به نام خدا

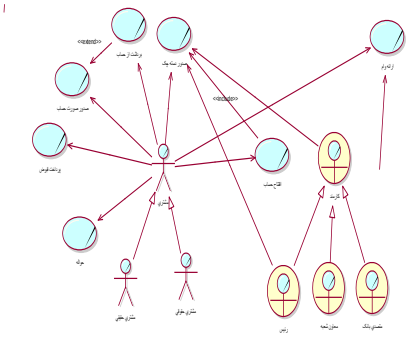
جزوه درس

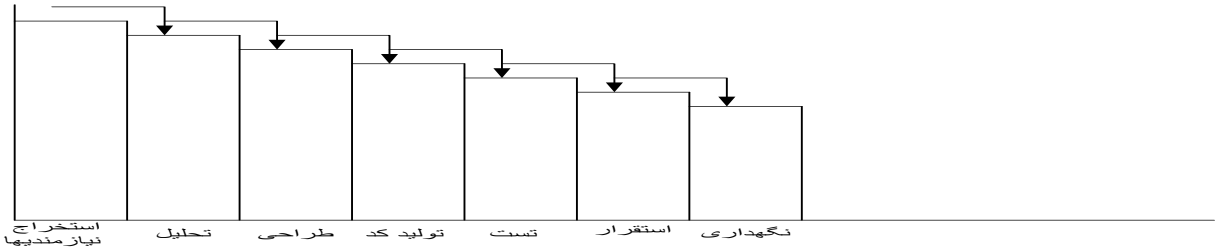
مهندسی نرم افزار 1

مهندس محمد ظروفی

جمع اوری :

نیما مسقدی





دانشگاه ملایر

نیمسال تحصیلی 902

Role ها :

1- برنامه ساز یا برنامه نویس Programmer/Developer

2- اشکال زدایی و اشکال یابی Test

3- تحلیل Analysis

4- طراحی Design

5- استقرار نرم افزار (نصب نرم افزار) Deployment

فرق یا ویژگی یک محصول نرم افزاری چیست که انرا ازمحصولات دیگر مثلا محصولات سخت افزاری متمایز می کند ؟

ویژگی یک محصول نرم افزاری در مقایسه با دیگر محصولات عبارتند از :

1- فرایند جایگزینی و استقرار نرم افزار نمی تواند به صورت دفعه ای صورت پذیرد. یکی از چالش های data است (داده) مثلا اگر نرم افزاری 30 سال در شرکتی باشد و نرم افزار جدیدی نصب شود باید تمام داده ها به نرم افزار جدید منتقل شود . در صورتی که انتقال به صورت رکورد به رکورد باشد مهاجرت موفقی را نداشته ایم .

2- نرم افزار قابل کپی برداری نیست. به عبارت دیگر ، بر خلاف دیگر محصولات ، نرم افزار تولید نمیشود بلکه توسعه پیدا می کند . مثلا مثل سخت افزار نیست که یک مدل طراحی شود و نمونه های بعدی از آن تولید شوند.

مولفه – قطعه Component نگرش مولفه گرا Component Oriented

برای نرم افزار های اختصاصی به دلیل متمایز بودن نیاز های کاربران نمیتوان یک نرم افزار اختصاصی در کاربردهای مختلف مورد استفاده قرار گیرد. استفاده از مولفه پیش ساخته باعث کاهش هزینه های می شود.

مهندسی نرم افزار به مقوله مدلسازی می پردازد. تجزیه و تحلیل به صورت شی گراست .

عناوین مطالب :

1- اشنایی با مفاهیم شی گرا

2- اشنایی با مفاهیم مهندسی نرم افزار

3- اشنایی با مدل های مختلف توسعه نرم افزار

4- اشنایی با نرم افزار رشنال رز

5- اشنایی با مفاهیم مدیریت پروژه های نرم افزاری

نرم افزار مجموعه ای از دستور العمل هاست که تحت عنوان یک برنامه کامپیوتری (نرم افزار)برای انجام مقاصد تعریف شده مورد استفاده قرار می گیرد.

هدف مهندسی نرم افزار **توسعه سیستم های نرم افزاری** می باشد Software Development

فرایند توسعه نرم افزار به منظور توسعه ی (**شروع از صفر و به انتها رسیدن**) یک سیستم نرم افزاری باید دنبال شود/طی شود.

مفهوم Process که یک مفهوم کلیدی در مهندسی نرم افزار است.

نگرش مولفه گرا در سیستم نرم افزاری قابل استفاده نیست چون نیاز های کاربران خاص است . ایجاد مولفه هایی که برای کاربردهای خاص هستند اسان نیست. مولفه مثل اجر در ساختمان سازی است . کوچکترین مولفه در دنیای نرم افزار دستور العمل است .

دستور العمل – متد – کلاس – مولفه – مولفه بزرگتر

هرچه مولفه بزرگتر شود اختصاصی تر می شود و استفاده از مولفه های کوچکتر کارامد نیست . بنابراین:

3- استفاده از نگرش مولفه گرا در فرایند توسعه نرم افزار بر خلاف دیگر محصولات امکان پذیر نیست .

علت : تنوع مشتریان – هرچه مولفه بزرگتر شود اختصاصی تر می شود – احتمال استفاده از آن در کاربردهای دیگر کاهش میابد.

انواع نرم افزار ها :

1- نرم افزاری های سیستمی System Software

نرم افزار هایی هستند که عمدتا به منظور سرویس دهی و ارائه خدمات به دیگر نرم افزارهای سیستمی و برنامه های کاربردی مورد استفاده قرار می گیرند. مثل OS ، سرویس دهنده های مختلف (سرویس دهنده وب ، دیتا ، سیستم فایل،پست الکترونیک)

کاربر بطور مستقیم با این نرم افزار در ارتباط نیست.

2- نرم افزارهای بی درنگ ، عکس العمل سریع real Time

در این نرم افزارها پاسخ باید در فاصله زمانی مشخصی داده شود . این نرم افزارها باید از جانب OS و سخت افزار و نرم افزارهای سیستمی حمایت شوند / پشتیبانی شوند .

این نرم افزارها از مولفه های زیر تشکیل شده اند :

1- مولفه جمع اوری اطلاعات 2- مولفه کنترل و پردازش و تحلیل 3- مولفه پالایش و هماهنگی 4- مولفه احراز صدور فرمان

3- نرم افزار های تجاری

این نرم افزار ها عمدتا به منظور پردازش حجم بالایی از اطلاعات مورد استفاده قرار میگیرند . کار آنها عمدتا ایجاد ، حذف و ویرایش داده و جستجو است . این نرم افزارها معمولا با یک یا چند سرویس دهنده بانک اطلاعاتی در ارتباط هستند .

4- نرم افزارهای علمی مهندسی

این نرم افزارها برای محاسبات پیچیده مورد استفاده قرار می گیرند.

CASE : Computer Aided Software Engineering نرم افزاری در مقوله توسعه نرم افزار که مورد استفاده تیم توسعه نرم افزار قرار می گیرد.

Autocad : در رشته عمران و معماری Computer Aided Design

5- نرم افزار های تعبیه شده Embedded Software

برای استفاده در لوازم خانگی ایجاد شده اند . مثل نرم افزارهایی که روی موبایل یا GPS قرار دارند.

6- نرم افزار های تحت وب Web Application

از دو مولفه تشکیل شده اند :

- مشتری (سرویس گیرنده) : کارش تولید درخواست ،ارسال درخواست و تحلیل پاسخ است .
- سرویس دهنده (خدمت گزار): کارش دریافت درخواست ،پردازش آن و تولید پاسخ است .

سرویس دهنده و مشتری از پروتکلی به نام http برای مقاصدشان استفاده می کنند.

این نرم افزار ها به زبان های خاصی برای تولید نیاز دارند.

7- نرم افزار های خانگی

مثل game ها و multimedia

8- نرم افزارهای هوش مصنوعی Artifitial Inteligenece

در این نوع نرم افزار قطعیت وجود ندارد ، بر اساس قواعدی که وجود دارد استنتاج هایی انجام می شود. بر اساس ورودی های مختلف خروجی های مختلفی داریم . از یک سری قوانین برای کاربردهای خاص استفاده میشود. فاقد قطعیت در خروجی هستند ، در اجراهای مختلف ممکن است خروجی متفاوت داشته باشند. مثل نرم افزارهای تکاملی مراحل مختلفی دارند. اجراهای مختلفی انجام می شود و خروجی هر مرحله ورودی مرحله قبل است .

ذهنیت های اشتباه

1- ذهنیت های اشتباه در حوزه مدیریت پروژه نرم افزاری

2- ذهنیت های اشتباه توسط توسعه دهندگان

3- ذهنیت های اشتباه توسط مشتریان

در فرایند فرایند توسعه نرم افزار استفاده از قوانین ،استاندارد ها و متدلوژی ها اجتناب ناپذیر است.

بحرانی در دهه 40 و 50 میلادی به نام بحران نرم افزار بوجود امد . منشا این بحران ها فقدان قوانین و استاندارد ها بود و پروژه ها با Project Failer روبرو می شدند.

مشکلاتی که وجود دارد :

1- هزینه های گزاف 2- تخطی از مهلت زمانی 3- عدم سازگاری یا مطابقت بین نیاز های مشتری (انتظارات و خواسته ها) و محصول تولید شده (عدم پیگیری نیاز مشتری باعث این مورد می شود)

این بحران باعث به وجود آمدن استاندارد ها و متدلوژی هایی شد که مشکلات و تبعات را کاهش دهد .

ذهنیت های اشتباه در حوزه مدیریت پروژه ی نرم افزاری

1- لیست کاملی از استانداردها ، قوانین و روال ها برای توسعه نرم افزار موجود است که تیم توسعه از ان بهره خواهد برد.

واقعیت موجود عدم استفاده اعضای تیم توسعه از این روال ها و استانداردها یا عدم اطلاع آنها از این اطلاعات است .

2- ذهنیت حوزه ی مدیریت از وجود آخرین و به روز ترین ابزار ارائه شده در حوزه ی نرم افزار (CASE) است .

واقعیت عدم استفاده تیم توسعه از این نرم افزارهاست.

منابع موجود در فرایند توسعه (منابع محدود هستند)

1- زمان 2- هزینه (بودجه) 3- staff

به دلایلی پروژه نرم افزاری با تاخیر روبرو می شود. در این صورت مدیریت پروژه نرم افزار چگونه این عقب افتادگی را جبران می کند ؟

یک مدیر موفق در زمانبندی خودش **فاکتور شناوری** را اضافه می کند یعنی اگر یکی از task ها به t واحد زمانی نیاز دارد به آن $\mathbf{t+\Delta t}$ زمان اختصاص می دهد. یعنی با Δt عقب افتادگی را پیش بینی می کند. اگر تاخیر بیشتر از این مقدار باشد به مراحل بعدی منتقل می شود و کل پروژه دچار تاخیر می شود.

روش دیگری برای جبران عقب افتادگی : **تزریق نیروی انسانی**

مفهوم نفر ساعت

10 نفر ساعت = کار با یک نفر 10 ساعته انجام می شود .

1*50 = 2*25 = 5*10 ?

این تساوی درست نیست چون افراد باید با هم هماهنگ شوند .

هر چه تعداد انسان ها بیشتر شود به همان نسبت فاصله این عبارت ها از هم بیشتر خواهد بود.

بنابراین تزریق نیروی انسانی نمی تواند این عقب افتادگی را جبران کند. در بحث گروهبندی سعی میشود اعضای گروه زیاد بزرگ انتخاب نشود.

3- در صورت عقب افتادن از برنامه زمان بندی می توان با تزریق نیروی انسانی به پیکره تیم به جبران عقب افتادگی پرداخت.

واقعیت این است که افراد و نیروی انسانی جدید نیاز به هزینه و زمانی برای اموزش توسط دیگر اعضای تیم توسعه دارند که این خود باعث تشدید عقب ماندگی خواهد شد.

برون سپاری out Source

نرم افزار inhouse نرم افزاری که در خود سازمان انجام شود.

پروژه ها یا out source هستند یا inhouse

وقتی یک مدیر نرم افزار با مشکل عقب ماندگی مواجه می شود آیا میتواند قسمتی را inhouse و قسمتی را out source انجام دهد؟ وقتی پروژه out source باشد دیگر روی ان مدیریت نداریم.

4- استفاده از برون سپاری یا بخشی از پروژه به یک پیمان کار می تواند عقب افتادگی را تا حدی جبران نماید.

بدلیل عدم کنترل مدیریت این بخش از پروژه احتمال به تاخییر افتادن آن و هزینه های مربوط به هماهنگی این بخش از این پروژه برون سپاری شده با دیگر بخش های پروژه ممکن است باعث تشدید عقب افتادگی شود.

ذهنیت های اشتباه توسعه دهندگان

برنامه نویسان بیشتر مایلند وارد مقوله کدنویسی شوند ولی به همان نسبت تاخیر در پروژه بوجود می آید.

مراحل یا گام های توسعه نرم افزار

1- شناخت نیازمندی ها 2- تحلیل 3- طراحی 4- تولید کد 5- تست 6- استقرار 7- نگهداری

نگرش مدلهای مختلف به این مراحل متفاوت است .

وقتی developer بخواهد زود وارد مقوله کد نویسی شود یعنی سه مرحله قبل را به خوبی انجام نداده (به خوبی به انها پرداخته نشده)

60 تا 80 درصد هزینه های تولید نرم افزار در سه مرحله اول انجام میشود.

1- هر چه زودتر وارد مقوله کدنویسی شویم محصول سریع تر نهایی خواهد شد.

واقعیت این است که هرچه زودتر وارد مقوله کد شویم به دلیل عدم اشراف به نیازمندی های مشتری بدلیل عدم تحلیل جامع روی خواسته های مشتری و عدم طراحی مناسب و کارآمد با تاخیر زمانی روبرو خواهیم شد .

واحد های تولیدی واحدی به نام QC (Quality Control) دارند . همچنین بخشی را در مقوله نرم افزار داریم که به آن QA (Quality Assurance) می گویند.

بحث QA باید به موازات تمام مراحل توسعه نرم افزار با آن پیش رود و نباید فقط به انتهای آن وارد شود . اما توسعه دهندگان مایلند در آخر وارد مرحله QA شوند.

2- بحث ارزیابی کیفیت نرم افزار و اطمینان از کیفیت به انتهای توسعه نرم افزار موکول شود.

واقعیت این است که ارزیابی کیفیت بایستی به موازات مراحل مختلف توسعه نرم افزار انجام شود.

خروجی فرآیند توسعه نرم افزار چیست ؟

1- نرم افزار 2- مستندات (مستندات تحلیل ، طراحی ، تولید کد ،تست و....) 3- داده

مستندات باید توسط اعضای مختلف تیم توسعه ایجاد و به همراه نرم افزار به مشتری تحویل داده شود که اگر مشتری خواست تغییراتی روی نرم افزار بدهد بتواند. مستندات مثل کامنت در برنامه نویسی هستند.

3- تنها بخش تحویل داده شده به مشتری نرم افزار است .

واقعیت این است که مستندات هم بخش دیگری از خروجی فرایند توسعه نرم افزار است که مشتری در صورت نیاز بتواند تغییرات لازم را اعمال کند.

مستندات بخش انکار ناپذیر فرایند توسعه نرم افزار است پس باید به مقوله مستند سازی بپردازیم . به افراد مستند ساز نیاز داریم. همه ی افراد از جمله برنامه نویس و ... باید خوشان مستند ساز باشند . اعضای تیم توسعه نیز مستندات را مورد استفاده قرار می دهند.

4- تیم توسعه فکر می کند مقوله مستند سازی یک کار غیر ضروری است که منجر به طولانی شدن زمان انجام پروژه می شود.

واقعیت این است که یکی از اهداف مهندسی نرم افزار مستند سازی است . مستند سازی باعث می شود تا بحث QA را داشته باشیم . مستند سازی لازمه ی رسیدن محصول با کیفیت مورد قبول است .

ذهنیت های اشتباه مشتریان

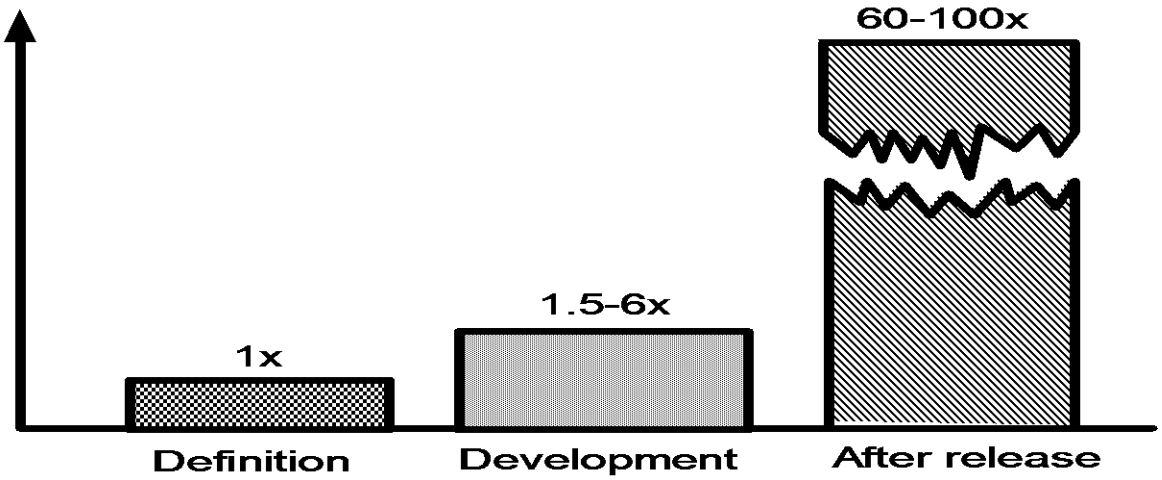
1- در ابتدای فرایند توسعه مشتری خیلی علاقه مند است با یک تعریف کلی از اهداف و نیازمندی های خود تیم توسعه را وارد مقوله توسعه نماید.

اما واقعیت این است که یکی از عوامل شکست پروژه نرم افزاری تعریف کلی و مبهم مساله است . و لذا باید تعریف دقیقی از domain صورت گیرد.

Domain حوزه : هر نرم افزار در یک حوزه مورد استفاده قرار می گیرد . در بدو توسعه باید شناختی از domain داشته باشیم که اگر نداشته باشیم بعضی از قابلیت ها در نرم افزار لحاظ نمی شود .

2- اعلام نیازمندی ها به صورت تدریجی . مشتری علاقه مند است که به صورت تدریجی نیازمندی ها و خواسته های خود را به تیم توسعه ابلاغ کند. با این ذهنیت اشتباه که هزینه های اعمال تغییر در نرم افزار ناچیز است اما واقعیت این است که تبعات و هزینه های اعمال تغییرات در نرم افزار شدیدا بستگی به این نکته دارد که شما در چه مرحله ای از توسعه نرم افزار قرار دارید.

