط هستند. این مقدار هم می‌شود.

* هزینه ارتباط هر دو نفر برابر 25 loc/month است.

$\binom{4}{2} = 6$ ⟵ تعداد ارتباط‌ها

باید هزینه ارتباط‌ها را از عدد حاصل کم کنیم:

$24000 - 12 \times 6 \times 25 = 24000 - 1800 = 22200$ loc/year

PqPCO

39

Subject:
Year. Month. Date. ( )

حالت دیگری را در نظر بگیرید که در یک پروژه یکساله، در مدت زمان ۲ ماه به پایان پروژه، ۲ نفر جدید اضافه می‌شود.

۴ نفر ۶ نفر
ماه ۱۰

$(4 \times 10 \times 500) + (6 \times 2 \times 500) - (6 \times 10 \times 25 + 16 \times 2 \times 25)$

در صورتی که این عدد کمتر شد یعنی اضافه شدن دو نفر صرفه‌جویی داشته باشد. باید سایر پارامترهای موثر اضافه شدن این دو نفر را نیز در نظر بگیریم مثلاً هزینه train

۳ effort روش توزیع

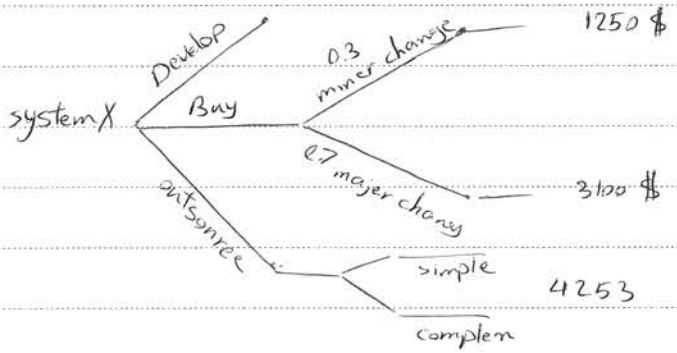
یکی از روش‌های برآورد هزینه روش pressman است 40-20-40 Rule

40٪ برای آنالیز و طراحی

20٪ برای پیاده‌سازی

40٪ برای تست

تصمیم گیری؟



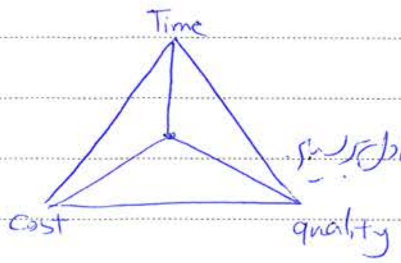
expected cost = $\sum$ (path probability) × (estimated path cost)

در مثال تصمیم گیری بر اساس هزینه راه‌های مختلف، تصمیم می‌گیریم که کدام اتفاق بیافتد.

P4PCO

Subject:
Year. Month. Date. ( )

مدیریت زمان پروژه :



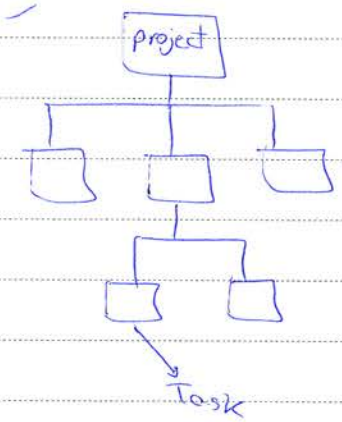
سه عامل مهم در انجام یک پروژه به صورت روبرو است که باید به نقطه تعادل برسیم.

مسائل مطرح در زمان بندی :

۱- شناسایی فعالیت و تعیین Task ها
۲- مشخص کردن وابستگی بین Task ها
۳- تخصیص زمان به هر Task (تاریخ شروع و پایان)
۴- اعتبار سنجی Task ها
۵- انتساب وظایف به اعضای تیم
۶- تعیین دست وردها
۷- تعیین milestone ها

الف) تعیین لیست فعالیت ها (WBS , Work Breakdown structure)

در این روش یک پروژه را آنقدر خرد می کنیم تا به فعالیت های کوچک و منطقی برسیم.



PAPCO

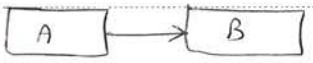
Subject :
Year . Month . Date . ( )

ب) تعیین وابستگی Task ها:

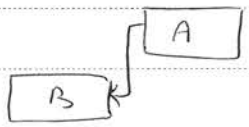
انواع وابستگی ها عبارتند از:

- وابستگی اجباری و جزو ذات کار است.
- وابستگی اختیاری (Discretionary) که توسط تیم تعیین می‌شود.
- وابستگی خارجی (External) ، از خارج از پروژه اعمال می‌شود. مثلاً یک نرم‌افزار یا از خارج از شرکت خریداری می‌گردد.

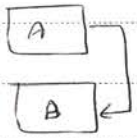
- Finish to start :



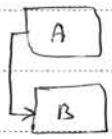
- start to finish :



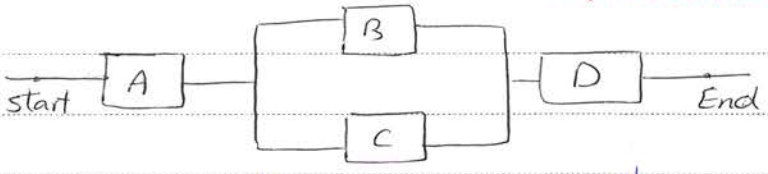
- Finish to finish :



- start to start :



دیاگرام شبکه (Network Diagram) :



براساس وابستگی های موجود این وابستگی ها برای Task ها دیاگرام بالا رسم می‌شود.

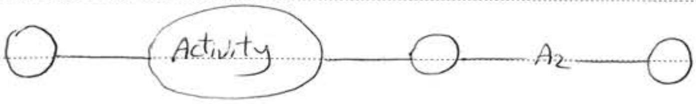
(ادامه در اسلایدها)

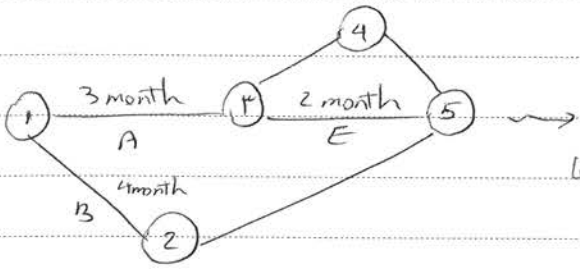
P4PCO

Subject:
Year. Month. Date. ( )

۵. Arrow Diagraming Method

تحت عنوان روش ADA شناخته می‌شود. در این نوع دیاگرام بردار رسم می‌شود، یال‌ها فعالیت‌ها را نشان می‌دهد





روی یال‌ها هزینه انجام task ها نشان داده می‌شود و نودها فقط ارتباط‌ها را نشان می‌دهد.

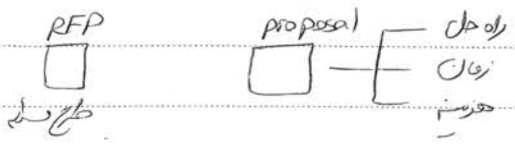
تکنیک‌های کاهش زمان پروژه

۱ـ اضافه کردن منابع جدید: task هایی که روی مسیر بحرانی هستند را در نظر می‌گیریم و منابع مورد نیاز آن را زیاد می‌کنیم

۲ـ کاهش scope پروژه: در صورتی که زمان کم بود انجام کارها را محدودتر می‌کنیم

۳ـ تغییر توالی وظایف: شاید با جابجا کردن Task ها کارها سریعتر انجام شود

۴ـ time boxing: زمان انجام پروژه را به بازه‌هایی می‌شکنیم و افق زمانی‌های جزئی را مشخص می‌کنیم. در صورتی که انجام یک task از این زمان بیشتر شد آن Task متوقف شده و بقیه کارها را ادامه می‌دهیم



شرکت‌ها هم پروپوزال

Subject:
Year. Month. Date. ( )

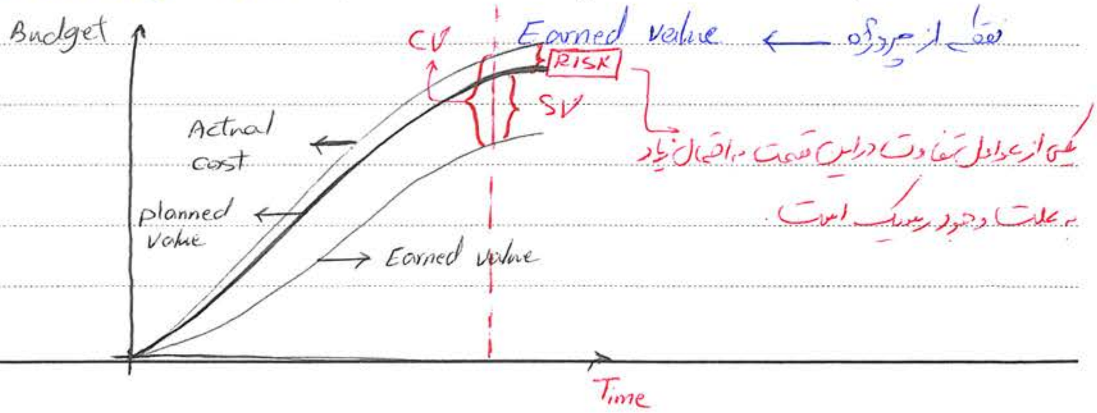
Monitoring & Tracking :

پیگیری و پایش پروژه در طول اجرا است که روش‌های مختلفی برای انجام آن وجود دارد.

- برپایی نشست‌های اداری
- ارزیابی نتایج بازبینی
- کنترل milestone ها
- استفاده از روش تحلیل ارزش حاصله

روش تحلیل ارزش حاصله (Earned value Analysis) :

- Budget cost of work scheduled (BCWS) : در برنامه‌ریزی زمان‌بندی ایجاد می‌کند و فرض آن مشخص می‌شود چه میزان کار تا زمان مشخص باید کامل شده باشد ← planned value
- Actual cost of work performed (ACWP) : مجموع effort ی که واقعاً تا این نقطه از پروژه برای task ها ثبت شده است. ← actual cost
- Budget cost of work performed (BCWP) : ارزش حاصل از کارهای انجام شده تا یک نقطه از پروژه ← Earned value



Subject:
Year. Month. Date. ( )

معیارها:

- معیار واریانس زمانبندی: میزان دهنده واریانس محاسبه از زمان بندی برنامه ریزی شده است.

Earned value

$$SV = EV - PV$$

SV: schedule variance — PV: Planned value

- معیار واریانس هزینه پروژه:

$$CV = EV - AC$$

CV: cost variance — AC: Actual cost

$CV > 0$ ⇝ under Budget — از آن بودجه پیش بینی شده کمتر مصرف کرده ایم.

$CV < 0$ ⇝ over Budget — بیشتر از بودجه پیش بینی شده مصرف کرده ایم.

$SV > 0$ ⇝ ahead of schedule — از زمان بندی جلو تر هستیم و این خوب است.

$SV < 0$ ⇝ Behind the schedule — از زمان بندی پروژه عقب هستیم.

- معیار کارایی زمان بندی (schedule performance index): SPI

$$SPI = \frac{EV}{PV}$$

هرچه این عدد از یک بیشتر شود یعنی از اول درست تخمین زده نشده است.

P4PCO

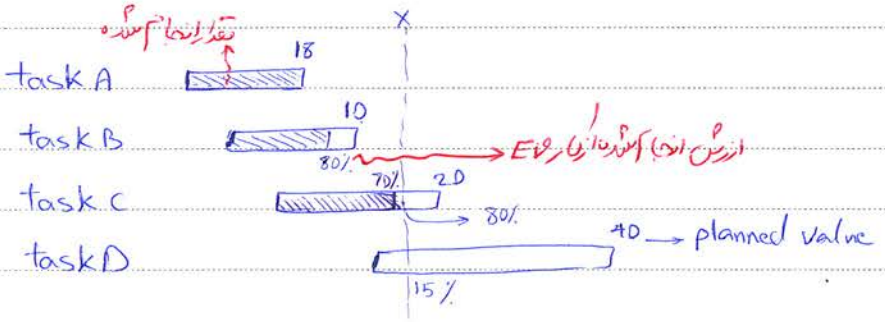
Subject:
Year. Month. Date. ( )

- معیار کارایی هزینه:

$$CPI = \frac{EV}{AC}$$

در اینجا نیز هرچه این مقدار یک است و در صورتی که بزرگتر از یک شود یعنی از زمان بندی جلوتر هستیم. در صورت جلو تر بودن از زمان یا هزینه خوب است یعنی توانستیم به خوبی تخمین بزنیم.

مثال: فرض کنید نمودار Gantt یک پروژه به صورت زیر در اختیار است:



فرض کنید actual cost = 45

وضعیت پروژه در نقطه X

$PV = 18 + 10 + 80\% \times 20 + 6 \times 15\% = 50$

$EV = 18 + 8 + 14 + 0 = 40$

$SV = EV - PV = 40 - 50 = -10 < 0$ ⟶ behind the schedule

$SPI = \frac{40}{50} = 0.8$ ⟶ 80%

$CV = EV - AC = 40 - 45 = -5 < 0$ ⟶ over Budget

$CPI = \frac{40}{45} = 0.89 = 89\%$

- برآورد اولیه این هزینه انجام شود = PV = 100 $
- هزینه واقعی که برای این هزینه صرف شده = AC = 110 $
- ارزش کار انجام شده = EV = 80 $

این اندازه‌ها وضعیت را مشخص می‌کند.

P4PCO

Subject:
Year. Month. Date. ( )

- بودجه تکمیل پروژه:

$BAC = \sum (BCWS_k)$ for all task K

- شاخص درصد برنامه ریزی برای تکمیل پروژه:
نشان دهنده ی ارزش برنامه ریزی کار تکمیل شده

percent scheduled for completion = $\frac{BCWS}{BAC}$

- شاخص درصد تکمیل شدن پروژه:
نشان می دهد تکمیل پروژه در زمان t

percent complete = $\frac{BCWP}{BAC}$

مدیریت منابع انسانی

- بحث Resource Allocation:

نمونه های سازمانی: (RAM) Responsibility Assignment Metrix

code	Task name	project sponser	Analyst	project manager	...	Developer
	task A	C	R	A		R
	task B	A	I			

معنای هر خانه جدول از واژه زیر انتخاب می شود:
- Responsible: کسی که کار را انجام می دهد.
- Accountable: کسی که مسئولیت صحت انجام کار را برعهده دارد.
- consulted: کسی که باید با او درمورد task مشورت شود.
- Informed: کسی که باید درمورد task مطلع شود.

Subject:
Year. Month. Date. ( )

هدف از انجام این جدول انتساب وظایف به نفس ها است ولی مشخص نمی کند که هر فردی آن کار را انجام می دهد.

رابطه نفس ها و افراد در یک پروژه:

	Analyst	Design	code
Al.	✓		
Hassan		✓	
:			✓

وضعیتی که هیچ پروژه به صورت همزمان انجام نمی شود به صورت ترتیبی عمل می کنیم:

	Analyst	Design	code
project1	Al.	..	
project2	Hassan		

مدیریت ریسک:

Morphy's Law: " Anything that can go wrong, will go wrong"

مفهوم مدیریت ریسک: ریسک پدیده ای است که ممکن است در آینده رخ دهد و باعث اختلال و تأثیرات منفی در روند پروژه شود.

دو ویژگی کلیدی ریسک:

- عدم قطعیت: ریسک ممکن است اتفاق بیفتد یا نه.
- ضایعات: اگر ریسک به وقوع بپیوندد حتماً ضایعاتی در پی خواهد داشت.

رویکردها در مقابله با ریسک:

- واکنشی (reactive): پس از وقوع ریسک به رفع اثرات می پردازد.

Subject:
Year. Month. Date. ( )

- پیش واکنشی (proactive) : مدام ریسک گیری دارد

در رویکرد واکنشی در همان لحظه وقوع تصمیم می‌گیریم در حالی که در پیش واکنشی از قبل تصمیم‌گیری می‌شود.

مراحل انجام کار:
1) شناسایی ریسک
2) اولویت بندی
3) تهیه برنامه مدیریت ریسک ⟵ Risk Management plan

1- شناسایی ریسک:
ریسک ها بطور کلی دو نوع است:
- ریسک کلی : قطعی برق یا اینترنت
- ریسک خاص یک پروژه : کم شدن نیروهای انسانی

ثبت و بالا بردن ریسک:
یکی از روش های بیان ریسک که توسط pressman مطرح شده به صورت زیر می‌باشد:

Condition . Transition Consequence (CTC)

Given that <condition> then there is concern that (possibly) <consequence>

پس از بیان ریسک باید آن را ثبت و پالایش کنیم روش پیشنهادی Risk mitigation monitoring & management (RMMM)

P4PCO

48

Subject:
Year. Month. Date. ( )

نکات:

1) اجتناب از ریسک: طوری برنامه ریزی می کنیم که احتمال ریسک ایجاد شود و تمهیداتی را در نظر می گیریم که مشکلی ایجاد نشود ولی باید در نظر داشت که با کم کردن تمهیدات مشکلی ایجاد نکند (سعی بر جلوگیری از ریسک)

2) نظارت بر ریسک: بررسی ریسک در طول انجام پروژه و یا اقدامات پیشگیرانه دوره ای است. تشخیص اطلاعات در مورد ریسک

3) مدیریت ریسک و برنامه ریزی احتیاطی: در صورتی که ریسک اتفاق بیفتد تمهیدات مربوط را اعمال می کند. در صورت بروز ریسک چه کارهایی انجام می شود.

تاثیر ریسک: (Risk impact)

- فاجعه بار (catastrophic)
- بحرانی (critical)
- حاشیه ای (marginal)
- قابل چشم پوشی (Negligible)

گروه های متعادت ریسک:

- ریسک پروژه ای ← زمان - هزینه - نیروی انسانی
- ریسک فنی ← کیفیت نرم افزار
- ریسک های تجاری (بیرونی) ← ریسک بازار، رقابتی، عوامل اقتصادی

PAPCO

Subject:
Year. Month. Date. ( )

احتمال وقوع (Risk probability):

- زیاد (high)
- متوسط (moderate)
- پایین (low)

Risk Table:

ردیف	عنوان ریسک	گروه ریسک	احتمال وقوع	تأثیر	وزن ریسک	RMMM
		فنی	زیاد	بحرانی	w	

هرکدام از موارد بالا مشخص می‌شوند و به صفت‌های کیفی مقدار عددی را نسبت می‌دهیم. RMMM نشانی است به جدول ریسک که مشخصات آن ریسک در آن مشخص شده است.

وزن = احتمال × تأثیر

پس از بدست آوردن وزن جدول را براساس آن sort می‌کنیم و از یک جا به بعد ریسک‌ها قابل چشم‌پوشی هستند و یک cut off line رسم می‌کنیم.

تحلیل خطر ریسک: (Risk Exposure)

$RE = prob \times cost$

مشخص می‌کند که در صورتی که ریسک اتفاق بیفتد چقدر خسارت ایجاد می‌کند.

Subject:
Year. Month. Date. ( )

میزان صرف هزینه برای ریسک:

براساس قانون 20 _ 80 ، 80٪ هزینه‌ها و خسارت های پروژه مربوط به اشکالاتی به در 20٪ پروژه است
موقع بررسی است. درنتیجه باید ریسک های مهم شناسایی و کنترل شود.

نقطه مقابل ریسک ، serendipity است یعنی حال حوادثی در مسئله پیش آید و انتظارش نبود ولی فرصتی
برای حل آن پیش می‌آید.

نکته 8 : ریسک جزء فعالیت های جدی است و در طول انجام پروژه همیشه باید در نظر گرفته شود.

(فرم RMMM درون Document ها موجود است.)

proposal ← درون اسلایدها قرار دارد.

project plan ← درون اسلایدها قرار دارد.