اگر فرایند تولید نرم افزار را در سه حالت کلی تعیین کنیم :



محور عمودی بیانگر هزینه و محور افقی بیانگر زمان است .

فرایند

یک فرایند مجموعه ای از گام ها و اقدامات قابل پیش بینی است . (نقشه راه)

دررسیدن به نتیجه با کیفیت بالا در یک زمان مشخص و محدود کمک می کند.

یک فرایند نرم افزار چارچوبی است که از یک سری وظایف و اقدامات تشکیل شده است که به منظور رسیدن به یک نرم افزار با کیفیت بالا الزامی است .

چه کسی فرایند (نقشه راه) را تعریف می کند ؟

مهندسین نرم افزار و مدیران پروژه علاوه بر این ،این 2 گروه بر اساس ماهیت نرم افزار سازگاری های لازم را در سطح فر ایند انجام می دهند.

فرایند از چه مراحلی تشکیل شده است ؟

این مراحل بسته به ماهیت نرم افزار که در صدد تولید و توسعه آن هستید مجموعه اقدامات و گام ها در فرایند تعیین میشوند.

محصول یا فراورده هایی که در انتهای این راه و مسیر تولید شده چه چیزهایی هستند ؟

نرم افزار و مستندات و داده

نشانه های مربوط به درستی کار (صحت انجام کار) چیست؟

وجود ابزار های ارزیابی و همچنین مکانیزم هایی برای ارزیابی ، بحث کیفیت و تداوم و حضور طولانی یک نرم افزار در سطح یک سازمان.

مهندسی نرم افزار :

یک نگرش قابل اندازه گیری سیستماتیک و ساخت یافته ،سازمان دهی شده به منظور توسعه و راه اندازی و نگهداری یک نرم افزار.

مهندسی ، تحلیل ، طراحی ، ایجاد ،تایید یا اثبات و مدیریت موجودیت های تکنیکی .

اقداماتی که در بحث مهندسی نرم افزار وجود دارد به 3 مرحله دسته بندی می شود:

فارغ از اینکه چه سیستمی در حال توسعه است و اندازه پروژه چقدر است و میزان پیچیدگی سیستم به چه صورت است :

1- تعریف مساله Problem Definition 2- توسعه development 3- پشتیبانی support

تعریف مساله :

روی چه سیستمی قرار است انجام شود . یعنی چه سیستمی باید توسعه پیدا کند، چه اطلاعاتی باید مورد پردازش قرار بگیرد. منظور شناخت حوزه کاری است . چه روال ها و چه توابعی باید توسعه داده شود. چه کارایی مورد انتظار مشتری است . رفتار سیستم چگونه است ؟ چه رفتار هایی در سیستم قابل انتظار است . چه رابط هایی باید پایه گذاری شود . محدودیت های مربوط به طراحی سیستم چه چیزهایی هستند. شناخت نیازمندی های کلیدی .

در بحث تعریف مساله ما 3 کار اصلی انجام می دهیم :

1- مهندسی سیستم 2- برنامه ریزی پروژه (هزینه ،نیروی انسانی و...) planning 3- تحلیل و استخراج نیازمندی ها Requirement Capture

مهندسی سیستم

یکی از اقدامات و نقش هایی که به عهده تیم تعریف مساله است .

سیستم : مجموعه ای از اجزای به هم مرتبط که برای تحقق هدف یا اهداف از پیش تعیین شده با یکدیگر در تعامل هستند.

قبل از تعریف مساله نیاز به شناخت سیستمی که می خواهیم توسعه دهیم داریم . اول باید بدانیم سیستم از چه اجزایی تشکیل شده است .

1- محدوده Boundry 2- ورودی ها و خروجی ها input and output (باید بدانیم چه چیزی داخل و خارج شده است)

3- اجزا یا زیر سیستم ها subsystems 4- ماموریت یا فرایند های این سیستم goal 5- محدودیت های constrains 6- رابط ها Interface

هر سیستم برای خودش روابط عمومی داشته باشد . صرفا در سطح سیستم ها نیستند بلکه در سطح زیر سیستم ها و مولفه های نرم افزاری هم وجود دارد (هر مولفه نرم افزاری باید یک زیر سیستم داشته باشد.)

استخراج این 6 مورد و شناخت در باره سیستم قبل از تعریف مساله باید انجام شود .

توسعه

داده ها را چگونه سازماندهی کنیم ، چگونه توابعی را پیاده سازی کنیم ،چگونه طراحی قابل ترجمه به زبان برنامه سازی باشد ، روال های تست چگونه قابل انجام است.

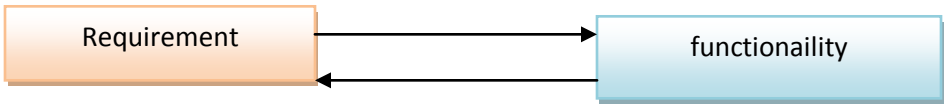
در بحث توسعه سه کار اصلی باید انجام شود :

1- طراحی نرم افزار design 2- پیاده سازی coding 3- تست test

توضیح در ارتباط یا تست :

فرق بین functionality (امکانات- قابلیت)و Requirement (نیازمندی – خواسته – انتظار)

Requirement توسط مشتری ایجاد میشود و functionality توسط تیم توسعه . ارتباط این دو مرحله در مرحله تست مورد بررسی قرار میگیرد. به ازای هر نیاز مندی باید قابلیت داشته باشیم و برعکس هر قابلیتی که داریم باید در نظر داشته باشیم که قابلیتی لحاظ نشده باشد که نیازمندی برای ان نباشد.



پشتیبانی

بعد از مرحله تست هیچ تضمینی وجود ندارد که اشکالی وجود نداشته باشد.

در بحث پشتیبانی اصلاحات (modification) ،سازگاری (adaptation)، ارتقا (Enhancement) ، جلوگیری (Preventation)و بحث مدل ها مطرح است . مدل هایی که در فرایند توسعه نرم افزار به آن نیاز داریم.

اصلاحات : افزودن قابلیت هایی در نرم افزار ممکن است موجب عدم سازگاری بین مولفه های نرم افزار شود . باید بکوشیم تا ناسازگاری را بین این مولفه ها از بین ببریم .

جلوگیری : جلوگیری از خطر سقوط نرم افزار یا منسوخ شدن آن (مثلا برای حجم بالایی از اطلاعات و درخواست ها نرم افزار باید بتواند پاسخگو باشد) .

ارتقا : مثل خدمات پس از فروش.

مدل های توسعه نرم افزار Software Process Models

هدف استفاده از مدل های مختلف این است که ریسک را کاهش دهیم و هزینه ها قابل قبول باشند.

استراتژی های توسعه ،روش ها یا متد ها بایستی توسط تیم توسعه نرم افزار به منظور حل یک مشکل خاص اتخاذ شود.

یک مدل فرایند برای مهندسی نرم افزار بر اساس ماهیت پروژه و نرم افزار به کار برده ، متدها و ابزارهایی که مورد استفاده قرار میگیرند انتخاب می شود .

معرفی مدل ها :

مدل ها در دو حوزه متفاوت دسته بندی می شوند :

- مدل های خطی
- مدل های تکاملی

مدل های متفاوت ممکن است نگرش های متفاوتی داشته باشند .

مدل ترتیبی خطی (آبشاری) Waterfall

نگرش این مدل به توسعه نرم افزار نگرش ترتیبی و خطی است .

این مدل یک رویکرد ترتیبی برای توسعه نرم افزار پیشنهاد می کند که از سطح سیستم شروع می شود و مراحل استخراج نیازمندی ها ، تحلیل ، طراحی ، کد نویسی یا پیاده سازی ، تست و پشتیبانی را پشت سر می گذارد. این مدل فعالیت های زیر را شامل می شود :

1- مدل سازی و مهندسی سیستم : شناسایی نیازمندی ها برای تمام سیستم و اختصاص بخشی از این نیازمندی ها به نرم افزار

2- تحلیل نیازمندی های نرم افزار : درک ماهیت برنامه ای که در حال توسعه است یک تحلیلگر باید درک درستی از محدوده سیستم (دامنه) ،روال های مورد نیاز ، رفتار و کارایی داشته باشد ، این نیازمندیها با تعامل تحلیل گر و مشتری مستند می شود و مورد بازبینی قرار می گیرد.

سه نگرش را در مقوله تحلیل سیستم داریم :

- بالا به پایین Top Down
- پایین به بالا Bottom Up
- شروع از وسط Middle Out

3- طراحی : یک فرایند چند مرحله ای است که روی 4 مشخصه کلیدی متمرکز می شود :

الف) ساختمان داده ها ب) معماری نرم افزار ج) رابطه ها د) جزئیات مربوط به هر کدام از روال هایی که وجود دارد .

یک طراح نیازمندی های مشتری را به شکلی ترجمه می کند که امکان ارزیابی آن قبل از ورود به مرحله پیاده سازی وجود داشته باشد.

4- تولید کد : مستندات طراحی بایستی به یک شکل قابل فهم برای ماشین ترجمه شود و در صورتی که طراحی با جزئیات کامل انجام شده باشد تولید کد به صورت خودکار قابل انجام است .

5- تست : این فعالیت بر روی منطق برنامه متمرکز است و برای اطمینان از اینکه تمامی عبارتها به درستی کار میکند مورد استفاده قرار می گیرد و این که در نگرش خارجی مجموعه ای از داده های ورودی خروجی مورد نظر را تولید می کند .

6- پشتیبانی : بعد از تحویل نرم افزار انجام می شود . تغییرات یک ضرورت است . ریشه های تغییر خطاهای نرم افزار ، ارتقا یا انتظارات جدید کاربر می تواند باشد .

ضعف های موجود در مدل آبشاری

1- پروژه های واقعی به ندرت یک جریان ترتیبی را دنبال می کنند. بنابر این هر تغییری که در طول روند تولید و توسعه نرم افزار از سوی مشتری اعلام شود منجر به سردرگمی اعضای تیم خواهد شد.

2- برای مشتری غیر ممکن است که بتواند تمامی نیاز هایش را یک جا و به صراحت اعلام کند. در حالی که مدل آبشاری این پیش فرض را دارد (لازمه این مدل)

3- مشتری باید صبور باشد . نسخه عملیاتی برنامه در انتهای این فرایند قابل دسترس خواهد بود.

4- وضعیت block شده (مسدود). این مدل منجر به ایجاد پدیده ای به نام block یا مسدود خواهد شد . برخی از اعضای تیم در انتظار انجام فعالیت توسط دیگر اعضای تیم و ارائه مستندات جهت اغاز فعالیتشان هستند.

5- از معایب دیگر این مدل بودن توازی و پیوستگی در انجام عملیات است چون کارها به صورت موازی انجام نمی شود بهره وری نمی تواند افزایش یابد.

6- ضعف دیگر عدم حضور مشتری در فرایند توسعه نرم افزار است . وقتی نیاز مشتری را بدانیم ریسک کاهش میابد چون سریع می توانیم ناسازگاری را تشخیص بدهیم و زود وارد عمل شویم . وقتی مشتری نیاز ها را تدریجی عنوان کند این مدل به درد نمی خورد.

مزیت های مدل آبشاری :

1- زمانی که با تغییرات جزئی در سطح نرم افزار رو به رو هستیم یا هدف تحقق نیازهای خاص و جزئی در نرم افزار است این مدل به درد می خورد.

2- به عنوان یک چارچوب جهت یادگیری و درک تکنیک های مختلف موجود در فرایند تولید و توسعه نرم افزار است.

3- به عنوان یک گذر در مدل حلزونی

4- به عنوان یک چارچوب برای یک مدل چرخه ای

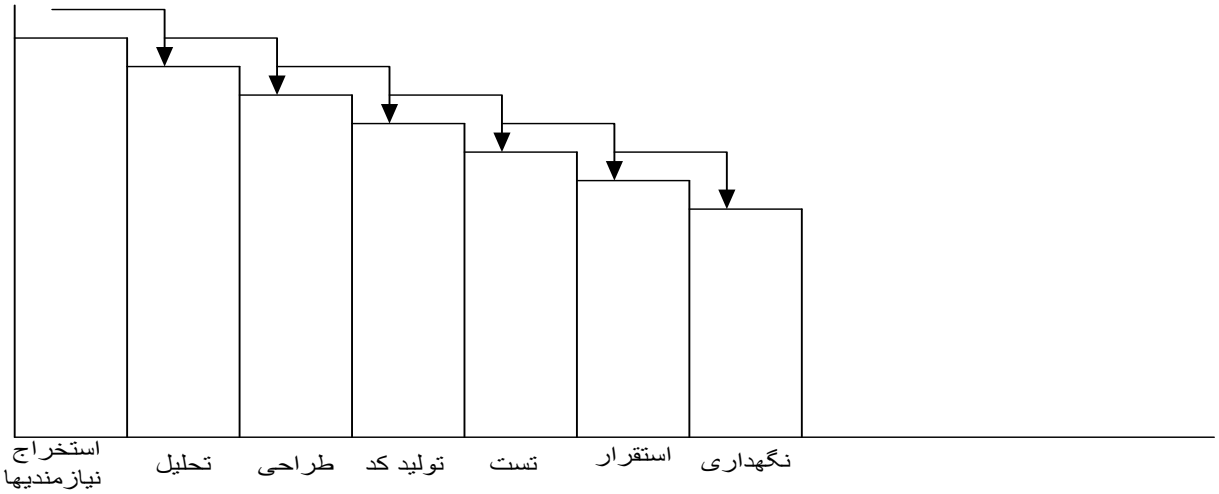
5- به عنوان یک تکنیک توسعه سریع برای پروژه های کوچک با یک تعداد محدود از توسعه دهندگان

این مدل از جنبه مطالعاتی مورد استفاده قرار میگیرد (مدل پایه محسوب می شود) مثلا به عنوان base band برای پروژه دیگری مورد استفاده قرار می گیرد .

همچنین زمانی که پیچیدگی پروژه کم است می تواند مورد استفاده قرار بگیرد.

7- استقرار : قرارگیری مدل نرم افزاری در یک محیط منطقی

مدل ابشاری نگرشی ره به صورت زیر به نرم افزار دارد :



عدم بازگشت به عقب مشخصه اصلی این مدل است .

مدل نمونه سازی ، مدل پیش نمونه سازی Prototyping Model

اغلب یک مشتری یک سری اهداف کلی را از نرم افزار مورد نظر در اختیار تیم توسعه قرار می دهد . بنابر این جزئیات مربوط به آن اهداف اعلام یا شناسایی نمی شود . این مدل با مرحله جمع اوری نیازمندی های مشتری اغاز می شود . توسعه دهندگان و متشری ملاقات هایی را با هم انجام می دهند تا اهداف کلی نرم افزار را تعیین کنند. بعد از مرحله فوق یک طراحی سریع quick Design بر اساس نیاز مندی های اعلام شده انجام خواهد شد . طراحی سریع عمدتا بر روی جنبه هایی از نرم افزار تمرکز دارد که قابل نمایش در نرم افزار است و در ادامه این طرح ارائه شده منجر به یک نمونه ازمایشی می شود . این نمونه ازمایشی توسط مشتری و کاربر نهایی مورد ارزیابی قرار می گیرد. در صورت نیاز یک سری پیشنهادات در صورت وجود در اختیار تیم توسعه قرار میگیرد تا نهایتا به توافق برسند . به صورت ایده آل این مدل به عنوان مکانیسمی برای شناسایی نیاز مندی های مشتری سودمند است و برای نیازمندیهای مشتری مورد استفاده قرار می گیرد .

هدف مدل کوچکتر:

با کمترین هزینه دید واقع بینانه ای از سیستم داشته باشیم و اگر قرار است تغییراتی را لحاظ کنیم زمانی که با مشتری به توافق رسیدیم تغییرات را لحاظ کنیم . بوسیله این مدل ریسک را کاهش می دهیم . ارتباطمان با مشتری را زیاد می کنیم تا زمانی که از ذهنیت حاصل او اطمینان پیدا کردیم وارد مقوله توسعه شویم.

دواستراتژی بعد از نهایی شدن پیش نمونه روبروی تیم توسعه است :

- کنار گذاشتن پیش نمونه و آغاز مجدد
- انجام اصلاحات دوم روی پیش نمونه جهت نیل به محصول واقعی

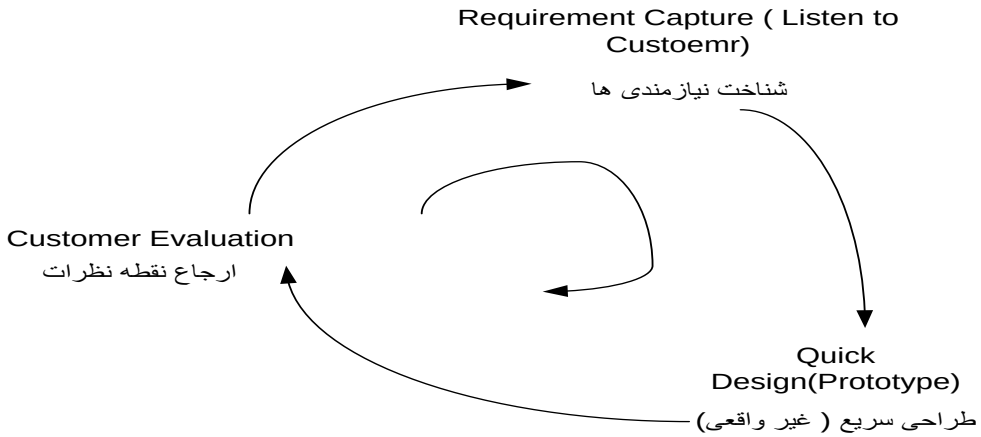
در بحث پیش نمونه سازی کیفیت زیاد لحاظ نشده (مثل ساختمان چند طبقه ای که همین طور روی هم قرار گرفته) پس رویکرد اول بهتر است.

در رویکرد دوم :

- مصالحه : تیم توسعه بدلیل عدم به کارگیری الگوریتم کارامد به ناچار به اراده محصول می پردازد.
- مشتری تلقی می کند که این مدل یا این پیش نمونه همان محصول واقعی است در صورتی که این طور نیست .

در بحث پیش نمونه سازی علاوه بر مشتریان تیم توسعه هم متمایل به استفاده از این نگرش دوم هستند.

مزیت این مدل حصول اطمینان و توافق بر سر خواسته ها و انتظارات مشتری است .



ضعف های مدل نمونه سازی :

1- ذهنیت مشتری

مشتری به اشتباه می پندارد که در نرم افزار نهایی چه ملاحظات و قابلیت هایی وجود دارد و غافل از این که پیش نمونه یک سیستم غیر واقعی است و ملاحظات مربوط به ارزیابی کیفیت نرم افزار در ادامه قابلیت نگهداری نرم افزار در نظر گرفته نشده است .

2- توسعه دهنده developer : به دلیل بهره گیری از ایده رسیدن پیش نمونه به ایده واقعی ناچار است یک مصالحه انجام دهد. آن مصالحه این است که الگوریتم کارآمد جایگزین الگوریتم ناکار آمد نخواهد شد و یا یک سیستم عامل نامناسب کنار نخواهد رفت و یک زبان برنامه سازی کنار گذاشته نمی شود.