PAPCO

Subject:
Year. Month. Date. ( )

بخش سوم: فرایندهای توسعه نرم افزار

مدل لایه ای نرم افزار:

ابزار (Tool) ← به صورت خودکار یا نیمه خودکار به روش فرایندیک می کند.
روش (method) ← چگونگی انجام کارها
فرایند (process) ← چیدمان فعالیت ها برای رسیدن به هدف
کیفیت (Quality) ← هدف اولیه

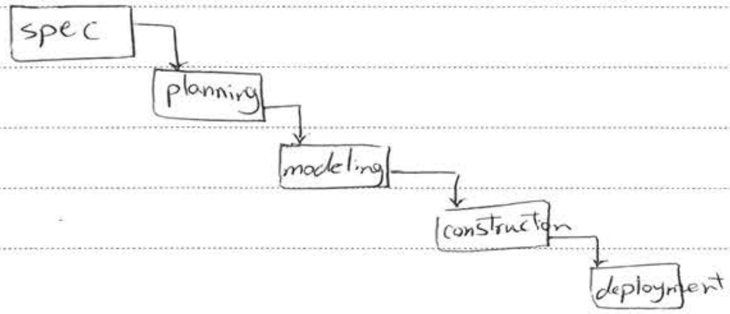
مدل عمومی فرایند

فعالیت های کلی که در یک فرایند می تواند انجام شود یکی از موارد زیر است:

۱_ ارتباط (communication)
۲_ برنامه ریزی (planning)
۳_ مدل سازی (modeling)
۴_ ساخت (construction)
۵_ استقرار (Deployment)

waterfall model (1):

قبل از همه روش ها Build & Fix بوده است که ابتدا به صورت کامل ساخته می شود سپس به برطرف کردن اشکالات آن می پردازند. روش بعدی به صورت آبشاری است که هر یک از مراحل جداگانه ابتدا به تمامی انجام می شود و سپس به مرحله بعدی می رویم.



- زمانی که کیفیت مهم‌تر است مناسب می‌باشد.
- یکپارچه‌سازی پروژه از ذی‌نفعان ...
- خاصیت problem-solving که خاصیت نرم‌افزار است در آن وجود ندارد.

Subject:
Year. Month. Date. ( )

مشکلات:

زمانی استفاده می‌شود:

1) تغییر نیازمندی‌ها مشکل ایجاد می‌کند. — 1) نیازها به خوبی شناخته شده باشد.

2) تمام نیازمندی‌ها از ابتدا مشخص نیست. — 2) تعریف محصول ثابت باشد.

3) ایجاد نسخهٔ نرم‌افزار و نمایش آن طول می‌کشد. — 3) تکنولوژی جدید و پروژه باشد.

4) ایجاد Blocking state: یعنی شرایطی که بخشی از تیم منتظر کار بخش دیگر است.

4) ورژن جدیدی از یک محصول موجود باشد.

5) محصول موجود را در platform جدیدی ارتقا دهند.

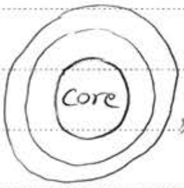
مزایا:

1) برای پروژه‌هایی که نیازمندی‌ها از ابتدا مشخص است مناسب می‌باشد.

II) Incremental model: (مدل افزایشی)

خصوصیات مدل آبشاری با تکرار همراه می‌شود. در ابتدا هسته‌ای از نرم‌افزار ایجاد می‌شود که نیازمندی‌های پایه‌ای است (core) در نهایت آن هسته تکمیل می‌شود.

مثلاً در نرم‌افزار word ابتدا صفحه برای نوشتن و edit فراهم می‌شود سپس ابزارهای بعدی به آن اضافه می‌شود.



- ریسک تغییر نیازها کم می‌شود. - مشتری به هر build پاسخ می‌دهد.
- تحویل اولیه سریع‌تر می‌شود. - planning و design خوبی نیاز دارد.
- مشتری می‌تواند سیستم را سریع‌تر ببیند. - هزینهٔ کلی پروژه کمتر نیست.

1) از مزایای این روش این است که می‌توان قسمتی از کار را زودتر تحویل داد و متناسب با پروژه‌های ارزان تحویل داده می‌شود.

2) مزیت دیگر آن نیروی انسانی است که به صورت همزمان می‌توان از آن‌ها استفاده کرد. (مدیریت کمتر)

3) درصورتی که در مرحلهٔ خاصی نیاز به تجهیزات سخت‌افزاری خاصی باشد. زمانی مناسب است که:

1- تکنولوژی جدیدی باشد

2- عملکرد اولیه سریع‌تر داده شود

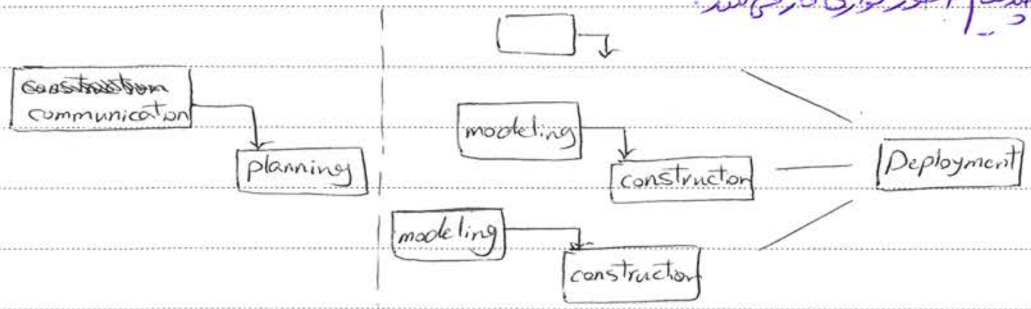
- RAD (rapid Application development):

محصول در یک دورهٔ زمانی کوتاه می‌خواهیم آماده شود. در نتیجه scope پروژه و نیازمندی‌ها مشخص است منظور بازهٔ زمانی کوتاه 60 تا 90 روز است.

PAPCO

Subject:
Year. Month. Date. ( )

- استفاده از ابزارها و code generator تاکید دارد. ← تیم های مختلفی تقسیم می شوند
- چندین تیم به طور موازی کار می کنند.



مشکلات:

- مشتری های ابزار (sophisticated) : با اهمیت سرعت رسیدن انجام پروژه آشنا باشد.
- در صورت وجود ریسک فنی بالا مناسب نیست
- باید سیستم قابل شکستن به ماژول های مشخص باشد.
- برای پروژه های بزرگ نیاز به منابع انسانی زیاد است.

III) Evolutionary model : (مدل تکاملی)

در مدل incremental یک مدل ناقص ایجاد می کنیم و سپس تکمیل می شود ولی در مدل تکاملی ابتدا یک prototype ایجاد می کنیم ولی قسمت های اول پیاده نشده است اما همه ویژگی های پروژه را دارد

3.1- prototype : به دو دسته تقسیم می شوند:

+ Through_away : دور ریختنی است و بعدا توسعه دهندگان برای انجام خود پروژه از آن استفاده می کنند.
+ operational : همین نمونه مرحله به مرحله تکمیل می شود

PAPCO

Subject:
Year. Month. Date. ( )

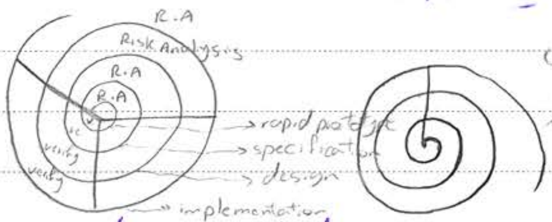
مشکلات:

۱- اگر تکنولوژی مناسب و روش مناسب در توسعه نرم افزار استفاده نشود. (ممکن است)

۲- مشتری نمی‌داند برای اجرای پروژه با آزمایش به هم چسبیده است!

3.2 - spiral : (مدل حلزونی) ← - امکانات بررسی ریسک را فراهم می‌کند که در هر مرحله باید
- کاربران می‌توانند علت عدم ایجاد شدن سیستم را بررسی کنند

نکاتی که در این روش وجود دارد عبارت است از: - مشتری‌های ریسک حساس از ابتدا در نظر گرفته می‌شود
1) توسط آقای Bohem پیشنهاد شده است. - لازم نیست که design آن کامل باشد
2) روش بهینه‌سازی را با شرایط توجه به ریسک انجام می‌دهد. (Risk Driven Dev.)

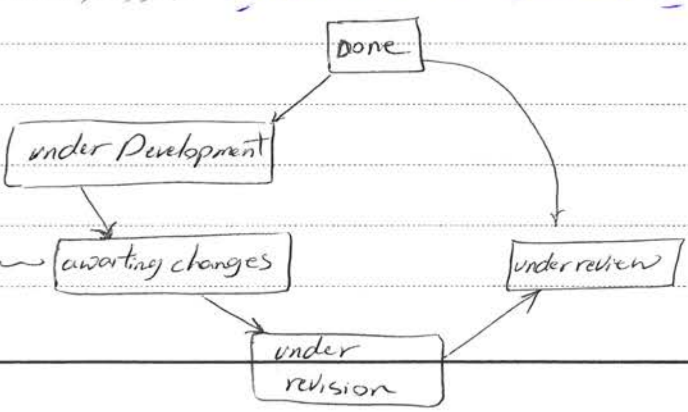


یک مدل ساده شده از spiral مانند waterfall است
در هر مرحله از آن همراه با ارائه و مدیریت ریسک می‌باشد

به هر میزان که پیش می‌رویم ریسک‌ها را در نظر می‌گیریم، در صورتی که ریسک غیر قابل کنترل بود پروژه را قطع می‌کنیم
- به کاربران به سرعت feedback داده می‌شود

برای پروژه‌های بزرگ مناسب است و هم زمان با تکامل نرم افزار فرایند نیز تکامل می‌یابد.
(هزینه زیاد، خوانده دشواری در ارائه‌ها وجود است.)

3.3 - Concurrent Development (توسعه هم روند):
در این روش مسیر حرکت یک state در نظر می‌گیرد. ← که هر مرحله از process model این کارها را انجام می‌دهد



سه تا پیدا کردن خطا ← awaiting changes

Subject:
Year. Month. Date. ( )

هر یک مراحل کلی و اصلی باید این زیر مراحل state machine را طی کنند در طول پروژه هر یک از آنها درگیر این state ها هستند

IV) مدل های فرایند تخصصی:

4.1- Component base devel :

نرم افزار را به یک سری component می‌شکنیم و سعی می‌کنیم از component های موجود برای این کار استفاده کنیم. مثلاً یک سری function های کلان در نظر گرفته به صورت dll یا jar وجود دارند و از همان dll های موجود برای این کار استفاده می‌کنیم

در استفاده مجدد از مولفه‌ها نکات زیر باید رعایت شود:

1) برای محصول مورد نظر همه محصولات مبتنی بر مولفه موجود شناسایی می‌شود

2) مسائل مربوط به یکپارچگی مولفه‌ها در نظر گرفته می‌شود

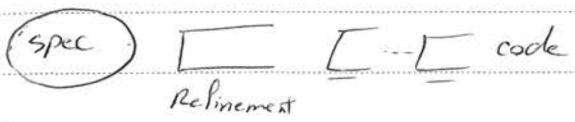
3) معماری نرم افزار تعیین شود

4) مولفه‌ها در ساختار معماری قرار گیرد

5) Integrity test (تست جامع بودن) انجام شود

4.2- Formal method :

نرم افزار را براساس مدل‌های ریاضی بیان کرده و با استفاده از مدل های ریاضی آن را حل می‌کنیم



Subject:
Year. Month. Date. ( )

spec را به زبان ریاضی و با ابزارهای موجود [illegible] تعریف می‌کنیم و سپس ابزارهای مناسب برای مراحل بعدی بستر بوجود می‌آید. به عنوان [illegible] مثال هم‌ترازان بوسیله؟ logic همین مراحل را طی می‌کند.

زبان های توصیف صوری:

- Z
- object Z
- RSL
- B_method

Drugstore
dr: IPdrug
lc: IP location
store: Drug ⇸ location

روش‌های ریاضی هم معمولاً هزینه‌بر هستند اما بیشتر برای پروژه‌هایی که حساسیت کاری بالاست قابل استفاده‌اند.

معایب:

1) هزینه تولید بالاست.
2) ارتباط با مشتری دشوار است.
3) هزینه آموزش برای تیم توسعه

4.3 _ Aspect oriented sw Development

در بعضی از کدها چند ماژول به دغدغه‌های یکسانی درگیر هستند در این حالت تکرار کد آن را کمینه می‌کند. برای حل این مشکل از aspect oriented استفاده می‌کنیم.

tangling: چند concern در یک ماژول آمده است. مثلاً log کردن رخدادها

معمولاً concern ها به خاطر نیازهای non-Functional ایجاد می‌شود.

Subject:
Year. Month. Date. ( )

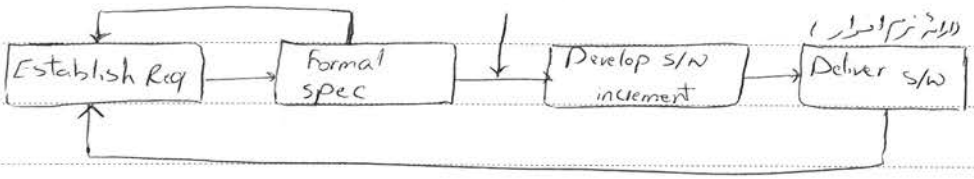
: Cleanroom software Dev process

- نکته ۱: تاکید بیشتری از Defect است و می خواهیم از همان اول از بروز آن جلوگیری کنیم.

- نکته ۲: تاکید بر استفاده از روش های صوری (Formal methods)

- نکته ۳: تاکید بر استفاده از کنترل کیفیت به صورت آماری

Forzen specification ← freeze می کنند و دیگر آن را تغییر نمی دهد.



- نکته ۴: تاکید بر روی کد افزایشی (incremental)

و واقعا این نیز یک process model است که در آن به صورت incremental استفاده می شود.

RUP
: (Rational unified process) Unified process model
IBM ←

در این روش سعی کرده است بهترین ایده هایی که در همه مدل ها هست را در آن قرار دهد.

- Jacobson
- Booch
- Rumbaugh

} UML (unified modeling Language)

مدل RUP با استفاده از زبان UML تعریف ها و عینیات کلیه نرم افزار را کامل می کند. اهدافی که از VP انتظار می رفت عبارتند از:

1- Use-Case Driven

P4PCO

Subject:
Year. Month. Date. ( )

۲- Architecture centric

۳- Iterative & incremental

ویژگی‌های کلیدی RUP:

باید اهداف و انتظارات را در بر بگیرد.

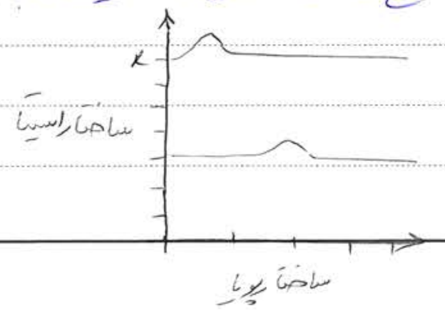
روح RUP:

- حمله مستمر بر ریسک ← spiral
- ایجاد ارزش افزوده برای مشتری: باید همیشه تمرکز این باشد که کار درست و محصول با ارزش خوب ایجاد شود.
- تأکید بر داشتن نرم‌افزار قابل اجرا در هر مقطع پروژه
- توجه به مدیریت تغییرات
- وجود معماری مستحکم و قابل اجرا
- تولید با استفاده از مؤلفه‌های نرم‌افزاری (component base)
- فعالیت در قالب یک تیم ← در مدل RAD هم تیم است
- کیفیت: در همه فعالیت‌ها ← کیفیت نباید به عنوان یک فاز در نظر گرفته شود بلکه باید در کل پروژه جریان دارد

RUP چهار فاز در طول زمان پروژه ایجاد می‌کند به عنوان ساختار پویا مطرح می‌شود. این چهار فاز عبارتند از:

۱) ساختار آغازین (Inception)
۲) تشریح (Elaboration)
۳) ساخت (construction)
۴) انتقال (transition)

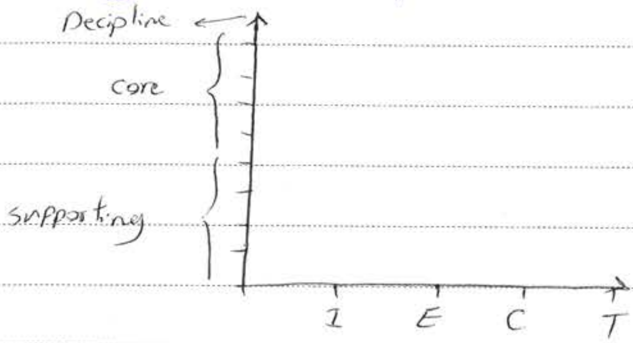
سپس ویژگی‌های پروژه را در ساختار ایستا مطرح می‌کند و مشخص می‌کند میزان توجه به هر ویژگی در هر کدام فاز بیشتر است.



مشخص می‌شود در این کاربرد در کدام فاز ← توجه بیشتری دارد

Subject:
Year. Month. Date. ( )

مباحث کار آسیا شامل Decipline هایی است که به دو دسته Core و supporting تقسیم می‌شود.



- RUP یک process Framework است. RUP مجموعه‌ای از دستورالعمل‌ها در زمینه‌ها Decipline های مختلف را برای پروژه خاصی مورد نیاز و مفید است را مطرح می‌کند

- RUP هم برای پروژه‌های کوچک و هم پروژه‌های بزرگ است، زیرا انعطاف‌پذیری دارد از همه دستورالعمل‌های موجود استفاده نمی‌شود و می‌توان به دلخواه از آنها استفاده کرد

فازهای RUP:

1. RUP هم برای پروژه‌های بزرگ و هم پروژه‌های کوچک می‌توان استفاده کرد

- Inception communication & planning

- Elaboration: modeling
Architecture

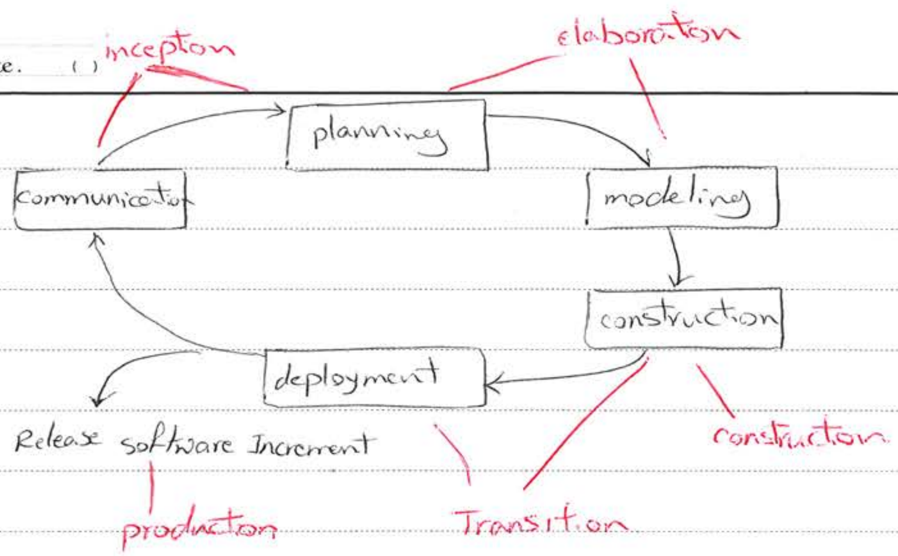
- construction: ساخت نرم افزار

- Transition: نرم افزار که تست شده در اختیار مشتری قرار می‌گیرد ← در پایان این فاز نرم افزار قابل تحویل نهایی به مشتری است.