مزیت های مدل نمونه سازی :

پیش نمونه سازی یک الگوری کارآمد در مهندسی نرم افزار است . نکته کلیدی در آن است که قواعد بازی در ابتدا تعریف می شود. این به این معنی است که مشتری و توسعه دهنده هر دو بر روی پیش نمونه اتفاق نظر داشته باشند . پیش نمونه ای که به عنوان مکانیزمی برای شناسایی نیازمندی های مشتری مطرح میشود.

مدل توسعه سریع کاربرد RAD (Rapid Application Development)

نگرش توسعه نرم افزار به طور سریع با بهره گیری از ارائه و به کارگیری مولفه های از پیش ساخته با هدف اسمبل کردن مولفه های نرم افزاری است .

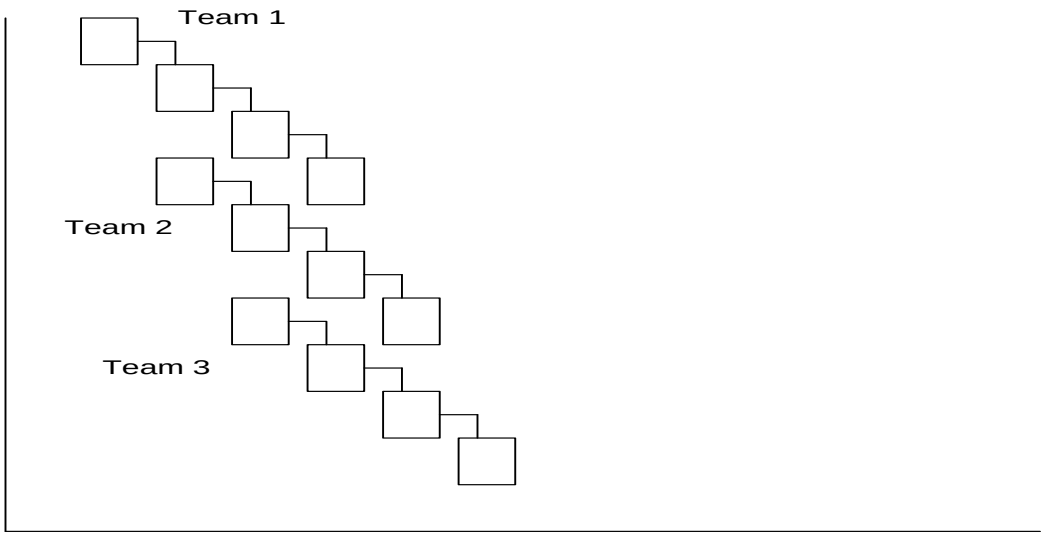
تعریف دیگر : توسعه برنامه های کاربردی مورد نیاز مشتری از روی مولفه های از پیش ساخته جهت کاهش زمان و هزینه توسعه سیستم نرم افزاری. ما در مدل RAD نگرش بر تولید یک سیستم نرم افزاری در یک مدت زمان محدود و کوتاه هستیم (60 تا 90 روز)

این مدل یک نسخه خاص از مدل ترتیبی خطی است که در آن با استفاده از ایده استفاده از مولفه های از پیش ساخته توسعه نرم افزار انجام می گیرد.

موفقیت این مدل (تولید یک سیستم نرم افزاری قابل قبول) مستلزم این است که :

نیازمندی ها بدرستی شناسایی شده باشد و دامنه پروژه محدود باشد و پروژه متوسط به پایین باشد.

طراحی هر کدام از سیستم ها همزمان انجام می شود یعنی در زمانی که یک سیستم در حال تحلیل است سیستم دیگر عملیات پیاده سازی یا ... را انجام می دهد .



این مدل از مراحل یا فازهای زیر تشکیل شده است :

1- مدل سازی سیستم تجاری Business Modeling 2- مدل سازی داده Data Modeling

3- مدلسازی فرایند Process Modeling 4- یکپارچه سازی برنامه کاربردی Integration

5- تست و تحویل به مشتری Test(تست می کنیم که آیا زیر سیستم ها درست کار می کنند؟)

مدلسازی سیستم تجاری :

جریان یا گردش داده ها و اطلاعات بین روال های مختلف سیستم جاری بایستی شناسایی و مدل گردد. و معمولا با سوال های زیر بدست می اید :

- چه اطلاعاتی در فرایند های مختلف سیستم وجود دارد ؟
- چه داده هایی را تولید می کنیم ؟
- چه کسی داده ها را تولید می کند ؟
- این داده ها و اطلاعات به کجا می روند ؟
- چه کسی داده ها را پردازش می کند ؟

مدل سازی داده :

گردش اطلاعات که به عنوان بخشی از فاز مدل سازی سیستم جاری تعریف می شود به یک سری از اشیا داده ای خلاصه می شود که در ادامه مورد استفاده قرار می گیرند. مشخصات هر یک از اشیا شناسایی و تعریف می شود و ارتباط های بین این اشیا شناسایی می شوند.

مدل سازی فرایند ها :

اشیا داده ای که در مدلسازی داده ها تعریف شده است به شکلی خواهند شد که به منظور تحقق یک جریان اطلاعات مورد استفاده قرار می گیرند. در واقع در مدلسازی فرایند شناسایی فرایند به منظور افزودن ، اصلاح ، حذف یا بازیابی یک شی داده ای مورد استفاده قرار می گیرد.

یکپارچه سازی برنامه کاربردی :

به جای ایجاد نرم افزار به روش متعارف RAD بر روی استفاده از قطعات از پیش ساخته متمرکز می شود و یا یک سری قطعات قابل استفاده ایجاد می کند . در تمامی حالات ابزارهایی وجود دارد که به منظور ایجاد و بازسازی نرم افزار مورد استفاده قرار می گیرد(اتخاذ تصمیمات به منظور یکپارچه سازی زیر سیستم هایی که توسط کارگروه های مختلف توسعه یافته است)

تست و تحویل به مشتری:

هزینه کلی مرحله تست نسبت به مراحل دیگر کمتر خواهد بود و کاهش پیدا می کند زیرا مولفه های پیش ساخته به کار گرفته شده قبلا مورد تست قرار گرفته اند. هر چند که مولفه های ایجاد شده نیاز به تست دارند و چون هر یک از SubSystem ها روالهای تست را در مراحل قبل پشت سر گذاشته اند در این مرحله عمدتا تست یکپارچه سازی انجام می شود.

ضعف های مدل RAD

1- برای پروژه های بزرگ و یا Scalable (مقیاس پذیر) نیاز به منابع انسانی کافی برای ایجاد تیم های کاری مورد نیاز می باشد.

2- اسیب پذیری این سیستم در برابر فاکتور زمان. این مدل این پیش فرض را دارد که توسعه دهندگان و مشتریان به سیستم نهایی وعملیاتی در یک فریم زمانی مشخص دست پیدا خواهند کردو اگر به هر دلیلی تیم یا تیم هایی از پروژه با تاخیری مواجه شود با شکست مواجه خواهند شد.

3- اگر سیستم نتواند به درستی تجزیه شود (تکه تکه شود) ایجاد مولفه ها یا به کارگیری مولفه هایی که مورد نیاز برای توسعه application است با مشکل مواجه خواهد شد .

4- هرچه ریسک فنی پروژه بالا باشد خطر شکست پروژه افزایش میابد.

هرچه تعامل بین کار گروه ها بیشتر باشد باعث می شود ریسک بیشتر شود چون سیستم پیچیده تر می شودو خطر شکست پروژه افزایش میابد.

مدل های تکاملی فرایند Evalutionary Models

حقیقت : نمی توان یک محصول نرم افزاری را در یک Pass (گذر) توسعه داد . (هر بار scan کد برنامه یک گذر است)

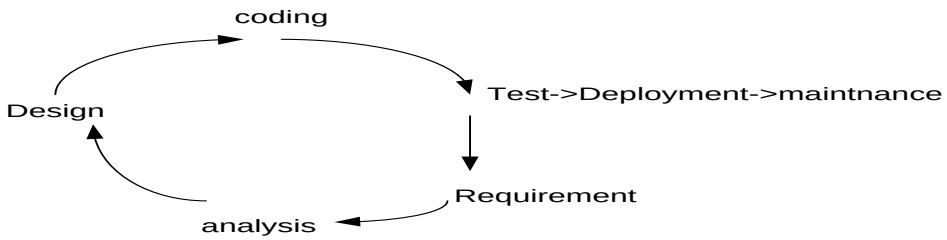
تک گذره : سرعت بالا

چند گذره : کد بهینه تولید میشود.

هدف : ارائه مدل هایی برای توسعه نرم افزار است که به طور واقع بینانه به مقوله توسعه نرم افزار نگاه داشته باشند.

مدل حلزونی :

این مدل مثل مدل ابشاری است با این تفاوت که این مراحل را چندین دور می توان انجام داد.



ایده : تکرار بیش از یکبار مراحل هفت گانه مدل آبشاری برای رسیدن به محصول نرم افزاری

ضعف های مدل حلزونی :

فاکتور زمان و احتمال تاخیر در پروژه نرم افزاری

امکان بازگشت به گذشته تا زمان تکمیل یک چرخه وجود ندارد.

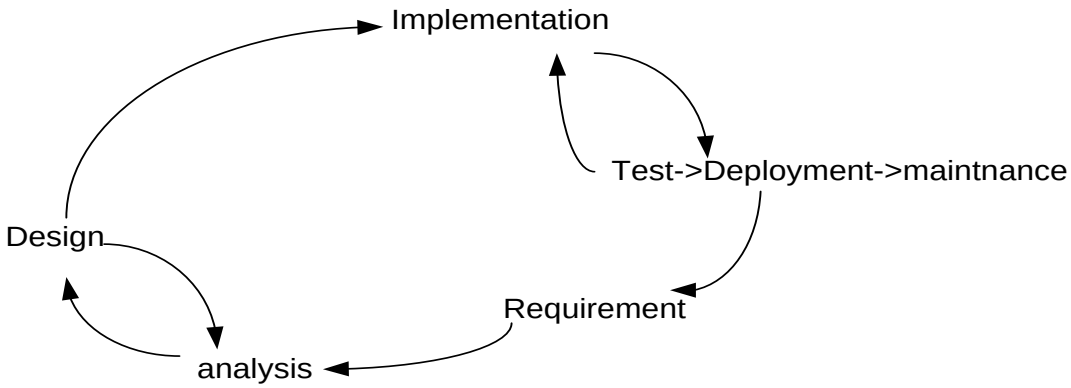
غیبت کاربران نهایی و مشتریان در فرایند توسعه نرم افزار

مستندات تولید شده در یک مرحله (گذر) در گذر بعدی منسوخ می شود و به طور کامل کنار گذاشته می شود.

این مدل زمانی به انتها می رسد که رضایت مشتری جلب شود .

کاربر یا کاربران در انتهای هر سیکل از روند پیشرفت پروژه مطلع می شوند. مدل حلزونی هنوز هم کامل نیست . عملا مدل ابشاری در این مدل به کار گرفته می شود با این تفاوت که هر کاری سه بار یا بیشتر انجام می شود.

مدل چرخه ای Iterative model



همانند مدل حلزونی تاکید این مدل در ایجاد و توسعه سیستم در چندین مرحله است . این مدل این امکان را فراهم می کند که بتوانیم در یک گذر به مراحل قبل بازگردیم. (در صورت وجود اشتباه فنی)

این مدل در موارد زیر می تواند موثر باشد :

1- مستندات تولید شده در هر چرخه یا سیکل مانند مدل حلزونی دور ریخته نمی شود و اصلاحات لازم روی آن مستندات انجام میشود.

مستندات تولید نمی شوند تا زمانی که اصلاحات لازم در مستندات مرحله قبلی اعمال شود.

2- ابزارهای توسعه نرم افزاری وجود دارند که این مدل را پشتیبانی می کنند و این امکان را فراهم می آورند که درصد مطابقت (سازگاری) یک محصول یا یک مستند با مشخصه ی آن مستند یا نیازمندی تعیین شود .

3- فازهای کلاسیک موجود در این مدل تعیین می کند که چه فعالیت هایی باید انجام شود و در چه سیستمی باید حرکت کنیم.

4- این مدل این امکان را فراهم می آورد که در کمترین زمان ممکن به رفع مشکل بوجود امده بپردازیم .

کاربران نهایی و مشتریان در انتهای فاز توسعه در جریان خروجی قرار میگیرند و در طی توسعه غایب هستند.

مدل افزایشی Incremental Model

نگرش : تقسیم مساله به مسائل کوچکتر (تقسیم بندی نیاز مندی ها و اولویت بندی انها)

تجزیه سیستم به زیر سیستم ها و توسعه نرم افزار به صورت تدریجی انجام میشود. مبنای انتخاب (اولویت گذاری) زیر سیستم ها اهمیت زیر سیستم ها با انتخاب مشتری است .

مثلا دانشگاه یک سیستم و آموزش یک زیر سیستم آن است .

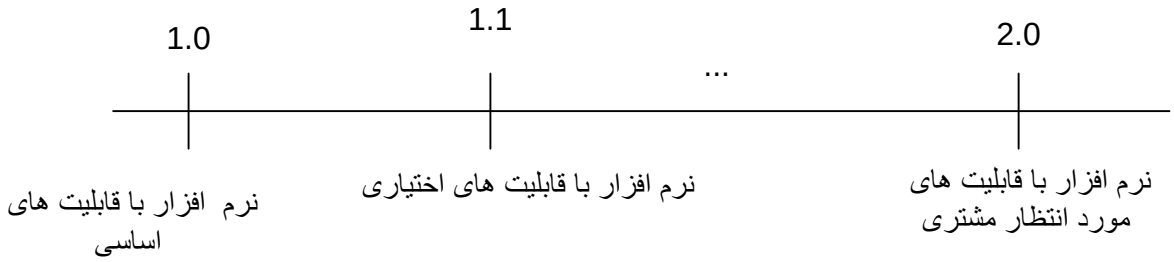
تقسیم بندی نیاز مندی ها:

1- نیازمندی با بالاترین اولویت (حیاتی) 2- نیازمندی های خیلی مهم 3- نیازمندی های مهم 4- نیازمندی های متوسط

5- نیازمندی های اختیاری

Versioning : نسخه های مختلفی از نرم افزار با قابلیت های افزوده ارائه و در اختیار کاربران نهایی قرار داده می شود.

Subsystem کلیدی : بیشترین وابستگی subsystem ها به آن وجود دارد .



نگرش مدل تکاملی نگرش واقع بینانه ای است .

ادغام یا ترکیبی از مدل های مختلف Rup (Rational Unified Process)

تلفیقی از مدل های ابشاری ،افزایشی ، حلزونی و چرخشی و مبتنی بر نگرش شی گراست .

متدلوژی Methodology

تشریحی از گام ها (task activity : محصول و مستند (بیانی artifact – نهایی)) و اقدامات لازم که تیم توسعه برای رسیدن به یک محصول با کیفیت نرم افزاری باید دنبال کند.

در واقع متدلوژی یک روش سیستماتیک برای انجام کارهاست.

تعاریف دیگر متدلوژی :

گام ها و اقدامات و فعالیت هایی است که تیم توسعه با توجه به نقش ها انجام می دهند تا به محصولات (لزوما محصول نهایی نیست ، ممکن است محصول میانی باشد) و مستندات (راهنمای کاربران ، مستندات تست ،مستندات طراحی ، تحلیل ، نیازمندی ها) برسند.

مثلا اگر در آرایشگاه مدل مو را انتخاب کنی ، مراحل رسیدن به این مدل مو متدلوژی است .

یک متدلوژی به موارد زیر می پردازد :

1- فلسفه موجود در هر یک از فاز ها 2- گردش یا جریان کاری که در داخل فعالیت باید انجام شود و فعالیت های خاصی که باید در هر فاز انجام شود . 3- خروجی هایی که باید تولید شود 4- وابستگی بین خروجی ها 5- علائمی که برای تولید هر کدام از خروجی ها مورد استفاده قرار می گیرد. به عنوان مثال در متدلوژی RUP از زبان شی گرای UML استفاده می شود . 6- ضرورت ایجاد مدل اشیا پویا

ویژگی های متدلوژی (یک متدلوژی خوب به موضوعات زیر می پردازد):

1- زمانبندی (Scheduling) یک فعالیت در چه زمانی و با چه مدت زمان انجام پذیر است 2- مدیریت منابع Resource Management تخصیص منابع Resource Allocation 3- تشریحی از فعالیت ها 4- برنامه ریزی Planning 5-خروجی ها Output

6- آموزش End User Training

Developer Training : بخشی از مستندات است که به آن User Manual می گویند . راهنمای نرم افزار مثل فایل های pdf ی است که کنار نرم افزار قرار می گیرد.

7- گردش های کاری Work Flow یک سری work Flow ها روی متدلوژی وجود دارد .

8- نقش ها role ها باید بر اساس نقش ها در سازمان شرح وظایف داشته باشیم.

شناوری : برای کاهش احتمال تغییر در نرم افزار شناوری را برای فعالیت در نظر می گیریم تا جلوی انتشار تاخیر در مراحل بعد را بگیریم. بعضی فعالیت ها ماهیتا شناوری ندارند پس اگر با تاخیر روبرو شدند کل پروژه با تاخیر روبرو می شود.

مسیر بحرانی Critical Path

مسیری از مجموعه ای از فعالیت ها که شناوری این فعالت ها صفر است که هر کدام از این فعالیت ها با تاخیر روبرو شوند کل پروژه نرم افزاری با تاخیر روبرو خواهد شد .

مزیت استفاده از متدلوژی :

1- امکان مستند سازی Documention

2- کاهش مدت زمان تاخیر

3- افزایش شانس تحویل سیستم با هزینه تخمینی در زمان تخمینی

4- قابلیت تکرار پذیری

مدل سازی نیازمندی های سیستم :

تصمیم گیری راجع به اینکه چه قابلیت هایی در نرم افزار پیشنهادی باید لحاظ شود .

سوالاتی نظیر چگونه می توان رکورد دانشجویی را در زمان ثبت نام تغییر داد در این مدل مطرح میشود.

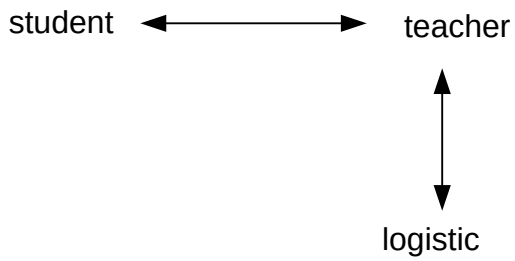
برای درک اینکه با چه موجودیت هایی در ارتباط هستیم بایستی نسبت به موضوعات زیر شناخت داشته باشیم .

1- موجودیت ها یا پدیده های مرتبط Relevant Entities

2- مشخصه های هرکدام از موجودیت ها Their Properties

مثلا برای دانشجو (نام و نام خانوادگی – شماره دانشجویی – تلفن – ادرس و ...)

3- ارتباط بین موجودیت ها Inter-Relationship



دانشجو که شامل موجودیت ها و رفتار است تشکیل یک کلاس می دهد.

در مرحله تحلیل سوالاتی نظیر سوالات زیر مطرح می شود :

- چه کالاهایی در این فروشگاه عرضه می شوند ؟
- این کالا ها از کجا تامین میشوند ؟
- قیمت کالا ها چند است ؟

تفاوت مرحله تحلیل و طراحی چیست ؟

طراح تصمیم گیرنده راجع به انتخاب تکنولوژی جهت توسعه تیم نرم افزاری است . زبان برنامه سازی ، محیط توسعه ،سیستم مدیریت بانک اطلاعاتی و ... توسط تیم طراحی اعلام می شود .

در مرحله طراحی هدف حل مساله می باشد یا ارائه راهکار طراحی سیستم ،سیستم به زیر سیستم های منطقی تقسیم می کند . این زیر سیستم ها همان فرایند ها هستند . علاوه بر این طراحی سیستم ،سیستم را به زیر سیستم های فیزیکی هم تقسیم می کند. طراح تصمیم می گیرد چه ماشین هایی با یکدیگر در ارتباط باشند . (سیستم های کامپیوتری) . تکنولوژی درست و به جا را برای اهداف خاص انتخاب می کند. به عنوان مثال اینکه برای توسعه application از چه زبانی استفاده شود .

در طراحی subystem ها تصمیم گیری انجام می شود که چگونه زیر سیستم های منطقی به قطعات کد کارامد و موثر و شدنی تقسیم شوند(تقسیم پذیر باشد) به عنوان مثال در یک سیستم انبار آیتم ها یا اقلام چگونه داخل یک Hash Table قرار گیرند.

مشخصه : این اصطلاح توسط توسعه دهندگان مختلف تعبیر های مختلفی دارد . به عنوان مثال خروجی فاز جمع آوری نیازمندی ها مشخصه سیستمی است که در صدد توسعه آن هستیم یا خروجی مرحله تحلیل مشخصه سیستمی است که تعیین می کند با چه پدیده هایی مواجه هستیم . در موارد زیر مورد استفاده قرار میگیرد:

- به عنوان مبنایی برای طراحی روال های تست

- به منظور بررسی یا نشان دادن اینکه نرم افزار صحیح توسعه پیدا کرده به خصوص برای سیستم های نرم افزاری بحرانی حساس Life Critical
- برای مستند سازی مولفه های نرم افزار که قرار است توسط دیگر تیم ها (پیمان کار) توسعه پیدا کند .
- برای توصیف اینکه قطعه کد تولید شده چقدر امکان استفاده در بخش های دیگر را دارد .

پیاده سازی : نوشتن قطعات کد برای شکل دهی subsystem ها مورد استفاده قرار میگیرد. پیاده سازی باید به گونه ای انجام شود که با مشخصه سیستم مطابقت و هم خوانی داشته باشد . رابطه های مولفه های نرم افزاری بایستی به صورت صحیح طراحی و مستند سازی شود. جزئیات پیاده سازی بخش های داخلی مولفه های نرم افزاری به توسعه دهنده واگذار شود. بخش های بیرونی بایستی بر اساس مشخصه مولفه توسعه یابد.

تست: تست مطابقت Conformance Testing با هدف مطابقت نیازهای مطرح شده از سوی مشتری و مشخصه آن با پیاده سازی صورت گرفته . توسعه دهنده می تواند در مقوله تست ایفای نقش نماید (با انجام تست در سطح مولفه ها)

استقرار :

- ایجاد بستر ارتباطی (شبکه) برای سیستم های نرم افزاری
- آموزش کاربران ،شناخت طیف مختلف کاربران نهایی سیستم (کاربران معمولی – راهبران – مدیران – مدیران فنی)
- راه اندازی سرویس دهنده های مختلف (اوراکل – دایرکتوری فعال – پست الکترونیک و ...)
- پیکر بندی سیستم نرم افزاری در سطح شبکه
- پیکر بندی تجهیزات شبکه (سوئیچ ها – مسیر یاب هاو ...)

پشتیبانی و نگهداری :

1- ارتقا 2- اصلاح (رفع عیب)3- اجتناب 4- سازگاری

ایده تلفیق مدل های مختلف در فرایند تولید و توسعه نرم افزار :

عکس العمل سریع تغییرات Responsive to Change (Extreme Programming)

XP (Xtreme Programming)

یک متدلوژی (نگرش) برای توسعه نرم افزار مبتنی بر مدل توسعه سریع کاربرد (RAD)می باشد . از دو استراتژی برخوردار است :

1- برنامه نویسی دو به دو یا جفتی Pair Programming

هدف بهره گیری از ایده های دیگری به منظور کاهش احتمال بروز خطا در توسعه نرم افزار است . پای هر سیستم دو نفر می نشینند و کار توسعه را انجام می دهند . این کار باعث همفکری میشود و قبل از مقوله تولید کد ضعف های دیگری را پوشش می دهند.

2- توسعه مبتنی بر تست Test Driven Development

هر توسعه باید مبتنی بر تست انجام شود تا از انتشار خطا به مراحل دیگر جلوگیری شود .

مولفه های قابل استفاده مجدد باید در مراحل ابتدایی توسعه نرم افزار تست شوند.

یکی از فاکتور های توسعه نرم افزار استفاده از ابزار case است.

انتظارات از یک ابزار case

1- قابلیت پیگیری Traceability : این امکان باید فراهم شود که ارتباط بین مستندات (خروجی) یک مرحله (فاز) و مستندات مرحله دیگر برقرار شود .

2- سوابق تغییرات Change History : امکان پیگیری و مشاهده تغییرات صورت گرفته در مستندات مختلف به منظور بازگشت به ویرایش های قبلی ، در واقع برخورداری از کلکسیونی از تغییرات اعمال شده در مستندات که در صورت نیاز امکان بازگشت به ویرایش قدیمی تر وجود داشته باشد .

3- کنترل دسترسی چند کاربره Multi-User Accsess : اختصاص سطوح دسترسی به افراد شرکت کننده در روند توسعه و تولید نرم افزار بر اساس نقش هر یک از آن ها. جلوگیری از دسترسی همزمان کاربران سیستم به مستندات به منظور اعمال تغییرت در مستندات .

4- افزونگی کاهش یافته Reduced Redundancy : در صورت ایجاد تغییر در برخی مستندات امکان مشاهده این تغیرات برای دیگر کاربران فراهم باشد . اطمینان از اینکه برای اعمال تغییر نیاز به ثبت ان تغییر در بخش های مختلف نمی باشد . هدف کاهش نیاز اعمال تغییرات در بیش از یک مستند .

5- بررسی سازگاری و جامعیت Consistency Checking : در صورت ایجاد تغییر در یک مستند تغییرات لازم در مستندات مرتبط اعمال گردد. تا از بروز ناسازگاری در مستندات اجتناب گردد. یعنی بین مستندات مختلف ایجاد شده توسط ابزار case بایستی سازگاری وجود داشته باشد و در صورت بروز ناسازگاری تدبیری جهت رفع آن ها اتخاذ گردد.

6- عملیات تحت شبکه Network Operation : امکان دسترسی همزمان کاربران (افراد تیم توسعه) نرم افزار ز طریق سیستم های کامپیوتری تحت شبکه.

7- تست و ارزیابی محصولات : ابزار case باید این قابلیت را داشته باشد که تست های اولیه را اعمال کند تا در صورت وجود تناقض از نهایی شدن تغییرات ممانعت شود .

زبان مدلسازی UML (Unified Modeling Language)

زبانی است که جهت مدل سازی سیستم ها مورد استفاده قرار می گیرد .

هدف از مدلسازی ارائه یک تصویر واحد از واقعیت عنوان شده است .

هر چه بتوانیم به تصویر واقعی نزدیک تر شویم موفقیت در پروژه عملی تر است .

دو نقش در مدلسازی :

- مدلساز Modeler
- مخاطب Audience : مشتری و کار فرما – دیگر اعضای تیم توسعه

در مقوله مدلسازی باید از یک زبان مشترک یا قرار داد مشترک استفاده کنیم . یک زبان و قرار داد مشترک بین مدلساز و مخاطب (تیم توسعه یا مشتری) وجود دارد . این زبان باید دقیق باشد و به قالب ذهنی مخاطب نزدیک باشد . باید ابهام نداشته باشد . برای جلوگیری از ابهام و نزدیکی زبان مدلسازی به قالب ذهنی (مدل ذهنی) مخاطب از علائم و سمبل های گرافیکی استفاده می شود .

این زبان از مفاهیم کلاس و شی گرایی به طور کامل پشتیبانی می کند . زبان uml با استفاده از نماد های گرافیکی به مدلسازی می پردازد.

کلیشه Sterotype :

علامت گرافیکی تعریف شده در uml است .

رشنال رز Rational Rose :

یکی از ابزار های case که مبتنی بر زبان مدلسازی uml بوده و فرایند RUP را پشتیبانی می کند .

نمودار های موجود در UML

1- نمودار مورد استفاده Use case Diagram

دو نوع میباشد :

- Business Use Case Diagram
- System Use case Diagram

2- نمودار فعالیت Activity Diagram

3- نمودار کلاس Class Diagram

4- نمودار توالی Sequence Diagram

5- نمودار همکاری Collaboration Diagram

6- نمودار حالت / وضعیت Statechart Diagram

7- نمودار مولفه Component Diagram

8- نمودار استقرار Deployment Diagram

9- نمودار مدل داده Data Diagram

تعاریف

مورد استفاده (کاربرد)Use Case :

System Use Case

سرویس یا خدماتی که از سوی سیستم در اختیار مراجعه کنندگان قرار می گیرد در قالب مورد استفاده مطرح می گردد.

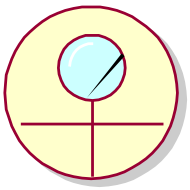
به هر نوع خدمت یا سرویس ارائه شده در سیستم پیشنهادی توسط سیستم اطلاعاتی گفته می شود . این سرویس می تواند مورد استفاده موجودیت های خارجی و موجئدیت های درون سازمانی قرار گیرد.

کنشگر Actor :

Actor

یک موجودیت خارجی که باعث راه اندازی یک مورد استفاده شده و یا از اطلاعات ارائه شده توسط یک usecase بهره می برد.

کارگر Worker :



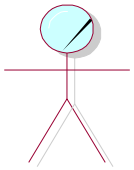
Worker

هر موجودیتی که در سطح سیستم اقدام به ارائه سرویس مربوط به کنشگر می کند کارگر یا worker نامیده می شود . در واقع یک موجودیت درون سازمانی که برای تحقق یک خدمت به کنشگر وارد عمل می شود.

کنشگر می تواند یک فرد ،یک زیر سیستم در سطح سیستم و یا یک سیستم خارجی باشد .

Business Actor :

یک موجودیت خارجی که در سیستم جاری راه انداز یک سرویس است .

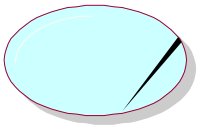


Business Actor

Business Use Case :

یک خدمت عرضه شده در سیستم جاری

سرویس ارائه شده در سیستم جاری به موجودیت های ارائه شده در سیستم جاری ، موجودیت های درون سازمانی یا خارجی که ممکن است در سیستم پیشنهادی منسوخ شده یا به روش نوین ارائه گردد.



Business Use case

نیاز مندی ها :

- **Business Modeling :** خدمات ارائه شده سیستم جاری – هدف ایجاد تصویر گویا از سیستم جاری ، زیر سیستم ها ، روالها، تعاملات ع وابستگی ها و ... است .
- **System Requirement Modeling :** خدمات مورد انتظار در سیستم پیشنهادی (باید لحاظ شود) - انتظارات و خواسته های مشتری (اسپانسر) از سیستم نرم افزاری مورد تقاضا در این مدل مطرح می شود .

Business use case diagram :

این دیاگرام به منظور نمایش قابلیت های ارائه شده توسط سازمان مورد استفاده قرار می گیرد. سوالاتی نظیر این که سیستم جاری چه وظایفی بر عهده دارد توسط این نمودار پاسخ داده می شود. این نمودار به عنوان یک اساس جهت ایجاد use case ها ایفای نقش می کند.

Business Worker :

مثل ارتباط Business Actor و Actor

Use case Diagram :

دیاگرام های use case تعاملات بین actor ها و use case ها را نمایش می دهد.

Use case ها قابلیت های ارائه شده توسط سیستم را شامل می شود. Actor ها افراد یا سیستم هایی هستند که اطلاعات را در اختیار سیستم قرار می دهند یا اطلاعاتی را از سیستم دریافت می کنند. نمودار های use case نشان می دهند که چه actor ی یک use case را راه اندازی می کند یا چه actor ی اطلاعاتی را از یک use case دریافت می کند. Use case diagram در واقع نیازمندی های سیستم پیشنهادی را نشان می دهد در حالی که Business use case به این موضوع توجهی نداشته و صرفا روالهای جاری را مدل می کند.

نکته :

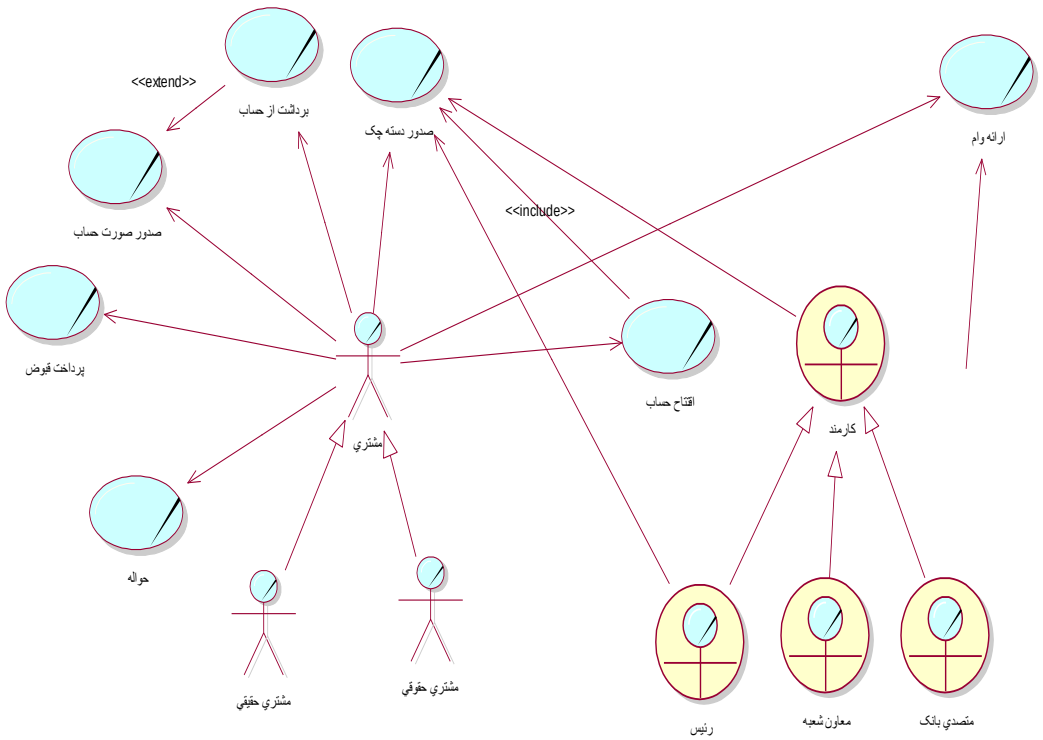
یک رابطه یک به یک بین Business Use case و Use case وجود ندارد . یک Business Use case تنها گاها نیاز به 30 تا Use case برای پیاده سازی دارد.

نمودار فعالیت Activity :

این نمودار برای نمایش جریان کار در سیستم مورد استفاده قرار می گیرد. این نمودار در مدلسازی سیستم جاری مورد استفاده قرار می گیرد تا گردش کاری در سیستم جاری را نشان دهد.

مثال:

بانک - هدف : ترسیم و ایجاد نمودار مورد استفاده برای سیستم بانکی



ارتباط بین مورد های استفاده :

1- مشارکت اجباری (include) 2- مشارکت اختیاری (extend)

این نمودار در مرحله جمع آوری نیازمندی ها جهت نمایش جریان رخداد ها در خلال یک use case مورد استفاده قرار میگیرد. این نمودار مشخص می کند که کجا گردش کاری اغاز و کجا پایان میابد و چه فعالیت هایی باید در این بین انجام شود و به چه ترتیبی.

نکته :

ارتباط ارث بری بین موجودیت ها برقرار است . ارتباط ارث بری بین مورد های استفاده نباید و نمیبایستی برقرار باشد.در ارتباط اجباری یک مورد استفاده توام با اجرای مورد استفاده دیگر است . اما اجرای یک مورد استفاده می تواند توام با اجرای مورد استفاده دیگری باشد .

دیاگرام فعالیت لزوما برای هر گردش کاری ایجاد نمی شود و صرفا برای گردش های کاری پیچیده و بزرگ مورد استفاده قرار می گیرد.

نیازمندی ها و مدیریت نیازمندی ها Requirement and management

نیازمندی Requirement :

1- یک قابلیت و توانمندی طرح شده توسط کاربر نهایی یا مشتری است که جهت حل یک مساله در سطح نرم افزار باید پیش بینی گردد.

2- یک قابلیت (Functionality) که باید توسط نرم افزار ارائه شده تا یک استاندارد , یک مشخصه , و یا بندی از یک قرارداد را پوشش دهد.

3- یک محدودیت دیکته شده توسط حامی پروژه (نیازمندی می تواند یک محدودیت باشد)

سیستم بایستی از درج نمرات دانشجویان که خارج از محدوده صفر تا 20 می باشد اجتناب به عمل آورد. سیستم بایستی طبق استاندارد ISO22001 مبادرت به تولید مستندات یا گزارش های مورد نیاز کاربر نماید.

بین نیازمندی و قابلیت باید سازگاری وجود داشته باشد .

منابع طرح نیازمندی :

1- کاربران نهایی End User 2- مشتری یا کارفرما Customer 3- حامیان پروژه Sponser 4- تیم توسعه (نیازمندی ها را بر اساس شناخت یا تشخیص اعلام و لحاظ می کنند)

انواع نیازمندی :

- نیازمندی های کارکردی یا عملیاتی Functional Requirement
- نیازمندی های غیر کارکردی یا غیر عملیاتی یا تکمیلی nonFunctional Requirement

فرق این دو نیاز مندی :

نیاز مندی اول به طور مستقیم مورد استفاده کاربر نهایی قرار می گیرد اما نیازمندی نوع دوم به طور مستقیم و غیر مستقیم جهت ارائه قابلیت های کارکردی و پشتیبانی از آن ارائه و پیاده سازی می شود .