در واقع در RUP لزوماً با تمام یک درصد مراحل را به کار بردیم و فازهای مختلف در طول پروژه وجود دارند.

PAPCO

Subject:
Year. Month. Date. ( )



در این روش از ایده incremental استفاده می‌کند. تفاوت RUP با incremental این است که هر فعالیت در RUP در تمام مسیر پروژه دیده می‌شود یعنی req درست است این ویژگی در فعالیت ها implementation و construction بیشتر است اما در incr. می‌گویند اول فقط req و بعد ساخت و ...

Decipline ها:

- Decipline های اصلی (Core)

۱- مدل سازی کسب و کار (Bussiness modeling) ← ذهنیت ذینفعان

۲- نیازمندیها (Req)

۳- تحلیل و طراحی (Analysis & Design)

۴- پیاده سازی (Imp)

۵- تست (Test)

۶- استقرار (Dep)

- Decipline های پشتیبانی (supporting):

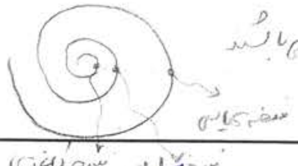
۱- مدیریت پروژه (project management)

۲- مدیریت تغییرات و پیکربندی (change & configuration)

۳- محیط (Environment): بخش هایی که لازم است تا تیم نرم افزار بتواند کار کند و محصول را تولید کند. (تجهیزات لازم - سخت افزار - نرم افزار)

PAPCO

Subject:
Year. Month. Date. ( )



ردیگر Incremental است و سقف‌ها رو به بهبود می‌باشد
برخلاف spiral

برای گویا بودن پروژه مستنداتی آماده می‌شود که در RUP به آن Artifact می‌گویند مجموعه‌ای از این Artifact ها عبارتند:

: Inception

- Vision : انتظارات سطح بالا (چشم‌انداز) در ابتدا RFP و proposal آماده می‌شود که فعلی است چنین حالتی باشد در این زمان فعلی است Req ها تغییر کنند و انتظارات از سیستم مشخص تغییر کند. در نتیجه سند چشم‌انداز در حال ابتدا تعیین می‌شود و به توافق می‌رسند.

- Risk : مجموعه ریسک‌های پروژه را مشخص می‌کند.

: Elaboration

- Use case specification (نیازمندی‌های Functional)
- supplementry spec (نیازمندی‌های nonFunctional)
- Architecture (توصیف معماری)

در فاز construction و Transition نیز مستنداتی لازم است.

پیشنهاد RUP این است که در پایان هر فاز یک major milestone ایجاد کرده و مشخص کنیم آیا کارهای لازم در آن فاز انجام شده است یا نه. در صورتی که به نتیجه نرسیده باشد باید چند کار تکمیل شود سپس به فاز بعدی برود.

RFP — proposal — project plan — vision — usecase spec — Architecture — SW

- Tools ← برای رسیدن به هدف هر یک از کارها باید چه ابزاری استفاده شود
- method ← برای رسیدن به هدف هر یک از کارها به چه صورت انجام شود.
- process ← برای رسیدن به هدف چه کارهایی با چه ترتیبی انجام شود؟
- Quality ← به چه هدفی می‌خواهیم برسیم

methodology

: Decipline

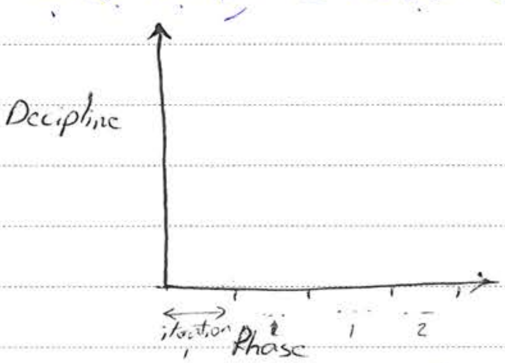
درلغت به معنای نظم و انضباط است. درواقع نوعی ارتباط منطقی بین مجموعه فعالیت ها، نقش ها و دستاوردهای هر Decipline وجود دارد.

Role ← چه کسی
artifact ← چه کاری
activity ← چگونه
workflow ← چه موقع

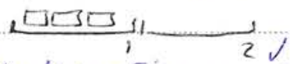
از دید RUP هر پروژه ای می توان استفاده کرد به شرط اینکه customize شود. RUP یک process framework است که آنچه مورد نیاز است باید از آن بیرون کشیده شود.

: Iteration

در RUP یک iteration یک سیکل پروژه تعریف می شود که برای انجام آن از همه Decipline های RUP استفاده می شود و به عنوان خروجی آن یک سیستم قابل اجرا ولی ناقص تحویل می گردد.



گاهی در انتهای یک فاز می بینیم نمی توانیم به سوالات انتهای فاز پاسخ دهیم و به دستاوردهای مورد نظر نرسیده ایم در نتیجه باید کارهای آن فاز را دوباره تکرار کنیم.



incremental بخشی از iteration است.

- هر iteration فعالیت های بسیاری را (برنامه ریزی - هماهنگی - ارزیابی نتایج) خواهد داشت بنابراین تکرار iteration در هر فاز RUP باید انجام شود.

Subject:
Year. Month. Date. ( )

برای انجام یک پروژه می‌توان چندین iteration در فازهای مختلف داشته باشیم هر دو این
به صورت incremental کارها را انجام می‌دهیم RUP پیشنهاد می‌دهد که مثلاً construction دو فاز
باشد و بقیه را در یک فاز انجام دهیم

- فاز شناخت (proof-of-concept prototype)

+ vision از طریق این فاز می‌توانیم سوالات اولیه پروژه را درک کنیم (پیاده‌سازی sample

- فاز بسط و رشد (architectural prototype)

+ جنبه‌های فنی یک سری prototype آماده می‌کنیم که جنبه‌های فنی را نشان می‌دهد performance قابل دستیابی است

- فاز ساخت (تولید محصول بتا (Beta-Realease

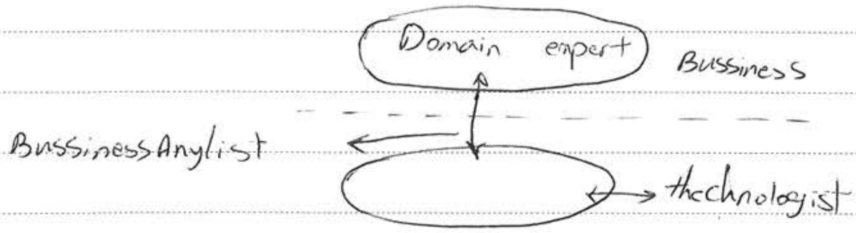
تولید محصول در این مرحله جاری کنترل ???

- فاز انتقال و تحویل محصول نهایی)

در نتیجه همان طور که در بالا گفته شد در هر فاز به یک محصول به صورت ناقص می‌رسیم

Domain Driven Design (DDD)

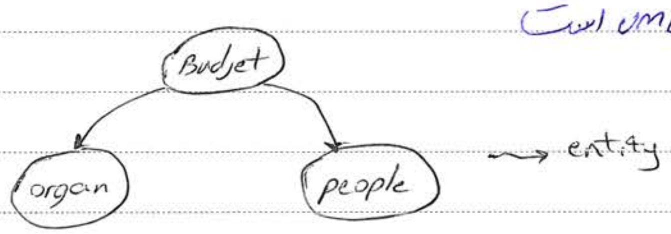
پروفسور در این حوزه Eric Evans است که کلیت تئوری آن به این صورت است که
برای ایجاد یک نرم‌افزارهای پایدار باید از اول با آن‌ها آشنا شویم و با آن کار کنیم. معمولاً برای برقراری این
ارتباط فردی به اسم Bussiness Analyst اضافه می‌شود.



PAPCO

Subject:
Year. Month. Date. ( )

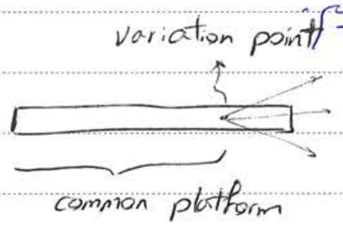
درنتیجه یک زبان فراگیر (Ubiquitous language) قابل فهم برای Domain expert و Technology Expert باشد.
یکی از زبان‌های قابل استفاده UML است




```
while (project Funded)
{
  showlatest software to domain ();
  updateDomainModelTogether ();
  updateCodeToReflectDomainModel ( );
}
```

8 software product line

در ابتدا در صنعت خودروسازی خط تولیدی ایجاد شد (mass production) ولی مشکلی که در آن وجود داشت این بود که تنوع (Diversification) در آن وجود ندارد برای داشتن هر دو این ویژگی‌ها یا به عبارتی بنابراین حالت mass-customization را داشتیم.
برای داشتن این حالت یک platform کلی ایجاد کرده (common platform) و سپس از آن که به تنوع لازم شد از آن نقطه مسیر را چند شاخه می‌کنیم.



software product line: is a set of systems that share a common set of core assets.

PAPCO

Subject:
Year. Month. Date. ( )

(MDD): model_Driven Development

a model is an abstract representation or a simplification of sth.

abstraction می‌گوید در هر سطح همان چیزهایی دیده می‌شود که نیاز است و لازم نیست وارد جزئیات شویم. در یک سیستم اگر بخواهیم با همه جزئیات سیستم درگیر شویم کار دشوار می‌شود ولی در صورتی که یک مدل از آن را در نظر بگیریم راحت‌تر خواهد بود. تأکید اصلی این روش بر این است که در فضای مدل شروع کنیم

- با انجام مدل سازی تولید [illegible] کد به طور خودکار انجام می‌شود.

model_Driven Architecture سه سطح در نظر می‌گیرد.

computation development model (CIM)

platform-independent model (PIM)

platform-specific Model (PSM)

Code

مدل ها می‌توانند مستقل از platform و یا وابسته به platform باشند. در سطح پایین‌تر مدلی که تعریف کرده‌ایم برای platform خاص قابل استفاده است.

یکی از ابزارهای مورد استفاده برای ایجاد مدل استفاده از نرم افزار webratio است که از سرور tomcat استفاده می‌کند و برای ایجاد connection string بهتر است از Apache DerBy استفاده شود.

PAPCO

Subject:

Year. Month. Date. ( )

Subject:
Year. Month. Date. ( )

رویکرد Agile:

خصوصیات پروژه های نرم افزاری:

- تولید اغلب نرم افزارها یک فرآیند تولید انبوه قابل پیش بینی نیست
- توسعه هر نرم افزار، تولید یک محصول جدید است.
- تولید قابل پیش بینی یک پارادایم اشتباه در تولید نرم افزار است.
- توسعه نرم افزار یک فرآیند تعریف شده نیست، چون ورودی ها فعالیت های فرآیند افراد هستند و وابسته به افراد است.
- در رویکرد چابک تأکید بر مقاومت در برابر تغییرات طی زمانبندی را به تولید محصول مورد نظر مشتری ترجیح نمی دهیم.

در سال 2001 بیانیه Agile یا Agile Manifesto توسط جمعی از متخصصین نرم افزار ارائه شد.

بیانیه agile:

- افراد و تعاملات بر ابزارها و فرآیندها ارجحیت دارد.
- نرم افزار قابل استفاده بر مستندسازی جامع ارجحیت دارد.
- همکاری مشتری بر مذاکرات قرارداد اولویت دارد.
- پاسخگویی به تغییرات بر پیروی از برنامه اولویت دارد.

PAPCO

Subject:
Year. Month. Date. ( )

روش‌های Agile یا چابک در یک طیف قرار دارند که به صورت زیر است:

traditional	Agile
- predictive	- adaptive
- process-oriented	- people oriented

متودولوژی‌های agile:

انواع متودولوژی‌های به کار رفته در agile عبارتند از:

- extreme programming (XP)
- Adaptive software Development (ASD)
- Dynamic systems development method (DSDM)
- ~~Dynamic~~ static system (SCRUM)
- (crystal)
- Feature driven Development (FDD)

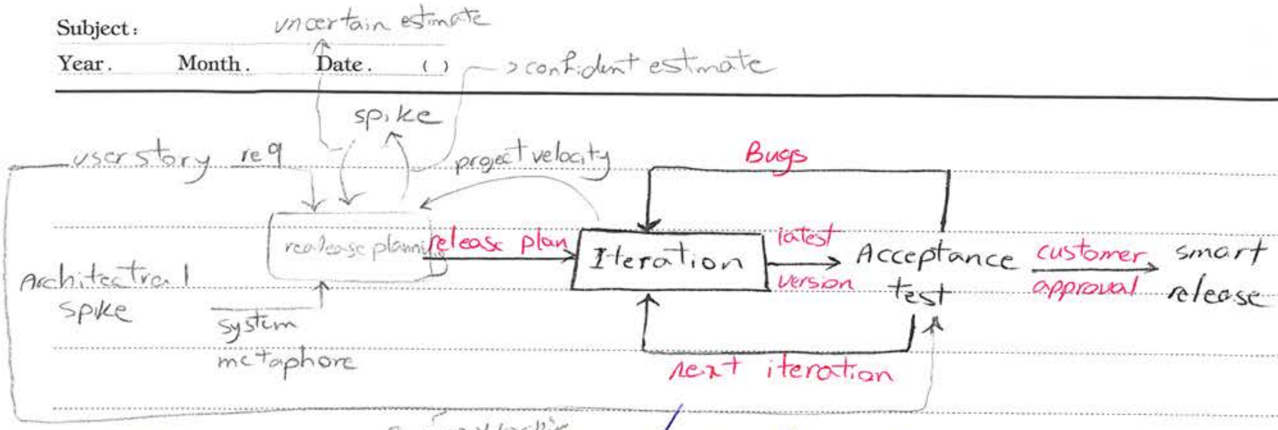
XP:

این روش توسط Kent Beck ابتکار شد. value های مختلف در این روش عبارت است از:

XP Values:

- communication
- Simplicity → تیم‌ها سعی کنند کارها را ساده انجام دهند
- Feedback
- Carriage → شجاعت
- Respect → احترام بین افراد تیم باید باشد.

Subject: Year. Month. Date. ( )



User stories : در xp مشتری خودش چیزی را که می‌خواهد می‌نویسد (توصیف نیازمندی ها)

Architectural spike : راه حل مشخص برای مسائل فنی

در ورودی user stories و Architectural spike برای رسیدن به Release planning لازم هستند

spike : برای برنامه ریزی سوالاتی داریم که برای جواب spike را بکار می‌برند

spike solution : spike در xp به معنای تلاش برای کاهش ریسک مرتبط با یک موضوع ناشناخته سیستم است (راه حل بالقوه)

Accept test : از دید بالا و مشتری باید تست شود.

system metaphore : یک تصویر کلی که با یک ایده از سیستم کلی ایجاد می‌کند، تا حدودی نقش معماری سیستم را دارد.

نکته: باید حدود یک تا ۳ هفته برای پیاده سازی زمان داشته باشید. در این روش مشتری دائماً در حال تحویل گرفتن یک چیز بدرد بخور است.

پس آنچه در ذهن مشتری است و آنچه باید پیاده سازی شود فاصله کم است که در زمان کوتاهی انجام می‌شود.

PAPCO

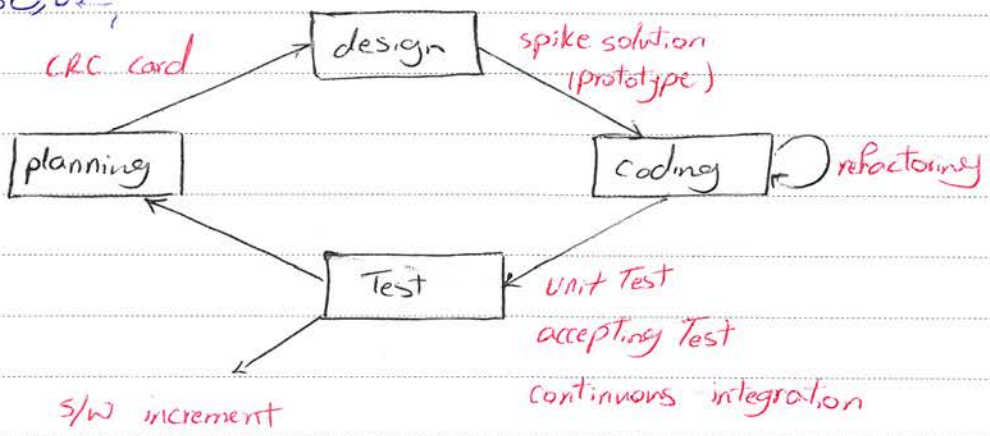
Subject:
Year. Month. Date. ( )

project velocity : سرعت پاره سازی و تحقق user story ها در این مرحله مشخص می گردد
سرعت نهایی تیم برای پروژه پیش رو است و با این سرعت برنامه زمان بندی می شود.

XP : قواعدی

- برنامه ریزی (planning)
- طراحی (designing)
- کدنویسی (coding)
- تست (Testing)

- بازنگری های مکرر برای بیان user story



Refactoring : به معنای تغییر در پاسخ پاره سازی کد بدون تغییر عملکرد آن است.

(I) برنامه ریزی:
- user story
- project velocity

(II) طراحی:
- system metaphor
- spike solution
- Refactoring

(you aren't gonna need it) YAGNI

Subject:
Year. Month. Date. ( )

III) برنامه‌نویسی:

- pair-programming: پشت هر دستگاه دو تا مهندس نرم‌افزار می‌نشینند یکی کد می‌نویسد و دیگری با نظارت بر آن در سطح بالاتر آن را چک می‌کند.

- Test driven: برای هر قسمت یک unit test آماده می‌شود

IV) تست:

- تست واحد (unit test): از طرف برنامه‌نویس و در سطح چک کردن کد است.

- تست پذیرش (acceptance test): از طرف کارفرما برای تحویل داده شده و تست می‌شود.

Software processing Improvement (SPI):

نمی‌توانیم انتظار داشته باشیم که یک شرکت از ابتدای کار خود از یک مدل فرایندی استفاده کند و تغییر نکند. در نتیجه می‌توان از یکی از روش‌های زیر برای کمبود مدل فرایندی استفاده کرد تا بگوید چگونه می‌توانیم آن را تغییر دهیم.

- CMM (capability maturity model)

↓

- CMMI

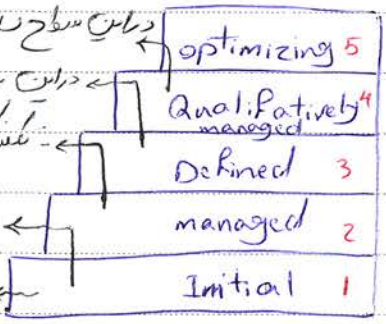
- PSP ← personal software process
- TSP ← Team software process

PAPCO

Subject:
Year. Month. Date. ( )

1 capability maturity model CMM :

توسط SEI در دانشگاه CMU بیان شد که مراحل به کار رفته در این روش به صورت زیر است:



5 ← در این سطح سازمان فرایندهای خود را از طریق جمع‌آوری داده‌ها و تحلیل آنها و کنترل و پایش قرار می‌دهد

4 ← در این سطح با توجه به گسترده استانداردسازی، استانداردها و یکپارچه‌سازی فرایندها سازمان خود را ایجاد می‌کند.

3 ← تکنیک‌های اولیه برای مدیریت پروژه استفاده می‌شود

2 ← فرایند تولید نرم‌افزار نامشخص و بی‌نظم است.

1 ← موفقیت بر مبنای تلاش‌های مشخص است و غیرقابل تکرار است
چون فرایند تعریف نشده است

optimizing: فرایندها از طریق بازخورد و پایش فرایندهای موجود به طور دائم بهبود می‌یابد و فرایندهای ابتکاری جدیدی برای پاسخگویی به نیازهای سازمان معرفی می‌شود.