برای مثال نرم افزار کاربردی باید بتواند در گزارش های مختلف گزارشی تهیه نماید که در آن لیست دانشجویانی که سه ترم متوالی مشروط شده اند تهیه گردد (یک Functionality در سیستم)

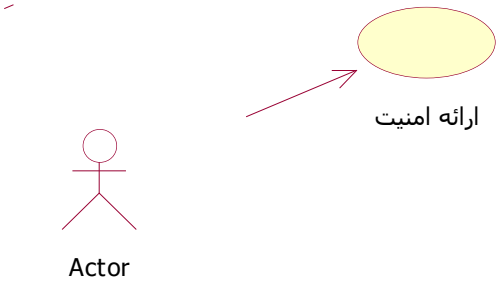
نرم افزار باید از ورود کاربران نامعتبر , hacker ها به سیستم جلوگیری نماید. یا نرم افزار باید بتواند بدون خرابی به عملیات مختلف تعریف شده بپردازد . (Non Functional)

Use case (مورد استفاده یا مورد کاربرد) یک سرویس کارکردی (عملیاتی) در سیستم پیشنهادی محسوب می شود.

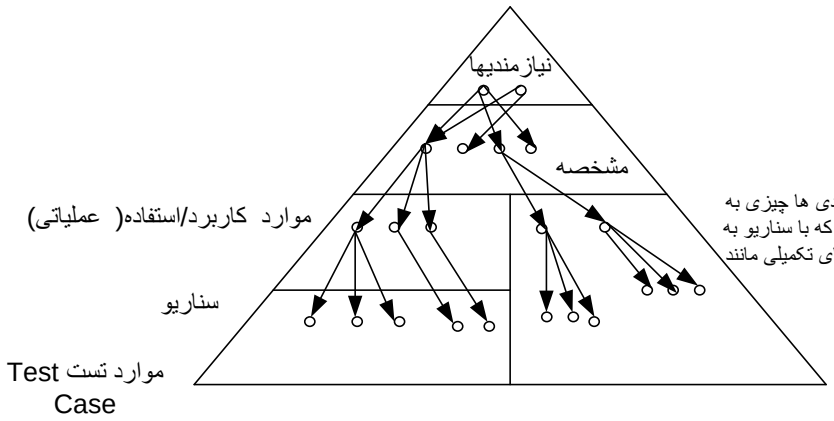
برای تحقق نیازمندی های غیر کارکردی از چند use case برای تحقق و پیاده سازی استفاده می شود .

نیازمندی های غیر کارکردی مثت امنیت نمی توانند به صورت یک یا چند مورد استفاده در نمودار مورد استفاده قرار گیرد . برای تحقق آن باید چند use case استفاده شود و در زمان پیاده سازی به آن نیازمندی بپردازند.

پس شکل زیر اشتباه است :



نکته :



نیاز مندی های تکمیلی : با این که سناریو ندارند اما می توانند use case داشته باشند.

در مرحله استخراج نیازمندی یک نیازمندی خوب چه ویژگی هایی باید داشته باشد ؟

1- عدم ابهام 2- شدنی (امکان پذیر و واقع گرایانه) Feasible

3- قابل ازمون یا تایید باشد. بتوانیم یک test case برای هر نیازمندی چه عملیاتی چه غیر کاربردی داشته باشیم تا ببینیم در نرم افزار لحاظ شده یا نه .

4- شفاف و صریح باشد . 5- atomic باشد (غیر قابل تجزیه) چون باید جزئی باشد .

6- درست باشد . 7- قابل فهم باشد (برای کسی که قرار است آن را در مرحله توسعه نرم افزار لحاظ کند)

8- ضروری باشد. 9- مستقل باشد (بین نیازمندی ها وابستگی وجود نداشته باشد. باید نیازمندی ها به صورت نیازمندی های صریح عنوان شوند. هر نیاز باید بدون توجه به نیاز قبلی خودش را معرفی کند.

عبارت مبهم	راهبرد رفع ابهام
کافی یا قابل قبول بودن	معیار های مقبولیت در سیستم را تعریف کنید . بیان کنید که چگونه سیستم می تواند در باره مقبولیت قضاوت داشته باشد .
حداقل , حداکثر, نه بیشتر , نه کمتر	مقادیر و یا شرایط قابل قبول برای حداقل و حداکثر را تعریف کنید .
بین , محدوده	حد بالا و پایین باید مشخص شود.
کارایی	چگونگی استفاده و بهره برداری سیستم از منابع موجود بایستی تعریف شود و تعیین شود که بهینه بودن در این استفاده چگونه است .
بستگی , وابستگی	ماهیت وابستگی را تشریح کنید . تلقی شما از وابستگی چیست ؟ آیا مصداق وابستگی دادهه ها نیست که یک سیستم به عنوان ورودی در اختیار سیستم دیگر قرار می دهد ؟ آیا قبل از نصب نرم افزار نرم افزارهایی باید نصب شوند ؟
انعطاف پذیریFlexibility	در پاسخ به شرایط تغییر نیازهای مشتری سیستم چه راهبردهایی را ارائه می دهد
بهبود , بهتر , سریع تر	مبتنی بر عدد و رقم بیان نمایید که چه میزان سریع تر , بهتر و یا برتر منجر به بهبود در یک حوزه کارکردی می گردد.
اختیاری optional , option هایی که end user انتخاب می کند یا تیم توسعه	دقیق بیان می کند که تلقی شما از اختیاری بودن چیست . آیا این که انتخاب سیستمی است ؟ آیا انتخاب یک کاربرد نهایی است یا انتخاب یک توسعه دهنده است ؟ انتخاب از سوی چه کسی و توسط چه کسی است ؟
در صورت نیاز	توضیح دهید چگونه می توان به طور صحیح یا در جای مناسب بر روی موارد مذکور قضاوت انجام داد؟ کی ؟ شرایط زمانی است ؟ مکانی است ؟ داده ای است ؟
قابل اتکاDependable	تعریف کنید که چگونه سیستم با استثنا ها مواجه شده و با شرایط محیط و اجرایی پیش بینی نشده پاسخ مناسب و در خور می دهد.
نبایستی	به جای استفاده از کلمه نبایستی از کلمه بایستی استفاده شود . تلاش کنید نیازمندی ها و قابلیت ها را به صورت بایستی بیان کنید.
چندین و چند بار	ذکر نمایید که منظور از چند چیست؟ در صورت امکان محدوده بالا و پایین را برایش تعریف کنید.
پشتیبانی	مثال : بایستی بتواند پشتیبانی موثری از کاربر نهایی داشته باشد (پشتیبانی در چه مقوله ای ؟) به طور دقیق بگویید چه سوال ها و توابعی بایستی توسط سیستم انجام گیرد تا پشتیبانی از قابلیت ها مفهوم پیدا کند .
کفایت و کافی بودن	بگویید چه میزان از چه چیزی کفایت و کافی بودن است ؟
نوع نیازمندی	یک نوع نیازمندی به مفهوم کلاس نیازمندی است و با توجه به تعدد و تنوع نیازمندی ها در یک پروژه نرم افزاری با مقیاس بزرگ می توان دسته بندی و طبقه بندی بین نیازمندی ها مدیریت موثری را بر روی انها انجام داد . مثلا (درجه اهمیت نیازمندی یا کارکرد نیازمندی یا حوزه پیاده سازی نیازمندی ها (در چه مرحله ای از توسعه باید به ان پرداخته شود)) نیازمندی باید درست باشد یعنی نیازمندی های دیگر را نقض نکند و با دیگر نیازمندی ها سازگاری داشته باشد.
قابلیت درگیری	گاها در نیازمندی های عنوان شده توسط حامیان پروژه مواردی از تضاد و تناقض مشاهده می شود لذا از آنجائیکه انتظارات حامیان پروژه به ویژگی ها و قابلیت های محصول مرتبط می باشد لذا نیازمندی های درخواست شده وابستگی کامل به دیگر نیازمندی ها خواهند داشت که بایستی بین نیازمندی ها سازگاری را داشته باشد.

تعیین نیاز مندی های سیستم Requirement Capture

ورود بهه مرحله تحلیل مستلزم برخورداری از درک درست سیستم و در ادامه انتظارات و خواسته های کاربران نهایی است .

سهگام برای تحلیل داریم :

1- استخراج / شناسایی نیازمندی ها Requirement Identification

2- سازماندهی نیازمندی ها Requirement Structuring

3- انتخاب Selection

استخراج نیازمندی ها نیاز به مصاحبه با مشتری دارد .

سازماندهی نیاز مندی ها , آنها را مستند کرده و تضاد و تناقض را برطرف می کند.

مرحله انتخاب : از بین Requirement های استخراج شده کدام را انتخاب و در مرحله توسعه نرم افزار لحاظ کنیم.

یک تحلیلگر که باید پای صحبت های end user بنشیند , باید یک سری مشخصه ها داشته باشد.

- جسارت : جسارت را داشته باشی تا هر سوالی در مورد محصول مورد نظر می خواهی بپرسی
- بی طرف بودن : باید انتظارات را بدون توجه به تعصبات و گرایش هایی که اعضای تیم توسعه دارند استخراج و مستند کرد.
- عدم مقاومت در برابر تغییرات : تغییرات در نیازمندی ها انکار ناپذیر است به عنوان تحلیلگر باید در مقابل تغییر نیازمندی ها کاملا انعطاف پذیر بود و تغییرات را لحاظ کرد .
- توجه به جزئیات : زمان استخراج نیازمندی ها با جزئیات کامل نیازمندی ها را لحاظ و مستند کنید . از کلی گویی با طرح سوال های جرئی تر دید بهتری از خواسته را داشته باشید.
- قالب دهی مجدد Reframing : نباید قالب های از پیش طی شده را برای سیستم های جدید اعمال کرد. باید قابلیت این را داشته باشیم که در فضای جدید به خواسته های جدید بپردازیم. نباید گفت من قبلا روی پروژه ای کار کرده ام الان باید مثل همان پروژه کار کنم.

خروجی های مرحله استخراج نیازمندی های سیستم در جدولی در کتاب خوانده شود .

روش های مختلف استخراج نیاز مندی ها (روش تعیین نیازمندی های سیستم)

1- مصاحبه : یکی از روش های سنتی روش مصاحبه و شنیدن است Interview (مصاحبه فردی)

برای مصاحبه قواعد خاصی وجود دارد . باید مکان و زمان آن از قبل مشخص شده باشد . ماهیت مصاحبه باید از قبل تعیین شده باشد. حوزه طرح سوالات باید مشخص باشد تا مشتری بتواند از پس ان برآید. باید check list از مصاحبه و موضوعات داشته باشید تا بر اساس آن روند مصاحبه در دست داشته باشید و از چارچوب خارج نشوید . این فهرست باید کاوشگرانه باشد. بر اساس پاسخ های دریافت شده سوالات بعدی را بپرسید تا جزئیات لازم برای آن مساله را دریابید.

ماهیت سوالات:

- **متن باز Openended تشریحی (سوالاتی که فرصت بیشتری برای تامل و تهیه پاسخ دارند)**
- **بسته Closeended گزینه ای (به مخاطب اجازه بیان نقطه نظرات را نمیدهد)**

طرح سوالات به صورت گزینه ای مستلزم این است که درک درستی از پاسخ های محتمل برای سوالات داشته باشید.

از تلفیق این دو نوع سوال هم می توان استفاده کرد .سوالات متن باز زمان گیر هستند.

مثال : کدام یک از موارد زیر بهترین مورد درباره سیستم اطلاعاتی است ؟

1- دسترسی آسان به کلیه داده ها 2- زمان پاسخگویی 3- توانایی اجرای سیستم همگام

راهنمایی برای انجام مصاحبه :

- سوال را به گونه ای طرح نکنید که پاسخ صحیح یا غلط بودنش مشخص شود .
- به مخاطب به دقت گوش کنید , یادداشت برداری کنید یا صدا را ضبط کنید.
- در حین مصاحبه انتظاراتی را ایجاد ننمایید. به چگونگی رسیدن به خواسته ها نپردازید . از ارائه هر گونه راه کاربردی برای تحقق ان خواسته به مشتری جلوگیری کنید .

تهیه پرسش نامه :

در صورت زیاد شدن مخاطبان می توان از پرسش نامه استفاده کرد . در مقطع مشخصی می توان پرسش نامه را در اختیار گروه زیادی قرار داد و بعد به بحث Selection پرداخت .

مصاحبه به علت تعاملی که بین مصاحبه شونده و مصاحبه کننده وجود دارد از انعطاف بالاتری برخوردار است .

4 روش برای انتخاب مصاحبه شوندگان :

- میتوان کاربران سهل الوصول را انتخاب کرد . افرادی هستند که تمایل بیشتری برای مصاحبه دارند.
- استفاده از گروه تصادفی : end user سیستم فعلی یا سیستم تصادفی
- نمونه هدفمند : فقط افرادی که شرایط خاصی دارند انتخاب نمایید که اطلاعات کامل تر و حرفه ای تری را در اختیار قرار می دهند.
- نمونه چند قشری : سوالات را صرفا به قشر و گروه کاربری خاصی محدود نکنیم . از مدیران , مدیران اجرایی , کارمندان , end user هایی که با سیستم کار می کنند و ... استفاده کنیم.

بحث ابهام :

مثلا : شما در چه فاصله زمانی از فایل های کامپیوترتان نسخه پشتیبان تهیه می کنید ؟

1- معمولا 2- گاهی اوقات 3- به ندرت 4- هرگز

اشکالات این سوال :

تلقی هر فرد از گاهی با دیگری فرق می کند . ممکن است منظور یک هفته باشد یا در طول یک روز.

ابهام دیگر در صورت مساله سوال است . منظور از فایل های کامپیوتری چیست؟ فایل های ارشیو یا ....

میتوان سوال را به صورت زیر تغییر داد :

شما در چه فاصله زمانی از فایل های کامپیوترتان بر روی hard disk که معمولا متنی هستند و بیشتر از 50 درصد کارتان مورد استفاده قرار می گیرند نسخه پشتیبان تهیه می کنید ؟

1- معمولا (حداقل یک بار در هفته)

2- گاهی اوقات (از یک تا سه بار در ماه)

3- به ندرت (یک بار در ماه یا کمتر)

4- هرگز (هیچوقت)

نقطه قوت پرسش نامه : امکان ارائه پرسش نامه در یک مقیاس بزرگ وجود دارد.

نقطه ضعف پرسش نامه : عدم برخورداری از ورود به جزئیات یک موضوع یا مورد خاص و عدم تعامل بین مخاطب و طراح سوال

مصاحبه گروهی :

از گروهی از افراد در زمان مشخص برای مصاحبه استفاده می شود .

مزیت : صرفه جویی در زمان (برای یک فرد خاص باید زمانی را برای تحلیل قرار داد تازع بعد ممکن است تضاد ها معلوم شود)

به افراد اجازه می دهد نقطه نظرات یکدیگر را بشنوند و افراد بتوانند در ادامه نکاتی را ذکر کنند که سودمند باشد .

امکان توافق یا عدم توافق در مورد موضوع خاصی را می دهد (در مورد سامانه خاص)

نقطه ضعف : ایجاد هماهنگی برای تعیین زمان سخت است . مخصوصا اگر افراد نقش های مدیریتی داشته باشند.

مشاهده کاربران (مشاهده مستقیم عملکرد کاربران) : نشستن به عنوان یک همراه یا همکار کنار کاربران و مشاهده اینکه در طول روز چه فعالیت هایی انجام می دهند . باید زمان زیادی را در نظر گرفت تا تام پروسه هایی که فرد انجام می دهد را بررسی کرد . می توان مشاهدات را مستند کرد .

ضعف ها : بزرگ نمایی (تظاهر) بسیاری از فعالیت های کاربران ممکن است Undimand باشد. ممکن است فرد در طول روز کاری نتواند تمام عملیاتی را که باید انجام دهد . ممکن است در حین کارش لازم باشد کار یکسانی را انجام دهد و باید زمان زیادی را در نظر گرفت تا تام پروسه هایی را که فرد انجام می دهد بررسی کرد.

نمیتوان به صورت online روال های موازی را استخراج کرد. مثلا کار دو کارمند به صورت همزمان در دو بخش مختلف

مزیت: کاملا در جریان روال های قرار میگیریم (مشاهده مستقیم عملیاتی که در حال انجام است)

تحلیل مستندات و رویه ها :

نگرش تحلیل مستندات این است که فرم ها , رویه ها و هرچه در سیستم تعریف شده است دریافت کرده و روی آنها مطالعه داشت و با مطالعه آنها درک بهتری از سیستم داشت .

میتوان با استفاده از این موارد مشکلات سیستم موجود را بررسی کرد. با مشاهده این رویه ها می توان work flow را استخراج کرد.

روش های جدید تعیین نیاز مندی های سیستم

طراحی برنامه کاربردی به روش مشارکت JAD (Joint Application Design)

JAD یک Framework است . در واقع JAD همان مصاحبه گروهی با قالبی دیگر است. جلسات JAD جلسات متعددی است و به جلسه خاصی محدود نمی شود . ممکن است این جلسات تا 4 جلسه طول بکشد و هزینه های گزافی را به اسپانسر یا حامی پروژه تحمیل کند.

نقش ها در جلسه JAD :

1- رئیس جلسه (هماهنگ کننده) Leader : که سازمان دهی جلسه و هماهنگی بین اعضای مختلف با او خواهد بود.

2- کاربران User : کاربران کلیدی سیستم تحت مطالعه (بعنوان اپراتور پای سیستم می نشینند و روالهای مختلف را انجام می دهند)

3- مدیران در سطوح مختلف

4- پشتیبان مالی : باید در جریان نشست ها قرار بگیرد.

5- تحلیلگران : مستند کردن برداشت و صحبت هایی که مورد استفاده قرار بگیرد. تحلیل گر بر حسب scope به طرح سوال می پردازد.

6- منشی : کار مستند کردن را دارد.

7- پرسنل سیستم اطلاعاتی : از دیگر اعضای تیم توسعه هستند مثل طراحان و برنامه نویسان که به عنوان نماینده در جلسه حضور داشته باشند. (تصویر شکل 7-6 کتاب)

در جلسات JAD باید از گروه های پشتیبانی سیستم JSS استفاده کنیم .

ممکن است کاربر شهامت انتقاد از رویه ها و سیسم مدیران را نداشته باشد . باید تدبیری اتخاذ شود تا جلسات JAD از کارایی برخوردار باشد. بنابراین برای این که افراد جسارت بیان انتقادات را دشاته باشند از ابزاری به نام JSS استفاده می شود .

در این سیستم سوالات به صورت Broadcast در اختیار افراد قرار میگیرد . پس end user ها شهامت پیدا می کنند ضعف را عنوان کنند و تحلیل گر online , feed back ها را دریافت کند .

با استفاده از این ابزار همه فرصت طرح پاسخ را دارند . مشکل این است که چون سوالات broadcast است و پاسخ ها خصوصی عنوان می شود نمی توان به تفاهم و توافق رسید .

در جلسات JAD می توان از ایده PROTOTYPE استفاده کرد یعنی برداشت ها و درخواست های کاربر را در قالب پیش نمونه قرار داد.

اشنایی با زبان مدلسازی UML

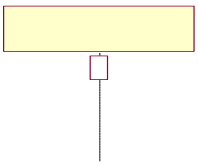
نمودار مورد استفاده : ارائه تصویری از سیستم در قالب سرویس ها و خدمات ارائه شده به موجوودیت های درون سازمانی.

نمودار توالی : یک نمودار به منظور نمایش تعاملات بین اشیا مختلف در یک سیستم

اشیا در کلاس های نمودار توالی : این کلاس ها متعلق به سطح تحلیل بوده و لزوما کلاس های مرحله طراحی نخواهند بود.

کلیشه های موجود در نمودار توالی :

1- شی



2- محور زمان Time Line |

3- پیام

4- علامت نابودی

سناریو : به جریان های قابل رخداد (پیش بینی) در حین طراحی یک مورد استفاده سناریو گفته می شود .

مثال : انتقال بانکی

1- عدم انتقال به دلیل ناکافی بودن موجودی حساب انتقال دهنده

2- عدم انتقال به دلیل فراموش کردن رمز عبور (ورود) کارت

3- عدم انتقال به دلیل پایین بودن سطح مبدا

4- عدم انتقال به دلیل پایین بودن مقصد

5- انتقال موفقیت آمیز

تشریح سناریو :

شرح مرحله به مرحله هر یک از سناریوهای قابل پیش بینی برای مورد استفاده در یک فایل متنی .

مثال :

1- مشتری کارت خود را در دستگاه کارت خوان وارد می کند .

2- مشتری رمز عبور را وارد می کند.

3- صفحه نمایش دستگاه خود پرداز گزینه مربوط به عملیات بانکی را به رویت مشتری می رساند.

4- مشتری گزینه انتقال کارت به کارت را انتخاب می کند .

نمودار توالی برای سناریوی عملیات موفقیت آمیر انتقال :

سوال : چگونه می توان کلاس های مختلف موجود در سیستم را شناسایی تا در ادامه اشیایی از انها را ایجاد و در نمودار توالی مورد استفاده قرار داد ؟

منبع : تشریح و توصیف سناریوی مورد استفاده

شناسایی و استخراج تمامی اسامی به کار رفته در سناریوی مورد استفاده

اسامی به کار رفته :

1- مشتری 2- دستگاه کارت خوان 3- رمز عبور 4- شماره حساب 5- صفحه نمایش 6- شماره کارت 7- مبلغ 8- موجودی 9- رسید

10- شماره تراکنش 11- کارت 12- حساب

تفاوت شی و داده :

داده یک جز اطلاعاتی فاقد رفتار است ولی شی جزء یا ابزار اطلاعاتی است دارای رفتار .

به ازای هر سناریو در خصوص مورد استفاده یک نمودار توالی اختصاصی باید ترسیم شود .

رسید یک کلاس است اما رفتاری ندارد . در نمودار تراکنش هم هیچ عملیاتی انجام نمی دهد :

1- مبلغ 2- شماره حساب مبدا 3- نوع رسید 4- شماره حساب مقصد 5- تاریخ تراکنش 7- شماره رسید

زمان های استفاده از علامت تخریب / نابودی Destuction Marker

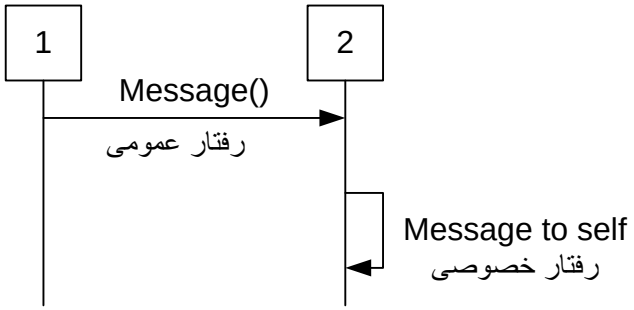
1- شی مربوطه عملیات خاصی را انجام نمی دهد .

2- شی مربوطه درخواستی از دیگران ندارد.

3- شی مربوطه درخواستی از سوی دیگران ندارد.

نمودار کلاس سطح تحلیل :

ارتباط بین کلاس ها در نمودار کلاس



1- ارتباط انجمنی (Association) : ————

2- ارتباط کل به جز (Whole-Part) تجمیع (bydirectionalیا undirectional است)

3- ارتباط ترکیب (Composition)

4- ارتباط ارث بری (Generalization)

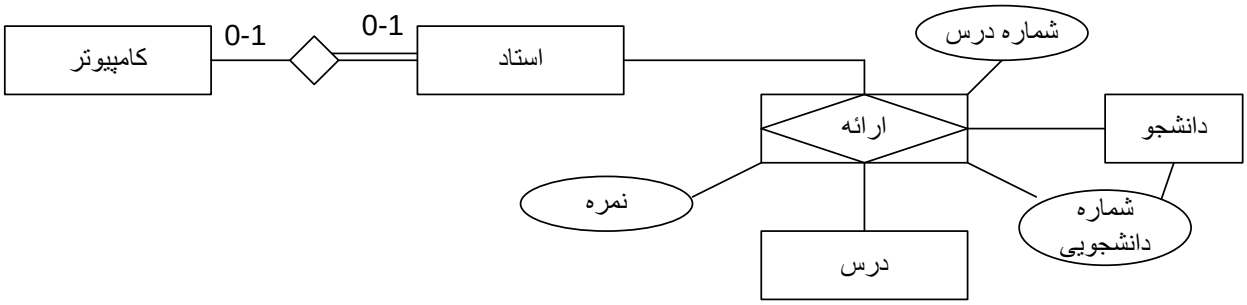
چندگانگی Multiplicity

به تعداد نمونه اشیا ایجاد از یک کلاس که با اشیایی از کلاس (های) دیگر در ارتباط هستند.

کلاس انجمنی

کلاسی است که حاصل ارتباط انجمنی بین دو کلاس است .

در صورت از بین رفتن این ارتباط کلاس مزبور از بین خواهد رفت.



نمودار فعالیت Activity Diagram

برای نمایش جریان های کاری موجود در سیستم/ واحد سازمانی مورد استفاده قرار می گیرد (work flow)

کلیشه موجود در نمودار فعالیت

1- Swim Lane نمایانگر یک واحد سازمانی ،موجودیت که در جریان کاری اقدامات خاصی انجام می دهد.

2- Activity مجموعه اقدامات لازم در یک مقطع زمانی خاص که توسط یک واحد سازمانی انجام می شود.

3- وضعیت State حالاتی که یک جریان کاری ممکن است تجربه کند.

4- تصمیم گیری Decision انتخاب یک روند در جریان کاری بسته به شرایط و محدودیت ها

5- گذار Transition ترتیب/تقدم فعالیت های صورت گرفته توسط هر واحد سازمانی

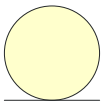
حالات مختلف یک فعالیت

شرایط زمانی :

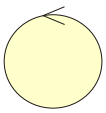
- **No Entry** مرحله شروع
- **Do** حین اجرای برنامه
- **No Exit** مرحله پایان

انواع کلاس های سطح تحلیل :

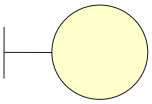
1- کلاس موجودیت Entity



2- کلاس کنترلی Controller : به منظور کاهش هزینه نگهداری نرم افزار می توان منطق موجود در سیستم را در یک کلاس کنترلی کپسوله کرد تا در صورت تغییر صرفا این کلاس تغییر پیدا کند.



3- کلاس واسط روابط Boundary



- واسط انسان – ماشین
- واسط ماشین – ماشین
- زیر سیستم – زیر سیستم
- مولفه – مولفه
- سیستم – سیستم

طراحی

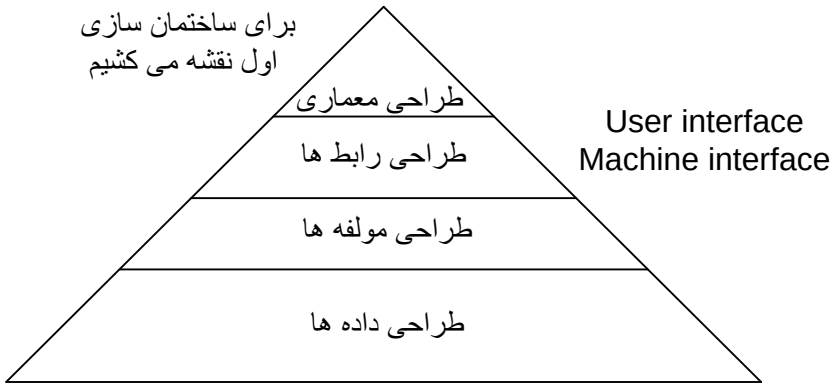
مبحث شناخت نیازمندی ها و تحلیل را پشت سر گذاشتیم و وارد مبحث طراحی می شویم.

طراحی چگونگی پیاده سازی / تحقق مدل تحلیل در قالب یک سیستم نرم افزاری است .

طراحی در 4 حوزه مطرح می شود و ملاحظاتی برای آن وجود دارد .

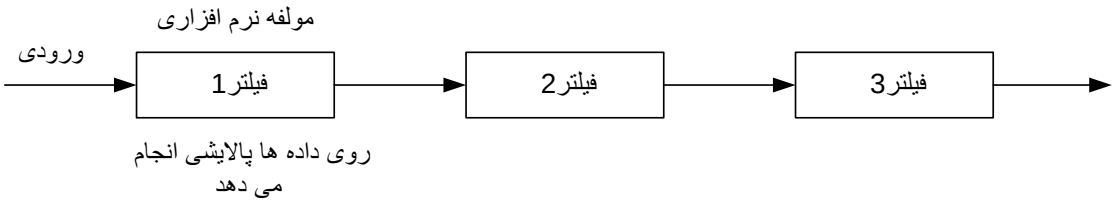
در مرحله تحلیل بدون توجه به اینکه سیستم در چه پلتفرمی تحقق میابد به تحلیل می پردازیم.

در مرحله طراحی باید در مورد زبان برنامه سازی ، انتخاب سکو (platform) ، انتخاب معماری ،انتخاب و به کارگیری الگوهای مختلف طراحی (Design Pattern) ،انتخاب سرویس دهنده ها (بانک اطلاعاتی – وب سرور و...) ،انتخاب توپولوژی شبکه (از ویندوز یا یونیکس و... استفاده کنیم) ،پروتکل ها و استاندارد هایی که باید وجود داشته باشد ملاحظاتی وجود داشته باشد. (صفحه 207 کتاب پرسمن)



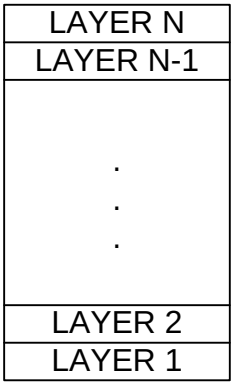
انواع معماری ها Architecture

1- Pipeline (معماری لوله فیلتر)



2- Layered

مثال مدل OSI که هفت لایه دارد . داده از لایه بالا دریافت می شود و در اختیار لایه پایین قرار میگیرد. هر لایه به لایه بالایش سرویس می دهد و از لایه زیرین خود سرویس میگیرد.



لایه N موظف به ارائه سرویس ها و خدماتی به لایه N+1 به صورت شفاف میباشد . در عین حال این لایه از سرویس های ارائه شده توسط لایه N-1 برای این منظور بهره می گیرد.

3- Black Board (تخته سیاه یا داده مجدد)

داده ها را در Black Board قرار می دهیم .

تعریف معماری :

معماری از دو جز تشکیل شده است : اجزا و ارتباط بین آنها

در رشنال رز انواع کلاس های سطح طراحی را داریم . این کلاس ها عبارتند از :

1- کلاس ایستا یا استاتیک Static Class

وقتی حافظه ای در اختیار متغیری قرار می گیرد و در نهایت در اختیار آن برنامه قرار میگیرد.

در زبان جاوا کلاس های ایستا داریم .

در آن واحد تنها یک نسخه از کلاس ایستا می تواند وجود داشته باشد.

بعد از ایجاد یک object تا انتها object وجود دارد .

2- کلاس های موقتی (گذرا)

اطلاعات موجود در اشیا ایجاد شده از کلاس های گذرا تا انتهای برنامه scope در حافظه باقی می ماند و پس از آن از بین می رود.

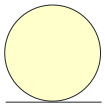
(اطلاعاتی کلاس موجودیت در جایی ذخیره و از آن استفاده می شود . کلاس موجودیت را در سطح تحلیل میاوریم.)

3- کلاس های دائمی (مانا) Persistant

اطلاعات مربوط به اشیا ایجاد شده در این کلاس از بین نمی روند حتی اگر شی مربوطه در انتهای برنامه از بین برود.

کلاس های سطح تحلیل موجودیت (entity) از مصادیق کلاس های مانا در سطح طراحی هستند (تا بعد مدلسازی داده را انجام دهیم)

در سطح طراحی باید بر اساس ضوابط و قواعدی که وجود دارد نام گذاری کلاس و رفتار آن انجام شود. لزوما تناظر یک به یک بین کلاس های سطح تحلیل و سطح طراحی وجود ندارد.



حساب مشتري

مانا تعریف می کنیم چون می خواهیم در مدلسازی داده ها جدولی برای حساب مشتری داشته باشیم تا اطلاعات حساب را در آن قرار دهیم پس باید انرا مانا تعریف کنیم تا بتوانیم از ان استفاده کنیم.

با مانا کردن یک کلاس سطح طراحی در فرآیند ایجاد مدل داده Data Modeling کلاس مورد نظر به عنوان یک جدول table در سطح بانک اطلاعاتی مطرح می گردد که صفات و ویژگی های کلاس مورد نظر در قالب فیلد های بانک اطلاعاتی در سطح جدول خواهد بود.

```
Class Account{
int inventory; // موجودی
long int accountID;//شماره حساب
long int cardID;//شماره کارت
int checkInventory(int); // بررسی موجودی
void deposit(int);//واریز به حساب
};
Int main(){
Account akbari(121212);
akbari.deposit(100);}
```

پرونکل ها امکان تعامل بین نرم افزار ها را به وجود میا ورند.

دسته بندی کلاس های سطح طراحی

1- کلاس های با قاعده Regular : به صورت پیش فرض تمام کلاس های سطح طراحی Regular هستند.

2- کلاس های پارامتر پذیر : این کلاس ها برای ایجاد خانواده ای از کلاس ها مورد استفاده قرار میگیرد. مثل template ها در c++

Vector<int>

Vector<dataType>

Vector<int> stdList(20);

Vector<struct Student> list(20);

3- کلاس نمونه سازی شده Instantiate Class

در صورتی که برای پارامتر های کلاس پارامتر پذیر مقادیری در نظر گرفته شود کلاس مورد نظر نمونه سازی شده خواهد بود.

4- کلاس utility ابزار سودمند

کلاس های سودمند معمولا فاقد صفات هستند. حاوی متدها و رفتارهایی هستند که می توانند در کاربردهای مختلف در پیاده سازی متد های مختلف دیگر کلاس ها مورد استفاده قرار گیرند. (مثل کلاس math در زبان جاوا که فقط از توابع آن استفاده میکنیم یا کلاس sort در c++ که روش های مختلف مرتب سازی را دارد مثل merg sort)

5- کلاس های تلفیقی

ترکیب کلاس های سودمند و پارامتر پذیر

کلاس های سودمند (جزوه استاد)

این کلاس از مجموعه ای از عملیات تشکیل شده است. به عنوان مثال کلاس math در زبان جاوا که برای پیاده سازی توابع مختلف ریاضی مورد استفاده قرار میگیرد. از جمله توابع abs ، توابع مثلثاتی . این توابع کنار هم قرار گرفته و تشکیل کلاس سودمند را می دهند. این کلاس ها توسط کلاس های دیگر مورد استفاده قرار میگیرند.

کلاس های سودمند پارامتر پذیر (جزوه استاد)

یک کلاس پارامتر پذیر حاوی مجموعه ای از عملیات است به عبارتی مجموعه ای از کلاس سودمند و پارامتر پذیر است . در واقع یک قالب یا template می باشد.

کلاس سودمند نمونه سازی شده Instantiated class utility

یک کلاس پارامتر پذیر که حاوی مقادیر برای پارامتر ها باشد.

رابط ها Interface

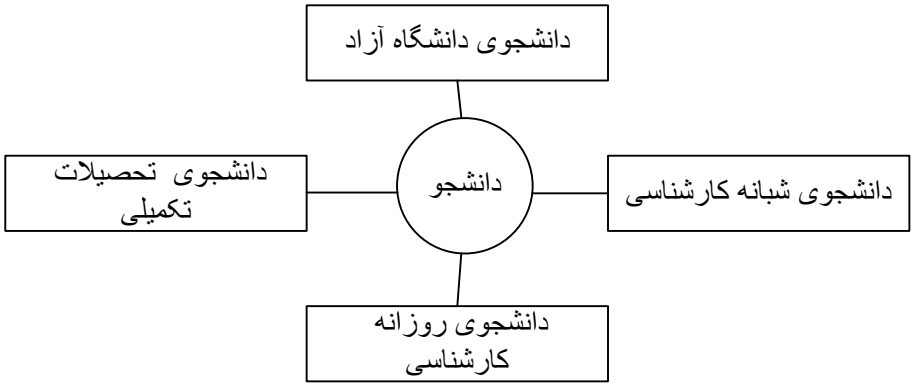
به منظور جدایی پیاده سازی کلاس از رابط آن استفاده می شود. رابط ها صرفا حاوی امضای پارامتر ها بوده و فاقد پیاده سازی می باشند. از یک رابط می توان چندین پیاده سازی متفاوت ارائه نمود .

مثال : کارت گرافیک و سیستم عامل

هرکس بخواهد کارت گرافیک تولید کند باید قابلیت هایی را در آن قرار دهد تا سیستم عامل بتواند فرامین آنرا اجرا کند. کارت گرافیکی بعد از قرار دادن آن بدون نصب کردن کار می کند به خاطر پروتکل آن . یعنی متد ها و فرامین در کارت گرافیکی لحاظ می شود. هر تولید کننده می تواند در کنار آن فرامین قابلیت هایی اضافه کند .

پروتکل هایی را در قالب متد داخل رابط قرار می دهیم . و کلاس های مختلف مستلزم پیاده سازی رفتار ها در سطحinterface هستند ولی می توانند رفتار های اختصاصی هم داشته باشند.

مثال : روال هر کدام از این دانشجویان فرق دارد ولی هر کدام وجوه مشترکی دارند مثل انتخاب واحد .



نقطه اشتراک تمام کلاس ها :

1- ثبت نام کردن(از ارث بری استفاده نمی کنیم چون روال ها باید متفاوت باشد .) 2- فراغ التحصیلی

نکته : رفتار ها را بدون پیاده سازی در سطح interface قرار می دهیم ولی کلاس ها را ملزم می دانیم تا با interface ارتباط برقرار کنند. پس کلاس ها باید رفتار مشترک داشته باشند نه لزوما پیاده سازی یکسان داشته باشند.