در واقع هر سازمان از سطح اول شروع کرده تا وقتی که به بالاترین سطح برسد کارهای خود را در سطح level های بالا پیش می‌برد و در هر دوره می‌تواند در یکی از این سطوح باشد.

personal software process (psp):

هدف: کمک به مهندسین برای تولید محصول باکیفیت و با کمترین نقص (zero effect) در زمان‌بندی مشخص است. در واقع PSP شامل مجموعه‌ای از دستورالعمل‌ها و Deciplne برای این منظور است.

Team software process (TSP):

یک سطح بالاتر از حالت قبلی است و تأکید بر بهبود کارایی در سطح افراد و تیم‌ها و پاسخ به بازار است. در TSP یک سری دستورالعمل برای مهندسین دارد. لازم است هرکدام از اعضای تیم دستورات PSP را رعایت کند.

نکته: الان در وضعیت بهبود در ابعاد یک پروژه هستیم.

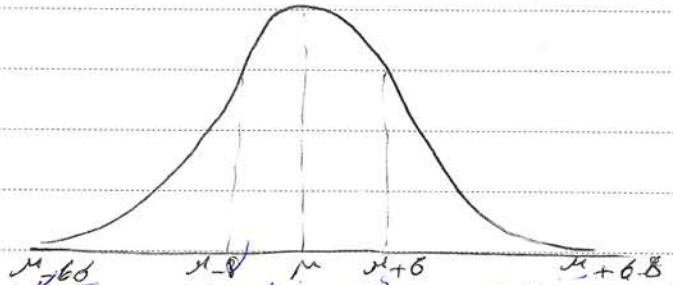
PAPCO

Subject:
Year. Month. Date. ( )

روش های کاربردی برای کنترل کیفیت محصول وجود دارد که عبارت اند از:

Sim Sigma:

یک فرایند تولید داریم و هدف این است که کمترین انحراف معیار کیفیت محصول را داشته باشد. این روش می گوید برای اینکه کمترین محصولات را داشته باشیم باید انحراف معیار محصولات از نرمال در بیشتر باشد.



در واقع ابتدا داریم پراکندگی محصولات در چه بازه ای است که بازه ای را تعریف کرده ایم که کیفیت محصول در آن محدوده تغییر می کند.

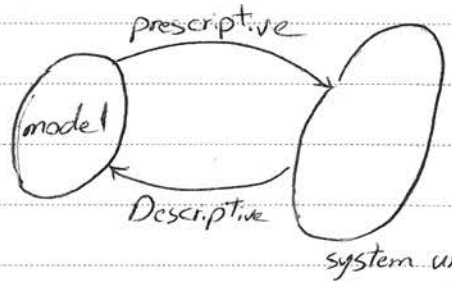
74

Subject:
Year. Month. Date. ( )

فصل چهارم

تحلیل و طراحی نرم افزار (SAD) با دیدگاه OOAD

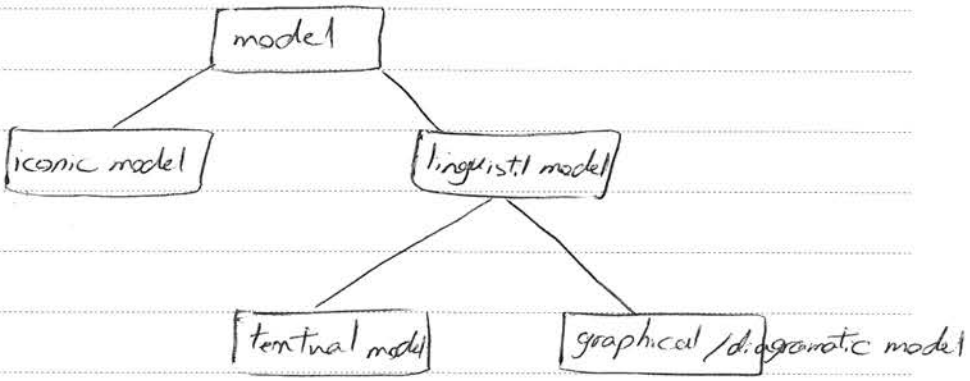
همانگونه که در مهندسی نرم افزار مدل سازی است. با مدلسازی می توان در هر سطحی به جزئیات آن سطح توجه کرد.



دو نوع مدل می توان در نظر گرفت:

(I) Descriptive: یک واقعیتی وجود دارد و می خواهیم برای آن مدلی ایجاد کنیم.

(II) prescriptive: واقعیتی وجود ندارد و فقط تصورات ذهنی است نه باید برای آن مدلی ایجاد کرد و محصول هنوز تولید نشده است و یک specification تولید می شود.



یک نوع دسته بندی از مدل ها به صورت زیر است.

(1) iconic model: یک آیکون مدلی را برای یک واقعیت بیرون ایجاد می کنیم و با شکل آن را نشان می دهیم.

Subject:
Year. Month. Date. ( )

2) textual model: به وسیله عبارت های نوشتاری و یا فرمول ها آن را بیان می کنیم $E=mc^2$

3) graphical models: به صورت گرافیکی و دیاگرام نمایش داده می شود در چون در مهندسی نرم افزار یکی از اهداف ارتباط برقرار کردن است این روش مناسب تر است. مانند RUP و UML

علی تحلیل و طراحی را می خواهیم با استفاده از زبان UML انجام دهیم (unified modeling lang) ارکان هایی که شامل syntax و semantic است به آن زبان می گوییم.

تحلیل ساخت یافته:
شامل دیاگرام های زیر است:
1) DFD
2) ERD
3) STD

که در این روش داده و پردازش را از هم جدا کرده است ولی در object-oriented تلفیق هر دو داده و پردازش را درون یک کلاس قرار داده و با هم هستند.

UML:
- دیدگاه مدل کاربر: از یک دیدگاه سعی می کنیم فقط کاربر ببینیم سیستم را از دیدگاه کاربر نشان می دهد.

- دیدگاه مدل ساختاری (structral model view): ساختار ایستای کلاس ها و اشیاء را مدل سازی می کند.

- دیدگاه رفتاری (Behaivral model view): جنبه های رفتاری و پویای سیستم را مدل می کند.

- دیدگاه پیاده سازی (Impl. model view): جنبه های ساختاری و رفتاری برای پیاده سازی مدل می شود.

PAPCO

76

دیدگاه مدل محیط (Environment model view): نمایشی از جنبه های ساختاری و رفتاری محیط است که سیستم در آن ساخته می شود.
لازم است که وقتی درباره یک سیستم صحبت می کنیم محیط دور و بر روی سیستم را نیز بررسی کنیم.

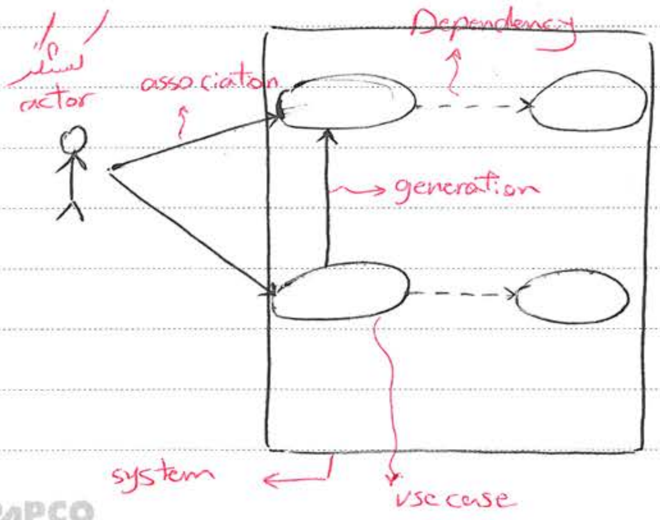
مدل مورد کاربرد (use case model):
هدف این است که از دیدگاه بیرونی به آن نگاه کند که از سه بخش تشکیل می شود:

1) Use case Diagram
2) use case narrative
3) use case scenarios

I) use case Diagram: یک دید سطح بالا به صورت black box (یعنی از بیرون به سیستم نگاه می کند) از سیستم ارائه می دهد.

II) use cas narrative: در خود مورد کاربرد باید شرح کلی برای هر مورد کاربرد ارائه شود.

III) use case scenarios: یک مسیر منطقی در یک مورد کاربرد است. یک [illegible] توالی عملی از گام های یک مورد کاربرد



Subject:
Year. Month. Date. ( )

سیستم: محدوده سیستم را نسبت به کنشگرها و تابعیتهای سیستم مشخص می‌کند

کنشگر: یک نقش است که توسط یک فرد، سیستم یا وسیله ایجاد می‌گردد و معمولاً در عملکرد و توفیق سیستم دارد.

مورد کاربرد: یک قابلیت کلیدی سیستم را مشخص می‌کند. یک هدف که سیستم باید برآورده کند.

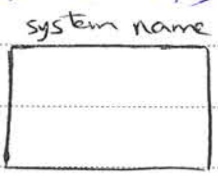
ارتباطات: تعامل بین کنشگرها و موارد کاربرد است.

وابستگی: نسبت های ارتباطی بین دو مورد کاربرد

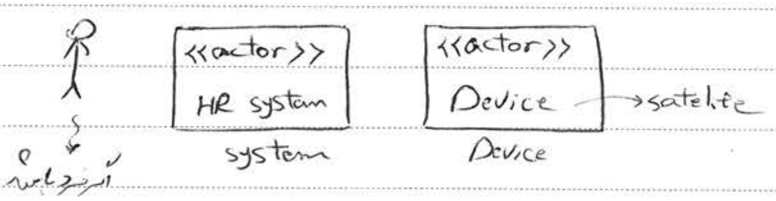
عمومی سازی: رابطه بین دو کنشگر، دو مورد کاربرد که در آن یکی از دیگری ارث می‌برد.

actor ها نیز می‌توانند با یکدیگر ارتباط داشته و از یکدیگر ارث برده شوند.

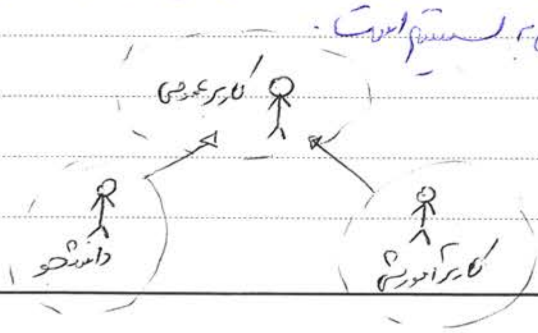
سیستم



Use case



نکته: use case های شامل چندین نقش‌ها هستند و چندین نفر می‌توانند یک نقش را داشته باشند و تعیین کننده سطح دسترسی به سیستم است.



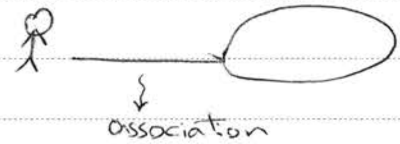
PAPCO

Subject:
Year. Month. Date. ( )

Use case (update account) (ثبت نام) بیانگر حالت action در صورتی که پدیده در آن مشخص باشد.

نکته: در usecase لازم نیست که به جزئیات پیاده سازی داخل بپردازیم بلکه فقط از دید کاربر قابلیت های سیستم را مشخص می کنیم

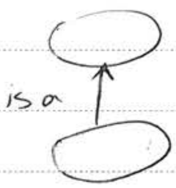
روابط



دارینه ها << >>

داشتن به ما کمک می کند رابطه بین دو usecase را مشخص کنیم و حالت های زیر را به خود می گیرد
<< include >>
<< extend >>

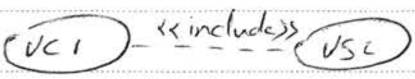
عمومی سازی

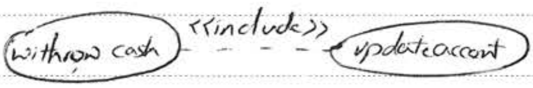


← مفهوم ارث بری

include: هنگامی که usecase برای اجرا شدن به usecase دیگر احتیاج دارد

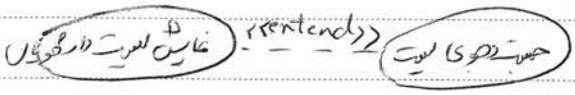
اجبار





Subject:
Year. Month. Date. ( )

extend : استفاده از آن در UC دیگر اختیاری است و الزامی به اجرای UC2 وجود ندارد و بسط داده شده آن است. علت استفاده از این حالت ها و تقسیم بندی های استفاده یکی بار دیگر در یک UC است.



مراحل ایجاد نمودار موردکاربرد (Use case Diagram):

قدم اول: تعیین content و محدوده سیستم

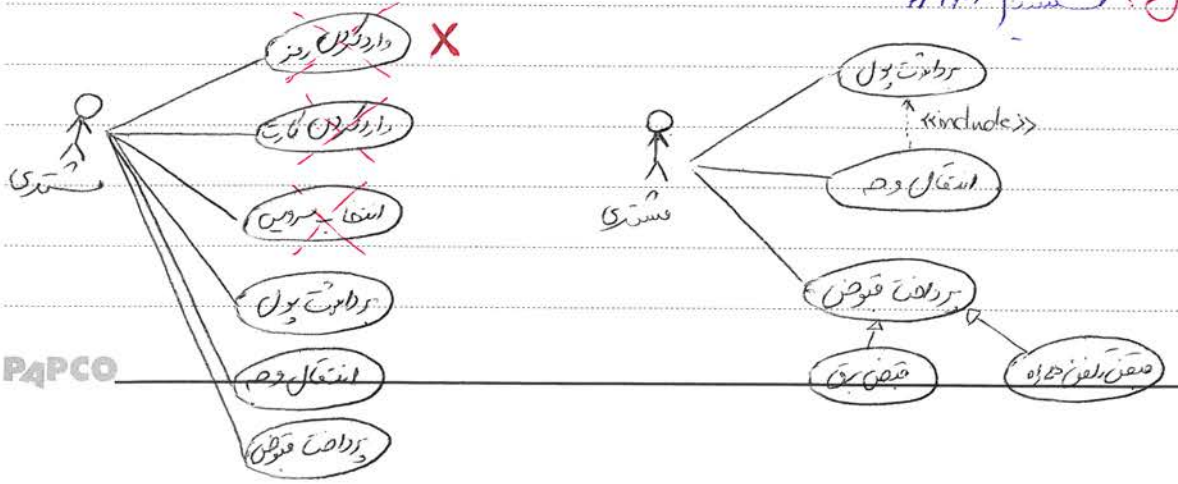
قدم دوم: تعیین کنشگر (actor) ←
1- سیستم چه ورودی هایی از کنشگر دریافت می کند.
2- سیستم چه خروجی هایی برای کنشگر آماده می کند.
3- سیستم برای انجام چه کارهایی به کنشگر کمک می کند.

قدم سوم: تعیین موارد کاربرد

قدم چهارم: تعیین ارتباط بین کنشگرها و موارد کاربرد

قدم پنجم: انجام بهبودهای لازم:
- وابستگی «include»
- وابستگی «extend»
- بررسی قابلیت عمومی سازی

مثال: سیستم ATM

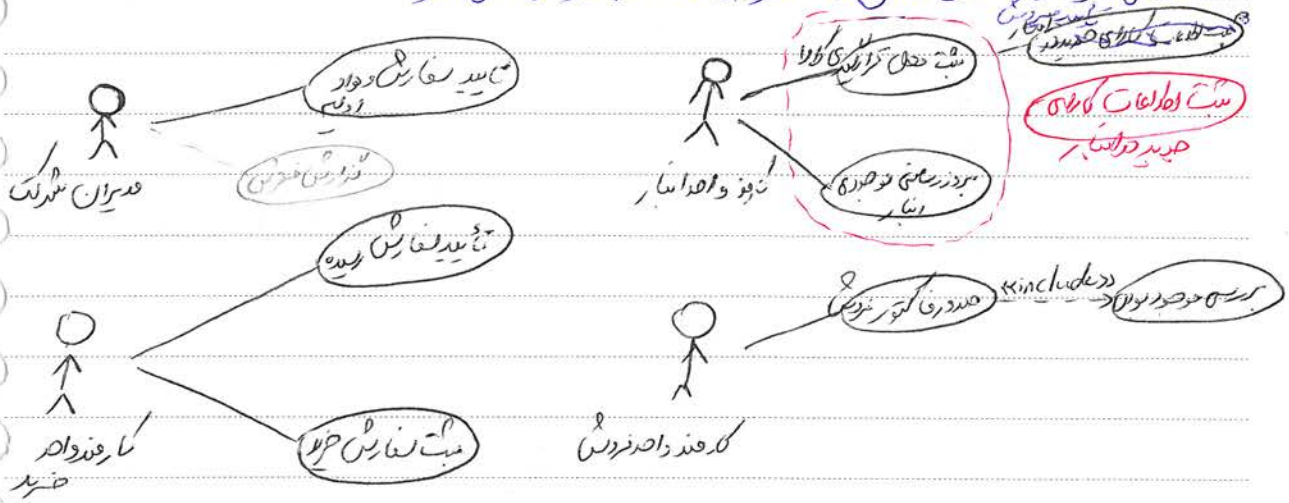


Subject:
Year. Month. Date. ( )

تعاملات کاربر با سیستم در use case ها نمایش داده می‌شود.

مثال: در یک شرکت تولید و عرضه محصولات داریم که وجود دارد:

- در واحد خرید براساس نظر مدیران شرکت، سفارشات در سیستم ثبت می‌شود و از طریق پست برای شرکت های تولیدکننده ارسال می‌گردد. کارمندان این بخش فهرست های دریافتی را با سفارش مقایسه کرده و تأیید خرید را برای واحدهای ارسال می‌کنند.
- در واحد انبارداری انبارگردان دلیل قرارگیری سفارشات رسیده از تولیدکنندگان را مشخص کرده و محصولات را در محل خود قرار می‌دهند. موجودی انبار و محل قرارگیری کالاها بروزرسانی می‌شود.
- در واحد فروش لیست سفارش خرید از مشتریان دریافت شده و در صورت موجود بودن کالا فروش صادر می‌شود و کالاهای مشخص شده عرضه شده در نظر گرفته می‌شود.



Subject:
Year. Month. Date. ( )

8. Usecase narrative

در UC قبلاً انتخاب واحد برای عنبر دانشجو مفهوم ندارد در narrative یک توصیف واضح از UC ارائه می‌دهیم.

این قسمت از چندین بخش تشکیل می‌شود:

۱- فرضیات (Assumption): بیان کننده وضعیتی از سیستم هستند که باید قبل از استفاده از مورد کاربرد برقرار باشند. این شرایط توسط مورد کاربرد تست نمی‌شود.

۲- پیش شرط‌ها (preconditions): بیان کننده وضعیتی از سیستم هستند که باید قبل از استفاده از مورد کاربرد برقرار باشند. این شرایط توسط خود مورد کاربرد تست می‌شود و در صورت برقرار نبودن، مورد کاربرد خاتمه می‌یابد.

برداشت حساب ← Authentication ← Assumption

مثال: مبلغ مورد نظر [ ] سقف HT ...۲۰۰ ولی ۳...HT۴ می‌زنیم در precondition جلوی ما را می‌گیرد و وارد فضای حساب و ... نمی‌شود.

۳- شروع مورد کاربرد (usecase initation): تعیین کننده نحوه شروع یک مورد کاربرد است. قبلاً انتخاب کاربر.

۴- دیالوگ: توصیف گام به گام تعامل بین کاربر (کنشگر) با سیستم برای مورد مشخص

عمل کاربر	پاسخ سیستم

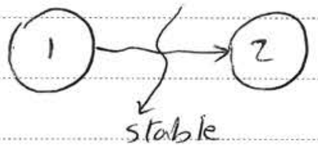
۵- خاتمه مورد کاربرد: روش‌های مختلف خاتمه مورد کاربرد را بیان می‌کند. ممکن است این خاتمه به روش طبیعی (خاتمه اجرا) باشد یا به واسطه بروز خطا، استثنا یا ...