فصل پنجم : برنامه ریزی Planning

برنامه ریزی تلاش برای تخمین نزدیک به واقعیت برای میزان هزینه ، کار ،زمان ، ریال برای توسعه ی یک سامانه نرم افزاری است .

مواردی که در تخمین تاثیر مستقیم دارد

پیچیدگی نزم افزار ، اندازه نرم افزار ، درجه عدم قطعیت

اهداف برنامه ریزی پروژه

فراهم اوردن چارچوبی است که مدیر را قادر به برآورد منطقی منابع ، هزینه ها و زمانبندی کند.

اقداماتی که برای برنامه ریزی پروژه بایستی انجام شود :

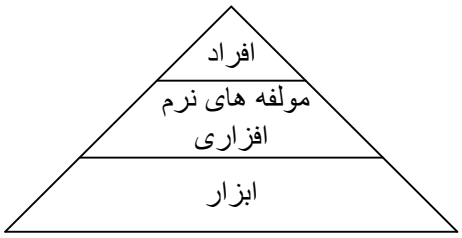
1- تعیین دامنه کاربرد نزم افزار (مهندسی سیستم)

مطالعه امکان سنجی feasibility study

- مسائل هزینه ای و ریالی
- فناوری و تکنولوژی
- زمان
- منابع

2- منابع Resource

افراد ، مولفه هایی که باید توسعه پیدا کنند ، ابزار های سخت افزاری و نرم افزاری .



منابع نرم افزاری قابل استفاده مجدد

1- مولفه های آماده : بدون هیچ تغییری عینا قابل استفاده اند .

2- مولفه های تجربه کامل : بایستی تغییراتی در آنها ایجاد شود تا در پروژه قابل استفاده باشند اما هزینه این تغییر کم است چون احتمالا افرادی که در این پروژه کار می کنند در پروژه قبلی حضور داشته اند .

3- مولفه های تجربه ناقص : اگر هزینه این مولفه سنگین باشد از نو مولفه ای ایجاد می کنیم . اما اگر اینطور نباشد انها را customize کرده و دوباره استاده می کنیم .

4- مولفه های جدید

تخمین پروژه نرم افزاری

روش های برآورد (تخمین) پروژه نرم افزاری

1- برآورد تاخیر تا اواخر پروژه (درجمه قطعیت 100 درصد)

2- استناد به پروژه های مشابه قبلی انجام شده برای بر آورد

3- تجزیه استفاده از تکنیک های تجزیه برای اورد پروژه

$C(p) > C(p1) + C(p2)$

مساله اصلی > مساله فرعی 1 + مساله فرعی 2

4- بهره گیری از مدل های تجربی برای برآورد

تکنیک های تجزیه ای نرم افزار

صحت برآورد پروژه به چند عامل بستگی دارد :

1- درجه درستی براورد اندازه محصول توسط برنامه ریز

2- توانایی تفسیر برآورد اندازه به صورت کار انسانی به زمان تقویمی و هزینه ها

3- درجه انعکاس توانایی های تیم نرم افزاری در برنامه ریزی

4- پایداری خواسته های پروژه و محیطی که کار مهندسی نرم افزار را پشتیبانی می کند .

روش های برآورد در تجزیه

1- برآورد مبتنی بر مساله 2- برآورد مبتنی بر فرآیند (توجه به مراحل مختلف توسعه نرم افزار و تخصیص زمان ، هزینه و منابع لازم برای هر یک از مراحل توسعه نرم افزار

بر اورد مبتنی بر مساله

- براورد بر اساس تعداد خطوط کد Line Of Code (LOC)
- برآورد بر اساس نقاط عملکرد Function Point (FP)

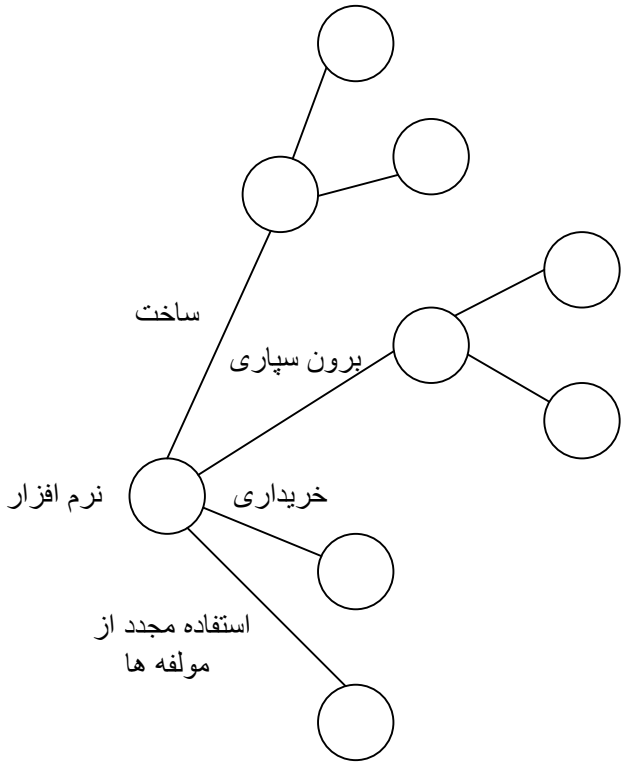
برآورد مبتنی بر فرآیند تخمین بر اساس شیوه فرآیند مورد استفاده

تصمیم گیری ساخت / خرید

چند انتخاب پیش رو داریم :

- نرم افزار را برای مدت معینی اجاره کرده یا مجوز استفاده از آن ها را بگیریم .
- مولفه های تجربه ناقص را خریده و نرم افزار را تولید کنیم.
- برون سپاری

بر اساس این موارد درخت تصمیم کشیده می شود.



تعریف ریسک

یک اتفاق یا رخداد قابل پیش بینی در سطح پروژه نرم افزاری که عدم توجه به ان می تواند تبعاتی را برای پروژه نرم افزاری به دنبال داشته باشد.

مراحل مدیریت ریسک :

1- شناسایی عوامل ریسک Risk Identification

2- تحلیل Risk Analysis

3- بررسی تبعات ریسک Outcome 4 - برنامه ریزی احتیاطی Contigency Planning

فرق دو واژه ریسک Risk و ضایعه ، خرابی Hazard :

هر خرابی لزوما ریسک نیست ، عدم توجه به ریسک موجب خرابی می شود.

مثال : ساعت 24 از همدان به سمت تهران حرکت می کنیم و می خواهیم 8 صبح در جلسه ای حضور داشته باشیم. چند نمونه ریسک :

پنچر شدن ماشین ،ضایعه : نرسیدن به برنامه مورد نظر

نمی توانیم جلوی پنچر شدن را بگیریم اما می توانیم با برنامه ریزی تبعات را به حداقل کاهش دهیم مثلا زاپاس با خودمان حمل کنیم.

مثالی نر م افزاری :

ریسک : فردی در مینه فرآیند توسعه بخواهد تیم توسعه را ترک کند .

برنامه ریزی : مستندات فرد را در اختیار بگیریم تا در نبود او تبعات به حداقل برسد یا فردی را در کنار او قرار دهیم تا از مستنداتی که او ایجاد می کند استفاده کند و در مستندات خودش انها را لحاظ کند و جانشین فردی باشد که تیم توسعه را ترک می کند.

راهبرد های مواجهه با ریسک

1- راهبرد های عکس العملی

هیچ برنامه ریزی انجام نمی دهیم وقتی در عمل انجام شده قرار میگیریم عملی را انجام می دهیم (مدیریت بحران) مثل kill مردن پروسه ها در انحصار متقابل

2- راهبردهای پیش کنشی Proactive

برنامه ریزی میان مدت یا بلند مدت که تبعات حادثه ای مثل زلزله را به حداثل برسانیم. این روش هوشمندانه است چون برنامه ریزی لازم برای ریسک انجام می شود .

ممکن است احتمال وقوع رخدادی بالاتر و یا تبعات آن سنگین تر باشد . اگر عکس این حالت وجود داشت انها را ignore می کنیم. (این قسمت حتما از کتاب خوانده شود)

هدف مدیریت ریسک حذف عوامل ریسک و خود ریسک نیست ،بلکه تبعات و خسارت هایی را که ریسک مورد هدف قرار داده تا حد امکان کاهش می دهد.

ریسک های نرم افزاری

1- عدم قطعیت : احتمال وقوع یک ریسک در پروژه های نرم افزاری $0<p<1$ است . ریسکی که اختمال وقوع آن 1 باشد دیگر ریسک نیست بلکه امکان قطعی است .

2- ضایعات Hazard : هزینه ناشی از وقوع یک ریسک در پروژه نرم افزاری

نمونه ای از یک خرابی در پروژه نرم افزاری : نبود یک فرد در پروژه نرم افزاری و در اختیار نداشتن مستندات او

Outcome : در مرحله بعدی خودش را نشان می دهد .

انواع ریسک ها

1- ریسک های پروژه ای : ریسک هایی که در مقوله پروژه نرم افزاری نمود پیدا می کند و برنامه ریزی پروژه را تهدید می کند. مثل زمانبندی ،تخمین هزینه ، تخمین منابع انسانی و ...

2- ریسک های فنی : به مواردی اطلاق می شود که در خصوص مسائل فنی پروژه مطرح می شود. مثل استفاده از یک فناوری در نرم افزار یا اشکالات و ضعف هایی که در مرحله طراحی و مستندات وجود دارد .

3- ریسک های تجاری :به مواردی در خصوص استقرار نرم افزار ،فروش نرم افزار ، بازاریابی و .. مطرح میشوند. مثلا نرم افزاری را توسعه دهیم که برای ان مشتری وجود ندارد .

ریسک ها :

- قابل پیش بینی
- غیر قابل پیش بینی

ریسک های قابل پیش بینی را می توان با تجربه بدست آورد.

در ریسک های غیر قابل پیش بینی هیچ برنامه ریزی انجام نشده و تبعات آن می تواند مشکل آفرین باشد . ریسک های غیر قابل پیش بینی ماهیتا غیر قابل پیش بینی بوده و باید به صورت ضمنی در برنامه ریزی لحاظ شوند . مثلا با وجود n ریال براورد مالی $n+\Delta n$ ریال را برای موارد پیش بینی نشده در نظر بگیریم. یا $t+\Delta t$ برای زمان و ...

شناسایی ریسک Risk Identification

تعریف : ارائه راهکاری برای شناسایی عواملی در پروژه نرم افزاری که به صورت مستقیم یا غیر مستقیم چک لیستی را تهیه می کنیم (از پروژه های قبلی استخراج می کنیم یا از منابع مشابه استفاده می کنیم)

در این چک لیست نوع ریسک ،item آن ، احتمال ان ، ضایعات آن را در نظر می گیریم.

دسته بندی چک لیست ها در حوزه های مختلف (گروه هایی که می توانند در چک لیست وجود داشته باشند)

1- اندازه محصول : ریسک های ناشی از عدم تشخیص درست اندازه (میزان) محصولی که قرار است توسعه یابد.

2- تاثیر تجاری : ریسک های ناشی از بازار و وجود رقبایی در توسعه محصول های مشابه (باید کیفیت را بالا و قیمت و زمان توسعه را کاهش داد)

3- ویژگی های مشتری : ریسک های ناشی از ناتوانی در برقراری ارتباط سازنده با مشتری

4- تعریف فرآیند ،ریسک های ناشی از به کار گیری یک مدل توسعه نرم افزار ، متدلوژی توسعه

5- محیط توسعه : ریسک های ناشی از ابزار ها و محیط هایی که برای توسعه محصول مورد نظر لازم است . (مثلا برای توسعه نرم افزار می خواهیم از JBuilder استفاده کنیم و قبلا از آن استفاده نکرده ایم. این یک ریسک است چون افراد قبلا با آن کار نکرده اند. وقتی از محیط شناخته نشده ای استفاده کنیم LOC افزایش میابد.

6- فناوری که قرار است توسعه یابد :بهره گیری از فناوری های توسعه نرم افزار که در موارد مشابه مورد استفاده قرار گرفته اند.

7- تجربه و تعداد پرسنل : به کارگیری افرادی با تجربه ضعیف در فرایند توسعه نرم افزار

سنجش یا تحلیل ریسک (از کتاب خوانده شود)

باید اطمینان حاصل کنیم که کاربر می خواهد از نرم افزار استفاده کند اگر کاربر نهایی نخواهد از محصول استفاده کند یا ساز مخالف بزند یک ریسک محصوب می شود .

ناپایدار بودن دامنه پروژه باعث می شود نیاز ها تغییر پیدا کنند و SCOPE سیستم دست خوش تغییر شود .

مولفه ها و محرک های ریسک :

1- ریسک کارایی 2- ریسک هزینه ای 3- ریسک زمانبندی 4- ریسک پشتیبانی

ریسک کارایی :میزان عدم قطعیت براورده شدن خواسته ها توسط محصول و مناسب بودن برای کاربرد

ریسک هزینه ای : در مقوله برنامه ریزی بودجه و هزینه برنامه ریزی درستی نداشته باشیم (عدم قطعیت در حفظ بودجه پروژه)

ریسک زمانبندی : مربوط به مسائل زمانبندی

ریسک پشتیبانی : عدم قطعیت در سهولت تصحیح ، نتوان پشتیبانی درستی از نرم افزار را داشت و بحث ارتقا و سازگاری را نتوانیم داشته باشیم .

کتاب بر اساس این 4 تا ITEM جدولی ترسیم کرده . ریسک ها فاجعه بار از کتاب خوانده شود .

پیش بینی ریسک

باید 4 فعالیت را در این زمینه انجام داد :

1- تعیین مقیاس برای احتمال ریسک (از مقیاسی برای واحد ریسک استفاده کنیم)

2- پیامد های رسیک Hazard

3- براورد تاثیر ریسک بر پروژه و محصول outcome

4- توجه به صحت ریسک : مطمئن شویم موارد استخراج شده درست استخراج شده است .

جدول ریسک

ریسک	گروه	احتمال	تاثیر	برنامه احتیاطی
پیش بینی تعداد کاربران کم تر از حد انتظار	ps	٪ 30	**2**	

این جدول بر اساس احتمال و تاثیر می تواند مرتب شود .

Ps: اندازه پروژه st : ریسک پرسنلی cu : ریسک مشتری

بعد از کامل کردن جدول ریسک در اختیار مدیر قرار می گیرد و مدیر بر اساس ریسک هایی که وجود دارد تصمیم گیری می کند که در مواجهه با چه ریسک هایی باید برنامه RLM وجود داشته باشد و برنامه ریزی برای مواجه شدن با ریسک را انجام دهد. از یک سری ریسک ها صرف نظر می شود و برای بعضی ریسک ها راهبردهای مشخصی ارائه می شود.

اگر ریسکی تاثیرش زیاد باشد ولی احتمال آن کم باشد از آن چشم پوشی می شود.

اگر ریسکی تاثیرش کم باشد ولی احتمال ان بالا باشد مورد توجه قرار می گیرد.

اگر ریسکی تاثیرش کم ولی احتمال ان متوسط باشد مورد توجه قرار می گیرد.

با این موارد می توانیم روی ریسک ها دسته بندی داشته باشیم.

میزان قرار گرفتن در معرض ریسک Risk Expose

RE = C*P= هزینه × احتمال

مثال : فقط 70 ٪ از مولفه های نرم افزاری قابل استفاده مجدد هستند. یعنی 30 ٪ آن را خودمان باید وسعه دهیم. در این مثال ریسک 70 ٪ است .

نکته : برآورد از مولفه های نرم افزاری یک ریسک است .

مثال : اگر هزینه توسع هنرم افزاری برای هر LOC ، 14 پوند باشد و هر مولفه نرم افزاری 100 خط باشد و تعداد کل مولفه های نرم افزاری 60 تا باشد همچنین اگر احتمال ریسک 80٪ باشد داریم:

18*14*100 = 25200 RE = 25200*0.80 = 20200 هزینه ای که برای ریسک باید بپردازیم

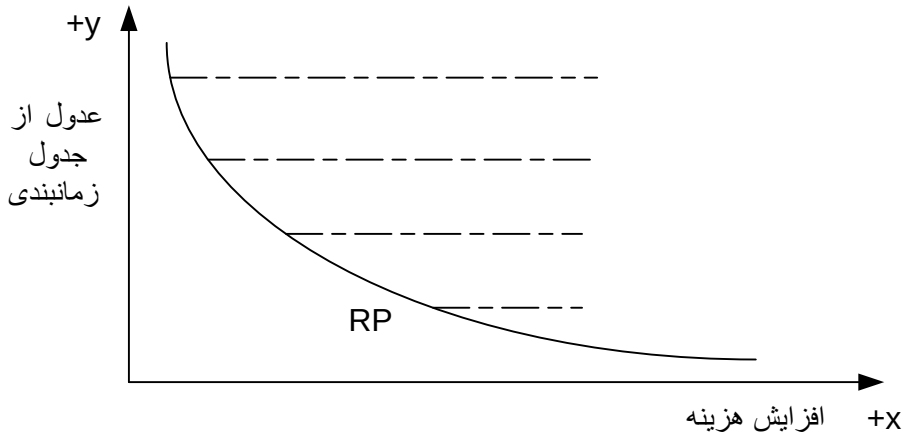
در بحث ریسک اگر بتوانیم 3 ایتم r_i ، l_i، X_i را استخراج کنیم می توانیم اطلاعات خوبی را بدست اوریم.

X_i: میزان در معرض قرار گرفتن RE

r_i : ریسک l_i احتمال ریسک

در مقوله ریسک مفهوم نقطه مرجع (Refference Point) را داریم :

باید ببینیم هزینه ها ایجاب می کند پروژه را ادامه دهیم یا نه ؟ در واقع با استفاده از نقطه مرجع می توانیم تصمیم بگیریم .



پالایش ریسک

پالایش روی نیاز مطرح شده انجام می شود تا مولفه نرم افزاری بدست بیاید.

(این قسمت حتما از کتاب خوانده شود)

نمایش ریسک به صورت (شرط Condition، گذار Transition ، پیامد Consequence outcome (تبعاتی که در صورت بروز ریسک اتفاق افتاده است)) است.

گذار یعنی چطور از این ریسک که در حال اتفاق افتادن است عبور کنیم.

مثال : بافرض (شرط x)این احتمال وجود دارد که (پیامد) نتیجه شود (این قسمت حتما از کتاب خوانده شود)

شرط قرعی 1 : مولفه های قابل استفاده مجدد توسط یک شخص ثالث ایجاد شده که هیچ اطلاعی از استانداردهای طراحی پروژه داخل سازمان ندارد. این ریسک در مرحله support خودش را نشان می دهد .

اگر شخص ثالث از استاندارد های ما استفاده نکرده باید هزینه هایی را برای مطابقت دادن با استانداردهای سازمان داشته باشیم.

تبعات : ناسازگاری در استانداردهای طراحی در مراحل بعدی و هزیننه های نگهداری افزایش میابد.

شرط فرعی 2 : استانداردهای طراحی برای مولفه ها رعایت نشده و ممکن است برخی مولفه های نرم افزاری مطابقت نداشته باشند.

شرط فرعی 3 : مولفه های قابل استفاده مجدد که قرار است در سطح نرم افزار استفاده شود با زبان برنامه نویسی و در محیط دیگری ایجاد شده اند.

تبعات هر شرط با شرط دیگر فرق دارد .

نظارت Monitoring ، مدیریت Management، تقلیل Mitigation (MMM)

باید بتوانیم به ترفند های مختلف احتمال ریسک را کاهش دهیم چون ترجیح می دهیم ریسک اتفاق نیفتد . این مستلزم مدیریت و نظارتی است که در مرحله توسعه نرم افزار باید از ابتدا تا به انتها داشته باشیم .

مثلا در مورد تقلیل باید لاستیک را عوض کنیم ،سرپیچ ها با سرعت بالا نپیچیم و ... تا احتمال پنچر شدن کاهش یابد. یا در مورد نظارت باید هر نیم ساعت وضعیت تایر را بررسی کنیم .

مدیریت در قالب برنامه ریزی احتیاطی نمود پیا می کند مثل زاپاس .

باید در پروژه ها به جای فرد محور بودن ،Task محور باشیم تا در صورت نبود فرد خاص از روی مستندات ان فرد بتوانیم پروژه را ادامه دهیم .

مولفه های نرم افزاری :

- **Ready :** میتوانند همان طور که هستند در پروژه لحاظ شوند .
- **تجربه ناقص :** با ایجاد تغییراتی در پروژه لحاظ می شوند .
- **تجربه کامل**
- **مولفه هایی که باید خودمان ایجاد کنیم یا مهندسی شود .**

ریسک های امنیتی :

امنیت نرم افزار وقتی نمود پیدا می کند که نر م افزار در اختیار مشتری است و مشتری از ان استفاده می کند . (صفحه 155 پرسمن)

RISK information Sheet				
شماره ریسک	**تاریخ**	**احتمال**	**تاثیر(زیاد – متوسط – کم – قابل چشم پوشی)**	**توصیف یا شرح ریسک**

مثال :

توصیف ریسک : 70 درصد مولفه های نرم افزاری قابل استفاده مجدد هستند .

شرط فرعی 1

شرط فرعی 2

...

تقلیل / نظارت :

1- تماس باشخص ثالث برای مطابقت با استاندارد های طراحی

2- اعمال فشار برای مطابقت کامل استانداردهای واسط

3- تعیین تعداد مولفه ها در شرط فرعی 3 تعیین اینکه پشتیبانی زبان قابل حصول هست یا نه ؟

مدیریت طرح احتیاطی

(این فصل (6) حتما از روی کتاب خوانده شود)

زمانبندی پروژه نرم افزاری Scheduling

زمانبندی باید از زمانی که پروژه شروع می شود تا زممانی که محصول با کیفیت در اختیار مشتری قرار گیرد دغدغه مدیر باشد.

در مورد زمانبندی باید یک سری فعالیت ها را در روند توسعه نرم افزار شناسایی کنیم . این فعالیت ها در مراحل و روند های مختلف توسعه نرم افزار صورت می گیرد .

مثلا شناسایی ریسک فعالیت های مرتبط در مرحله شناخت سیستم (مهندسی سیستم) ،تحلیل ، طراحی ، پیاده سازی ، تست

صرفا شناخت فعالیت ها کفایت نمی کند بلکه باید وابستگی بین فعالیت ها را هم در نظر گرفت. Dependency

در بحث زمانبندی برای هر یک از فعالیت ها باید برآوردی داشته باشیم که به هر فعالیت چقدر زمان اختصاص دهیم . (بعضی فعالیت ها موازی هستند بعضی وابسته هستند. باید این موارد را مد نظر قرار داد)

دیرکرد Late

هر تاخیری که در مرحله توسعه نرم افزار با آن مواجهیم (ریسک ، زمانبندی غیر واقعی و ...) باعث دیرکرد می شود . اگر پروژه نرم افزاری با تاخیر مواجه شود ممکن است باعث شکست پروژه نرم افزاری شود .

عوامل اساسی دیرکرد :

1- مهلت غیر واقع بینانه Duration

2- تغییر در درخواست ها و انتظارات مشتری

3- براورد کمتر از حد واقع فعالیت ها و زمان و هزینه لازم برای هر یک

4- ریسک های قابل پیش بینی و غیر قابل پیش بینی در روند توسعه نرم افزار

5- مشکلات فنی در روند توسعه گریبان تیم توسعه را بگیرد.

6- مشکلات انسانی Human factor

7- ارتباطات سوء میان پرسنل

8- شکست مدیریت پروژه در شناسایی دیرکرد و عدم تلاش برای جبران آن

هدف : تجزیه پروژه به زیر سیستم های کوچکتر جهت حصول اطمینان از زمانبندی واقع بینانه

وابستگی : شناسایی وابستگی های موجود بین فعالیت های مختلف . فعالیتی که به دیگر فعالیت ها وابستگی دارد لزوما در انتظار انجام فعالیت های مورد نظر است .

تخصیص زمان : بر اساس براورد صورت گرفته از هر فعالیت زمان لازم و منابع انسانی مورد نیاز جهت انجام در نظر گرفته می شود.

اعتبار سنجی فعالیت ها (Validation)

ارائه یک مکانیسم جهت حصول اطمینان مدیریت پروژه از انجام موفقیت امیز یک فعالیت.

در روند توسعه نرم افزار یک سری milestone تعریف می کنیم . وقتی در نقاط milestone قرار میگیریم میدانیم چه مراحلی را پشت سر گذاشته ایم و چه مراحلی را پیش رو داریم. وقتی اعتبار سنجی انجام میشود نمی توانیم اطمینان حاصل کنیم که در یک milestone خاص قرار گرفتیم.

تخصیص انتخاب مسئولیت

تعیین اینکه هر یک از فعالیت ها با زمان تخمینی توسط چه فرد یا افرادی بایستی انجام شود . س

تعیین پیامدها outcomes

شناسایی محصول یا مستندی که حاصل انجام فعالیت مورد نظر است .

تعیین نقاط عطف (milestone)

یک پروژه نرم افزاری از چندین نقطه عطف تشکیل شده است که بایستی تعیین شود. انجام (پشت سر گذاشتن) چه فعالیت هایی پروژه را در یک نقطه عطف خاص قرار می دهد.

نکته : قرار نیست به ازای هر فعالیت milestone داشته باشیم.

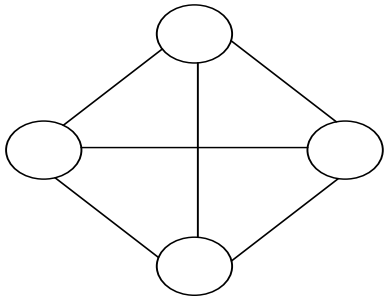
نمونه ای از milestone این است که طراحی را به طور کامل انجام داده باشیم (یا پیاده سازی یا تست و ...)

رابطه بین افراد و ک ار (مثال از کتاب خوانده شود)

در اینجا بحث اجتماع افراد در قالب تیم ها مطرح است .

گره ها در گراف : نیروی انسانی

یال ها در گراف : مسیر ارتباطی بین افراد



$Kn = n(n-1)/2$

مثال : اگر 4 نفر جداگانه پروژه هایی را انجام دهند در پایان سال هر کدام 5000 خط کد loc تولید می کنند. اگر این افراد با هم کار کنند 6 ارتباط به وجود می ایدو با این فرض که 250 خط کد به ازای هر ارتباط کم شود داریم :

4*5000 – 6*250 = 18500

و این یعنی 7/5 درصد کاهش کارایی

اگر دو نفر را در دو ماه پایانی به پروژه اضافه کنیم :

تعداد ارتباط 6*5/2 = 15

بهره وری هر نفر در ماه 20000/120 = 840

20000+2*(1680)-(15*250)

انواع پروژه های نرم افزاری و مجموعه وظایف را از کتاب بخوانید.

مجموعه وظایف / فعالیت ها برای پروژه نرم افزاری

یک مجموعه وایف عبارت است از یک سری وظایف کاری ،نقاط عطف ، و قطعات قابل تحویل (قطعات نرم افزاری (قطعه کد)) که باید برای کامل شدن پروژه ان ها را انجام داد.

این مجموعه وظایف به منظور دستیابی به کیفیت بالا در محصول باید از انضباط کافی برخوردار باشد و در عین حال نباید تیم پروژه را به کار بیهوده مشغول کند.

انواع پروژه ها :

1- پروژه های توسعه مفهوم : هدف توسعه یک محصول نرم افزاری نیست بلکه ایجاد یک مفهوم و نهادینه کردن ان در سطح یک جامعه هدف (target) است .

2- پروژه توسعه محصول نرم افزاری جدید : هدف تولید یک محصول نرم افزاری برای به کارگیری ان در سطح جامعه هدف است.

3- پروژه های بهسازی : در صورتی که محصول نرم افزاری از جنبه هایی مانند عملکرد ، کارایی ، امنیت و... دارای ضعف هایی باشد می توان با اتخاذ تدابیر به بهبود و بهسازی ان پرداخت.

4- پروژه نگهداری محصول نرم افزاری : هدف شناسایی و دریافت نقطه نظرات مشتری / کاربر نهایی از نرم افزار و اعمال موارد در سطح نرم افزار است (بحث ارتقا در بهسازی مورد توجه قرار میگیرد) . بحث ارتقا ، تصحیح و تطبیق وجود دارد .

5- پروژه مهندسی مجدد (reengineering) : هدف دوباره سازی یک محصول نرم افزاری که می تواند توام با منسوخ کردن یک فناوری و جایگزین کردن یک فناوری دیگر باشد .

درجه دشواری : بسته به مهارت نیروی انسانی دخیل در پروژه ها و انجام پروژه های نرم افزاری مشابه متغیر اند .

دسته بندی سطوح مختلف دشواری :

1- موردی (حتما از کتاب خوانده شود) : خواسته های مربوط به مستند سازی کاهش پیدا می کند ولی دستورالعمل ها و اصول توسعه محصول نرم افزاری در ان لحاظ می شود .

2- ساخت یافته : در این گروه درجه دشواری فعالیت های مختلفی برای حصول اطمینان از کیفیت بالای محصول به اجرا گذاشته می شود. وظیفه اطمینان از کیفیت نرم افزار ، مستند سازی و انداره گیری به شیوه کار امد اجرا می شود .

3- دقیق : فرآیندکامل برای این نوع پروژه با درجه ای از انظباط اجرا می شودکه کیفیت بالای محصول را تضمین می کند. در این نوع پروژه مستند سازی قوی باید انجام شود.

4- واکنش سریع : به خاطر یک وضعیت اظطراری فقط وظایف ضروری برای حفظ کیفیت اجرا می شود . مثلا پروژه های بهسازی ،نگهداری ، توسعه یک محصول نرم افزاری جدید

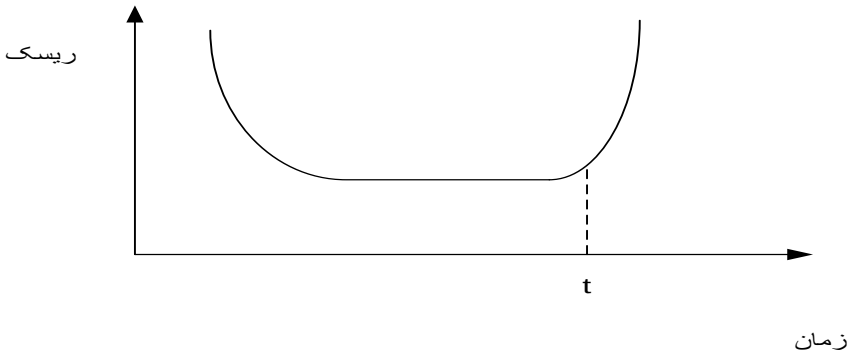
ملاک های تطبیق (مطابقت)

مشخص می کند یک پروژه نرم افزاری بر اساس ماهیت و scope خود در کدام حوزه جای می گیرد.

11 ملاک تطبیق داریم :

1- اندازه پروژه 2- تعداد کاربران بلقوه :end user هایی که در آخر قرار است از محصول استفاده کنند . 3- اهمیت ماموریت Criticality مثلا نرم افزار واژه پرداز word و نرم افزاری که برای یک نیروگاه هسته ای می نویسند اصلا قابل مقایسه نیستند. 4- طول عمر برنامه کاربردی

به علت بروز نیازهای کاربران یا ضعف در پروژه که نمی تواند خودش را با نیاز ها همسو کند بعد از مدت زمانی ریسک بالا می رود.



5- پایداری خواسته ها 6- سهولت برقراری ارتباط با مشتری یا کاربر نهایی 7- بلوغ فناوری به کاربرده در پروژه : در پروژه از چه فناوری می خواهیم استفاده کنیم ،اگر فناوری از بلوغ بالاتری استفاده کند درجه دشواری افزایش پیدا می کند .

8-شرایط مرزی / حدی کارایی : مثلا درج درکورد در بانک اطلاعاتی باید در کسری از زمان انجام شود.

9- ویژگی های تعبیر شده : قابلیت هایی که می خواهیم در نرم افزار لحاظ شود .

10- پرسنل پروژه : از لحاظ میزان مهارت در پروژه تاثیر دارد.

11- عوامل مهندسی مجدد : چطور می توانیم از ایده مهندسی مجدد در توسعه نرم افزار استفاده کنیم .

از روی این 11 مورد جدولی استخراج خواهد شد که برای هر یک از مقادیر آن ، مقادیر 1-5 در نظر می گیریم.

هدف از استخراج ملاک های تطبیق بدست اوردن درجه دشواری و در ادامه زمانبندی است . (صفحه 165 پرسمن)

بعد از مشخص شدن درجه دشواری فعالیت های اصلی در سطح کلان را مشخص می کنیم. برای فعالیت اصلیTask های اصلی ، برای هر Task فعالیت های فرعی و برای هر یک نیرو و زمان لازم را تخصیص می دهیم.

زمانبندی CPM (Critical Pass Method)

مسیر بحرانی : یک مسیر در ین فعالیت های پروژه که زمان کلی پروژه را مشخص می کند.

فعالیت های روی مسیر بحرانی شناوری شان صفر است . هر تاخیر در این مسیر روی کل پروژه تاثیر می گذارد.

مثلا اگر سقف انتخاب درس برای دانشجویی برداشته شود باید به پیش نیاز ها توجه کرد . اگر درسی را بیفتد یک ترم عقب می ماند. (نمودار خط زمانی صفحه 137 کتاب)

پیگیری :می خواهیم ببینیم از زمانبندی عقب افتاده ایم یا نه .

1- یکی از روش ها نشست های ادواری با افراد تیم پروژه است و هرکدام از افراد باید به مواردی که با ان مواجه شده اند بپردازند (برای گزارش گیری)

2- ارزیابی نتایج کلی بازبینی ها 3- مقایسه شروع واقعی با شروع برنامه ریزی شده 4- برگزاری جلسات رسمی با سازندگان نرم افزار (مدیران اجرایی)

پیگیری خطاها : وسیله ای برای ارزیابی وضعیت یک پروژه در حال اجرا

بازدهی رفع نقص DRE

نقص Defect : در سطح نرم افزار و بعد ازتحویل ان وجود دارد . توسط مشتری شناسایی می شود.

خطا Error : در سطح نرم افزار و قبل از تحویل ان وجود دارد . تست های مختلف برای شناسایی و رفع خطا انجام میشود.

فرمول مهم بازدهی رفع نقص : شناسه ای از موثر بودن فعالیت های تضمین کیفیت

$$DRE = \frac{E}{E + D}$$

این کسر یعنی از بین مجموعه خطاها در سطح نرم افزار چند تای انها قبل از تحویل نرم افزار شناسایی شده اند. اگر این کسر به صفر نزدیک شود یعنی ضعف زیاد است و خطاهای زیادی شناسایی شده است.

با استفاده از خطاهای شناسایی شده ، عملکرد فرد مورد بررسی قرار می گیرد. تعیین پیشرفت حاصل شده در جریان حرکت پروژه نرم افزاری موارد زیر را می توانیم استخراج کنیم :

1- تعداد نواقص و خطاها به ازای هر صفحه تعیین نیازمندی ها

2- تعداد خطا به ازای هر مولفه

3- تعداد خطاها به ازای هر مولفه در سطح طراحی

4- تعداد خطاها به ازای هر مولفه در سطح پیاده سازی

خطاهای مولفه : واسط – نحوی – عملکردی

مدلسازی وب Web Modeling

امروزه بسیاری از سیستم های نرم افزاری مبتنی بروب هستند.

سیستم های نرم افزاری :

- **Stand alone Application**
- **Web Application**

این بحث در مرحله Design از مراحل هفت گانه توسعه نرم افزار صورت میگیرد.

انواع کلاس های وب :

1- کلاس مشتری Client Page

2- کلاس فرم Web Form

3- کلاس سرویس دهنده Server Page : تمامی صفحات توسعه یافته به یکی از زبان های وب (...,php,perl,asp.net) که وظیفه پردازش درخواست های مشتری وتولید پاسخ مناسب را برعهده دارند.

انواع ارتباط های موجود بین کلاس های وب :

1- ارتباط بین دو صفحه مشتری (پیوند)

در صورتی که در یکی از صفحات وب از تگ a استفاده شده باشد بین این دو صفحه مشتری یک ارتباط لینک با کلیشه <<link>> در سطح نمودار کلاس وجود خواهد داشت.

2- ارتباط بین صفحه مشتری و هر کدام از کلاس های فرم (حاوی Aggrigation)

یک صفحه مشتری می تواند حاوی صفر ، یک و یا چند کلاس فرم برای مقاصد مختلف باشد .

3- ارتباط بین کلاس مشتری و سرویس دهنده (build)

حاصل ترجمه یک کلاس سرویس دهنده توسط وب سرور یک صفحه مشتری است که برای نمایش به سمت مرورگر کاربر ارسال می شود.

4- ارتباط بین کلاس فرم و سرویس دهنده (submit تسلیم)

اطلاعات گرداوری شده توسط کاربر در یک کلاس فرم جهت پردازش در اختیار یک کلاس سرویس دهنده قرار می گیرد.

```
<form method =”get” action=”register.php”>
...
</form>
```

صفحه register.php متولی پردازش درخواست ماست .

5- ارتباط بین دو صفحه سرویس دهنده (forward)

یک صفحه سرویس دهنده می تواند پس از دریافت یک درخواست از سوی مشتری و انجام پردازش لازم روی آن برای انجام پردازش های بیشتر این درخواست را به یک صفحه سرویس دهنده دیگر ارسال نماید.

این مواردی که ذکر شد برخی کلاس های وب بودند.