۶- پس شرط‌ها (post condition): باید سیستم پس از اجرای مورد کاربرد در وضعیت پایداری باشد. بیان کننده وضعیتی از سیستم است که در هنگام خاتمه مورد کاربرد باید برقرار باشد.

PAPCO

Subject:
Year. Month. Date. ( )

به عنوان مثال تا وقتی که دستگاه پول را از کارت نگرده عملیات تکمیل نشود و یا باز کردن فایل یا ارتباط با DB باید حتماً بسته بشود.



مثال: سیستم انتخاب واحد [کد درس]

1) assumption ← مجاز بودن کاربر برای دسترسی به سیستم

2) pre condition ← کد درس مجاز باشد

3) usecase initiation ← انتخاب گزینه توسط کاربر (گزینه مثلاً درس)

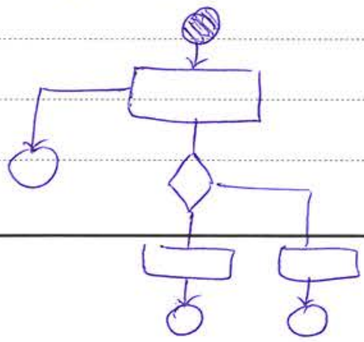
4) Dialog ←

	عمل کاربرد	پاسخ سیستم
	انتخاب گزینه درس	انتخاب درس برای دانشجو
سناریو اصلی (موفق)		
	" " "	نمایش پیغام تداخل
سناریو فرعی		نمایش پیغام بروز خطا به هر دلیل

5) usecase termination ← خاتمه طبیعی
← نمایش پیغام خطا

6) post condition: بستن ارتباط با داده، عدم کاهش ظرفیت درس در صورت بروز خطا

usecase senario: یک سری منطق در مورد کار چه است. در هر usecase یک باید سناریو وجود دارد. سناریوهای فرعی و اصلی هدف دنبال می‌واد به صورتی که در دیالوگ سیستم دنبال می‌شود را دارد.

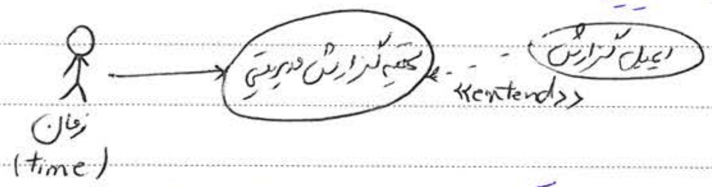


PAPCO

پروژه لدم: براساس انتشار RUP سند توصیف موارد کاربرد (Usecase spec) تهیه شود

Subject:
Year. Month. Date. ( )

مثال: در یک سیستم اتوماسیون اداری انتظار می رود نرم افزار به صورت خودکار در زمان های مشخص گزارشات مدیریتی تهیه نماید و برای مدیر شرکت ایمیل کند.



actor خود سیستم یا نرم افزار است علاوه بر آن زمان یا timer نیز به عنوان actor بیان می شود.

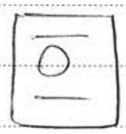
: class Diagram

توصیف ساختار ایستای سیستم را بر عهده دارد. اجزای تشکیل دهنده class diagram عبارتند از:

- Attribute
- operations
- association
- steriotypes
- inheritence

اجزای تشکیل دهنده کلاس: نام کلاس، خصوصیات، عملگرها

class {
private
≡ ⇝ value
operation



قابلیت مشاهده (visibility):

- public (+)
- private (-)
- protected (#): کلاسهایی که از کلاس اصلی ارث می برند.
- package (~): تمام مواردی که در یک package هستند.

Subject:
Year. Month. Date. ( )

قالب کلی بیان attribute در UML :

visibility, attribute name : datatype = default value {constraints (قید)}

studentName : String = "null" ;

توصیف opperation :

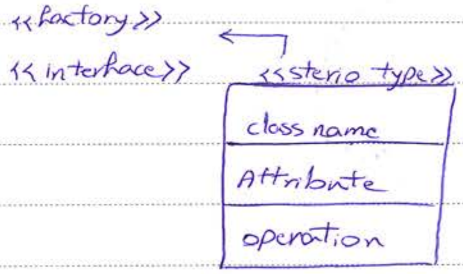
visibility, operation Name (array : datatype {constraint} {condition}

return datatype {constraints}

+ public AddCourse (courseID: STRING), return Bool

تحلیل و طراحی شیءگرا (OOAD) :

- UML
- use case model
- class diagram
- Design pattern
- class

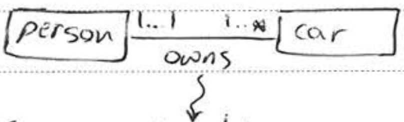


ارتباط بین کلاس ها :

1) Association :

بحث multiplicity ، یعنی تعداد را باید مشخص کنیم اگر از هر طرف (ا) باید از دو نقطه از زاویه دید ببینیم یا از زاویه فرد بررسی کنیم

حالت کلی در UML این است که هر طرف max و min دارد به عنوان مثال یک نفر یک تا بی‌نهایت ماشین می‌تواند داشته باشد.



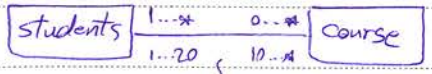
رابط association می‌تواند اسم داشته باشد.

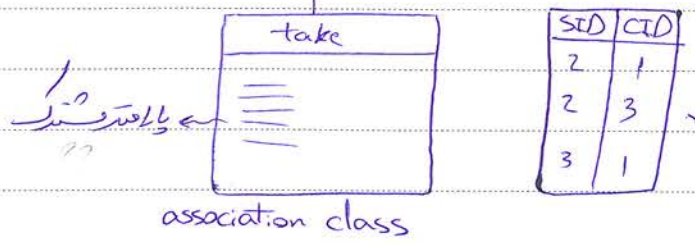
Subject :
Year . Month . Date . ( )

حالات خاص :

اگر 1...1 ← می تواند فقط و فقط یکی از آن وجود داشته باشد

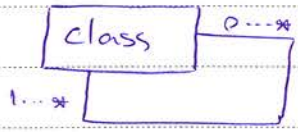
حالت تعمیم یافته : (many to many)

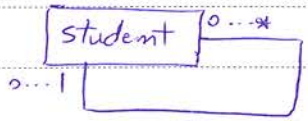




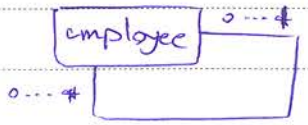
8. Reflexive Association 12

به معنای ارتباط اشیاء یک کلاس با یکدیگر است.

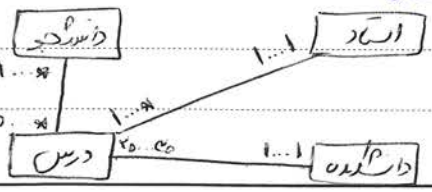




هر دانشجو می تواند با یک نفر رابطه داشته باشد ←



مثال : تهیه بخشی از کلاس برای سیستم پرسنل آموزشی



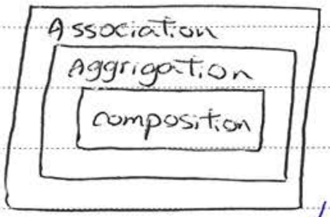
دانشجو
درس
استاد
دانشکده

PdPCO

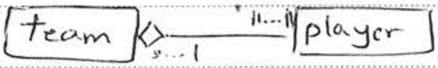
Subject:
Year. Month. Date. ( )

3) Aggrigation :

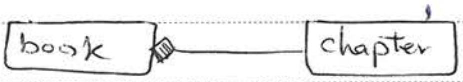
گاهی بین دو کلاس رابطه Association وجود دارد ولی کافی نیست. مثلاً اگر player نباشد پس اصلاً تیمی هم نخواهد بود.



در واقع کل اشیاء درون هم یک complex obj را تشکیل می‌دهند. به عنوان مثال obj های کلاس team از obj های کلاس بازیکن تشکیل شده است.



4) Composition :



کتاب بدون chapter معنی ندارد

در مرحله aggrigation ، lifespan مستقل از player است. از آن جایی که در بیشتر موارد نیاز است که یک موجودیت obj های مربوط به خودش را کنترل کند از این دو مفهوم استفاده می‌کنیم.

در حالت aggr. ، طول دوره حیات obj ها دست اولی و obj های اصلی نیست ولی در composition همین طور است.

```
aggrigation → class player {

              }
              class team {
                 list Team;
                 Add(player *p);
                 Team.add(*p);
              }
              player *p = new ----→
```
این obj به صورت مستقل وجود دارد در بازی.

```
remove();
```
team ندارد وجودش هست. در واقع بازیکن دوره حیات خودش را دارد.

Subject:
Year. Month. Date. ( )

composition →

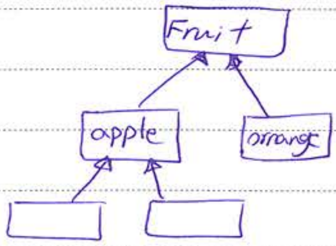
```
class Team {
    list Team;
    Add ( )
    {
      player *P=new player
      Team.add(*P);
    }
```

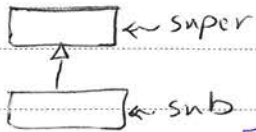
در این حالت اگر این obj یعنی team از بین برود بازیکن نیز از بین می رود.

نقطه مشترک : وجود یک obj کلی complex.

composition نوعی از agg است و agg نوعی از Ass. می باشد

5) رابطه Generalization :
از مفهوم ارث بری است





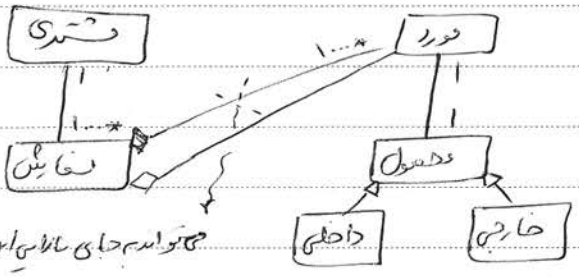
+ super class
+ sub class
+ Abstract class ← نمی توان یک instance از آن به نوعی روی مفهوم ارث بری ایجاد کرد
در واقع بر این مفهوم است که حداقل یک operation آن پیاده سازی نکرده است
مثلاً در C++ یک متد از آن virtual است.
+ concrete class ← همه operation های آن پیاده سازی شده اند

الزاماً موظف هستیم اول ارث ببریم سپس از آن instance بسازیم. (sub class)

مثال (سوال میانترم): برای شکل زیر عوارض class diagram را ترسیم کنید.
در یک شرکت توزیع دارو، سفارش مشتریان ثبت می شود. هر سفارش می توان شامل یک یا چند مورد باشد
هر موردی می تواند محصول داخلی یا وارداتی باشد. در این سیستم هر مشتری مجاز به ارائه یک یا چند سفارش

P4PCO

Subject:
Year. Month. Date. ( )



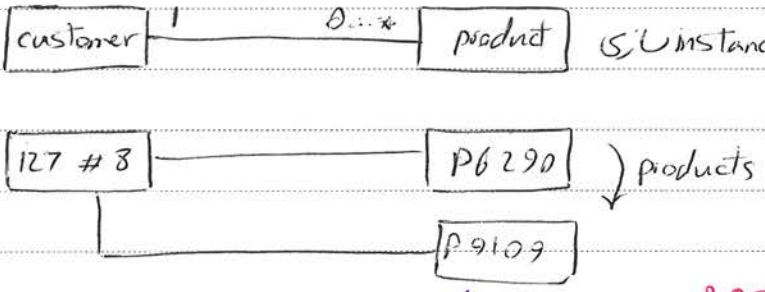
می‌توانید جای نمایش این دو را عوض کنید

مشتری شماره ۱۲۳ ، نمایش شماره ۱۲ را درخواست کرد شامل موارد ۳۴ ، ۴۵ و ۳۷ است. فروش شماره ۲۴ محصول #A13 است.

: Object Diagram

می‌خواهیم ببینیم در حین اجرا رابطه‌هایی که از کلاس‌ها ایجاد می‌شوند به چه صورت مدل می‌شوند

نمونه‌ای از اجرا



: Activity Diagram

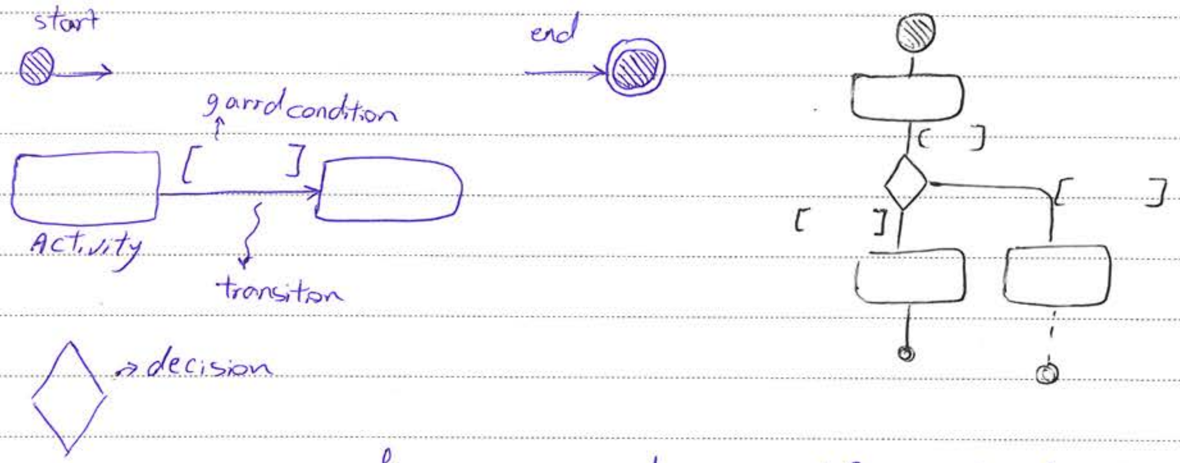
نمونه Flowchart و گردش کار را مدل می‌کند.

اجزای سازنده:

1. start
2. Activities
3. transition
4. Gaurd condition
5. Decisions
6. Merge point
7. concurency
8. Synchoronization
9. swimlane
10. end

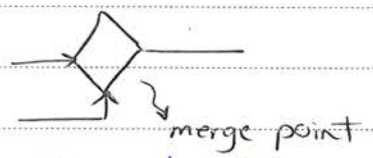
Subject:
Year. Month. Date. ( )

علامت‌ها و notation های به‌کار رفته برای هر یک از اجزاء به صورت زیر است:

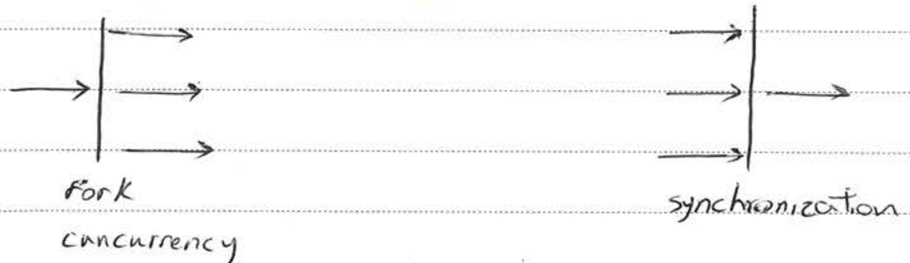


در واقع gaurd condition شرط برقراری رابطه در transition را نشان می‌دهد.

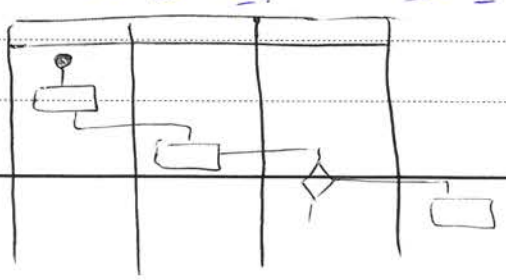
هر مسیر درون activity Diagram از شروع به پایان یک سناریو را مشخص می‌کند. گاهی دو مسیر در دیاگرام به یکدیگر می‌رسند: به نقطه تلاقی دو مسیر merge point می‌گویند.



در بعضی مواقع چندین مسیر به صورت موازی در همراه چندین thread به صورت همروند یا پارالل کار می‌کنند.
در این شرایط از notation زیر برای نمایش استفاده می‌کنیم که در حالت Fork چندین thread با هم شروع به کار می‌کنند و synchronization محل رسیدن این thread ها به پارالل را نشان می‌دهد.



در swimlane باید activity ها این نقش‌ها و actor ها را مشخص کنیم.



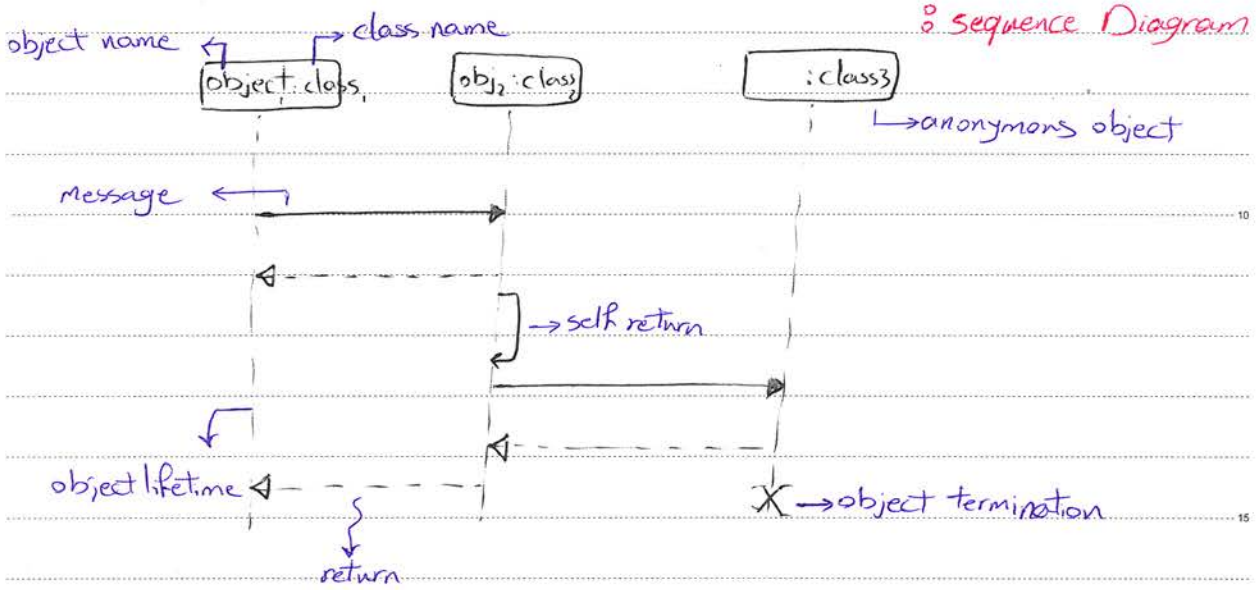
هر activity Diagram نشان دهنده یک usecase است که برای یک سناریو موفق و چندین سناریو فرعی برای خطاها و استثناها می باشد.

Subject:
Year. Month. Date. ( )

مدل کردن رفتارهای پویای سیستم (Dynamic)

behavior ←
- sequence Diagram } Interaction Diagram
- Collaboration Diagram }
- statechart Diagram

sequence Diagram



anonymous object یک کلاس است که استفاده می کنیم و استفاده ای از object نمی کنیم و نام آن اهمیتی ندارد.

self return یک متد از کلاس خودرا صدا می زند.

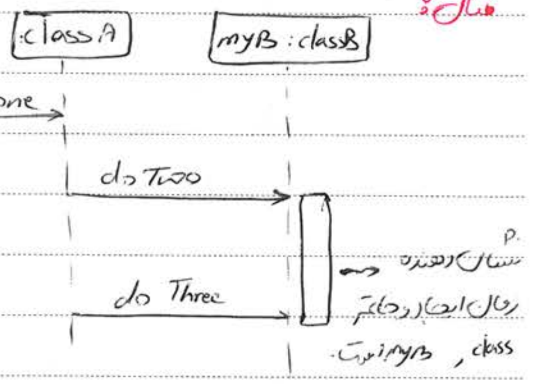
در این حالت دو message داریم ① سنکرون ② آسنکرون که در حالت آسنکرون زمانی پیامی را ارسال کرده و به ادامه کار خود می پردازد و در حالت عادی فلش را توخالی تر می دهیم ولی در حالت سنکرون صبر می کند تا آن کار به تمام انجام شود فلش صدا زدن متدها به صورت سنکرون در برنامه نویسی در این حالت فلش را توپر نشان می دهیم.

Subject:
Year. Month. Date. ( )

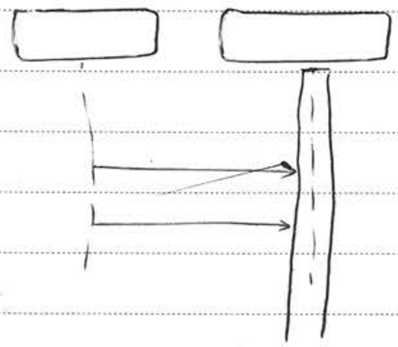
name زاده نشان دهنده آن instance است که در هنگام وقوع می خواهیم به ازای هر instance از این کلاس موجود کاری انجام شود در این حالت از anonymous object استفاده می کنیم.

مثال:

```
public class A
{
    private B myB = new B();
    public void done()
    {
        myB.doTwo();
        myB.dothree();
    }
    ....
}
```

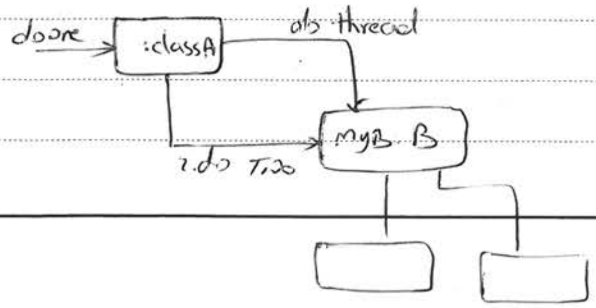


نکته: در صورتی که instance ساختن از کلاس B در خارج از کلاس A ساخته می شود به صورت زیر دیاگرام را رسم می کنیم.



Collaboration Diagram:

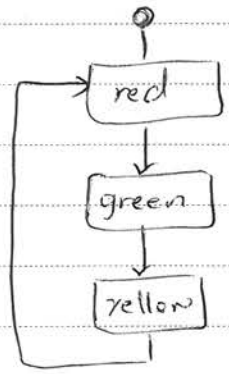
می خواهیم تعاملات بین کلاس های مختلف را نشان دهیم در این اشکال ارتباط هر کلاس با کلاس های دیگر که از یک زاده خاص استفاده می کنند به صورت یک خط نشان داده می شود دقیقا Message های تبادل شده نشان داده می شود.



PAPCO

Subject :
Year . Month . Date . ( )

state chart



stateهایی که در run time ایجاد می‌شود و حالت‌های مختلفی که سیستم در آن قرار می‌گیرد را نشان می‌دهد.

usecase realization

یکی از روش‌ها برای توصیف requirment استفاده از usecase model است و دنباله حالت‌ها سیستم را طراحی می‌کند.
هدف از این درس این است که از یک specification به code برسیم.

spec [... [... [code

what (چه چیزی می‌خواهیم) — how (چطور پیاده‌سازی کنیم)

بین specification ها اولین مدل usecase است زیرا چگونگی انجام کارها را نشان نمی‌دهد و فقط نیازها بیان می‌شود. بنابراین برای رسیدن به کد و مشخص کردن how از دو مدل زیر استفاده می‌شود:

uc realization ← { class Diagram → static
sequence Diagram → Dynamic

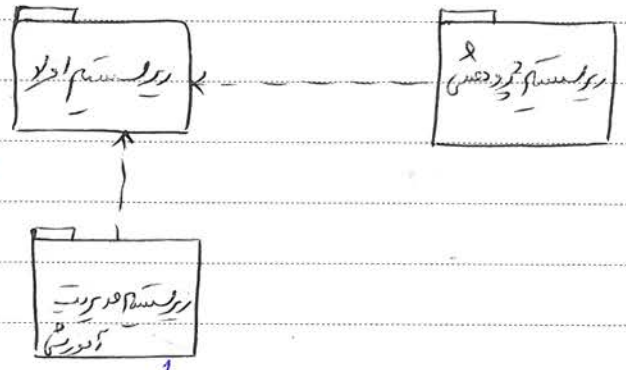
درنتیجه برای نشان دادن uc realization که پاسخی برای usecase model است باید class Diagram و sequence Diagram رسم شود.

P4PCO

Subject:
Year. Month. Date. ( )

package Diagram :

گاهی سیستم آنقدر بزرگ است که نیاز داریم به چندین زیرسیستم تبدیل کنیم. در نتیجه نرم‌افزار vscase گرچه کمک می‌کند و می‌توانیم زیرسیستم‌ها را به صورت مجزا بررسی کنیم.

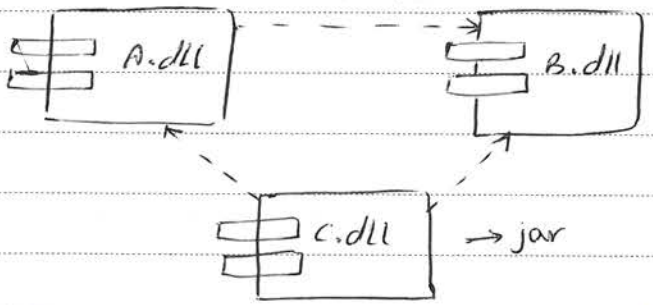


logical

طبق این دیاگرام می‌توانیم پیاده‌سازی هر زیرسیستم را به یک تیم واگذار کنیم و همچنین وابستگی‌ها بیان می‌کند برای انجام پیاده‌سازی زیرسیستم خرید و فروش به زیرسیستم انبار نیاز داریم.

Component Diagram :

تفاوت آن با package Diagram این است که در مثل تقسیم‌بندی‌ها و زیرسیستم‌ها به صورت منطقی انجام شده است ولی در component در جهت پیاده‌سازی عمل می‌کند.



physical

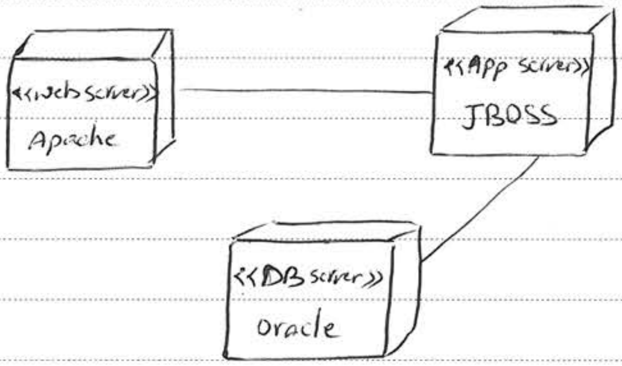
قابلیت آن این است که می‌توان با تغییر یک component نیاز سیستم را برآورده کرد و همچنین از component های قبلی استفاده مجدد داشته باشیم.

P4PCO

Subject:
Year. Month. Date. ( )

Deployment Diagram :

در سطح استقرار سیستم اجزا و سخت افزارهای مورد نیاز سیستم را تعیین می کنیم.



معماری نرم افزار :

معماری یعنی ارائه توصیفی فنی از یک سیستم که نشان دهنده ساختار اجزای آن، ارتباط بین آنها و اصول و قواعد حاکم بر طراحی آن و تکامل آن در طول زمان باشد.

معماری : ایجاد یک سیستم ساده می کنیم که کیفیت بهتر و هزینه کمتری را به دنبال داشته باشد. معماری یک دیدی از نرم افزار را فراهم می کند.

چه زمانی معماری لازم است:

1) ابعاد بزرگ
2) پیچیدگی زیاد
3) نیاز نوعی خاص
4) طول عمر زیاد
5) انعطاف پذیری در برابر تغییرات

معماری نرم افزار یعنی ارائه توصیف فنی از سیستم که نشان دهنده ساختار اجزای آن، ارتباط بین آنها و اصول و قواعد حاکم بر طراحی آن و تکامل آنها در طول زمان باشد.

P4PCO

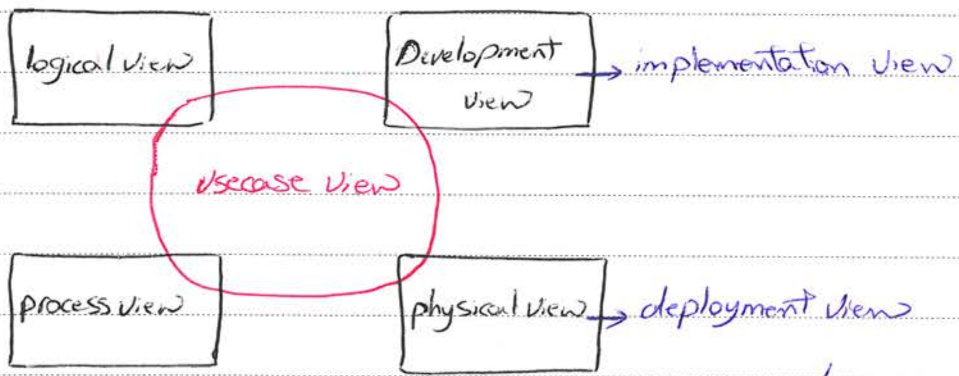
Subject:
Year. Month. Date. ( )

دو لایه‌های معماری و پیاده‌سازی را مشخص می‌کند.

4+1 View :

برای توصیف معماری باید 4+1 را پیاده کنیم. مبتنی بر UML است

4 view دیگر RUP پیشنهاد داده است عبارتند از:



Usecase مشخص کننده میزان تحقق functional requ. هاست به همین دلیل از 4 view دیگر جدا شده است.

1) Usecase view: به توصیف موارد کاربرد کلیدی سیستم که رفتارهای تأثیرگذار می‌پردازد

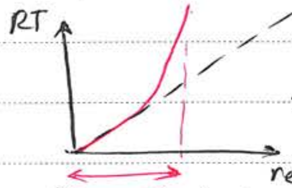
موارد کلیدی که در سیستم تأثیرگذارند

Subject:
Year. Month. Date. ( )

2_ دیدگاه منطقی: ساختار منطقی سیستم را مشخص می‌کند که شامل موارد زیر است:
- class diagram
- package diagram
- usecase realization

3_ process view: پردازه‌هایی (فرایند) که در زمان اجرا وجود دارند به همراه تعامل آنها نشان
کلاس‌ها و زیرسیستم‌ها به process و thread ها در زمان اجرا

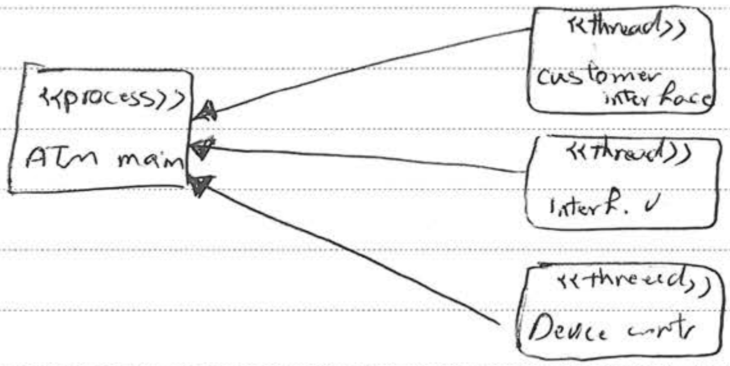
مسائل مهم در زمان اجرا عبارتند از:
1) deed lock
2) scalability ← باید با افزایش تعداد request ها زمان response time خیلی طولانی نباشد
3) throupút
4) concurrency ← تعداد thread همزمان در حال اجرا باشد.
5) Fault tolerance



Fault اگر در اثر قطع سرور mail و ... سیستم باید پایداری در نظر گرفت که در صورت وقوع اتفاق افتادن آن را طوری کنترل کند که user به end برساند.

large scale software: نرم‌افزارهایی که تعداد کاربران بسیار زیادی دارد و نیاز دارد تدابیری برای برقراری همه موارد بالا در نظر گرفته شود

مثال: برای سیستم ای‌تی‌ام داریم:



PAPCO

Subject:
Year. Month. Date. ( )

۴) دیدگاه توسعه‌دهنده (Development)
این دیدگاه نحوه دسته‌بندی کردن برنامه در قالب Component Diagram توصیف دارد.

۵) دیدگاه استقرار (Physical):
نحوه استقرار سیستم اشاره دارد. نحوه توزیع فیزیکی سیستم مختلف می‌شود.
(Deployment Diagram)

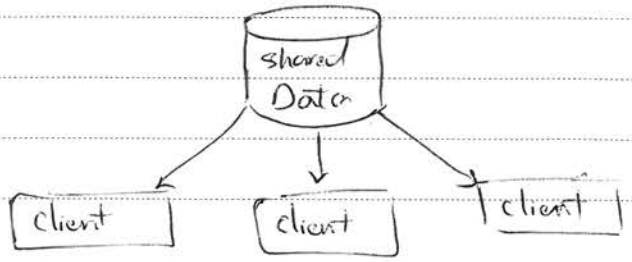
8 Architectural proof-of-concept
پاسخ به این سؤال که آیا حداقل یک راه‌حل برای نیازمندی‌های مطرح شده وجود دارد یا خیر.
در صورتی که یک سیستم را با 4+1 view توصیف کنیم این حوزه به خوبی پوشش داده می‌شود.

ویژگی‌های ایده‌آل:
- تکنولوژی‌های موجود
- مدل‌سازی با UML
- تیم‌سازی
- نسخه‌ی قابل اجرا

8 Architectural style

۱۱ سبک 8 Data centered

الف) Repository: برای تراکنش‌ها عمل می‌کند.



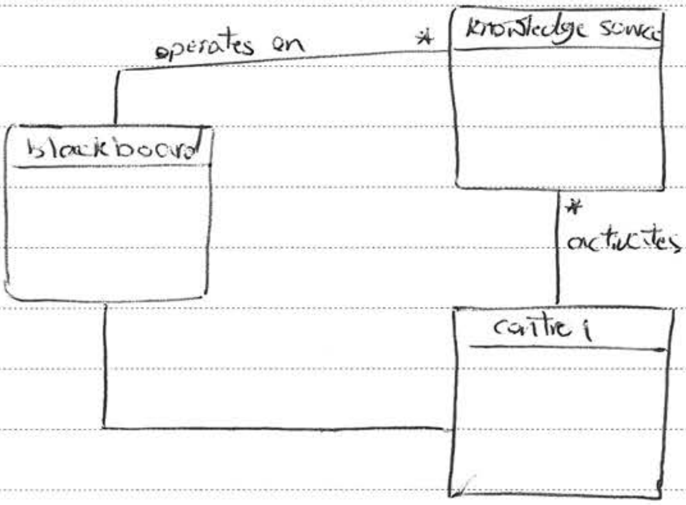
PAPCO

98

Subject:
Year. Month. Date. ( )

ب) black board

برای کاربردهای هوش مصنوعی نظیر پردازش سیگنال، پردازش تصویر و ... استفاده می‌شود.

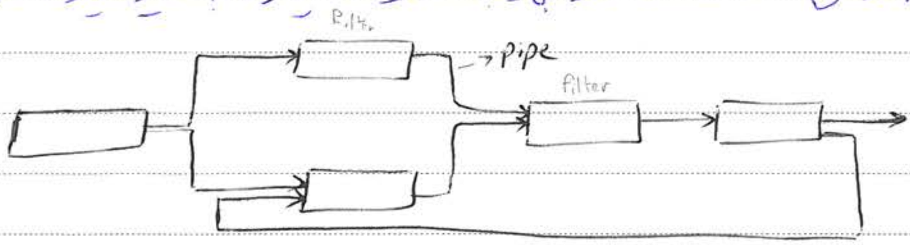


بعنوان مثال برای speech-to-text می‌توان از این معماری استفاده کرد. برای moderator دارد
و به عنوان controller هر کس هر کاری می‌تواند انجام دهد و هر بخشی از افراد دانا وجود دارد که هر یک برای
از دانش را دارند
هر جایی که لازم باشد که هر بخشی از تخصص‌های مختلف با یکدیگر استفاده کنند و ترتیبی در آن‌ها وجود داشته باشد
از این معماری استفاده می‌کنیم

12) سبک Data flow:

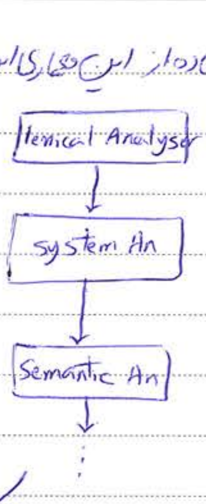
الف) pipe and filters: فیلتر یک جریان از داده‌ها را به عنوان ورودی می‌گیرد و یک جریان از داده‌ها را در خروجی تولید می‌کند.

pipe در نقش connector خروجی ایجاد شده توسط فیلتر را به سایر فیلترها منتقل می‌کند.



Subject:
Year. Month. Date. ( )

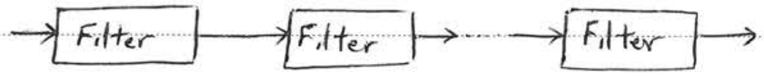
مثال: طراحی یک کامپایلر یکی از موارد استفاده از این معماری است



هر فیلتر ورودی داده تا بگیرد می گذارد و به مولفه بعدی ارسال می کند

1) سبک batch sequental:

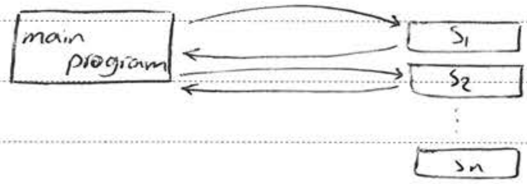
در صورتی که هر این فیلترها در یک خط به صورت سیال قرار گیرند تحت عنوان batch sequental نامیده می شوند.



یعنی در آن هیچ Fork وجود نداشته باشد.

3) سبک call and return:

الف) main program/subroutines:



subroutine ها مفاهیم مشابه function ها را دارند و می توان آنها را جدا در نتیجه آن را در main استفاده کنیم.

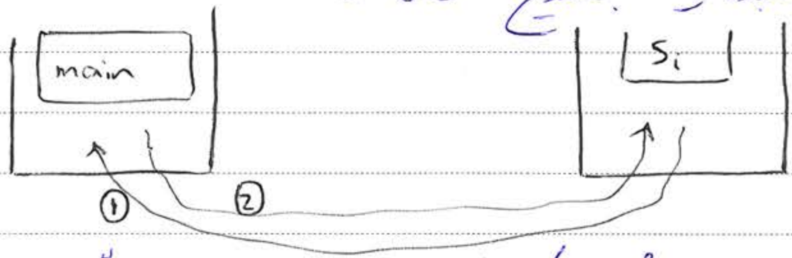
P4PCO

00

Subject:
Year. Month. Date. ( )

ب) Remote procedure call (RPC):

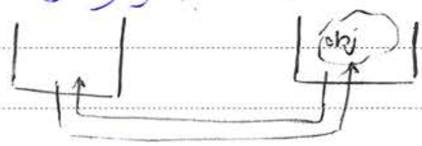
در ساده‌ترین حالت main program و subroutine ها بر روی یک ماشین قرار دارند.
در اینجا اجزاء را بر روی یک شبکه توزیع شده قرار می‌دهد.



یکی از کاربردهای جالب این روش در یک service oriented است. مثلاً Authentication در یک سیستم که از پورتال های مختلف استفاده می‌کند می‌تواند بر روی یک سرور جدا باشد تا لازم به وارد کردن username و password برای هر پورتال آموزشی، گردشگری یا ... نباشد.

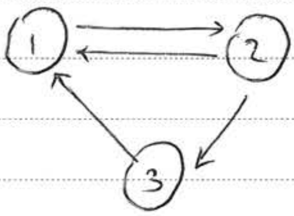
ج) Remote method invocation (RMI):

پیاده‌سازی آن در java وجود دارد. subroutine را درون یک obj قرار داده و فراخوانی می‌دهد. subroutine معادل فراخوانی یک متد از یک obj است. این روش مجاری را برای استفاده در object oriented آماده می‌کند که در آن obj ها برای یکدیگر message هایی را ارسال می‌کنند.



14) سبک object oriented:

ارتباط در اینجا براساس ارسال پیام است و طراحی را بر مبنای اشیاء و قابلیت انجام می‌دهد.



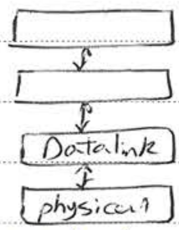
15) سبک layered (لایه‌ای):

هر لایه سرویسی برای سایر لایه‌ها ارائه می‌کند. برای ارتباط بین لایه‌ها از پروتکل مشخص استفاده می‌شود.

PAPCO

Subject:
Year. Month. Date. ( )

OSI model



در مدل های لایه ای از لایه های مختلف استفاده کرده که در آن هر لایه قسمتی از وظایف را انجام میدهد. هر مشکل وجود در یک لایه می توانند به راحتی عوض شود و تنها یک تغییر در آن ارتباط بین لایه هاست.

مزیت:

1) طراحی مناسب‌تر

2) تکامل راحت تر: در غیر این صورت لازم بود برای یک تغییر برنامه را عوض کنیم. هر لایه وظیفه تخصصی خود را دارد و منطق داخلی آن به صورت مستقل در سایر لایه ها قابل تغییر است.

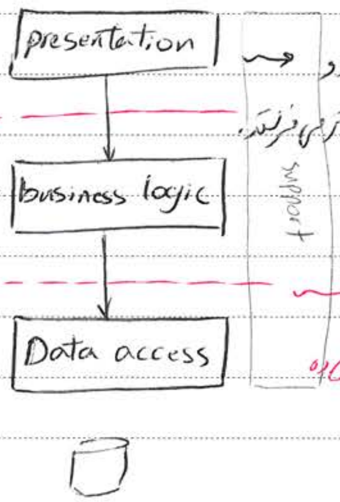
معایب:

1) overhead Function call برای API های میانی برای ارتباط لایه ها اضافه می شود.

2) همیشه به سادگی نمی توان وظایف را در قالب لایه ها تفکیک کرد.

سبک چندلایه (multi layer):

نسلی که در حرفه از لایه های بیشتر از همه استفاده می شود سبک 3 layer است که ساختار آن به صورت زیر می باشد:



UI و تعامل با کاربر را مدیریت می کند و از آن request ی برای لایه پایین تر می فرستد.

درخواست ها به چه حقوق را می گیرد قسمت از فرمول مربوط فقط فقط را می گیرد.

منطق کسب و کار در اینجا تأکید می کند.

نحوه ارتباط داده ها persistance، ذخیره و بازیابی اطلاعات (مدیریت داده ای)

اطلاعات در جایی ذخیره می شود: مستقل از اینکه از چه پایگاه داده و شیوه ای استفاده شده است.

PAPCO

Subject:
Year. Month. Date. ( )

جنبه‌های support مانند امنیت یا ورود کردن را در هر لایه‌ها قرار می‌دهیم

.Net	Java
C# form Asp.net	JSF
class BLL {	Spring
class DAL {	DAO

```
class BLL {
    call salary(id) {
        gethour( )
    }
}
class DAL {
    get hour(id)
    {
    }
}
```

تکنولوژی جدید در دات نت که مابین لایه‌های برنامه می‌شود LINQ است.

تفاوت Tier و Layer:

در multi layer و multi Tier از یک طرف لایه‌ها را براساس نحوه کارکرد آن‌ها جدا می‌کنیم حال ممکن است هر لایه بار پردازشی زیادی داشته باشد و نیاز باشد هر لایه بر روی یک ماشین مجزا انجام شود.

multi tier: می‌توانند بر روی یک یا چند ماشین مختلف با شرایط و زمان اجرای متفاوت قرار گیرد.

multi layer: layer یک مفهوم منطقی است و همه لایه‌ها بر روی یک ماشین قرار می‌گیرد.

در برنامه‌های client/server می‌توان از لایه‌های متفاوتی استفاده کرد:

one tier
two tier
3-tier
N-tier

مزیت این روش تقسیم کار مناسب است.

PAPCO

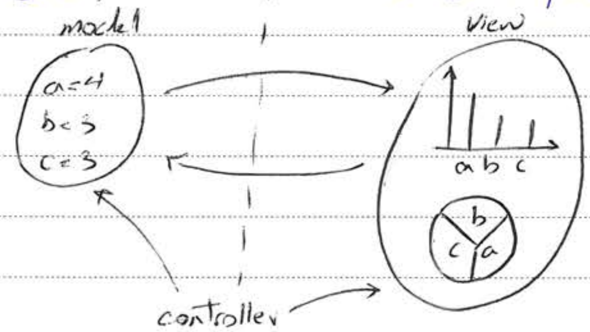
Subject:
Year. Month. Date. ( )

الگوی معماری (architectural pattern):

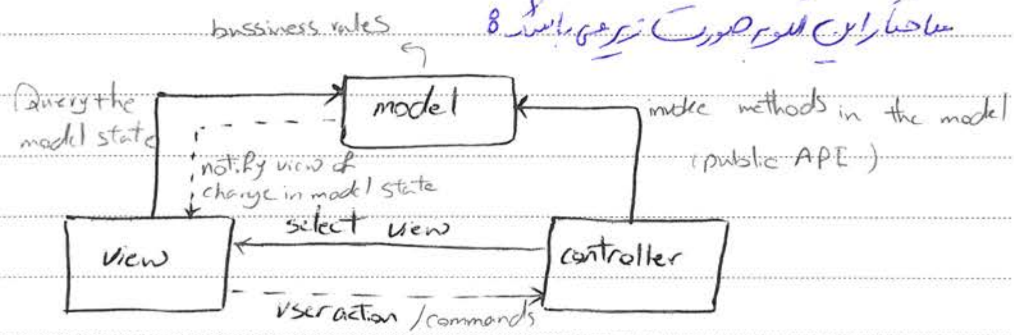
الگو راه حل آماده برای مسئله مشخص است. یکی از این الگوهای مشخص MVC pattern است.

MVC pattern (model view control):

یک مدل از مسئله می‌توانند در view ها بر گروه‌های متفاوتی نشان داده شود. در این صورت مدل مسئله را از view ها جدا می‌کنیم و بوسیله یک controller آنها را به یکدیگر متصل می‌کند.



ساختار این الگو به صورت زیر می‌باشد:



——→ method Invocation

– – –→ Events

در جاوا می‌توان از MVC struts برای این الگو استفاده کرد.

(ADL) Architecture Description language:

زبان توصیف معماری است که بدون استفاده از شکل در قالب یک زبان آن را مطرح می‌کنیم.

(SOA) Service oriented Architecture:

مانند تبادل انسان‌ها با یکدیگر است. دنیایی از سرویس‌ها با هماهنگی فراخوانی سرویس‌ها به یک هدف می‌رسیم

چگونگی ترتیب فراخوانی سرویس‌ها اهمیت دارد.

Subject:
Year. Month. Date. ( )

بخش پنجم:

کیفیت نرم افزار:

کیفیت میزان مطابقت محصول با انتظارات کاربر

تأمین کیفیت بر فعالیت‌های بازبینی و بازبینی محصولات پروژه و فرایندها جهت اطمینان از انطباق آنها با استانداردها و روش‌های اجرایی مناسب است.

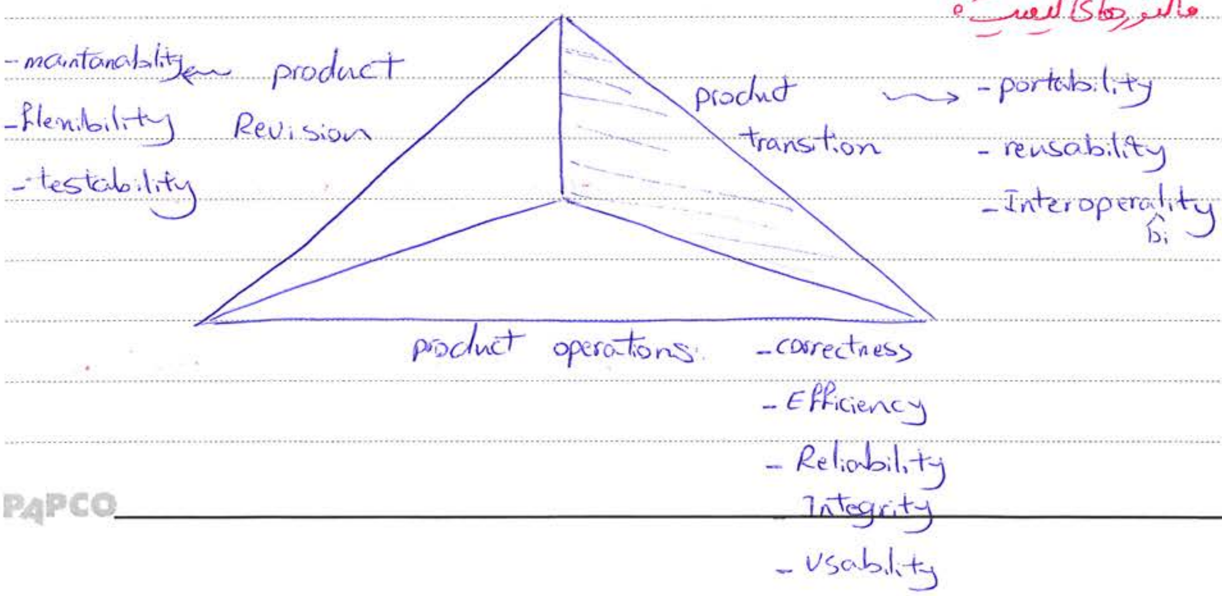
ابعاد کیفیت نرم افزار:

1) کیفیت داخلی: خصوصیاتی است که متناسب با مشخصات ثابت داریای که بر مبنا تعریف شده و توسط تیم توصیف و توسعه نرم افزار (برنامه‌نویس) اندازه‌گیری می‌شود.

2) کیفیت خارجی: خصوصیاتی است که متناسب با مشخصات پویای نرم افزار در زمان اجرا و توسط کاربر سیستم اندازه‌گیری می‌شود.

3) کیفیت استفاده: کیفیت استفاده از سیستم که توسط کاربر اندازه‌گیری می‌شود برحسب اینکه نرم افزار تا چه حد پاسخگوی نیازهای کاربر در محیط اجرای سیستم است.

فاکتورهای کیفیت:



PAPCO

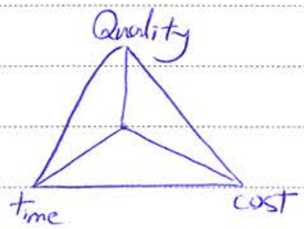
Interoperability: نرم‌افزار قابلیت همکاری با نرم‌افزارهای دیگر را دارد.

Flexibility: مشخص می‌کند چقدر راحت می‌توانیم نرم‌افزار را تغییر دهیم

Maintanability: تعیین کننده این است که چقدر راحت می‌توانیم تعمیر و نگهداری داشته باشیم

+ FURPS: نمی‌تواند به تنهایی برای بررسی کیفیت باشد. عوامل دیگر از استانداردها برای بررسی کیفیت محصول ISO 9126 است.

: Quality Dilemma



سوال: تا چه حد کیفیت نرم‌افزار لازم است؟

پاسخ: متناسب با هزینه و زمان

درصورتی که بخواهیم از حالت تعادل سه پارامتر تغییری داشته باشیم و Quality بیشتری ایجاد کنیم لازم است زمان و هزینه بیشتری را صرف کنیم.

: Good enough

اگر بخواهیم منتظر باشیم تا کیفیت به همان اندازه مورد نظر ایجاد شود ممکن است در زمان مورد نظر محصول آماده نشود. در نتیجه می‌توان بعضی از کیفیت‌ها را در مرحله دوم فراهم کرد در نتیجه برخی خصوصیات کیفی در مرحله اول نادیده گرفته و در مرحله بعدی اعمال می‌گردد.

مشکلاتی که این روش ممکن است ایجاد کند عبارت است از:

۱) نارضایتی مشتری و از دست دادن مشتری

۲) در سیستم‌های real time نمی‌توان استفاده کرد. به این حالت فرصتی برای تکمیل نداریم زیرا در همان لحظه استفاده می‌گردد و باید سریع‌تر باشد.

Subject:
Year. Month. Date. ( )

کیفیت محصول : در قبل بررسی شده است

کیفیت فرایند تولید محصول :

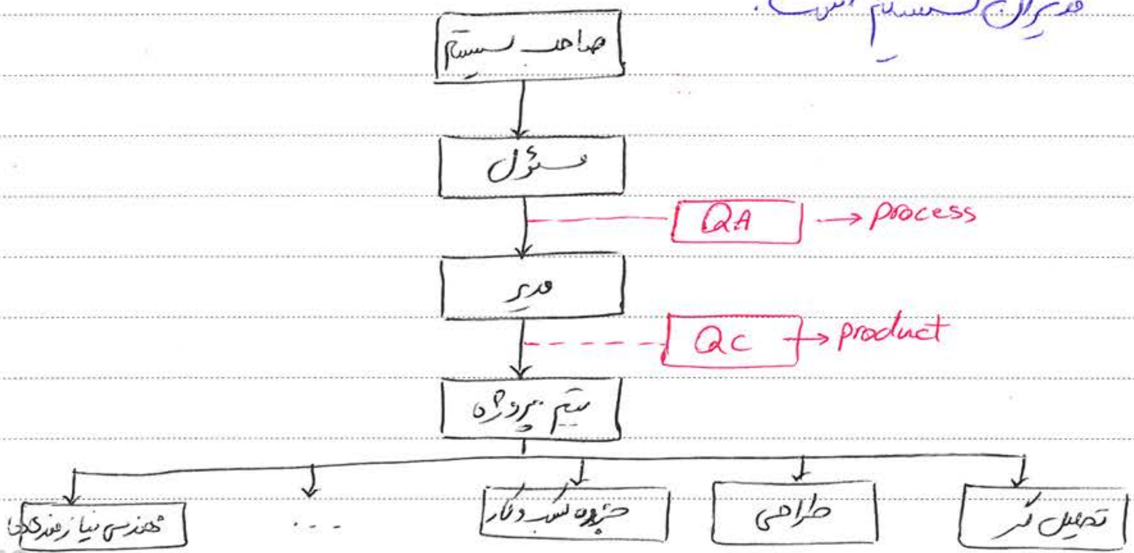
کنترل کیفیت (Quality control) : مجموعه‌ای از بررسی‌ها، مرورها و تست‌هایی که در سراسر فرایند توسعه نرم‌افزار مورد استفاده قرار می‌گیرند تا انطباق محصول با نیازمندی‌ها اطمینان حاصل شود.

تاکید بر feedback در چرخه تولید است. بوسیله یک سری کد تولید می‌شود که پس از آن بخش QC به بررسی و تست آن می‌پردازد و می‌تواند با دادن feedback ابزار برنامه‌نویس وضعیت محصول را مشخص و کنترل کند.

تضمین کیفیت (Quality Assurance) : مجموعه‌ای از عملکردهای گزارش‌گیری و ممیزی که کارایی و کامل بودن فعالیت‌های کنترل کیفیت را ارزیابی می‌کند.

Audit
گزارش صادره می‌تواند مورد استفاده برای مدیران باشد.

هدف از تضمین کیفیت تهیه داده‌های لازم در مورد کیفیت محصول و ایجاد اطمینان خاطر برای مدیران سیستم است.



Subject:
Year. Month. Date. ( )

QC در کنار تیم پروژه و در قرار گرفته و با بررسی کار تیم پروژه به مدیر feedback داده می‌شود و اگر کیفیت رعایت نشود باز گردانده خطا برگشت می‌رود.

QA کارهای نظارتی را انجام می‌دهد که آیا مدیر به درستی همه قسمت‌ها را نظارت می‌کند QC گزارشات مناسب از کیفیت داده می‌شود.

صفحه ۵:

۱۱) نیازمندی‌های نرم‌افزار پایه و اساس اندازه‌گیری کیفیت است.

۱۲) مجموعه استانداردهای تعریف شده به عنوان معیارهای توسعه نرم‌افزار در نظر گرفته شود.

۱۳) به نیازمندی‌های ضمنی (غیر صریح) توجه شود.
در واقع نیازمندی‌های سیستم به دو دسته implicit و explicit تقسیم می‌شود یعنی نیازهایی که در ذهن مشتری است ولی آن را بیان نمی‌کند به این دسته از نیازها implicit می‌گویند.

نکات تکمیلی ۵

- SQA به عنوان نماینده مشتری در تیم توسعه عمل می‌کند یعنی اعضای این گروه باید به نرم‌افزار از دیدگاه مشتری نگاه کنند.

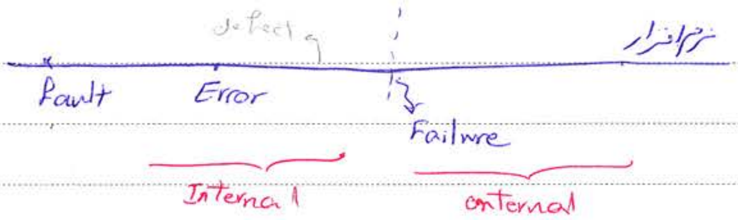
- بازبینی نرم‌افزار (Software Review) بازبینی نرم‌افزار مانند فیلتر عمل می‌کند و در مراحل مختلف فرایند تولید اعمال می‌شود.

- Formal Technical Review (FTR): جلساتی که نماینده‌های مختلف از بخش‌های گوناگون را بررسی کرده و در مورد کیفیت تصمیم می‌گیرند.

نکته: هدف از FTR یافتن error ها قبل از تبدیل شدن به Defect است.

PAPCO

Subject:
Year. Month. Date. ( )



Fault یعنی یک نقص که در سیستم وجود دارد و ممکن است هرگز خود را نشان ندهد، اگر در یک دوره اجرای برنامه به error تبدیل می شود. error ممکن است در بیرون پیدا کند یا نکند، اگر پیدا کند آن Failure می شود.

در صورتی که error در مرحله خودش شناسایی نشود تبدیل به defect می شود. مثلاً در مرحله نیازمندی ها اشتباهی داشته ایم اگر این اشتباه به مرحله پیاده سازی منتقل شود defect می شود.

اهداف جلسه FTR:

1) شناسایی خطاها
2) بررسی تطبیق نرم افزار با نیازمندی ها
3) بررسی رعایت استانداردها در نرم افزار
4) آشنایی اعضا با تمام بخش های نرم افزار

فایده: هر بخش نرم افزار با کمک بخش های دیگر آشنایی پیدا می کنند

نکات تکمیلی:

- FTR به صورت جلسه است.

- نکات جلسه FTR:

✓ افراد از قبل جلسه آماده باشند
✓ بین ۳ تا ۵ نفر
✓ طول جلسه کمتر از ۲ ساعت

PAPCO

Subject:
Year. Month. Date. ( )

گزارش جلسه بازبینی باید به سه سوال پاسخ دهد:

۱) چه چیزی بازبینی شده؟
۲) چه کسانی آن را بازبینی کردند؟
۳) نتایج جلسه چه بود؟

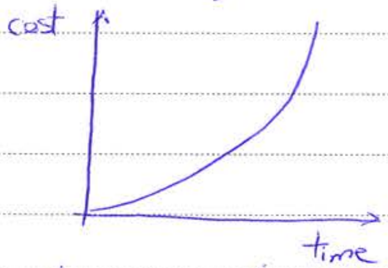
توصیه‌ها برای جلسه بازبینی:

۱- محصول بازبینی می‌شود نه تولیدکنندگان
۲- تعیین سرفصل دستور جلسه
۳- پرهیز از بحث و مجادله
۴- یادداشت نکات

تست نرم افزار:

تاکید بر یافتن اشتباهات، ناسازگاری‌ها و نواقص است.

در verification و validation هر دو از طریق تست بررسی می‌شود.

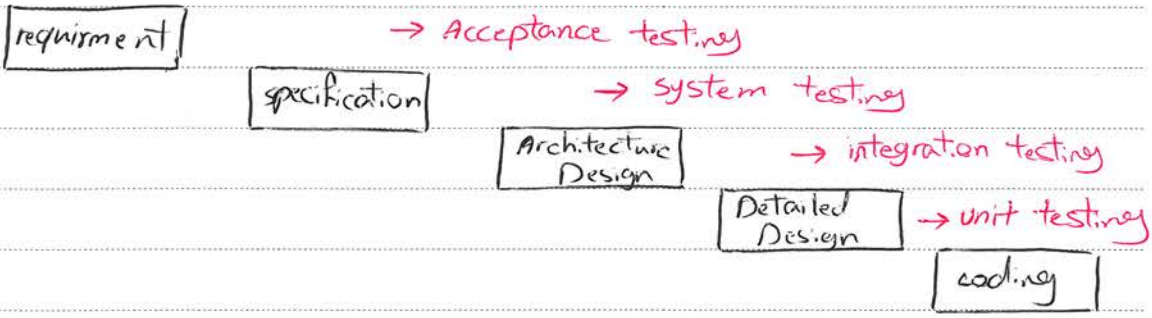


با توجه به نمودار بالا می‌توان نتیجه گرفت هزینه رفع یک error خیلی کمتر از رفع Defect است.

سطوح تست:

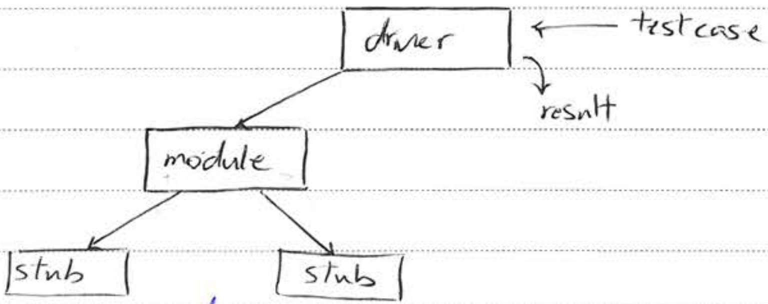
نموداری به نام V-model وجود دارد که این را به صورت کامل برای ما مشخص می‌کند.

Subject:
Year. Month. Date. ( )



۱) تست واحد (unit test):

کوچکترین عنصر قابل تست (وظیفه یا ماژول) از سیستم را تست می‌کند. در این تست مسیرهای اجرایی برنامه چک می‌شود. مثلاً ممکن است از برنامه یک ماژولی است که عمل sort را انجام می‌دهد حال می‌توان نقاط حساس را بررسی کرد که کار خود را درستی انجام می‌دهد.



driver می‌تواند یک main و یا scope ی برای اجرای ماژول باشد ولی ممکن است خود ماژول داشته باشیم زیر ماژول های دیگر باشد حال می‌توان بوسیله stub ها خروجی های حاصل از زیر ماژول های دلخواه را شبیه سازی کند. نمونه هایی مانند Junit می‌توانند unit test ها را بصورت اتوماتیک ایجاد کنند.

۲) تست یکپارچگی (Integration test):

هدف تست ماژول پس از قرارگیری در کنار یکدیگر است. مشکلات ممکن این است که در هنگام قرارگیری ماژول ها در کنار هم ممکن است برخی داده ها در روابط بین ماژول ها حذف گردد.

یکی از این روش ها big Bang همه ماژول ها را کنار یکدیگر قرار دهیم و یک جا تست کنیم که روش

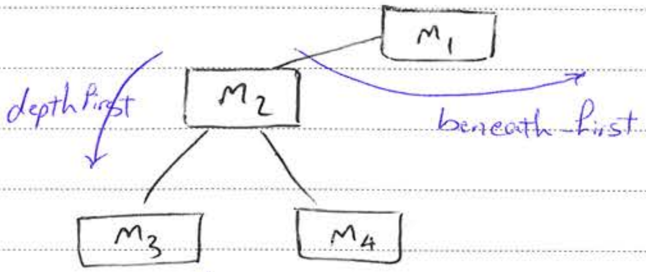
PAPCO

Subject:
Year. Month. Date. ( )

سخت است در روی حالت های مختلف را استوار می کند. یکی دیگر از روش ها، رویکرد افزایشی است:

رویکرد افزایشی:

۱) روش بالا به پایین:



در این حالت در ابتدا به صورت عمقی پیش می رویم و یا یک سطح را در ابتدا تست می کنیم. مشکلی که این روش دارد این است که اتصال این وجود دارد که زیر ماژول های مربوط به یک ماژول زیرلایه هنوز پیاده سازی نشده باشد.

۲) روش پایین به بالا:

ابتدا از زیر ماژول ها شروع کرده و با اتصال آنها به یکدیگر ماژول ها، cluster های بزرگتر ایجاد می شود.

Regression testing:

با اضافه شدن هر ماژول به نرم افزار در مرحله یکپارچه سازی تغییرات پیش آمده در برخی بخش ها تست می شود. تست رگرسیون به معنای انجام دوباره بخشی از تست های قبلی است.
این test مشکل دو روش قبلی را برطرف می کند و لازم است به ازای هر تغییر مجدداً محل هایی که تحت تأثیر قرار می گیرند بررسی شود.

P4PCO

1/2

Subject:
Year. Month. Date. ( )

۱۳) روش sandwitch:

یک روش ترکیبی است که قسمتی از بالا و بخش دیگری از پایین انجام می‌شود.

smoke testing: با بزرگ شدن نرم‌افزار ممکن است قسمتهایی از برنامه که تحت تست قرار می‌گیرند بسیار زیاد شود و تست کردن آنها زمان‌بر باشد. برای این کار از این روش استفاده می‌کنیم. یک زیرمجموعه از موارد تست که کارکردهای اصلی سیستم را شامل می‌شود و مسیرهایی که احتمال خطا بیشتری دارد را تست می‌کند.

ایده آن از سخت‌افزار می‌آید که همه قطعات را وصل کرده و اجرا می‌کنیم اگر دود بلند شد مشکل پیدا می‌شود.

13) تست سیستم (system test):

نرم‌افزار کامل شده مورد تست قرار می‌گیرد و با spec مطابقت آن بررسی می‌شود. علاوه بر این بررسی می‌شود که نرم‌افزاری که جزء یک سیستم مبتنی بر کامپیوتر است نیاز دارد که با سایر اجزا همکاری کند.

4) تست پذیرش (Acceptance Test):

نرم‌افزار آماده شده به وسیله کاربران نهایی سیستم به منظور یقین حاصل کردن برای استقرار نرم‌افزار تست می‌شود.

انواع دیگر تست:

- validation test: هدف بررسی اعتبار نرم‌افزار از دید کاربر نهایی است.

- Alpha test: در محل تیم توسعه‌دهنده انجام می‌شود و کاربران با حضور اعضای تیم توسعه نرم‌افزار را تست می‌کنند و خطاها یادداشت و ثبت شده و برطرف می‌گردد. این نوع تست در محیط کنترل شده است.

- beta test: در محل سازمان مشتری انجام می‌شود. برخلاف تست آلفا، اعضای تیم توسعه حضور ندارند. کاربران مشکلات را ثبت کرده و بعداً گزارش می‌کنند.

PAPCO

- پروژه نرم‌افزار
- vending machine
- پروژه کتابخانه

- configuration test: (تست پیکربندی) ارزیابی چگونگی عملکرد یک هدف تست در پیکربندی‌های مختلف.

پیکربندی یعنی در سیستم از چه ابزارهایی استفاده می‌شود (مثلاً DBMS یا MySQL) و متناسب با شرایط چه پیکربندی مناسب تعیین می‌شود. یا مثلاً استقرار نرم‌افزار در ۲۰ نقطه، ۳۰ نقطه آیا امکان مدیریت یابد.

- Recovery Test: بعد از بروز fault، نرم‌افزار بتواند دوباره فعالیت خود را آغاز کند یا اینکه fault-tolerant باشد.

در این تست شرایط failure برای سیستم فراهم شده و رفتار آن از نظر recovery تست می‌شود.

- security test: تست امنیت است. افرادی که تخصص دارند می‌توانند انواع حملات را بررسی کنند مثل SQL injection و XSS.

- stress testing: تست ارزیابی کارایی. هدف تست در شرایط غیرمعقول و افراطی یا بحرانی، مانند کم کردن منابع یا افزایش بیش از حد کاربران ... تا کاربران ... کاربر بررسی می‌کنیم.

- performance test: هدف ارزیابی قابل قبول بودن کارایی هدف تست با استفاده از پیکربندی‌های مختلف در شرایط عملیاتی ثابت.

با فرض اینکه ۱۰۰ تا کاربر همزمان داریم یک بار برنامه را با oracle10g بررسی کرده و یک بار با oracle 11g و کارایی سیستم بررسی می‌شود.

- load test (تست بار): ارزیابی قابل قبول بودن کارایی هدف تست در شرایط متغیر عملیاتی مانند تعداد کاربران در حالتی که پیکربندی ثابت است.

- installation Test (تست نصب): ارزیابی نحوه نصب نرم‌افزار در پیکربندی‌های مختلف و شرایط مختلف (کمبود حافظه اصلی) مثلاً در هنگام نصب اندازه و صفحات سیستم را بررسی می‌کنیم.

Subject:
Year. Month. Date. ( )

تست جعبه سیاه (black box testing):

کاری به نحوه پیاده سازی نداریم و رفتار سیستم را در سطح واسط کاربری بررسی می‌کنیم بدون توجه به نحوه پیاده سازی و منطق داخلی نرم افزار

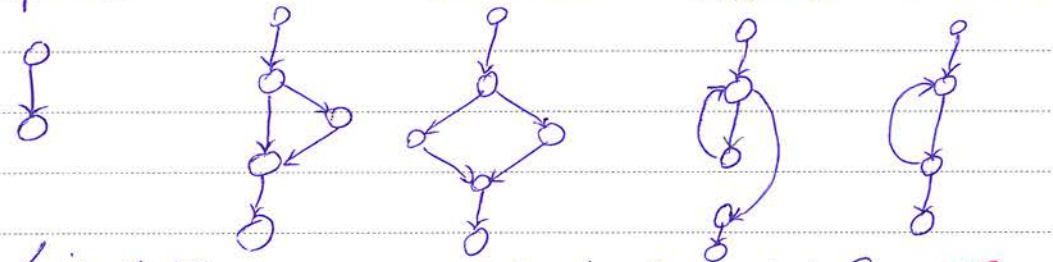
تست جعبه سفید (white box testing):

منطق داخلی نرم افزار و مسیرهای مختلف تست می‌شود. برای هر use case لازم است همه مسیرهای منطقی را بررسی کنیم تا حالت‌های نسبتاً زیادی ایجاد می‌شود و عملی نیست. می‌توان قسمت‌های مهم و حساس با این روش تست شود و خطا را حذف کنیم.

Basis path Testing

هدف: یافتن یک مجموعه پایه از مسیرهای اجرایی و تعیین test case برای هر مسیر

sequence if if-then-else For/while Do-while



تعداد نودهای داخلی گره‌ها هستند

قدم دوم: تعیین cyclomatic Complexity

CC: # region + 1 ← تعداد گره‌ها

CC: # decision + 1

CC: L - N + 2

L ← تعداد نودها

تعداد یال‌ها ↓

قدم سوم: تعیین مسیرهای مستقل در برنامه

قدم چهارم: استخراج Test case ها

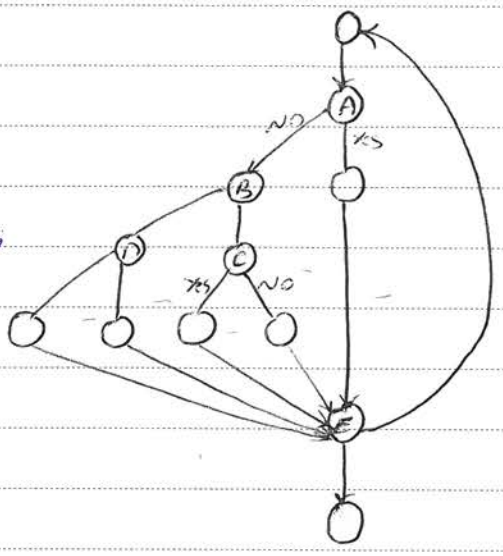
+ مجموعه تست باید به گونه‌ای باشد که همه مسیرها در مجموعه یا هم حتماً اجرا شود.

PAPCO

Subject:
Year. Month. Date. ( )

مثال: مطلوبست رسم گراف جریان کنترل و محاسبه پیچیدگی سیکلومتیک

```
Do
{ if (A) then {...};
  else {
     if(B) then {
        if(C) then {...};
        else {...}
     }
  else if(D) then {...}
     else {...};
  }
} while(E);
```



regions+1 : 5+1

decision+1 : 5+1

L_N+2 : 16-12+2 = 6

حال باید بتوانیم چند مسیر مستقل از گراف پیدا کنیم و پس برای این هر یک از مسیرها اجرا شود ورودی ها را به صورت مناسب تعیین کنیم.

1, 2, 3, 11, 12

1, 2, 4, 5, 8, 11, 12

⋮

مدیریت تغییرات (Change management)

در صورتی که تغییرات زیادی بوجود می آید لذا نیاز به مدیریت دارد. حال یک فعالیت چتری را با مدیریت تغییرات داریم که باید با روشهای مشخص انجام شود.

این بخش تحت عنوان software configuration Mang (SCM) شناخته می شود.

Subject:
Year. Month. Date. ( )

منابع تغییر:

- شرایط جدید بازار یا کسب و کار
- نیازهای جدید مشتری
- سازماندهی مجدد کسب و کار
- محدودیت بودجه و زمان

تفاوت بین change Management و Maintanance:

تفاوت آن این است که change managment از شروع پروژه وجود دارد و باید بررسی شود ولی maintanance پس از استقرار پروژه بوجود می آید و از همان تکنیک های changemanagment استفاده می کند.

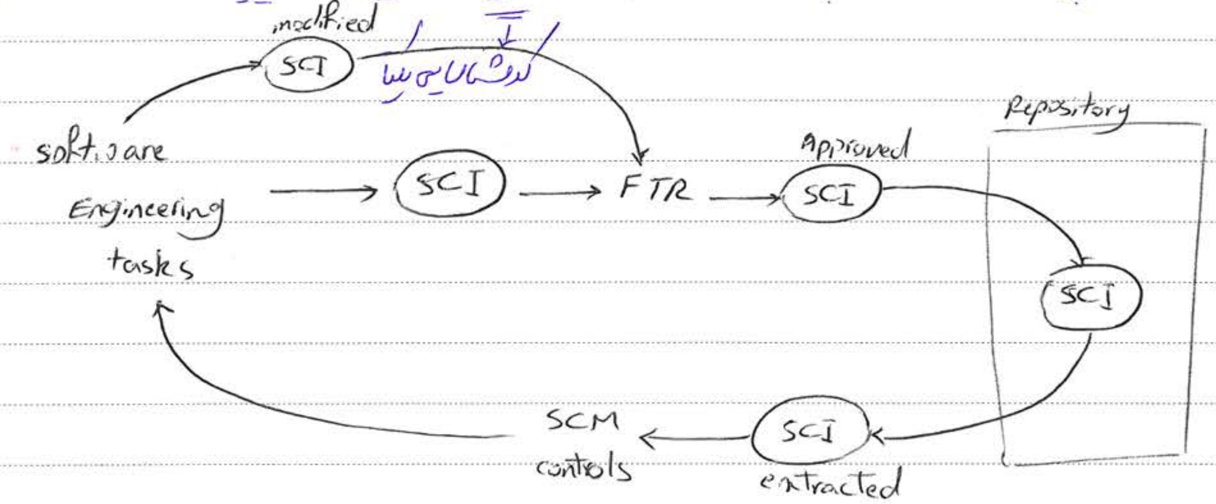
خروجی فرایند تولید نرم افزار:

- برنامه کامپیوتری (کد منبع، شکل قابل اجرا)
- مستندات
- داده ها

پیکربندی نرم افزار (software configuration) - SC:

software configuration item

- مجموعه مواردی که دربرگیرنده همه اطلاعات تولید شده به عنوان بخشی از فرایند نرم افزاری هستند (SCI)
- هر یک از اطلاعات که به صورت یک واحد به صورت یکتا شناسایی شده و تغییرات آن کنترل می شود.



change control bored

Subject:
Year. Month. Date. ( )

یک release از نرم‌افزار شامل هر چیزی از SCI ها است.

نکته: SCM جزو وظایف QA است و باید به آن توجه شود می‌تواند به نحوه انجام کار کنترل داشته باشد

سطوح بوده در Baseline:
specification _
Requirement _
Design _
source code _
Test/plan _

نکته: در صورت استفاده از ابزارهای مورد استفاده نیز باید در مدیریت پیکربندی دخالت شود.

قابلیت یک SCM:
ماهیتاً یک نرم‌افزار است و باید قابلیت‌های زیر را داشته باشد

versioning: نسخه‌های مختلف از SCI ها قابل نگهداری باشد
Dependency checking & check management: توانایی حفظ روابط بین SCI ها و حفظ پایداری آنها باشد.
requirement tracing: backward ، Forward باید بتوان هر قسمت از برنامه را به نیازهای موجود map کرد.
Configuration management: نگهداری از ورژن‌های مختلف برنامه و SCI های مختلف در کنار یکدیگر configuration را انجام می‌دهد.
مدیریت پیکربندی‌های مختلف نرم‌افزار برای release ها، milestone ها

PAPCO

Subject:
Year. Month. Date. ( )

- Audit trail : چه زمانی، چرا و توسط چه کسی تغییرات ایجاد شده

نرم‌افزارهایی از این نوع افزارها SVN و CVS است و در ویژوال جدید تر آن برای .net با قابلیت‌های Team Foundation است.

SCM process :

گام‌های لازم عبارت است از:

1) تعیین مواردی که در کنار هم یک پیکربندی را تعریف می‌کند.
2) مدیریت تغییرات
3) تسهیل ساخت نسخه‌های مختلف
4) اطمینان از اینکه کیفیت نرم‌افزار با تکامل پیکربندی در طول زمان حذف نمی‌شود.

load Test :

چندین پارامتر وجود دارد که باید در نظر گرفته شود:

N : تعداد کاربران همزمان

$B = N * T * D$

T : تعداد تراکنش‌های online در هر واحد زمانی

D : بار پردازشی مورد نیاز برای هر تراکنش

مثال: فرض کنید ۲۰۰۰۰ کاربر همزمان در دو دقیقه یک درخواست (تراکنش) داشته باشند. اگر هر تراکنش نیاز مند بارگذاری 3kB اطلاعات است. پهنای باند لازم؟

$$B = (20000 \times \frac{1}{2} \times 3kB) / 60 = 500\ kB/sec$$

P4PCO